ŠOLSKI CENTER VELENJE
ELEKTRO IN RAČUNALNIŠKA ŠOLA
TRG MLADOSTI 3, 3320 VELENJE
MLADI RAZISKOVALCI ZA RAZVOJ SAŠA REGIJE

RAZISKOVALNA NALOGA

NAPRAVA ZA NITNO UMETNINO

Tematsko področje: aplikativni inovacijski predlogi in projekti

Avtor:
Tin Mustač, 4. letnik

Mentor:
Jožef Hrovat, dipl. inž.

Velenje, 2024

Raziskovalna naloga je bila opravljena na Šolskem centru Velenje, Elektro in računalniška šola.

Mentor: Jožef Hrovat, dipl. inž.

Datum predstavitve: Marec 2024

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD Elektro in računalniška šola, šolsko leto 2023/2024

KG avtomatizacija/ umetnina/ nit/ vrv/ nizkocenoven/ izdelava robota

AV MUSTAČ, Tin

SA HROVAT, Jožef

KZ 3320 Velenje, SLO, Trg Mladosti 3

ZA Elektro in računalniška šola

LI 2024

IN **NAPRAVA ZA NITNO UMETNINO**

TD Raziskovalna naloga

OP VII, 74 str., 89 sl., 36 vir., 5 pr.

IJ SL

JI sl/en

AI Če se osredotočimo na področje umetnosti, robotika in avtomatizacija nista videti kot njena neposredna sopotnika. Pogosto se misli, da je postopek izdelave umetnine nekaj, kar zmore le človek, ampak to ni res. Umetniški ustvarjalni procesi se v današnji dobi nenehno razvijajo in vključujejo vedno bolj abstraktne metode. Eden od takšnih je uporaba robotov, s katerimi si olajšamo delo izdelovanja. V raziskavi je predstavljen postopek izdelave, delovanje ter končni izdelek enega od takšnih strojev. Vrsta umetnine, ki je ustvarjena v procesu se imenuje »String art« in deluje na principu povezovanja niti med žeblji. Končni izdelek je slika, katere cilj je čim boljše prekopirati originalno sliko. Rezultati dela kažejo, da roboti niso zgolj orodje za simulacijo, ampak imajo sposobnost prevzemanja aktivne vloge pri celotnem procesu izdelave umetniških del, kar predstavlja ključen mejnik v zlitju tehnologije in umetnosti.

KEY WORDS DOCUMENTATION

ND Elektro in računalniška šola, šolsko leto 2023/2024

CX Automatization/ art/ string/ cheap/ constructing robot

AU MUSTAČ, Tin

AA HROVAT, Jožef

PP 3320 Velenje, SLO, Trg Mladosti 3

PB Elektro in računalniška šola

PY 2024

TI **STRING ART MACHINE**

DT RESEARCH WORK

NO VII, 74 pg., 89 fig., 36 ref., 5 ap.

LA SL

AL sl/en

AB If we focus on the field of art, robotics and automation do not appear to be its immediate companions. It is often believed that the process of creating art is something only humans can do, but that is not true. Artistic creative processes are constantly evolving in today's age and involve increasingly abstract methods. One such method is the use of robots, which facilitates the creation process. The research presents the process of creation, operation, and the final product of one of these machines. The type of art created in the process is called "String art," and it operates on the principle of connecting threads between nails. The final product is an image whose goal is to replicate the original picture as closely as possible. The results of the work show that robots are not just tools for simulation but have the ability to take an active role in the entire process of creating art, representing a crucial milestone in the merger of technology and art.

.KAZALO VSEBINE

KAZALO SLIK

KAZALO PRILOG

KAZALO OKRAJŠAV IN SIMBOLOV

WB – Kanvas za nit (angl. »Workpiece board«)

MY – koračni motor Y

MX – koračni motor X

1 UVOD

Sodobni časi prinašajo vedno več avtomatizacije produkcije izdelkov in povečujejo učinkovitost izdelovalnih postopkov. V tem kontekstu sem se odločil avtomatizirati dolgotrajen in monoton tradicionalen način ustvarjanja umetnine iz niti. Naloga se osredotoča na razvoj stroja, ki s pomočjo tehnologije, kot so arduino mikrokrmilnik, koračni motorji, 3D tiskanje, lasersko rezanje, tako kot ročne spretnosti sestavljanja samodejno ustvari izdelek.

Enoličnost in repetitivnost ustvarjanja vzorca iz niti lahko pogosto ovira ustvarjalni proces. Prav tako je ročno ustvarjanje vzorca s podobo slike bolj komplicirano, kot ustvarjanje simetričnih vzorcev. Vzorci, ki so popolnoma simetrični, so pogosto izračunani z matematičnimi formulami in sledijo določenemu zaporedju. Ročno sledenje takšnemu vzorcu se hitro vtisne v mišični spomin, kar omogoča hitrejše in bolj intuitivno delo. Pri vzorcih, katerih končni produkt je komplicirana slika, pa tega mišičnega spomina ne more biti, kajti je zaporedje žebljev, okoli katerega naviješ nitko, navidezno naključno. Zato je za takšno delo najbolj primeren robot, ki lahko hitro in efektivno ustvari tovrsten izdelek.

Pričakovani rezultati ne vključujejo zgolj prostoročne produkcije umetnine, temveč tudi razširitev razumevanja avtomatizacije v umetnosti.

Slika 1 Simetrični vzorec (1)

Slika 2 Kompliciran vzorec (2)

1.1 HIPOTEZE

- Izdelal bom delujočo napravo
- Končni izdelek bo podoben vhodni sliki
- Končni izdelek bo razviden tudi brez ogleda začetne slike
- Stroj bom naredil iz preprosto dostopnih materialov
- Naprava bo preprosta za uporabo

2 CILJ

Cilj naloge je združiti umetnost in avtomatizacijo ter razviti funkcionalnega avtonomnega robota, katerega glavna naloga je ustvarjanje vzorcev iz niti. Ta naloga se bo izvedla s povezovanjem žebljev, postavljenih ob obodu krožnega kanvasa, z uporabo Arduino krmilnila ter koračnih motorjev. Končni izdelek je nitna slika, ki si prizadeva doseči popolno reprodukcijo originalne slike prek prepletene nitne mreže. Ta mreža, sestavljena iz natančno povezanih niti, ustvarja vizualno privlačno in prepoznavno grafiko, ki združuje umetniški izraz s tehnološko inženirsko inovacijo.

Slika 4 Vhodna slika kompliciranega vzorca (35)

Slika 3 Izhodna slika kompliciranega vzorca (34)

3 TEORETIČNE OSNOVE

3.1 NITNE SLIKE

Nitna umetnost (angl. String art), je prepoznavna po razporeditvi barvne ali brez barvne niti, napete med točkami tako, da tvorijo geometrijske vzorce ali preproste reprezentativne oblike kot naprimer drevo, jelen. Nit, žica ali vrv je prepletena okoli žebljev zabitih v krogličnі leseni disk. Ta je tudi pogosto prekrit s kakšnim žametom ali pa pobarvan, da dobimo željeno barvo ozadja. Čeprav je celotna slika sestavljena iz ravnih črt, se zaradi različnih kotov, pod katerimi se nitke sekajo, ustvarja videz Bézierjevih krivulj. S spreminjanjem pozicije žebljev, prav tako lahko dobimo drugačen videz in obliko končnega izdelka.

Slika 6 Nitna slika jelena (4)

Slika 5 Nitna slika drevesa (5)

Izvor umetnosti z nitjo sega v konec 19. st, kjer je to umetnostno obliko izumila Mary Everest Boole, ki je poskušala matematične ideje približati otrokom. Postala je priljubljena dekorativna obrt v poznih šestdesetih letih preko kompletov za ustvarjanje in knjig.

Najnovejša oblika nitne umetnosti in oblika, katero sem jaz uporabil, vključuje tudi računalništvo, kjer lahko ustvarimo fotorealistična umetniška dela. To je prvič bilo predstavljeno s strani Petra Vrellisa leta 2016.

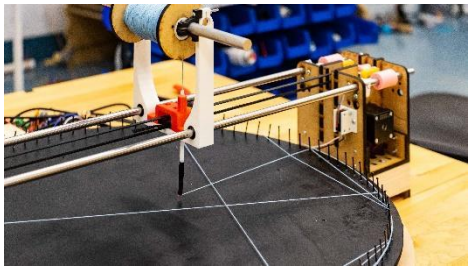
3.2 NAPRAVA ZA NITNO UMETNINO (STRING ART MACHINE)

»String art machine« ali naprava za nitno umetnino je naprava, ki avtomatizirano ustvarja umetniške izdelke z metodo ovijanja napete niti okoli žebljev. Gre za specializirano napravo, ki je na trgu manj prisotna, saj obstaja omejeno povpraševanje. Ocenjujem to na osnovi dejstva, da sem našel le eno spletno stran, ki takšne stroje ponuja (alinedeco). Posledično se večinoma zgodi, da je večina teh strojev narejena doma.

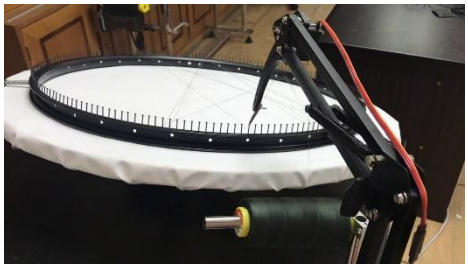
Delovanje takšne naprave je podobno, kot ročno izdelovanje nitne umetnine, le da žeblje poveže robot, ki lahko to opravi hitreje in z večjo preciznostjo. Kakor sem videl, obstajajo tri različne izvedbe naprave za nitno umetnino.

3.2.1 *PRVI NAČIN IZDELAVE*

Prva deluje na principu fiksnih pozicij žebljev, kjer so žeblji enako medsebojno razporejeni na disku. Ta način je v primerjavi z drugim neizmerno lažji in je tudi način, po katerem sem si zadal izdelati svojega. Takšen robot ima eno rotirajočo os, na kateri je kanvas z žeblji ter drugo os, ki je lahko ali linearna ali rotirajoča, ki pa premika roko, s katero ovijemo nit okoli žeblja.



Slika 7 Izvedba 1 (foto: Erika Lu)

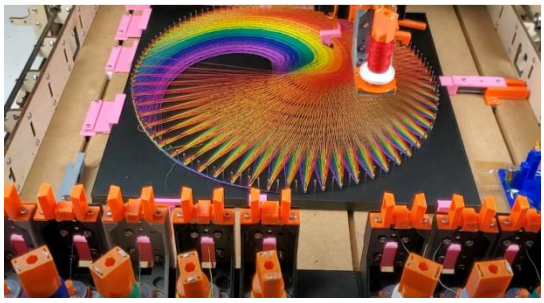


Slika 8 Izvedba 1 (foto: Wonderina)

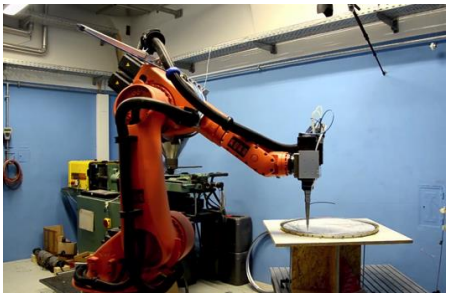
Izvedbe takšnega robota so ustvarili: Barton Dring (6), Alinedeco (7), Wonderina (8), Erika Lu (9), Yoavkleiner (10).

3.2.2 *DRUGI NAČIN IZDELAVE*

Drugi način delovanja deluje zelo podobno kot prvi, z eno ključno razliko: namesto preproste roke z enim motorjem se uporablja robotska roka ali kartezijski robot. Kljub temu da so žeblji še vedno postavljeni na fiksnih pozicijah v krogu, se v tem primeru gibanje ne odvija v samem krogu žebljev, ampak v sami roki.



Slika 10 Izvedba 2 (foto: K. Dluzen)

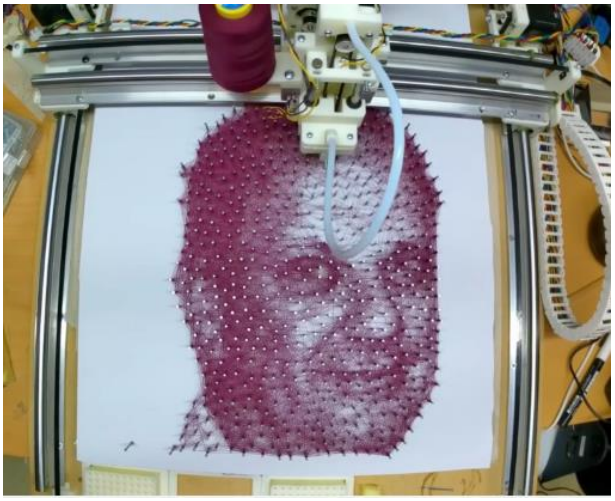


Slika 9 Izvedba 2 (foto: TU Wien)

Izvedbe takšnega robota so ustvarili: Kevin Dluzen (11), TU Wien (12), Marc Sallent (13).

3.2.3 *TRETJI NAČIN IZDELAVE*

Pozicije žebljev v tem načinu niso fiksne, temveč jih generira program in kasneje tudi generiramo, katere povežemo, da ustvarimo sliko. Ti roboti so zelo bolj kompliciran. Kanvas je ponavadi iz stiroporja ali nekega drugega mehkega materiala, v katerega robot sam vtakne žeblje, nato pa te žeblje med seboj poveže. Tipično ima 3 osi: X, Y in Z. S kombinacijo vseh treh osi manevrira med žeblji, da pride do željene točke po točno določeni poti. Slike, ki nastanejo so tudi veliko bolj realistične.



Slika 11 Izvedba 3 (foto: Paul MH)

Izvedbe takšnega robota so ustvarili: Paul MH (14)(15), Laarco studio (16).

3.3 AVTOMATIZACIJA

V predelovalni industriji se avtomatizacija proizvodnih procesov hitro uveljavlja kot standardno orodje. Namesto da bi bila grožnja zaposlenim, je avtomatizacija postala ključno sredstvo za optimizacijo proizvodnje v vsakem podjetju. Poleg tradicionalne uporabe na proizvodnih linijah, lahko avtomatizacija seže tudi na področja, kot so upravljanje inventarja, naročil, nakupov in celo trženja. Ta napredna tehnologija vključuje uporabo programske opreme za upravljanje proizvodnje ter robotskih orodij za izboljšanje fizične proizvodnje. S tehnološkim napredkom se je povečalo število operacij, ki jih lahko posamezno avtomatizirano orodje izvede, kar posledično prispeva k učinkovitejšim in izboljšanim proizvodnim procesom.

3.4 AVTOMATIZACIJA V VSAKODNEVNEM ŽIVLJENJU

Avtomatizacija močno vpliva na vsakodnevno življenje in prinaša številne prednosti. Povečuje učinkovitost časa, na primer s hitrimi plačili prek avtomatiziranih sistemov. Zmanjšuje napake in zagotavlja natančnost v ponavljajočih se nalogah, s tem pa zmanjšuje možnost človeških napak zaradi utrujenosti ali zmedenosti. Ekonomičnost avtomatizacije je očitna, saj nadomešča delo z orodji strojne ali programske opreme, kar bistveno zmanjšuje ročno posredovanje. Dodatna varnost je dosežena zgodnjim zaznavanjem potencialnih groženj s strani avtomatiziranih sistemov, ki hitro ukrepajo pred morebitnimi napadi ali vdori. Povečuje produktivnost na različnih področjih poslovanja, še posebej pri večjih projektih, kjer so zahtevane natančnost in doslednost. Na splošno avtomatizacija prispeva k prihranku časa, zmanjšanju človeškega napora, hitrejšim proizvodnim ciklom in večji natančnosti, kar vodi v večji uspeh in rast.

4 UPORABLJENI MATERIALI

4.1 ARDUINO

Arduino je odprtokodna platforma, ki temelji na prijazni strojni in programski opremi. Plošče lahko sprejemajo različne vhode, kot so svetloba, pritisk na gumb ali sporočila s Twitterja in jih pretvarjajo v izhode, kot so zagon motorja, prižiganje LED-diod ali objavljanje na internetu. Z uporabo programske opreme Arduino (IDE) in programiranja v jeziku Arduino lahko uporabniki dajo svoji plošči natančna navodila za delovanje. Skozi leta je Arduino postal ključen del številnih projektov, od preprostih gospodinjskih pripomočkov, do zapletene znanstvene opreme. Okoli te platforme se je oblikovala globalna skupnost ustvarjalcev, ki so s svojimi prispevki omogočili dostop do obsežnega znanja, koristnega tako za začetnike kot strokovnjake. Sprva ustvarjen kot orodje za hitro prototipiranje na Inštitutu za interakcijski dizajn Ivrea, je Arduino z razvojem prilagodil svojo ponudbo novim potrebam, ki vključujejo nosljive naprave, 3D-tisk, vgrajena okolja in aplikacije interneta stvari.

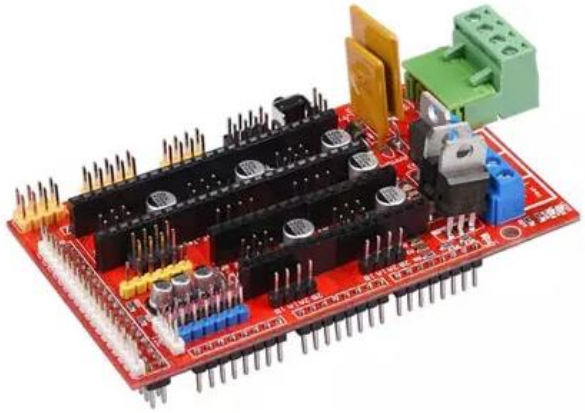
4.1.1 *ARDUINO MEGA 2560*

Arduino Mega 2560 je mikrokrmilniška plošča, ki temelji na ATmega2560. Ima 54 digitalnih vhodno/izhodnih pinov (od katerih jih 15 lahko uporabite kot PWM izhode), 16 analognih vhodov, 4 UARTe (strojne serijske porte), 16 MHz kristalni oscilator, USB povezavo, napajalno vtičnico, ICSP glavo in gumb za ponastavitev. Primerna je za projekte, ki zahtevajo več I/O linij, več pomnilnika za skico in več RAM.

