# Kinematički kontroler robota RM501

**Golec, Matija**




SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

**Matija Golec**

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

Mentor: Student:

Prof. dr. sc. Mladen Crneković, dipl. ing. Matija Golec

Zagreb, 2019.

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<br>Fakultet strojarstva i brodogradnje | |
|---|---|
| Datum | Prilog |
| Klasa: | |
| Ur.broj: | |

# ZAVRŠNI ZADATAK

Student: **MATIJA GOLEC** Mat. br.: 0035205711

Naslov rada na hrvatskom jeziku: **KINEMATIČKI KONTROLER ROBOTA RM501**

Naslov rada na engleskom jeziku: **KINEMATIC CONTROLLER FOR RM501 ROBOT**

Opis zadatka:

Edukacijski robot Mitsubishi RM501 izrađen je krajem 80-tih godina prošloga stoljeća. Iako su mehanička konstrukcija i pogon robota zastarjeli, u potpunosti su upotrebljivi. Najveće promjene doživio je upravljački sustav robota i njega bi trebalo unaprijediti. Naime, komunikacije ugrađene u robota danas se više ne koriste pa ga više nije moguće spojiti s današnjim računalima. Kako detaljni nacrti elektroničkih sklopova nisu poznati, njihova zamjena nije moguća. Unapređenje rada robota je potrebno izvesti izradom vanjskog uređaja za prilagodbu komunikacije i dodavanjem osnovnog kinematičkog modela robota.
U radu je potrebno:

- projektirati i izraditi sklop za prilagodbu komunikacije (USB i Bluetooth) na paralelnu Centronics komunikaciju,
- ostvariti mogućnost zadavanja vanjskih koordinata robota.

Potrebno je navesti korištenu literaturu i ostale izvore informacija, te eventualno dobivenu pomoć.

Zadatak zadan:
29. studenog 2018.

Rok predaje rada:
**1. rok:** 22. veljače 2019.
**2. rok (izvanredni):** 28. lipnja 2019.
**3. rok:** 20. rujna 2019.

Predviđeni datumi obrane:
**1. rok:** 25.2. - 1.3. 2019.
**2. rok (izvanredni):** 2.7. 2019.
**3. rok:** 23.9. - 27.9. 2019.

Zadatak zadao:

Prof.dr.sc. Mladen Crneković

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

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

Zahvaljujem se svom mentoru prof. dr. sc. Mladenu Crnekoviću za savjete i pomoć tijekom izrade ovog završnog rada.

Zahvaljujem se i svojim roditeljima, obitelji i djevojci za svu potporu, vjeru i strpljenje.

Matija Golec

# SADRŽAJ

# POPIS SLIKA

# POPIS TABLICA

# SAŽETAK

Zadatak ovog završnog rada je projektirati kinematički kontroler pomoću kojeg će se ostvariti komunikacija između Mitsubishijevog robota RM-501 i računala pomoću USB kabla i Bluetootha. Također je potrebno ostvariti mogućnost pomicanja robota upisivanjem njegovih vanjskih koordinata. U uvodnom dijelu ovog rada predstavljen je te je ukratko opisan problem zbog kojeg je bilo potrebno izraditi kinematički kontroler te je dano idejno rješenje. U drugom dijelu rada opisan je robot i pojedine komponente od kojih se sastoji te su navedene njegove karakteristike. Treće poglavlje opisuje projektirani kinematički kontroler, opisuju se komponente od kojih je sastavljen, način njegovog projektiranja i postupak izrade. Četvrto poglavlje opisuje način programiranja, kod korišten u mikrokontroleru kinematičkog kontrolera te način pokretanja i komuniciranja s robotom.

Ključne riječi: industrijski robot, RM501, kinematički kontroler

## SUMMARY

The purpose of this final paper is to design a kinematic controller that will communicate between Mitsubishi's RM-501 robot and a computer using a USB cable and Bluetooth. It is also required to be able to move the robot by typing its external coordinates. The introductory part of this paper presents and briefly describes the problem that necessitated the creation of a kinematic controller and provided a preliminary solution. The second part of the paper describes the robot and its individual components and describes its characteristics. The third chapter describes the designed kinematic controller, describes the components of which it is composed, the method of designing it, and the manufacturing process. The fourth chapter describes the programming method, the code used in the kinematic controller microcontroller, and how to start and communicate with the robot.


Keywords: industrial robot, RM501, kinematic controller

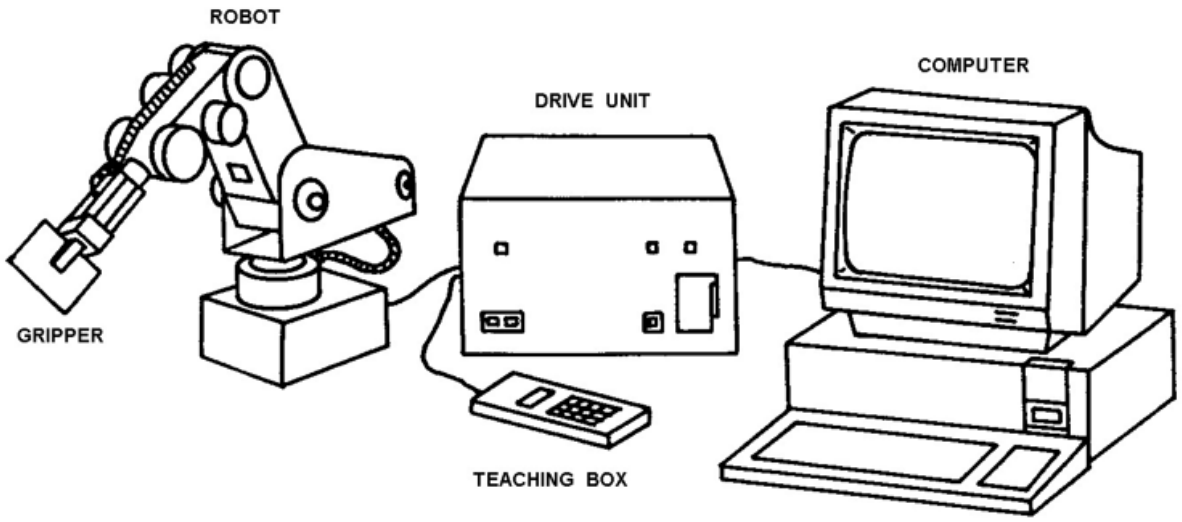
# 1. UVOD

Mitsubisihjev robot RM-501 napravljen je osamdesetih godina prošlog stoljeća te je zamišljen kao malen i lagan robot namijenjen edukaciji za upotrebu pravih odnosno većih industrijskih robota. Isti takav robot nalazi se u jednom od laboratorija našeg fakulteta gdje je bio korišten za edukaciju i upoznavanje studenata s industrijskim robotima. Trenutno se ne koristi jer je zastario te su nabavljeni novi i moderniji roboti, ali pošto je i dalje ispravan može ga se prilagoditi današnjoj tehnologiji te ga tako učiniti korisnim za daljnju edukaciju. Najveći problem je u komunikaciji između računala i samog robota. Robot komunikaciju s računalom ostvaruje preko paralelnog Centronics priključka s 36 pinova. Najveća prednost paralelne komunikacije je brzina prijenosa podataka pa se koristi za komunikaciju s uređajima gdje je potrebna velika brzina prijenosa ili prijenos podataka u realnom vremenu. S vremenom i napretkom tehnologije brzina serijske komunikacije postala je veća te je većina paralelnih priključaka uklonjena iz upotrebe, osobito na prijenosnim računalima te je zamijenjena serijskom komunikacijom. Sam robot na upravljačkoj jedinici ima serijski RS232 priključak, ali iz nekog razloga isti nije ispravan. Iz tog razloga direktno povezivanje računala s robotom više nije moguće. Ovaj problem riješen je od strane profesora Crnekovića i profesora Zorca izradom kinematičkog kontrolera. Napravljen je uređaj koji omogućuje povezivanje paralelne komunikacije na strani robota sa serijskom komunikacijom na računalu. Na jednoj strani kinematičkog kontrolera nalazi se Centronics priključak koji se spaja na robota, a s druge strane se kinematički kontroler spaja na računalo preko RS232 priključka. Kinematički kontroler upravljan je od strane Atmelovog 8-bitnog mikrokontrolera na kojem se nalazi program koji računa kinematički problem robota te povezuje serijsku s paralelnom komunikacijom. Za rad kontrolera potrebno je dovesti 12V vanjskog napajanja pa je s time potrebno izvesti i stabilizator napona na 5V radi mikrokontrolera. Ovaj kinematički kontroler izrađen je prije malo manje od 10 godina te je zbog već spomenutog ekstremno brzog razvoja tehnologije potrebno napraviti novi i suvremeniji kontroler. Najveći problem ponovno se stvara kod povezivanja s računalom, na novim prijenosnim računalima postepeno je nestao prije korišteni RS232 priključak pa je i povezivanje s kontrolerom postalo otežano. Današnja prijenosna računala većinom su opremljena USB priključcima te će se izrada novog kinematičkog kontrolera bazirati upravo na njegovom povezivanju s računalom preko USB-a. Također većina računala opremljena je Bluetooth tehnologijom koja omogućuje bežično povezivanja pa će se omogućiti i bežično povezivanje računala i kinematičkog kontrolera.

# 2. ROBOT

## 2.1. Komponente

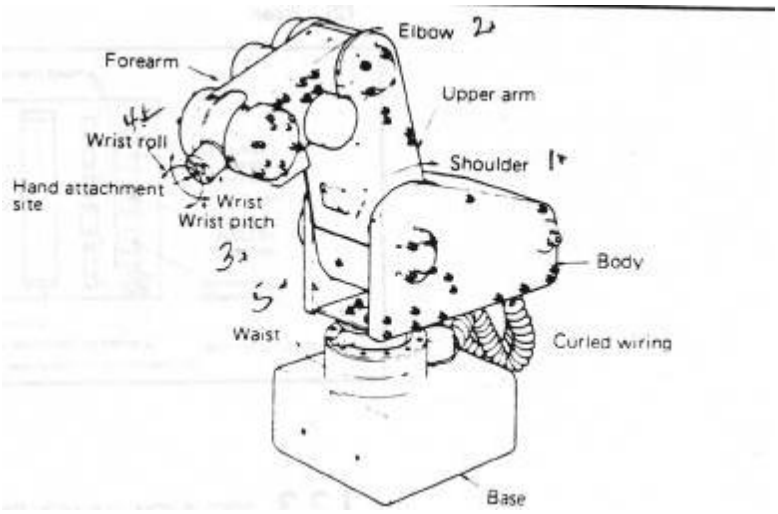
Mitsubishijev robot Movemaster II RM-501 sastoji se od robotske ruke te upravljačke kutije koje su međusobno povezane pomoću dva kabla. Na upravljačku jedinicu povezan je i privjesak za učenje koji se može koristiti za pokretanje i programiranje robota. Također za rukovanje predmetima na kraju robotske ruke montirana je standardna prihvatnica.




**Slika 1. Robotski sustav [1]**

### 2.1.1. Robotska ruka

Sama robotska ruka sastoji se od postolja koje je fiksno i pričvršćeno za stol. Na postolje se nastavlja tijelo robota koje je povezano pomoću rotacijskog zgloba te može rotirati za 300 stupnjeva. Sljedeća komponenta robotske ruke je nadlaktica koja je na tijelo također povezana pomoću rotacijskog zgloba te joj je omogućena rotacija od 130 stupnjeva. Podlaktica robotske ruke ima mogućnost rotacije od 90 stupnjeva te je rotacijskim zglobom povezana sa nadlakticom. Na nadlakticu je rotacijskim zglobom povezana šaka robota. Na šaku robota montira se prihvatnica pomoću koje robot obavlja određeni zadatak. Šaka robota ima omogućenu rotaciju u samome zglobu za ±90 stupnjeva, a također joj je omogućena i rotacija oko vlastite osi za ±180 stupnjeva pomoću koje se postiže željena orijentacija same prihvatnice.



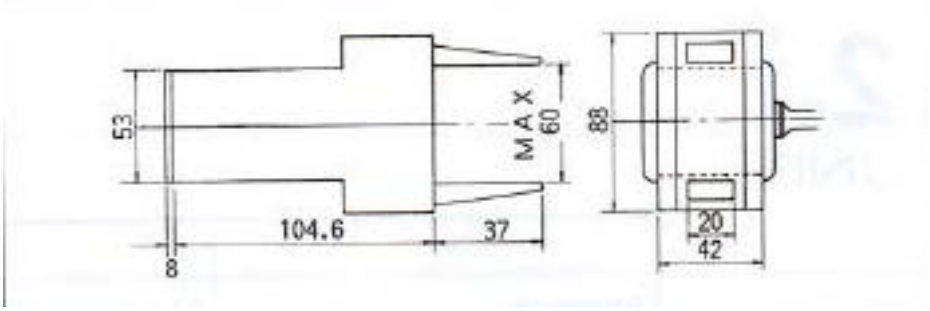

**Slika 2. Robotska ruka [1]**

### 2.1.2. Upravljačka jedinica

Upravljačka jedinica robota s robotom je povezana pomoću pet metara dugačkih kabela te služi za primanje, obradu i izvršavanje naredbi. Na prednjoj strani upravljačke jedinice nalaze se prekidači pomoću kojih uključujemo i isključujemo robotsku ruku, prekidač za testiranje robota i svjetlosni signali za napajanje i grešku. Sa stražnje strane upravljačke jedinice nalaze se konektori koji povezuju upravljačku jedinicu s računalom, privjeskom za učenje, robotskom rukom i izvorom napajanja. Spomenuto je da je robotska ruka povezana pomoću dva kabla, jedan kabel služi za davanje napona motorima, a drugi za prijenos signala k motorima.

### 2.1.3. Prihvatnica

Prihvatnice se montiraju na robotske ruke da bi one mogle izvršavati određene zadatke kao što su prenošenje predmeta pomoću prihvatnica s prstima ili pomoću prihvatnica s vakumskim ili magnetskim hvataljkama. Na našem robotu montirana je standardna prihvatnica s dva prsta maksimalnog zahvata 60 mm i maksimalne sile primanja od 4kg. Sama prihvatnica pogonjena je DC servomotorima i teži oko 700 grama. Dimenzije prihvatnice mogu se vidjeti sa slike 3.



