Upravljački sklop za električnu hvataljku

Domjanić, Filip

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Filip Domjanić

Zagreb, 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Prof. dr. sc. Bojan Jerbić, dipl. ing.
Dr. sc. Bojan Šekoranja, mag. ing.

Student:

Filip Domjanić

Zagreb, 2016.

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

Zahvaljujem mentorima prof.dr.sc. Bojanu Jerbiću i dr.sc. Bojanu Šekoranji na stručnoj pomoći i razumijevanju tijekom izrade ovog rada.

Također zahvaljujem roditeljima Damiru i Jadranki na pruženoj podršci i razumijevanju tijekom studija.

Filip Domjanić

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

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

ZAVRŠNI ZADATAK

Student: **FILIP DOMJANIĆ** Mat. br.: 0035185973

Naslov rada na hrvatskom jeziku: **UPRAVLJAČKI SKLOP ZA ELEKTRIČNU HVATALJKU**

Naslov rada na engleskom jeziku: **CONTROLER FOR ELECTRIC GRIPPER**

Opis zadatka:

U sklopu zadataka potrebno je predložiti povezivanje industrijske servo-električne hvataljke Schunk PT-AP 70-R s računalnom upravljačkom okolinom. Pri tome je potrebno:

- omogućiti upravljanje servo-električnom hvataljkom pomoću Arduino upravljačkog sklopa,
- izraditi standardnu knjižnicu upravljačkih algoritama, koja će omogućiti upravljanje hvataljkom na proizvoljnom robotu pomoću TCP/IP komunikacije (indirektno preko Arduino upravljačkog sklopa).

Zadatak zadan:
25. studenog 2015.

Rok predaje rada:
1. rok: 25. veljače 2016
2. rok (izvanredni): 20. lipnja 2016.
3. rok: 17. rujna 2016.

Predviđeni datumi obrane:
1. rok: 29.2., 02. i 03.03. 2016.
2. rok (izvanredni): 30. 06. 2016.
3. rok: 19., 20. i 21. 09. 2016.

Zadatak zadao:
Prof. dr. sc. Bojan Jerbić

Predsjednik Povjerenstva:
Prof. dr. sc. Zoran Kunica

Sadržaj:

POPIS SLIKA

POPIS TABLICA

POPIS KRATICA

BCC	**Block Check Character**
DC	**Direct current**
IP	**Internet Protocol**
TCP	**Transmission Control Protocol**
OSI	**Open Systems Interconnection**
TTL	**Transistor Transistor Logic**
UART	**Universal Asynchronous Receiver/Transmitter**
UDP	**User Datagram Protocol**
USB	**Universal Serial Bus**
WWW	**World Wide Web**

SAŽETAK

Zadatak ovog rada je primjena servo-električne hvataljke kojom se upravlja pomoću RS232 serijske veze. Komunikacija današnjih robota s okolinom najčešće se ostvaruje putem TCP/IP protokola, koji postupno istiskuje klasičnu serijsku komunikaciju iz upotrebe. Zbog toga moderne upravljačke jedinice više nemaju serijski priključak što onemogućuje izravno povezivanje s ovakvom vrstom hvataljke.

Da bi se hvataljka i dalje mogla koristiti potreban je posrednik koji će imati ulogu prevoditelja signala između hvataljke i upravljačkog računala robota. U takvoj situaciji obično se koristi računalo kao posrednik u upravljanju serijskom vezom, ali je takvo rješenje skupo. Daleko jednostavnije i jeftinije rješenje je primjena mikrokontrolera pomoću kojeg se realizirala komunikacija između hvataljke i upravljačke jedinice robota. Mikrokontroler može komunicirati s robotom putem TCP/IP protokola, a naredbe prema hvataljci prilagođavati obliku prihvatljivom hvataljci te ih upućivati RS232 komunikacijskim protokolom na izvršavanje. Isto tako, vrijedi i obrnuto, informaciju iz hvataljke primljene putem RS232 komunikacije prilagođava se upravljačkoj jedinici robota te šalje TCP/IP protokolom upravljačkoj jedinici robota.

Ključne riječi: hvataljka, robot, mikrokontroler

SUMMARY

Task of this work is application of servo-electric gripper controlled by Serial RS232 connection. Today's robots usually communicate with the environment via TCP/IP protocol which gradually displaces conventional serial communication from use. Therefore, modern control unit no longer have a serial port so gripper direct connection is not possible.

If we want to use gripper anyway we need same device which will convert signals between gripper and control unit of a robot. In that situation desktop computer is usually used for controlling serial connection, but that solution is expensive. Far simpler and cheaper solution is usage of a microcontroller with which is realized the communication between the gripper and robot control unit. The microcontroller can communicate with the robot via TCP / IP protocol, and adjust commands for gripper to form acceptable to gripper and sends it via RS232 communication for execution. Likewise, the information received from the grippers trough RS232 communication is adapted for the control unit of the robot and sent trough TCP/IP protocol to control unit of the robot.

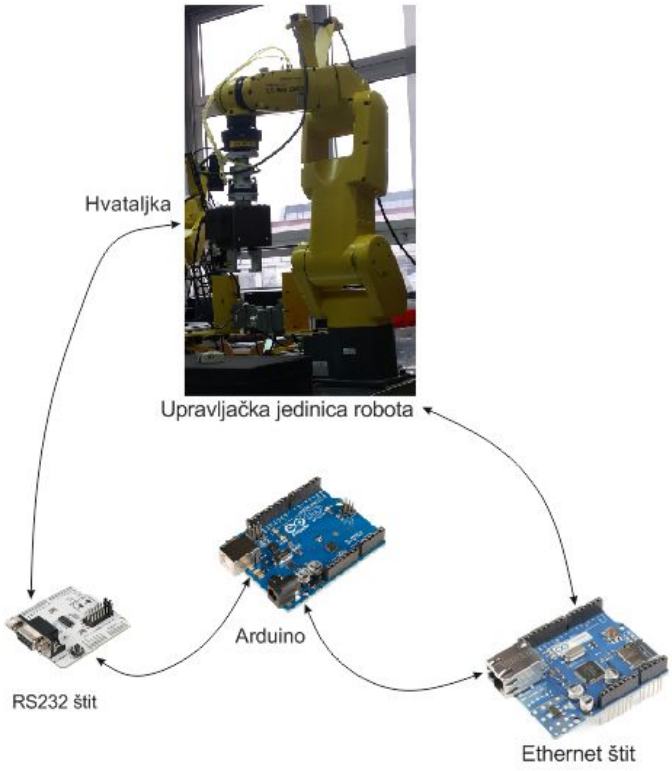
Key words: gripper, robot, microcontroller

1 UVOD

Rad se bavi povezivanjem električne robotske hvataljke Schunk PT-AP 70-R s robotskom upravljačkom jedinicom putem TCP/IP protokola. Problem je što primijenjena hvataljka koristi isključivo serijsku RS232 komunikaciju. Na suvremenim robotskim upravljačkim sustavima već je odavno klasična serijska komunikacija zamijenjena suvremenijim protokolima, poglavito mrežnim sučeljima. Stoga je potrebno osmisliti ”međusklop” koji će premostiti serijsku komunikaciju s mrežnim protokolom. Do sada se taj problem rješavao uz posredovanje samostojećeg računala. No takvo je rješenje previše skupo. Cilj ovog zadatka je stoga realizacija međusklopa koji će omogućiti svakom robotu komunikaciju i upravljanje električnom hvataljkom putem TCP/IP mrežnog protokola. Odabrani posrednik za ostvarenje tog zadatka je Arduino mikrokontroler koji će se programirati tako da prilagođava signale hvataljci odnosno upravljačkom računalu robota.

2 Dijelovi sustava

Glavni dijelovi sustava su servo električna hvataljka i upravljačka jedinica robota kojima treba omogućiti međusobnu komunikaciju. To ostvarujemo Arduino mikrokontrolerom, no osim njega potrebni su i Arduino mrežni i RS232 štitovi (engl. Shield) koji se povezuju na njega, te mrežni i serijski kabel. Nakon nabavke svih dijelova preostaje još razviti Arduino program koji će i s aplikacijske strane omogućiti komunikaciju između hvataljke i upravljačke jedinice robota i riješiti problem ovoga zadatka.



Slika 1 - Smjerovi komunikacije

2.1 Hvataljka Schunk PT-AP 70-R

2.1.1 Karakteristike

Schunk PT-AP 70-R je servo električna hvataljka s dva paralelna mehanička prsta pogonjena elektroničkim komutiranim motorom (engl. Brushless DC motor). Za ispravan rad potreban joj je istosmjerni napon od 24V. RS232 sučelje se koristi za komunikaciju s upravljačkim uređajem.

Moguće je upravljati brzinom gibanja i ubrzanjem mehaničkih prstiju, silom kojom će mehanički prsti nešto uhvatiti i razmakom na koji ih prethodno želimo rastvoriti. Osim što

prima informacije hvataljka šalje povratne informacije, tako u svakom trenutku imamo podatak o razmaku na kojem su prsti ili kojom silom djeluje na predmet koji je zahvatila.

Raspon sile kojom može djelovati je od 30 do 200N, što korisniku omogućuje prenošenje kako osjetljivijih i krhkih objekata, tako i onih veće mase koji za manipulaciju zahtijevaju veću silu.

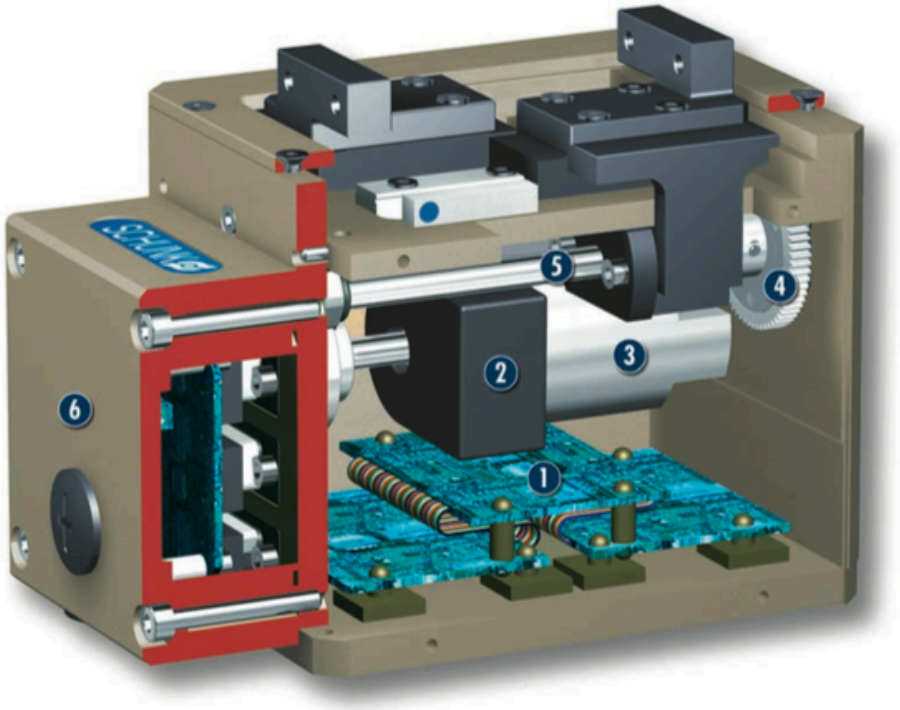
Mehaničke prste hvataljke moguće je razmaknuti do 70mm.

Ostale karakteristike vidljive su u slijedećoj tablici.