Slika 12 Arduino mega 2560 (20)

4.2 RAMPS 1.4

RAMPS 1.4 je elektronska plošča za krmiljenje 3D tiskalnikov. RAMPS pomeni RepRap Arduino Mega Pololu Shield. To je dodatek za Arduino Mega 2560, ki omogoča povezavo do petih koračnih motorjev, senzorjev, grelnih elementov in drugih komponent.



Slika 13 RAMPS 1.4 (22)

4.3 KORAČNI GONILNIK A4988

A4988 je učinkovit voznik mikrokoračnih motorjev s prevajalnikom, zmožen poganjati bipolarne korakne motorje do 16 mikrokorakov. Ima regulator toka, omogoča delovanje v počasnem ali mešanem načinu propadanja ter enostaven vmesnik za premikanje motorja s samo enim impulzom. Idealno za aplikacije brez kompleksnega mikroprocesorja.

Slika 14 A4988 driver (24)

4.4 KORAČNI MOTORJI

Koračni motorji so vrsta elektromotorjev, ki delujejo na principu diskretnega premikanja, kjer se rotor premika v diskretnih korakih namesto v kontinuirani rotaciji. To dosežejo z ustvarjanjem zaporedja impulzov električnega toka, ki ustvarjajo magnetna polja, ki privlačijo rotor in ga s tem premikajo za določen kot.

Koračni motorji so sestavljeni iz statorskega in rotorskega dela ter navitij. Statorski del vsebuje elektromagnet, ki ga sestavljajo navitja, medtem ko je rotorski del sestavljen iz magnetnega materiala. Električni tok, ki kroži skozi navitja statorskega dela, ustvari magnetno polje, ki privlači rotorski del in ga premika. Premik rotorskega dela se izvede v določenem koraku, odvisno od zaporedja impulzov, ki se uporabljajo.

Koračni motorji so priljubljeni zaradi svoje visoke natančnosti, visokega navora in enostavne krmiljivosti. Poleg tega so zelo zanesljivi in imajo dobro odzivnost. Uporabljajo se v različnih aplikacijah, kot so CNC-stroji, 3D-tiskalniki, avtomatski sistemi za pakiranje, robotika, medicinska oprema in številne druge aplikacije, kjer je potrebna natančna in stabilna kontrola gibanja.

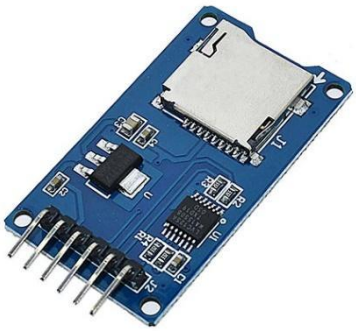
4.4.1 *NEMA 17HS441*

Nema 17hs441 je bipolarni koračni motor z 1,8º korakom (200 korakov / revolucijo) in 40N.cm (56oz.in) zadrževalnim navorom. Motor MX upravljam v konfiguraciji za 1/16th step, zato ima 0.11 º korak (3200 korakov / revolucijo), motor MY pa v konfiguraciji za 1/4th step, zato ima 0.45 º (800 korakov / revolucijo).

Slika 15 Nema 17HS4401 koračni motor (25)

4.5 ARDUINO MODUL ZA SD KARTICO

Arduino modul micro SD kartica je modul za branje in pisanje iz micro SD kartice preko SPI komunikacije. Ima vgrajen 3,3V regulator in pretvornik nivoja signalov iz 5 na 3,3V. (26) Uporabil sem jo zaradi omejenih zmogljivosti pomnilnika (RAM) v Arduino Mega 2560, ki znaša le 8 KB. Ta količina prostora ni zadostna za shranjevanje vseh koordinatnih točk potrebnih za reprezentacijo ene slike. Na primer, že samo ena slika s 3000 točk zavzame približno 10 KB, kar presega trenutno razpoložljivi pomnilniški prostor, jaz jih pa želim imeti več.



Slika 16 Modul micro SD kartica (26)

4.6 LEŽAJI

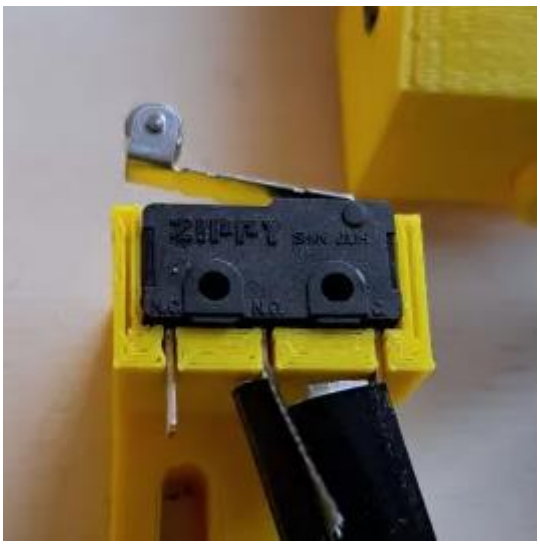
Uporabil sem pet ležajev v konstrukciji. Tri od njih, dimenzij 19 mm x 6 mm, sem vgradil kot kolesa v noge, medtem ko sem enega postavil v sredino za vzdrževanje središča teže. Dodatno sem uporabil ležaj dimenzij 22 mm x 8 mm za izboljšano rotacijo dodatnih kotalnih koles.

Slika 17 Ležaj 19mm (foto: T.Mustač)

Slika 18 Ležaj 22mm (foto: T.Mustač)

4.7 KOTALNO KONČNO STIKALO

Kotalno končno stikalo je vrsta končnega stikala, ki se uporablja za preklapljanje električnega tokokroga, ko se doseže določena meja gibanja. Takšne naprave se uporabljajo med spremljanjem in nadzorom opreme, ki jo je treba redno preverjati za mobilnost. Končno stikalo je nameščeno na konstrukcijah, kjer je potrebna kontrola premikov posameznih elementov. (27) Kotalno končno stikalo ima na vrhu še majhno koleščke, ki omogoča aktivacijo z malo trenja.



Slika 19 Kotalno končno stikalo (foto: T. Mustač)

4.8 VENTILATOR 12V

Majhen 40mx40m 12V ventilatorček je uporabljen, da piha hladen zrak na a4988 gonilnike in jih s tem hladi. Tako preprečim, da se pregrejejo in uničijo. Uporabljam ga namesto izmenjalnika toplote.

Slika 20 Uporabljen ventilatorček (foto: T. Mustač)

4.9 NAPAJANJE / POVEZAVA Z RAČUNALNIKOM

Za napajanje naprave uporabljam DC pretvornik, ki pretvori vhodni izmenični tok v enosmernih DC 12V 8A, ki jih lahko uporabim za napajanja ramps 1.4 ter koračnih motorjev.

Z računalnikom se povežem preko 1.5m dolgega USB A v USB B kabla, s posamezno magnetno zanko, ki omogoča jasnejši in hitrejši prenos podatkov.

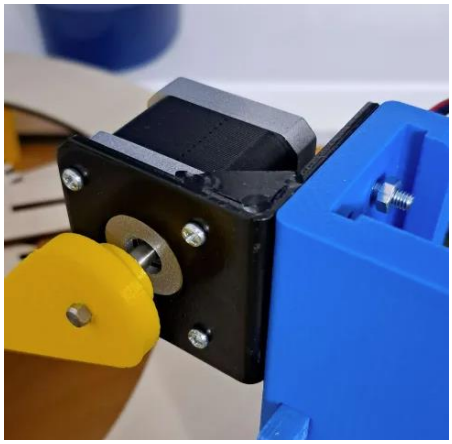
4.10 NAPENJALNIK NITI

Niti napenjam s pomočjo montažne enote za napenjanje niti B2015-372-0A0, ki je primerna za uporabo z JUKI MB-372, MB-373 in MB-377 šivalnimi stroji. Z njo preprečim neželene pojave, kot so povešanje niti in morebitne zataknitve.

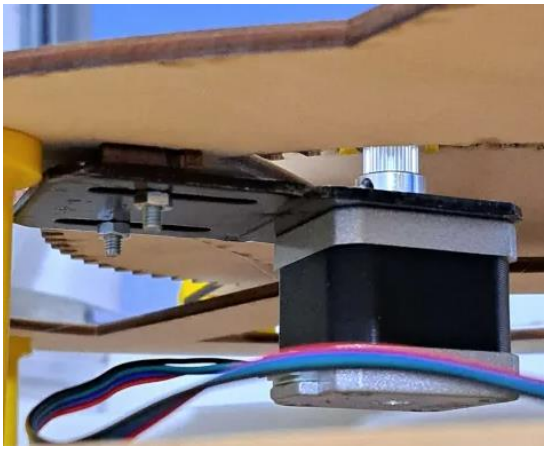
Slika 21 Napenjalnik niti (28)

4.11 DRŽALO ZA KORAČNI MOTOR

Koračna motorja držim z dvema identičnima 90 º držaloma, le da sem enega s pomočjo preše izravnal, da lahko drži motor vodoravno. Z njima lahko tudi hitro in preprosto nastavim točno pozicijo motorja.



Slika 23 Držalo za motor (foto: T. Mustač)



Slika 22 Izravnano držalo za motor (foto: T. Mustač)

4.12 3D TISKANI KOSI

3D-tisk je bil uporabljen za proizvodnjo komponent z različnimi geometrijskimi zahtevami, ki niso bile izvedljive z običajnimi proizvodnimi metodami, kot je rezanje iz vezane plošče. Ta tehnologija je omogočila izdelavo majhnih, kompleksnih in nekonvencionalnih oblikovanih kosov. Skupno je bilo natisnjenih približno 35 različnih delov

4.13 MATERIAL ZA LASERSKO REZANJE

Za vse večje kose, ki jih nisem mogel natisniti, sem jih z laserskim rezalnikom izrezal iz 5mm debele brezove vezane plošče. Vse skupaj je izrezanih 11 kosov, 2 sem moral za vsako sliko ponovno izrezati, kajti sestavljata kanvas, na katerem vlečem niti.

3mm akrilična plastika je bila uporabljena za izrez 72-zobnega zobnika.

4.14 OSTALI MATERIALI

Za električno povezovanje komponent sem uporabil tri različne vrste žičk. Za vzpostavitev povezave med ramps 1.4 in koračnimi motorji, sem uporabil štiri-žični JST PH kabel. Za povezavo z micro SD kartičnim modulom sem uporabil žičke tipa dupont. Za povezavo stikala in končnega stikala sem s spajkanjem povezal konice ženskih dupont žičk z daljšo črno žičko.

Žeblji, ki so postavljeni ob obodu WB so 1.4x25mm kolarski žeblji, izdelani po DIN 1152, ter EN 10230-1 standardu.

Vsi kosi, ki niso trajno pritrjeni, sem jih pritrdil z M 4x20mm vijaki, izdelanimi po DIN 965 standardu.

Za konico roke, sem uporabil GOLD_EYE Tapestry 18/22 igle, katere sem upognil. S tem sem preprečil povijanje niti okoli igle. Te igle imajo dolgo in široko odprtino, kar mi omogoča lažje vstavljanje niti in manj trenja, ko se nit premika.

Slika 25 Uporabljeni žeblji ter vijaki (foto: T.Mustač)

Slika 24 Uporabljene igle (foto: T.Mustač)

5 PROGRAMSKI DEL

5.1 ZAPOREDJE TOČK

Za določanje zaporedja točk, ki jih mora robot povezati v specifičnem vrstnem redu, uporabljam odprtokodni spletni program, ki je dostopen na platformi GitHub. Program je bil razvit pred petimi leti s strani uporabnika z imenom halfmonty, ki ga je objavil in omogočil uporabo kode vsem, ki pridobijo kopijo. Kodo lahko najdete pod virom 29.

Program ima spletni vmesnik (Priloga C), v katerem lahko vidiš, kako naj bi izgledala končna slika. Končni rezultat dobim v obliki texta s številkami ločenimi z vejcami.

Primer: 0, 43, 198, 23, 32, 87, 38, 20, 34, 99, 198, 200 ...

Vmesnik vsebuje sledeče spremenljivke, katere lahko sam nastavljaš: Number of Pins, Number of Lines, Line Weight. Vsaka spremeni izgled končne slike. Sama vhodna slika tudi drastično spremeni izgled končnega izdelka

5.1.1 *SPREMENLJIVKA »NUMBER OF PINS«*

Number of Pins ali število žebljev okoli WB, določa, koliko medsebojno enako oddaljenih žebljev ali točk bo imela slika. Povečanje števila točk pripomore k izboljšanju kakovosti slike, vendar pa tudi pomeni, da bodo točke bolj skupaj, če je WB manjši. Dva različno velika WB lahko imata enako število točk in bosta oba dosegla željeni vzorec, saj se relativne pozicije niti in njihovih presečišč ne spremenijo z večanjem WB. Večje WB vodi le do povečanja razmika med črtami, kar lahko vodi do razbleditve slike.

5.1.2 *SPREMENLJIVKA »NUMBER OF LINES«*

Number of lines ali število črt, ki jih bo povezalo med nitmi, močno vpliva na izgled, čas generiranja točk ter čas izdelovanja slike. Eno črto predstavlja povezava med dvema žebljema. Željeno število črt je predvsem odvisno od števila žebljev/točk. Če je to število preveliko, bo nastala slika imela več podrobnosti, a bo tudi temnejša. Z velikim številom črt in manjšim številom žebljev, lahko nastanejo tudi neželjeni krogličnі vzorci v središču slike. Premajhno število črt pa lahko vodi do slabše prepoznavnosti slike zaradi pomanjkanja podrobnosti. Prav tako bo slika svetlejša. Če je to število zelo premalo, bo nastali vzorec

popolnoma neprepoznaven. Pomembno vlogo pri izbiranju te vrednosti ima tudi sama slika, ki jo želiš reproducirati. Nekatere lahko izgledajo boljše z manjšim številom, nekatere pa boljše z višjim številom.

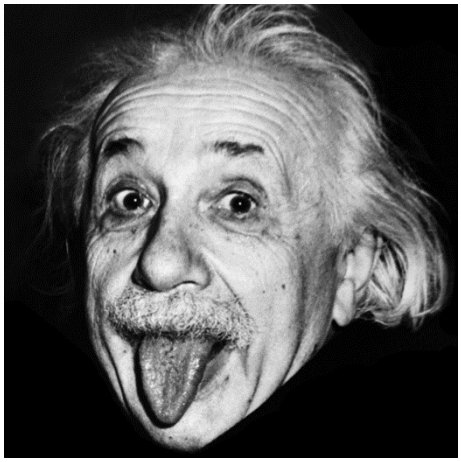
V mojem primeru sem si z 200 točkami in 40 cm premerom WB izbral zgornjo mejo 3000 ter spodnjo okoli 1300. Po lastnih izkušnjah več ali manj kot to, vodi do slabšega rezultata. Zgornja meja 3000 je tudi, ker je to ravno pod 1 km sukanca, kar je lažje pridobiti kot kakšn daljša dolžina.

5.1.3 *SPREMENLJIVKA »LINE WEIGHT«*

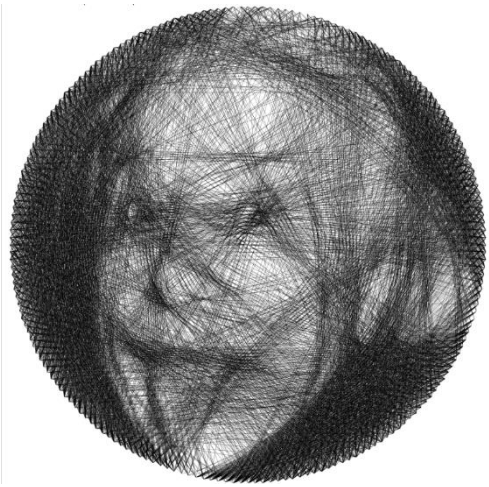
Line weight oziroma debelina črt je spremenljivka, ki spremeni krepkost niti. Za najboljše rezultate jo je najbolje pustiti na vrednosti, na katero je že nastavljena in sicer 20. V spletnem vmesniku spreminjanje te vrednosti ne pokaže nobene vizualne razlike.

5.1.4 *VHODNA SLIKA*

Najpomembnejša spremenljivka je sama vhodna slika. Kontrast, barve, kompliciranost, podrobnost, svetle točke, temne točke, pozicija likov in subjektov na sliki, razlika med sosednjimi barvami, razlika v svetlobi sosednjih barv so vse stvari, ki lahko popolnoma spremenijo izgled nitne slike. Določene slike že od začetka kažejo popolnost, zlasti tiste z enobarvnim ozadjem in velikim kontrastom v liku. Zahteva, da dobra slika izpolnjuje ta pogoj, izhaja iz dejstva, da če se slika pretvori v črno-belo, se opazi znatna svetlobna kontrastnost med osrednjim subjektom in ozadjem, kar preprečuje zameglitev barv.

