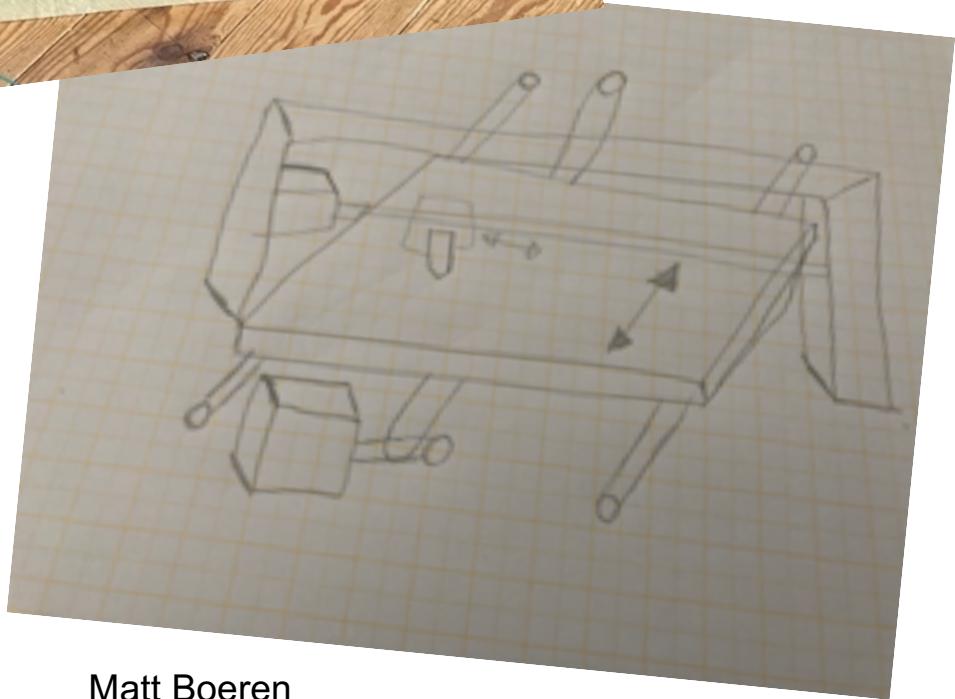
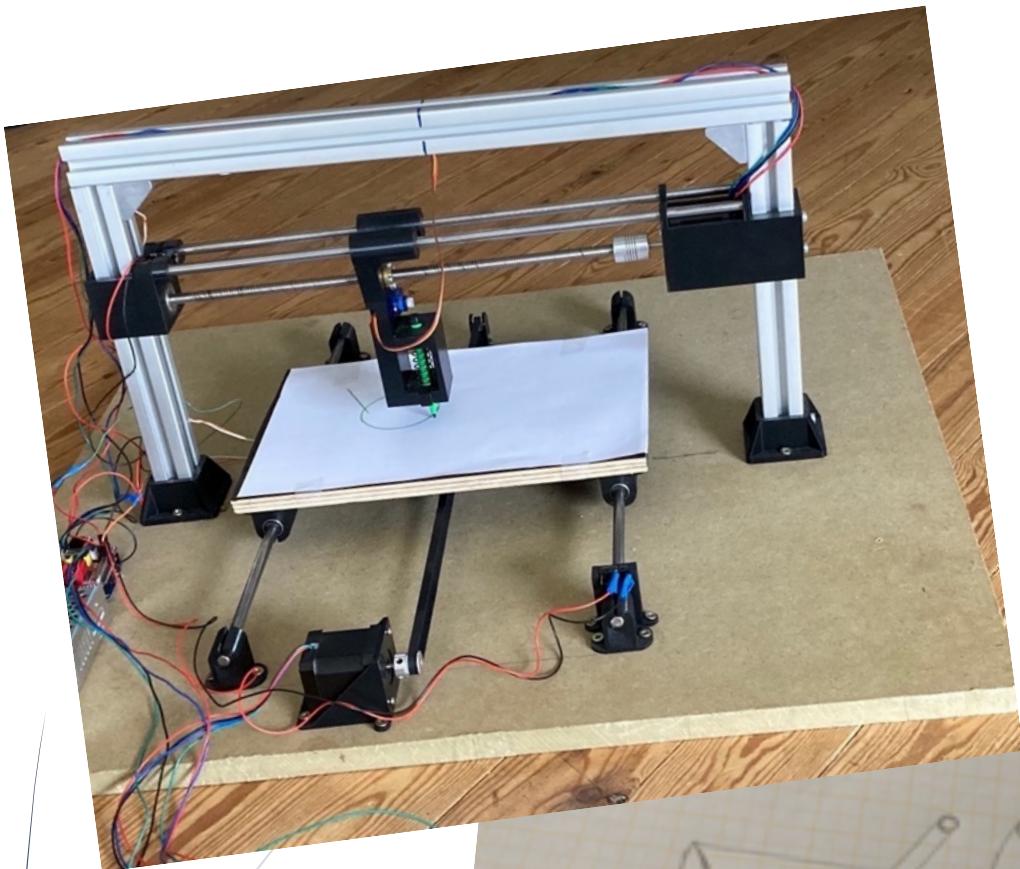




2021-2022

# GIP : CNC penplotter



Matt Boeren  
KLAS : 6IW

# GIP : CNC penplotter

Matt Boeren

Industriële wetenschappen  
Mevrouw M. Vanhoof  
Tweede leerjaar - derde graad  
Schooljaar 2021-2022



## **Samenvatting**

Mijn geïntegreerde proef gaat over een CNC penplotter. Een CNC penplotter is een computergestuurde 2D machine die kan tekenen met een pen. Die kan bewegen volgens twee assen. Deze heb ik zelf volledig opgebouwd van nul. Dit heb ik gedaan door middel van zelfontworpen 3D geprinte stukken en materialen dat ik gekocht heb. Ik ben tot een geslaagd product gekomen dat volledig werkt.

Nadat de bouw gedaan was ben ik begonnen met het programmeren van mijn penplotter. In het begin heb ik hiervoor gebruik gemaakt van bestaande software maar ik heb ook mijn eigen software geschreven voor basisfiguren, zoals lijnen, rechthoeken, cirkels, cirkelbogen en ellipsen. Mijn penplotter wordt aangestuurd door middel van een Arduino. Dit is een microcontroller die de motoren kan doen bewegen. Voor mijn eigen programmatie werkt deze als vertaler tussen mijn computer en mijn penplotter. In de computer wordt er een commando ingegeven en de Arduino vertaalt deze door middel van mijn programma naar bewegingen die de motoren moeten maken om de pen de figuur te laten tekenen.

## Voorwoord

Ik heb mijn onderwerp gekozen omdat ik aan de jury wil tonen dat ik meer kan dan gewoon informatie opzoeken en deze verwerken. Ik kan de informatie zelf genereren in de vorm van een praktische uitwerking. Ik heb ervoor gekozen om een CNC penplotter te bouwen en te programmeren. Dit heeft de uitdagingen concept ontwikkeling, mechanisch en electrisch ontwerp, 3D ontwerpen en programmeren.

Het bouwen van mijn CNC penplotter ging vlot maar ik heb wel enkele stukken terug moeten ontwerpen. Desondanks ben ik snel aan het resultaat dat ik wou gekomen. Bij het programmeren van mijn penplotter had ik al snel een basisprogramma maar dit debuggen heeft wel enkele weken geduurd.

Het schrijven van mijn geïntegreerde proef liep niet zo vlot als het bouwen zelf. Het moeilijke eraan was dat iemand zonder een technische achtergrond het moet begrijpen. Na enkele evaluaties is het mij wel gelukt om een begrijpbare GIP te schrijven.

Ik zou graag mijn vader (industrieel ingenieur) bedanken voor het lenen van zijn 3D-printer aan mij voor het 3D printen van de stukken. Ook wil ik mijn moeder (industrieel ingenieur) en Mevr. Vanhoof (leerkracht fysica en chemie) bedanken om mijn GIP na te lezen.

I choose my project because I wanted to show the jury that I can do more than just look up information and process it. I can make the information myself in the form of a practical elaboration. I chose to build and program a CNC penplotter because it has the challenge of 3D-designing and programming.

The process of building my CNC penplotter did run smoothly but I had to redesign some parts. Nevertheless I got to the result I wanted fairly quickly. After building it I started programming the machine. I had a rough program quickly but the debugging took a couple of weeks.

The writing of my integrated project didn't go as smoothly as the build itself.

The hard part of the writing was that someone without a technical background had to understand it. But after a couple of evaluations it became understandable.

I would like to thank my father (industrial engineer) for lending his 3D-printer to me for the 3D printed parts. Also I would like to thank my mom (industrial engineer) and Ms. Vanhoof (teacher physics and chemistry) for evaluating my integrated project.



## Inhoudsopgave

Samenvatting .....	3
Voorwoord .....	4
1 Principe en voordelen van CNC machines .....	8
2 Voorbeelden van CNC machines.....	8
2.1 CNC frees .....	8
2.2 3D printer .....	9
3 Eigen CNC penplotter.....	9
3.1 Algemene informatie .....	9
3.2 Gedachtengang van het ontwerp .....	10
3.3 Bestellijsten .....	11
4 Mechanisch ontwerp van de y-as .....	12
4.1 3D geprinte onderdelen van de y-as .....	13
4.1.1 Lineaire kogellager klem voor het schrijfbed.....	13
4.1.2 Klem voor de geleidingsassen .....	13
4.1.3 Timing belt klem .....	14
4.1.4 Motorbevestigingsstuk y-as.....	14
4.1.5 Timing belt pulley houder .....	15
4.1.6 Eindeloopschakelaar houder y-as.....	15
4.2 Montage van de y-as.....	16
4.2.1 Schrijfbed .....	16
4.2.2 Montage aan de plaat .....	17
5 Mechanisch ontwerp van de x-as .....	18
5.1 3D geprinte onderdelen van de x-as .....	19
5.1.1 Motorbevestigingsstuk x-as.....	19
5.1.2 Kogellager klem .....	19
5.1.3 Penhouder.....	20
5.1.4 Montagerail voet.....	22
5.1.5 Eindeloopschakelaar houder x-as.....	22
5.2 Montage van de x-as.....	23
6 Elektrisch ontwerp.....	24
6.1 Aansluiten van de componenten.....	24
6.2 Aansluitschema.....	25
6.3 Motorcontrollers instellen en microstepping.....	26
7 CNC principes van mijn CNC penplotter.....	27
7.1 Referentiepunt bepalen van de machine .....	27
7.2 Stappen per millimeter .....	27
8 CNC penplotter aansturen met bestaande software.....	28
8.1 Instellingen .....	28
8.2 Commando's .....	29

9	Eigen software voor basis vormen.....	30
9.1	Algemene code .....	30
9.2	Homing sequence .....	33
9.3	Naar een bepaalde plaats bewegen .....	35
9.4	Lijn tekenen.....	38
9.5	Rechthoek tekenen .....	39
9.6	Cirkel en cirkel bogen tekenen.....	40
9.6.1	Cirkel .....	40
9.6.2	Cirkel bogen .....	41
9.7	Ellips tekenen.....	42
9.8	User interface.....	43
10	Besluit .....	45

## 1 Principe en voordelen van CNC machines

CNC staat voor Computer Numerical Control. Door met behulp van een computerprogramma een 2D of 3D ontwerp te maken, kan het programma berekenen hoe een bepaalde tool van het betreffende CNC machine moet bewegen om het werkstuk te maken. Deze ontwerpen worden gemaakt in CAD-software. CAD staat voor Computer Aided Design. Het maken van stukken met behulp van computers noemt CAM of Computer Aided Manufacturing. Hoe de CNC machine dit stuk maakt hangt af van machine tot machine.

De berekende bewegingen worden opgeslagen in een gcode. Een gcode is een lang bestand vol instructies die de computer vormt voor de machine. De machine zal deze stap per stap uitwerken. Wanneer deze stappen uitgevoerd zijn zal het stuk gemaakt zijn.

CNC machines worden heel veel gebruikt omdat deze veel accurater en sneller zijn dan een manuele bediening. Door deze machines in te zetten is er niemand meer nodig om de machine te bedienen en is het gemakkelijk om aan massaproductie te doen.

In sommige bedrijven waar ze werken met CNC machines is het zelfs niet meer nodig om het werkstuk te vervangen omdat dit volkomen automatisch gebeurt door een robotarm.

## 2 Voorbeelden van CNC machines

Om aan te tonen waarvoor CNC machines in het dagelijks leven gebruikt worden en hoe deze werken geef ik enkele voorbeelden. Ik heb gekozen voor een CNC frees en een 3D printer.

### 2.1 CNC frees

Een CNC frees werkt door een frees aan te sturen die materiaal wegneemt bijvoorbeeld uit hout of metaal. Dit is een verspanende techniek. Van deze techniek bestaan er uitvoeringen met verschillende assen<sup>1</sup>. Zo kan men 3, 4 of 5 assig frezen.

Bij 3-assig frezen kan het werk具ug bewegen volgens twee assen (x en y) en de derde as bepaalt de diepte (z). Dit is de meest simpele versie van een CNC frees. Bij 4-assige machines kan het werkstuk ook nog kantelen volgens één as, bij 5-assig frezen volgens twee assen. In het algemeen geldt dat met meerassige freesmachines complexere stukken kunnen gemaakt worden.



Afbeelding 1 : Voorbeeld 3 assen CNC frees



Afbeelding 2 : Voorbeeld 4 assen CNC frees



Afbeelding 3 : Voorbeeld 5 assen CNC frees

<sup>1</sup> <https://www.youtube.com/watch?v=OKFFRZevXIM&t=136s>

## 2.2 3D printer

Een 3D printer is ook een CNC machine. Deze neemt geen materiaal weg maar maakt nieuw materiaal aan in de vorm van gesmolten kunststof die weer stolt. Dit is een additieve techniek.

Als men een stuk wil 3D printen maakt een computerprogramma verschillende lagen van het 3D ontwerp. Dit soort programma's wordt ook wel een slicer genoemd.

Deze maakt dan een gcode bestand voor de 3D printer. De printer print dan laag per laag het stuk. De z-as van de machine zal altijd een vaste hoeveelheid naar boven bewegen per laag. Dit noemt men de layerheight van het geprinte stuk.