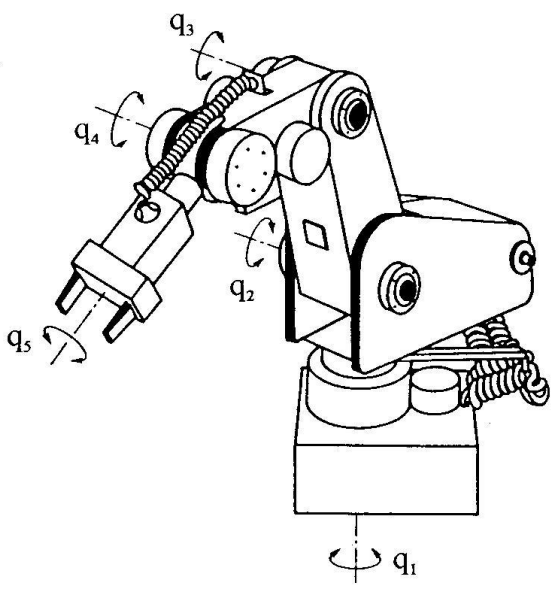

**Slika 3. Prihvatnica robota [1]**

## 2.2. Karakteristike robota

Robotska ruka ima 5 stupnjeva slobode gibanja koji su ostvareni pomoću ranije navedenih rotacijskih zglobova. Na slici 4 su prikazani stupnjevi slobode koji su u tablici 1 povezani sa svojim ograničenjima.

**Tablica 1. Ograničenja stupnjeva slobode**

| Stupanj slobode | Ograničenje |
|---|---|
| q1=struk | 300° |
| q2=rame | 130° |
| q3=lakat | 90° |
| q4=nagib | 90° |
| q5=valjanje | 180° |




**Slika 4. Stupnjevi slobode robotske ruke**

Svi zglobovi u robotu pokretani su pomoću istosmjernih motora koji pogon prenose pomoću zupčanika i remenja te se pritom ostvaruje točnost ponavljanja od ±0,5mm. Pozicija se određuje pomoću enkodera.

Maksimalna deklarirana nosivost robotske ruke je 1,2kg dok je njena približna težina 27kg. Prihvatnica se može kretati maksimalnom brzinom od 400mm/s, a brzina se može postaviti u jednu od 9 ponuđenih opcija.

# 3. KINEMATIČKI KONTROLER

Zbog ranije navedenog problema sa povezivanjem robota i računala potrebno je osmisliti i izraditi kinematički kontroler koji će služiti kao poveznica za komunikaciju između računala i upravljačke jedinice robota. Kinematički kontroler moguće je izraditi kao jednu tiskanu pločicu na kojoj se nalaze sve komponente potrebne za rad ili kao kombinaciju više tiskanih pločica odnosno modula koji se spajaju na tiskanu pločicu sa mikrokontrolerom.

## 3.1. Komponente

Za izradu kinematičkog kontrolera potreban nam je jedan mikrokontroler, FT232RL modul, HC-05 Bluetooth modul te ostale elektroničke komponente kao što su otpornici, kondenzatori, led diode itd.

### 3.1.1. Mikrokontroler

Da bi naš kinematički kontroler uopće mogao funkcionirati potreban mu je mikrokontroler. Mikrokontroler je zapravo maleno računalo koje je namijenjeno za izvršavanje samo jednog programa. Sam mikrokontroler kao i ostala računala sastavljen je od procesora, RAM, ROM i flash memorije te vanjskih uređaja koje kontrolira ili pomoću kojih je kontroliran.

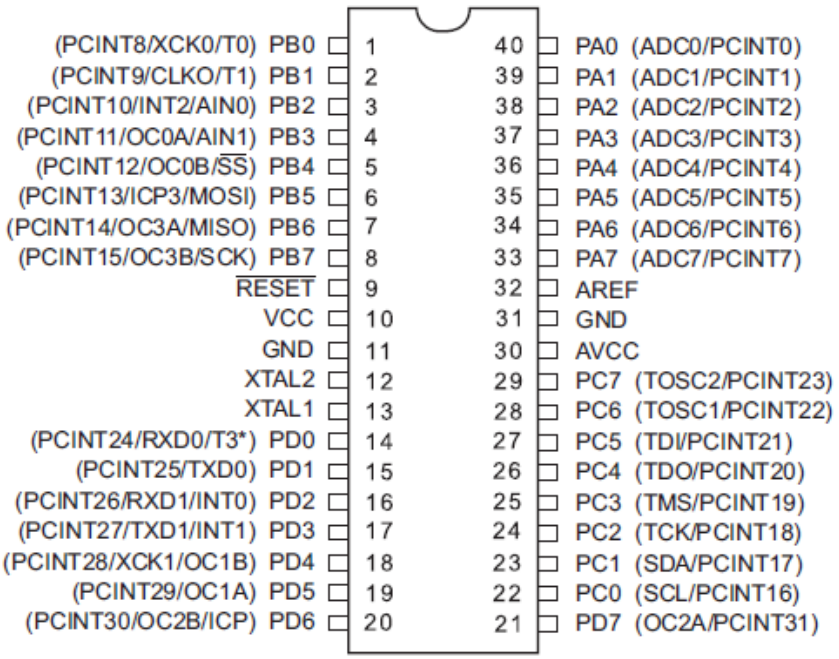

**Slika 5. ATmega1284P mikrokontroler [5]**

Za izradu kinematičkog kontrolera izabran je mikrokontroler ATmega1284P. ATmega1284P je 8 bitni AVR mikrokontroler visokih performansi koji izrađuje američka tvrtka Microchip. Ovaj mikrokontroler sastoji se od 128kB ISP flash memorije, 4kB EEPROM i 16kB SRAM memorije. Iz mikrokontrolera izlazi 40 pinova od kojih se 32 mogu koristit kao ulazni ili izlazni pinovi. Jedna od važnijih stvari koje ovaj mirkokontroler nudi jest mogućnost spajanja dvije UART komunikacije istovremeno. Ova mogućnost će na biti od velike koristi zbog korištenja

USB i Bluetooth komunikacije. Komunikacija je još ostvariva preko tri SPI i jednog $I^2C$ komunikacijskog sučelja.

**Tablica 2. Karakteristike mikrokontrolera**

| Parametri | Vrijednost parametra |
|---|---|
| Tip programske memorije | Flash |
| Količina programske memorije | 128kB |
| Brzina procesora | 20(MIPS/DMIPS) |
| Količina SRAM memorije | 16kB |
| Količina EEPROM memorije | 4kB |
| Digitalna komunikacijska sučelja | 2 x UART<br>3 x SPI<br>1 x $I^2C$ |
| Vremenski brojači | 2 x 8-bitni<br>2 x 16-bitni |
| Broj komparatora | 1 |
| Raspon radne temperature | -40°C – 85°C |
| Raspon radnog napona | 1,8V – 5,5V |
| Broj pinova | 40 |

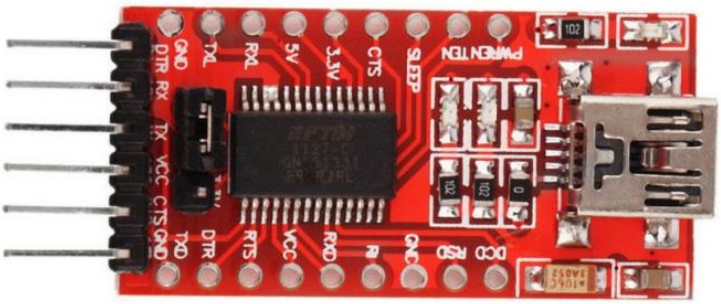



**Slika 6. Raspored pinova mikrokontrolera [5]**

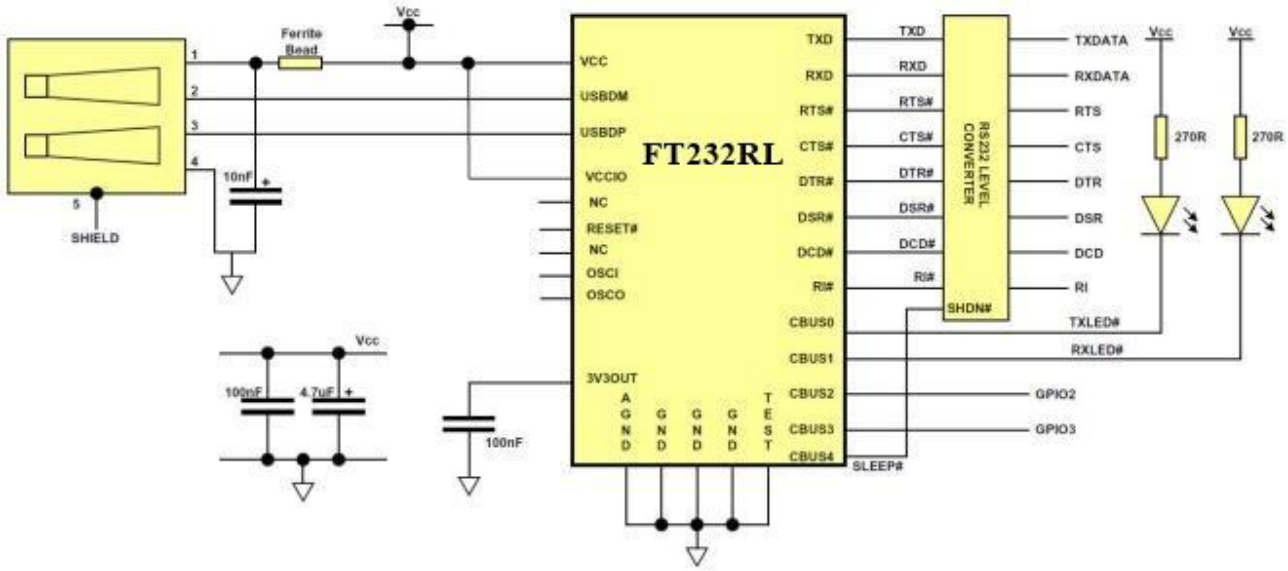
### 3.1.2. FT232RL modul

FT232RL modul potreban nam je za ostvarivanje komunikacije između računala i mikrokontrolera pomoću USB kabla. Sam modul sastoji se od USB mini konektora na koji se

spaja kabel za komunikaciju s računalom, FTDI-evog FT232RL čipa koji služi kao sučelje za komunikaciju između USB-a i UART-a, konektora sa 6 pinova pomoću kojeg se spaja sa mikrokontrolerom te ostalih komponenti kao što su zavojnice, LED diode, otpornici itd.



**Slika 7. FT232RL modul [6]**

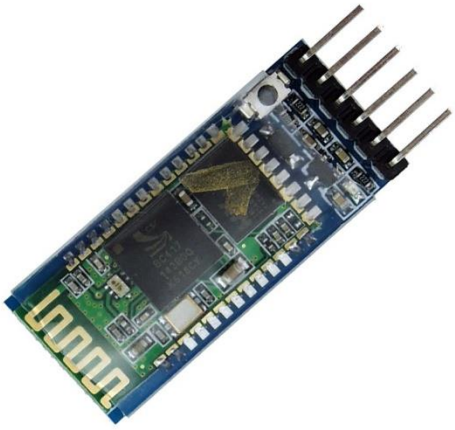



**Slika 8. Shema FT232RL modula [6]**

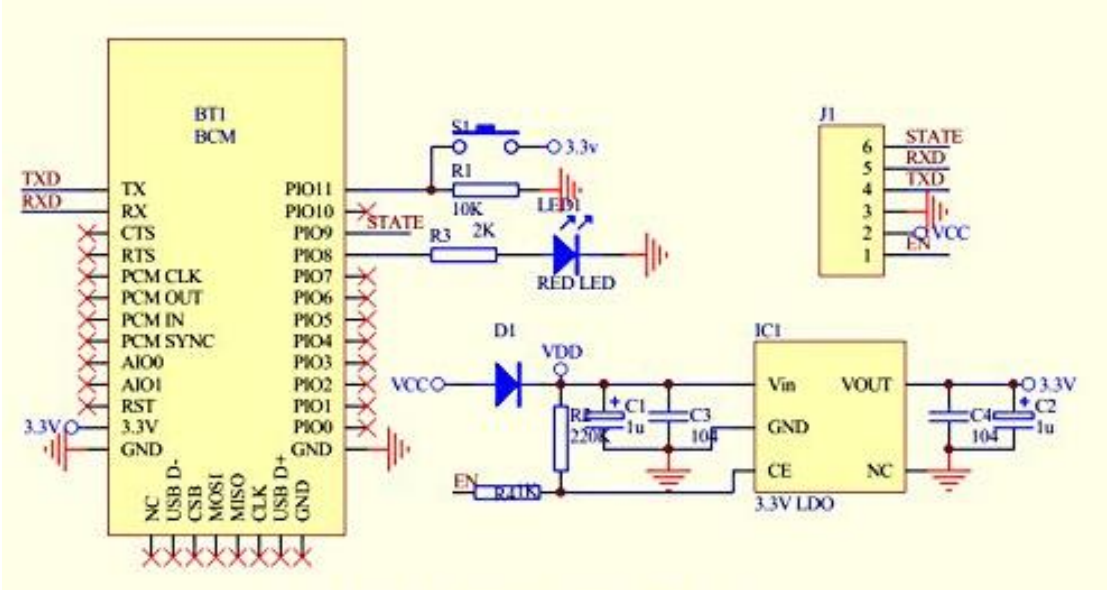
### 3.1.3. HC-05 modul

HC-05 je Bluetooth modul koji se povezuje s mikrokontrolerom te ovisno o načinu spajanja omogućuje bežično programiranje i komunikaciju s mikrokontrolerom ili samo komunikaciju s mikrokontrolerom. Komunikaciju s mikrokontrolerom ostvaruje preko serijskog UART sučelja pa je samo spajanje modula s mikrokontrolerom dosta jednostavno. Ovaj modul može raditi i u master i u slave modu pa se tako može bežično primati i slati podatke između dva modula na dva različita mikrokontrolera ili između mikrokontrolera i pametnog telefona ili

računala. Prilikom kupovine modula može ga se kupiti zasebno ili povezanog sa tiskanom pločicom na kojoj se još nalazi konektor koji omogućuje lakše povezivanje sa mikrokontrolerom, LED dioda koja prikazuje stanje modula, stabilizator napona i dugme pomoću kojeg se mijenjaju načini rada modula. Zadani način rada modula je način za prijenos podataka (eng. „data mode“) u kojem modul šalje i prima podatke od drugih Bluetooth uređaja. Uz pomoć spomenutog dugmeta način rada se mijenja u komandni način rada (eng. „Command mode“) u kojem se mogu mijenjati zadane postavke samog modula.