Technical data

Designation		PG 70
Mechanical gripper operating data	ID	0306087
Stroke per finger	mm [in]	35.0 [1.378]
Constant gripping force (100 % continuous duty)	N [lbf]	200.0 [45]
Max. gripping force	N [lbf]	200.0 [45]
Min. gripping force	N [lbf]	30.0 [6.7]
Weight	kg [lbs]	1.4 [3.09]
Recommended workpiece weight	kg [lbs]	1.0 [2.20]
Closing time	s	1.1
Opening time	s	1.1
Max. permitted finger length	mm [in]	140.0 [5.512]
IP rating		20
Min. ambient temperature	°C [°F]	5.0 [41]
Max. ambient temperature	°C [°F]	55.0 [131]
Repeat accuracy	mm [in]	0.05 [0.0020]
Positioning accuracy	mm [in]	on request
Max. speed	mm/s	82.0
Max. acceleration	mm/s²	328.0
Electrical operating data for gripper		
Terminal voltage	V	24.0
Nominal current	A	2.2
Maximum current	A	on request
Resolution	mm [in]	0.25 [0.010]
Controller operating data		
Integrated electronics		Yes
Voltage supply	VDC	24.0
Nominal current	A	0.5
Sensor system		Encoder

Slika 2 - Karakteristike hvataljke



Slika 3 - Dijelovi hvataljke

Na slici 3 je prikazana hvataljka u presjeku na kojem se vide glavni sastavni dijelovi:

1- Upravljačka elektronika

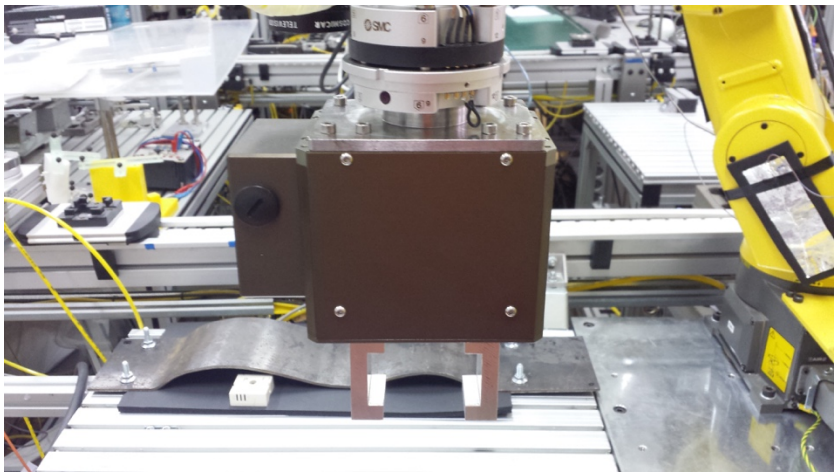
2- Enkoder

3- Motor

4- Mehanizam za prijenos snage s motora na pogonsko vreteno

5- Vreteno

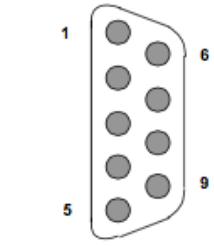
6- Kućište priključnih vodova



Slika 4 - Hvataljka na robotu

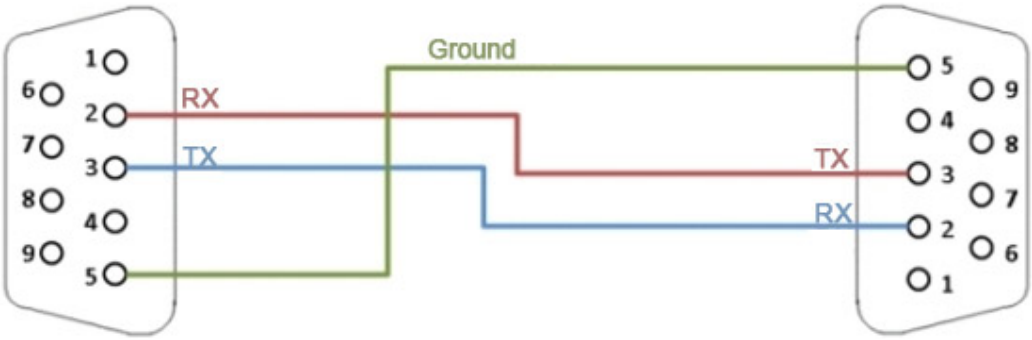
2.1.2 Upravljanje hvataljkom

Upravljanje se odvija serijskom RS232 komunikacijom. Hvataljka ima ženski DE-9 priključak i koristi kontakte 2, 3 i 5.

Connector (female)	Pin	Color	Description
	1	-	not used
	2	brown	RxD
	3	white	TxD
	4	-	not used
	5	Shield	GND
	6	-	not used
	7	-	not used
	8	-	not used
	9	-	not used

Slika 5 - Priključak na hvataljku

Za uspostavljanje veze između hvataljke i Arduinovog RS232 štita potreban je Null-modem kabel. On služi za izravno serijsko povezivanje dvaju uređaja. Razlika između običnog i null modem kabela je u zamijenjenim kontaktima 2 i 3. (Slika 6).



Slika 6 – DE-9 Null-modem kabel

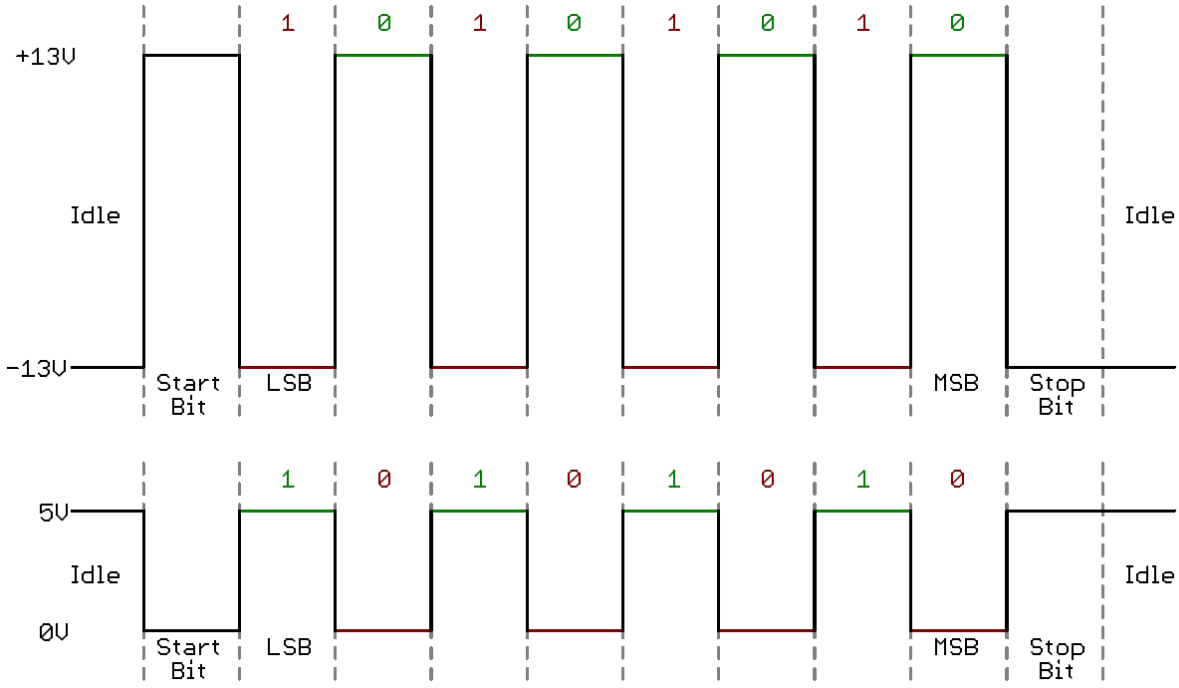
2.1.3 RS232 komunikacija

Serijska veza je proces slanja podataka preko komunikacijskog kanala slijedno bit po bit. Norma uređaja određuje raspon napona za logička stanja jedan i nula. Logičko stanje nula određuju naponi između +3V i +25V, a za logičko stanje jedan naponi moraju biti između -3V i -25V. Naponi između -3V i +3V su u području zabranjene zone. Na većini računala naponi se u području između -13 i +13V. Visoki naponi omogućuju prijenos podataka na veće udaljenosti.

Najmanje značajan bit (eng. Least Significant Bit - LSB) se kod serijske komunikacije šalje prvi, a najviše značajan bit (eng. Most Significant Bit - MSB) zadnji. Osim korisne informacije, uređaj šalje "start bit" i "stop bit" koji označavaju početak i kraj poruke te bit za provjeru valjanosti poruke.

2.2 RS232 štit

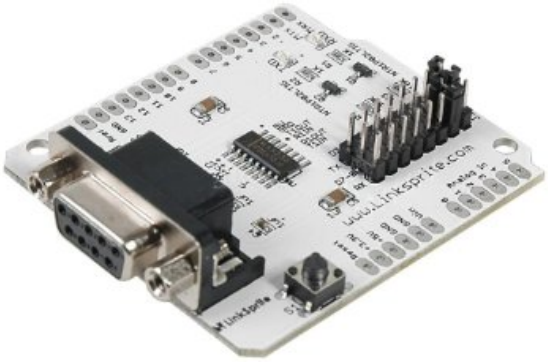
Serijski portovi hvataljke i Arduina su kompatibilni samo aplikacijski, signali su im istih oblika, ali različitih vrijednosti. Arduino koristi UART, univerzalni asinkroni prijemnik/odašiljač koji se povezuje zajedno s RS-232. UART prevodi podatke između paralelnih i serijskih formata te može podešavati brzine prijenosa i oblik slanja podataka. Za pravilan rad, UART na strani primatelja i na strani odašiljača mora biti isto podešen. Takva komunikacija na Arduinu je TTL serijska. Vrijednosti signala u serijskoj TTL komunikaciji su u području između 0V i vrijednosti napona napajanja. Vrijednost napona za logičko stanje jedan je vrijednost napona napajanja, dok je logičko stanje nula određeno s 0V.



Slika 7 - TTL i RS232 signal

Da bi se uspješno povezalo hvataljku i Arduino potrebno je prilagoditi komunikacijske signale. Prvo se obavlja logička operaciju NE tj. komplementiraju se logička stanja, a zatim

im se mijenja vrijednost napona. Najjednostavniji način za to je primjena MAX-232 integriranog kruga između spomenuta dva uređaja. MAX232 je temeljni dio RS232 štita koji prilagođava RS232 signal u TTL signal i obrnuto. Kada na ulaz dobije TTL logičko stanje 0 onda na izlazu vlada napon raspona +3 i +15V, a kada na ulaz dovedemo TTL logičko stanje 1, na izlazu vlada napon između -3 i -15V. Analogna je prilagodba RS232 u TTL. Dovođenjem napona između +3 i 15V na ulaz, na izlazu će biti TTL logičko stanje 0, a dovođenjem napona između -3 i -15V na izlazu ćemo dobiti logičko stanje 1.



Slika 8 - RS232 štit

Logičko stanje	RS232 napon	TTL napon
"0"	+3V do +15V	0V
"1"	-3V do -15V	5V

Tablica 1 - Razlike u naponima logičkih stanja

2.3 Fanuc LR Mate 200iC5L

Fanuc robot LR Mate, serije 200iC/5L manji je robot iz Fanucove ponude. Idealan je za razne primjene u automatizaciji. Ima 6 stupnjeva slobode gibanja, nosivost mu je do 5 kg, ponovljivost do ±0,03 mm. Za razliku od osnovnog modela ovaj robot ima veći radni doseg. Na taj robot je postavljena hvataljka. Tehničke specifikacije su mu sljedeće:

Broj osi		6
Masa		29kg
Doseg		892mm
Točnost ponavljanja		±0.03mm
Nosivost		5kg
Opseg gibanja	J1	340 º
	J2	230 º
	J3	373 º
	J4	380 º
	J5	240 º
	J6	720 º
Maksimalna brzina	J1	270º/s
	J2	270º/s
	J3	270º/s
	J4	450º/s
	J5	450º/s
	J6	720º/s

Tablica 2 - Tehničke karakteristike Fanuc robota LR Mate

Slika 9 - Fanuc LR Mate 200iC5L

Upravljanje robotom vrši upravljačka jedinica R-30iA Mate na brzom operativnom sustavu koji se nije sukladan sa Windows operativnim sustavima. Može upravljati sa do 40 stupnjeva slobode gibanja. Pokrene se u manje od jedne minute. Za komunikaciju s drugim uređajima koristi mrežni sustav (engl. Ethernet) s RJ-45 priključkom.

Slika 10 - R-30iA upravljačka jedinica

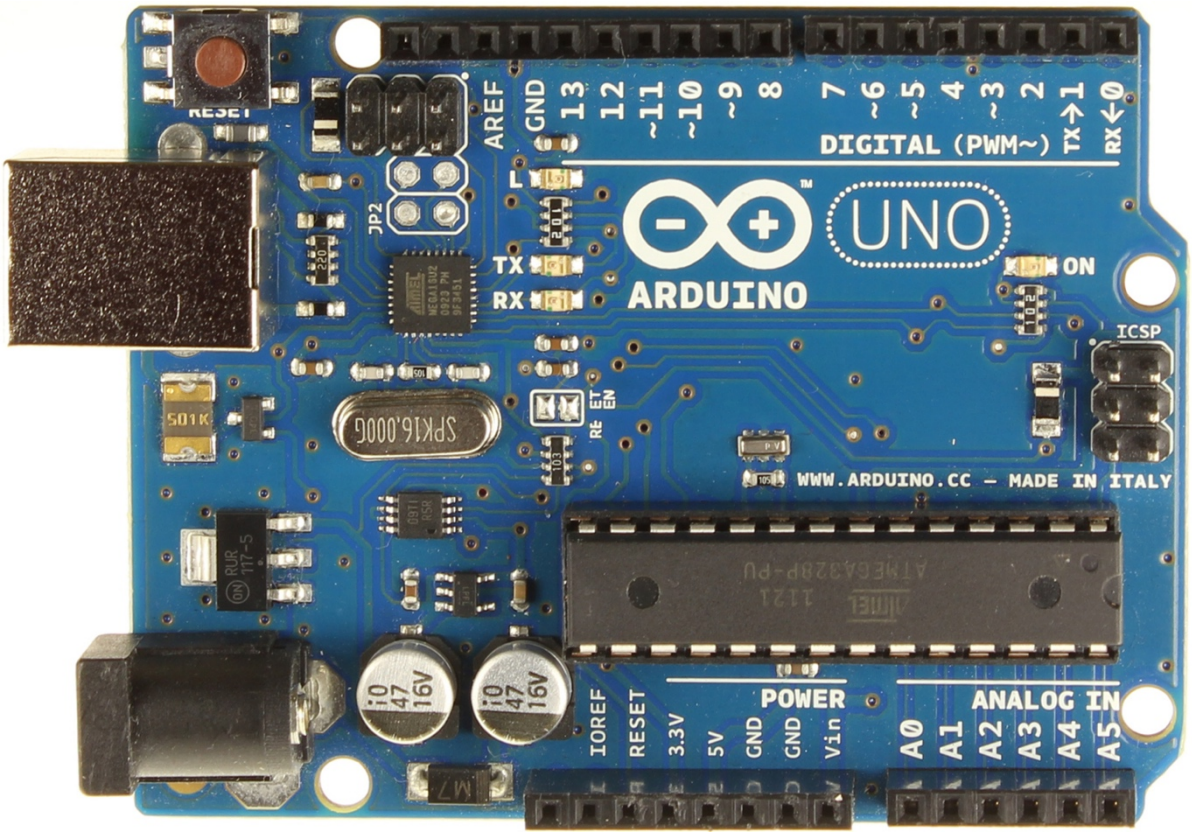
Programiranje robota se vrši preko privjeska za učenje. Privjesak za učenje je složeni ručni uređaj preko kojega se mogu kreirati novi programi, učitavati i prepravljati postojeći, može se direktno pomicati robota, obavljati kalibracije te mnoge druge funkcije. Ima zaslon u boji na kojem se može otvoriti do tri prozora, što uvelike olakšava programiranje i praćenje interakcija.

