Vrije Universiteit Brussel
2e kandidatuur Burgerlijk Ingenieur
Werkcollege werktuigkunde – elektrotechniek

Claire Giannone
Lieselotte Monteyne
Kenny Goossens
Jan Verstreken
Thomas De Schampheleire

Inhoudsopgave

Deel 1: Technisch verslag

1.1 Opgave en doelstellingen

Opgave

De opgave is om een automatische machine te maken. Er is gekozen voor het bouwen van een plotter: een soort printer voor geometrische figuren (lijnstukken, rechthoeken, cirkels, ellipsen, …). Hierbij waren er verschillende mogelijkheden:

- Zullen de figuren met losse punten of met volle lijnen getekend worden?
- Zal het blad vastliggen en beweegt er een as over, of beweegt het blad zoals in een echte printer?
- Als het blad blijft vastliggen: zullen er dan twee beweegbare assen gebruikt worden met daartussen de stift, of slechts één beweegbare as waarop een wagentje met de stift beweegt?
- Welke geometrische figuren zullen geprogrammeerd worden?

Er is uiteindelijk besloten om een toestel te maken dat volle lijnen tekent op een vast blad, en waarbij er slechts één beweegbare as is. Hiertoe dienden volgende onderdelen gemaakt te worden:

- Een mechanische constructie waarmee een stift naar elk punt van het blad gebracht kan worden. Deze moet bestaan uit een kader, een beweegbare as en een wagentje.
- Een mechanisme dat een stift vasthoudt en deze op en neer laat bewegen.
- Een softwareprogramma dat de motoren naar de juiste posities laat bewegen en zo verschillende basisfiguren kan tekenen.

Enkele doelstellingen

- Volgende basisfiguren moeten getekend kunnen worden: lijn, rechthoek, cirkel en cirkelboog
- De plotter moet voldoende stevig gebouwd zijn
- Er moet zo weinig mogelijk materiaal gebruikt worden
- De pen moet vervangbaar zijn
- Het programma moet duidelijk geschreven worden

1.2 Eindresultaat

De gebouwde plotter is in staat om lijnen, rechthoeken, cirkels en ellipsen te tekenen. Er is een resolutie van 0,70 mm, wat voldoende klein is om vloeiende figuren te verkrijgen. Voor de berekening van deze resolutie, zie bladzijde 12)

De gebruiker kiest via de computer wat hij wil tekenen en moet zo ook alle parameters ingeven. Naast het tekenen via de computer, kan je ook “vrij” tekenen met een controller, waarbij je niet beperkt bent tot geometrische figuren. Voor beide opties bestaan aparte programma’s waarvan er slechts een tegelijk kan geladen worden.

Afmetingen en resolutie

In onderstaande tabel staat een overzicht van de verschillende afmetingen van de constructie:

Onderdeel	Lengte (mm)	Breedte (mm)	Hoogte (mm)
Houten grondplaat	440	360	18
Lego kader	465	320	50

Beweegbare as	365	170	165 (60 zonder dradentoren)
Wagentje	105	105	max.180 / min.160
Controller	64	32	70
Tekenbereik	265	180	

Gebruikte materiaal en programmeeromgeving
De plotter is volledig opgebouwd uit lego-blokjes. Het probleem hierbij was dat er maar een zeer beperkt aantal blokjes voorhanden waren. Daarom zijn er een aantal ideeën niet gebruikt kunnen worden in het eindontwerp.
Verder worden er 2 stappenmotoren gebruikt, 1 gelijkstroommotor, 2 drukknoppen en 1 lichtsensor. In de controller pad zitten ook nog eens 3 drukknoppen en 3 lichtsensoren. De constructie is gelijmd op een houten plaat zodat het geheel stevig blijft.

De software is geschreven in BrickC, een aangepaste versie van C++. Het gecompileerde programma mag niet groter zijn dan 32 KB, wegens de beperkte geheugencapaciteit van de PowerBrick, de gebruikte microprocessor-eenheid (zie ook bladzijde 13).

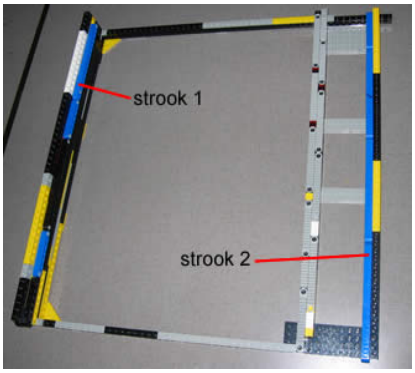
1.3 Uitleg bij de constructie

De plotter bestaat in feite uit vier belangrijke onderdelen: het kader, de beweegbare as, het wagentje en het stiftsysteem.

1.3.1 Het kader

Het kader is de basis van de plotter: hierover rolt de beweegbare as en daarbovenop het wagentje met de stift. Aan één zijde van het kader is er een rail en aan de overstaande zijde een gladde strook. De twee andere zijdes hebben voornamelijk een verstevigende functie.
Het kader is vastgelijmd op een houten plank: dit verzekert stevigheid en stabiliteit en zorgt ervoor dat de plotter gemakkelijk verplaatst kan worden.

Het oorspronkelijke ontwerp (figuur K-1) bestond uit een gladde strook aan één zijde (strook 1), en een rail en een tweede gladde strook (strook 2) aan de andere zijde. De beweegbare as steunde op strook 1 met een glad plaatje en op strook 2 met wielen. Verder was er een rij van drie tandwielen die over de rails rolden. Strook 2 is verdwenen in latere versies aangezien die niet echt nodig bleek: de beweegbare as kreeg voldoende steun door de rails en strook 1.



Figuur K-1


Figuur K-2

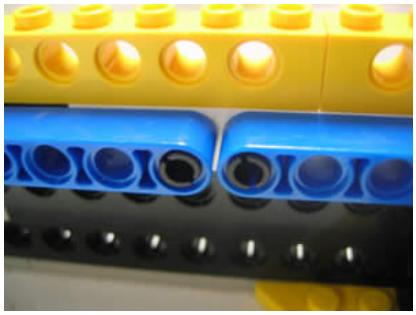
De rail
Omdat het beschikbare bouwmateriaal nogal beperkt was, zijn er doorheen het project verschillende wijzigingen gebeurd in verband met de rails. Er waren een aantal blokjes met “goede” ononderbroken rail en een aantal langere stukjes rail die echter gaatjes aan de uiteinden hadden. Om dit probleem te omzeilen is er een “dubbele” rail gemaakt zodanig dat de gaatjes op de rails afgewisseld werden tussen beide parallelle delen. Om de

volledige lengte van het kader te kunnen bedekken zijn er tussen twee railonderdelen soms gladde verlengstukjes geplaatst. Dit alles is duidelijk te zien op figuur K-2.

Later zijn de goede en slechte rails omgewisseld tussen kader en beweegbare as. De rails op het kader hebben veel meer gewicht te dragen dan de rails op de as (die alleen maar het wagentje draagt). Daarom is het belangrijk dat de beweegbare as over een zo vlot mogelijke rail op de kader kan rollen.

De gladde strook

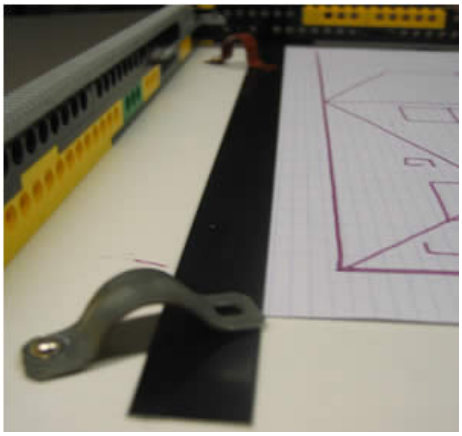
De strook bestaat uit lange afgeronde balken die vastgeklikt worden op de opstaande zijden van het kader. Het vastklikken gebeurt met de zwarte pinnetjes en niet met de grijze omdat de eerste een veel stevigere binding creëren. Bij de overgang tussen twee balken is er weliswaar een klein putje (figuur K-3), maar dit geeft geen problemen voor de beweging van de as.



Figuur K-3

Inklemming van het blad papier

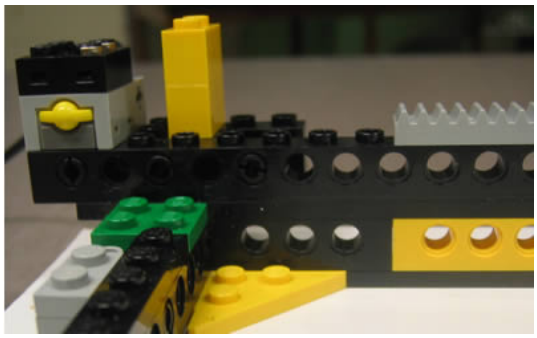
Om het blad stevig op zijn plaats te houden, wordt er gebruik gemaakt van een klemsysteem dat bestaat uit twee metalen latjes en vier klemmen (figuur K-4). Om het blad te vervangen is het voldoende om de klemmen opzij te draaien en een nieuw blad onder de latjes te schuiven. De correcte positie van het blad wordt aangeduid door een stuk karton en de positie van de latjes door twee strepen op de houten plank.



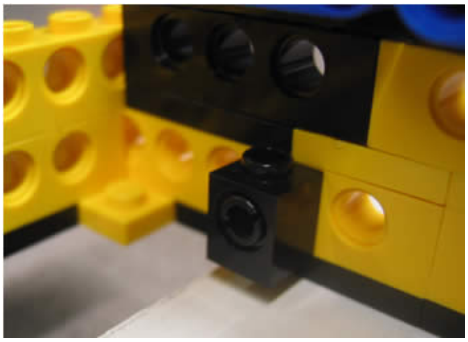
Figuur K-4

Andere elementen op het kader

Bij het opstarten van het programma verplaatst de beweegbare as zich terug naar zijn beginpositie. Hierbij wordt gebruik gemaakt van een drukknop die zich aan het uiteinde van de rail bevindt en die ingedrukt wordt wanneer de beweegbare as aangekomen is. Zo weet het programma wanneer hij de motoren moet stopzetten (figuur K-5).



Figuur K-5



Figuur K-6

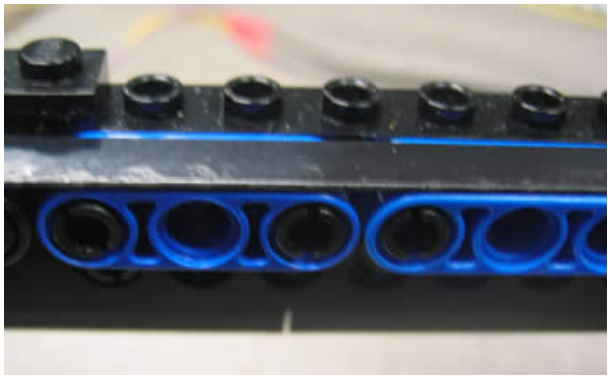
Om te vermijden dat de beweegbare as buiten het kader rolt, wordt in de broncode op geprogrammeerde grenzen gecontroleerd. Toch zijn er ook fysieke barrières aan beide

uiteinden van de as. Wanneer het programma zou falen, dan wordt de as toch nog tegengehouden (zie de gele blokjes op figuur K-5 en het zwarte blokje op figuur K-6)

Om het kader te verstevigen zijn er een aantal bijkomende elementen aangebracht zoals vastgeklikte blokjes en driehoekige plaatjes in de hoekpunten. Om het beschrijfbare gebied van het blad papier te vergroten is het kader in een richting verlengd. Omdat de houten plank echter op maat gezaagd was, steekt het kader aan een zijde nu een tweetal centimeters uit. Hier zijn de twee driehoekige plaatjes dus verwijderd.

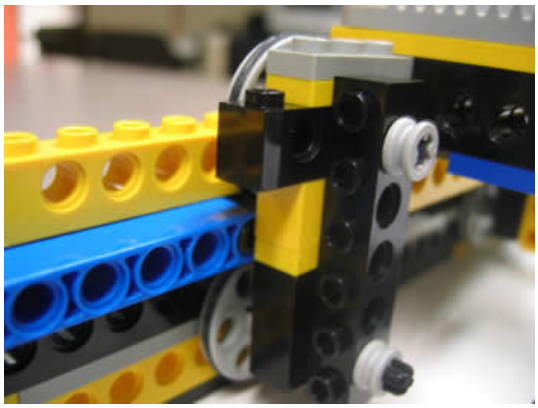
1.3.2 De beweegbare as

De beweegbare as heeft twee evenwijdige “armen” waarover het wagentje rijdt. De ene is een rail voor tandwielen en de tweede een gladde strook.
Zoals al eerder vermeld bestaat de rail niet uit kleine blokjes met doorlopende rail, maar wel uit grotere waartussen delen zonder tandjes te vinden zijn. Doordat deze delen minder hoog zijn en de tandwielen er geen grip op hebben, blokkeert het wagentje op die plaatsen. Daarom is er een tweede rail geplaatst naast de eerste en worden op het wagentje telkens twee tandwielen naast elkaar gemonteerd. Op die manier is er op elk moment zeker een van de twee tandwielen in contact met de rail. Een moeilijkheid bij het plaatsen van deze tweede rij was dat de tandjes van de twee rails zijdelings in elkaars verlengde moesten liggen. Daarom zijn er tussen twee railblokjes soms andere gladde blokjes geplaatst (figuur K-2).
Langs de andere arm van de beweegbare as is er een gladde strook met daarop een stalen lat (figuur A-1). De strook zal ondersteuning bieden aan het wagentje en de stalen lat dient enkel om het wagentje rechter te zetten.



Figuur A-1

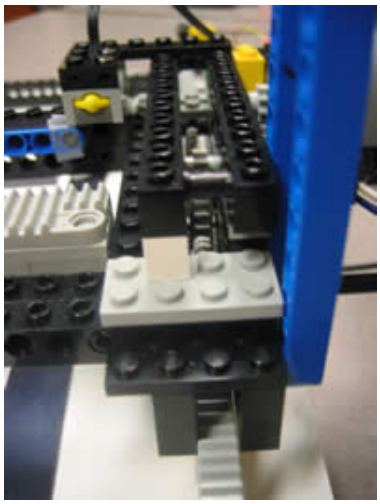
Om doorbuiging van beide armen te beletten bestaan deze uit lagen van platte en hoge legoblokjes. Door de platte blokjes overlappend met de hoge te plaatsen zodat de scheidingen niet overeenkomen, is de constructie veel steviger.



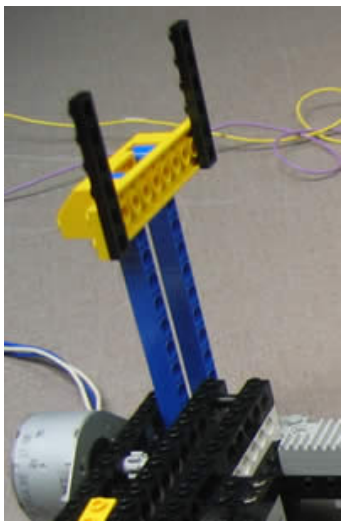
Figuur A-2

Aan het ene uiteinde van de as bevinden zich twee rijen van twee wielen. De bovenste twee wielen rollen over een gladde strook terwijl de onderste twee dienen om de as in te klemmen (figuur A-2). Er worden telkens twee wielen gebruikt omwille van het evenwicht.

Aan het andere uiteinde wordt de as aangedreven door een stappenmotor. Op deze stappenmotor is een klein tandwieltje met 8 tanden geplaatst dat een tandwiel met 24 tanden aandrijft. Op dezelfde as als dit groter tandwiel staat er een tweede kleine tandwiel (met 8 tanden) dat weer een tandwiel met 24 tanden doet draaien. Dit tandwiel rust samen met twee andere “losse” tandwielen op de rail van het kader. Doordat er twee tandwielen met 8 en 24 tanden op dezelfde as staan, ontstaat er zo een verkleining van de resolutie met een factor drie (24 gedeeld door 8). Dit geeft een opmerkelijke prestatie-verbetering.



Figuur A-3



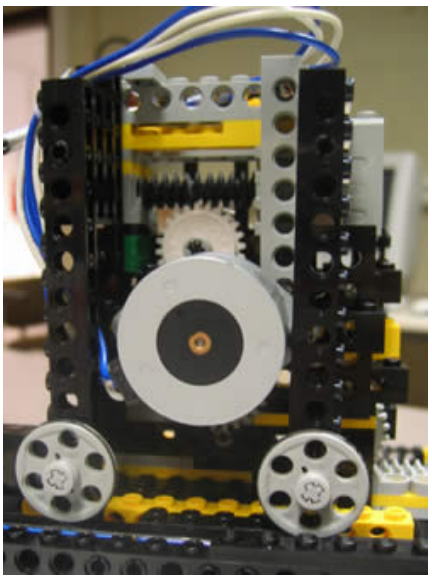
Figuur A-4

Langs weerszijden van de tandwielen “hangen” er twee muurtjes die ervoor zorgen dat de as niet dwars af de rail geschoven kan worden (figuur A-3).

Bovenop de constructie rond de motor is er ook nog een H-vormige toren gebouwd om de bedrading weg van de rails en de tandwielen te houden (figuur A-4)

1.3.3 Het wagentje

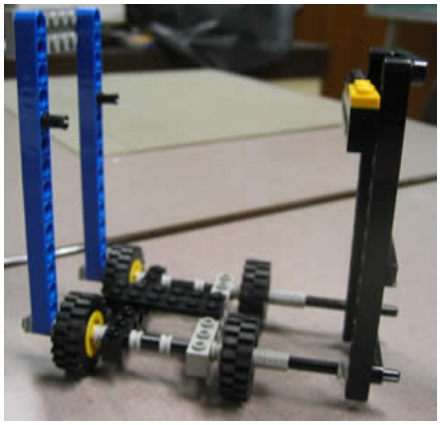
Het wagentje wordt aangedreven met een stappenmotor. Het rijdt over de beweegbare as en houdt de stiftconstructie vast. Het moet dus enerzijds stevig genoeg zijn, maar langs de andere kant ook niet te zwaar.
In een eerste fase van het ontwerp is er niet veel aandacht aan het wagentje besteed: het moest gewoon kunnen rijden. De rails waarover het wagentje rolde, waren toen nog de “goede” ononderbroken rails. Zoals reeds vermeld is deze rail later verwisseld met die van het kader omdat de rail op het kader meer gewicht moet dragen.



