Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Leon Golob

Preprosta naprava za lasersko graviranje

diplomsko delo na univerzitenem študiju

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana 2013

Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

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: 01925/2013
Datum: 03.04.2013

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

Kandidat: **LEON GOLOB**

Naslov: **PREPROSTA NAPRAVA ZA LASERSKO GRAVIRANJE**
A SIMPLE DEVICE FOR LASER ENGRAVING

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Načrtujete preprosto napravo za lasersko graviranje v plastiko. Krmilnik naprave naj temelji na mikrokrmilniku STM32F4 z jedrom ARM Cortex-M4. Procesorski krmilnik mora krmiliti koračne motorje ter vezje za krmiljenje laserske diode. Zaradi lažjega doseganja časovnih zahtev med posameznimi opravili in prekinitveno-servisnimi podprogrami, le-te načrtujte v odprtokodnem operacijskem sistemu za delo v realnem času FreeRTOS. Za izvedbo mehanskega sestava gravirne naprave uporabite mehanske dele (vodila, koračne motorje, ipd.) iz osebnega računalnika.

Mentor:
izr. prof. dr. Patricio Bulić

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Leon Golob, z vpisno številko **63060052**, sem avtor diplomskega dela z naslovom:

PREPROSTA NAPRAVA ZA LASERSKO GRAVIRANJE

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 30. junij 2013 Podpis avtorja:

Zahvaljujem se mentorju izr. prof. dr. Patriciu Buliću za pomoč in vodenje pri opravljanju diplomskega dela. Zahvaljujem se tudi Andražu za vse predloge in nasvete.

Posebej pa se zahvaljujem družini za podporo in razumevanje v času študija. Hvala staršem, ker sta vsa leta omogočala in vedno spodbujala moje izobraževanje.

Kazalo

Seznam uporabljenih kratic

Povzetek

Abstract

1 Uvod **1**

2 Strojna oprema **3**

2.1 Uporabni deli iz računalnika 3

2.2 Kupljeni deli 4

2.3 Sestava posameznih delov 7

2.3.1 Laser 7

2.3.2 Koračni motorji 12

2.4 Celotna naprava 17

2.5 Opis mikrokrmilnika 25

3 Programska oprema **27**

3.1 Osnovni program 27

3.2 FreeRTOS 31

3.2.1 Osnovni program prilagojen za FreeRTOS 35

3.2.2 Razhroščevanje opravil 42

4 Priprava in potek graviranja **45**

4.1 Rezultat graviranja 49

5 Sklepne ugotovitve **51**

Priloga **53**

Seznam uporabljenih kratic

ADJ	Adjust – vhodna nožica regulatorja LM317
ARM	Advanced RISC Machine
ATX	Advanced Technology eXtended
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CNC	Computer Numerical Control
DMIPS	Dhrystone Million Instructions Per Second
FPU	Floating Point Unit
FTDI	Future Technology Devices International
GND	Ground – masa, zemlja
LD	Laser Diode – laserska dioda
LDD	Laser Diode Driver (Constant current laser driver)
LED	Light Emitting Diode
LQFP	Low profile Quad Flat Package
MOS-FET	Metal–Oxide–Semiconductor Field-Effect Transistor
RAM	Random Access Memory
RTOS	Real Time Operating System
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
ZDA	Združene Države Amerike

Povzetek

V okviru diplomske naloge sem izdelal preprosto napravo za lasersko graviranje. Posebnost le-te je izdelava iz obrabljenih ali starih delov računalnika. Za premikanje po dvodimenzionalnem prostoru, sem iz dveh enot CD in DVD uporabil koračna motorja. Spremenjeno lasersko diodo pisalne glave enote DVD sem uporabil za graviranje. Sistem krmili mikrokrmilnik STM32F4 z jedrom ARM Cortex-M4 z nameščenim odprtokodnim operacijskim sistemom za delo v realnem času FreeRTOS.

Najprimernejše slike za graviranje so emblemi. Zaradi preprostih oblik jih je lahko spremeniti v binarno sliko. Napisal sem program, ki izbrani emblem spremeni v črno-belo sliko, pri čemer sem posamezne slikovne točke zapisal v tekstovno datoteko kot niz ničel in enic. To datoteko sem preko vmesnika USART (*Universal Synchronous Asynchronous Receiver Transmitter*) poslal gravirni napravi, ki po vrsticah bere te vrednosti. Ko naleti na enico vklopi laser in ga izklopi, ko je na vhodu ničla. Najprimernejši material za graviranje je črna plastika ali mehkejši les.

Največja pomankljivost moje gravirne naprave je majhna površina graviranja. V primeru večje površine je priporočljivo izboljšati algoritem za samodejno graviranje.

Ključne besede: koračni motor, laserska dioda, naprava za graviranje, operacijski sistem za delo v realnem času, STM32F4-Discovery

Abstract

For the purpose of the thesis we developed a simple laser engraving device. Its construction is particularly interesting as it consists of used or old computer parts. We used two stepper motors from CD and DVD drives to achieve two-dimensional motion. A changed laser diode on the DVD write head was adapted for engraving. The system is controlled by an STM32F4 microcontroller with an ARM Cortex-M4 core; it has FreeRTOS-an open source real time operating system-installed on the device.

The most appropriate engraving images are emblems or logos. Their simple form is easy to change into a binary image. We wrote a program that changes a selected emblem into a black and white image, where individual pixels are written in a text file as a string of zeros and ones. The text file is then sent to the engraving device through the USART interface (Universal Synchronous Asynchronous Receiver Transmitter), which reads the values line by line. The device turns the laser on when the input is one and turns it off when the input is zero. The most appropriate engraving materials are black plastic or softer wood.

Our device has a small engraving surface, which is also its major deficiency. For larger surfaces, the algorithm for automatic engraving should be improved.

Key words: stepper motor, laser diode, engraving device, real time operating system, STM32F4-Discovery

Poglavje 1

Uvod

Ko nam računalnik ne služi več, ga velika večina ljudi zamenja. Zaradi hitrega razvoja strojne in programske opreme, predvsem računalniških iger, računalnik v roku nekaj let postane prepočasen in zamenljiv. Po izteku življenjske dobe ga nekateri računalničarji morda uporabijo za domači strežnik, večina pa jih konča na smetišču.

Na spletnem portalu video vsebin YouTube sem zasledil posnetek, kako lahko spremenimo namen uporabe določenim napravam. Najbolj zanimiv je bil posnetek o spremembi moči laserja iz enote DVD, kjer lahko prižigamo vžigalico ali prerežemo oziroma prežgemo plastičen material temnejše barve. Z opisanim posnetkom je prišla zamisel o izgradnji podobnega laserja. Laserski modul sem kasneje dopolnil s koračnimi motorji in nastala je preprosta naprava za lasersko graviranje.

Profesionalna naprava za graviranje je običajno računalniško vodena naprava, kjer obdelovalnemu stroju podamo podatke v numerični obliki. Tako imenovana *G-koda* [5] je programski jezik za podajanje navodil napravi CNC (*Computer Numerical Control*) [3]. V osnovi ta navodila opisujejo kaj naj naprava CNC dela in kako naj to stori. Glede na vnešene podatke, kot so hitrost, pozicija in pot premika, se potem izvršijo ukazi za premik elektromotorjev na sami napravi. Postopek se ponavlja, dokler željena vrednost ni enaka dejanski vrednosti oziroma poziciji.

Moja naprava za graviranje je preprosta zato, ker ne uporablja numeričnega vnosa podatkov v obliki *G-kode*, ampak je vhod tekstovna datoteka z zapisi ničel in enic. To datoteko dobimo iz črno-bele slike tipa *bitmap* [2], kjer namesto vrednosti za črno slikovno točko zapišemo vrednost ena, belim slikovnim točkam pa zapišemo vrednost nič. Lasersko graviranje poteka po vrsticah, pri čemer naprava proži laser, ko je na vhodu enica.

Za izdelavo sem uporabil stare dele računalnika, kot so napajalnik, enota za branje zgoščenk CD, enoto za branje in zapisovanje zgoščenk DVD, tipke računalniškega ohišja, večino povezovalnih kablov osnovne plošče ter hladilne enote. Pri sami izvedbi sem imel več možnosti, saj navodil za izdelavo gravirne naprave iz odpadnih delov ni. Zaradi tega sem na koncu izdelal dve napravi oziroma prototipa. Za krmiljenje same naprave sem uporabil mikrokrmilnik ARM podjetja STM, ki je primeren zaradi velikega števila vhodno-izhodnih nožic, precejšne podpore perifernih naprav in ima možnost uporabe operacijskega sistema za delo v realnem času.

Samo vezje, kontrolnik nad laserjem in kontrolnike nad koračnimi motorji sem sestavil na testni ploščici. Kasneje sem izdelal še tiskano vezje, saj je samo vezje zaradi tega manj občutljivo na motnje in zato bolj zanesljivo. Polega tega je končni izdelek bolj podoben pravi napravi. Sheme in datoteke za izdelavo tiskanine so prikazane v prilogi na koncu diplomske naloge.

Posebno pozornost je potrebno nameniti zaščiti oči z očali za določeno valovno dolžino laserja. Zaščita je še toliko bolj pomembna, ker sem spremenil moč laserja in ker naprava za graviranje ni zaščitena s pokrovom. Varnostni razred zaprte enote DVD spada v razred 1, zaradi zgoraj navedenih razlogov pa se je razred moje naprave za graviranje dvignil v varnostni razred 3B [4].

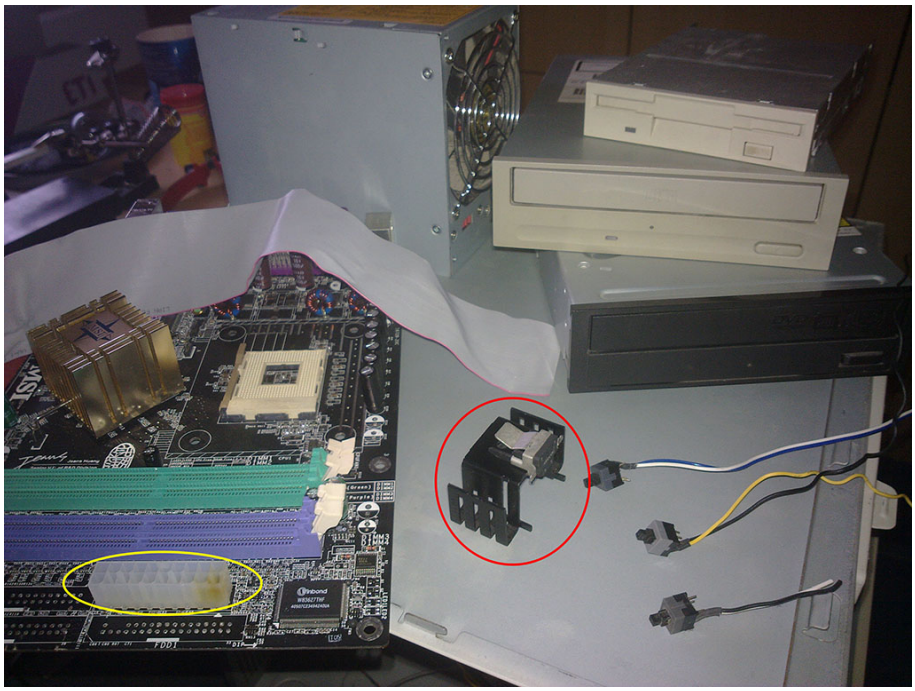
Poglavje 2

Strojna oprema

2.1 Uporabni deli iz računalnika

Vzrokov za menjavo računalnika je veliko. Nekatere narekuje trend, druge boljše tehnološke zmogljivosti, velika večina ljudi pa računalnik zamenja zaradi pokvarjene komponente. Slednje je bilo tudi vzrok za menjavo mojega računalnika, saj je bil nakup novega smotrnejši kot popravilo le-tega. Odločil sem se, da spremenim njegov namen v nekaj relativno uporabnega. Na sliki 2.1 so vsi uporabni deli, ki sem jih pri projektu uporabil.

Za povsem osnovno delovanje sem potreboval obe enoti za predvajanje zgoščenk CD in DVD. Napajalnik mi je služil za napajanje, razni podatkovni vodniki pa za povezovanje elementov. Osnovna plošča je vir vsestransko uporabnih elementov, kot so kondenzatorji, svetleče diode LED, hladilni elementi in posamezne nožice. Pri ročnem upravljanju so mi prišle prav razne tipke v samem ohišju računalnika, enoto za diskete sem porabil za držalo laserskega modula in za pladenj vgraviranega materiala.

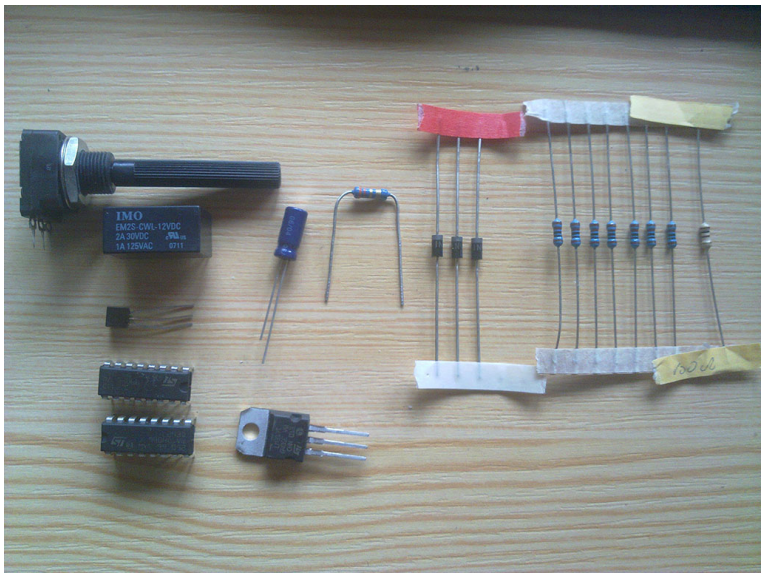


Slika 2.1: **Uporabni deli starega računalnika.** Vzrok za menjavo računalnika je bil kratek stik na priključku za napajanje (obkroženo z rumeno elipso). Rdeča elipsa prikazuje hladilni element, ki sem ga koristno porabil pri hlajenju linearnega tokovnega regulatorja LM317.

2.2 Kupljeni deli

Zbirko novih delov prikazuje slika 2.2. Linearni regulator, rele, tranzistor, upori in diode so cenovno zelo ugodni in dostopni deli. Ustrezni laserski modul sem naročil iz ZDA, saj ga v Evropi ni mogoče dobiti. Lahko bi uporabil ohišje laserja za označevanje, vendar bi imel precej težav z natančno postavitvijo laserske diode in nastavitvijo leče.

Mikrokrmilnik STM32F4 sem izbral po predlogu mentorja, saj je zaradi nizke cene in velike uporabnosti precej primeren za projekt.



Slika 2.2: **Kupljeni deli.** Levi del slike od zgoraj navzdol nam prikazuje potenciometer, rele, tranzistor in dva H-mostička. Desno je linearni tokovni regulator LM317. Ostanejo še kondenzator, diode in različni upori.

Na sliki 2.3 je najpomembnejši in najdražji del celega projekta. Zaščitna očala so obvezen del zaščite, saj nam lahko laserski žarek zaradi svoje narave močno poškoduje oči. Pri nakupu je pomembno, da izberemo kvalitetno zaščito za pravo valovno dolžino.