**Slika 9. Bluetooth modul [6]**



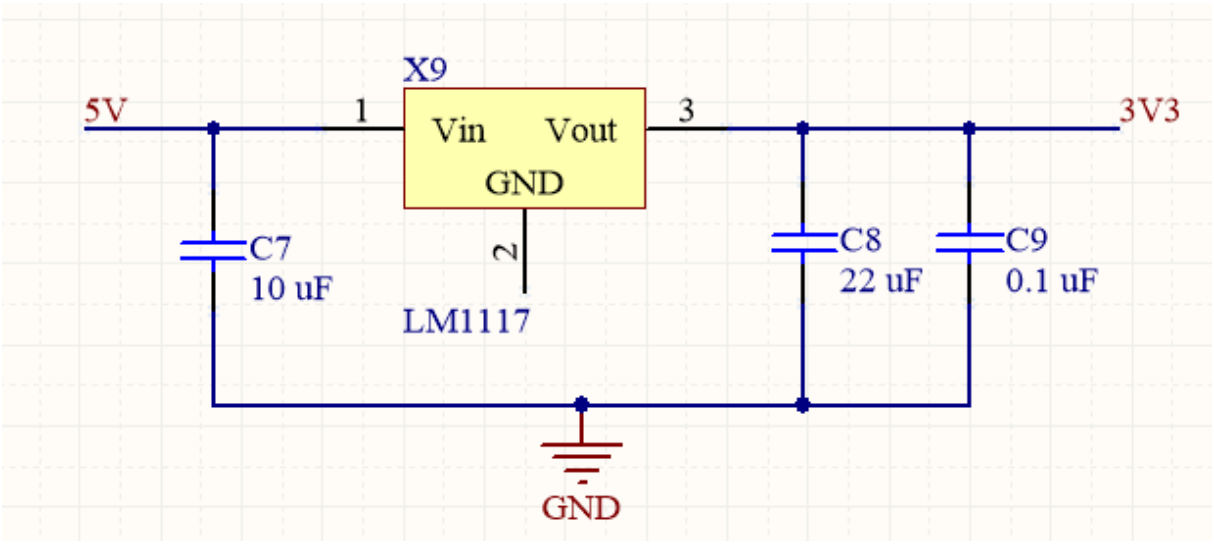

**Slika 10. Shema Bluetooth modula [6]**

### 3.2. Povezivanje komponenata i izrada kinematičkog kontrolera

Ranije je spomenuto da se kinematički kontroler može izraditi na dva načina, koji god da se način izrade koristi spojevi između pinova komponenata ostaju isti. Sam kinematički kontroler nije u stvarnosti izrađen, ali je projektiran te će se u ovom dijelu opisivati način projektiranja i proces koji bi bilo potrebno obaviti prilikom izrade.

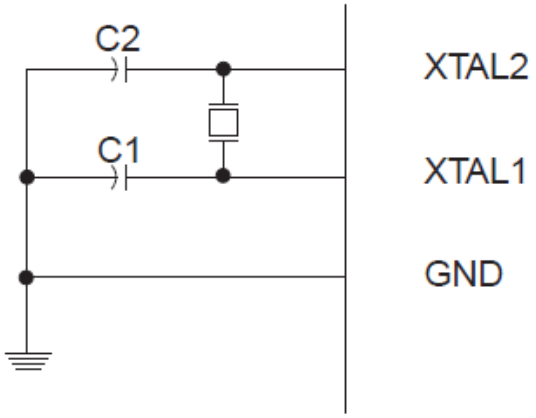
#### 3.2.1. Povezivanje komponenata

Bazni dio kinematičkog kontrolera je mikrokontroler koji se nalazi na glavnoj tiskanoj pločici. Da bi mikrokontroler radio potreban mu je izvor napajanja. Izvor napajanja za mikrokontroler, a samim tim i za cijeli kinematički kontroler dobiva se od računala, prijenosne baterije ili direktno iz utičnice, pomoću punjača za mobitel, preko USB priključka koji se nalazi na FT232RL modulu. Napon na USB priključcima je 5V, a taj napon odgovara svim komponentama osim Bluetooth modula koji koristi napon od 3.3V. Ako se kinematički kontroler izrađuje sa svim komponentama na jednoj pločici potrebno je izvesti stabilizator napona na 3.3V (Slika 11), a ako kupujemo Bluetooth modul na posebnoj tiskanoj pločici za spajanje on dolazi sa ugrađenim stabilizatorom napona pa ga spajamo na 5V od mikrokontrolera.



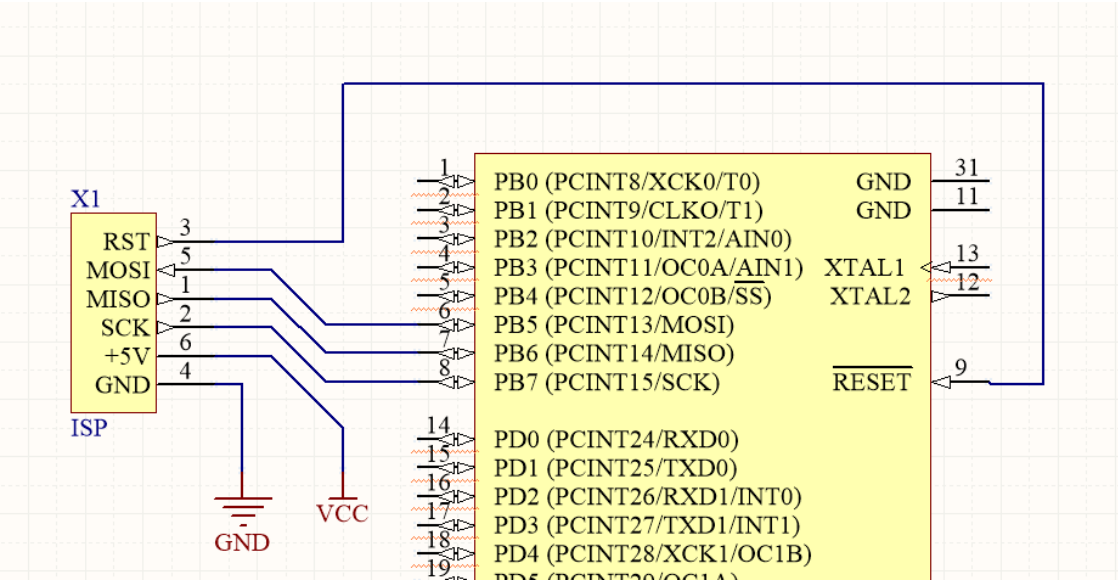

**Slika 11. Stabilizator napona**

Na mikrokontroler je također potrebno povezati kvarcni kristal koji će služiti kao oscilator odnosno davati takozvani clock signal koji usklađuje sve unutarnje operacije mikrokontrolera. Kristal se povezuje na XTAL1 i XTAL2 pinove mikrokontrolera prema shemi na slici, a vrijednosti kondenzatora izabrane su prema preporukama proizvođača mikrokontrolera.




**Slika 12. Način povezivanja kvarcnog kristala [5]**

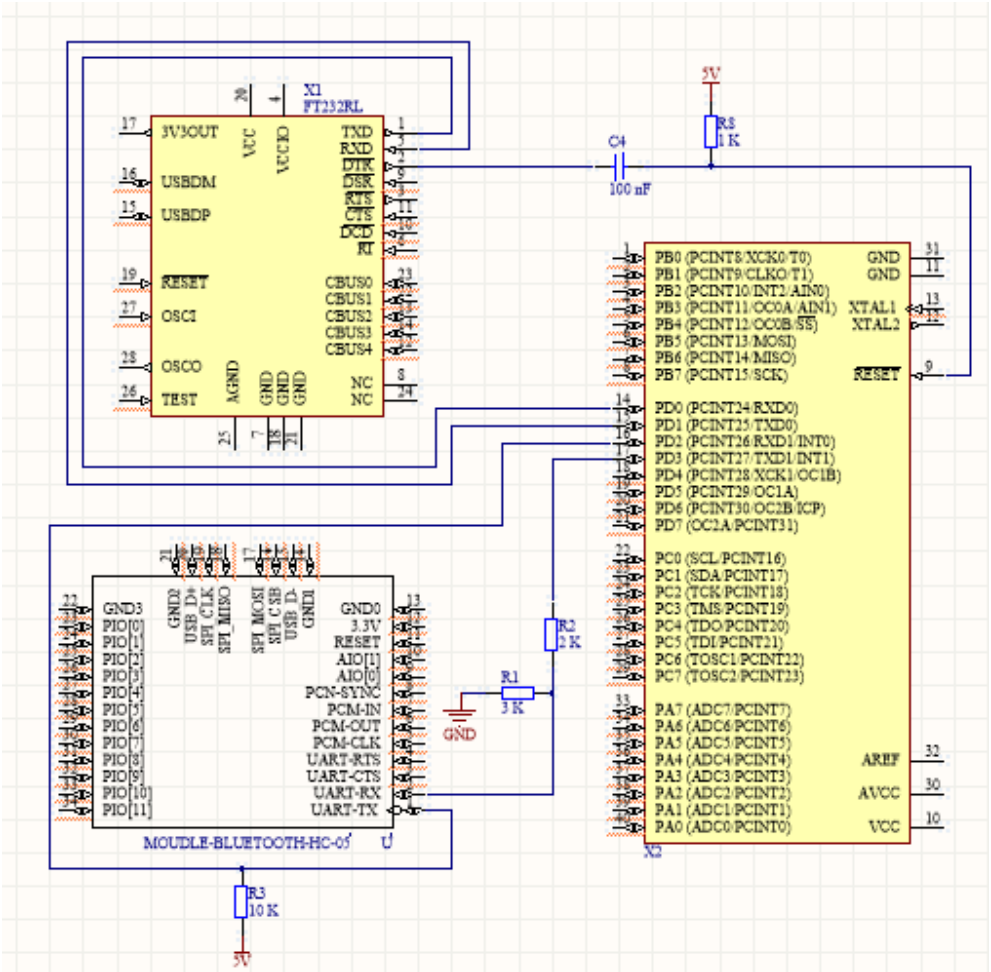
Na tiskanoj pločici sa mikrokontrolerom također se nalazi i ISP konektor. ISP (eng. In system programing) konektor služi za programiranje mikrokontrolera, a spaja se na mikrokontroler preko SPI komunikacijskog sučelja prema slijedećoj shemi prikazanoj na slici 12. SPI komunikacija funkcionira po master slave principu.




**Slika 13. Shema spajanja ISP konektora**

U slučaju izrade kinematičkog kontrolera na jednoj pločici Bluetooth modul i FT232RL modul se u komponentama ugrađuju na pločicu, a u slučaju da se kupuju gotovi moduli potrebno je na tiskanoj pločici s mikrokontrolerom postaviti konektore na koje će se moduli spajati. I Bluetooth i FT232RL modul se na mikrokontroler spajaju pomoću konektora sa 6 pinova te za

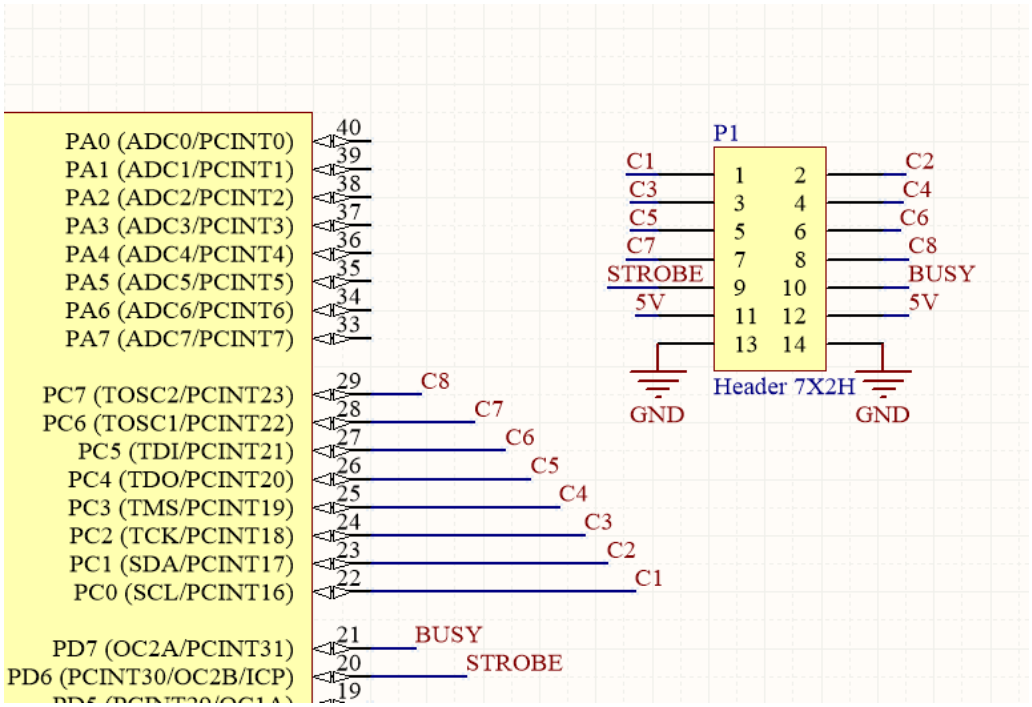
komunikaciju koriste serijsko UART sučelje. Korištenja UART sučelja podrazumijeva komunikaciju preko dvije linije od koje se jedna linija koristi za primanje podataka, a druga za slanje podataka. Pinovi preko kojih se uspostavljaju ove komunikacijske linije nazivaju se RXD (eng. recive data) i TXD (eng. transmit data). Ranije je spomenuto kako ATmega1284P mikrokontroler raspolaže sa dva UART sučelja što znači da ima dva para RXD i TXD pinova koji su nazvani RXD0, TXD0, RXD1 i TXD1. Na TXD0 pin mikrokontrolera spaja se RXD pin sa FT232RL modula, a na pin RXD0 se spaja pin TXD sa FT232RL modula. Ista situacija je i sa spajanjem Bluetooth modula samo što se njegov RXD pin spaja na TXD1 pin, a njegov TXD pin na RXD1 pin od mikrokontrolera. Problem kod spajanja komunikacije između Bluetooth modula i mikrokontrolera je razlika u naponima signala. Visoki napon signala na mikrokontroleru je 5V, a na Bluetooth modulu je 3.3V. Iz tog razloga se linija koja povezuje RXD1 pin mikrokontrolera s TXD pinom Bluetooth modula povezuje preko djelitelja napona, a linija koja povezuje pin TXD0 sa pinom RXD na Bluetooth modulu se povezuje preko otpornika od 10kΩ na napajanje od 5V.




