

Vakgroep SYSTeMS



Vakoverschrijdend project 2013-2014

Slingercontrole door voorwaartse sturing

Groep 3:

Simon Delva

Kilian Drubbel

Wouter Van Dael

Halewijn Wayenberg

Begeleiders:

Prof. Dr. Ir. Mia Loccufier

Prof. Dr. Ir. Alain Sarlette

Inhoudstafel

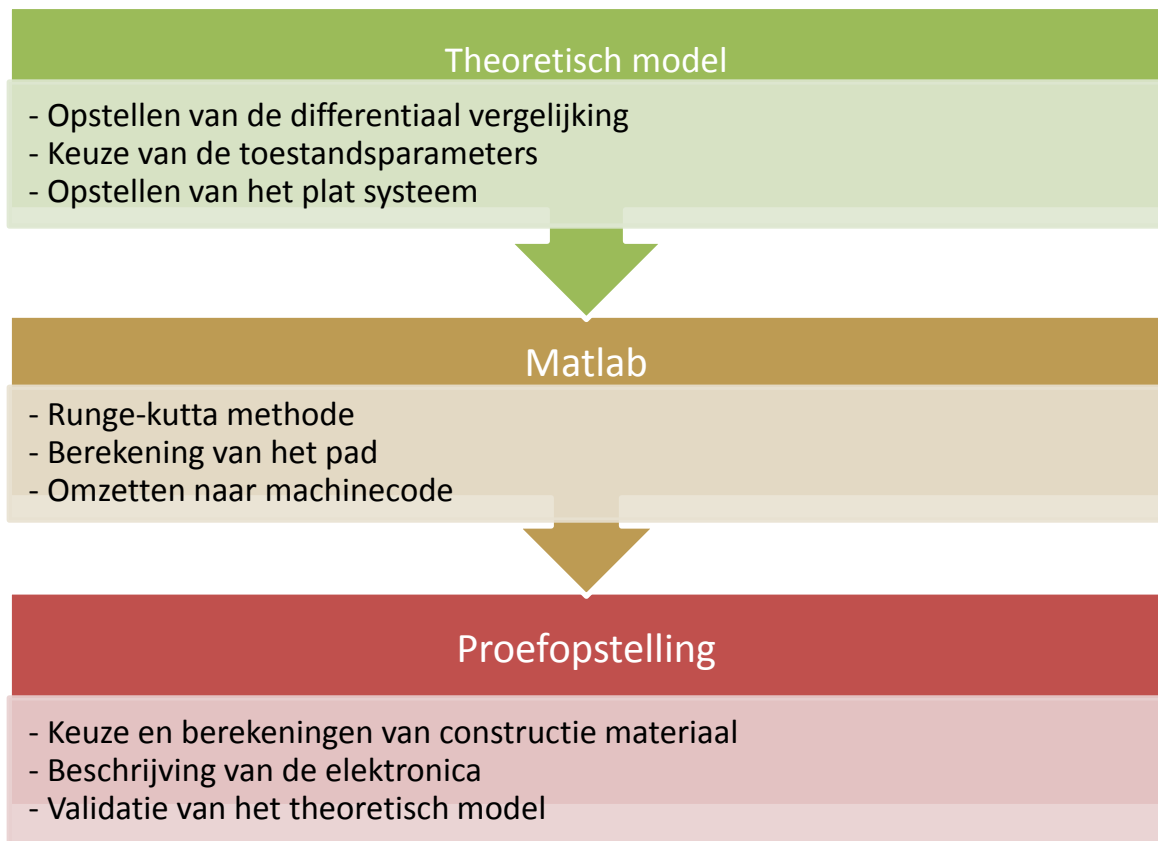
| | |
|---|----|
| Inleiding | 3 |
| Theoretisch model..... | 4 |
| Opstellen van de differentiaal vergelijkingen | 4 |
| Opstellen van het toestandsmodel | 7 |
| Opstellen van de platte output | 7 |
| Keuze van een geschikt pad | 8 |
| Interpolatie functie als pad | 10 |
| Proefopstelling | 12 |
| Mechanisch ontwerp en keuzes..... | 12 |
| Basisontwerp | 12 |
| Doorbuiging van het U-profiel..... | 14 |
| Elektronica..... | 15 |
| Stappenmotoren | 16 |
| De stepper driver..... | 17 |
| De microcontroller | 18 |
| De schakeling..... | 19 |
| Programma..... | 20 |
| Validatie van het model door de proefopstelling | 22 |
| Conclusie | 22 |
| Bijlages..... | 23 |

Inleiding

Een groot probleem bij kabelbanen is de slingering van de cabine, wat inhoudt dat de cabine in de langsrichting (langs de kabel), in de dwarsrichting of in de verticale richting ongecontroleerde bewegingen kan maken. De doelstelling van dit vakoverschrijdend project is om een controlestrategie op te stellen die de slingering bij kabelbanen binnen perken zal houden, dempen of neutraliseren. Er is ook gevraagd deze controlestrategie te testen met behulp van MATLAB en een proefopstelling te maken die steunt op het opgestelde model.

Er waren drie verschillende uitgangspunten: in de eerste plaats was er een feedbacklus met een actuator die een massa binnen in de cabine verplaatst, om zo de slingering te beperken op een actieve manier. Daarnaast was er een passieve methode die met een mechanische constructie een demping van slingering moest voorzien. De derde en laatste controlestrategie is zuiver theoretisch: door middel van inverse kinematica wordt een manier gezocht waarmee, aan de hand van wenswaarden voor de positie van de cabine, de versnelling van de kabelbaan geregeld wordt, en wel zo dat de cabine effectief deze positie zal aannemen. Wij hebben gekozen voor het geval waarbij we gebruik maken van inverse kinematica. We steunen hiervoor op een theoretische techniek omtrent platte systemen. Een plat systeem is een systeem waarbij men de inputparameters (versnelling van de kabel) kan schrijven in functie van de outputparameters (positie van de cabine) en zijn afgeleiden (snelheid & versnelling van de cabine). Op deze manier kunnen we de slinger een bepaald pad in de ruimte laten volgen en zo de slingering volledig sturen. Om de slingering volledig te kunnen sturen hebben we nood aan een versnellingsregeling van de motor en de lengteregeleing van de kabel waarmee de cabine aan de kabelbaan bevestigd is. Dit systeem is eigenlijk eerder van toepassing op bijvoorbeeld een torenkraan met slingercontrole.

In dit verslag zullen we initieel het theoretisch model van het plat systeem opstellen en dit daarna verifiëren door middel van een numerieke simulatie in MATLAB. Vervolgens bespreken we de proefopstelling. Welke keuzes we qua opstelling, materiaal en elektronica maken en waarom, zal kort worden uitgelegd. Ten slotte zullen we het theoretisch model proberen te valideren aan de hand van de proefopstelling, de verschillen tussen de werkelijkheid en het theoretisch model zullen we dan ook analyseren en verklaren.

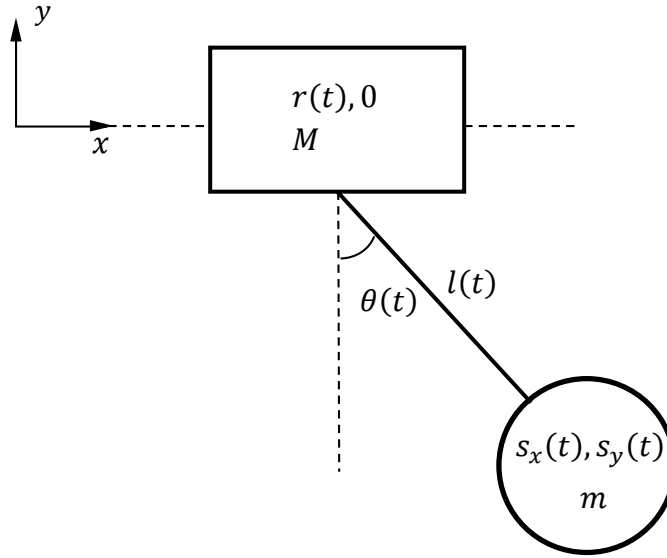


Figuur 1: een flowchart van het ontwerpproces

Theoretisch model

Opstellen van de differentiaalvergelijkingen

Voor ons theoretisch model gaan we uit van een kar die zich horizontaal over een starre rail voortbeweegt, waaraan een slinger hangt waarvan de lengte regelbaar is, we beschrijven eerst de kinematica van het gekozen mechanisme, vervolgens stellen we een toestandsmodel op en daarna beschouwen we het plat systeem. We gaan ervan uit dat de slinger een massa m heeft en de kar een massa M . Het koord tussen de slinger en de kar heeft een variabele lengte $l(t)$ en een slingerhoek $\theta(t)$, zie Figuur 2. Ter vereenvoudiging zullen we de luchtweerstand die op de slinger inwerkt en de wrijving tussen de kar en de rail verwaarlozen voor ons theoretisch model. We kiezen ons assenstelsel zo dat de rail zich langs de x-as bevindt.



