Izdelava univerzalnega robotskega 4 osnega manipulatorja

Jan Slemenšek

[1]*FERI, Smetanova ulica 17, 2000 Maribor*
E-pošta: jan.slemensek@gmail.com

The making of 4 axis robotic manipulator

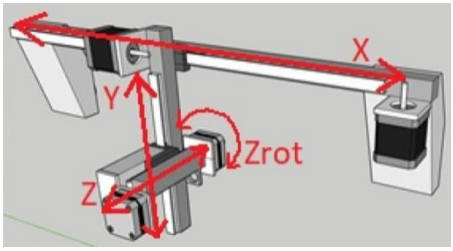
Abstract. *In the last 30 years, robots with it's revolution have shaped the world as we know it today. With every day, robots are getting smarter, faster and more accurate.*

1 Uvod

Roboti so v zadnjih 30-ih letih s svojim razvojem revolucionirali ter oblikovali svet kot ga poznamo danes. Z vsakim dnem postajajo roboti pametnejši, hitrejši ter bolj natančni.

2 Moje delo

Roboti so me od nekdaj fascinirali s svojimi zmogljivostmi.
Že nekaj časa sem si želel izdelati nekakšen robotski manipulator, ob izdelavi katerega bi se veliko naučil, hkrati pa bi ga lahko programiral, da bi manipulator izvajal neko ponovljivo nalogo.
Zamislil sem si 4 osni manipulator, kjer bi imel 3 translacijske osi *(X,Y,Z)* in eno rotacijsko os *(Zrot).*



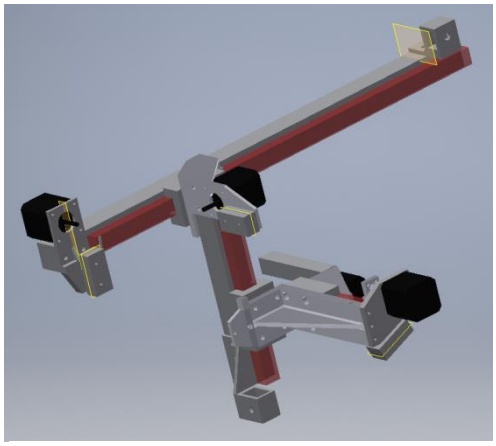
Slika 1: 3D skica manipulatorja

Na os četrtega motorja *(Zrot)* bi lahko nameščal različne "glave" manipulatorja. Zamislil sem si več različnih glav, ki bi omogočile manipulatorju opravljanje različnih nalog. Tako bi lahko izdelal nekakšno prijemalo, s katerim bi prijemal različne izdelke ter manipuliral z njimi.
Lahko bi izdelal glavo za 3D tisk, ter manipulator uporabil za 3D tiskanje, na glavo bi lahko namestil avtomatski spajkalnik ali pa mikroskop.

Najprej sem narisal skico, da sem si lahko lažje predstavljal celoten koncept ter kaj vse bo potrebno izdelati. Dobil sem tudi grob občutek glede potrebnih dimenzij linearnih vodil, jermenov in drugih komponent.

2.1 Izdelava

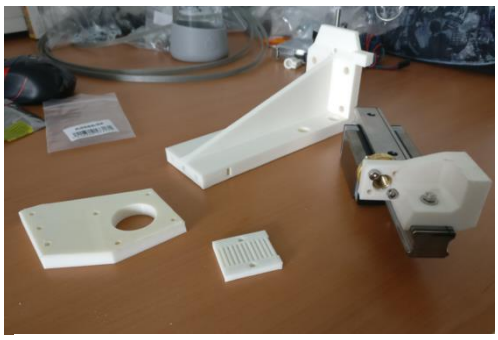
Narisal sem natančen 3D model *(assembly)* mehanizma, ter začel zbirati material.
Odločil sem se, da bom vse potrebne nosilce za mehanizem natisnil na navadnem *FDM* 3D tiskalniku.



Slika 2: Natančen 3D model mehanizma

Obdelovalni volumen bi pri taki geometriji znašal: 450 x 200 x 100 mm *(x,y,z).*
X in Y os uporabljata jermen in pripadajoče jermenice medtem ko Z os spremeni rotacijsko gibanje osi motorja v linearno gibanje drsnika s pomočjo vretena in pripadajoče matice. Korak navoja vretena je 1mm.