Slika 2.3: **Zaščitna očala.** Zaščitna očala nudijo zaščito za elektromagnetno valovanje v obsegu od 590 nm do 780 nm. Spremenjena laserska dioda sveti v rdečem spektru z valovno dolžino 658 nm.

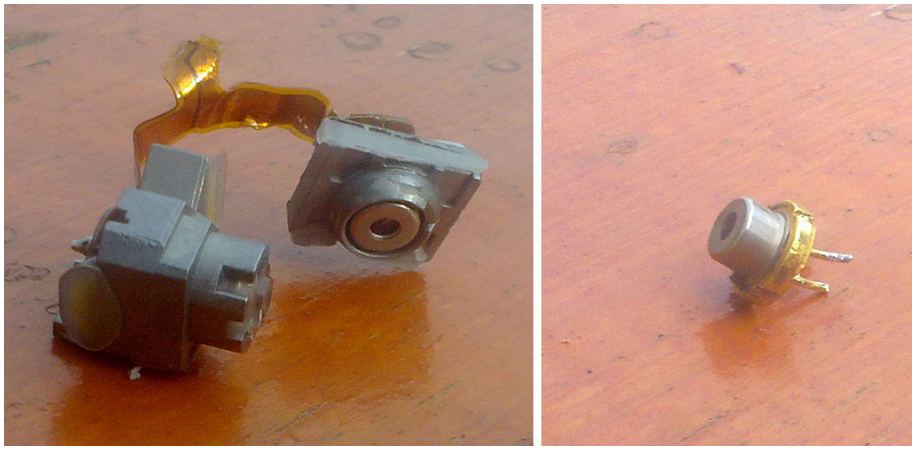
2.3 Sestava posameznih delov

2.3.1 Laser

Za izdelavo naprave moramo najprej poiskati podatke in osnovne specifikacije, kot so nazivna napetost, moč, fizične dimenzije ipd. Predvsem pri rabljenih napravah oziroma elementih je večina podatkov in specifikacij neznanih. Najboljši način za zbiranje informacij je iskanje kakršnih koli oznak na elementu samem ali pa na celotni enoti. Slika 2.4 kaže primer, kako sem pridobil potrebne informacije.

Slika 2.4: **Oznake na enotah DVD in CD.** Iz podatkov, obkroženih z rdečo barvo, izvemo model DVD pisalnika in informacijo, kdaj je bil proizveden. S pomočjo teh informacij sem na spletu poiskal željene podatke [6]. Manj dela z iskanjem podatkov je bilo na enoti CD, saj so bile željene informacije zapisane na oznaki, obkroženi z rumeno barvo.

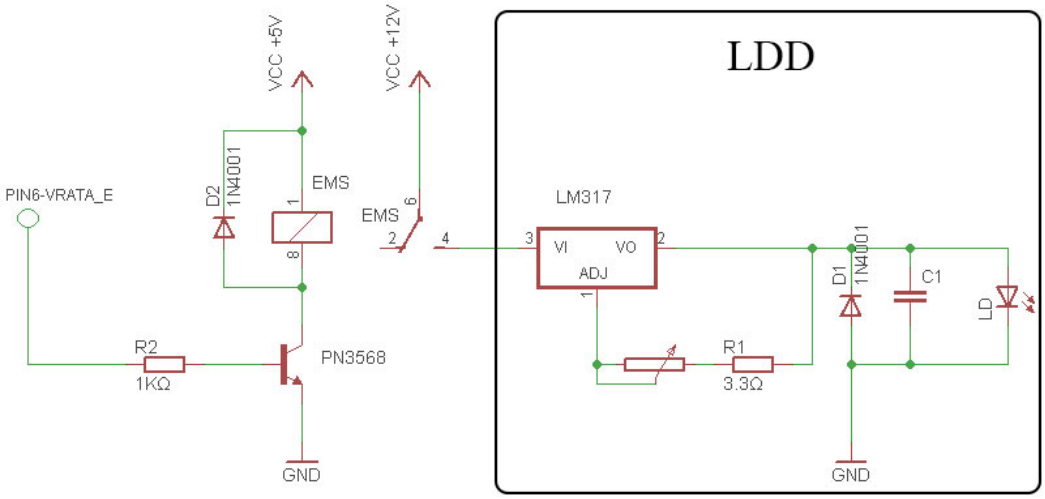
Enota CD je imela lasersko diodo samo za branje. Z optično močjo okoli 0,1 mW je prešibka za graviranje. Bralno-pisalna enota DVD ima dve laserski diodi, prikazani na sliki 2.5.



Slika 2.5: **Laserska dioda.** Levi del slike prikazuje dve laserski diodi iz enote za DVD. Dioda brez zaščitnega svinca je prikazana na desni strani slike. Pri odstranjevanju laserske diode iz ohišja bralno-pisalne glave moramo paziti, da lasersko diodo čim manj tresemo, saj so zaradi same zgradbe zelo občutljive na bolj grobe premike.

Prva je infrardeča LD (Laserska Dioda) za branje in pisanje CD-jev, ki je z močjo 5 mW neuporabna. Druga, rdeča LD, ki se uporablja za branje in pisanje DVD-jev, ima izhodno moč 200 mW [6] in je z ustrezno spremembo vhodnega toka najbolj primerna za projekt. Z valovno dolžino 658 nm, spada v rdeči spekter vidnega elektromagnetnega valovanja.

Laserska dioda je tokovno voden element, zato zanjo potrebujemo konstantno tokovno voden gonilnik z regulatorjem LM317 [1]. Shema celotnega vezja je prikazana na sliki 2.6.



Slika 2.6: **Shema vezja LDD skupaj s stikalom.**

Da sem zagotovil varno uporabo LD, sem za vezje za krmiljenje laserske diode (LDD) uporabil naslednje elemente:

- regulator LM317,
- upor 3,3 Ω,
- potenciometer 100 Ω,
- dioda 1N4001,
- kondenzator 47 μF 16 V.

Na sliki 2.6 je s črnim kvadratom označena shema za vezje LDD. Izhodni tok regulira LM317, katerega referenčna napetost je vedno 1.25 V. Na nožici ADJ in V_{OUT} sem povezal upor vrednosti 3,3 Ω in potenciometer za natančnejše reguliranje izhodnega toka. S preprostim izračunom po Ohmovem zakonu (2.1) in enačbo za moč (2.3), sem dobil maksimalen tok (2.2) ter izhodno moč (2.4) laserske diode.

Podatki potrebni za izračun: U_{REF}*=1,25 V in* R*=3,3* Ω

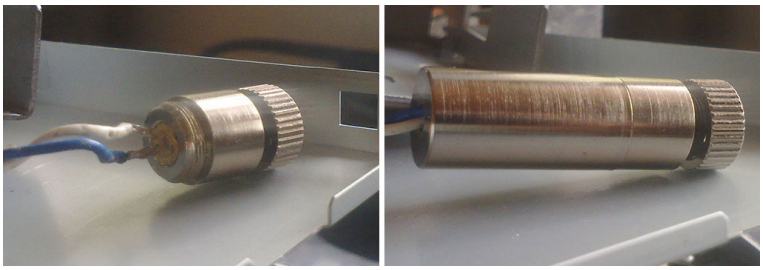
$$I = U_{REF}/R \tag{2.1}$$

$$I = 1,25\ V/3,3\ \Omega \doteq \mathbf{380\ mA} \tag{2.2}$$

$$P = (U_{REF})^2/R \tag{2.3}$$

$$P = (1,25\ V)^2/3,3\ \Omega \doteq \mathbf{474\ mW} \tag{2.4}$$

Stara vrednost izhodne moči laserske diode je bila 200 mW. Če upoštevamo novo vrednost izhodne moči, ki je 474 mW, in to zapišemo v odstotkih, sem moč laserske diode povečal za dodatnih 130 %. Posledica povečane obremenitve je zmanjšana življenjska doba delovanja in povečano segrevanje LD. Za odvajanje toplote sem uporabil namensko ohišje [8]. Na sliki 2.7 je prikazan laserski modul po vstavitvi diode v ohišje. Znotraj ohišja je zbiralna leča, pri kateri je s sprednje strani mogoče nastavljati goriščno razdaljo.



Slika 2.7: **Laserski modul**.

Po mnenju nekaterih raziskovalcev na forumu [7] naj bi bilo laserske diode mogoče krmili s tokom do 600 mA, kar drastično zmanjša njihovo življenjsko dobo. Pri tako velikem toku nastanejo tudi težave z odvajanjem toplote in samim akrilnim steklom, ki se uporablja za lečo. Za tako velike izhodne moči je priporočljiva leča iz stekla.

S potenciometrom lahko tok reguliramo bolj natančno, kar pride prav pri določanju moči za različno globino oziroma debelino graviranja. Kondenzator je potreben za zaščito LD v primeru napetostnih špic. Do teh lahko pride v napajalniku zaradi neobremenjenosti. Zaščitno diodo 1N4001 sem uporabil za zaporni tok v primeru, da pride do pomote pri priklopu napajalnika, in za primer praznenja kondezatorja.
Za vklapljanje vezja LDD sem uporabil naslednje elemente:

- upor 1 kΩ,
- rele EMS 12 VDC, 1 A,
- bipolarni tranzistor NPN PN3563,
- dioda 1N4001.

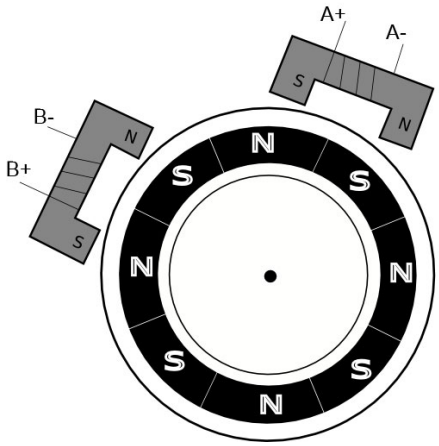
Izhodno nožico iz mikrokrmilnika sem pripeljal preko upora vrednosti 1 kΩ na bazo tranzistorja, emitor je vezan na maso, kolektor pa na tuljavo mehanskega releja, ki vklaplja vezje LDD. Poleg tranzistorja za vklop uporabljam rele, saj LD porablja relativno visoko napetost oziroma tok. Uporabil sem mehanski rele, ki pa zaradi tuljave in posledično indukcije pri ugašanju povzroča povratni tok. Težavo sem rešil z zaporno smerjo diode. Za hitrejše preklope sem pri izdelavi tiskanega vezja uporabil polprevodniški rele MOSFET.

2.3.2 Koračni motorji

Enoti za CD in DVD imata vsaka po tri različne enosmerne motorje:

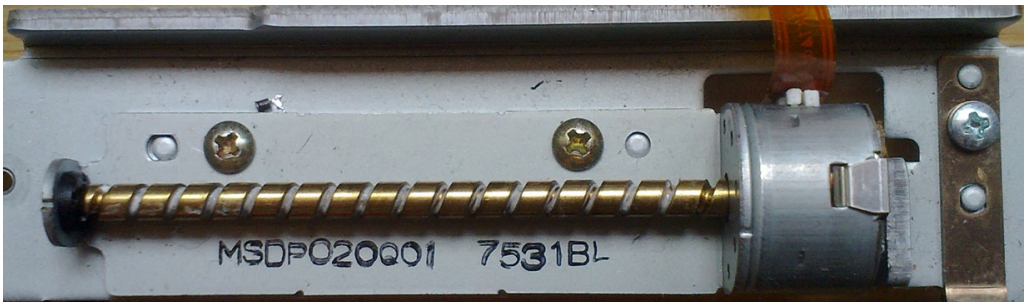
- motor za odpiranje in zapiranje nosilca, kamor položimo zgoščenko,
- brezkrtačni motor za vrtenje zgoščenke,
- koračni motor za premikanje bralno-pisalne laserske glave.

Koračni motorji se delijo glede na uporabljeni tip tuljav. Ko se v tuljavah zamenja smer toka in se spremeni magnetno polje, motor naredi en korak. Od tod ime koračni motor. Poznamo unipolarne in bipolarne koračne motorje, ki se razlikujejo v tem, kako se ta sprememba doseže. Bipolarni motor ima eno navitje na par polov (Slika 2.8), za spremembo smeri pa potrebuje štiri preklopna stikala.



Slika 2.8: **Konceptualna zgradba koračnega motorja [18].** V rotorju motorja se izmenjuje sodo število polaritet. Več kot je izmenjanih parov polaritet, manjši je korak premika. V ilustriranem primeru je teh polov osem, pri čemer je kot premika velik 45°.

Koračna motorja v obeh enotah sta bipolarna. V nadaljevanju bo s pojmom **motor** mišljen bipolarni motor. Na sliki 2.9 je eden od dveh bipolarnih koračnih motorjev, ki sem ju uporabil pri projektu.

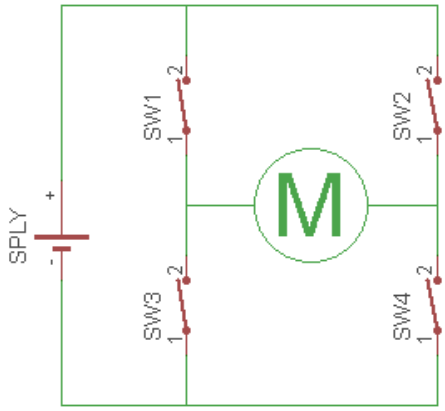


Slika 2.9: **Bipolarni koračni motor s polžastim vijakom.** Dolžina vijaka je 48 mm, vendar je zaradi varnosti ter same zgradbe uporabnih le 36 mm dolžine. Največja vgravirna površina je tako velikosti 36 mm × 36 mm.

Za krmiljenje sem najprej želel uporabiti integrirano vezje, ki se nahaja v samih enotah CD ali DVD, ampak je veliko lažje uporabiti H-mostiček. S slike 2.10 lahko opazimo značilno obliko vezja, ki spominja na črko H.
S takšnim vezjem krmilimo klasičen enosmerni krtačni motor. Z sklenjenima stikaloma **SW1** in **SW4** se motor vrti v smeri urinega kazalca. V primeru zaprtih stikal **SW2** in **SW3** se motor vrti v nasprtotno smer urinega kazalca. Na takšno vezje lahko povežemo tudi koračne motorje. Štiri preklopna stikala krmilijo eno navitje od dveh tuljav, ki so v koračnem motorju. Skladno s povedanim potrebujemo za kontrolo koračnega motorja osem preklopnih stikal ali dvojni H-mostiček. Uporabil sem vezje L293D [11], ki je prikazano na sliki 2.11. Ker sem uporabil dva koračna motorja, sta za krmiljenje obeh potrebna dva H-mostička L293D.
Koračni motorji imajo tri različne možnosti zaporedja za krmiljenje [9]:

- polni korak - korak je sestavljen iz štirih sekvenc,
- polovični korak - korak je sestavljen iz osmih sekvenc,
- mikro korak - sekvenca je amplituda toka v obliki sinusnega valovanja.

En korak je sestavljen iz zaporedja štirih različnih vhodnih kombinacij signalov, ki so strnjeni v tabeli 2.1. Pri naslednjem koraku se sekvenca ponovi.