Figuur 2: een schets van het gebruikte model

De tijdsafhankelijke variabelen stellen we voor zonder hun argument t te specificiëren, dus r , θ , l , s_x en s_y in plaats van $r(t)$, $\theta(t)$ enz. Vanaf hier zullen we vectoren ook aanduiden door gebruik te maken van vetgedrukte letters, en afgeleiden door een punt boven de variabele: $\frac{d}{dt}x = \dot{x}$.

$$\mathbf{p}_{kar} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

$$\mathbf{p}_{kar} = \mathbf{v}_{kar} = \begin{bmatrix} \dot{r} \\ 0 \end{bmatrix}$$

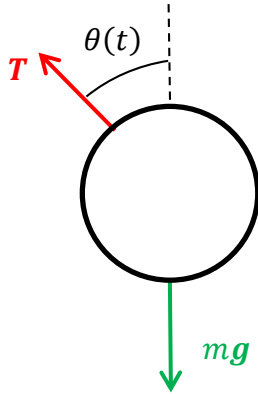
$$\mathbf{p}_{kar} = \mathbf{a}_{kar} = \begin{bmatrix} \ddot{r} \\ 0 \end{bmatrix}$$

$$\mathbf{p}_{slinger} = \begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} r + \sin(\theta) l \\ -\cos(\theta) l \end{bmatrix}$$

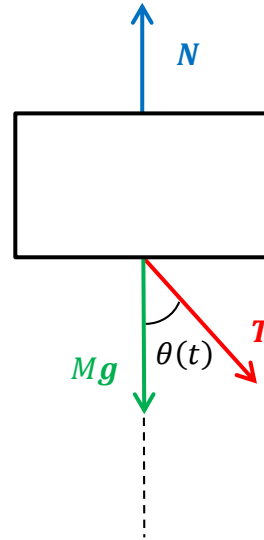
$$\mathbf{p}_{slinger} = \mathbf{v}_{slinger} = \begin{bmatrix} \dot{s}_x \\ \dot{s}_y \end{bmatrix} = \begin{bmatrix} \dot{r} + \dot{\theta} l \cos(\theta) + \sin(\theta) \dot{l} \\ \dot{\theta} l \sin(\theta) - \cos(\theta) \dot{l} \end{bmatrix}$$

$$\mathbf{p}_{slinger} = \mathbf{a}_{slinger} = \begin{bmatrix} \ddot{s}_x \\ \ddot{s}_y \end{bmatrix} = \begin{bmatrix} \ddot{r} - \sin(\theta) \ddot{\theta} l + \cos(\theta) \ddot{\theta} l + 2 \cos(\theta) \dot{\theta} \dot{l} + \sin(\theta) \ddot{l} \\ \cos(\theta) \ddot{\theta} l + \sin(\theta) \ddot{\theta} l + 2 \sin(\theta) \dot{\theta} \dot{l} - \cos(\theta) \ddot{l} \end{bmatrix}$$

Om de dynamica van het mechanisme te beschrijven, splitsen we het mechanisme op in twee vrijelichaamsdiagrammen (zie Figuur 3 en Figuur 4). Op elk van deze diagrammen zullen we de tweede wet van Newton toepassen, dit geeft ons de volgende vergelijkingen.



Figuur 3: vrijelichaamsdiagram van de slinger



Figuur 4: vrijelichaamsdiagram van het karretje

$$\sum \mathbf{F}_{slinger} = m \mathbf{a}_{slinger}$$

$$\begin{bmatrix} -T \sin(\theta) \\ T \cos(\theta) - mg \end{bmatrix} = \begin{bmatrix} m \ddot{s}_x \\ m \ddot{s}_y \end{bmatrix} = \begin{bmatrix} m(\ddot{r} - \sin(\theta) \ddot{\theta} l + \cos(\theta) \dot{\theta} l + 2 \cos(\theta) \dot{\theta} \dot{l} + \sin(\theta) \ddot{l}) \\ m(\cos(\theta) \ddot{\theta} l + \sin(\theta) \dot{\theta} l + 2 \sin(\theta) \dot{\theta} \dot{l} - \cos(\theta) \ddot{l}) \end{bmatrix}$$

$$\sum \mathbf{F}_{kar} = M \mathbf{a}_{kar}$$

$$\begin{bmatrix} T \sin(\theta) \\ -T \cos(\theta) - Mg + N \end{bmatrix} = \begin{bmatrix} M \ddot{r} \\ 0 \end{bmatrix}$$

Om de differentiaalvergelijking van het systeem te vinden moeten we de parameter T elimineren uit bovenstaande vergelijkingen. Dit kunnen we gemakkelijk doen door de eerste vergelijking om te vormen naar T .

$$T = \frac{-m(\ddot{r} - \sin(\theta) \ddot{\theta} l + \cos(\theta) \dot{\theta} l + 2 \cos(\theta) \dot{\theta} \dot{l} + \sin(\theta) \ddot{l})}{\sin(\theta)}$$

Na substitutie in de tweede vergelijking en vereenvoudigingen bekomen we de volgende uitdrukking.

$$\begin{aligned} m(\cos(\theta) \ddot{r} + \ddot{\theta} l + 2 \dot{\theta} \dot{l} + g \sin(\theta)) &= 0 \\ \Leftrightarrow l \cos(\theta) \ddot{r} + \frac{d}{dt}(l^2 \dot{\theta}) + l g \sin(\theta) &= 0 \end{aligned}$$

We merken op dat bovenstaande differentiaalvergelijking niet lineair is.

Het tweede vrijelichaamsdiagram (Figuur 4) kunnen we gebruiken om de spankracht in het touw en de normaalkracht op de kar te bepalen. Als we dit stelsel uitwerken bekomen we de volgende vergelijkingen. Deze krachten kunnen reeds een eerste goede schatting geven voor onze materiaalkeuze.

$$N = \frac{M(g \sin(\theta) + \cos(\theta) \ddot{r})}{\sin(\theta)} \quad T = \frac{M \ddot{r}}{\sin(\theta)}$$

Opstellen van het toestandsmodel

De regelbare inputparameters van het mechanisme zijn de lengte van het koord en de versnelling van de kar. We definiëren momenteel nog geen outputparameters vermits we later gebruik zullen maken van platte outputparameters.

De inputparameters zijn: $u_1 = \ddot{r}(t)$, $u_2 = l(t)$.

Voor de toestandsparameters kiezen we de volgende variabelen, vermits we met een vierde orde systeem zitten moeten we ook vier toestandsparameters definiëren. Met behulp van de tweede wet van Newton en de differentiaalvergelijking leiden we gemakkelijk het niet-lineaire toestandsmodel af.

$$\begin{cases} x_1 = \theta \\ x_2 = l^2 \dot{\theta} \\ x_3 = r \\ x_4 = \dot{r} \end{cases} \Rightarrow \begin{cases} \dot{x}_1 = \frac{x_2}{u_2} \\ \dot{x}_2 = u_2 g \sin(x_1) - u_1 u_2 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = u_1 \end{cases}$$

Opstellen van de platte output

Om eerst een beeld te geven wat platte output is, geven we initieel een algemene definitie die we daarna zullen toepassen op ons systeem. De algemene definitie luidt als volgt.

Een niet lineair systeem S

$$S = \left\{ \begin{array}{l} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{u}(t) \in \mathbb{R}^m \\ \mathbf{x}(t) \in \mathbb{R}^n \end{array} \right\}$$

is een plat systeem als er een output $\mathbf{y}(t) = (y_1(t), \dots, y_m(t))$ kan gevonden worden, die voldoet aan de volgende voorwaarden:

- De outputparameters y_i met $i = 1, \dots, m$ kunnen uitgedrukt worden in functie van de toestandsparameters x_i met $i = 1, \dots, n$ en van de inputparameters u_i met $i = 1, \dots, m$ en een eindig aantal van de afgeleiden naar de tijd $u_i^{(k)}$ met $k = 1, \dots, \alpha$
- De toestandsparameters x_i met $i = 1, \dots, n$ en de inputparameters u_i met $i = 1, \dots, m$ kunnen uitgedrukt worden in functie van de platte outputparameters y_i met $i = 1, \dots, m$ en een eindig aantal afgeleiden naar de tijd $y_i^{(k)}$ met $k = 1, \dots, \gamma$
- De componenten van $\mathbf{y}(t)$ zijn differentiaal onafhankelijk van elkaar, dit wil zeggen dat ze niet voldoen aan een (differentiaal)vergelijking van de vorm $\phi(\mathbf{y}, \dot{\mathbf{y}}, \dots, \mathbf{y}^{(\gamma)}) = 0$

Voor dit project zijn we geïnteresseerd in de gevolgde weg van de slinger, we kiezen dus als platte-outputparameters de coördinaten van de slinger, s_x en s_y . We willen nu de inputparameters en de toestandsparameters bepalen in functie van s_x en s_y en hun afgeleiden naar de tijd. We gaan als volgt te werk.

Uit de kinematica van het mechanisme volgen de uitdrukkingen voor de platte-outputparameters, we verifiëren ook direct dat deze vergelijkingen voldoen aan de eerste en de derde voorwaarde voor een plat systeem.

