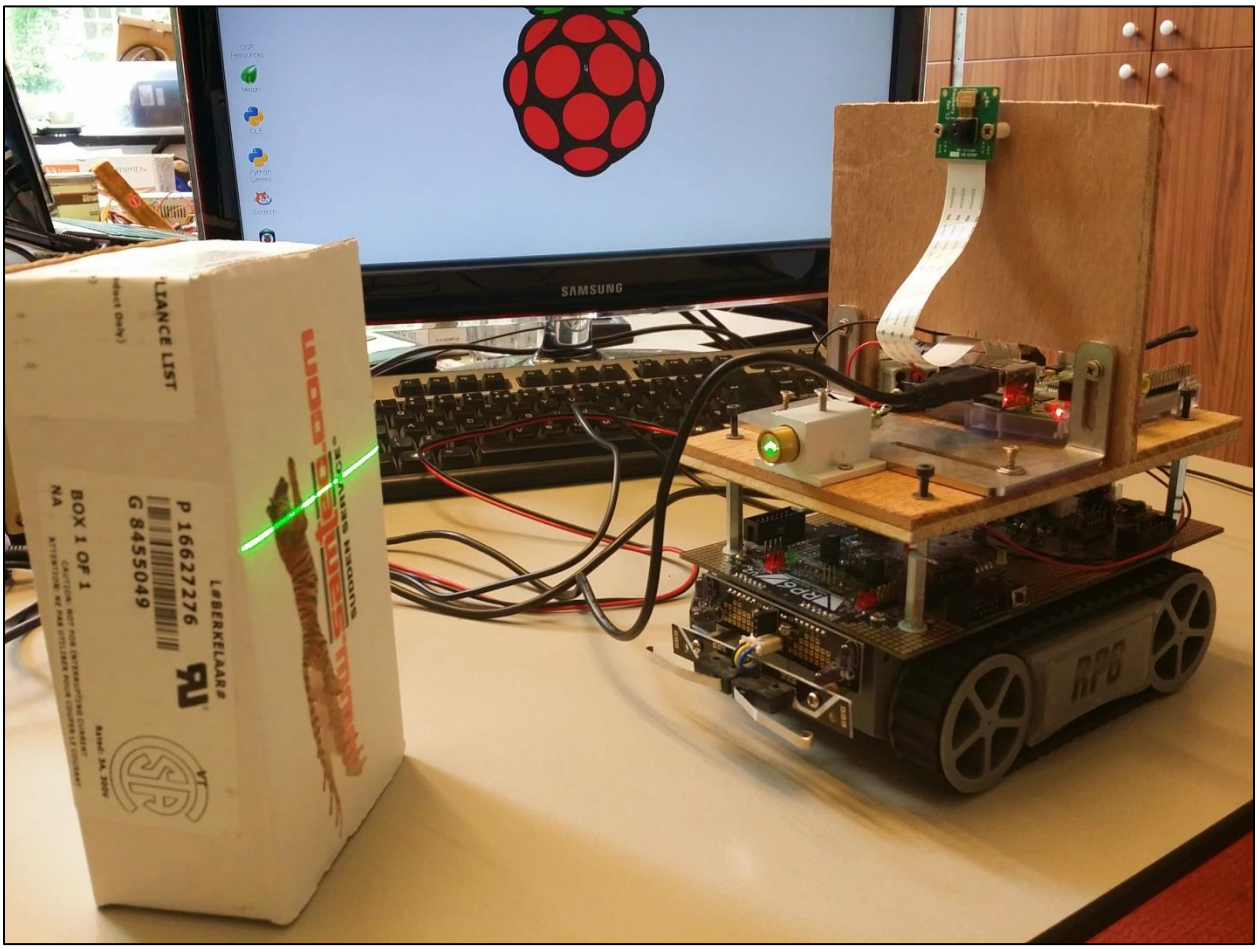
# Afstudeerverslag

## Het ontwikkelen van een real-time lokaliseringssysteem





| | |
|---|---|
| Student: | Menno Sman |
| Studentnummer: | 09001662 |
| Onderwijsinstelling: | De Haagse Hogeschool |
| Opleiding: | Technische Informatica |
| Afstudeerperiode: | 10 februari 2014 - 6 juni 2014 |
| Afstudeerbegeleider: | Anthony van Geest |
| Tweede examinator: | Tony Andrioli |
| Bedrijf: | Berkelaar Meet- en Regeltechniek |

# Referaat

Dit is het afstudeerverslag van Menno Sman, Technische Informatica student aan de Haagse Hogeschool te Delft. Het afstudeerverslag omschrijft het ontwikkelingsproces van een real-time lokaliseringssysteem. Het systeem maakt gebruik van een laser en een camera-module om nabije objecten te lokaliseren. Het doel van het systeem is om autonome robots nabije objecten te laten detecteren en omzeilen. Het ontwikkelingsproces bestaat uit zowel het onderzoek doen naar als het ontwikkelen van een lokaliseringssysteem en een autonome robot.

*De afstudeeropdracht is uitgevoerd in de periode van 10 februari 2014 tot 6 juni 2014.*

**Descriptoren**:

- Technische Informatica
- De Haagse Hogeschool
- Berkelaar Meet- en Regeltechniek
- Embedded-systemen
- Optische triangulatie
- Odometrie
- Programmeren
- Ontwikkelen
- Raspberry Pi
- OpenCV
- AVR
- C
- Dead Reckoning
- Computer-vision
- Robotica
- Real-time
- Laser
- C++

# Voorwoord

*Ik wil graag de heren Wiebe Berkelaar, Armin Oonk en Jorrit Scharloo bedanken voor mijn tijd bij Berkelaar Meet- en Regeltechniek. Ik ben erg onder de indruk van de wijze waarop ze in staat zijn om interessante opdrachten binnen te halen en met succes uit te voeren. Met drie medewerkers ontwikkelen ze producten die uiteenlopen van autonome robots tot zweeftreinen. Ik zie het als een bijzondere kans dat ik bij een bedrijf heb kunnen werken dat zich bezig houd met zulke interessante onderwerpen. Ik heb veel van ze geleerd. Daarnaast wil ik de heren Anthony van Geest en Tony Andrioli bedanken voor hun begeleiding en feedback vanuit De Haagse Hogeschool.*

*Menno Sman*

*4 juni 2014*

# Inhoudsopgave

# 1. Inleiding

In 2011 heeft Berkelaar Meet- en Regeltechniek een robot ontwikkeld die in staat is om autonoom te navigeren. De robot kan zijn eigen positie te bepalen door middel van RFID-chips in de vloer. De robot kan botsingen met andere objecten te vermijden door middel van infrarood-sensoren. Deze technieken laten echter veel te wensen over. Berkelaar Meet- en Regeltechniek heeft plannen voor de robot die met deze technieken niet kunnen worden doorgevoerd. Berkelaar Meet- en Regeltechniek is daarom geïnteresseerd in een alternatieve techniek.

In opdracht van ingenieursbureau Berkelaar Meet- en Regeltechniek dient er een prototype van een systeem ontwikkeld te worden dat in staat is om door middel van een camera-module en een lijnlaser nabije objecten te detecteren.  In dit afstudeerverslag wordt in gegaan het ontwikkelingsproces van dit systeem.

**Leeswijzer**

Het afstudeerverslag is opgedeeld in vijf onderdelen. Het eerste onderdeel wordt gevormd door de bedrijfsomschrijving, opdrachtomschrijving en aanpak. Het tweede deel wordt gevormd door de eerste iteratie. In deze iteratie wordt met name aandacht besteed aan optische triangulatie. Het derde deel is de tweede iteratie. In deze iteratie worden technieken ontwikkeld die een robot nodig heeft om optische triangulatie te kunnen gebruiken. Het vierde deel is de derde iteratie. In deze iteratie wordt onderzoek gedaan naar de samenkomst van alle eerder onderzochte technieken. Er wordt gekeken hoe deze technieken kunnen worden gecombineerd om een robot zelfstandig te laten navigeren. In het laatste deel komen de conclusie, evaluatie en referenties van het ontwikkelingsproces en afstudeerverslag naar voren.

# 2. Organisatiebeschrijving

Berkelaar Meet- en Regeltechniek is een klein ingenieursbureau in Delft. Berkelaar Meet- en Regeltechniek is in 2007 opgericht door Wiebe Berkelaar. Berkelaar Meet- en Regeltechniek telt drie vaste medewerkers, alle drie oud-studenten aan de Technische Universiteit te Delft. Door hun gezamenlijke expertise (Luchtvaart- en Ruimtevaarttechniek, Bouwkunde en Technische Informatica) zijn ze in staat om een grote verscheidenheid aan opdrachten uit te voeren. Deze opdrachten lopen uiteen van het ontwikkelen van kleine gespecialiseerde elektronicaproducten tot het ontwikkelen van complete machines.

Berkelaar Meet- en Regeltechniek heeft verschillende opdrachten uitgevoerd voor het Science Centre van de TU Delft. Het Science Centre is een museum dat de interesse van kinderen in techniek moet stimuleren. Veel van de opdrachten zijn te zien in het museum. Een van deze opdrachten betreft het ontwikkelen van een rijdende binnenhuisrobot. De robot is in staat om kinderen waar te nemen en met hen te communiceren. Het is een van de meest populaire attracties van het Science Centre. Hoewel de robot al een groot succes is, hebben Berkelaar Meet- en Regeltechniek en het Science Centre verschillende ideeën om de robot te verbeteren. Een van deze ideeën betreft het ontwikkelen van een navigatiesysteem waarmee de robot de bezoekers van het museum kan gaan rondleiden. De huidige sensoren van de robot kunnen dit echter niet ondersteunen. Er is behoefte aan een nieuw systeem dat in combinatie met zijn huidige sensoren de robot in staat kan stellen de taak te vervullen.

# 3. Opdrachtomschrijving

## 3.1 Achtergrond

In 2011 heeft Berkelaar Meet- en Regeltechniek een robot ontwikkeld die in staat is om autonoom te navigeren. De robot kan zijn eigen positie te bepalen door middel van RFID-chips in de vloer. De robot kan botsingen met andere objecten te vermijden door middel van infrarood-sensoren. De robot is ontwikkeld in opdracht van het Science Centre van de TU Delft. Het doel van de robot is om de bezoekers van het museum te vermaken.

## 3.2 Probleemstelling

Door middel van infrarood-sensoren kan de robot objecten detecteren. De robot is niet in staat om de locatie van deze objecten te bepalen. Hierdoor moet de robot door middel van 'trial and error' zijn weg naar een doel vinden. Hoewel de robot uiteindelijk het doel zal bereiken is het verre van efficiënt. De robot is hierdoor niet in staat om taken uit te voeren die complexer zijn dan het vermaken van bezoekers.

Door middel van RFID-chips is de robot in staat om zijn eigen locatie te bepalen. RFID-chips zijn betrouwbaar maar onhandzaam. Iedere ruimte waar de robot in moet functioneren, moet worden voorzien van RFID-chips. Het is duur en onrealistisch om alle plaatsen waar de robot moet kunnen functioneren te voorzien van deze chips. De robot is hierdoor alleen in staat om in (kleine) van te voren bepaalde ruimtes te functioneren.

## 3.3 Motivatie

De robot van Berkelaar Meet- en Regeltechniek staat in een aparte ruimte in het Science Centre. In deze ruimte moet de robot bezoekers van het museum vermaken. Het Science Centre heeft echter plannen met de robot. De robot moet in 2014 de bezoekers van het museum gaan verwelkomen en rondleiden. Zijn huidige technieken zijn hier niet toe in staat. De robot zal zijn doelen op de meest efficiënte manier moeten bereiken. De robot kan niet door middel van 'trial and error' een pad gaan vinden als er bezoekers staan te wachten. Daarnaast is het plaatsen van de RFID-chips door het museum duur en onrealistisch. Er is behoefte aan een nieuwe techniek. Een techniek waarmee de robot zonder externe middelen (chips) de bezoekers op een efficiënte manier kan rondleiden.

Naast het Science Centre hebben andere bedrijven aangegeven behoefte te hebben aan een alternatief voor RFID en infrarood. Deze bedrijven maken (net zoals Berkelaar Meet- en Regeltechniek) gebruik van RFID en infrarood omdat alternatieve technieken te duur zijn. De onderdelen waar deze alternatieve technieken uit bestaan (camera-modules, microcontrollers en lasers) worden echter steeds beter betaalbaar. Berkelaar Meet- en Regeltechniek wil daarom onderzoek doen naar de mogelijkheid om zelf een alternatieve techniek te ontwikkelen.

## 3.4 Vooronderzoek

Berkelaar Meet- en Regeltechniek heeft onderzoek gedaan naar een alternatief voor RFID en infrarood-sensoren. De methode die het beste naar voren kwam is optische triangulatie. Optische triangulatie is een manier om met een camera-module en een laser de afstand tot nabije objecten te kunnen bepalen. Omdat Berkelaar Meet- en Regeltechniek plannen heeft om deze techniek zelf te ontwikkelen (en verkopen) zijn ze met name geïnteresseerd in de mogelijkheid om deze techniek te ontwikkelen met betaalbare hardware. Om deze reden heeft Berkelaar Meet- en Regeltechniek al een aantal onderdelen (camera-module, microcontroller en laser) geselecteerd.

### 3.5 Doelstelling

Het doel van de afstudeeropdracht is het ontwikkelen van:
"Een betaalbaar systeem dat door middel van optische triangulatie nabije objecten kan lokaliseren."

### 3.6 Resultaten

De afstudeeropdracht dient te resulteren in:

- Een prototype (software- en hardware-module (camera-module, microcontroller en laser)).
- Documentatie betreffende de ontwikkeling en werking van het systeem (afstudeerverslag).

### 3.7 Eisen

Aan de volgende eisen moet worden voldaan:

- Het systeem moet afstanden kunnen bepalen door de coördinaten van de locatie van objecten te bepalen.
- Het systeem moet beschikken over een gezichtsveld dat net zo breed is als het diep is (1:1).
- Het systeem moet in staat zijn om objecten tot op minimaal 3 meter afstand te kunnen lokaliseren.
- Het systeem moet afstanden kunnen bepalen met een afwijking kleiner dan 2 %.
- Het systeem moet kunnen functioneren op een frequentie van minimaal 5 Hz.
- Het systeem moet lasers hanteren die omstanders (waaronder kinderen) niet kunnen schaden.
- Het systeem moet ontwikkeld worden in de programmeertaal C++.
- Het systeem moet kunnen functioneren met uitsluitend een camera-module, microcontroller en laser.
- Het systeem moet minder kosten dan commercieel beschikbare systemen (minder dan 300 euro).

### 3.8 Grenzen

Het systeem moet in staat zijn om de coördinaten te bepalen van locaties waar nabije objecten zich bevinden. Het systeem zal uitsluitend de coördinaten bepalen van locaties waarin objecten in het zicht van de camera het licht van de laser kruisen. Het systeem zal na het bepalen van de coördinaten er verder niets mee doen.

### 3.9 Organisatie

De duur van de afstudeeropdracht is zeventien weken (10 februari 2014 t/m 6 juni 2014). De heer Wiebe Berkelaar zal optreden als begeleider vanuit Berkelaar Meet- en Regeltechniek. De heer Anthony van Geest zal optreden als begeleider vanuit de Haagse Hogeschool. De heer Tony Andrioli zal optreden als tweede examinator.

# 4. Aanpak

4.1 Inleiding

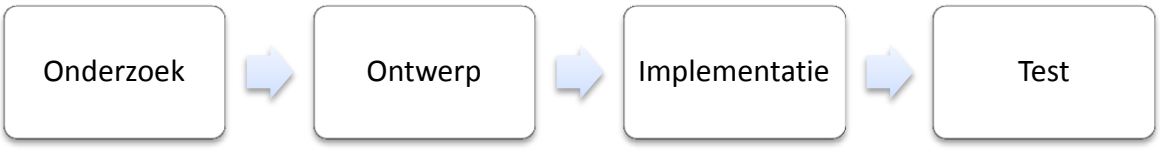
Een weldoordachte aanpak is van belang om de kwaliteit van een ontwikkelingsproces te kunnen waarborgen. Een aanpak kan bestaan uit een aantal ontwikkelelementen of uit een vaste ontwikkelmethodiek. Het voordeel van een ontwikkelmethodiek is dat de aanpak voor alle ontwikkelaars van een uit te voeren opdracht duidelijk en consistent is. Daarnaast zijn ontwikkelmethodieken effectief wanneer er sprake is van veel hiërarchie. In deze situatie is het waardevol om van te voren te kunnen aanduiden wanneer contact opgenomen zal worden met de klant. Ontwikkelmethodieken zijn minder wenselijk in situaties met één ontwikkelaar. Sommige aspecten van ontwikkelmethoden zijn in deze situaties redundant en de tijd niet waard. Het selecteren van een paar ontwikkelelementen om op deze wijze de aanpak zelf vorm te geven kan in deze situaties beter zijn.

Binnen Berkelaar-Meet en Regeltechniek gaat alle communicatie over de korte lijn. Er zijn weinig werknemers en iedereen bevindt zich in dezelfde kamer. Het is hierdoor gemakkelijk om elkaar direct te benaderen. Daarnaast zijn ontwikkelmethodieken voornamelijk gericht op pure softwareontwikkeling. Opdrachten waarbij onderzoek centraal staat zijn minder goed aan te sluiten op de werkwijze van ontwikkelmethodieken. Bij Berkelaar Meet- en Regeltechniek wordt er daarom ook geen gebruik gemaakt van vaste ontwikkelmethodieken. Dezelfde redenen gelden voor deze afstudeeropdracht. Er zal daarom ook geen gebruik worden gemaakt van een vaste ontwikkelmethodiek. Er zal een aparte aanpak worden opgesteld door middel van een losse ontwikkelelementen.

4.2 Ontwikkelmethode

De afstudeeropdracht zal worden uitgevoerd in een periode van zeventien weken. Deze zeventien weken zullen worden opgedeeld in minstens twee iteraties. Alleen de eerste iteratie staat vast. Het probleemdomein is namelijk redelijk onbekend. Aan de hand van de bevindingen van de eerste iteratie zal bepaald worden hoeveel iteraties er zullen volgen. Iedere iteratie bestaat uit een onderzoeksfase, ontwerpfase, implementatiefase en testfase. Het doel van de onderzoeksfase is het verzamelen van zo veel mogelijk informatie over het probleemdomein (en alle verwante technieken/software/hardware). Het doel van de ontwerpfase is de verzamelde informatie te gebruiken om een systeem te ontwerpen dat aan sluit bij het probleemdomein. Het doel van de implementatiefase is het implementeren van het ontwerp. Het doel van de testfase is het testen en evalueren van het systeem om vervolgens te kunnen bepalen hoe de volgende iteratie het beste kan worden ingericht.

Aangezien er voor de eerste iteratie zeven weken zijn gereserveerd, kan er worden gesteld dat er sprake zal zijn van minstens twee iteraties. Een paragraaf aan het begin van iedere iteratie licht toe wat het doel en de motivatie is om de iteratie op de desbetreffende manier in te richten.

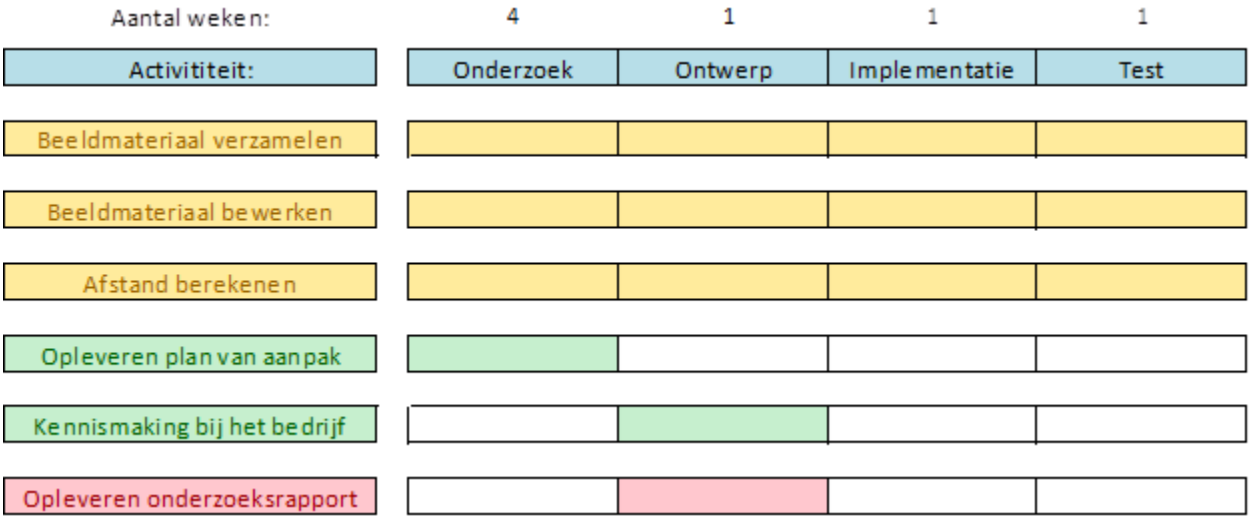



***Figuur 4.1*** *Een iteratie bestaande uit vier fasen.*

# **De eerste iteratie:** Optische triangulatie

*van 17 februari 2014 t/m 4 april 2014 (7 weken)*

Het doel van de eerste iteratie is onderzoek doen naar en het ontwikkelen van de basis van optische triangulatie. De resultaten van deze iteratie zijn bepalend voor de wijze waarop de volgende iteraties worden ingevuld. De eerste iteratie is aanzienlijk langer dan de andere iteraties. De eerste iteratie zal namelijk de basis vormen voor de gehele opdracht. Er zal eerst onderzoek gedaan worden naar de essentie van optische triangulatie om later in te zoomen op de verschillende onderdelen waar optische triangulatie uit bestaat. De iteratie bestaat uit drie weken onderzoek, twee weken ontwerp, één week implementatie en één week testen.