Hoe groter deze is hoe ruwer het stuk zal zijn maar hoe sneller de printer het stuk zal printen. Deze technologie wordt tegenwoordig vaak gebruikt omdat deze goedkoop is en zeer toegankelijk.

Voorafgaand voorbeeld is een klassieke 3D printer. Andere 3D printers werken bijvoorbeeld met een kunststof poeder dat gesmolten wordt door een laser. Om een nieuwe laag te leggen zal er boven op het gelazerde stuk een nieuwe laag poeder gelegd worden. Dit noemt men een poederbed 3D printer.



Afbeelding 4 : Foto van een 3D printer

## 3 Eigen CNC penplotter

### 3.1 Algemene informatie

Een CNC penplotter is een computergestuurde 2D machine die kan tekenen met een pen. Die kan bewegen volgens twee assen. Hierdoor kan de pen bewegen over heel de oppervlakte die de assen vormen. Bij mijn eigen CNC penplotter is de grootte van het vlak gelijk aan een A4 papier. De x-as kan bewegen over de lange zijde. De y-as kan bewegen over de korte zijde. Bij gevolg is het bereik van mijn penplotter een A4 papier.

Mijn machine wordt aangestuurd door een Arduino die twee stappenumotoren (x- en y-as) aanstuurt. Een Arduino is een opensource programmeerbare microcontroller. De stappenumotoren doen de assen bewegen. Het type Arduino dat ik gebruik is de Arduino UNO.

Ik maak ook gebruik van een CNC shield. Hiermee kan ik de motoren en motorcontrollers gemakkelijker aansluiten op de Arduino. Voor de motorcontrollers gebruik ik DRV8825 motorcontrollers.

Om de pen van mijn penplotter naar boven en naar beneden te bewegen om al dan niet te schrijven, wordt de pen bediend door een servomotor. Dit is eigenlijk een z-as maar deze dient enkel om de pen omhoog te brengen om ervoor te zorgen dat bij bewegingen waar de pen niet moet schrijven dit ook niet gebeurt.

Mijn CNC penplotter is gemaakt van gekochte onderdelen en zelfontworpen 3D geprinte stukken. Deze 3D geprinte stukken heb ik ontworpen in het tekenprogramma Autodesk Fusion 360. De stukken zijn geprint met behulp van een Ultimaker S5. Alle stukken zijn geprint met PLA (Poly Lactic Acid - Poly Melkzuur).

De programmatie van mijn penplotter is geschreven in de Arduino IDE. Dit is een programmeer terminal van Arduino om hun microcontrollers te programmeren.

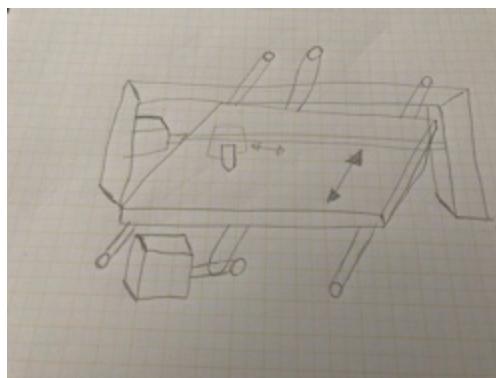
### 3.2 Gedachtengang van het ontwerp

Voor het ontwerp van mijn CNC penplotter heb ik mij gebaseerd op de 3D printers van Creality. Deze hebben een bewegend bed voor hun y-as. Dit bed kan dan naar voor en naar achter bewegen. Voor hun x-as hebben ze een vaste brug waarover de nozzle van de 3D printer kan bewegen. Bij deze 3D printers bewegen de x- en y-as door een stappenmotor in combinatie met een timing belt. Een timing belt is een getande riem die zeer exact zonder slip kan bewegen.

Ikzelf heb ervoor gekozen om mijn y-as volgens het timingbelt principe te doen bewegen en mijn x-as door een leadscrew met een leadscrew moer. Een leadscrew is een speciale draadstang die gemaakt wordt om een nauwkeurige spoed te hebben. De spoed van een draadstang is hoeveel een moer zich verplaatst als je de draadstang 1 omwenteling ( $360^\circ$ ) draait. Ik heb gekozen om gebruik te maken van een leadscrew zodat ik een ander principe van een as aandrijving van een CNC machine kan bespreken en toepassen.

Voor het bereik van mijn penplotter heb ik gekozen voor de grootte van een A4 papier. Ik heb geopteerd om van de lange zijde van het blad de x-as te maken en de korte zijde van het blad de y-as. Dit heb ik gedaan omdat zo de lengte van de geleidingsassen van de y-as korter zijn en het geheel dan compacter is. Dit is omdat de geleidingsassen een minimale lengte moeten hebben van twee keer de lengte waar je op wil tekenen in de y-richting.

Aan de hand van voorafgaande beslissingen en mijn eerste schets heb ik kunnen bepalen wat ik moest bestellen (zie 3.3) en wat ik moest ontwerpen om te 3D printen (zie 4.1 en 5.1).



Afbeelding 5 : Mijn eerste schets



Afbeelding 6 : Voorbeeld van een Creality 3D printer (Creality Ender 3)

### 3.3 Bestellijsten

Voor mijn CNC penplotter heb ik volgende materialen besteld (tabel 1). In tabel twee heb ik aangegeven welke materialen ik reeds ter beschikking had.

Webshop Tinytronics		Webshop 123-3D		
Producten	Aantal	Winkelwagentje		Product
		Aantal	Artikelnr	
 S3003 Servo - Model: S3003	1	2	DME00021	Staaf as glad 8 mm x 100 cm voor X- of Y-as
 SG90 Mini Servo - Model: SG90	1	1	DLS00002	Leadscrew moer, TR8x8
 2-Pin DuPont Jumper Draad Female-Female - 70cm - Model: DUPO2PINCABLE	2	Winkelwagentje		
 DRV8825 Motor Driver Module - Model: DRV8825DRIVERMOD	2	1	DDK00052	Jumper kabels met dupond connector mannelijk naar vrouwelijk 10 cm (40 stuks)
 GT2 Pulley - 20 tanden - 5mm as - Model: GT2PULLEY20T5MM	1	2	DME00003	Kogellager 608ZZ (1 stuk)
 Flexibele Motorkoppeling - 5mm naar 8mm - Model: FLEXCOUPLERS-8MM	1	4	DME00065	Veer 35 mm, buitendiameter 14 mm
 CNC Shield V3 - Model: CNCSHIELDV3	1	1	DFC00043	Aluminium profiel 3030 extrusion lengte 1 m (123-3D huismerk)
 LM8LUU Lineaire Kogellager - Model: LM8LUU	6	2	DFC00044	Aluminium hoekverbinding 3030 inclusief bevestigingsmateriaal (123-3D huismerk)
 GT2-20 Pulley - 20 Tanden - 3mm as - Met Rollager - Model: GT2320T	1			
 GT2 Tandriem - Timing Belt - 6mm - 1m - Model: GT2TIMINGBELT6MM	1			

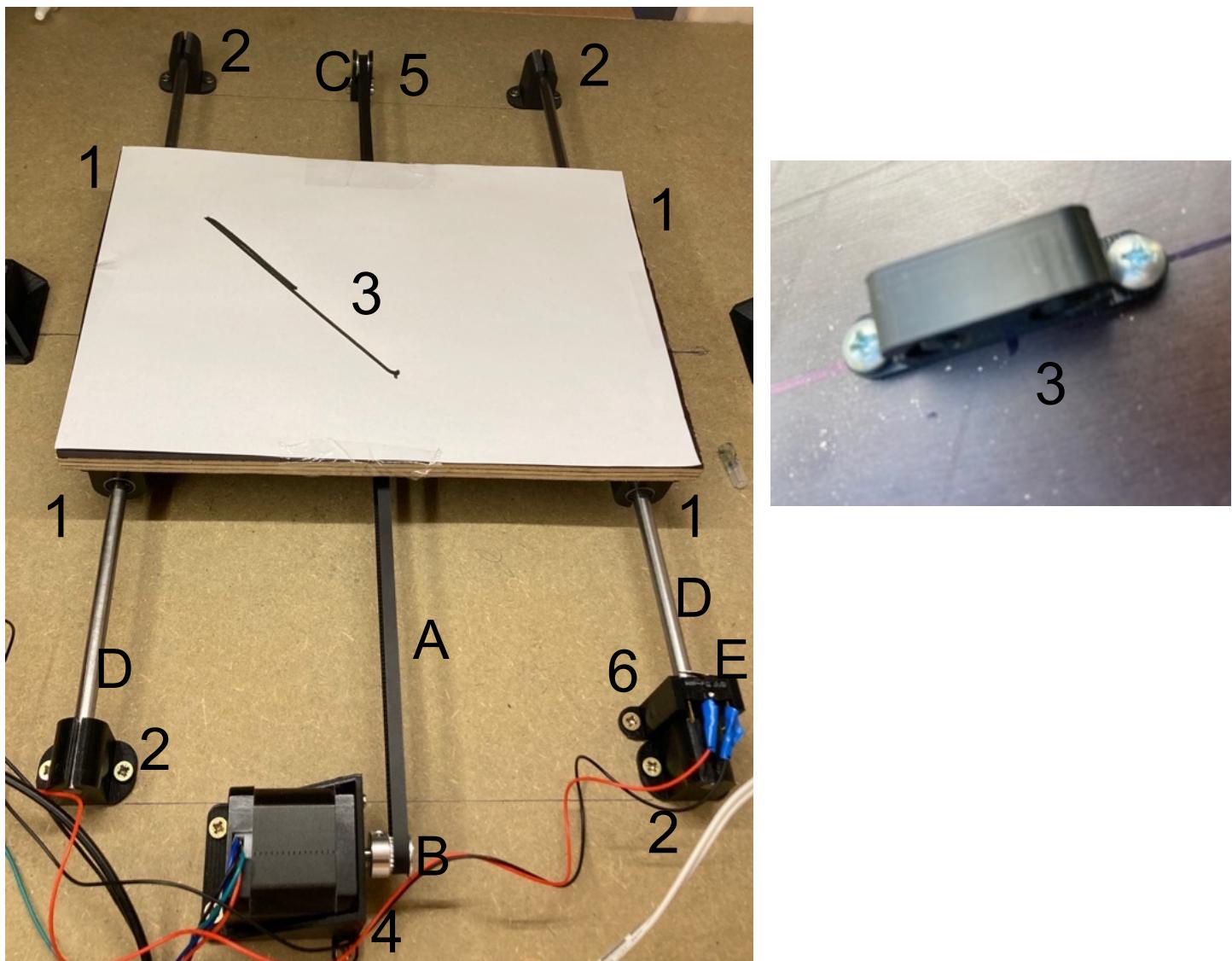
Tabel 1 : Bestellijsten

- 2 stuks NEMA 17 0,55 Nm 1,5 A stappen motoren
- Leadscrew 8x8
- 6 stuks M6x12 bouten
- 6 stuks M6 glijmoeren

Tabel 2 : Materiaal uit voorraad

#### 4 Mechanisch ontwerp van de y-as

De y-as van mijn CNC penplotter wordt bediend met een stappenmotor in combinatie met een timing belt (A). Deze timing belt zit vast met een klem (3) aan het schrijfbed. Het schrijfbed is een plaat waarop het papier bevestigd wordt. De afstand die de y-as aflegt bij één motoromwenteling is de omtrek van de schijfpulley waar de timing belt overloopt. Om het bed op gelijke hoogte te houden zijn er geleidingsassen in combinatie met lineaire kogellagers voorzien. Die assen staan vast op de montageplaat en daar kunnen de kogellagers die aan het schrijfbed bevestigd zijn over schuiven.



Afbeelding 7 : Foto's y-as genummerde componenten

- A Timing belt
- B Motor met motor pulley
- C Timing belt pulley
- D Geleidingsassen
- E Eindeloopschakelaar

- 1 Lineaire kogellager klem voor het schrijfbed
- 2 Klem voor geleidingsassen
- 3 Timing belt klem (onderaan het schrijfbed)
- 4 Motorbevestigingsstuk
- 5 Timing belt pulleyhouder
- 6 Eindeloopschakelaar

De houten plaat is de montageplaat.

De 3D geprinte onderdelen worden in volgorde besproken in de volgende paragrafen.

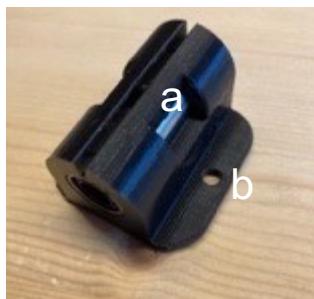
## 4.1 3D geprinte onderdelen van de y-as

### 4.1.1 Lineaire kogellager klem voor het schrijfbed

Om de geleidingsassen van de y-as te verbinden met het schrijfbed maak ik gebruik van lineaire kogellagers en een 3D geprint stuk om deze te klemmen en te bevestigen.

Om de klem te ontwerpen heb ik twee iteraties nodig gehad. Bij de eerste iteratie is er een opening gelaten (a) voor een bout en een moer om de kogellager nog meer te klemmen maar na dit ontwerp te printen bleek echter dat de kogellager vast genoeg zat zonder bout en moer. Hierdoor heb ik dit in mijn tweede iteratie weggelaten.

Bij de eerste iteratie was er ook het probleem dat de vijzen waarmee het stuk aan het bed zou vastgemaakt worden nog door het bed zouden komen. Dit heb ik opgelost door een kokertje aan de voorziene gaten (b) te tekenen zodat de vijs minder diep in de plaat komt.



Afbeelding 8 : Iteratie 1  
lineaire kogellager  
klem



Afbeelding 9 : 3D tekening  
Iteratie 2

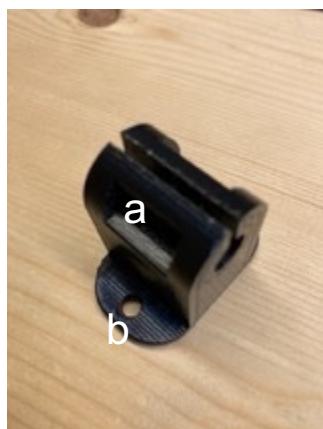


Afbeelding 10 : Iteratie  
2 lineaire kogellager  
klem

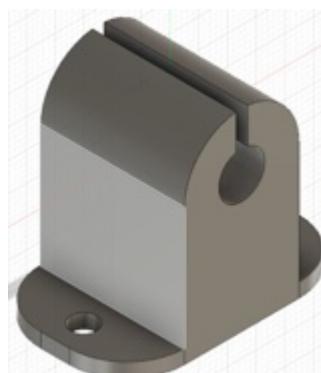
### 4.1.2 Klem voor de geleidingsassen

Om de geleidingsassen van de y-as te monteren aan de montageplaat heb ik een onderdeel ontworpen om deze te klemmen en te monteren.

