UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Luka Levstek

Sistem za krmiljenje gravirne naprave

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: izr. prof. dr. Patricio Bulić

Ljubljana, 2013

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00539 / 2013
Datum: 15.9.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **LUKA LEVSTEK**

Naslov: **SISTEM ZA KRMILJENJE GRAVIRNE NAPRAVE**
A SYSTEM TO CONTROL AN ENGRAVING DEVICE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Implementirajte sistem za krmiljenje gravirne naprave. Osnovni krmilni modul za krmiljenje motorjev gravirne naprave naj bo zgrajen na osnovi sistema na čipu STM32F407 z jedrom ARM Cortex M4. Uporabniški vmesnik, ki krmilnemu modulu pošilja podatke s koordinatami pa načrtujete v programskem jeziku Visual Basic. Za potrebe testiranja sistema implementirajte preprosto gravirno napravo.

Mentor:
izr. prof. dr. Patricio Bulić

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani **Luka Levstek**, z vpisno številko **63050165**, sem avtor diplomskega dela z naslovom

SISTEM ZA KRMILJENJE GRAVIRNE NAPRAVE

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom **izr. prof. dr. Patricia Bulića,**
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 20.12.2013 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju izr. prof. dr. Patriciu Buliću za pomoč in nasvete pri opravljanju diplomskega dela.

Posebej se zahvaljujem staršem, ki so mi študij omogočili in mi vsa ta leta dajali vzpodbudo.

Kazalo

Seznam uporabljenih kratic

0DL	Nič,desno,levo
ADC	Analog to Digital Converter – analogno digitalni pretvornik
ARM	Advanced Risc Machine
CCW	Clockwise – smer urinega kazalca
CNC	Computer Numerical Control – računalniška numerična kontrola
CSS	Cascading Style Sheets - kaskadne stilske podloge
CW	Counter Clockwise – obratna smer urinega kazalca
DAC	Digital to Analog Converter – digitalno analogni pretvornik
EXT	External – zunanje
GND	Ground – masa, ozemljitev
HTML	Hyper Text Markup Language - jezik za označevanje nadbesedila
I2C	Inter Integrated Circuit
LED	Light-Emitting Diode – Svetleča dioda
LPT	Line Print Terminal – vrata, namenjena tiskalnikom
LSB	Least Significant Bit – Bit z najnižjo težo
MIT	Massachusetts Institute of Technology
MSB	Most Significant Bit – Bit z najvišjo težo
NC	Numerical Control – numerična kontrola
PMMA	Poly Methyl MethAcrylate – polimetil metaakrilat
SPI	Serial-Peripheral interface – serijski vmesnik za periferijo
TTL	Transistor-Transistor Logic – tranzistorska logika
USART	Universal Synchronus Asynchronus Receiver Transmitter – univerzalini sinhroni asinhroni sprejemnik oddajnik
VB.NET	Visual Basic Dot Net – programski jezik
XML	EXtensible Markup Language – razširljiv označevalni jezik

Povzetek

V okviru diplomske naloge sem izdelal napravo CNC za graviranje. Predvsem me je zanimal celoten postopek od same zamisli, načrtovanja ter nato izdelave. Pri izdelavi sem se srečal z vrsto težav, katere sem uspel odpraviti.

Naprava CNC je sestavljena iz strojnega dela, krmilnega dela in uporabniškega vmesnika. Za premikanje obdelovalnega orodja sem uporabil tri koračne motorje, ki so gnani z gonilnikom L297-8. Za krmiljenje koračnih motorjev in komunikacijo med uporabniškim in strojnim delom skrbi razvojni sistem, ki je zasnovan na mikrokrmilniku STM32F407 z jedrom ARM Cortex M4. Komunikacija poteka preko serijskega komunikacijskega vmesnika USART (*Universal Synchronus Asynchronus Receiver Transmitter).*

Programska koda uporabniškega vmesnika je napisana v programskem jeziku Visual Basic. Uporabniški vmesnik v okviru diplomske naloge omogoča vpis koordinat za risanje linearnih linij.

Pri izdelavi diplomske naloge sem uporabil večino znanj, ki mi jih je dala fakulteta v času študija.

Ključne besede: Naprava CNC, Graviranje, ARM Cortex M4 , STM32F4 Discovery

Abstract

In my thesis I have made CNC machine for engraving. In particular, I was interested in the whole process from the very idea of planning to construction. During the construction process, I met with a number of problems, which I managed to overcome.

CNC machine consists of a machine part, the control component and the user interface. For movement of the engraver I have used three stepping motors that are driven by driver L297-8 feed. To control the stepper motors and communication between the user part and the machine the system STM ARM32F4 Discovery developement board is here. Communication takes place via an interface USART *(Universal Synchronus Asynchronus Receiver Transmitter)*.

The code of the user interface is written in the programming language Visual Basic. The user interface in the thesis enables to enter the coordinates for drawing linear lines.

While making the thesis I have used most of the skills that I have managed to obtain from the faculty during the study.

Keywords: CNC machine, Engraving, ARM Cortex M4, STM32F4 Discovery

1. Uvod

Takoj po koncu 2. svetovne vojne je v ameriški letalski industriji narasla želja po večji natančnosti in kompleksnejši izdelavi polizdelkov. Prav tako so tovarne strmele k manjšim proizvodnim stroškom. Odgovor na to je bila prva NC *(Numerical Control)* naprava, razvita na univerzi MIT*(Massachusetts Institute of Technology)*, ki je bila leta 1954 predstavljena širši javnosti. Z razvojem elektronike so ročno krmiljenje zamenjali računalniki, kar danes poznamo kot CNC *(Computer Numerical Control)*[1].

Množična proizvodnja naprav CNC je botrovala k cenovni dostopnosti, tako da danes skorajda ne najdemo večje kovinske ali lesne industrije, ki ne bi imela naprave CNC.

V primerjavi z delavcem lahko naprava CNC deluje 24 ur na dan, se nikoli ne utrudi in konstantno dela z nespremenljivo natančnostjo. Kljub temu je v večini primerov potrebna tudi prisotnost delavca – operaterja naprave CNC. Operater naprave CNC skrbi za pravilnost obdelave, vloži obdelovanec na delovno mizo in umakne končni izdelek.

Danes poznamo široko paleto obdelovalnih naprav CNC. Od nizkocenovnih, katere najdemo v hobi delavnicah, do takih, katerih os X meri več kot 100 metrov in so namenjene obdelovanju največjih polizdelkov iz enega kosa.

V diplomski nalogi je predstavljen potek izdelave gravirne naprave CNC. Obenem so opisane teoretične osnove posameznih komponent, ki sestavljajo napravo. Na koncu je predstavljen končni rezultat graviranja in predlogi za izboljšavo.

2. Strojni del

2.1 Konstrukcija

Konstruiranje naprave je dejanje pri katerem svoje zamisli in ideje prenesemo na papir. Pri tem je potrebno predvideti vse morebitne težave in napravo konstruirati tako, da težave v celoti odpravimo oz. karseda zmanjšajmo njihov učinek.

Pri konstruiranju naprave je zelo pomemben izbor materialov. V primeru, da uporabimo neustrezne materiale, se to kasneje pozna pri samem delovanju naprave. Praviloma velja, da je kvaliteta pogojena s ceno. S ceno smo navzgor omejeni, zato je potrebno uporabiti najboljši material, ki je še vedno za nas cenovno sprejemljiv. V naslednjem podpoglavju je opis komponent, katere sem uporabil pri konstrukciji naprave CNC.

2.1.1 Osnovni material

Osnovna konstrukcija naprave CNC je narejena iz 15-milimeterskih PMMA *(Poly methyl methacrylate)* plošč (Slika 2.1), nam bolje poznane pod imenom pleksi akrilatno steklo. Glavna lastnost pleksi akrilatnega stekla je trdnost in lahka obdelava materiala. Vrtanje lukenj in vrezovanje navojev v material ni predstavljalo večjih težav. Plošče se lahko ob segrevanju z visoko temperaturo ukrivijo. Obdelane plošče sem prebarval z belo barvo na vodni osnovi.

Slika 2.1: Kos pleksi akrilatne plošče

2.1.2 Navojno vreteno

Navojno vreteno (Slika 2.2) je dimenzij 14x4. Vreteno je široko 14mm in ima hod 4 mm. Hod navoja je razdalja med grebeni. Navojno vreteno je bilo uporabljeno za premikanje orodja v smeri X, Y in Z.

Slika 2.2: Navojno vreteno

2.1.3 Matica s trapeznim navojem

Matica s trapeznim navojem (Slika 2.3) je dimenzij 14x4. Notranji navoj je širok 14mm in ima hod 4mm. Skozi notranjost matice poteka navojno vreteno.

Slika 2.3: Matica s trapeznim navojem

2.1.4 Vodilna os

Vodilna os (Slika 2.4) premera 12mm je izdelana iz kaljenega jekla. Namenjena je natančnemu gibanju orodja po točno določeni poti, katero vodilna os določa.

Slika 2.4: Vodilna os

2.1.5 Drsna puša

Drsna puša (Slika 2.5) ima notranji premer 12mm. Zunanji del je iz jekla, notranji pa je iz teflona. Teflon omogoča lažje drsenje po kaljenem jeklu.

Slika 2.5: Drsna puša

2.1.6 Kroglični ležaj

Kroglični ležaj (Slika 2.6) je vpet na obeh koncih navojnega vretena. Ležaj zmanjšuje trenje pri vrtenju navojnega vretena, gnanega s koračnim motorjem. Zunanji premer ležaja je 22mm, notranji 8mm.

Slika 2.6: Kroglični ležaj

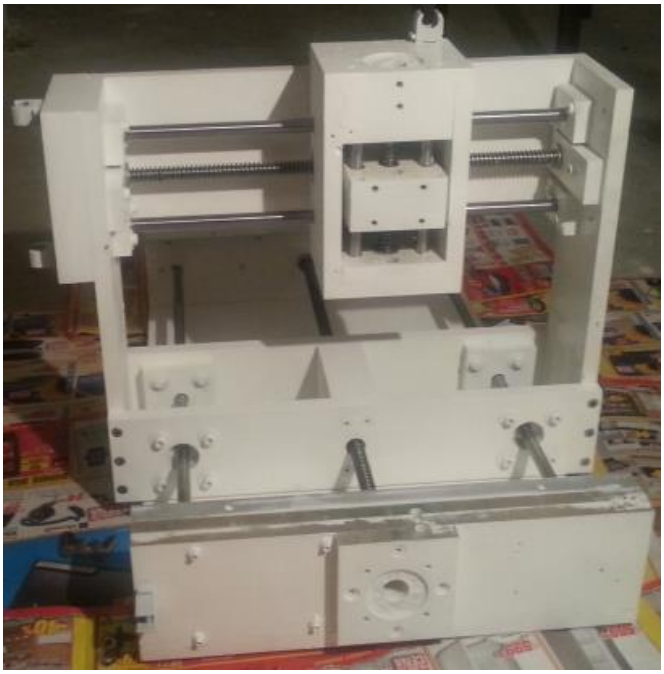
2.1.7 Imbus vijak

Za vijačenje plošč sem uporabil imbus vijake dimenzije M6 z nizko imbus glavo (slika 2.7).

Slika 2.7: Imbus vijak z nizko glavo

2.1.8 Slika konstrukcije

Na sliki 2.8 je prikazana konstrukcija naprave brez koračnih motorjev in vodnikov.