Nosilce sem natisnil iz bele *ABS* plastike.



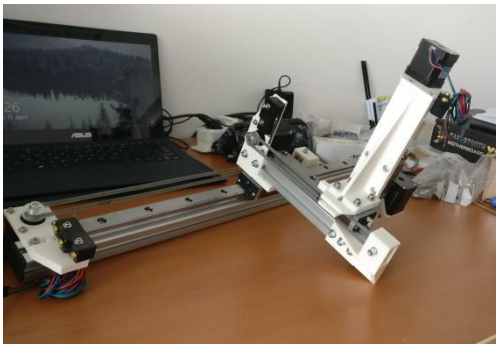
Slika 3: nekaj 3D natisnjenih nosilcev

2.2 Sestavljanje

Ko sem prejel ves potreben material, sem lahko pričel s sestavljanjem manipulatorja.
Zaradi potreb po dobri natančnosti in ponovljivosti, sem se odločil za uporabo linearnih tirnih vodil.

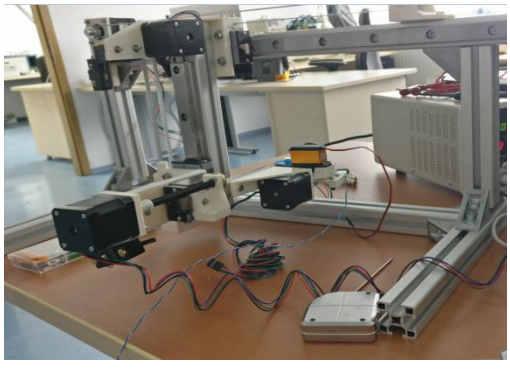
Slika 4: Sestavljena Z os, testiranje.

Uporabil sem koračne motorje *Nema 17*. Ti motorji nimajo povratne informacije o svoji poziciji *(Open-loop)*. Potrebno je uporabiti končno stikalo. Tako lahko po ponovnem zagonu krmilnika, z uporabo končnega stikala najdemo izhodiščno točko. Koračnemu motorju lahko nato posredujemo število korakov, ki naj jih motor opravi v izbrano smer.



Slika 5: Sestavljena konstrukcija

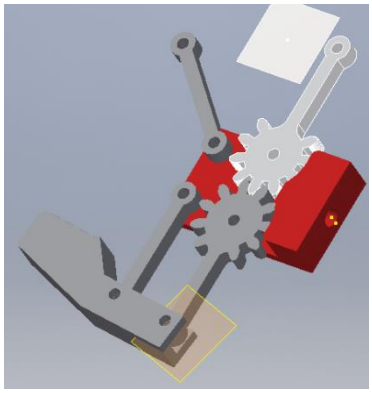
Nosilec za mehanizem je izdelan tako, da je 4. rotacijska os manipulatorja orientirana horizontalno.



Slika 6: Končan manipulator montiran na nosilec

2.3 Prijemalo

Najprej sem izdelal prijemalo za manipulacijo z objekti. Prijemalo je moralo biti zmožno objekt prijeti in spustiti.



Slika 7: 3D model prijemala

Slika 1: Končano prijemalo

Za odpiranje/zapiranje prijemala sem uporabil 6 V servo motor.

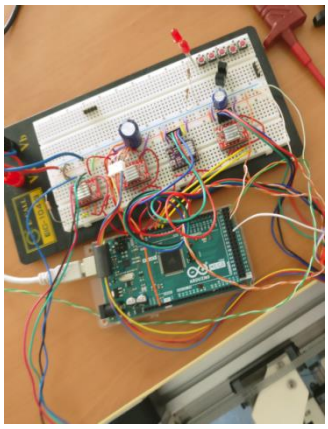
Želel sem, da bi prijemalo lahko opravljalo čim več različnih funkcij.

V "prst" prijemala sem namestil še fotoupor, ki omogoča zaznavanje osvetljenosti. S pomočjo fotoupora bi lahko preverjali ali je prijemalo v pravilni poziciji.