**Slika 14. Povezivanje Bluetooth i FT232RL modula preko UART sučelja**

Da bi se mikrokontroler mogao ponovno pokrenut pin RESET koji se nalazi na mikrokontroleru se povezuje na napajanje i DTR pin od FT232RL modula. Kada signal dođe s modula na RESET pinu mikrokontrolera se pojavi nisko stanje i ako ono potraje dulje od dvije mikro sekunde mikrokontroler se ponovno pokreće.

Povezivanje s robotom ostvaruje se preko 14 pinskog konektora smještenog na tiskanoj pločici mikrokontrolera. Na konekcijskom kablu robota se trenutno nalazi paralelni Centronics konektor sa 36 pinova, ali od tih 36 za prijenos podataka koristi se njih 10, 2 pina su za napajanje, a 2 su za uzemljenje.

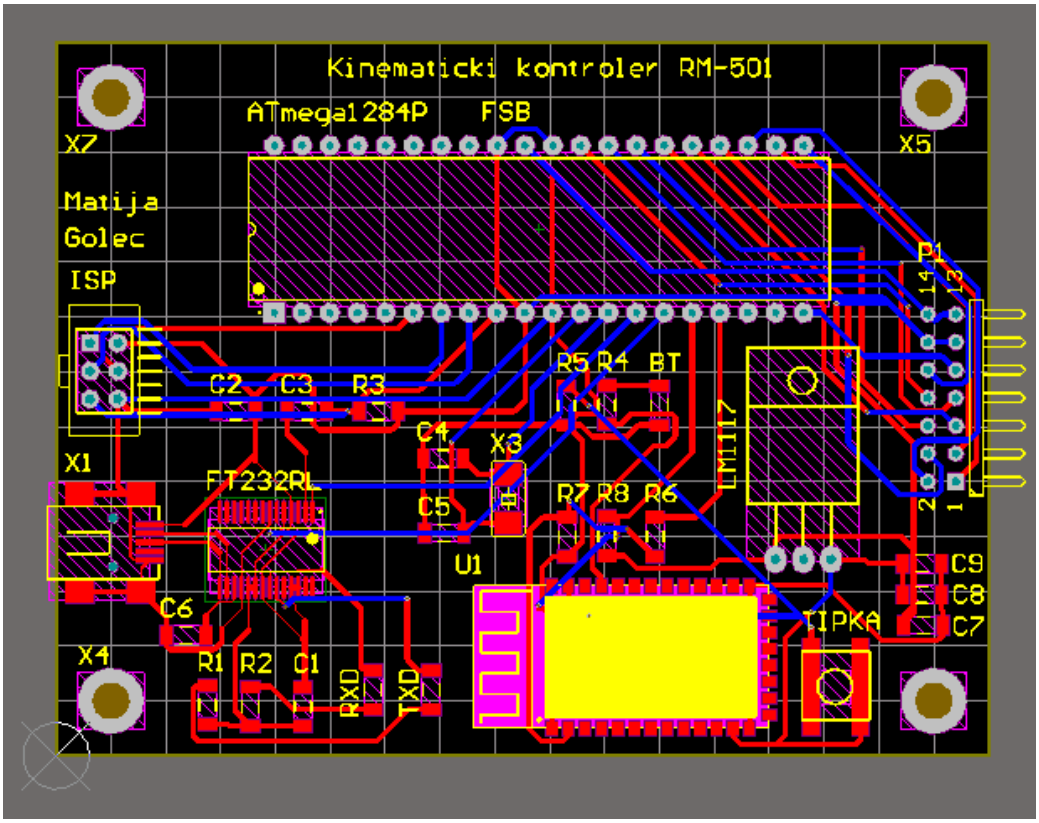



**Slika 15. Shema povezivanja konektora za spajanje s robotom s mikrokontrolerom**

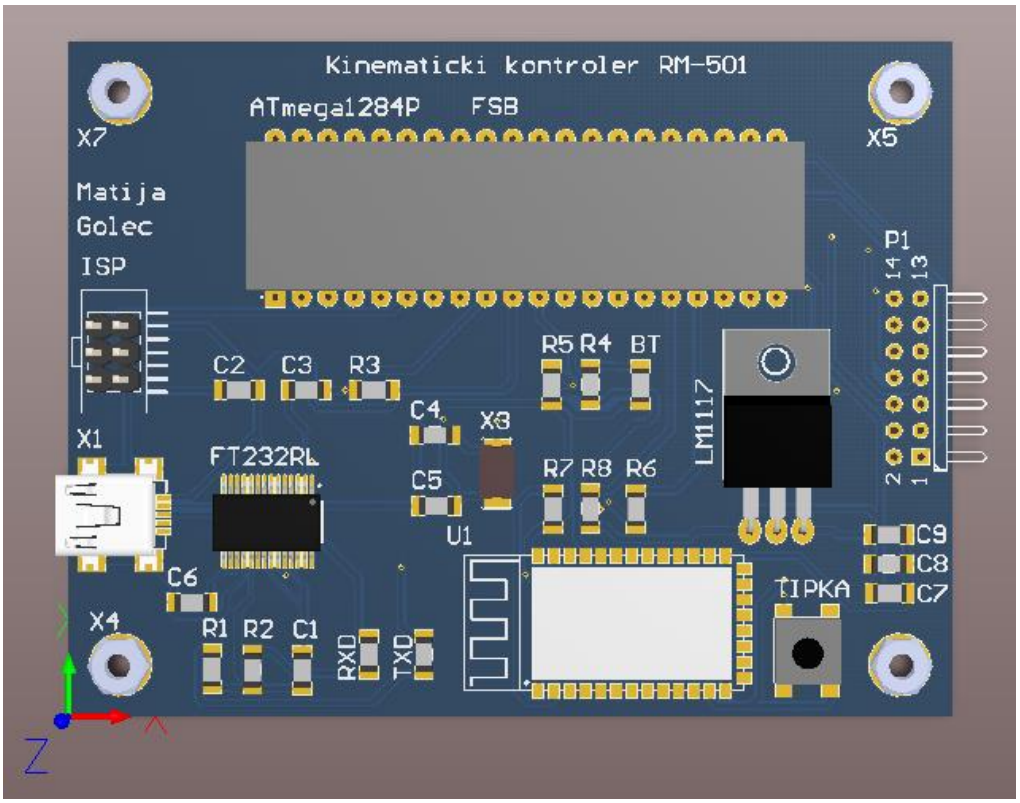
### 3.2.2. Izrada kinematičkog kontrolera

Prvi dio izrade bilo koje tiskane pločice je crtanje njezine sheme i izrada same tiskane pločice u računalnom programu. Program korišten za izradu ove pločice je Altium Designer. U programu se mogu izrađivati ili uvesti već izrađeni modeli stvarnih komponenata te ih se jednostavno postavlja i povezuje. Nakon izrade sheme tiskane pločice prelazi se na izradu modela pločice. Iz datoteke u kojoj je shema možemo povući sve komponente i njihove veze u PCB sučelje za modeliranje pločice. Komponente onda razmještamo u željeni raspored te određujemo konačnu veličinu i oblik tiskane pločice. Na rubove pločice postavljene je linija

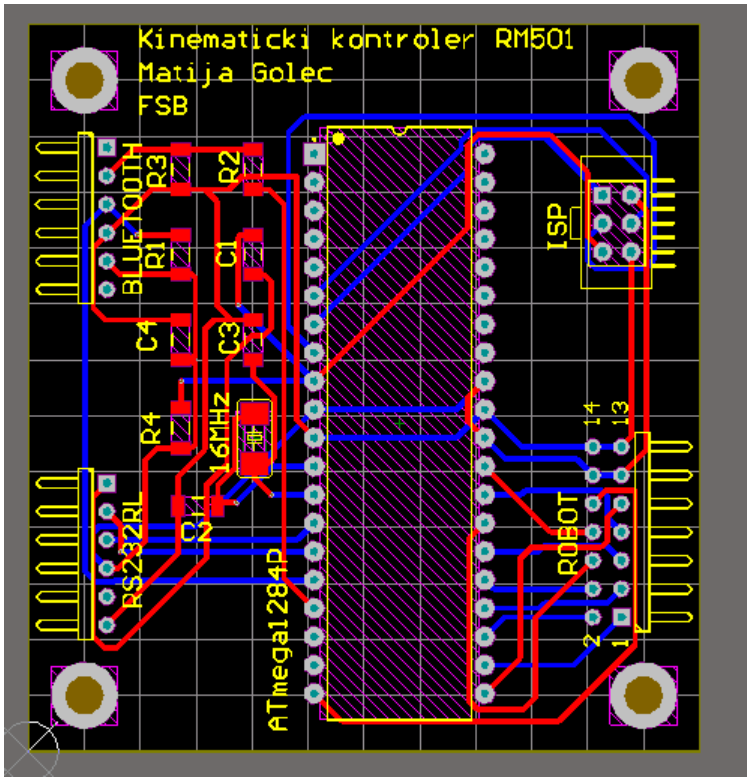
koja će kasnije biti korištena prilikom izrade tiskane pločice. Na gornji i donji sloj tiskane pločice stavljeni su slojevi bakra koji su povezani na uzemljenje te su pomoću opcije za automatsko povezivanje komponenata postavljene ostali vodovi. Prije nego se vodovi postave potrebno je odrediti pravila kao što su minimalna i maksimalna širina vodova, minimalna udaljenost između dva različit voda i veličine provrta. Iz usporedbe ograničenja nekoliko firmi za izradu tiskanih pločica uzeta su ograničenja da vodovi ne smiju biti tanji od 0.2mm, također razmak između različitih vodova ne smije biti manji od 0.2mm, a minimalni provrt je 0.3mm. Kada smo gotovi sa izradom pločice na prozirnu foliju ispisujemo pozitiv gornjeg i donjeg sloja pločice. Kao što je već spomenuto tiskana pločica se može izraditi na dva načina pa su dane slike i sheme za oba načina izrade.



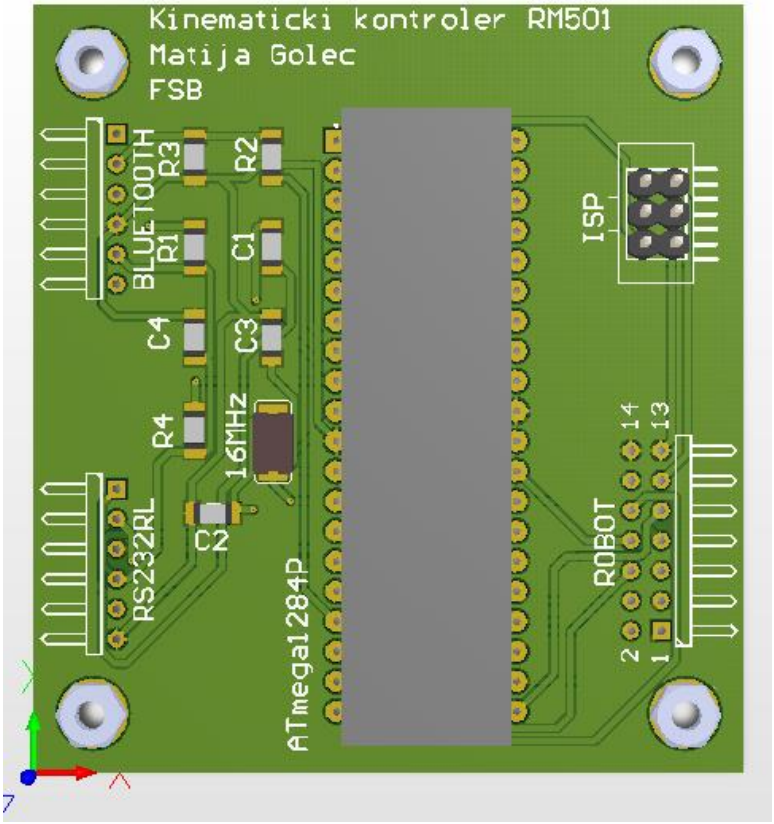

**Slika 16. PCB sa svim komponentama sa povezanim vodovima**




**Slika 17. 3D prikaz PCB-a sa svim komponentama**




**Slika 18. PCB s konektorima za module sa povezanim vodovima**




**Slika 19. 3D prikaz PCB-a s konekotrima za module**

Izrada same tiskane pločice može se obaviti slanjem projektirane pločice u neku specijaliziranu tvrtku za izradu tiskanih pločica, a uz malo truda i znanja tiskana pločica može se napraviti i u kućnim uvjetima. Postupak ovdje opisan prikazan je na jednom od kolegija prilikom kojeg se izrađivala tiskana pločica te se tim istim postupkom može izraditi i ovdje projektirana tiskana pločica.

Kupljene tiskane pločice uglavnom su napravljene od vitroplasta na kojem se s jedne ili obje strane, ovisno trebamo li dvostranu ili jednostranu pločicu, nalazi bakrena folija. Prvo je potrebno kupljenu pločicu prilagoditi veličini projektirane pločice, a zatim očistiti površine bakrene folije. Čišćenje je potrebno zato što bakar u dodiru sa zrakom korodira te stvara oksidacijski sloj koji je potrebno ukloniti, ali i zbog uklanjanja masnoće nastale zbog dodirivanja same pločice. Postupak čišćenja može se provesti na razne načine kao što su: brušenje finim brusnim papirom, pranjem pomoću nitro razrjeđivača, vibracijskom bušilicom, čeličnom vunom ili već nekim drugim postupkom. Nakon čišćenja se bakrena površina pločice više ne smije dodirivati prstima.

Sljedeći postupak je nanošenje zaštitnog sloja odnosno maske na dijelove bakrene folije na kojima želimo ostaviti foliju koja će služiti kao vodovi na našoj tiskanoj pločici. Nanošenje

maske može se izvesti na razne načine kao što su: vodootporni flomaster, ljepljiva traka, sitotisak, fotopostupak itd. Za nanošenje maske izabran je fotopostupak.