Figuur W-1

Op de as van de stappenmotor staat een tandwiel met 8 tanden. Dit tandwiel brengt het koppel over naar een tandwiel met 24 tanden, dat op de rail rust. Langs weerszijden van dit grote tandwiel, zijn er nog twee andere grote tandwielen geplaatst, die een ondersteunende functie hebben. Ze zijn uiteraard niet in contact met het middelste tandwiel, ze dienen gewoon als wiel.
Aan de andere kant van het wagentje zijn twee wielen bevestigd, die op de gladde strook van de beweegbare as steunen. Later zijn de dikke banden vervangen door dunnere, omdat deze veel minder stroef meedraaien (figuur W-1).

Om ervoor te zorgen dat het wagentje niet kan omkantelen, is er getracht het wagentje in te klemmen. Hiervoor is eerst een soort van liftstructuur bedacht (figuur W-2), maar deze maakte het wagentje veel te log. Bovendien beweegt zo de beweegbare as door het gewicht veel moeizamer, wanneer het wagentje op het uiteinde van de as staat. Er is nu een veel elegantere oplossing waarbij de gladde strook ingeklemd wordt met de twee gewone wielen langs boven en met twee kleine wieltjes langs onder. Dit zorgt echter voor een stroevere beweging van het wagentje, omdat er te veel wrijving is. Daarom zijn de onderste twee wieltjes een eenheid naar beneden geplaatst. Hierdoor kan het wagentje nog wel in beperkte mate kantelen, maar het kan er niet afvallen. Bovendien ligt het zwaartepunt nog voldoende boven de as zodat het wagentje uit zichzelf ook zal terugkantelen.



Figuur W-2

Opdat het karretje niet van de beweegbare as zou kunnen rijden is er aan beide kanten een mechanische stop gebouwd. Verder wordt er in het programma op ingestelde grenzen gecontroleerd waar het wagentje niet voorbij mag. Aan een uiteinde is er ook een schakelaar, die door het wagentje wordt ingedrukt wanneer het bij de kalibrering in zijn beginpositie is aangekomen (dit is op de achtergrond van figuur A-3 te zien).

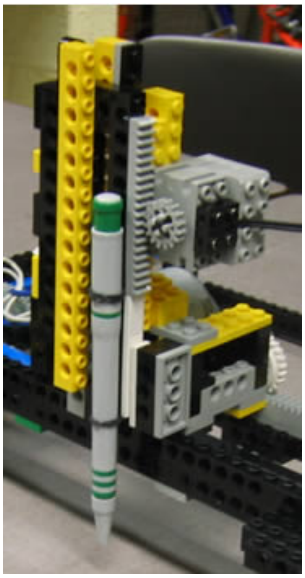
Tijdens het bouwen van het wagentje is er altijd gelet op stevigheid. Door langs de zijkanten dwarse balken op de staven te pinnen kan het geheel niet meer loskomen en kan het wagentje dus ook niet zomaar uiteenvallen.

Om de slechte punten van het wagentje volledig te kunnen elimineren is er de derdelaatste week besloten om het nog eens helemaal af te breken en terug op te bouwen. Ook de stiftconstructie is mee hermaakt. Het wagentje met de stift is daardoor één geheel geworden, terwijl het voordien eigenlijk twee aparte, op elkaar geplaatste onderdelen waren.
Twee uur voor het einde van de laatste werkdag werd ons nog een methode aangereikt om door middel van een tandwieloverbrenging de resolutie met een factor drie te verbeteren. Hiervoor was nog een kleine aanpassing nodig, waardoor de motor iets hoger is geplaatst. Voor meer uitleg over deze tandwieloverbrenging, zie bladzijde 12.

1.3.4 De stiftconstructie

De uiteindelijke stiftconstructie is tot stand gekomen door het samenvoegen van de positieve eigenschappen van de vier andere systemen die in de zes werkweken geconstrueerd werden.

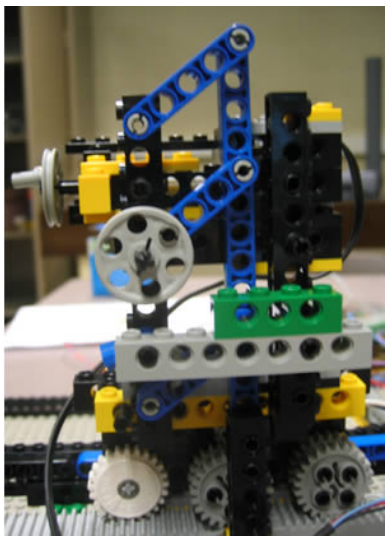
Eerste systeem (figuur S-1)
Een tandwiel wordt aangedreven door een gelijkstroommotor. Dit tandwiel beweegt over een rail en zet zo een cirkelvormige beweging om in een verticale.



Figuur S-1

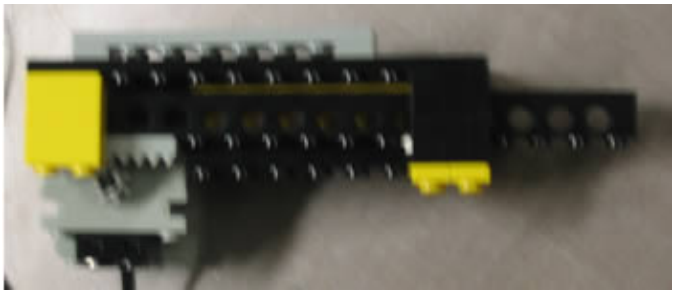
Het voordeel van dit systeem is dat de stift hier enkel verticaal beweegt en geen zijdelingse verplaatsing ondergaat. De nadelen zijn het grote gewicht, de speling van de stift en de rechtstreekse verbinding tussen de gelijkstroommotor en de rail. Voor dit laatste moet er een soort vertragingsmechanisme worden ingebouwd zodat de rotatiesnelheid van de motor niet onmiddellijk overgedragen wordt naar de rail.

Tweede systeem (figuur S-2)
Hier doet de gelijkstroommotor via een wormwiel-mechanisme een tandwiel roteren dat op zijn beurt een stang op en neer beweegt. Zo ontstaat er een hefboomsysteem dat duidelijk minder zwaar is dan de vorige constructie en op het eerste gezicht minder speling vertoont. Het gebruik van een wormwiel vertraagt bovendien de beweging en zorgt ervoor dat de stift niet terug omhoog kan "gedrukt" worden tijdens het schrijven.
Een nadeel hier is dat de stift niet enkel verticaal beweegt waardoor er bij het drukken op het blad altijd wat gewrongen wordt. Door testen bleek ook dat het probleem van de speling nog niet opgelost was.



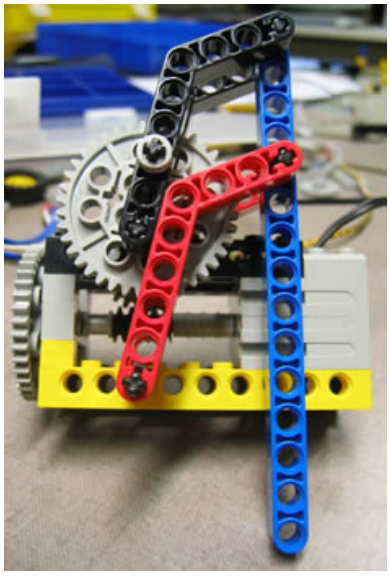
Figuur S-2

Derde systeem (figuur S-3)
Dit is een verbetering van het eerste systeem. De constructie is minder zwaar gemaakt en de verticaal bewegende staaf is beter ingeklemd. Een goed systeem om de stift zelf te bevestigen is er nog niet en ook het gebruik van het wormwielmechanisme is hier moeilijk te integreren. Het massamiddelpunt ligt bovendien niet boven de as.



Figuur S-3

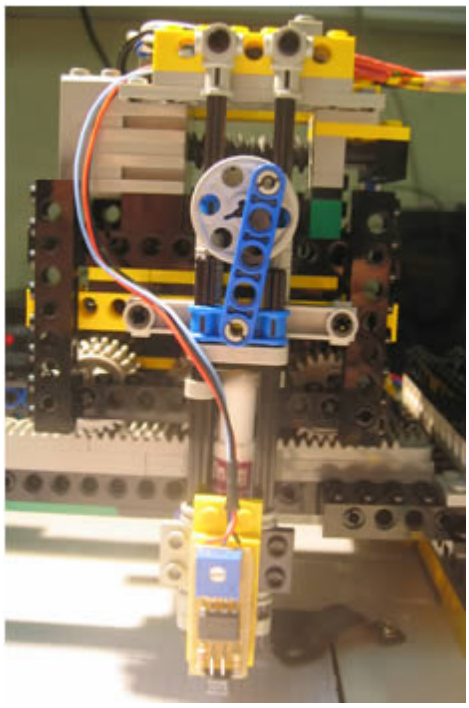
Vierde systeem (figuur S-4)



Hier is het tweede systeem verbeterd door minder materiaal te gebruiken en toch de speling te verminderen. Bovendien ligt het massamiddelpunt in het midden tussen de assen. Het systeem is echter vrij breed en daardoor verkleint de beschrijfbare oppervlakte van het blad sterk. Ook hier blijft het nadeel van de zijdelingse beweging van de stift.

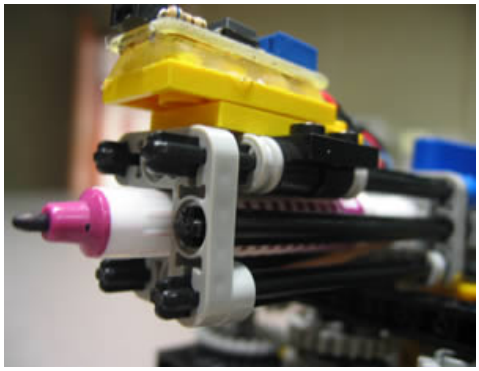
Figuur S-4

Definitief systeem (figuur S-5)
In het definitieve systeem zijn alle positieve eigenschappen van de voorgaande samengebracht: het gebruik van het wormwielmechanisme voor de vertraging van de beweging en de verticale beweging van de stift. Verder zijn er nog enkele bijkomende interessante punten zoals het gebruik van een elastiek en de manier waarop de lichtsensor is vastgemaakt.



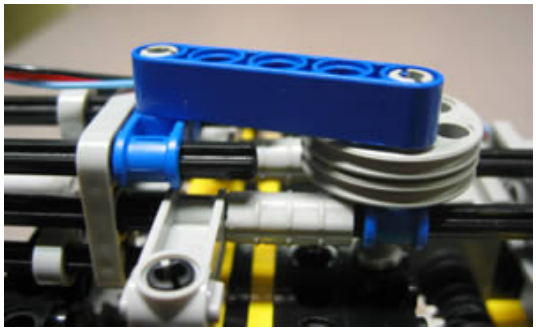
Figuur S-5

De basis van het op en neer bewegen van de stift is een gelijkstroommotor. Deze wekt een rotatiebeweging op die via een staaf overgedragen wordt naar een tandwiel. Met een wormwiel wordt de beweging vertraagd en via een speciaal wit tandwiel overgebracht naar een gewoon rond wiel. Het witte tandwiel slipt door op zijn as wanneer er een draaimoment wordt aangelegd van meer dan 2,5-5 Ncm. Dit zorgt ervoor dat het wagentje intact blijft bij een eventuele blokkering.
Het ronde wiel heeft verschillende gaten waarvan er een gebruikt wordt om een blauwe stang op te bevestigen. Deze bevestiging gebeurt met een grijze pin (en geen zwarte) omdat de stang nog vrij moet kunnen draaien. De verbinding zorgt ervoor dat er slechts beweging mogelijk is in één vlak.
De stift zelf is ingeklemd in een kooi bestaande uit staven en lego-elementen met een rechte hoek (figuur S-6). Deze kooi is bevestigd op de blauwe stang. Om ervoor te zorgen dat het systeem niet horizontaal kan bewegen zijn er staven verticaal vastgemaakt aan het wagentje. Bovendien kunnen ze door gaten in de kooi van de stift glijden waardoor men een op- en neergaande beweging bekomt. Er wordt met andere woorden een rotatiebeweging omgezet in een rechtlijnige beweging door een kruk-drijfstangmechanisme (figuur S-7).



Figuur S-6

Om ervoor te zorgen dat men weet wanneer de gelijkstroommotor zijn beweging moet stopzetten is er een lichtsensor op de kooi bevestigd die aangeeft wanneer de stift het blad voldoende raakt. Ook is er een marge voorzien door een elastiek over de stift te spannen zodat eventuele oneffenheden op het blad geen hindernissen vormen bij het tekenen. Een andere functie van de elastiek is ervoor te zorgen dat men voldoende drukkracht uitoefent op het blad, eigenlijk de eigenschap van een veer.



Figuur S-7

1.4 Extra uitleg over enkele onderdelen en mechanismen

1.4.1 Stappenmotor

Een stappenmotor bestaat uit twee delen: een stator en een rotor. De stator heeft een aantal wikkelingen (elektromagneten) die, als ze in de juiste volgorde bekrachtigd worden, een permanente magneet op de as van de motor doen draaien. Bij het bekrachtigen van een wikkeling door er een stroom door te sturen, wordt de elektromagneet magnetisch. De poolparen op de rotor zullen zich richten volgens het zo ontstane magnetische veld. Voor de beweging van de as en het wagentje is er een stappenmotor met 4 windingen en 12 poolparen gebruikt. Als deze windingen achtereenvolgens bekrachtigd worden dan draait de kern (en dus ook de verbonden staaf) 7,5 graden per stap.

Berekening van de afgelegde afstand bij een stap van de stappenmotor
We zullen nu voor verschillende lego-tandwielen de afstand berekenen die de motor na een stap heeft afgelegd over een rail, wanneer dit tandwiel op zijn as wordt aangebracht.
De motoras draait 7,5° per stap. In radialen komt dit overeen met 0,13 rad.
Een fundamentele lego-unit (FLU) komt overeen met 8 mm.

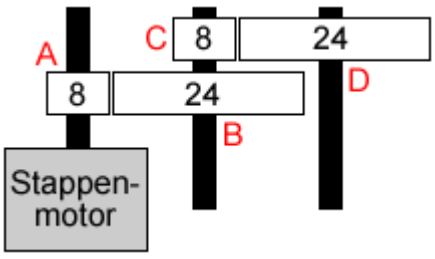
Afstand = straal * hoek
Afstand = straal * 0,13 rad

Gebruikte tandwiel	Straal (FLU)	Afstand (FLU)	Afstand (mm)
8 tanden	0,5	0,065	0,52
16 tanden	1	0,13	1,05
24 tanden	1,5	0,196	1,57

Op de motoren van de plotter worden tandwielen met 8 tanden bevestigd. Omdat een stap in het programma overeenkomt met vier stappen van de motor, is de kleinst haalbare afstand (de resolutie) dus 4*0,52 mm = 2,09 mm.

1.4.2 Tandwieloverbrenging

Omdat een resolutie van twee millimeter vrij grof is, wordt er aan beide stappenmotoren gebruik gemaakt van een tandwieloverbrenging om de resolutie te verkleinen (figuur E-1). Wanneer tandwiel A over een bepaalde hoek draait, dan draait tandwiel B over een hoek die drie keer kleiner is (24/8 = 3). De snelheid van de motor wordt ook met een factor drie verkleind, en het koppel met een factor drie vergroot. De afgelegde afstand blijft echter dezelfde, omdat de stralen van beide tandwielen ook in een factor drie verschillen. Er is dus nog niets gebeurd door enkel tandwielen A en B te gebruiken.

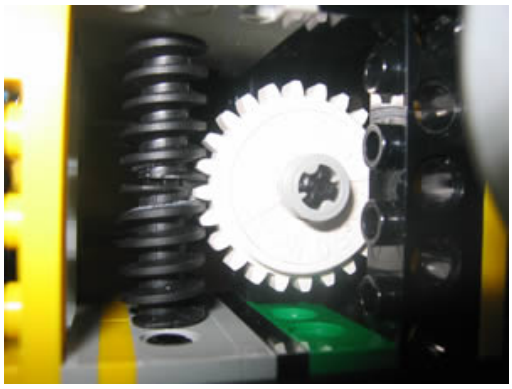


Figuur E-1

Wanneer tandwiel B over een bepaalde hoek draait dan zal tandwiel C over dezelfde hoek draaien, omdat het op dezelfde as is geplaatst. Dit keer is de afgelegde afstand wel kleiner geworden, wat gemakkelijk gezien kan worden in de formule afstand = straal * hoek, waarin de hoek dezelfde blijft en de straal met een factor drie verkleind is. De afstand is dus ook met een factor drie verkleind. Het koppel en de snelheid zijn uiteraard nog dezelfde (dezelfde as).

Door nu tandwiel D op tandwiel C te laten ingrijpen, wordt het koppel weer met een factor drie vergoot en de snelheid met dezelfde factor verkleind. De afstand blijft in deze laatste stap dezelfde, net zoals in de overgang van A naar B.
Tussen tandwielen A en D is er dus een verlaging van de snelheid met een factor 9, een vergroting van het koppel met een factor 9 en een verkleining van de afstand met een factor 3. De resolutie is dus verkleind van 2,09 mm tot 0,70 mm, een merkbaar verschil.