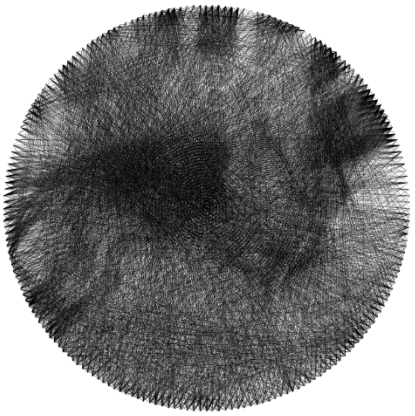


Slika 27 Dobra vhodna slika



Slika 26 Dobra izhodna slika

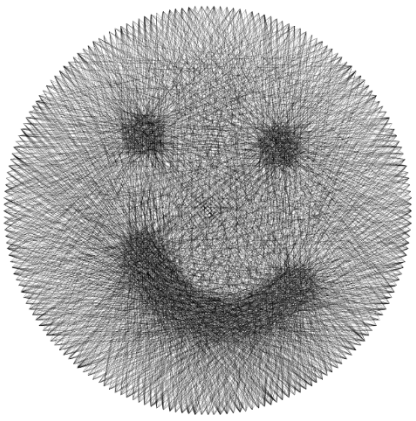
Slika 29 Slaba vhodna slika (foto: Britannia)



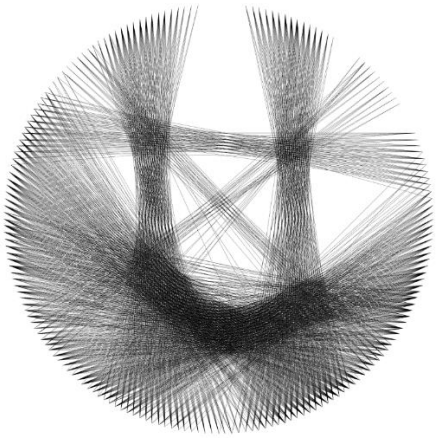
Slika 28 Slaba izhodna slika

Če slika ne izpolnjuje teh pogojev, sem ugotovil, da lahko narediš dobro sliko na enega od dveh načinov: narediš preprosto 2D risbo ali obdelaš sliko v programu za urejanje slik.

1) Vse preproste risbe, ki jih lahko na roko narišeš v programu Paint oz. Slikar izpadejo zelo dobro in jih najbolje narišeš že s samo 1300 črtami, kar je približno 360-470 m niti v mojem primeru. Vendar mora barva ozadja biti svetlo siva, barva, s katero pa rišeš pa temno siva, ker samo bela in črna barva ne izpadeta dobro.



Slika 30 Primer izhoda preproste slike z odtenki sive

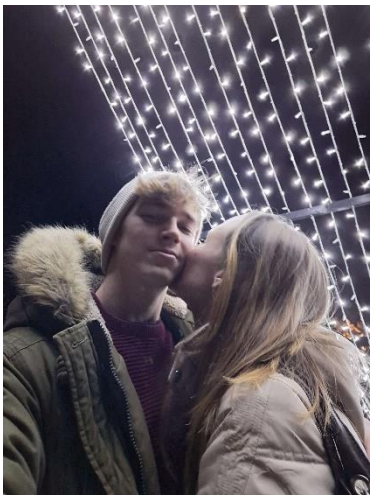


Slika 31 Primer izhoda črno-bele preproste risbe

2) Če željena slika ni takšna, kot bi morala biti, jo je potrebno bolj obdelati v programu namenjenem za obdelavo slik. Lahko se uporabi katerikoli program, a jaz sem uporabil Gimp 2.10.22. Postopki, ki izboljšajo izhodni vzorec so sledeči:

- *Spremeni sliko v črno belo.* Tako bo dosti lažje nadalje urejati sliko.
- *Povečaj subjekt in ga zbližaj.* S tem omogočiš večim podrobnostim, da se ohranijo.

- *Malo podrobnosti.* Najboljši rezultat pride iz slike, ki je zelo jasna in vsebuje malo majhnih podrobnosti. Lastni sliki lahko tudi odstranimo nekatere podrobnosti, tako da preprosto pobarvamo čez njih.
- *Uporabi sharpness filter.* Filter poostri vse črte na sliki in jih naredi bolj vidne, kar pomaga pri ločitvi likov na nitni sliki.
- *Enobarvno ozadje* omogoča čim več podrobnosti na samem subjektu slike. Barva ozadja naj ne bo čisto bela ali čisto črna, temveč nekaj vmes, čeprav včasih gre tudi s črno.
- *Prilagodi kontrast temnih in svetlih območij* na sliki s čopičem, s čimer se prepreči prekomerno izpostavljanje temnih ali svetlih področij na originalni sliki. S čopičem z nasprotno barvo in z nizko stopnjo prosojnosti oz. »opacity«, previdno prehodi skozi svetla/temna območja, ki povzročajo težave.
- *Deli slike, ki se dotikajo roba* oz. kroga v predogledu, naj bodo označeni z barvo ozadja.
- *Kontura* okoli osrednjih subjektov pomaga pri ločitvi lika od ozadja in prepreči meglenost. Priporočljivo je, da je kontura temnejša od ozadja in osrednjega subjekta ter da ima primerno debelino.



Slika 33 Slika pred prilagoditvijo (foto: T.Mustač)

Slika 32 Slika po prilagoditvi

5.2 KODA ZA IZRAČUN PREDVIDENE DOLŽINE NITI

Kode, ki mi pove približek dolžine niti sem napisal v programskem jeziku Python. Koda izračuna to s preprosto trigonometrijo. Najprej preberem .txt datoteko in vse točke prepišem v novo matriko. Nato v *for* zanki, ki se ponovi tolikokrat, kot je členov matrike, s pomočjo

sinusnega izreka izračunam razdaljo med tema dvema točkama in jo prištejem spremenljivki »totalLenght«. Enačba, ki jo uporabljam je sledeča:

$$d = 2r\sin\left(\frac{a}{2}\right) \quad ...(1)$$

d- razdalja med točkama [mm]
r-razdalja med središčem WB in središčem žebljev (v mojem primeru 195mm)
a-kot med dvema žebljema

S pomočjo te kode sem prišel do sledečih ugotovitev:

- 1000 črt ≈ 360m niti
- 1200 črt ≈ 440m niti
- 1300 črt ≈ 470m niti
- 1500 črt ≈ 540m niti
- 1600 črt ≈ 630m niti
- 3000 črt ≈ 920m niti

5.3 KODA V ARDUINO MIKROKRMILNIKU

Program uporabljen na Arduino mikrokrmilniku sem napisal v programu VSCode. Uporabil sem razne programske dodatke, ki so mi omogočili uporabljati ta program, da komuniciram z Arduinom, mu nalagam kodo in jo urejam. Ti »extensioni« so sledeči: Serial Monitor, PlatformIO IDE, Arduino, Arduino-snippets.

Koda prebere točke shranjene na .txt datoteki na SD kartici, preko SPI komunikacijskega protokola in izračuna, za koliko korakov se mora koračni motor obrniti ter v katero smer, da pride WB v najkrajši možni poti v tisto pozicijo. Nato s kombinacijo MY in MX zavijemo nit okoli žeblja. Ta postopek se nato ponavlja, dokler ne pridemo do konca datoteke s točkami, kjer se robot ustavi (Priloga A).

Za komunikacijo s SD spominsko kartico uporabljam knjižnici SPI in SD. Celotna koda je napisana v okoli 420 vrsticah. V Serial monitor konstantno piše na katerem postopku je in

kaj dela. Torej iz katere, na katero točko gre, v katero smer se bo zavrtela in kolikšna je stopnja dokončanosti.

5.3.1 *BRANJE TOČK IZ DATOTEKE*

S funkcijo *readNextPointFromFile()* najprej preverim, če je še kaj točk za prebrati v datoteki. Če sem prišel do konca, zaprem datoteko in vrnem false, da označim konec branja. Z uporabo *.readStringUntil*, preberemo do naslednje točke in s *.trim* odstranim vodilne in končne presledke.

```
bool readNextPointFromFile() {
  if (myFile.available()) {
    // Beri po eno številko iz datoteke
    String rawString = myFile.readStringUntil(',');
    rawString.trim(); // Odstrani vodilne in končne presledke
    rawString.toCharArray(buffer, sizeof(buffer));
    return true;
  } else {
    myFile.close(); // Zapri datoteko, ko ni več številk
    return false;
  }
}
```

Slika 34 Program za branje točk iz datoteke

5.3.2 *ISKANJE REFERENČNE TOČKE*

Pred začetkom vsake uporabe stroj opravi kalibracijo, da zagotovi točnost. Enkratnost prvotne operacije zagotovimo s tem, da kličemo funkcijo v startup zanki, ki se opravi samo enkrat. MX se v pogojni zanki premika za en korak v smeri urinega kazalca, dokler ne aktivira končnega stikala. Ko pride do te pozicije, se zanka zapre in se MX premakne na začetno pozicijo 0. Ker je končno stikalo na levi od pozicije 0, kar pomeni, da je na koraku 529, odštejem njegovo pozicjo *zeroStep* od korakov za totalno revolucijo WB

stepsPerWorkpieceRevolution, da minimiziram pot nazaj do ničle točke. Ko vem, da sem na željeni poziciji, nastavim trenutno število korakov na 0.

```
void findZeroPosition() {
  Serial.println("Iskanje ničelne točke");
  // Premikaj motor, dokler ne najdeš ničle pozicije
  while (digitalRead(zero_pin) != LOW) {
    moveStepperX(1, HIGH);
    delay(10);
  }
  moveStepperX((zeroStep), LOW);
  // Resetiraj trenutni korak
  currentStep = 0;
  Serial.println("Ničelna točka najdena");
  delay(500);
}
```

Slika 35 Enkratno iskanje ničle točke

5.3.3 *PREMIK MOTORJA X*

S funkcijo *moveStepperX(steps,direction)* premaknem MX, za dano število korakov in v dano smer. To naredim s pomočjo for zanke, katero ponovim za število korakov. V for zanki premikam po en korak s pomočjo *digitalWrite()*, s katerim nastavim pin *X_STEP_PIN* na *HIGH*, nato pa zopet na *LOW*. Podobno funkcijo uporabljam tudi za premik MY.

```
digitalWrite(X_STEP_PIN, HIGH);
delayMicroseconds(500);
digitalWrite(X_STEP_PIN, LOW);
delayMicroseconds(500);
```

Slika 36 Premik koračnega motorja X

Dodal sem tudi možnost, da motor pospeši in upočasni, če je število korakov večje kot izbrana vrednost. V njej pospešuje prvih nekaj korakov, nato se giblje enakomerno, dokler ne začne zavirati zadnjih nekaj korakov. Hitrost gibanja spreminjam, s spremembo časovnega zamika *delayMicroseconds()*. Koliko mora ta zamik biti, dobim z uporabo *map()* funkcije, katera prevede vrednost iz enega obsega v drugega.

```
if (acceleration == 1 && steps > 999) {
      if (i < AcceleratedSteps) {
        currentDelay = map(i, 0, AcceleratedSteps, maxDelay, minDelay);
      }
      else if (i < steps - AcceleratedSteps) {
        currentDelay = minDelay;
      }
      else {
        currentDelay = map(i, steps - AcceleratedSteps, steps, minDelay, maxDelay);
      }


      digitalWrite(X_STEP_PIN, HIGH);
      delayMicroseconds(currentDelay);
      digitalWrite(X_STEP_PIN, LOW);
      delayMicroseconds(currentDelay);
    }
```

Slika 37 Pospeševanje motorja X

5.3.4 *RAČUNANJE POTREBNIH KORAKOV DO NASLEDNJE TOČKE*

Koliko korakov potrebujem do naslednje točke je ključnega pomena, kajti moram vedeti, za koliko korakov naj obrnem motor in v katero smer, da pride to te točke. Izračun tega naredim v funkciji *moveMotorToPoint()*. V njej izračunam, v katero smer naj se premakne MX, da pride WB do željene pozicije po najkrajši poti. To je pomembno zato, ker če WB kadarkoli opravi pod daljšo kot 180˚ oziroma polovico rotacije, se bo izvlečena nit, ki je do te točke bila napeta, začela povešati, kar lahko vodi do kakšne zataknitve.

```
dLeft = (currentPoint - targetPoint + numOfPins) % numOfPins;
dRight = (targetPoint - currentPoint + numOfPins) % numOfPins;
...
int direction = (dLeft < dRight) ? HIGH : LOW;
...
// V katero smer bo šlo (low  = dRight (smer kontra uringa kazalca), high =
dLeft(smer uringa kazalca)) (Odvisno od orientacije priključka)
int stepsToMove = min(dLeft, dRight) * stepsPerPoint;
moveStepperX(stepsToMove, direction);
...
currentStep = targetPoint * stepsPerPoint;
```

Slika 38 Izbira smeri rotacije motorja X

dLeft izračuna razdaljo v smeri urinega kazalca od trenutne točke. Od *currentPoint* odšteje *targetPoint*, doda *numOfPins* (da zagotovi pozitivno vrednost) in nato vzame ostanek po deljenju z *numOfPins* (da upošteva krožno naravo vzorca). Enak postopek je z *dRight*, le da od *targetPoint* odšteje *currentPoint.*

V isti funkciji tudi kličem funkcijo za premik motorja *moveStepperX().* Ko pridem do ciljne točke, nastavim trenutno pozicijo na točko, v katero se je WB premaknil.

5.4 POVIJANJE NITI

Funkcija za povijanje niti vključuje premik obeh motorjev. Najprej premaknem MY na drugo stran žebljev, nato premaknem MX za eno mesto v desno oz. za število korakov potrebnih za eno točko nazaj. Če s tem pridem v negativne korake, ali pa prekoračim število totalnih korakov v enem krogu, jih povijem nazaj na začetek, ali pa jih odštejem od totalnih korakov. Nato premaknem MY nazaj na prvotno mesto.

```
void loopNail(){
  moveStepperY(30, HIGH);
  delay(300);
  moveStepperX(stepsPerPoint, HIGH);
  currentStep -= stepsPerPoint;
  if (currentStep < 0) {
    currentStep = stepsPerWorkpieceRevolution + currentStep;
  } else if (currentStep > stepsPerWorkpieceRevolution) {
    currentStep = currentStep - stepsPerWorkpieceRevolution;
  }
  delay(300);
  moveStepperY(30, LOW);
}
```

Slika 39 Povijanje niti

5.4.1 *GLAVNA ZANKA*

Glavno zanko začnem s preverjanjem, če je še kakšna točka za prebrati iz datoteke. Če je ni, pomeni, da sem prišel do konca datoteke in je slika končana. V serijski monitor napišem, da je delo končano, nato pa ustvarim neskončno zanko, da se program ustavi in čaka. Če pa je še kakšna številka za prebrati, vrne funkcija *readNextPointFromFile() TRUE,* kar vodi program v glavni del. Najprej nastavim spremenljivko *findOffset* na 1, kar pomeni, da bo program začel slediti odstopanje pri obračanju. To ni dobro imeti zmeraj vklopljeno, saj pri nekaterih operacijah ni željeno merjenje odstopanja. Nato z uporabo funkcije *moveMotorToPoint()* premaknem WB na pozicijo naslednje prebrane točke. Nato dodam zamaknitev in ponovim celotni postopek.

Sproti tudi štejem, koliko žebljev sem že povezal in v serijski monitor izpišem, koliko umetnine je že narejene v procentih. Na koncu funkcije, torej ko je izdelek narejen, pa izpišem, koliko časa je potreboval za izdelavo.

```
while (digitalRead(pause_pin) != LOW) {
    if (readNextPointFromFile()) {
      findOffset = 1; // Premakni motor na ciljno točko
      moveMotorToPoint(atoi(buffer));
      delay(250);
      loopNail();       // Povij žebelj
      delay(250);
      findOffset = 0;

      currentLine++;
      Serial.print("Progress: ");
      float Progress = static_cast<float>(currentLine) * 100 / numOfLines;
      Serial.print(Progress);
      Serial.println("%");

    } else {
      Serial.println("Končano!");
      ...
      while (1) {
        // Ne naredi ničesar
      }
    }
```

Slika 40 Glavna zanka

V glavno zanko sem vključil tudi način za odpravljanje napak, v kateri se aktivira, ko je spremenljivka Debug = 1. Z njim sem lahko ročno premikal motor MX, kar mi je močno pomagalo pri nastavljanju referenčne točke poleg drugih reči. Lahko sem ga premikal za določeno število korakov in v željeno smer, z uporabo predznaka d, ali pa sem ga avtomatsko poslal v željeno točko, s predznakom p. Imam tudi na izbiro, da povijem žebelj, na katerem sem. Dodal sem tudi možnost nadaljevanja programa iz željenega mesta v datoteki točk, katero lahko uporabim s predznakom r. Če pa napišem »end« v serijski monitor, se program nadaljuje kot normalen, s tem da zopet preveri odstopanje s funkcijo *findZeroOffset()*.

6 PRAKTIČNI DEL

6.1 ZASNOVA

Način izdelave, da bodo posamezni kosi iz 5mm vezane plošče sem dobil deloma iz sestavljanja lego kock, kjer preprosto postaviš eno na drugo. Bilo mi je zelo težko začeti modelirati iz nič in ta način mi je zelo olajšal delo, kajti mi je podal vsaj eno mero z katero lahko začnem (debelina 5mm). Zelo je pomagalo tudi, da nisem rabil razmišljati v 3D izgledu celotnega izdelka, ampak sem se lahko osredotočil na preproste 2D sloje iz plošče, ki sem jih samo polagal drug na drugega.