Fotopostupak nanošenja maske započinje nanošenjem fotoosjetljivog filma na čistu bakrenu foliju pločice uz prigušeno svijetlo. Fotoosjetljivi film može biti u obliku folije ili u spreju. Nakon nanošenja fotoosjetljivog filma potrebno ga je osušiti, sušenje na sobnoj temperaturi je dosta sporo pa ga je moguće ubrzati sušenjem u pećnici. Nakon sušenja na tiskanu pločicu pozicionira se pozitiv predložak projektirane tiskane pločice te se vrši osvjetljavanje UV svijetlom oko dvije do tri minute. Na ovaj način se naš pozitiv predložak preslikava u negativ predložak na tiskanoj pločici. Postupak treba provesti na obje strane tiskane pločice. Nakon osvjetljavanja tiskana pločica stavlja se u razvijač. Kao razvijač se najčešće koristi otopina natrijevog karbonata ili otopina natrijeve lužine. Postupak razvijanja prilikom upotrebe svježeg razvijača traje oko jedne minute. Nakon razvijanja pločica se ispere vodom te su sa pločice uklonjeni svi dijelovi zaštitne maske osim onih koji će kasnije biti vodovi na našoj pločici.

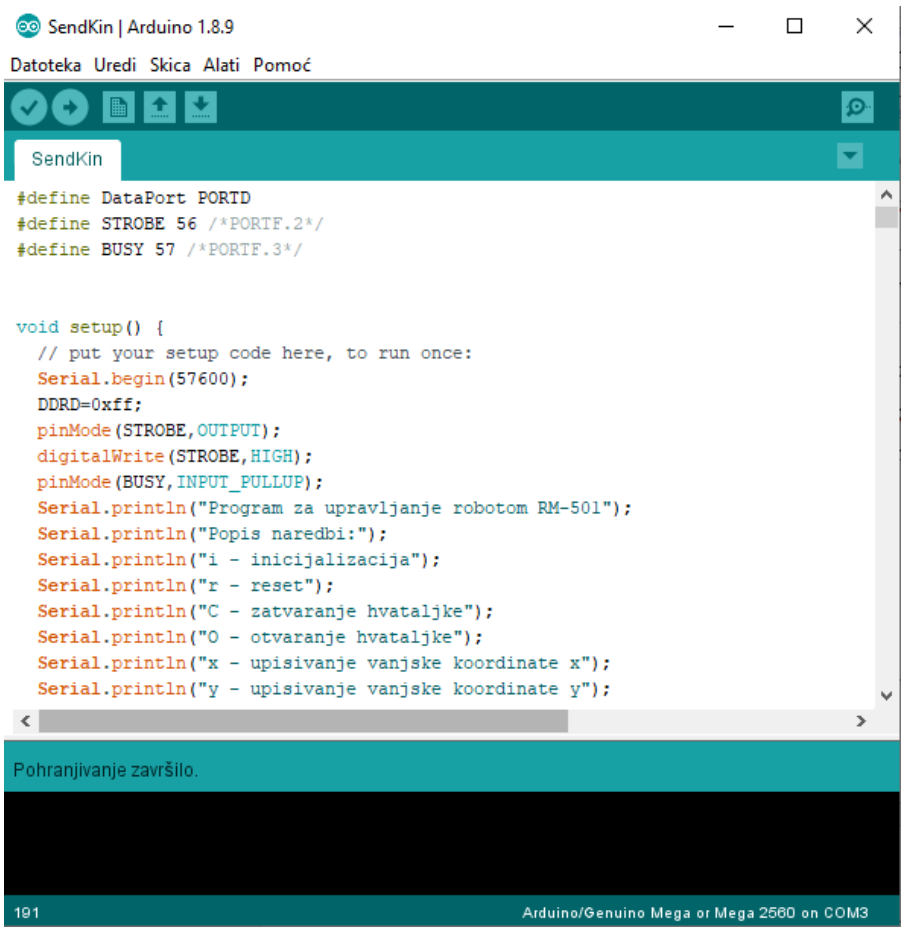
Sljedeći postupak je jetkanje. Jetkanjem se uklanja nepotrebni dio bakrene folije, a provodi se upotrebom kemijskih sredstava od kojih je najbolje upotrijebiti otopinu vode, solne kiseline i vodikovog superoksida. Ova otopina je opasna za korištenje pa ju je potrebno koristiti uz upotrebu zaštitne opreme (rukavice, maska, pregača, zaštitne naočale). Jetkanje se provodi uranjanjem tiskane pločice u otopinu koju smo ulili u plastičnu posudu. Da bi ubrzali postupak lagano ljuljamo posudu. Kada se uklone nepotrebni dijelovi folije bakra jetkanje je završeno, a pločica se ispere vodom te se osuši.

Nakon jetkanja potrebno je skinuti zaštitnu masku s ostatka bakrene folije odnosno s vodova pločice. Skidanje maske provodi se otapalom kao što je aceton ili nitro razrjeđivač. Nakon skidanja maske imamo čiste bakrene vodove koje od korodiranja možemo zaštititi plastic lakom ili ih se može pokositriti pomoću tankog sloja lemne žice.

Pločica je sad spremna za bušenje provrta koje se mora provoditi pažljivo na stolnoj bušilici pri čim većim brzinama svrdla i uz upotrebu posebnih svrdla za bušenje vitroplast pločica. Upotrebom običnih svrdla se ona brzo zatupe pa se ne dobiju jednako kvalitetni provrti kao kod posebnih svrdla. Promjer provrta ovisi o komponenti koja se montira, a kreće se od 0,8 mm do 1,2 mm. Nakon bušenja provrta sve je spremno za postavljanje i lemljenje komponenata na pločicu.

# 4. PROGRAMIRANJE MIKROKONTROLERA

Jedan od razloga zašto je izabran ATmega1284p mikrokontroler je i taj što mu je programski kod moguće pisati i učitavati pomoću Arduino programa, što znači da se mikrokontroler može relativno jednostavno programirati, jer je Arduino porgram te donekle modificirani programski jezik C++ koji program koristi poprilično jednostavan za koristiti. Kod modificiranog C++ jezika misli se da program koristi većinu sintakse C++ programskog jezika uz neke izmjene koje su napravljene radi specijalizirane primjene Arduino programa za programiranje mikrokontrolera. Upotrebu i programiranje u Arduino programu najviše pojednostavljuje njegova popularnost pa samim time i jako velika internetska pokrivenost primjerima koda i objašnjenjima naredbi. Arduino program napravljen je za programiranje Arduino pločica sa različitim AVR mikrokontrolerima. ATmega1284P trenutno se ne koristi ni na jednoj arduino pločici pa ga nije moguće samo izabrati iz ponuđenih pločica u programu nego je potrebno instalirati jezgru posebno za naš mikrokontroler. Arduino jezgra za ATmega1284P dostupna je na internetu te nakon što se instalira u Arduino program on uz svoje standardne pločice nudi i pločicu sa ATmega1284P mikrokontrolerom.




**Slika 20. Arduino programsko sučelje**

### 4.1. Bootloader

Da bi se naš mikrokontroler mogao programirati preko serijskog UART sučelja te da bi se s njime moglo komunicirati preko sučelja za komunikaciju koje je u Arduino programu nazvano serial monitor potrebno mu je instalirati bootloader. Bootloader je kod koji je pohranjen u posebnoj memoriji mikrokontrolera, odvojenoj od memorije u koju se pohranjuje glavni kod programa te se pokreće prije glavnog koda i može primati promjene u kodu preko serijskog UART komunikacijskog sučelja. Kad ne bi instalirali bootloader sve promjene koje bi željeli napraviti u našem programu trebali bi mikrokontroleru slati preko ISP konektora, koji koristi STL komunikacijsko sučelje, uz pomoć ISP programera. Zbog toga je i bootloader potrebno instalirati na taj način, a kao ISP programer može se koristiti Arduino UNO pločica. Nakon instaliranja bootloadera našu tiskanu pločicu spajamo s računalom pomoću USB kabla te svaku promjenu u kodu ubacujemo direktno iz Arduino programa preko USB kabla u naš mikrokontroler.

### 4.2. Kinematički problem

Da bi se naš kontroler uopće mogao nazivati kinematičkim kontrolerom on mora moći riješiti kinematički problem robota. Rješavanje kinematičkog problema omogućuje robotu da u svakom trenutku zna svoj položaj i orijentaciju, a samim time se omogućuje i točno obavljanje zadatka robota.

Za rješavanje kinematičkog problema potrebno je definirati dva vektora koordinata, vektor vanjskih koordinata $\boldsymbol{r} = [x\ y\ z\ \varphi\ \psi]^T$ i vektor unutarnjih koordinata $\boldsymbol{q} = [q_1\ q_2\ q_3\ q_4\ q_5]^T$. Zbog razlučivosti koordinata $q_1$, $q_2$ i $q_3$ od 0,025° te koordinata $q_4$ i $q_5$ od 0,075°za pomak svakog stupnja slobode zadaje se višekratnik razlučivosti koordinate odnosno parametar $a = [a_1\ a_2\ a_3\ a_4\ a_5]^T$. Također zbog nedovoljne preciznosti mikroprekidača na robotu parametar $a$ potrebno je ispraviti za izmjerenu konstantu $c = [0\ 163\ 8\ 925\ 925]^T$. Iz toga se za svaku unutarnju koordinatu uspostavlja sljedeća veza:

$$q_1 = -0.025(a_1 + c_1) \tag{4.2.1}$$

$$q_2 = 55 - 0.025(a_2 + c_2) \tag{4.2.2}$$

$$q_3 = 55 - 0.025(a_3 + c_3) \tag{4.2.3}$$

$$q_4 = \frac{-0.075}{2}(a_4 + c_4 - a_5 - c_5) \tag{4.2.4}$$

$$q_5 = \frac{-0.075}{2}(a_4 + c_4 + a_5 + c_5) \tag{4.2.5}$$

### 4.2.1. Direktni kinematički problem

Rješavanjem direktnog kinematičkog problema iz vektora unutarnjih koordinata $\boldsymbol{q}$ dobivamo vektor vanjskih koordinata $\boldsymbol{r}$**.** Za rješavanje direktnog kinematičkog problema korištene su sljedeće jednadžbe:

$$x_3 = \cos(q_1)\,[L_2 \sin(q_2) + L_3 \sin(q_2 + q_3)] \tag{4.2.6}$$

$$y_3 = sin(q_1)\,[L_2\, sin(q_2) + L_3 sin(q_2 + q_3)] \tag{4.2.7}$$

$$z_3 = L_1 + L_2 \cos(q_2) + L_3 \cos(q_2 + q_3) \tag{4.2.8}$$

$$\varphi = 90° - (q_2 + q_3 + q_4) \tag{4.2.8}$$

$$\psi = q_5 \tag{4.2.9}$$

$$x = x_3 + (L_4 + L_5) cos\varphi \cos(q_1) \tag{4.2.10}$$

$$y = y_3 + (L_4 + L_5) cos\varphi cos(q_1) \tag{4.2.11}$$

$$z = z_3 + (L_4 + L_5) sin\varphi \tag{4.2.12}$$

### 4.2.2. Inverzni kinematički problem

Rješavanjem inverznog kinematičkog problema iz poznatih vanjskih koordinata $\boldsymbol{r}$ izračunavamo unutarnje koordinate $\boldsymbol{q}$**.** Za rješavanje inverznog kinematičkog problema korištene su sljedeće jednadžbe:

$$q_1 = \arctan(\tfrac{y}{x}) \tag{4.2.13}$$

$$x_3 = x - (L_4 + L_5) cos\varphi cos(q_1) \tag{4.2.14}$$

$$y_3 = y - (L_4 + L_5) cos\varphi cos(q_1) \tag{4.2.15}$$

$$z_3 = z - (L_4 + L_5) sin\varphi \tag{4.2.16}$$

$$d = \sqrt{x_3^2 + y_3^2 + (z_3 - L_1)^2} \tag{4.2.17}$$

$$cos\beta = \tfrac{L_2^2 + d^2 - L_3^2}{2 L_2 d} \tag{4.2.18}$$

$$tan\gamma = \frac{z_3 - L_1}{\sqrt{x_3^2 + y_3^2}} \tag{4.2.19}$$

$$q_2 = 90° - (q_2 + q_3 + \varphi) \tag{4.2.20}$$

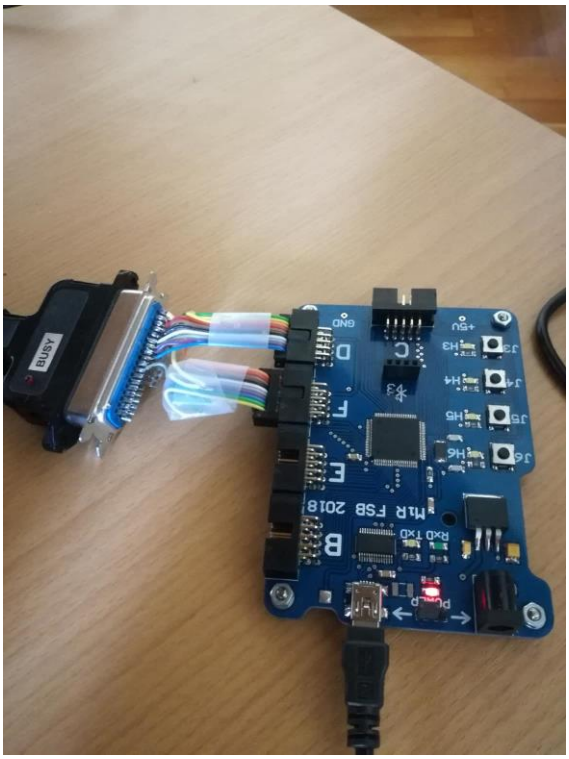
$$q_3 = \arccos\left(\frac{d^2 - L_2^2 - L_3^2}{2L_2L_3}\right) \tag{4.2.21}$$

$$q_4 = 90° - (q_2 + q_3 + \varphi) \tag{4.2.22}$$

$$q_5 = \psi \tag{4.2.23}$$

### 4.3. Program

Svaki program napisan u Arduino programu počinje sa dvije funkcije od kojih se prva izvršava samo jednom prilikom pokretanja programa i naziva se setup(), a druga se izvršava uzastopno i naziva se loop(). Pošto se projektirana tiskana pločica za kinematički kontroler nije izrađivala kod je napisan i isproban na tiskanoj pločici profesora Crnekovića koja je izrađena da funkcionira kao Arduino Mega koji koristi ATmega2560 mikrokontroler (Slika 21.). Jedina razlika koja bi se trebala prepraviti da je kod pisan za ovdje projektiranu tiskanu pločicu sa ATmega1284P mikrokontrolerom bila bi zbog razlike u rasporedu pinova na mikrokontrolerima pa bi pinovi trebali biti drugačije definirani.




