Mestna občina Celje
Komisija Mladi za Celje

AVTOMATIZACIJA POMIKA DESNEGA VODILA NA FORMATNI ŽAGI SCM Si 400 NOVA

RAZISKOVALNA NALOGA

AVTORJA
Gašper Pistotnik
Andraž Pevcin

MENTORJA
Tomislav Viher, univ. dipl. org.
Karmen Kotnik, univ. dipl. inž.

Celje, marec 2017

Šolski center Celje
Gimnazija Lava

AVTOMATIZACIJA POMIKA DESNEGA VODILA NA FORMATNI ŽAGI SCM Si 400 NOVA

RAZISKOVALNA NALOGA

Avtorja:
Gašper Pistotnik, 2. e
Andraž Pevcin, 2. e

Mentorja:
Tomislav Viher, univ. dipl. org.
Karmen Kotnik, univ. dipl. inž.

Mestna občina Celje, Mladi za Celje

Celje, marec 2017

Povzetek

V raziskovalni nalogi sva opisala, kako sva avtomatizirala pomik desnega vodila na formatni žagi SCM Si 400 Nova, kar bo v pomoč mizarjem pri njihovem delu. To sva naredila z Genuinom in koračnim motorjem, saj sva želela izbrati najcenejšo pot in obenem ohraniti natančnost in uporabnost že obstoječih dražjih različic. Ta dodatek je uporaben tudi na ekvivalentnih strojih drugih tipov oziroma podjetij. Najin izdelek sva primerjala z izdelki, ki so trenutno na trgu. Ugotovila sva, da je razmerje med ceno in kakovostjo v primerjavi z drugimi rešitvami zelo dobro.

Zahvala

Zahvaljujeva se mentorjema Tomislavu Viherju in Karmen Kotnik za nesebično pomoč in podporo, ki sta nama jo nudila med nastajanjem te naloge. Prav tako se zahvaljujeva Simoni Jereb, prof., za jezikovni pregled naloge. Nenazadnje pa se ne smeva pozabiti zahvaliti najinim staršem, ki so ves čas verjeli v naju, nama zaupali in naju spodbujali z materialnimi sredstvi in strokovnimi nasveti.

Kazalo vsebine

Kazalo slik

1 UVOD

SCM Si 400 Nova je formatna žaga podjetja SCM iz Italije, prikazana na spodnji sliki. Iz izkušenj veva, da je najtežje tisto delo, za katerega vemo, da bi ga lahko opravili na lažji način, a si tega zaradi pomanjkanja znanja ali zaradi pomanjkanja denarja enostavno ne moremo privoščiti. Odločila sva se, da bova mizarjem pomagala olajšati delo in avtomatizirala pomik desnega vodila, pocenila serijsko izvedbo, ki se jo da dokupiti pri nabavi stroja ali kupiti kot dodatek, recimo TigerFence (https://www.tigerstop.com/products/tigerfence/), hkrati pa ohranila natančnost nastavljanja vodila.

Slika 1: Formatna žaga SCM Si 400 Nova
(Vir: http://www.lestroj.si/novi-stroji/formatna-zaga-scm-si-400-ep-nova.html).

1.1 Opis problema in rešitve

Mizarjev cilj je, da bi izdelke naredil čim natančneje, brez napak. To je pri ročnem nastavljanju desnega vodila skoraj nemogoče, saj se lahko hitro zmotimo. Malo smo nepozorni ali pa na merilni trak, ki se nahaja pod vodilom, gledamo pod napačnim kotom in že se zgodi napaka.

Slika 2: Žaga z ročno nastavitvijo desnega vodila
(Vir: http://www.lestroj.si/novi-stroji/formatna-zaga-scm-si-400-ep-nova.html).

To napako lahko omilimo z digitalnim odčitovalcem, a je vseeno nastavljanje zamudno, saj moramo za mero nad 50 cm iti okrog voza za vzdolžni rez in nastaviti desno vodilo.

Slika 3: Žaga z digitalnim odčitovalcem na desnem vodilu
(Vir: http://www.lestroj.si/novi-stroji/formatna-zaga-scm-si-400-ep-nova.html).

To zamudnost lahko rešimo z avtomatiziranimi pomiki vodil, a ti so cenovno manj dostopni kot navadne, ročno nastavljive žage. Pri teh sta dodana avtomatski dvig in nagib žaginega lista, na zaslonu pa se izpiše še hitrost vrtenja žaginega lista. Cena za avtomatski pomik desnega vodila se giblje od 3.128 € do 6.055 €. Za 3.125 € (READY 3 verzija) dobimo avtomatski pomik vodila, ki se premika z jekleno vrvjo. Mero odčituje na magnetnem traku, enako kot pri digitalnem odčitovalcu. Tipkovnica za vnos mere je na nepraktičnem mestu, pri stikalih za vklop in izklop stroja, kjer se pokrije z večjimi kosi obdelovanca. Za 4.112 € (READY 3 UP verzija) tudi dobimo pomik z jekleno vrvjo, a je tipkovnica za vnos nad žago, zraven pa so tudi stikala za vklop in izklop stroja. Pri zgoraj navedenih avtomatskih pomikih se vodilo pritrdi s pnevmatskim cilindrom, ki je označen na sliki.

Slika 4: Žaga s programatorjem READY 3 UP
(Vir: http://www.lestroj.si/novi-stroji/formatna-zaga-scm-si-400-ep-nova.html).

Slika 5: Cilinder za pritrditev vodila
(Vir: http://www.lestroj.si/novi-stroji/formatna-zaga-scm-si-400-ep-nova.html).

Za 6.055 € (READY 3 UP PLUS verzija) dobimo pomik z ležajnim navojnim vretenom, kjer se mera odčituje glede na število obratov navojnega vretena (z enkoderjem). (Vse zgoraj navedene cene so pridobljene pri podjetju Lestroj iz Trzina in ne vsebujejo DDV).

Odločila sva se, da bova avtomatizirala pomik desnega vodila z Arduinom, pocenila izvedbo in ohranila enako natančnost, kot je zagotovljena pri dodatku s strani SCM in TigerFence. Najin produkt bo kompatibilen z vsemi samostojnimi (to pomeni, da ne velja za kombinirne stroje, ki imajo lahko več operacij) formatnimi žagami. Po najinih izračunih naj bi za ves potreben material porabila približno do 600 €.

Slika 6: TigerFence (Vir: http://www.windowmachinery.com/products-positioner-tigerfence.html).

1.2 Hipoteze

- Z avtomatiziranjem pomika desnega vodila na formatni žagi SCM Si 400 Nova lahko izboljšamo natančnost njegovega nastavljanja.
- Avtomatizacija pomika desnega vodila omogoča hitrejše delo.
- Z Genuinom lahko naredimo cenejšo, a enako natančno različico, kot sta TigerFence in dodatek, ki ga lahko dokupimo pri prodajalcu SCM Si 400 Nova.

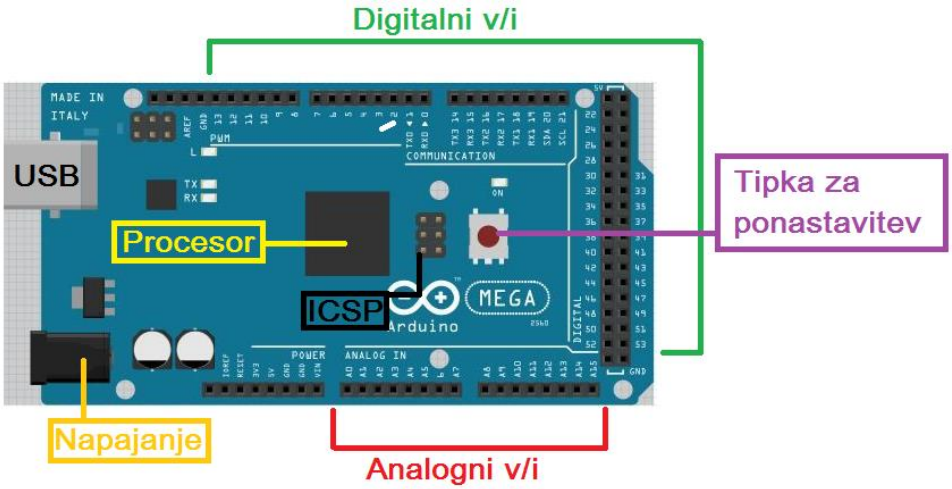
1.3 Metode dela

- Poiskala in primerjala bova že obstoječe rešitve.
- Za osnovo bova uporabila programsko rešitev s spletne strani Brainy-Bits, ki jo bova spremenila in nadgradila.
- S koračnim motorjem bova vrtela krogličnо vreteno, preko katerega se bo premikalo vodilo.
- S pomočjo nekaj jeklenih sestavnih delov bova matici pritrdila na vodilo.
- S konstrukcijskim jeklom bova pritrdila navojno vreteno in motor na stroj.
- Na formatni žagi SCM Si 400 Nova v mizarski delavnici bova testirala delovanje.