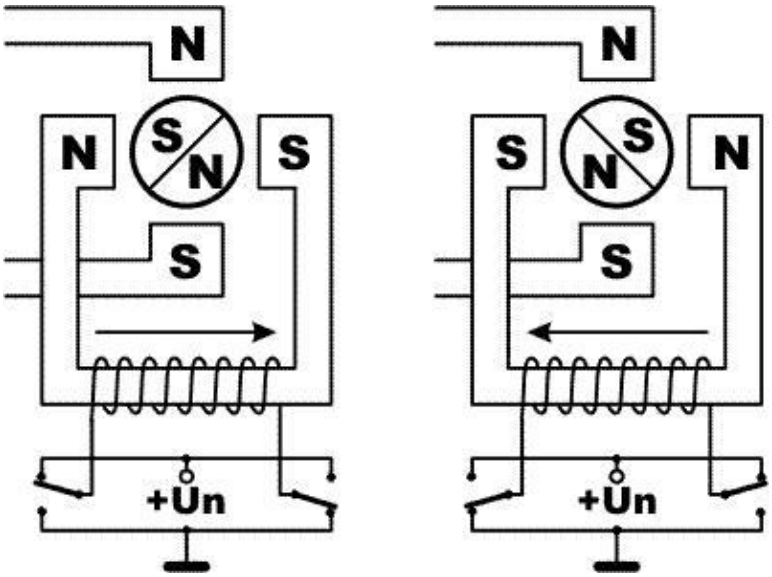


Slika 2.8: Konstrukcija naprave

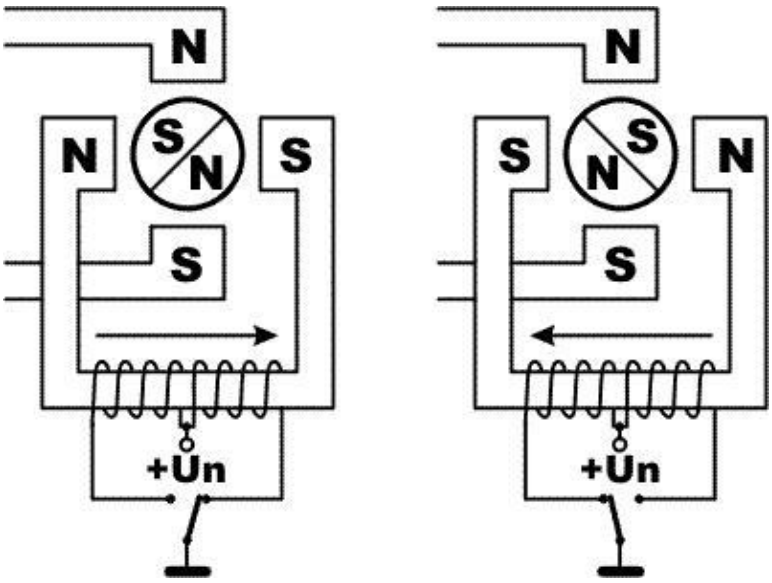
2.2 Koračni motor

2.2.1 Krmiljenje koračnega motorja

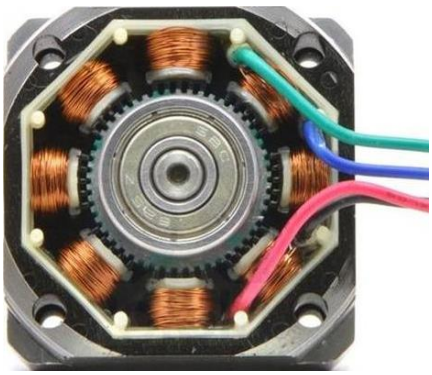
V koračnem motorju se rotor premakne, ko se smer toka v tuljavi spremeni, zato se spremeni tudi magnetno polje na statorskih polih. Glede na izvedbo tuljav ločimo bipolarne in unipolarne koračne motorje. Razlika med unipolarnimi in bipolarnimi motorji je v tem, kako je ta sprememba dosežena. Bipolarni koračni motor ima eno navitje na en polov par, krmili pa se z dvema preklopnima stikaloma, ki izmenično preklapljata napajalno napetost (Slika 2.9).Unipolarni koračni motor ima dve navitji na en polov par in se krmili z enim preklopnim stikalom (Slika 2.10).
Navor pri koračnem motorju je premosorazmeren z gostoto magnetnega pretoka v statorski tuljavi. Odvisen je od števila navitij in jakosti električnega toka, ki teče skozi tuljavo[2].



Slika 2.9: Model bipolarnega koračnega motorja[2]



Slika 2.10: Model unipolarnega koračnega motorja[2]

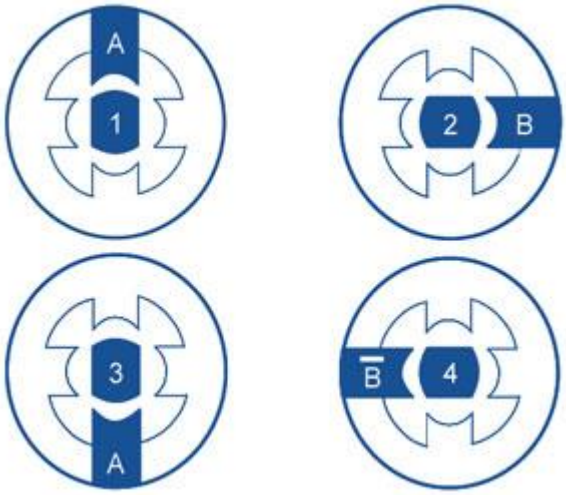


Slika 2.11: Prerez koračnega motorja, ki ima 4 pare navitij [3].

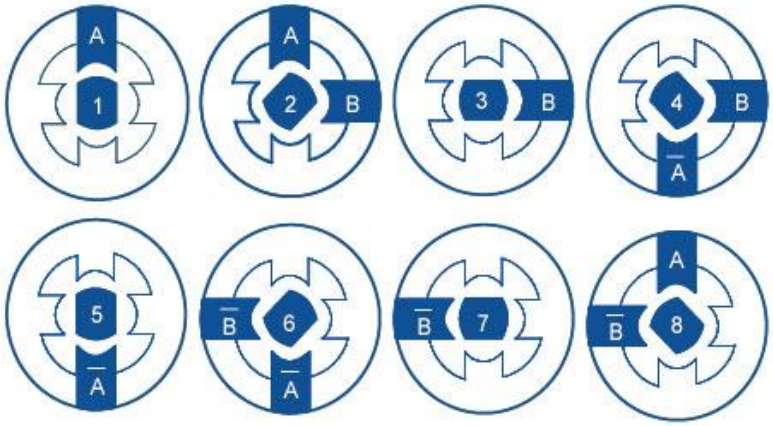
Glede na potrebe lahko koračni motor krmilimo na načine:

- polni korak (eng. Full Step)
- polovično korakanje (Half Stepping)
- in mikro korakanje (Microstepping).

Pri krmiljenju motorja v polnokoračnem (Slika 2.13) načinu spreminjamo tok le na enem navitju naenkrat, medtem ko pri polkoračnem (Slika 2.14) izmenično spreminjamo tok na navitju. Enkrat v eni tuljavi, drugič v obeh tuljavah. Da bi dosegli enako hitrost moramo na motorju s polkoračnim krmiljenjem dovesti dvakratno frekvenco, ki bi jo potrebovali za krmiljenje koračnega motorja v polnokoračnem načinu.

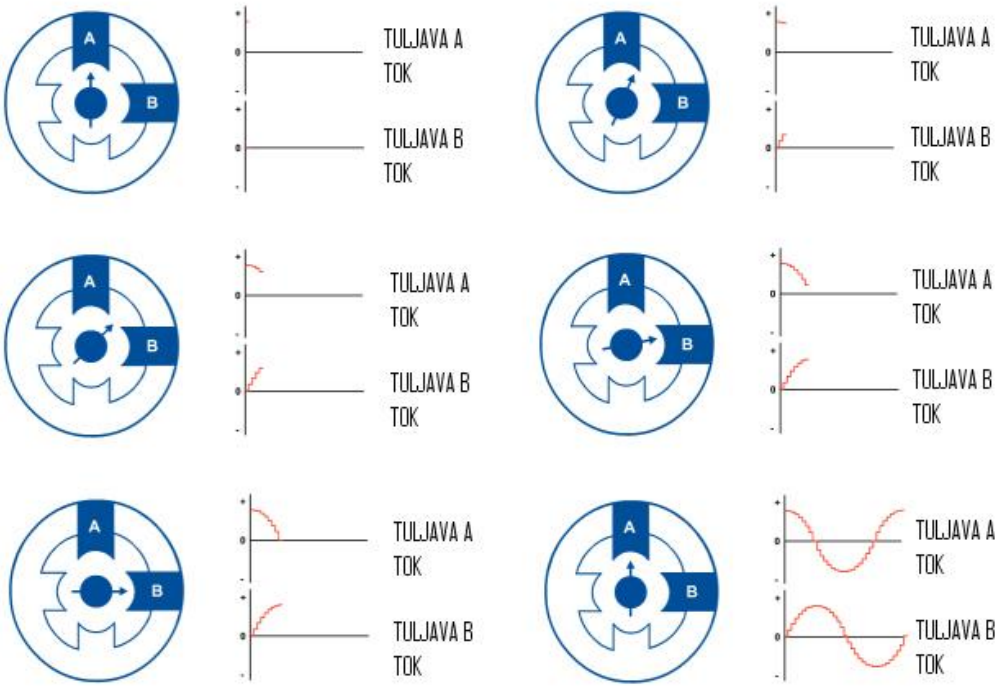


Slika 2.13: Krmiljenje motorja v polnokoračnem načinu [4].



Slika 2.14: Krmiljenje koračnega motorja v načinu polovičnega korakanja(Half Stepping) [4].

Mikrokorakanje je način pri krmiljenja, pri katerem koračni motor teče najbolj gladko. Tok v tuljavah je izmenični oz. sinusni. Potek dovajanja električnega toka v tuljavo je prikazan na sliki 2.15.



Slika 2.15: Krmiljenje koračnega motorja v načinu mikrokorakanja [4]

2.2.2 Izbor koračnega motorja

Glede na velikost obdelovalne mize sem se odločil za koračni motor 23LM-C352-08, proizvajalca Astrosyn/Minebea. Motor pri polnokoračnem načinu krmiljenja potrebuje 200 korakov za en obrat, torej se rotor motorja pri vsakem koraku premakne za kot 1,8°. Navor motorja je 71 Ncm. Motor se lahko priklopi v unipolarnem ali bipolarnem načinu[5].

Slika 2.16: Uporabljen koračni motor

3. Krmiljenje

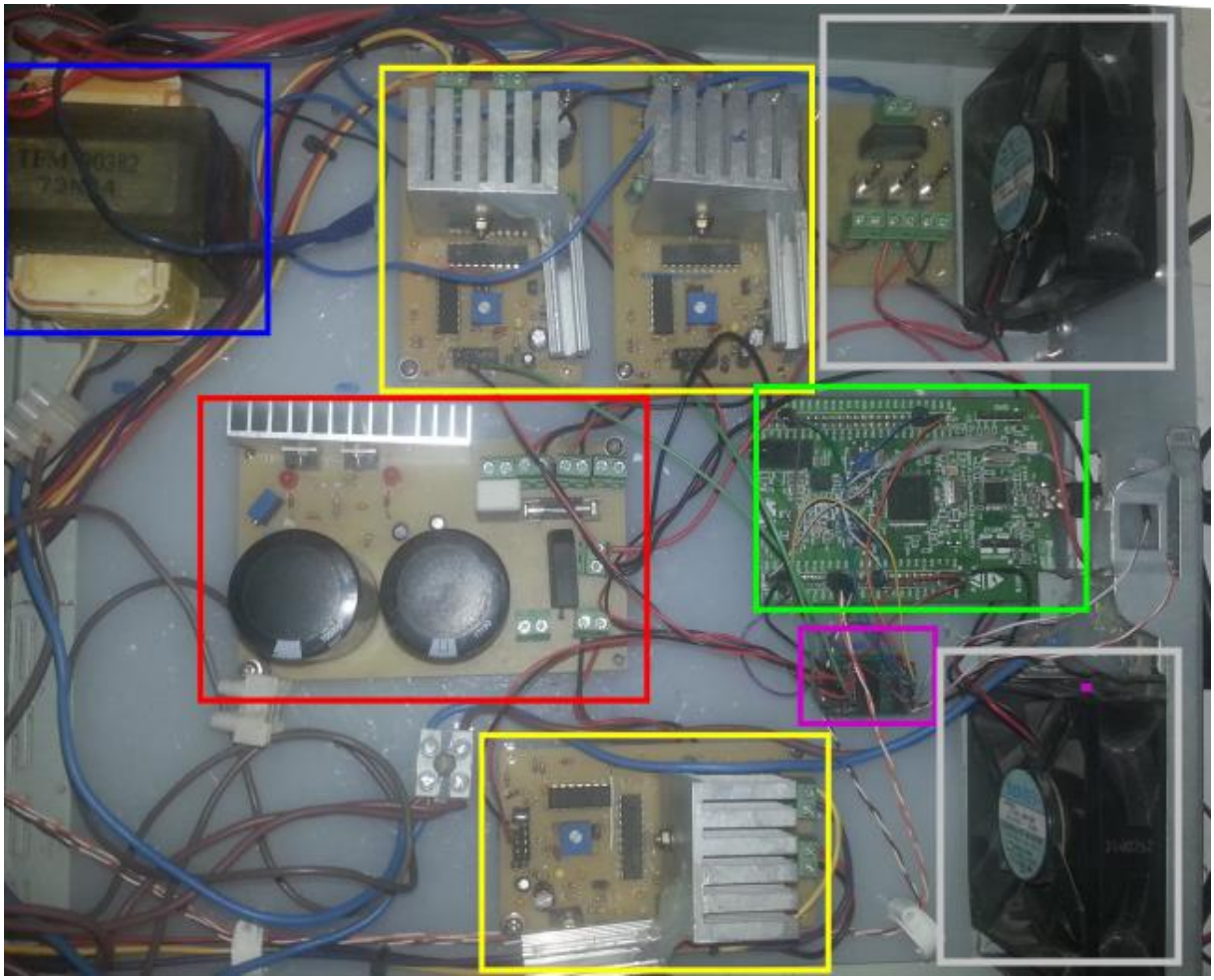
3.1 Krmilna enota

3.1.1 Ohišje krmilne enote

Za ohišje krmilne enote sem uporabil ohišje namiznega računalnika IBM Thinkcentre S50. Zeleno stikalo na ohišju je namenjeno vklopu ventilatorjev in rdeče stikalo vklopu napajanja gonilnikov koračnih motorjev. Iz krmilne enote so izpeljani vsi vodniki, kateri so priklopljeni na koračne motorje.