***Figuur P.1*** *De planning van de eerste iteratie. Het ontwikkelen van optische triangulatie.*

# 5. Onderzoeksfase

*van 17 februari 2014 t/m 14 maart 2014 (4 weken)*

In de eerste onderzoeksfase wordt onderzoek gedaan naar de kern van optische triangulatie. Er wordt gekeken naar wat optische triangulatie is, hoe het kan worden gerealiseerd en wat voor problemen er bij komen kijken. Er wordt onder andere onderzoek gedaan naar hardware, software en wiskunde die er bij komen kijken. Het onderzoek is een combinatie van een literatuurstudie (papers, artikels) en verschillende praktijktesten. Het doel van deze onderzoeksfase is om er achter te komen hoe optische triangulatie kan worden geïmplementeerd en wat erbij komt kijken. Om dit doel te bereiken zal naar antwoord worden gezocht op de volgende vragen:

*Wat is optische triangulatie? (**5.1**)*

*Hoe kan optische triangulatie worden gerealiseerd? (**5.2**, **5.3**, **5.4**)*

*Met wat voor hard- en software kan optische triangulatie worden geïmplementeerd ? (**5.5**, **5.6**)*

*Wat voor problemen komen er kijken bij de implementatie van optische triangulatie? (**5.7**)*

Tijdens deze onderzoeksfase wordt er gebruik gemaakt van twee verschillende opstellingen. De eerste opstelling is een microcontroller (Raspberry Pi), camera-module (Raspicam) en lijnlaser (5 mw). De tweede opstelling is een notebook (MSI-GP60), camera-module (USB-webcam) en lijnlaser (5 mw). De microcontroller ondersteunt het besturingssysteem Raspbian (Unix) en is via een HDMI-kabel aangesloten aan een beeldscherm. Er kan hierdoor direct op de microcontroller worden geprogrammeerd. De microcontroller heeft echter niet de rekenkracht die een notebook kan bieden. Hierdoor kost met name het compileren van de code veel tijd. De notebook beschikt ook over een Linux-distributie (Ubuntu) en alle vereiste software-bibliotheken zijn cross-platform. Hoewel het daadwerkelijke systeem uiteindelijk gebruik zal moeten maken van de microcontroller, is het veel efficiënter om op de notebook te ontwikkelen. Om deze reden zal daarom het merendeel van het onderzoek erop plaats vinden.

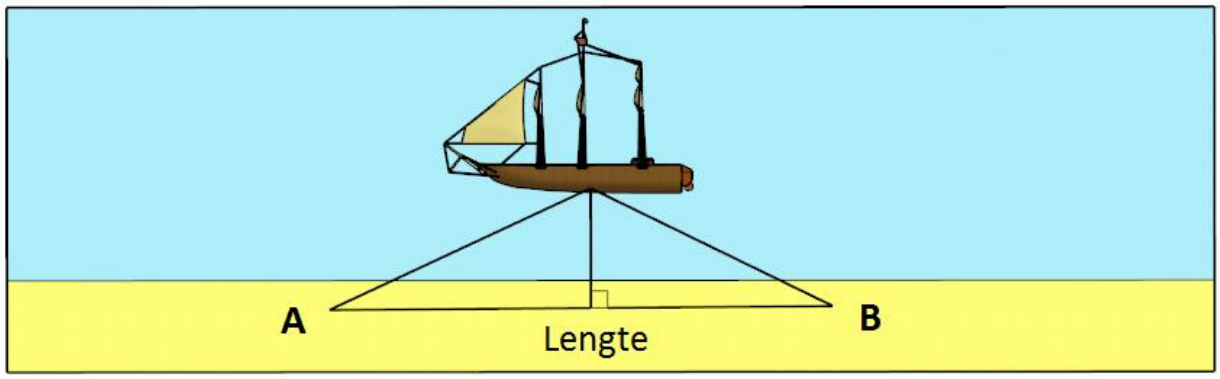
# 5.1 Optische triangulatie

Optische triangulatie is een techniek waarmee de afstand tot nabije objecten kan worden bepaald. Optische triangulatie maakt gebruik van een camera-module en lijnlaser om objecten waar te nemen en afstanden te bepalen. Hoewel er meerdere technieken bestaan om afstanden te bepalen door middel van lasers is optische triangulatie de enige techniek waarmee de afstand tot alle nabije objecten in een meting kan worden bepaald. Waar laser pointers en de time-of-flight techniek met name worden gebruikt om accuraat afstanden te bepalen tussen specifieke punten, kan optische triangulatie worden ingezet om snel een omgeving in kaart te brengen. Naast optische technieken zijn er andere technieken waarmee afstand kan worden bepaald. Een van deze technieken is ultrasone afstandsbepaling. Ultrasone afstandsbepaling heeft voor- en nadelen ten opzichte van optische afstandsbepaling. Ultrasone afstandsbepaling is bijvoorbeeld in staat om afstanden te bepalen onafhankelijk van de hoeveelheid (zon-) licht en de kleur van de objecten maar het is minder accuraat omdat geluid minder goed geconcentreerd kan worden dan licht. De reden waarom ultrasone afstandsbepaling voor Berkelaar Meet- en Regeltechniek geen goed alternatief is, is omdat er meerdere ultrasone sensoren nodig zijn om objecten te lokaliseren. Één ultrasone sensor kan alleen informatie verschaffen over de afstand van objecten, maar deze sensor kan geen informatie verschaffen over de locatie van deze objecten. Net zoals een oog is is om beelden te zien, maar niet in staat is om diepte waar te nemen. Er zijn meerdere sensoren vereist om een kaart te maken. Deze sensoren kunnen echter niet tegelijk te functioneren omdat de signalen elkaar storen. Optische triangulatie vereist maar één sensor en is daarmee zowel aanzienlijk goedkoper als sneller dan ultrasone afstandsbepaling.

Voordat er in zal worden gegaan op het onderzoek naar en de ontwikkeling van optische triangulatie is het belangrijk om te begrijpen hoe optische triangulatie gebruikt kan worden om afstanden te bepalen. In de volgende drie paragrafen zal de essentie van optische triangulatie daarom worden toegelicht. Optische triangulatie is een combinatie van de meetmethode “triangulatie” en het natuurkundige verschijnsel “parallax”.

### 5.1.1 Triangulatie

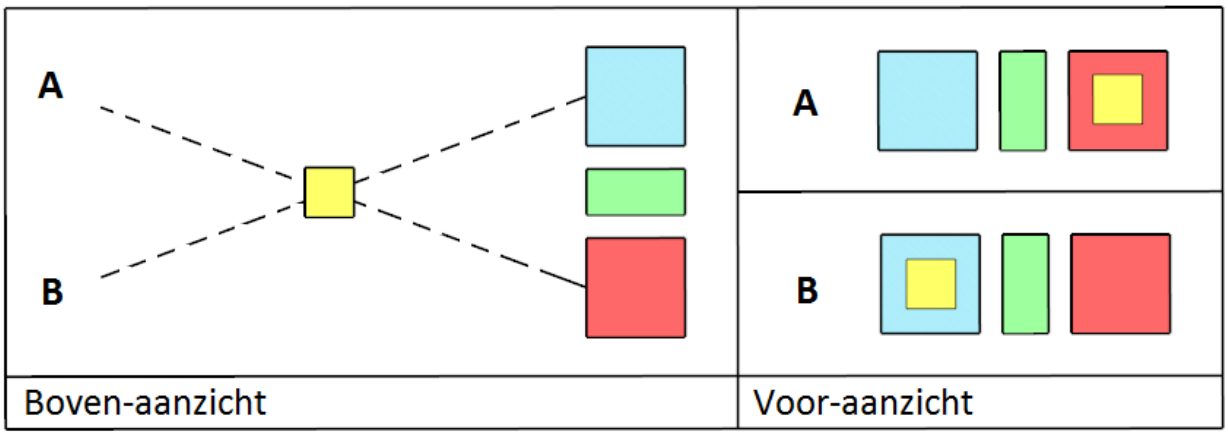
Triangulatie is een meetmethode die gebruik maakt van het gegeven wanneer er twee hoeken en een zijde van een driehoek bekend zijn, alle hoeken en zijdes kunnen worden berekend. De techniek is al honderden jaren oud. De techniek werd gehanteerd in de scheepsvaart om te bepalen hoe ver schepen zich bevonden van de kust. In figuur 5.1 is te zien dat de afstand van een schip tot de kust kan worden bepaald door er vanuit twee punten naar te kijken. De hoek waarin het schip vanuit de twee punten kan worden gezien en de afstand tussen de twee punten is genoeg informatie om alle andere hoeken en zijdes te kunnen berekenen (law of sines/cosines[1]).




***Figuur 5.1*** *De afstand tot de kust bepalen door middel van triangulatie.*

### 5.1.2 Parallax

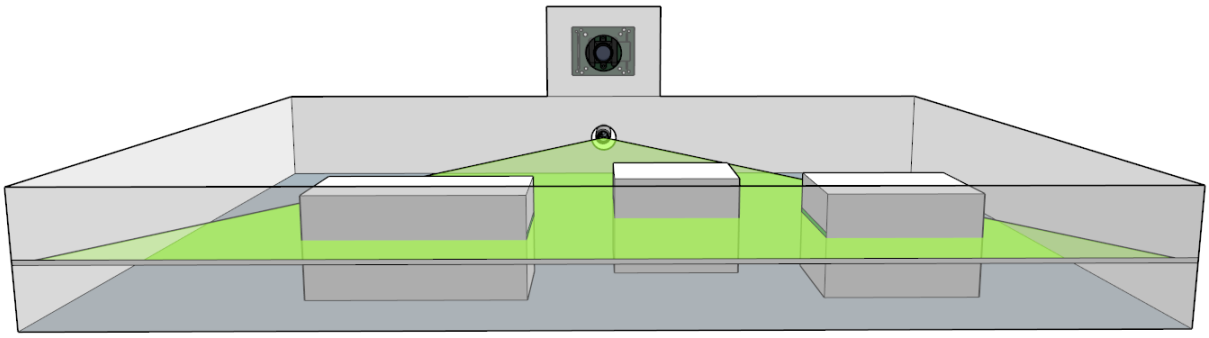
Parallax is een natuurkundig verschijnsel waardoor de schijnbare positie van een voorwerp verschilt door er vanuit verschillende perspectieven naar te kijken. In figuur 5.2 is te zien dat de schijnbare positie van het gele voorwerp verschilt door er vanuit twee perspectieven naar te kijken. Als de daadwerkelijke positie van het voorwerp bekend is, kan door middel van zijn schijnbare positie, informatie worden afgeleid over de daadwerkelijke positie van de voorwerpen op de achtergrond. Deze informatie (hoeken) kan vervolgens worden benut door met een meetmethode (triangulatie) de afstand tot deze voorwerpen te berekenen.




***Figuur 5.2*** *Parallax door vanuit twee perspectieven naar een voorwerp te kijken.*

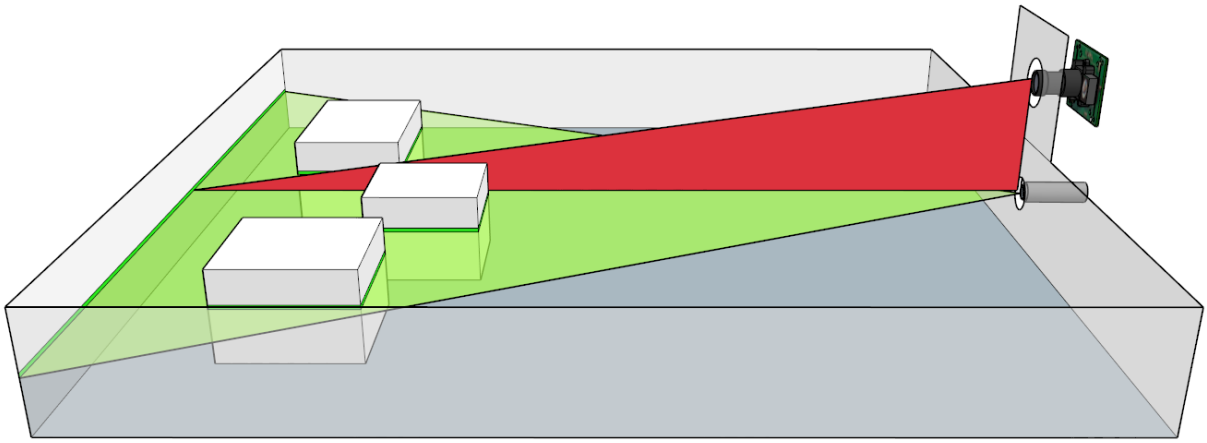
### 5.1.3 Optische triangulatie

Optische triangulatie is een techniek waarbij door middel van het parallax-verschijnsel hoeken worden bepaald om vervolgens door middel van triangulatie afstanden te kunnen berekenen. In figuur 5.3 is een opstelling weergegeven waarin door middel van optische triangulatie de afstand tot de afgelegen voorwerpen kan worden bepaald. De opstelling bestaat uit een lijnlaser en een camera-module. De enige eisen zijn dat de camera-module in staat is om het licht van de lijnlaser waar te nemen en dat de hoeken waarin ze gepositioneerd zijn bekend zijn.



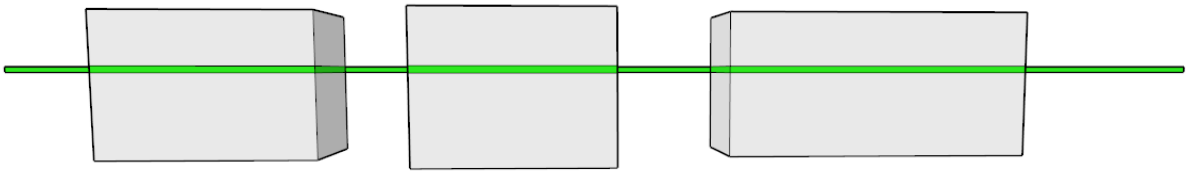
***Figuur 5.3*** *Voor-aanzicht van optische triangulatie opstelling. (lijnlaser, camera-module)*

De eerste stap in het bepalen van de afstand is het definiëren van een driehoek. In figuur 5.4 is een driehoek weergegeven. De hoeken van het driehoek raken de lijnlaser, de camera-module en de muur. De afstand tot de muur (en andere voorwerpen) kan worden bepaald door de onderzijde van het driehoek te berekenen. Om alle zijdes van een driehoek te kunnen berekenen moeten minstens twee hoeken en een zijde van het driehoek bekend zijn. Er zijn echter maar een hoek en een zijde bekend. De bekende hoek is de hoek waarin de camera-module boven de lijnlaser is geplaatst (90°) en de bekende zijde is het hoogteverschil tussen de lijnlaser en de camera-module. De derde hoek die nodig is om de afstand te bepalen kan worden berekend door middel van parallax.



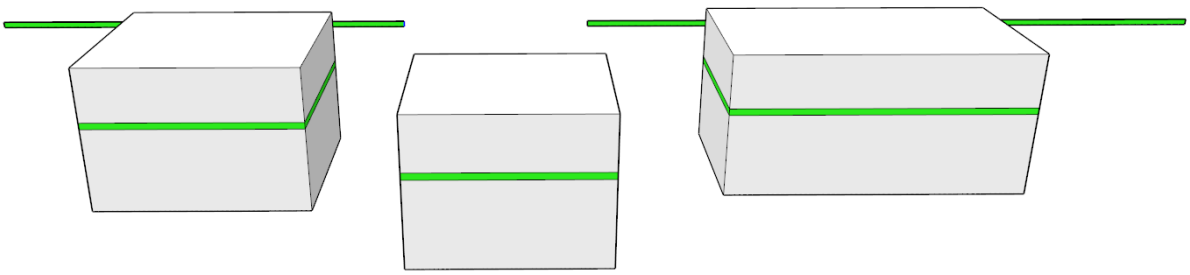
***Figuur 5.4*** *Het te bepalen driehoek tussen de muur, de lijnlaser en de camera-module.*

In figuur 5.5 is het voor-aanzicht  weergegeven vanuit het perspectief van de laser. Het licht is nu een rechte lijn.



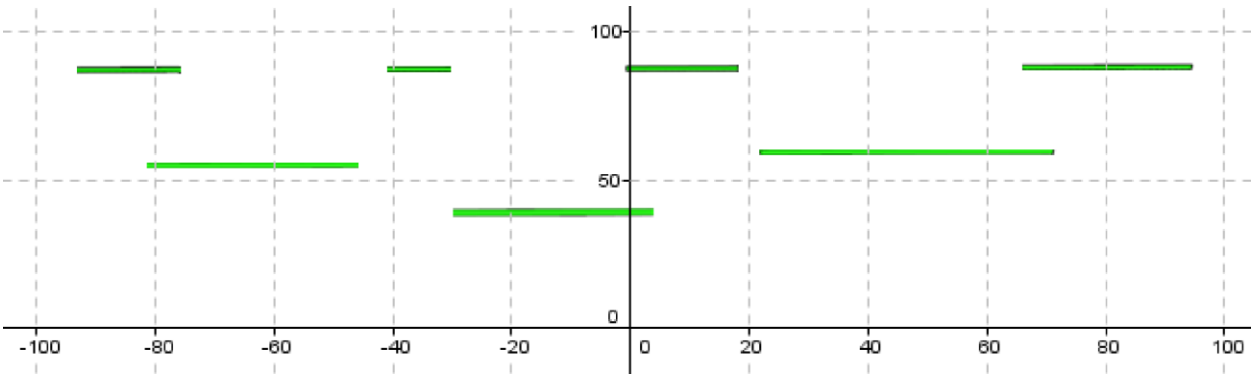
***Figuur 5.5*** *Het licht van de laser is vanuit het perspectief van de laser een rechte lijn.*

Figuur 5.6 geeft het voor-aanzicht vanuit het perspectief van de camera-module weer. De lijn is nu gebroken.



***Figuur 5.6*** *Het licht van de laser is vanuit het perspectief van de camera-module een gebroken lijn.*

In figuur 5.7 is een voorbeeld-foto vanuit het perspectief van camera-module weergegeven. De positie van de lijn in de foto is afhankelijk van de afstand tot de lijn in het echt. De lijn kan naar boven en naar onder verschuiven tot de lijn buiten de beeldhoek van de camera-module valt. Er kan daarom gesteld worden dat de derde hoek bekend is als de lijn op de boven- of onderkant van de foto valt. De derde hoek is dan namelijk direct af te leiden van de verticale beeldhoek van de camera-module. Iedere andere positie is echter ook te berekenen. Het verband tussen de positie in de foto en de hoek is namelijk lineair. Oftewel, door een foto te maken en de positie van de lijn in de foto te bepalen, kan de derde afstand worden bepaald (mits de positionering van de camera-module bekend is).




***Figuur 5.7*** *Het licht van de laser zoals het op een foto zou kunnen worden waargenomen.*

Nu de afstand tot de voorwerpen kan worden berekend is de daadwerkelijke locatie nog niet bekend. Daarvoor moet ook de afstand van de voorwerpen tot het middelpunt bepaald worden. Deze afstand kan tevens met triangulatie worden bepaald. De bekende hoeken zijn de 90° hoek tussen de lijnlaser en de camera-module en de verticale beeldhoek van de camera. De bekende zijde is de eerder berekende afstand tot het voorwerp.

Een computer is in staat om optische triangulatie uit te voeren in drie stappen. De eerste stap is het verzamelen van beeldmateriaal (foto's) van het licht van de laser. De tweede stap is het bewerken van het beeldmateriaal om de lijnen te kunnen onderscheiden. De derde stap is het berekenen van de afstand aan de hand van de lijnen. In paragraaf 5.2, 5.3 en 5.4 worden deze stappen nader toegelicht.

## 5.2 Beeldmateriaal verzamelen

De eerste stap in het uitvoeren van optische triangulatie is het verzamelen van beeldmateriaal. De kwaliteit van het beeldmateriaal is bepalend voor de betrouwbaarheid van de triangulatie. De kwaliteit wordt bepaald door drie factoren. Deze factoren zijn: de camera-module, de camera-instellingen en de laser.

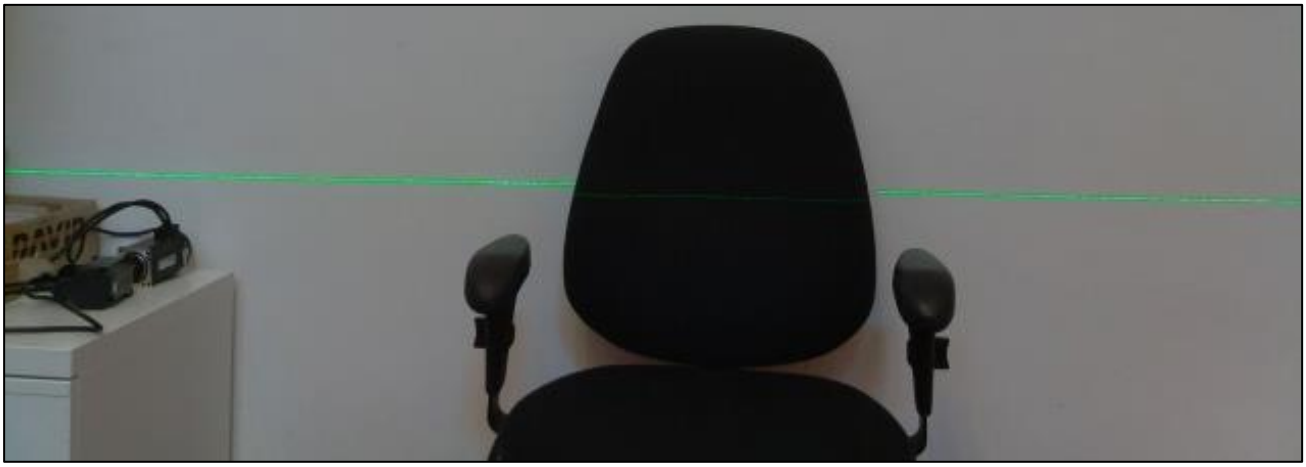
### 5.2.1 Camera-module

De kwaliteit van het systeem is sterk afhankelijk van de kwaliteit van de camera-module. De kosten van het systeem zijn echter ook van belang. Er zal daarom een goede afweging gemaakt moeten worden tussen de prijs en kwaliteit van de camera-module. Een voorbeeld van een betaalbare maar goede camera-module is de Raspicam. Deze camera-module is speciaal ontwikkeld om gebruikt te worden door de Raspberry Pi (zie 5.5).

### 5.2.2 Camera-instellingen

De camera-instellingen van de Raspicam worden aangestuurd door een driver. Helaas laat deze driver veel te wensen over. Sommige functies zijn slecht of helemaal niet geïmplementeerd. Gebruikers zijn niet in staat hier veel aan te doen. De camera-instellingen worden namelijk afgehandeld door de GPU en de code van de GPU is niet open-source. In principe zou de GPU omzeild kunnen worden door direct de camera-module aan te spreken en de grafische bewerkingen zelf uit te voeren, maar een ARMv6 processor is niet in staat om zulke zware bewerkingen uit te voeren.

Op 28 februari 2014 heeft Broadcom aangegeven bereid te zijn om de code van de GPU vrij te willen geven. De complexiteit van de problemen moet echter niet worden onderschat. De problemen zijn bij Broadcom al langer bekend, toch zijn ze nog niet in staat geweest om de problemen op te lossen.