Om dit stuk te ontwerpen heb ik twee iteraties nodig gehad. Net zoals bij de lineaire kogellager klem had ik bij mijn eerste iteratie een opening gelaten voor een bout en een moer (a). In mijn tweede iteratie heb ik deze weg gelaten omdat deze al genoeg geklemd zat zonder bout en moer. Zo is het stuk ook sterker omdat er meer materiaal rond de klemming zit. Aan het stuk zijn twee voetjes voorzien voor het stuk vast te maken aan de montageplaat (b).



Afbeelding 11 : Iteratie 1  
geleidingsas klem



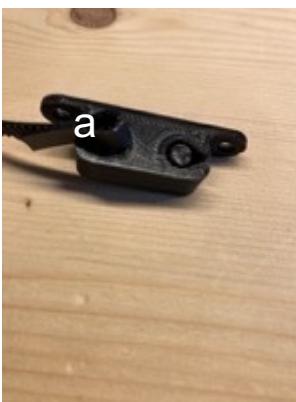
Afbeelding 12 : 3D  
tekening Iteratie 2



Afbeelding 13 : Iteratie 2  
geleidingsas klem

#### 4.1.3 Timing belt klem

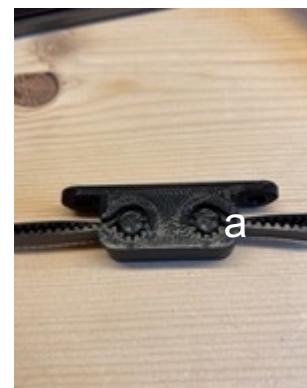
De timing belt moet bevestigd worden aan het schrijfbed zodat men door de motor te laten draaien het schrijfbed kan laten bewegen. De timing belt kan geklemd worden door deze dubbel te plooien zodat het een lusje (a) vormt en zo in de klem kan passen. Wanneer er een opening gelaten wordt met de grootte van de dubbelgevouwen timing belt en een cilindertje om het lusje tegen te houden zal deze er niet uitkunnen. Door dit principe langs beide uiteinden toe te passen kan ik van de timing belt één grote lus maken. Door de lus te spannen tussen de motor en de timing belt pulley zal als de motor draait het bed bewegen. In mijn eerste iteratie was er echter niet genoeg plaats voor de timing belt waardoor deze niet in het stuk paste. Bij mijn tweede iteratie heb ik de opening wat aangepast zodat dit wel lukte.



Afbeelding 14 : Timingbelt klem iteratie 1



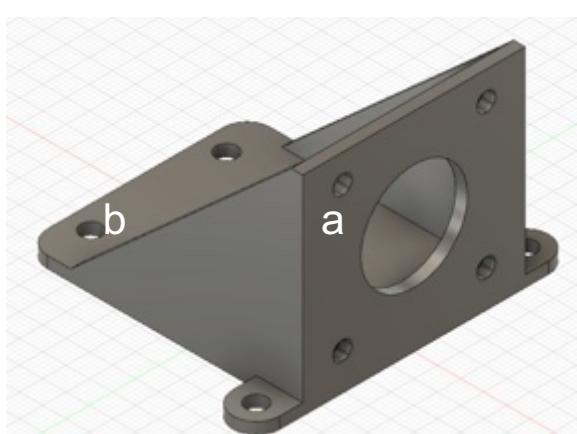
Afbeelding 15 : 3D tekening timingbelt klem iteratie 2



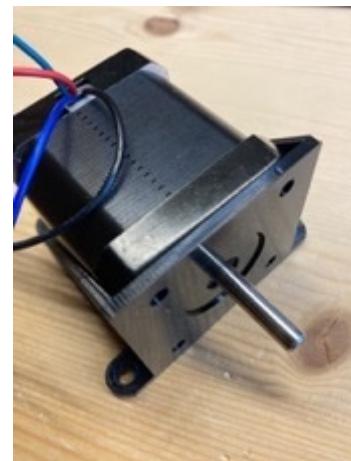
Afbeelding 16 : Timingbelt klem iteratie 2

#### 4.1.4 Motorbevestigingsstuk y-as

Om de stappenumotor te bevestigen op de montageplaat is er een stuk nodig dat de motor op zijn plaats houdt. Dit stuk is voorzien van een opening voor de as van de motor en openingen waarmee het motorblok aan het stuk bevestigd kan worden (a). Via openingen aan de onderzijde wordt het geheel vastgemaakt aan de montageplaat van de penplotter (b). Voor dit stuk heb ik maar één iteratie nodig gehad.



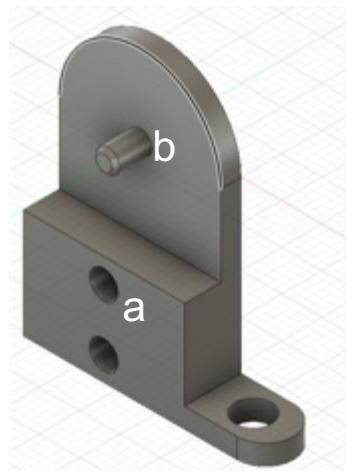
Afbeelding 17 : 3D tekening motorbevestigingsstuk y-as



Afbeelding 18 : Motorbevestigingsstuk y-as met motor

#### 4.1.5 Timing belt pulley houder

Om van de timing belt een grote lus (zie 4.1.3) te maken en deze aan te spannen heb ik een stuk nodig om deze aan de kant waar de motor niet staat op zijn plaats te houden. Hiervoor gebruik ik een aangekochte timing belt pulley en een 3D geprint stuk om deze op zijn plaats te houden. Dit stuk heb ik in twee delen geprint die identiek zijn. Deze kunnen dan verbonden worden door middel van twee bouten en moeren (a). Aan dit stuk is een asje (b) voorzien waar de kogellager van de pulley op past. Voor dit stuk heb ik één iteratie nodig gehad.



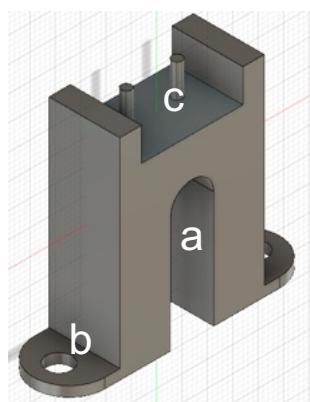
Afbeelding 19 : 3D tekening timingbelt pulley houder



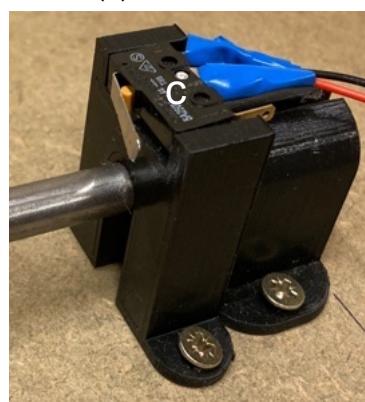
Afbeelding 20 : Timingbelt pulley houder met pulley

#### 4.1.6 Eindeloopschakelaar houder y-as

Om het nulpunt van de y-as te bepalen (zie 7.1) heb ik een houder voor een eindeloopschakelaar ontworpen. Wanneer het schrijfbed tegen de eindeloopschakelaar botst zal dit het nulpunt zijn van de y-as.  
Aan het stuk heb ik een opening gelaten om over de geleidingsas van de y-as te schuiven (a). Om het stuk vast te zetten aan de plaat zijn er twee voetjes voorzien met elk een opening voor een houtvijs (b).  
Voor de eindeloopschakelaar op zijn plaats te houden heb ik aan het stuk twee pinnen voorzien. Deze zijn getekend op de plaats waar er openingen zijn aan de schakelaar om deze te monteren (c).



Afbeelding 21 : 3D tekening eindeloopschakelaar houder y-as



Afbeelding 22 : Eindeloopschakelaar houder y-as

## 4.2 Montage van de y-as

### 4.2.1 Schrijfbed

De montage ben ik begonnen met de lineaire kogellagers in de klemmen te duwen door middel van een bankschroef. Het schrijfbed heb ik uit een restje betonplex van 12 mm gezaagd. Het bed heeft de grootte van een A4 papier.

Door middel van houtvijzen heb ik de lineaire kogellagers in hun klem verbonden met het schrijfbed. Dit heb ik gedaan met een geleidingsas in de kogellagers zodat de uitlijning perfect was (zie fig 23). Een perfecte uitlijning betekent dat de kogellagers recht achter elkaar staan. Bij gevolg zal het bed vlot kunnen bewegen over de geleidingsassen.

Hierna heb ik het midden van het bed afgemeten en aan de hand daarvan bepaald waar de timing belt klem moet komen. Het midden van de timing belt klem komt overeen met het midden van het schrijfbed. Deze heb ik net zoals de kogellagers vastgezezen met houtvijzen.



Afbeelding 23 : Montage lineaire kogellagers in hun klem aan het schrijfbed



Afbeelding 24 : Montage timing belt klem aan het schrijfbed

#### 4.2.2 Montage aan de plaat

De montageplaat is een mdf plaat van 18 mm.

Om het schrijfbed aan de plaat vast te maken heb ik eerst de geleidingsassen op maat geslepen. Deze hebben elk een lengte van 500mm. Daarna heb ik de klemmen voor de geleidingsassen langs één kant van de as er op geduwd. Ik heb de assen door de kogellagers gestoken en de andere klemmen bevestigd aan de geleidingsassen. Hierna heb ik geprobeerd of de klemmen op dezelfde plaats bleven wanneer ik het bed bewoog. Wanneer dit het geval is wil dit zeggen dat de lineaire kogellagers perfect zijn uitgelijnd.

Ik heb de plaats afgemeten waar de klemmen moesten komen op de plaat.

De klemmen heb ik aan twee kanten vastgezet en getest of het bed nog vlot kon bewegen, wat het geval was.

De motorpulley en de pulley in de houder moeten op eenzelfde lijn staan in het midden van het schrijfbed. Ik heb dit afgemeten en het motorbevestigingsstuk en de pulleyhouder vastgemaakt met houtvijzen aan de montageplaat.

Om de timing belt vast te maken aan het schrijfbed heb ik één geleidingsas terug los gemaakt zodat ik aan de onderkant van het bed kon. Ik heb de timingbelt door de pulley gestoken en langs één kant in de timing klem bevestigd door het kleine lusje te maken in de klem. Dan heb ik de grote lus gemaakt door aan de andere kant van de klem ook het kleine lusje te maken (zie 4.1.3). Dit heb ik in verschillende stappen gedaan zodat ik telkens de timing belt meer kon aanspannen tot deze genoeg gespannen was. Wanneer dit het geval was heb ik de geleidingsas terug vast gemaakt en nu beweegt het bed als je aan de motor draait.

Ten slotte heb ik de eindeloopschakelaar in zijn houder gedrukt en tegen een klem van de geleidingsassen geschroefd.



Afbeelding 25 : Foto tijdens de montage van de y-as

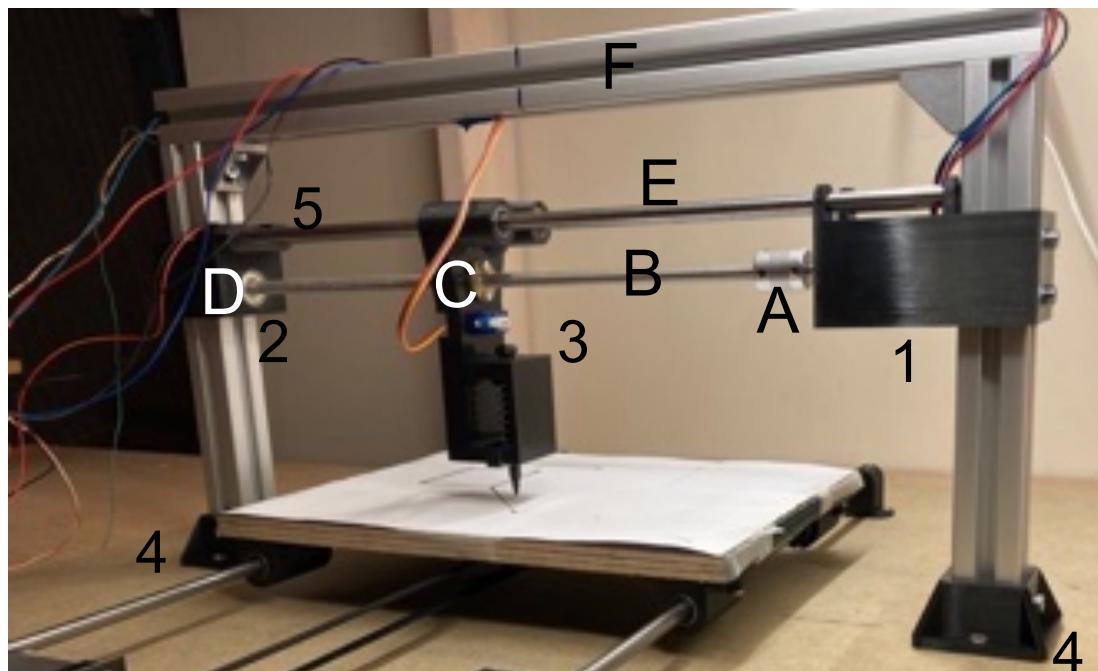


Afbeelding 26 : Volledig gemonteerde y-as

## 5 Mechanisch ontwerp van de x-as

De x-as van mijn CNC penplotter werkt met een leadscrew (B) en een leadscrew moer (C). Wanneer je aan de leadscrew draait zal de leadscrew moer naar links of rechts bewegen. De afstand die de x-as zal afleggen hangt af van de hoek die de motor draait. De leadscrew is van het type 8x8. Dit wil zeggen dat deze een diameter van acht millimeter heeft en als deze één toer draait ( $360^\circ$ ) zal de leadscrew moer acht millimeter opschuiven.