2 ARDUINO/GENUINO

Podjetje Arduino proizvaja strojno in programsko opremo, s katero lahko sestavljamo naprave, ki zaznavajo in kontrolirajo okolje. Arduinove plošče so odprtokodne in so komercialno dostopne, predsestavljene ali kot naredi-si-sam komplet. Večina jih uporablja Atmelov 8-, 16-, 32-biten AVR- mikrokrmilnik, plošče narejene po letu 2015 pa tudi druge vrste. Uporabljajo tudi različne vrste procesorjev, ki so večinoma Atmelovi in Intelovi. Na ploščah je več analognih ali digitalnih vhodov in izhodov, na katere lahko povežemo razširitvene plošče (shield, driver) ali druga vezja. Vse plošče imajo vmesnik za serijsko komunikacijo, ki ga uporabimo za nalaganje programov. To je največkrat USB (univerzalno serijsko vodilo). Podjetje Arduino je razvilo tudi programsko okolje (IDE), v katerem lahko pišemo program, ki se bo izvajal na mikrokrmilniku. Tega programiramo v programskem jeziku, ki je mešanica jezikov C in C++. Arduino priskrbi tudi dodatne knjižnice, ki omogočajo lažje programiranje.

Uporabila bova Genuino MEGA 2560. Plošče Genuino so identične ploščam Arduino, le da se prodajajo na evropskem trgu. To preimenovanje je delna rešitev za spor glede pravic do blagovnega imena Arduino (http://arduinox.si/). Najina plošča uporablja Atmelov mikrokrmilnik ATmega2560. Ima 54 digitalnih priključkov, 16 analognih priključkov, 4 serijske vhode strojne opreme, 16 MHz-kristalni oscilator, USB-povezavo, napajalni vhod, ICSP-glavo in tipko za ponastavitev. Za napajanje potrebuje enosmerni tok (DC) od 5 V do 12 V.

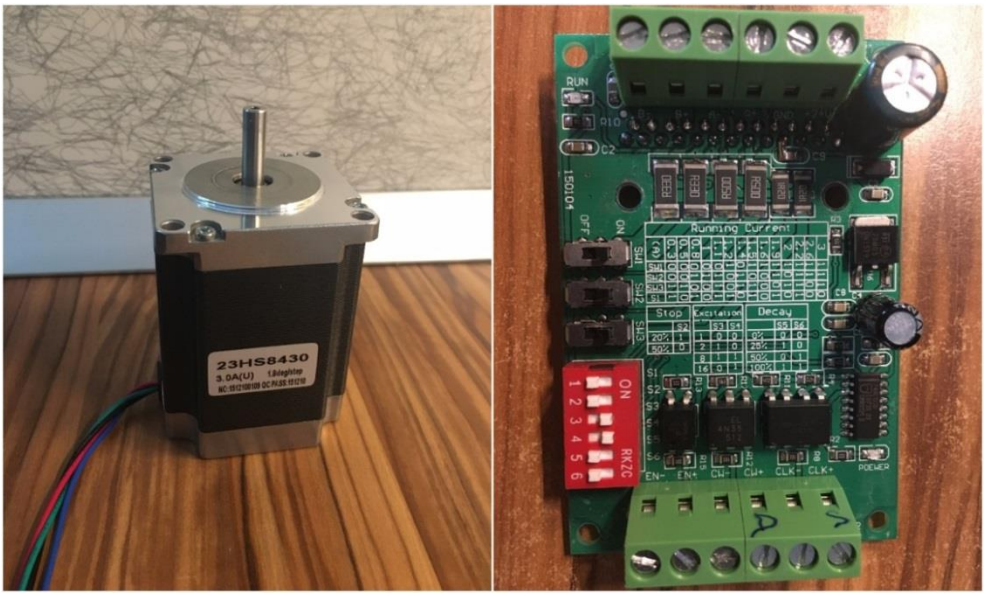


Slika 7: Matična plošča Genuino mega 2560.

3 DRUGE KOMPONENTE

3.1 Koračni motor in krmilnik

Sprva sva uporabila koračni motor NEMA 23 in krmilnik TB6560.



Slika 8: Koračni motor NEMA 23 in krmilnik TB6560.

Ko sva dobila kroglično vreteno in matici, sva ugotovila, da nama ne zadostuje, saj koračni motorji zelo izgubljajo navor s hitrostjo in nisva dobila zadostne potisne moči pri željeni hitrosti (približno 30 sekund od 3 mm do 1250 mm, če je pomik počasnejši, je ta dodatek neuporaben, to je 12,5 obratov na sekundo, 750 RPM). Zato sva uporabila drug motor, in sicer NEMA 34, ki ima pa tudi drug krmilnik DQ860MA in napajanje z napetostjo 60 V. Napajanje imava sedaj ločeno za matično ploščo ter motor, saj matična plošča ne deluje pod višjimi napetostmi kot 12 V.

Slika 9: Pretvornik iz 220-230 V na 60 V in krmilnik koračnega motorja.

Novi motor NEMA 34 ima pri najbolj optimalnih pogojih 8,7 Nm navora, kar nama zadostuje.

Slika 10: NEMA 34 koračni motor.

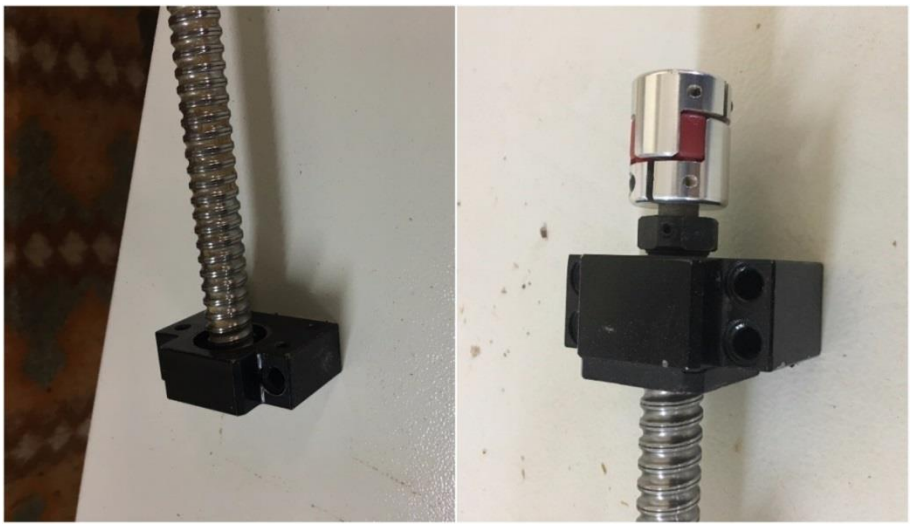
3.2 Krogličnо vreteno, matica, stranska ležaja in sklopka

Krogličnо vreteno sva uporabila, ker je najnatančnejše. Pri matici ni prostora, kot na primer pri trapeznem vretenu, pri katerem ima matica malo prostora, poleg tega pa se tudi obrablja. Prodajalec nama je povedal, da ima to vreteno korak 5 mm, to pomeni, da ko se vreteno obrne za en obrat, se matica prestavi za 5 mm. A ko sva to zmontirala na žago, sva ugotovila, da nama laže za 6 desetink milimetra, če prestaviva vodilo s 100 mm na 1250 mm. Nato sva program predelala, da je 1 mm meril 1 obrat. S tem sva dosegla to, da sva zasukala motor natančno za 200 obratov in odčitala razdaljo, ki se je pri tem zasuku spremenila. Na podlagi tega sva izračunala, da je korak vretena 4,998 mm. To sva izračunala tako, da sva odštela začetno pozicijo od končne in razliko delila z 200.

Slika 11: Linearni aktuator.

Matica ima kroglice, ki minimizirajo trenje, da se navojno vreteno čim lažje vrti in da se maksimira prenos moči z vretena na matico (iz krožnega gibanja v premo gibanje). Ta črna cev je v matici zato, da se kroglice v njej ne izgubijo. Tudi ko se matica daje na vreteno, je treba paziti, da se ta cev izpodrine z vretenom, da kroglice ne padejo iz matice.

Slika 12: Krogličnа matica.



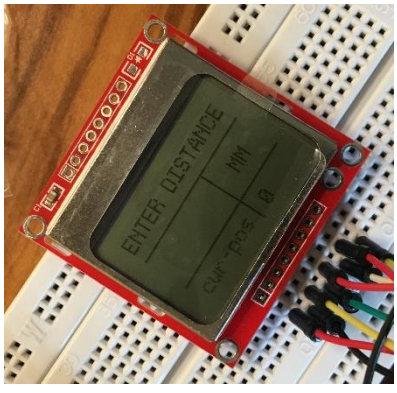
Slika 13: Stranska ležaja.