Slika 3.1: Ohišje krmilne enote

Na dno ohišja sem vstavil pleksi ploščo in nanjo pritrdil posamezne komponente. Ohišje ima na zgornji strani dvižno stranico. V osrednji del sem postavil usmernik, ki je na sliki 3.2 označen z rdečo barvo in vse tri gonilnike za koračne motorje, označene z rumeno barvo. Transformator, označen z modro barvo, skrbi za napajanje sistema z napetostjo. Na ohišje so vgrajeni trije ventilatorji. Dva ohlajata gonilnike za koračne motorje, tretji pa izpihuje vroč zrak iz zaprtega sistema. Razvojni sistem z mikrokrmilnikom STM32F407 je označen z zeleno barvo, z vijolično barvo pa pretvornik napetosti iz 3.3V na 5V.



Slika 3.2: Notranjost krmilne enote

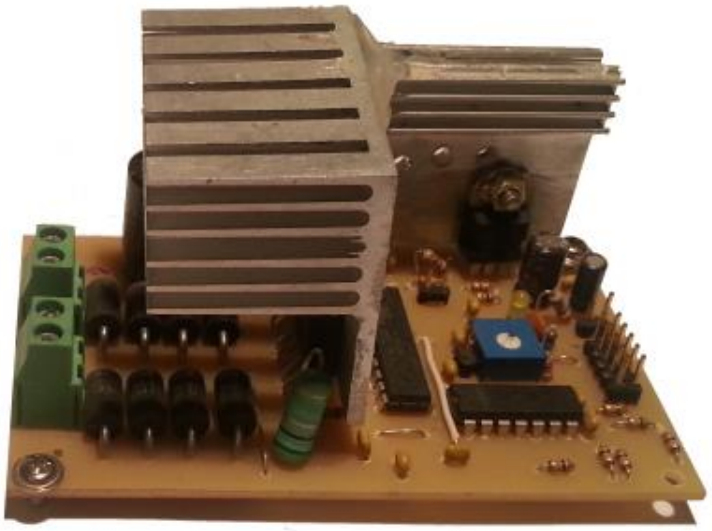
3.1.2 Napajanje krmilne enote

Za napajanje sistema z električno energijo sem uporabil transformator v kombinaciji z usmernikom.
Transformator sestavlja en par vhodnih in dva para izhodnih priključkov. Vhodni priključek je priklopljen na **230V** izmenične električne napetosti. Napetost na prvem izhodu transformatorja znaša **30V** in je uporabljena za napajanje gonilnikov koračnih motorjev. Na drugem izhodu je napetost **12V** in je namenjena za napajanje ventilatorjev. Ker je transformirana napetost izmenična jo moramo preko usmernika spremeniti v enosmerno. Načrt za usmernik sem našel na spletni strani [6].

3.2 Krmiljenje koračnih motorjev

3.2.1 Gonilnik koračnega motorja

Na spletni strani [7] sem našel načrt za izdelavo gonilnika za bipolarni koračni motor (Slika 3.3). To je vezje, imenovano L297-8. Za svoje delovanje uporablja čipa L297 in L298.



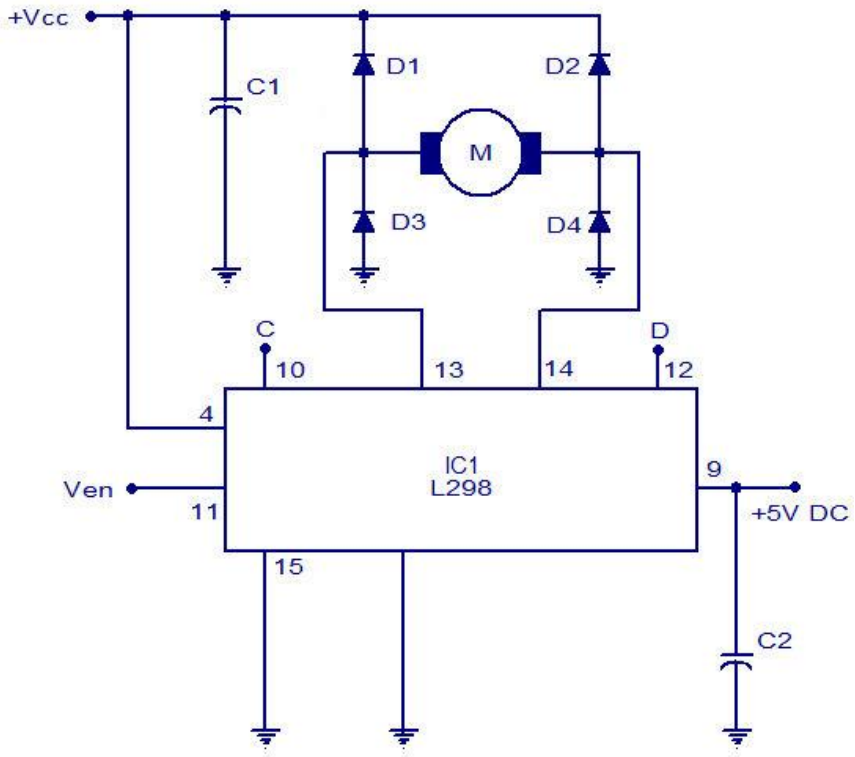
Slika 3.3: Gonilnik koračnega motorja

Čip L297 skrbi za logično krmiljenje rotorja pri koračnem motorju. Vsaka sprememba iz pozitivne urine fronte na negativno premakne rotor za en korak. Hitrost premika rotorja motorja je pogojena z vhodno frekvenco. Višja je frekvenca, hitreje se rotor vrti[8].

Čip L298D je t.i. H – mostiček, ki omogoča da se rotor vrti v smer urinega kazalca oziroma obratno ter omogoča hitro zaustavitev rotorja. Oznaka D pomeni da ima en čip dva mostička. To zadostuje za krmiljenje dveh tuljav oziroma enega koračnega motorja[9].

V kolikor želimo imeti nadzor nad vrtenjem motorja, moramo vklopiti signal V_{en} (slika 3.4). S spreminjanjem vrednosti na pinih 10 (C,INPUT3) in 12 (D,INPUT4) se spreminja tudi vrednost signala na pinih 13 (OUTPUT3) in 14 (OUTPUT4) [7].

V tabeli 3.4 je prikazana funkcija motorja glede na vhodne vrednosti.



Slika 3.4: Vezava čipa L298 in koračnega motorja [10]

Vhod		Funkcija
V_{en}=1	C=1; D=0	Vrtenje CW *(Counter Clockwise)*
	C=0; D=1	Vrtenje CCW *(Counter Clockwise)*
	C=D	Motor se hitro ustavi
V_{en}=0	C=X ; D=X	Motor prosto teče do ustavitve

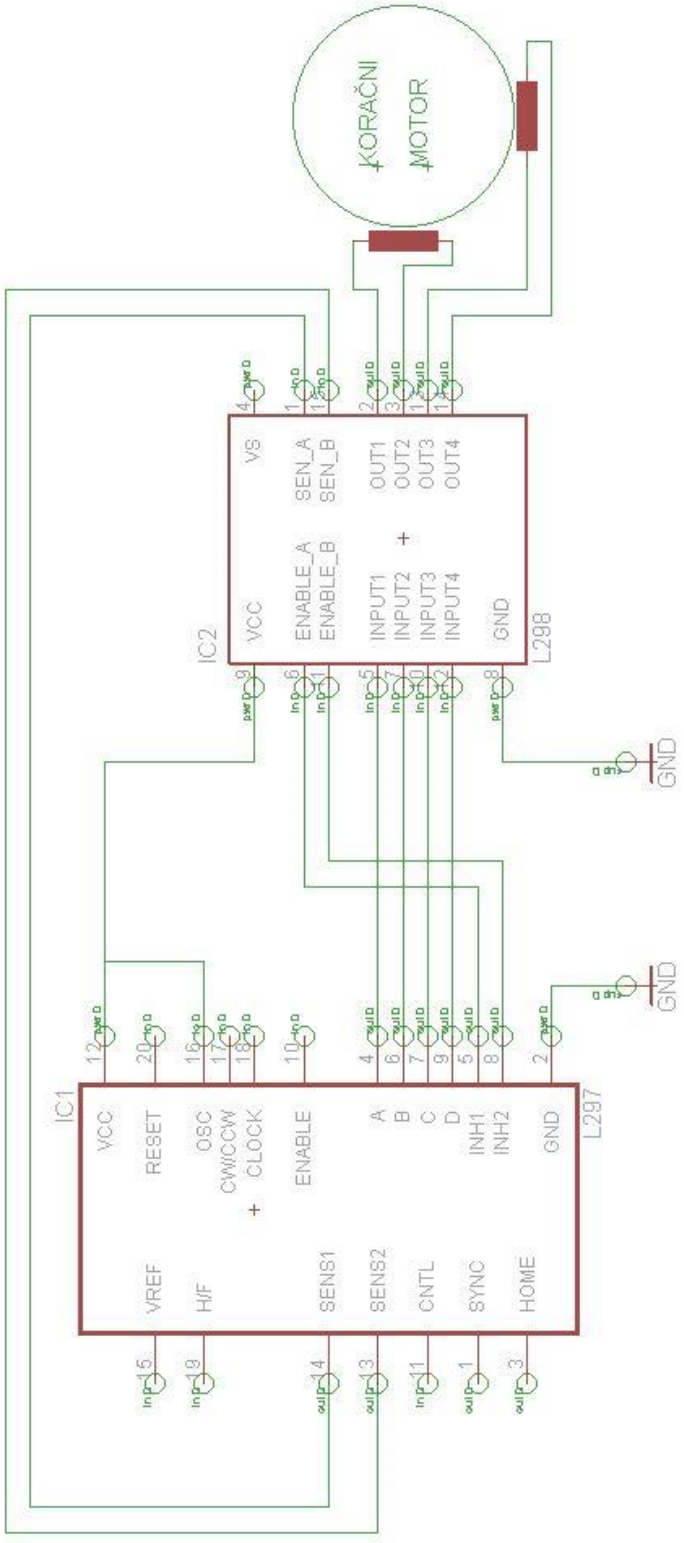
Tabela 3.1: Funkcija H-mostička glede na vhodne vrednosti

Gonilnik koračnega motorja L297-8 ima štiri vhodne priključke (Slika 3.5), ki so preko vodnikov povezani na sistem ARM. Signal **ENABLE** je namenjen vklopu oz. izklopu gonilnika. Ko predvidimo, da motorja dalj časa ne bomo potrebovali, ga je smiselno izklopiti. Tako zmanjšamo možnost, da se bo motor začel pregrevati.

Z vhodnim signalom **STEP** kontroliramo hitrost vrtenja koračnega motorja. Vhod ima vgrajen preprost RC filter. Minimalna širina pulza je 3µs. Maksimalna frekvenca za korak je 40kHz.V praksi večina koračnih motorjev izgubi navor pr 1kHz pri polnem koraku oz. 2kHz polovičnem korakanju.
Vhodna frekvenca na priključku **STEP** je generirana z razvojnim sistemom ARM.

Signal **DIR** določa smer vrtenja koračnega motorja. Če je signal v nizkem stanju, potem se motor vrti v smeri urinega kazalca (CW). V nasprotnem primeru, torej če je **DIR** v visokem stanju, se motor vrti v nasprotni smeri urinega kazalca (CCW) [11].