***Figuur 5.8*** *Een lage belichting door de automatische belichting van de Raspicam.*

Een van de meest essentiële camera-instellingen is de belichting. De belichting is bepalend voor de hoeveelheid licht die op een foto wordt weergegeven. Het streven is om de hoeveelheid licht zo te doseren, dat al het licht van de laser goed waarneembaar is, maar de foto niet overbelicht is. Het is namelijk lastig om op een overbelichte foto onderscheid te maken tussen het licht van de laser en het licht van andere lichtbronnen. In sommige situaties is het niet mogelijk om aan beide eisen te voldoen De intensiteit van het licht van de laser kan in een foto soms te

veel verschillen. Moderne technieken zoals High Dynamic Ranging (HDR) en Auto Exposure Bracketing (AEB) gaan hiermee om door een aantal foto's te maken met verschillende belichtingen. Deze verschillende foto's worden vervolgens gecombineerd. Het resultaat is een ideaal belichte foto.

De Raspicam is niet in staat om gebruik te maken van de moderne technieken vanwege een te lage FPS. Eer de Raspicam een tweede of derde foto van een beeld kan maken is het beeld al weer veranderd. De Raspicam zal daarom gebruik moeten maken van een eendimensionale, lineaire belichting. In de documentatie van de Raspicam worden drie verschillende belichtingsvormen naar voren gebracht. Een constante stand, een automatische stand en een aantal automatische standen voor speciale scenario's (*night, snow, beach, sports, backlight, spotlight, verylong, fixedfps, antishake* en *fireworks*).

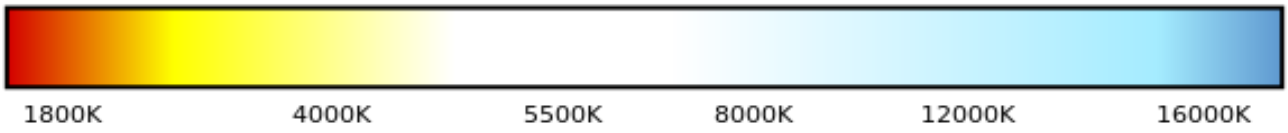
***Figuur 5.9*** *Een hoge belichting door de automatische belichting van de Raspicam.*

Hoewel de constante stand staat aangegeven in de documentatie van de Raspicam is hij niet operationeel. Broadcom heeft aangegeven dat sommige functies vanwege regressie niet meet functioneren. De Raspicam is daarom toegewezen op het gebruik van de automatische stand. Helaas is de automatische stand van de Raspicam niet zo betrouwbaar. De automatische stand zal trachten om voor iedere foto een optimale balans te vinden. Soms is de Raspicam echter niet in staat een goede keuze te maken. In figuur 5.8 is een afbeelding weergegeven van een onderbelichte foto. In figuur 5.9 is een afbeelding weergegeven van een overbelichte foto. De Raspicam is zonder moderne technieken zoals HDR en AEB niet in staat om alle delen van de foto te belichten en kan daarom geen goede balans vinden.

In figuur 5.8 en 5.9 is te zien dat de kleur en het materiaal van de achtergrond bepalend zijn voor de intensiteit van het licht. Door de foto op verschillende manieren te belichten kan het licht van de laser toch overal worden waargenomen. In figuur 5.10 is een ander probleem weergegeven. Een overvloed aan licht. De overvloed aan licht is niet direct te zien. De Raspicam laat ter compensatie veel minder licht door. De Raspicam is echter niet in staat om onderscheid te maken tussen het licht van de laser en andere lichtbronnen. Hierdoor is zowel de overvloed aan licht als het licht van de laser niet goed meer waar te nemen. Hoe meer licht, des te minder goed het licht van de laser waargenomen kan worden. Alleen moderne camera-modules zijn in staat om deze problemen te vermijden (HDR, AEB).

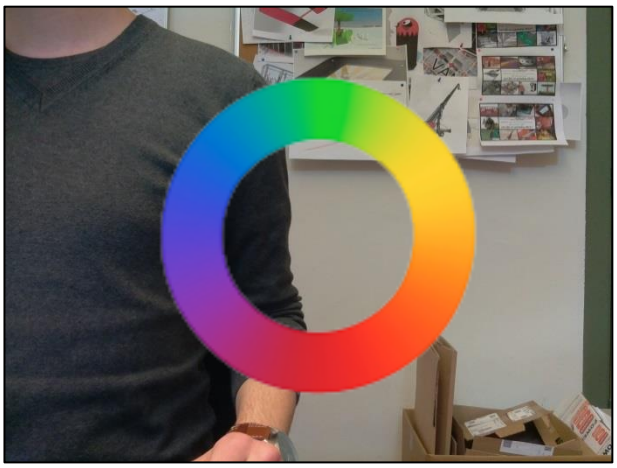
***Figuur 5.10*** *Een overvloed aan zonlicht en een lage belichting door de automatische belichting van de Raspicam.*

Lichtbronnen hebben naast een intensiteit ook een kleurtemperatuur. De kleurtemperatuur van een lichtbron bepaalt hoe het licht kan worden waargenomen. In figuur 5.11 zijn de kleurtemperaturen van licht weergegeven in Kelvin. Het is echter niet de bedoeling dat een foto er bij verschillende lichtbronnen anders uit gaat zien. De camera-instelling die hier verantwoordelijk voor is heet de witbalans. Net zoals bij een aantal andere instellingen is alleen de automatische stand beschikbaar.




***Figuur 5.11*** *De kleurtemperatuur van licht bij verschillende temperaturen (Kelvin).*

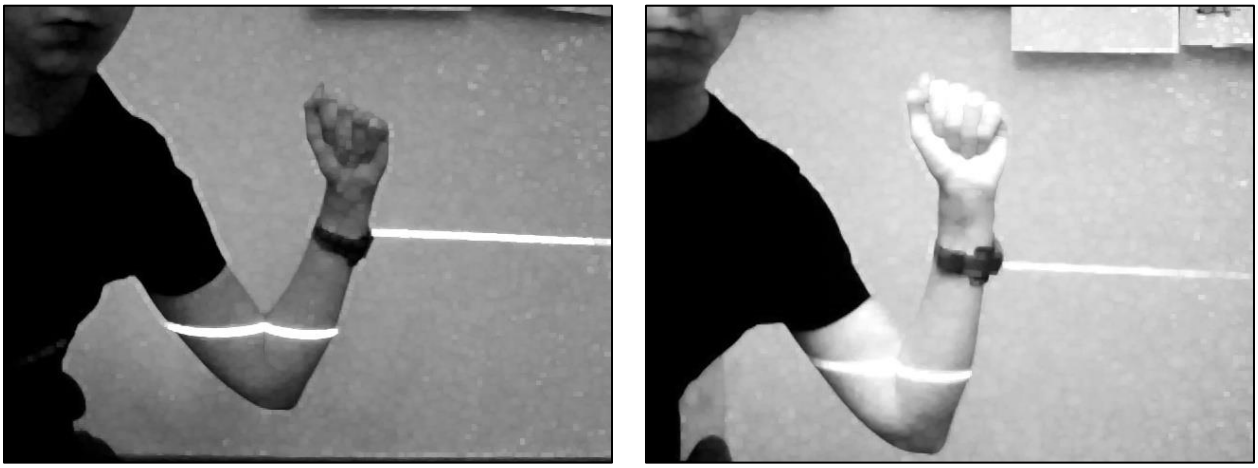
De automatische stand van de witbalans is helaas niet zo betrouwbaar. De witbalans zal bij het waarnemen van een overvloed van een bepaalde kleur compenseren door de complementaire kleur te versterken. Als een foto een overvloed aan blauw bevat zal de witbalans compenseren door oranje tinten te versterken. Hoe meer er van een kleur in beeld is, hoe meer de witbalans zal compenseren.



***Figuur 5.12*** *Links: originele foto met complementaire kleuren. Rechts: onstabiele witbalans*

5.2.3 Laser

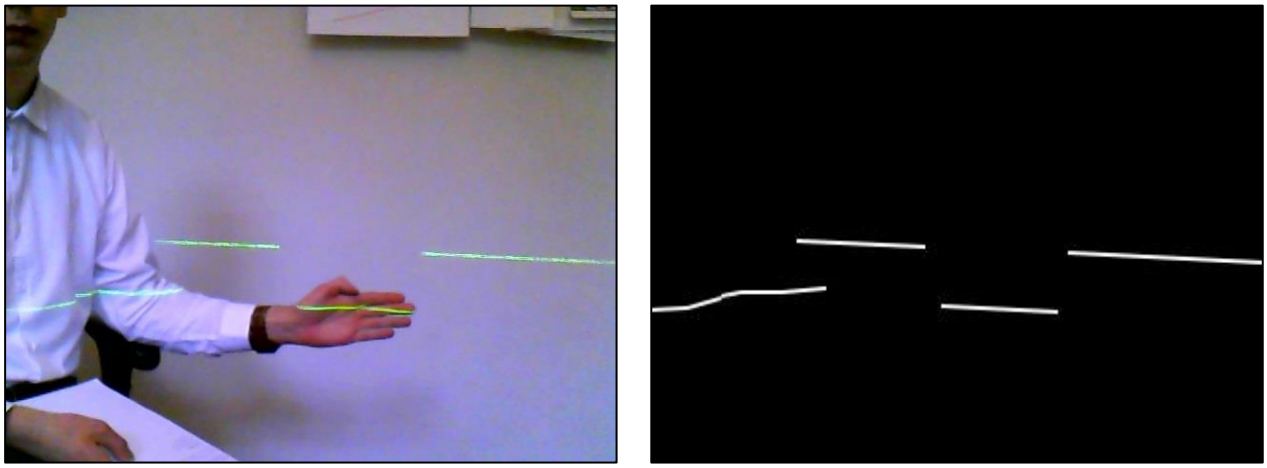
De laser heeft twee eigenschappen die van invloed zijn op de kwaliteit van het beeldmateriaal. De intensiteit (mw/cm²) en de kleur (nm). In figuur 5.13 zijn deze eigenschappen weergegeven. De eerste afbeelding toont het groene RGB-kanaal van een foto van een groene laser (532 nm). De tweede afbeelding toont het rode RGB-kanaal van een foto van een rode laser (671 nm). Hoewel het licht op beide foto's goed te zien is, is de camera-module beter in staat om het groene licht te onderscheiden. Een hogere intensiteit zou de camera-module in staat stellen om het rode licht ook te kunnen onderscheiden, maar het gebruik van meer dan 5 mw/cm² is niet verantwoord.



***Figuur 5.13*** *Links: groene laser (groen RGB-kanaal). Rechts: rode laser (rood RGB-kanaal)*

# 5.3 Beeldmateriaal bewerken

De tweede stap in het uitvoeren van optische triangulatie is het bewerken van beeldmateriaal om het licht van de laser te kunnen onderscheiden. Lijnen zijn te onderscheiden aan de hand van drie eigenschappen: de intensiteit, de kleur en de vorm. Lijnen detecteren aan de hand van minder dan drie eigenschappen zal resulteren in ruis. Het doel van het bewerken van het beeldmateriaal is een binaire representatie van de foto waarin het licht van de laser duidelijk te zien is. In figuur 5.14 (rechts) is het ideaalbeeld weergegeven.



***Figuur 5.14*** *Links: oorspronkelijke weergave van de foto. Rechts: optimale binaire weergave.*

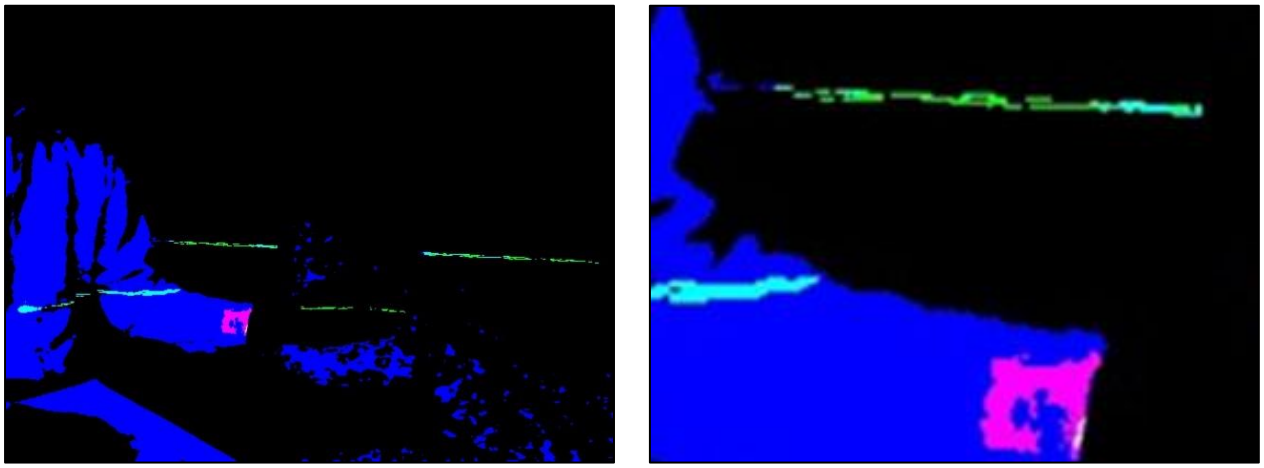
5.3.1 Intensiteit

Een eigenschap waarmee lijnen onderscheiden kunnen worden is de intensiteit van het licht. Hoe groter de intensiteit, des te meer contrast met de achtergrond. De laser in figuur 5.14 heeft een vermogen van 5 milliwatt. Hiermee ontstaat er een aanzienlijk contrast met de achtergrond. Toch is het niet mogelijk om aan de hand van het contrast alleen de lijnen te onderscheiden. Er zijn namelijk andere lichtbronnen die een vergelijkbaar contrast creëren. De intensiteit is namelijk niet alleen afhankelijk van het vermogen van de lichtbron, maar ook van het reflecterend vermogen van de achtergrond.

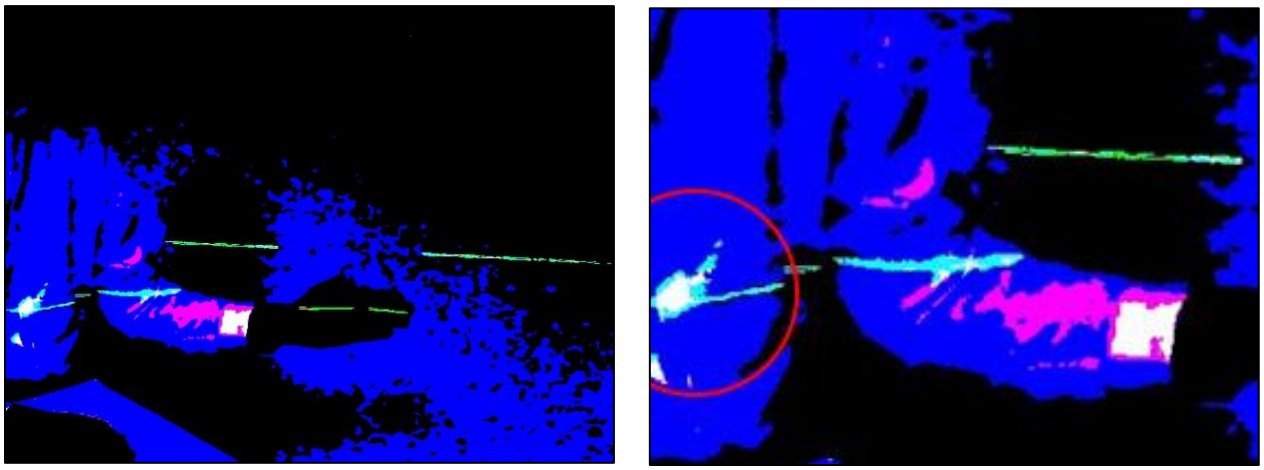
In figuur 5.15 (links) is een threshold weergegeven van de originele afbeelding. Een threshold is een segmentatie aan de hand van intensiteit. De computer neemt drie verschillende segmenten waar. Het groene segment is het licht van de laser. Het blauwe segment is gereflecteerd zon- en lamplicht en het paarse segment is dat eveneens. Hoewel de laser een grotere intensiteit heeft dan het zon- en lamplicht worden deze toch weergegeven. De afbeelding geeft namelijk maar een reflectie van het licht weer en deze reflectie is sterk afhankelijk van het reflecterend vermogen van de achtergrond. Het witte overhemd heeft een groter reflecterend vermogen dan de witte achtergrond. In figuur 5.15 (rechts) is te zien dat er daardoor eigenlijk vier segmenten ontstaan. De reflectie van de laser op de muur heeft een andere intensiteit dan de reflectie van de laser op het overhemd. Hoewel de verschillende reflecties complexiteit toevoegen, is het licht van de laser nog steeds goed te onderscheiden.

Een ander probleem ontstaat door absorptie van de achtergrond. In de threshold van de afbeelding is niet de hele lijn waar te nemen. Alleen de binnenkant van de lijn kan goed worden waargenomen. De binnenkant van de lijn heeft namelijk een hogere intensiteit dan de buitenzijde. Het is mogelijk om de buitenzijde aan de afbeelding toe te voegen door de nauwkeurigheid van de threshold te verkleinen. Hierdoor ontstaat echter een ander probleem.

Door de nauwkeurigheid te verkleinen zal ook meer zon- en lamplicht worden toegelaten. In figuur 5.16 is te zien dat het niet meer mogelijk is om onderscheid te maken tussen zon- en lamplicht en licht van de laser.



***Figuur 5.15*** *Links: threshold weergave van de foto. Rechts: vier verschillende threshold segmenten.*

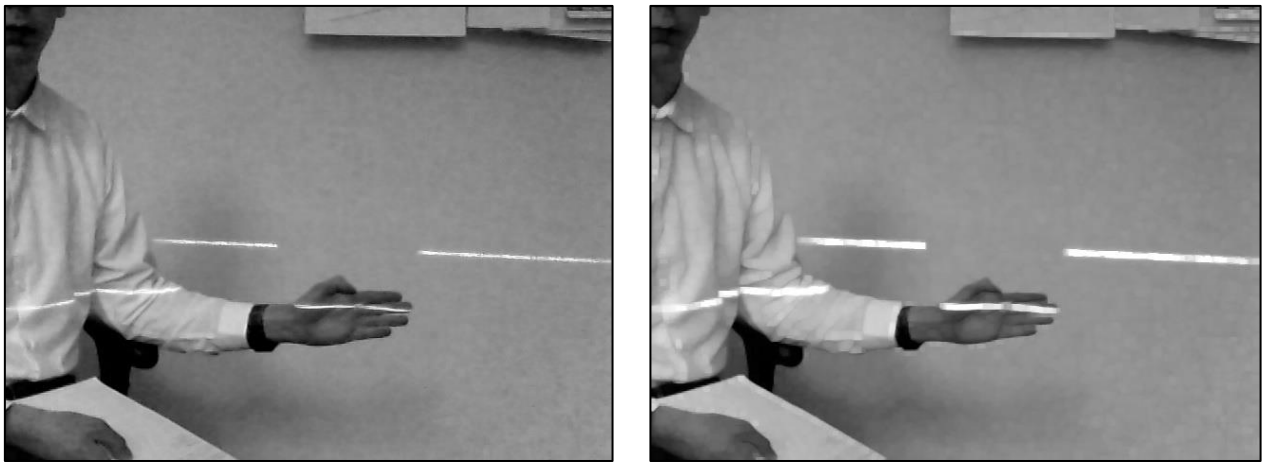


***Figuur 5.16*** *Links: threshold met kleinere nauwkeurigheid. Rechts: threshold met ruis van lichtbronnen*

5.3.2 Kleur

Een andere manier om onderscheid te maken tussen lijnen en de achtergrond is door middel van kleur. Een lijn bestaat niet alleen uit verschillende intensiteiten, maar ook uit verschillende kleuren. Afhankelijk van het reflecterend vermogen van de achtergrond kan de kleur variëren van donkergroen tot bijna wit. Hierdoor is het niet zinloos om een threshold uit te voeren aan de hand van RGB-waardes. De ruimte tussen de minimale en maximale RGB-waarde zal zo groot moeten zijn dat er een overvloed aan ruis doorheen zal komen. Het is beter om de afbeelding op te splitsen in drie RGB kanalen. In het rode kanaal komen alle rode kleuren naar voren, in het groene kanaal alle groene kleuren en in het blauwe het kanaal alle blauwe kleuren. In figuur 5.17 / 5.18 is te zien dat de (groene) laser aanzienlijk beter zichtbaar is in het groene kanaal. In figuur 5.18 is tevens te zien dat het contrast met de achtergrond vergroot kan worden door de afbeelding te dilateren.

***Figuur 5.17*** *Links: groene laser in rode RGB-kanaal. Rechts: groene laser in blauwe RGB-kanaal.*

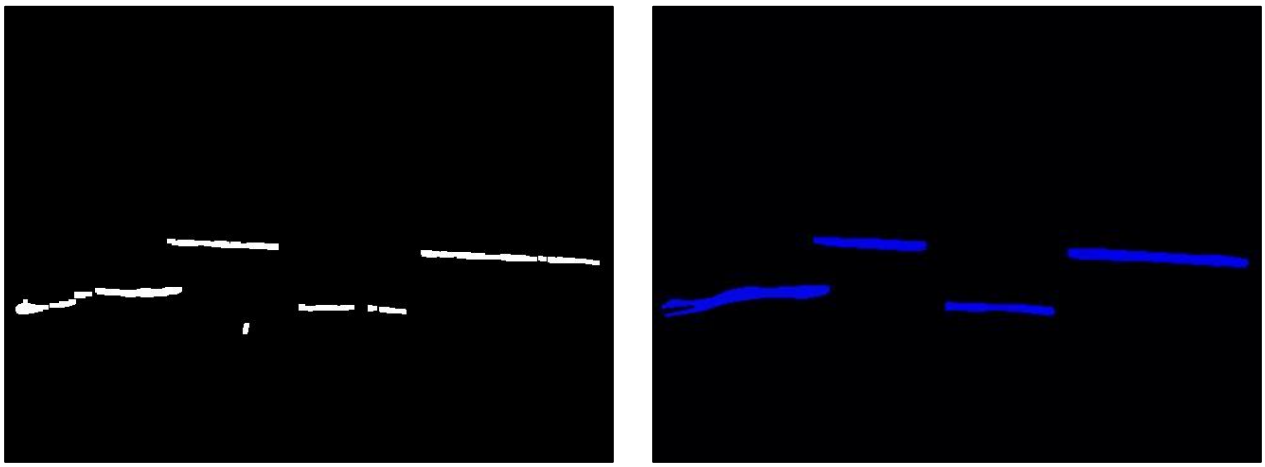


***Figuur 5.18*** *Links: groene laser in groene RGB-kanaal. Rechts: gedilateerde weergave groene kanaal.*

Door de intensiteit en kleur filters te combineren ontstaat er een binaire afbeelding. In figuur 5.19 is deze afbeelding weergegeven (links). In figuur 5.14 is de ideale afbeelding weergegeven (rechts). Het verschil tussen de twee afbeeldingen is dat de echte afbeelding bestaat uit gebroken lijnen en ruis. Het zijn problemen die ontstaan door lijnen te onderscheiden aan de hand van intensiteit. Omdat er sprake is van zowel een te kort aan lijn (gebroken lijnen) als een overvloed aan lijn (ruis) heeft het geen zin om de threshold te vergroten of verkleinen.