1.4.3 Wormwiel

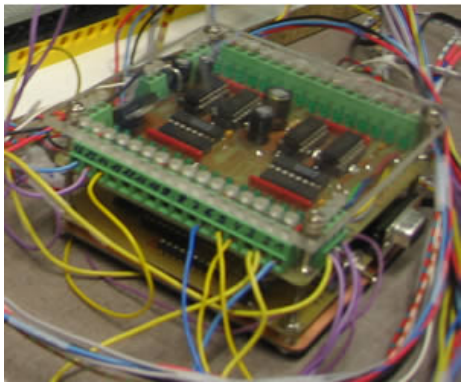


Figuur E-2

Een wormwiel (figuur E-2) is een uitvinding van Archimedes en lijkt sterk op zijn waterschroef. Wanneer het gebruik van een wormwiel gecombineerd wordt met dat van een gewoon rond tandwiel, dan wordt een reductie gecreëerd. Elke omwenteling van het wormwiel verdraait het tandwiel juist over een tand. Voor een volledige omwenteling van een tandwiel met 24 tanden zijn er dus 24 omwentelingen nodig van het wormwiel. Een andere interessante eigenschap van dit mechanisme is dat de beweging slechts in één richting werkt: je kan het wormwiel niet laten roteren door aan het tandwiel te draaien.

1.5 Uitleg bij de sturing van de plotter

Het bouwen van de constructie is één ding. De plotter de vooraf gespecificeerde taken correct te laten uitvoeren een ander. Opdat een mechanische machine zelfstandig taken kan uitvoeren, heeft het een stuureenheid nodig dat alles in goede banen kan leiden. De hier gebruikte eenheid is de PowerBrick (figuur P-1), die bestaat uit een centrale verwerkingseenheid, een volatiel RAM-geheugen van 32 KB, een seriële poortconnectie en 16 parallelle in- en uitgangen. De centrale verwerkingseenheid is een Motorola 68HC11-microcontroller met een kloksnelheid van 2 MHz. Via de parallelle uitgangen kunnen motoren (actuatoren) aangestuurd worden en via de ingangen kunnen gegevens ontvangen worden van verschillende sensoren. De seriële poort kan gebruikt worden voor interactie met de gebruiker via de computer.



Figuur P-1

In het geheugen van de PowerBrick kan een programma geladen worden, ook via de seriële poort. Het programma wordt eerst geschreven in de programmeertaal BrickC en vervolgens gecompileerd tot machinetaal. BrickC is een aangepaste versie van C++, maar uitgerust met extra functionaliteit eigen aan de PowerBrick.

1.5.1 Bespreking broncode

Om het algemene verloop van het programma te verklaren, worden hier eerst enkele belangrijke deelfuncties uitgelegd.

- **Delay (duur)**

Deze procedure wordt gebruikt om de snelheid van de motoren te bepalen. Hierbij wordt gebruik gemaakt van een eenvoudige teller. Zolang de processor aan het tellen is, worden er geen andere taken uitgevoerd.

- **PenOmlaag ( )**

Om de stift op het papier te plaatsen wordt een gelijkstroommotor gebruikt. De procedure *PenOmlaag* zorgt voor een spanningsverschil tussen de polen van de motor, waardoor deze gaat draaien. De motor stopt wanneer de optische sensor die tegen de stift hangt, aangeeft dat hij zich vlak boven het papier bevindt.

- **PenOmhoog ( )**

Om de stift weer omhoog te heffen wordt gedurende een bepaalde tijd een omgekeerde spanning aangelegd dan bij *PenOmlaag*. Zo heeft de motor de tijd om de pen van het papier weg te brengen.

- **VraagCoord (x, y)**

In deze procedure worden twee coördinaten aan de gebruiker gevraagd. Deze worden gebruikt voor het tekenen van elke geometrische figuur en worden by-reference teruggegeven.

- **Beweeg (motor, snelheid, richting)**

De plotter maakt gebruik van twee stappenmotoren, en beide worden door deze procedure gestuurd. Elke motor heeft een nummer dat als parameter wordt meegegeven bij de procedureaanroep. Voor de parameter *snelheid* geldt: hoe hoger de waarde, hoe groter de delay tussen elke stap en dus hoe trager de beweging. Deze waarde mag echter niet te klein zijn, omdat de motor dan niet voldoende tijd heeft om zijn magnetisch veld op te bouwen (zie ook de uitleg over stappenmotoren op bladzijde 12).

- **Positioneer (x, y)**

De procedure *Positioneer* verplaatst de pen naar een welbepaalde positie op het blad. Ze wordt voornamelijk gebruikt om de pen naar een beginpositie te bewegen bij de start van een tekening.

Bovenaan het programma worden de globale variabelen en constanten gedeclareerd. Voor elke actuator wordt een element in een array van *outputdevices* gereserveerd. In de procedure *Init* wordt vervolgens aan elk element een fysieke uitgang van de PowerBrick gekoppeld. Ook de sensoren, die van het type *inputdevice* zijn, worden in deze procedure aan een welbepaalde ingang gekoppeld. Dit laat toe om vanuit alle procedures in het programma met deze poorten te werken om bijvoorbeeld motoren aan te sturen of de waardes van sensoren uit te lezen. Daartoe moet wel de procedure *Init* aanroepen worden, wat dan ook als eerste gebeurd in de hoofdprocedure *start*.
Na het starten van de plotter wordt onmiddellijk de procedure *Reset* aangeroepen. Deze procedure tilt de pen van het papier indien dit nodig zou zijn, en verplaatst het wagentje vervolgens naar de beginpositie op de beweegbare as. Bij aankomst op deze positie zal het wagentje een schakelaar induwen, en daardoor weet het programma dat het wagentje is aangekomen. Vervolgens wordt de beweegbare as naar zijn beginpositie gebracht. Ook hier wordt bij aankomst een schakelaar ingedrukt. Doordat de plotter telkens in deze positie zal komen, ongeacht of iemand het wagentje met de hand heeft verplaatst, is dit een soort van kalibrering.

Nu de plotter klaar is voor gebruik, krijgt de gebruiker via de computer een keuzemenu te zien met een lijst van geometrische basisfiguren. Hiervoor wordt in de hoofdprocedure de procedure *KeuzeMenu* opgeroepen die het menu zal printen naar het terminalscherm. Wanneer een keuze gemaakt is, worden de nodige parameters (breedte, hoogte, straal, middelpunt, …) gevraagd. De waardes die de gebruiker ingeeft worden gecontroleerd om na te gaan of deze binnen het bereik van de plotter liggen. Indien dit zo is, wordt de procedure aangeroepen die instaat voor het tekenen van de gekozen figuur, anders krijgt de gebruiker opnieuw het keuzemenu te zien.

De gebruiker krijgt volgende opties in het keuzemenu:

1. Rechthoek
2. Lijnstuk
3. Cirkel
4. Ellips
5. Programma afsluiten

Een rechthoek tekenen gebeurt met de procedure *Rechthoek*. De gebruiker dient de coördinaten van de linkeronderhoek, de hoogte en de breedte van de rechthoek in te voeren. De beweegbare as en het wagentje met de pen zullen beurtelings in de juiste richting over de juiste afstand bewegen, zodat er een mooie rechthoek op papier gezet wordt. Uiteraard wordt de pen in het begin omlaag gelaten, en op het einde weer omhoog.

Voor het tekenen van een lijnstuk met de procedure *Lijn* zijn vier gegevens vereist: de coördinaten van het begin- en eindpunt. Uit deze gegevens worden de richtingscoëfficiënt en zijn omgekeerde berekent. Er wordt met de omgekeerde van de richtingscoëfficiënt gewerkt bij zwak stijgende of zwak dalende lijnen, want wegens geheugenbeperkingen kan er niet gewerkt worden met kommagetallen.
Afhankelijk van de relatieve posities van het begin- en eindpunt ten opzichte van elkaar, worden verschillende gevallen behandeld om de motoren op de juiste manier te sturen. Ook het geval van een lijnstuk evenwijdig met een van de assen wordt in de procedure afzonderlijk behandeld. De richtingscoëfficiënt is dan ofwel gelijk aan 0, ofwel aan oneindig.
De eigenlijke bewegingen worden verzorgd door aanroepen naar de procedure *Beweeg*.

Aangezien een cirkel een speciaal geval is van een ellips, wordt voor beide figuren eenzelfde procedure gebruikt, de procedure *Ellips*. De op te geven parameters zijn de coördinaten van het middelpunt en de lengtes van beide halve assen (langs x- en y). Schuine ellipsen zijn niet mogelijk. Om een cirkel te krijgen volstaat het voor beide halve assen eenzelfde waarde op te geven, maar via het keuzemenu wordt deze waarde maar een keer gevraagd.
Bovenaan in het programma worden een aantal cosinuswaarden in een array gestoken, en deze worden hier gebruikt om een aantal punten op de ellips/cirkel te berekenen. Tussen deze punten worden gewone lijnstukjes getekend met de procedure *Lijn*, maar omdat er per kwadrant zes punten berekend worden valt dit nauwelijks op.

1.5.2 Een leuk extraatje: vrij tekenen met de controller

Door gebruik te maken van een controller pad kan de gebruiker zelf de plotter bedienen en zo willekeurige figuren tekenen zonder parameters in te moeten geven. De controller

bestaat uit enkele optische sensoren en schakelaars die de verschillende bewegingsrichtingen voorstellen (meer uitleg over de controller vindt u op bladzijde 18). Het programma voor deze bediening is grotendeels hetzelfde. In de procedure *Init* worden aan de sensoren fysieke poorten gekoppeld. De functies *Delay, Beweeg, Reset, PenOmhoog* en *PenOmlaag* blijven dezelfde.
De pen op en neer bewegen gebeurt in dit programma met een tussenstap. In de controller staat maar één knop om de pen te bedienen, en het programma moet dus zelf de juiste actie uitvoeren. De status van de pen wordt in de variabele *laatsteActie* opgeslagen.
De belangrijkste procedure in dit programma is de procedure *Control*, die in feite *KeuzeMenu* vervangt. Zolang de gebruiker niet heeft aangegeven dat hij wil stoppen, wordt *Control* in een lus aangeroepen. *Control* controleert telkens welke sensor de gebruiker geactiveerd heeft en roept vervolgens de procedure *Beweeg* aan met de juiste parameters. Bij elke beweging worden de nieuwe coördinaten in variabelen opgeslagen. Deze gegevens worden gebruikt om na te gaan of de pen zich niet buiten het bereik van de plotter begeeft. In dat geval krijgt de gebruiker een melding op het scherm.

1.5.3 Opgedoken problemen tijdens het programmeren

Omdat C++ een sterk objectgeoriënteerde taal is, is in eerste instantie ook geprobeerd om hiervan gebruik te maken, en te werken met klassen. Er is echter gebleken dat werken met klassen in BrickC niet van een leien dakje gaat en dat het veel problemen met zich meebrengt.
Toch is er verder geprobeerd om zonder klassen zo'n overzichtelijk mogelijke programma's te schrijven. Door voor elke deeltaak aparte procedures te schrijven die enkel doen wat ze moeten doen, kan de code gemakkelijk gelezen en begrepen worden en kunnen ook gemakkelijk aanpassingen aan de code worden gedaan. Bovendien wordt het programma zo korter, omdat veelvoorkomende acties maar een keer worden geschreven. Ook dit is niet zonder problemen verlopen, maar uiteindelijk is er toch in het opzet geslaagd om een duidelijk, bondig en werkend programma te schrijven.

Naast de problemen op vlak van de programmeertaal zelf, waren er de problemen in verband met de PowerBrick. Soms deed deze (en doet hij nog steeds) onverklaarbaar raar. Een cirkel wordt bijvoorbeeld soms als een achthoek getekend, maar als je onmiddellijk daarna dezelfde cirkel laat tekenen, doet hij het wel goed.
Verder heeft de PowerBrick maar een geheugen van 32 KB, wat een enorme beperking betekent voor het programma. Meermaals werd de limiet overschreden, maar toch is het steeds gelukt om het programma terug te brengen tot onder de 32 KB. Door te experimenteren is er vastgesteld dat er een aantal aanpassingen het nodige geheugen merkbaar kunnen verkleinen:

- Indien een bepaalde waarde constant zal blijven gedurende heel het programma, loont het om hiervan een expliciete constante te maken met het woord *const*. Voor drie constanten van het type *short* betekent dit een totale besparing van 1 KB.
- Floating-point getallen gebruiken zorgt voor een enorme geheugentoename. Niet alleen omdat die types op zichzelf meer ruimte innemen, maar ook omdat de microcontroller daarvoor een heel extra instrumentarium nodig heeft om er mee te kunnen rekenen. Die bijkomende code zal automatisch in de gecompileerde versie gestoken worden door de compiler.
- Meestal gebruik je voor getallen altijd het integer-type, omdat in moderne computers geheugen vaak geen probleem vormt. Er zijn echter andere numerieke types die

minder geheugen in beslag nemen, maar waarvan de getallen ook beperkter in grootte zijn. In onderstaande tabel staat een overzicht van enkele integrale types in BrickC en hun bereik.

Naam van het type	Soort getal	Bereik
Int	32 bit integer	-2147483648 tot +2147483647
Short	16 bit integer	-32768 tot +32767
Unsigned short	16 bit integer	0 tot 65535
Char	8 bit integer	-128 tot +127
Unsigned char	8 bit integer	0 tot 255

Door rationeel om te springen met de beschikbare types (rekening houdend met het bereik) kan er een bijkomende geheugenbesparing verkregen worden.

- Een letterteken neemt 8 bit (1 byte) geheugen in, en een stuk tekst dus al snel een aantal bytes. Door hierop in te spelen en de tekst in het keuzemenu zo beperkt mogelijk te houden, neemt het programma ook minder geheugen in.
- Parameters *by-reference* doorgeven neemt blijkbaar meer geheugen in dan ze *by-value* door te geven.

Een gevolg van dit beperkte geheugen is dat er nu twee aparte programma's zijn: een voor het tekenen via de computer, en een voor het tekenen met de controller. Hoewel beide programma's voor een groot stuk gelijk zijn, is het niet mogelijk om een programma te maken dat beide functies combineert en binnen de grenzen van het geheugen blijft. Dit komt vooral door het feit dat voor de controller extra poorten gebruikt worden op de PowerBrick. Het benutten van een in- of uitgang van de PowerBrick vraagt relatief veel geheugen.

1.6 Handleiding voor het tekenen via de computer

Nadat het juiste programma in de PowerBrick geladen is, start u de plotter door op de resetknop te drukken. De pen zal zichzelf kalibreren door zowel de beweegbare as als het wagentje met de pen naar hun respectievelijke beginposities te bewegen. Op het scherm wordt een keuzemenu getoond met als opties de verschillende basisfiguren. Om een figuur te tekenen typt u het bijhorende cijfer in, en drukt u op de entertoets. Om te stoppen geeft u het cijfer 5 in, ook gevolgd door de entertoets. Wanneer u een ongeldig cijfer ingeeft, krijgt u opnieuw het keuzemenu te zien.

Het gebruikte coördinatenstelsel is een orthonormaal assenstelsel. Alle gegevens die bij het tekenen moeten ingevoerd worden, worden uitgedrukt in dit coördinatenstelsel. De ingegeven waardes zijn evenredig met het aantal stappen dat de stappenmotor moet zetten, maar met een factor 4 verschil. Eén stap in het coördinatenstelsel komt overeen met 4 stappen van de stappenmotor. Er kan enkel met natuurlijke getallen gewerkt worden, aangezien alles zich in het eerste kwadrant afspeelt. De maximale waardes die binnen het bereik vallen zijn 379 voor x en 249 voor y. Wanneer men andere dan deze toegelaten waarden invoert, krijgt men opnieuw het keuzemenu te zien.

Het tekenen van een rechthoek

Nadat u het juiste cijfer hebt ingegeven in het keuzemenu zullen u vier gegevens gevraagd worden.

- BeginX: de x-coördinaat van de linkeronderhoek van de rechthoek, gezien met de oorsprong linksonder van de kader.
- BeginY: de y-coördinaat van de linkeronderhoek.
- Breedte
- Hoogte

Het tekenen van geroteerde rechthoeken is niet mogelijk met deze functie. U kan natuurlijk wel vier lijnstukken tekenen en zelf de juiste coördinaten ingeven.

Het tekenen van een lijnstuk
Voor een lijnstuk worden ook vier parameters gevraagd:

- BeginX: de x-coördinaat van het beginpunt van het lijnstuk
- BeginY: de y-coördinaat van het beginpunt
- EindX: de x-coördinaat van het eindpunt van het lijnstuk
- EindY: de y-coördinaat van het eindpunt van het lijnstuk

Het tekenen van een cirkel
Bij een cirkel dient u achtereenvolgens volgende gegevens op te geven:

- BeginX: de x-coördinaat van het middelpunt
- BeginY: de y-coördinaat van het middelpunt
- Straal

Het tekenen van een ellips
Voor het tekenen van een ellips worden ook de coördinaten van het middelpunt gevraagd, en twee stralen:

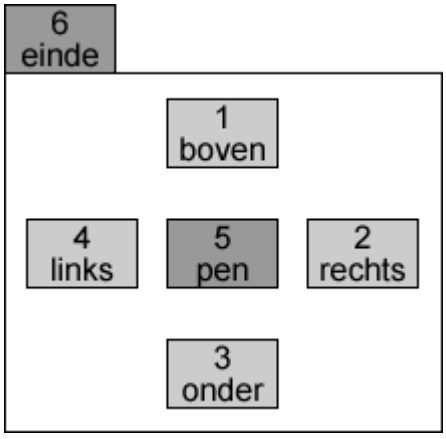
- BeginX
- BeginY
- HalveAsX: de lengte van de halve as volgens de x-richting
- HalveAsY: de lengte van de halve as volgens de y-richting

Het tekenen van schuine ellipsen is niet mogelijk met deze functie.