Stranska ležaja držita palico, da se ne premika, saj sta v nosilcih in ji omogočata, da se vrti z minimalnim trenjem. Sta pa dva: nesnemljiv in snemljiv ležaj. Nesnemljivi je pri motorju, da je vreteno fiksno vpeto v ležaj, tudi pritrjen je s štirimi vijaki, snemljivi ležaj je na drugem koncu vretena, pritrjen pa je z dvema vijakoma.

Slika 14: Sklopka

Sklopka prenaša gibanje iz motorja na navojno vreteno. Sklopka tudi omogoča, da se sname motor z vretena, ne da odtegnemo vijaka na sklopki, s katerima jo pritrdimo na vreteno in motorno os. Zaradi zamenjave motorja sva morala polovico sklopke, ki je pritrjena na motor, predelati, povečati luknjo s premerom 6 mm na 14 mm.

3.3 Zaslon



Slika 15: Zaslon Nokia 5110.

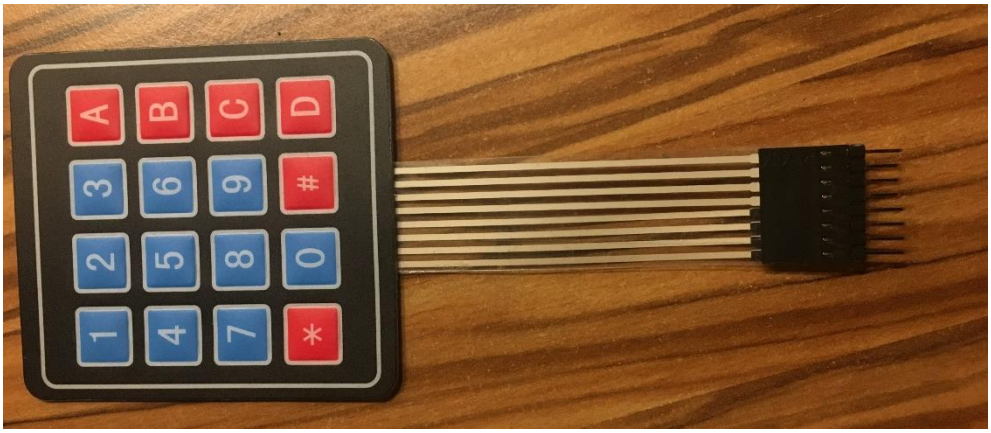
Sprva sva uporabila Nokijin zaslon 5110, a sva po posvetu z mizarjem ugotovila, da je premajhen, zato sva ga zamenjala z Adafruit ILI9341. Ta zaslon ima tudi zelo veliko prednost pred Nokijinim, saj ne potrebuje kablov za povezavo z matično ploščo. Potrebno je bilo samo prerezati 3 kontakte in pospajkati nove tri zaradi uporabe plošče Genuino mega 2560 (v rumenih pravokotnikih na sliki 16).

Slika 16: Prirejanje zaslona Adafruit ILI9341.

Slika 17: Zaslon Adafruit ILI9341.

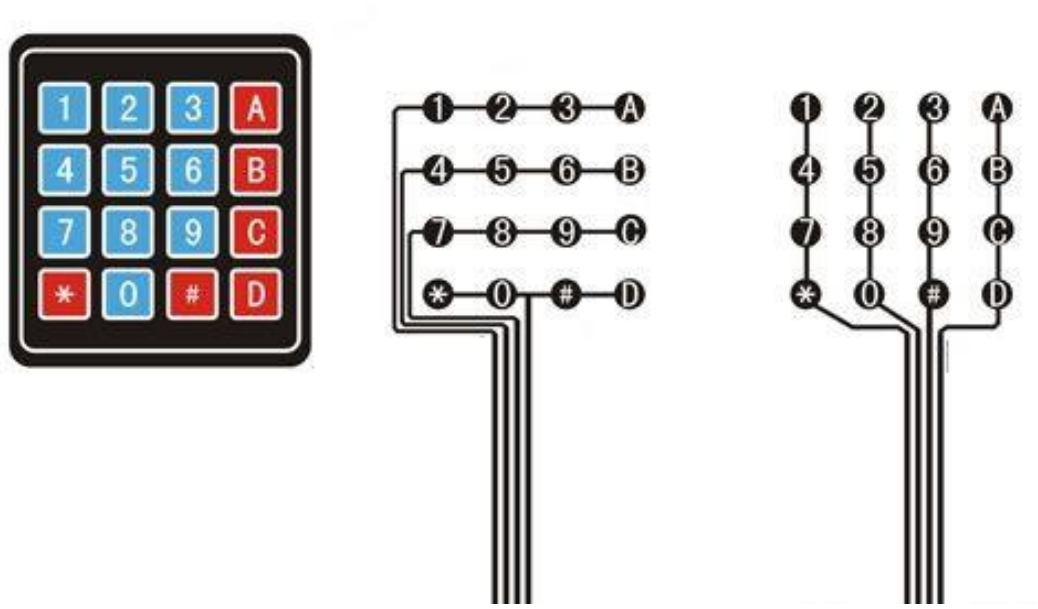
3.4 Tipkovnica

Uporabila sva 4 x 4-membransko tipkovnico z znaki: 1, 2, 3, 4, 5, 6, 7, 8, 9, *, 0, #.



Slika 18: 4 x 4-membranska tipkovnica.

Ta tipkovnica uporablja kombinacijo štirih stolpcev in štirih vrstic, s čimer pridobimo stanje tipk na mikrokontrolerju. Možnih ima 16 kombinacij in vsaka od teh ima svoj znak.



Slika 19: Shema 4 x 4-membranske tipkovnice.

Če program na primer naredi na vseh priključkih za vrstice (slika 19 leva shema) "LOW", na priključkih za stolpce pa izmenično naredi (slika 19 desna shema) "HIGH"**,** potem program bere vhode za vrstice, in kjer je "HIGH" (to pomeni, da je bil zaznan kontakt) lahko določi, katera tipka je bila pritisnjena. ("HIGH" je stanje na priključku, ko napetost na priključku ni enaka 0 V, "LOW" je stanje na priključku, ko je napetost na priključku enaka 0 V.)

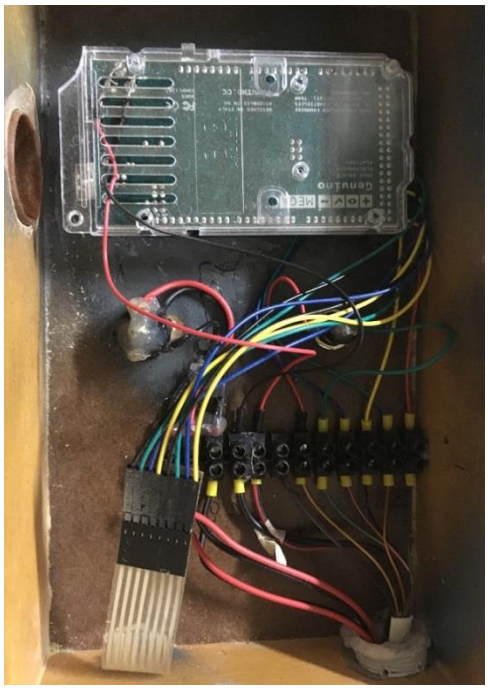
3.5 Ohišje

3.5.1 Ohišje za primarno elektroniko

V ohišje za primarno elektroniko sva namestila zaslon, ki je prilepljen na ohišje. Nanj sva priključila Genuina. Dve okrogli luknji sta za stikali, desno zeleno stikalo je za prižiganje in ugašanje elektronike za pomik desnega vodila, v njem gori lučka, ko je prižgano, rdeče tipkalo pa je za ugašanje formatne žage in predrezila.

Slika 20: Ohišje za primarno elektroniko.

Tipkovnica je preko 8 podaljškov pritrjena na mikrokrmilnik.



Slika 21: Pogled od zadaj v ohišje za primarno elektroniko.

3.5.2 Ohišje za sekundarno elektroniko

V ohišju za sekundarno elektroniko sta dva pretvornika z 220 V na 60 V in na 12 V. Za napajanje Genuina se potrebuje od 5 V do 12 V, a ker sva sprva imela koračni motor NEMA 23, sva uporabila kar 12 V-napajalnik, da si nisva delala dodatnih stroškov z nakupom napajalnika z manjšo napetostjo. Za napajanje koračnega motorja pa sva uporabila 60 V-pretvornik. Poleg njiju se nahaja še krmilnik za koračni motor.

Slika 22: Pogled v ohišje za sekundarno elektroniko.