Slika 11 - Privjesak za učenje, iPendant

2.4 Arduino

Arduino je elektronička platforma otvorenog koda (*eng. Open Source Platform*) temeljena na sklopovlju (engl. Hardware) i programskoj podršci (engl. Software) koja je prilagodljiva i jednostavna za korištenje. Malene dimenzije obrnuto su proporcionalne mogućnostima realizacije mnogobrojnih tehničkih sustava.

Na Arduino elektroničkoj pločici je 8-bitni ATmega328P mikrokontroler sa pripadajućim komponentama koje omogućavaju programiranje i povezivanje s mnogobrojnim senzorima, modulima, aktuatorima i sl. Uređaj je moguće napajati iz univerzalne serijske sabirnice (engl. Universal Serial Bus – USB) ili drugim vanjskim izvorom istosmjernog napona od 5 do 12V. Velika prednost Arduina je normiran raspored kontakata koji omogućava lako povezivanje s dodatnim modulima. Kontakti 0 i 1 služe za serijsku TTL komunikaciju.

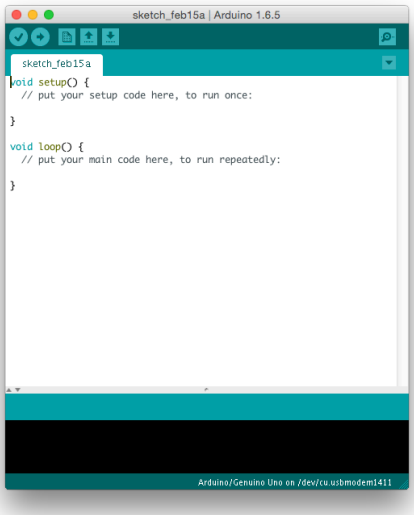


Slika 12 - Arduino Uno

Mikrokontroler	**ATmega328P**
Radni napon	5V
Napon napajanja (preporučeni)	7-12V
Digitalni ulazi/izlazi	14
PWM digitalni ulazi/izlazi	6
Analogni ulazi	6
Struja na ulazu/izlazu	20mA
Struja na 3.3V	50mA
Flash memorija	32kB (0.5kB koristi bootloader)
SRAM	2kB
EEPROM	1kB
Oscilator	16 MHz
Duljina	68.6mm
Širina	53.4mm
Masa	25g

Tablica 3 - Tehničke specifikacije Arduina Una

Mikrokontroler se programira putem Arduino programskog jezika nazvanog Arduino IDE koji je napisan u programskom jeziku Processing, isti je kombinacija Jave, C-a i C++-a. Temeljen je na C++ knjižnici Wiring koja čini uobičajene ulazno izlazne operacije veoma jednostavnim. Arduino IDE ujedno služi za pisanje programa i za prenošenje samog programa na mikrokontroler. Za prenošenje koda na mikrokontroler, Arduinu za razliku od većine ostalih mikrokontrolera nije potreban fizički programator. Na samom Arduinu nalazi se Arduino „bootloader“, mali program koji služi za prenošenje programa na sam mikrokontroler bez dodatnog uređaja.



Slika 13 - Arduino IDE

2.4.1 Programiranje u Arduinovom programskom jeziku

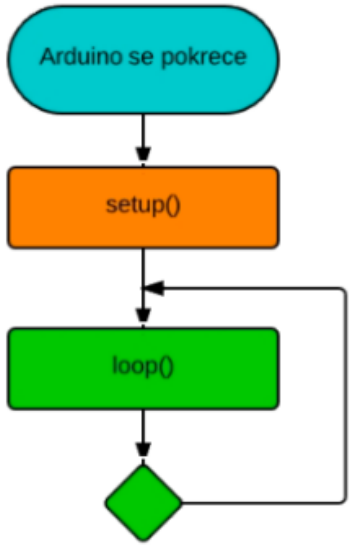
Na samom početku kreiranja novog programa mora se, ukoliko je to potrebno, uvesti određena vanjska knjižnica naredbi. U tim knjižnicama je veliki broj prethodno napisanih naredbi u C++ programskom jeziku koje uvelike olakšavaju i ubrzavaju proces programiranja. Ako je primjerice potrebno izračunati kvadratni korijen nekog broja, nije potrebno unositi komplicirani algoritam za takovu operaciju, dovoljno je pozvati knjižnicu *math.h* u kojoj se nalazi široka skupina matematičkih algoritama, kvadratni korijen jednostavno se računa naredbom *sqrt()*.

Slijedeći korak je deklariranje varijabli. Varijable mogu biti različitih tipova, ovisno o tome kakvog tipa je podatak koji u nju pohranjujemo. Najčešće korištene varijable su slijedećih tipova: Interger (cjelobrojni broj), Float (decimalni), Char (znak), String (skup znakova - riječ).

Svaki Arduino program mora imati dvije glavne naredbe kako bi se ispravno izvršavao: *setup()* i *loop()*.

Naredba *setup()* koristi se za postavljanje ulazno-izlaznih portova (engl. I/O ports) kao što su svjetleće diode (engl. Light Emitting Diode), senzori, motori, serijski portovi ili pak uspostavljanje neke veze, npr. serijske ili Ethernet. *setup()* funkcija se izvršava samo jednom i to nakon što je Arduino pokrenut ili ponovno pokrenut (engl. restart).

Po izvršenju *setup()* funkcije program nailazi na *loop()* funkciju, riječ je o beskonačnoj petlji u kojoj se kontrolira ulazno izlazne portove pomoću uobičajenih funkcijskih petlji, primjerice: *if ()*, *for ()*, *while ()*, *switch()*.



Slika 14 - Dijelovi programskog koda

2.5 TCP/IP komunikacija

TCP/IP je oznaka za grupu protokola pomoću kojih se izmjenjuju podaci između računala. Naziv potječe od dva najvažnija protokola te skupine TCP (engl. Transmission Control Protocol) te prema IP (engl. Internet Protocol) protokolu. TCP/IP omogućava povezivanje mrežnih čvorova određivajući kako se podaci moraju konvertirati u pakete, adresirati, slati, prenositi te primati na odredištu. TCP/IP spada u treći i četvrti sloj OSI referentnog modela (engl. Open Systems Interconnection Basic Reference Model). OSI je najkorišteniji apstraktni opis arhitekture mreže. Opisuje komunikaciju sklopovlja, aplikacija i protokola pri mrežnim komunikacijama. Koriste ga proizvođači pri projektiranju mreža, kao i stručnjaci pri proučavanju mreža.

Fizički sloj definira električka i fizička svojstva mrežnih uređaja, naponske razine, brojeve kontakata te uređaje kao što su ponavljači, mrežni koncentratori itd.

Podatkovni sloj kontrolira razmjenu podataka između mrežnih uređaja i korigira možebitne greške na fizičkom sloju.

Mrežni sloj je zadužen za pretvaranje logičke IP adrese u fizičku MAC adresu kao bi podaci stigli od jednog mrežnog čvora do drugog.

Prijenosni sloj se brine o paketima koji putuju između dva računala. Primjeri protokola na prijenosnom sloju su TCP i UDP. U slučaju nestanka nekog paketa, TCP će zatražiti od pošiljatelja da ponovno pošalje taj isti paket

Sloj sesije bavi se uspostavom i sinkronizacijom veze između krajnjih korisnika.

Prezentacijski sloj služi za izvođenje kodiranja za sustav koji koristimo npr. TXT-datoteke na Unix-u, Mac OS-u i Windowsima na različite načine označavaju prelazak u novi red. Prezentacijski sloj zadužen je za usklađivanje kodiranja.

Aplikacijski sloj je najbliži korisniku, pruža mrežne usluge korisničkim aplikacijama. Od ostalih slojeva OSI modela razlikuje se po tome što ne pruža usluge drugim slojevima, već samo aplikacijama van OSI modela.

OSI referentni model
7. aplikacijski sloj
6. prezentacijski sloj
5. sloj sesije
4. prijenosni sloj
3. mrežni sloj
2. podatkovni sloj
1. Fizički sloj

Tablica 4 - OSI model

Korisnik/poslužitelj (engl. Client/Server) komunikacija

Svako računalo i usmjernik (engl. Router) ima jedinstvenu IP adresu, 32 bitni broj, podijeljen u 4 grupe, od kojih svaka sadrži jedan 8 bitni broj; na primjer, 192.168.123.55. Te adrese su nužne da bi se paketi upućeni s izvorišnog računala mogli preusmjeriti do odredišnog. Osnovna je zamisao da postoji jedno centralno računalo koje ima ulogu poslužitelja i koje obavlja dio posla te pruža podatke i izvršava zahtjeve drugog računala koje nazivamo korisnik, odnosno računalnog programa na tom računalu. Takvim rješenjem računalo koje postavlja zahtjeve (korisnik) zapravo ima ulogu korisničkog sučelja za program koji se izvršava na centralnom računalu, poslužitelju.

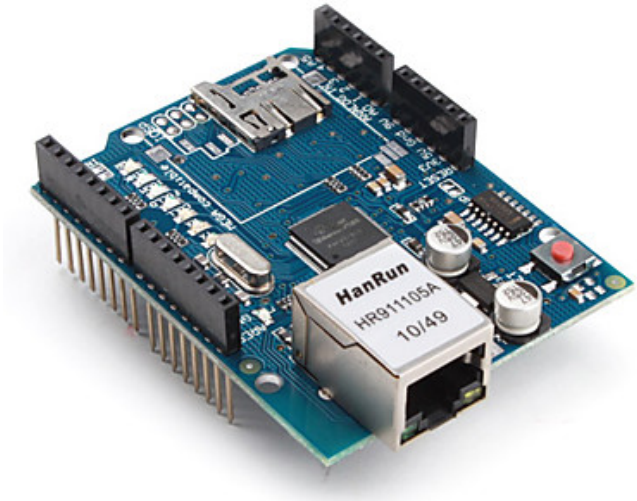
Budući da na jednom poslužitelju najčešće ima više posluživačkih aplikacija koje nude usluge korisnicima (web server, ssh, ftp, itd.) potrebno je razlikovati različite aplikacije na poslužitelju. To se čini pomoću jednog broja koji se naziva port i koji je pridružen svakoj poslužiteljskoj aplikaciji. Kada korisnik želi započeti komunikaciju s nekim poslužiteljem, on otvara utičnicu (engl. socket). Socket je apstrakcija mrežne programske podrške koji vrši komunikaciju preko mreže. Sa stanovišta aplikacijskog programera komunikacija se sastoji od

pisanja u socket i čitanja iz njega. Može ga se opisati kao jedan od dva kraja na kanalu komunikacije između poslužitelja i korisnika.

Principi rada:

- Aplikacija (na poslužitelju) radi na određenom računalu (zadanom s IP adresom) i "osluškuje" na unaprijed određenom portu
- Korisničkoj aplikaciji je poznata adresa poslužitelja i port na kojem sluša te mu šalje zahtjev za uspostavu komunikacije (pokušava otvoriti socket za komunikaciju)
- Kada je poslužitelj spreman komunicirati s korisnikom, on otvori novi socket (različit od onog na kojem osluškuje) te sada poslužitelj i korisnik komuniciraju kroz tako otvoreni kanal komunikacije.
- Kada je komunikacija završena i poslužitelj i korisnik zatvaraju svoje sockete i na taj način se zatvaraju kanali komunikacije.