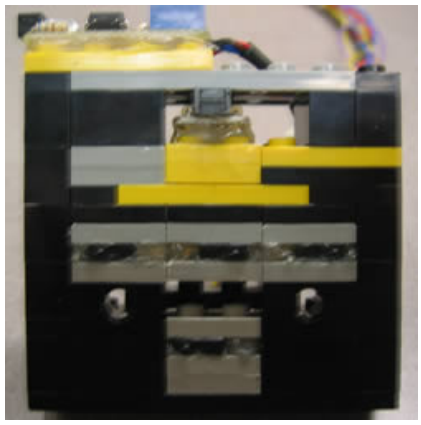
1.7 Handleiding voor het tekenen met de controller

Nadat het juiste programma in de PowerBrick geladen is, start u de plotter door op de resetknop te drukken. De pen zal zichzelf kalibreren door zowel de beweegbare as als het wagentje met de pen naar hun respectievelijke beginposities te bewegen.
U heeft nu de volledige controle over de pen met behulp van de controller. Op dit toestel staan zes knoppen (figuur P-2 en P-3).

Om te tekenen verplaatst u de pen naar de gewenste positie van het blad met behulp van de richtingsknoppen 1 tot en met 4. Met knop 5 kan u de pen tegen het blad aandrukken. Gebruik nu opnieuw de richtingsknoppen om iets te tekenen. U kan de pen weer opheffen door opnieuw op de penknop te duwen. Om het programma te beëindigen drukt u op knop 6.



Figuur P-2



Figuur P-3

1.8 Conclusie

Er is geslaagd in het opzet een apparaat te ontwerpen dat in staat is op een aanvaardbaar niveau automatisch basisfiguren te tekenen. Van de doelstellingen zijn er twee zaken niet uitgevoerd: het tekenen van cirkelbogen en het makkelijk vervangen van de stift. Langs de andere kant zijn er ook een aantal zaken die de verwachtingen overstegen: het tekenen met de controller, de vrij goede resolutie, het tekenen van ellipsen, …

Hieronder worden kort enkele pluspunten en tekortkomingen aangehaald.

1.8.1 Pluspunten

- De belangrijkste basisfiguren kunnen getekend worden: lijn, rechthoek, cirkel en ellips. Dit alles gebeurt met een zeer aanvaardbare resolutie zodat alle lijnen er zeer vloeiend uitzien.
- De constructie is makkelijk verplaatsbaar en ook stevig, hoewel er maar weinig materiaal is gebruikt. Hierdoor is de plotter ook heel compact en is er op geen enkele plaats een probleem met doorbuiging. De overbruggingen die gemaakt worden zijn beperkt in lengte.
- Het papier is zeer stevig ingeklemd. Het papier kan dan ook onmogelijk door invloeden van de pen verschoven worden. Dit zorgt voor een grotere nauwkeurigheid van de tekeningen.

1.8.2 Tekortkomingen

- Hoewel de constructie in staat is de belangrijkste basisfiguren te tekenen, kunnen er geen cirkelbogen en veelhoeken getekend worden. De voornaamste oorzaak hiervan is het beperkte geheugen van de PowerBrick. Een programma bereikt vrij snel de maximumgrootte van 32 KB en er zijn verscheidende inspanningen nodig geweest om het binnen de vereiste grootte te krijgen.
- De penconstructie is uiteindelijk goed ontworpen. Alleen het vervangen van de pen is een moeilijke operatie. Dit zou beter moeten.
- Door het gebrek aan drukschakelaars is er gebruik gemaakt van optische sensoren. Een groot nadeel hiervan is dat de constructie sterk beïnvloed wordt door de omgevingsbelichting.

- Door de vele sensoren, schakelaars en de drie motoren zijn er veel draden nodig om al deze toestellen te verbinden met de PowerBrick. Dit zorgt voor een wirwar van kabels, wat niet echt elegant is. Toch is geprobeerd de kabels zo goed mogelijk te ordenen en te groeperen, zodat ze zo weinig mogelijk invloed hebben op de goede werking van de plotter.

1.8.3 Onopgelost probleem

Wanneer een cirkel getekend moet worden, tekent de plotter soms een achthoek. Maar wanneer onmiddellijk hierna hetzelfde commando ingetikt wordt, dan tekent de plotter vaak wel de juiste cirkel. Hetzelfde probleem doet zich soms voor met het tekenen van ellips. Hier wordt dan een lijn getekend.
Er wordt veronderstelt dat de oorzaak dus noch in het ontwerp, noch in het programma ligt, maar wel ergens in de PowerBrick.

Deel 2 Boekhouding

Het gehele project moest voltooid worden in 6 werkdagen. Dit komt overeen met 240 manuren (5 man x 6 dagen x 8 uren/dag).
Het werk is in twee groepen verdeeld in functie van de eigenschappen van de verschillende groepsleden. Het bouwteam hield zich bezig met de mechanische constructie en het programmeerteam met de software. Doordat ieder taken deed die hem interesseerde, kon ieder zo goed mogelijk bijdragen tot het project. Dit betekent echter niet dat er geen onderling overleg was tussen de teams, of dat er iemand van het programmeerteam nooit mee bouwde…
De eerste drie dagen bestond het bouwteam uit Jan, Lieselotte en Claire, en het programmeerteam uit Kenny en Thomas. Vanaf de vierde dag heeft Jan een beetje gependeld tussen beide teams.

In volgende tabel staat een overzicht van het gewerkte aantal manuren per deeltaak van het project.

Deeltaak	**Aantal manuren**
Bouwen	88
Programmeren	81
Onderlinge bespreking	35
Werken aan het verslag	25
Overige	11
Totaal	**240**

2.1 Bouwen van de mechanische constructie

Zoals reeds aangehaald in het technische verslag zijn er hoofdzakelijk vier onderdelen in de mechanische constructie. De verdeling van de werktijd per onderdeel wordt meer gedetailleerd weergegeven in onderstaande tabel.

Onderdeel van de constructie	**Aantal manuren**
De stiftconstructie	29
Het wagentje	19
De basisconstructie	14
De beweegbare as	13
Overige	13
Totaal	**88**

De basisconstructie bestaat uit het kader met de rails van de vaste as en de inklemming van het blad. De beweegbare as bestaat uit de aandrijving die steunt op het kader, de rails, de wielen die zorgen voor de inklemming met het kader en de constructie voor de bedrading. Het wagentje is de constructie met motor die de stift draagt en die over de beweegbare as rijdt. Onder stiftconstructie wordt het hele mechanisme op het wagentje bedoeld, dat de stift vasthoudt en op en neer doet bewegen. Bij “overige” horen onder andere het solderen en ontwarren van de draden, het verwijderen van onnodige blokjes en het zoeken of maken van extra onderdelen, zoals de latten voor de inklemming van het blad.

Uit de tabel blijkt duidelijk dat de stiftconstructie veruit voor de meeste problemen heeft gezorgd. Het bedenken, bouwen en dan weer afbreken van de verschillende stiftconstructies heeft veel tijd gevergd, maar heeft dan ook tot een beter resultaat geleid.

2.2 Programmeren

Bij het programmeren hoort niet alleen het effectieve schrijven aan het programma, maar ook het oefenen met de programmeertaal tijdens de eerste werkdag, het debuggen van fouten, het compacter schrijven van de code, enzovoort.

2.3 Onderlinge bespreking

Hieronder vallen alle momenten waarbij het team rond de tafel ging zitten om nieuwe ideeën uit te werken, mogelijke problemen te bespreken en op te lossen. Ook het opstellen van de dagplanning en taakverdeling, het schrijven van het logboek en de nabespreking van de dag werden hieronder verrekend.

2.4 Overige

Dit onderdeel bevat de onderlinge uitleg die tussen bouw- en programmeerteam werd georganiseerd. Het bouwteam legt de constructie uit en het programmeerteam het programma. Verder vallen hieronder het maken van berekeningen en het opzoeken van informatie.

Buiten de voorziene uren is er nog aan het verslag en de presentatie gewerkt.

Deel 3 Reflectie

3.1 Nabespreking van het project

Iedereen van de groep is het erover eens dat het project “Lego Plotter” goed geslaagd is op verschillende vlakken. Enerzijds is er uiteraard het eindresultaat. Ondanks de beperkte materiële mogelijkheden hebben we toch een product ontworpen dat zijn taak correct volbrengt en dat bovendien ook nog stevig en compact is. De controller die toelaat om “met de losse hand” te tekenen, vergroot bovendien de aantrekkelijkheid van de plotter.

Iets wat echter nog belangrijker is, is hoe we tot dit resultaat zijn gekomen. Hierbij komen niet alleen kennis en vaardigheden kijken, maar ook voor een groot stuk teamwerk. Het team met de vijf intelligentste studenten heeft immers niet noodzakelijk het beste eindresultaat. Doordat we goed op elkaar afgestemd waren en doordat ieder van ons specifieke kwaliteiten had, is het hele project relatief vlot verlopen.
Bij de start van het project zijn we samen gaan zitten om te denken over wat we wilden maken en hoe we dit zouden aanpakken. Doordat er al wat opzoekwerk was verricht vóór de aanvang van het werkcollege, was er al enig inzicht in de problematiek van een plotter. Gebruiken we twee bewegende assen of slechts één met daarop een wagentje? Of zullen we misschien enkel een wagentje maken en het papier laten bewegen? Zal de plotter punten zetten of volle lijnen tekenen?
Tijdens het project waren er drie processen te onderscheiden die continu op elkaar inspeelden en in elkaar overvloeiden: ontwerp, bouw en programma.
De ontwerpfase is uiteraard een belangrijke fase en hoewel we het niet expliciet zo hebben genoemd, was deze fase wel degelijk aanwezig. Zowel in het begin als tussendoor zijn er momenten geweest dat we, allemaal samen of enkel het bouwteam, rond de tafel zijn gaan zitten om systemen uit te denken voor een bepaald onderdeel van de plotter. Aansluitend op deze ontwerpfase was er een bouwfase die de ideeën in de praktijk omzette, en een programmeerfase die de nodige aanpassingen in het programma deed.
Door deze nauwe samenwerking konden fouten of problemen heel snel opgemerkt en aangepast worden. Vanaf het moment dat het bouwteam een onderdeel klaar had, kon snel een klein testprogramma geladen worden om het geheel uit te testen. De hierbij ontstane feedback werd vervolgens gebruikt om verbeteringen aan te brengen aan het onderdeel en vervolgens opnieuw te testen.

In het logboek hebben we systematisch neergeschreven welke inzichten we verworven hebben tijdens elke dag. Hieronder staan de belangrijkste nog even op een rijtje:

- Stevigheid is een belangrijk aspect van een machine. Op elkaar gestapelde lego-blokjes zijn niet stevig. Je moet zoveel mogelijk gebruik maken van kliksystemen met pinnetjes of van andere verstevigingmethoden.
- Door tandwielen op een rationele manier te gebruiken, kan je de resolutie heel wat verkleinen.
- Je moet rekening houden met de plaatsing van gewicht: zo dicht mogelijk bij de aandrijving en zo symmetrisch mogelijk boven een as.
- Soms is het efficiënter om een bepaald onderdeel helemaal af te breken en terug van nul te beginnen, dan te proberen het onderdeel te verbeteren.
- Lichtsensoren zijn te veel afhankelijk van omgevingslicht en zijn daardoor niet betrouwbaar genoeg voor bepaalde doeleinden. Drukknoppen daarentegen werken altijd en zijn dus ook te verkiezen wanneer het kan.

- Bij de programmering van een programma voor een microcontroller dien je rationeel om te springen met geheugen. Gebruik constanten waar mogelijk en denk goed na over de variabelentypes die je gebruikt (characters of shorts in plaats van altijd die integers; geen floating point-getallen omdat ze gebruik maken van heel wat extra code in de compiler).
- Het is in programmeeropdrachten van groot belang om op voorhand duidelijke richtlijnen vast te leggen in verband met naamgeving van variabelen, methodes, bestanden enzovoort. Bij het gebruik van versienummers dient ook afgesproken te worden wanneer een nieuw nummer gestart te worden.

3.2 Het groepsgebeuren

Een van de belangrijke troeven van onze groep is dat we alle vijf goed met elkaar overweg kunnen. Bovendien is ieder van ons sterk gemotiveerd en geïnteresseerd, wat zeker een vereiste is voor goed groepswerk. Toch is groepswerk geen gemakkelijke zaak en geeft het aanleiding tot enkele problemen. Hieronder staan een aantal zaken die we zijn tegengekomen. Namen van groepsleden zijn weggelaten.

- Doordat niet iedereen altijd hetzelfde denkt, moet je in een team naar compromissen zoeken. Soms moet je toegeven aan iemand anders, en soms geeft iemand anders toe aan jou. Het is hierbij heel belangrijk om met iedereen rekening te houden.
- Karakters kunnen botsen. Een combinatie van drie koppige individuen geeft soms aanleiding tot wervelingen. Hoewel dit meerdere malen is voorgekomen, is het gelukkig nooit uitgedraaid op ruzie.
- Communicatie is uiterst belangrijk, maar ook hier kan het soms mislopen:
    - Wanneer iemand een idee heeft, dan zou hij dit duidelijk moeten uitleggen aan de rest van de groep. Het is niet bevorderlijk voor teamwerk wanneer die persoon zijn mening probeert door te dringen terwijl de rest nog niet beseft wat hij bedoelt. Te kort uitleg geven is dus niet echt goed.
    - Anderen geven dan weer te uitgebreid antwoord op vragen wanneer korte antwoorden volstaan. Dit kan tot irritaties leiden bij andere groepsleden.
    - Wie kritiek op iemand heeft, mag deze uiteraard uiten. Daarbij moet er echter wel op gelet worden dat de kritiek niet afbrekend is, maar eerder een constructief karakter heeft. Op die manier kan je veel van elkaar leren.
- In een groep moet je de taken verdelen op basis van kwaliteiten en interesses van ieder groepslid. Hierbij ontstaan twee problemen:
    - Er is niet altijd een duidelijk beeld op wat er nog allemaal moet gebeuren
    - Doordat je in aparte groepjes werkt, heb je geen idee van hoe het andere groepje zijn taak heeft aangepakt, je ziet enkel het resultaat. Je kunt wel aan elkaar vertellen wat er allemaal is gebeurd, maar toch is het onmogelijk om hierbij de volledige problematiek naar voren te brengen.

3.3 Evaluatie van het werkcollege zelf

We zijn het er allemaal over eens dat er weinig materiaal aanwezig was. Tijdens het werkcollege werd dit eerder als een negatief punt ervaren. Achteraf bekeken is dit echter nog niet zo slecht: je kan niet zomaar alles maken. Je hebt bepaalde beperkingen en daardoor moet je wat langer nadenken, moet je wat inventiever zijn. Het is een deel van de uitdaging en naar volgende jaren toe moet dit niet echt veranderd worden.

Het bouwen met Lego was ook heel goed gedocumenteerd met de tekst “The Art of LEGO Design”.

Iets wat als lastig werd ervaren is het aantal beschikbare computers. In het lokaal was er één computer aanwezig. Doordat er in ieder geval een programma geschreven moet worden is deze praktisch altijd bezet. Wanneer er aan het logboek gewerkt moest worden, of later aan het verslag, stelde dit wel problemen. Gelukkig was er in onze groep iemand met een draagbare computer, maar zelfs dat was niet altijd voldoende. Iemand naar de computerzalen in gebouw K was een optie, maar niet altijd efficiënt omdat er soms in overleg iets getypt moet worden. Ideaal zouden er twee computers voorzien moeten worden door de dienst werktuigkunde, en dan kunnen studenten eventueel zelf nog een laptop meenemen.

Op vlak van de programmeertaal was er nogal weinig ondersteuning. Er was weliswaar een documentatiebundel voorzien, maar deze was eerder beperkt op sommige vlakken. Zeker tijdens de eerste werkdagen zou het handig zijn geweest indien er iemand was komen horen of er problemen waren. Dit moet natuurlijk iemand zijn die de taal BrickC goed kent.

Verder zijn er nog volgende puntjes die we even willen aanhalen:

- In de eerste werkdagen hadden we nogal de indruk dat er eerder pure kritiek kwam op ons mechanisch gedeelte, en dat deze pas later constructief geuit werd.
- De richtlijnen in verband met het gebruik van bijkomend materiaal waren nogal onduidelijk.
- Spijtig dat er zo weinig geheugen aanwezig is in de PowerBrick. Langs de ene kant is dit ook een uitdaging, maar dit heeft ons verplicht twee aparte programma’s te maken: een voor het tekenen via computer, en een tweede om te tekenen met de controller.

Deel 4 Bijlagen

4.1 Logboek

4.1.1 Maandag 9 februari

De eigenschappen van de teamleden
Claire: beslisser, organisatorisch, geen programmeerkennis
Lieselotte: doener, analytische geest, geen programmeerkennis
Jan: creatieve geest, Lego-expert, probleemoplosser
Kenny: organisatorisch, doener, kennis van programmeren
Thomas: analytische geest, probleemoplosser, kennis van programmeren

Het project
In plaats van een kopieermachine te bouwen die punten op een blad zet, hebben wij geopteerd voor een plotter die lijnen trekt. Het oorspronkelijke idee was om enkel eenvoudige geometrische figuren te laten tekenen, maar tijdens een eerste brainstormsessie kwamen we op volgende uitbreidingen:
- de mogelijkheid om met een soort control pad in real-time te tekenen
- eventueel een functie-plotter voor wiskundige functies zoals parabolen, ... (dit vereist vooral programmeerwerk en geen hardwarematige aanpassingen)

Projectplanning:
Week 21 t/m 23: bouwen + programmeren
Week 24 en 25: Integreren van hard- en software, bijhorende problemen oplossen en verfijnen
Week 26: Grondig testen en eventueel werken aan de presentatie en het rapport
Week 28: Presentatie

Dagplanning (9 februari)
Voormiddag: planning maken, taakverdeling, systemen en mechanismen uitdenken, wennen aan programmeertaal
Namiddag: Verder mechanismen uitdenken, proberen de motoren te doen werken via Powerbrick

Taakverdeling
Bouwen met Lego: Claire, Lieselotte, Jan
Programmeerwerk: Kenny, Thomas

Realisatie
Voormiddag: geplande taken, kader van plotter gemaakt
Namiddag: Het legogedeelte van een eerste prototype van de plotter is af. Deze moet nog wel getest worden met de Powerbrick, waarna zeer waarschijnlijk nog verscheidene aanpassingen moeten doorgevoerd worden.
Het programmeerwerk ging in het eerste deel van de middag goed, maar daarna doken er onverwachte problemen op die tegen het einde van de dag opgelost zijn. De stappenmotor kan reeds aangestuurd worden via enkelvoudige bekrachtiging.