Slika 2: Fotoupor v prstu

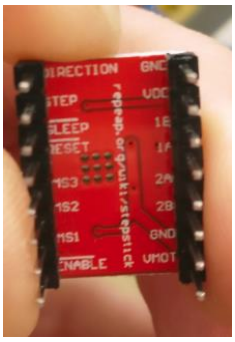
2.4 Električna vezava

Uporabil sem krmilnik *Arduino Mega 2560*, ki upravlja koračne motorje ter servo motor na prijemalu. Uporabljena so tudi končna stikala, ki zagotovijo določitev ničelne točke.
Za krmiljenje koračnih motorjev sem uporabil dodatne gonilnike *A4988*.



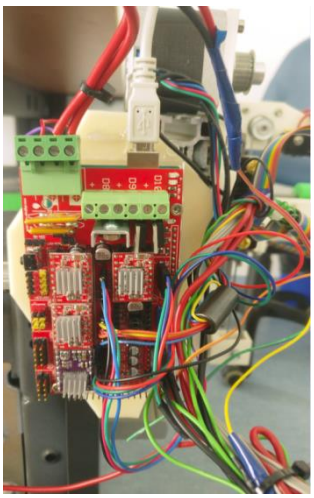
Slika 30: Vezava

Gonilnik se enostavno poveže na arduino preko treh povezav (*DIR, STEP, GND*).
V programu, ki je naložen na krmilnik, lahko enostavno definiramo, kako naj se naš koračni motor giblje. *DIR* definira smer sukanja robota (0V-levo, 5V-desno), medtem ko *STEP* (*PWM signal*) določa, koliko korakov naj motor izvede v prej definirani smeri.



Slika 41: A4988

2.4.1 Ramps 1.4



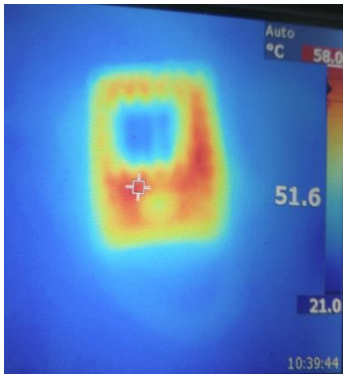
Slika 52: Ramps 1.4

Z osnovno vezavo (*Mega+A4988*) sem potrdil delovanje sistema. Hotel sem še dodatno izboljšati zanesljivost in kompaktnost elektronike z *Ramps 1.4* ploščo.

Ramps 1.4 je razširitvena plošča za *Arduino Mega*, namenjena gonilnikom koračnih motorjev. Najpogosteje se uporablja za krmilje 3D tiskalnikov, priklopi se direktno na *Arduino ploščo*.
Podpira priklop do pet koračnih motorjev, hkrati pa ima veliko število različnih vhodov (analog, digital) ter izhodov (3 x 12V tranzistor). Gonilniki *A4988* se priklopijo na *Ramps 1.4* (*slika 12)*.

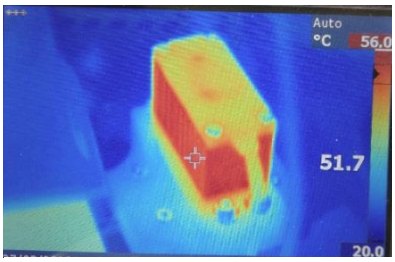
2.5 Termična stabilnost mehanizma

Ob daljšem delovanju sistema, se je gonilnik *A4988* Y osi (nosi največjo težo), pregrel do termične zaščite, ki je vgrajena v čipu, tako, da je prenehal delovati (dokler je zaščita vklopljena).
Problem je nastal zaradi prevelikih koračnih motorjev, ki so imeli nizko notranjo upornost, posledično pa so porabili preveč amperov iz gonilnika. Zamenjal sem Y koračni motor za enak, vendar manjši motor. Zamenjava motorjev je zmanjšala temperaturo gonilnika iz 110 °C na 60 °C.



Slika 63: Termična slika gonilnika

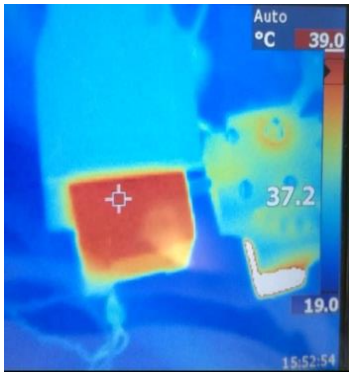
Servo motor na prijemalu se je ob navadnem delovanju na 6 V segrel na 76 °C. To temperaturo lahko zmanjšamo, da nekoliko znižamo napajalno napetost do točke, ko servo motor še lahko normalno opravlja nalogo.



