

Ward Debecker

Nick Vlaeyen

PXL Tech

2EAI groep B

22 mei 2015

Mechatronica project: Eshapeoko

eshapeoko programmeren met Screvle

Inhoud

[1. Inleiding 2](#_Toc420010760)

[2. Uitwerking 3](#_Toc420010761)

[3. Uitleg project 4](#_Toc420010762)

[4. Kennismaking programmeeromgeving Eclipse 4](#_Toc420010763)

[5. Kennismaking programmeeromgeving Lua 5](#_Toc420010764)

[6. Analyseren Eshapeoko-tafel 6](#_Toc420010765)

[7. Basis motoraansturing 8](#_Toc420010768)

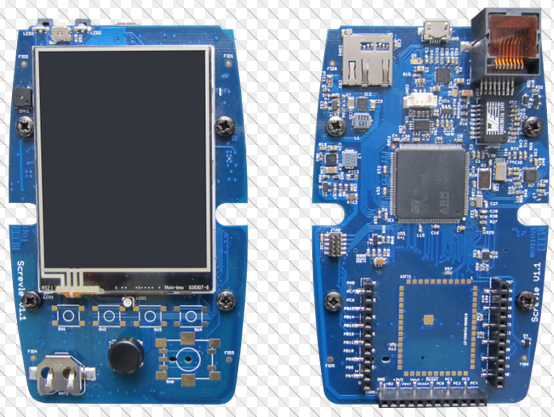
[8. Verdere uitwerking project 8](#_Toc420010769)

[9. Problemen 9](#_Toc420010770)

[10. Bronnen 10](#_Toc420010771)

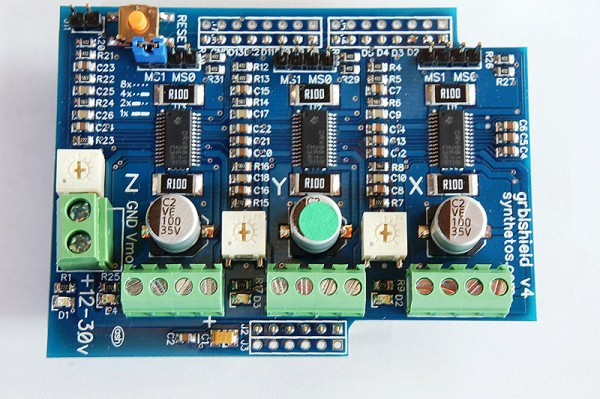
# Inleiding

De bedoeling van dit project is het programmeren van een CNC-tafel, met XYZ positie, te programmeren met Screvle. Het is een closed-source programmeeromgeving, ontworpen door 1 persoon, waardoor er weinig informatie van te vinden is, ten opzichte van open-source programmeeromgevingen zoals Arduino. Screvle is een multifunctioneel meet- en controle apparaat dat verbindt met sensoren en actuatoren en een makkelijke verbinding voorziet om zo je machines, apparaten en netwerkinfrastructuren te programmeren. Het programma dat gebruikt wordt om alles mee te programmeren heet Eclipse. Om het te programmeren kan men ook gebruik maken van de Lua-omgeving.



Figuur Screvle

De Eshapeoko CNC-machine was al helemaal gemonteerd door een projectgroep van vorig academiejaar, hierdoor moesten we het enkel programmeren. Om onze machine uiteindelijk te laten werken moeten we natuurlijk ook de motoren aansturen. Voor de stappenmotorsturing is er een gShield, of grblShield, voorzien. Dit is een shield speciaal gemaakt voor Arduino, maar wij doen dit dus met Screvle in plaats van met Arduino.