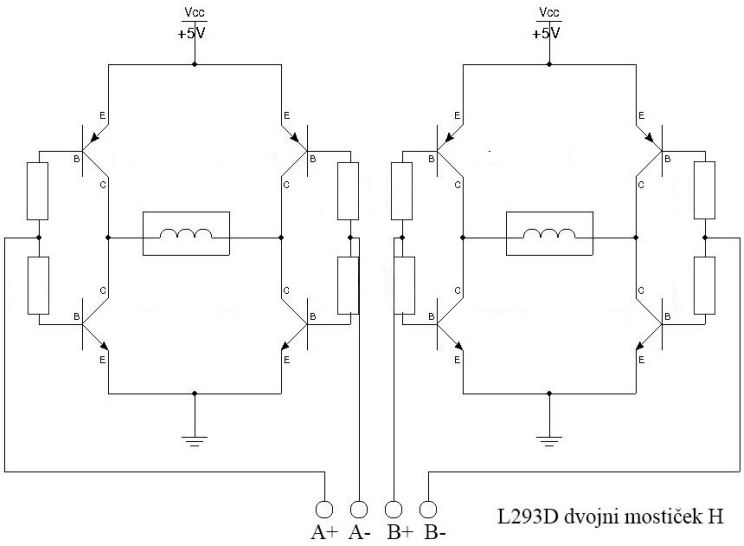


Slika 2.10: **H-mostiček.** Preprosta ilustracija poenostavljenega H-mostička s preklopnimi stikali. V pravem vezju so stikala tranzistorji, manjkajo pa tudi upori in diode za zaščito pred povratnim tokom.

sekvenca\tuljava	A_+	A_-	B_+	B_-
1.	H	L	H	L
2.	L	H	H	L
3.	L	H	L	H
4.	H	L	L	H
1.	H	L	H	L

Tabela 2.1: **Štiri vhodne kombinacije za en korak.** H predstavlja visoko stanje na izhodu, L pa nizko stanje.

Koračni motor, ki sem ga uporabil, ima dvajset parov polaritet. Razlika med dvema vhodnima kombinacijama je obrat za 18° oziroma za 0,15 mm, Pri enem polnem koraku je tako to obrat za 72° ali 0,6 mm pomika. Polžasti vijak je dolg 48 mm, zaradi same zgradbe nosilca bralno-pisalne glave pa je uporabnih le 37 mm. Pri 60 korakih tako pridemo do dolžine 36 mm. En korak oziroma nekaj manj kot 1 mm dolžine sem pustil zaradi varnostnih

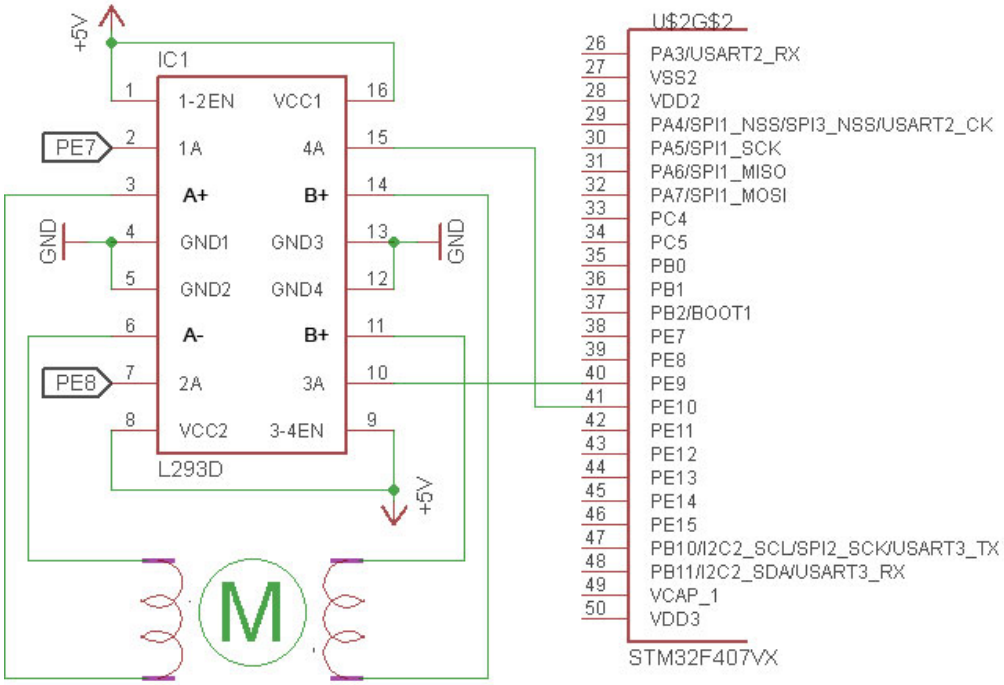


Slika 2.11: **Vezje L293D.** En H-mostiček krmili eno navitje tuljave. Črka D nam pove, da je v vezju dvojni H-mostiček, s čimer nadzorujemo dve tuljavi oziroma en motor.

razlogov. V primeru polovičnega koraka, bi tako za razmik enega koraka dobil 0,3 mm, s čimer bi dvakratno povečal ločljivost.

Pri mikrokoraku se sekvenca podaja v obliki sinusnega valovanja toka. Med dvema korakoma osnovne frekvence sta tuljavi različno inducirani. Ker so spremembe na tuljavi posledica toka, ki se zvezno spreminja po sinusu, dobimo mikropremike [17].

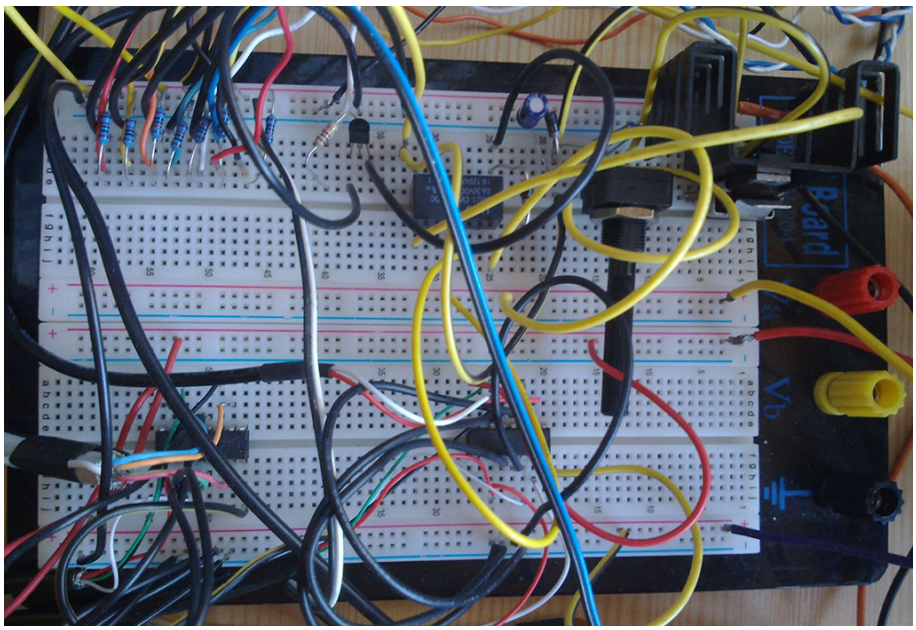
Za svoj projekt sem uporabil polni korak, saj razmik 0,6 mm zadošča za uporabno graviranje. V tem načinu je tudi navor motorja največji, kar mi je koristilo zaradi teže samega nosilca za laserski modul. Shema vezave enega motorja je prikazana na sliki 2.12.



Slika 2.12: **Shema vezja koračnega motorja.**

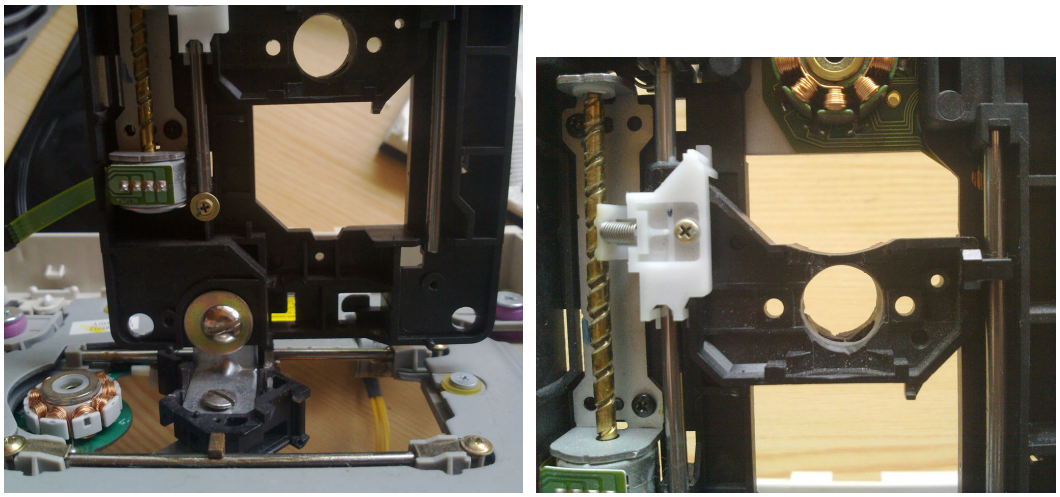
2.4 Celotna naprava

Preden sem se lotil sestave vezja, sem iz osnovne plošče računalnika odspajkal vse priklopne nožice. Te sem skupaj s podatkovnimi vodniki in ostalimi žicami uporabil za povezovanje elementov na testni ploščici. Shema na sliki 5.1 v prilogi prikazuje vezavo laserskega vezja LDD in vezavo koračnih motorjev s H-mostički. Poleg tega je vidna tudi vezava desetih "pull-down" uporov za tipke in za senzorje. Navadno so takšni upori desetkratnik vhodne impedance nožice, tako da sem uporabil 10 kΩ upore. Slika 2.13 prikazuje zgoraj opisano shemo realizirano na testni ploščici.



Slika 2.13: **Vezje na testni ploščici.** Levo zgoraj je prikaz sedmih uporov za tipke in dva senzorja. Desno se s tranzistorjem začne in zaključi z LM317 vezje za laserski modul. Spodaj je vezava dveh H-mostičkov za koračna motorja.

Za izdelavo preproste naprave za graviranje nisem imel nobenih načrtov, zato sem na koncu izdelal dva prototipa, ki sta opisana v nadaljevanju.



(a) Kotni nosilec. *(b)* Držalo laserskega modula.

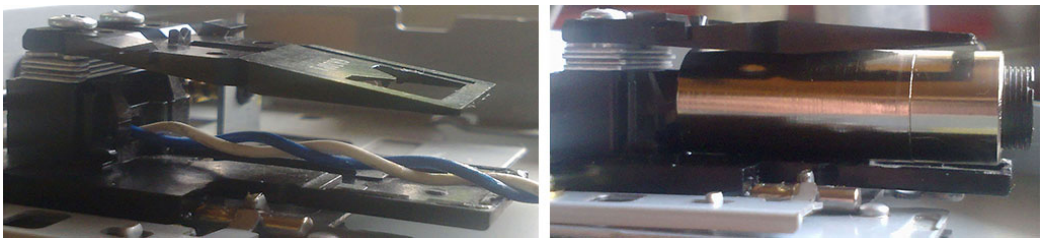
Slika 2.14: **Obdelava obeh nosilcev bralno-pisalnih glav.**

Iz ohišij bralno-pisalnih glav obeh enot sem najprej odstranil vse nepotrebne elemente zato, da je bila skupna teža čim manjša. Poleg tega sem prazna ohišja tudi lažje obdeloval. Za prvi prototip sem na ohišje vodoravne bralno-pisalne glave pritrdil kotni nosilec za ohišje navpičnega vodila (slika 2.14a). V ohišje bralno-pisalne glave navpičnega nosilca pa sem zvrtal odprtino, kamor sem vstavil laserski modul, kar je dobro vidno na sliki 2.14b.

Za drugi prototip sem obe ohišji bralno-pisalnih glav dodatno očistil in opilil, da sem dobil ravni površini. Na vodoravno ohišje prve bralno-pisalne glave sem prilepil zunanji kovinski okrov enote za diskete, ki služi kot pladenj za vgraviran material (slika 2.15).

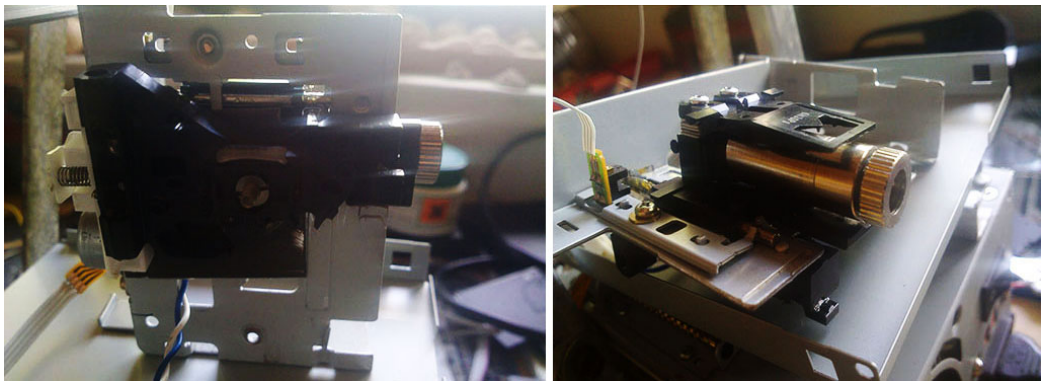
Slika 2.15: **Vodoravno ohišje bralno-pisalne glave z nosilcem.**

Poleg kovinskega pokrova enote za diskete, sem uporabil še celoten nosilec, na katerem je manjši koračni motor skupaj z bralno-pisalno glavo. Z manjšimi spremembami mi je bralno-pisalna glava služila za držalo laserskega modula, kar je razvidno s slike 2.16.



Slika 2.16: **Držalo za laserski modul.** Odstranjeno vezje bralno-pisalne glave je pustilo primerne utore za držanje laserskega modula

Celoten nosilec z laserskim modulom sem pritrdil na navpično ohišje bralno-pisalne glave. Na sliki 2.17 se to dobro vidi iz različnih kotov.
Koračni motor enote za diskete sem uporabil za fokusiranje laserja, da mi ni bilo potrebno ročno nastavljati leče. To mi je koristilo pri različni debelini vgraviranega materiala.



Slika 2.17: **Navpično ohišje bralno-pisalne glave z laserskim nosilcem.**

Motorja iz enot za CD in DVD sem skupaj z vodili uporabil za premikanje po oseh X in Y. Če si predstavljamo sliko, ki jo bomo gravirali v kartezičnem koordinatnem sistemu, potem morata biti motorja z vodili pravokotna drug na drugega. Na sliki 2.18 sta prikazani dve možni izvedbi takšnega sistema. Paziti je treba, da sta oba nosilca vodil karseda pravokotna, saj je vsako odstopanje dobro vidno na končnem izdelku.

Napajalnik računalnika je bil izdelan po standardu ATX (*Advanced Technology eXtended*) [12], kar pomeni, da s tipko za vklop ne dobimo nikakršne napetosti na vodnikih. Vijolični vodnik zagotavlja začetno napetost za osnovno ploščo, ki pri vklopu preko zelenega vodnika sporoči napajalniku, naj se vklopi. Preprosta rešitev za delovanje napajalnika brez osnovne plošče je vezava zelenega vodnika na ozemljitveni vodnik GND, kar prikazuje slika 2.19.