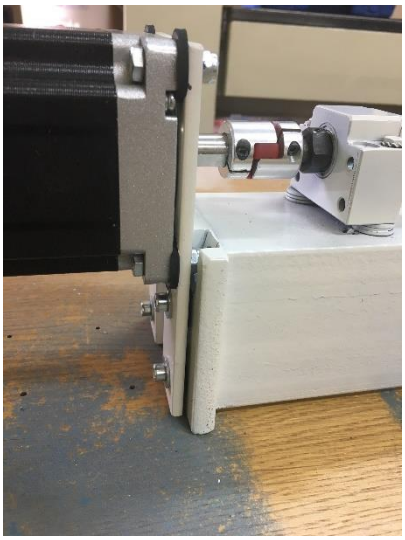
3.6 Pritrditev linearnega aktuatorja

Pod vodilo, ki je na formatni žagi, je pritrjeno vroče valjano kotno konstrukcijsko jeklo 8 x 8 cm in dolžine 162 cm. Nanj sta na vsaki strani pritrjena nosilca vretena, na strani, na kateri je motor, pa je privarjena plošča, da sva dobila ravnino. Na to ravnino sva privijačila dve vroče valjani ploščati konstrukcijski jekli. Med kotnim in ploščatim konstrukcijskim jeklom je gumijasta podložka, da minimizira tresljaje motorja, ki bi se prenesli na žago, in s tem utiša delovanje motorja.

Slika 23: Linearni aktuator na nosilcu.

Slika 24: Vroče valjani ploščati konstrukcijski jekli.

Na dve vroče valjani ploščati konstrukcijski jekli je privit motor, vmes pa so tudi gumijaste podložke.



Slika 25: Pritrditev motorja na nosilec.

To kotno jeklo sva pritrdila na žago že s pomočjo obstoječih lukenj, saj s tem ne vplivava na garancijo stroja.

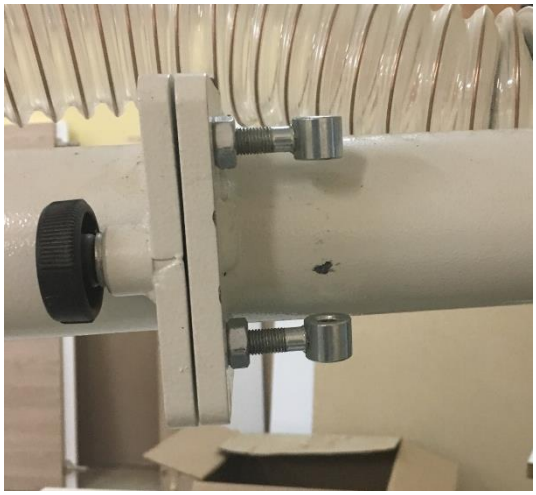
Vodilo in pogon sta povezana s posebej izdelanim sestavnim delom, ki trdno spaja ta dva elementa in omogoča prestavljanje vodila preko matice.

Slika 26: Posebej izdelan sestavni del.

Ta sestavni del sva s štirimi navojnimi zatiči pritrdila na vodilo.

Slika 27: Slika pritrjenega linearnega aktuatorja.

Ohišje za primarno elektroniko sva nastavila nad glavna stikala na stroju in nad žaginim listom, da ni preveč v napoto, kot je vidno na sliki 27. To stojalo je z vratnim okovjem pritrjeno na obstoječo roko za odsesovalno kapo.



Slika 28: Nosilca za nosilec primarne elektronike.

Uporabila sva kar staro stojalo za televizijo, ki je sicer dosti večje od ohišja, a se lahko zraven z magnetom pripne še list, na katerem so mere oz. optimizacijski list.

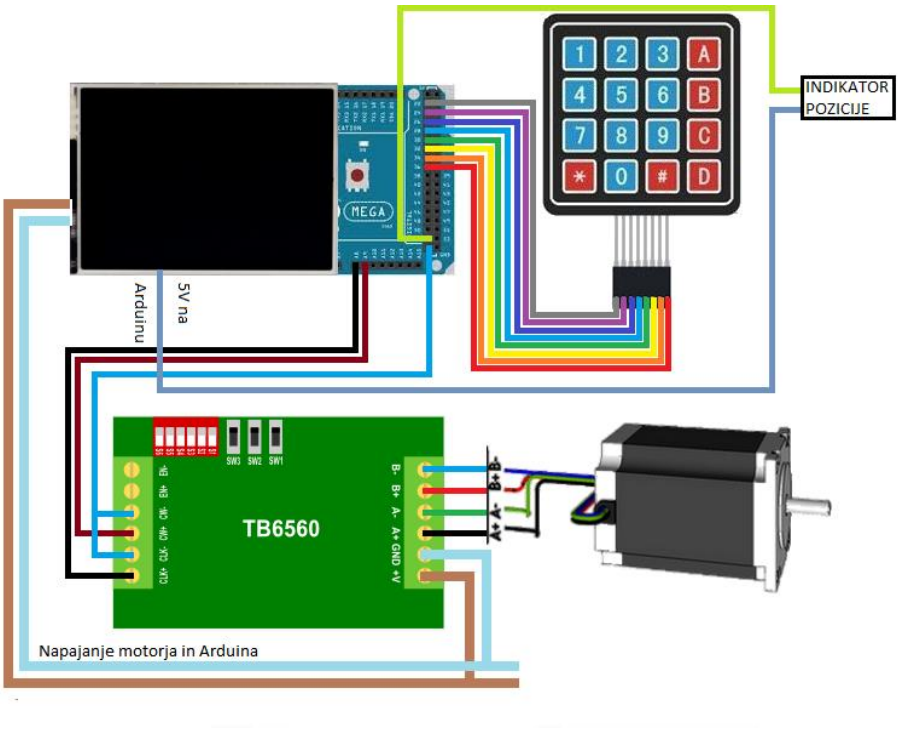
Slika 29: Nosilec primarne elektronike.

Slika 30: Slika stroja z dodatkom.

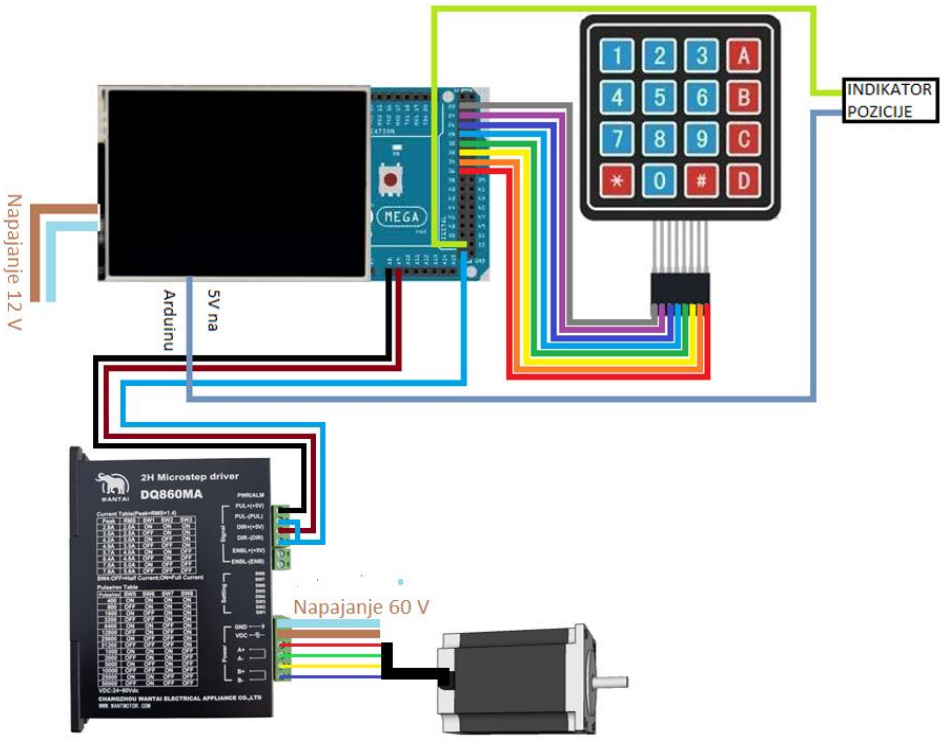
Povezavo med ohišjem za primarno in sekundarno elektroniko sva speljala po kabelski cevi, da so kabli zavarovani. Kabelsko cev sva speljala po roki za odsesovalno kapo. Uporabila sva tri vodnike 2 x 0,75 mm^2, rdeč/črn, ti so za zeleno stikalo, ki ugaša in prižiga elektroniko za napajanje Genuina, ter ploščat omrežni kabel UTP za prenos signalov z Genuina na krmilnik koračnega motorja in za ugašanje žage.

Slika 31: Napeljava po roki za odsesovanje.

4 SHEMA IN POVEZAVE MED KOMPONENTAMI



Slika 32: Shema z motorjem NEMA 23.



Slika 33: Shema z motorjem NEMA 34.