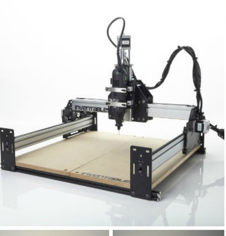


Figuur GrblShield

# Uitwerking

Om dit project tot een goed einde te brengen moesten we eerst een plan van aanpak opstellen om het project tot een goed einde te brengen. Omdat dit een vrij zwaar project was met een zeer onbekend closed-source programmeeromgeving, hebben we hier veel tijd voor ingepland. Onze plan van aanpak is zo opgesteld dat we na vele uren werken, het project tot een goed resultaat probeerden te brengen. Maar na vele stribbelingen met het programmeerplatform, zijn we op veel obstakels geland. Ook na infomomenten met onze contactpersoon Ronald Vanschoren.

1. Uitleg project door Ronald Vanschoren
2. Kennismaking programmeeromgeving Eclipse
3. Kennismaking programmeeromgeving Lua
4. Analyseren Eshapeoko tafel
   1. Analyseren grblShield
   2. Analyseren project vorig academiejaar
5. Basis motoraansturing met Screvle
6. Verdere uitwerking project
   1. Site maken voor uploaden g-code
   2. Screvle als stand alone server laten draaien



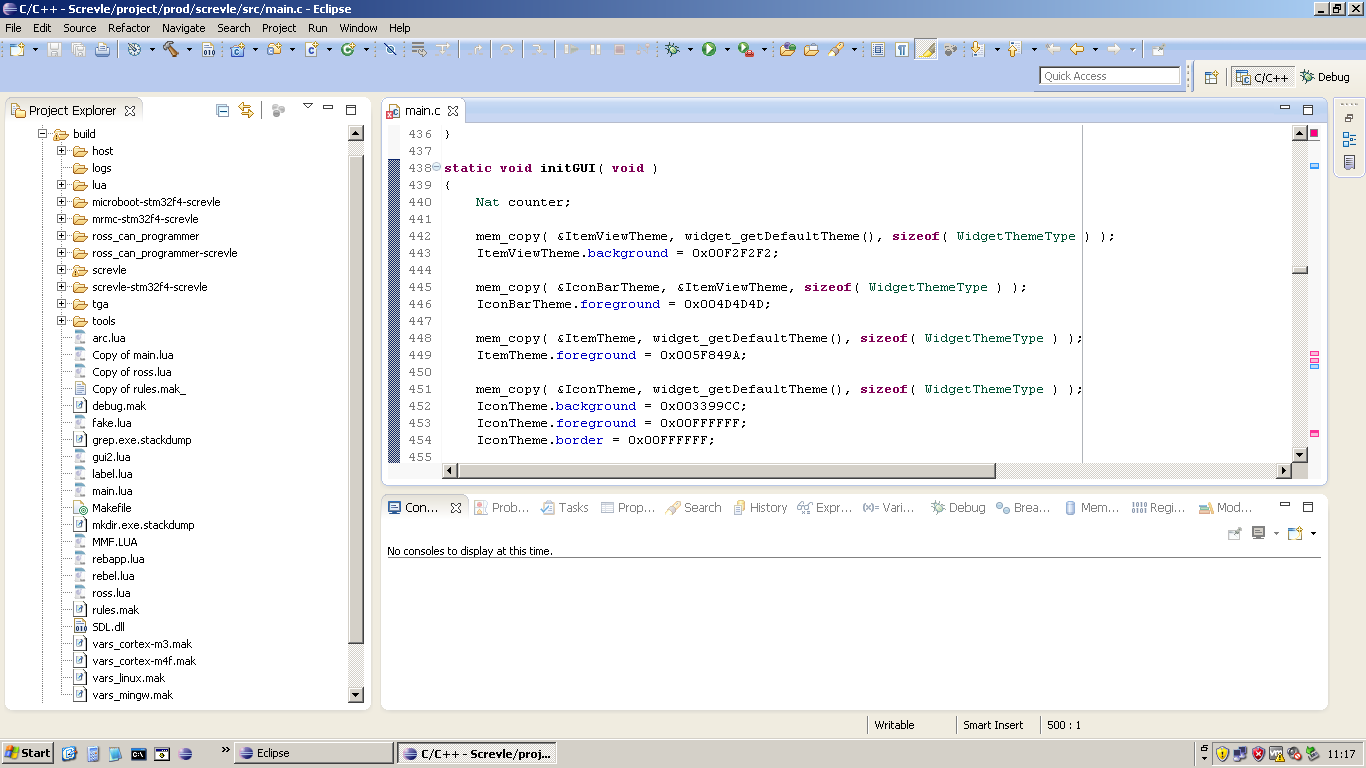
Figuur Eshapeoko CNC Machine

# Uitleg project

Allereerst kregen we uitleg over de programmeeromgeving waarin we moesten programmeren, dit was voor ons beiden een leerrijk moment omdat we Screvle en Eclipse zelf nog niet kenden. Om zelf te kunnen programmeren met Screvle moesten we de libraries van onze contactpersoon zelf hebben, en dus hebben we zijn VMWare-omgeving meegekregen. Ook hebben we een Screvle mee om het uiteindelijk te programmeren, maar omdat deze te duur was kregen we er maar 1.