Geboekte inzichten
- Stilstaand blad en bewegende pen is verkozen boven rollend blad met vaste plotter omdat het gemakkelijker realiseerbaar is
- Het oorspronkelijk plan was om twee bewegende assen te gebruiken, maar wegens praktische bezwaren in verband met het vasthouden van de pen, is van dit idee afgestapt. Het huidige prototype werkt met een bewegende as waarop de pen kan bewegen.
- Wegens het beperkt aantal lego-onderdelen konden we niet aan elke zijde van het plotterframe met tandwielen en rails werken. Dit is momenteel opgelost door aan een zijde te werken met tandwielen/rails en aan de andere zijde met een schuivende verbinding.
- Het is belangrijk gebleken om de constructie te voorzien van voldoende stevigheid

To do
- Verder programmeerwerk
- Gebouwde constructie testen met de bewegende motoren
- De bevestiging van de pen herzien en verbeteren

4.1.2 Maandag 16 februari

Dagplanning
Voormiddag:
- De controller (afstandsbediening) laten werken zodat het bouwteam het prototype gemakkelijk kan testen en eventueel aanpassingen aan de constructie doen.
- De bevestiging van de stift herzien en verbeteren
- Verder de motoren programmeren

Namiddag:
- programmeren: lijnen, rechthoeken en cirkels laten tekenen
- bouwen: problemen die opgemerkt worden door verdere tests oplossen; bevestiging stift afwerken en de sturing hiervan testen met de motoren
- Rond 16h aan elkaar de opbouw van het prototype uitleggen zodanig dat iedereen weet waarvoor een bepaald onderdeel dient; logboek schrijven

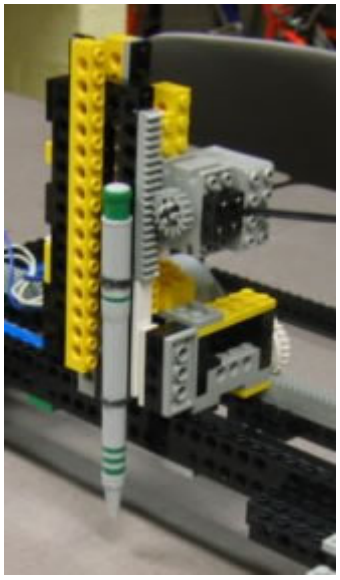


Foto 1: de oorspronkelijke bevestiging van de pen

Taakverdeling
Kenny en Thomas: programmeren
Lieselotte, Jan en Claire: bouwen. Lieselotte, Claire en Jan verschillende systemen voor de bevestiging van de stift uitwerken en vervolgens de voor-en nadelen onderling bespreken.

Realisaties: bevestiging stift
1) Eerste systeem (zie foto 1)
Problemen: niet stevig genoeg, heel zwaar, het gewicht is niet goed verdeeld en er is veel speling

2) Tweede systeem: hetzelfde principe als ervoor maar lichter, steviger, kleiner, stabieler en met veel minder speling.
Probleem: de massa is niet goed verdeeld

3) Derde systeem (zie foto 2): Hier wordt er gewerkt met een soort hefboom. Er is hier een zeer goede verdeling van de massa en er is een minieme speling.
Probleem: de (DC-)motor die de pen op en neer moet kunnen bewegen, draait heel snel. We moeten via tandwielen de rotatiesnelheid verminderen

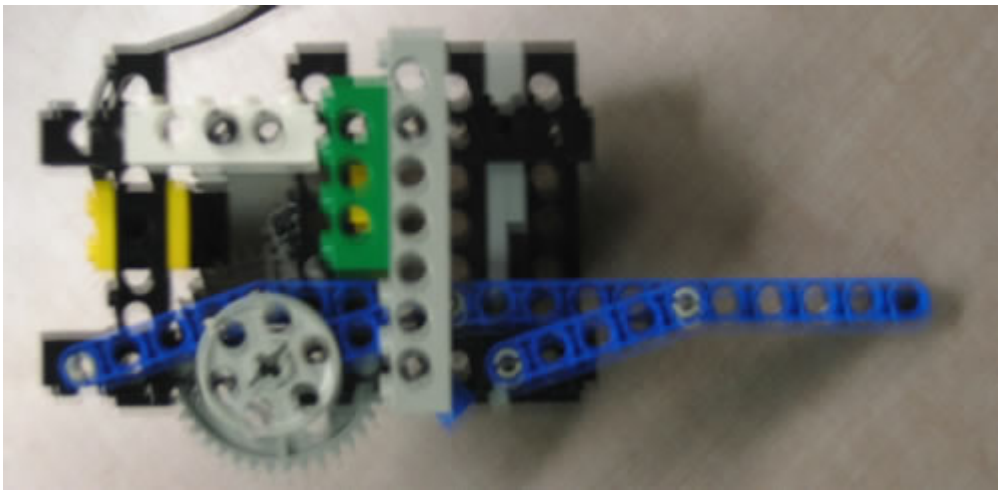


Foto 2: Het hefboomsysteem (sorry voor de wazige foto)

4) Vierde en huidige oplossing: Hetzelfde principe als het derde systeem maar dan door gebruik te maken van een wormwiel.

Verdere bouwrealisaties
- Door een Legoplaat als ondergrond te gebruiken is de hele constructie stabieler en steviger geworden.
- Onnodige onderdelen zijn verwijderd (bijvoorbeeld de blauwe dubbele wand aan de vaste rails, vergelijk tussen foto's 3 en 4)
- Om de werking van de plotter niet te hinderen zijn de aansluitdraadjes van de motoren verlengd (met wat

soldeerwerk). Uit praktische overwegingen zijn bij elkaar horende draadjes vervolgens samengebonden.

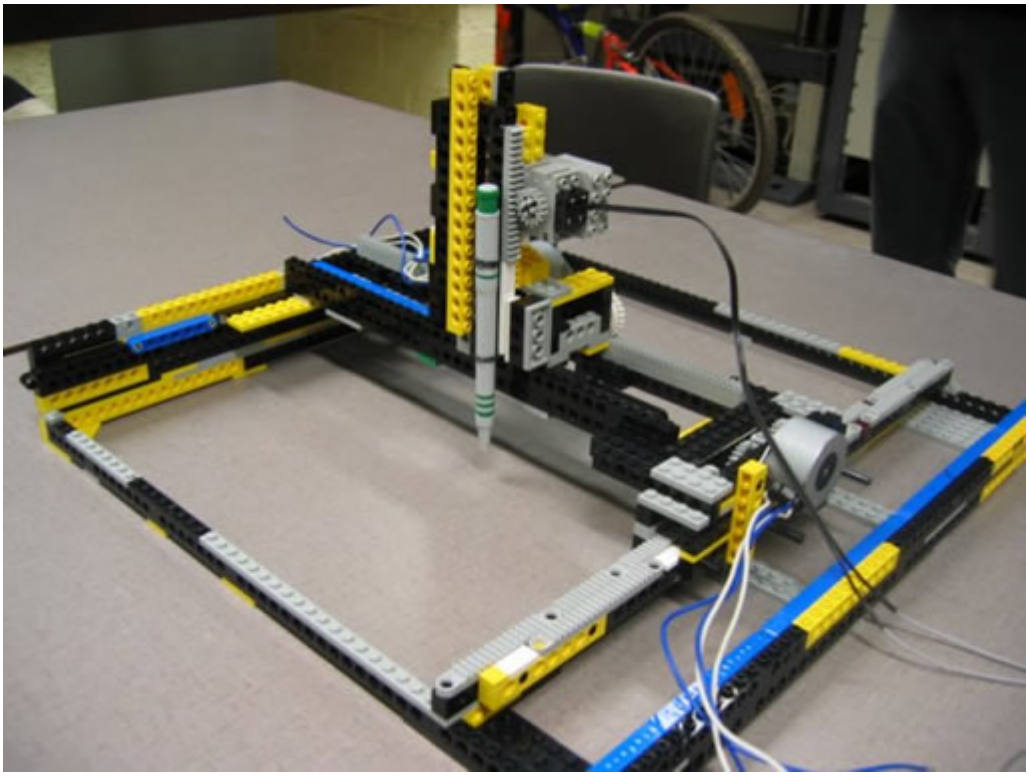


Foto 3: Prototype aan het begin van de dag

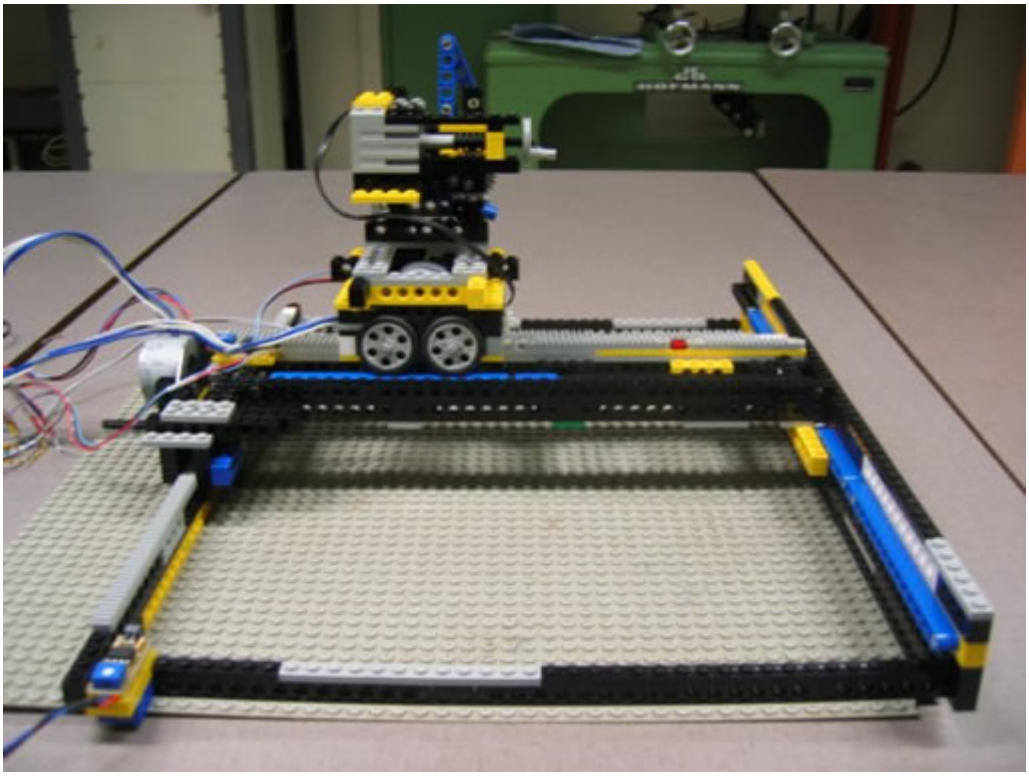


Foto 4: Prototype aan het einde van de dag

Realisaties bij het programmeren

- *Werken met controller:* bijna in orde, er zijn nog enkele problemen met het op en neer bewegen van de pen.
- Er is begonnen met een 'hoofdprogramma' om de plotter te doen werken. Bij inschakeling vertrekt hij naar de

vervolgens beweegt.
- Extra moeilijkheid: de structuur van de taal: er duiken moeilijkheden op om het programma overzichtelijk en bondig te houden. Daardoor is een heleboel tijd verloren. We zijn bijgevolg niet geraakt tot het tekenen van basisfiguren.

Geboekte inzichten
Programma's nemen vrij snel veel geheugen in, vooral de sensoren en de drukknoppen nemen veel geheugen in.
Mogelijke oplossingen:
- De controller en het "automatische" programma in aparte programma's steken en dus ook apart inladen in de Powerbrick
- Het aantal knoppen op de controller trachten te verminderen
- Code compacter maken (dit heeft echter reeds tot problemen geleid met de programmeertaal)

- Het wisselen van de "goede" en de "slechte" rails heeft een grote optimalisatie teweeg gebracht. De goede rails dienen nu voor het deel van de constructie dat de grootste last moet dragen.
- Er is een kleine fout bij het werken met tandwielen als de bewegingsrichting verandert.
- We hebben veel inzicht gekregen met het uitdenken van de stift-constructie, zoals het zo licht mogelijk proberen maken van een constructie, dus zo weinig mogelijk stukken gebruiken. We hebben leren werken met tandwielen en worm gears.
- Het gewicht wordt best zoveel mogelijk boven de rails geplaatst en niet op de gewone wielen. Zoniet kantelt het wagentje.

To do
- De grootte van een stap van de stappenmotor berekenen, d.w.z. welke afstand afgelegd wordt bij gebruik van verschillende tandwielen.
- Draden ontwarren en labelen!
- Verder verbeteren van de huidige constructie
- Verbeteren van de rails op de beweegbare as

- Bij het programmeren is het belangrijkste dat nu gerealiseerd moet worden het automatisch tekenen van basisfiguren door de plotter. Dit zal hoogst waarschijnlijk nog enkele gebreken aan de constructie en de programma's aan het licht brengen.

4.1.3 Maandag 23 februari

Dagplanning
Voormiddag:
- Draden ontwarren en labelen
- De bewegende (slechte) rail herbekijken
- Grondplaat herbekijken
- Het bewegen van de pen doen werken
- Bouwteam: huidige code bekijken zodanig dat ze gewend raken met de programmeertaal (om later eventueel mee te kunnen programmeren)
- Programmeerteam: Nagaan of het programma waar we vorige keer mee geëindigd waren, werkt

Namiddag:
- De plotter figuren proberen te laten tekenen
- Bouwteam kan mee programmeren (is uiteindelijk niet gebeurd)
- De grootte van een stap van de stappenmotor berekenen en dit nagaan in de praktijk

Realisaties
- De grondplaat is momentaal verwijderd want die was niet mooi vlak. Daardoor trok het de hele constructie een beetje scheef. Volgende week zal de plotter gemonteerd worden op een houten plank.
- Het witte tandwiel is even verplaatst naar het mechanisme van de pen, maar zorgde daar voor te veel speling. Momenteel is het dus gewoon zoals ervoor.
- De draden zijn allemaal losgemaakt en vervolgens netjes gebundeld teruggestoken.
- Het bouwteam heeft het huidige programma gelezen en de zaken die onduidelijk waren, zijn uitgelegd door het programmeerteam.
- Het programma is verder uitgebreid zodanig dat de plotter een rechthoek kan tekenen. Er zijn voor een cirkel en lijnstuk al procedures maar die werken nog niet naar behoren. Eens Cirkel echter werkt, zijn we ook in staat (dankzij de huidige structuur van de procedure Cirkel) om veelhoeken te tekenen. Cirkel kan bovendien ook makkelijk uitgebreid worden naar Ellips en dan hebben we eigenlijk alle basisgeometrische figuren.
- Het op en neer bewegen van de pen zorgde vorige week nog voor problemen, maar nu werkt dit (op vlak van

software) goed.
- Het huidige programma is qua code iets compacter, wat een voordeel is met de beperkte geheugenruimte van de PowerBrick (32 KB). Dit heeft wel wat tijd gekost (zie ook verder).
- Claire en Lieselotte zijn reeds begonnen met het schrijven van het verslag.

Berekening van een stap
1 stap = 7,5° = 0,13 rad
1 Lego-unit = 0,8 cm

- voor een tandwiel met 8 tandjes: straal = 0,5 units
> afstand = 0,065 units
> per stap: 0,052 cm
- voor een tandwiel met 16 tandjes : straal = 1 unit
> afstand = 0,13 units
> per stap: 0,104 cm
- voor een tandwiel met 24 tandjes : straal = 1,5 units
> afstand = 0,195 units
> per stap: 0,156 cm

De afgelegde afstand hangt af van het tandwiel dat we op de motoras bevestigen.

Bij het controleren van de berekeningen met de plotter bleek er een factor 4 verschil te zitten. De reden was snel gevonden: een stap in het programma komt eigenlijk overeen met 4 stappen in de praktijk.

Problemen (al dan niet opgeloste)
- Bij het tekenen van rechthoeken bleek er nogal wat speling op de pen te zitten, waardoor de hoeken afgerond werden. Ook de pen ging niet voldoende omhoog bij het teruggaan naar de oorsprong. We hebben dit allemaal opgelost en rechthoeken tekenen gaat nu vrij vlot.

- Cirkels en lijnen tekenen geeft momenteel nog volgende problemen:

- Met de bekomen resolutie blijkt het moeilijk om schuine lijnen en cirkels mooi te tekenen.
- De berekening punten cirkel is blijkbaar niet helemaal correct
- Bewegen in de vier richtingen zorgt voor een lang stuk code dat moeilijk te vereenvoudigen is

Bekomen inzichten
- Het tekenen van basisfiguren blijkt ingewikkelder dan oorspronkelijk gedacht. De moeilijkheid zit niet zo zeer in het uitvoeren op zich, maar om in mooie code een mooi resultaat op papier te zetten.
De rechthoek tekenen is relatief eenvoudig: vraag de gebruiker de begincoördinaten, breedte en hoogte van de rechthoek. Vervolgens tekenen we de vier zijden door telkens de juiste assen te laten tekenen.
Een schuine lijn wordt getekend door de twee motoren telkens één voor één kleine stapjes te laten nemen.
Ook het tekenen van een cirkel hanteert hetzelfde principe, maar daar veranderen het aantal stapjes gedurende de beweging. Dit geeft echter nog niet het gewenste resultaat.