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} r + l \sin(\theta) \\ -l \cos(\theta) \end{bmatrix} = \begin{bmatrix} x_3 + u_2 \sin(x_1) \\ -u_2 \cos(x_1) \end{bmatrix}$$

De tweede wet van Newton levert ons de volgende vergelijkingen op, deze kunnen we gebruiken na eliminatie van de parameter T . Waaruit zal blijken dat het systeem kan voldoen aan de tweede voorwaarde voor een plat systeem.

$$\begin{bmatrix} m\ddot{s}_x \\ m\ddot{s}_y \end{bmatrix} = \begin{bmatrix} -T \sin(\theta) \\ T \cos(\theta) - mg \end{bmatrix}$$

Hier uit halen we dat:

$$\begin{aligned} \begin{bmatrix} m\ddot{s}_x \\ m\ddot{s}_y + mg \end{bmatrix} &= \begin{bmatrix} -T \sin(\theta) \\ T \cos(\theta) \end{bmatrix} \\ \frac{m\ddot{s}_x}{m\ddot{s}_y + mg} &= -\frac{T \sin(\theta)}{T \cos(\theta)} \\ \frac{\ddot{s}_x}{\ddot{s}_y + g} &= -\tan(\theta) \end{aligned}$$

Na wat omvormen en na gebruik te maken van de kinematica van de slinger bekomen we de volgende uitdrukkingen:

$$\begin{aligned} r = x_3 &= s_x - \frac{\ddot{s}_x s_y}{\ddot{s}_y + g} \\ l = u_2 &= \sqrt{(s_y)^2 + \left(\frac{\ddot{s}_x s_y}{\ddot{s}_y + g}\right)^2} \\ \theta = x_1 &= \arcsin\left(\frac{\ddot{s}_x s_y}{\ddot{s}_y + g} / l\right) \end{aligned}$$

We merken nu al op dat de positie r (gemeten langs de rail) van de kar afhankelijk is van de tweede afgeleiden van de positieparameters (s_x en s_y) van de slinger, dus zal de versnelling van de kar (wat de inputparameter u_1 is) afhankelijk zijn van de vierde afgeleide. Een eerste extra voorwaarde voor ons geval is dat de functies viermaal continu afleidbaar moet zijn, de tweede extra voorwaarde volgt uit het feit dat we vertrekken uit een steady-state toestand (= eigenschap platte output). De eerste vier afgeleiden van de platte outputparameters moeten dus allen gelijk zijn aan nul in de begintoestand.

Keuze van een geschikt pad

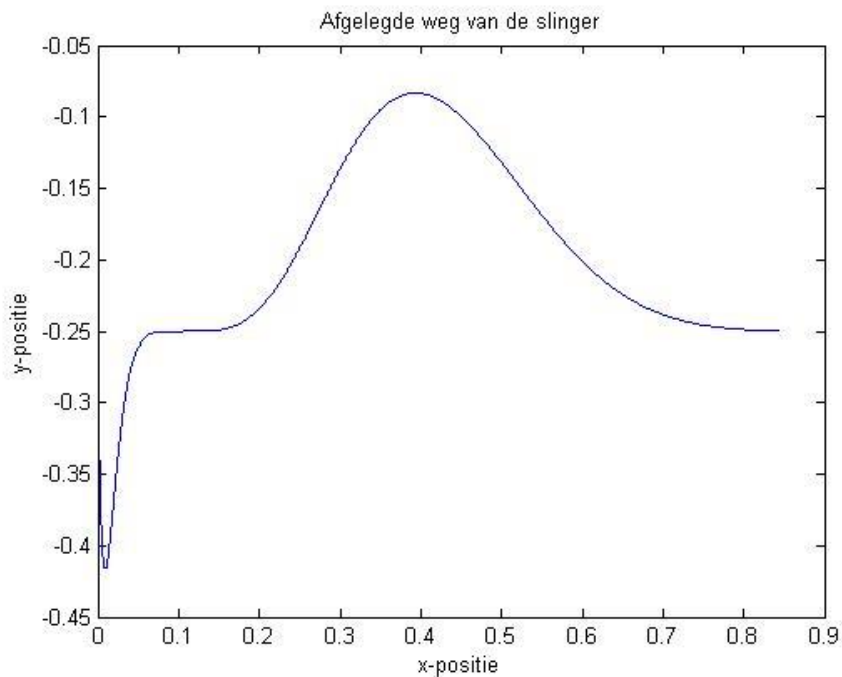
In theorie bestaan er oneindig veel geschikte paden om deze methode op toe te passen, zolang de voorwaarden van de inverse kinematica maar voldaan zijn. In dit project beperken we ons echter tot het uitvoeren van één enkel pad. Proefondervindelijk zijn we tot onderstaand pad gekomen.

$$\mathbf{p}_{slinger} = \begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \left(\arctan\left(\frac{t}{D}\right) - \left(\frac{t}{D}\right) + \left(\frac{t}{D}\right)^3\right) \cdot A \\ \left(-2 \sin\left(\frac{t}{D}\right)^5\right) \cdot B - C \end{bmatrix}$$

Dit pad voldoet aan de voorwaarden die hierboven in de algemene uitleg over inverse kinematica gesteld zijn, onafhankelijk van de schaalfactoren A , B , C en D . Bij het kiezen van deze schaalfactoren

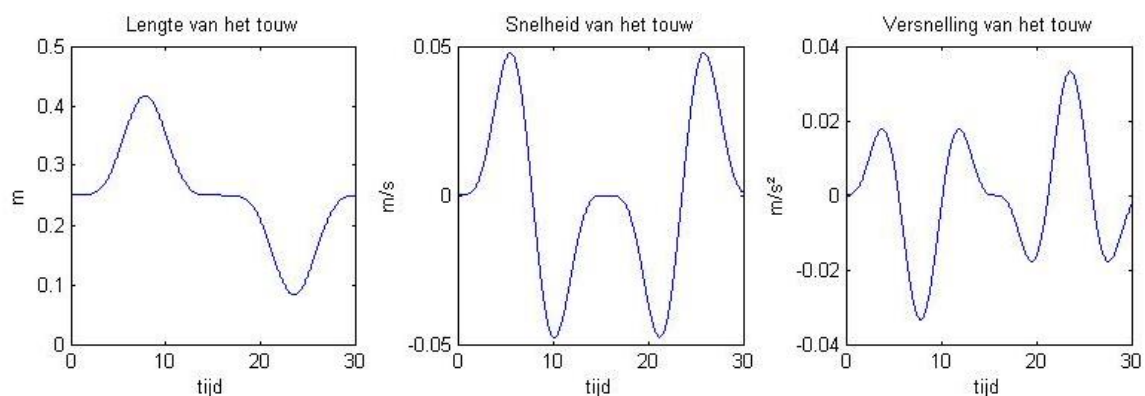
moet echter rekening gehouden worden met de maximale versnelling die de motoren kunnen leveren en het feit dat de massa geen grotere versnelling naar beneden kan ondergaan dan de valversnelling van $9,81\text{m/s}^2$.

De schaalfactoren kiezen we zo dat ze voldoen aan de proefopstelling die we later zullen beschrijven, we willen namelijk dat de maximale uitwijking van de slinger binnen de 0,25 meter blijft de rail heeft een lengte van 0,8 meter en we stellen dat het tijdsinterval gelijk is aan 30 seconden. Om aan deze parameters te voldoen werken nu verder met de waarden $A = 1/80$, $B = 1/6$, $C = 1/4$, $D = 5$. Het gekozen pad ziet er met bijhorende schaalfactoren uit zoals weergegeven in Figuur 5.

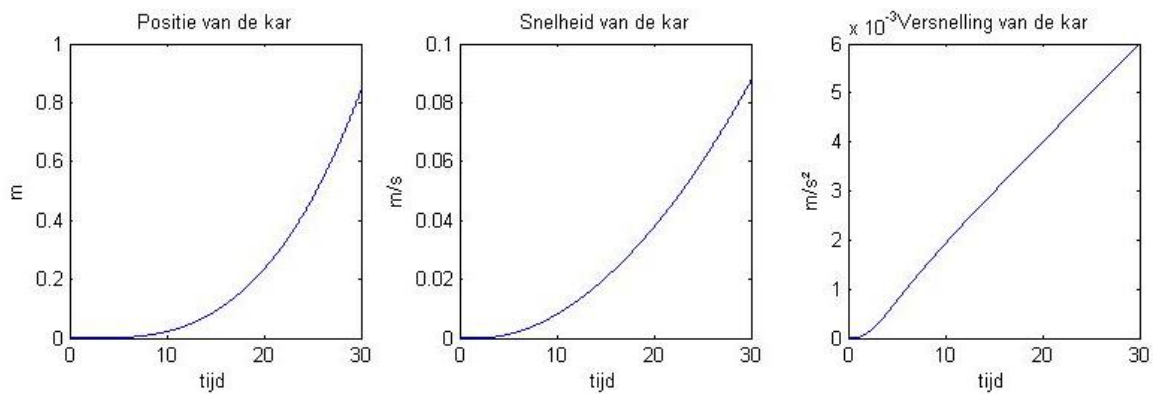


Figuur 5: de afgelegde weg van de cabine (slingerende massa)

Voor de gekozen waarden bekommen we de grafieken zoals weergegeven in Figuur 6 en Figuur 7 voor de positie/lengte, snelheid en versnelling van de kar en het touw.



Figuur 6: lengte van het touw en haar eerste twee afgeleiden



Figuur 7: positie, snelheid en versnelling van het karretje

We zien dat zowel de versnelling voor de kar en de slinger binnen haalbare waarden blijven. We zullen deze parameters voor lengte en versnelling dus ook toepassen voor onze proefopstelling.