2.5.1 Mrežni štit (engl. Ethernet Shield)



Slika 15 - Ethernet štit

Arduino mrežni štit omogućuje Arduinovo spajanje na mrežu. Zasnovan je na Wiznet W5100 Ethernet čipu koji omogućava mrežnu komunikaciju koristeći TCP i UDP protokole. Podržava do četiri istovremena socket povezna kanala. Za pisanje programa koji koriste ovaj

shield dostupna je Ethernet knjižnica. Povezivanje s Arduinom je izvedeno tako da su svi kontakti s Arduinove elektroničke pločice dostupni i na priključenom štitu, napajanje štita je također riješeno kroz spomenute kontakte. Time je omogućeno spajanje modula uz zadržavanje osnovnog rasporeda kontakata. Mrežni štit ima standardni RJ-45 mrežni priključak, s integriranim transformatorom i PoE (engl. Power over Ethernet) mogućnostima. Na elektroničkoj pločici se nalazi i utor za micro-SD karticu, koja se koristi za čuvanje datoteka koje će biti poslane preko mreže. Arduino komunicira sa W5100 i SD karticom SPI komunikacijom (engl. Serial Peripheral Interface). Taj port se nalazi na kontaktima 11, 12 i 13. Kontakt broj 10 se koristi za odabir W5100, a kontakt broj 4 za SD karticu. Iz tog razloga se ti kontakti ne mogu koristiti kao standardni ulazi/izlazi. Pošto W5100 I SD kartica dijele isti SPI (engl. Serial Peripheral Interface) priključak, ne mogu se koristiti u isto vrijeme. Tipka za ponovno pokretanje (engl. Reset) na štitu resetira i W5100 i osnovnu Arduino elektroničku pločicu.

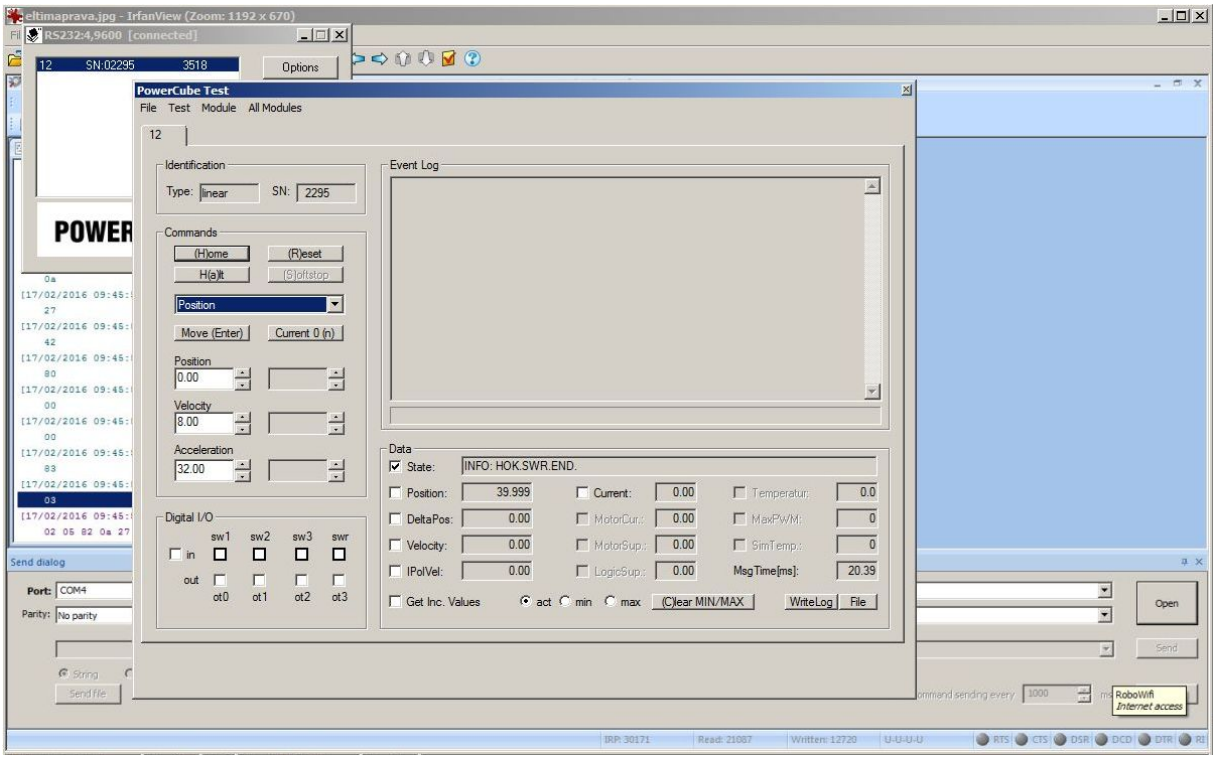
Na elektroničkoj pločici se nalazi nekoliko LED indikatora:

- PWR: označava da su Arduino i štit uključeni
- LINK: označava prisustvo mrežne konekcije i treperi kada štit šalje ili prima podatke
- FULLD: označava da mreža istovremeno izmjenjuje podatke u oba smjera (engl. Full - Duplex)
- 100M: označava prisustvo 100 Mb/s mrežne konekcije (u suprotnom brzina je 10 Mb/s)
- RX: treperi kada štit prima podatke
- TX: treperi kada štit šalje podatke
- COLL: treperi u slučaju kolizije na mreži

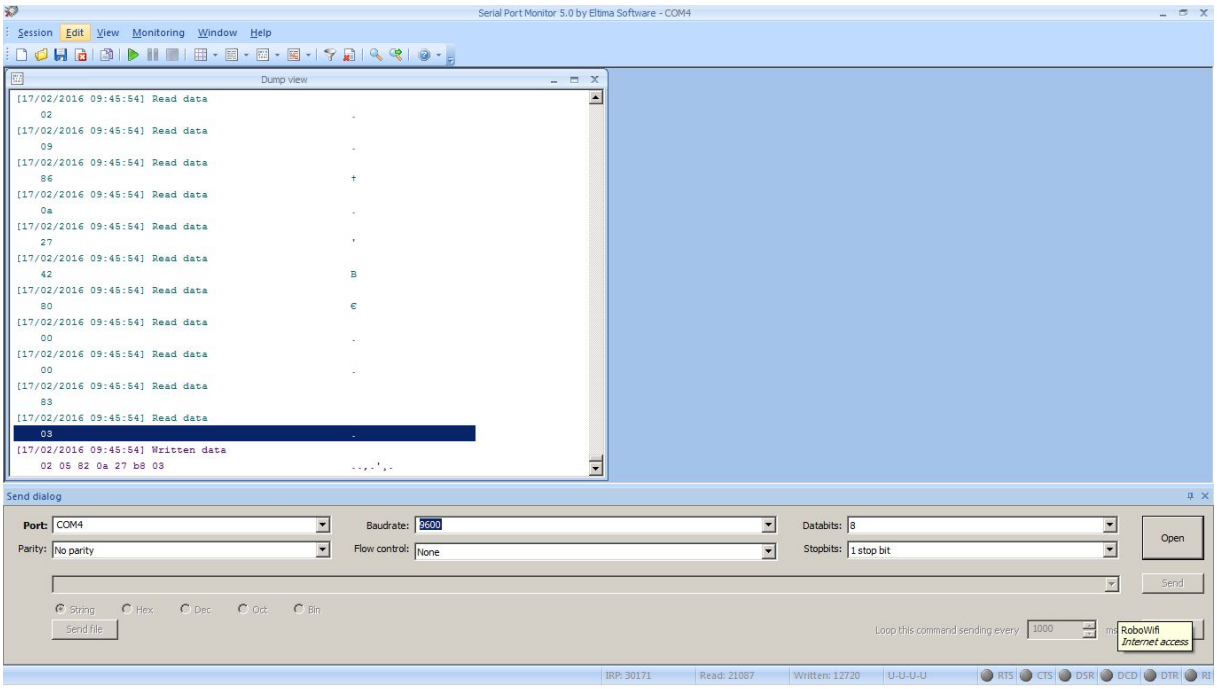
Postoje dvije klase *korisnik (engl. client)* i *poslužitelj (engl. server)*. Klasa *korisnik* se koristi za kreiranje korisnika na Arduinu koji se spaja na neki vanjski poslužitelj i s njime razmjenjuje podatke. Klasa *Poslužitelj* se koristi za kreiranje poslužitelja na Arduinu koji šalje i prima podatke od vanjskih korisnika koji su priključeni na njega.

3 Razvoj upravljačkog programa

Da bi se moglo krenuti s razvojem Arduino programa prvo je potrebno shvatiti način upravljanja i komunikaciju s hvataljkom. Na internetskoj stranici proizvođača hvataljke dostupna je PowerCube probna aplikacija koja se koristi za probno slanje naredbi u hvataljku. Za vrijeme slanja naredbi iz 'PowerCube' aplikacije u hvataljku, računalo bi u pozadini imalo pokrenutu aplikaciju 'Serial Monitor' koja bi pratila i spremala poruke koje prolaze serijskom vezom. To je jedan od načina da se stekne uvid u poruke koje hvataljka očekuje i kakve šalje uređaju koji s njome upravlja.



Slika 16 - PowerCube aplikacija



Slika 17 - Serial Port Monitor aplikacija

3.1.1 Oblik naredbi za hvataljku

Podaci se prenose u INTEL (Little endian) obliku. Little endian zamjenjuje redoslijed zapisa podatka u memoriju. Najmanje značajan oktet (engl. byte) je spremljen na najnižu adresu u memoriji, a najznačajniji na najvišu. Heksadecimalni broj će biti spremljen ovako:

0x12345678	0x78	0x56	0x34	0x12

Tablica 5 – Primjer zapisa heksadecimalnog broja u Little endian formatu

Poruka za hvataljku uvijek počinje sa znakom '02' i završava znakom '03'. Ako bi se ti znakovi trebali koristiti za zapis podataka bilo gdje drugdje unutar poruke, da ne bi došlo do nedefiniranog stanja potrebno ih je zamijeniti slijedećim kombinacijama znakova:

02→ 10 82

03→ 10 83

10→ 10 90

BCC - Block Check Charakter (Check Sum) služi za provjeru ispravnosti poruke. Njegova vrijednost je zbroj svih znakova u poruci osim znakova za početak ('02') i kraj poruke ('3'), dolazi na predzadnjem mjestu u poruci.

Nakon znaka '02' dolazi znak koja govori da li je poruka usmjerena hvataljci ili je odaslana iz nje. Ako je u poruci na drugom mjestu znak '5' poruka je namjenjena za hvataljku, a ako je znak '9' poruka je odaslana iz hvataljke.

Znakovi korišteni na trećoj poziciji su '86', '82' i '81'. Znak '86' dolazi u porukama u kojima će slijedeći znakovi u poruci određivati parametre gibanja (npr. u poruci u kojoj će se zadati određena brzina gibanja mehaničkih prstiju hvataljke).

Znak '82' dolazi u porukama koje su odaslane iz hvataljka, a šalju informaciju o nekom parametru (npr. na koji razmak su postavljeni mehaničke prsti).

U porukama u kojim su svi znakovi uvijek jednaki na trećem mjestu dolazi znak '81'.

Slijedeći znak koji dolazi je tzv. **CommandID**. On određuje vrstu operacije koja će biti poslana u hvataljku. U slijedećoj tablici su prikazane i objašnjene moguće vrijednosti.

Naredba	**Command ID**	**Duljina**	**Opis**
Reset	0x00	1 Byte	Brisanje greški
Home	0x01	1 Byte	Početna pozicija
Halt	0x02	1 Byte	Stop
SetExtended	0x08	3-6 Byte	Postavljanje parametara
GetExtended	0x0a	2 Byte	Informacija iz hvataljke
SetMotion	0x0b	6-8 Byte	Postavljanje načina gibanja

Tablica 6 – CommandID

Naredbe Reset (brisanje grešaka), Home (pozicioniranje u početni položaj), i Halt (trenutno zaustavljanje) su određene istim znakovima i nikad se ne mijenjanju.

Reset: 02 05 81 00 86 03

Home: 02 05 81 01 87 03

Halt: 02 05 81 10 82 88 03

'SetExtended' ('08') dolazi u porukama u kojima se definira određena brzina ili ubrzanje kojim će se gibati mehanički prsti hvataljke. Ako se postavlja brzina, nakon znaka '08' dolazi znak '4f', a ako se postavlja ubrzanje na petom mjestu u poruci mora biti znak '50'.

'GetExtended' ('0a') dolazi u porukama u kojima se šalje zahtjev hvataljci da vrati vrijednost nekog parametra.

'SetMotion' ('0b') se postavlja u porukama koje pokreću gibanje. Ako se želi odabrati gibanje s prethodno definiranim brzinom i ubrzanjem nakon znaka '0b' dolazi znak '04', a ako se želi gibanje određenom strujom koja neće mijenjati svoj iznos sve dok ne pošaljemo drugu naredbu iza znaka '0b' dolazi znak '08'.

Slijedeća tablica prikazuje koji heksadecimalni znakovi predstavljaju određeni iznos razmaka, brzine odnosno ubrzanja mehaničkih prstiju hvataljke. Tako je npr., za razmak mehaničkih prstiju od 31.4mm heksadecimalna kombinacija 00 3d. Ista ta heksadecimalna kombinacija je ekvivalentna za brzinu iznosa 31.4mm/s i za ubrzanje iznosa 31.4mm/s^2. Vrijednosti koje su između vrijednosti prikazanih u tablici se izračunavaju linearnom interpolacijom.