Za napajanje projekta sem uporabil naslednje vodnike:

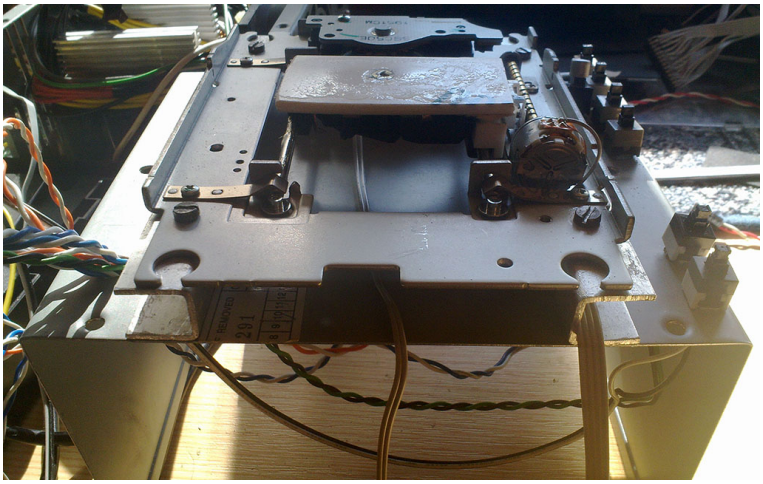
- rdeči vodnik napetosti 5 V,
- oranžni vodnik napetosti 3,3 V,
- rumeni vodnik napetosti 12 V.

Slika 2.18: **Izvedbi pravokotnega sistema.** Leva slika prikazuje prvi prototip, kjer sem navpični nosilec pritrdil na ohišje bralno-pisalne glave vodoravnega nosilca. Druga izvedba prikazuje ločena nosilca vodil. Sama struktura je bolj trdna in ne povzroča neželjenih vibracij. Poleg tega je tudi bolj podobna pravi CNC napravi.

Slika 2.19: **Rešitev za vklop napajalnika.** Zeleni vodnik je treba vezati na ozemljitveni vodnik GND.

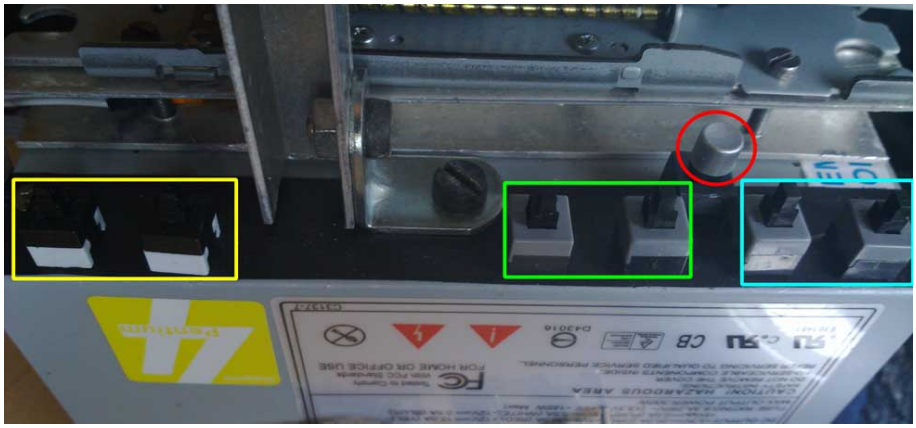
Ostale vodnike, ki jih nisem potreboval, sem primerno zložil v notranjost napajalnika.

Na sliki 2.20 se vidi sprememba zgornjega dela okrova napajalnika. Na osrednji del sem pritrdil dva distančnika za vodoravni nosilec vodil. Zraven sem na eni strani zvrtal odprtine za tipke, na drugi strani pa večjo odprtino za uporabljene vodnike iz napajalnika in vodnike tipk. Na okrov sem pritrdil še celoten sistem navpičnega vodila. S tem sem pridobil kompaktno konstrukcijo, primerno za prenašanje, poleg tega sem imel manj dela z določanjem pravih kotov.

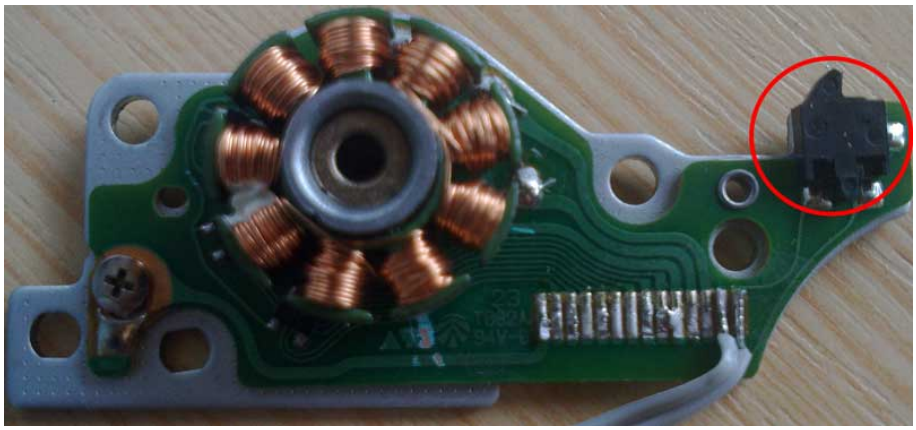


Slika 2.20: **Zgornji del okrova napajlnika.** Po sredini sta vidna dva kovinska distančnika za vodoravni nosilec. Na desni strani so smerne tipke. Za vsak motor sem potreboval dve tipki. Ena tipka se uporablja za vklop in izklop laserja. Na levi strani je prehodna luknja za njihove vodnike.

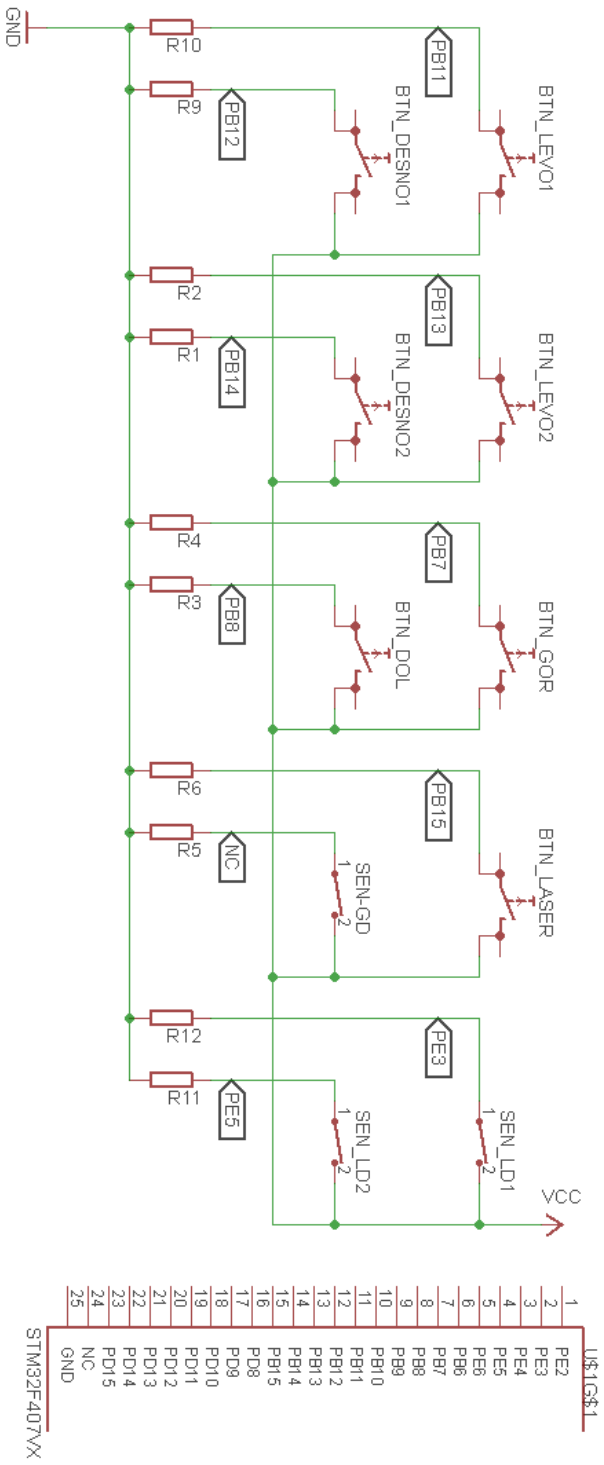
Slika 2.21 prikazuje končno postavitev tipk za ročno krmiljenje naprave. Senzorja, ki sem ju uporabil za določanje robnega položaja, sta mehanski mini detektorski stikali (slika 2.22). Shema vezave vseh tipk in senzorjev na mikrokrmilnik je predstavljeno v sliki 2.23.



Slika 2.21: **Postavitev krmilnih tipk.** Tipki obkroženi z rumeno barvo sta namenjene krmiljenju navpičnega koračnega motorja za premik laserskega modula. Tipki obkroženi z zeleno barvo krmilita vodoravni pladenj, tipki obkroženi s turkizno barvo pa krmilita navpično ohišje celotnega laserskega nosilca. Z rdečo barvo je obkrožena tipka za vklop oziroma izklop laserja.



Slika 2.22: **Senzor za detekcijo robnega položaja.** Obkroženi del prikazuje mini detektorsko stikalo. Levo od njega je stator brezkrtačnega motorja za vrtenje zgoščenke.



Slika 2.23: **Shema vezja za tipke in senzorja.** Zastavica z oznako kaže na vodilo, ki je povezano s pripadajo nožico mikrokrmilnika.

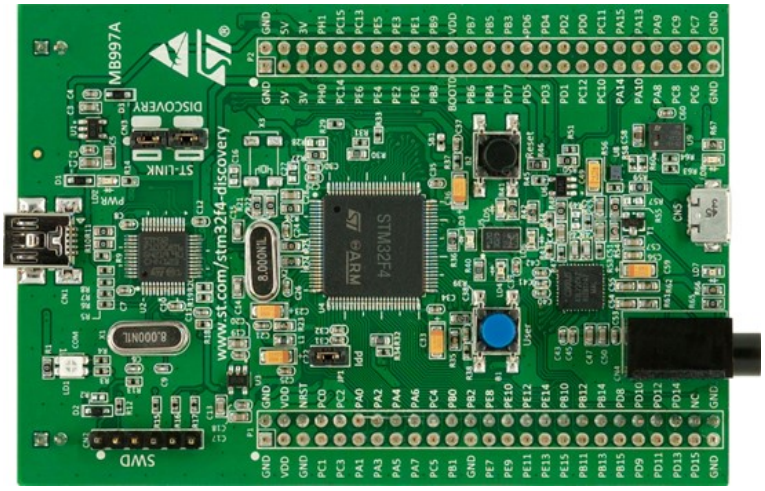
2.5 Opis mikrokrmilnika

Razvojna ploščica STM32F4 Discovery [13], slika 2.24, je osnovana na 32-bitnem ARM-u z jedrom Cortex-M4F nove generacije (168 MHz jedro STM32F407VGT6 [23] s podporo računanja v plavajoči vejici (*FPU*) z 210 DMIPS, 17 časovnikov, podpora za Ethernet). Uporablja 1 MB velik Flash pomnilnik ter 192 KB pomnilnika RAM. Poleg senzorja gibanja, zvočnega vhoda, 8 vgrajenih svetlečih diod LED in dveh tipk nudi podporo še za digitalno procesiranje signalov.

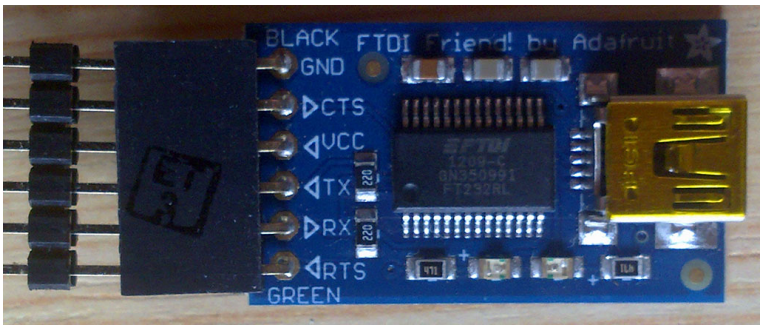
Eden od razlogov za izbiro tega mikrokrmilnika je bil ohišje LQFP100 (*Low profile Quad Flat Package 100 pins*), ki ponuja veliko število vhodno-izhodnih nožic za enostavno povezovanje in sondiranje. Za preprostejše programiranje in razhroščevanje ima razvojna ploščica vgrajen programator ST-LINK/V2, ki ga povežemo z računalnikom preko vmesnika USB.

Na voljo je veliko različnih orodij za razvoj programske kode. Sam sem uporabil razvojno okolje IAR–Ewarm [22]. Podjetje STMicroelectronics ponuja za izbrani mikrokrmilnik *firmware* z dvaindvajsetimi primeri [13]. To mi je pomagalo, da sem se z mikrokrmilnikom kar najhitreje spoznal. Za mikrokrmilnik STM32F4 so na voljo tudi izbrane razširitve, kot so zaslon LCD občutljiv na dotik, digitalna kamera, microSD in osnovni periferni vhodi oziroma izhodi.

Periferne vhode in izhode vrat C mikrokrmilnika sem uporabil za komunikacijo z računalnikom. Slika 2.25 prikazuje vmesnik FT232R [14], ki poskrbi za serijski prenos podatkov na protokol USB.



Slika 2.24: **Razvojni sistem ARM STM32F4 Discovery.** Za nizko ceno sem dobil odlično izbiro perifernih naprav.



Slika 2.25: **FTDI USART vmesnik za serijski prenos na protokol USB.** Za moje potrebe je bilo dovolj, da sem uporabil samo nožice R_x, T_x in GND. Omogoča tudi programiranje mikrokrmilnikov AVR.

Poglavje 3

Programska oprema

Podjetje STMicroelectronics je pripravilo številne začetne programe, t. i. *firmware* **STM32F4-Discovery_FW_V1.1.0** [19], za raziskovanje zmogljivosti mikrokrmilnika STM32F4. Moj projekt je zahteval delo z vhodno-izhodnimi nožicami, zato sem uporabil primer *IO_toggle*, ki se nahaja v mapi *Peripheral_Examples*. V podpoglavju 3.1 je program opisan bolj podrobno.

Osnovni program je imel nekaj pomanjkljivosti. Veliko dela sem imel z določanjem časovnih zamikov pri pošiljanju sekvenc motorju. Pojavilo se je neželjeno tresenje, kar se je opazilo na končnem rezultatu graviranja. Časovni zamiki so bili težava tudi pri preverjanju tipk in z izvajanjem funkcij, ki jim ustrezajo. Po predlogu mentorja sem uporabil precej popularen operacijski sistem za vgrajene naprave FreeRTOS [15]. Z uporabo opravil, vrst in semaforjev sem težave zadovoljivo rešil.

3.1 Osnovni program

Testni primer *IO_toggle* je imel v celoti nastavljeno začetno konfiguracijo. To pomeni, da so bili nastavljeni vektorji za prekinitve, sklad, programski števec in sistemska ura. S temi nastavitvami in uporabo delilnikov ure, t. i. *prescaler*, so bile določene tudi ure perifernih vodil, ki pa so na začetku onemogočene. Za uporabo vhodno-izhodnih nožic sem moral najprej vklopiti

uro vodila AHB (koda 3.1).

```
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE, ENABLE);
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
```

Koda 3.1: **Vklop ure.** Omogočil sem delovanje vhodno-izhodnih nožic za sklop vrat A, B, E in D. Vsaka posamezna vrata imajo 16 nožic.

S kodo 3.2 sem nastavil tip posameznih nožic. Vrata B in A sem uporabil za vhod tipk. Kot vhod sem delno določil tudi vrata E za senzorje, ostale nožice vrat E pa sem določil za izhod za kontrolo nad motorji. Vrata D so izhodna; uporabil sem jih za prikaz premikov z vgrajenimi LED diodami.