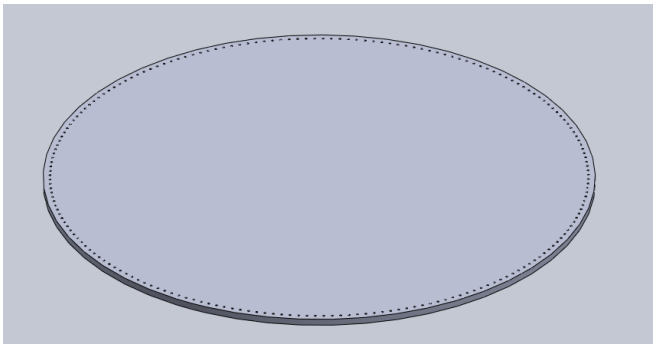
Prvo skico sem si naredil na kos papirja, kjer sem si približno določil, kako želim, da moj končni izdelek izgleda. Narisal sem jo v obliki razstavljene skice. (Priloga D)

6.2 MODELIRANJE KOSOV

Vse kose, torej tudi tiste, ki sem jih izrezal s pomočjo laserja, sem modeliral na programu Sloidworks 2023. Solidworks je programski paket za računalniško podprto konstruiranje in inženirske analize. Uporablja se za ustvarjanje 3D modelov, sestavov in tehnične dokumentacije. Najprej sem naredil kose iz vezane plošče, da sem dobil osnovno obliko, o tem, kako bo izgledala naprava. Po večih iteracijah, sem prišel na 8 različnih ključnih modelov. Poleg teh sem izdelal še nekaj testnih kosov.

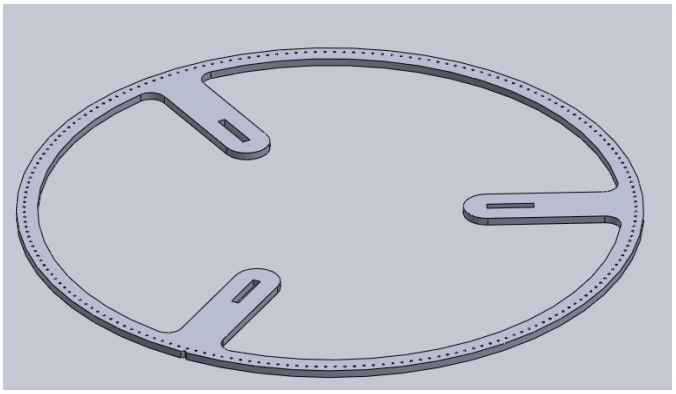
6.2.1 *LASERSKO IZREZANI*

1) Začel sem z najpomembnejšim kosem in to je WB. Moral je biti dovolj velik in imeti 200 lukenj za žeblje. Odločil sem se, da bo imel premer 40 cm in luknje 5 mm od zunanjega roba.



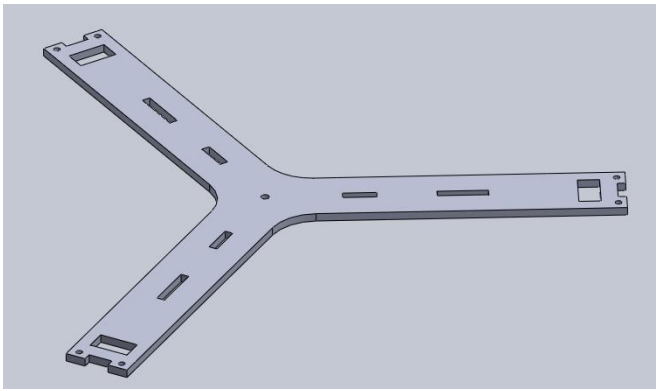
Slika 41 Model Workpiece

2) Druga stvar na listu je bil drugi del WB in to je spodnji kos, s katerim ga pritrdim na držalo. WB je sestavljen iz dveh delov, zato da imajo žeblji več materiala za podporo, in da lahko kos pritrdim s spodnjim delom in zgornji del nima lukenj. Mere so enake, le da ima ta na sredini odvzet nepotreben material in oznako, kje je žebelj 0



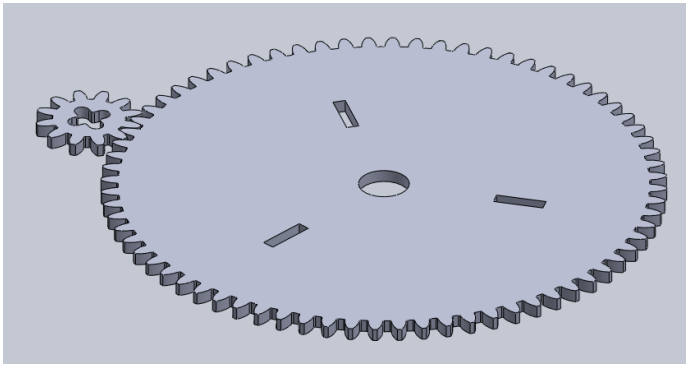
Slika 42 Model Workpiece_Bottom

3) Stvar, ki bo držala WB, sem poimenoval WorkpieceHolder. Ima tri noge in na koncu vsake pride 3D natisnjen kos z ležajem. Ima tudi luknje, v katere lahko vstavim povezovalni element, da zagotovim netrajno spajanje v primeru, da moram zamenjati kakšen kos.



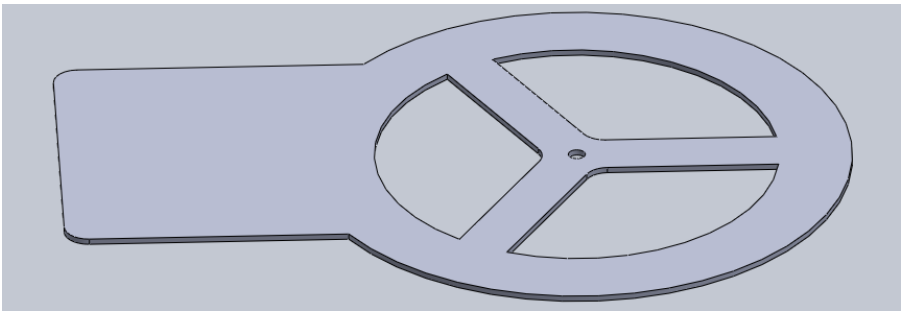
Slika 43 Model WorkpieceHolder

4 & 5) Vse to poganjam z dvema zobnikoma z razmerjem 72:12 (gnani: poganjalni) torej 6:1. Oba kosa sem prvotno nameraval narediti iz vezane plošče, a sem na koncu uporabil 3D natisnjen 12-zobni zobnik, ker se je bolje ujemal z 72-zobnim, katerega sem naredil iz 3 mm debele plastične plošče, ker je bila bolj ravna kot vezana plošča. Uporabljen modul na obeh je 3 mm, »pressure angle« 27. V sredini imata oba luknje, za vezanje z drugimi kosi.



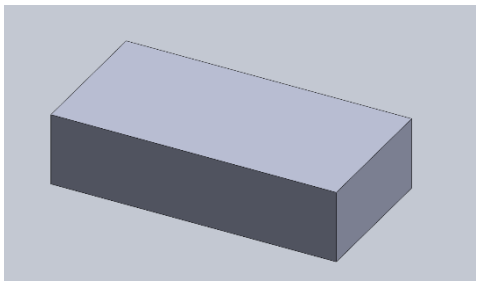
Slika 44 Model obeh zobnikov

6) Največji izmed kosov je tako imenovana Platforma. Je kos na katerem vsi ostali sedijo. Po njem se tudi gibljejo koleščka na WorkpieceHolderju. Ima veliko ploščad, na katero sem prostoročno izvrtal luknje, za pritrditev raznih drugih kosov. V končnem izdelku, sem tudi dodal nanj graviran napis.

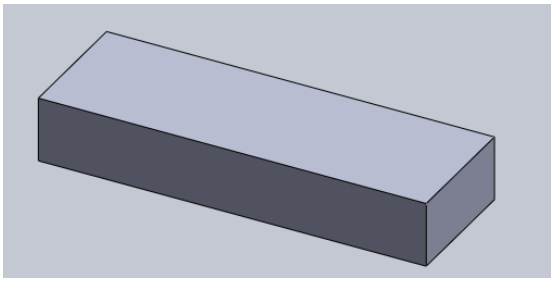


Slika 45 Model Platforme

7 & 8) Izdelal sem dva različna povezovalna elementa. Prvi povezuje držalo obdelovanca (Workpiece Holder) z 72-zobnim zobnikom, medtem ko drugi povezuje držalo obdelovanca s spodnjim delom delovnega stola (Workbench, WB). Ustvaril sem jih z namenom, da zmanjšam število trajnih povezav med elementi, kar omogoča lažje zamenjavo posameznih delov v primeru potrebe. Oba povezovalna elementa se pojavljata trikrat v izdelku.

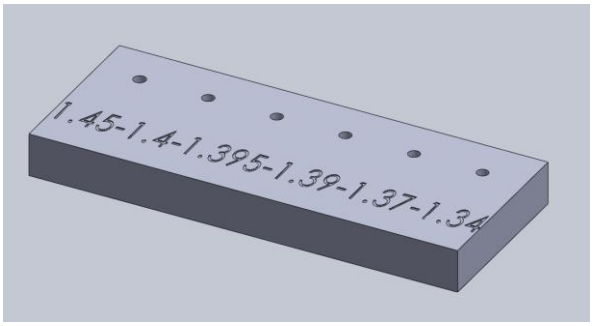


Slika 47 Model povezovalnega elementa 1



Slika 46 Model povezovalnega elementa 2

Ker nisem želel izrezati obeh kosov WB, samo da bi ugotovil, da so luknje premale ali prevelike, sem izdelal testni kos, na katerem imam luknje raznih dimenzij. Z njimi sem kasneje ugotovil, kako velika naj bo luknja za žeblje, da so dovolj tesno pripeti, da se ne usločijo ali padejo ven.

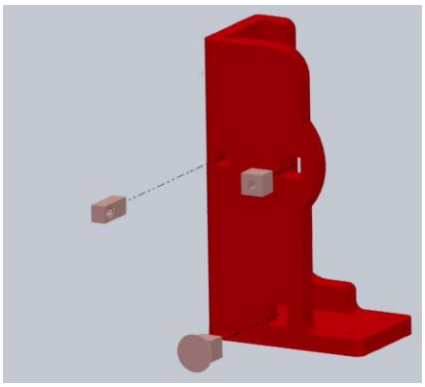


Slika 48 Model testnega kosa

6.2.2 *3D NATISNJENI*

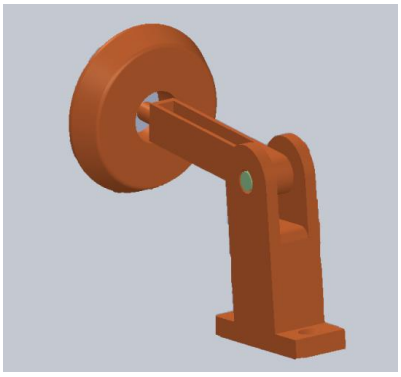
Ko sem imel narejene vse lasersko izrezane kose, sem se lahko lotil izdelovanja 3D natiskanih kosov. Za lažjo izdelavo, sem sproti vstavljal vse modele v assembly datoteko, ki mi omogoča imeti vse kose na enem mestu in jih zlagati, da lahko dobim realno predstavo (Priloga E). Modelov namenjenih za tiskanje je precej več in sicer okoli 25. V večini je več kosov sestavilo eno komponento.

1) Najpomembnejša komponenta je tako imenovan Arm, ki drži koračni motor in usmerja nit v Hand. Je največji od vseh kosov, zato ima 3 luknje, s katerimi ga lahko pritrdim na ploščad. Ob strani ima luknje, v katere lahko dodam tako imenovane dodatke, ki usmerjajo nit. Nisem jih hotel dodati k sami roki, zaradi lažjega tiskanja in izmenjave samega dela, če ta ne bo utrezen.



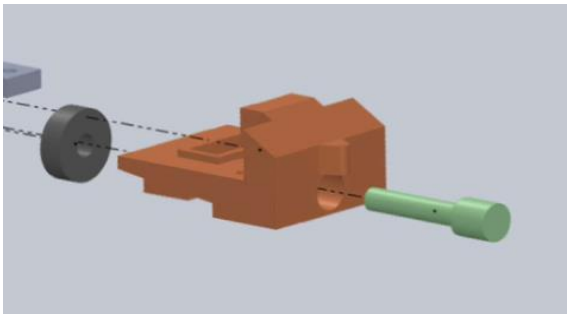
Slika 49 Model Arm ter dodatki

2) Del, ki se vrti po vrhu nitke in jo potiska dol, da se ne zabije v konico roke, sem izdelal iz 4 kosov + ležaj. Na dnu sem dodal luknje za pritrditev na ploščad. V samem koleščku sem pa namestil prej omenjeni ležaj.



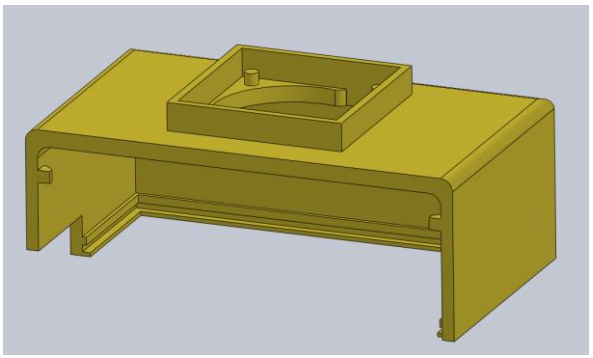
Slika 50 Model Rollerja

3) Na vse konice WorkplaceHolderja sem naredil mala držala za 19 mm (*zunanji premer*) ležaj in dve luknji, kjer ga lahko pritrdim. En kos ima izboklino, ki mi omogoča da najdem referenčne točke in kalibracijo. Ležaj pritrdim z natisnjenim zatičem.



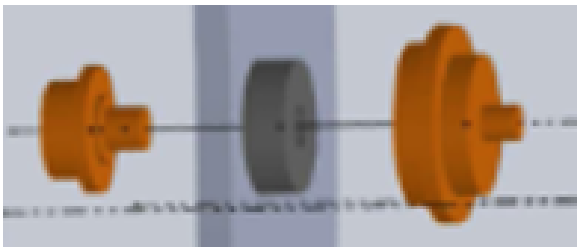
Slika 51 Model HolderHand ter zatič

4) Arduino shield ramps 1.4 pokrijem s kosom, ki drži 12V ventilatorček, omenjen med uporabljenimi materiali. Ta model sem razdelil v 2 dela, za lažje tiskanje. Naredil sem tudi vrata za ta kos, a jih kasneje nisem uporabil.



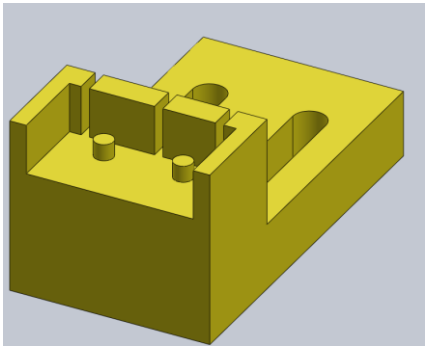
Slika 52 Model ramps 1.4 cover

5) V središče vrtenja sem vstavil enak ležaj, kot tisti, ki sem ga uporabil za kolesa. Zanj sem moral natisniti spodnje in zgornje držalo. *Spodnja slika je obrnjena za 90˚ za prihranitev prostora.*



Slika 53 Model centralnih držal

6) Končno stikalo držim s posebno modeliranim kosom, ki ga postavi na točno višino, da lahko najde pozicijo WB. Pritrdim ga z dvema vijakoma, saj je zelo ključno, da se ne premika.

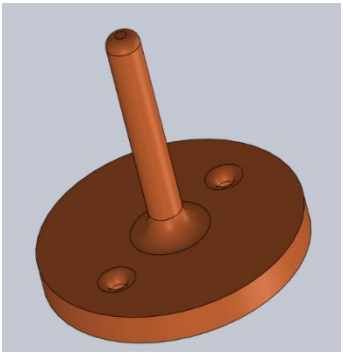


Slika 54 Model držala končnega stikala

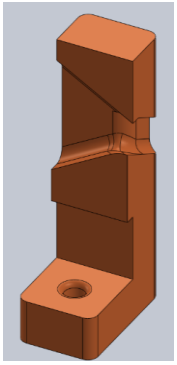
7) Celoten izdelek leži na 5 nogah, ki so ravno dovolj visoke, da pod platformo pride še koračni motor.

Slika 55 Model nog

8) Za upravljanje niti imam še dva kosa. Eden, na katerem se nit vrti in drugi, ki preusmeri nit v željeno smer.

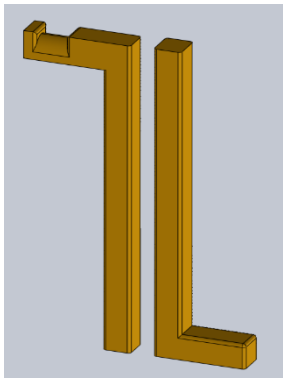


Slika 57 Model držala niti



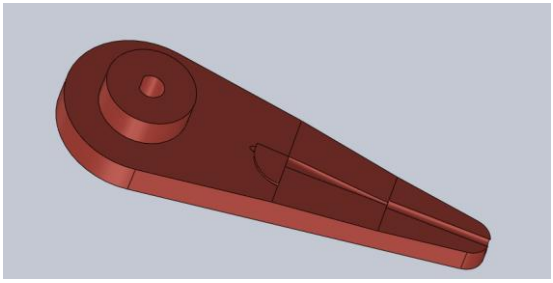
Slika 56 Model usmerjala niti

9) Da sem lahko uporabljal tudi večje sukance niti, sem modeliral kos, ki bo to nit vlekel iz vrha sukanca, ker bi potreboval preveč sile, da ga obrnem. Kos sem razdelil na dva dela, da je lahko natisnjen na tiskalniku.