Iznos razmaka, brzine odnosno ubrzanja	Heksadecimalna kombinacija
0	00 00
0.48	00 3a
1.96	00 3b
7.84	00 3c
31.4	00 3d
69.4	8e 3d
80	a3 3d
125	00 3e
320	a3 3e

Tablica 7 - Prikaz vrijednosti razmaka, brzine odnosno ubrzanja i heksadecimanih znakova koji zamjenjuju te iznose u porukama poslanim hvataljci

3.1.1.1 Primjer naredbe

Za pozicioniranje mehaničkih prstiju hvataljke na razmak od 30mm, brzinom 20mm/s i ubrzanjem od 8mm/s^2 robotskoj hvataljci se šalju sljedeće poruke:

02 05 86 08 4f 00 00 a3 3c c2 03

02 05 86 08 50 00 00 10 83 3c 23 03

02 05 86 0b 04 00 00 f5 3c cc 03

Sve tri gore navedene poruke počinju znakom za početak poruke '02', a završavaju znakom za kraj poruke '03'. Znakovi na drugom i trećem mjestu su također jednaki u sve tri poruke, oni govore da će u hvataljku biti poslana naredba gibanja. Četvrti znak određuje da li je to poruka za postavljanje parametara '08' ili pak ona koja će pokrenuti gibanje '0b'.

U prvoj poruci definirana je željena brzinu gibanja. Da bi se moglo zadati brzinu, peti znak u poruci mora biti ['4f'] . Iznos brzine je određen sa dva ekvivalentna znaka na osmom i devetom mjestu u poruci. Predzadnji član u poruci je BCC, njegova vrijednost je zbroj svih članova osim prvog i zadnjeg: 05+86+08+4f+a3+3c odnosno u dekadskom sustavu: 5+134+8+79+163+60 što iznosi 449, ali BCC mora biti broj između 0 i 255 pa se od zbroja oduzima 255 sve dok se ne završi između tih vrijednosti. 449-255=194 odnosno u heksadecimalnom sustavu 'c2'.

Drugom porukom se definira željeno ubrzanje gibanja. Da bi se moglo zadati ubrzanje peti znak u poruci mora biti ['50']. Ubrzanje od 8mm/s^2 je određeno znakovima '03' i '3c'. Ali znak '03' ne smijemo upisati unutar poruke jer je njegova funkcija prekidanje poruke i uvijek treba biti na kraju. Zamjenjujemo ga znakovima '10' i '83'. Računanje BCC-a je sada malo drugačije, u zbroj ne ulaze 10 i 83, nego se umjesto njih pribraja 3 čija je vrijednost prvotno tu trebala i biti. 5+134+8+80+**3**+131+60 što iznosi 290, ali vrijednost joj mora biti manja od 255. 290-255=35 odnosno 23 u heksadecimalnom.

Treća poruka će pokrenuti gibanje znakom '0b' upisanim na četvrto mjesto. Na 8. i 9. mjesto dolaze znakovi koji predstavljaju razmak mehaničkih prstiju od 30mm – 'f5' i '3c'. BCC se izračunava na isti način: 5+134+11+4+245+60=459. 459-255=204 odnosno 'cc' u heksadecimalnom brojevnom sustavu.

3.1.2 Dijelovi Arduino upravljačkog programa

```
#include <SPI.h>
#include <Ethernet.h>
#include <math.h>
```

Slika 18 - Uvoz knjižnica

Na početku programa su pozvane tri knjižnice naredbi. Taj poziv se ostvaruje naredbom *#include*.

SPI.h je knjižnica koja Arduinu omogućuje komunikaciju s SPI (engl. Serial Peripheral Interface) uređajima i treba ju pozvati ako koristimo mrežni štit jer je on SPI uređaj. SPI je vrsta brze serijske komunikacije koja se koristi za manje udaljenosti.

Osim Ethernet štita, za povezivanje na mrežu nam treba i knjižnica *Ethernet.h* koja omogućuje da Arduino bude poslužitelj ili korisnik. Moguće su četiri istovremene konekcije: dolazne, odlazne ili kombinacije istih.

U *math.h* knjižnici su spremljene matematičke funkcije.

```
byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
IPAddress ip(192, 168, 123, 50);
IPAddress server(192, 168, 123, 25);
EthernetClient client;
```

Slika 19 - Definiranje adresa i Arduina klijentom

mac[] je adresa Ethernet adaptera Arduino uređaja, nalazi se na naljepnici na Arduinu.

U *ip* varijablu je spremljena IP adresa Arduina, a u *server* IP adresa robota. S funkcijom *EthernetClient* arduino je postavljen za korisnika, odnosno poslužitelj će biti upravljačka jedinica robota.

Nakon toga slijedi deklaracija glavnih i pomoćnih varijabli te petlja *setup()*.

```
void setup() {
  Serial.begin(9600);
  for (i = 0; i < 15; i = i + 1)
    Serial.write(reset[i]);
  for (i = 0; i < 15; i = i + 1)
    Serial.write(HOME[i]);
  Ethernet.begin(mac, ip);
  delay(1000);
  client.connect(server, 5555);
}
```

Slika 20 - setup() petlja

Funkcija *Serial.begin()* pokreće serijsku komunikaciju. U zagradu se upisuje brzina prijenosa. Najčešće su: 4800, 9600, 14400, 19200. S obzirom da se *setup()* uvijek izvršava samo nakon što je Arduino pokrenut ili ponovno pokrenut (engl. restart) korisno je u hvataljku poslati naredbe za brisanje eventualnih grešaka i naredbu za pozicioniranje u početni položaj.

Rad s mrežom započinje s pokretanjem naredbe *Ethernet.begin (mac, ip, dns, gateway, subnet);*. Mora se navesti jedino *mac* adresa, ostale se mogu dodijeliti automatski. Opis argumenata je:

- **mac** je adresa Ethernet adaptera Arduino uređaja
- **ip** je IP adresa Arduina. Ne mora se navesti ako koristimo uređaj koji koristi DHCP (engl. Dynamic Host Configuration Protocol) - mrežni protokol korišten od strane mrežnih računala za dodjeljivanje IP adresa i ostalih mrežnih postavki
- **gateway IP** je adresa mrežnog prolaza (engl. Gateway)
- **subnet** je maska mreže

S *client.connect()* spajamo Arduino na *ip adresu* robota i port 5555.

```
void loop() {
  while (client.available()) {
    char newchar = client.read();
    if (newchar == 0x4b) {
      funkcija(commandString);
```

Slika 21 - Početak loop() petlje

Upravljačka jedinica robota Arduinu šalje poruku u ovome obliku: Pa@b@c@dK

'P' označava početak poruke, *'a'* željenu brzinu gibanja hvataljke, *'b'* željeno ubrzanje hvataljke, *'c'* željeni razmak mehaničkih prstiju hvataljke, *'d'* silu kojom se želi zatvoriti hvataljka dok *'K'* označava kraj poruke. Sa znakovima *'@'* se razdvaju parametri.

Ako se za brzinu i ubrzanje ili položaj unese veći iznos od maksimalnog, smatrati će se da je korisnik htio najveće moguće iznose te će se oni i poslati u hvataljku. Ako se za brzinu i ubrzanje unese nula, za brzinu će se poslati 10mm/s, a za ubrzanje 15mm/s^2. Kada korisnik unese iznos za silu, ostali parametri se ne razmatraju.

Ostale definirane poruke:

"P500@0@0@0" – pozicioniranje u početni položaj

"P600@0@0@0" – brisanje greški

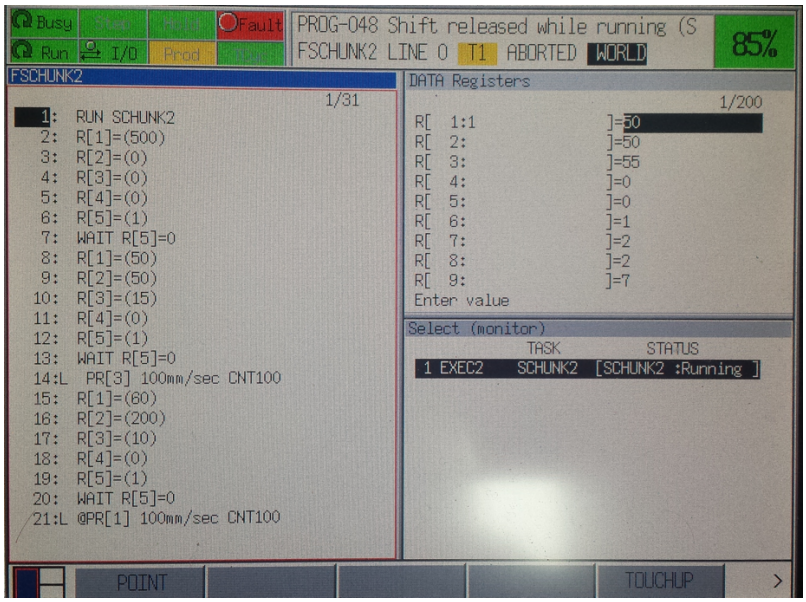
"P700@0@0@0" – dozvoljava ručno pomicanje mehaničkih prstiju hvataljke

"P800@0@0@0" – informacije o položaju i sili

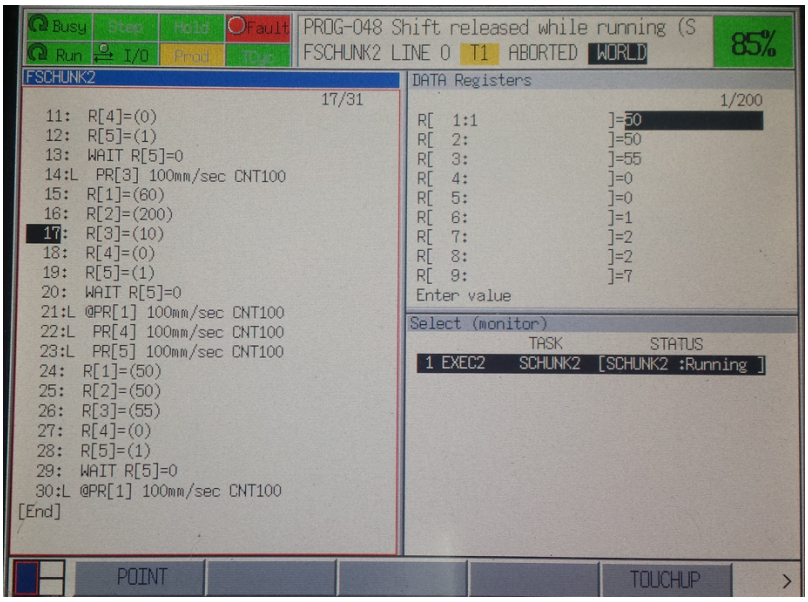
Uz pomoć naredbe *client.avaliable()* ispituje se da li ima podataka poslanih od poslužitelja prema korisniku. Sve dok podaci pristižu, znak po znak se čita i sprema u varijablu tipa String 'commandString'. Kada se upiše znak *'K'*, ('*4B'* u heksadecimalnom zapisu) simbol za kraj poruke, String se šalje u funkciju *'funkcija'*.

U toj funkciji se ispituje kakva je poruka stigla. Nakon što se otkrila funkcija pristigle naredbe, ta naredba se i izvršava. Ako je riječ o naredbi gibanja, program treba napraviti kodove za zadane parametre i poslati ih pravilnim redoslijedom u robotsku hvataljku. Nakon njihovog slanja u hvataljku, hvataljci se šalje zahtjev da nam vrati poruku iz čijeg sadržaja je vidljivo da je gibanje završilo. Poslije dobivanja potvrdnog odgovora, u hvataljku se šalje zahtjev da vrati položaj i iznos sile. Na kraju se prazni varijabla tipa String 'commandString' u koji se sprema poruka iz upravljačke jedinice, vračamo se na početak programa i čekamo novu naredbu iz poslužitelja (upravljačke jedinice robota).