**Slika 21. Oprema korištena za programiranje**

#### 4.3.1. Definiranje pinova i početak programa

Na početku programa potrebno je definirati pinove mikrokontrolera koje ćemo koristiti za slanje podataka prema upravljačkoj jedinici robota. Definira se DataPort kao port D, port D sastoji se od 8 pinova preko kojih će se slati podaci robotu. Također je potrebno definirati STROBE i BUSY pin strobe definiran je kao pin 56, a busy pin kao pin 57. Svi pinovi i konstante definiraju se pomoću naredbe #define.

```
#define DataPort PORTD
#define STROBE 56 /*PORTF.2*/
#define BUSY 57 /*PORTF.3*/
```

**Slika 22. Definiranje pinova**

Nakon definiranja pinova potrebno je napisati što će se dogoditi prilikom pokretanja programa, odnosno programirati funkciju setup(). U funkciji setup() određuju se vrste pinova i postavlja se njihovo početno stanje. Tako je uz pomoć naredbe pinMode() definiran pin strobe kao izlazni pin, a pin busy kao ulazni pin s omogućenim unutarnjim pull-up otpornikom. Pomoću funkcije digitalWrite() strobe pin postavlja se u visoko stanje. U setup() funkciji se također pomoću naredbe Serial.println() prilikom pokretanja programa ispisuju ime programa i moguće naredbe.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
  DDRD=0xff;
  pinMode(STROBE,OUTPUT);
  digitalWrite(STROBE,HIGH);
  pinMode(BUSY,INPUT_PULLUP);
  Serial.println("Program za upravljanje robotom RM-501");
  Serial.println("Popis naredbi:");
  Serial.println("i - inicijalizacija");
  Serial.println("r - reset");
  Serial.println("C - zatvaranje hvataljke");
  Serial.println("O - otvaranje hvataljke");
  Serial.println("h - povratak u početnu poziciju");
  Serial.println("s - povratak u poziciju za isključivanje");
  Serial.println("x - upisivanje vanjske koordinate x");
  Serial.println("y - upisivanje vanjske koordinate y");
  Serial.println("z - upisivanje vanjske koordinate z");
  Serial.println("f - upisivanje vanjske koordinate fi");
  Serial.println("p - upisivanje vanjske koordinate psi");
  Serial.println("M - pokreće računanje kinematičkog problema");
  Serial.println("Primjer unosa: x300,y-27,z200,f-90,p0,M");
  Serial.println();
```

**Slika 23. Funkcija setup()**

### 4.3.2. Slanje podataka robotu

Slanje podataka robotu ostvaruje se preko ranije definiranog porta D kao DataPorta uz pomoć dvije funkcije. Da bi se podaci mogli slati od mikrokontrolera do upravljačke jedinice robota potrebna je nisko stanje busy pina i visoko stanje strobe pina. Strobe i busy pinovi se koriste za zaustavljanje i pokretanje prijenosa podataka. Kada je strobe linija u viskom stanju to je znak da je upravljačka jedinica robota uključena i povezana s kinematičkim kontrolerom. Pomoću linije busy upravljačka jedinica robota upravlja brzinom prijenosa podataka, kad je busy u viskom stanju upravljačka jedinica robota ne može primati podatke od kontrolera, podaci od kontrolera se šalju tek kad je busy u niskom stanju. Kada je kontroler spreman poslati podatke prema upravljačkoj jedinici postavlja strobe signal u nisko stanje na jednu milisekundu što daje znak upravljačkoj jedinici da su podaci spremni za preuzimanje. Prva funkcija nazvana WriteParallel() napisana je tako da postavlja jedan znak upisane naredbe na DataPort i provjerava stanja busy i strobe pinova te ranije opisanim postupkom javlja upravljačkoj jedinici da preuzme podatke.

```
void WriteParallel(unsigned char c){
    DataPort=c;
    while(digitalRead(BUSY)==HIGH);
    digitalWrite(STROBE,LOW);
    delay(1);
    digitalWrite(STROBE,HIGH);
    while(digitalRead(BUSY)==HIGH);
    }
```

**Slika 24. Funkcija WriteParallel**

Funkcija SendToRobot() koristi se za slanje naredbe robotu na način da upisanu naredbu uzima znak po znak i uzastopno dok god ima znakova poziva funkciju WriteParallel() pomoću koje se ti znakovi šalju prema upravljačkoj jedinici robota.

```
void SendToRobot(String STR){
  unsigned char i;
  i=0;
  while(STR[i]){
    WriteParallel(STR[i]);
    i++;
  }
  WriteParallel(0x0D); //add CR
  delay(10);
  }
```

**Slika 25. Funkcija SendToRobot()**

### 4.3.3. Prepoznavanje upisane naredbe

U glavnom dijelu koda odnosno u loop() funkciji konstantno se pozivaju dvije funkcije recive() i toRobot(). Funkcija recive() provjerava je li išta upisano u naredbenu liniju te prilikom upisa naredbu sprema u varijablu. Funkcija toRobot() pomoću switch case strukture provjerava koja je naredba upisana i ako je naredba važeća poziva funkciju koju ta naredba izvršava. Slika 26 prikazuje samo kraći dio funkcije toRobot().

```
void loop() {
  // put your main code here, to run repeatedly:

   recive();

   toRobot();

}

void recive(){
  while(Serial.available()>0 && go==false){
   rd=Serial.read();
    go=true;
  }
}


void toRobot(){
  if(go==true){
    switch(rd){
      case 'C':
      SendToRobot("GC \r");
      go=false;
      break;
```

**Slika 26. Funkcije recive() i toRobot()**

### 4.3.4. Računanje kinematičkog problema i pokretanje robota

Upisivanjem jedne ili više od naredbi x, y, z, fi ili psi praćene iznosom na koji ih želimo postaviti (npr. x100,y200,) switch case prepoznaje naredbu te poziva jednu od toFloat funkcija. Funkcija toFloat() napisana je tako da preuzima znakove napisane u naredbenu liniju prilikom postavljanja iznosa koordinate te ih pretvara u decimalni broj i sprema u odgovarajuće varijable. Ovaj dio koda je potreban da bi program mogao računati sa upisanim vrijednostima.

```
void toFloatx(){
  while (Serial.available() > 0) {
    int inChar = Serial.read();

    if (inChar != 'e') {
      inString += (char)inChar;

    }
    else {
      Serial.print("Unesena vrijednost: ");
      Serial.print(inString);
      x=inString.toFloat();
      Serial.print("Nakon pretvorbe u float x=");
      Serial.println(x);
      inString = "";
      k=false;
    }
  }
  }
```

**Slika 27. Funkcija toFloat()**

Ako na kraju naredbe za promjenu koordinata dodamo naredbu „M“ (npr. x100,y200,M) switch case poziva funkciju moveRobot() koja računa inverzni kinematički problem robota pomoću ranije navedenih jednadžbi i dobivene parametre sprema i zaokružuje jer se robotu kao parametri mogu poslati samo cijeli brojevi.

```
q1=atan(y/x);
fi*=pi/180;
x3=x-(L4+L5)*cos(fi)*cos(q1);
y3=y-(L4+L5)*cos(fi)*sin(q1);
z3=z-(L4+L5)*sin(fi);
d=sqrt(pow(x3,2)+pow(y3,2)+pow((z3-L1),2));
b=(pow(L2,2)+pow(d,2)-pow(L3,2))/(2*L2*d);
g=(z3-L1)/(sqrt(pow(x3,2)+pow(y3,2)));

if(b<=-1 || b>=1){
Serial.println("Greška prilikom izračuna kosinusa.");
return 0;
}

fi*=180/pi;
beta=acos(b)*180/pi;
gama=atan(g)*180/pi;
q1*=180/pi;
q2=90-beta-gama;
q3p=(pow(d,2)-pow(L2,2)-pow(L3,2))/(2*L2*L3);
q3p*=pi/180;
q3=acos(q3p)*180/pi;
q4=90-(q2+q3+fi);
q5=psi;

a1=static_cast<int>((-q1/0.025)-C1);
a2=static_cast<int>((-((q2-55)/0.025)-C2));
```

**Slika 28. Dio funkcije moveRobot() za izračun i spremanje parametara**

Nakon računanja kinematičkog problema i spremanja parametara funkcija moveRobot() računa razliku između parametara točke u koju šaljemo robota i parametara točke u kojoj se robot trenutno nalazi i te vrijednosti sprema u odgovarajuće delta varijable te provjerava jesu li one u dopuštenim granicama (Slika 29.). Nastale delta varijable su cijeli brojevi, a robotu se mogu slati samo znakovi pa je potrebno brojeve spremiti u string oblik pomoću naredbe String(). Svaka varijabla delta je sada spremljena kao string u odgovarajuću varijablu deltas. Sve varijable deltas je sada potrebno zajedno sa naredbom „MI“ pospremiti u jedan string koji se zove zaRobota. Na kraju se string zaRobota pomoću funkcije SendToRobot() šalje prema upravljačkoj jedinici robota i robot se pokreće u željene koordinate (Slika 30.).

```
  deltal=(al-alp);
  delta2=(a2-a2p);
  delta3=(a3-a3p);
  delta4=(a4-a4p);
  delta5=(a5-a5p);

// provjera jesu li parametri unutar zadanih granica
  if(deltal<=-6000||deltal>=6000){
    Serial.println("Greska deltal");
    return 0;
    }

  if(delta2<=-2600||delta2>=2600){
    Serial.println("Greska delta2");
    return 0;
    }
```

**Slika 29. Dio funkcije moveRobot() za računanje i provjeru parametara**

```
  // Pretvaranje decimalnih brojeva u znakove
    deltals=String(deltal);
    delta2s=String(delta2);
    delta3s=String(delta3);
    delta4s=String(delta4);
    delta5s=String(delta5);

  // Stvaranje jedne naredbe za slanje robotu
    zaRobota+="MI ";
    zaRobota+=deltal;
    zaRobota+=",";
    zaRobota+=delta2;
    zaRobota+=",";
    zaRobota+=delta3;
    zaRobota+=",";
    zaRobota+=delta4;
    zaRobota+=",";
    zaRobota+=delta5;
    zaRobota+=",";
    zaRobota+="0";