Slika 3.5: Vhodni signali na gonilniku koračnega motorja [7]

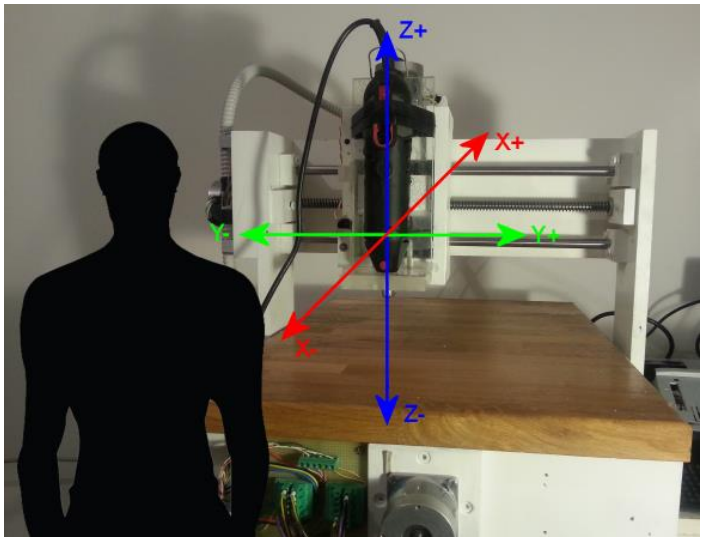


Slika 3.6: Vezava čipov L297 in L298

3.3 Premikanje

3.3.1 Koordinatni prostor

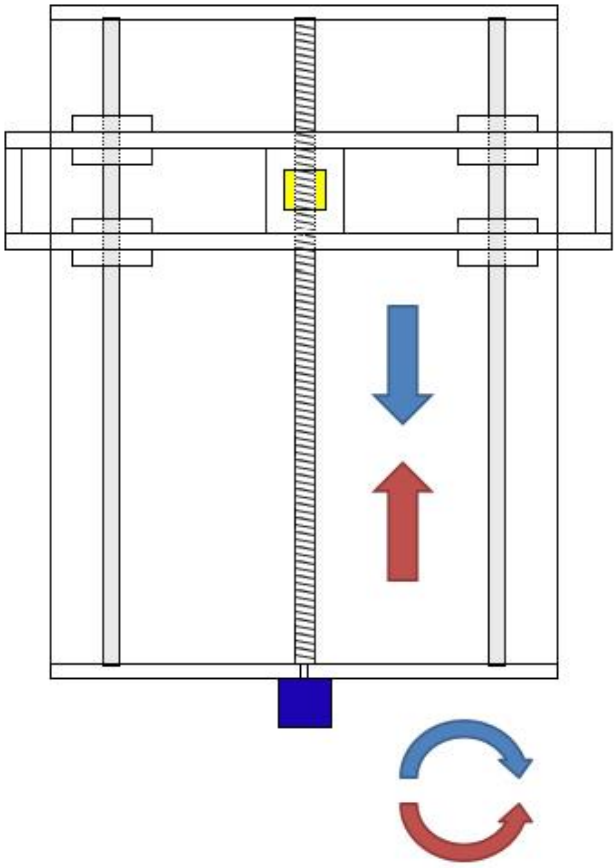
Obdelovalno orodje lahko obišče vse točke v prostoru, omejenem z dimenzijami osi X,Y in Z. S strani operaterja, postavljenega pred stroj se obdelovalni stroj pomika naprej in nazaj v **smeri X**, levo in desno v **smeri Y** ter gor in dol v **smeri Z**. Obdelovalno orodje se lahko v smereh X in Y premika istočasno, z različnimi hitrosti. V smeri Z se premika takrat, ko sta X in Y v mirovanju.



Slika 3.7: Koordinatni prostor

3.3.2 Premik obdelovalnega orodja

Na koračni motor je z reducirnim členom vpeto navojno vreteno, ki gre skozi matico s trapeznim navojem. Pravokotno na navojno vreteno je postavljena pleksi akrilna plošča, ki je na obeh straneh omejena z vodilno osjo. Med vodilno osjo in pleksi akrilno ploščo je vgrajena drsna puša. Ko se motor premika v smeri urinega kazalca se obdelovalno orodje pomika v smeri proti koračnem motorju (Slika 3.8).



Slika 3.8: Premik obdelovalnega orodja v smeri osi X

3.3.3 Obdelovalno orodje

Za obdelovalno orodje (v nadaljevanju orodje) sem uporabil premi brusilnik znamke Skill (Slika 3.9). Število vrtljajev v minuti se nastavlja s potenciometrom in znaša od 15000 do 35000. Na orodju je vpet kronski rezkar, premera 3 mm.

Slika 3.9: Gravirnik znamke Skill, vpet na nosilec

3.3.4 Omejitev premika

V smeri osi Z je premik orodja omejen z dvemi končnimi stikali, kot je prikazano na sliki 3.10. Stikali, označeni z modro barvo, sta nastavljivi po višini. S pozicijo spodnjega stikala nastavimo globino graviranja, s pozicijo zgornjega pa optimiziramo čas pomika orodja iz stanja graviranja v stanje, ko ne želimo gravirati. Stikalo se sproži v trenutku, ko del , na sliki 3.10 označen z rdečo barvo, zadene stikalo. Koda 3.1 prikazuje programsko rešitev (ki se izvaja na mikrokrmilniku STM32F407) za premik orodja gor (ne graviraj) in dol (graviraj) v smeri osi Z.

Slika 3.10: Omejitev premika v smeri osi Z

```
void ne_graviraj(){

  ustavi_x();
  ustavi_y();
  GPIO_SetBits(GPIOC, GPIO_Pin_7);  //določimo smer premika orodja - gor v smeri Z
do
{
resume_z();
}
while (GPIO_ReadInputDataBit(GPIOD,GPIO_Pin_2)!=1);  //ponavljaj, dokler ni zgornje
stikalo aktivno

ustavi_z();
 graviranje=0;

}

void graviraj(){

ustavi_x();
ustavi_y();
GPIO_ResetBits(GPIOC, GPIO_Pin_7); //določimo smer premika orodja - dol v smeri Z
do
{
resume_z();
}
while (GPIO_ReadInputDataBit(GPIOD,GPIO_Pin_4)!=1); //ponavljaj, dokler ni spodnje
stikalo aktivno
ustavi_z();
graviranje=1;
}
```

Koda: 3.1: Programska rešitev procedure za premik orodja

3.3.5 Resolucija naprave CNC

Resolucija naprave CNC nam pove razdaljo premika obdelovalnega orodja pri obratu koračnega motorja za en korak. To je tudi najmanjši možen premik obdelovalnega orodja. Ob obratu navojnega vretena za 360° se obdelovalno orodje premakne za dolžino 4mm. Koračni motor potrebuje 200 korakov za en obrat pri polnokoračnem načinu krmiljenja. V našem primeru koračni motor krmilimo v polkoračnem načinu. Potrebujemo dvakrat več korakov za en obrat, torej 400. Sprememba kota pri enem koraku se iz 1,8° prepolovi in znaša 0,9°.
Izračunano vrednost sem dobil z enačbo (3.1) in znaša **0,01mm** (3.2).

$$R = \frac{\Delta\varphi\ (en\ korak) \times hod\ vretena}{\Delta\varphi\ (en\ obrat)} \qquad (3.1)$$

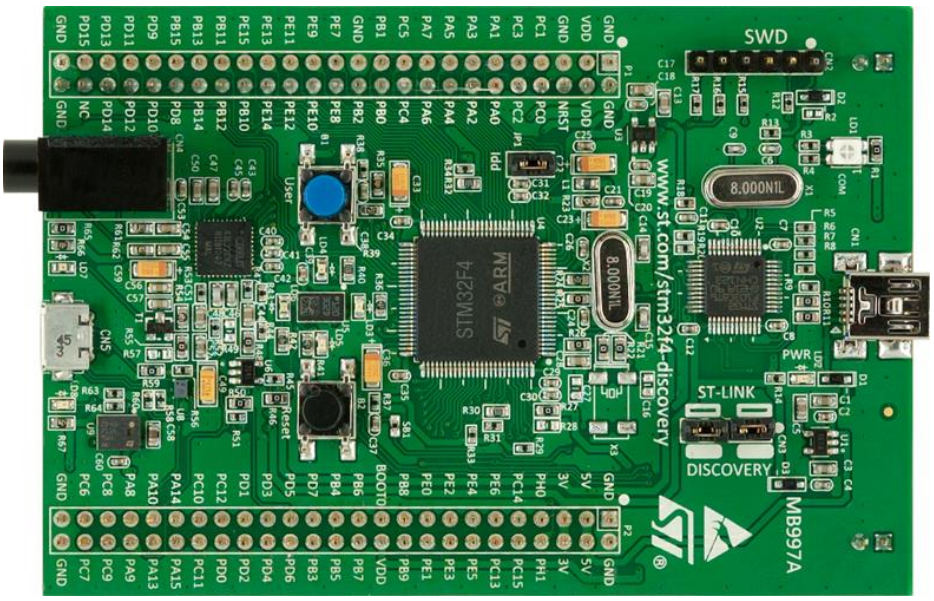
$$R = \frac{0{,}9° \times 4mm}{360°} = \boldsymbol{0{,}01\ mm} \qquad (3.2)$$

3.4 Razvojni sistem STM32F4 DISCOVERY

Razvojni sistem z mikrokrmilnikom STM32F407 (v nadaljevanju sistem ARM) sem uporabil kot vmesni del med uporabniškim vmesnikom in CNC napravo. Ta skrbi za komunikacijo z uporabniškim vmesnikom in krmiljenje koračnih motorjev.

Sistem ima visoko zmogljiv 32-bitni ARM *(Advanced Risc Machine)* procesor s Cortex M4 jedrom. Ta deluje s taktom 168 MHz. Sistem ima na voljo 1MB flash pomnilnika in 192KB RAM-a. Na razvojni plošči je vgrajen senzor gibanja, priključek za mikrofon ali zvočnik, DAC *(Digital Analog Converter)*, štiri vgrajene LED *(Light-Emitting Diode*) diode,… Poleg tega ima kar 84 vhodno-izhodnih priključkov.

Na proizvajalčevi spletni strani je na voljo več programov, ki demonstrirajo delovanje vgrajenih komponent. Od enostavnega prižiganja in ugašanja LED diod na razvojni plošči do premikanja miškinega kursorja na ekranu s pomočjo vgrajenega giroskopa. Programi so mi bili v veliko pomoč pri spoznavanju funkcij razvojne plošče.[12]



Slika 3.11: Razvojna plošča z mikrokrmilnikom STM32F4

3.5 Vmesnik USART

Komunikacija med uporabniškim vmesnikom in sistemom ARM poteka preko serijskega komunikacijskega vmesnika USART. Glavna razlika med sinhronsko in asinhronsko komunikacijo je ta, da se pri sinhronski komunikaciji poleg podatkov pošilja še ura.

Ker novejši računalniki nimajo serijskih vrat, sem moral za serijsko komunikacijo uporabiti **vmesnik PL2302HX** (Slika 3.12)**.** Vmesnik emulira serijski port preko USB vmesnika. Vmesnik je preko signalov Tx, Rx in GND povezan na razvojno ploščo STM32F4 Discovery.

Na sistemu ARM je vmesnik USART definiran na vhodno-izhodnih pinih PB6 (Tx) in PB7(Rx) [12].