De uiteindelijke bedoeling van dit project was het maken van een website waarop we onze g-code kunnen uploaden, g-code is de code nodig om met de grblShield de motoren aan te sturen. De screvle zou dan zelf als webserver fungeren waardoor het een volledig stand alone systeem is. Wanneer er dan op deze website g-code ge-upload werd kreeg u dat te zien op het touchscreen van de Screvle, waar u tussen de verschillende bestanden kan kiezen en selecteren welke u wilt gebruiken. De Screvle op zijn beurt zou dan geconnecteerd zijn met het grblShield, waardoor de g-code naar het grblShield gestuurd kan worden. Eens in het grblShield aangekomen zou deze de stappenmotoren aansturen, en het doorgestuurde bestand laten printen, als we er een stift op zouden plaatsen, of laten uitfrezen.

# Kennismaking programmeeromgeving Eclipse



Figuur Programmeeromgeving Eclipse

Onze eerste kennismaking met Eclipse was bij onze contactpersoon, maar eens dat wij de VMWare op onze pc’s klaarzette, kregen we niet dezelfde layout als Ronald. Ook waren er meerdere folders, die we nodig hadden om te programmeren, verdwenen. Na meerdere mails naar onze contactpersoon, snapte hij uiteindelijk wat ons probleem was en gaf ons zo toegang tot zijn BitBucket-account, zodat we rechtstreeks aan zijn code konden. Hierdoor hadden we een grotere achterstand dan we eerst dachten.