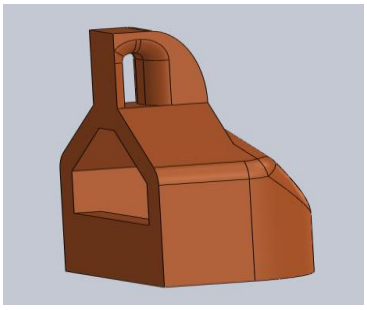
Slika 58 Model usmerjalnika niti za večje sukance

10) Stvar, ki ovije nit okoli žeblja, se pritrdi na MY. Ima majhno izboklino, ki vodi nit proti luknji na vrhu, v katero sem vstavil upognjeno šivilsko iglo, ki vodi nit med žeblji.



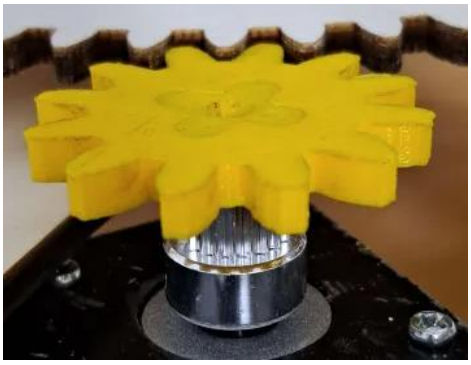
Slika 59 Model roke za povijanje žebljev

11) Da preprečim popuščanje napetosti niti, ko se roka vrača, uporabim utež, ki jo drži napeto ob povratnem gibu roke. Ima votlo notranjost, v katero lahko dam kakšen material, ki bo celoten kos utežil.



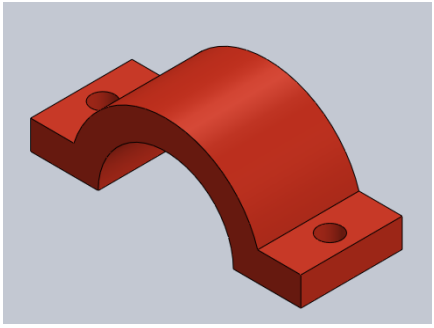
Slika 60 Model StringWeight

12) Manjši od dveh zobnikov pritrdim na motor z uporabo 3D natisnjenega kosa. Prvotno naj bi se ta pritrdil na izrezan manjši zobnik, ampak sem se nato odločil da ga natisnem, zaradi boljšega ujemanja zob. Kasneje sem ga trajno pritrdil na kovinsko glavo koračnega motorja, da preprečim deformacije plastike ob daljši uporabi.



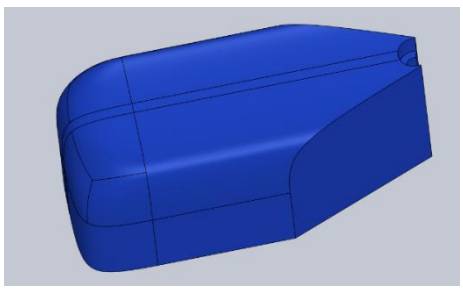
Slika 61 Manjši zobnik (foto: T.Mustač)

13) Naredil sem tudi kos, s katerim lahko pritrdim vse kable ob ploščad oz. kabelsko sponko.

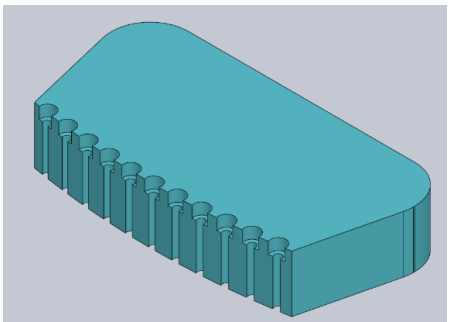


Slika 62 Model kabelske sponke

14) Za zagotovitev pravokotne pritrditve žebljev sem izdelal dva specializirana pripomočka, ki bistveno olajšata izvedbo postopka izdelave.



Slika 64 Specializirano orodje 1



Slika 63 Specializirano orodje 2

6.3 IZDELAVA KOSOV

6.3.1 *3D TISKANJE*

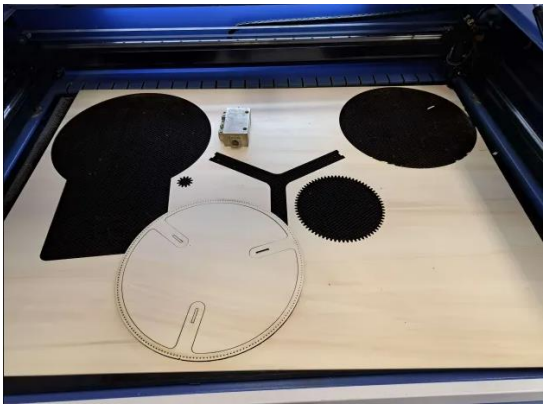
Tiske sem pridobil na treh različnih 3D tiskalnikih. Eden je bil od mojega dedija, druga dva pa iz šole in od mentorja. Pri printanju sem uporabil razne barve in metode segmentiranja (angl. slicing) modelov. Kjer je bilo potrebno, sem dodal podporo. Več tiskov sem zavrgel, ker niso ustrezali pričakovanim standardom kakovosti, so imeli težave z adhezijo, ali pa so se pojavile napake med postopkom tiskanja. Tako sem izdelal okoli 50-60 kosov, mnogo od njih ni prišlo do končnega izdelka zaradi raznih sprememb in napak.

6.3.2 *LASERSKO REZANJE*

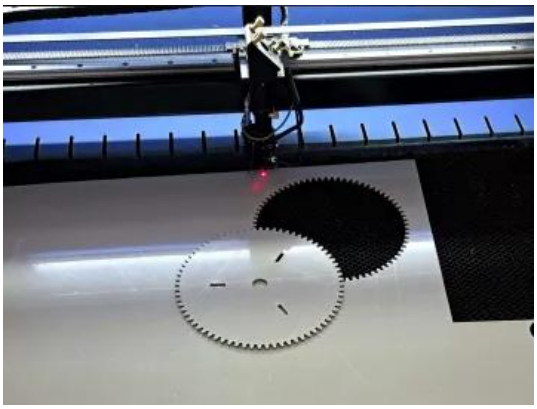
Za laserski izrez sem uporabil stroj FL-1390. Laser FL-1390 ponuja vsestransko delovno površino 1300 × 900 × 280 mm z zaprtimi vrati in 1300 × ∞ × 30 mm z odprtimi vrati. Opremljen je s CO2 stekleno cevjo moči 100W/130W, ki ima življenjsko dobo do 10.000 ur in se hladi s pomočjo industrijskega hladilnika CW 5000/5200. Stroj ima samodejno nastavljivo mizo, lečo za fokus z izbirnimi velikostmi ter podpira različne grafične formate. S sposobnostjo rezanja debeline do 25 mm in združljivo programsko opremo, kot je Ruida RD Works, zagotavlja natančnost in učinkovitost v kompaktnem dizajnu dimenzij 1830 × 1320 × 1120 mm ter teže 380 kg.

Vse kose sem izrezal enkrat, razen tiste, ki sestavljajo WB, katere moram za vsako novo sliko ponovno izrezati.

Ponovno sem izrezal velik zobnik iz 3-milimetrske plastične plošče, saj tisti iz vezane plošče ni bil dovolj ploščat in je posledično imel neenakomerno oblikovana zobe.



Slika 66 Rezanje kosov na laserskem stroju (foto: T.Mustač)

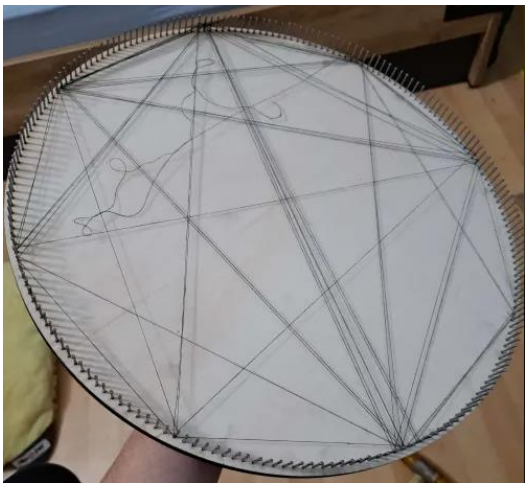


Slika 65 Rezanje zobnika na laserskem stroju (foto: T.Mustač)

6.3.3 *IZDELAVA WB*

Izdelava delovne površine za delo z nitno umetnostjo, imenovane tudi "WB" ali platno, je zahtevala največ časa v celotnem procesu. Uporabil sem kladivo in specializiran pripomoček za vstavljanje žebljev, da sem pritrdil žeblje v podlago. Poleg tega sem izdelal podlogo za zabijanje žebljev iz prazne pločevinke, da bi preprečil, da bi žeblji prodrli preveč globoko in poškodovali tla sobe. Celoten postopek izdelave WB mi je vzel približno 50 minut.

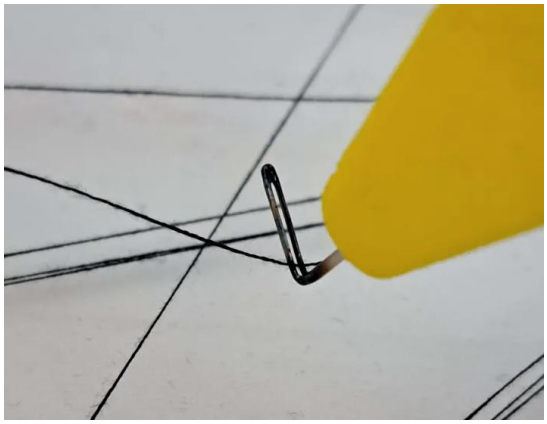
Slika 68 Sestavljanje WB (foto: T.Mustač)



Slika 67 WB z vsemi žeblji (foto: T.Mustač)

6.3.4 *MANJŠI KOSI*

Iglo sem upognil s pomočjo vžigalnika in klešč. Po večih neuspelih poskusih sem prišel do ugotovitve, da mora biti luknja, skozi katero poteka nit, čim večja, hkrati pa mora biti ukrivljena pod določenim kotom, saj bi se drugače povila okoli igle in posledično onemogočila nadaljne delo stroja. Kot, pod katerim sem ugotovil, da nit teče najbolje, je okoli 115˚. Iglo sem nato pritrdil z vročim lepilom.



Slika 69 Upognjena igla (foto: T.Mustač)

Eno od držal koračnega motorja sem naredil s pomočjo pnevmatičnega stiskala, ker nisem mogel z običajnimi metodami. Najprej sem s kladivom zbil nosilce, ki so ga držali pod 90˚ kotom, nato sem s pomočjo profesorja Seitla uporabil stiskalnico narejeno na Strojni šoli, da sem z lahkoto sploščil ta kos. Na koncu, sem ga moral še malo s kladivom izravnati.

Da sem lahko povezal zunanje komponente z Arduino mikrokrmilnikom, sem prerezal dupont žičke in jih spajkal z daljšo enojno jedrno žico, s katero sem kasneje povezal končno stikalo in stikalo za ustavitev naprave. Za dodatno izolacijo, sem jo ovil z istobarvnim duct tape lepilnim trakom. Bolje bi jo bilo poviti s termoskrčljivo folijo, a je nisem imel na voljo tisti čas.

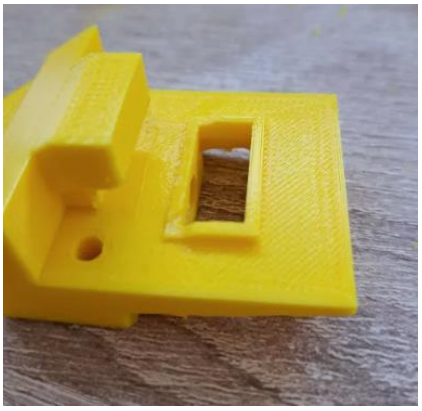
Slika 70 Izdelava povezovalnih žičk (foto: T.Mustač)

Vse kose, ki so bili premajhni ali preveliki, sem popravil s pomočjo rezalnega noža z zamenljivimi rezili, brusnega papirja in maskirnega traku. Vse kar je bilo preveliko, sem odrezal ali pobrusil. Vse kar je pa bilo premalo, sem pa povil oz. dodal sloje maskirnega lepilnega traku, da sem dosegel željene dimenzije.

6.4 SESTAVA NAPRAVE

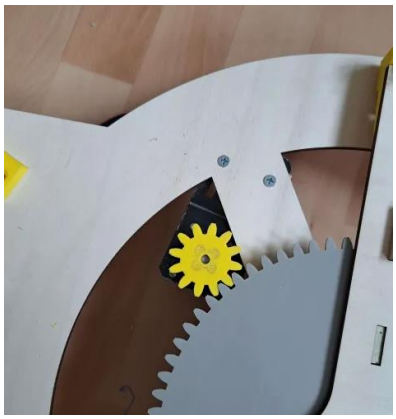
Napravo sem sestavil po delih, ki sem jih nato dal skupaj. Najprej sem sestavil držalo WB, na katerega sem pritrdil 3D-natisnjene dele in štiri ležaje. Tri ležaje sem namestil ob obodu kot kolesa, medtem ko sem četrti ležaj namestil v os držala. Nato sem to celoto pritrdil na spodnjo ploščad. Pri tem sem nekatere 3D-natisnjene dele moral dodatno obdelati z olfa nožem, da sem zagotovil, da bodo ustrezali nameščeni v svojo pozicijo.

Slika 72 Držalo WB na platformi (foto: T.Mustač)

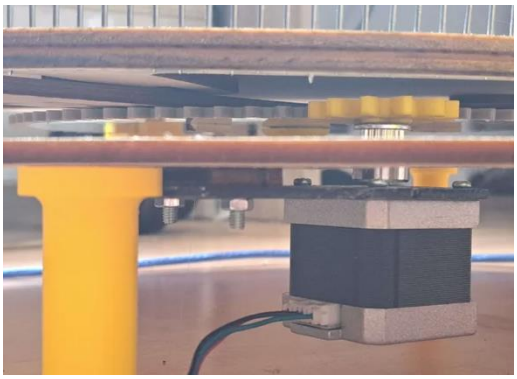


Slika 71 Dodatna obdelava z olfa nožem (foto: T.Mustač)

Za tem sem ploščad opremil s petimi nogami, katere sem pritrdil z vročim lepilom. Noge sem postavil na mesta, na katerih se mi je zdelo, da bo največ koristi. Za tem sem koračni motor pritrdil na ploščad. Pritrdil sem ga pod kotom, da ga lahko preprosto nastavljam in prilagodim za popolno ujemanje zobnikov. Prav tako sem moral med držalom motorja in ploščadjo pritrditi nekaj materiala, ker je gred koračnega motorja stala previsoko (5 mm preostali kos vezane plošče).



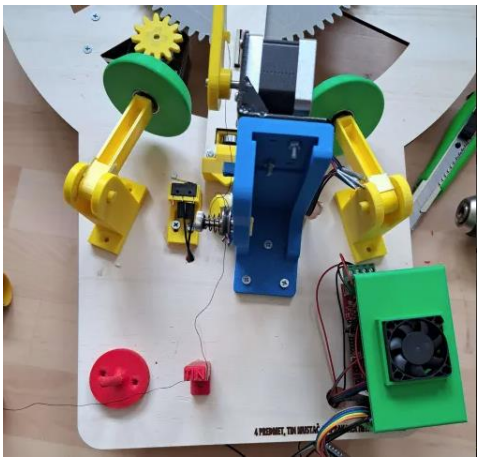
Slika 73 Pritrditev motorja na ploščad (foto: T.Mustač)



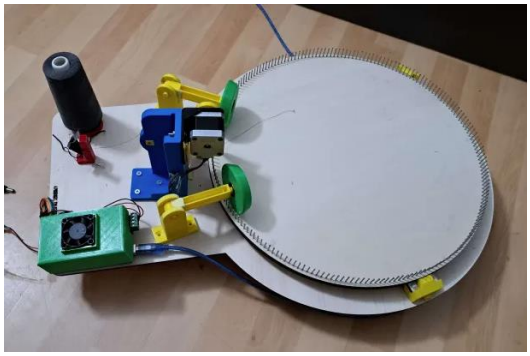
Slika 74 Razmik motorja od ploščadi (foto: T.Mustač)

Sprva sem načrtoval pritrditev vseh preostalih 3D-natisnjenih kosov z vijaki, vendar sem zaradi pomanjkanja le-teh večino kosov pritrdil na ploščad z vročim lepilom. MY sem pritrdil na držalo, nato pa to držalo na roko, uporabljajoč tri vijake. Postavil sem jo pod takšnim kotom, da je os gredi MY vzporedna z navidezno tangentno črto WB. Ob vsaki strani roke sem namestil dva rolerja, ki tiščita nit stran od konice roke, da preprečujeta trčenje. Koleščka sem postavil čim bliže žebljem. Direktno poleg roke sem postavil držalo končnega stikala, katerega sem najprej pritrdil z dvema vijakoma, ko sem pa določil točno željeno pozicijo, sem ga trajno pritrdil še z vročim lepilom.

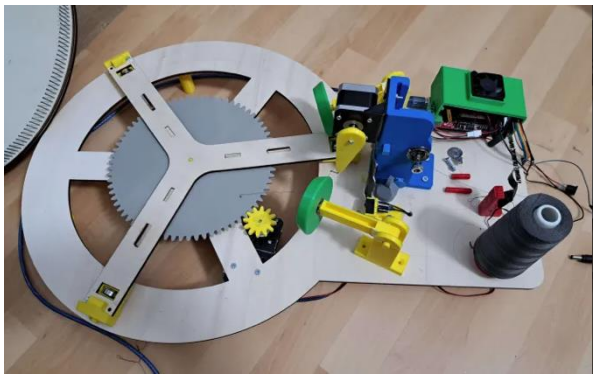
Na levi strani sem postavil držalo za nit in nekoliko desno od tega usmerjevalnik niti. Na drugi strani sem namestil Arduino mikrokrmilnik. Čim več žic sem skušal skriti pod platformo, da ne bi bile vidne. Te sem pritrdil s 3D-natisnjenimi kosi. Arduino sem pritrdil z žeblji in vročim lepilom na njih, da ne padejo ven.