Slika 3.12: Vmesnik PL2303HX za serijsko komunikacijo na strani računalnika[13]

Pri inicializaciji USART vmesnika je potrebno nastaviti parametre:

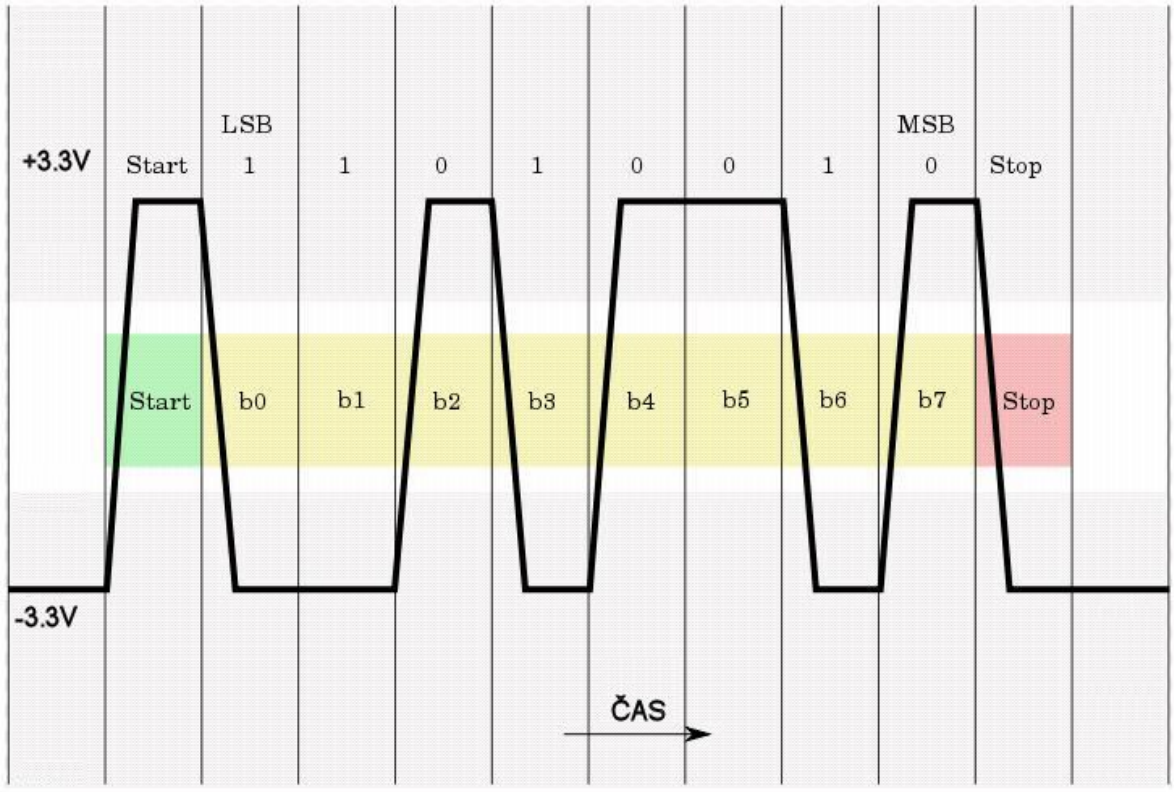
- Baudrate - število prenesenih simbolov ali pulzov na sekundo.
- Frame size - število podatkovnih bitov, ki jih prenašamo v paketu. Nastavimo vrednost 5-8.
- Stop bit- število stop bitov – stop bit je postavljen za MSB. Označuje konec zaporedja podatkovnih bitov. Nastavimo vrednost 1-2.
- Parity –pariteta. Opcijsko nastavimo pariteto kot kontrolo za pravilnost komunikacije. Nastavimo vrednost liha ali soda.

Pri tem morajo biti podatki na obeh napravah identični, saj je v nasprotnem primeru komunikacija neuspešna.

Pošiljatelj prejemniku pošlje zaporedje bitov (Slika 3.13). **Start bit** označuje začetek prenosa, zatem sledi zaporedje podatkovnih bitov v zaporedju od **LSB** *(Least Significant Bit)* do **MSB** *(Most Significant Bit)*. Konec podatkovnih bitov označuje eden ali dva **stop bita**.

BIT ŠT.	1	2	3	4	5	6	7	8	9	10	11
	START BIT	8 PODATKOVNIH BITOV								STOP BIT	
	START	D0	D1	D2	D3	D4	D5	D6	D7	STOP	STOP

Slika 3.13: Zaporedje prenosa bitov pri komunikaciji USART



Slika 3.14: Signali pri prenosu vrednosti 0xD2

V mojem primeru sem dolžino prenesenega niza v obe smeri omejil na dva znaka. S tem sem se izognil dodatnemu preverjanju dolžine prejetega ukaza in njegovi interpretaciji.

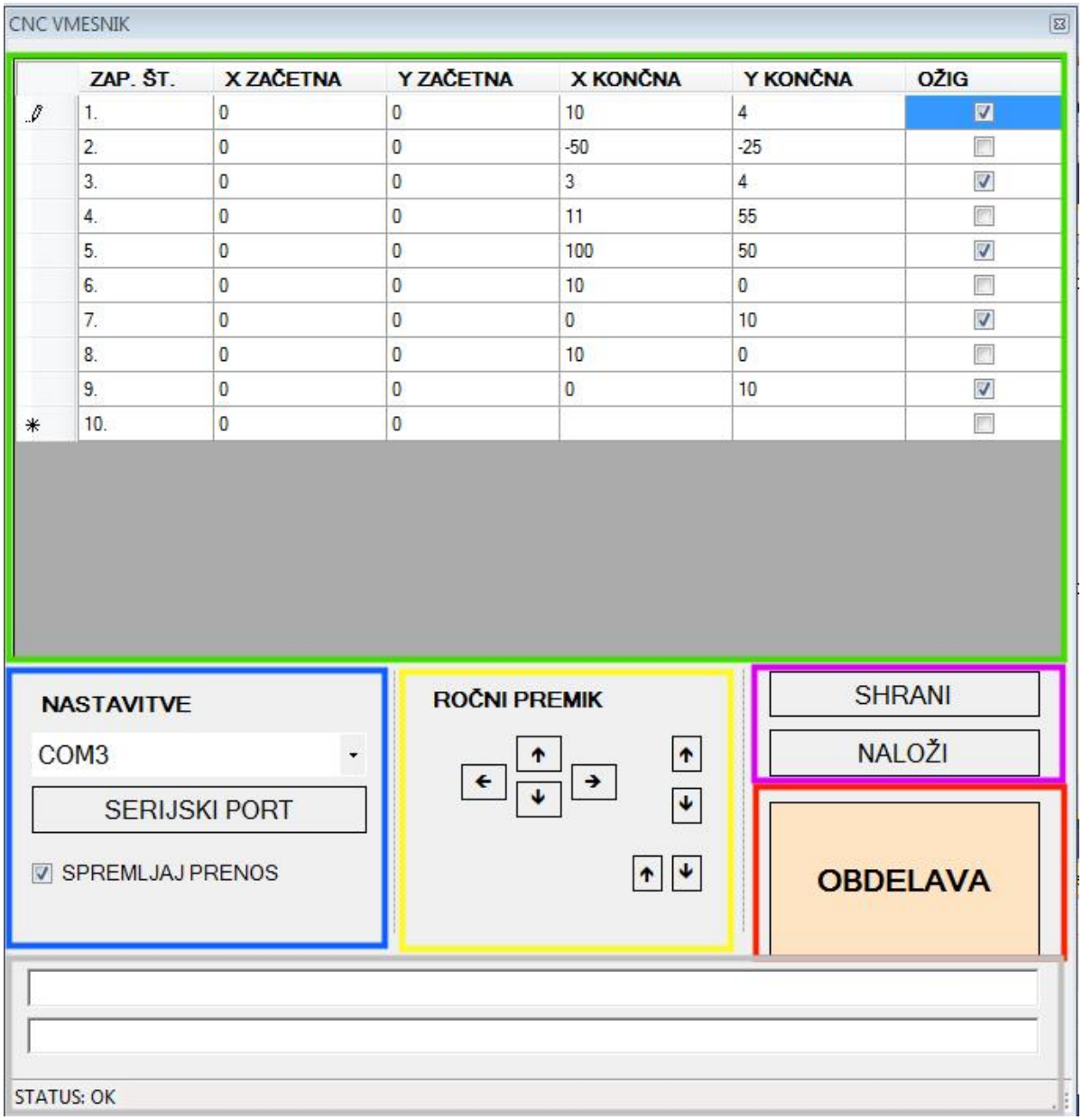
4. Programska oprema

4.1 Uporabniški vmesnik

Uporabniški vmesnik je napisan v razvojnem okolju Microsoft Visual Studio 2010 Professional. Okolje podpira programske jezike VB.NET, Visual C++ in Visual C. Prav tako podpira tudi označevalne jezike XML*(eXtensible Markup Language)*, HTML *(Hyper Text Markup Language)*, Javascript in CSS *(Cascading Style Sheets)*.

Gradnik DataGridView, na sliki 4.1 označen z zeleno barvo, je namenjen vnosu končnih koordinat X in Y. Začetna točka je definirana z vrednostjo X=0 in Y=0, v polji **X_KONČNA** in **Y_KONČNA** pa uporabnik vpiše željene koordinate. Če želimo gravirati, potem je potrebno v polje **OŽIG** nastaviti kljukico. V nasprotnem primeru se izvede le premik orodja, vendar brez graviranja.

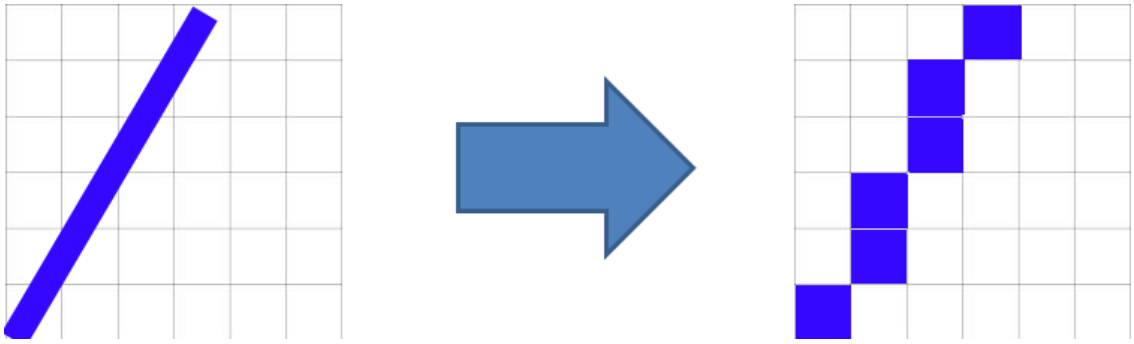
Polje **NASTAVITVE** je označeno z modro barvo. V tem polju nastavimo serijska vrata, preko katerih želimo komunicirati z sistemom ARM. Z gradnikom **SPREMLJAJ PRENOS** vključimo ali izključimo tekstovno polje, namenjeno izpisu prenesenih vrednosti za tabeli **TABELA_X** in **TABELA_Y** v sistem ARM. Te vrednosti se izpisujejo v gradnikih, označenih s sivo barvo.
Z rumeno barvo je označeno polje **ROČNI PREMIK**. V kolikor želimo ročno nastavljati pozicijo orodja, lahko to počnemo s smernimi tipkami.
Uporabniški vmesnik omogoča shranitev koordinat. Te se zapišejo v tekstovne datoteke. Ob pritisku gumba naloži, se vrednosti iz tekstovne datoteke prenesejo v tabelo.
Gumb **OBDELAVA** je označen z rdečo barvo. Do trenutka, dokler v vmesniku ne določimo vrat za serijsko komunikacijo in dokler se izvaja obdelava, je gumb onemogočen.



Slika 4.1: Uporabniški vmesnik

4.2 Bresenhamov algoritem za risanje linije

Bresenhamov algoritem se v računalniški grafiki uporablja za risanje daljic na rastrskem zaslonu. Če bi hoteli narisati točno linijo, potem bi potrebovali zaslon z velikostjo neskončno točk, kar v praksi ni možno. Ker smo pri risanju daljice na zaslon omejeni s končnim številom točk, je potrebno določiti najboljši približek(Slika 4.2). Bresenhamov algoritem rasterizira črte s pomočjo inkrementalne celoštevilske aritmetike[15].



Slika 4.2: Pretvorba linije v rastrski prostor

Predpostavimo da rišemo daljico med dvema točkama (x_0, y_0) in (x_1, y_1) v 1. oktantu.