- Tipkovnica je povezana na digitalne priključke 22, 24, 26, 28, 30, 32, 34, 36.
- Krmilnik motorja je povezan na analogne priključke A8 (na krmilniku PUL+ oz. CLK+), A9 (na krmilniku DIR+ oz. CW+), PUL- oz. CLK- in DIR- oz. CW- na krmilniku sta pa povezana na gnd vhod/izhod.
- Ekran je narejen tako, da se pritrdi neposredno na matično ploščo, zato je povezan z naslednjimi vhodi/izhodi, ki so na shemi pod njim: SCL, SDA, AREF, GND, IOREF, RESET, 3,3 V, 5 V, Vin, ICSP glavo (MISO, SCK, Reset, Vcc, MOSI, GND), digitalni priključki 0-13 in analogni priključki A0-A5.
- V matično ploščo pride enosmerni tok (DC) z napetostjo od 7 V do 12 V.
- Na krmilniku je povezana na B- modra žica motorja, na B+ rumena oz. rdeča žica motorja, na A- zelena žica motorja, na A+ rdeča oz. črna žica motorja. Na vrata VDC oz. V+ je priklopljen fazni vodnik napajalnika (s 60 V oz. 12 V napetosti), na vrata GND pa ničelni vodnik napajalnika.

Na shemah je narisan indikator pozicije, ki ga nisva uporabila, saj sva pri poskusu kalibracije uničila kar dva Genuina. Zaradi tega sva ga izvzela iz projekta. Ugotovila pa sva tudi, da kalibracije ni potrebno izvajati, saj se pri več zaporednih nastavitvah natančnost vodila ne izgubi.

5 PRISTOP K PROGRAMSKI REŠITVI

Ker sva pri programiranju začetnika (programirati sva se naučila pri projektih z roboti Lego in s pomočjo spletnih vodičev), sva si na začetku pomagala z že narejenim programom, ki je uporabljen za podoben namen, kot je najin.

Za osnovo sva uporabila program v mešanici jezikov C in C++ z dodatnimi knjižnicami s spletne strani Brainy Bits (https://brainy-bits.com/tutorials/diy-stepper-miter-box/). Urejala sva ga v programskem okolju Arduino IDE, ki je brezplačen in privzet urejevalnik za Arduino/Genuino. Uporabila sva Genuino Mega 2560 kot mikrokrmilnik, EasyDriver kot krmilnik za koračni motor, Nokia 5110 kot prikazovalnik trenutne mere in mero, ki jo vnašamo, adapter za računalnik (12 V, 4.0 A), membransko tipkovnico 4 x 4 (z znaki 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, #, A, B, C, D) in močnejši motor, kot je bil na spletni strani Brainy Bits, Nema 23 (3.0 A, 1.8 stopinj/korak).

Ko sva dobila te stvari, sva ugotovila, da sva bila nepozorna na specifikacije krmilnika za koračni motor. EasyDriver podpira koračne motorje do 0.75 mA, imela pa sva 3.0 A motor. Zamenjala sva ga s krmilnikom TB6560, ki podpira do 3.5 A-koračne motorje.

Po posvetu z mizarjem sva še ugotovila, da imava premalo viden ekran oz. da so številke premajhne, zato sva zamenjala še ekran z Adafruit ILI9341 ekranom na dotik. Ekran Nokia 5110 je meril 61 mm v diagonalo, ekran Adafruit ILI9341 pa meri 71 mm, a je razlika zelo velika. ILI9341 ima barvni zaslon, zato se bolje opazi, ker ima večji kontrast. Zaslon je tudi občutljiv na dotik.

Ko sva dobila navojno vreteno, sva ugotovila, da ima motor premalo moči, saj s povečanjem hitrosti vrtenja zelo izgublja navor. Zato sva naročila nov motor, in sicer NEMA 34 z 8,7 Nm navora. Ta motor ima tudi drug krmilnik, DQ860MA in drug napajalnik, saj motor potrebuje tok 5,6 A in ima večjo napetost, 60 V.

Prizadevala sva si narediti čim bolj strnjen program, kar pomeni, da bo čim preglednejši. Program s spletne strani Brainy Bits sva uporabila le za osnovo, s pomočjo katere sva napisala svoj program, ki je prilagojen najini strojni opremi.

Ko vnesemo novo številko v mm, se številke sproti prikazujejo na zaslonu. Na tipkah A, B, C in D so nastavljene določene mere, ki se največkrat uporabljajo. Tipko samo pritisnemo in vodilo se premakne. Če se zmotimo, lahko zbrišemo številko z *. Ko vnesemo željeno številko, z # poženemo koračni motor, da prestavi vodilo na željeno mero. Program preračuna korake, ki so potrebni za premik vodila z ene mere na drugo mero.

$$\frac{(l_1 - l_0) * 400}{4{,}998}$$

V enačbi je l_1 vrednost, ki smo jo vpisali, l_0 pa vrednost trenutne mere. Vse skupaj množimo s 400 zato, da dobimo podatek, koliko korakov potrebujemo za ta premik (koračni motor ima le 200 korakov, a krmilnik te korake razdeli še vsaj na polovico, lahko tudi na manjše korake, zato 400, ker je to najmanjše možno število korakov za ta krmilnik). Na koncu še vse skupaj delimo s 4,998, kar je korak navojne palice (korak navojne palice

je pot, ki jo opravimo pri enem obratu). Iz te formule dobimo podatek, za koliko korakov se mora prestaviti koračni motor. Ko se motor prestavi, se ta mera zapiše v pomnilniške celice in se izpiše kot trenutna mera na zaslonu. Mera, ki jo vnašamo, je omejena med 3 in 1250, zato da se ne približa preveč žaginemu listu oz. da se ne oddalji preveč in česa ne poškoduje.

6 PROGRAM

#include

```
//knjižnice
#include <SPI.h> //Komunikacija med mikrokontrolerjem in zaslonom
#include <Adafruit_GFX.h> //Grafične funkcije zaslona
#include <gfxfont.h> //Prepoznavanje pisav
#include <Adafruit_ILI9341.h> //Prikazovanje na zaslonu , grafične funkcije
#include <EEPROM.h> //Shranjevanje vrednosti v Genuinov spomin
#include <Keypad.h> //Funkcije za tipkovnico
#include <AccelStepper.h> //Funkcije za upravljanje koračnega motorja
#include <MultiStepper.h> //Funkcije za upravljanje več koračnih motorjev
```

Slika 34: Program: knjižnice.

V program vse knjižnice in druge datoteke, ki jih potrebujeva.

- SPI (Serial Peripheral Interface) omogoča komunikacijo med mikrokrmilnikom in zaslonom.
- Adafruit_GFX in Adafruit_ILI9341 vsebujeta funkcije, ki olajšujejo prikaz na zaslonu.
- Knjižnica gfxfont pomaga mikrokrmilniku razumeti pisave.
- EEPROM omogoča shranjevanje podatkov v Genuinov spomin (flash pomnilnik velikosti 4 KB).
- Keypad vsebuje funkcije za lažje programiranje tipkovnice.
- AccelStepper in MultiStepper vsebujeta funkcije za lažje upravljanje koračnega motorja (regulirata pospeševanje, pojemanje, hitrost in omogočata krmiljenje več koračnih motorjev naenkrat), ne delujeta ena brez druge.

```
//pisave
#include <Fonts/FreeSans9pt7b.h>
#include <Fonts/FreeSansBold24pt7b.h>
#include <Fonts/FreeSans18pt7b.h>
```

Slika 35: Program: pisave.

Poleg tega dodava tri pisave, ki sva jih uporabila. Te pisave Genuino razume s pomočjo knjižnice "gfxfont".

Globalne spremenljivke

```
//spremenljivke
byte stevilke[] = {99, 99, 99, 99}; //seznam števk iz katerih nastane število za premik
int i = 0; //pove koliko števk je vpisanih
int steviloZaPrikaz; //stevilo, ki se prikaze na zaslonu
int celoStevilo; //število, ki pove za koliko se mora pomakniti koračni motor premakniti
String trenutnaMera = ""; //Trenutna pozicja desnega vodila (ko je vodilo stacionarno)
int zacetnaMera; //
```

Slika 36: Program: spremenljivke.

V tem sklopu sva definirala vse globalne spremenljivke.

- V tabeli "stevilke" se nahajajo številke, iz katerih nastane mera, na katero se mora premakniti vodilo. Vsebuje štiri elemente, ki so na začetku nastavljeni na 99 (99 namreč pomeni, da ta številka na tem mestu še ni bila vpisana). Tabela je tipa "byte", to pomeni, da je vsak element število z intervala [0, 255].
- Število "i" pove, koliko številk je vpisanih do sedaj. Tudi ta spremenljivka je tipa "byte". Na začetku je nastavljen na 0, ker še ni bila vpisana nobena številka.
- V spremenljivkah "steviloZaPrikaz" in "celoStevilo" je shranjena velikost premika v milimetrih. Vrednost jima je dodeljena šele pozneje v programu.
- Niz znakov ("String") "trenutnaMera" vsebuje trenutno mero (pozicijo) vodila, ko se to ustavi. Ta niz se po premiku prikaže na zaslonu.

```
//tipkovnica
const byte ROWS = 4; //število vrstic v tipkovnici
const byte COLS = 4; //število stolpcev
//tu so definirane tipke na tipkovnici
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
}; //seznam vseh tipk na tipkovnici
//tu so definirani vhodi/izhodi za tipkovnico
byte rowPins[ROWS] = {23, 25, 27, 29}; //priključki za vrstice
byte colPins[COLS] = {31, 33, 35, 37}; //priključki za stolpce
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ); //definicija tipkovnice
```

Slika 37: Program: definicija tipkovnice.