3.1.3 Programiranje robota



Slika 22 – Pick and Place 1/2



Slika 23 – Pick and Place 2/2

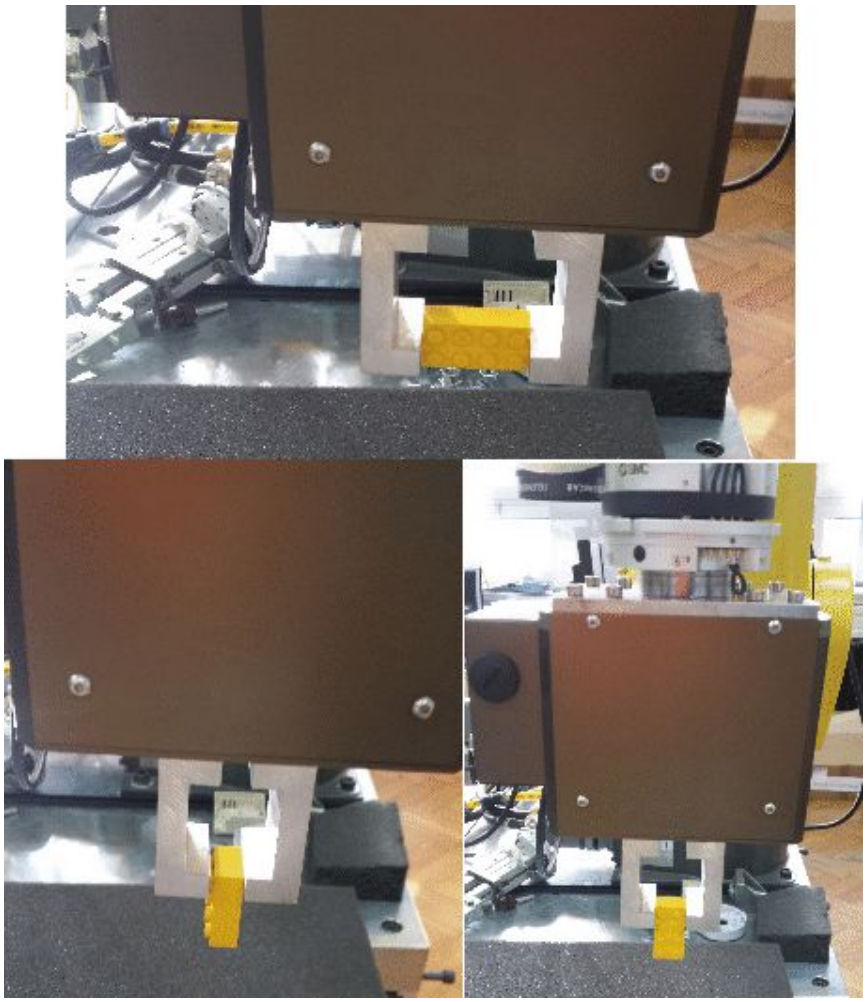
Na slikama 22 i 23 je prikazan zaslon Fanucovog privjeska za učenje. Zaslon je podijeljen na 3 prozora. Na lijevom, najvećem je prikazan jednostavan program. On obavlja funkciju prenošenja predmeta s jednog mjesta na drugo (engl. Pick and Place). Na početku programa se poziva *'SCHUNK2'* jednostavan program napisan programskim jezikom KAREL koji se nakon pokretanja konstantno izvodi u pozadini. KAREL programski jezik je jezik niske razine sličan programskom jeziku Pascal. KAREL programi se pišu na vanjskom računalu pomoću kojega se i prenose u upravljačku jedinicu robota . Na upravljačkoj jedinici

robota KAREL programi se ne mogu otvarati i preuređivati. *'SCHUNK2'* osigurava mrežno povezivanje i izmjenu podataka s Arduinom. Željeni parametri za hvataljku spremaju se u registre. U FANUC operativnom sustavu oznaka registra je 'R', imaju dva parametra, redni broj registra i vrijednost koja je u njega pohranjena. Redni broj se nalazi u uglatim zagradama iza oznake R, a vrijednost zapisana u registar poslije znaka '=' – npr. R[1]=60. U registar 1 upisujemo željenu brzinu, u registar 2 ubrzanje, u registar 3 položaj, a u registar 4 iznos sile. U registar 5 upisujemo znak *'1'*, on je okidač za slanje poruke. Kada se gibanje hvataljke obavi, vrijednost u registru 5 se vraća u *'0'* i program se može nastaviti, dočekan je uvjet u WAIT funkciji. Za lakše praćenje tijeka događaja, na desnoj strani ekrana moguće je stalno nadgledati vrijednosti koje se mijenjaju u registrima i status KAREL programa.

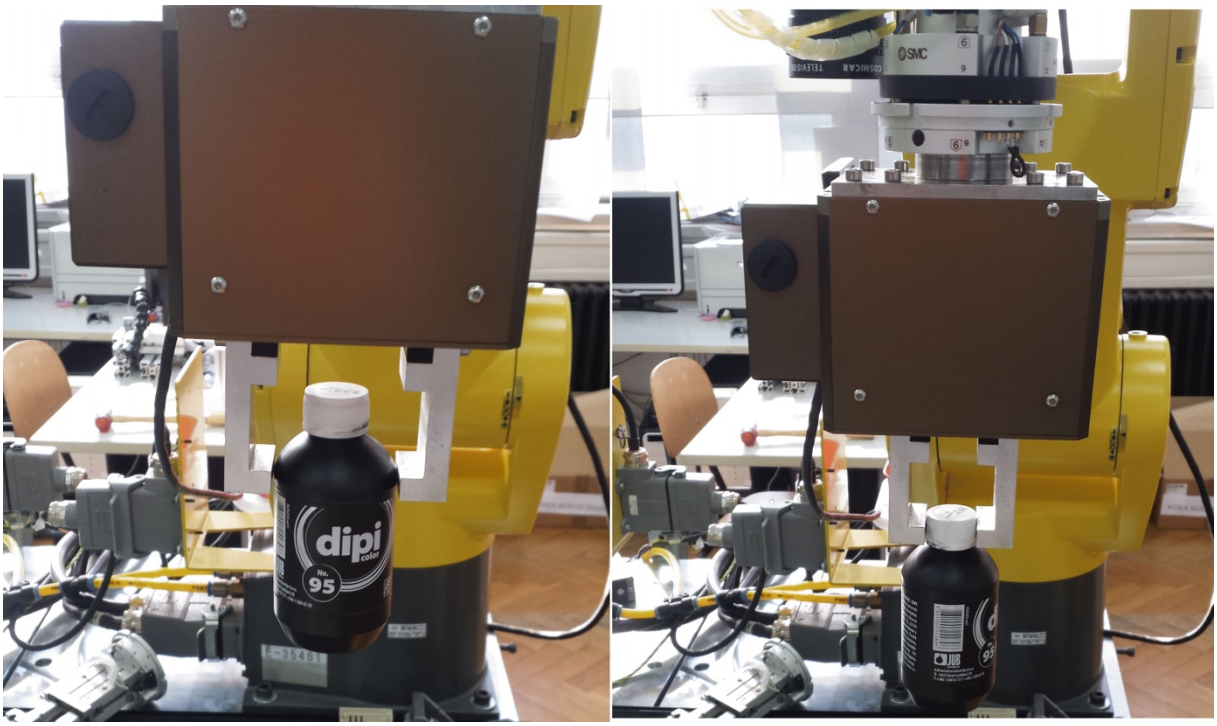
3.1.4 Laboratorijska provjera

Nakon razvoja upravljačkog programa, hvataljka je postavljena na robotsku ruku te su potom isprobane njene funkcije. Hvataljkom se uspješno upravljalo s privjeskom za učenje robota. Hvatanje stezanjem zahtijeva gibanje mehaničkih prstiju hvataljke u svrhu postizanja neophodne stezne sile.

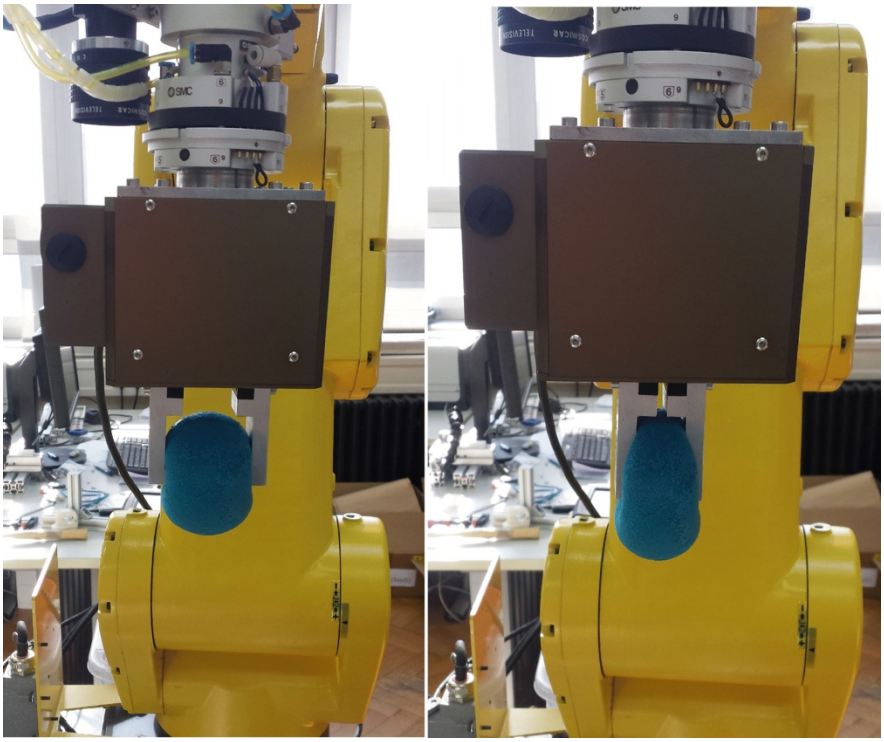
Hvataljkom je prenošena ista LEGO kockica tri puta, ali ju je svaki puta uhvatila za drugu dimenziju (slika 24). Izrazito je bitan čvrst kontakt između prstiju hvataljke i predmeta kojeg prenosi jer gibanje predmeta uslijed inercije ili vlastite težine ne smije biti moguće za vrijeme njegovog prenošenja. Stabilnost predmeta u hvataljci se mora osigurati efektivnom steznom silom u dodirnim točkama ili aktivnoj površini između radnog dijela i prstiju hvataljke. U testiranju se prenosila i bočica valjkastog oblika, prvo uhvaćena za širi dio samog tijela, a potom i za poklopac koji je također cilindrični, ali manjega promjera (slika 25). Teoretska površina dodira između bočice i mehaničkih prstiju hvataljke je određena linijom. No usprkos maloj dodirnoj površini hvataljka je mogla ostvariti dovoljno veliku silu za prihvatljivo prenošenje bez neželjenih pomaka. Ako primjena najveće moguće sile kojem hvataljka može djelovati na tijelo koje prenosi nije dovoljna za stabilno prenošenje na hvataljku se mogu postaviti drugi prsti koji svojim oblikom osiguravaju veću aktivnu površinu i time smanjuju potrebnu steznu silu (slika 27). Iz toga se zaključuje da hvataljci dimenzije i oblici ne stvaraju problem kod hvatanja predmeta zadanom silom. Hvataljku se također testiralo s prenošenjem predmeta najveće dopuštene mase što je hvataljaka isto uspješno odradila. Na slici 26. je prikazano djelovanje hvataljke različitim silama. Prvo se primjenila najmanja moguća sila od 30N, a potom najveća moguća od 200N, razlika se primjećuje na deformaciji spužvaste loptice. Hvataljkom je moguće prenijeti i šuplja tijela s kojima bi kontakt ostvarila na unutarnjim plohama. U tom slučaju silom djelujemo u suprotnom smjeru (slika 28).



Slika 24 - Hvataljka i LEGO kocka

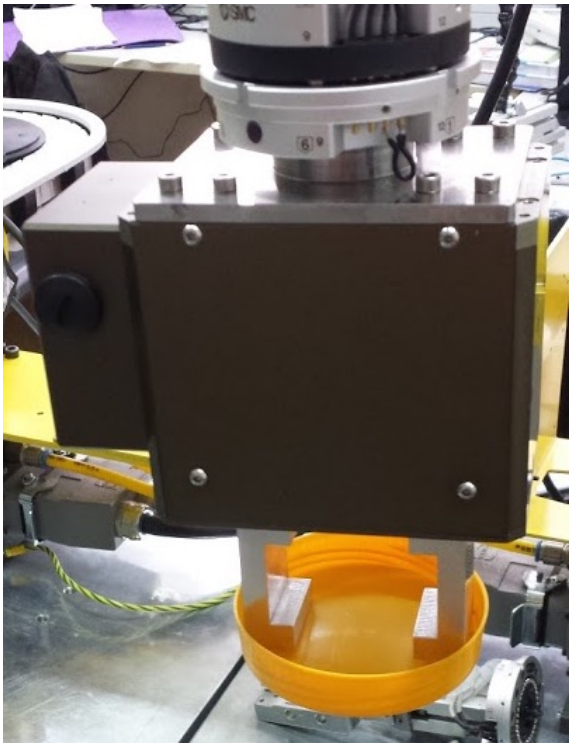


Slika 25 - Hvataljka i valjkasta boca



Slika 26 - Pritisak hvataljke različitim silama

Slika 27 - Mehanički prst hvataljke prikladan za hvatanje cilindričnih tijela



Slika 28 - Prenošenje predmeta s kontaktom ostvarenim na unutarnjim plohama

4 Zaključak

Hvataljka Schunk PT-AP 70-R kvalitetna je hvataljka velikih mogućnosti, no ima manu što se ne može priključiti na moderne robote zbog zastarjele vrste komunikacije. Da bi se mogla i dalje koristiti, ovim radom napravljen je međusklop koji povezuje do sada češće korištenu RS232 serijsku komunikaciju s danas najčešćim oblikom komunikacije temeljenim na računalnim mrežnim protokolima, odnosno- TCP/IP protokolom.