Het geheel wordt bevestigd op een brug boven het schrijfbed. De brug is gemaakt van aluminium montagerail van 30x30 mm (F).



Afbeelding 27 : Foto x-as genummerde componenten

- A Motorkoppeling
- B Leadscrew 8x8
- C Leadscrew moer
- D Kogellager
- E Geleidingsassen
- F Montagerail

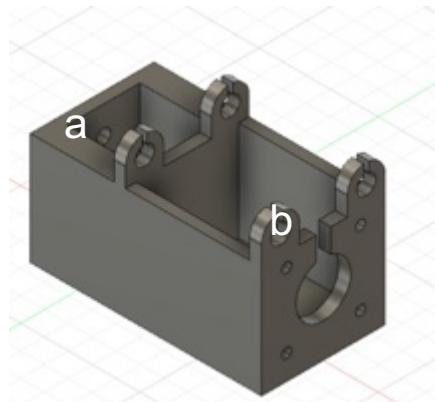
- 1 Motorbevestigingsstuk x-as
- 2 Kogellager klem
- 3 Penhouder
- 4 Montagerail voet
- 5 Eindeloopschakelaar houder x-as

De 3D geprinte onderdelen worden in volgorde besproken in de volgende paragrafen.

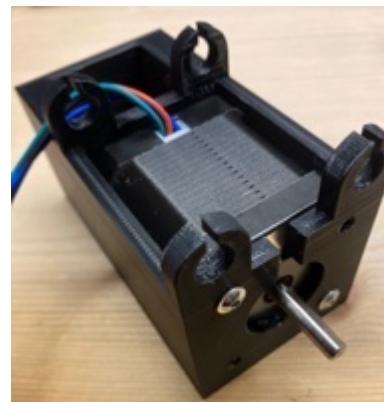
## 5.1 3D geprinte onderdelen van de x-as

### 5.1.1 Motorbevestigingsstuk x-as

Om de stappenmotor van de x-as te bevestigen aan de brug heb ik een stuk ontworpen waar de motor in vast geschroefd kan worden. Deze kan dan vastgemaakt worden aan de montage rail (F) door de daarvoor voorziene opening (a) en gaten om bouten door te steken. In de rail zitten er glijmoeren waardoor het stuk tegen de montage rail geklemd wordt. Een glijmoer is een moer dat in montagerails gebruikt wordt om een stuk te bevestigen. Aan de bovenzijde zijn er ringetjes (b) voorzien om de geleidingsassen te monteren.



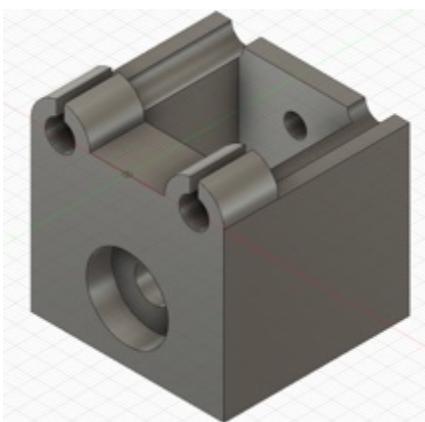
Afbeelding 28 : 3D ontwerp  
motorbevestigingsstuk x-as



Afbeelding 29 :  
Motorbevestigingsstuk x-as

### 5.1.2 Kogellager klem

Om het andere uiteinde van de leadscrew te bevestigen maak ik gebruik van een kogellager in een klem. Deze klem heeft een opening voor de montage rail en een opening waar de kogellager net in past. Er is een kleinere opening met hetzelfde middelpunt van de kogellager voor het deel van de leadscrew dat voorbij de kogellager komt. In dit stuk zijn er openingen gelaten voor bouten die schroeven in de glijmoeren in de montagerail.



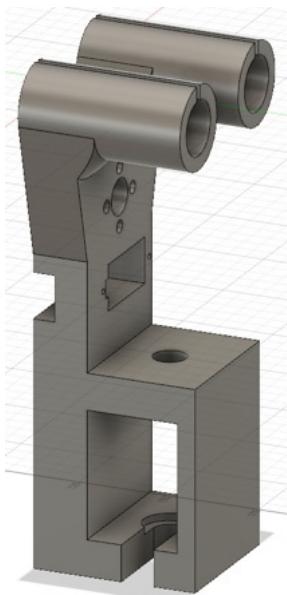
Afbeelding 30 : 3D tekening  
kogellager klem



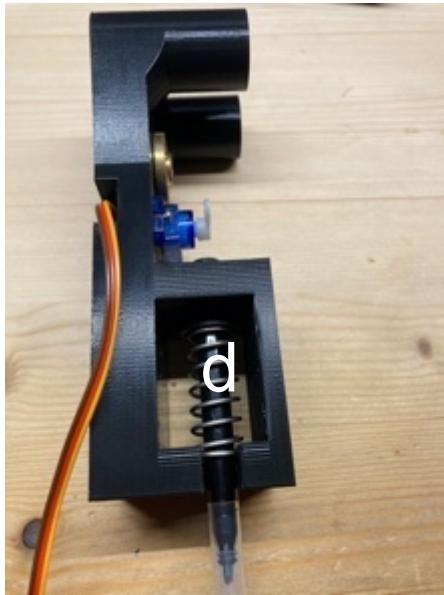
Afbeelding 31 : Kogellager klem

### 5.1.3 Penhouder

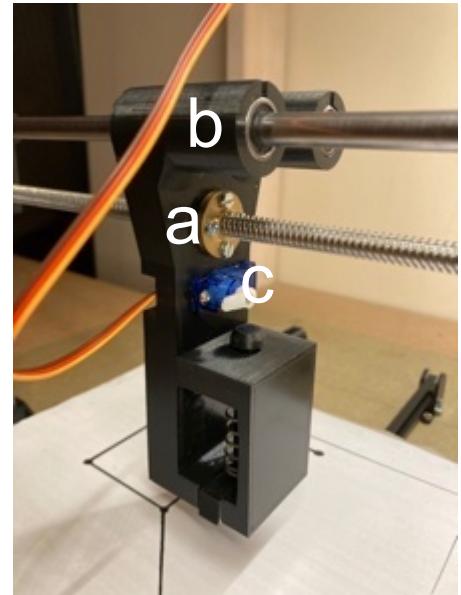
Aan de penhouder moeten 6 dingen gemonteerd worden. Twee lineaire kogellagers, de leadscrew moer, de servomotor en de pen met een veer.



Afbeelding 32 : 3D tekening penhouder



Afbeelding 33 : Foto 1 van de penhouder



Afbeelding 34 : Foto 2 van de penhouder

(a)

De leadscrew moer dient om de pen van links naar rechts te kunnen verschuiven door middel van de leadscrew die gedraaid wordt door de stappenmotor. Hiervoor heb ik aan de penhouder een ronde opening voorzien met de diameter van het kokertje (1) van de leadscrew moer. Rondom deze openingen heb ik vier kleinere ronde openingen voorzien om de leadscrew moer vast te kunnen maken aan de penhouder door middel van moeren en bouten.



Afbeelding 35 : Leadscrew moer

(b)

De twee lineaire kogellagers zorgen ervoor dat de pen altijd volgens een rechte lijn zal bewegen en dat de penhouder niet zou beginnen rond draaien. Om deze aan de penhouder te bevestigen heb ik twee kokers voorzien met een gat in. In deze kokers kunnen de twee kogellagers ingeduwd worden.

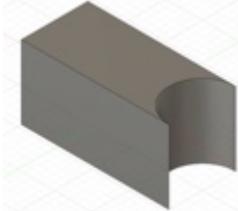
(c)

De servomotor moet dienen om de pen naar boven en naar beneden te duwen om al dan niet te schrijven. Om deze te monteren heb ik in de penhouder een rechthoekige opening voorzien met de afmetingen van de servomotor. Links en rechts naast deze opening heb ik nog twee kleine ronde openingen voorzien om de servomotor vast te schroeven. Aan de achterkant van de penhouder en aan de linkerkant van de opening zijn er rechthoekige openingen gelaten voor de kabels.

(d)

Om de pen en de veer in de penhouder te monteren heb ik aan de penhouder een grote rechthoekige opening gelaten. Aan de bovenkant van de rechthoekige opening heb ik een ronde opening ontworpen die net iets groter is dan de diameter van de pen. Door deze opening komt de pen door zodat de servomotor op de pen kan duwen. Onderaan de rechthoekige opening is er een kleinere rechthoekige opening met een afronding gelaten om de pen te kunnen vervangen. Aan het uiteinde van de afronding is een ronde inkeping met de diameter van de veer om deze op zijn plaats te houden.

Om de pen op zijn plaats te houden heb een onderdeel ontworpen dat in de rechthoekige opening met de afronding past. Dit onderdeel is een balk met een uitsnede van een halve cirkel met als diameter de diameter van de pen.



Afbeelding 36 : 3D tekening stukje om de pen tegen te houden

Om ervoor te zorgen dat de veer mee naar beneden gedrukt wordt wanneer de servomotor op de pen duwt heb ik een onderdeel ontworpen. Dit is een cilinder met in het midden een ronde opening met de diameter van de pen. Dit cilindertje kan op de pen geduwd worden en zal niet bewegen. In dit onderdeeltje is ook een inkeping voorzien waar de veer in past.



Afbeelding 37 : 3D tekening veer houder

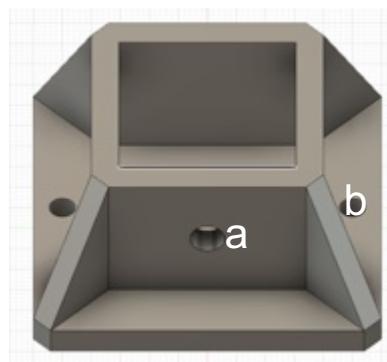
Om ervoor te zorgen dat wanneer de servomotor op de pen duwt dit altijd zal zorgen voor een gelijk verschil in hoogte heb ik een pendop ontworpen. Dit onderdeel is een afgeplatte halve bol met een opening voor de pen. Wanneer de arm van de servomotor op het afgeplatte deel komt zal de pen niet meer naar beneden geduwd worden.



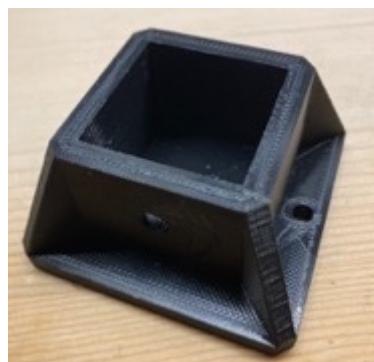
Afbeelding 38 : 3D tekening pendop

#### 5.1.4 Montagerail voet

Om de montagerail vast te maken aan de montageplaat heb ik een stuk ontworpen waar deze in vast gemaakt kan worden een glijmoer. In het stuk is er voor de bout een ronde opening voorzien (a). Aan dit stuk zijn noch twee ronde openingen voorzien om het stuk vast te vijzen aan de montageplaat (b).



Afbeelding 39 : 3D tekening  
montagerail voet

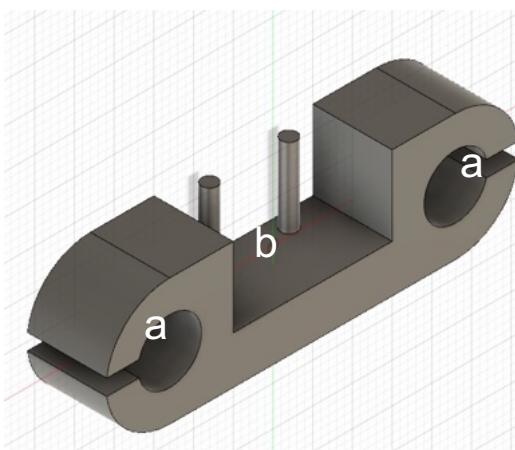


Afbeelding 40 : Montagerail voet

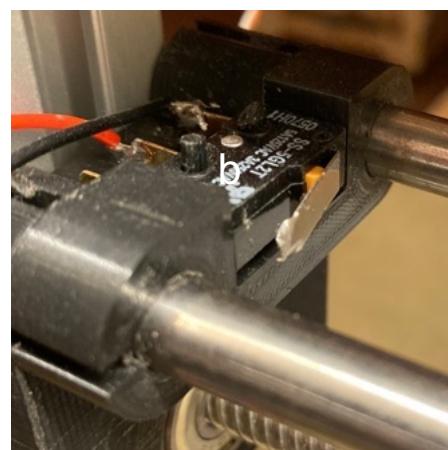
#### 5.1.5 Eindeloopschakelaarhouder x-as

Net zoals bij de y-as moet het nulpunt van de x-as bepaald worden. Dit gebeurt met een eindeloopschakelaar. Het nulpunt zal bereikt zijn wanneer de penhouder tegen de schakelaar botst.

Voor deze schakelaar heb ik een houder ontworpen. Deze wordt op zijn plaats gehouden door twee klemmen (a). Deze klemmen op de geleidingsassen van de x-as. Net zoals bij de y-as heeft deze houder ook twee pinnen op het stuk waar er aan de schakelaar openingen gelaten zijn voor montage (b).



Afbeelding 41 : 3D tekening  
eindeloopschakelaarhouder x-as



Afbeelding 42 : Eindeloopschakelaar  
houder x-as

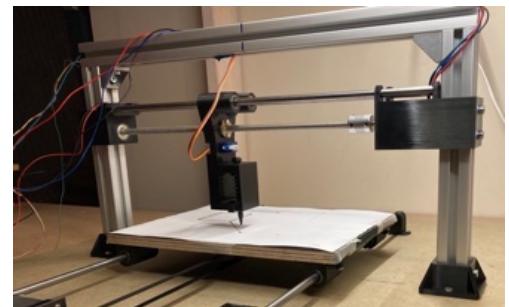
## 5.2 Montage van de x-as

Om de x-as te monteren ben ik begonnen met de leadscrew moer vast te schroeven in de penhouder. Daarna heb ik de lineaire kogellagers in de penhouder gedrukt met behulp van een bankschroef. De servomotor heb ik in de opening gedrukt en vastgeschroefd in de daarvoor voorziene openingen. Daarna heb ik op de pen de veerhouder gedrukt en de pen samen met de veer in de penhouder gestoken. Tenslotte heb ik het pendopje op de pen gezet en het stukje om de pen tegen te houden gemonteerd aan de penhouder.