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 | GPIO_Pin_12|
                    GPIO_Pin_13| GPIO_Pin_14| GPIO_Pin_15;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

Koda 3.2: **Primer nastavitev nožic.** Najprej sem izbral nožice in jim določil način dela kot vhod. Tip priključka sem nastavil na Push-Pull ter omogočil notranje Pull-Down upore. Nastavil sem še hitrost delovanja in celo strukturo določil za izbrana vrata B.

Program se najprej začne v ničelnem položaju (0,0), kar pomeni, da se oba motorja premikata desno, dokler ne prideta do senzorja. V kodi 3.3 sem ponastavil še oba števca korakov. Števca sem potreboval za omejitev korakov na levi strani vodil, da se ne prekoračijo fizične omejitve polžastega vijaka.

Po začetni postavitvi v ničelni položaj, program v zanki pregleduje vhodne nožice vrat B za morebitne pritiske tipk. Na začetku sem najprej preveril tipko USER_BUTTON, ki ob pritisku na njo preklopi v samodejni način delovanja. Nato sem preverjal vhode smernih tipk za oba motorja in vhod tipke za vklop laserja. Sledi še zakasnitev, ki sem jo določil s poskušanjem (koda 3.4).

```
void InitZeroZero()
{
  while((GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_3))||(
      GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_5)))
  {
    PremikDesno1();
    PremikDesno2();
    stGD=0; //zacetna pozicija v 0
    stLD=0;
  }
}
```

Koda 3.3: **Ničelni položaj.**

```
while (1)
  {
    if(GPIOA->IDR & 0x0001) //tipka USER_BUTTON
    {
      automatic();
    }
    if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_11))
    {
      PremikLevo();
    }
  ...
    Delay(0xFFFF); //kontroliram kako tekoce zaznava moje
        pritiske tipk
  }
```

Koda 3.4: **Kontrola tipk v zanki.** Zapisa v pogojnih stavkih sta sicer različna, vendar oba prebereta rezultat izbrane vhodne nožice. Na prav tak način se preverjajo še ostale tipke.

Koda 3.5 prikazuje vrstice ukazov za premik motorja glede na zaporedje v tabeli 2.1. Na začetku sem napisal pogoj za izvajanje koraka, v nadaljevanju pa sem s postavljanjem nožic na visok in nizek nivo izvedel en korak.

```
void PremikLevo()
{
  if(stLD<60) //omejitev korakov
  {
    GPIO_SetBits(GPIOD, GPIO_Pin_12); //vklop LED diode

    GPIO_SetBits(GPIOE, GPIO_Pin_8|GPIO_Pin_10);
    Delay(zak);
    GPIO_ResetBits(GPIOE, GPIO_Pin_8|GPIO_Pin_10);

    GPIO_SetBits(GPIOE, GPIO_Pin_8|GPIO_Pin_9);
    Delay(zak);
    GPIO_ResetBits(GPIOE, GPIO_Pin_8|GPIO_Pin_9);

    GPIO_SetBits(GPIOE, GPIO_Pin_7|GPIO_Pin_9);
    Delay(zak);
    GPIO_ResetBits(GPIOE, GPIO_Pin_7|GPIO_Pin_9);

    GPIO_SetBits(GPIOE, GPIO_Pin_7|GPIO_Pin_10);
    Delay(zak);
    GPIO_ResetBits(GPIOE, GPIO_Pin_7|GPIO_Pin_10);

    GPIO_ResetBits(GPIOD, GPIO_Pin_12); //izklop LED diode

    stLD++;
  }
}
```

Koda 3.5: **Sekvenca za polni korak premika v levo.** *GPIO_SetBits()* nastavi nožico na visok nivo, *GPIO_ResetBits()* pa na nizek nivo.

Zakasnitev **Delay()** je preprosta funkcija, kjer ostanemo v zanki *while* dokler se ne izteče podana vrednost. S preskušanjem različnih vrednosti zakasnitve sem določil hitrosti premikov za različne tipe slik. Na primer šestnajstiška vrednost 0xFFFF je pri polni moči laserja primerna za graviranje polne slike, medtem ko je vrednost 0x7FFFF primerna za elipso ali kvadrat, ročno upravljanje pa je najlažje z vrednostjo 0xFFFFFF. Za premik v nasprotno smer

sem ukaze napisal v obratnem vrstnem redu. Pri tem sem zaporedje zamaknil za eno sekvenco, saj bi sicer pri prehodih iz levega v desni premik in obratno prihajalo do zamika ali preskoka koraka.

Ob pritisku na tipko USER_BUTTON se začne izvajati samodejno graviranje predhodno določene slike. Potek in priprava slike je opisana v podpoglavju 4. V prikazani kodi 3.6 sem za vsak stolpec v izbrani tabeli pogledal vrednosti. Če je vrednost 1, sem vklopil laser, v nasprotnem primeru sem laser izklopil. Pri premiku v desno sem moral vrstico tabele brati od zadnjega elementa proti prvemu elementu. Paziti sem moral na robne pogoje, saj je vedno potreben en premik manj, kot pa je število stolpcev.

3.2 FreeRTOS

Mikrokrmilnik ST32F4 z jedrom ARM Cortex-M4 je idealen za sistem s takojšnjim odzivom. Časovne zahteve takšnih sistemov so lahko nekritične. To pomeni, da sistem še vedno deluje kljub prekoračitvi časovnega okvira izvajanja. Primer tega je neposreden video prenos, kjer prekoračitve ne povzorčijo dejanske negledljivosti, je pa takšno zatikanje lahko moteče.

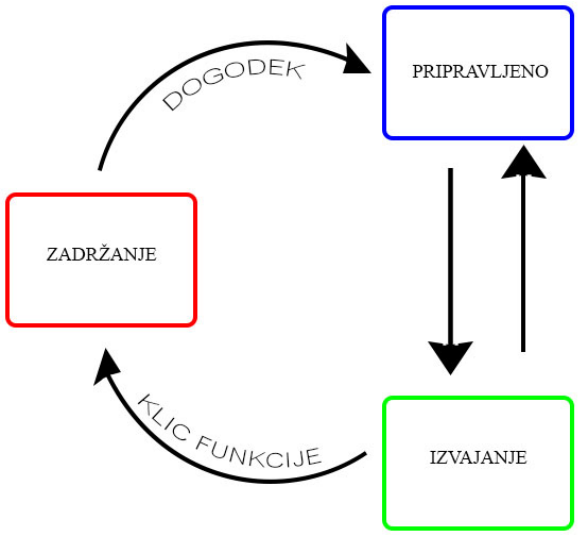
Poznamo še časovno kritične zahteve oziroma zahteve za dosledno delo v realnem času, kjer prekoračitev časovnega izvajanja povzroči odpoved celotnega sistema. Letalski nadzorni sistem ali srčni vzpodbujevalec sta le nekatera izmed njih.

FreeRTOS [15] je časovni razvrščevalnik, ki na mikrokrmilniku omogoča izvajanje kritičnih zahtev sistema s takojšnjim odzivom. Pri tem uporablja niti oziroma opravila, vrste in semaforje. Mikrokrmilnik STM32F4 ima vgrajen sistemski časovnik SYSTICK, ki je vir prekinitev za preklapljanje opravil v FreeRTOS. Opravila so funkcije v programskem jeziku C, ki lahko kot parameter sprejmejo kazalec tipa *void*, vračajo pa ničesar. Na opravilo lahko gledamo kot na manjši samostojen program, ki se izvaja v neskončni zanki, dokler ga ne prekine drugo opravilo višje ali enake prioritete.

Na sliki 3.1 so prikazana stanja, v katerih se opravilo lahko nahaja. Vsa opravila se začnejo v stanju pripravljenosti. Tisto opravilo, ki ima najvišjo prioriteto, se lahko začne izvajati. V primeru dveh opravil z enako prioriteto, se po preteku časovne rezine izmenjujeta v stanjih pripravljenosti in izvajanja.

Za izvajanje opravil z nižnjimi prioritetami imamo dve možnosti. Prva je spreminjanje nivoja prioritet, druga pa, da s pomočjo zakasnitev pošljemo trenutno opravilo na zadržanje. Zadržano opravilo gre po preteku časa v stanje pripravljenosti. Če ima višjo prioriteto takoj zamenja trenutno izvajano opravilo. V stanje zadrževanja se opravilo premakne tudi pri čakanju odgovora vrste ali semaforja. Vrsta se uporabi za komunikacijo med opravili in pri pošiljanju podatkov, semaforji pa služijo za komunikacijo med opravili in prekinitveno servisnimi rutinami.

FreeRTOS za preklapljanje opravil uporablja **prednostno** razvrščanje. To pomeni, da se takoj začne izvajati opravilo z najvišjo prioriteto, pri čemer prekine izvajanje opravila z nižjo prioriteto. Temu se je mogoče izogniti z **vzajemnim** razvrščanjem, saj le-ta dopušča tekočemu opravilu, da se izvede do konca. Opravilo višje priorite mora tako počakati v stanju pripravljenosti. Vzajemno razvrščanje je v tem pogledu enostavnejše, vendar lahko povzroči slabšo odzivnost sistema.



Slika 3.1: **Možna stanja opravil.** Obstaja še stanje IZKLJUČEN vendar na sliki ni prikazano, ker ga nisem uporabil.

```
void automatic(){
      for(i=0;i<37;i++)
      {
        for(j=0;j<60;j++)
        {
          if(L==0)  //samo znacka kdaj levo in kdaj desno
          {
            if(tabFRI[i][j]==1)
            {
              VklopLaser();
            }
            else
            {
              IzklopLaser();
            }
            if(j!=59)
            PremikLevo();
            Delay(7FFFF); //doloca hitrost pisanja 6xF=1s
          }
          else
          {
            if(tabFRI[i][59-j]==1)
            {
              VklopLaser();
            }
            else
            {
              IzklopLaser();
            }
            if(j!=59)
            PremikDesno();
            Delay(7FFFF);
          }
        }
       PremikDol();
      }
  }
```

Koda 3.6: **Samodejni način delovanja.** Program po vrsticah bere vrednosti in glede nanje vklaplja ali izklaplja laser.

3.2.1 Osnovni program prilagojen za FreeRTOS

Za novi program sem napisal naslednja opravila:

- premik v levo in desno za posamezen koračni motor s prioriteto 2,
- vklop in izklop laserja s prioriteto 2,
- začetni ničelni položaj s prioriteto 2,
- branje vhodnih tipk s prioriteto 1,
- samodejni način graviranja s prioriteto 0,
- obdelava prejetih podatkov s prioriteto 1.

Vsega skupaj sem napisal dvanajst opravil. Koda 3.7 prikazuje funkcijo, s katero ustvarimo novo opravilo znotraj glavne metode. Ko sem napisal funkcije za ostala zgoraj napisana opravila, sem nato klical funkcijo *vTaskStartScheduler()*, ki zažene razvrščevalnik v prednostnem načinu. Sedaj za vse klice opravil v samem sistemu skrbi algoritem znotraj FreeRTOS.

```
xTaskCreate(
    vTaskAutomatic,          /* Kazalec na funkcijo opravila*/
  (const signed char*) "Automatic", /* Ime opravila za DEBUGG*/
    configMINIMAL_STACK_SIZE,          /* Sklad*/
    (void*) NULL,          /* Kazalec na parameter opravila */
    tskIDLE_PRIORITY + 0UL,            /* Prioriteta opravila*/
    &xAutoHandle                    /* Spreminjanje prioritete*/
);
```

Koda 3.7: **Funkcija, kjer se ustvari opravilo.** Prioriteta opravila samodejnega načina je na začetku 0. Po pritisku tipke USER_BUTTON se prioriteta zviša za en nivo. S tem postane samodejni način glavno opravilo izvajanja. Prioriteto opravila ročnega načina sem skladno s tem znižal za en nivo.

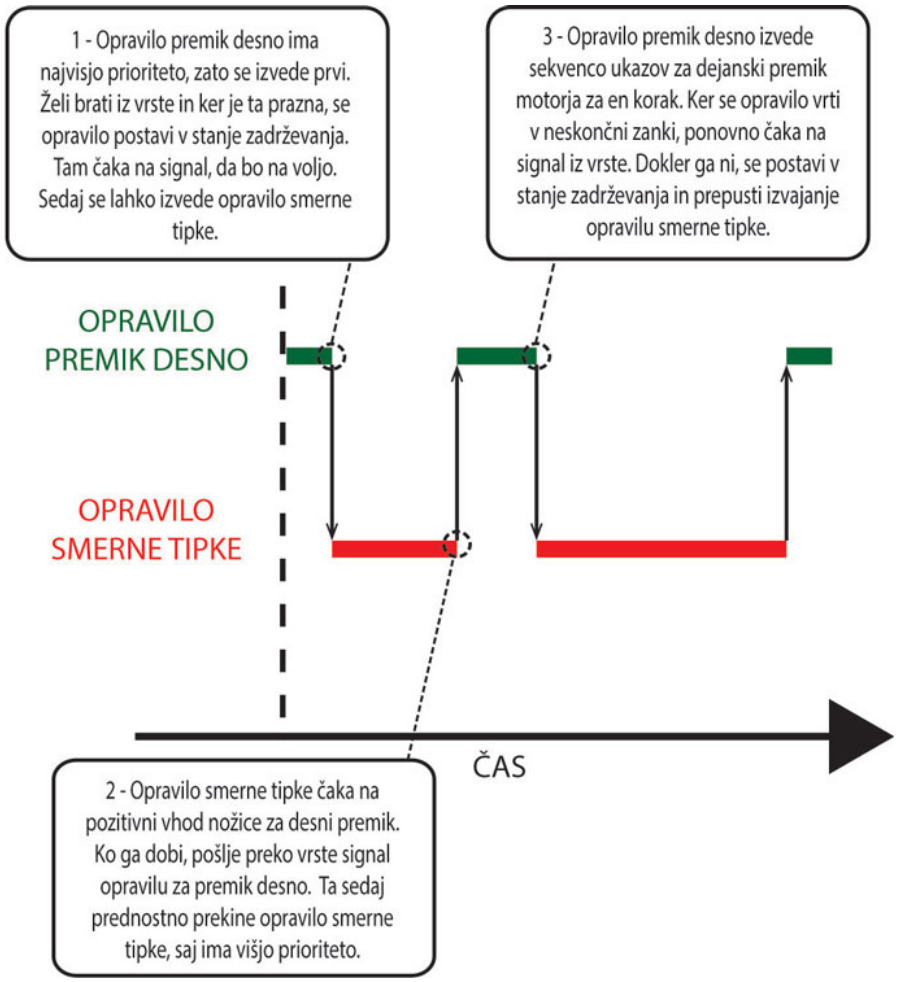
Opravila za premike, delo z laserjem in začetni ničelni položaj, imajo v sistemu najvišjo prioriteto. Do deljenja časovne rezine ne pride, ker sem vsa opravila, razen opravila začetni položaj, postavil v zadržanje. Pri zagonu programa se tako opravilo začetni položaj izvede prvo. Po postavitvi sem mu znižal prioriteto za dva nivoja (koda 3.8).

```
void vTaskInitZeroZero (void *pvParameters){

  portBASE_TYPE xStatusVrste;
  unsigned portBASE_TYPE uxPriority;
  unsigned char ucKontrola; //sprejeti signal za KONTROLO
  /* Dobi prioriteto samega sebe, drugace uporabis TaskHande */
  uxPriority = uxTaskPriorityGet(NULL);

  /* The task is implemented in an infinite loop */
  while (1) {
        zak=0x7FFF; //za vracanje je najhitrejsa mozna hitrost
         while((GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_3))||
              (GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_5)))
          {
            xStatusVrste = xQueueSend(xVrstaKontrolaD1, &
                ucKontrola,0);
            xStatusVrste = xQueueSend(xVrstaKontrolaD2, &
                ucKontrola,0);
          }
          stGD=0;
          stLD=0;
          L=0;
         zak=0x5FFFF; //se vrne na hitrost  za rocno upravljanje
   vTaskPrioritySet(NULL, (uxPriority-2));
  }
}
```

Koda 3.8: **Funkcija opravila začetni položaj nič nič.** Zanka se izvaja, dokler nista oba senzorja sprožena. Nato se ponastavijo vsi števci. Z ukazom **xQueueSend** se pošlje signal določenemu opravilu v zadržanju. V opisani kodi se signal izmenično pošilja opraviloma za premik obeh motorjev v desno.