Interpolatiefunctie als pad

Bovenstaande methode is echter niet geschikt om toe te passen in de praktijk, omdat de versnelling van de kar snel zal oplopen en de massa na het afleggen van bovenstaande paden niet in stilstand is. Een betere oplossing is echter het verplaatsen van de massa van één stilstaande positie naar een andere stilstaande positie op dezelfde hoogte, met deze methode zullen we echter de lengte van de slinger niet meer als variabele beschouwen we zullen dus enkel de versnelling van de motor sturen.

$$l(t) = c^{te} \Rightarrow s_y(t) \approx c^{te}$$

Om de interpolatie functie te kunnen opstellen maken we gebruik van het feit dat het plat systeem aan de volgende begin- en eindvoorwaarden moet voldoen als we in rust willen vertrekken en eindigen.

$$\begin{array}{ll} s_x(t_0) = x_0 & s_x(t_1) = x_1 \\ \dot{s}_x(t_0) = 0 & \dot{s}_x(t_1) = 0 \\ \ddot{s}_x(t_0) = 0 & \ddot{s}_x(t_1) = 0 \\ \dddot{s}_x(t_0) = 0 & \dddot{s}_x(t_1) = 0 \\ \dots s_x(t_0) = 0 & \dots s_x(t_1) = 0 \end{array}$$

De interpolatiefunctie moet dus voldoen aan vijf beginvoorwaarden en vijf eindvoorwaarden, ze zal dus minimum van de negende graad zijn. De berekeningen van deze interpolatiefunctie laten we achterwege in dit verslag, een mogelijke interpolatie functie is de volgende:

$$s_x(t) = x_0 + (x_1 - x_0)\tau^5(126 - 420\tau + 540\tau^2 - 315\tau^3 + 70\tau^4)$$

$$\text{met } \tau = \frac{t-t_0}{t_1-t_0}$$

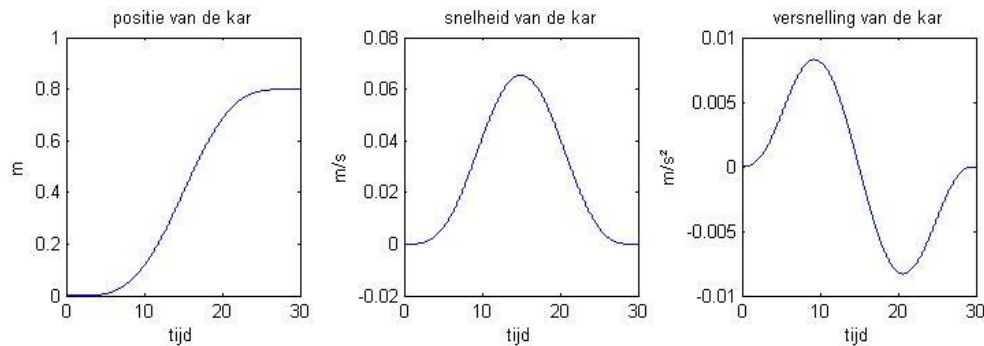
Voor onze proefopstelling zullen we stellen dat $t_0 = 0s$, $t_1 = 30s$, $x_0 = 0m$ en $x_1 = 0,8m$. Met deze waarden bekomen we volgende functie:

$$s_x(t) = \frac{(0,8)t^5}{30^5} \left(126 - \frac{420t}{30} + \frac{540t^2}{30^2} - \frac{315t^3}{30^3} + \frac{70t^4}{30^4} \right)$$

Met behulp van deze functie kunnen we nu de positie, snelheid en versnelling van de kar bepalen door middel van de afgeleide formules, we merken nu wel op dat de afgeleiden naar de y-positie van het pad zich zullen herleiden naar nul. De formule voor de positie ziet er dus uit als volgt:

$$r(t) = s_x(t) - \frac{\dot{s}_x(t)s_y}{g}$$

We bekomen de grafieken zoals weergegeven in Figuur 8 voor de positie, snelheid en versnelling van de kar.



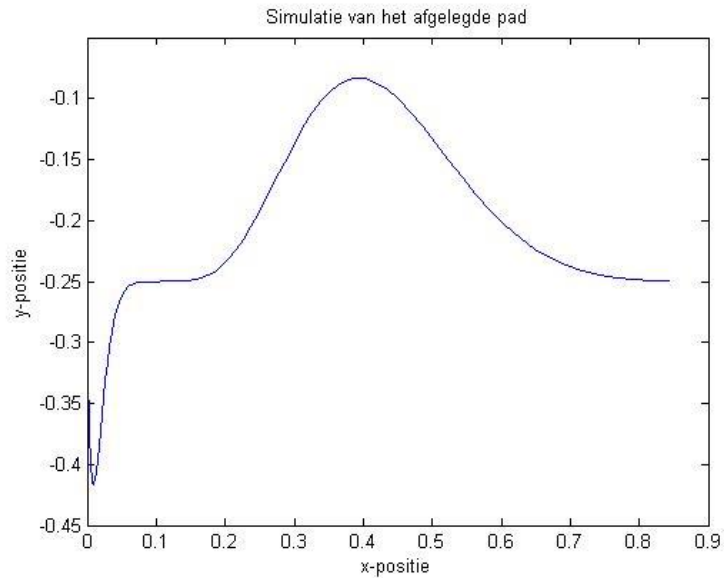
Figuur 8: positie, snelheid en versnelling van het karretje als de lengte van het touw constant gehouden wordt

Dit is wel een nuttig pad voor een hijskraan: een mooi verloop van de snelheid, met niet al te grote versnellingen.

Verificatie door numerieke simulatie in MATLAB

Door middel van MATLAB simuleren we het plat systeem door gebruik te maken van het toestandsmodel dat eerder werd opgesteld. We lossen het toestandsmodel op door middel van een numerieke methode: de methode van Runge-Kutta, deze is standaard beschikbaar in MATLAB. Deze methode laat ons toe de toestandsparameters te bepalen op ieder ogenblik van de simulatie, we kiezen hiervoor een tijdsstap van $50ms$. De MATLAB-code kan men terugvinden in de bijlage van dit verslag.

Zoals waarneembaar op Figuur 9 is de numerieke oplossing van het gekozen pad bijna exact gelijk aan het gekozen pad. Deze simulatie uitvoeren op het interpolatiepad geeft ons uiteraard een horizontale rechte vermits we uitgaan van constante slingerlengte, op de animatie zien we ook dat de slingering van de massa stopt in het eindpunt. We kunnen dus uit onze simulatie besluiten dat ons theoretisch model werkt voor beide gevallen en dus toepasbaar zou moeten zijn in de praktijk.



Figuur 9: posities van de slinger berekend in een simulatie mbv MATLAB

Proefopstelling

Mechanisch ontwerp en keuzes

Basisontwerp

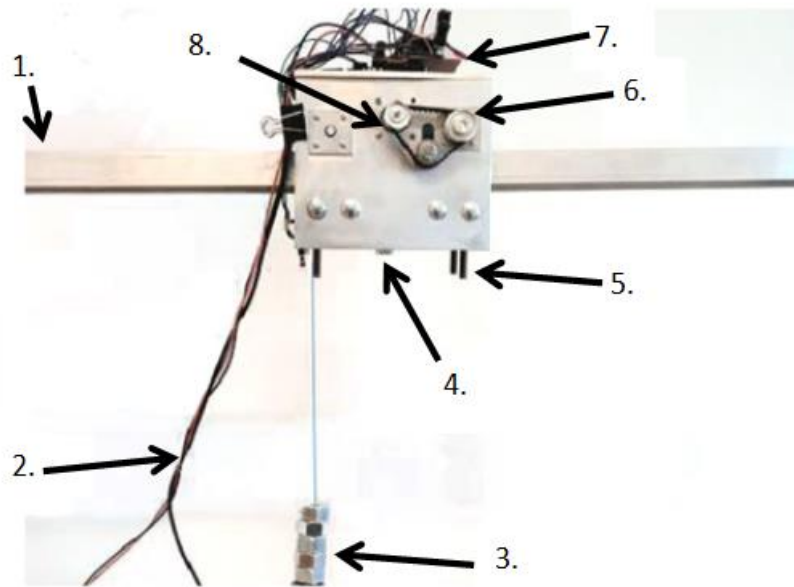
Het basisconcept van onze proefopstelling is een karretje dat over een rail rijdt zoals zichtbaar in Figuur 10. De voornaamste reden waarom we voor deze opstelling hebben gekozen is omdat we zowel de positie en de lengte van het touw moeten regelen en dus nood hebben aan twee motoren, deze motoren kunnen we nu gemakkelijk verwerken in het frame van het karretje.

Het werken met een rail in plaats van een kabel, heeft als voordelen dat we ons volledig kunnen focussen op de outputregeling, en zo geen rekening moeten houden met de doorbuiging van onze kabel. De rail zou eventueel ook kunnen verlengd worden, met extra steunpunten onderweg, om een grotere afstand te overbruggen.

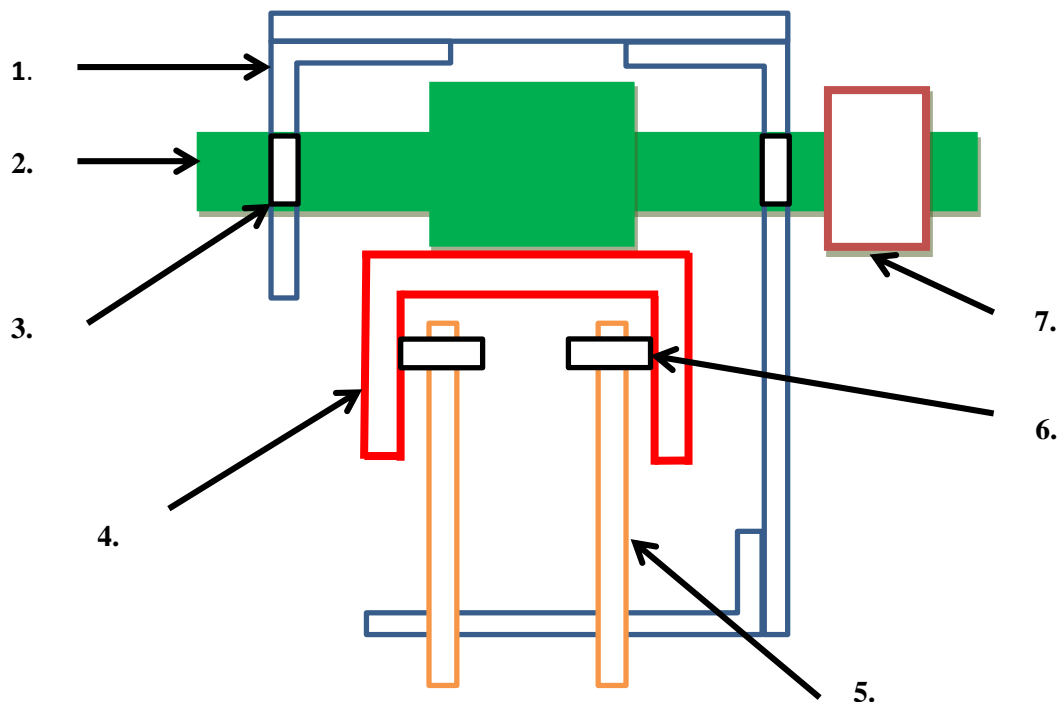
Op Figuur 10 zien we een foto van de constructie, waarop alle onderdelen zijn aangeduid. Een schematische voorstelling van het vooraanzicht van het frame van het karretje zien we duidelijk op Figuur 11. De componenten in Figuur 10 zijn:

1. Rail
2. Bedrading
3. Massa
4. Motor (1) om touw aan te drijven om de last op te hijsen
5. Draadstaaf
6. Groot riemwiel aan as, aangedreven door motor (2) via een tandriem
7. Sturingselektronica

8. Klein riemwiel aangedreven door motor (2)



Figuur 10: foto van de kar en de last, met aangeduide onderdelen



Figuur 11: dwarsdoorsnede ter verduidelijking van de montage van de kar op de rail

Op Figuur 11 kunnen we de volgende onderdelen van elkaar onderscheiden.