TCP omogućuje jednostavnije korištenje i implementaciju raznih rješenja te je upravo iz tog razloga potisnuo ostale načine komunikacije i postao dio svakodnevnog života kako u industrijskoj tako i u primjeni za komunikaciju između uređaja koje koristi gotovo svako današnje kućanstvo (pisaći, mobiteli, računala).

Sustav je uspješno realiziran pomoću Arduino platforme. Arduino sa svojim jednostavnim sklopovljem i s velikom bazom primjera omogućava brzo savladavanje osnova rada. Velik broj gotovih knjižnica dodatno olakšava spajanje i upotrebu raznih uređaja. Važno je spomenuti i vrlo pristupačnu cijenu. Zajedno Arduino i oba štita koštaju 160kn što je gotovo cijeli iznos koji je potrošen na izradu ovoga rada.

Hvataljka je postavljena na robota i uspješno provjerena njena primjena.

Literatura

[1] https://en.wikipedia.org/wiki/RS-232

[2] https://hr.wikipedia.org/wiki/TTL

[3] https://www.sparkfun.com/tutorials/215

[4] https://web.math.pmf.unizg.hr/nastava/rp2/pred10/pred10.html

[5] https://en.wikipedia.org/wiki/OSI_model - Layer_3:_Network_Layer

[6] http://titan.fsb.hr/~bosekora/nastava/ims/ims-Vjezba1.pdf

[7] https://www.arduino.cc/

[8] https://www.youtube.com/watch?v=vcOE2XAQHzY

[9] OM_AU_PG__EN.pdf

[10] OM_PT-A_komplett__EN.pdf

[11] http://www.fanucrobotics.com/cmsmedia/datasheets/LR Mate 200iC Series_10.pdf