Naslednje opravilo z najvišjo prioriteto je opravilo za ročni način. V tem opravilu sem sistematično pregledoval vhode smernih tipk, tipke za vklop laserja in vhod tipke USER_BUTTON. Glede na nivo vhoda sem z ukazom *xQueueSend(QueueHandle, &Data)* pošiljal ustrezne signale pripadajočim

opravilom v zadržanju. Slika 3.2 prikazuje primer preklopa med opravili za premik v desno. Na podoben način delujejo ostala opravila za premike in vklop laserja.



Slika 3.2: **Preklop med opravili.** V primeru pritiska dveh tipk naenkrat se opravili za premike izvedeta izmenično. Opravilo smerne tipke je dejansko ročni način vodenja naprave.

Ob pritisku na tipko USER_BUTTON sem izvedel znižanje prioritete opravilu za ročni način. S tem sem poslal v izvajanje opravilo za samodejni način (koda: 3.9).

```
void vTaskAutomatic (void *pvParameters)
{
  uxPriority = uxTaskPriorityGet(NULL);
        vTaskPrioritySet(NULL, (uxPriority+1));
        k=0;

  while (1)
  {
     USART\_puts(USART6, "Vpisi velikost. FORMAT \%08X30!\r\n");

      xStatusVrste = xQueueReceive(xVrstaAutomatic, &
          ucKontrolaSprejeti,portMAX_DELAY);

      /**********************PROGRAM*****************/
     ...
      /*************KONEC PROGRAM***************/
USART_puts(USART6, "KONEC RISANJA\n");
     for (i = 0; i < Y; i++)
      {
        for(j=0; j< X; j++)
        {
          while( !(USART6->SR & 0x00000040) );
          USART_SendData(USART6, *(tabela+i*X+j));
        }
        while( !(USART6->SR & 0x00000040) );
        USART_puts(USART6, "\n");
      }
      vPortFree(tabela );
       uxPriority = uxTaskPriorityGet(NULL);
        vTaskPrioritySet(NULL, (uxPriority-1));
  }
}
```

Koda 3.9: **Funkcija opravila za samodejni način.** Po koncu programa, ki je podoben tistemu v kodi 3.6, sem sliko izrisal še na računalniškem terminalu.

V tem opravilu sem najprej dvignil nivo lastne prioritete. Preko serijske povezave sem poslal na terminal prošnjo za podatek o velikosti slike, nato pa sem čakal na signal vrste iz opravila za obdelavo podatka (koda: 3.10).

```
xStatusVrste = xQueueReceive(xVrstaObdelava, &
    ucKontrolaSprejeti,portMAX_DELAY);
USART_puts(USART6, "V obdelavi!\r\n");

if(xStatusVrste)//ce je sprejeto
   USART_puts(USART6, "Nalozi sliko\n");

Y=(atoi(&size_string[0])*10)+atoi(&size_string[1]);
X=(atoi(&size_string[3])*10)+atoi(&size_string[4]);
tabela = (char*) pvPortMalloc(Y*(X+1)); //allokacija tabele

xStatusVrste = xQueueReceive(xVrstaObdelava, &
    ucKontrolaSprejeti,portMAX_DELAY);
USART_puts(USART6, "Slika nalozena\n");

k=strlen(received_string);
for(i=0; i<k; i++)
  {
    if(received_string[i]!='\n'){
      tabela[m*X+n]=received_string[i];
      n++;
    }
    else{
      m++;
      n=0;
    }
  }
USART_puts(USART6, "Tabela pripravljena\n");
//******SEDAJ VODI AUTOMAIC**********//
xQueueSend(xVrstaAutomatic, &ucKontrola,0);
```

Koda 3.10: **Funkcija opravila za obdelavo podatka.**

Pri opravilu za obdelavo podatka sem najprej počakal na serijski prenos podatka o velikosti slike. Nato sem rezerviral pomnilniški prostor za sliko in počakal na njen prenos. Za alokacijo poskrbi pomnilniški upravljalec FreeRTOS. Podatki o sliki so preneseni kot znakovni niz, ki sem ga za lažjo predstavo pretvoril v tabelo vrstic in stolpcev. Ko je bila tabela pripravljena, sem poslal opravilu za samodejni način signal, da lahko nadaljuje z dejanskim graviranjem slike. Sliko sem nato izrisal še na računalniški terminal. Opravilu za samodejni način sem po končanem graviranju znižal prioriteto, s tem pa sem ga poslal na zadržanje. Prioritetno prednost ima tako ponovno opravilo za ročni način vodenja.

Koda 3.11 je prekinitveno servisni program za sprejem podatkov preko vmesnika USART. Napisal sem preproste pogoje za prepoznavo vhodnega podatka. V primeru da se niz začne z znakom %, se vsi nadaljni znaki prenesejo v spremenljivko ***size_string[]***. V nasprotnem primeru se vsi podatki prenesejo v spremenljivko ***received_string[]***. Po končanem prenosu sem dodal pogoje za sprožitev signalov preko vrste za opravilo obdelave.

```
void USART6_IRQHandler(void){
 // check if the USART6 receive interrupt flag was set
 if( USART_GetITStatus(USART6, USART_IT_RXNE) ){

   static long cnt = 0; // stevec znakov
               static int u=0;  // znacka kateri string
   char t = USART6->DR; // znak iz data-reg USART6 gre v t
 /*znak % doloca da je podatek o velikosti*/
   if( (cnt < MAX_STRLEN) && (t=='%'))
                {
                  size_string[cnt] = t;
                  u=1;
                  cnt++;
                }
                else
                {
                if(u==0)
                  {
```

```
                received_string[cnt] = t;
                cnt++;
                    if(cnt==(Y*(X+1)))
                      xQueueSend(xVrstaObdelava, &
                          ucKontrola,0);
              }
            else
              {
                size_string[cnt] = t;
                cnt++;
                if(t=='\r')
                  {
                    xQueueSend(xVrstaObdelava, &ucKontrola
                        ,0);
                    USART_puts(USART6, size_string);
                    cnt=0;
                    u=0;
                  }
              }
```

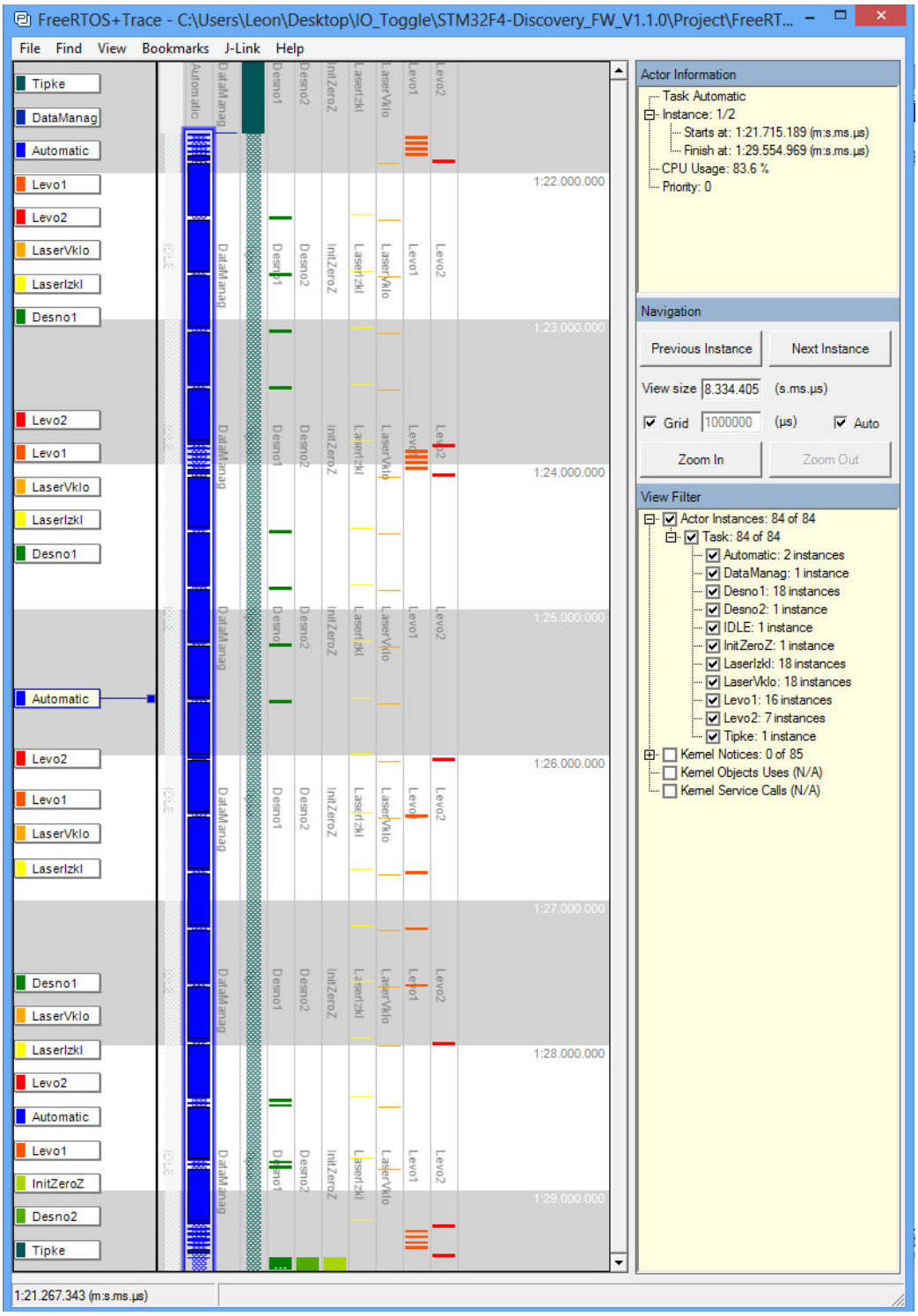
Koda 3.11: **Prekinitveno servisni program za UART6.**

3.2.2 Razhroščevanje opravil

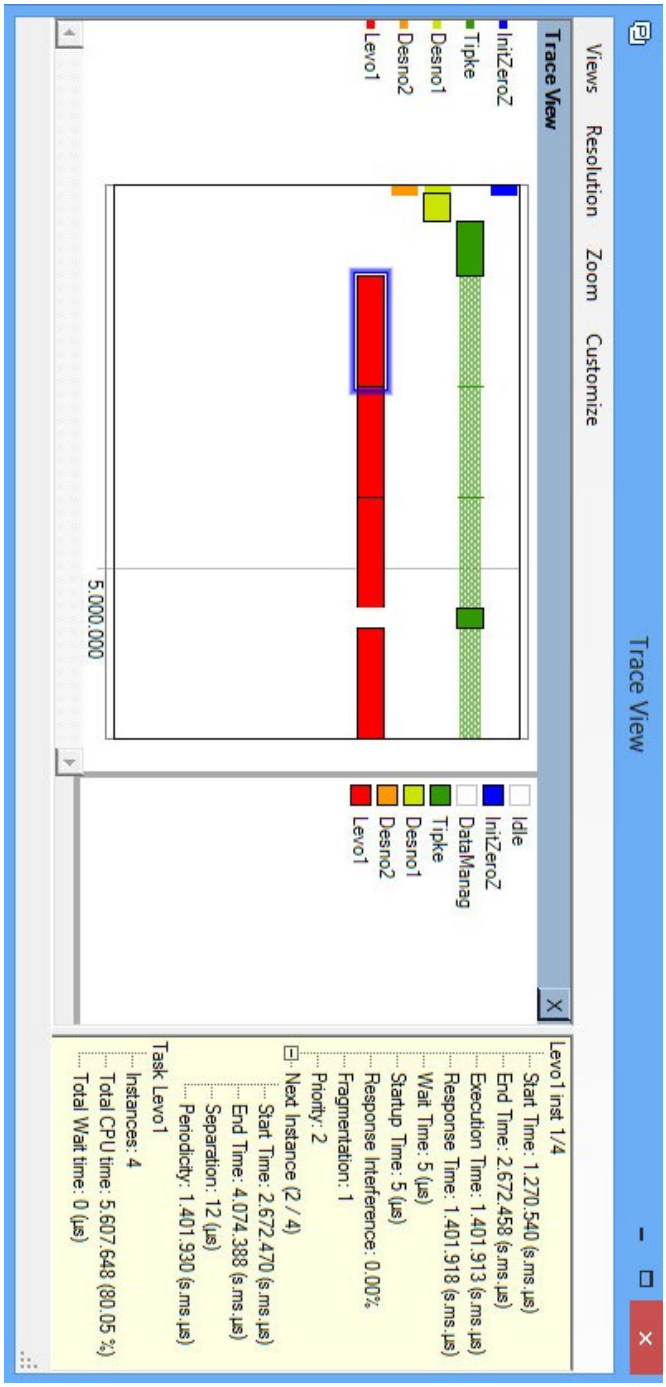
FreeRTOS+Trace [21] je program za grafično analizo sledenja opravilom. V prosti verziji nam program FreeRTOS+Trace pokaže kolikokrat se je posamezno opravilo izvedlo, koliko časa je minilo od začetka in do konca opravila, kakšna je njegova prioriteta in koliko procesorskega časa je izbrano opravilo porabilo.

Z razhroščevanjem in analizo podobnih grafov, kot je ta na sliki 3.3, sem dobil boljši vpogled v točno preklapljanje med določenimi opravili. S spreminjanjem nivoja prioritet posameznim opravilom in z ustvarjanjem vrst kot signal, da naj se neko opravilo izvede, sem dosegel preklope opravil v točno določenih časih. V samodejnem algoritmu sem ukaze *Delay()* zamenjal z ukazi *vTaskDelay(ticks)*, saj mi omogočajo, da vrednosti zakasnitev zapišem v milisekundah ali pa s številom prekinitev števca SYSTICK. Točni časovni preklopi se poznajo tudi na končnem izdelku graviranja.

Slika 3.3 prikazuje del časovnega poteka delovanja različnih opravil v samodejnem načinu. Na začetku poteka opravilo za preverjanje tipk, kar označuje temno zelena barva. Zatem sledi opravilo za samodejni način, ki je označen z modro barvo. To opravilo se nato izmenično prekinja z opravili prvega koračnega motorja za premike v levo in desno, ter z opravili za vklop in izklop laserja. Vsa omenjena opravila so označena z rdečimi, oranžnimi in rumenimi odtenki barv. FreeRTOS+Trace med drugim ponuja vodoraven prikaz časovnega poteka z dodatnimi informacijami. Primer na sliki 3.4 prikazuje ročni način delovanja, kjer sem štirikrat pritisnil tipko za levi premik prvega koračnega motorja. Pred opravilom preverjanja tipk se je najprej izvedlo opravilo za ničelni položaj, ki je sprožilo opravili za premike v desno. Pri tem je prvi motor potreboval več časa za vrnitev v izhodišče, saj je bil od njega bolj oddaljen kot drugi motor. Za prve tri premike sem tipko držal, jo spustil, ter ponovno pritisnil. Na grafu je to prikazano z rdečo barvo. S klikom na enega od opravil, oziroma barvnih kvadratov, se nam na desni strani izpišejo bolj podrobni podatki o začetku izvajanja opravila, koliko časa se je izvajalo, kakšen je bil odziven čas, kdaj se ponovi isto opravilo ipd.