1. **Het frame** bestaat uit geëxtrudeerde aluminium L-profielen die we aan elkaar bevestigen door middel van bouten en moeren. We delen het frame op in vier verschillende delen om willen

van productiemogelijkheden en het eenvoud bij de assemblage. We hebben gekozen voor aluminium om de massa van het karretje te beperken.

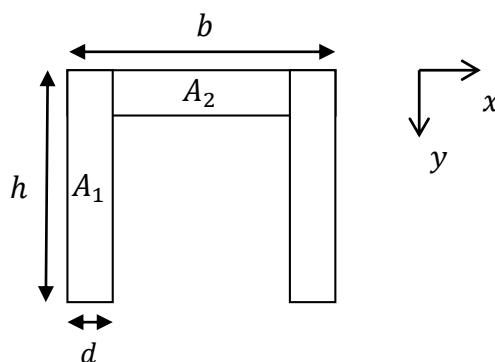
2. **De assen** zijn ook gemaakt van aluminium, omdat dat makkelijk bewerkbaar is en stevig genoeg voor onze doeleinden. De as is in het midden veel dikker dan aan de uiteinden, en dit omdat we de verbreding als wiel willen gebruiken. Om voldoende wrijving te hebben, en een vlotte beweging van het karretje over de rail te proberen bereiken, hebben we de as omhuld met rubber. Aan de uiteinden is de as gelagerd aan het frame.
3. **De lagering van de as aan het frame** bestaat uit goedkope skateboard lagers.
4. **De rail** is in een aluminium U-profiel waarop het karretje zich zal voortbewegen, dit profiel zullen we aan beide uiteinden vastvrijzen op twee poten, zodat de rail op ca 60cm van de grond staat.
5. Deze onderdelen dienen voor de **geleiding volgens de lengterichting** van het karretje: de kar kan niet verdraaien op de rail omdat tegen de binnenkant van de rail vier kogellagers bevestigd zijn.
6. Dit zijn de kogellagers waarover hierboven sprake is.
7. We werken met een tandriem, en dit is het **groot riemwiel**, dat de wielen van het karretje aandrijft.

Doorbuiging van de rail

om de maximale doorbuiging van het profiel te berekenen gaan we ervan uit dat het karretje zich halverwege bevindt op het profiel, het uitgeoefend moment is hier namelijk het grootst. Bij deze berekeningen moeten we uiteraard rekening houden dat we in werkelijkheid met een dynamische belasting te maken zullen hebben, door de minimale gebruikstijd van de proefopstelling kunnen we echter stellen dat deze berekening een goede schatting zal geven.

Gegevens

De massa van de kar $m_{kar} = 6kg$, de massa van de slinger $m_s = 0,5kg$, de afmetingen van de rail zijn (zoals weergegeven in Figuur 12) $b = 80mm$, $h = 30mm$, $t = 3mm$ en $L = 1115mm$. De vloeigrens van aluminium $\sigma = 160MPa$, de elasticiteitsmodulus $E = 69GPa$ en de dichtheid $\rho = 2700 \frac{kg}{m^3}$.



Figuur 12: doorsnede van de rail, met maataanduidingen

Berekeningen

De aangeduide oppervlaktes zijn:

$$A_1 = 3mm \cdot 30mm = 90mm^2$$

$$A_2 = 3mm \cdot 74mm = 222mm^2$$

Het zwaartepunt volgens het gekozen assenstelsel op Figuur 12 is gelegen op:

$$y_{tot} = \frac{2.15mm \cdot 90mm^2 + 1.5mm \cdot 222mm^2}{2.90mm^2 + 222mm^2} = 7,54mm$$

Relatief tegenover het midden van de horizontale balk is dit: $7,54mm - 1.5mm = 6,04mm$. Dus het oppervlaktetraagheidsmoment kunnen we berekenen als:

$$\begin{aligned} I &= 2I_{vert} + I_{hor} \\ &= 2 \cdot \left(\frac{1}{12} \cdot 3mm \cdot (30mm)^3 + (30 - 7.54)^2 \cdot 30mm \cdot 3mm \right) \\ &\quad + \left(\frac{1}{12} \cdot 74mm \cdot (3mm)^3 + 6.04^2 \cdot 74mm \cdot 3mm \right) \\ &= 233778.7mm^4 \end{aligned}$$

De kracht die in het slechtste geval (grootste doorbuiging want grootste moment) wordt uitgeoefend op het midden van de balk:

$$F = 9,81 \frac{N}{kg} \cdot 6,5 kg = 63,77 N$$

Het maximale moment zal zich dus in het midden van het profiel bevinden, en kunnen we berekenen via de formule:

$$M = \frac{FL}{4} = \frac{63.77N \cdot 1115mm}{4}$$

De doorbuiging is dan:

$$f = \frac{FL^3}{48EI} = \frac{63.77N \cdot (1115mm)^3}{48 \cdot 69GPa \cdot 233778.7mm^4} = 0,1142mm$$

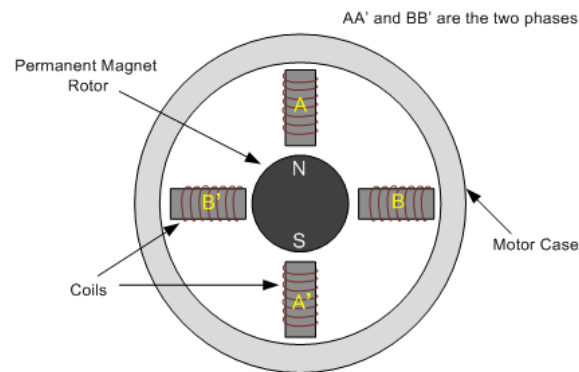
Wat we aanvaardbaar vinden.

Elektronica

Door de aard van platituderegeling, is het bij deze opstelling belangrijk dat de last en het karretje op een nauwkeurige en herhaalbare manier bewogen kunnen worden. Precies om die reden hebben we gekozen voor stappenmotoren: een speciale soort motoren die (mits de juiste sturing) discrete posities aannemen. Omdat het niet evident is om deze sturing zelf te ontwerpen, en het weinig nut heeft om het wiel opnieuw uit te vinden, maken we gebruik van makkelijk verkrijgbare *drivers*. De motoren, de *drivers* en toebehoren halen we allemaal uit de 3d-printerwereld, de eisen voor een 3d-printer zijn immers vergelijkbaar: nauwkeurig en eenvoudig te sturen. De *drivers* zorgen ervoor dat een *microcontroller* noodzakelijk wordt, om deze *drivers* aan te sturen, en dus hebben we gekozen voor een Arduino, een toegankelijke en gebruiksvriendelijke *microcontroller* met bijhorende *programming environment*.

Stappenmotoren

Stappenmotoren (in het Engels: *stepper motors*) zijn een speciale soort motoren die door een speciale manier van opbouwen discrete rotatiestappen kunnen zetten. De motoren die wij kozen zijn bipolaire tweefasige stappenmotoren en zetten stapjes van 1.8° , dus zijn er 200 stappen nodig voor een volledige rotatie. Deze soort stappenmotor bestaat uit een rotor met permanente magneten en een stator (buitenkant) met twee spoelenparen (fasen), met elk twee klemmen, dus aan de buitenkant van de motor vindt men vier draden terug. De spoelenparen ((A,A') en (B,B')) in Figuur 13) zijn twee tegengesteld gewikkelde spoelen in serie. Door een speciale plaatsing van de spoelen, kan voor een specifiek bekrachtigingspatroon een rotatie over een kleine hoek bekomen worden.