### 5.3.3 Vorm

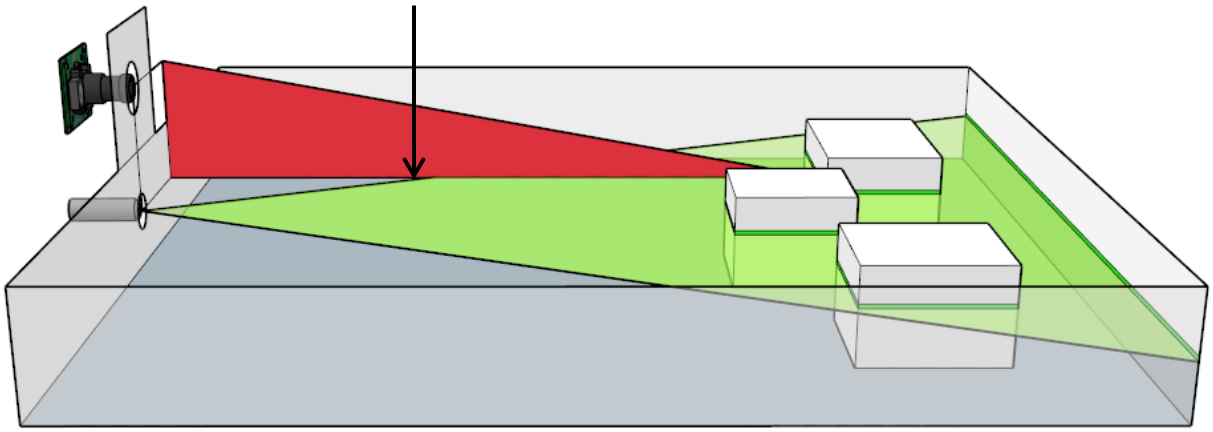
Een andere manier waarmee lijnen herkend kunnen worden is de vorm. Een hough transformatie is een methode waarmee bepaalde vormen in een afbeelding kunnen worden gedetecteerd. Een hough-lines transformatie is in staat om hele specifieke lijnen te onderscheiden. De methode is in staat om lijnen te onderscheiden aan de hand van de rotatie, minimale lengte en maximale lengte. Het resultaat van een hough-lines transformatie is weergegeven in 5.19 (rechts). De transformatie is in staat om de gebroken lijnen te repareren en de aanwezige ruis te negeren. Het enige probleem is dat de hough-lines transformatie geen onderscheid kan maken tussen het einde van een lijn en het einde van de afbeelding. Hoewel de lijnen eigenlijk de randen zouden moeten raken, is dit niet het geval. Toch is het resultaat van de hough-lines transformatie bijna identiek aan de ideale binaire afbeelding.



***Figuur 5.19*** *Links: resulterende binaire afbeelding. Rechts: gedetecteerde lijnen (hough-lines transformatie).*

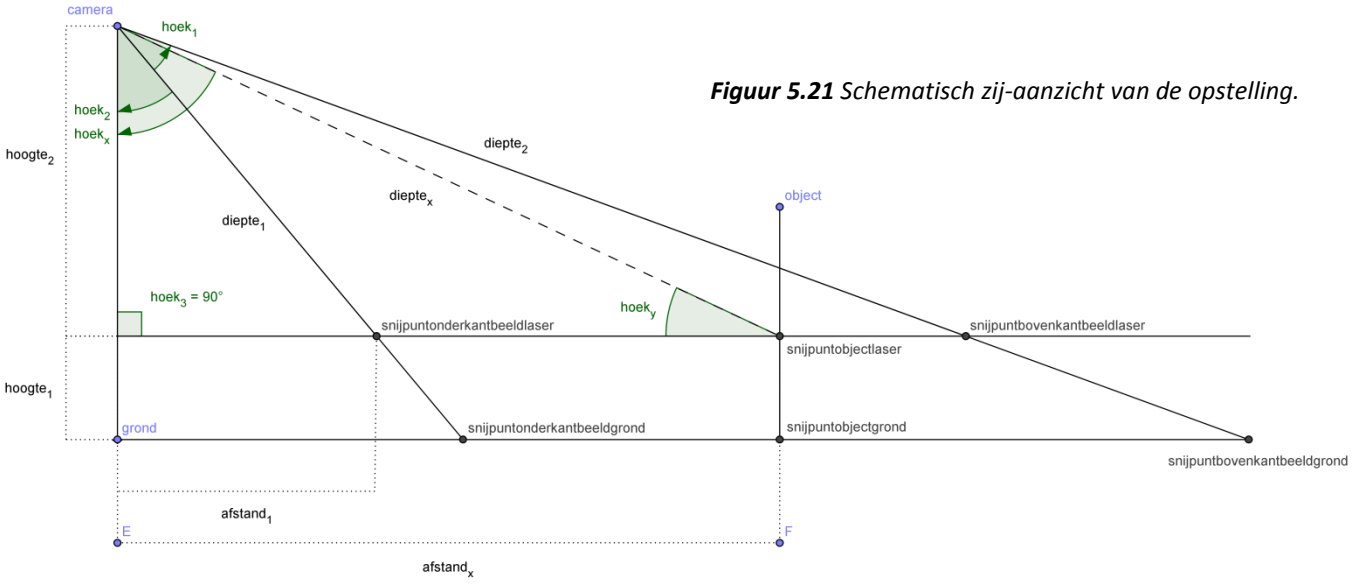
# 5.4 Afstand berekenen

De derde en laatste stap in het uitvoeren van optische triangulatie is het berekenen van de afstanden. De afstanden kunnen worden geïnterpreteerd als coördinaten in een assenstelsel waarin de camera-module in de oorsprong en loodrecht op de x-as is geplaatst. De waarde *afstandx* is daarom niet de afstand van de laser tot het object, maar de afstand van de x-as tot het object. De waarde *afstandx* is weergegeven in figuur 5.20.



***Figuur 5.20*** *Het zij-aanzicht van de opstelling met daarin afstandx aangegeven met een pijl.*

In figuur 5.21 is de waarde *afstandx* op een schematische wijze weergegeven. Deze waarde kan worden berekend door middel van triangulatie. Om triangulatie te kunnen uitvoeren moeten twee hoeken en een zijde van en driehoek bekend zijn. De bekende hoek is *hoek3*, de hoek waarin de camera-module boven de lijnlaser is geplaatst. De bekende zijde is *hoogte2*, het hoogteverschil tussen de camera-module en de lijnlaser. De hoek die moet worden bepaald is *hoekx*. Deze hoek kan worden bepaald door middel van het *rijnummer* van de pixel in de foto (verticale positie). Aan de hand van het *rijnummer* en het *maximale aantal rijen* kan bepaald worden waar in de (verticale) beeldhoek van de camera-module het licht van de laser (en het object) is waargenomen.




***Figuur 5.21*** *Schematisch zij-aanzicht van de opstelling.*

De waarde *hoekx* kan berekend worden door middel van de volgende formule:

$$hoekx = \left( (hoek1 - hoek2) * \left( \frac{rijnummer}{aantalrijen} \right) \right) + hoek2$$

Aan de hand van hoekx kan nu door middel van triangulatie de waarde van afstandx worden bepaald:

$$afstandx = \tan(hoekx) * hoogte2$$

De waarde *afstandy* is weergegeven in figuur 5.22. De waarde *afstandy* is de afstand tussen het object en de y-as. De waarde *afstandy* kan op dezelfde wijze worden berekend als *afstandx*. Om triangulatie te kunnen uitvoeren moeten twee hoeken en een zijde van de driehoek bekend zijn. De bekende hoek is *hoek3*, de hoek waarin de camera-module boven de laser is geplaatst. De bekende zijde is *afstandx*, de eerder berekende afstand van de x-as tot het object. De onbekende hoek is *hoeky*. Deze hoek kan worden bepaald door middel van het *kolomnummer* van de pixel in de foto (horizontale positie). Aan de hand van het kolomnummer en het *maximale aantal kolommen* kan bepaald worden waar in de (horizontale) beeldhoek van de camera-module het licht van de laser (en het object) is waargenomen. De waarde *hoeky* kan door middel van de volgende formule worden berekend:

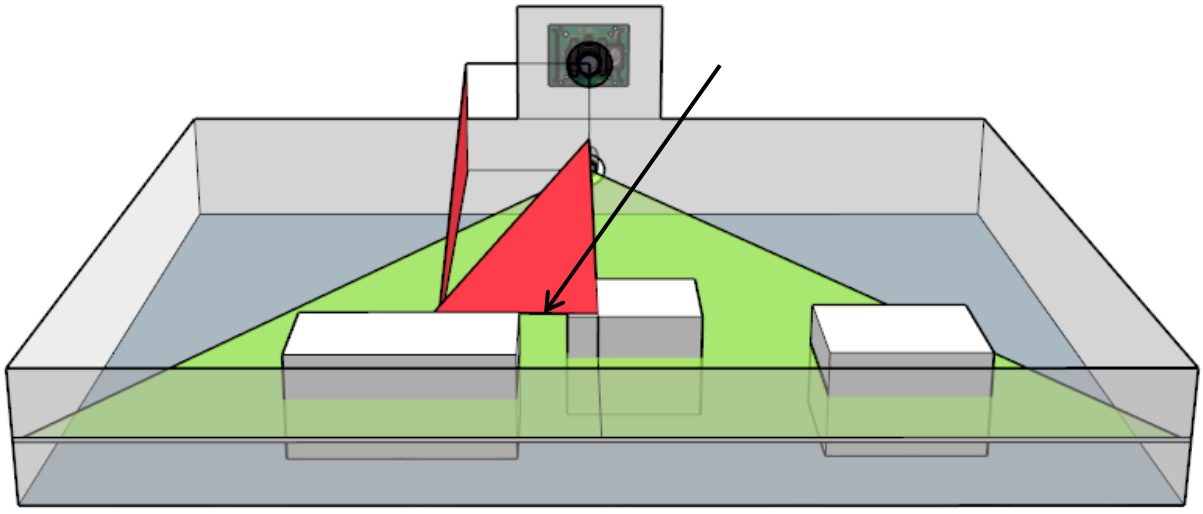
$$hoeky = \left( \frac{horizontalebeeldhoek}{aantalkolommen} \right) * kolomnummer$$

Aan de hand van *hoeky* kan nu door middel van triangulatie de waarde van *afstandy* worden bepaald:

$$afstandy = \tan(hoeky) * afstandx$$

De waarde hoekx is eveneens de waarde *coördinaatx*. De waarde *hoeky* is niet automatisch een coördinaat. Om de waarde *hoeky* om te zetten in een coördinaat in het assenstelsel kan de volgende formule worden gehanteerd:

$$coördinaaty = -(\tan\left(\frac{hoek4}{2}\right) * afstandx) + hoeky$$



***Figuur 5.22*** *Het voor-aanzicht van de opstelling met daarin afstandy aangegeven met een pijl.*

# 5.5 Hardware

Optische triangulatie kan worden gerealiseerd door middel van een camera-module, een microcontroller en een lijnlaser. De kwaliteit van deze componenten is bepalend voor de kwaliteit van de afstandsbepaling. Omdat Berkelaar Meet- en Regeltechniek plannen heeft om deze techniek zelf te ontwikkelen (en verkopen) zijn ze met name geïnteresseerd in de mogelijkheid om deze techniek te ontwikkelen met betaalbare hardware. Om deze reden heeft Berkelaar Meet- en Regeltechniek al een aantal onderdelen geselecteerd. Deze onderdelen zijn: een microcontroller (Raspberry Pi), een camera-module (Raspicam) en een laser (5 mw lijnlaser).

5.5.1 Raspberry Pi

De Raspberry Pi (Rpi) is ontwikkeld om kinderen kennis te laten maken met programmeren zonder daar veel kosten bij te laten komen kijken. Hierdoor heeft de Rpi een uitstekende prijs-kwaliteitverhouding. De Rpi is ook makkelijk te gebruiken. Door de aanwezigheid van HDMI-, USB-, en ethernet-poorten en een besturingssysteem kan er direct op de microcontroller worden ontwikkeld.

In de onderstaande tabel zijn de specificaties van de Rpi weergegeven:

| | |
|---|---|
| Richtprijs | € 35 |
| SOC | Broadcom BCM2835 (CPU + GPU + DSP) |
| CPU | 700 MHz ARM1176JZF-S core ARM11 |
| GPU | Broadcom VideoCore IV |
| Harde schijf | SD-kaart van 2GB of meer |
| Geheugen (SDRAM) | 512 MiB gedeeld met GPU |
| USB-poorten | 2 USB2.0-poorten |
| Video-uitgangen | Composite RCA (PAL en NTSC), HDMI , LCD via DSI |
| Audio-uitgangen | 3,5 mm-jack, HDMI |
| Netwerk | 10/100 ethernet |
| Energieverbruik | 700 mA (3,5 W) |
| Stroomvoorziening | 5 V via micro-USB of optionele GPIO header |

***Figuur 5.23*** *Specificaties van de Rasberry Pi microcontroller.*

De meest limiterende factor van de Rpi is de rekenkracht. Het is voornamelijk te merken aan het aantal ‘frames per second’ dat de Rpi kan behalen bij het verzamelen van beeldmateriaal.

In de onderstaande tabel is het gemiddelde aantal ‘frames per seconde’ bij verschillende resoluties weergegeven.

| | |
|---|---|
| 320 x 240 | 30,78 |
| 640 x 480 | 12,98 |
| 1280 x 960 | 3,60 |
| 1920 x 1080 | 2,08 |

***Figuur 5.24*** *Frequenties waarmee de Raspberry Pi foto’s kan maken bij verschillende resoluties.*

### 5.5.2 Raspicam

De Raspicam is een camera-module die speciaal ontwikkeld is om te werken op een Rpi. De Raspicam heeft een betere kwaliteit dan USB-camera's uit zijn prijscategorie omdat de Raspicam in staat is om door een flatcable direct te communiceren met de GPU van de Rpi. De GPU doet het meeste werk, hierdoor worden er aan de camera-module minder eisen gesteld. De flatcable heeft echter ook een nadeel. Veel open-source bibliotheken (voor beeldbewerking) zijn alleen in staat om te functioneren met een USB-camera. Het formaat van het beeldmateriaal moet eerst omgezet worden door de GPU voordat de bibliotheken in staat zijn om er iets mee doen. Dat betekend dat de bibliotheken moeten samenwerken met de driver van de GPU. De driver is hier echter niet op bedacht. De driver is niet open-source en heeft maar een aantal publieke functies. De bibliotheken zijn daarom volkomen afhankelijk van deze publieke functies. De slecht of helemaal niet functionerende functies moeten daarom voor lief worden genomen. De GPU zou in principe vermeden kunnen worden door het beeldmateriaal met de CPU zelf te verwerken, maar de CPU heeft niet dezelfde efficiëntie als de GPU en is hierdoor vele malen langzamer. Deze situatie kan in de toekomst echter verbeteren. Op 28 februari 2014 heeft Broadcom aangegeven bereid te zijn om de code van de GPU vrij te willen geven. Het is daarom een kwestie van tijd voordat de slecht en niet functionerende functies worden verbeterd. De Raspicam blijft interessant.

In de onderstaande tabel zijn de specificaties van de Raspicam weergegeven:

| | |
|---|---|
| Sensor | OV5647 (QSXGA) (5-MP CMOS) |
| Sensorgrootte | 3.67 x 2.74 mm |
| Resolutie | 2592 x 1944 |
| Pixelgrootte | 1.4 x 1.4 um |
| Lens | F = 3.7 mm (f/2.9) |
| Beeldhoek | 54 x 41 ° |
| Gezichtsveld | 1 : 0,677 : 1 |
| SLR lens equivalent | 35 mm |
| Gefixeerde focus | > 1 m |
| Video resolutie | 1920 x 1080 (30 fps) (H.264) |
| Grootte | 25 x 24 mm |

***Figuur 5.25*** *Specificaties van de Raspicam camera-module.*

### 5.5.3 Lijnlaser

Een lijnlaser is een laser met een lens waarmee in plaatst van een punt een lijn geprojecteerd kan worden. Lasers hebben een bepaalde intensiteit en een bepaalde kleur. De intensiteit van een laser is afhankelijk van het vermogen (milliwatt) en de concentratie van het licht. Hoe hoger de intensiteit van de laser, des te beter de laser is waar te nemen. Het is echter niet te verantwoorden om met meer dan 5 mw te werken wanneer er omstanders aanwezig kunnen zijn. Een laser met een vermogen van 5 mw valt in veiligheidsklasse 3. Deze staat omschreven als "Ongevaarlijk, maar kan potentieel gevaarlijk zijn wanneer direct in de laserbundel wordt gekeken". De kleur is hierbij ook van belang. Een klasse 3 laser met infrarood licht is namelijk wel gevaarlijk. Omstanders kunnen het licht namelijk niet zien en weten dan ook niet of zij in de laserbundel kijken. Een klasse 3 laser met zichtbaar licht is niet zo problematisch omdat mensen automatisch knipperen wanneer zij in de laser kijken. Hierdoor kan schade aan de retina worden vermeden. De verschillende kleuren van zichtbaar licht hebben voornamelijk invloed op de waarneembaarheid van het licht. Het menselijk oog is bijvoorbeeld het beste in staat om licht waar te nemen met een golflengte van 495 nm tot 570 nm (groen). Een camera is ook het beste in staat om licht waar te nemen met een golflengte van 495 nm tot 570 nm omdat het het beste te onderscheiden is van zonlicht.

## 5.6 Software

Om optische triangulatie te kunnen realiseren moet de computer in staat zijn om het licht van de laser in het beeldmateriaal te kunnen herkennen. De computer is hier toe in staat door middel van 'computer-vision'. Computer-vision is een verzamelnaam voor functionaliteit waarmee beeldmateriaal kan worden bewerkt. Hoewel er verschillende computer-vision bibliotheken beschikbaar zijn, lag de keuze al snel op de bibliotheek OpenCV.

OpenCV (Open Source Computer Vision) is een computer-vision bibliotheek ontwikkeld voor real-time systemen. OpenCV heeft de voorkeur ten opzichte van andere bibliotheken vanwege verschillende redenen. OpenCV is speciaal ontwikkeld om te kunnen functioneren op real-time systemen. OpenCV is tevens een van de meest populaire bibliotheken. Hierdoor is er veel documentatie en ondersteuning beschikbaar. OpenCV is bekend bij zowel de ontwikkelaar van het systeem als de werknemers van Berkelaar Meet- en Regeltechniek. Daarnaast is OpenCV cross-platform, open-source en ondersteunt het onder andere de programmeertalen C, C++ en Java.

Er zit echter ook een nadeel aan OpenCV. OpenCV is speciaal ontwikkeld voor USB-camera-modules. De Raspicam is echter via een flatcable verbonden met de microcontroller. OpenCV kan zonder interface niet met de Raspicam communiceren. Omdat OpenCV zo populair is zijn hier echter verschillende interfaces voor ontworpen. De interfaces bieden helaas niet veel functionaliteit. De interface zal namelijk met de GPU van de microcontroller moeten communiceren om het beeldmateriaal van de Raspicam te kunnen converteren. De driver van de GPU is echter niet open-source. Hierdoor kan de interface alleen gebruik maken van de standaard functies van de driver. Het resultaat is dat OpenCV alleen in staat is om een camera-module te selecteren, een foto te maken en de camera-module vrij te geven. Deze functionaliteit is echter voldoende om optische triangulatie uit te kunnen voeren. Er zit niet veel verschil tussen de beschikbare interfaces. Alle interfaces moeten namelijk met de GPU communiceren en hebben hierdoor dezelfde limitaties.

De communicatie tussen OpenCV en de Raspicam zal in de toekomst waarschijnlijk verbeteren. Op 28 februari 2014 heeft Broadcom namelijk aangegeven bereid te zijn om de code van de GPU vrij te willen geven. Het is daarom een kwestie van tijd voordat de slechte communicatie zal worden verbeterd.

## 5.7 Problemen

In deze paragraaf worden een aantal bekende problemen van optische triangulatie naar voren gebracht. Hoewel niet alle problemen verholpen kunnen worden, is het identificeren van de problemen de eerste stap naar het vinden van een oplossing. Deze problemen zijn met name complex omdat ze zich in hele specifieke situaties voordoen. Het is hierdoor lastig om er altijd rekening mee te kunnen houden. De problemen zijn gevonden door middel van een literatuurstudie (papers, artikels) en verschillende praktijktesten.

Bij een overvloed aan externe lichtbronnen zoals zon- en lamplicht is het voor de camera-module lastiger om onderscheid te maken tussen het licht van de laser en het licht van externe lichtbronnen. Zonlicht is met name sterk in het rode deel van het kleurenspectrum (671 nm). Hierdoor is rood licht voor optische triangulatie een slechte keuze. Als externe lichtbronnen direct in de lens van de camera-module schijnen zal niets meer werken.

Het licht van de laser kan alleen worden waargenomen als iets het licht reflecteert. Objecten die licht absorberen zijn daarom problematisch. Als de camera-module het licht niet kan waarnemen, zal de optische triangulatie er vanuit gaan dat er geen object is. Niet veel objecten zullen het licht volkomen absorberen. Objecten die licht gedeeltelijk absorberen zullen tot meer problemen leiden. Het is een complexe taak om het licht van de laser te herkennen wanneer de intensiteit ervan in een foto aanzienlijk kan verschillen.

Hoewel reflectie vereist is voor de camera-module om licht te kunnen waarnemen, kan het ook problemen veroorzaken. Reflectie kan op verschillende manieren problemen veroorzaken. Reflecterende oppervlakken kunnen externe lichtbronnen versterken waardoor het lastig is om deze te onderscheiden van het laserlicht. Daarnaast kunnen reflecterende oppervlakken het licht van de laser buigen. In een ‘worst case scenario’ kan een reflecterend oppervlak de camera-module verblinden door het licht van de laser direct in de lens te reflecteren.

Als het licht van de laser een hoek of ronde vorm raakt zal maar een deel van het licht naar de camera-module worden gereflecteerd. Hierdoor zal de intensiteit van het licht aanzienlijk lager zijn. Net zoals bij absorptie zal het een complexe taak zijn om het licht van de laser de herkennen wanneer de intensiteit in een foto kan verschillen.