Najprej izračunamo smerni koeficient daljice (4.1). Dy je razlika med končno in začetno točko, glede na y os (4.2) in dx je razlika med končno iz začetno točko glede na os X(4.3). Naklon daljice je razmerje med dy in dx (4.4)

$$m = \frac{dy}{dx} \qquad (4.1)$$

$$dx = y_k - y_z \qquad (4.2)$$

$$dy = x_k - x_z \qquad (4.3)$$

$$m = \frac{yk - yz}{xk - xz} \qquad (4.4)$$

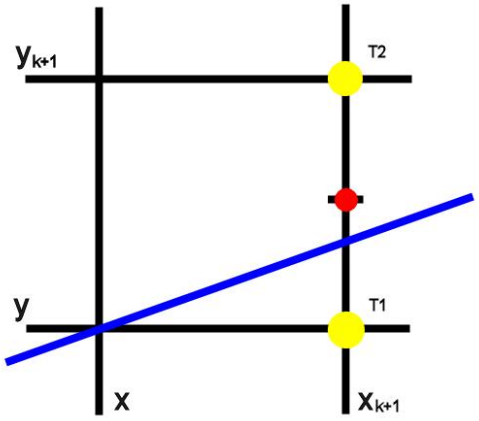
Nato se pomikamo v smeri osi X s korakom 1. Pri tem imamo možnosti da narišemo točko

- T1 pri koordinatah (x_k+1, y)

ali

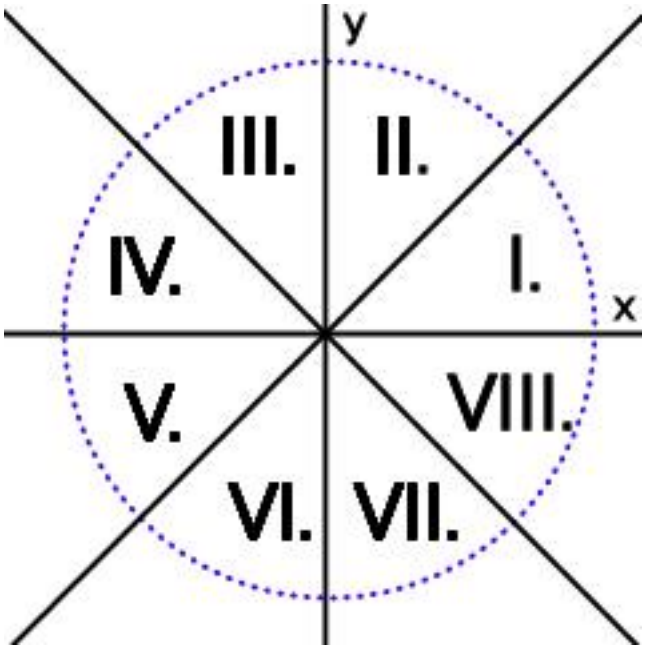
- T2 pri koordinatah (x_k+1, y_k+1)

Pri tem izberemo točko, ki je bližja točki midpoint, na sliki 4.3 označena z rdečo piko.



Slika 4.3: Izbira točke po Bresenhamovi formuli

Prikazan primer je za prvi oktant. Določitev oktanta je odvisna od smernega koeficienta in vrednosti začetne in končne točke. Tabela 4.4 prikazuje določitev oktanta glede na vrednost smernega koeficienta m in začetne ter končne točke.



Slika 4.4: Koordinatni prostor, razdeljen na oktante

Oktant	Vrednost m	Vrednost začetne in končne točke
I.	$0 \leq m \leq 1$	$x_z < x_k$
II.	$1 \leq m \leq \infty$	$y_z<y_k$
III.	$-1 > m > -\infty$	$y_z<y_k$
IV.	$0 \geq m \geq -1$	$x_k < x_z$
V.	$0 \leq m \leq 1$	$x_k < x_z$
VI.	$1 < m < \infty$	$y_k<y_z$
VII.	$-1 > m > -\infty$	$y_k<y_z$
VIII.	$0 > m \geq -1$	$x_z < x_k$

Tabela 4.4: Določitev oktanta

4.3 Vrednost 0DL

Profesionalne naprave CNC za svoje delovanje uporabljajo t.i. G - kodo [16].
Namesto G - kode sem naredil svoj koncept. Vsaka vrstica predstavlja svojo linijo. Izhaja iz dveh parov točk:

- x_začetna
- y_začetna

in

- x_končna
- y_končna

Točki **x_začetna** in **y_začetna** imata privzeto vrednost 0 in se ju ne more spremeniti.
Uporabnik mora vpisati točki **x_končna** in **y_končna**.

Ko se po Bresenhamovem algoritmu izračunajo vrednosti za X in Y, se te zapišejo v tabelo. Namesto da bi preko serijske komunikacije prenašali vse koordinate, naredimo novo tabelo v **formatu 0DL** *(Nič,desno,levo).* Pri tem gremo čez tabelo in primerjamo sosednja elementa.

Pri tem velja:

- če sta sosednja elementa enaka, zapišemo 0
- če je naslednji element večji od trenutnega, zapišemo D
- če je naslednji element je manjši od trenutnega, zapišemo L

	TABELA_X	TABELA_Y		TABELA_X_0DL	TABELA_Y_0DL
Pozicija 1	0	0		0	0
Pozicija 2	1	1		D	D
Pozicija 3	2	2		D	D
Pozicija 4	3	3		D	D
Pozicija 5	4	4		D	D

Tabela 4.2: Številske vrednosti konvertiramo v vrednosti 0DL

V primeru da imamo končne točke v negativnih vrednostih, se izvede obdelava za pozitivne vrednosti, nato pa na podlagi predznakov, ki so opisani v tabeli Tabela 4.3 naredi ustrezno zamenjavo.

X_KONČNA	Y_KONČNA	FUNKCIJA
+	+	Ne spreminjaj 0DL
-	+	zamenjaj D in L v tabeli tabela_X
+	-	zamenjaj D in L v tabeli tabela_Y
-	-	Zamenjaj D in L v tabelah tabela_x in tabela_y

Tabela 4.3: Sprememba vrednosti 0DL glede na vrednosti vhodnih koordinat

Koda 4.1 prikazuje proceduro **toggle_x.** Procedura se sprehodi čez tabelo in zamenja vrednosti D in L. Vrednosti 0 ne spreminja.

```
'Zamenjaj vrednosti D in L v tabeli Tabela_x

    Sub toggle_x()

        Dim stevec As Integer

        For stevec = 0 To tabela_x_0DL.count - 1 ' sprehod čez tabelo

            If tabela_x_0DL(stevec) = "D" Then    'zamenjaj »D« z »L«
                tabela_x_0DL(stevec) = "L"



            ElseIf tabela_x_0DL(stevec) = "L" Then ' zamenjaj »L« z »D«
                tabela_x_0DL(stevec) = "D"


            ElseIf tabela_x_0DL(stevec) = "0" Then ' »0« pusti pri miru
                tabela_x_0DL(stevec) = "0"


            Else : tabela_x_0DL(stevec) = "E"    'napaka.vrednost ni »0«, »D«
ali »L«
```

```
            End If
        Next
```

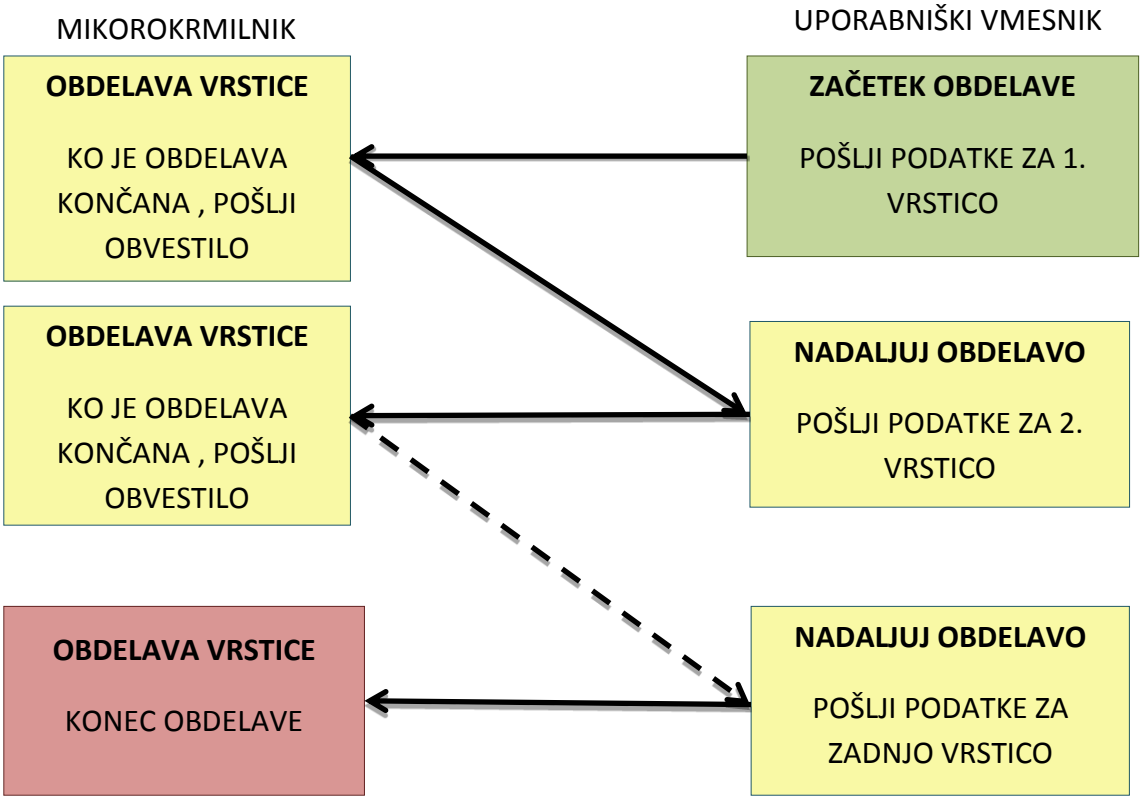
Koda 4.1: Zamenjava vrednosti D in L v tabeli Tabela_x

4.4 Obdelava

4.4.1 Začetek obdelave

V trenutku, ko je v uporabniškem vmesniku pritisnjen gumb **OBDELAVA**, se v sistem ARM prenesejo izračunane vrednosti za prvo vrstico. Te vrednosti so v **formatu 0DL.**
Ko sistem ARM konča z izvedbo prve vrstice in uporabniškemu vmesniku pošlje ukaz da je zaključil, mu ta ponovno pripravi vrednosti v formatu 0DL za naslednjo vrstico. Ta postopek se ponavlja do zadnje vrstice. Med obdelavo se uporabniški vmesnik zaklene. S tem preprečimo da bi pomotoma še enkrat zagnali obdelavo in tako zmotili obdelovalni proces.

Slika 4.5 prikazuje diagram poteka med uporabniškim vmesnikom in sistemom ARM.



Slika 4.5: Diagram poteka obdelave

4.4.2 Procedura DodajXY

Procedura **DodajXY** na podlagi prejetega niza izvede določeno akcijo. Za prejem niza je na strani sistema ARM prekinitveno-servisna procedura USART1_IRQHandler (Koda 4.2).Procedura prestreza znake, poslane s strani uporabniškega vmesnika. Prejete znake sestavi v dvoznakovni niz in jih uporabi kot vhodni parameter za proceduro **DodajXY** (Koda 4.3). Nabor možnih vrednosti za proceduro DodajXY je opisan v tabeli 4.4.

```
void USART1_IRQHandler(void){

	//  Preveri USART1 prekinitveno zastavico
	if( USART_GetITStatus(USART1, USART_IT_RXNE) ){

		static uint8_t cnt = 0; // števec za dolžino prejete besede
		char t = USART1->DR; // znak iz USART1 shrani v t


		if( (t != '\n') && (cnt < MAX_STRLEN) ){
			received_string[cnt] = t; //če je vrednost manjša od 2, potem
dodaj črko k nizu

			    cnt++;




		}
		else{ //  resetiraj števec, interpretiraj prejeti niz
			cnt = 0;
			DodajXY(received_string);


		  }
	}
}
```

Koda 4.2: Sprejem dvoznakovnega niza na strani sistema ARM

Prejeti niz	Dejanje
NN	Avtomatski dvig orodja (ne graviraj)
TT	Avtomatski spust orodja (graviraj)
ZG	Ročni dvig orodja
ZD	Ročni spust orodja
DX	Ročni pomik orodja desno v smeri X
LX	Ročni pomik orodja levo v smeri X
DY	Ročni pomik orodja desno v smeri Y
LY	Ročni pomik orodja levo v smeri Y
X {0,D,L}	V tabelo tabela_x doda 0,D ali L
Y{0,D,L}	V tabelo tabela_y doda 0,D ali L
GG	Zažene proceduro obdelava, ki je opisana v točki 4.4.3