Afbeelding 43 : Montage van de x-as deel 1

Wanneer de penhouder volledig gemonteerd was heb ik de motor in het motorbevestigingsstuk gemonteerd. Hierna heb ik de geleidingsassen en de leadscrew op maat geslepen en de montagerail op maat gezaagd en in elkaar gevezen. Verder heb ik de kogellagers in hun klemmen geduwd en de leadscrew gemonteerd aan de motor. Dan heb ik de geleidingsassen en de leadscrew gemonteerd en de montagerailvoeten op hun plaats geschroefd. Voordat ik het geheel aan de brug heb geschroefd heb ik eerst de eindeloopschakelaar nog gemonteerd aan de geleidingsassen tegen de kogellager klem. Hierna heb ik alles gemonteerd aan de montagerail en deze vast gemaakt in de montagerail voeten. Ik heb alles uitgelijnd zodat de pen het bed raakt wanneer de servomotor de pen naar beneden duwt.



Afbeelding 44 : Montage van de x-as deel 2

## 6 Elektrisch ontwerp

### 6.1 Aansluiten van de componenten

Voor alle aansluitingen in een schema zie aansluitschema 6.2.

Om de aansluitingen van de motoren en de eindeloopschakelaars op de Arduino gemakkelijker te maken heb ik gebruik gemaakt van een CNC shield. Op het shield zijn headers en pinnen op voorzien om alle componenten aan te sluiten. Hierop kunnen de motorcontrollers, stappenmotoren, eindeloopschakellaars en de servomotor op aangesloten worden. Het shield zorgt dan voor de verbinding van deze pinnen en headers naar de Arduino.

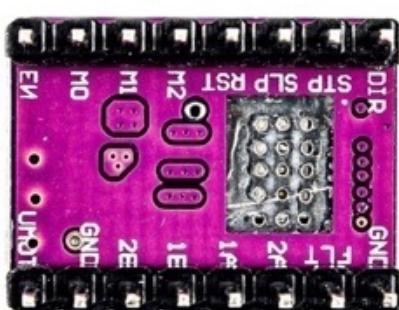
Bij het aansluiten van de componenten ben ik begonnen met het aansluiten van de motorcontrollers. Deze heb ik aangesloten op het shield zodat de enable pin (EN) van de controller overeenkomt met de enable pin van het shield.

De stappenmotoren heb ik aangesloten op het shield. Hoe men deze aansluit op het shield maakt eigenlijk niet uit. Het enige verschil gaat zijn dat de draairichting omgekeerd gaat zijn maar dit kan softwarematig nog aangepast worden. Bij mij zijn de motoren aangesloten met de zwarte kabel van de motor het dichtste bij de letter van de as.

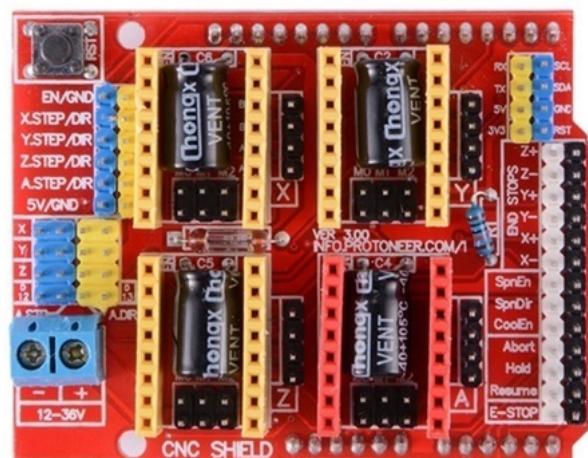
De x- en y-as hebben elk een eindeloopschakelaar om het nulpunt te bepalen van de penplotter. Deze zijn aangesloten op de end stops van het shield. Die van de x-as zit aangesloten aan de X+ pin en die van de y-as aan de Y- pin.

De z-as wordt bediend met een servomotor maar het shield is gemaakt voor stappenmotoren. Om dit op te lossen kan men gebruik maken van de pin die zorgt voor de eindeloopschakelaar van de z-as omdat men deze niet nodig heeft. Buiten een aanstuur pin heeft een servomotor ook nog een 5V en een GND (ground, minpool) pin. Deze kunnen aangesloten worden op de 5V en GND pinnen van het shield.

Ik gebruik een 12V voeding om de motoren aan te sturen. Deze wordt aangesloten op de 12-36V schroefverbinding op het CNC shield.

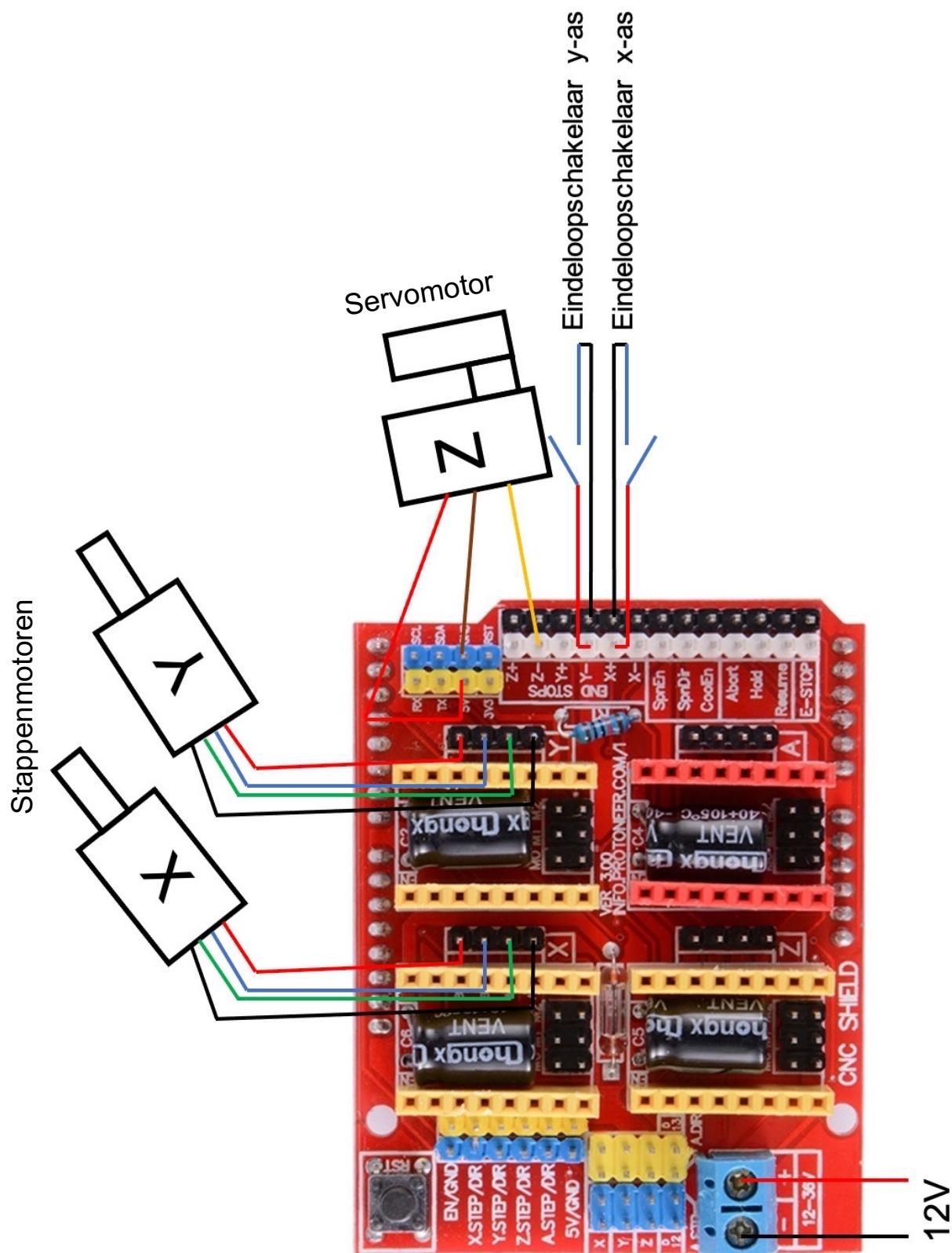


Afbeelding 45 : Pin lay-out  
DRV8825 motorcontroller



Afbeelding 46 : Pin lay-out CNC shield

## 6.2 Aansluitschema



Afbeelding 47 : Aansluitschema

### 6.3 Motorcontrollers instellen en microstepping

Na het aansluiten van de componenten heb ik mijn penplotter uitgeprobeerd maar deze trilde zeer hard en maakte veel geluid. Eerst dacht ik dat dit een mechanisch probleem was maar vond hier geen oplossing voor. Na wat opzoekwerk ben ik erop uitgekomen dat ik de motorcontrollers nog niet had ingesteld. Hierdoor krijgen de motoren te veel stroom.

De motorcontrollers instellen kan door de referentiespanning van de controller aan te passen met het schroefje op de controller (a). Deze spanning wordt gemeten tussen het schroefje en de minpool van de spanningsbron.

Bij de DRV8825 controller is de referentie spanning gelijk aan de maximale stroom van de motor gedeeld door twee:  $U_{ref} = \frac{I_{max}}{2}$ .

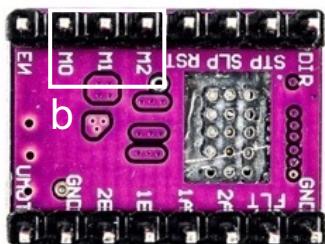
Bij een NEMA 17 0,55 Nm stappenmotor bedraagt de maximale stroom 1,5 A dus moet de referentiespanning gelijk zijn aan 0,75 V.

Om het trillen van de constructie nog meer te beperken heb ik microstepping ingesteld bij de controllers. Microstepping verdeelt een stap van de stappenmotoren nog eens in kleinere stappen. Als de motor meer stappen moet doen om een omwenteling te bekomen zal deze minder hard trillen. Dit instellen kan door bepaalde M pinnen van de controller van spanning te voorzien (b). Deze combinatie van pinnen die van een spanning voorzien zijn zal een bepaalde verkleining geven (zie tabel 3). Op het CNC shield kunnen deze pinnen van spanning voorzien worden door een verbinding te maken bij de pinnen van M0, M1 en M2 (c) onder de voorziene plaats voor de controllers. Ik heb gekozen voor een 1/16 verkleining voor mijn y-as en een 1/4 verkleining voor mijn x-as. Dit wil zeggen dat elke stap van de motor van mijn y-as nog eens in 16 stappen zal verdeeld worden. Elke stap van mijn x-as zal dan nog eens verdeeld worden in 4 stappen. Zonder verkleining hebben mijn stappenmotoren 200 stappen/omwenteling. Met deze verkleining zal de motor van mijn y-as 3200 stappen/omwenteling hebben en die van mijn x-as 800 stappen/omwenteling. De kleinere stapverkleining voor de x-as heb ik gekozen door problemen bij het zelf programmeren van mijn penplotter (zie 9.3).

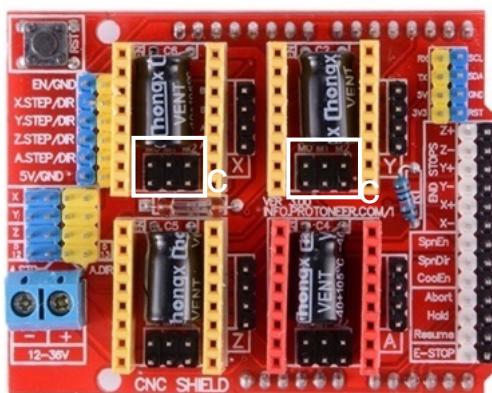
M0	M1	M2	Stappen verkleining
0	0	0	1
1	0	0	1/2
0	1	0	1/4
1	1	0	1/8
0	0	1	1/16
1	0	1	1/32

0 = geen spanning 1 = spanning

Tabel 3 : Stapverkleining



Afbeelding 49 : Pin lay-out  
DRV8825 motorcontroller



Afbeelding 50 : Pin lay-out CNC shield



Afbeelding 48 : Foto van de motorcontroller

## 7 CNC principes van mijn CNC penplotter

### 7.1 Referentiepunt bepalen van de machine

Het referentiepunt bepalen van de machine wordt in het Engels de homing sequence genoemd. Dit moet gebeuren omdat een stappenmotor in heel kleine stappen kan bewegen maar niet weet waar deze zich bevindt of hoeveel stappen deze al gemaakt heeft.

Wanneer een machine deze taak uitvoert bewegen zijn assen in een vooraf bepaalde richting waar eindeloopschakelaars staan. Op het moment dat een eindeloopschakelaar van een as ingedrukt wordt is dit het referentiepunt van die as. Vanaf het referentiepunt bepaald is kan de machine bijhouden hoeveel stappen de assen in welke richting gemaakt hebben. Aan de hand hiervan weet de machine waar die zich bevindt ten opzichte van het referentiepunt.

Bij mijn CNC penplotter wordt het referentiepunt bepaald voor x- en y-as. Het referentiepunt ligt op het punt X0 Y0 en bevindt zich in de linkerbovenhoek van het blad. Als ik over mijn penplotter spreek gebruik ik dus het woord nulpunt in plaats van referentiepunt.

### 7.2 Stappen per millimeter

Stappen per millimeter is een getal dat aangeeft hoeveel stappen de stappenmotor moet maken om het werktuig één millimeter te verplaatsen volgens een as van de machine. Dit getal kan verschillend zijn voor de verschillende assen van de machine. Dit getal kan berekend worden wanneer men weet hoeveel stappen de motor nodig heeft om één omwenteling te maken en hoeveel millimeter het werktuig hierdoor verplaatst wordt. Hoe hoger dit getal hoe nauwkeuriger de as zal zijn.

Voor de x-as:

De x-as van mijn CNC penplotter beweegt door een 8x8 leadscrew.

Eén omwenteling zal de pen acht millimeter verplaatsen volgen de x-as. We weten ook dat de stappenmotor 800 stappen nodig heeft om één omwenteling te maken.

Het aantal stappen per millimeter is dus gelijk aan  $\frac{800 \text{ stappen}}{8 \text{ mm}} = 100 \frac{\text{stappen}}{\text{mm}}$ .