Prilozi

I. CD-R disc

II. Arduino programski kod

```
#include <SPI.h>
#include <Ethernet.h>
#include <math.h>

byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
IPAddress ip(192, 168, 123, 50);
IPAddress server(192, 168, 123, 25);
EthernetClient client;


String commandString = "";
float razmak;
float brz;
float ubrz;
int sila = 0;
int brojac = 0;
int i, j;
int duljina;
int a = -1;
int a1 = -1;
int b = -1;
int b1 = -1;
int c = -1;
int p = -1;
String brzina1 = "";
String ubrzanje1 = "";
String polozaj1 = "";
String force1 = "";
float hvatS = 0;
int hvatS1;
float polhvat = -2;
int polhvat1;
int incomingByte;
float ax = 0;
int cx = 0;
int px = 0;
int rr = 0;
float bx = 0;

int reset[] = {0x02, 0x05, 0x81, 0x00, 0x86, 0x03 };
int HOME[] = {0x02, 0x05, 0x81, 0x01, 0x87, 0x03 };
int brzina[] = {2, 5, 134, 8, 79, 166, 155, 68, 59, 164, 3 };
int ubrzanje[] = {0x02, 0x05, 0x86, 0x08, 0x50, 0x9e, 0xef, 0x27, 0x3e, 0xd7, 0x03 };
int paramG[] = {0x02, 0x05, 0x86, 0x08, 0x50, 0x9e, 0xef, 0x27, 0x3e, 0xd7, 0x03 };
int pozicija[] = {2, 5, 134, 11, 4, 0, 0, 117, 61, 159, 3 };
int struja[] = {0x02, 0x05, 0x86, 0x0b, 0x08, 0x00, 0x00, 0x19, 0xbf, 0xab, 0x03 };
int struja0[] = {0x02, 0x05, 0x86, 0x0b, 0x08, 0x00, 0x00, 0x00, 0x00, 0x9e, 0x03 };
int getpoz[] = {0x02, 0x05, 0x82, 0x0a, 0x3c, 0xcd, 0x03};
int dohvat[] = {0x00, 0x00, 0x0, 0x00, 0x0, 0x0, 0x00, 0x00, 0x0, 0x0, 0x00};
int getcur[] = {0x02, 0x05, 0x82, 0x0a, 0x4d, 0xde, 0x03};
int dohvatS[] = {0x00, 0x00, 0x0, 0x00, 0x0, 0x0, 0x00, 0x00, 0x0, 0x0, 0x00};
int state[] = {0x02, 0x05, 0x82, 0x0a, 0x27, 0xb8, 0x03};
int brk[] = {0x00, 0x00, 0x0, 0x00, 0x0, 0x0, 0x00, 0x00, 0x0, 0x0, 0x00, 0x00};

void setup() {
  Serial.begin(9600);
  for (i = 0; i < 15; i = i + 1)
    Serial.write(reset[i]);
  for (i = 0; i < 15; i = i + 1)
    Serial.write(HOME[i]);
  Ethernet.begin(mac, ip);
  delay(1000);
  client.connect(server, 5555);
}

void loop() {
  while (client.available()) {
    char newchar = client.read();
    if (newchar == 0x4b) {
      funkcija(commandString);
      if (sila == 0) {
        if (razmak >= 0.49 && razmak < 1.96) {
          pozicija[8] = 58;
          pozicija[7] = ((razmak - 0.49) / 1.47) * 255;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
          while (pozicija[9] > 255)
```

```
            pozicija[9] = pozicija[9] - 255;
          if (pozicija[7] == 2) {
            pozicija[7] = 16;
            pozicija[8] = 130;
            pozicija[9] = 58;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 2 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 3) {
            pozicija[7] = 16;
            pozicija[8] = 131;
            pozicija[9] = 58;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 3 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 16) {
            pozicija[7] = 16;
            pozicija[8] = 144;
            pozicija[9] = 58;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 16 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
        }
        if (razmak >= 1.96 && razmak < 4) {
          pozicija[8] = 59;
          pozicija[7] = ((razmak - 1.96) / 2.04) * 124;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
          while (pozicija[9] > 255)
            pozicija[9] = pozicija[9] - 255;
          if (pozicija[7] == 2) {
            pozicija[7] = 16;
            pozicija[8] = 130;
            pozicija[9] = 59;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 2 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 3) {
            pozicija[7] = 16;
            pozicija[8] = 131;
            pozicija[9] = 59;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 3 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 16) {
            pozicija[7] = 16;
            pozicija[8] = 144;
            pozicija[9] = 59;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 16 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
        }
        if (razmak >= 4 && razmak < 7.84) {
          pozicija[8] = 59;
          pozicija[7] = ((razmak - 4) / 3.84) * 131 + 124;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
          while (pozicija[9] > 255)
```

```
            pozicija[9] = pozicija[9] - 255;
          if (pozicija[7] == 2) {
            pozicija[7] = 16;
            pozicija[8] = 130;
            pozicija[9] = 59;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 2 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 3) {
            pozicija[7] = 16;
            pozicija[8] = 131;
            pozicija[9] = 59;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 3 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 16) {
            pozicija[7] = 16;
            pozicija[8] = 144;
            pozicija[9] = 59;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 16 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
        }
        if (razmak >= 7.84 && razmak < 15.5) {
          pozicija[8] = 60;
          pozicija[7] = ((razmak - 7.84) / 7.66) * 125;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
          while (pozicija[9] > 255)
            pozicija[9] = pozicija[9] - 255;
          if (pozicija[7] == 2) {
            pozicija[7] = 16;
            pozicija[8] = 130;
            pozicija[9] = 60;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 2 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 3) {
            pozicija[7] = 16;
            pozicija[8] = 131;
            pozicija[9] = 60;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 3 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 16) {
            pozicija[7] = 16;
            pozicija[8] = 144;
            pozicija[9] = 60;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 16 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
        }
        if (razmak >= 15.5 && razmak < 31.4) {
          pozicija[8] = 60;
          pozicija[7] = (((razmak - 15.5) / 15.9) * 127) + 125;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
          while (pozicija[9] > 255)
```

```
            pozicija[9] = pozicija[9] - 255;
          if (pozicija[7] == 2) {
            pozicija[7] = 16;
            pozicija[8] = 130;
            pozicija[9] = 60;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 2 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 3) {
            pozicija[7] = 16;
            pozicija[8] = 131;
            pozicija[9] = 60;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 3 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 16) {
            pozicija[7] = 16;
            pozicija[8] = 144;
            pozicija[9] = 60;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 16 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
        }
        if (razmak >= 31.4 && razmak < 50) {
          pozicija[8] = 61;
          pozicija[7] = ((razmak - 31.4) / 18.6) * 76;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
          while (pozicija[9] > 255)
            pozicija[9] = pozicija[9] - 255;
          if (pozicija[7] == 2) {
            pozicija[7] = 16;
            pozicija[8] = 130;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 2 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 3) {
            pozicija[7] = 16;
            pozicija[8] = 131;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 3 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 16) {
            pozicija[7] = 16;
            pozicija[8] = 144;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 16 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
        }
        if (razmak >= 50 && razmak < 60) {
          pozicija[8] = 61;
          pozicija[7] = ((razmak - 50) / 10) * 41 + 76;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
          while (pozicija[9] > 255)
```

```
            pozicija[9] = pozicija[9] - 255;
          if (pozicija[7] == 2) {
            pozicija[7] = 16;
            pozicija[8] = 130;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 2 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 3) {
            pozicija[7] = 16;
            pozicija[8] = 131;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 3 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 16) {
            pozicija[7] = 16;
            pozicija[8] = 144;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 16 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
        }
        if (razmak >= 60 && razmak < 69.4) {
          pozicija[8] = 61;
          pozicija[7] = ((razmak - 60) / 9.4) * 25 + 117;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
          while (pozicija[9] > 255)
            pozicija[9] = pozicija[9] - 255;
          if (pozicija[7] == 2) {
            pozicija[7] = 16;
            pozicija[8] = 130;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 2 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 3) {
            pozicija[7] = 16;
            pozicija[8] = 131;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 3 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
          if (pozicija[7] == 16) {
            pozicija[7] = 16;
            pozicija[8] = 144;
            pozicija[9] = 61;
            pozicija[10] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5]
+ pozicija[6] + 16 + pozicija[8];
            pozicija[11] = 3;
            while (pozicija[10] > 255)
              pozicija[10] = pozicija[10] - 255;
          }
        }
        if (razmak >= 69.4) {
          pozicija[8] = 61;
          pozicija[7] = 142;
          pozicija[9] = pozicija[1] + pozicija[2] + pozicija[3] + pozicija[4] + pozicija[5] +
pozicija[6] + pozicija[7] + pozicija[8];
          pozicija[10] = 3;
        }
```

```
        if (brz >= 80) {
          paramG[4] = 0x4f;
          paramG[7] = 162;
          paramG[8] = 61;
          paramG[10] = 0x3;
          paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
        }
        else
          funkcija9(brz);
        paramG[4] = 0x4f;
        for (i = 0; i < 15; i = i + 1)
          Serial.write(paramG[i]);
        if (ubrz >= 300) {
          paramG[4] = 0x50;
          paramG[7] = 255;
          paramG[8] = 62;
          paramG[10] = 0x3;
          paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
        }
        else
          funkcija9(ubrz);
        paramG[4] = 0x50;
        for (i = 0; i < 15; i = i + 1)
          Serial.write(paramG[i]);
        for (i = 0; i < 12; i = i + 1)
          Serial.write(pozicija[i]);
        Serial.flush();
        while (Serial.available())
          Serial.read();
        for (i = 0; i < 12; i = i + 1)
          brk[i] = 0;
        do {
          for (i = 0; i < 7; i = i + 1)
            Serial.write(state[i]);
          brejk ();
        }
        while (brk[9] != 0x85 && brk[10] != 0x45 && brk[10] != 0x43 && brk[9] != 0x83);
        info ();
      }
      if (sila >= 30 && sila < 72.5) {
        struja[8] = 191;
        struja[7] = ((sila - 30) / 42.5) * 230 + 25;
        struja[9] = struja[1] + struja[2] + struja[3] + struja[4] + struja[5] + struja[6] +
struja[7] + struja[8];
        struja[10] = 3;
        while (struja[9] > 255)
          struja[9] = struja[9] - 255;
        if (struja[7] == 2) {
          struja[7] = 16;
          struja[8] = 130;
          struja[9] = 191;
          struja[10] = struja[1] + struja[2] + struja[3] + struja[4] + struja[5] + struja[6] +
2 + struja[8];
          struja[11] = 3;
          while (struja[10] > 255)
            struja[10] = struja[10] - 255;
        }
        if (struja[7] == 3) {
          struja[7] = 16;
          struja[8] = 131;
          struja[9] = 191;
          struja[10] = struja[1] + struja[2] + struja[3] + struja[4] + struja[5] + struja[6] +
3 + struja[8];
          struja[11] = 3;
          while (struja[10] > 255)
            struja[10] = struja[10] - 255;
        }
        if (struja[7] == 16) {
          struja[7] = 16;
          struja[8] = 144;
          struja[9] = 191;
          struja[10] = struja[1] + struja[2] + struja[3] + struja[4] + struja[5] + struja[6] +
16 + struja[8];
```

```
          struja[11] = 3;
          while (struja[10] > 255)
            struja[10] = struja[10] - 255;
        }
        for (i = 0; i < 15; i = i + 1)
          Serial.write(struja[i]);
        Serial.flush();
        while (Serial.available())
          Serial.read();
        for (i = 0; i < 12; i = i + 1)
          brk[i] = 0;
        do {
          for (i = 0; i < 7; i = i + 1)
            Serial.write(state[i]);
          brejk ();
        }
        while (brk[9] != 0x85 && brk[10] != 0x45 && brk[10] != 0x43 && brk[9] != 0x83);
        info ();
      }
      if (sila >= 72.5 && sila <= 200) {
        struja[8] = 192;
        struja[7] = ((sila - 72.5) / 127.5) * 134;
        struja[9] = struja[1] + struja[2] + struja[3] + struja[4] + struja[5] + struja[6] +
struja[7] + struja[8];
        struja[10] = 3;
        while (struja[9] > 255)
          struja[9] = struja[9] - 255;
        if (struja[7] == 2) {
          struja[7] = 16;
          struja[8] = 130;
          struja[9] = 192;
          struja[10] = struja[1] + struja[2] + struja[3] + struja[4] + struja[5] + struja[6] +
2 + struja[8];
          struja[11] = 3;
          while (struja[10] > 255)
            struja[10] = struja[10] - 255;
        }
        if (struja[7] == 3) {
          struja[7] = 16;
          struja[8] = 131;
          struja[9] = 192;
          struja[10] = struja[1] + struja[2] + struja[3] + struja[4] + struja[5] + struja[6] +
3 + struja[8];
          struja[11] = 3;
          while (struja[10] > 255)
            struja[10] = struja[10] - 255;
        }
        if (struja[7] == 16) {
          struja[7] = 16;
          struja[8] = 144;
          struja[9] = 192;
          struja[10] = struja[1] + struja[2] + struja[3] + struja[4] + struja[5] + struja[6] +
16 + struja[8];
          struja[11] = 3;
          while (struja[10] > 255)
            struja[10] = struja[10] - 255;
        }
        for (i = 0; i < 15; i = i + 1)
          Serial.write(struja[i]);
        Serial.flush();
        while (Serial.available())
          Serial.read();
        for (i = 0; i < 12; i = i + 1)
          brk[i] = 0;
        do {
          for (i = 0; i < 7; i = i + 1)
            Serial.write(state[i]);
          brejk ();
        }
        while (brk[9] != 0x85 && brk[10] != 0x45 && brk[10] != 0x43 && brk[9] != 0x83);
        info ();
      }
      delay (500);
    }
    else
      commandString += newchar;
  }
}
```

```
void funkcija (String naredba) {
  Serial.println (naredba);
  if (naredba.indexOf("reset") > -1 || naredba.indexOf("RESET") > -1 ||
naredba.indexOf("P600@0@0@0") > -1) {

    for (i = 0; i < 15; i = i + 1)
      Serial.write(reset[i]);
    for (i = 0; i < 15; i = i + 1)
      Serial.write(HOME[i]);
    commandString = "";
    brojac = 0;
    razmak = 100;
    sila = 1000;
    return;
  }
  if (naredba.indexOf("P500@0@0@0") > -1 || naredba.indexOf("home") > -1 ||
naredba.indexOf("HOME") > -1) {
    for (i = 0; i < 15; i = i + 1)
      Serial.write(reset[i]);
    for (i = 0; i < 15; i = i + 1)
      Serial.write(HOME[i]);
    Serial.flush();
    while (Serial.available())
      Serial.read();
    for (i = 0; i < 12; i = i + 1)
      brk[i] = 0;
    do {
      for (i = 0; i < 7; i = i + 1)
        Serial.write(state[i]);
      //drugi put funkcija
      brejk ();
    }
    while (brk[9] != 0x85 && brk[10] != 0x45 && brk[10] != 0x43 && brk[9] != 0x83);
    //treci put funkcija
    info ();
    commandString = "";
    brojac = 0;
    razmak = 100;
    sila = 1000;
    return;
  }

  if (naredba.indexOf("P800@0@0@0") > -1 || naredba.indexOf("INFO") > -1 ||
naredba.indexOf("info") > -1 )
  {
    info ();
    commandString = "";
    brojac = 0;
    razmak = 100;
    sila = 1000;
    return;
  }


  if (naredba.indexOf("rucno") > -1 || naredba.indexOf("RUCNO") > -1 ||
naredba.indexOf("P700@0@0@0") > -1 ) {

    for (i = 0; i < 15; i = i + 1)
      Serial.write(struja0[i]);

    commandString = "";
    brojac = 0;
    razmak = 100;
    sila = 1000;
    return;
  }

  duljina = naredba.length();

  for (i = 0; i < duljina; i++) {

    if (naredba.charAt(i) == 'P' && a == -1)
      p = i;
```

```
    if (naredba.charAt(i) == '@' && a == -1)
    { a = i;
      a1 = 2;
    }
    else if (naredba.charAt(i) == '@' && a1 == 2 && b1 == -1)
    { b = i;
      b1 = 2;
    }
    else if (naredba.charAt(i) == '@' && b1 == 2)
    {
      c = i;
    }
  }

  brzina1 = naredba.substring((p + 1), a);
  ubrzanje1 = naredba.substring((a + 1), b);
  polozaj1 = naredba.substring((b + 1), c);
  force1 = naredba.substring(c + 1);

  brz = brzina1.toFloat();
  ubrz = ubrzanje1.toFloat();
  razmak = polozaj1.toFloat();
  sila = force1.toInt();
  Serial.println (commandString);
  Serial.println (brz);
  Serial.println (ubrz);
  Serial.println (razmak);
  Serial.println (sila);



  a = -1;
  a1 = -1;
  b = -1;
  b1 = -1;
  c = -1;

  commandString = "";
}


void funkcija9 (float x) {

  if (x == 0) {
    paramG[8] = 60;
    paramG[7] = 35;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;




  }



  if (x >= 0.49 && x < 1.96) {
    paramG[8] = 58;
    paramG[7] = ((x - 0.49) / 1.47) * 255;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 58;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
```

```
      paramG[8] = 131;
      paramG[9] = 58;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 58;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }



  if (x >= 1.96 && x < 4) {
    paramG[8] = 59;
    paramG[7] = ((x - 1.96) / 2.04) * 124;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 59;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 59;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 59;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }


  if (x >= 4 && x < 7.84) {
    paramG[8] = 59;
    paramG[7] = ((x - 4) / 3.84) * 131 + 124;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 59;
```

```
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 59;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 59;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }


  if (x >= 7.84 && x < 15.5) {
    paramG[8] = 60;
    paramG[7] = ((x - 7.84) / 7.66) * 125;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 60;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 60;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 60;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }



  if (x >= 15.5 && x < 31.4) {
    paramG[8] = 60;
    paramG[7] = (((x - 15.5) / 15.9) * 127) + 125;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
```

```
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 60;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 60;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 60;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }



  if (x >= 31.4 && x < 50) {
    paramG[8] = 61;
    paramG[7] = ((x - 31.4) / 18.6) * 76;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }
```

```
  if (x >= 50 && x < 60) {
    paramG[8] = 61;
    paramG[7] = ((x - 50) / 10) * 41 + 76;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }

  if (x >= 60 && x < 69.4) {
    paramG[8] = 61;
    paramG[7] = ((x - 60) / 9.4) * 25 + 117;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
```

```
        paramG[10] = paramG[10] - 255;
    }
    return;
  }

  if (x >= 69.4 && x < 80) {
    paramG[8] = 61;
    paramG[7] = ((x - 69.4) / 10.6) * 21 + 142;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }

  if (x >= 80 && x < 125) {
    paramG[8] = 61;
    paramG[7] = ((x - 80) / 45) * 92 + 163;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 61;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 61;
```

```
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }

  if (x >= 125 && x <= 300) {
    paramG[8] = 62;
    paramG[7] = ((x - 125) / 175) * 255;
    paramG[9] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] +
paramG[7] + paramG[8];
    paramG[10] = 3;
    while (paramG[9] > 255)
      paramG[9] = paramG[9] - 255;
    if (paramG[7] == 2) {
      paramG[7] = 16;
      paramG[8] = 130;
      paramG[9] = 62;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 2 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 3) {
      paramG[7] = 16;
      paramG[8] = 131;
      paramG[9] = 62;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 3 +
paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    if (paramG[7] == 16) {
      paramG[7] = 16;
      paramG[8] = 144;
      paramG[9] = 62;
      paramG[10] = paramG[1] + paramG[2] + paramG[3] + paramG[4] + paramG[5] + paramG[6] + 16
+ paramG[8];
      paramG[11] = 3;
      while (paramG[10] > 255)
        paramG[10] = paramG[10] - 255;
    }
    return;
  }
  return;
}

void funkcija5 () {
  rr = 0;
  for (i = 0; i < 11; i = i + 1)
    dohvat[i] = 0;
  for (i = 0; i < 7; i = i + 1)
    Serial.write(getpoz[i]);
  Serial.flush();
  if (Serial.available() > 0 && rr == 0 ) {
    if (Serial.read() == 2) {
      for (j = 0; j < 10; j = j + 1) {
        dohvat[j + 1] = Serial.read();
        if (rr == 0) {
          for (i = 0; i < 11; i = i + 1) {
            if (dohvat[i] == 0x03 )
              rr = 1;
          }
        }
        if (dohvat[2] == 0x86) {
          if (dohvat [8] == 58 && dohvat [7] >= 0 && dohvat [7] <= 255)  {
            ax = dohvat[7];
            polhvat = ax / 255 * 1.47 + 0.49;
          }
          if (dohvat [8] == 59 && dohvat [7] >= 0 && dohvat [7] < 124)  {
            ax = dohvat[7];
            polhvat = ax / 124 * 2.04 + 1.96;
          }
```

```
          if (dohvat [8] == 59 && dohvat [7] >= 124 && dohvat [7] <= 255)  {
            ax = dohvat[7];
            polhvat = (ax - 124) / 131 * 3.84 + 4;
          }
          if (dohvat [8] == 60 && dohvat [7] >= 0 && dohvat [7] < 125)  {
            ax = dohvat[7];
            polhvat = ax / 125 * 7.66 + 7.84;
          }
          if (dohvat [8] == 60 && dohvat [7] >= 125 && dohvat [7] <= 255)  {
            ax = dohvat[7];
            polhvat = (ax - 125) / 130 * 15.9 + 15.5;
          }
          if (dohvat [8] == 61 && dohvat [7] >= 0 && dohvat [7] < 76)  {
            ax = dohvat[7];
            polhvat = ax / 76 * 18.6 + 31.4;
          }
          if (dohvat [8] == 61 && dohvat [7] >= 76 && dohvat [7] < 117) {
            ax = dohvat[7];
            polhvat = (ax - 76) / 41 * 10 + 50;
          }
          if (dohvat [8] == 61 && dohvat [7] >= 117 && dohvat [7] <= 142)  {
            ax = dohvat[7];
            polhvat = (ax - 117) / 25 * 9.4 + 60;
          }
        }
      }
    }
  }
}
void funkcija7 () {
  cx = 0;
  for (i = 0; i < 11; i = i + 1)
    dohvatS[i] = 0;
  for (i = 0; i < 7; i = i + 1)
    Serial.write(getcur[i]);
  Serial.flush();
  if (Serial.available() > 0 && cx == 0 ) {
    if (Serial.read() == 2) {
      for (i = 0; i < 10; i = i + 1)
        dohvatS[i + 1] = Serial.read();
      if (cx == 0)
        for (i = 0; i < 11; i = i + 1)
          if (dohvatS[i] == 0x03 )
            cx = 1;
      if (dohvatS[2] == 0x86) {
        if (dohvatS [8] == 191 && dohvatS [7] >= 25 && dohvatS [7] <= 255)  {
          bx = dohvatS[7];
          hvatS = (bx - 25) / 230 * 42.5 + 30;
        }
        if (dohvatS [8] == 192 && dohvatS [7] >= 0 && dohvatS [7] < 134)  {
          bx = dohvatS[7];
          hvatS = bx / 134 * 127.5 + 72.5;
        }
        if (dohvatS [8] == 0 && dohvatS [7] == 0)
          hvatS = 0;
      }
    }
  }
}
void info () {
  while (Serial.available())
    Serial.read();
  polhvat = -2;
  while (polhvat == -2)
    //cetvrti put funkcija
    funkcija5 ();
  polhvat = round(polhvat);
  polhvat1 = (int)polhvat;
  while (Serial.available())
    Serial.read();
  hvatS = -2;
  while (hvatS == -2)
    funkcija7 ();
  hvatS = round(hvatS);
  hvatS1 = (int)hvatS;
  client.print ("P");
  client.print (polhvat1);
  client.print ("@");
```

```
  client.print (hvatS1);
  client.println ("K");
}

void brejk () {
  px = 0;
  Serial.flush();
  for (i = 0; i < 7; i = i + 1)
    Serial.write(state[i]);
  if (Serial.available() > 0 && px == 0 ) {
    if (Serial.read() == 2) {
      for (i = 0; i < 12; i = i + 1) {
        brk[i + 1] = Serial.read();
      }
    }
  }
}
```