Slika 75 Postavitev kosov (foto: T.Mustač)



Slika 77 Naprava iz desne strani (foto: T.Mustač)



Slika 76 Naprava iz leve strani, brez WB (foto: T.Mustač)

6.5 UPORABA

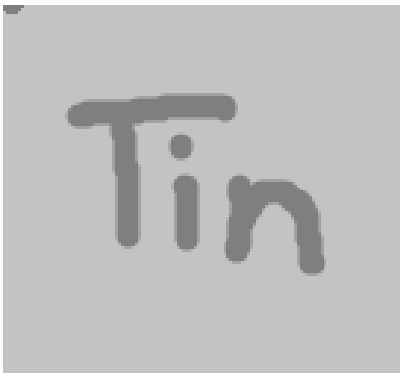
Kljub temu da je cilj te naloge čim bolj avtomatizirati postopek ustvarjanja tovrstne umetnine, je še vedno potrebno kar nekaj človeške intervencije in pomoči pri izdelavi. Prva stvar, ki jo moraš kot uporabnik narediti, je najditi ustrezno sliko oz. če ta ni takšna, jo obdelati. Sledi pretvorba te slike v točke s pomočjo prej omenjenega spletnega programa. Nato je treba ročno vstaviti vseh 200 žebljev v izrezane kose, katere moraš tudi za vsako sliko ponovno izdelati na laserskem stroju. Ko imaš to opravljeno, moraš v kodi spremeniti, katero datoteko naj program bere. Šele za tem lahko vstaviš nit v stroj in jo pritrdiš na točko 0. Pred vsako izvedbo programa je dobro pregledati, če se končno stikalo aktivira v predvideni poziciji, torej če je referenčna točka še vedno na isti poziciji. Ko je izdelek končan, je potrebno nit pritrditi na zadnji žebelj.

Za optimalno delovanje naprave je najbolje uporabljati računalnik. Priporočljivo je uporabiti prenosni računalnik zaradi njegove prenosljivosti, kar omogoča enostavno premikanje naprave na želeno lokacijo. Računalnik je potreben tudi za uporabo naprave, saj je za vsako novo sliko potrebno prilagoditi specifične parametre v kodi ter na SD kartici.

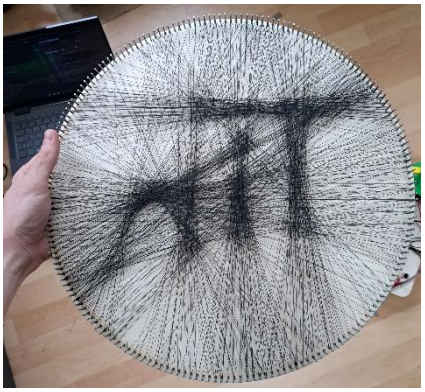
Omembe vredno je tudi, da za uporabo naprave ni nujno potrebna slika, ki bi jo stroj poskušal kopirati. Namesto tega je mogoče ustvariti tudi geometrijske vzorce. Vse, kar mora uporabnik narediti, je poiskati zaporedje številk na spletu ali napisati funkcijo, ki generira želen vzorec. Nato je uporaba enaka, kot če bi želel narediti fotorealistično sliko.

6.6 KONČNI IZDELKI

Pri prvem celotnem preizkusu izdelka sem uporabil sliko podpisa mojega imena. Načrtovanih je bilo 1300 črt, vendar mi je niti zmanjkalo po malo več kot 400 črtah. Slika je bila zrcaljena zaradi napake v moji kodi. Največja težava med prvim daljšim preizkusom, kot je bil ta, je bilo število trkov roke z žebljem. Če ne bi bil nenehno prisoten in hitro popravil napake, bi se stroj popolnoma zataknil in bi lahko prišlo do uničenja.



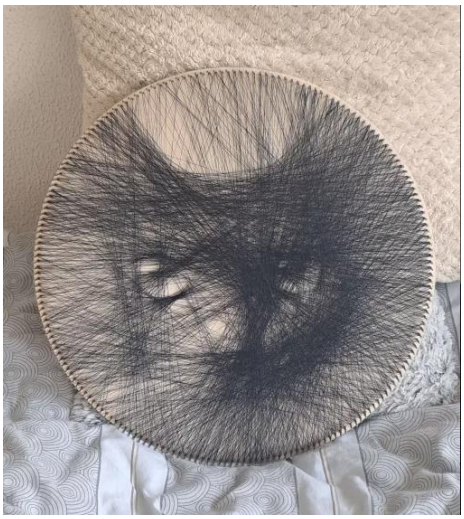
Slika 79 Prvi test - vhodna slika



Slika 78 Prvi test - izhodna slika (foto: T.Mustač)

V drugem preizkusu sem opazil izboljšanje rezultatov, vendar še vedno niso bili popolni, kar je posledica moje napake. Kljub temu je naprava delovala brezhibno in se ni nikoli zaletela v žebelj. Ponovno sem poskusil z vhodno sliko, ki je bila načrtovana za 1300 točk, vendar sem nenamerno ustavil program, ko je bil izdelek približno 80 % dokončan. Kljub tej napaki je končni izdelek podoben želeni sliki. Druga napaka, ki sem jo naredil, je bila, da nisem popravil problema zrcaljenja, zaradi česar je izhodna slika še vedno zrcaljena. Kljub temu je bilo opaziti izboljšanje in napredek v primerjavi s prejšnjim preizkusom.

Slika 81 Drugi test - vhodna slika



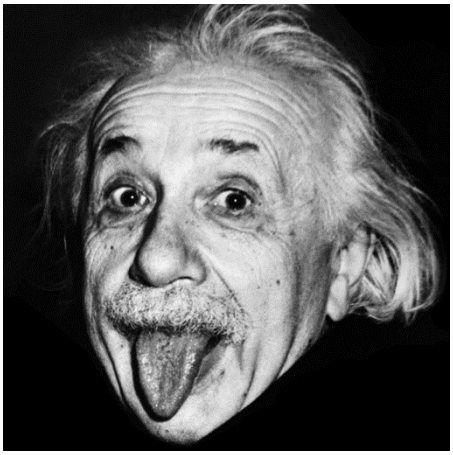
Slika 80 Drugi test - izhodna slika (foto: T.Mustač)

V tretjem testu sem dosegel najslabše rezultate, kar pripisujem številnim napakam, do katerih je prišlo med postopkom ustvarjanja. Uporabljal sem drugo nit, ki se je med ustvarjanjem, zaradi manjših zataknitev dvakrat pretrgala. Večino krivde za to pripisujem slabi pritrditvi igle, ki se je večkrat obrnila in posledično trčila v kakšen žebljiček, kar je povzročilo dodatne težave. Zaradi tega sem moral večkrat improvizirati in poskusiti ponovno začeti program. Končni izdelek na koncu ni bil skoraj nič podoben željeni sliki. Možen razlog za to je tudi prekomerna kompleksnost vhodne slike. Sliko, ki naj bi imela 3000 črt, sem zato nekje okoli 1300 črt prenehal izdelovati.

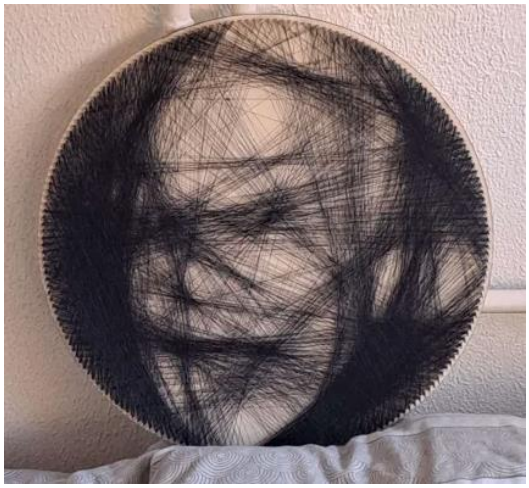
Slika 83 Tretji test - vhodna slika

Slika 82 Tretji test - izhodna slika (foto: T.Mustač)

Četrti test je pokazal zelo dobre rezultate. Končna nitna slika je vsaj zame zelo podobna vhodni sliki. Naprava je delovala brez napak, le občasno sem moral kaj premakniti, vendar bi verjetno tudi brez tega vse potekalo gladko. Slika ima 1900 povezanih črt od načrtovanih 3000, vendar že zdaj izgleda odlično. To mi pove, da za dosego željene kakovosti nitne slike ne potrebujem toliko črt. Seveda manjše število črt pomeni nekoliko manj podrobnosti, vendar pa tudi krajši čas izdelave, manj možnosti za napake in tako naprej.

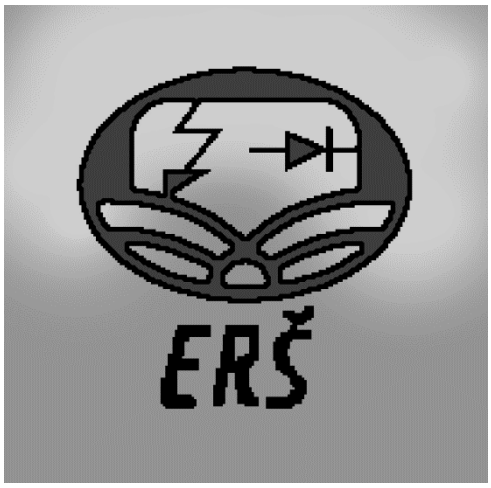


Slika 84 Četrti test - vhodna slika



Slika 85 Četrti test - izhodna slika (foto: T.Mustač)

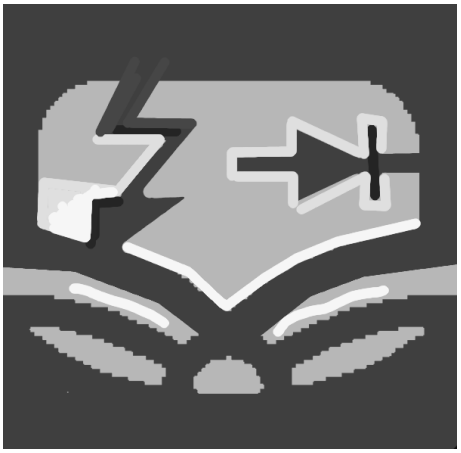
Peti test je prav tako pokazal zelo dobro delovanje naprave, saj je bil prvi test, ki sem ga v celoti opravil. To pomeni, da sem sliko dopolnil v celoti, do 100 %. Žal slika ni dobra, kar je verjetno posledica omejitev nitne umetnine in morda tudi slabe vhodne slike. Nitna slika je bila sestavljena iz 2000 črt in je potrebovala 160 minut za izdelavo oz. 2 uri in 40 minut.



Slika 87 Peti test - vhodna slika

Slika 86 Peti test - izhodna slika (foto: T.Mustač)

V šestem testu sem ponovno poskusil izdelati logo moje šole (ERŠ). Rezultati so bili precej bolj uspešni kot v prejšnjih poskusih. Slika vsebuje 1900 povezanih niti. Med izdelavo mi je enkrat skoraj zmanjkalo niti, a sem jo preventivno zamenjal tako, da sem program zaustavil in konec in začetek zavezal za isti žeblj. Po menjavi niti sem imel nekaj težav, saj se je robot nenadoma začel zabijati v žeblje, čeprav pred tem ni bilo nobenih težav pri povezovanju približno 1200 črt. Ampak po tem je naprava zopet delovala kot normalno. Ker sem med delovanjem večkrat ustavil program, se čas ni štel od celotnega začetka, zato nimam časa izdelave kosa.



Slika 89 Šesti test - vhodna slika

Slika 88 Šesti test - izhodna slika (foto: T.Mustač)

6.7 TEHNIČNE ZNAČILNOSTI

Dolžina	~ 66cm
Širina	~ 46cm
Višina	~ 30cm
Teža	~ 2.5kg
Volumen	~ 3000cm³
Čas izdelave	~ 4.8145s / 1 črto

Naprava ima dolžino 66 centimetrov, širino 46 centimetrov in višino približno 30 centimetrov. Njena teža je ocenjena na 2.5 kilograma, s prostornino približno 3000 kubičnih centimetrov. Za izdelavo ene črte potrebuje okoli 4.8145 sekunde. Tako da izdelava slike s 3000 črtami traja približno 240 min.

Sestavljena je iz vezanega lesa breze in 3D-natisnjenih komponent. Poleg tega vsebuje tudi različne kovinske elemente (ležaji, vijaki, žeblji), vključno z električnimi (Arduino, koračni motorji, Ramps 1.4, žičke, 12V ventilator ...).

Povezave in pritrditve so izvedene z različnimi sredstvi, med katerimi so uporabljeni čepi, vijaki, matice in vroče lepilo.

7 ZAKLJUČEK

Z rezultatom izdelka sem zelo zadovoljen, kajti dosegel sem večino ciljev, ki sem si jih zadal. Rezultati začetnih hipotez so sledeči:

- *Izdelal bom delujočo napravo*

  Ta hipoteza nedvomno velja za potrjeno, saj naprava deluje uspešno. Opravil sem več uspešnih testov in ustvaril končne izdelke. Z napravo lahko ustvarim katerikoli nitni vzorec, bodisi geometrijski ali fotorealističen, ki si ga zaželim. Vse to pod pogojem, da imam na voljo točke za njegovo ustvarjanje.

- *Končni izdelek bo podoben vhodni sliki*

  Končni izdelek je brez dvoma podoben vhodni sliki, če ju primerjamo eno ob drugi. Kljub temu da se na nitnemu izdelku ne vidi vsake podrobnosti, je iz večine primerov dovolj razvidno, da jo lahko povežeš z ustrezno dano vhodno sliko. Zato lahko to hipotezo potrdim.

- *Končni izdelek bo razviden tudi brez ogleda začetne slike*

  Čeprav je končni izdelek nedvomno prepoznaven tistim, ki so videli vhodno sliko, lahko drugače rečemo za tiste, ki nimajo predhodnega znanja o tem, kako naj bi izdelek izgledal. Vendar se ta situacija redko pojavi, saj tisti, ki opazujejo izdelek, običajno lahko prepoznajo njegovo podobnost s prvotno sliko, če dobro opazujejo. Poleg tega lahko ljudem, ki nimajo konteksta, enostavno povemo, kaj naj bi predstavljala nitna umetnina, kar jim pomaga razumeti končni izdelek. Ampak, ker se še vedno lahko zgodi, da nekdo ne vidi te similarnosti z vhodno sliko, bom žal moral zavrniti to hipotezo.

- *Stroj bom naredil iz preprosto dostopnih materialov*

  Stroj je bil narejen popolnoma iz preprosto dostopnih materialov. 3D tiskalniki so zelo pogosti v današnjih časih, torej tudi če ga nimaš je zagotovo nekdo ali nekakšna ustanova v tvoji bližini, ki bi ti jih dovolila uporabljati. Vezana plošča je tudi zelo dostopna, kajti kupiš jo lahko povsod po svetu. Laserski rezalnik, bi mogoče predstavljal problem, ampak tudi obstajajo podjetja, ki lasersko izrezujejo po naročilu.

Vse ostale kose sem kupil v lokalnih trgovinah ali pa jih naredil sam iz stvari, ki so ležale doma. Zaradi tega bom to hipotezo tudi potrdil.

- *Naprava bo preprosta za uporabo*

  Za upravljanje naprave v njenem trenutnem stanju je potrebna uporaba računalnika. Seveda je možno predhodno naložiti programsko opremo ter točke na mikrokrmilnik in SD kartico, vendar v tem primeru uporabnik ne bo imel vpogleda v trenutne operacije naprave in ne bo vedel, kaj naprava trenutno dela in koliko časa še potrebuje za dokončanje izdelka. Poleg tega je potrebno najti ustrezno sliko, jo ustrezno obdelati ter pretvoriti v točke, ki jih je nato potrebno naložiti v spominsko kartico. Zaradi kompleksnosti in potrebnih postopkov bom to hipotezo zavrnil.

7.1 MOŽNE IZBOLJŠAVE

Kot končni izdelek bi tudi želel, da je celotna naprava bolj vzdržljiva. Zato bi moral za osnovo uporabiti material, ki je bolj trden in odporen na obremenitve. Prav tako bi bilo lepo, če pride naprava v enem tako rečenem »paketu«, kjer izgleda ko ena sama stvar, da ima ohišje in se posamezne komponente ne vidijo.

Nekaj, kar je tudi precej pomanjkljivo, je enostavnost uporabe in uporabniška izkušnja. Če bi dodal notranji računalnik, v katerega bi uporabnik samo vstavil sliko in bi ta vse ostalo naredil sam, bi izjemno olajšalo in poenostavilo uporabo. Seveda bi poleg tega potreboval tudi nekakšne vmesnike med programi ter zaslon, ki bi uporabniku prikazal vse informacije. Sem bi tudi spadala dodatna izboljšava, da ima celotna naprava samo en kabel, za povezavo z elektriko.

Potencialna optimizacija bi izhajala iz uporabe dvojnih vijačnih zobniških koles. Ta pristop bi znatno povečal natančnost celotne naprave, saj bi se zobnika med seboj bolj tesno stikala. Vendar pa bi izdelava takšnih zobnikov na laserskem rezalniku bila neizvedljiva, zato bi bila potrebna alternativna proizvodna metoda. Lahko bi tudi uporabil kakšen zobati jermen, ampak bi moral narediti več prostora med motorjem in središčem WB.

Največja izboljšava po mojem mnenju bi bila funkcija avtomatskega vstavljanja žebljev, kajti to predstavlja večino fizičnega dela, potrebnega za vsak izdelek. Lahko bi tudi naredil drugačno različico WB, ki ne potrebuje žebljev, ampak se nit povije oz zatakne okoli samega kanvasa. Za to bi tudi moral spremeniti celotno napravo.