Voor de y-as:

De motorpulley (GT2-20) van de y-as heeft 20 tanden en de timing belt heeft een pitch van 2 mm. Dit wil zeggen dat wanneer de motorpulley van een tand naar de tand erna draait het schrijfbed 2mm opgeschoven zal zijn. Deze motorpulley heeft 20 tanden dus zal het schrijfbed 40 mm opschuiven bij één omwenteling. We weten ook dat de stappenmotor van de y-as 3200 stappen nodig heeft om één omwenteling te maken.

Het aantal stappen per millimeter is dus gelijk aan  $\frac{3200 \text{ stappen}}{40 \text{ mm}} = 80 \frac{\text{stappen}}{\text{mm}}$ .

## 8 CNC penplotter aansturen met bestaande software

Ik stuur mijn CNC penplotter aan met grbl mi. Dit is een software waarmee je door middel van commando's dingen tekenen. Deze software maakt gebruik van de Arduino IDE. Door de serial monitor van de Arduino IDE kan je dan commando's sturen.

### 8.1 Instellingen

Deze instellingen zijn niet alle instellingen van grbl mi maar voor mijn cnc penplotter de belangrijkste.<sup>2</sup>

\$3=1 (dir port invert mask:00000001)

Deze instelling kan de positieve draairichting van bepaalde assen omdraaien. Na het uittesten van de standaard instellingen van grbl mi bleek echter dat dit nodig was voor de x-as van mijn penplotter. Om dit op te lossen heb ik deze instelling veranderd van 0 naar 1. Het programma weet over welke as het gaat door het getal om te zetten naar een binair getal en te kijken naar de eerste drie bits. Als de eerste bit een 1 is zal deze de draairichting voor de x-as omdraaien. Als de tweede bit 1 is voor de y-as. Tenslotte ook voor de z-as maar deze gebruik ik niet bij mijn penplotter.

\$22=1 (homming cycle, bool)

Deze instelling zorgt ervoor dat ik mijn machine de homming sequence van grbl mi kan gebruiken. Dit doe ik door deze instelling gelijk te stellen aan 1.

\$23=3 (homming dir invert mask:00000011)

De grbl mi instellingen gaan er van uit dat de eindeloopschakelaars voor de homming sequence van de machine aan het einde van de positieve richting van elke as staan. Bij mij is dit niet het geval deze staan aan het begin van mijn assen in mijn nulpunt. Om dit op te lossen heb ik deze instelling aangepast naar 3. Net zoals bij het omkeren van de draairichting zal de software dit getal omzetten daar een binair getal en kijken naar de laatste drie bits om te weten over welke as het gaat.

\$24=25.000 (homming feed, mm/min)

\$25=500.000 (homming seek, mm/min)

Deze instellingen zijn de snelheden voor de homming sequence van de machine. De homming seek snelheid zal de snelheid zijn waar de assen aan zullen bewegen naar de eindeloopschakelaars toe. Wanneer beide eindeloopschakelaars geraakt zijn zullen de assen een klein stukje terug bewegen aan de homming feed snelheid terug naar de eindeloopschakelaars bewegen om een nauwkeuriger resultaat te hebben.

\$100=100.000 (x, step/mm)

\$101=80.000 (y, step/mm)

Deze instellingen geven mee aan het programma hoeveel stappen de motoren van beide assen moeten maken voor één millimeter te moeten verplaatsen (zie 7.2).

---

<sup>2</sup> <https://github.com/gnea/grbl/wiki/Grbl-v1.1-Configuration>

$\$130=295.000$  (x max travel, mm)  
 $\$131=210.000$  (y max travel, mm)

In deze instellingen worden de uiterste randen meegegeven aan de machine. Voor mij zal dit bij de x-as 295 mm zijn en volgens de y-as 210 mm. Dit zijn ongeveer de afmetingen van een A4-papier.

## 8.2 Commando's

Grbl mi werkt door middel van commando's. Om de penplotter aan te sturen zullen er commando's naar de Arduino gestuurd worden door middel van de seriële monitor van de Arduino IDE.

Wanneer men mijn penplotter wil gebruiken moet men eerst het commando "\$H" gebruiken om het nulpunt van de machine te bepalen. Wanneer de machine zijn nulpunt bereikt geeft men het commando "G92 X0 Y0" in. Met dit commando weet de machine dat hij zich in zijn nulpunt bevindt. Dit is nodig omdat men eindeloopschakelaars in het uiterste punt van elke as kan zetten en van dit punt kan refereren.

Om de penplotter te bewegen naar een bepaald punt gebruikt men het commando "G1 X Y F". G1 betekent dat de penplotter een rechtlijnige beweging zal maken. Na de X en de Y geeft men dan de coördinaat in waar men de pen wil naartoe bewegen. Na de F volgt een snelheid waarmee de pen moet bewegen in mm/min. Als men de pen wil bewegen naar het punt X150 Y25 met een snelheid van 2000 mm/min wordt het commando "G1 X150 Y25 F2000".

Om de pen te laten tekenen op het papier gebruikt men het commando "m3 s". De m betekent hier dat de servomotor gaat draaien. De 3 wil zeggen dat men de servomotor wil laten bewegen met een bepaalde hoek. De hoek wordt dan aangegeven door het getal achter de s. Als men de servomotor met een hoek van 90° willen laten draaien wordt het commando "m3 s90". De hoek die de servomotor moet maken om de pen het blad te laten raken hangt af van de dikte van het blad. Als de pen gedaan heeft met tekenen kan de servomotor terug naar zijn startpositie gebracht worden met het commando "m5". Wanneer men een lijn wil tekenen van X15 Y30 tot X100 Y95 kan dit door deze reeks van commando's:  
"\$H" "G92 X0 Y0" "G1 X15 Y30 F2000" "m3 s100" "G1 X100 Y95 F200" "m5".  
Stel dat men hierna nog een lijn wil tekenen hoeft het niet meer om het commando voor de homing sequense te gebruiken.  
Bijvoorbeeld nog een lijn van X10 Y10 naar X125 Y40:  
"G1 X10 Y10 F2000" "m3 s100" "G1 X125 Y40 F200" "m5".

## 9 Eigen software voor basis vormen

Bij mijn eigen programmatie werkt de Arduino als vertaler tussen mijn computer en mijn penplotter. In de computer wordt er een commando ingegeven en de Arduino vertaalt deze door middel van mijn programma naar bewegingen die de motoren moeten maken.

### 9.1 Algemene code

In het algemeen deel van het programma definiere ik variabelen en constanten die in het volledige programma meerdere malen gebruikt worden of dingen die moeten gebeuren om de Arduino meer informatie mee te geven. Op volgende bladzijden wordt deze code beschreven.

```
//pins x axis + micros for the steps
#define pulseX 2
#define dirX 5
unsigned long Xmicros;

//pins y axis + micros for the steps
#define pulseY 3
#define dirY 6
unsigned long Ymicros;

//enable pin
#define enablePin 8

//limitswitches
#define limitX 9
#define limitY 10

//speeds in mm/s
#define moveSpeed 15
#define drawSpeed 5
#define homeSpeed 10

//steps/mm for the axis
#define Xstepsmm 100
#define Ystepsmm 80

//locations
float X;
float Y;

//z axis
#include <Servo.h>
Servo Zaxis;
int angle = 100;
```

Programmatie algemene code deel 1

```
void setup() {
    pinMode(pulseX, OUTPUT);
    pinMode(dirX, OUTPUT);

    pinMode(enablePin, OUTPUT);

    pinMode(limitX, INPUT_PULLUP);
    pinMode(limitY, INPUT_PULLUP);

    digitalWrite(enablePin, HIGH);
    digitalWrite(pulseX, LOW);
    digitalWrite(dirX, LOW);

    digitalWrite(pulseY, LOW);
    digitalWrite(dirY, LOW);

    Zaxis.attach(11);
    Zaxis.write(0);

    Serial.begin(9600);

    Home();
}
```

Programmatie algemene code deel 2

In deel één van de algemene code maak ik variabelen aan en definieer ik constanten.

Mijn stappenmotoren werken met een puls-, richting- en enable pin. Doordat ik gebruik maak van het CNC shield is de enable pin maar één pin voor beide stappenmotoren.

De enable pin zal hoog (5 V) of laag (0 V) geschakeld worden. Als deze laag geschakeld wordt zullen de motoren kunnen draaien. In tegenstelling tot wanneer deze hoog geschakeld wordt dan zullen de motoren niet kunnen draaien.

De richting pin van de motor zal hoog (5 V) of laag (0 V) geschakeld worden.

Aan de hand hiervan weet de motor of deze urwijzerzin of tegen urwijzerzin moet draaien.

De puls geeft aan wanneer de motor een stap moet nemen. Elke keer dat de puls pin een kort hoog signaal krijgt zal deze een stap nemen in de meegegeven richting.

Om deze signalen naar de motoren te sturen moet de Arduino weten op welke pinnen van de microcontroller de pulse-, richting- en enable pinnen zijn aangesloten. Dit doe ik door middel van een #define gevuld door een logische naam voor de pinnen en welke pin het is. Wanneer de naam die achter de #define komt later in het programma gebruikt zal worden wordt deze vervangen door het getal er achter. Dit gebeurt voor de code op de Arduino gezet wordt dus door een #define te gebruiken gaan deze getallen niet opgeslagen worden in het geheugen van de Arduino. Deze #define gebruik ik ook voor de pinnen van de eindeloopschakelaars (limitX en limitY), de snelheden en de stappen/mm van elke as.

Dit gebruik ik niet voor de pin van de servomotor aan te geven, omdat ik hier een bibliotheek voor gebruik. Een bibliotheek is een al bestaande code die ik kan implementeren in mijn eigen code. Wanneer men een bibliotheek wil gebruiken begint men met een #include gevuld door <"naam van de bibliotheek".h>.

In het geval van de servomotor: #include <Servo.h>. In het geval van de servomotorbibliotheek geeft men eerst de servomotor een naam door na de implementatie van de bibliotheek volgende functie te gebruiken: "Servo + naam van de servo". In mijn geval: "Servo Zaxis".

Voor elke stappenumotor heb ik een variabele aangemaakt met de namen X- en Ymicros met het data type unsigned long. Dit is om later in het programma te kunnen meten hoeveel tijd er al verlopen is na de laatste stap (zie 9.2 en 9.3).

Met een unsigned long kan een positief geheel getal dat binair geschreven kan worden in 4 bytes of 32 bits. Dit zijn de getallen van 0 tot 4 294 967 294.<sup>3</sup>

Om de hoek van de servomotor bij te houden heb ik een variabele aangemaakt met de naam "angle" van het type int. Dit heb ik gedaan omdat de hoek waar de motor zou moeten staan om op het blad te tekenen kan variëren door de dikte van het blad. Het type int betekent dat het hier gaat over gehele getallen die binair geschreven kunnen worden in 2 bytes of 16 bits.

Dit zijn de getallen van -32768 tot +32767.<sup>4</sup>

Om de positie van de pen bij te houden heb ik ook een variabele X en een variabele Y aangemaakt van het type float. Het type float kan kommagetallen opslagen die binair geschreven kunnen worden in 4 bytes of 32 bits.

Dit zijn de getallen van  $-3.4028235 \cdot 10^{38}$  tot  $3.4028235 \cdot 10^{38}$ .<sup>5</sup>

---

<sup>3</sup> <https://www.arduino.cc/reference/en/language/variables/data-types/unsignedlong/>

<sup>4</sup> <https://www.arduino.cc/reference/en/language/variables/data-types/int/>

<sup>5</sup> <https://www.arduino.cc/reference/en/language/variables/data-types/float/>

Het tweede deel van mijn algemene code is geschreven in de “void setup(){}”. Hierin schrijft men code dat maar één keer moet gebeuren.

Ik geef hier aan welke pinnen een ingang (INPUT) of een uitgang (OUTPUT) zijn. Een ingang kan uitgelezen worden en een uitgang kan iets aansturen. De pinnen van de stappenmotoren zijn uitgangen en de eindeloopschakelaars zijn ingangen. Om te voorkomen dat er een kortsluiting optreedt zijn de eindeloopschakelaars ingegeven als “INPUT\_PULLUP”. Hiermee maakt de Arduino gebruik van een interne weerstand voor deze pinnen. Het nadeel van deze techniek is dat de logica van deze schakelaars omgekeerd zal zijn. Wanneer de schakelaars open zijn zullen deze een 1 of een hoog signaal geven als men deze uitleest.

Wanneer deze gesloten zijn zullen ze een 0 of een laag signaal geven.

Een andere manier zou zijn om een grote externe weerstand te schakelen tussen de digitale pin en de ground (negatieve kant van de arduino). Dit kan ik niet toepassen want op het CNC shield zit er geen externe weerstand.

In dit deel schrijf ik ook de enable pin hoog zodat de motoren in elk geval niet bewegen. Dit doe ik door “digitalWrite(enablePin, HIGH);” in de setup te schrijven. Ook heb ik om fouten te voorkomen de andere pinnen van de stappenmotoren laag geschreven.

De pin van de servomotor geef ik aan met “Zaxis.attach(11);”. Hierna breng ik de servomotor naar zijn nulpunt door “Zaxis.write(0);”.

Om commando's van mijn computer naar de Arduino te sturen gebruik ik in het setup gedeelte de lijn “Serial.begin(9600);”.

Het tweede deel van de algemene code sluit af met de machine naar zijn nulpunt te brengen (zie 9.2).

## 9.2 Homing sequence

Dit deel van het programma is de homing sequence. Door dit deel van het programma uit te voeren zal de pen naar zijn referentie punt bewegen en van dit punt zijn nulpunt maken. Op de volgende bladzijde wordt deze code beschreven.

```
void Home(){

    digitalWrite(enablePin, LOW);

    //calculate time
    int timeX = 1000000/(homeSpeed*Xstepsmm);
    int timeY = 1000000/(homeSpeed*Ystepsmm);

    //direction
    digitalWrite(dirX, LOW);
    digitalWrite(dirY, HIGH);

    Xmicros = micros();
    Ymicros = micros();
```

Programmatie homing sequence deel 1

```
//move the motors until they hit the limit switch
while((digitalRead(limitX) == 1) || (digitalRead(limitY) == 1)){

    if(digitalRead(limitX) == 1){
        if((micros() - Xmicros) >= timeX){
            digitalWrite(pulseX, HIGH);

            Xmicros = micros();
        }
        digitalWrite(pulseX, LOW);

        if(digitalRead(limitY) == 1){
            if((micros() - Ymicros) >= timeY){
                digitalWrite(pulseY, HIGH);
                Ymicros = micros();
            }
            digitalWrite(pulseY, LOW);
        }
        digitalWrite(enablePin, HIGH);

        X = 0;
        Y = 0;

    }
}
```

Programmatie homing sequence deel 2

Om mijn CNC penplotter naar zijn nulpunt te brengen heb ik in het programma een functie geschreven. Wanneer deze functie opgeroepen wordt zal de machine naar zijn nulpunt bewegen. Een functie schrijven in de software van Arduino gebeurt als volgt: void + "naam van de functie" + () +{ + "code van de functie" + }.