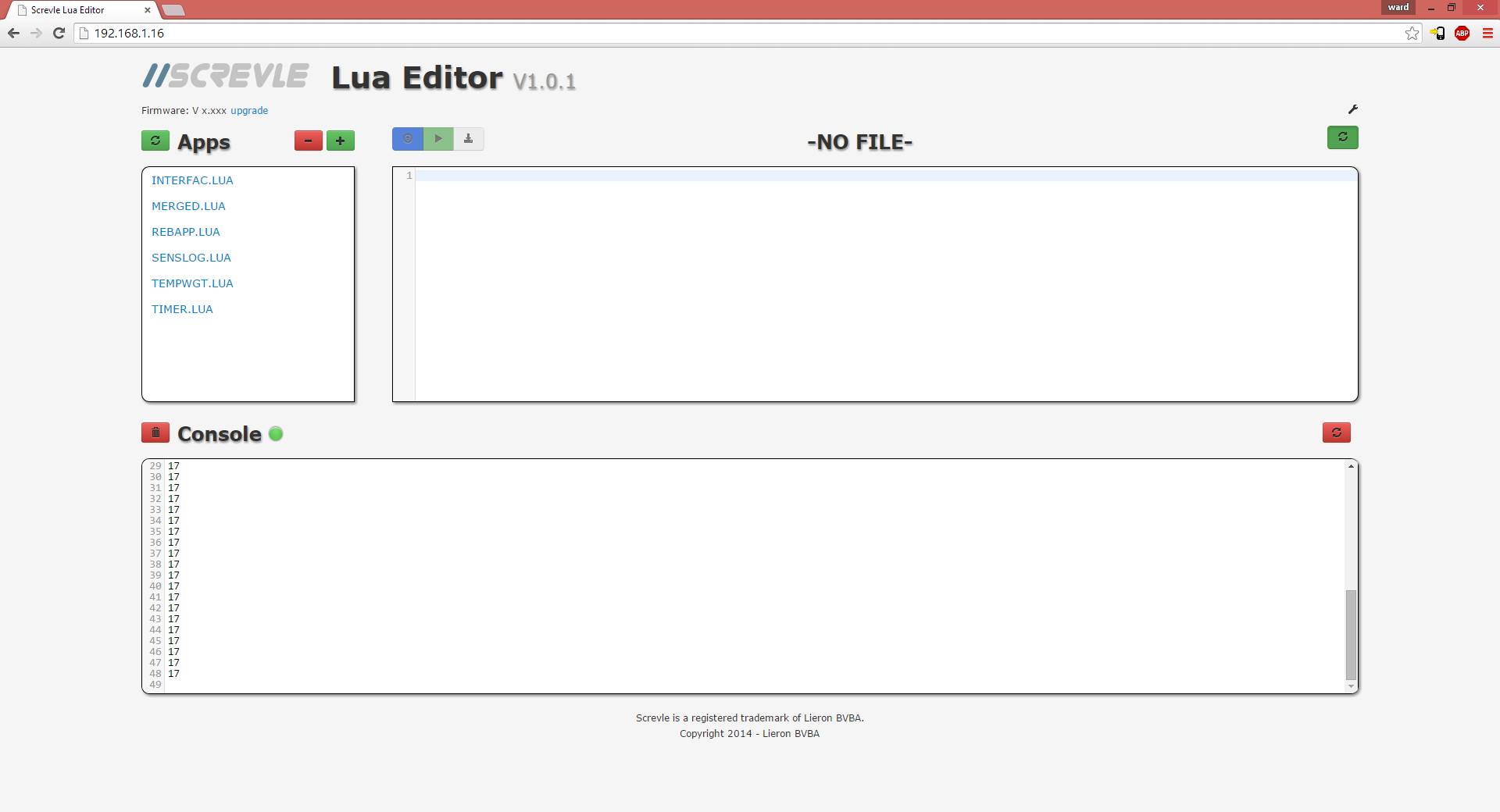
Eens we aan de code konden we beginnen programmeren met Eclipse, dit was voor ons een grote uitdaging omdat we niet wisten hoe te programmeren in deze omgeving. Ook moesten zoeken waar we onze code moesten schrijven, dit was ook al een uitdaging in de vele files en folders in het screvle project. Toen we dit wisten konden we beginnen aan onze code te schrijven, echter leek het niet zo makkelijk als eerst verwacht, omdat er weinig documentatie van Screvle op internet te vinden was moesten we zelf een beetje proberen te zoeken hoe we onze outputs moesten sturen.

Maar om te kunnen programmeren met Screvle, moesten we verbinding maken met Olimex ARM-USB-TINY-H. Hier kwamen we nog een groot probleem tegen, geen van beide computers die we gebruikten konden niet verbinden omdat openOCD niet voldoende werkt op windows. Dit hebben we dan proberen op te lossen door een pre-build versie van openOCD te vinden, en deze .exe op te roepen met een batch file, maar ook dit werkte niet helemaal.

# Kennismaking programmeeromgeving Lua

Om met Lua te programmeren moeten we de ethernet-poort van de Screvle verbinden met onze laptop, en dan naar het ip-adres van de Screvle te verbinden. Eens dit gebeurd was konden we programmeren in de Lua omgeving en zo de interface op het touchscreen helemaal aan te passen.