- In de huidige opstelling kunnen de lichtsensoren beschadigd geraken bij een eventueel falen van de software. Het is "good practice" om met zoiets rekening te houden en de sensoren zo te plaatsen dat ze geen contact kunnen maken met de opstelling.

To do
- Het programma laten controleren op "out of bound"-fouten
- De lichtsensoren verplaatsen
- Een manier vinden om de bik stevig vast te maken, want momenteel gebruiken we hier een gewoon stukje kleefband voor.
- De variabelen in het begin van het programma allemaal op nul zetten (we denken dat hierdoor onverklaarbare onregelmatigheden met PowerBrick opgelost kunnen worden)
- Verder de basisfiguren uitwerken

Discussies in de groep
Er is een soort van meningsverschil binnen de groep op vlak van programmeerstijl. Thomas wil liever ineens "proper" object-georiënteerd programmeren terwijl Kenny en Jan liever eerst een eventueel "niet-propere" snel werkende code schrijven en daarna de code wat opkuisen. Iedereen is akkoord dat er in de oorspronkelijke code een aantal zaken enorm lastig waren tijdens het programmeren (doordat we de poorten en sensoren locaal definiëren moeten we deze bij elke procedure-aanroep meegeven als parameter).

De eerste dag van het project hadden we dus een stuk niet-propere code die werkt, en die hebben we op dag 1 of 2 herschreven met behulp van klasses. Deze code werkte niet onmiddellijk en we hebben dan in eerste instantie geprobeerd om deze wel te doen werken (naar Thomas zijn zin), een beetje tegen de zin in van Kenny. We zijn nog steeds niet zeker of de gebruikte taal BrickC goed overweg kan met klasses, aangezien we hiermee verscheidene malen op problemen stootten. Thomas geeft wel degelijk toe dat er hierdoor heel wat tijd is verloren, maar is nog steeds overtuigd van het nut van propere code reeds tijdens de ontwikkelfase van een procedure en niet pas nadat ze werkt.
Bij het begin van de derde dag (vandaag) hebben we geprobeerd om de niet-helemaal-propere, werkende code een beetje in te korten op een iets properdere manier (maar zonder objecten/klasses). Ook dit werkte niet onmiddellijk, en daar hebben we alweer een heel klein beetje tijd verloren, maar dit keer heeft Thomas sneller beseft dat zijn methode niet echt efficiënt was gezien de beperkte tijd, en is in hoofdzaak doorgegaan met het werkende programma. Omdat we vandaag echter een laptop bijhadden en omdat Jan ondertussen mee was gaan programmeren heeft Thomas op de laptop getracht om de werkende code toch properder te schrijven. Ditmaal werkte de propere code wel en vervolgens hebben Kenny en Jan verder geprogrammeerd in een gedeeltelijk properder programma. Thomas is dan verder gegaan met het properder maken van de code en tegen het einde van de dag is er een propere werkende code ontstaan, waar iedereen tevreden over is.

Het compromis van (desnoods vuil) programmeren om iets werkend te krijgen en simultaan iemand die de code proper maakt, blijkt vrij goed te werken en is zeker geen bijkomend tijdverlies, aangezien je toch niet met meer dan twee personen op één computer kan werken.

4.1.4 Maandag 1 maart

Dagplanning
Voormiddag:
- Procedure Lijn() verbeteren
- Lichtsensoren verplaatsen
- Dubbele inklemming wagentje
- Bevestiging bik

Namiddag:
- Grenzen beperken
- User interface
- Wieltjes plaatsen aan de andere kant van de vaste as (sleep verminderen)

Realisaties
- De kader is op een houten plank gelijmd.
- De lichtsensoren zijn verplaatst.
- Wieltjes geplaatst aan andere kant vaste as
- Er zijn een aantal overbodige blokjes weggehaald van de kader
- In de voormiddag zijn er wieltjes geplaatst aan de onderkant van de beweegbare as om het wagentje in te klemmen, en is de bik op een andere manier vastgemaakt.

- Omdat het hele wagentje eerder een opeenstapeling was geworden van allerlei onderdelen die na elkaar waren gebouwd, is in de namiddag heel het wagentje uiteen gehaald om een nieuw, verbeterd wagentje te kunnen maken. Er is nu een nieuwe constructie gevonden voor de bik waarbij enkel nog een verticale beweging gebeurt, en geen horizontale meer (we gebruiken geen hefbomen meer).
Gevolg: alles is lichter en kleiner, maar toch nog niet optimaal qua stevigheid.

- Op vlak van het programma: de problemen die er vorige week nog waren (o.a. met de procedure Lijn) zijn opgelost. De plotter kan nu volgende figuren tekenen:

- Rechthoek / vierkant
- Lijnen: schuin/horizontaal/verticaal
- Cirkels *
- Ellipsen *

* zijn nog niet met de plotter zelf getest, maar de coördinaten worden wel correct berekend (dit hebben we met de hand nagegaan door de berekende coördinaten uit te zetten in een orthogonaal assenstelsel)

Problemen
Het programma naderde tegen de namiddag de geheugenlimiet van 32 KB, terwijl er toch nog een aantal zaken in het programma moesten inkomen. Helemaal op het einde is dit toch iets verbeterd door zoveel mogelijk chars

en shorts te gebruiken in plaats van altijd integers. Door de teksten die naar de gebruiker worden gestuurd zoveel mogelijk in te korten is er ook nog wat ruimte vrijgekomen.
Resultaat: de .s19-file is van 31,7 KB naar 29,8 KB gekrompen.
Maar: we hebben dit laatste programma nog niet kunnen testen op de Powerbrick. We moeten nu goed nagaan of de beperkingen opgelegd door het gebruik van chars en shorts, niet te streng zijn en eventueel in het programma laten controleren op overflows.

Inzichten
- Alles moet heel licht, zo weinig mogelijk materiaal gebruiken
- Het bouwen: we hadden voor het wagentje te veel apart gewerkt, dus uiteindelijk 4 afzonderlijke stukken op elkaar moeten vastzetten wat zeker niet optimaal was (zie ook persoonlijke commentaar).
- Iets helemaal uit elkaar halen en vervolgens terug iets ineensteken is een heel vruchtbare methode om een machine te verbeteren.
- Bij het schrijven van programma's die uitgevoerd zullen worden op een computer, is geheugengebruik geen belangrijke factor. Wanneer men echter programma's gaat schrijven die op een kleine controller moeten werken (voor bijvoorbeeld de Powerbrick) moet men hier sterk rekening mee houden. Het beschikbare geheugen is daar immers vaak heel beperkt.

Persoonlijke commentaar
Claire: Volgens mij hebben we weer te veel apart gewerkt (voor het wagentje opnieuw te construeren). Ik vond dat we moesten beginnen bouwen met in ons achterhoofd wat er allemaal op het wagentje bevestigd moest worden. Maar als we met meerdere personen tegelijk bouwen, betekent dat dat we zeer veel moeten communiceren. En dat ging niet: iedereen was weer veel te veel bezig met het optimaliseren van zijn eigen deeltje.
Lieselotte: Eigenlijk zouden we allemaal ons eigen (volledige) prototype kunnen bouwen, daarna vergelijken en de beste onderdelen van elk bijeen nemen. Maar we hebben hiervoor niet genoeg materiaal. Momenteel moeten we onmiddellijk iets afbreken als we iets willen veranderen. Dit dwingt ons wel eerst goed na te denken.

To do
- Wagentje verstevigen en toch wat optimaliseren
- Cirkel en ellips testen (die nu in theorie kloppen)
- Probleem met geheugen verder bekijken en testen
- Enkele figuren aan het menu toevoegen zoals bijvoorbeeld veelhoeken
- Zorgen dat het wagentje niet buiten de fysische grenzen kan rijden

4.1.5 Maandag 8 maart

Dagplanning en taakverdeling
De constructie moest vandaag volledig af zijn, zodanig dat de laatste werkdag (15 maart) gebruikt kon worden voor noodgevallen, en om verder aan het verslag te werken.

- Bevestiging van de pen / constructie wagentje: Claire, Lieselotte en Jan
- Inklemming papier: Lieselotte
- Cirkel en ellips doen werken / softwarematige grenzenbeperker: Kenny
- Aan verslag werken: Thomas en Claire

Realisaties
Op mechanisch vlak:
- Er is eindelijk een stevig karretje met niet te veel materiaal dat niet uit elkaar kan vallen. Het staat ook stevig op de as door een dubbele inklemming met wielen
- De constructie van de pen is nu ook compleet. Er wordt een elastiekje gebruik om een soort vering te creëren. De lichtsensor is erop bevestigd om te weten wanneer de pen beneden is.
- Door een dun stalen plaatje (ca. 0,6 mm dik) te plaatsen op de gladde lat van de bewegende as staat het wagentje steviger op de as terwijl de beweging toch nog vlot verloopt.
- De kabels zijn een klein beetje "gekanaliseerd" zodat ze niet in de weg lopen tijdens het plotten
- Een van de twee lichtsensoren is nog een keer verplaatst
- Er zijn vier klemmetjes gevezen op de houten plaat om het papier vast te houden, maar nu moeten er nog twee latjes tussen komen voor een betere inklemming.

Op vlak van programmeren:
- Alle basisfiguren kunnen worden getekend (Lijn, rechthoek, cirkel, ellips)

Voor het verslag:

- Er is al wat geschreven voor het technische verslag, maar dit is nog niet voldoende grondig. Er moet veel meer specifieke informatie over de werking en opbouw van de plotter.
- Er is verder al het grootste stuk van de boekhouding gemaakt van het aantal werkuren

Problemen
- Het huidige programma is in feite net iets te groot: ongeveer 32,6 KB. toch klaagt Powerbrick er nog niet over dus tot nu toe is dat nog niet zo'n ramp.
- De figuren "sluiten" soms niet goed: het beginpunt komt niet altijd mooi overeen met het eindpunt.

Bekomen inzichten
- Het gebruik van een elastiek/veer hadden we eigenlijk eerder moeten gebruiken. Tot nu toe bekwamen we een goed contact tussen stift en blad door de dc-motor nog wat extra te laten draaien nadat de sensor op high springt. Door de vering is dit niet meer nodig.
- Het is nodig duidelijke richtlijnen te hebben bij het geven van bestandsnamen voor de broncode, zoniet ontstaat er soms verwarring over welk programma nu nog werkte en welk niet meer (zeker wanneer je in verschillende versies begint te wijzigen)

To do
- Latjes maken voor een betere inklemming van het blad papier
- Wat testen welke de optimale snelheid is om te stappenmotoren te laten draaien
- Kijken hoe we het programma toch nog iets kleiner kunnen maken
- In de procedure cirkel nagaan of er geen verwisseling is gebeurt tussen x en y. Daar gebeurt immers nog een eigenaardigheid...
- Nagaan hoe het komt dat de figuren niet mooi sluiten.
- Meer commentaar schrijven bij de broncode (deze moet afgedrukt bij het verslag gestoken worden)
- Een flowchart maken van het programma (ook als bijlage bij het verslag)
- De grenzen van het blad verifiëren en de waarde aanpassen in het programma.
- Verder werken aan het verslag
- Er zouden veel detailfoto's moeten gemaakt worden van de uiteindelijke plotter om zo de tekst in het verslag visueel te verduidelijken
- Beginnen aan de presentatie

4.1.6 Maandag 15 maart

Dagplanning
- Latjes bevestigen om het papier in te klemmen
- Nagaan waarom de figuren niet sluiten
- De grenzen van het blad verifiëren
- Een vaste plek voor het blad markeren, zodanig dat het coördinatenstelsel nog behouden blijft wanneer er een nieuw blad geplaatst wordt

- Nagaan of de plotter versneld kan worden (variabele SNELHEID aanpassen) zonder de goede werking van de plotter te wijzigen
- X en Y omwisselen bij ellips (daar was nog iets mis...)
- Het programma becommentariëren
- Een flowchart maken van de code
- De aanroep naar PenOmlaag() in procedure Lijn() weghalen: dit wordt immers gebruikt in Ellips() waardoor de pen elke keer op en neer beweegt tijdens het tekenen.

- Aan het verslag werken
- Nadenken over de presentatie
- Veel detailfoto's trekken voor in het verslag
- Alles testen
- Een demoprogramma maken voor op de presentatie

Realisaties
- Het blad staat nu stevig dankzij de latjes en bijhorende klemmen. Er is een stuk karton gebruikt om aan te duiden waar het blad zich moet bevinden.
- Op het wagentje is er een kleine wijziging aangebracht met serieuze gevolgen voor de vlotheid waarmee het wagentje beweegt. Er was een klein niveauverschil tussen de tandwielen en de rails waardoor de wieltjes een beetje kromgetrokken werden.
- Door een 'gear train reduction' te gebruiken is de resolutie verkleind met een factor 3. Dit geeft veel vloeiendere tekeningen, maar verkleint ook het hele coördinatenstelsel. Dit zou betekenen dat het beschrijfbare gedeelte van het blad ook sterk verkleind zou worden, omdat we voor de coördinaten met chars werken die maar een waarde

tot 128 kunnen hebben. Door de chars om te zetten naar shorts is ditvprobleem verholpen terwijl we toch binnen de geheugengrens van 32 KB blijven.
Hiermee is ook het probleem van de niet-sluitende figuren opgelost: nu het wagentje rustige, kleine stapjes neemt, trilt het niet zo wat een exactere beweging meerbrengt.
- Door enkele verbeteringen aan de kader is er een groter beschrijfbaar gebied
- De lichtsensoren op de X- en Y-as zijn vervangen door drukknoppen. De lichtsensoren hingen immers te sterk af van de belichting van de omgeving, iets wat nefaste gevolgen kan hebben op de presentatie.
- Enkele draadjes zijn nog langer gemaakt zodat ze niet in de weg lopen

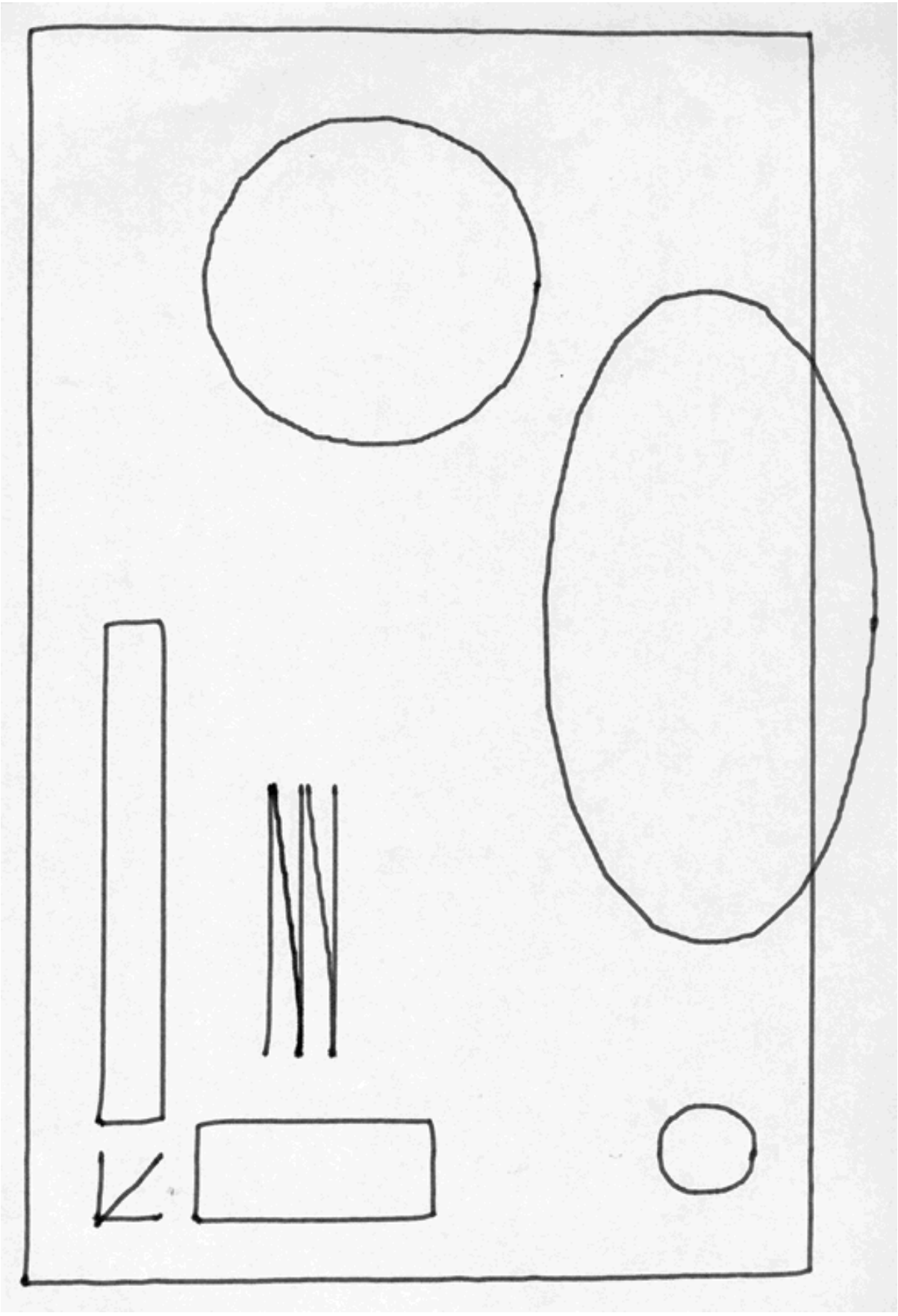
- Een cirkel of ellips tekenen gaat veel vlotter nu de pen niet heel de tijd op en neer gaat
- In het hele programma staat commentaar om uit te leggen hoe het werkt.
- Verder is de code volledig. Alle figuren kunnen naar behoren getekend worden.
- Het programma is getest op afhandeling van onverwachte gegevens: negatieve getallen, decimale getallen en te grote getallen.
- De controller is een beetje veranderd en door een ander programma te laden in de PowerBrick kan deze gebruikt worden om "freestyle" te tekenen.