   Serial.println(zaRobota);
   SendToRobot(zaRobota);
   zaRobota="";
```

**Slika 30. Dio funkcije moveRobot() za slanje naredbe robotu**

### 4.4. Povezivanje preko Bluetootha

Bluetooth je na ovom kinematičkom kontroleru projektiran tako da nema mogućnosti promjene glavnog goda na samom mikrokontroleru nego nakon što se program učita pomoću USB kabla preko Bluetootha se mogu samo upisivati naredbe za pokretanje robota i dobivati informacije od mikrokontrolera. Da bi se povezali potrebno je aktivirati Bluetooth modul te spojiti računalo s Bluetooth modulom što na računalu stvara virtualni COM port kojeg izabiremo za spajanje u Arduino programu te preko serial monitora bežično šaljemo naredbe robotu.

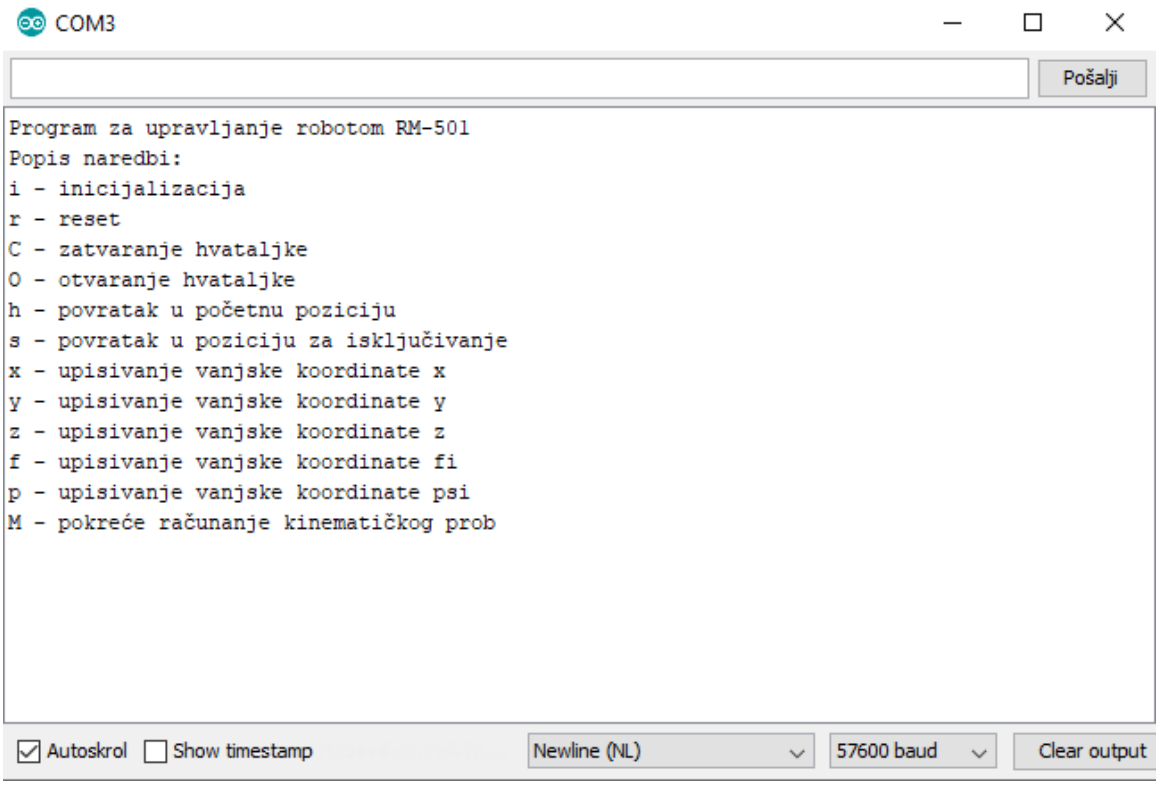
### 4.5. Pokretanje robota i zadavanje naredbi

Nakon što smo robota upalili i povezali sa računalom potrebno je pokrenuti inicijalizaciju robota koja izvršava određeni set naredbi koje provjeravaju mirkoprekidače robota te postavljaju robota u početnu poziciju. Inicijalizacije se mora provesti nakon svakog pokretanja robota jer ga inače nije moguće pomicati.

Naredbe se zadaju u serial monitoru Arduino programa jednostavnim upisivanjem jedne od ponuđenih naredbi koje nam program ispiše prilikom pokretanja. Nakon upisivanja naredbe program nas dalje navodi uputama.

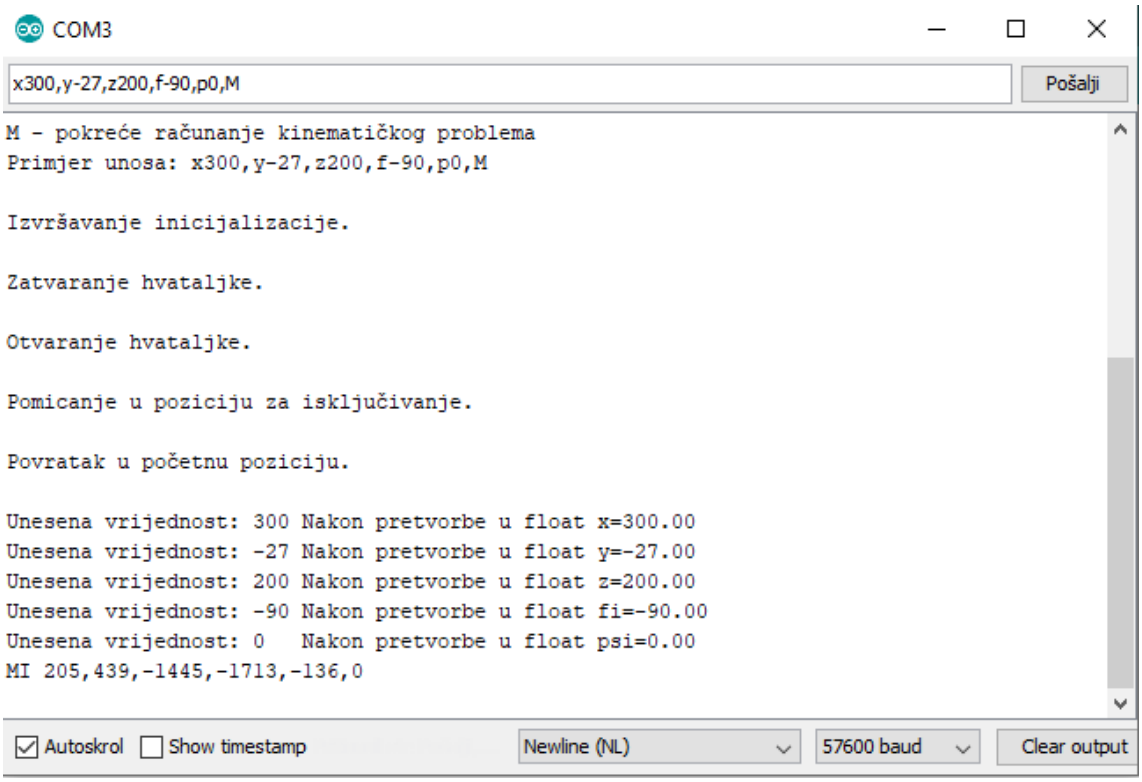
**Tablica 3. Naredbe**

| UPISANA NAREDBA | IZLAZNI KOD | OPIS |
|---|---|---|
| i | NT<br>MI -6000,-2600,1800,1200,-1200,0<br>MI 0,2400,-1600,-2000, 140,0<br>HO | Inicijalizacija |
| h | OG | Povratak u početnu poziciju |
| C | GC | Zatvaranje hvataljke |
| O | GO | Otvaranje hvataljke |
| r | RS | Reset |
| s | MI 0, 2400, -1600, -2000, 140, 0 | Pomak u poziciju za isključivanje |
| x100,y200,M | MI delta1s,delta2s,delta3s… | Pomiče robota u vanjske koordinate |




**Slika 31. Početak programa**

Upisivanjem jedne od naredbi program šalje odgovarajuću naredbu robotu te nam javlja koju smo naredbu izabrali.



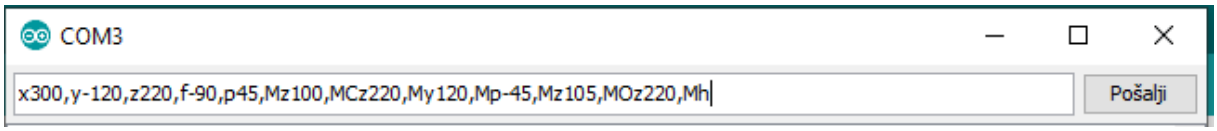

**Slika 32. Odgovori programa na upisane naredbe**

Ukoliko dođe do greške prilikom izračuna ili se robot fizički ne može pomaknuti u zadane koordinate sučelje javlja grešku.

```
Unesena vrijednost: -300        Nakon pretvorbe u float x=-300.00
Unesena vrijednost: 200 Nakon pretvorbe u float y=200.00
Greška prilikom izračuna kosinusa.
Unesena vrijednost: 300 Nakon pretvorbe u float x=300.00
Unesena vrijednost: 200 Nakon pretvorbe u float y=200.00
Greska delta2
```

**Slika 33. Javljanje greške**

Upisivanjem više naredbi odjednom u naredbenu liniju moguće je programirati neku od lakših zadaća kao što je uzmi i postavi (eng. pick and place) gdje robot uzima predmet s jedne strane prepreke te ga prenosi i postavlja s druge strane iste prepreke.




**Slika 34. Naredba za pick and place**

# 5. ZAKLJUČAK

Ovaj kinematički kontroler iako vrlo jednostavan izvršava svoju namjenu. Omogućeno je prilagođavanje USB i Bluetooth komunikacije paralelnoj Centronics komunikaciji te je samim time ostvareno povezivanje robota s računalom. Također je izračunavanjem inverznog kinematičkog problema omogućeno pokretanje robota zadavanjem vanjskih koordinata robota.

Mogućnosti za unaprjeđivanje ovog rada su velike. U ovome trenutku za upisivanje naredbi korisiti se serial monitor programa Arduino koji je funkcionalan, ali vrlo brzo postaje nepregledan. Da se taj problem riješi bilo bi potrebno izraditi program sa grafičkim sučeljem u kojem bi se nalazile sve naredbe robota te bi bile ostvarive uz klik miša ili dodir prsta ako se izradi aplikacija za android pomoću koje bi se preko Bluetootha moglo upravljati robotom. Nadogradnjom koda moguće je ostvariti obavljanje kompliciranijih zadataka kao što je paletizacija ili depaletizacija odnosno programirati robota da izvršava neki određeni zadatak i na taj način ga uvelike približiti današnjim edukacijskim robotima.

Sve u svemu uz malo uloženog truda, vremena i novčanih sredstava moguće je relativno lako modernizirati zastarjelu opremu te ju tako spasiti od skupljanje prašini i učiniti je ponovno korisnom. Iako je ovaj robot star te je zamijenjen novim robotima i dalje je koristan za edukaciju, sam primjer je ova modernizacija kroz koju se može mnogo naučiti o izradi tiskanih pločica, mikrokontrolerima, programiranju i ostalim aspektima robotike.

# LITERATURA

[1] Uputstva za upotrebu za Mitsubishi RM-501

[2] Šurina T., Crneković M., Industrijski roboti, 1990. Školska knjiga, Zagreb

[3] Crneković M., Zorc D., Kinematic controller for a mitsubishi RM501 robot, 2012. FAMENA

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

[5] https://www.microchip.com

[6] https://www.sparkfun.com

# PRILOG

1. CD
2. Programski kod
3. Shema spajanja pločice sa svim komponentama
4. Shema spajanja pločice sa konektorima za module

Programski kod:

```
#define DataPort PORTD
#define STROBE 56 /*PORTF.2*/
#define BUSY 57 /*PORTF.3*/


void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
  DDRD=0xff;
  pinMode(STROBE,OUTPUT);
  digitalWrite(STROBE,HIGH);
  pinMode(BUSY,INPUT_PULLUP);
  Serial.println("Program za upravljanje robotom RM-501");
  Serial.println("Popis naredbi:");
  Serial.println("i - inicijalizacija");
  Serial.println("r - reset");
  Serial.println("C - zatvaranje hvataljke");
  Serial.println("O - otvaranje hvataljke");
  Serial.println("h - povratak u početnu poziciju");
  Serial.println("s - povratak u poziciju za isključivanje");
  Serial.println("x - upisivanje vanjske koordinate x");
  Serial.println("y - upisivanje vanjske koordinate y");
  Serial.println("z - upisivanje vanjske koordinate z");
  Serial.println("f - upisivanje vanjske koordinate fi");
  Serial.println("p - upisivanje vanjske koordinate psi");
  Serial.println("M - pokreće računanje kinematičkog problema");
  Serial.println("Primjer unosa: x300,y-27,z200,f-90,p0,M");
  Serial.println();


}
//Postavlja jedan znak na DataPort
void WriteParallel(unsigned char c){
    DataPort=c;
    while(digitalRead(BUSY)==HIGH);
    digitalWrite(STROBE,LOW);
    delay(1);
    digitalWrite(STROBE,HIGH);
    while(digitalRead(BUSY)==HIGH);
    }

//Funkcija za slanje naredbe robotu
void SendToRobot(String STR){
  unsigned char i;
  i=0;
  while(STR[i]){
    WriteParallel(STR[i]);
    i++;
```

```
  }
  WriteParallel(0x0D); //add CR
  delay(10);
  }

//definiranje varijabli
boolean go=false;
boolean k=true;
char rd;
String inString="";
float x,y,z,fi,psi;
float x3, y3, z3, d, b, g;
float q1, q2, q3, q4, q5, q3p;
float q1r, q2r, q3r, q4r, q5r, fr;
float beta, gama;
const float pi=3.14159;
int L1, L2, L3, L4, L5;
int a1, a2, a3, a4, a5;
int C1, C2, C3, C4, C5;
int a1p, a2p, a3p, a4p, a5p;
int delta1, delta2, delta3, delta4, delta5;
String delta1s,delta2s,delta3s,delta4s,delta5s;
String zaRobota="";
int x1,y1,z1;



// Funkcija za izvršenje inicijalizacije robota
void Init1(){
  SendToRobot("NT");
  SendToRobot("MI -6000, -2600, 1800, 1200, -1200, 0");
  SendToRobot("HO");
}
// Funkcije pomoću kojih se upisana vrijednost koordinate pretvara u decimalni broj
void toFloatx(){
  while (Serial.available() > 0 && k==true) {
    int inChar = Serial.read();

    if (inChar != ',') {
      inString += (char)inChar;

    }
    else {
      Serial.print("Unesena vrijednost: ");
      Serial.print(inString);
      x=inString.toFloat();
      Serial.print("\tNakon pretvorbe u float x=");
      Serial.println(x);
      inString = "";
      k=false;
```

```
   }
  }
  }

void toFloaty(){
  while (Serial.available() > 0 && k==true) {
    int inChar = Serial.read();

    if (inChar != ',') {
      inString += (char)inChar;

    }
    else {
      Serial.print("Unesena vrijednost: ");
      Serial.print(inString);
      y=inString.toFloat();
      Serial.print("\tNakon pretvorbe u float y=");
      Serial.println(y);
      inString = "";
      k=false;
    }
  }
  }

  void toFloatz(){
  while (Serial.available() > 0 && k==true) {
    int inChar = Serial.read();

    if (inChar != ',') {
      inString += (char)inChar;

    }
    else {
      Serial.print("Unesena vrijednost: ");
      Serial.print(inString);
      z=inString.toFloat();
      Serial.print("\tNakon pretvorbe u float z=");
      Serial.println(z);
      inString = "";
      k=false;
    }
  }
  }

  void toFloatfi(){
  while (Serial.available() > 0 && k==true) {
    int inChar = Serial.read();

    if (inChar != ',') {
      inString += (char)inChar;
```

```
    }
    else {
      Serial.print("Unesena vrijednost: ");
      Serial.print(inString);
      fi=inString.toFloat();
      Serial.print("\tNakon pretvorbe u float fi=");
      Serial.println(fi);
      inString = "";
      k=false;
    }
  }
  }

void toFloatpsi(){
  while (Serial.available() > 0 && k==true) {
    int inChar = Serial.read();

    if (inChar != ',') {
      inString += (char)inChar;

    }
    else {
      Serial.print("Unesena vrijednost: ");
      Serial.print(inString);
      psi=inString.toFloat();
      Serial.print("\tNakon pretvorbe u float psi=");
      Serial.println(psi);
      inString = "";
      k=false;
    }
  }
  }


// Funkcija koja računa kinematički problem te pokreće robota u željene koordinate
unsigned char moveRobot(){
L1=250;
  L2=220;
  L3=160;
  L4=215;
  L5=0;