Lua op zich was tof om mee te programmeren, hier hebben we niet veel problemen mee gekend. Maar omdat dit niet echt een prioriteit was van dit project hebben we eerst gezorgd dat we Eclipse werkend zouden krijgen, aangezien lua veel overhead heeft en hierdoor ook traag werkt.



Figuur Programmeeromgeving Lua

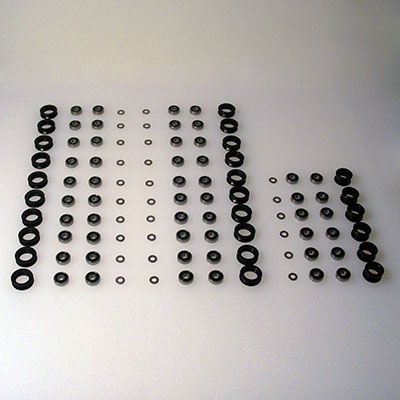
# Analyseren Eshapeoko-tafel

Al het voorgaande is van belang om de CNC machine aan de praat te krijgen, de Eshapeoko. Deze was al gemonteerd voor ons, maar laat ons toch even alle onderdelen apart toespitsen. In onderstaande foto (figuur 6) zien we de elektrische onderdelen. We zien onder andere een voedingskabel, 4 motoren, connectoren, arduino en het grblShield.



Figuur Elektronische componenten

In onderstaande 2 figuren ziet u wieltjes die gemonteerd moeten worden op de assen om het rijtuig waar alles op gemonteerd worden, te kunnen verplaatsen.



Figuur links: wieltjes en tussenwieltjes rechts: rijtuigen

# Y Axis ImageGantry Image

Figuur Railings en rijtuigen

In de figuur hierboven ziet u de railings om de assen waarop alles beweegt te monteren, zo krijgen we de x- en y-as. De foto hieronder toont dan weer alles aan wat je nodig hebt voor de z-as.

# z-axis exploded

Figuur Benodigdheden z-as

Als laatste hebben we natuurlijk nog het werkblad, deze wordt onderaan gemonteerd zodat al het vuil op deze plaat vallen. Ook worden alle rails hierop bevestigd zodat alles 1 mooi geheel word.



Figuur Onderframe

Dit alles resulteert in een volledig werkende CNC-machine. Zoals eerder vermeld werd dit alles al gemonteerd voor ons, maar stond wel voor ons ter beschikking.

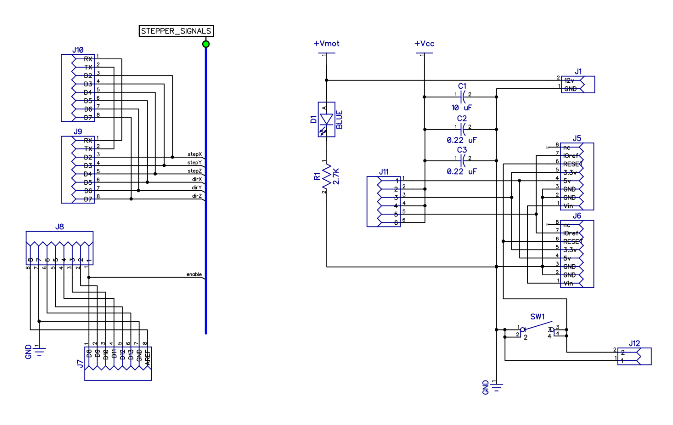


Figuur CNC-machine

# Basis motoraansturing

Wanneer we op de website kijken van shapeoko, nl: <http://docs.shapeoko.com/>, zien we ook dat hier al een code online staat om de machine te testen, een zogenaamde “Hello world”-code. Dit is een testcode die door de fabrikanten ter beschikking wordt gesteld om te weten of de CNC-machine werkt. Hier wordt er ook van een programma voorzien om de tafel naar een gewenste positie te verplaatsen.