Objecten zoals tafels en stoelen kunnen misleidend zijn voor optische triangulatie. De camera-module zou onder de tafel door kunnen kijken en alleen de stoelpoten kunnen zien. Zodra een systeem gebruik maakt van deze informatie zou het systeem in de problemen kunnen komen. Optische triangulatie is daarom alleen betrouwbaar wanneer er sprake is van objecten die van de grond tot de top dezelfde vorm aanhouden.

Omstanders en bewegende voorwerpen kunnen tot problemen leiden omdat de informatie die de optische triangulatie verschaft bijna direct onbruikbaar is. Het is daarom essentieel dat de optische triangulatie met een dermate hoge frequentie wordt uitgevoerd dat er continu nieuwe informatie binnenkomt. Omstanders kunnen tevens problemen veroorzaken door te dicht bij de camera-module te komen (en de laser te blokkeren) waardoor de camera-module niet meer in staat is om het licht van de laser waar te nemen.

Het licht van de laser heeft niet één intensiteit. Het middelpunt van de lijn is aanzienlijk feller. Om het middelpunt heen is licht waar te nemen met een lagere intensiteit. Dit komt door lichtspreiding. Als een hele hoge nauwkeurigheid vereist is, kan lichtspreiding problemen leveren. De afwijking is echter zo klein dat het voor generieke systemen (zoals navigatie systemen) niet gauw problemen zal veroorzaken.

# 6. Ontwerpfase

*Van 17 maart 2014 t/m 21 maart 2014 (1 week)*

In de eerste ontwerpfase wordt de basis van het lokaliseringssysteem ontworpen. Deze basis moet in staat zijn om beeldmateriaal te verzamelen, beeldmateriaal te bewerken en afstand te berekenen. De software-module is geprogrammeerd in de programmeertaal C++ en maakt met name gebruik van de software-bibliotheek OpenCV. De hardware-module maakt gebruik van een microcontroller, een camera-module en een lijnlaser.

Gedurende deze ontwerpfase zal er aandacht worden besteed aan de volgende onderwerpen:

*Aan welke eisen moet het systeem voldoen? (**6.1**)*

*Hoe kan de software-module het beste worden gerealiseerd? (**6.2**)*

*Hoe kan de hardware-module het beste worden gerealiseerd? (**6.3**)*

# 6.1 Eisen

De eisen die aan het systeem worden gesteld moeten garanderen dat de kwaliteit van de informatie goed genoeg is om andere systemen er mee te laten navigeren. De eisen zijn het resultaat van het onderzoek uit de onderzoeksfase en gesprekken met de werknemers van Berkelaar Meet- en Regeltechniek. Er wordt onderscheid gemaakt tussen functionele eisen en niet-functionele eisen. Functionele eisen geven de kerntaken van het systeem weer en niet-functionele eisen geven de kwaliteit en eigenschappen van het systeem weer.

**Functionele eisen:**

*Het systeem moet door middel van optische triangulatie nabije objecten kunnen lokaliseren.*

**Niet-functionele eisen:**

*Het systeem moet de locatie van objecten kunnen weergegeven in de vorm van een x- en y-coördinaat.*
*Het x-coördinaat is de afstand vanaf het midden van de horizontale beeldhoek tot een object. Het y-coördinaat is de afstand vanaf een lijn die ter hoogte van de camera-module loodrecht op het midden van de horizontale beeldhoek valt tot het object.*

*Het systeem moet beschikken over een gezichtsveld dat minstens net zo breed is als het diep is (1:1).*
*De breedte en diepte van het gezichtsveld moeten overeenkomen. Op een meter afstand moet de camera-module een meter breed beeld hebben. Op drie meter afstand moet de camera-module drie meter breed beeld hebben.*

*Het systeem moet in staat zijn om objecten tot op minimaal 3 meter afstand te kunnen lokaliseren.*
*Deze afstand is afhankelijk van de kwaliteit van de camera-module (om het licht te kunnen waarnemen) en de hoek waarin de camera-module is gepositioneerd (om een gezichtsveld met drie meter diepte te kunnen creëren).*

*Het systeem moet afstanden kunnen bepalen met een afwijking die niet groter is dan 2%.*
*De afwijking is onder andere afhankelijk van de nauwkeurigheid waarmee de maten, afstanden en posities van de hardware componenten van de opstelling bekend zijn in de softwarematige implementatie.*

*Het systeem moet kunnen functioneren met een frequentie van minimaal 5 Hz.*
*Omdat er sprake is van mobiele systemen (en omstanders) moet het systeem in staat zijn om vijf keer per seconde optische triangulatie uit te voeren. Hoe lager de resolutie van de camera-module, des te hoger de frequentie.*

*Het systeem moet lasers hanteren die omstanders (waaronder kinderen) niet kunnen schaden.*
*Lasers met meer dan 5 mw kunnen niet gehanteerd worden zonder het risico te lopen omstanders te schaden. Een zwakkere laser is minder problematisch omdat mensen automatisch knipperen wanneer zij in de laser kijken.*

*Het systeem moet ontwikkeld worden in de programmeertaal C++.*
*Berkelaar Meet- en Regeltechniek geeft de voorkeur aan de objectgeoriënteerde programmeertaal C++. Aangezien meer van hun systemen deze programmeertaal gebruiken is de optische triangulatie makkelijk te integreren.*

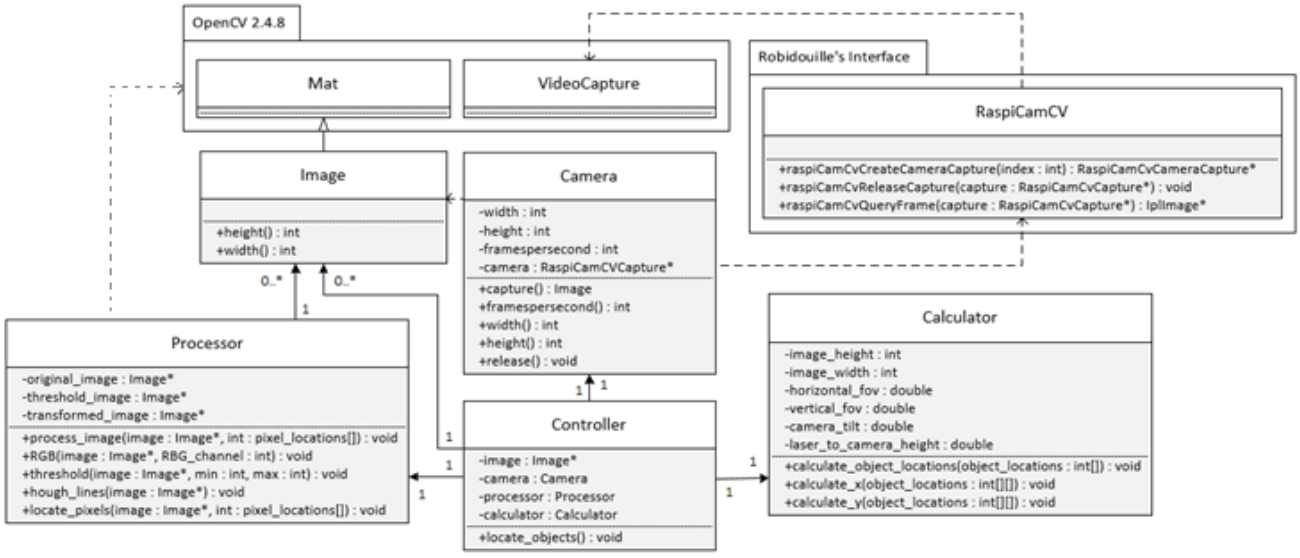
*Het systeem moet minder kosten dan commercieel beschikbare optische-triangulatiesystemen (< 300 euro).*
*Om de ontwikkeling van het systeem lucratief te houden moeten de kosten van de microcontroller, camera-module en lijnlaser te samen niet meer bedragen dan de systemen die anno 2014 op de markt beschikbaar zijn.*

# 6.2 Software-module

De software-module van het systeem dient geprogrammeerd te worden in de objectgeoriënteerde programmeertaal C++. De structuur van de software-module kan daarom worden weergegeven door middel van modelleertaal UML. De structuur van de software-module is weergegeven in een klassendiagram (figuur 6.1). De sequentiële werking van de software-module is weergegeven in een sequentiediagram (figuur 6.2).

### 6.2.1 Klassendiagram

In figuur 6.1 is het klassendiagram van de software-module weergegeven. Het klassendiagram bestaat uit vijf interne klassen, een externe interface (*Robidouillle's interface*) en een computer-vision bibliotheek (*OpenCV 2.4.8*). De structuur van het systeem is gebaseerd op de stapsgewijze implementatie van optische triangulatie zoals deze in de onderzoeksfase is weergegeven. De stappen zijn: het verzamelen van beeldmateriaal (**Camera**), het bewerken van beeldmateriaal (**Processor**) en het berekenen van afstand (**Calculator**). De stappen hebben geen onderlinge interactie. Alle interactie loopt via een interface (**Controller**). Omdat de verschillende stappen geen onderlinge afhankelijkheden hebben kunnen de stappen gemakkelijker worden aangepast. Het klassendiagram bevat tevens een klasse **Image**. Deze klasse is een representatie van de klasse **Mat** uit de computer-vision bibliotheek *OpenCV 2.4.8*. Deze klasse is weergegeven om de structuur van het ontwerp te verhelderen. De klasse **Image** bevat geen additionele functionaliteit. Naast de klasse Mat is met name de klasse **VideoCapture** uit de computer-vision bibliotheek interessant. Het is de klasse die verantwoordelijk is voor het maken van foto's. Zonder een interface is OpenCV echter niet in staat om te communiceren met de (flatcable) Raspicam camera-module. Hiervoor wordt gebruik gemaakt van *Robidouille's Interface*. Deze interface bevat een aantal C functies die de communicatie tussen de Raspicam en OpenCV kunnen afhandelen. Het resultaat is een software-module die in staat is om stapsgewijs beeldmateriaal te verzamelen, beeldmateriaal te bewerken en afstand te berekenen.

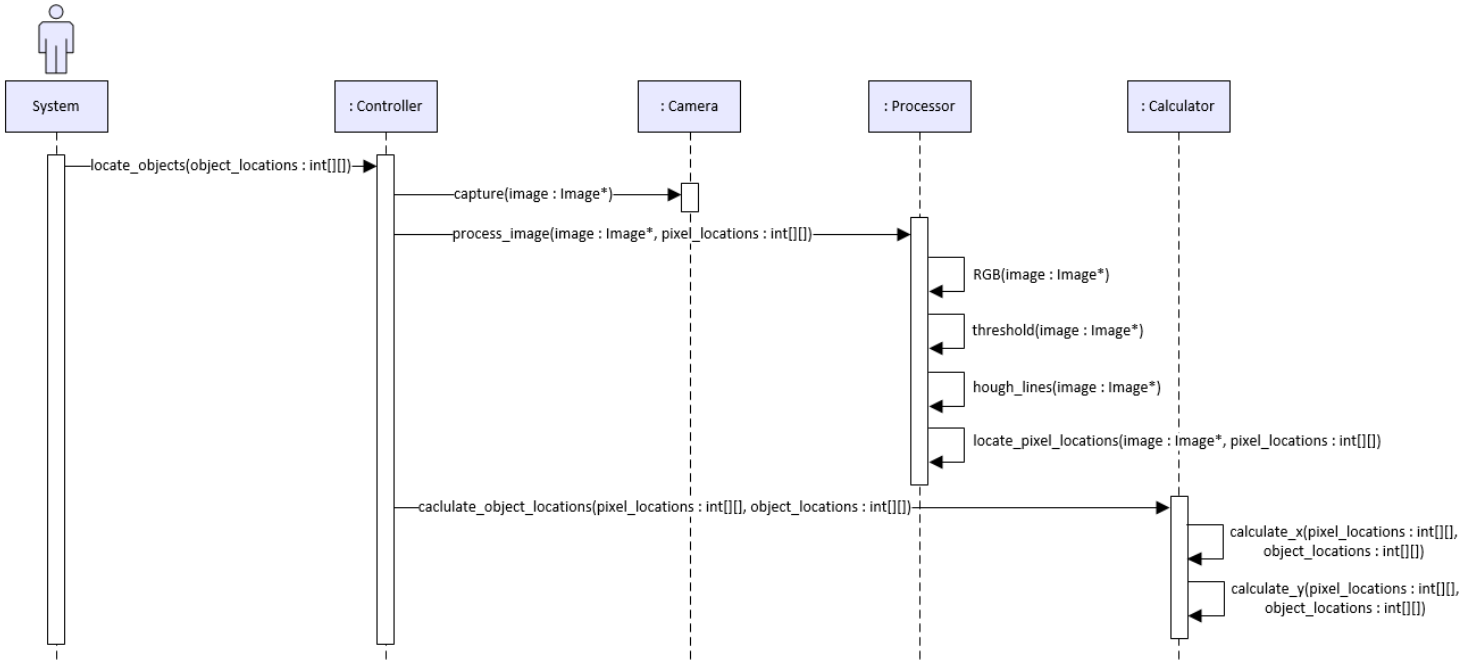



***Figuur 6.1*** *Klassendiagram van het lokaliseringssysteem (eerste iteratie)*

### 6.2.2 Sequentiediagram

In figuur 6.2 is het sequentiediagram van de software-module weergegeven. Het geeft de interactie tussen de klassen **Controller**, **Camera**, **Processor** en **Calculator** weer. Alle interactie loopt vanuit de klasse **Controller**. In het klassendiagram heeft de functie *locate_objects*() van de klasse **Controller** geen parameters of return-waardes omdat de interactie tussen de software-module en andere systemen buiten deze iteratie valt. In het sequentiediagram is echter wel ter verheldering een extern systeem weergegeven die door middel van een parameter (pointer) de locaties van nabije objecten kan opvragen. De **Controller** geeft aan de klasse **Camera** een **Image** pointer mee om een foto te maken en die in de pointer te stoppen. De interactie tussen de **Camera** en *Robidouille's Interface* (**RaspiCamCV**) vindt in de functie *capture*() plaats. De **Controller** geeft de **Image** pointer die de foto bevat vervolgens in de functie *process_image*() mee aan de klasse **Processor**. Nagenoeg alle communicatie met de computer-vision bibliotheek *OpenCV* vind plaats deze klasse. De **Processor** zal meerdere functies (3) van zichzelf aanroepen waarin de foto uit de **Image** pointer wordt omgezet van een oorspronkelijke foto tot een binaire afbeelding. In de binaire afbeelding is het licht van de laser weergegeven met een witte kleur en al het andere is zwart. De **Processor** zal vervolgens met de functie *locate_pixel_locations*() de locaties van de pixels waarin het licht is waargenomen registreren. Deze locaties zullen vervolgens door de **Controller** worden doorgegeven aan de klasse **Calculator**. De klasse **Calculator** is in staat om door middel van de *pixel_locaties* en de beeldhoeken van de camera-module de locaties (x- en y-coördinaten) van de objecten te berekenen. Het resultaat is een **Controller** die beschikt over een tweedimensionale array met de coördinaten van de locaties van de objecten. In de derde iteratie wordt er in gegaan wat de **Controller** vervolgens met deze locaties kan doen.

*Een interessante toevoeging voor de derde iteratie zou de klasse **Location** zijn. Deze klasse zou onder andere de pixel_locations en object_locations en image_resolution kunnen bevatten. Het zou ook informatie kunnen verstrekken over richtingen waarin zich wel of niet objecten bevinden. Daarnaast is het versturen van tweedimensionale arrays is niet overzichtelijk. Gedurende de eerste iteratie zal het echter voldoen.*




***Figuur 6.2*** *Sequentiediagram van use-case "locate nearby objects" (eerste iteratie)*

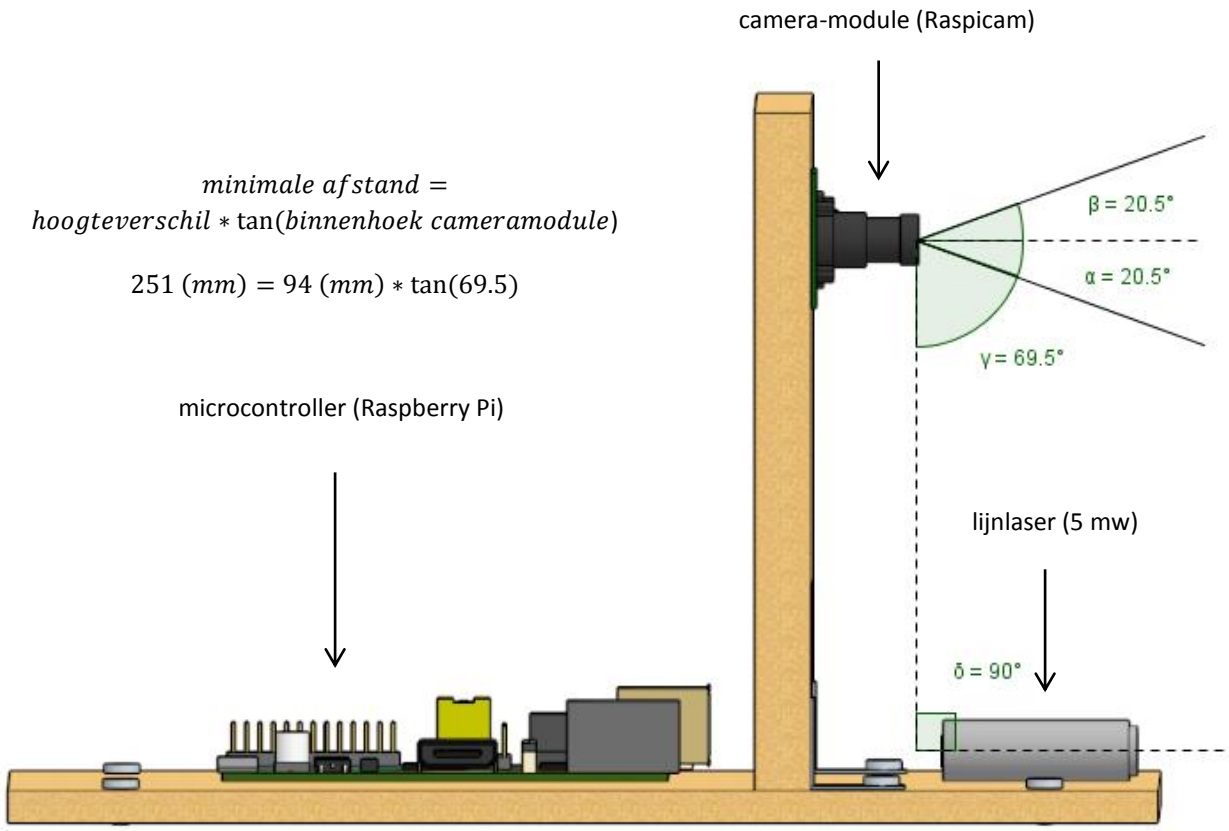
# 6.3 Hardware-module

De hardware-module van het systeem bestaat uit een microcontroller (Raspberry Pi), camera-module (Raspicam) en lijnlaser (5 mw). De specificaties van deze componenten zijn weergegeven in de onderzoeksfase (5.5). De wijze waarop deze componenten worden opgesteld hebben invloed op onder andere het maximale en minimale bereik van de optische triangulatie en de eenvoud waarin verschillende afstanden van elkaar kunnen worden onderscheiden. In figuur 6.3 is een ontwerp van de hardware-module weergegeven (niet op schaal). Het ontwerp is gebaseerd op de beschikbare middelen (een houten plank en de hardwarecomponenten). Aan de hand van het ontwerp kunnen de minimale en maximale afstand waarmee optische triangulatie kan worden uitgevoerd worden berekend. Deze afstanden zijn afhankelijk van de verticale beeldhoek van de camera-module, de hoek waarin de camera-module is gepositioneerd en de verticale afstand tussen de camera-module en de laser.

De minimale afstand waarmee optische triangulatie kan worden uitgevoerd is te berekenen met de formule:

$$minimale\ afstand = hoogteverschil * \tan(binnenhoek\ cameramodule) = 94 * \tan(69.5) = \ 251\ (mm)$$

Een maximale afstand is niet te berekenen. Deze is er namelijk niet. De maximale afstand is de afstand tot de bovenkant van de beeldhoek het licht van de laser kruist. In de opstelling in figuur 6.3 zal dit niet gebeuren. Hiervoor zal de camera meer dan 20.5 graden naar beneden moeten draaien. De maximale afstand zal in deze situatie daarom niet bepaald worden door het zichtveld maar door de capaciteiten van de camera-module en laser. Dit maakt de hardware-module uitermate geschikt om de prestaties van de componenten te testen.