Tabela 4.4: Seznam ukazov v sistemu ARM

```
void DodajXY(volatile char ukaz[])
{


  if ((ukaz[0]=='F')&&(ukaz[1]=='F'))   //ustavi vse motorje
     {

TIM_Cmd(TIM3, DISABLE);
TIM_Cmd(TIM4, DISABLE);
TIM_Cmd(TIM5, DISABLE);

     }


  if ((ukaz[0]=='N')&&(ukaz[1]=='N'))  //dvig orodja-ne graviraj
     {

      ne_graviraj();

     }

    if ((ukaz[0]=='T')&&(ukaz[1]=='T')) //spust orodja - graviraj
     {

       graviraj();

     }


  if ((ukaz[0]=='Z')&&(ukaz[1]=='G')) //ročni premik orodja - gor v smeri Z
     {

       rocni_premik_z_gor();

     }

  if ((ukaz[0]=='Z')&&(ukaz[1]=='D')) //ročni premik orodja - dol v smeri Z
     {
```

```
      rocni_premik_z_dol();

    }


  if ((ukaz[0]=='D')&&(ukaz[1]=='X')) //ročni premik orodja - desno v smeri X
    {

      rocni_premik_x_desno();

    }
    if ((ukaz[0]=='L')&&(ukaz[1]=='X')) //ročni premik orodja - levo v smeri X
    {

      rocni_premik_x_levo();

    }

    if ((ukaz[0]=='D')&&(ukaz[1]=='Y')) //ročni premik orodja - desno v smeri Y
    {

      rocni_premik_y_desno();

    }
    if ((ukaz[0]=='L')&&(ukaz[1]=='Y')) //ročni premik orodja - levo v smeri Y
    {

      rocni_premik_y_levo();

    }




    if (ukaz[0]=='X')
    {


      tabela_x=realloc(tabela_x,2+2*st_el_tab_x*sizeof(char)); //dinamično alociranje
velikosti tabele
      tabela_x[st_el_tab_x]=ukaz[1];  //dodajanje vrednosti 0,D ali L v tabel
tabela_x
      st_el_tab_x++;

    }

    if (ukaz[0]=='Y')
    {


      tabela_y=realloc(tabela_y,2+2*st_el_tab_y*sizeof(char)); //dinamično alociranje
velikosti tabele
      tabela_y[st_el_tab_y]=ukaz[1];  //dodajanje vrednosti 0,D ali L v tabel
tabela_y
       st_el_tab_y++;

    }

       if ((ukaz[0]=='G')||(ukaz[1]=='G'))
    {
```

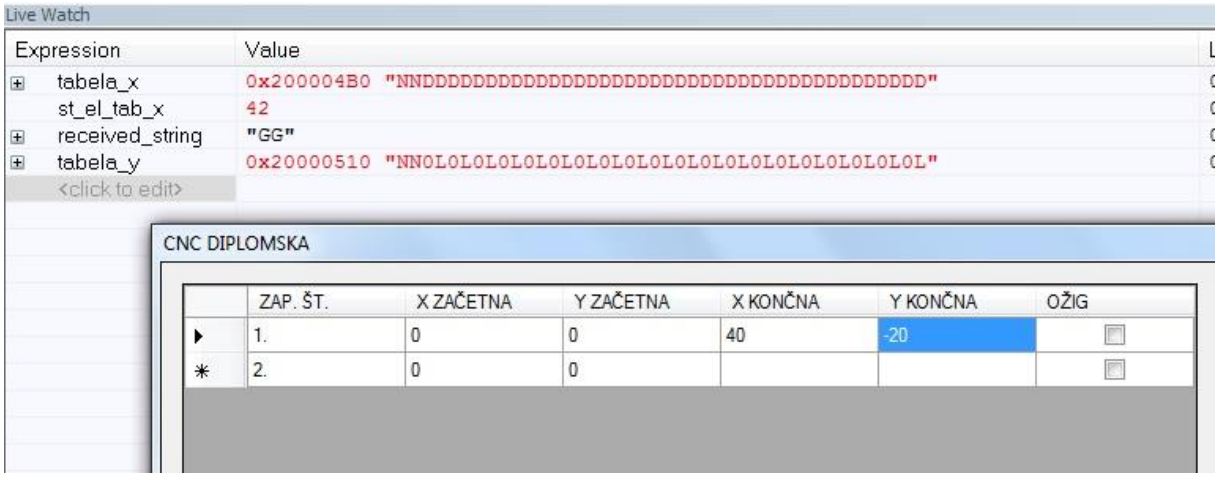
```
   obdelava();
   stevec_x=0;              //izprazni tabeli tabela_x in tabela_y
   stevec_y=0;
   st_el_tab_x=0;
   st_el_tab_y=0;
   USART_puts(USART1, "V"); //uporabniškemu vmesniku pošljemo obvestilo o končani
obdelavi

     }


}
```

Koda 4.3: Programska rešitev procedure DodajXY

Razvojno okolje IAR Embedded Workbench IDE vključuje funkcijo Live Watch (slika 4.6) , ki omogoča spremljanje vrednosti spremenljivk v času izvajanja programa. Z omenjeno funkcijo sem spremljal pravilnost prenosa iz uporabniškega vmesnika v sistem ARM. Na sliki so prikazane vrednosti za premik (0,0)→(40,-20)



Slika 4.6: Vrednosti v formatu 0DL, prenesene v pomnilnik sistema ARM.

4.4.3 Procedura Obdelava

V trenutku, ko mikrokrmilnik prejme niz **»GG«**, kar pomeni obdelavo vrstice, se sprehodi čez tabeli **Tabela_x** in **Tabela_y**, shranjeni v pomnilniku sistema ARM in potem na podlagi vrednosti naredi določeno akcijo. Nabor akcij opisuje tabela 4.5.

Vrednost	Akcija
N	Avtomatski dvig orodja
T	Avtomatski spust orodja
0	Orodja ne premikaj
D	Orodje premakni v desno
L	Orodje premakni v levo

Tabela 4.5: Dejanje orodja glede na vrednosti v tabeli

Ko je sprehod čez tabelo končan, se tabela izprazni. Ob izvedbi novega ukaza se v tabelo vpišejo nove vrednosti.

```
void  obdelava(){     //začni obdelavo vrstice



TIM_Cmd(TIM4, ENABLE);  //vklop motorja x
TIM_Cmd(TIM5, ENABLE); //vklop motorja y

vklop_x();



   for (int i=0;i<st_el_tab_x;i++)            //sprehod čez tabelo
    {

      if ((tabela_x[i]=='N')||(tabela_x[i]=='T'))  //primer da imamo znak za dvig ali
spust orodja
      {
         if ((tabela_x[i]=='N') && (graviranje==1))
           ne_graviraj();    //dvig orodja

         if ((tabela_x[i]=='T') && (graviranje==0))
           graviraj();  //spust orodja


      }
        else
        {

          ustavi_x();
          ustavi_y();

             if (tabela_x[i]=='0')   //ustavi motor x
         {
          ustavi_x();
         }

        if (tabela_y[i]=='0')  //ustavi motor y
         {
         ustavi_y();
         }


        if (tabela_x[i]=='D')  //vrtenje motorja x v desno
        {
          resume_x();
```

```
          GPIO_ResetBits(GPIOD, GPIO_Pin_13);
        }

      if (tabela_x[i]=='L') //vrtenje motorja x v levo
         {
         resume_x();
         GPIO_SetBits(GPIOD, GPIO_Pin_13);
         }


        if (tabela_y[i]=='D')  //vrtenje motorja y v desno
        {
          resume_y();
          GPIO_ResetBits(GPIOA, GPIO_Pin_1);
        }

      if (tabela_y[i]=='L') //vrtenje motorja y v levo
         {
         resume_y();
         GPIO_SetBits(GPIOA, GPIO_Pin_1);
         }


               if ((tabela_x[i]=='N')&&(tabela_y[i]=='N'))
        {

         ustavi_x();
          ustavi_y();


        }

        Delay(0X001FFFFF);   // Glavna zakasnitev


    }



    }


  ustavi_x();
  ustavi_y();
  ustavi_z();

  free(tabela_x);
  free(tabela_y);
   tabela_x = (char*) malloc (2);
  tabela_y =(char*) malloc (2);


}
//konec obdelave
```

Koda: 4.4: Programska rešitev procedure Obdelava

5. Ugotovitve

Material, na katerem sem izvedel graviranje, je stirodur plošča, proizvajalca Fibran. Stirodur je v primerjavi z lesom mehkejši, kar pomeni lažje graviranje. Ko sem poskusil gravirati na les, se je po določenem času rezkar segrel do te mere, da je od vročine začel žareti. V kolikor bi želel izvesti daljšo gravuro na les, bi moral kupiti profesionalni namenski rezkar.

Za preizkus natančnosti sem izvedel graviranje napisa »I♥FRI«. Posamezen znak sem prenesel na koordinatno mrežo (Slika 7.3) in izpisal vse linearne premike (tabela 7.2). Dobljene vrednosti sem vpisal v uporabniški vmesnik in pognal obdelavo. Rezultat je prikazan na sliki 4.7.

Kvaliteta graviranja je odvisna od vrednosti števca v proceduri **obdelava**. Čim manjša je vrednost, tem natančnejša je obdelava, vendar za predstavitev linije potrebujemo več podatkov. Optimalno vrednost sem dobil s poskušanjem in znaša **(0X001FFFFF).**

Slika 4.7: Rezultat graviranja

6. Zaključek

Pri samem delu sem spoznal kako pomembna je faza načrtovanja. Če kakšna komponenta ni bila dobro premišljeno načrtovana, so se pojavile težave in sem potem pri odpravljanju le teh porabil veliko časa. Primer tega je gonilnik koračnega motorja, ki je zasnovan na 5V TTL logiki, razvojna ploščica ARM pa na 3.3V TTL logiki. Za uspešno delovanje sem moral vgraditi dodatni **čip SN74LVC4245A** [17], ki pretvori napetostni nivo iz 3,3V na 5V.

V začetku sem nameraval krmiljenje izvesti preko vrat LPT *(Line Print Terminal)*. Kasneje se je izkazalo, da je maksimalna frekvenca, ki jo lahko generiramo preko vrat LPT, premajhna.

Vsekakor bi se z dražjimi komponentami natančnost še povečala. Denimo da bi medeninasto matico s trapeznim navojem, zamenjal z matico, katera ima v notranjem delu kroglice. Te kroglice zmanjšajo trenje med navojem in matico. Tako premikanje naprave deluje lahkotneje.

Pri komunikaciji bi lahko količino prenosa podatkov preko USART-a zmanjšal, da bi vmesnik poslal koordinate sistemu ARM in bi potem ta izračunal koordinate in izvedel obdelavo.

Risanje okroglin je zahtevalo veliko število manjših linearnih premikov, kar je bilo časovno zamudno. Za natančno risanje krivulj bi bilo potrebno implementirati Bresenhamov algoritem za risanje krivulj.

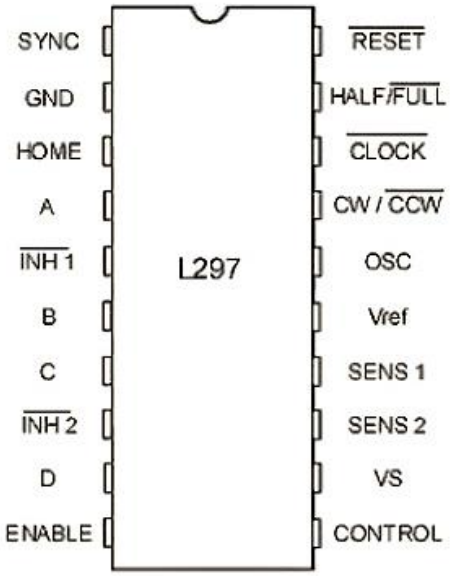
Kljub nekaterim pomanjkljivostim menim, da sem glede na nizek finančni vložek uspel narediti gravirno napravo CNC, katera ima dovolj veliko preciznost za domačo uporabo.

Vesel sem, da sem tekom izdelave diplomske naloge pridobil veliko izkušenj in praktičnih znanj, za katere verjamem, da mi bodo služili pri prihodnjem ustvarjanju.