In onderstaande figuur zien we hoe we ons grblShield moeten aansturen zodat de motoren zullen bewegen. Zo zien we dat we op J10 en J9 een signaal moeten sturen, hierdoor zal stepX, stepY, stepZ en dirX, dirY, dirZ aangestuurd worden afhankelijk van waar de motor heen moet bewegen.



Figuur Aansturen motor

# Verdere uitwerking project

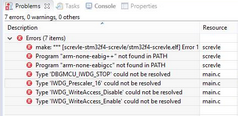
Tot dit deel van het project zijn we niet geraakt omdat we vele tegenslagen zijn tegengekomen onderweg naar het eindproduct. Hetgeen we hier wouden realiseren was dat we een website zouden maken waarin we onze code zoude kunnen laden. Deze code zou dan verzonden kunnen worden naar onze Screvle die daar op zijn beurt zou opgeslagen kunnen worden op een sd-kaart.

Maar om deze code naar de Screvle te kunnen versturen moesten de Screvle als een stand alone server fungeren zodat we alles konden versturen vanuit een website. Als we dan onze Screvle zouden aan zetten, zouden al tekeningen, of beter gezegd g-code, hierop opgeslagen staan. Hierdoor zouden we kunnen kiezen welke we zouden doorsturen naar het grblShield, en zo de machine in gang zetten om het traject af te lopen.

Spijtig genoeg zijn we hier niet aan kunnen starten. Omdat we van mening waren dat we eerst onze motoren moesten kunnen aansturen, vooraleer we de verdere uitwerking van dit deel zouden doen.

# Problemen

Zoals eerder vermeld zijn we vele problemen tegengekomen onderweg waardoor dit project niet is afgeraakt. Zo hebben we toen we de eerste keren wouden programmeren, enkele errors tegengekomen, in onderstaande figuur enkele voorbeelderrors. Dit waren errors in de code die recht van onze contactpersoon kwam.



Figuur Voorbeelderrors

Wanneer we deze errors dan wouden opzoeken, vonden we geen documentatie. Dit omdat het een closed-source programmeeromgeving is. Ook onze contactpersoon wist niet juist wat de problemen waren met de errors die we kregen.

Hierdoor hebben we uiteindelijk weer contact gehad met onze contactpersoon, deze heeft ons dat toegang gegeven tot zijn BitBucket account, waar de code opstond die we konden gebruiken. Toen we hiermee wouden starten om te programmeren kwamen we weer in de problemen als we onze code wouden compilen. Hier leek de fout te zijn dat we met Windows werkten in plaats van Linux. Als oplossing hebben we dan bedacht om de main.c file met FTP naar een Linux server te sturen. Daarna zouden we dan van daaruit kunnen compilen en het gecompileerde bestand terug te sturen en dit te programmeren.

Dit leek op het eerste moment een goede oplossing want onze code kon compileren, maar wanneer we dan onze teruggestuurde code wouden programmeren lukte dit niet omdat de openOCD voor te programmeren nog niet werkte. OpenOCD werkt normaal vanuit Eclipse, maar dit werkte niet in Eclipse. Voor dit hebben we nog geen oplossing gevonden.

Als laatste oplossing wouden we dan een stand alone openOCD maken/programmeren, maar wanneer we nu de code gingen compileren kwam dit in conflict met de makefile.

Ook was het zeer moeilijk om te vinden hoe we het grblshield per bit moesten aansturen aangezien dit een shield specifiek voor de arduino gemaakt is, en er natuurlijk geen header files bestaan voor de screvle.

# Bronnen

<http://www.shapeoko.com/wiki/index.php/Main_Page>

Dit is de wikipedia van Shapeoko

<https://github.com/Glenn-Kerselaers/ESHAPEOKO_1-MYRIO>

Dit is de GitHub link van degene die vorig jaar dit project had met MyRio.

<https://screvle.atlassian.net/wiki/display/SCREV/Screvle+Manual+and+API+Documentation>

<https://github.com/synthetos/grblShield/wiki>

<https://github.com/synthetos/grblShield/tree/master/hardware/gshield_v5_schematic>