Tipkovnico definirava s konstruktorjem "Keypad" (iz knjižnice Keypad.h). Ta konstruktor potrebuje naslednje argumente:

- "Keymap", ki ga naredi funkcija "makeKeymap()" iz dvodimenzionalne tabele znakov "keys";
- priključke vrstic in stolpcev, s pomočjo katerih določimo vrstico in stolpec, v katerem je pritisnjena tipka;
- število vrstic in stolpcev.

Adafruit_ILI9341'

```
//zaslon
#define TFT_DC 9
#define TFT_CS 10
#define BLACK 0x0000 //črna barva
#define WHITE 0xFFFF //bela barva
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC); //definicja zaslona
```

Slika 38: Program: definicija zaslona.

S konstruktorjem "Adafruit_ILI9341" (iz knjižnice Adafruit_ILI9341) in stavkom #define definirava vse, kar je povezano z zaslonom.

- Priključek 9 služi kot DC-priključek, ki priskrbi zaslonu 5 V-enosmernega toka.
- Priključek 10 služi kot CS ("chip select")-priključek, s pomočjo tega pa mikrokrmilnik izbere enega od čipov v zaslonu.
- Barve so zapisane v šestnajst bitnem zapisu (prvih 5 bitov pove nasičenost rdeče barve, naslednjih 6 pove nasičenost zelene barve, zadnjih 5 pa nasičenost modre barve). Za predstavitev šestnajstih bitov potrebujemo 4 šestnajstiške številke. Da pa mikrokrmilnik prepozna številko v šestnajstiškem sistemu, mu moramo dodati kodo "0x" pred zapisom barve.
- Konstruktor "Adafruit_ILI9341" potrebuje dva argumenta, in sicer DC-priključek (9) in CS-priključek (10).

AccelStepper

```
//koračni motor
AccelStepper stepper(1, A12, A11); //definicija koračnega motorja
```

Slika 39: Program: definicija koračnega motorja.

Koračni motor definirava s konstruktorjem "AccelStepper". Ta potrebuje tri argumente:

- "MotorInterfaceType" pove mikrokrmilniku, na kakšen način je nanj povezan koračni motor. V tem primeru je na tem polju vpisano število 1, to pomeni, da sta povezana preko krmilnika ("driverja");
- "Step" priključek, ki daje krmilniku koračnega motorja signale. Za vsak signal se koračni motor obrne za en korak (v najinem primeru za 0,9 °);
- "Dir" priključek, ki pove motorju, v katero smer naj se premika (5 V je v negativni smeri, 0 V pa v pozitivni smeri);
- s "stepper.setMaxSpeed()" in "stepper.setAcceleration()" se nastavi najvišja hitrost vrtenja in pospešek motorja, a to se nastavi v "setup-u()".

6.1 Setup

EEPROM.read()

```
//z EEPROM.read preberemo iz pomnilnika vrednosti, ki so zapisane, v najinem primeru je to trenutna mera
if (EEPROM.read(3) != 99) {
  zacetnaMera = EEPROM.read(0) * 1000 + EEPROM.read(1) * 100 + EEPROM.read(2) * 10 + EEPROM.read(3);
}
else if ( EEPROM.read(2) != 99) {
  zacetnaMera = EEPROM.read(0) * 100 + EEPROM.read(1) * 10 + EEPROM.read(2);
}
else if ( EEPROM.read(1) != 99) {
  zacetnaMera = EEPROM.read(0) * 10 + EEPROM.read(1);
}
else {
  zacetnaMera = EEPROM.read(0);
}
trenutnaMera = String(zacetnaMera);
```

Slika 40: Program: EEPROM.read.

Ko se mikrokrmilnik ugasne, se trenutna mera shrani v njegov spomin. To narediva tako, da se vsaka števka te mere shrani v svojo pomnilniško celico. Ob zagonu pa mora program prebrati te vrednosti, da ve, kje se nahaja vodilo. To naredi tako, da prebere pomnilniško besedo [3], če vrednost te ni enaka 99, potem ve, da je mera štirimestno število (vse vrednosti pomnilniških besed so različne od 99). Iz tega lahko sklenemo, da bo v pomnilniški besedi [0] zapisana tisočica, v [1] stotica, v [2] desetica in v [3] enica, zato lahko zapišemo shranjeno število kot (v oglatih oklepajih so naslovi pomnilniških besed):

$$10^3 * [0] + 10^2 * [1] + 10 * [2] + [3]$$

Če je pomnilniška beseda [3] enaka 99, vemo, da ima število 3 ali manj mest, zato prebere pomnilniško besedo [2]. Če je njena vrednost različna od 0, vemo, da je v pomnilniški besedi [0] zapisana stotica, v [1] desetica in v [2] enica. Shranjeno število izračunamo kot:

$$10^2 * [0] + 10 * [1] + [2]$$

Če je [2] enaka 99, bo program preveril pomnilniško besedo [1]. Če vrednost te ni enaka 99, je v [1] zapisana enica, v [0] pa desetica. Shranjeno število izračunamo kot:

$$10 * [0] + [1]$$

Če je vrednost pomnilniške besede [1] enaka 99, program ve, da je shranjeno število enomestno, to pa pomeni, da je v [0] enica. Shranjeno število izračunamo kot:

$$[0]$$

tft

V funkciji "setup()" je nastavljen tudi ekran.

```
tft.setRotation(3);//nastavi se orientacija zaslona
tft.fillScreen(white);//nastavi se barva ozadja
tft.drawFastHLine(0, 140, 320, BLACK);//nariše se črta, ki ločuje predel z trenutno mero
tft.drawFastHLine(0, 141, 320, BLACK);// in mero, ki jo vnašamo, dvakrat zato, da je bolj vidna
tft.setTextColor(BLACK);//nastavi se barva pisave
tft.setFont(&FreeSans9pt7b);//nastavi se vrsta pisave
tft.setCursor(0, 15);//postavi kazalko na koordinato (0, 15)
tft.println("VNESI MERO");//pri kazalki se izpiše besedilo
tft.setCursor(5, 130);
tft.setFont(&FreeSansBold24pt7b);
tft.setCursor(230, 130);
tft.setFont(&FreeSans18pt7b);
tft.println("mm");
tft.setCursor(0, 155);
tft.setFont(&FreeSans9pt7b);
tft.println("TRENUTNA POZICIJA");
tft.setCursor(5, 230);
tft.setFont(&FreeSansBold24pt7b);
tft.println(trenutnaMera);
tft.setCursor(230, 230);
tft.setFont(&FreeSans18pt7b);
tft.println("mm");
```

Slika 41: Program: nastavi zaslon.

- "tft.setRotation()" nastavi orientacijo zaslona,
- "tft.fillScreen()" zapolni ozadje ekrana,
- "tft.drawFastHLine()" nariše horizontalno črto (HLine), in sicer nariševa 2, da sta bolj vidni,
- "tft.setTextColor()" nastavi barvo pisave,
- "tft.setFont()" nastavi vrsto pisave,
- "tft.setCursor()" postavi kazalko na koordinate,
- "tft.println()" zapiše neko vrednost (številsko ali zaporedje znakov "String"),
- kasneje v programu z "tft.fillRect()" nariševa zapolnjen pravokotnik (z enako barvo, kot je ozadje), ki se nahaja na območju, na katerem se je spremenila številka, da je vidna samo nova številka. To je bolj praktično, saj ni treba osveževati celega zaslona.

6.2 Loop

Funkcija "loop()" se začne izvajati, ko se konča funkcija "setup()" in se ponavlja, dokler ne ustavimo programa ali mikrokrmilniku ne prekinemo napajanja.

```
char keypressed = keypad.getKey();  //shrani tipko, ki je bila pritisnjena
//poda vrednost pritisnjene tipke na tipkovnici
if (keypressed != NO_KEY) { //Perveri če je bil gumb pritisnjen
  switch (keypressed) {
```

Slika 42: Program: preveri, če je bila pritisnjena tipka.