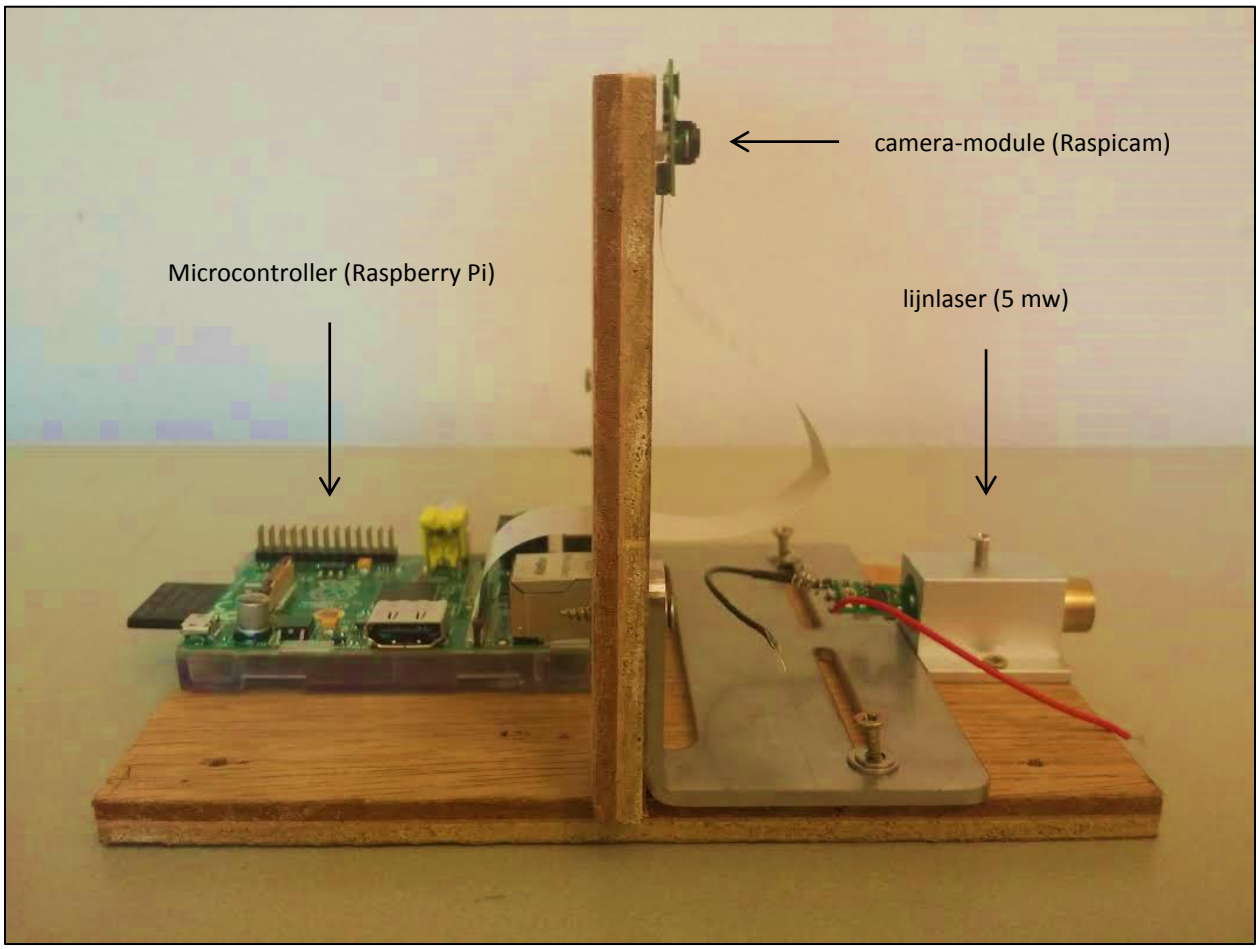



***Figuur 6.3*** *Zij-aanzicht van de hardware-module met aangegeven componenten.*

# 7. Implementatiefase

*van 24 maart 2014 t/m 28 maart 2014 (1 week)*

In de eerste implementatiefase worden twee versies van het ontwerp geïmplementeerd. De eerste versie wordt geïmplementeerd op een notebook (MSI-GP60). Deze versie maakt gebruik van de ingebouwde camera-module van de notebook en een 5 mw lijnlaser. De tweede versie wordt geïmplementeerd op de Raspberry Pi en maakt gebruik van de Raspicam camera-module en dezelfde 5 mw lijnlaser. Het verschil tussen de twee versies is de communicatie tussen de camera-module en OpenCV. Het verschil is waar te nemen in de Camera klasse. Beide versies zijn op de notebook ontwikkeld. De processor van de microcontroller is namelijk niet sterk genoeg om snel te kunnen compileren. Cross-compilen vanaf de notebook naar de microcontroller is helaas ook niet mogelijk. Om te kunnen cross-compilen moeten alle afhankelijkheden (code en bibliotheken) beschikbaar zijn op beide systemen, maar de notebook heeft geen toegang tot de drivers die de communicatie tussen de Raspicam en OpenCV ondersteunen. Beide versies zijn echter met succes op de notebook ontwikkeld en op de desbetreffende systemen geïmplementeerd. Gedurende deze implementatiefase is tevens de hardware-module van het systeem gerealiseerd. Deze hardware-module is weergegeven in figuur 6.4.




***Figuur 7.1*** *Foto van het zij-aanzicht van de hardware-module*

```
/*************************************************************************|
| The function determine_course determines the coordinates of nearby objects |
\*************************************************************************/
double Controller::determine_course()
{
    // An image will be captured by the Camera.
    camera.capture(&image);
    imwrite("original.jpg", image);                         /* Debugging */
    imread("stored_original.jpg").copyTo(image);            /* Debugging */

    // The locations of pixels that contain the light of laser will be determined  by the Processor.
    int pixel_locations[image.width()];
    processor.process_image(&image, pixel_locations);

    // The coordinates of the locations that the pixels represent will be calculated by the Calculator.
    double object_locations[image.width()][2];
    calculator.calculate_coordinates(pixel_locations, object_locations, image.width(), image.height());

    // The coordinates of the locations of objects will be examined to determine a viable course.
    return calculator.calculate_course(object_locations, image.width(), image.height());
}
```

***Figuur 7.2*** *Voorbeeld-code: de implementatie van de functie die de gehele optische triangulatie routine controleerd.*

```

// Calculates the distance corresponding with the given pixel coordinates in the image
void Calculator::calculate_coordinates(int* pixel_locations, double object_locations[][2], int columns, int rows)
{
    for(int i = 0; i < columns; i++)
    {
        if(pixel_locations[i] != -1)
        {
            object_locations[i][1] = calculate_y(pixel_locations[i], rows);
            object_locations[i][0] = calculate_x(i, columns, object_locations[i][1]);
        }
        else
        {
            object_locations[i][1] = -1; object_locations[i][0] = -1; // returns -1 if none coordinates are detected
        }

        /* cout << i << " " << object_locations[i][1] << " " << object_locations[i][0] << endl; */
    }
}

// Returns the y coordinate of the pixel location
double Calculator::calculate_y(double row_number, int rows)
{
    return (LASER_TO_CAMERA_HEIGHT * tan((((row_number/rows) * CAMERA_FULL_VERTICAL_FIELD_OF_VIEW) +
            (CAMERA_POSITIONING_ANGLE - (CAMERA_FULL_VERTICAL_FIELD_OF_VIEW / 2))) * ( PI / 180)));
}

// Returns the x coordinate of the pixel location
double Calculator::calculate_x(double column_number, int columns, double y)
{
    double column_percentage = ( ( column_number / columns ) * 100 );

    if(column_percentage < 50) // Left of the middle results in a negative value
    {
       return (0 - (y *(tan(((CAMERA_FULL_HORIZONTAL_FIELD_OF_VIEW/2) - (CAMERA_FULL_HORIZONTAL_FIELD_OF_VIEW * (column_number/columns))) * (PI / 180)))));
    }
    else if(column_percentage > 50) // Right of the middle results in a positive value
    {
        return (y *(tan(((CAMERA_FULL_HORIZONTAL_FIELD_OF_VIEW) * ((column_number/columns)-0.5)) * (PI / 180))));
    }
    else
    {
        return 0;
    }
```

***Figuur 7.3*** *Voorbeeld-code: de implementatie van de functies die de coördinaten van de objecten berekenen.*

# 8. Testfase

*van 31 maart 2013 t/m 4 april 2014 (1 week)*

In de eerste testfase staat het testen van de optische triangulatie centraal. De tests zijn gebaseerd op onder andere de functionele eisen (P. 6.1), niet-functionele eisen (P. 6.1) en bekende problemen (P. 5.7). Iedere test bestaat uit vijf aspecten. Het eerste aspect is wat er getest wordt. Het tweede aspect is waarom er getest wordt. Het derde aspect is hoe er getest wordt. Het vierde aspect is wat de verwachte testresultaten zijn. Het vijfde aspect is het daadwerkelijke testresultaat. Door op deze wijze te testen kan er goed in kaart worden gebracht wat voor consequenties de testresultaten hebben op de kwaliteit van het systeem.

In de onderstaande lijsten zijn de eisen en problemen weergegeven waarop de tests gebaseerd zijn.

Functionele en niet-functionele eisen:

1. *Het systeem moet de locatie van objecten kunnen weergegeven in de vorm van een x- en y-coördinaat.*
2. *Het systeem moet afstanden kunnen bepalen met een afwijking die niet groter is dan 2%.*
3. *Het systeem moet beschikken over een gezichtsveld dat minstens net zo breed is als het diep is (1:1).*
4. *Het systeem moet in staat zijn om objecten tot op minimaal 3 meter afstand te kunnen lokaliseren.*
5. *Het systeem moet kunnen functioneren met een frequentie van minimaal 5 Hz.*
6. *Het systeem moet lasers hanteren die omstanders (waaronder kinderen) niet kunnen schaden.*
7. *Het systeem moet ontwikkeld worden in de programmeertaal C++.*
8. *Het systeem moet minder kosten dan commercieel beschikbare optische-triangulatiesystemen (< 300 euro).*

Bekende problemen:

9. *Over- en onderbelichte scenario's*
10. *Absorberende objecten*
11. *Reflecterende objecten*
12. *Hoeken en ronde objecten*
13. *Objecten met misleidende vormen*
14. *Bewegende objecten*
15. *Onscherpe projecties*

Alle eisen en bekende problemen resulteren in een groot aantal testcases waarvan iedere testcase bestaat uit meerdere foto's en tabellen. Om deze reden wordt er met name in gegaan op de testresultaten die niet overeenkomen met de verwachte testresultaten of testresultaten die grote implicaties hebben voor de ontwikkeling van het systeem. Van de overige testcases worden de testresultaten kort naar voren gebracht.

<u>1. Het systeem moet de locatie van objecten kunnen weergegeven in de vorm van een x- en y-coördinaat.</u>

Door middel van de pixel locaties, horizontale beeldhoek en verticale beeldhoek kunnen beide coördinaten worden bepaald. Door de wijze waarop door middel van triangulatie afstanden kunnen worden berekend zijn de afstanden per definitie coördinaten en niet directe afstanden van objecten tot de camera-module of laser.

2. Het systeem moet afstanden kunnen bepalen met een afwijking die niet groter is dan 2%.

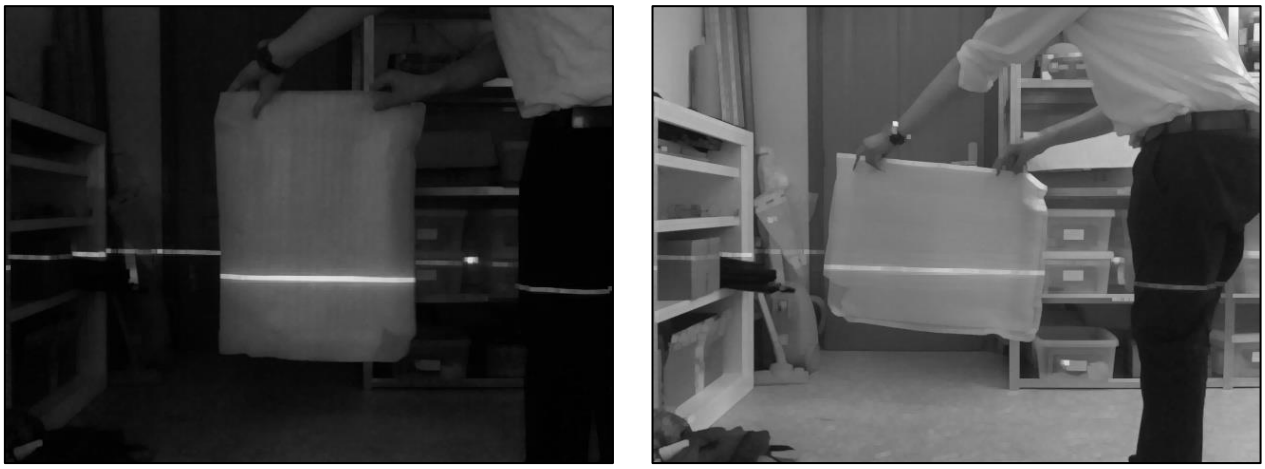
Mits de beeldhoek van de camera-module en de afstand tussen de laser en de camera-module nauwkeurig kunnen worden bepaald, kan de afstand met grote nauwkeurigheid worden bepaald. Deze nauwkeurigheid neemt echter af naarmate de afstand tot de objecten toeneemt. In testcase 4 wordt hier verder op ingegaan.

3. Het systeem moet beschikken over een gezichtsveld dat minstens net zo breed is als het diep is (1:1).

De specificaties van de Raspicam geven een beeldhoek weer met een (breedte : hoogte : diepte verhouding) van (1 : 0.677 : 1). Door foto's te maken en de maten op te meten zijn deze verhoudingen geverifieerd.

4. Het systeem moet in staat zijn om objecten tot op minimaal 3 meter afstand te kunnen lokaliseren.

Het kunnen lokaliseren van afstanden is afhankelijk van twee factoren: het kunnen waarnemen van het licht van de laser en het kunnen berekenen van de afstand. De eerste factor is problematisch omdat deze factor met name afhankelijk is van de kwaliteit van de camera-module en de hoeveelheid zon- en lamplicht in de foto.



***Figuur 8.1*** *Links: 1 meter afstand in een donkere omgeving. Rechts: 2 meter afstand in een lichte omgeving.*

Bij de tweede factor ontstaat een probleem dat bij het ontwerp over het hoofd is gezien. De camera-module is loodrecht boven de laser geplaatst. Het middelpunt van de beeldboek van de camera-module is daarom een horizontale lijn. Het licht van de laser is ook een horizontale lijn. Omdat beide lijnen horizontaal lopen zullen ze elkaar nooit kruisen. Het resultaat is dat alle afstanden worden weergegeven in onderste helft van de foto. Het licht van de laser zal ook nooit meer buiten de beeldhoek van de camera-module vallen. Hierdoor worden letterlijk alle afstanden weergegeven in de onderste helft van de foto. Hoe verder weg de afstand, des te dichterbij het middelpunt van de foto. Naarmate de afstand toeneemt wordt het steeds moeilijker om de verplaatsing waar te nemen. Hierdoor kan deze opstelling alleen maar worden ingezet om korte afstanden waar te nemen. Als de afstanden groter worden dan 2 meter wordt het moeilijk om het licht waar te nemen en het wordt moeilijk om afstanden accuraat te af te leiden van de foto. De eerste factor kan worden verbeterd door een betere camera-module te gebruiken. De tweede factor kan worden verbeterd door de camera-module naar beneden te kantellen.

5. Het systeem moet kunnen functioneren met een frequentie van minimaal 5 Hz.

Het systeem is een 'real-time' systeem en moet dus kunnen voldoen aan tijdsgebonden eisen. De door Berkelaar Meet- en Regeltechniek gestelde eis is dat het systeem moet kunnen functioneren met een frequentie van minimaal 5 Hz. Deze frequentie is essentieel vanwege twee redenen. De eerste reden is dat het systeem uiteindelijk op een mobiel platform zal worden geïmplementeerd. Het feit dat het platform zich continu zal verplaatsen betekent dat de informatie die de optische triangulatie verschaft bijna direct verouderd is. Er moeten daarom continu nieuwe informatie worden geleverd. De tweede reden is dat het systeem ingezet moet kunnen worden als een omstanders aanwezig zijn. Deze omstanders zullen net zoals het bewegende platform er voor zorgen dat de verschafte informatie van de optische triangulatie snel verouderd is.

In figuur 8.2 zijn de testresultaten van de Raspberry Pi weergegeven. De testresultaten bestaan uit de minimale, maximale en gemiddelde frequenties van verschillende onderdelen van de optische triangulatie. De gehele optische triangulatie kan worden uitgevoerd met een frequentie van gemiddeld 5.95 Hz. Het systeem voldoet hiermee aan de door Berkelaar Meet- en Regeltechniek gestelde eis van minimaal 5 Hz.

| Features | Minimum | Maximum | Gemiddelde |
|---|---|---|---|
| Beeldmateriaal verzamelen | 11.73 Hz | 14.25 Hz | 13.61 Hz |
| Beeldmateriaal verzamelen + bewerken | 5.69 Hz | 6.37 Hz | 6.13 Hz |
| Beeld materiaal verzamelen + bewerken + berekenen | 5.78 Hz | 6.24 Hz | 5.95 Hz |

***Figuur 8.2*** *De frequenties waarmee de Raspberry Pi verschillende taken kan uitvoeren.*

6. Het systeem moet lasers hanteren die omstanders (waaronder kinderen) niet kunnen schaden.

De laser valt in de laser-veiligheidsklasse 3R. Deze klasse is gespecificeerd als: "Alleen gevaarlijk wanneer direct in de laserbundel wordt gekeken". Een klasse 3 laser met zichtbaar licht is niet zo gevaarlijk omdat mensen automatisch knipperen wanneer zij in de laser kijken. Hierdoor kan schade aan de ogen worden vermeden.

7. Het systeem moet ontwikkeld worden in de programmeertaal C++.

Het systeem is ontwikkeld in de programmeertaal C++. Alleen Robidouille's Interface bevat de programmeertaal C.

8. Het systeem moet minder kosten dan commercieel beschikbare optische-triangulatiesystemen (< 300 euro).

Het optische triangulatie systeem bestaat uit drie componenten die in totaal €103,47 kosten.

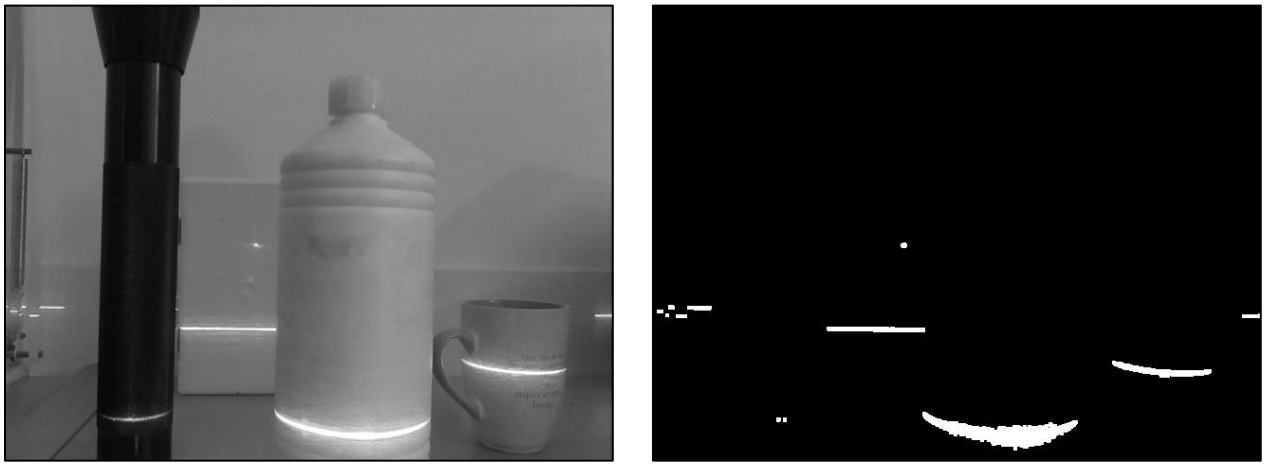
| | | |
|---|---|---|
| Raspberry Pi model B 512 MB | (€39, 99) | www.conrad.nl |
| Raspicam camera-module | (€44, 99) | www.conrad.nl |
| 5 mW 532 nm lijnlaser-module | (€18.49) | www.odicforce.com |

9. Over- en onderbelichte scenario's

Over- en onderbelichte scenario's kunnen op verschillende manieren problemen veroorzaken. Het grootste probleem is dat de camera-module in overbelichte scenario's niet goed in staat is om het licht van de laser te onderscheiden van andere lichtbronnen. Er kunnen echter ook problemen ontstaan die niet direct met het licht zelf te maken hebben. In hele over- en onderbelichte scenario's kan de camera-module foto's compenseren. Deze compensatie kan vervolgens zelf tot problemen leiden (P 5.2.2). In de tests kwam echter naar voren dat het met name overbelichte scenario's zijn waarin objecten worden waargenomen die zon- en lamplicht reflecteren.

10. Absorberende objecten

Donkere objecten reflecteren minder goed licht dan lichte objecten. Hierdoor kan het moeilijker of zelfs onmogelijk zijn om deze objecten waar te nemen op de foto. In figuur 8.3 is een duidelijk verschil te zien tussen het licht van de laser op een zwarte zaklamp en het licht van de laser op een witte fles. Links is het groene RGB-kanaal van de oorspronkelijke foto weergegeven. Rechts is de op intensiteit gebaseerde threshold weergegeven.



***Figuur 8.3*** *Links: het groene RGB-kanaal van een foto. Rechts: een threshold van de links weergegeven foto.*

11. Reflecterende objecten

Reflecterende objecten kunnen problematisch zijn omdat ze zon- en lamplicht naar de camera-module kunnen reflecteren. Licht dat direct in de camera-module wordt gereflecteerd heeft een hele hoge intensiteit en zal daardoor moeilijk te onderscheiden zijn van het licht van de laser. In figuur 8.4 zijn twee scenario's weergegeven. Links een scenario met lamplicht en rechts een scenario zonder lamplicht. Niet alleen wordt het lamplicht met hoge intensiteit gereflecteerd, het licht van de laser is ook nog eens minder goed waar te nemen.

***Figuur 8.4*** *Links: een foto zonder lamplicht (geen reflectie) . Rechts: een foto met lamplicht (duidelijke reflectie).*

12. Hoeken en ronde vormen

Het probleem met hoeken en ronde objecten is niet dat ze het licht van de laser niet reflecteren, maar dat ze het licht van de laser niet naar de camera-module reflecteren. De reflectie wordt niet direct weerkaatst. Hierdoor is d camera-module minder goed in staat om de intensiteit van het licht waar te nemen. Het verschil in intensiteit is echter verwaarloosbaar ten opzichte van de problemen die ontstaan bij absorberende en reflecterende objecten.

13. Objecten met misleidende vormen

Objecten zoals stoelen en tafels kunnen problemen opleveren omdat de laser alleen in staat is om objecten waar te nemen op een specifieke hoogte. Objecten die zich boven of onder de laser bevinden kunnen niet worden waargenomen. Zonder de camera-module te bewegen of meerdere lasers te gebruiken is dit onvermijdelijk.

14. Bewegende objecten

Het systeem moet ook in staat zijn om optische triangulatie uit te voeren wanneer er omstanders aanwezig zijn. Het systeem zal omstanders gewoon als objecten zien. Het enige probleem met omstanders is dat ze (snel) kunnen bewegen. Dit kan resulteren in twee soorten problemen. Het eerste probleem is dat de informatie over de locatie van de bewegende objecten per direct verouderd is. Acties ondernemen aan de hand van deze informatie kan vervolgens tot problemen leiden. Het tweede probleem is dat het maken van foto's van bewegende objecten een goede camera-module vereist. De Raspicam heeft al gauw moeite om bewegende objecten scherp waar te nemen. Hoe lager de frequentie waarmee de foto's worden genomen, des te problematischer de foute informatie zal zijn.

15. Onscherpe projecties

Een laser is een lichtbron die geconcentreerd licht kan voort brengen. De mate van concentratie verschilt per laser en is afhankelijk van verschillende factoren (medium, methode, vermogen). In figuur 8.5 (rechts) is echter te zien dat het licht van de laser niet één concentratie heeft. Om de geconcentreerde lijn heen is een 'aura' van minder intens licht te zien. Aura's kunnen ontstaan bij onder andere slechte lasers en vieze lenzen. Aura's zijn beter te zien op objecten die goed kunnen reflecteren. In figuur 8.4 (links) bijvoorbeeld alleen een aura te zien op de metalen fles. Het aura verbreedt de projectie van de laser. Er kan daarom minder nauwkeurig afstand worden bepaald.