Slika 14: Termična slika servo motorja

Po zmanjšanju napajalne napetosti iz 6 V na 4,8 V, se je servo motorju temperatura zmanjšala za 25 ºC.

Nema 17 koračni motorji se segrejejo do 40 °C, medtem ko se Y motor zaradi drugačne izvedbe (manjši motor, enak gonilnik) segreje do 65 °C pri polni obremenitvi.



Slika 157: Termična slika Nema17 motorja
17 motorja

2.6 Program

Program je napisan v programskem jeziku C. Za nadzor koračnih motorjev se uporablja knjižnica *"AccelStepper"*, ki omogoča enostavno definicijo maksimalne hitrosti, pospeškov ter pojemkov. S to knjižnico lahko istočasno vodimo več motorjev.

Napisal sem program, kjer lahko vpišemo koordinate (v milimetrih), robotski mehanizem pa se pomakne v želene koordinate.

Void loop zanka programa je sledeča:

```
void loop() {
  while(Serial.available() > 0) // Don't read unless avaliable
  {
     TravelX=(Serial.parseInt()*64); //multiplied for input in mm.
     Serial.println(TravelX/64);
     TravelY=(Serial.parseInt()*64); //multiplied for input in mm.
Serial.println(TravelY/64);
     TravelZ=(Serial.parseInt()*156);
     Serial.println(TravelZ/156);
     TravelZr=(Serial.parseInt()); //3200=360deg, 1600=180deg//
     Serial.println(TravelZr);
//////////////////////////////LIMITS////////////////////////////
        if (TravelX>20000){TravelX=20000;}  //X stepper limits
           else if(TravelX<0) {TravelX=0;}
        if (TravelY>7000){TravelY=7000;}    //Y stepper limits
           else if(TravelY<0) {TravelY=0;}
        if (TravelZ>153615){TravelZ=153615;}  //Z stepper limits
          else if(TravelZ<0) {TravelZ=0;}
        if (TravelZr>153615){TravelZr=153615;}  //Zr stepper limits
           else if(TravelZr<0) {TravelZr=0;}
//////////////////////////////////////////////////////////////////
   stepperX.moveTo(TravelX);  // Set new move position for X Stepper
   stepperY.moveTo(TravelY);  // Set new move position for Y Stepper
   stepperZ.moveTo(TravelZ);  // Set new move position for Z Stepper
   stepperZr.moveTo(TravelZr);//Set new move position for Zr Stepper
}
  if ((stepperX.distanceToGo() != 0) || (stepperY.distanceToGo() !=0)
|| (stepperZ.distanceToGo() !=0) || (stepperZr.distanceToGo() !=0)) {
        stepperX.run();  // Move Stepper X into position
        stepperY.run();  // Move Stepper Y into position
        stepperZ.run();  // Move Stepper Z into position
        stepperZr.run();  // Move Stepper Zr into position
       }
}
```

Program omogoča vodenje mehanizma preko vpisanih koordinat (X,Y,Z,Zr). Koordinate vpišemo v serijski vmestnik v okolju Arduino. Istočasno lahko premikamo vse štiri osi. Uporabljen je absolutni koordinatni sistem. Koordinate vpisujemo po naslednjem zgledu:

```
0,0,0,0                 //x=0mm; y=0mm; z=0mm; zr=0º;
100,56,75,1800          // x=100mm; y=56; z=75; zr=90º;
100,70,75,0             //premakneta se samo Y os ter Zr.
```

3 Zaključek

Manipulator je dokončan. Sledi še izdelava različnih "glav", ki bi mehanizmu omogočale opravljanje različnih funkcij.

Razlaga

FDM: (Fused deposition modelling), navadni 3D tiskalniki, ki talijo plastično žico.
Assembly: skupek 3D modeliranih delov zloženih skupaj, da formirajo skupno geometrijo,
ABS: tip plastike, ki se uporablja pri FDM 3D tiskalnikih,
Open-loop: Odprto-zančni način krmiljenja.

Literatura

[1] https://reprap.org/ (citirano 1.7.2018)

[2] https://www.arduino.cc/en/Main/ArduinoBoardMegaADK (citirano 1.7.2018)

[3] https://www.pololu.com/product/1182 (citirano 1.7.2018)