Voor de homing sequence: void Home(){}. Tussen de accolades komt de code om de penplotter zijn nulpunt te bepalen. "void" betekent dat het gaat om een functie die geen informatie terugstuurt.<sup>6</sup>

In het eerste deel van deze code schakel ik de enable pin laag zodat de motoren kunnen draaien. Hierna bereken ik voor elke as de tijd die tussen de pulsen nodig is. Ik heb dit gedaan met een variabele zodat het programma kan berekenen hoeveel tijd er nodig is als de snelheid voor de homing sequence verandert.

$$\text{De berekening gebeurt als volgt: } \text{tijd} = \frac{1000000}{\text{snelheid} \left( \frac{\text{mm}}{\text{s}} \right) \frac{\text{stappen}}{\text{mm}}} = \frac{\mu\text{s}}{\text{stap}} .$$

Hierna schakel ik de richtingspinnen van de stappenummotoren zodat de assen naar de eindeloopschakelaars bewegen: voor de x-as laag, voor de y-as hoog.

Tenslotte maak ik de variabelen Xmicros en Ymicros gelijk aan het aantal microseconden dat het programma al aan het lopen is.

Het tweede deel van de code van de homing sequence begint met de lijn: "while((digitalRead(limitX) == 1) || (digitalRead(limitY) == 1)){".

De while in deze lijn betekent dat de code die tussen de accolades van dezelus staat zal blijven herhaald worden tot de voorwaarden die meegegeven worden niet meer juist zijn. In mijn geval zal de lus stoppen wanneer beide eindeloopschakelaars in gedrukt zijn. Het symbool "||" wil zeggen "of". Deze lus zal blijven lopen tot beide voorwaarden niet meer van toepassing zijn. Dit is nodig voor het geval dat slechts één as zijn eindeloopschakelaar al heeft in gedrukt. De plotter zal weten waar het nulpunt van deze as is maar nog niet van de andere as.

In de while lus begin ik met te controleren of de x-as zijn nulpunt nog niet bereikt heeft met de lijn: if(digitalRead(limitX) == 1){. Dit is nodig omdat de while lus blijft herhaald worden ook al heeft één as zijn nulpunt al bereikt. Als de x-as zijn nulpunt nog niet bereikt heeft ga ik verder met te controleren of de tijd tussen de laatste pulse en de puls die moet komen al verstrekken is met de lijn:

"if((micros() - Xmicros) >= timeX){". Hierin ga ik na of het verschil van het aantal microseconden dat het programma al loopt en het tijdstip dat de laatste puls was gebeurd groter of gelijk is aan de tijd tussen de pulsen van de x-as is. Als dit het geval is schakel ik de pulse pin van de x-as hoog en zet ik de tijd van de laatste puls naar het huidige aantal microseconden dat het programma al loopt. Nadat het programma dit al dan niet heeft uitgevoerd schakel ik de pulse pin laag. Hierna doe ik voor de y-as hetzelfde.

Wanneer beide voorwaarden van de while lus niet meer juist zijn en dus het nulpunt van de penplotter bereikt is schakel ik de enable pin terug hoog en maak de variabelen X en Y gelijk aan 0. Deze variabelen houden de positie van de pen bij.

---

<sup>6</sup> <https://www.arduino.cc/reference/en/language/variables/data-types/void/>

### 9.3 Naar een bepaalde plaats bewegen

Dit deel van de code zorgt ervoor dat de pen van zijn huidige locatie naar een ander punt kan bewegen. Ik heb hier een aparte functie voor geschreven omdat dan de programmatie van de figuren gemakkelijker is. Op de volgende bladzijde staat deel twee van de code. Op de bladzijde daarna staat de code beschreven.

```
void moveTo(float mX, float mY, int Speed){  
  
    //determine direction  
  
    if((mX - X) > 0){  
        digitalWrite(dirX, HIGH);  
    }  
    else{  
        digitalWrite(dirX, LOW);  
    }  
  
    if((mY - Y) > 0){  
        digitalWrite(dirY, LOW);  
    }  
    else{  
        digitalWrite(dirY, HIGH);  
    }  
  
    //calculate speed and time  
    float Length = sqrt(pow((mX - X),2)+(pow((mY - Y), 2)));  
  
    float speedX = (abs((mX - X))/Length)*Speed;  
    float speedY = (abs((mY - Y))/Length)*Speed;  
  
    float timeX = 1000000/(speedX*Xstepsmm);  
    float timeY = 1000000/(speedY*Ystepsmm);  
  
    double stepsX = 0;  
    double stepsY = 0;  
  
    //needed steps  
    double stepsNeededX = abs((mX - X))*Xstepsmm;  
    double stepsNeededY = abs((mY - Y))*Ystepsmm;
```

Programmatie bewegen naar een bepaalde plaats deel 1

```

//move the motors
digitalWrite(enablePin, LOW);

Xmicros = micros();
Ymicros = micros();

while((stepsX < stepsNeededX) || (stepsY < stepsNeededY)){

    if(stepsY < stepsNeededY){
        if((micros() - Ymicros) >= timeY){

            digitalWrite(pulseY, HIGH);
            stepsY += 1;
            Ymicros = micros();
        }
    }
    digitalWrite(pulseY, LOW);

    if(stepsX < stepsNeededX){
        if((micros() - Xmicros) >= timeX){

            digitalWrite(pulseX, HIGH);
            stepsX += 1;
            Xmicros = micros();
        }
    }
    digitalWrite(pulseX, LOW);
}

digitalWrite(enablePin, HIGH);

X = mX;
Y = mY;
}

```

Programmatie bewegen naar een bepaalde plaats deel 2

De code om naar een bepaalde plaats te bewegen heb ik geschreven met een functie. Om deze functie uit te voeren moeten er drie variabelen mee gegeven worden. Deze variabelen zijn de x- en y-coördinaat waar de pen naar moet bewegen en de snelheid waarmee de pen moet bewegen. Het bewegen van één plaats naar een andere plaats zal gebeuren aan de movespeed en het tekenen aan de drawspeed.

Deel één van deze code begint met het bepalen van de richting die de stappenmotoren van de assen moeten draaien. Dit doe ik door te onderzoeken of het verschil tussen de x- of y-coördinaat waar de pen naar toe moet en de x- of y-coördinaat van de pen positief of negatief is volgens die bepaalde as. Bij een positief resultaat voor de x-coördinaten zal deze de richting pin van de x-as hoog geschakeld worden anders laag. Bij de y-coördinaten zal bij een positief resultaat de richting pin van de y-as laag geschakeld worden anders hoog.

Om de snelheden van de x- en y-as te bepalen begin ik met de lengte te berekenen die de pen moet afleggen.

Dit doe ik door: "float Length = sqrt((pow((mX - X),2))+(pow((mY - Y), 2)));". Deze lijn brekend de lengte met behulp van de stelling van Pythagoras. Later kan ik hiermee de sinus en cosinus van de lijn berekenen om de snelheden te berekenen.

De snelheden zelf kunnen berekend worden door de absolute waarde van het verschil in coördinaten te delen door de lengte dat de pen moet bewegen. Dit is voor de x-as de cosinus van die lijn en de sinus voor de y-as. Dit vermenigvuldig ik met de snelheid die de pen moet bewegen.

Dit gebeurt voor de x-as met de lijn: "float speedX = (abs((mX - X))/Length)\*Speed;". Met de snelheden voor de x-en y-as kan ik net zoals bij de homing sequence de tijd tussen de pulsen berekenen.

Dit wordt voor de x-as: "float timeX = 1000000/(speedX\*Xstepsmm);".

Hierna maak ik nog twee variabelen stepsX en stepsY aan. Met deze variabelen hou ik bij hoeveel stappen de x- en y-as al gemaakt hebben. Hierna maak ik ook nog twee variabelen stepsNeededX en stepsNeededY aan. Voor deze variabelen bereken ik hoeveel stappen de x- en y-as moeten maken. Dit doe ik door de absolute waarde van het verschil tussen de coördinaten van de pen en de plaats waar de pen naar toe moet bewegen te vermenigvuldigen met de stappen/mm van de assen.

In deel twee van de code begin ik met de enable pin laag te schakelen en de X- en Ymicros gelijk te stellen aan het aantal microseconden dat het programma al loopt. Het bewegen van de motoren is gelijkaardig aan dat van de homing sequence enkel ga ik hier in de while lus na of de stappen van de x- en y-as al gelijk of groter is aan het aantal stappen dat de assen moeten maken (stepsNeededX en stepsNeededY). Dit gebeurt met: "while((stepsX < stepsNeededX) || (stepsY < stepsNeededY)){".

In deze lus begin ik met het nagaan of het aantal stappen van de x-as al groter is dan de stappen die nodig zijn. Als dit het geval is ga ik na of het verschil tussen de microseconden dat het programma al loopt en het tijdstip van de laatste pulse van de x-as groter is dan de tijd tussen de pulsen. Als dit het geval is schakel ik de pin hoog en tel ik bij het aantal stappen van de as één stap bij. En stel ik de Xmicros gelijk aan het aantal microseconden dat het programma al loopt. Als dit al dan niet gebeurd is schakel ik de pulse pin van de x-as laag. Voor de y-as doe ik hetzelfde als voor de x-as. Wanneer beide motoren al hun stappen gedaan hebben en dus de pen op het punt is waar deze moet zijn schakel ik de enable pin terug hoog.

Hierna stel ik de x-en y-coördinaat gelijk aan de x-en y-coördinaat waar de pen naartoe moet bewegen.

Bij dit deel van het programma had ik het probleem dat de y-as zijn stappen sneller voltooid had dan de x-as. Naar later toe om een lijn te tekenen is dit een probleem. Dit komt doordat de tijd tussen de pulsen van de x-as te kort is voor het programma. Om dit op te lossen heb ik de stapverkleining van de x-as terug vergroot naar 1/4 in de plaats van 1/16.

#### 9.4 Lijn tekenen

```
void line(int X1, int Y1, int X2, int Y2){  
  
    moveTo(X1, Y1, moveSpeed);  
  
    Zaxis.write(angle);  
  
    delay(500);  
  
    moveTo(X2, Y2, drawSpeed);  
  
    Zaxis.write(0);  
}
```

Programmatie lijn tekenen

Door de moveTo (naar een bepaalde plaats bewegen) functie is een lijn tekenen doormiddel van mijn eigen software vrij eenvoudig. Voor een lijn te tekenen heb ik een functie geschreven. Met deze functie worden 4 variabelen meegegeven. Deze zijn de x-en y-coördinaat van het startpunt en het eindpunt.

De functie begint met de pen te bewegen naar het beginpunt aan de moveSpeed. Dit doe ik doormiddel van de moveTo functie. Daarna stuur ik de servomotor aan om naar het punt te gaan dat de pen het blad zal raken.

Dit doe ik doormidden van: "Zaxis.write(angle);".

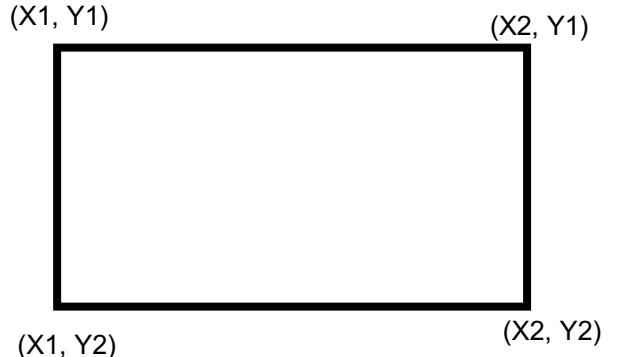
Na deze aan te sturen leg ik het programma een halve seconde stil zodat de servomotor zeker volledig op het blad staat voor de stappemotoren bewegen. Dit doe ik met: "delay(500);".

Hierna beweeg ik de motoren naar het eindpunt van de lijn aan de drawSpeed. Deze snelheid is trager en dus ook nauwkeuriger dan de moveSpeed. Wanneer de stappemotoren gedaan hebben met draaien en de pen zich op zijn eindpunt begint. Stuur ik de servomotor terug aan om naar zijn nulpunt te bewegen.

## 9.5 Rechthoek tekenen

```
void rect(int X1, int Y1, int X2, int Y2){  
    moveTo(X1, Y1, moveSpeed);  
    Zaxis.write(angle);  
    delay(500);  
    moveTo(X2, Y1, drawSpeed);  
    moveTo(X2, Y2, drawSpeed);  
    moveTo(X1, Y2, drawSpeed);  
    moveTo(X1, Y1, drawSpeed);  
    Zaxis.write(0);  
}
```

Programmatie rechthoek tekenen



Afbeelding 51 : voorbeeld van een rechthoek die de penplotter kan tekenen

Net zoals bij het tekenen van een lijn moet er bij de functie van een rechthoek ook vier variabelen meegegeven worden. Hier zijn het de x- en y-coördinaten van twee tegenovergestelde hoekpunten.

In de code begin ik met de pen te bewegen naar het eerste hoekpunt dat wordt meegegeven aan de moveSpeed. Hierna stuur ik de servomotor aan om naar het punt te gaan waar de pen het blad zal raken. Net zoals bij de lijn leg ik het programma weer een halve seconde stil.

Vanaf de pen op het blad staat tekent deze de rechthoek in deze volgorde: (X1, Y1) → (X2, Y1) → (X2, Y2) → (X1, Y2) → (X1, Y1). Nadat deze bewegingen gebeurt zijn stuur ik de servomotor terug aan om naar zijn nulpunt te gaan.