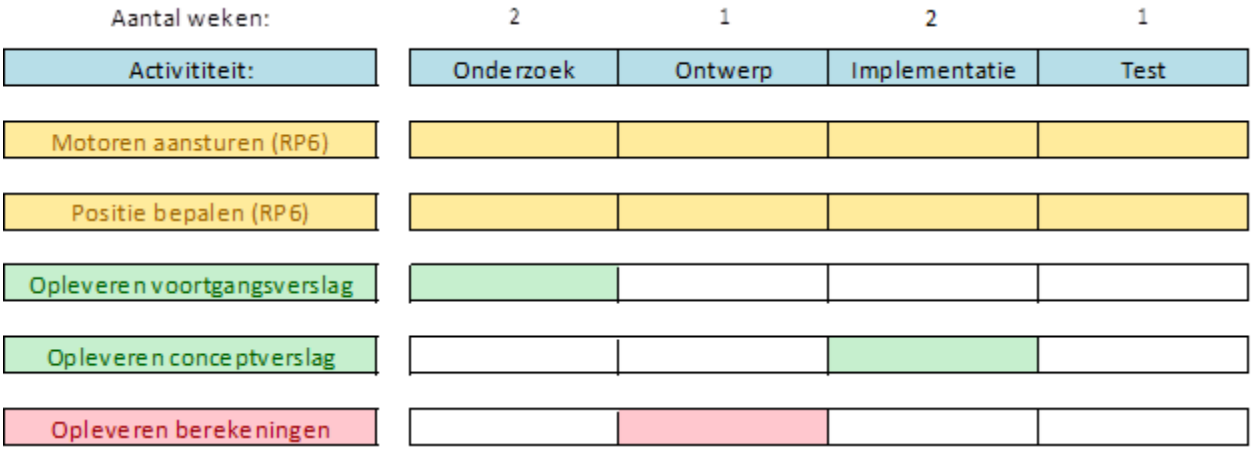

***Figuur 8.5*** *Links: reflecterende objecten geven een aura rondom de laser weer. Rechts: Het aura heeft een doffere kleur.*

# **De tweede iteratie:** Autonome systemen

*7 april 2014 t/m  9 mei 2014 (6 weken)*

In de tweede iteratie staat onderzoek naar systemen die gebruik kunnen maken van optische triangulatie centraal. Een alternatieve invulling van deze iteratie betreft het onderzoek doen naar methodes waarmee optische triangulatie kan worden verbeterd wanneer er sprake is van onder andere te veel licht en absorberende objecten. Er is hier echter niet voor gekozen omdat uit de eerste iteratie is gebleken dat deze problemen met name ontstaan (en verholpen kunnen worden) door de kwaliteit van de hardware en de status van de drivers. Deze iteratie kan daarom beter besteed worden door onderzoek te doen naar aspecten zoals de toepassing van het systeem.

In deze iteratie zal er onderzoek gedaan worden naar de aanvullende technieken die nodig zijn om een systeem door middel van optische triangulatie rond te laten rijden. Het doel van deze iteratie is een systeem dat in staat is om door een kamer te rijden en zijn eigen snelheid, locatie en oriëntatie kan bepalen. Niet alleen zijn de technieken die hier bij komen kijken essentieel voor autonome navigatie, het systeem zal ook als platform dienen waarop de optische triangulatie in de derde iteratie zal worden geïmplementeerd. De iteratie bestaat uit vijf weken en bevat zoals alle andere iteraties een onderzoeks-, ontwerp-, implementatie- en testfase.




***Figuur P.2*** *De planning van de tweede iteratie. Het ontwikkelen van een autonome robot.*

# 9. Onderzoeksfase

*7 april 2014  t/m 18 april 2014 (2 week)*

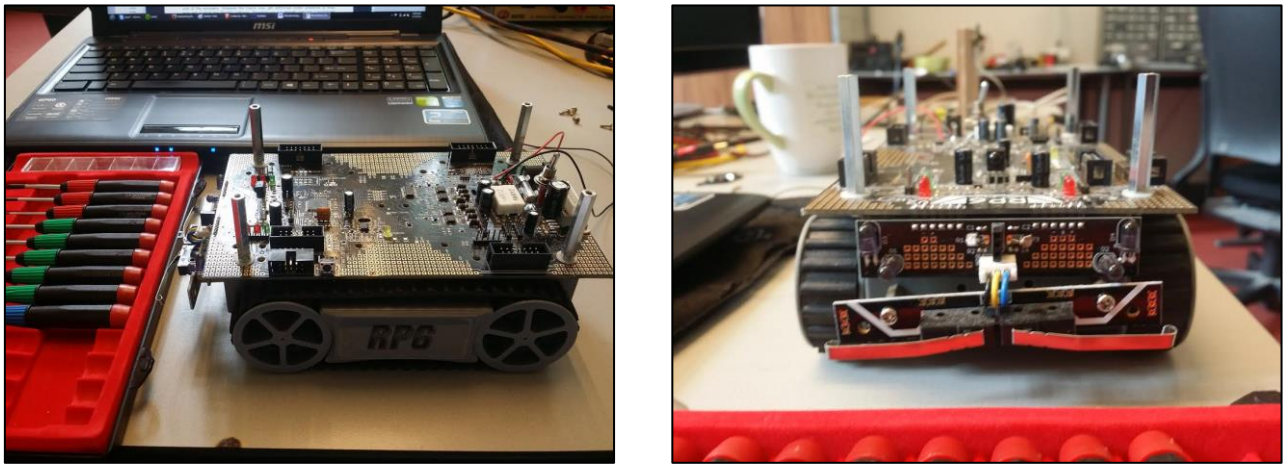
In de tweede onderzoeksfase wordt onderzoek gedaan naar de wijze waarop andere systemen gebruik kunnen maken van de informatie die met optische triangulatie kan worden verkregen. Er wordt met name onderzoek gedaan naar de aanvullende technieken waar deze systemen gebruik van maken. Het specifieke systeem waar onderzoek naar zal worden gedaan is weergegeven in figuur 9.1. Het systeem is een autonome robot (RP6) met rupsbanden. Deze robot zal gedurende de tweede iteratie dienen als platform voor het onderzoek naar en de ontwikkeling van technieken al deze technieken. Het doel van deze onderzoeksfase is onderzoeken hoe de RP6 kan worden ingezet om door een kamer te rijden en zijn eigen snelheid, locatie en oriëntatie te bepalen. Het grootste deel van het onderzoek bestaat uit goniometrie en zal worden benaderd met de wiskundesoftware “GeoGebra”.

Gedurende deze iteratie zal als leidraad gezocht worden naar antwoord op de volgende vragen:

*Welke technieken en sensoren zijn er nodig naast optische triangulatie? (**9.1**)*

*Hoe kunnen deze technieken en sensoren worden ingezet om snelheid, locatie en oriëntatie te bepalen? (**9.2**)*

*Hoe kan de robot worden aangestuurd aan de hand van zijn snelheid, locatie en oriëntatie? (**9.3**)*



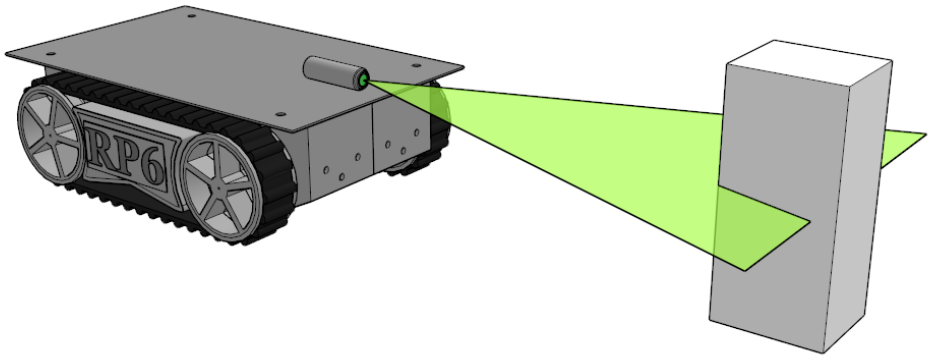

***Figuur 9.1** Foto’s van de autonome robot met rupsbanden (RP6) waarmee de tweede iteratie wordt gewerkt.*

# 9.1 Sensoren

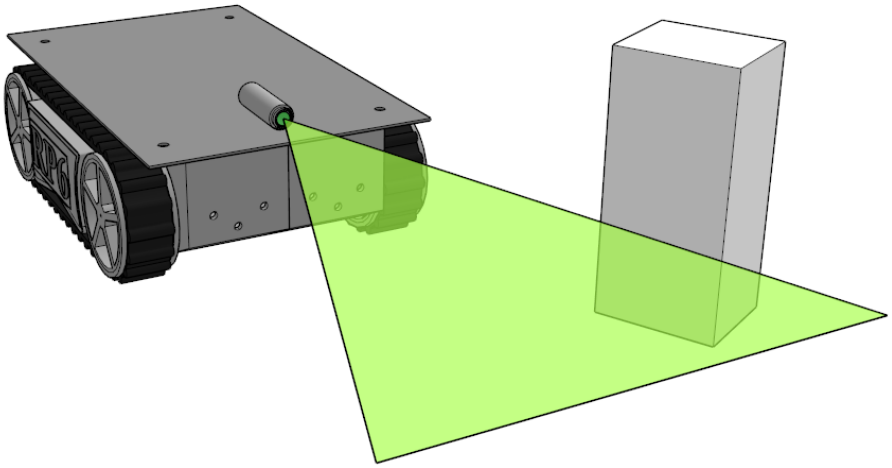
Optische triangulatie kan worden toegepast om de locatie van nabije objecten te bepalen. In principe kan een robot door middel van optische triangulatie navigeren. Een robot kan zijn eigen positie bepalen aan de hand van de afstand tussen de robot en nabije objecten. Helaas is deze methode niet in alle situaties even betrouwbaar. Er zullen andere technieken aan te pas moeten komen om de robot zelfstandig te kunnen laten navigeren.

### 9.1.1 Optische triangulatie

Optische triangulatie is een techniek waarmee de afstand tot nabije objecten kan worden bepaald. Deze afstanden zijn echter relatief en niet absoluut. Dat betekent dat deze afstanden alleen waardevol zijn vanuit de huidige positie van de robot of wanneer de positie van de robot bekend is. Een ander probleem dat zich voor zal doen met optische triangulatie is dat er niet altijd nabije objecten in het beeld van de camera zijn. In figuur 9.2 is een situatie weergegeven waarin een object het pad van de robot kruist. In figuur 9.3 is de situatie weergegeven waarin de robot om het object heen probeert te rijden. De robot zal niet meer in staat zijn het object waar te nemen. Zonder andere sensoren is de robot nu niet in staat om te bepalen waar hij is en waar hij heen moet.




***Figuur 9.2*** *De robot is in staat om informatie over zijn eigen positie af te leiden.*



***Figuur 9.3*** *De robot heeft geen idee meer waar hij is en waar hij heen moet.*

Sensoren zoals optische triangulatie vallen in de categorie ‘externe sensoren’. Externe sensoren maken gebruik van externe middelen om te kunnen functioneren. Voorbeelden van externe sensoren zijn GPS, RFID en IR-sensoren. Deze sensoren lopen echter tegen dezelfde problemen aan als optische triangulatie. De sensoren zijn afhankelijk van externe middelen zoals satellieten, RFID-chips en nabije objecten om te kunnen functioneren. Het alternatief zijn interne sensoren. Deze sensoren maken gebruik van de robot zelf om informatie te kunnen verkrijgen. Een van de meest voorkomende interne sensortechniek bij autonome navigatie is odometrie.

### 9.1.1 Odometrie

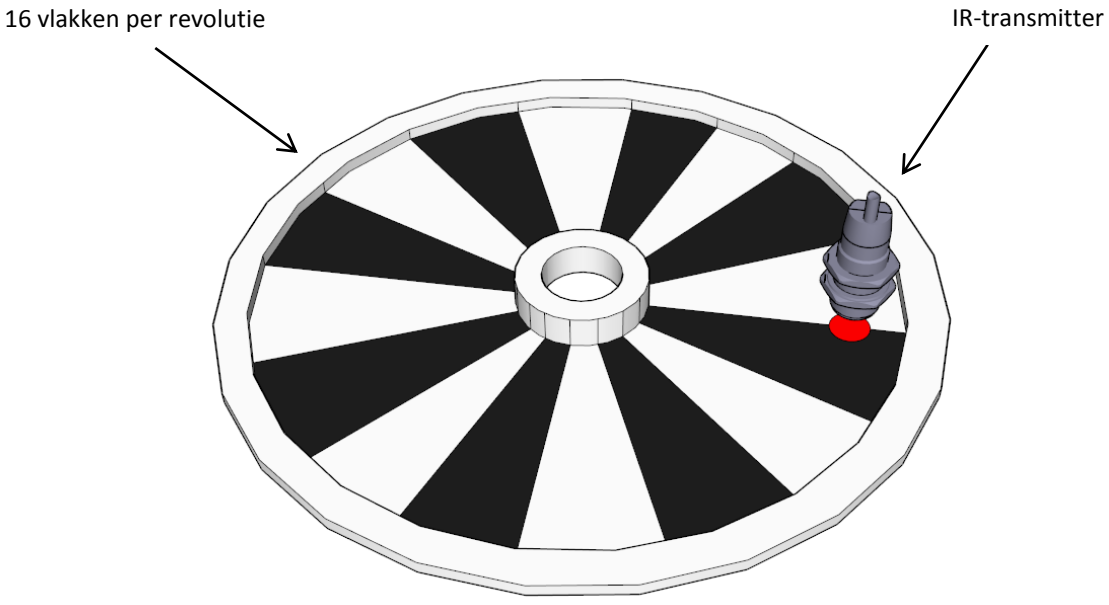
Odometrie is een techniek waarbij gekeken wordt naar de bewegende componenten van een robot om informatie te verkrijgen. Een van de meest voorkomende varianten maakt gebruik van wiel-encoders. In figuur 9.4 is een voorbeeld van een wiel-encoder weergegeven. Het is een print aan de binnenkant van een wiel waarop witte en zwarte vlakken zijn weergegeven. Boven de print is een infrarood-transmitter geplaatst. Deze IR-transmitter stuurt continu signalen. Als een signaal een wit vlak raakt weerkaatst het. Als een signaal een zwart vlak raakt gebeurt er niets. De IR-transmitter is hierdoor in staat om waar te nemen wanneer er een nieuw vlak voorbij komt. De wiel-encoder in figuur 9.4 heeft 16 vlakken. Als de IR-transmitter 16 keer een nieuw vlak voorbij ziet komen is het wiel een keer rond gegaan. Door een wiel-encoder te combineren met een timer kan veel worden berekend.

Als de diameter of straal van een wiel bekend is kan met de volgende formule de omtrek worden berekend:

$$omtrek = diameter * \pi = straal * 2 * \pi$$

Aan de hand van de omtrek van de wielen, het aantal keer dat de interrupt wordt aangeroepen en het aantal vlakken per revolutie kan de afstand worden berekend met de volgende formule:

$$afgelegde\ afstand = aantal\ interrupts * \frac{omtrek}{vlakken\ per\ revolutie}$$




***Figuur 9.4*** *Een wiel-encoder waarmee odometrie kan worden gerealiseerd.*

De snelheid van de robot kan vervolgens worden berekend met de volgende formule:

$$snelheid = \frac{(aantal\ interrupts * \frac{omtrek}{vlakken\ per\ wiel})}{verstreken\ tijd}$$

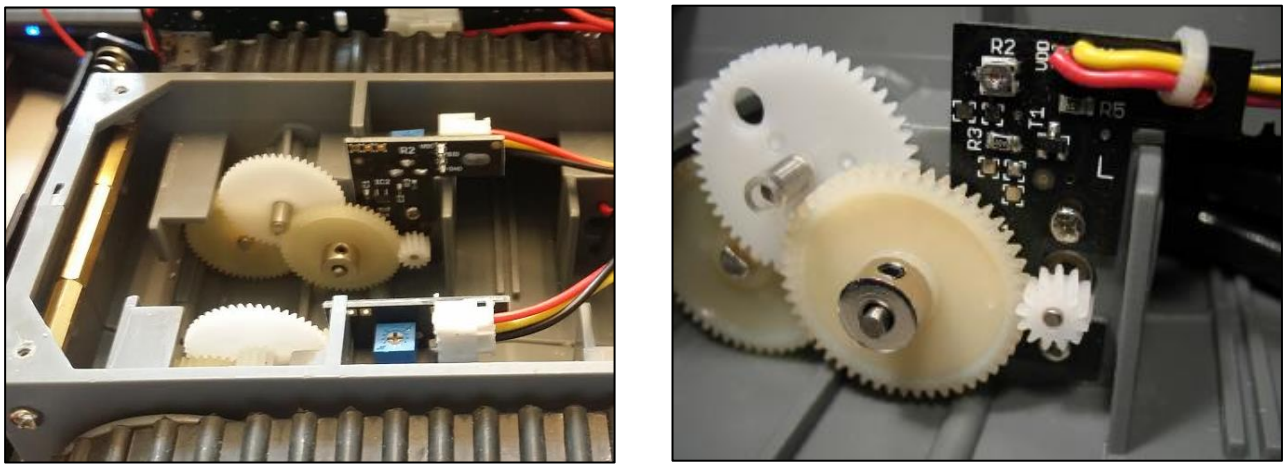
Deze formule is in staat om de gemiddelde afstand te berekenen over de verstreken tijd. Hoe vaker de snelheid wordt berekend, des te korter de verstreken tijd is en des te nauwkeuriger de gemiddelde snelheid is. Het voorbeeld in figuur 9.4 zal niet in staat zijn om tot een nauwkeurige snelheid te komen. De snelheid zal namelijk maar 16 keer per seconde kunnen worden berekend. In figuur 9.5 is weergegeven hoe de encoders van de RP6 met de wielen zijn verbonden. De “encoder vlakken” bevinden zich niet op het wiel zelf, maar op een tandwiel dat verbonden is met het daadwerkelijke wiel. Deze tandwielen zijn zo ingesteld dat de interrupt veel vaker dan 16 keer per ronde wordt aangeroepen. Het wiel zelf heeft 36 vlakken. De tandwielen bestaan uit 50 tanden aan de buitenkant en 12 tanden de binnenkant. Het aantal keer dat de interrupt per revolutie wordt aangeroepen is dus:

$$\frac{50}{12} * \frac{50}{12} * 36 = 17\frac{13}{36} * 36 = 625\ vlakken\ per\ revolutie$$

Door 625 keer per seconde de snelheid te kunnen meten kan de snelheid veel beter worden gecorrigeerd. De snelheid kan worden gezien als een golf-beweging. Het middelpunt van de golf-beweging is de gewenste snelheid. Des te vaker er gecorrigeerd wordt, des te hoger de frequentie en des te lager de amplitude zullen worden.

Bij het berekenen van afstand en snelheid door middel van wiel-encoders wordt er van uitgegaan dat de diameter van de wielen nauwkeurig is bepaald, dat de snelheid met een hoge frequentie wordt berekend en dat de robot rijdt door zijn wielen te laten draaien. De laatste eis klinkt wellicht vreemd maar het is lastig om er aan te voldoen. Een robot kan namelijk slippen en zodra de robot slipt zal de berekende afstand en snelheid niet meer kloppen. De robot is zonder andere sensoren niet in staat om dit waar te nemen en zal de foute resultaten niet herkennen.

Door middel van de hierboven weergegeven formules kan zowel de afgelegde afstand als de snelheid worden berekend. Deze formules zijn echter alleen maar toepasbaar wanneer de robot in een rechte lijn voor- of achteruit rijdt. Zodra de robot hoeken en bochten maakt zijn beide formules niet meer toepasbaar. Om in complexe situaties door middel van odometrie zijn positie te bepalen zal de robot gebruik moeten gaan maken van ‘dead reckoning’.

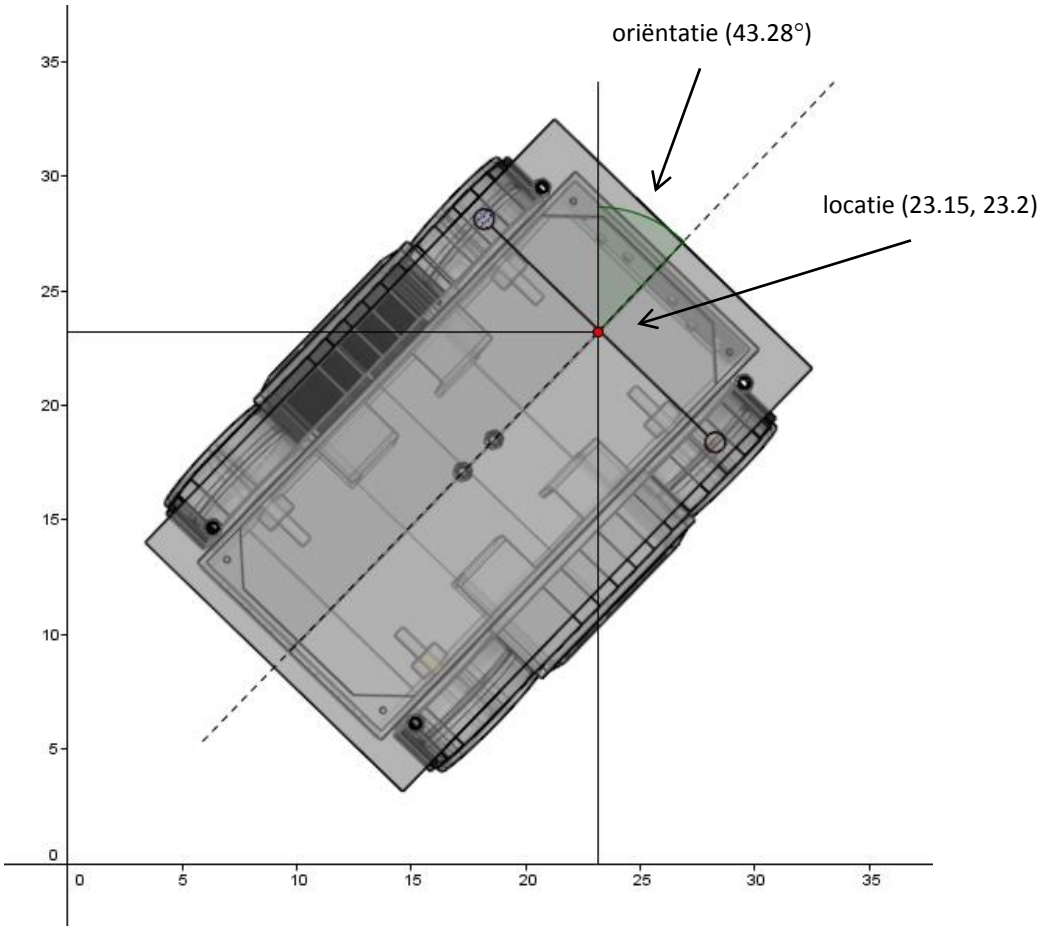



***Figuur 9.5*** *De door tandwielen aangedreven wiel-encoders van de RP6*

# 9.2 Dead reckoning

Dead reckoning (DR) is een techniek waarmee de locatie en oriëntatie van een robot kan worden benaderd door te kijken naar zijn oude positie en zijn huidige snelheid en oriëntatie. Net als optische triangulatie is dead reckoning al honderden jaren oud en werd het toegepast in de scheepsvaart om te navigeren. Dead reckoning is zo waardevol omdat het een van de weinige technieken is waarmee onafhankelijk van externe middelen locatie en oriëntatie kan worden bepaald. Zelfs vogels maken gebruik van deze techniek om hun koers aan te kunnen houden.