- Er is een programma gezocht en gevonden op internet waarmee van een stuk code automatisch een flow chart gemaakt wordt
- Voor het verslag zijn de bedachte penconstructies beschreven. Ook andere zaken zijn al beschreven, maar er is zeker nog veel werk aan.
- Er is een tijdsplanning voor de presentatie met een verdeling van wie wat vertelt. Voor de inhoud zullen we ons voornamelijk op het verslag baseren.
- Voor de demonstratie zal een 3D-legoblokje getekend worden, waarvoor de nodige coördinaten uitgerekend zijn. De procedure-aanroepen moeten nog wel geschreven worden.

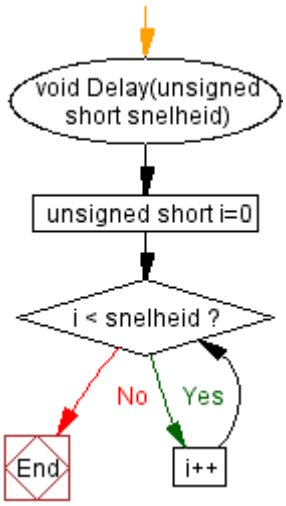
Geboekte inzichten
- Met gear train reduction kan een enorme prestatieverbetering verkregen worden. Alle figuren worden nu veel soepelder getekend en zien er niet meer zo amateuristisch uit.
- Lichtsensoren moet je enkel gebruiken wanneer je niet anders kan. Ze zijn te gevoelig aan omgevingslicht voor bepaalde doeleinden.

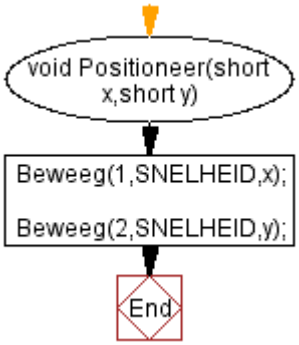
To do
- Verslag afwerken. We hebben de taken hiervoor verdeeld. Ieder zal zijn deel schrijven en daarna zal alles samengebracht worden in een uniforme layout.
- Controleren of de draadjes goed zijn gesoldeerd (of er overal goed contact is)
- De flowcharts maken voor elke procedure
- Meer foto's trekken: er zijn er nog veel te weinig om het verslag duidelijk te kunnen illustreren
- Het demoprogramma programmeren
- De presentatie voorbereiden

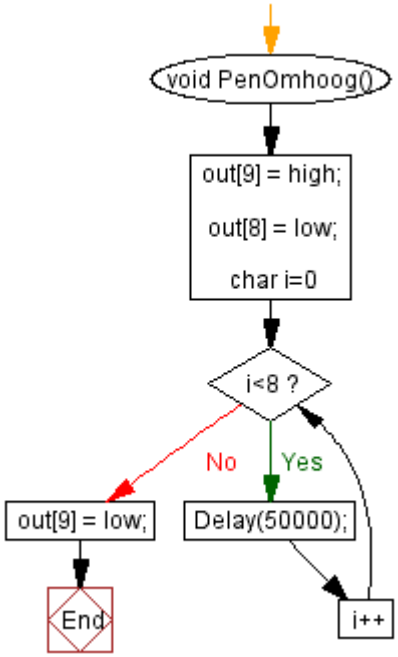
4.2 Portfolio

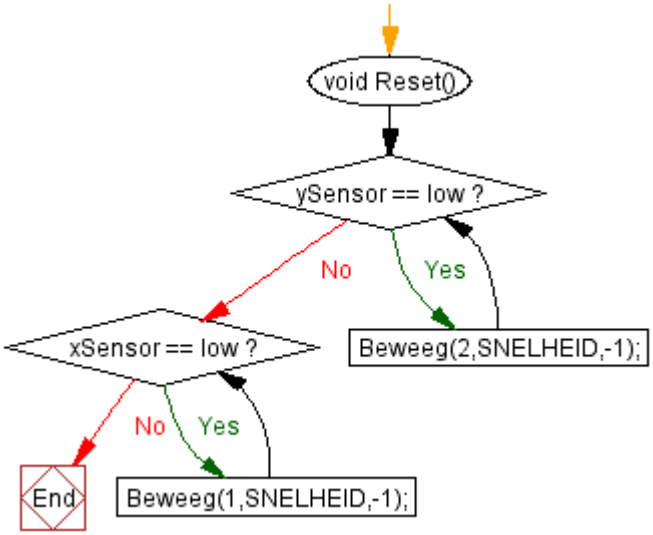


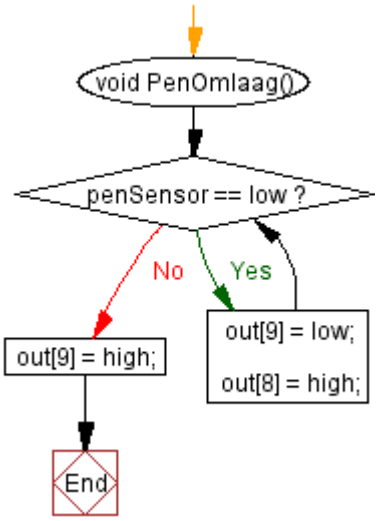
4.3 Flowcharts

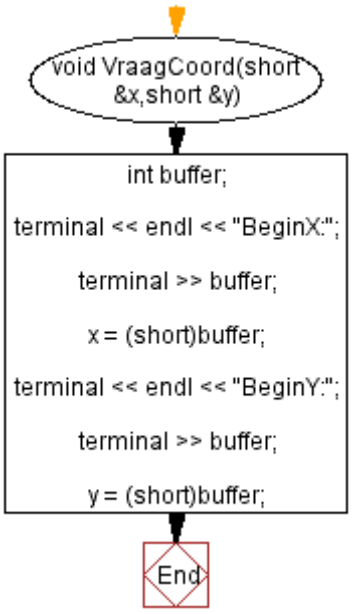


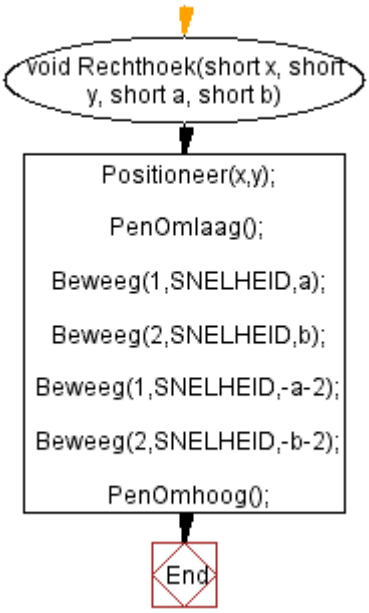


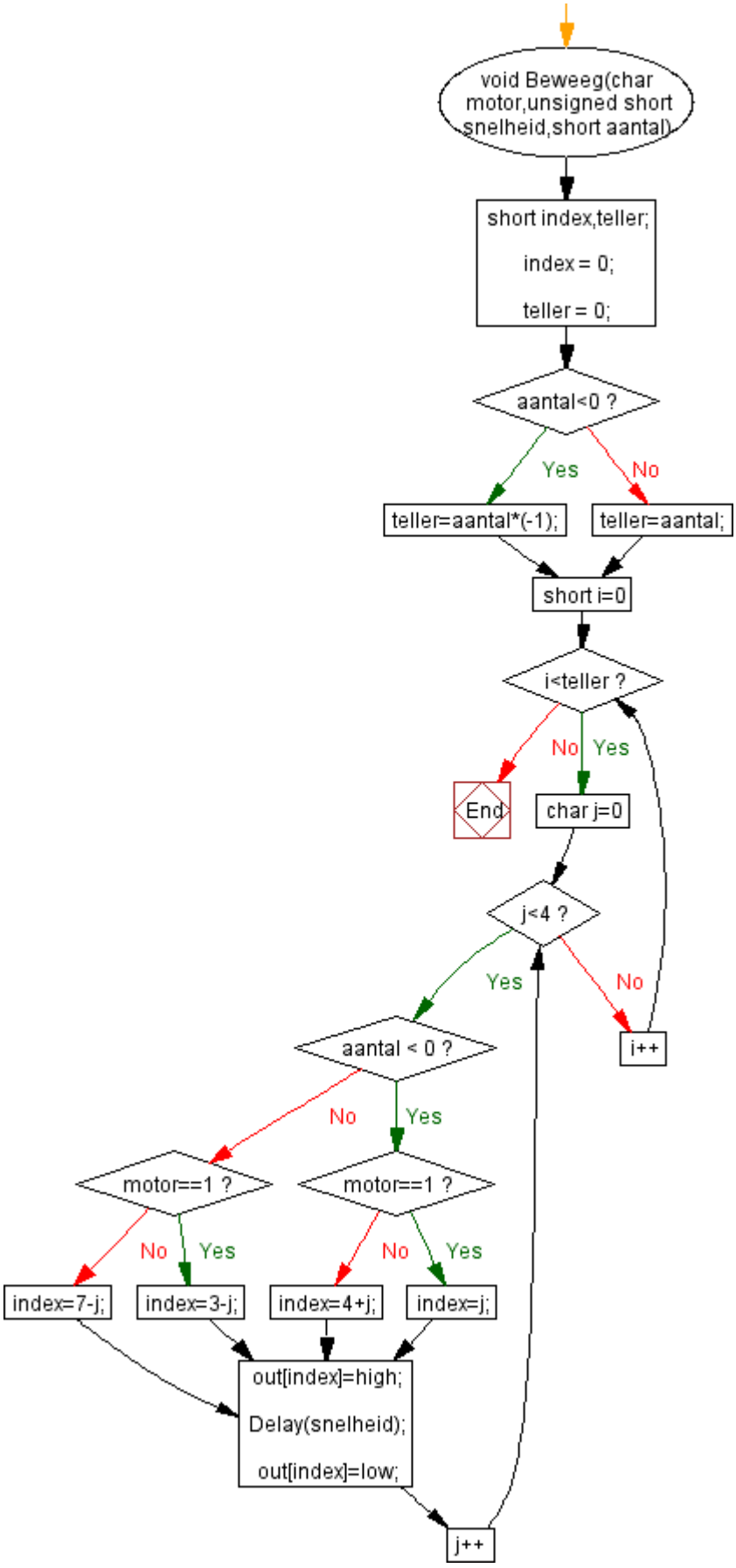


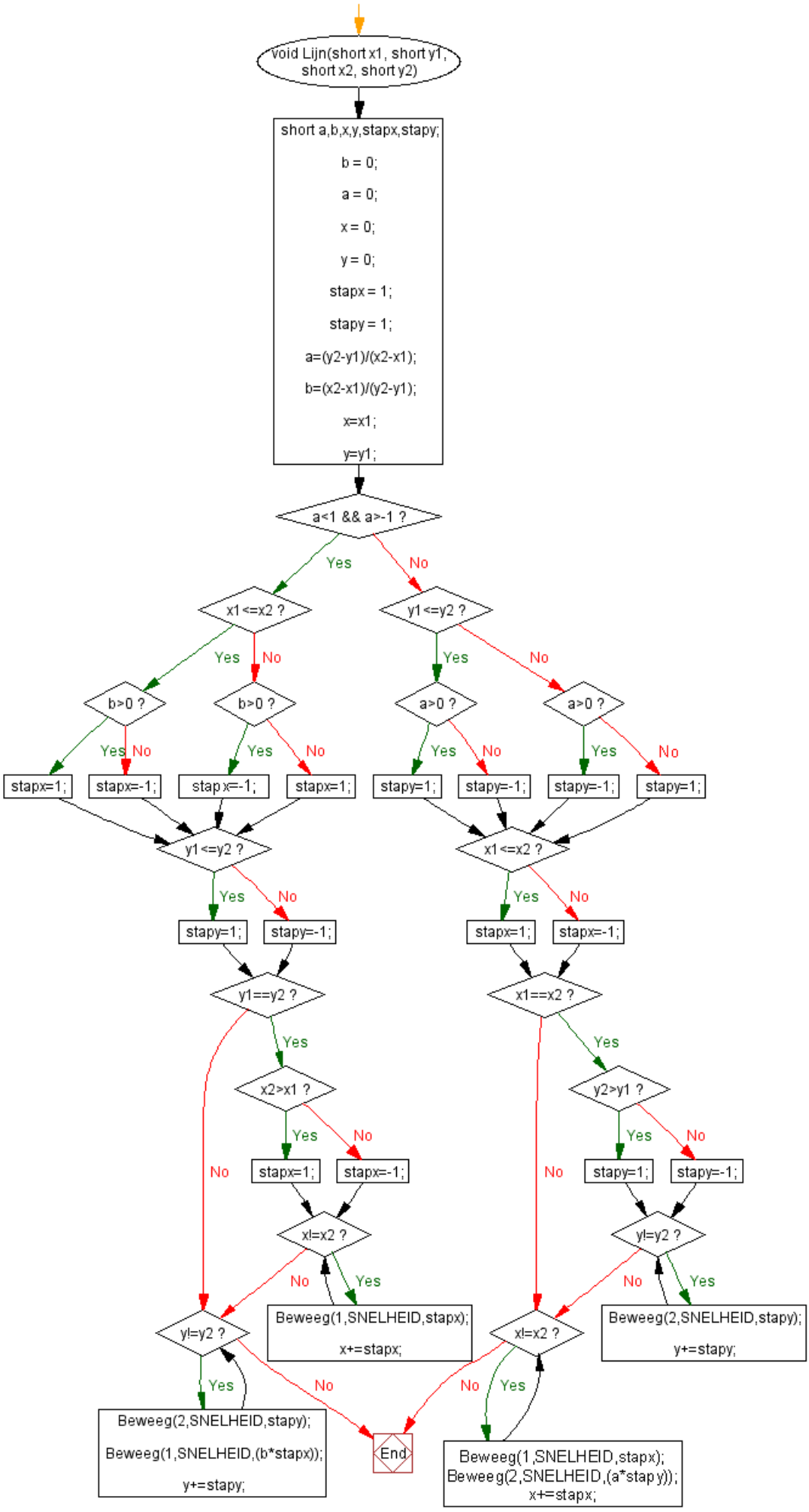


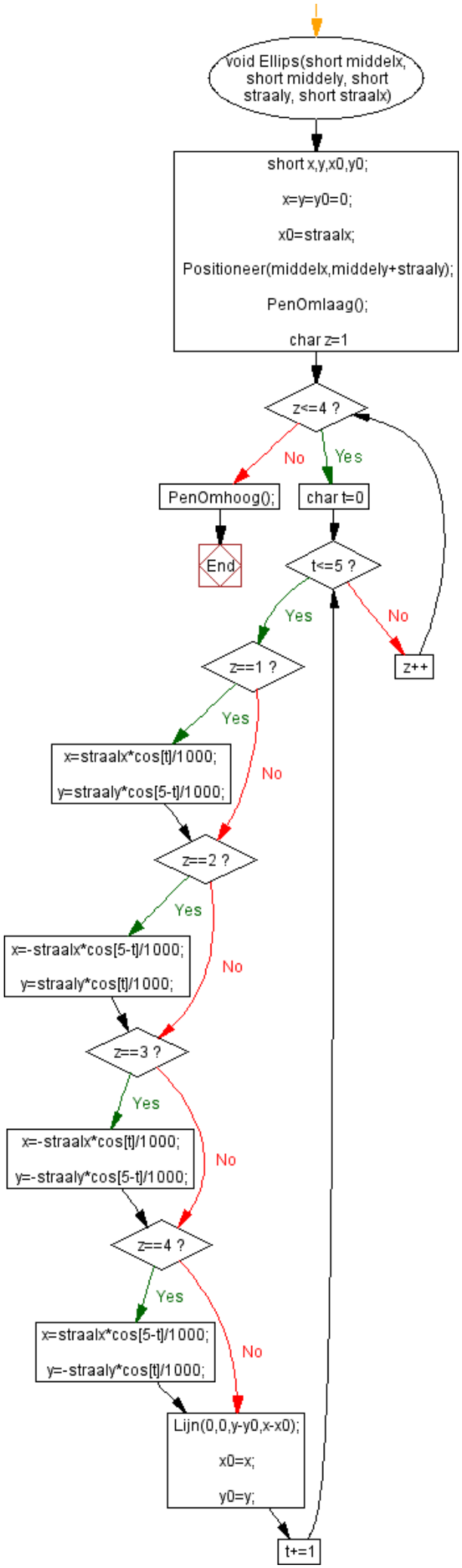







void Beweeg(char motor,unsigned short snelheid,short aantal)
short index,teller;
index = 0;
teller = 0;
aantal<0 ?
Yes
No
teller=aantal*(-1);
teller=aantal;
short i=0
i<teller ?
No
Yes
End
char j=0
j<4 ?
Yes
No
i++
aantal < 0 ?
No
Yes
motor==1 ?
motor==1 ?
No
Yes
No
Yes
index=7-j;
index=3-j;
index=4+j;
index=j;
out[index]=high;
Delay(snelheid);
out[index]=low;
j++


void Lijn(short x1, short y1, short x2, short y2)
short a,b,x,y,stapx,stapy;
b = 0;
a = 0;
x = 0;
y = 0;
stapx = 1;
stapy = 1;
a=(y2-y1)/(x2-x1);
b=(x2-x1)/(y2-y1);
x=x1;
y=y1;
a<1 && a>-1 ?
Yes
No
x1<=x2 ?
y1<=y2 ?
b>0 ?
a>0 ?
stapx=1;
stapx=-1;
stapy=1;
stapy=-1;
y1<=y2 ?
x1<=x2 ?
y1==y2 ?
x1==x2 ?
x2>x1 ?
y2>y1 ?
x!=x2 ?
y!=y2 ?
Beweeg(1,SNELHEID,stapx);
x+=stapx;
Beweeg(2,SNELHEID,stapy);
y+=stapy;
Beweeg(2,SNELHEID,stapy);
Beweeg(1,SNELHEID,(b*stapx));
y+=stapy;
Beweeg(1,SNELHEID,stapx);
Beweeg(2,SNELHEID,(a*stapy));
x+=stapx;
End


void Ellips(short middelx, short middely, short straaly, short straalx)
short x,y,x0,y0;
x=y=y0=0;
x0=straalx;
Positioneer(middelx,middely+straaly);
PenOmlaag();
char z=1
z<=4 ?
No
Yes
PenOmhoog();
char t=0
End
t<=5 ?
Yes
No
z++
z==1 ?
Yes
No
x=straalx*cos[t]/1000;
y=straaly*cos[5-t]/1000;
z==2 ?
Yes
No
x=-straalx*cos[5-t]/1000;
y=straaly*cos[t]/1000;
z==3 ?
Yes
No
x=-straalx*cos[t]/1000;
y=-straaly*cos[5-t]/1000;
z==4 ?
Yes
No
x=straalx*cos[5-t]/1000;
y=-straaly*cos[t]/1000;
Lijn(0,0,y-y0,x-x0);
x0=x;
y0=y;
t+=1