V njej je sprva definirana spremenljivka tipa "char" (znak) z imenom "keypressed". Dodeljena ji je vrednost metode "keypad.getKey()", ki vrne trenutno pritisnjeno tipko na tipkovnici, če ni pritisnjena nobena tipka, pa vrne ključno besedo: "NO_KEY". Program preveri, ali je bila na tipkovnici pritisnjena tipka. Če je pritisnjena katerakoli tipka, se s stavkom "switch" še preveri, katera tipka je to bila. Ta stavek nam omogoča, da določeno spremenljivko primerjamo s seznamom različnih vrednosti. Če je vrednost spremenljivke enaka vrednosti s seznama (case), se bo izvedla koda, ki je zapisana pod njim.

```
//case 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 dodajo v seznam številk, izpiše jo tudi na ekranu
case '1':
  dodajStevko(1);
  break;
case '2':
  dodajStevko(2);
  break;
case '3':
  dodajStevko(3);
  break;
case '4':
  dodajStevko(4);
  break;
case '5':
  dodajStevko(5);
  break;
case '6':
  dodajStevko(6);
  break;
case '7':
  dodajStevko(7);
  break;
case '8':
  dodajStevko(8);
  break;
case '9':
  dodajStevko(9);
  break;
case '0':
  dodajStevko(0);
  break;
```

Slika 43: Program: vrednosti tipk 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Če pritisnemo številko na tipkovnici, to pomeni, da hočemo dodati števko k meri na zaslonu. Izvede se metoda "dodajStevko()".

```
/*case A, B, C, D najprej ponastavi številke, nato pa doda vrednosti
  za hitro nastavljanje in premakne vodilo*/
case 'A':
  ponastaviStevilke();
  dodajStevko(1);
  dodajStevko(0);
  dodajStevko(0);
  premakniStepper(celoStevilo);
  break;
case 'B':
  ponastaviStevilke();
  dodajStevko(2);
  dodajStevko(0);
  dodajStevko(0);
  premakniStepper(celoStevilo);
  break;
case 'C':
  ponastaviStevilke();
  dodajStevko(5);
  dodajStevko(0);
  dodajStevko(0);
  premakniStepper(celoStevilo);
  break;
case 'D':
  ponastaviStevilke();
  dodajStevko(1);
  dodajStevko(2);
  dodajStevko(5);
  dodajStevko(0);
  premakniStepper(celoStevilo);
  break;
```

Slika 44: Program: vrednosti tipk A, B, C, D.

Na strani tipkovnice ležijo tipke A , B, C, D. Če pritisnemo eno od njih, se vodilo pomakne na vnaprej določeno mero. Tipki A je določena mera 100 mm, B 200 mm, C 500 mm, D 1250 mm. Program najprej uporabi metodo "ponastaviStevilke()", ki nastavi vrednost vseh mest v tabeli "stevilke" na 99, nastavi število števk (spremenljivko "i") na nič in počisti zaslon.

Nato v tabelo doda vsako števko mere zase in pokliče metodo "premakniStepper()", ki vodilo premakne na določeno mero.

6.3 Metode

dodajStevko()

```
//doda v seznam številko, ki je bila podana s tipkovnico
void dodajStevko(int napisanaStevilka) {
  if (i < 4) {
    stevilke[i] = napisanaStevilka;
    if (napisanaStevilka == 0) {
      if (i != 0 ) {
        i++;
      }
    }
    else {
      if (i == 0) {
        tft.fillRect(0, 20, 200, 120, WHITE);
      }
      i++;
    }
  }
  sestavaPrikaza();
}
```

Slika 45: Program: metoda dodajStevko().

Metoda "dodajStevko()" prejme en argument, ki ima vrednost pritisnjene številke na tipkovnici. Ker je največja možna mera enaka 1250 mm, lahko vpišemo samo 4 številke. Če je spremenljivka "i" (število vpisanih številk) manjša od 4, program dodeli naslednjemu mestu v tabeli, ki je enako 99, vrednost pritisnjene številke na tipkovnici. Ker nočemo, da se na zaslonu pojavljajo vodilne ničle, program preveri logično izjavo:

$$\text{vpisana številka} \neq 0 \text{ ALI število številk} \neq 0.$$

Če ta izjava drži, vemo, da ni bila na prvem mestu vpisana ničla, zato lahko povečamo število številk za 1.

Če ta izjava ne drži, vemo, da je na prvem mestu vpisana številka 0, zato števila številk ne povečamo, saj bi ta ničla drugače zasedla eno mesto v številu in bi bilo največje možno število le trimestno, saj je na prvem mestu 0. Za tem program kliče metodo "sestavaPrikaza()".

sestavaPrikaza()

```
void sestavaPrikaza() {
  for (int j = 0; j < 4; j++) {
    if (stevilke[j] != 99) {
      steviloZaPrikaz += stevilke[j] * ceil(pow(10, i - 1 - j));
    }
  }
  //s tem zavarujema premik nad največjo mero
  if (steviloZaPrikaz > 1250) {
    tft.fillRect(0, 20, 200, 120, WHITE);
    steviloZaPrikaz = 1250;
  }
  zaslon(String(steviloZaPrikaz));
  celoStevilo = steviloZaPrikaz;
  steviloZaPrikaz = 0;
}
```

Slika 46: Program: metoda sestavaPrikaza().

Metoda "sestavaPrikaza()" iz števk v tabeli "stevilke" sestavi mero, ki se pozneje prikaže na zaslonu. To sva naredila tako, da sva vsako števko iz tabele "stevilke" pomnožila z ustrezno potenco števila 10.

Če vrednost spremenljivke "steviloZaPrikaz" preseže 1250, ji program dodeli vrednost 1250, saj je to maksimalna mera. Če pa je pod 3, pa ji dodeli vrednost 3, saj je to najmanjša mera. Program nato pokliče metodo "prikaz()". Ko se ta konča, zapiše vsako števko iz tabele "stevilke" v svojo pomnilniško besedo. Spremenljivki "celoStevilo" dodeli vrednost spremenljivke "steviloZaPrikaz", tej pa vrednost vrne na 0.

izbrisiStevko()

```
//* zbriše zadnjo dodano števko v seznam
case '*':
  izbrisiStevko();
  break;
```

*Slika 47: Program: vrednost tipke *.*

Če na tipkovnici pritisnemo tipko *, program kliče metodo "izbrisiStevko()".

```
//ta funkcija se kliče z * in izbriše zadnjo dodano številko
void izbrisiStevko() {
  if (i != 0 ) {
    stevilke[i - 1] = 99;
    i--;
    tft.fillRect(0, 20, 200, 120, white);
    sestavaPrikaza();
  }
}
```

Slika 48: Program: izbrisiStevko().

Ta metoda izbriše zadnjo vpisano številko v tabeli "stevilke" in jo prikaže na zaslonu preko metode "sestavaPrikaza()" in "prikaz()". Če je vpisana vsaj ena številka (če spremenljivka "i", število vpisanih številk, ni enaka 0), program zmanjša število številk (spremnljivko "i") za 1 in spremeni vrednost zadnjega vpisanega mesta na 99. Nato pa izvede metodo "sestavaPrikaza()".

premakniStepper()

```
//# premakne vodilo na mero, ki je vpisana
case '#':
  premakniStepper(celoStevilo);
  break;
```

Slika 49: Program: vrednost tipke #.

Če na tipkovnici pritisnemo tipko # ali katerokoli tipko s črko, se prikliče metoda premakniStepper().

```
//s funkcijo premakniStepper premaknemo motor za izračunano mero naprej oz. nazaj
void premakniStepper(int premik) {
  if (premik > 1250) {
    premik = 1250;
  }
  if (premik < 3){
    premik = 3;
  }
}
  trenutnaMera = String(premik);
  tft.fillRect(0, 159, 200, 80, WHITE);
  tft.setCursor(5, 230);
  tft.setFont(&FreeSansBold24pt7b);
  tft.println(trenutnaMera);
  premik = premik - zacetnaMera;

  long izracun = ((premik * 400L) / 4.998); //4.998020529
  stepper.runToNewPosition(izracun);
  ponastaviStevilke();
}
```

Slika 50: Program: metoda premakniStepper().

Najprej program preveri, ali je "premik" (vpisana mera) znotraj dovoljenega območja, če ni, jo priredi tako, da jo nastavi na 1250, če je večja od 1250, oziroma nastavi na 3, če je manjša od 3.

Premik izračunava tako, da od vpisane številke odštejeva mero, ki jo program prebere iz pomnilnika.

Za izračun števila potrebnih korakov uporabiva podatkovni tip long, ki je 64-bitni zapis celega števila. Izračuna se po naslednji formuli:

$$izracun = ((premik * 400) / 4{,}998)$$

Premik sva pomnožila s številom korakov, potrebnih za en obrat. Ta produkt sva delila s 4,998, kar je korak navojne palice, kot sva razložila že na strani 7.

S "stepper.runToNewPosition(izracun)" sporočimo motorju, za koliko in v katero smer naj se obrne. Ko se vodilo prestavi, pa pokliče metodo "ponastaviStevilke()".

zaslon()

```
//s funkcijo zaslon zapišema na določene koordinate vrednost y
void zaslon(String y) {
  tft.setCursor(5, 130);
  tft.setFont(&FreeSansBold24pt7b);//DAJ VEČJO
  tft.println(y);//izpiše vrednost spremenljivke y

  //vpiše številke v pomnilnik preko EEPROM.write funkcije
  for (int j = 0; j < 4; j++) {
    EEPROM.write(j, stevilke[j] );
  }
}
```

Slika 51: Program: metoda zaslon().