Figuur 13: (schematische) doorsnede van een tweefasige bipolaire stappenmotor met permanente magneten op de rotor

In de meest eenvoudige opstelling, worden de vier klemmen bekrachtigd volgens het schema in Tabel 1. De klemmen zijn telkens vernoemd naar de spoel waarop ze aangesloten zijn, en een '+' houdt in dat deze klem aan de motorspanning (= 12V) wordt gehouden, terwijl een '-' inhoudt dat de klem aan de grond (= 0V) wordt gehouden. Een nadeel van deze eerste bekrachtigingsmethode is dat telkens slechts één van de twee spoelen gebruikt wordt, en dat er dus minder koppel gegenereerd wordt. Een alternatief is het schema in Tabel 2: een ander mogelijk bekrachtigingsschema voor een stappenmotor, maar dat heeft dan weer andere nadelen.

| Index | A | B | A' | B' |
|-------|---|---|----|----|
| 1 | + | - | - | - |
| 2 | - | + | - | - |
| 3 | - | - | + | - |
| 4 | - | - | - | + |

Tabel 1: mogelijk bekrachtigingsschema voor een stappenmotor

| Index | A | B | A' | B' |
|-------|---|---|----|----|
| 1 | + | - | - | + |
| 2 | + | + | - | - |
| 3 | - | + | + | - |
| 4 | - | - | + | + |

Tabel 2: een ander mogelijk bekrachtigingsschema voor een stappenmotor

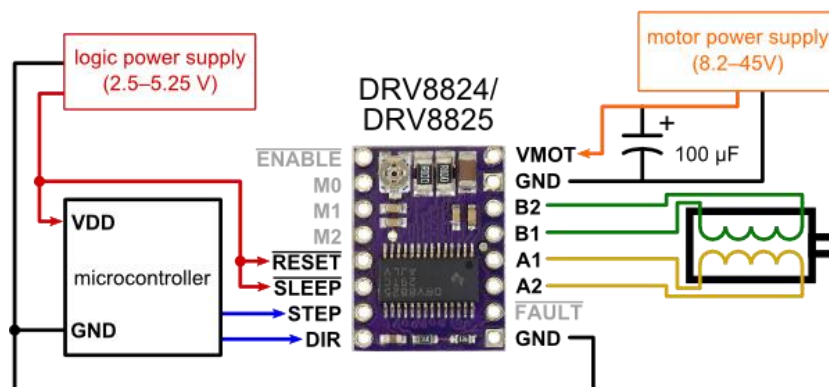
Dit zijn slechts de twee meest eenvoudige bekrachtigingsschema's, maar het werkingsprincipe is hier al duidelijk. Aan het schema kan men zien dat klemmen A en A' altijd een verschillende toestand hebben: de stroom keert dus telkens van zin om, daarom wordt dit een bipolaire stappenmotor genoemd. Door de stroom door de spoelen gradueel te variëren in plaats van ze volledig aan of uit te zetten, kan men *microstepping* verwezenlijken, waarbij men kleinere staphoeken kan bereiken. Op deze manier bereiken de stappenmotoren in deze opstelling véél kleinere hoeken (factor $\frac{1}{32}$) dan 1.8° .

Zoals reeds eerder vermeld is, gebruiken wij een gespecialiseerde driver om dit te bereiken. Verder heeft onze motor een koppel van $4.8N.cm$, bij een stroom van $0.6A$ door de windingen.

De stepper driver

We gebruiken een module van Pololu, een bedrijf dat zich vooral bezighoudt met mechatronische componenten beschikbaar te maken voor de consument. Over de module zelf valt niet veel te zeggen, dit is voor ons een *black box*, maar we moeten natuurlijk weten hoe we de module moeten aansluiten en aansturen.

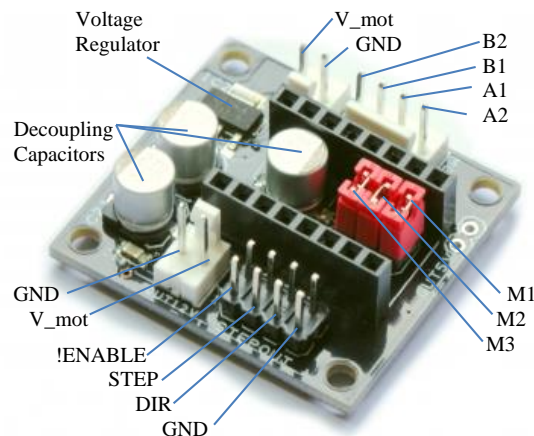
Dit vinden we terug op de website van Pololu, zie Figuur 14. Dit is de meest basische aansluitmethode, wij doen het nog iets anders om de *microstepping* modus te activeren.



Figuur 14: aansluitingsschema voor de stappenmotordriver

Op dit *breakout board* staan, naast de chip, enkele andere componenten, dit zijn weerstanden, condensatoren en een potentiometer. Een rappe blik op de datasheet van de DRV8825 chip, leert ons dat de weerstanden vermoedelijk *sense resistors* zijn, dit zijn zeer kleine (in weerstandswaarde), nauwkeurig opgemeten weerstanden waarover de spanning wordt gemeten, om zo de stroom te berekenen. Deze *driver* werkt met terugkoppeling van (onder andere) de stroomgrootte. De stroomamplitude kan ingesteld worden via de potentiometer, we hebben deze afgesteld op de nominale stroom voor onze motor: $600mA$. De belangrijkste externe contacten zijn de spanningsbronnen (V_{mot} en V_{logic}), de controller pinnen (STEP, DIR en IENABLE, waarbij ! staat voor "niet", in de figuur aangeduid door overstreping), de GND en natuurlijk de klemmen van de motor zelf. Als we met *microstepping* willen werken, hebben we ook pin M0, M1 en M2 nodig. Merk ook op dat er een (vrij grote) condensator tussen GND en V_{mot} is aangesloten, dit is een *decoupling capacitor*, een condensator die ruis op het circuit wegfiltert. Gemakshalve hebben we gekozen om ook hiervoor een module (zie Figuur 15) te gebruiken. Deze maakt de belangrijkste pinnen makkelijk bereikbaar, en maakt de standaardaansluitingen (!RESET aan !SLEEP) al op voorhand. De twee zwarte rijen op deze printplaat, passen op de pinnetjes die in de gaatjes van de *driver* worden vastgesoldeerd, zo is de *driver* module zelf makkelijk uitwisselbaar indien er iets misloopt. De pinnen om te communiceren met de *microcontroller* zijn aangeduid onderaan op de afbeelding. De *microstepping* pinnen (M1, M2 en M3) zijn met behulp van *jumpers* (de rode blokjes) kortgesloten aan de voedingsspanning van de *driver* chip. Deze voedingsspanning is niet V_{mot} , maar V_{logic} , en deze wordt opgewekt in de *voltage regulator* die ook is aangeduid op de figuur. Bovenaan het bordje komt de motorspanning V_{mot} binnen, dit is voor onze toepassing $12V$. De namen van de communicatiepinnetjes zijn redelijk vanzelfsprekend: de GND is de grond, DIR komt van *direction*, en geeft de richting door waarin stappen gezet moeten worden,

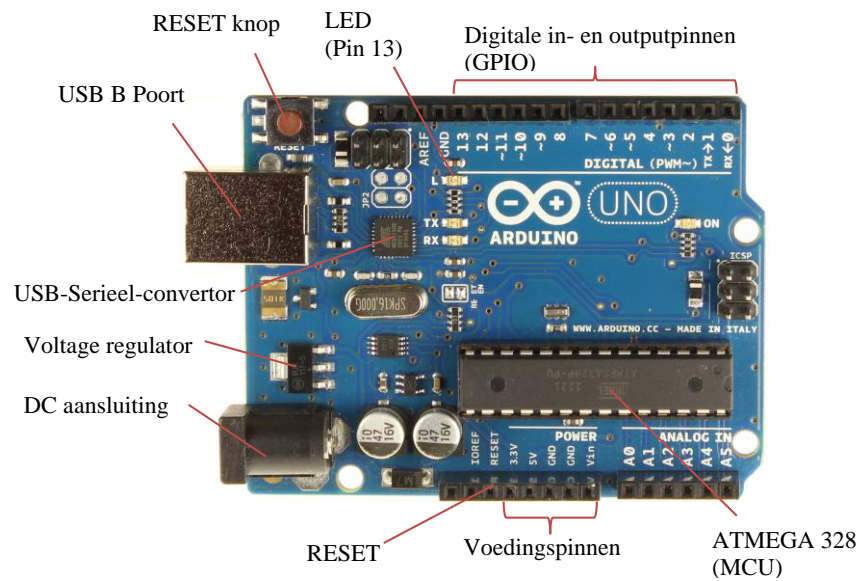
terwijl STEP aangeeft wanneer een stap moet gezet worden, door middel van een puls. En !ENABLE is een soort aan-uit-schakelaar: als deze pin op logische 0 (=0V) wordt gezet, is de motor ingeschakeld, als ze op logische 1 wordt gezet (=5V), is de motor uitgeschakeld. De naam is !ENABLE, wat staat voor “niet ENABLE”, dus dit is ook vanzelfsprekend.