4.4 Volledige broncode

```
#include <libhc11.h>

// ***************************** DECLARATIES ******************************

// poorten benoemen
outputdevice out[10];
inputdevice penSensor,xSensor,ySensor;

// variabelen benoemen
short cos[6];
const unsigned short SNELHEID = 3000;

// grensconstanten
const unsigned short XGRENS = 380;
const unsigned short YGRENS = 249;

// ### INIT ###
// initialiseert de nodige variabelen
void Init()
{
    // poortvariabelen aan fysieke poorten binden
    for (char j=1;j<=10;j++)
    {
        out[j-1].attach(outport[j]);
    }

    penSensor.attach(inport[16]);
    xSensor.attach(inport[15]);
    ySensor.attach(inport[14]);

    // voorgeprogrammeerde cosinuswaarden
    cos[0] = 1000;
    cos[1] = 951;
    cos[2] = 809;
    cos[3] = 588;
    cos[4] = 309;
    cos[5] = 0;
}

// ***************************** METHODES *******************************

// ### DELAY ###
// Last een pauze in in het programma. Dit wordt onder andere
// gebruikt in Beweeg
void Delay(unsigned short snelheid)
{
    for(unsigned short i=0; i < snelheid; i++) {}
}

// ### BEWEEG ###
// laat de stappenmotoren bewegen
void Beweeg(char motor,unsigned short snelheid,short aantal)
{
```

```
    short index,teller;
    index = 0; teller = 0;

    if (aantal<0) teller=aantal*(-1); // teller moet altijd positief zijn
    else teller=aantal;

    for (short i=0;i<teller;i++) // herhaal <aantal> keren
    {
        for (char j=0;j<4;j++) // voor elk draadje van de stappenmotor
        {
            if (aantal < 0) // voorwaarts
            {
                if (motor==1) index=j; // X-motor
                else index=4+j;        // Y-motor
            }
            else // achterwaarts
            {
                if (motor==1) index=3-j; // X-motor
                else index=7-j; // Y-motor
            }

            // motor een stapje laten zetten
            out[index]=high;
            Delay(snelheid);
            out[index]=low;
        }
    }
}

// ### VRAAGCOORD ###
// vraagt begincoördinaten
void VraagCoord(short &x,short &y)
{
    int buffer;

    terminal << endl << "BeginX:";
    terminal >> buffer; // om iets in te lezen moet je een integer
                        // gebruiken, vandaar deze tijdelijke buffer
    x = (short)buffer;  // die hier omgezet wordt in char

    terminal << endl << "BeginY:";
    terminal >> buffer;
    y = (short)buffer;
}

// ### POSITIONEER ###
// beweegt de pen naar de gewenste positie
void Positioneer(short x,short y)
{
    Beweeg(1,SNELHEID,x);
    Beweeg(2,SNELHEID,y);
}
```

```
// ### RESET ###
// brengt bij het opstarten de pen naar de oorsprong
void Reset()
{
    // eerst de Y-as verplaatsen, zodanig dat het gewicht van het
    // wagentje al dicht bij de X-motor is. Zoniet beweegt de X-as
    // nogal moeizaam
    while (ySensor == low)
    {
        Beweeg(2,SNELHEID,-1);
    }

    while (xSensor == low)
    {
        Beweeg(1,SNELHEID,-1);
    }
}

// ### PENOMHOOG ###
// brengt de pen omhoog
void PenOmhoog()
{
    // pen laten zakken
    out[9] = high;
    out[8] = low;

    for(char i=0;i<8;i++){
        Delay(50000); // zolang de delay duurt gaat de pen omhoog.
    }                 // Dit gebeurt in verschillende stappen van
                      // 50000 omdat Delay() enkel unsigned short
                      // aanvaardt (minder geheugengebruik)

    // pen stilzetten
    out[9] = low;
}

// ### PENOMLAAG ###
// brengt pen omlaag
void PenOmlaag()
{
    while (penSensor == low)
    {
        out[9] = low;  // zolang de pen niet beneden is blijft hij dalen
        out[8] = high; // hiervoor gebruiken we de lichtsensor
    }

    out[9] = high;
}

// ### RECHTHOEK ###
// tekent rechthoek met opgegeven begincoördinaten, breedte en hoogte
void Rechthoek(short x, short y, short a, short b)
{
```

```
    Positioneer(x,y); // naar de linkeronderhoek van de rechthoek
    PenOmlaag();

    Beweeg(1,SNELHEID,a);
    Beweeg(2,SNELHEID,b);
    Beweeg(1,SNELHEID,-a-2);
    Beweeg(2,SNELHEID,-b-2);

    PenOmhoog();
}

// ### LIJN ###
// tekent een lijn in willekeurige richting
void Lijn(short x1, short y1, short x2, short y2)
{
    short a,b,x,y,stapx,stapy;
    b = 0;
    a = 0;
    x = 0;
    y = 0;
    stapx = 1;
    stapy = 1;

    // parameters van de rechte berekenen
    a=(y2-y1)/(x2-x1); // richtingscoefficienten voor steile rechtes
    b=(x2-x1)/(y2-y1); // omgekeerde van de richtingscoefficient
    x=x1;
    y=y1;

    if (a<1 && a>-1) { // zwak steigende of zwak dalende lijnen

        // eerst bepalen of x en y vooruit of achteruit moeten bewegen
        if (x1<=x2){
            if (b>0)stapx=1;
            else stapx=-1;
        }
        else
        {
            if(b>0)stapx=-1;
            else stapx=1;
        }

        if (y1<=y2) stapy=1;
        else stapy=-1;

        // het geval van lijnen evenwijdig met de X-as apart behandelen
        if (y1==y2)
        {
            if(x2>x1)stapx=1;
            else stapx=-1;

            while(x!=x2) // zolang we nog niet in het eindpunt zijn
            {
```

```
                Beweeg(1,SNELHEID,stapx);
                x+=stapx;
            }
        }

        // het algemene geval
        while (y!=y2) // zolang we nog niet in het eindpunt zijn
        {
            Beweeg(2,SNELHEID,stapy);
            Beweeg(1,SNELHEID,(b*stapx));
            y+=stapy;
        }
    }

    else // sterk stijgende of sterk dalende lijnen
    {
        // eerst bepalen of x en y vooruit of achteruit moeten bewegen
        if (y1<=y2)
        {
            if (a>0)stapy=1;
            else stapy=-1;
        }
        else
        {
            if(a>0)stapy=-1;
            else stapy=1;
        }

        if (x1<=x2) stapx=1;
        else stapx=-1;

        // het geval van lijnen evenwijdig met de Y-as apart behandelen
        if (x1==x2)
        {
            if(y2>y1)stapy=1;
            else stapy=-1;

            while(y!=y2) // zolang we nog niet in het eindpunt zijn
            {
                Beweeg(2,SNELHEID,stapy);
                y+=stapy;
            }
        }

        // het algemene geval
        while (x!=x2) // zolang we nog niet in het eindpunt zijn
        {
            Beweeg(1,SNELHEID,stapx);
            Beweeg(2,SNELHEID,(a*stapy));
            x+=stapx;
        }
    }
}
```

```
// ### CIRKEL + ELLIPS ###
// cirkel is gelijk aan ellips met straalx en straaly gelijk
void Ellips(short middelx, short middely, short straaly, short straalx)
{
    short x,y,x0,y0;
    x=y=y0=0;
    x0=straalx;

    Positioneer(middelx,middely+straaly); // naar het beginpunt
    PenOmlaag();

    for (char z=1;z<=4;z++) // voor elk kwadrant
    {
        for (char t=0;t<=5;t+=1) // per kwadrant in 6 stappen
        {
            if(z==1) // eerste kwadrant
            {
                x=straalx*cos[t]/1000;
                y=straaly*cos[5-t]/1000;
            }
            if(z==2) // tweede kwadrant
            {
                x=-straalx*cos[5-t]/1000;
                y=straaly*cos[t]/1000;
            }
            if(z==3) // derde kwadrant
            {
                x=-straalx*cos[t]/1000;
                y=-straaly*cos[5-t]/1000;
            }
            if(z==4) // vierde kwadrant
            {
                x=straalx*cos[5-t]/1000;
                y=-straaly*cos[t]/1000;
            }

            Lijn(0,0,y-y0,x-x0); // een lijn trekken tussen het
                                 // vorige punt en het nieuwe
            x0=x;
            y0=y;
        }
    }

    PenOmhoog();
}

// ### KEUZEMENU ###
void KeuzeMenu()
{
    int buffer;
    char keuze;
    short x,y,a,b;
```

```
    x=y=a=b=0;

    while(a!=5)
    {
        terminal << "Kies uit:" << endl;
        terminal << "1.Rechthoek" << endl;
        terminal << "2.Lijn"<< endl;
        terminal << "3.Cirkel"<< endl;
        terminal << "4.Ellips"<< endl;
        terminal << "5.Afsluiten"<< endl;
        terminal >> buffer;
        terminal << endl;

        keuze = (char)buffer;

        // RECHTHOEK
        if (keuze == 1)
        {
            VraagCoord(x,y);

            terminal << "Breedte:";
            terminal >> buffer;
            a = (short)buffer;

            terminal << "Hoogte:";
            terminal >> buffer;
            b = (short)buffer;

            if (x<XGRENS && y<YGRENS && x>=0 && y>=0 &&
                x+a<XGRENS && a>0 && y+b<YGRENS && b>0)
            {
                Rechthoek(x,y,a,b);
                Reset();
            }
        }

        // LIJN
        if (keuze == 2)
        {
            VraagCoord(x,y);

            terminal<<"EindX:";
            terminal >> buffer;
            a = (short)buffer;
            terminal<<endl;

            terminal<<"EindY:";
            terminal >> buffer;
            b = (short)buffer;
            terminal<<endl;

            if (x<XGRENS && y<YGRENS && x>=0 && y>=0 &&
                a<XGRENS && a>=0 && b<YGRENS && b>=0)
```

```
            {
                Positioneer(x,y);
                PenOmlaag();
                Lijn(x,y,a,b);
                PenOmhoog();
                Reset();
            }
        }

        // CIRKEL
        if (keuze == 3)
        {
            VraagCoord(x,y);

            terminal<<"Straal:";
            terminal >> buffer;
            a = (short)buffer;
            terminal<<endl;

            if (x<XGRENS && x>0 && y<YGRENS && y>0 &&
                x+a<XGRENS && x-a>0 && y+a<YGRENS && y-a>0)
            {
                Ellips(x,y,a,a);
                Reset();
            }
        }

        // ELLIPS
        if (keuze == 4)
        {
            VraagCoord(x,y);

            terminal<<"HalveAsX:";
            terminal >> buffer;
            b = (short)buffer;
            terminal<<endl;

            terminal<<"HalveAsY:";
            terminal >> buffer;
            a = (short)buffer;
            terminal<<endl;

            if (x<XGRENS && x >0 && y<YGRENS && y>0 &&
                x+b<XGRENS && x-b>0 && y+a<YGRENS && y-a>0)
            {
                Ellips(x,y,b,a);
                Reset();
            }
        }
    }
}
```

```
// ************************** HOOFDPROGRAMMA ***************************
void start()
{
    Init(); // een aantal variabelen initialiseren

    // Bij de start van het programma de pen omhoog zetten
    if (penSensor == high)
    {
        PenOmhoog();
    }

    Reset(); // de pen naar de oorsprong verplaatsen (kalibreren)
    KeuzeMenu(); // het keuzemenu tonen
}
```

```
#include <libhc11.h>

// ***************************** DECLARATIES ******************************

// poorten benoemen
outputdevice out[10];
inputdevice penSensor,xSensor,ySensor;
inputdevice bovenKnop,onderKnop,linksKnop,rechtsKnop,penKnop, uitSensor;

// variabelen benoemen
short cos[6];
short buffer, x, y, vorigex, vorigey;
char laatsteActie;
const unsigned short SNELHEID = 4000;

// grensconstanten
const short XGRENS = 380;
const short YGRENS = 250;

// ### INIT ###
// initialiseert de nodige variabelen
void Init()
{
    laatsteActie = 1;
    buffer = 1;
    x = 0;
    y = 0;

    // poortvariabelen aan fysieke poorten binden
    for (char j=1;j<=10;j++)
    {
        out[j-1].attach(outport[j]);
    }

    penSensor.attach(inport[16]);
    xSensor.attach(inport[15]);
    ySensor.attach(inport[14]);

    // controller-schakelaars
    bovenKnop.attach(inport[1]);
    onderKnop.attach(inport[3]);
    linksKnop.attach(inport[4]);
    rechtsKnop.attach(inport[2]);
    penKnop.attach(inport[5]);
    uitSensor.attach(inport[6]);
}

// ***************************** METHODES *******************************

// ### DELAY ###
// Last een pauze in in het programma. Dit wordt onder andere
// gebruikt in Beweeg
void Delay(unsigned short snelheid)
```

```
{
    for(unsigned short i=0; i < snelheid; i++) {}
}

// ### BEWEEG ###
// laat de stappenmotoren bewegen
void Beweeg(char motor,unsigned short snelheid,short aantal)
{
    short index,teller;
    index = 0; teller = 0;

    if (aantal<0) teller=aantal*(-1); // teller moet altijd positief zijn
    else teller=aantal;

    for (short i=0;i<teller;i++) // herhaal <aantal> keren
    {
        for (char j=0;j<4;j++) // voor elk draadje van de stappenmotor
        {
            if (aantal < 0) // voorwaarts
            {
                if (motor==1) index=j; // X-motor
                else index=4+j;        // Y-motor
            }
            else // achterwaarts
            {
                if (motor==1) index=3-j; // X-motor
                else index=7-j; // Y-motor
            }

            // motor een stapje laten zetten
            out[index]=high;
            Delay(snelheid);
            out[index]=low;
        }
    }
}

// ### RESET ###
// brengt bij het opstarten de pen naar de oorsprong
void Reset()
{
    x = 0;
    y = 0;

    // eerst de Y-as verplaatsen, zodanig dat het gewicht van het
    // wagentje al dicht bij de X-motor is. Zoniet beweegt de X-as
    // nogal moeizaam
    while (ySensor == low)
    {
        Beweeg(2,SNELHEID,-1);
    }

    while (xSensor == low)
```

```
    {
        Beweeg(1,SNELHEID,-1);
    }
}

// ### PENOMHOOG ###
// brengt de pen omhoog
void PenOmhoog()
{
    // pen laten zakken
    out[9] = high;
    out[8] = low;

    for(char i=0;i<8;i++){
        Delay(50000); // zolang de delay duurt gaat de pen omhoog.
    }                 // Dit gebeurt in verschillende stappen van
                      // 50000 omdat Delay() enkel unsigned short
                      // aanvaardt (minder geheugengebruik)

    // pen stilzetten
    out[9] = low;
}

// ### PENOMLAAG ###
// brengt pen omlaag
void PenOmlaag()
{
    while (penSensor == low)
    {
        out[9] = low;  // zolang de pen niet beneden is blijft hij dalen
        out[8] = high; // hiervoor gebruiken we de lichtsensor
    }

    out[9] = high;
}

// ### PEN ###
// speelt tussenpersoon tussen penKnop en PenOmhoog/PenOmlaag
void Pen(char &laatsteActie)
{
    if (laatsteActie==0) // de pen staat omlaag
    {
        PenOmhoog();
        laatsteActie=1;
    }
    else // de pen staat omhoog
    {
        PenOmlaag();
        laatsteActie=0;
    }
}

// ### CONTROL ###
```

```
// voert de handelingen uit die de gebruiker ingeeft via de controller
void Control(char &laatsteActie)
{
    //verplaatst de pen in de gewenste richting en past de coordinaten aan
    if (bovenKnop == high && x<XGRENS)
    {
        Beweeg(1,SNELHEID,1);
        vorigex = x;
        x = x+1;
    }

    if (rechtsKnop == high && y<YGRENS)
    {
        Beweeg(2,SNELHEID,1);
        vorigey = y;
        y = y+1;
    }

    if (onderKnop == high && x>0)
    {
        Beweeg(1,SNELHEID,-1);
        vorigex = x;
        x = x-1;
    }

    if (linksKnop == high && y>0)
    {
        Beweeg(2,SNELHEID,-1);
        vorigey = y;
        y = y-1;
    }

    if (penKnop==high)
    {
        Pen(laatsteActie);
    }
}

// *************************** HOOFDPROGRAMMA ****************************
void start()
{
    Init();

    if (penSensor == high)
    {
        PenOmhoog(); //brengt de pen omhoog moest deze nog beneden staan
    }

    //naar oorsprong bewegen
    Reset();

    //oneindige lus, word onderbroken wanneer de gebruiker de gepaste
handeling uitvoert
```

```
    //lichtsensor bedekken
    for(;;)
    {
        Control(laatsteActie);

        if (uitSensor == high)
        {
            terminal<<"Het programma wordt beeindigd."<<endl;
            if(penSensor == high)
            {
                Pen(laatsteActie);
            }
            Reset();
            break;
        }
    }
}
```