K povečanju hitrosti izdelave bi bistveno prispevala uporaba močnejšega MX motorja, ki je odgovoren za rotacijo WB. Z uporabo zmogljivejšega motorja bi bilo mogoče doseči hitrejše premikanje med točkami, kar bi znatno skrajšalo čas izdelave.

Glasnost naprave bi tudi lahko zelo zmanjšal in sicer z uporabo tišjih motorjev in tišjega ventilatorja. Prav tako bi lahko podložil pot, po kateri se ležaji premikajo z nekakšno gumo, da zmanjšam vibracije in glasnost.

Za doseganje večje natančnosti in preprostosti kode bi bilo mogoče uporabiti drugačen senzor za zaznavanje referenčne točke. Optični ali laserski senzor bi bil najprimernejši, pri čemer bi lahko enostavno integrirali luknjo v zeroHand (roka na držalu), ki bi omogočila zaznavanje referenčne točke.

Z uporabo končnega stikala bi bilo mogoče zaznati ali je MY uspešno povil nit okoli žeblja ali se je zataknil. Tako bi se stroj samodejno ustavil, če pride do zatika, s čimer bi se izognili morebitnim večjim problemom in poškodbam.

K izboljšanju razvidnosti slike bi največ pripomogel večji WB in več točk oz. žebljev za pritrditev niti. Vendar bi večja velikost WB in več točk povzročila večjo težo, kar bi zahtevalo tudi večji motor za obračanje. Poleg tega bi takšna nitna slika potrebovala več črt, torej več niti za izvedbo, kar bi dodatno povečalo težo izdelka. Skupni izdelek bi postal bistveno večji in težji za prenašanje in shranjevanje.

Ena možnost izboljšave bi bila zamenjava žebljev, ki jih trenutno uporabljam. Trenutni žeblji imajo drastično različne glave, kar povzroča neenakomerno razporeditev med njimi, ko so v WB. Druga možnost bi bila, da vsakemu žeblju odrežem ali odščipnem glavo, kar bi omogočilo bolj enakomerno razporeditev med njimi. To bi izboljšalo reliabilnost naprave in zmanjšalo število zataknitev.

8 VIRI IN LITERATURA

(1) Slika preproste nitne umetnine. Ubuy, Baosity. https://www.ubuy.com.om/en/product/2Y115VAO-baosity-vintage-string-craft-geometric-d-figure-string-art-kits-diy-home-decor-for-adults-40x40cm (2.2.2024)

(2) Primer komplicirane nitne slike. Blue Thumb. https://bluethumb.com.au/maha-parastar/Artwork/string-art-portrait (4.2.2024)

(3) Dokumentacija nitne umetnine, ter njene zgodovine. HISOUR. https://www.hisour.com/string-art-17973/ (2.2.2024)

(4) Slika nitne umetnine jelena. The frosted flamingo. https://frostedflamingo.com/product/string-art/ (2.2.2024)

(5) Slika nitne umetnine drevesa. String of the Art. https://www.stringoftheart.com/products/oak-tree-string-art-kit (2.2.2024)

(6) Coward, C. Barton Dring Built a Spinning String Art Machine with an ESP32. https://www.hackster.io/news/barton-dring-built-a-spinning-string-art-machine-with-an-esp32-9b52ff70f32 (31.1.2024)

(7) Primer naprave za nitno umetnino 2. Alinedeco. https://www.alinedeco.com/products/string-art-machine (31.1.2024)

(8) Wonderina. Primer naprave za nitno umetnino 3. https://www.youtube.com/watch?v=IC3j9N8U0nA&ab_channel=Wonderina (31.1.2024)

(9) Lu, E. Primer naprave za nitno umetnino 4. https://www.youtube.com/watch?v=QCBl4WKxBGw&ab_channel=ErikaLu (31.1.2024)

(10) yoavkleiner. Primer naprave za nitno umetnino 5. https://www.youtube.com/watch?v=CaVLRoDt7dA&t=11s&ab_channel=yoavkleiner (31.1.2024)

(11) Dluzen, k. Primer naprave za nitno umetnino 5. https://kevindluzen.com/portfolio/string-art-machine/ (3.2.2024)

(12) Primer naprave za nitno umetnino 6. TU Wien. https://techxplore.com/news/2018-09-art-robot.html (3.2.2024)

(13) Sallent, M. Primer naprave za nitno umetnino 7. https://www.youtube.com/watch?app=desktop&v=NFEXO5FEKDs&ab_channel=MarcSallent (3.2.2024)

(14) Arduino team. This machine automatically threads beautiful string art. Arduino. https://blog.arduino.cc/2023/09/27/this-machine-automatically-threads-beautiful-string-art/ (31.1.2024)

(15) Paul MH. Building a string art machine. https://medium.com/@paulmorrishill/building-a-string-art-machine-eeee386a38db (31.1.2024)

(16) Walter, M. AUTOGRAPH: A STRING ART PRINTER. https://hackaday.com/2016/04/28/autograph-a-string-art-printer/ (31.1.2024)

(17) Kaj je avtomatizacija. Arhsol. https://www.arhsol.si/avtomatizacija/ (1.2.2024)

(18) Guha, S. What advantages does automation bring to your daily life?. https://www.linkedin.com/pulse/what-advantages-does-automation-bring-your-daily-life-shreya-guha (3.2.2024)

(19) Dokumentacija Mega 2560 mikrokrmilnika. Arduino. https://docs.arduino.cc/hardware/mega-2560/ (27.1.2024)

(20) Slika Arduino mikrokrmilnika. Conrad. https://www.conrad.si/sl/p/arduino-plosca-mega-2560-core-191790.html (27.1.2024)

(21) Morse, R. Dokumentacija ramps1.4. https://all3dp.com/2/ramps-1-4-review-the-specs-of-this-controller-board/ (27.1.2024)

(22) Slika ramps1.4. 3DSVET. https://www.3dsvet.eu/izdelek/ramps-1-4/ (28.1.2024)

(23) Dokumentacija koračnega gonilnika a4988. Allegro microsystems. https://www.allegromicro.com/en/Products/Motor-Drivers/Brush-DC-Motor-Drivers/A4988 (28.1.2024)

(24) Slika ramps1.4. NKX motor. https://www.nkxmotor.si/en/shop/stepper-motor-driver/a4988/a4988-stepper-motor-driver-with-heatsink/ (28.1.2024)

(25) Pinout. Dokumentacija, ter slika koračnega motorja. DatasheetCafe. https://www.datasheetcafe.com/17hs4401-datasheet-stepper-motor/ (28.1.2024)

(26) Dokumentacija, ter slika arduino modula za SD kartico. 3Dsvet. https://www.3dsvet.eu/izdelek/modul-micro-sd-kartica/ (28.1.2024)

(27) Končno stikalo Njegovo načelo delovanja in strukture. Punto Marieno. https://sl.puntomarinero.com/end-switch-its-principle-of/ (28.1.2024)

(28) Napenjalnik niti. Aliexpress. https://www.aliexpress.com/item/1005004890943780.html?spm=a2g0o.order_list.order_list_main.17.57781802ZGvmwu (30.1.2024)

(29) halfmonty. Koda StringArtGenerator. https://github.com/halfmonty/StringArtGenerator (26.1.2024)

(30) Kaj je program solidworks. SolidWorld. https://www.solidworld.si/ (6.2.2024)

(31) Dokumentacija laserskega rezalnika FL-1390. DBK. https://www.dbk.si/en/Laser-machines/CO2-laser-machines/FL-series/ArtMID/637/ArticleID/83/Laser-engraver-FL-1390 (6.2.2024)

(32) Kaj je arduino. Arduino. https://www.arduino.cc/en/Guide/Introduction (8.2.2024)

(33) Kaj je koračni motor. MPS. https://www.monolithicpower.com/stepper-motors-basics-types-uses (8.2.2024)

(34) Slika nitne umetnine Dali Salvadorja. Saatchi Art. https://www.saatchiart.com/art/Sculpture-Salvador-Dali-String-Art-Portrait/1167867/4672471/view (18.2.2024)

(35) Slika Dali Salvadorja. MGM. https://www.mgm.mo/en/macau/art/art-tour/salvador-dali (18.2.2024)

ZAHVALA

Zahvaljujem se mentorju Jožefu Hrovatu za pomoč in vodenje pri opravljanju naloge, ter radodarno donacijo materiala za izdelavo.

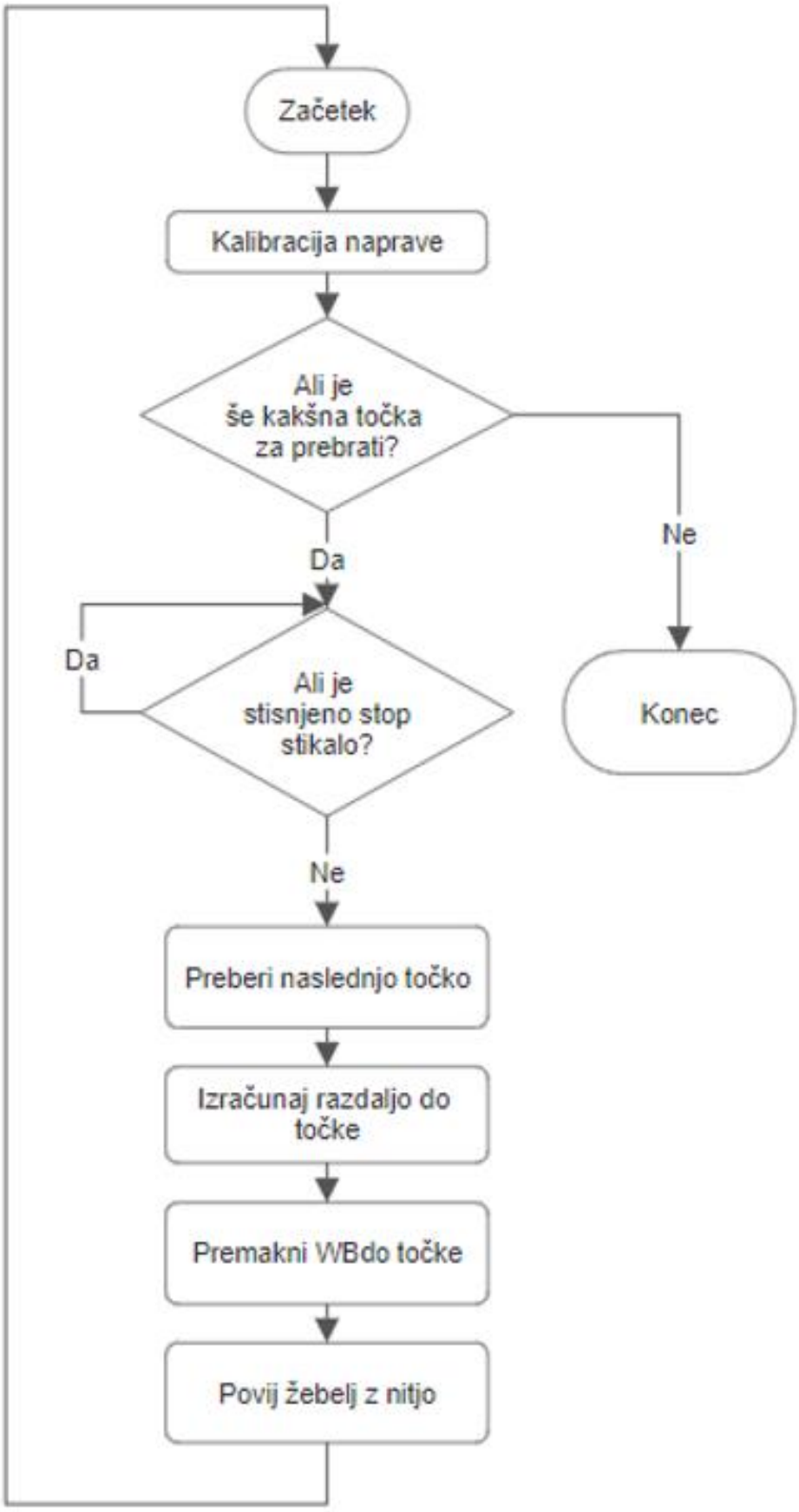
Zahvaljujem se profesorju Seitlu, za dovoljenje uporabe preše, ter donacijo materiala.

Zahvaljujem se mojemu dediju Silvu, za podporo pri izdelavi, donaciji materiala ter delitvi znanja na tem področju.

Zahvaljujem se profesorici dr. Nataši Meh Peer za njeno predano in natančno lekturo raziskovalne naloge.

Posebna zahvala velja staršem, ki mi omogočajo šolanje na ŠC Velenje – ERŠ.

PRILOGA A
DIAGRAM POTEKA KODE



PRILOGA B
KODA

```
#include <SPI.h>
#include <SD.h>

// Definiraj povezave za korake motorja
#define X_STEP_PIN 54
#define X_DIR_PIN 55
#define X_ENABLE_PIN 38

#define Y_STEP_PIN 60
#define Y_DIR_PIN 61
#define Y_ENABLE_PIN 56

// Definiraj stikalo za ničelno pozicijo
#define zero_pin 23
// Definiraj pin za začetek
#define start_pin 16
// Definiraj pin za ustavitev
#define pause_pin 17


// Konfiguracija razmerja zobnikov => Število zob na mizi / Število zob na
motorju
const int gearRatio = 6/1;
// numOfPins mora biti deljitelj števila 3200 (160, 200, 320, 400) -> (program
za točke gre samo do 288, zato uporabi 200)
const int numOfPins = 200;

float dLeft;
float dRight;

int Debug = 1; // 1 = način za odpravljanje napak, 0 = normalen način


////////////////////////////////////////////////////////////
//HIGH je smer urinega kazalca (dLeft)
//LOW je smer kontra uringea kazalca (dRight)
/*|       |
  | -   + |
  |--> <--|*/
////////////////////////////////////////////////////////////

// Nastavi ime datoteke na vrstici 99

int offset = 0; // Odmik od ničelne pozicije v enem premiku
int totalOffset = 0; // Skupni odmik od ničelne pozicije
int b = 1; // Da se zagotovi, da se stepTrackerSave shrani samo enkrat
const int zeroStepHigh = 529; // Pri katerem koraku bo visoka rotacija zadela
konec detektorja
```

PRILOGA B
KODA

```
const int zeroStepLow = 606; // Pri katerem koraku bo nizka rotacija zadela konec detektorja
int currentStep = 0; // Sledi trenutnemu položaju motorja
int currentLine = 0; // Sledi trenutnemu položaju v datoteki
int numOfLines = 0; // Za kalkulacijo koliko je že nareto
const int stepsPerRevolutionX = 3200;  // Spremeni to vrednost glede na specifikacije tvojega korakalnega motorja
const int stepsPerRevolutionY = 800;  // Spremeni to vrednost glede na specifikacije tvojega korakalnega motorja
const int stepsPerWorkpieceRevolution = stepsPerRevolutionX * gearRatio; // Število korakov, potrebnih za obračanje izdelka za eno polno vrtenje (3200*6=19200)

// Spremenljivke za pospeševanje
int acceleration = 1; // Določi, kdaj se izračuna pospeševanje (1 = vklopljeno, 0 = izklopljeno)
int currentDelay = 0; // Trenutna zakasnitev med koraki
const int maxDelay = 1000; // Maksimalna zakasnitev
const int minDelay = 200; // Minimalna zakasnitev
const int AcceleratedSteps = 700; // Število korakov za pospeševanje

const int stepsPerPoint = stepsPerWorkpieceRevolution / numOfPins; // Število korakov na točko
int currentPoint = currentStep / stepsPerPoint;

long startTime = 0;
long endTime = 0;

int stepTracker = 0; // Sledi korakom na trenutnem premiku
int stepTrackerSave = 0;
int findOffset = 0; // Nastavi, ko se izračuna odmik

const int chipSelect = 53; // CS pin za bralnik SD kartice
File myFile;
char buffer[10];

void setup() {
  pinMode(X_STEP_PIN, OUTPUT);
  pinMode(X_DIR_PIN, OUTPUT);
  pinMode(X_ENABLE_PIN, OUTPUT);

  pinMode(Y_STEP_PIN, OUTPUT);
  pinMode(Y_DIR_PIN, OUTPUT);
  pinMode(Y_ENABLE_PIN, OUTPUT);

  digitalWrite(X_ENABLE_PIN, LOW);
  digitalWrite(Y_ENABLE_PIN, LOW);
```

PRILOGA B
KODA

```
pinMode(start_pin, INPUT_PULLUP);
  pinMode(zero_pin, INPUT_PULLUP);
  pinMode(pause_pin, INPUT_PULLUP);

  Serial.begin(115200);

  // Inicializiraj SD kartico
  Serial.print("Inicializacija SD kartice...");
  if (!SD.begin(chipSelect)) {
    Serial.println("Inicializacija neuspešna!");
    while (1);
  }
  Serial.println("Inicializacija končana.");

  // Preveri če lahko odpreš datoteko
  // Datoteke na voljo (200 points): tin1300.txt, zoja3000.txt, anime-
preseren3000.txt, cat1300.txt, einstein3000.txt, ers3000.txt, marilin1300.txt,
smiley1300.txt, sšg03000.txt
  myFile = SD.open("ez.txt");
  if (!myFile) {
    Serial.println("Datoteka ni najdena");
    while (1);
  } else {
    Serial.print("Odpiranje datoteke:");
    Serial.println(myFile.name());
    Serial.println("Datoteka uspešno odprta");
  }

  // Preštej število vrstic v datoteki
  while (myFile.available()){
    String number = myFile.readStringUntil(',');
    numOfLines++;
  }
  myFile.seek(0);


  float totalSeconds = numOfLines * 4.8145;

  int hours = totalSeconds / 3600;
  int minutes = static_cast<int>(totalSeconds / 60) % 60;
  Serial.print("Predviden čas: ");
  Serial.print(hours);
  Serial.print("h ");
  Serial.print(minutes);
  Serial.println("min");


  // Počakaj na začetno sekvenco ali ukaz za začetek (start, q start)
  String command = "";
```