Figuur 15: pcb om de schakeling met de stappenmotordriver te vereenvoudigen

De microcontroller

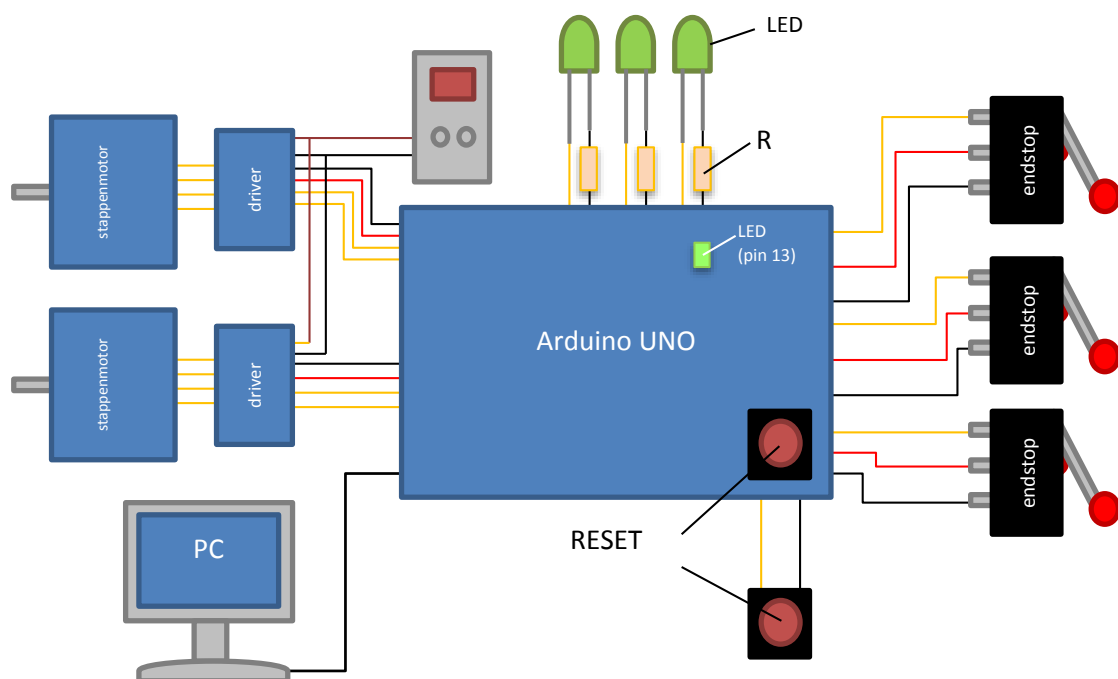
Voor de *microcontroller* hebben we een Arduino gekozen, omdat de bijhorende programmeertaal zeer goed gedocumenteerd is, en we hier al een beetje bekend mee waren. In Figuur 16 zijn de belangrijkste componenten voor ons aangeduid, we gaan deze rap eens overlopen. De voedingspinnen staan bovenaan op het bordje. Voor ons zijn 5V, GND en Vin belangrijk, de betekenis van deze namen spreekt voor zich, behalve misschien voor Vin. Als de microcontroller gevoed wordt via de DC aansluiting (bovenaan) of via de Vin pin, wordt de spanning gereduceerd tot 5V in de *voltage regulator*. Dit is geen bijzonder efficiënte conversie, want de overspanning wordt gewoon omgezet in warmte, voor hoge stromen (en dus hoge vermogens) is dit dus niet aan te raden. Bij ons wordt de gegenereerde 5V enkel gebruikt om enkele LED's aan te steken, en om de microcontroller en de *stepper drivers* (op logisch niveau, dus niet de stappenmotoren zelf) te voeden, en dit vergt slechts enkele tientallen milliwatts. Indien we hogere stromen zouden nodig hebben, zouden we moeten rekening houden met de warmteontwikkeling in de voltage regulator, en bijhorende maatregelen (koelinnen, ...) treffen. Omdat we er uiteindelijk voor gekozen hebben om de Arduino voortdurend aangesloten te laten aan de computer, die ook 5V levert via de USB-poort, is deze bedenking eigenlijk overbodig. Daarnaast zijn er nog de RESET-pin en de RESET-knop, deze dienen om de *microcontroller* opnieuw op te starten. Tijdens het uitvoeren van ons programma, is het nodig om met de computer te communiceren om data door te sturen, dit doen we via een seriële verbinding over de USB-poort. Omdat USB-verbindingen en seriële verbindingen niet helemaal hetzelfde zijn, moet er ergens een omzetting gebeuren, en dat gebeurt op een tweede *microcontroller*, deze is ook aangeduid. De digitale pinnen met nummers 0 en 1 (helemaal onderaan links op deze afbeelding), worden ook gebruikt voor deze seriële verbinding, en ze mogen in onze toepassing niet gebruikt worden, omdat er anders storingen in de communicatie met de computer zouden optreden. De digitale GPIO (*General Purpose Input & Output*) pinnen met nummers 2 tot en met 13, zijn wel bruikbaar. Het denkwerk van de Arduino wordt uitgevoerd op de microcontroller chip zelf (MCU), dit is een ATMEGA 328. Om het gedrag van de GPIO pinnen te sturen, kunnen we de microcontroller programmeren, dit gebeurt in een programmeertaal die wat weg heeft van C, C++ en JAVA.



Figuur 16: de Arduino UNO, de microcontroller waarvoor we gekozen hebben

De schakeling

In plaats van een schema waarop alle klemmen aangeduid zijn met de bijpassende naam, hebben we om het overzicht te behouden een soort blokdiagram (Figuur 17) van de componenten gemaakt.



Figuur 17: aansluitschema van alle componenten

Hierop staan wel alle onderdelen, maar niet de pinnummering of de lay-out op de pcb (*Printed Circuit Board* of printplaat in het Nederlands). Om te beginnen is er dus de Arduino: dit is de centrale component. Op de Arduino wordt wat simpel rekenwerk verricht, maar dat proberen we zoveel mogelijk te beperken. De hoofdbedoeling is om met behulp van de GPIO pinnen de toestand van

enkele knoppen (de *endstops*) in te lezen, en andere componenten aan te sturen. Zoals we reeds eerder gezegd hadden, zijn er vier draden nodig tussen de Arduino en de motorcontrollers, en hebben deze controllers ook een 12V voeding (PSU, *Power Supply Unit*) nodig. Daarnaast hebben we, voor de veiligheid en om automatisch te bepalen waar het karretje zich bevindt, ook enkele *endstops* geïnstalleerd. Dit zijn knoppen die ingedrukt worden met behulp van een hendeltje, waarop op het einde een wielletje zit. Van zodra dit wielletje tegen iets botst, wordt de knop ingedrukt, en dit kunnen we inlezen op de *microcontroller*. Een endstop heeft drie klemmen, eentje is de voelklem, en wordt ingelezen op de *microcontroller*, de andere hangen aan de grond (GND, 0V) of aan de voedingsspanning (Vd, 5V). Twee van deze *endstops* worden gebruikt om te detecteren of het karretje zich niet tegen het uiteinde van de rail (langs de x-as) bevindt, een derde om te detecteren of de last al helemaal tot boven is gehesen. Daarnaast hebben we ook drie LED's (*Light Emitting Diode*) voorzien, deze dienen vooral om tijdens het programmeren wat feedback te geven. Aan pin 13 van de Arduino hangt een klein LEDje dat op de Arduino zelf staat, maar omdat wij een *shield* maken dat bovenop de Arduino past, is dit onzichtbaar geworden, dus hebben we zelf 3 LED's aangesloten. Omdat de *microcontroller* op een spanning van 5V werkt, mogen we een LED niet zomaar aansluiten, maar moet dit over een weerstand R , zodanig dat haar maximale stroom (ca 30mA) bij nominale spanning (ca 2V) niet overschreden wordt. Ook moeten we opletten dat het juiste beentje op de juiste plaats is aangesloten: een LED is een diode, en geleidt slechts in één richting. De twee beentjes kunnen onderscheiden worden doordat het ene iets langer is dan de andere, of door te kijken naar de structuur binnenin de doorzichtige omkapping van de diode.

De laatste onbesproken component, is de RESET-knop. Deze staat er dubbel op: er is een resetknop gemonteerd op de Arduino zelf, maar omdat deze ontoegankelijk is door het *shield* dat we gemaakt hebben, hebben we besloten om de RESET-knop te kopiëren naar onze eigen pcb (*Printed Circuit Board*). Deze RESET-knop is handig om direct de microcontroller te laten stoppen, dus bijvoorbeeld als noodstop.

Programma

Zoals reeds gezegd, worden op de *microcontroller* enkel simpele bewerkingen uitgevoerd, het zware rekenwerk laten we aan MATLAB© over, en de resultaten sturen we door over de COM-poort. Dit doorsturen is ongetwijfeld de moeilijkste stap, vooral omdat we rechtstreeks uit MATLAB werken, en dit niet zo goed gedocumenteerd is als bijvoorbeeld communicatie via JAVA of iets dergelijk. We hebben besloten om vanuit MATLAB te werken, omdat het een krachtig programma is, waarin we al onze numerieke berekeningen voor het pad hebben uitgevoerd. Omdat de opstelling die we gemaakt hebben, eventueel later als practicumopstelling zal worden gebruikt, leek het ons ook handig om niet met bv. Processing© (een programma dat vaak in combinatie met Arduino wordt gebruikt, om te communiceren met een pc) te werken, omdat dat apart moet gedownload en geïnstalleerd worden.

Het programma werkt in grote lijnen als volgt: de Arduino wordt via USB aangesloten, en vervolgens wordt MATLAB opgestart. Het programma *PLATITUDE.m* wordt opgestart nadat de juiste COM-poort geselecteerd is in de code van dit MATLAB-script. Het script maakt automatisch verbinding, en reset daarbij de *microcontroller*. Vervolgens stuurt het alle belangrijke instellingen door: de kalibratiewaarden, de lengte van de x- en y-as enzovoorts. Dit zorgt ervoor dat men in principe, als men metingen wil doen met de opstelling, enkel het MATLAB-programma moet gebruiken, en er nooit nieuwe *firmware* (de software op de *microcontroller*) moet geïnstalleerd worden op de Arduino, dit leek ons een enorm handige functie om de opstelling daarna als meetopstelling te gebruiken. Elke keer

dat MATLAB iets verzendt naar de Arduino, verwacht het iets terug, een antwoord van minstens twee karakters.