  C1=0;
  C2=163;
  C3=8;
  C4=925;
  C5=925;

  q1=atan(y/x);
```

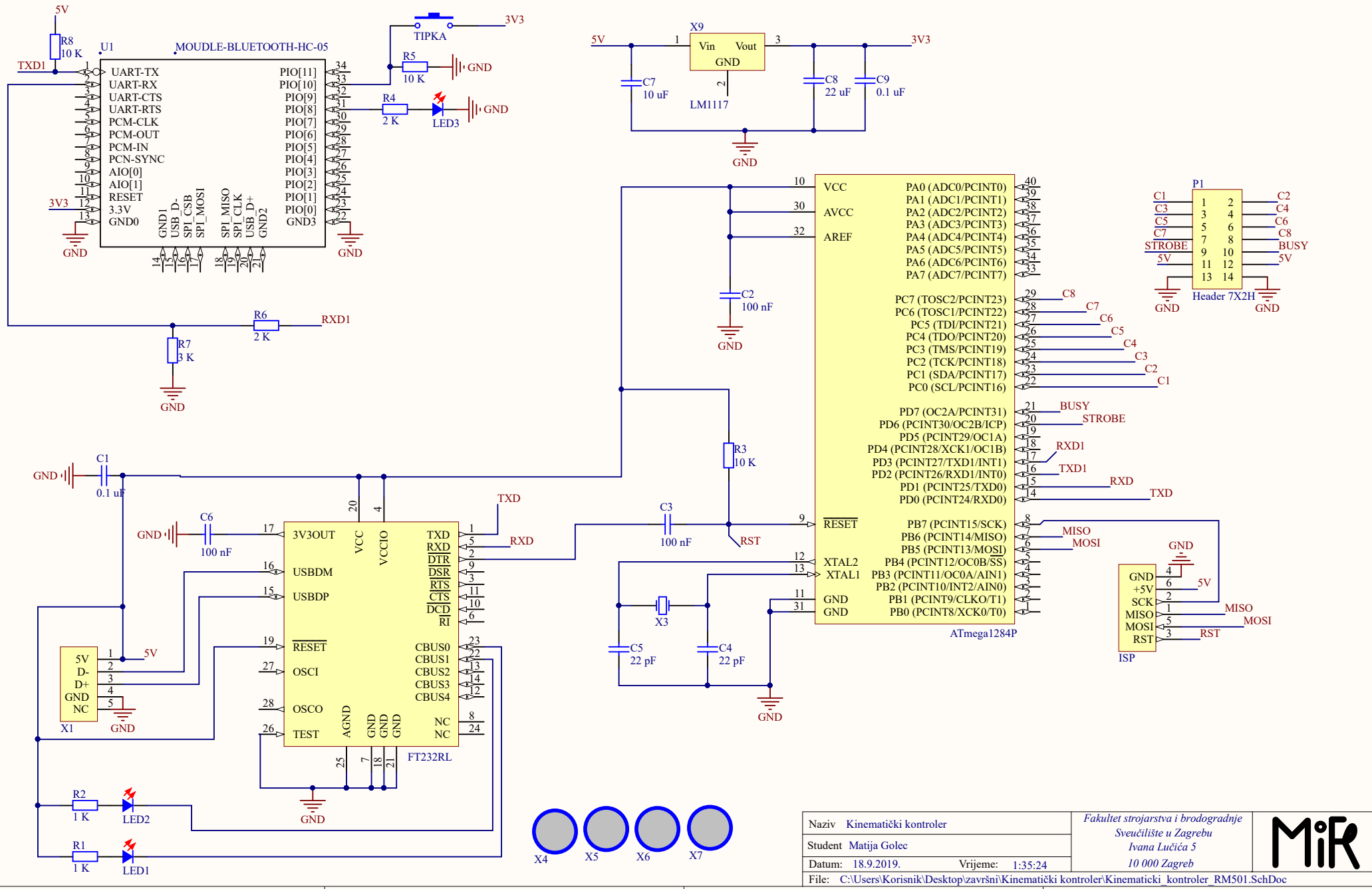
```
 fi*=pi/180;
 x3=x-(L4+L5)*cos(fi)*cos(q1);
 y3=y-(L4+L5)*cos(fi)*sin(q1);
 z3=z-(L4+L5)*sin(fi);
 d=sqrt(pow(x3,2)+pow(y3,2)+pow((z3-L1),2));
 b=(pow(L2,2)+pow(d,2)-pow(L3,2))/(2*L2*d);
 g=(z3-L1)/(sqrt(pow(x3,2)+pow(y3,2)));

if(b<=-1 || b>=1){
 Serial.println("Greška prilikom izračuna kosinusa.");
 return 0;
 }

 fi*=180/pi;
 beta=acos(b)*180/pi;
 gama=atan(g)*180/pi;
 q1*=180/pi;
 q2=90-beta-gama;
 q3p=(pow(d,2)-pow(L2,2)-pow(L3,2))/(2*L2*L3);
 q3p*=pi/180;
 q3=acos(q3p)*180/pi;
 q4=90-(q2+q3+fi);
 q5=psi;

 a1=static_cast<int>((-q1/0.025)-C1);
 a2=static_cast<int>((-((q2-55)/0.025)-C2));
 a3=static_cast<int>((-(q3-45)/0.025)-C3);
 a4=static_cast<int>((-(q4+q5)/0.075)-C4);
 a5=static_cast<int>(((q4-q5)/0.075)-C5);


 q1=-0.025*(a1+C1);
 q2=55-(0.025*(a2+C2));
 q3=45-(0.025*(a3+C3));
 q4=(-0.075*(a4+C4-a5-C5))/2;
 q5=(-0.075*(a4+C4+a5+C5))/2;

 q1r=q1*pi/180;
 q2r=q2*pi/180;
 q3r=q3*pi/180;
 q4r=q4*pi/180;
 q5r=q5*pi/180;
 fr=(pi/2)-(q2r+q3r+q4r);
 psi=q5;

 x1=cos(q1r)*(L2*sin(q2r)+L3*sin(q2r+q3r))+cos(fr)*cos(q1r)*(L4+L5);
 y1=sin(q1r)*(L2*sin(q2r)+L3*sin(q2r+q3r))+cos(fr)*cos(q1r)*(L4+L5);
 z1=L1+L2*cos(q2r)+L3*cos(q2r+q3r)+(L4+L5)*sin(fr);

 delta1=(a1-a1p);
```

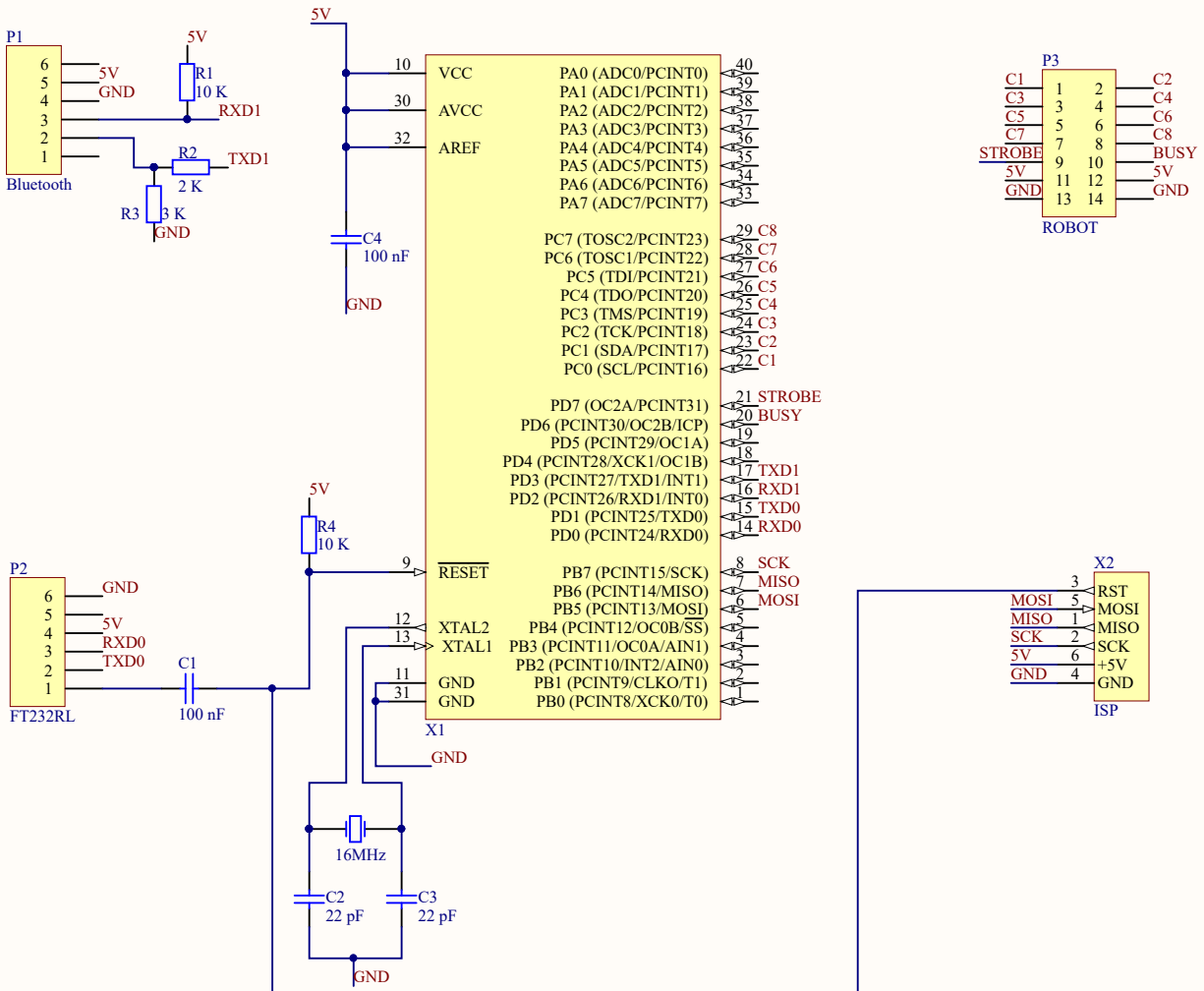
```
  delta2=(a2-a2p);
  delta3=(a3-a3p);
  delta4=(a4-a4p);
  delta5=(a5-a5p);

// provjera jesu li parametri unutar zadanih granica
  if(delta1<=-6000||delta1>=6000){
    Serial.println("Greska delta1");
    return 0;
    }

  if(delta2<=-2600||delta2>=2600){
    Serial.println("Greska delta2");
    return 0;
    }

  if(delta3<=-1800||delta3>=1800){
    Serial.println("Greska delta3");
    return 0;
    }

  if((delta4-delta5)<=-2400||(delta4-delta5)>=2400){
    Serial.println("Greska delta 4-5");
    return 0;
    }

  if((delta4+delta5)<=-9600||(delta4+delta5)>=9600){
    Serial.println("Greska delta4+5");
    return 0;
    }
//Sprema parametre
  a1p=a1;
  a2p=a2;
  a3p=a3;
  a4p=a4;
  a5p=a5;

// Pretvaranje decimalnih brojeva u znakove
  delta1s=String(delta1);
  delta2s=String(delta2);
  delta3s=String(delta3);
  delta4s=String(delta4);
  delta5s=String(delta5);

// Stvaranje jedne naredbe za slanje robotu
  zaRobota+="MI ";
  zaRobota+=delta1;
  zaRobota+=",";
  zaRobota+=delta2;
  zaRobota+=",";
```

```
  zaRobota+=delta3;
  zaRobota+=",";
  zaRobota+=delta4;
  zaRobota+=",";
  zaRobota+=delta5;
  zaRobota+=",";
  zaRobota+="0";

 Serial.println(zaRobota);
 SendToRobot(zaRobota);
 zaRobota="";

}

void loop() {
  // put your main code here, to run repeatedly:

   recive();

   toRobot();

}

//Funkcija koja sprema naredbu iz naredbene linije
void recive(){
  while(Serial.available()>0 && go==false){
   rd=Serial.read();
    go=true;
  }
}

// Funkcija koja izvršava unesenu naredbu
void toRobot(){
  if(go==true){
    switch(rd){
      case 'C':
      SendToRobot("GC");
      Serial.println("Zatvaranje hvataljke.");
      Serial.println();
      go=false;
      break;
      case 'O':
      SendToRobot("GO");
      Serial.println("Otvaranje hvataljke.");
      Serial.println();
      go=false;
      break;
      case 'h':
      SendToRobot("OG");
      Serial.println("Povratak u početnu poziciju.");
```

```
Serial.println();
a1p=0;
a2p=0;
a3p=0;
a4p=0;
a5p=0;
go=false;
break;
case 'i':
Init1();
Serial.println("Izvršavanje inicijalizacije.");
Serial.println();
go=false;
break;
case 'r':
SendToRobot("RS");
Serial.println("Reset.");
Serial.println();
go=false;
break;
case 'x':
k=true;
while(k==true){
  toFloatx();
  }
go=false;
break;
case 'y':
k=true;
while(k==true){
  toFloaty();
  }
go=false;
break;
case 'z':
k=true;
while(k==true){
  toFloatz();
  }
go=false;
break;
case 'f':
k=true;
while(k==true){
  toFloatfi();
  }
go=false;
break;
case 'p':
k=true;
```

```
    while(k==true){
      toFloatpsi();
      }
    go=false;
    break;
    case 'M':
    moveRobot();
    go=false;
    break;
    case 's':
    SendToRobot("MI 0, 2400, -1600, -2000, 140, 0");
    a1p=0;
    a2p=2400;
    a3p=-1600;
    a4p=-2000;
    a5p=140;
    Serial.println("Pomicanje u poziciju za isključivanje.");
    Serial.println();
    go=false;
    break;
    default:
    go=false;
    break;
    }
  go=false;
  }
}
```


MOUDLE-BLUETOOTH-HC-05
UART-TX
UART-RX
UART-CTS
UART-RTS
PCM-CLK
PCM-OUT
PCM-IN
PCN-SYNC
AIO[0]
AIO[1]
RESET
3.3V
GND0
PIO[11]
PIO[10]
PIO[9]
PIO[8]
PIO[7]
PIO[6]
PIO[5]
PIO[4]
PIO[3]
PIO[2]
PIO[1]
PIO[0]
GND3
GND1
USB_D-
SPI_CSB
SPI_MOSI
SPI_MISO
SPI_CLK
USB_D+
GND2
LM1117
Vin
Vout
GND
ATmega1284P
VCC
AVCC
AREF
RESET
XTAL2
XTAL1
FT232RL
3V3OUT
VCC
VCCIO
USBDM
USBDP
RESET
OSCI
OSCO
TEST
AGND
TXD
RXD
DTR
DSR
RTS
CTS
DCD
RI
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
NC
Header 7X2H
ISP
TIPKA
LED1
LED2
LED3
Naziv Kinematički kontroler
Student Matija Golec
Datum: 18.9.2019.
Vrijeme: 1:35:24
File: C:\Users\Korisnik\Desktop\završni\Kinematički kontroler\Kinematicki_kontroler_RM501.SchDoc
Fakultet strojarstva i brodogradnje
Sveučilište u Zagrebu
Ivana Lučića 5
10 000 Zagreb




| | | | |
|---|---|---|---|
| Naziv | | *Fakultet strojarstva i brodogradnje* | MiR |
| Student | | *Sveučilište u Zagrebu* | |
| Datum: 19.9.2019. | Vrijeme: 10:21:45 | *Ivana Lučića 5* | |
| File: C:\Users\Korisnik\Desktop\završni\Kinematički kontroler\KINkon.SchDoc | | *10 000 Zagreb* | |