PRILOGA B
KODA

```
Serial.println("Čakam na začetno sekvenco ali ukaz 'start'");
  while (digitalRead(start_pin) != LOW) {
    if (Serial.available()) {
      command = Serial.readStringUntil('\n');

      if (command == "start") {
        Serial.println("Prejet ukaz za začetek");
        break;
      } else if (command == "q start") {
        Serial.println("Preskakujem findZeroPosition()");
        break;
      }
    }
  }
  if (command != "start" && command != "q start") {
    Serial.println("Prejeta začetna sekvenca");
  }

  delay(2000);
  startTime = millis();
  Serial.println(command);
  if (command != "q start") {
    // Najdi ničelno pozicijo
    findZeroPosition();
  }
}

////////////////////////////////////////////////////////////////////////////
void loop() {
  // Način za odpravljanje napak
  while (Debug == 1) {
    Serial.println("////////Način za odpravljanje napak////////");
    while (1) {
      if (Serial.available()) {
        String input = Serial.readStringUntil('\n');
        input.trim(); // Odstrani vodilne in končne presledke
        if (input == "end") {
          Debug = 0;
          findZeroPosition();
          break;
        } else {
          // Premik po korakih (d[n] [dir] -> d500 1)
          if (input.startsWith("d")) {
            int commaIndex = input.indexOf(',');
            if (commaIndex != -1) {
              int steps = input.substring(1, commaIndex).toInt();
              int direction = input.substring(commaIndex + 1).toInt();
              moveStepperX(steps, direction);
```

PRILOGA B
KODA

```
            Serial.print("Število korakov: ");
            Serial.print(steps);
            Serial.print(" Smer: ");
            Serial.println(direction);
          }
          // Ukaz za premik na točko
        } else if (input.startsWith("p")) {
          int pointIndex = input.substring(1).toInt();
          findOffset = 1;
          moveMotorToPoint(pointIndex);
          Serial.print("Premaknjeno na točko: ");
          Serial.println(pointIndex);
          findOffset = 0;
          // Ukaz za ponovno nastavitev referenčne številke
        } else if (input.startsWith("z")) {
          Serial.println("Trenutna visoka ničla: ");
          Serial.println(zeroStepHigh);
          int tempZeroStep = input.substring(1).toInt();
          Serial.println("Iskanje ničelne pozicije");
          // Premakni motor, dokler ni dosežena ničelna pozicija
          while (digitalRead(zero_pin) != LOW) {
            moveStepperX(1, HIGH);
          }
          moveStepperX((tempZeroStep), LOW);
          // Ponastavi trenutni korak
          currentStep = 0;
          Serial.println("Ničelna pozicija najdena");
        } else if (input.startsWith("l")) {
          loopNail();
        } else if (input.startsWith("r")) {
          int pointIndex = input.substring(1).toInt();
          // end loop and continue from selected
          for (int i = 0; i < pointIndex; i++) {
            readNextPointFromFile();
          }
          Serial.print("line set ");
          Serial.println(pointIndex);
          Serial.println(atoi(buffer));
          currentLine = pointIndex;
          break;
        }
      }
    }
  }
}

while (digitalRead(pause_pin) != LOW) {
  if (readNextPointFromFile()) {
```

PRILOGA B
KODA

```
      // Premakni motor na ciljno točko
      findOffset = 1;
      moveMotorToPoint(atoi(buffer));
      delay(150);
      // Povij žebelj
      loopNail();
      delay(150);
      findOffset = 0;

      currentLine++;
      Serial.print("Progress: ");
      float Progress = static_cast<float>(currentLine) * 100 / numOfLines;
      Serial.print(Progress);
      Serial.println("%");
      Serial.println(currentLine);


    } else {
      Serial.println("Končano!");
      endTime = millis();
      Serial.print("Čas izvajanja: ");
      int totalTime = (endTime - startTime) / 60000; // Pretvori v minute
(1000ms = 1s, 60s = 1min
      Serial.print(totalTime);
      Serial.println("min");
      while (1) {
        // Ne naredi ničesar
      }
    }
  }
  Serial.println("PROGRAM PAUSED");
}
////////////////////////////////////////////////////////////////////////////

void findZeroPosition() {
  Serial.println("Iskanje ničelne pozicije");
  // Premakni motor, dokler ni aktivirano končno stikalo
  while (digitalRead(zero_pin) != LOW) {
    moveStepperX(1, HIGH);
  }
  moveStepperX((zeroStepHigh), LOW);
  // Ponastavi trenutni korak
  currentStep = 0;
  Serial.println("Ničelna pozicija najdena");
  delay(500);
}

bool readNextPointFromFile() {
```

PRILOGA B
KODA

```
  if (myFile.available()) {
    // Preberi naslednje število iz datoteke
    String rawString = myFile.readStringUntil(',');
    rawString.trim();
    rawString.toCharArray(buffer, sizeof(buffer));
    return true;
  } else {
    myFile.close(); // Zapri datoteko, ko ni več števil
    return false;
  }
}

void moveMotorToPoint(int targetPoint) {
  currentPoint = currentStep / stepsPerPoint;

  Serial.println("////////////");
  Serial.print("Premikanje iz točke: ");
  Serial.print(currentPoint);
  Serial.print(" v točko: ");
  Serial.println(targetPoint);


  // Zračunaj razdalje v obe smeri
  dLeft = (currentPoint - targetPoint + numOfPins) % numOfPins;
  dRight = (targetPoint - currentPoint + numOfPins) % numOfPins;
  Serial.print("dLeft = ");
  Serial.println(dLeft);
  Serial.print("dRight = ");
  Serial.println(dRight);

  // Izberi katera smer je najkrajša (Low  = dRight (smer kontra uringa
kazalca), high = dLeft (smer uringa kazalca)) (dependent on orientation of
connector)
  int direction = (dLeft < dRight) ? HIGH : LOW;


  // Izračunaj za koliko korakov se premakniti
  int stepsToMove = min(dLeft, dRight) * stepsPerPoint;
  moveStepperX(stepsToMove, direction);
  Serial.print("Steps taken to point: ");
  Serial.println(stepsToMove);
  Serial.println("////////////");

  // Posodobi trenutno pozicijo
  currentStep = targetPoint * stepsPerPoint;
}

void loopNail(){
```

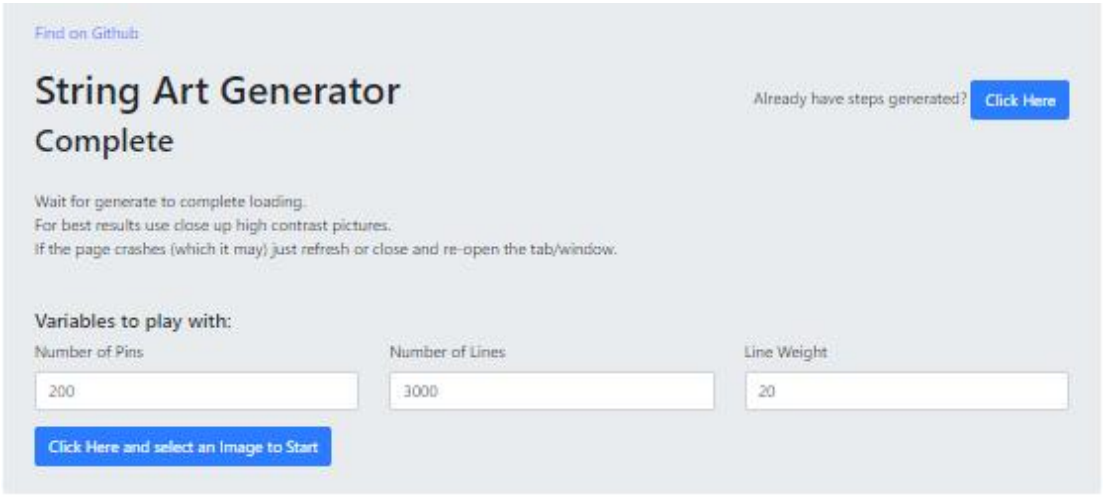
PRILOGA B
KODA

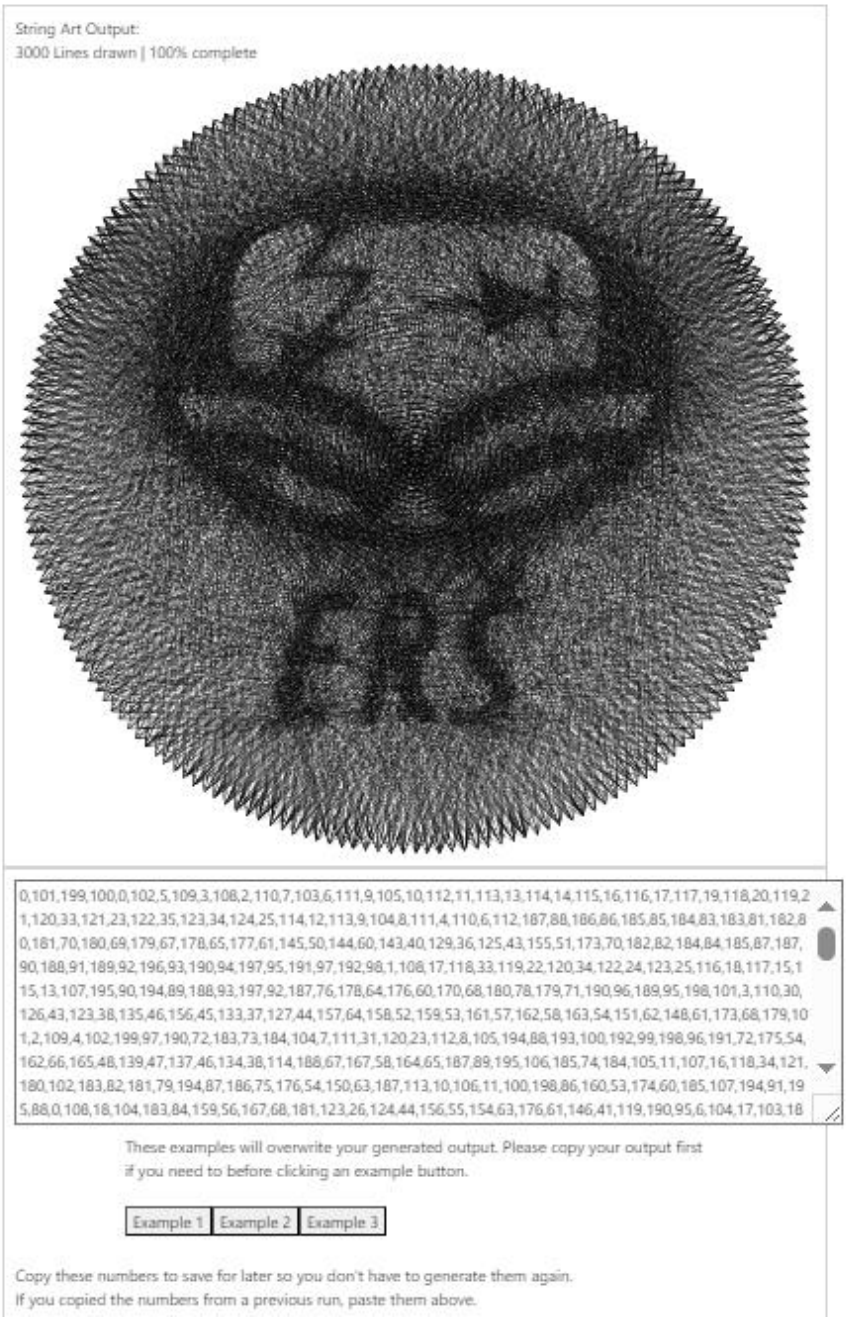
```
  moveStepperY(30, HIGH);
  delay(150);
  moveStepperX(stepsPerPoint, HIGH);
  currentStep -= stepsPerPoint;
  if (currentStep < 0) {
    currentStep = stepsPerWorkpieceRevolution + currentStep;
  } else if (currentStep > stepsPerWorkpieceRevolution) {
    currentStep = currentStep - stepsPerWorkpieceRevolution;
  }
  delay();
  moveStepperY(30, LOW);
}

void moveStepperX(int steps, int direction) {
  // Nastavi smer motorja
  digitalWrite(X_DIR_PIN, direction);
  // Premakni motor za določeno število korakov
  for (int i = 0; i < (steps); i++) {

    // Pospeševanje
    if (acceleration == 1 && steps > 999) {
      if (i < AcceleratedSteps) {
        currentDelay = map(i, 0, AcceleratedSteps, maxDelay, minDelay);
      }
      else if (i < steps - AcceleratedSteps) {
        currentDelay = minDelay;
      }
      else {
        currentDelay = map(i, steps - AcceleratedSteps, steps, minDelay,
maxDelay);
      }


      digitalWrite(X_STEP_PIN, HIGH);
      delayMicroseconds(currentDelay);
      digitalWrite(X_STEP_PIN, LOW);
      delayMicroseconds(currentDelay);
    } else {
      digitalWrite(X_STEP_PIN, HIGH);
      delayMicroseconds(500);
      digitalWrite(X_STEP_PIN, LOW);
      delayMicroseconds(500);
      }
}

void moveStepperY(int steps, int direction) {
  digitalWrite(Y_DIR_PIN, direction);
```

PRILOGA B
KODA

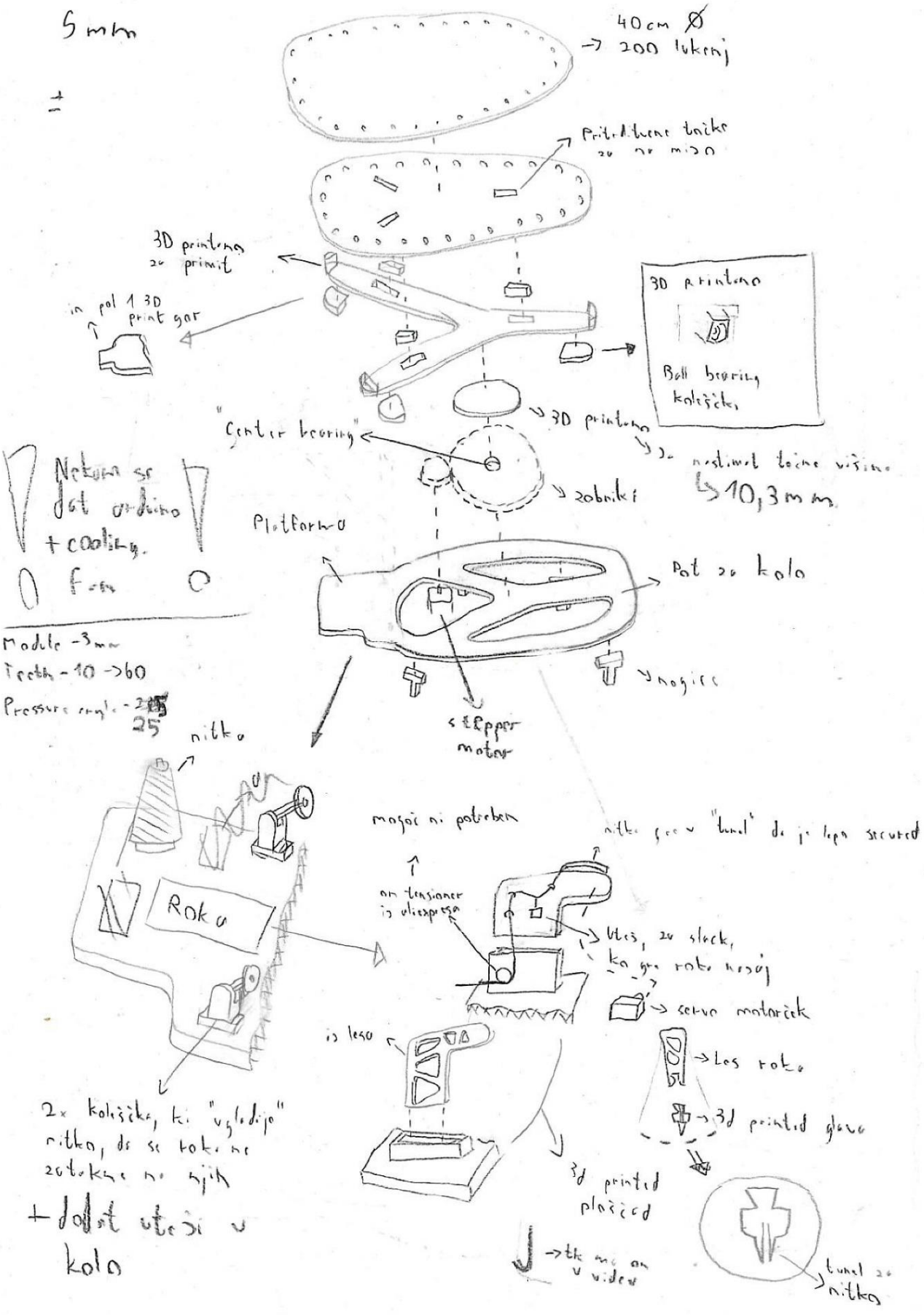
```
  for (int i = 0; i < (steps); i++) {
    digitalWrite(Y_STEP_PIN, HIGH);
    delayMicroseconds(1000);
    digitalWrite(Y_STEP_PIN, LOW);
    delayMicroseconds(3000);
  }
}
```

PRILOGA C
SPLETNI VMESNIK





PRILOGA D
KONCEPTNA SKICA



PRILOGA E
3D MODEL IZDELKA