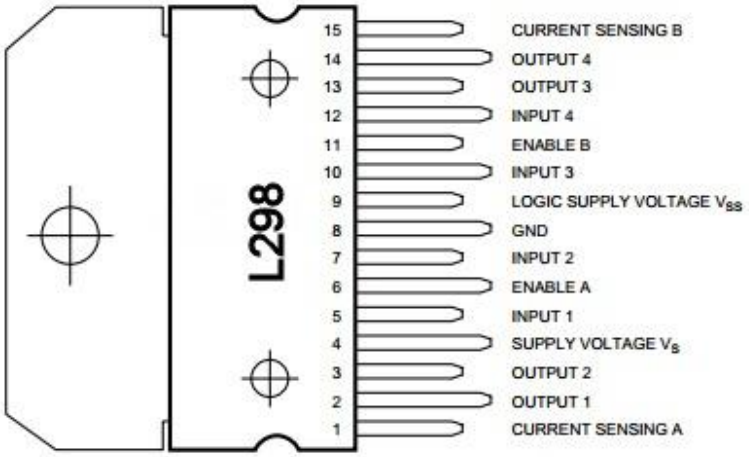
7. Priloge

Vhodnoizhodni priključki			
Port	Pin	Smer	Opis
GPIOD	GPIO_Pin_1	GPIO_Mode_OUT	stikalo graviranja izhod
GPIOD	GPIO_Pin_2	GPIO_Mode_IN	stikalo graviranja vhod
GPIOD	GPIO_Pin_3	GPIO_Mode_OUT	stikalo graviranja izhod
GPIOD	GPIO_Pin_4	GPIO_Mode_IN	stikalo graviranja vhod
GPIOC	GPIO_Pin_6;	GPIO_Mode_AF	timer3 krmiljenje po osi Z
GPIOC	GPIO_Pin_7	GPIO_Mode_OUT	smer po osi Z
GPIOD	GPIO_Pin_12	GPIO_Mode_AF	timer4 za krmiljenje po osi X
GPIOD	GPIO_Pin_13	GPIO_Mode_OUT	smer po osi X
GPIOD	GPIO_Pin_14	GPIO_Mode_OUT	vklop/izklop motora X
GPIOD	GPIO_Pin_15	GPIO_Mode_OUT	vklop/izklop motorja Y
GPIOA	GPIO_Pin_0	GPIO_Mode_AF	timer5 za krmiljenje po osi Y
GPIOA	GPIO_Pin_1	GPIO_Mode_OUT	smer po osi Y
GPIOB	GPIO_Pin_6;	GPIO_AF_USART1	USART Rx
GPIOB	GPIO_Pin_7	GPIO_AF_USART1	USART Tx

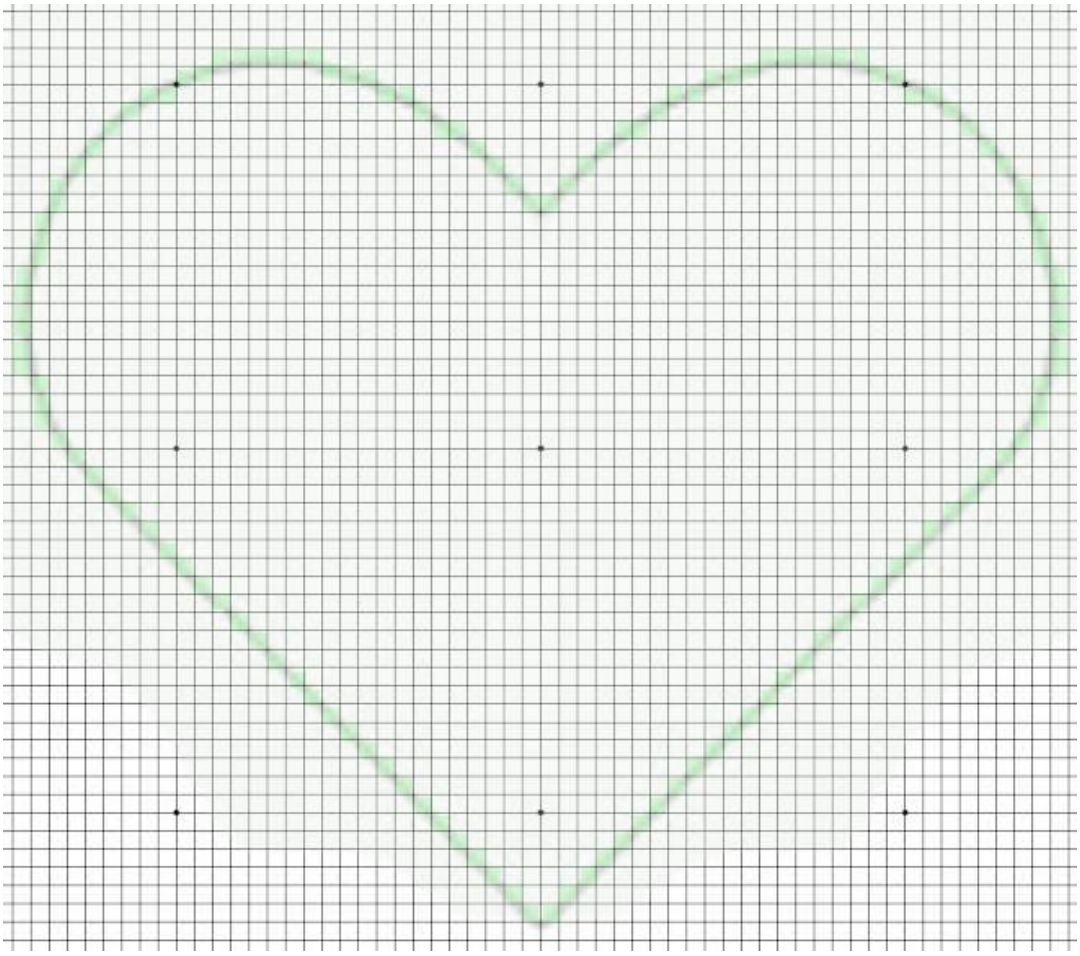
Tabela 7.1: Seznam uporabljenih pinov v sistemu ARM



Slika 7.1: Razpored pinov na čipu L297



Slika 7.2: Razpored pinov na čipu L298

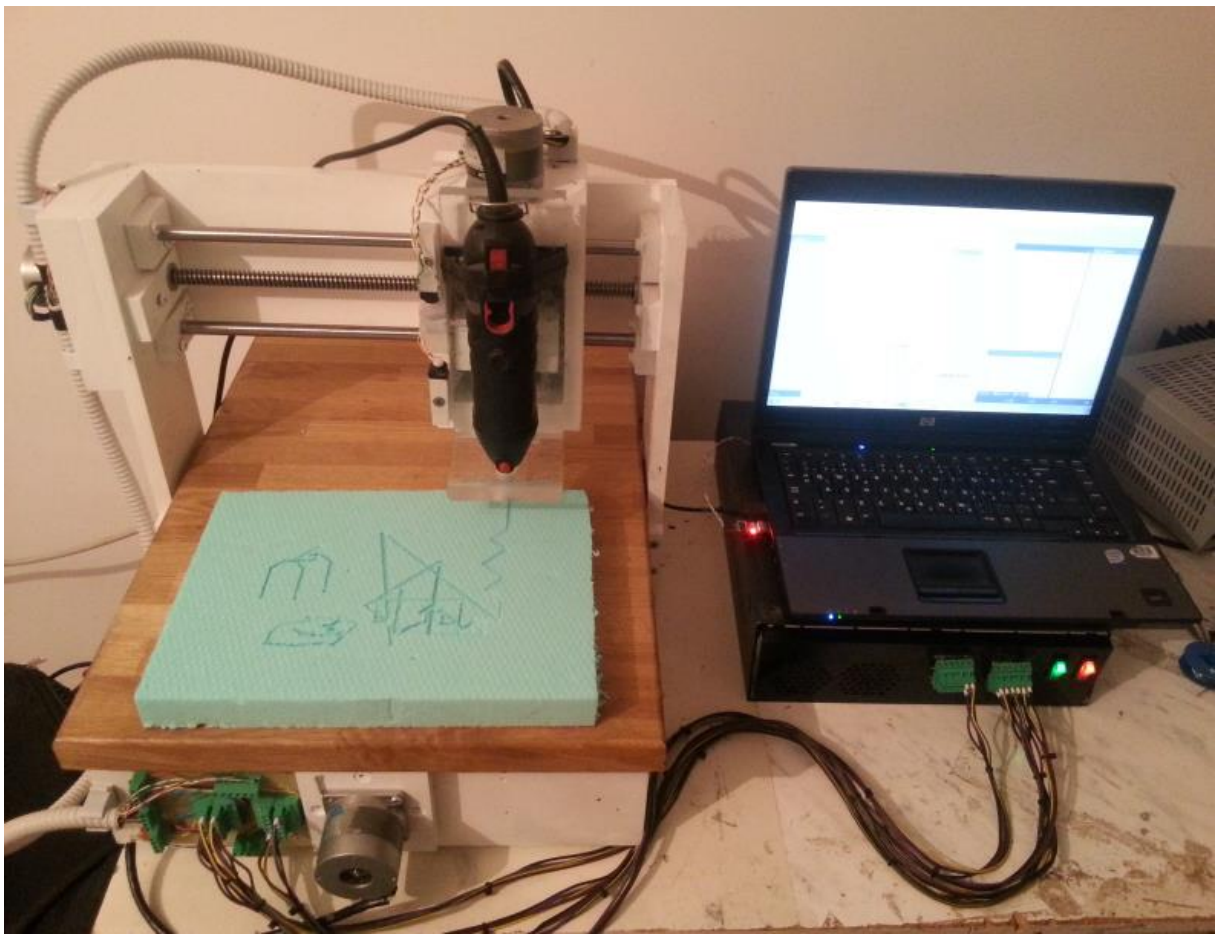


Slika 7.3: Silhueta srca

X_ZAČETNA	Y_ZAČETNA	X_KONČNA	Y_KONČNA	GRAVIRANJE
0	0	10	-10	DA
0	0	0	-2	DA
0	0	6	-6	DA
0	0	0	-2	DA
0	0	2	-2	DA
0	0	0	-4	DA
0	0	2	-2	DA
0	0	0	-10	DA
0	0	-2	-2	DA
0	0	0	-2	DA
0	0	-2	2	DA
0	0	0	-2	DA
0	0	-10	-10	DA
0	0	-2	0	DA
0	0	-2	2	DA

0	0	-4	0	DA
0	0	-2	-2	DA
0	0	-10	0	DA
0	0	-2	2	DA
0	0	-4	0	DA
0	0	-54	54	DA
0	0	54	54	DA
0	0	4	0	DA
0	0	2	2	DA
0	0	10	0	DA
0	0	2	-2	DA
0	0	4	0	DA
0	0	2	-2	DA
0	0	2	0	DA
0	0	10	-10	DA
0	0	0	-2	DA
0	0	-2	-2	DA
0	0	0	-2	DA
0	0	2	-2	DA
0	0	0	-10	DA
0	0	-2	2	DA
0	0	0	-4	DA
0	0	-2	-2	DA
0	0	0	-2	DA
0	0	-6	-6	DA
0	0	0	-2	DA
0	0	-10	-10	DA

Tabela 7.2: Zaporedje koordinat za izris srca



Slika 7.4: Končni izdelek med poskusnim graviranjem

Viri in literatura

[1] (2012) History of CNC Machining. Dostopno na http://www.cmsna.com/blog/2013/01/history-of-cnc-machining-how-the-cnc-concept-was-born/

[2] (2013) Krmiljenje koračnih motorjev v teoriji in praksi. Dostopno na http://www2.arnes.si/~sspslavr/k_motor/k_motor.html

[3] (2012) Stepper Motors or Step Motors
Dostopno na: http://www.engineersgarage.com/articles/stepper-motors

[4] (2013) Basics of Motion Control. Dostopno na:
http://www.orientalmotor.com/technology/articles/step-motor-basics.html

[5] (2013) Astrosyn koračni motorji. Dostopno na
http://astrosyn.com/section.php/20/1/nema_23

[6] (2009) DCS1 Power Supply. Dostopno na
http://pminmo.com/PMinMOwiki/index.php5?title=DCS1

[7] (2011) Gonilnik koračnih motorjev. Dostopno na
http://pminmo.com/PMinMOwiki/index.php5?title=L297-8

[8] (2013) Tehnična dokumentacija čipa L297. Dostopno na :
http://www.st.com/web/catalog/sense_power/FM142/CL851/SC1790/SS1160/PF63146

[9] (2013) Tehnična dokumentacija čipa L298. Dostopno na :
http://www.st.com/web/catalog/sense_power/FM142/CL851/SC1790/SS1555/PF63147

[10] (2013) H-Bridge controller using IC L298. Dostopno na
http://www.robometricschool.com/2013/11/electronic-circuit-schematic-h-bridge.html

[11] (2007) Krmilnik koračnega motorja L297-8. Dostopno na
http://www.pminmo.com/forsale/1X279805.pdf

[12] (2013) Razvojni sistem STM32F4 Discovery. Dostopno na http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00039084.pdf

[13] (2012) Prolific PL-2303HX USB to Serial Bridge Controller. Dostopno na: http://www.prolific.com.tw/UserFiles/files/ds_pl2303HXD_v1_4_4.pdf

[14](2012) USART prosojnice pri predmetu VGRS. Dostopno na https://ucilnica.fri.uni-lj.si

[15] (2013) Risanje linije s pomočjo Bresenhamovega algoritma. Dostopno na http://en.wikipedia.org/wiki/Bresenham's_line_algorithm

[16] (2013) G-koda. Dostopno na http://en.wikipedia.org/wiki/G-code

[17] (2005) Pretvornik napetosti SN74LVC4245A. Dostopno na http://www.ti.com/lit/ds/scas375h/scas375h.pdf