## 9.6 Cirkel en cirkel bogen tekenen

### 9.6.1 Cirkel

```
void circle(int midX, int midY, int R){  
  
    moveTo((midX+R), midY, moveSpeed);  
  
    Zaxis.write(angle);  
  
    delay(500);  
  
    for(int i = 1; i <= 100; i++){  
  
        float cX = midX + (cos((TWO_PI*i)/100)*R);  
        float cY = midY + (sin((TWO_PI*i)/100)*R);  
  
        moveTo(cX, cY, drawSpeed);  
    }  
    Zaxis.write(0);  
}
```

Programmatie cirkel tekenen

De code om een cirkel te tekenen heb ik ook binnen een functie geschreven. In deze functie worden drie variabelen meegegeven. Twee zijn voor de coördinaten van het middelpunt en de laatste voor de straal.

De functie begint met de pen te bewegen naar het startpunt van de cirkel. Dit startpunt heeft voor zijn x-coördinaat de som van de x-coördinaat van het middelpunt en de straal. Voor de y-coördinaat is dit de y-coördinaat van het middelpunt. Dit gebeurt aan de moveSpeed.

Vanaf de pen op zijn startpunt is beweegt de servomotor naar het punt dat deze het blad raakt. Hierbij wacht het programma net zoals de vorige figuren een halve seconde zodat de pen zeker het blad raakt voor de pen begint te bewegen.

Om een cirkel te tekenen deel ik deze op in 100 kleine deel intervallen. De software tekent eigenlijk een honderdhoek. Die kleine deel intervallen gaan bepalen hoe nauwkeurig de cirkel is. Deze verdelen doe ik in een for lus. Een for lus zal een bepaald aantal keer herhaald worden. Dit wordt bijgehouden door een variabele (in mijn geval i) die in de lus zelf gebruikt kan worden. In deze lus geeft men dan mee hoe groot i maximaal mag worden met een voorwaarde (bij mij i <= 100).

Na deze voorwaarden geeft men nog mee hoe groot de stap is die de variabele telkens groter of kleiner moet worden (bij mij i++ dit wil zeggen dat bij i elke keer één zal bij opgeteld worden). Deze lus zal dus 100 keer herhaald worden. Van i=1 tot i=100.

In deze lus bereken ik telkens de x-en y-coördinaat van het volgende punt op de cirkel aan de hand van de variabele i. Dit gebeurt voor de x-coördinaat met de lijn: "float cX = midX + (cos((TWO\_PI\*i)/100)\*R);". In deze lijn zal bij de x-coördinaat van het middelpunt de cosinus van  $2\pi$  maal i gedeeld door 100 maal de straal bij opgeteld worden. Voor de coördinaat is dit gelijkaardig enkel met de sinus en de y-coördinaat van het middelpunt.

Wanneer deze twee waarden berekend zijn voor een punt zal de pen naar dit punt bewegen met de drawSpeed.

Nadat de penplotter alle 100 deelintervallen getekend heeft en de cirkel dus getekend is zal deze de servomotor terug naar zijn nulpunt bewegen.

## 9.6.2 Cirkel bogen

```
void circleArc(int midX, int midY, int R, float Sangle, float Eangle){  
    moveTo((midX+(R*cos(Sangle*PI))), (midY+(R*sin(Sangle*PI))), moveSpeed);  
    Zaxis.write(angle);  
    delay(500);  
    int forSteps = abs((100*(Eangle-Sangle))/2);  
    for(int i = 1; i <= forSteps; i++){  
        float cX = midX + (cos((Sangle*PI)+((TWO_PI*i)/100))*R);  
        float cY = midY + (sin((Sangle*PI)+((TWO_PI*i)/100))*R);  
        moveTo(cX,cY,drawSpeed);  
    }  
    Zaxis.write(0);  
}
```

Programmatie cirkelbogen

De code om een cirkelboog te tekenen heb ook in een functie geschreven. Om deze functie uit te voeren moeten er vijf variabelen worden meegegeven. Twee variabelen zullen voor de coördinaten van het middelpunt zijn. Een andere voor de straal en de laatste twee voor de begin- en eindhoek. Deze hoeken worden uitgedrukt in het veelvoud van  $\pi$  dat bij de hoek in radianen zal staan. Stel dat men wil beginnen op  $90^\circ$  naar  $270^\circ$  zal de beginhoek 0,5 zijn en de eindhoek 1,5. De beginhoek zal de eerste hoek zijn die men met de klok mee tegen zal komen. De tweede zal dan de eindhoek zijn. Dus de  $90^\circ$  uit het vorige voorbeeld zou  $270^\circ$  op een goniometrische cirkel zijn. Dit is omdat mijn referentie punt de linkerbovenhoek van het blad is en dit punt de coördinaten (0, 0) heeft.

De functie zelf begint met de pen te bewegen naar het beginpunt. De x-coördinaat hiervoor zal de som zijn van de x-coördinaat van het middelpunt en de cosinus van de starthoek vermenigvuldigd met de straal. Voor de y-coördinaat is dit gelijkaardig enkel met de y-coördinaat van het middelpunt en de sinus van de beginhoek. Dit zal gebeuren aan de moveSpeed.

Wanneer de pen het beginpunt bereikt zal de servomotor naar het punt bewegen dat de pen het blad raakt. Hierbij wacht ik weer een halve seconde zodat de pen zeker het blad raakt voor de stappenmotoren beginnen te draaien.

Hierbij zou ik net zoals bij de cirkel een volledige cirkel opdelen in 100 kleine deel intervallen. Maar hierbij moeten al deze 100 deel intervallen niet getekend worden. Het berekenen van het aantal deel intervallen gebeurt door volgende lijn:

"int forSteps = abs((100\*(Eangle-Sangle))/2);".

Deze lijn berekent het verschil tussen de eindhoek en de beginhoek en vermenigvuldigt dit getal met de 100 kleine deel intervallen. Dat verschil zal echter liggen tussen 0 en 2 ( $2\pi$  rad =  $360^\circ$ ) dus moet ik de vermenigvuldiging nog delen door 2. Hier nemen ik de absolute waarde van want het verschil in hoeken kan ook negatief zijn. Bijvoorbeeld wanneer men een hoek van  $270^\circ$  tot  $90^\circ$  zou willen tekenen. Hierbij zal de beginhoek 1,5 zijn en de eindhoek 0,5. Het verschil tussen de twee zal dus uitkomen op -1.

Nadat deze berekening gebeurd is zal net zoals bij de cirkel de for lus beginnen lopen. Hierbij zal dit gebeuren van 1 tot het aantal deel intervallen dat de penplotter moet tekenen. De lijn om de volgende x-coördinaat te berekenen wordt dan: "float eX = midX + (cos((Sangle\*PI)+((TWO\_PI\*i)/100))\*R);".

Hierbij moet er nog rekening gehouden worden met de beginhoek. Deze wordt binnen de cosinus opgeteld bij de hoek die men bij de cirkel had.

Namelijk de hoek  $\frac{2\pi \cdot i}{100}$ . De volledige cosinus zal dan vermenigvuldigd worden met de straal en opgeteld bij de x-coördinaat van het middelpunt. Voor de volgende y-coördinaat is dit gelijkaardig enkel met de sinus en de y-coördinaat van het middelpunt. Wanneer de coördinaten van het volgende punt brekend zijn zal de pen naar dat punt bewegen aan de drawSpeed. Wanneer alle deelintervallen getekend zijn en de cirkelboog getekend is zal de servomotor terug bewegen naar zijn nulpunt.

## 9.7 Ellips tekenen

```
void ellips(int midX, int midY, int distX, int distY){

    moveTo((midX+(distX/2)), midY , moveSpeed);

    Zaxis.write(angle);

    delay(500);

    for(int i = 1; i <= 100; i++){

        float eX = midX + ((cos((TWO_PI*i)/100)*distX)/2);
        float eY = midY + ((sin((TWO_PI*i)/100)*distY)/2);

        moveTo(eX,eY,drawSpeed);
    }
    Zaxis.write(0);
}
```

Programmatie ellips

Net zoals bij de vorige figuren heb ik de code voor een ellips te tekenen in een functie geschreven. Om deze functie uit te voeren moeten er vier variabelen worden meegegeven. Twee van deze variabelen zijn voor de coördinaat van het middelpunt. De andere twee zijn voor de breedte en de hoogte van de ellips.

Deze functie begint met het bewegen naar het beginpunt. Bij de ellips zal dit punt zijn x-coördinaat gelijk zijn aan de som van de x-coördinaat van het middelpunt en de helft van de breedte van de ellips. De y-coördinaat zal gelijk zijn aan de y-coördinaat van het middelpunt. Dit zal gebeuren aan de moveSpeed. Hierna zal de servomotor draaien tot het punt dat de pen het blad raakt.

Net zoals bij de cirkel en de cirkelboog deelt de software de ellips in 100 kleine deel intervallen. De coördinaten van de punten van deze deelintervallen berekenen en te tekenen doe ik weer door gebruik te maken van een for lus. Het bereken van de x-coördinaat van zo een deel interval gebeurt met volgende lijn:

"float eX = midX + ((cos((TWO\_PI\*i)/100)\*distX)/2);".

Deze is gelijkaardig aan die van cirkel enkel zal de cosinus niet vermenigvuldigd worden met de straal maar met de breedte van de ellips en gedeeld worden door twee. Voor de y-coördinaat is dit hetzelfde maar dan met de sinus en de hoogte van de ellips.

## 9.8 User interface

Mijn penplotter werkt doormiddel van commando's. Deze commando's worden gestuurd door middel van de seriële monitor in de Arduino IDE. Hiermee kan er data gestuurd en ontvangen worden van de computer naar de Arduino.

```
void loop() {  
    Instructions();  
  
    while(Serial.available() == 0);  
  
    char com = Serial.read();  
  
    if(com == 'H'){  
        Home();  
    }  
  
    if(com == 'L'){  
        int data[4];  
  
        for(int i=0; i < 4; i++){  
            data[i] = Serial.parseInt();  
  
            if(i == 0){  
                Serial.print("punt 1: ");  
            }  
  
            if(i == 2){  
                Serial.println();  
                Serial.print("punt 2: ");  
            }  
  
            Serial.print(data[i]);  
            Serial.print(" ");  
        }  
        Serial.println();  
  
        line(data[0], data[1], data[2], data[3]);  
    }  
}
```

Programmatie user interface

Deze code is geschreven in de “void loop(){}”. Code dat hierin geschreven is zal altijd herhaald worden zolang de Arduino spanning heeft.

In deze lus begin ik met de zelfgeschreven functie “Instructions();” hiermee print ik een lijst met de uitleg van elke functie met zijn commando. In de seriële monitor zal dan het volgend komen.

```
Stuur 'H' om de machine terug naar zijn nulpunt te brengen bij een fout
Lijn
Stuur 'L' gevuld door de x-en y-coördinaten van het begin en eindpunt
L X1 Y1 X2 Y2

Rechthoek
Stuur 'R' gevuld door de x-en y-coördinaten van twee tegenovergestelde hoekpunten
R X1 Y1 X2 Y2

Cirkel
Stuur 'C' gevuld door de x-en y-coördinaten van het middelpunt gevuld door de straal
C X Y R

Ellips
Stuur 'E' gevuld door de x-en y-coördinaten van het middelpunt gevuld door de breedte en de hoogte
E X Y b h

Cirkelboog
Stuur 'B' gevuld door de x-en y-coördinaten van het middelpunt gevuld door de straal gevuld door de begin-en eindhoek
De beginhoek is de eerste hoek die men met de klok mee tegenkomt
B X Y R BH EH

Stuur '+' om de servohoek te vergroten met 5°
Stuur '-' om de servohoek te verkleinen met 5°
```

Afbeelding 52 : Instructies seriële monitor

Nadat de instructies gegeven zijn zal het programma blokkeren tot dat de Arduino een input heeft gekregen van de seriële monitor. Dit gebeurt met de lijn: “while(Serial.available() == 0);”.

Wanneer er wel iets in de seriële monitor zal gestuurd worden leest de Arduino het eerste karakter in. Dit gebeurt met de lijn: “char com = Serial.read();”. Dit zal bepalen welke functie uitgevoerd zal worden. Het bepalen van de functie gebeurt door voor elke functie een if lus te schrijven waarin het ingestuurde karakter vergeleken wordt met het karakter van de bepaalde functie. Voor een lijn zal dit gecontroleerd worden met de volgende lijn code: “if(com == 'L'){}”.

Hierna zal de data voor de bepaalde functie ingelezen worden. Uitzonderd bij de homingsequence en de hoek van de servomotor veranderen (commando's + en -). Hiervoor maak ik in de bepaalde if lus van de functie een array aan. Een array is een lijst met variabelen met een bepaalde lengte. Het inlezen zelf gebeurt met een for lus. In deze lus leest het programma stap per stap de data in voor die bepaalde functie.

Wanneer er data voor een functie is ingelezen print ik deze met extra uitleg in de seriële monitor.

Nadat alle data zijn ingelezen geef ik deze mee in de bepaalde functie. De figuur die ingegeven was zal dan getekend worden. Hierna zal het programma de instructies terug printen en wachten op het volgende commando.

## 10 Besluit

Het is me gelukt om een CNC penplotter van nul op te bouwen en deze voor basisfiguren zelf te programmeren.

Uit deze opdracht heb ik geleerd dat een goede planning zeer belangrijk is. Bij mij was dit eerst mijn penplotter bouwen. Moesten er in de bouw moeilijkheden zijn opgetreden had ik nog genoeg tijd om deze op te lossen. Daarna heb ik de bouw en het elektrisch ontwerp beschreven. Hierna had ik nog genoeg tijd om mijn eigen software te schrijven en deze te beschrijven.

Het bouwen van mijn penplotter ging vlot desondanks ik enkele onderdelen heb moeten herontwerpen.

Bij het programmeren had ik al snel een basisprogramma maar dit debuggen heeft wel enkele weken geduurd.

Het schrijven van mijn geïntegreerde proef was voor mij niet gemakkelijk. Het moeilijke eraan was dat iemand zonder een technische achtergrond mijn tekst moet begrijpen.

Uiteindelijk ben ik zeer tevreden van mijn GIP. Ik heb er zelf uit kunnen leren hoe ik technische informatie verstaanbaar kan overbrengen.

In volgend filmpje kan je mijn penplotter met mijn eigen software aan het werk zien. In het voorbeeld tekent deze een lijn en een cirkel.

<https://youtu.be/f9kwfoSGJ3w>