Metoda "zaslon()" zapiše na koordinate (5, 130) vrednost spremenljivke "y".

Zapiše pa tudi trenutno mero v pomnilnik (z zanko "for"). Za vsak "j", ki je v intervalu [0, 4), se izvede koda, ki je zapisana za pogojem, v zavitih oklepajih. Z "EEPROM.write" se zapiše v pomnilniško besedo z naslovom "j" vrednost številke, ki je na mestu "j".

ponastaviStevilke()

```
//s funkcijo ponastaviStevilke ponastavima številke za nov vnos
void ponastaviStevilke() {  // Reset numbers for next entry
  for (int j = 0; j < 4; j++) {
    stevilke[j] = 99;
  }
  i = 0;
  tft.fillRect(0, 20, 200, 120, WHITE);

}
```

Slika 52: Program: metoda ponastaviStevilke().

Metoda "ponastaviStevilke()" nastavi vrednost vseh mest v tabeli "stevilke" na 99 in nastavi število števk (spremenljivko "i") na nič.

7 TESTIRANJE

Po sestavi prvega prototipa, sva le-tega dala mizarju, da ga preizkusi. Ugotovila sva, da je zaslon Nokia 5110 premajhen in da ima motor NEMA 23 prepočasen pomik, zato sva ga nadgradila na NEMA 34, zaslon pa zamenjala z Adafruit ILI 9341.

Nov prototip sva spet dala mizarju, da ga je preizkusil in obneslo se je vse, razen kalibracije, zato sva se odločila, da jo bova začasno izpustila in jo razvila kasneje. Priredila sva še samo pospešek in najvišjo hitrost motorja, da premikanje poteka hitreje.

Mizar nama je povedal, da sva mu s tem pripomočkom olajšala in pohitrila razrezovanje. Ugotovili smo celo, da kupljeni digitalni odčitovalec, ki je že bil nameščen na vodilu in je namenjen bolj natančnemu ročnemu nastavljanju, ne kaže pravilno.

8 ZAKLJUČEK

Avtomatizacija desnega vodila na formatni žagi SCM Si 400 Nova nama je uspela. Vse hipoteze sva potrdila:

- Vodilo ima najvišjo izmerjeno napako manj kot 1 desetinko milimetra (0,1 mm), kar je v mizarstvu dopustna toleranca. Natančnost je ponovljiva, kar pomeni, da se tudi po večjem številu premikov (razrez materiala za standardno kuhinjo) napaka ne poveča.
- Z avtomatizacijo pomika vodila se res prihrani čas, saj lahko, medtem ko se vodilo nastavlja, naredimo kaj drugega, na primer poravnamo obdelovanec ali pa ga odrežemo po dolžini. Približni čas prestavitve z mere 3 mm na mero 1250 mm je 30 sekund, kar je zelo primerljivo temu, da bi šel okoli stroja, nastavil vodilo in se vrnil k žagi.
- Izvedba naju je stala 502,04 €, če vračunava stroške komponent, ki sva jih narobe kupila in jih nisva uporabila. Glede na cene avtomatiziranih pomikov TigerFence in SCM dodatka je bilo to zelo poceni.

Izdelek se že uporablja, trenutno deluje brezhibno. Predvidevava, da bo brezhibno delovanje trajalo več kot eno leto. Toliko časa imajo garancijo tudi podobni izdelki. Ker sva se veliko naučila na podlagi lastnih napak, verjameva, da bova ob morebitnih težavah hitro našla rešitev.

V prihodnosti nameravava najin izdelek še nadgraditi. Dodala bova kalibracijo, zamenjala koračni motor s servo motorjem in avtomatizirala nagib ter dvig žaginega lista, kar je uporabno tudi na drugih lesnoobdelovalnih strojih. Razmišljava pa še o optimizaciji razreza v povezavi z avtomatskim pomikom vodila.

9 VIRI

14CORE EDITOR. Using membrane keypad/keyboard on Arduino. [Online.] 14*Core.* [Citirano: 24. feb. 2017; 17:45]. Dostopno na spletnem naslovu: http://www.14core.com/using-membrane-keypadkeyboard-on-arduino/.

ADMIN. TB65603A Single Axis Stepper Motor Driver Board (HCMODU0022). [Online]. *Hobby components forum.* [Datum zadnje spremembe: 5. avg. 2012]. [Citirano: 27. jan. 2017; 21:36]. Dostopno na spletnem naslovu: http://forum.hobbycomponents.com/viewtopic.php?f=76&t=1371.

ARDUINO & GENUINO PRODUCTS: Arduino MEGA 2560 & Genuino MEGA 2560. [Online]. *Arduino.* [Citirano: 1. feb. 2017; 17:03]. Dostopno na spletnem naslovu: https://www.arduino.cc/en/Main/ArduinoBoardMega2560.

ARDUINO. [Online]. *Wikipedia, prosta enciklopedija.* [Datum zadnje spremembe: 6. jun. 2016; 12:39]. [Citirano: 6. feb. 2017; 15:27]. Dostopno na spletnem naslovu: https://sl.wikipedia.org/wiki/Arduino.

ARDUINO. [Online]. *Wikipedia, the free encyclopedia.* [Datum zadnje spremembe: 3. feb. 2017; 09:08]. [Citirano: 6. feb. 2017; 15:14]. Dostopno na spletnem naslovu: https://en.wikipedia.org/wiki/Arduino.

ARDUINO. [Online]. *Wikipedija, prosta enciklopedija.* [Datum zadnje spremembe: 17. feb. 2017; 15:32]. [Citirano: 4. mar. 2017; 19:13]. Dostopno na spletnem naslovu: https://sl.wikipedia.org/wiki/Arduino.

ARDUINO.[Online]. *Wikipedia, the free encyclopedia.* [Datum zadnje spremembe: 3. mar. 2017; 04:36]. [Citirano: 4. mar. 2017; 20:37]. Dostopno na spletnem naslovu: https://en.wikipedia.org/wiki/Arduino.

KKM: Matrix membranska tipkovnica 4 x 4. [Online]. *E-radionica.com.* [Citirano: 24. feb. 2017; 15:23]. Dostopno na spletnem naslovu: https://e-radionica.com/hr/blog/2015/08/19/kkm-matrix-membranska-tipkovnica-4x4/.

LESNOOBDELOVALNI stroji: Formatna Žaga SCM Si 400 Nova. [Online]. *Lestroj.* [Citirano: 15. jan. 2017; 16:43]. Dostopno na spletnem naslovu: http://www.lestroj.si/sl/novi-stroji/formatna-zaga-scm-si-400-ep-nova.htmlhttps://www.tigerstop.com/products/tigerfence/.

PRAVEEN. Interfacing hex keypad to arduino. [Online]. *Circuits today.* [Datum zadnje spremembe: 5. dec. 2014]. [Citirano: 31. jan. 2017; 13:17]. Dostopno na spletnem naslovu: http://www.circuitstoday.com/interfacing-hex-keypad-to-arduino.

SMAK, d. o. o. [Online]. *Arduinox.* [Citirano: 26. feb. 2017; 19:12] Dostopno na spletnem naslovu: http://arduinox.si/.

TIGERFENCE. [Online]. *Ameri-can machinery ltd.* [Citirano: 22. feb. 2017; 12:28]. Dostopno na spletnem naslovu: http://www.windowmachinery.com/products-positioner-tigerfence.html.

What is an Arduino? [Online]. *Sparkfun.* [Citirano: 4. mar. 2017; 18:12]. Dostopno na spletnem naslovu: https://learn.sparkfun.com/tutorials/what-is-an-arduino.

WHAT is Arduino? [Online]. *Arduino.* [Citirano: 3. mar. 2017; 21:47]. Dostopno na spletnem naslovu: https://www.arduino.cc/en/Guide/Introduction.

WIRE DIV268N to Steppers. [Online]. *Hobby CNC Australia.* [Citirano: 10. feb. 2017; 16:25]. Dostopno na spletnem naslovu: http://www.hobbycncaustralia.com.au/Instructions/iI72DIV268NToStepper.htm.

WIRING. [Online]. *Adafruit*. [Datum zadnje spremembe: 4. maj 2015]. [Citirano: 7. jan. 2017; 17:43]. Dostopno na spletnem naslovu: https://learn.adafruit.com/user-space-spi-tft-python-library-ili9341-2-8/wiring.

ELECTRONIC Miter box! Control a Stepper motor with an Arduino and Keypad. [Online]. *Brainy bits.* [Citirano: 17. nov. 2016; 18:37]. Dostopno na spletnem naslovu: https://brainy-bits.com/blogs/tutorials/diy-stepper-miter-box.

SOFTWARE. [Online]. *Arduino.* [Citirano: 17. nov. 2016; 19:21]. Dostopno na spletnem naslovu: https://www.arduino.cc/en/main/software.