Slika 3.3: **FreeRTOS+Trace.** Prikazan je del časovnega poteka delovanja različnih opravil pri samodejnem načinu delovanja naprave. Časovna sled poteka od zgoraj navzdol.



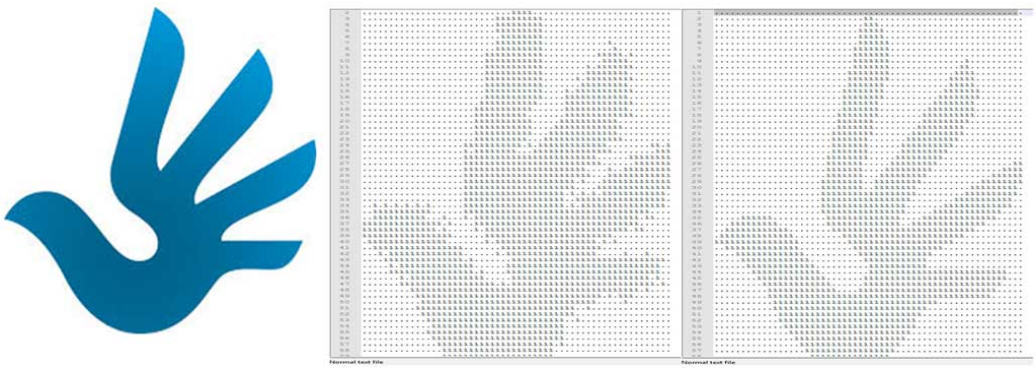
Slika 3.4: **FreeRTOS+Trace vodoravni pogled.** Primer ročnega načina delovanja naprave.

Poglavje 4

Priprava in potek graviranja

Najprimernejše slike za graviranje so emblemi raznovrstnih organizacij ali podjetij. Večinoma so enostavnih in prepoznavnih oblik in vsebujejo malo različnih barv. Manj kot je barv, lažje je pripraviti sliko in boljši je končni rezultat graviranja.

Izbrano sliko sem najprej pomanjšal na velikost ne večjo kot 60×60 slikovnih točk. Programi za delo s slikami omogočajo spreminjanje velikosti slike, vendar se pri tem lahko po kakovosti zelo razlikujejo. Preprost emblem in končni rezultat pomanjšanja prikazuje slika 4.1.



Slika 4.1: **Kakovost zmanjševanja slike.** Izvirna slika je emblem organizacije za človekove pravice [16].

Na sredini je izvirna slika pomanjšana s programom Microsoft Paint, desni

del pa prikazuje veliko boljši algoritem programa Photoshop [24]. Težava je predvsem v absolutni velikosti slike, saj je ta razmeroma majhna. Zaradi tega nisem izbiral embleme z veliko podrobnostmi, ker le-te po obdelavi niso vidne.

V programskem jeziku Java sem napisal program, ki mi pomanjšano sliko pretvori v črno-belo in zapiše sliko v obliki ničel in enic. Z enačbo 4.1 sem za vsako slikovno točko izračunal odtenek sivine.

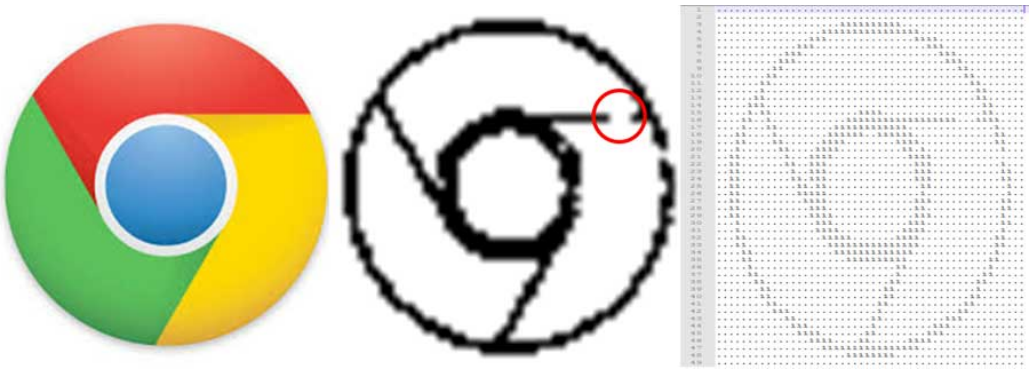
$$siva = ((0.299 * red) + (0.587 * green) + (0.114 * blue)) \tag{4.1}$$

S tem sem dobil vrednosti od 0 do 255, pri čemer je 0 črna barva in 255 bela barva. Za prag sem določil vrednost 128. Izkazalo se je, da je ta vrednost praga v večini primerov najboljša, ne pa v vseh. Primer, kjer prava izbira vrednosti za prag ni lahko določljiva, je viden na emblemu spletnega brskalnika Chrome (Slika 4.2). Težave povzroča večje število barv, njihova postavitev in sence oziroma osvetlitve določenih delov.

Slika 4.2: **Pretvorba izvirne slike v črno-bele odtenke.** Prva pretvorba je narejena pri privzeti vrednosti praga 128. Za drugi primer je bila vrednost povečana na 192. V obeh primerih je slika za graviranje nesprejemljiva.

Takšne slike je mogoče ustrezno pripraviti z določanjem robov. To je postopek, pri katerem z matematičnimi metodami poiščemo ostre spremembe v

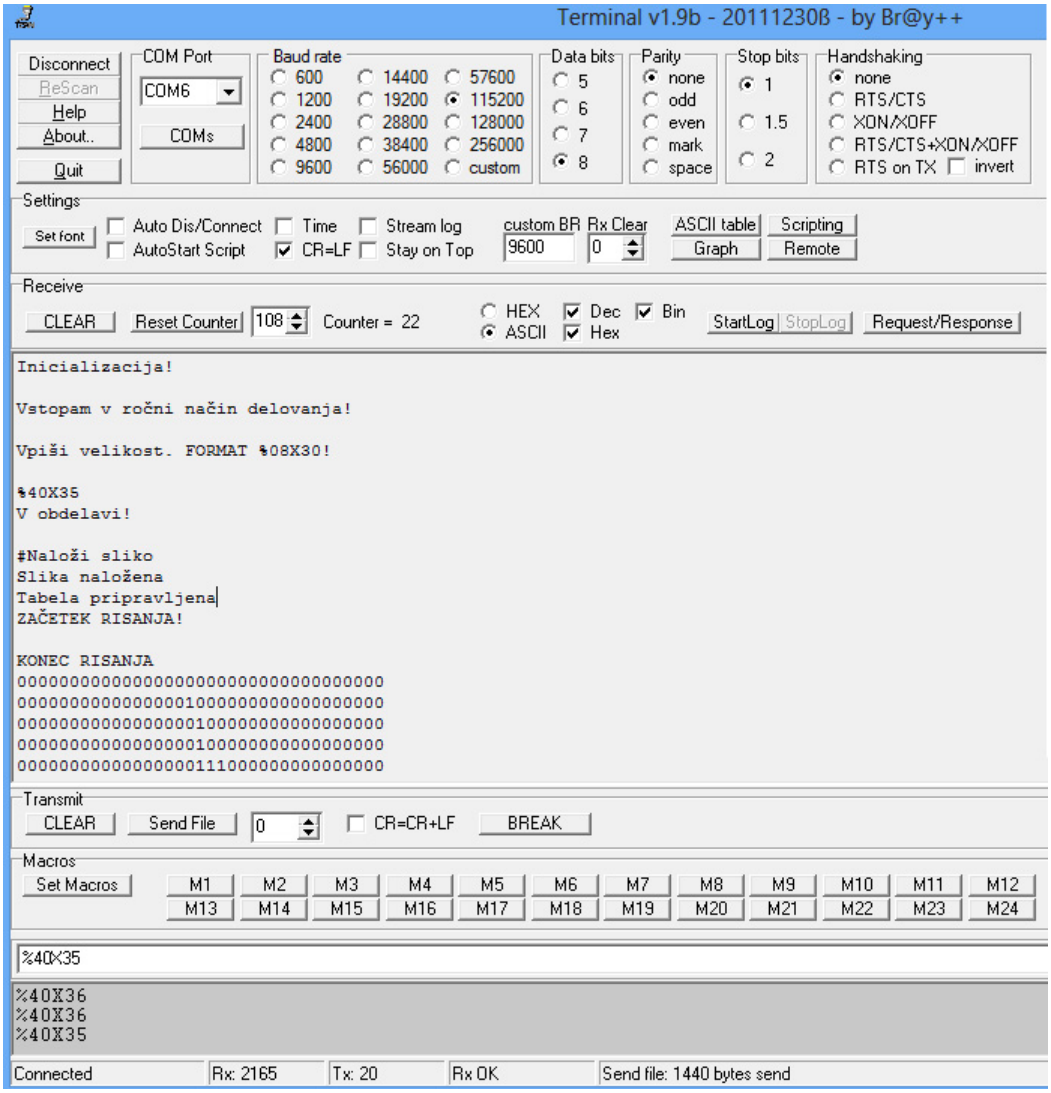
svetlosti slike. Na sliki 4.3 sem uporabil vgrajen filter *Stylize – Find Edges* programa Photoshop.



Slika 4.3: **Pretvorba izvirne slike s postopkom določanja robov.** Na označenem delu je vidna težava tega postopka. Lahko se zgodi, da robovi niso zvezni ali pa so napačni. V teh primerih je potrebno ročno popraviti sliko ali že pretvorjeno datoteko ničel in enic.

Ko sem bil zadovoljen s končnim izgledom slike, sem sivim slikovnim točkam, ki so bile manjše od praga, dodelil vrednost 1, večjim vrednostim pa 0. Slika je bila v obliki tekstovne datoteke pripravljena na prenos.

Za serijski prenos tekstovne datoteke sem uporabil program Terminal [20]. Na sliki 4.4 je viden izpis poteka programa. Najprej se izpiše pozdrav v ročnem načinu delovanja. Po pritisku tipke za samodejni način Terminal zahteva vpis podatka o velikosti. Format podatka je **%40x35**, kjer prva številka predstavlja število vrstic, druga pa število stolpcev.
Znak procent (%) pred podatkom sem dodal zaradi ločevanja med podatki o velikosti in podatki v tekstovni datoteki. Ko se slika uspešno prenese v RAM mikrokrmilnika, se prične graviranje. Po koncu graviranja sliko izrišem še na terminal. Nato se vrnem v ročni način vodenja, kar sem tudi izpisal na terminal.



Slika 4.4: **Terminal.** Zgoraj so vidne osnovne nastavitve, kot so baudna hitrost, število podatkovnih bitov, pariteta, število stop bitov in usklajevanje. Na osrednjem delu se izpisujejo prebrani podatki, spodnji del pa je namenjen pošiljanju. Na voljo je pošiljanje makrojev, datotek ali klasičnih nizov znakov.

4.1 Rezultat graviranja

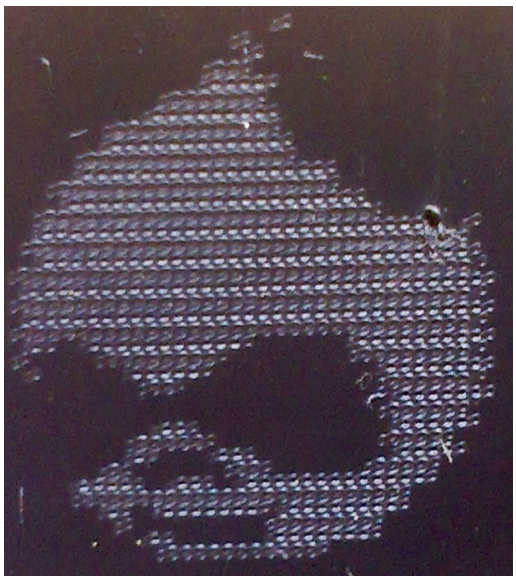
Na sliki 4.5 vidimo razliko končnih izdelkov graviranja med obema prototipoma. Na primeru (**4.5a**) se opazijo težave zaradi časovnih zamikov motorjev. Ti so povzorčali neželjene vibracije, ki so se zaradi same konstrukcije še okrepile. Rezultat vsega skupaj je neravno, poševno graviranje. Primer (**4.5b**) prikazuje bolj pravilno graviranje izvirne slike, saj sem z uporabo boljše konstrukcije prototipa 2 te težave odpravil. Seveda pa je, subjektivno gledano, svojevrsten vzorec na primeru (**4.5a**) lahko popolnoma korekten.

Posledice napačnih zakasnitev se dobro vidijo na robovih primera (**4.5c**). Pri premikih v naslednjo vrstico so se zaradi neusklajenosti začele pojavljati vrzeli na začetku nove vrstice. Zamik posamezne vrstice je sicer majhen, vendar se sosednji vrstici ne zamikata v isto smer in zato na koncu nastanejo opazne razlike. Rešitev je uporaba časovnih zakasnitev sistema FreeRTOS, kot je ukaz *vTaskDelay()* znotraj algoritma za samodejni način delovanja (**4.5d**). Lepo viden je vzorec prižiganja in ugašanja laserja. Končni izdelek je pravilno točkovno vgraviranje izvirne slike glede na vhodni tok ničel in enic.

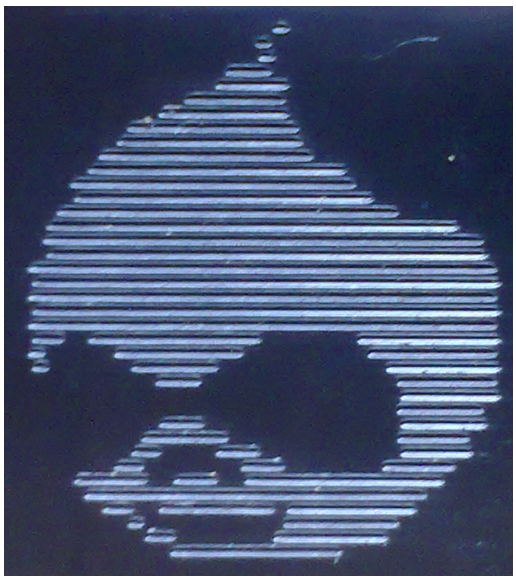
Za nadaljno uporabo sem izbral prototip 2. Boljša in natančnejša konstrukcija mi omogoča manj tresenja in popačitev na samem vgraviranem materialu. Z uporabo sistema FreeRTOS, opravil in njihovo sinhronizacijo je končni rezultat točno tak, kot sem želel.

Najboljši material za graviranje je takšen, ki absorbira največ sevalne energije. V tem primeru je to črna, zato sem uporabil plastične ovitke za CD oziroma DVD. Graviranje je sicer opazno tudi na ostalih temnejših ali svetlejših barvah, vendar se temu sorazmerno poveča čas ostajanja na isti lokaciji, kar z drugimi besedami pomeni manjša hitrost prehoda med točkama.