Nadat de instellingen verzonden zijn, staat de *microcontroller* in “waakmodus”, dit wil zeggen dat ze afwacht tot ze een nieuw commando krijgt. Over een seriële verbinding wordt (bij ons) tekst verstuurd, dit bracht ons op het idee om een soort protocol op te stellen, geïnspireerd op bijvoorbeeld MIDI in de muziekwereld, of g-code bij computergestuurde werktuigmachines (*cnc*'s, robots, ...). Elk commando bestaat uit een *header*, een *body* en een *delimiter* om het einde af te bakenen. De *header* is een letter die het soort commando aanduidt, bijvoorbeeld het commando ‘H’ zal de machine laten *homen* (automatisch naar haar nulpunt bewegen). De *body* kan uit willekeurige tekst bestaan, en bevat parameters die horen bij het commando. De *body* loopt tot de *delimiter*, een symbool dat we enkel gebruiken om het einde van een commando aan te geven, zodat de uitvoering van het commando kan gestart worden. We kozen als *delimiter* een *hashtag* ‘#’, lichtelijk geïnspireerd door de sociale media, maar vooral omdat dit een symbool is dat makkelijk te vinden is op het toetsenbord, en zelden gebruikt wordt voor andere dingen. Een commando ziet er dus bijvoorbeeld als volgt uit: ‘x400#’, wat wil zeggen: beweeg in de x-richting tot je op locatie 4000 (uitgedrukt in stappen tov het nulpunt) staat. Deze locaties worden doorgestuurd in een aantal stappen gedeeld door tien: willen we bijvoorbeeld naar $x = 380\text{mm}$ bewegen, dan moeten we x vermenigvuldigen met de kalibratiewaarde voor de x-as (die omzet naar een aantal stappen), dewelke ongeveer 100 is. Dus zouden we 38000 stappen moeten doorsturen, dit stoot echter op een praktisch probleem: de maximale waarde voor een geheel getal in de meeste programmeertalen, en dus ook deze van de Arduino, is 32768, en dus zal onze 38000 voor een *overflow* zorgen, wat aanleiding geeft tot fouten. Dit vermijden we door te delen door tien: we sturen 3800 door in plaats van 38000. We offeren dus wel een deel van onze resolutie op: het toestel zal 10x minder nauwkeurig werken. Daarnet zeiden we reeds dat de kalibratiewaarde (aantal stappen per millimeter) voor de x-as ongeveer 100 is, en dus is de nauwkeurigheid bij een verlies van factor 10 nog steeds één tiende van een millimeter, wat meer dan voldoende is voor onze toepassing. Zo hebben we nog enkele commando's opgesteld: bijvoorbeeld de letter ‘E’ vraagt de status van de *endstops* op, en ‘P’ geeft de positie volgens de Arduino (die bijgehouden wordt in een aantal stappen). Dit is geïnspireerd door bijvoorbeeld de MIDI-interface voor geluidsapparaten, of de g-code bij *cnc*'s en andere computergestuurde werktuigen.

Om een pad te volgen, moeten beide motoren aan veranderlijke snelheid worden aangestuurd, en hier botsen we op problemen. Met behulp van MATLAB berekenen we een lijst punten waarlangs de last moet passeren, en deze proberen we dan in het geheugen op te slaan. Deze lijst is echter te lang: we moeten op zoek naar een alternatief. Omdat we toch al een basisverbinding via USB hadden (om de commando's door te sturen, zoals hierboven beschreven), was het nieuwe plan om de data “gewoon live” door te sturen. Nadat we een programma hadden geschreven om voortdurend nieuwe coördinaten te vragen aan de computer, waren de testresultaten zeer slecht: de motoren schokten enorm, dit was vermoedelijk te wijten aan een te trage communicatie. Bij verder onderzoek kwam aan het licht dat de seriële verbinding vanuit MATLAB geen data kan verzenden aan de snelheden die wij nodig hebben, dit kunnen we linken aan de symptomen doordat er telkens heel even gewacht wordt alvorens het pad verder kan worden uitgevoerd. De data die wel werd doorgestuurd, was door deze vertraging dikwijls veranderd in de maximale waarde voor een geheel getal: 32768, hierdoor begon de machine vooruit te gaan tot tegen het einde van de rail.

Validatie van het model door de proefopstelling

We hebben helaas geen metingen kunnen doen op de opstelling. Omdat bij ons de positie van de last belangrijk was, en dat die niet rechtstreeks opmeetbaar is (met bijvoorbeeld een accelerometer), hadden we in gedachten om een opstelling te maken waarbij het karretje en de last een gekleurde vlek krijgen (bijvoorbeeld karretje groen en last rood), de uitvoering van ons pad te filmen, en dan door middel van kleurherkenning in MATLAB een benadering van de positie te vergelijken met de gewenste positie.

Conclusie

Uit fouten moet je leren, dus hebben we geprobeerd om hier onze conclusies uit te trekken: MATLAB is een zeer krachtig programma, dat vooral geschikt is om te werken met grote matrices. Er zijn enkele bijkomende functionaliteiten, waaronder de mogelijkheid tot seriële verbindingen, maar deze zijn vooral bedoeld om data in te lezen, en het verzenden ervan gebeurt niet snel genoeg voor deze toepassing. Indien we het programma opnieuw zouden ontwikkelen, zouden we vermoedelijk opteren voor JAVA of C++ (echte programmeertalen) of *LabView*, omdat dat een programma is dat speciaal ontworpen is om *realtime* data van en naar *microcontrollers* te verzenden/ontvangen.

We besluiten uit dit vakoverschrijdend project dat het vermoedelijk mogelijk is om door middel van inverse kinematica het pad van een slinger te sturen. Omdat het pad dat gevolgd moet worden aan bijzonder veel eisen moet voldoen, is het niet direct praktisch bruikbaar. Als we opteren voor een vaste slingerlengte, kan wel een bruikbaar (bijvoorbeeld voor een hijskraan) pad gevonden worden. Mits een goede regeling van de versnelling van het karretje zal de last bij verplaatsing van het karretje exact het pad volgen dat we willen. Dit is een volledig andere benadering dan het binnen de perken houden van slingering door middel van een feedbacksysteem: hierbij wordt de (neiging tot) slingering gemeten door middel van sensoren, en vervolgens wordt er ergens een actuator gebruikt om deze te beperken. In slechte omstandigheden (zware industrie) gaan sensoren rap kapot, wat leidt tot hoge onderhoudskosten en/of een onbetrouwbare machine. Daarom is de benadering via platte output zeker niet nutteloos in de hedendaagse industrie.

Bijlages

In de bijlage zit het matlabbestand dat gebruikt wordt om het in dit verslag beschreven pad om te zetten naar waarden voor de lengte van het koord en voor de positie van het karretje, alsook de simulatie om dit alles te verifiëren.

Zoals hierboven reeds gezegd werd: de sturing is nog niet in orde, dus daarom zit het programma voor de sturing hier nog niet bij als bijlage, dit kan eventueel later nog volgen...

```

function [uit,uit2] = testrecht2( )

clear;
clc;
clf;

l=@(t) 0.5; %input van de lengtefunctie
L0=0.62; %lengte van het touw als massa in rust op grond is
g=9.81;
theta0=0;
thetat0=0;
r0=0;
rt0=0;
rtt=@(t) .6584362144e-6*t^3*(126-14*t+3/5*t^2-7/600*t^3+7/81000*t^4)+.3292181072e-6*t^4*(-14+6/5*t-7/200*t^2+7/20250*t^3)+.3292181071e-7*t^5*(6/5-7/100*t+7/6750*t^2)+.2013566404e-6*t*(126-14*t+3/5*t^2-7/600*t^3+7/81000*t^4)+.4027132808e-6*t^2*(-14+6/5*t-7/200*t^2+7/20250*t^3)+.2013566404e-6*t^3*(6/5-7/100*t+7/6750*t^2)+.3355944006e-7*t^4*(-7/100+7/3375*t)+.3480238228e-11*t^5; %input versnellingsfunctie karretje

T=30;
stap=50/1000; %aantal milisecondes tussen elke iteratie
aantalit=T/stap
tspan=0:stap:T;

u1=@(t) rtt(t); %inputs voor het toestandsmodel;
u2=@(t) l(t);

x0=[theta0 thetat0*u2(0)^2 r0 rt0];

[t,x]=ode23(@rhs, tspan, x0 );
for i=1:aantalit+1
    L(i,1)=L0-l(tspan(i));
end
function xdot=rhs(t,x)
    xdot_1=x(2)/u2(t)^2;
    xdot_2=-u2(t)*sin(x(1))*g-u1(t)*u2(t)*cos(x(1));
    xdot_3=x(4);
    xdot_4=u1(t);

    xdot=[xdot_1; xdot_2; xdot_3; xdot_4];
end

[~,N]=size(tspan);

for it=1:N %animatie
    clf
    P(it)=x(it,3)+u2(tspan(it))*sin(x(it,1)); %x component
    Q(it)=-u2(tspan(it))*cos(x(it,1)); % y component
    plot([x(it,3),x(it,3)+u2(tspan(it))*sin(x(it,1))],[0,-u2(tspan(it))*cos(x(it,1))], 'k-o', 'LineWidth',1.5);
    hold on
    axis([0 0.8 -0.5 -0.1])
    drawnow

```

```
pbaspect([10,3,1])
pause(.001)

end
clf
plot(P,Q,'LineWidth',2)
axis([0 0.8 -0.5 0])

tel=1; %aantal stappen dat wordt overgeslaan voor communicatie met arduino
uitl=0; %lengte touw
uitr=0; %plaats kar
uitt=0; %tijd
teller=1;
for it=1:tel:N
    uitt(teller)=t(it)*1000;
    uitr(teller)=round(1000*x(it,3));
    uitl(teller)=round(1000*L(it));
    teller=teller+1;
end

mat=[t,round(x(:,3)*1000),round(L*1000)];%*10 000 is om de afstanden op 1/10 van een✓
milimeter door te geven
mat=transpose(mat);
uit=[uitt;uitr;uitl];
uit2=[P;Q];
csvwrite('tijd.txt',uit);

end
```