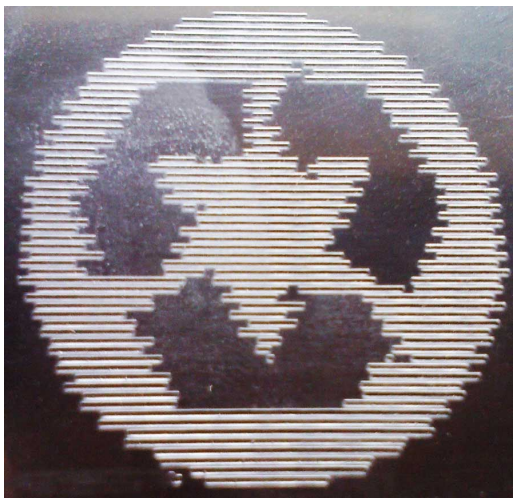
Zadovoljive rezultate dobim tudi pri graviranju na les. Najbolj primerna je balza ali kakšen drug mehkejši les. Za graviranje aluminija in ostalih kovin je laser preslab.



(a) Napake zaradi vibracij.



(b) Boljša konstrukcija.



(c) Zamiki na robovih.

(d) Boljša časovna uskladitev.

Slika 4.5: **Primeri obeh naprav.** Rezultati prototipa 1 so prikazani na primerih (**a**) in (**c**). Boljša rezultata (**b**) in (**d**) pa sta narejena na prototipu 2.

Poglavje 5

Sklepne ugotovitve

Cilj diplomske naloge je bil načrtovati preprosto napravo za lasersko graviranje v plastiko. Pri tem je bilo priporočljivo uporabiti mehanske dele osebnega računalnika. V mojem primeru mi je to v veliki meri uspelo, saj sem primerno uporabil različne konektorje, vodnike, nožice, kondenzatorje, motorje in diode. Veliko koristnih materialov sem porabil za samo strukturo prototipa, tako da so bili celotni stroški projekta minimalni. Največji strošek so bila zaščitna očala, ki so pri takem delu nujno potrebna. Z uporabo mikrokrmilnika STM32F4 in načrtovanjem v odprtokodnem operacijskem sistemu za delo v realnem času FreeRTOS, sem dosegel ustrezne časovne zahteve za preklope med opravili.

Pri vsem tem ni šlo brez težav. Že na začetku sem imel dilemo, kje in kako začeti. Ker nisem imel nobenega pravega načrta, sem naredil dva prototipa, pri čemer sem z izvedbo drugega odpravil pomankljivosti prvega. Prvi prototip je imel težave z neželjenim tresenjem navpičnega nosilca. Teža letega in vztrajnost pri premiku je povzročala nihanje, kar sem rešil z izdelavo drugega prototipa. Drugi prototip ima navpični in vodoravni nosilec ločen in dodatno utrjen z distančniki in oporniki. Druga velika težava je bila iskanje pravih informacij, kot so nazivna napetost, moč ali fizične lastnosti. Za določene elemente se podatki o njih najdejo takoj, za bolj pomembne, kot sta laserska dioda in koračni motor, pa je uporabnih podatkov malo ali nič.

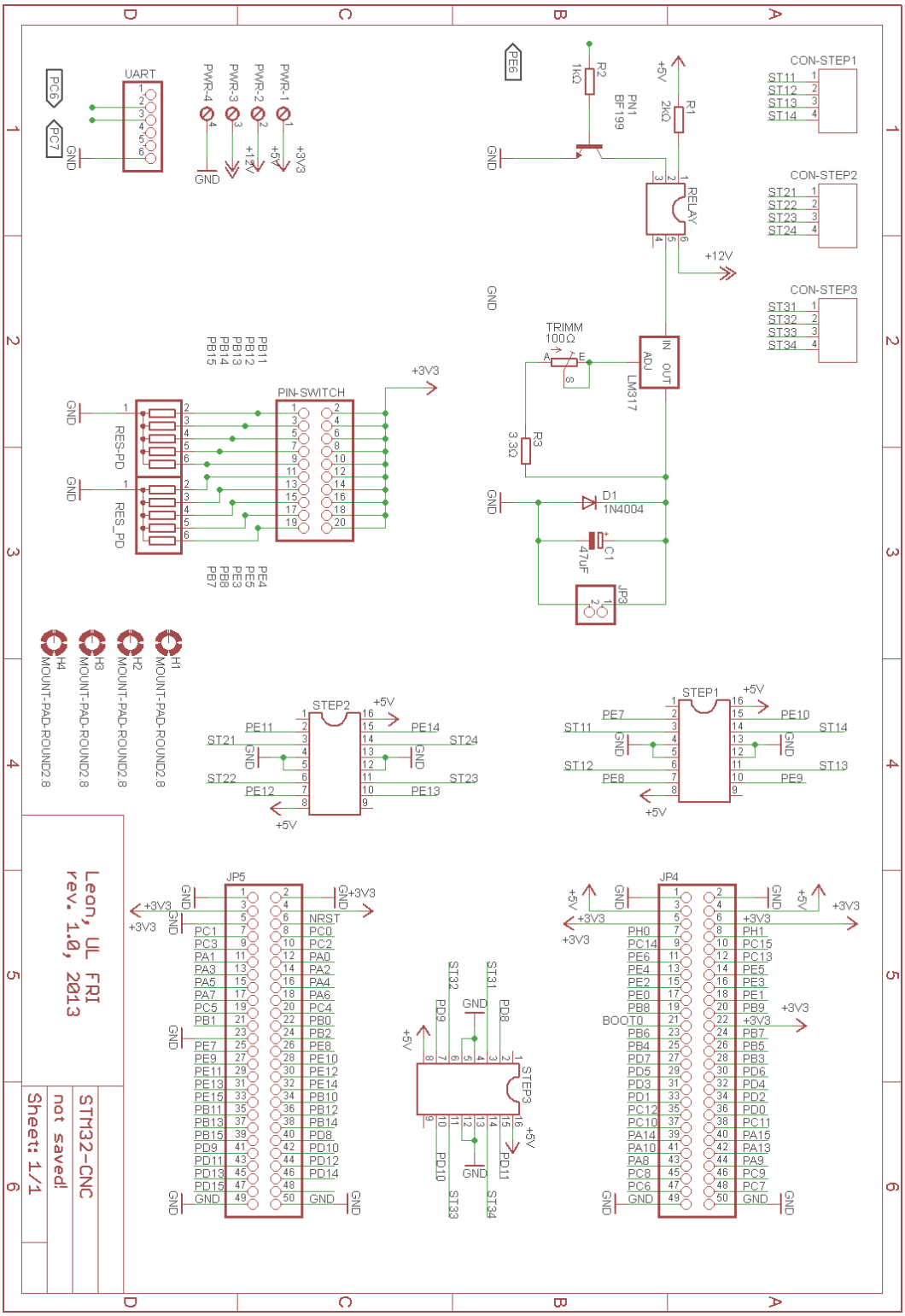
V primeru laserske diode sem se opiral na rezultate eksperimentov ostalih preizkuševalcev. Za podatke o koračnih motorjih sem enega od njih razstavil, kjer sem preštel število polovih parov za določanje velikosti kota pri premiku. Z digitalnim multimetrom sem pomeril upornost tuljav, ter tako določil ustrezno napetost za krmiljenje koračnih motorjev, ipd. V takšnih primerih sem se opiral tudi na znanje pridobljeno na fakulteti. Težave sem imel tudi pri nedelovanju celotnega sistema, kjer je bilo včasih težko ugotoviti, ali je napaka na programski ali na strojni opremi. Večina rešitev je prišla s časom ali s ponovno presojo napake, nemalokrat pa je bilo dobro preveriti tudi stike na testni ploščici in konektorjih.

Kot večina naprav tudi ta ni brez pomanjkljivosti. Prva najbolj opazna je majhna delovna površina. Cenovno ugodno in hitro rešitev ponuja zastarel tiskalnik, saj vsebuje dva večja in močnejša koračna motorja. Z uporabo vodil v tiskalniku bi lahko trenutno velikost površine povečal za osemkrat in tako izboljšal končno ločljivost graviranega izdelka. Druga pomankljivost je moč laserja. Moj laser je najbolj primeren za graviranje temnih plastičnih površin. Za graviranje kovin in lesa ali uporabo laserja za razrez bi le ta moral biti močnejši. To pa predstavlja precejšen strošek. Pomanjkljivosti se najdejo tudi v programski opremi, saj je lahko graviranje po vrsticah precej neučinkovito. Če ima pretvorjena slika veliko ničel, pomeni da so vsi premiki po praznem prostoru odveč. Rešitev je graviranje samo po enicah, ki bi jih program predhodno iskal. Programsko kodo bi lahko spremenil tudi za razčlenjevanje *G-kode*, s čimer bi omogočili uporabo programov CAD CAM. Navsezadnje je mogoče napisati prijaznejši uporabniški vmesnik ipd.

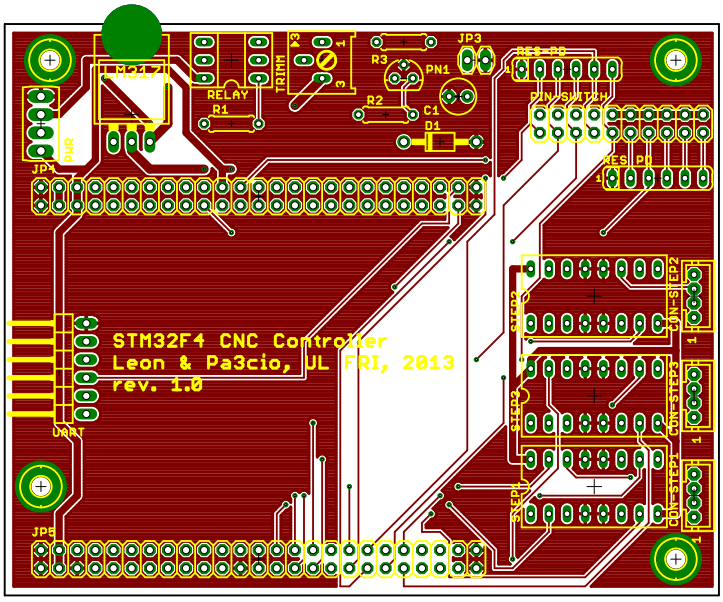
Pri pisanju diplomske naloge sem se veliko naučil o pisanju programov za mikrokrmilnik in o sestavnih delih naprav, ki jih vsakodnevno uporabljam. Zanimivo je bilo spoznati inženirske rešitve tehničnih problemov in delovanje posameznih komponent. Kljub teoretičnemu znanju te praktično delo vedno preseneti. Uničen upor ali tranzistor, nedelovanje sistema, nepazljvost so napake, iz katerih sem se največ naučil. Popestrile so mi delo in dobro je, da so se zgodile.

Priloga

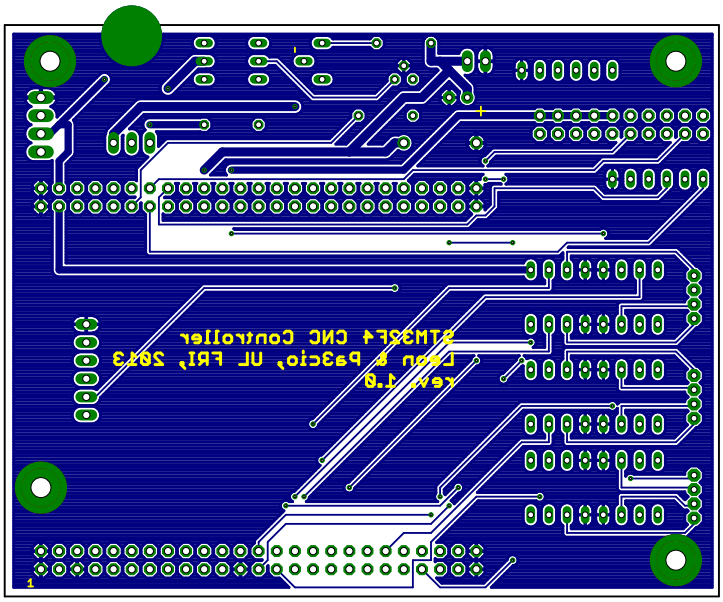
Na sliki 5.1 je celotna shema vezja, iz katere sem izrisal sliko za tiskano vezje (sliki 5.2 in 5.3).



Slika 5.1: **Shema celotnega vezja.**



Slika 5.2: **Tiskano vezje zgoraj.**



Slika 5.3: **Tiskano vezje spodaj.**

Literatura

[1] (2013) Linearni tokovni generator LM317. Dostopno na:
http://www.ti.com/lit/ds/symlink/lm117.pdf

[2] (2013) Bitmap Dostopno na:
http://en.wikipedia.org/wiki/Bitmap

[3] (2013) Numerical control (CNC). Dostopno na:
http://en.wikipedia.org/wiki/Numerical_control

[4] (2013) Laser safety class 3B. Dostopno na:
http://en.wikipedia.org/wiki/Laser_safety#Class_3B

[5] (2013) G-code, language for numerical control. Dostopno na:
https://en.wikipedia.org/wiki/G-code

[6] (2013) Laser Diode Power Output Based on DVD-R/RW specs.
Dostopno na:
http://elabz.com/laser-diode-power-output-based-on-dvd-rrw-specs/

[7] (2013) Laser pointer forums. Dostopno na:
http://laserpointerforums.com/f50/max-current-58753.html

[8] (2013) Laser module. Dostopno na:
http://www.aixiz.com/store/product_info.php/cPath/59/products_id/301

[9] Giorgos Lazaridis, "How stepper motor work", May 2010
Dostopno na:
http://www.pcbheaven.com/wikipages/How_Stepper_Motors_Work/

[10] (2013) Farnel prodaja elektronske opreme. Dostopno na:
http://farnell.com/

[11] (2013) Datasheet circuit L293D. Dostopno na:
http://www.datasheetcatalog.org/datasheet/texasinstruments/l293d.pdf

[12] (2013) Power Supply unit. Dostopno na:
http://en.wikipedia.org/wiki/Power_supply_unit_(computer)

[13] (2013) STM32Discovery review. Dostopno na:
http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419

[14] (2013) Ladyada, FTDI friend (USART Hardware). Dostopno na:
http://learn.adafruit.com/ftdi-friend

[15] (2013) Real Time Engineers Ltd., OS za vgrajene naprave FreeRTOS
Dostopno na: http://www.freertos.org/

[16] (2013) Logo Human rights Dostopno na:
http://www.humanrightslogo.net/en/terms-and-conditions

[17] Douglas W. Jones Microstepping of Stepping Motors, 1998
Dostopno na: http://homepage.cs.uiowa.edu/ jones/step/micro.html

[18] (2013) Physics of a real stepper motor
Dostopno na: http://www.societyofrobots.com/member_tutorials
/book/export/html/124

[19] (2013) STM32F4DISCOVERY board firmware package
Dostopno na: http://www.st.com/web/en/catalog/tools/PF257904

[20] Bray, Terminal com port development tool, 1997
Dostopno na: https://sites.google.com/site/terminalbpp/

[21] Percepio AB, FreeRTOS+Trace – Tracealyzer for FreeRTS, 2013
Dostopno na: http://percepio.com/tracealyzer/freertostrace/

[22] IAR Systems, Embedded Workbench for ARM, 2013
Dostopno na:
http://www.iar.com/Products/IAR-Embedded-Workbench/ARM/

[23] (2013) STM32F4-Discovery User Manual
Dostopno na:
http://www.st.com/st-web-ui/static/active/en/resource/technical
/document/user_manual/DM00039084.pdf

[24] Photoshop, 2013 Dostopno na: http://www.photoshop.com/