De RP6 kan gebruik maken van deze techniek om zijn locatie en oriëntatie bij te houden. Zoals weergegeven in figuur 9.6 zal de RP6 deze locatie en oriëntatie waarnemen als relatieve waardes binnen een assenstelsel. Het assenstelsel is een tweedimensionaal assenstelsel met het startpunt van de RP6 in de oorsprong. Aangezien optische triangulatie ook in relatieve coördinaten resulteert sluiten deze technieken goed op elkaar aan. Door middel van dead reckoning kan de RP6 zijn eigen positie ten opzichte van de waargenomen objecten bepalen.

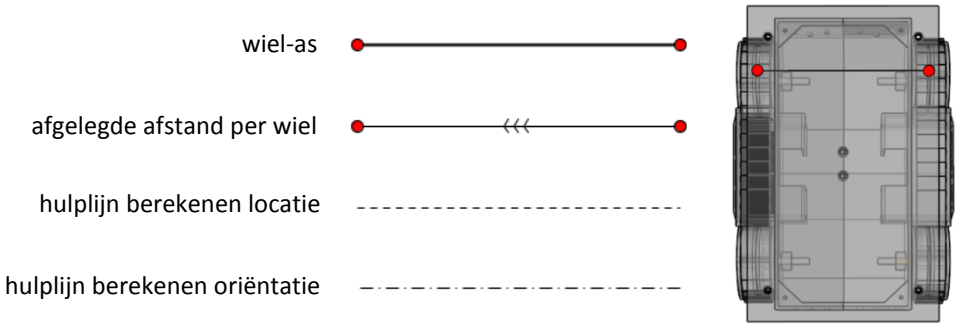



***Figuur 9.6*** *De relatieve locatie en oriëntatie van de RP6 weergegeven in een assenstelsel.*

Robots met wielen kunnen gebruik maken van odometrie om dead reckoning te implementeren. Door te kijken naar de stand van beide wielen kan zowel de locatie als de oriëntatie worden bepaald. Omdat dead reckoning gebaseerd is op odometrie kunnen verkeerd vastgestelde constanten en het slippen van de robot voor problemen zorgen. Het probleem met dead reckoning is dat de huidige locatie en oriëntatie altijd wordt bepaald door middel van de vorige locatie en oriëntatie. Dat betekend dat met iedere foute waarde wordt doorgerekend. Het resultaat is dat de absolute locatie en oriëntatie van de robot al gauw niet meer betrouwbaar zijn. De relatieve waardes blijven echter bruikbaar. De absolute waarde kan worden hersteld door middel van aanvullende sensoren en filters. Voor de RP6 is echter met name de relatieve waarde interessant.

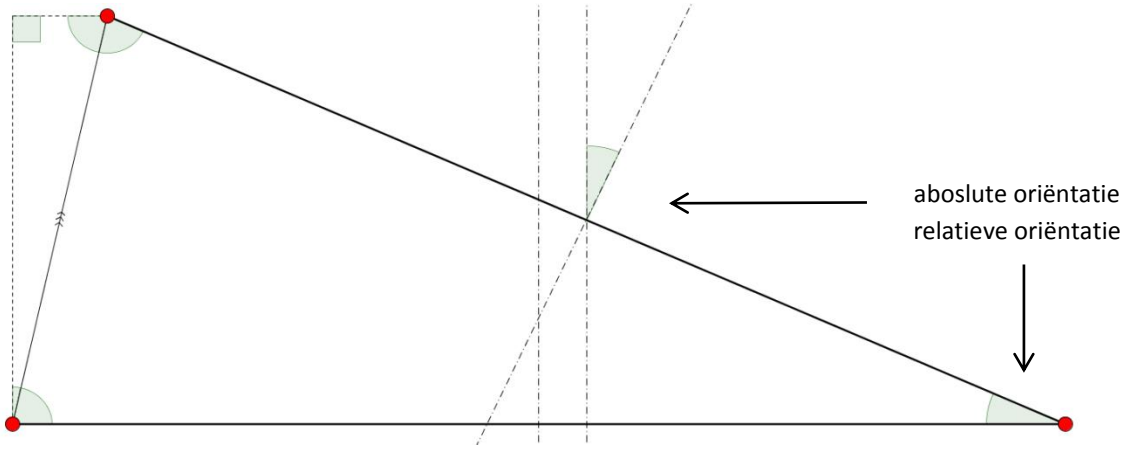
### 9.2.1 Oriëntatie

De wijze waarop de oriëntatie kan worden berekend is het beste weer te geven door middel van afbeeldingen. In figuur 9.7 is een legenda weergegeven van voorwerpen die in de afbeeldingen voorkomen.




***Figuur 9.7*** *Legenda van voorwerpen uit de hieronder weergegeven figuren.*

In figuur 9.8 zijn de wielen van de RP6 in twee verschillende situaties weergegeven. In de eerste situatie staan de wielen horizontaal. In de tweede situatie is heeft het linker wiel een stuk gereden. In de eerste situatie was de absolute en relatieve oriëntatie 0°. In de tweede situatie is de relatieve en absolute oriëntatie veranderd. Deze oriëntatie kan worden berekend. De oriëntatie is een hoek van een driehoek waarvan alle zijdes bekend zijn. De twee gelijkbenige zijdes van het driehoek zijn namelijk de wiel-as van de RP6. De korte zijde van het driehoek is de afgelegde afstand van het linker wiel. Deze afstand kan worden bepaald door middel van odometrie.

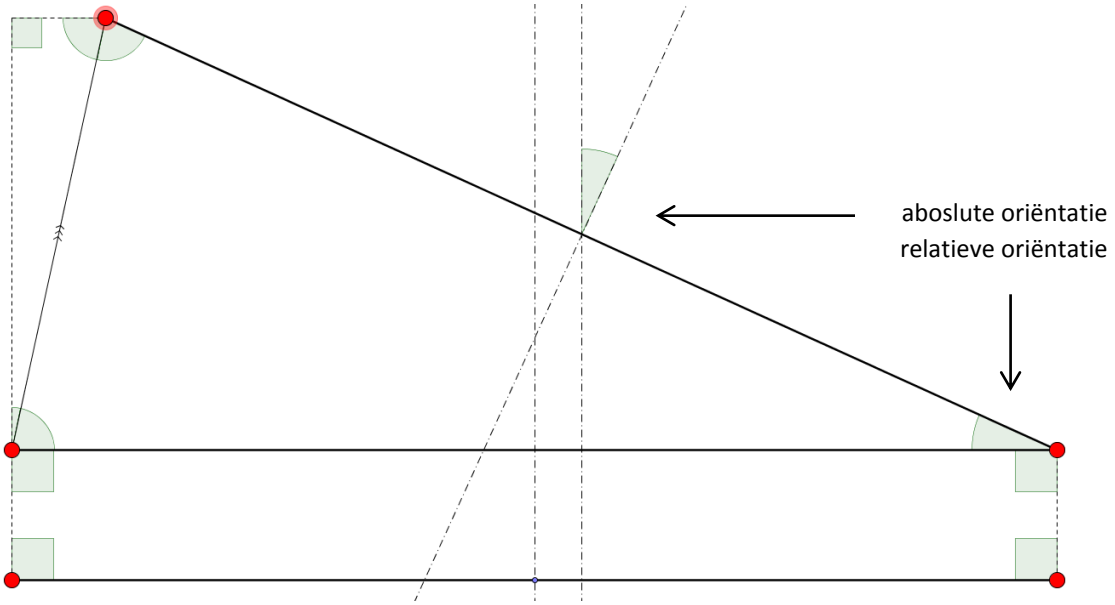



***Figuur 9.8*** *Oriëntatie kan bepaald worden aan de hand van twee sequentiële posities van de wielen.*

Als alle zijdes van het driehoek bekend zijn kan met de volgende formule de hoeken worden berekend:

$$relatieve\ oriëntatie = 180 - (cos^{-1}\frac{((afgelegde\ afstand^2) + (wiel\ as)^2 - (wiel\ as)^2)}{(2 * (afgelegde\ afstand) * (wiel\ as))}) * 2$$

De relatieve oriëntatie is positief als het linker wiel verder heeft gereden en negatief als het rechter wiel verder heeft gereden. De absolute oriëntatie is de som van alle (positieve en negatieve) relatieve oriëntaties. Het kan ook voorkomen dat beide wielen zich verplaatsen. In deze situatie wordt er aangenomen dat beide wielen eerst met een gelijke snelheid naar voren rijden. Als een van de twee wielen een langere afstand heeft afgelegd zal dat wiel vervolgens alleen verder rijden. In figuur 9.9 is het resultaat van deze interpretatie weergegeven. Deze aanpak is noodzakelijk omdat het niet mogelijk is om achteraf te bepalen wat er echt heeft plaats gevonden. Vanwege deze aanpak zal de oriëntatie iedere keer dat de wiel-encoder een interrupt genereerd moeten worden berekend. Als de oriëntatie niet zo vaak wordt berekend kan er door deze werkwijze een afwijking ontstaan.




***Figuur 9.9*** *De oriëntatie kan ook bepaald worden wanneer beide wielen zich verplaatsen.*

9.2.2 Locatie

De nieuwe locatie van de RP6 kan op een vergelijkbare wijze worden berekend. In figuur 9.10 zijn de afstanden weergegeven die moeten worden berekend om de nieuwe locatie te kunnen bepalen. Bij het berekenen van de locatie wordt er net als bij het berekenen van de oriëntatie van uitgegaan dat beide wielen met een gelijke snelheid verplaatsen. Als beide wielen zich verplaatsen zal daarom eerst een stuk rechtuit worden gereden. Deze nieuwe locatie zal daarom ook worden berekend door middel van de oude oriëntatie van de RP6. De nieuwe x- en y-coördinaten over een rechte afstand kunnen vervolgens met de volgende formules worden berekend:

$$nieuw\ x\ = oud\ x\ + (\sin(oude\ oriëntatie) * afgelegde\ rechte\ afstand)$$

$$nieuw\ y\ = oud\ y\ + (\cos(oude\ oriëntatie) * afgelegde\ afstand)$$

Na met de oude absolute oriëntatie het rechte stuk te hebben berekenen kan met de nieuwe relatieve oriëntatie het laatste stuk worden berekend. Deze berekening is ook van toepassing wanneer maar een van de twee wielen zich heeft verplaatst. In de eerste situatie zijn de oude x- en y-coördinaten het resultaat van de voorgaande formules. In de tweede situatie zijn de oude x- en y-coördinaten daadwerkelijk de coördinaten van zijn oude positie. De relatieve oriëntatie moet in deze berekeningen als positieve waarde worden toegepast.

Als het rechter wiel verder heeft gereden kan door middel van de volgende formules de locatie worden berekend:

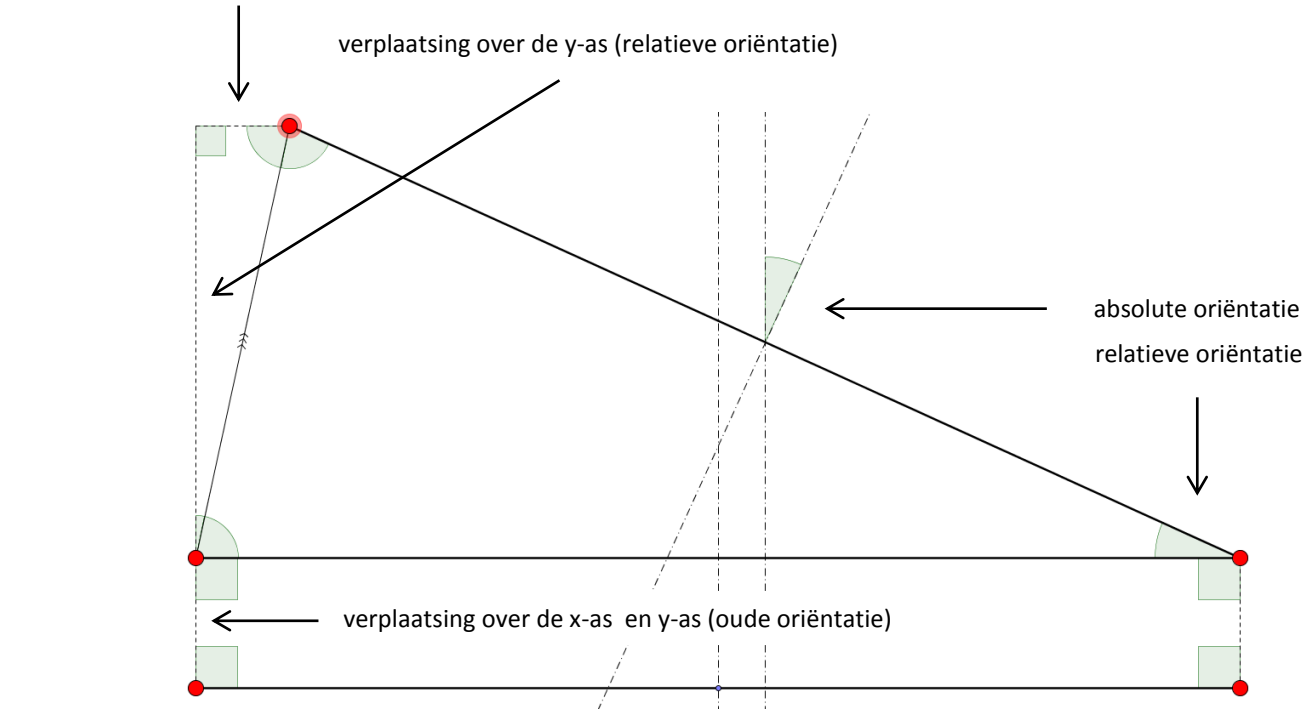
$$nieuw\ x = oud\ x + \frac{\cos(180 - \left(\frac{180 - relatieve\ oriëntatie}{2}\right) - oude\ oriëntatie) * afgelegde\ afstand}{2}$$

$$nieuw\ y = oud\ y + \frac{\sin(180 - \left(\frac{180 - relatieve\ oriëntatie}{2}\right) - oude\ oriëntatie) * afgelegde\ afstand}{2}$$

Als het linker wiel verder heeft gereden kan door middel van de volgende formules de locatie worden berekend:

$$nieuw\ x = oud\ x + \frac{\sin(90 - \left(\frac{180 - relatieve\ oriëntatie}{2}\right) + oude\ oriëntatie) * afgelegde\ afstand}{2}$$

$$nieuw\ y = oud\ y + \frac{\cos(90 - \left(\frac{180 - relatieve\ oriëntatie}{2}\right) + oude\ oriëntatie) * afgelegde\ afstand}{2}$$




***Figuur 9.10*** *De afgelegde afstand kan worden bepaald wanneer beide wielen zich verplaatsen.*

### 9.2.3 Problemen

Systemen kunnen door middel van dead reckoning hun positie benaderen. De techniek is niet betrouwbaar genoeg om de positie daadwerkelijk te bepalen. De techniek kan in combinatie met andere sensoren worden ingezet om informatie te verschaffen over locaties en oriëntaties. Hoewel dead reckoning op veel verschillende systemen kan worden toegepast (luchtvoertuigen, schepen, auto's), kampen al deze systemen met dezelfde soort problemen.

Bij het berekenen van locatie en oriëntatie wordt er gesproken over relatieve en absolute waardes. De relatieve waardes zijn de locatie en oriëntatie ten opzichte van de vorige positie van het systeem. De absolute waardes zijn de locatie en oriëntatie ten opzichte van de startpositie van het systeem. Absolute waardes worden bepaald door verder te rekenen met alle relatieve waardes. Het probleem met deze techniek is dat hele kleine fouten in de relatieve waardes, op de lange termijn zullen leiden tot hele grote fouten in de absolute waarde. Deze fouten kunnen zo klein zijn dat ze in de relatieve waardes niet waar te nemen zijn, maar toch de resultaten van de absolute waardes over een lange termijn onbruikbaar kunnen maken.

Kleine fouten in relatieve waardes kunnen op veel verschillende manieren ontstaan. Foute constante waardes zoals de maten van de wiel-as, de omtrek van de wielen en de positie van de draaipunten kunnen snel tot fouten leiden. Foute constante zijn een complex probleem omdat deze waardes vaak afhankelijk zijn van veranderende aspecten zoals de ondergrond, de massa van de robot en slijtage van de onderdelen.

Een probleem waar systemen met wielen mee te maken hebben is slippen. Bij dead reckoning wordt ervan uitgegaan dat wanneer wielen omwentelen er een specifieke afstand wordt afgelegd. Wanneer wielen slippen zullen de wielen omwentelen maar niet de verwachte afstand afleggen. Slippen zal hierdoor leiden in grote fouten in zowel de relatieve als absolutie locatie en oriëntatie. Het grootste probleem met slippen is dat het niet waargenomen kan worden. Omdat systemen niet weten wanneer het gebeurt, kunnen ze niet corrigeren.

De RP6 heeft geen wielen maar rupsbanden. De wijze waarop de locatie en oriëntatie bepaald kunnen worden zijn voor rupsbanden en wielen hetzelfde. Het probleem met rupsbanden is dat ze zich verplaatsen door te slippen. Oftewel, rupsbanden slippen altijd. Hoewel ze altijd slippen kan er niet worden bepaald hoeveel ze slippen. Dit is namelijk afhankelijk van aspecten zoals de snelheid, versnelling, massa en ondergrond. Met name het bepalen van de oriëntatie door middel van dead reckoning is niet realistisch wanneer er sprake is van rupsbanden.

Toch maken bijna alle autonome systemen gebruik van dead reckoning. In combinatie met andere sensoren kan het namelijk waardevolle informatie leveren. Door bijvoorbeeld de resultaten van dead reckoning te combineren de resultaten van een kompas, kan over een lange termijn op een betrouwbaardere wijze locatie en oriëntatie worden bepaald. Het Kalman-filter is een bekende manier om de resultaten van verschillende sensoren (bijvoorbeeld GPS) te combineren om tot betrouwbare resultaten te komen.

Voor optische triangulatie zijn de absolute locatie en oriëntatie minder interessant dan de relatieve waardes. Het systeem zal namelijk met name keuzes moeten maken aan de hand van zijn locatie en oriëntatie ten opzichte van nabije objecten. Bij rupsbanden zal het slippen echter ook leiden tot fouten in de relatieve waardes. Het is daarom niet aan te raden om optische triangulatie te implementeren op systemen met rupsbanden. Er is momenteel veel onderzoek gaande naar autonome navigatie van voertuigen met rupsbanden, maar de resultaten geven aan dat betrouwbare dead reckoning zonder additionele sensoren niet haalbaar is. [6][7]

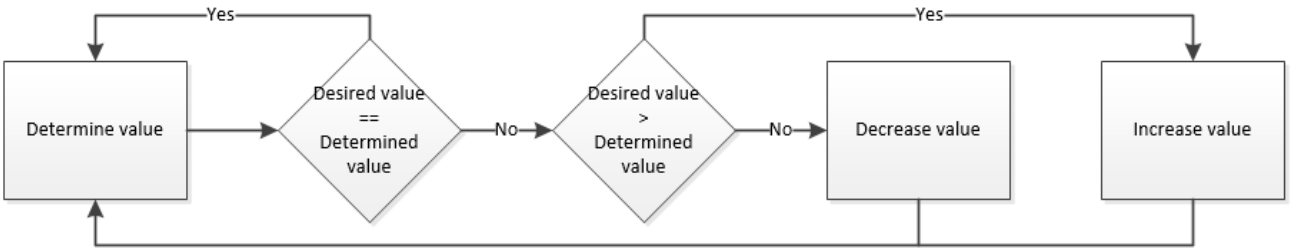
De RP6 blijft echter een testplatform en is gekozen omdat er makkelijk over te beschikken was. Omdat de testplatformen waarvoor het systeem eigenlijk bedoeld is wel wielen hebben wordt de ontwikkeling doorgezet.

# 9.3 Motoraansturing

Odometrie is een middel waarmee onder andere de snelheid van een robot bepaald kan worden. Zonder odometrie zou een robot niet in staat zijn om met een specifieke snelheid te rijden. De motoren van een robot weten namelijk niet wat snelheid is. De motoren van de RP6 worden aangestuurd door middel van pulsbreedtemodulatie. Pulsbreedtemodulatie is een techniek waarmee het elektrisch vermogen van een stroomtoevoer kan worden gecontroleerd. Hierdoor is het mogelijk om de RP6 met verschillende snelheden te laten rijden. Deze snelheden zijn het resultaat van verschillende 'duty-cycles' van de pulsbreedtemodulatie. De duty-cycle is een combinatie van onder andere de frequentie, amplitude en de duur van de puls. Door odometrie en pulsbreedtemodulatie te combinaren kan snelheid worden omgezet in een duty-cycle.

In figuur 9.11 is weergegeven hoe de motoren met een specifieke snelheid kunnen worden aangestuurd. Bij 'start' worden de motoren met een willekeurige (lage) snelheid aangezet. Vervolgens zal de snelheid van de motoren worden bepaald. Als de snelheid van de motoren lager is dan de gewenste snelheid zal de duty-cycle van de motoren worden verhoogd. Als de snelheid van de motoren hoger is dan de gewenste snelheid zal de duty-cycle van de motoren worden verlaagd. Door continu deze controle uit te voeren kan de snelheid worden beheerst.

De frequentie van deze routine en de toename en afname in snelheid per correctie zijn bepalend voor de precisie waarmee de snelheid kan worden beheerst. Deze routine is effectief wanneer de robot een bepaalde snelheid moet aanhouden. De routine is minder effectief wanneer de robot vanuit stilstand moet gaan rijden. De motoren zijn vanuit stilstand niet in staat om de frequentie toename in snelheid bij te houden. De routine zal echter continu de motoren opdragen sneller te rijden totdat ze de gewenste snelheid bereiken. Omdat de motoren achter liggen met het opvolgen van deze commando's zullen de motoren zodra ze wel op gang zijn gekomen voorbij de gewenste snelheid schieten. Vervolgens zullen ze weer dalen tot de gewenste snelheid. Het resultaat is dat de robot vanuit stilstand naar voren schiet om vervolgens de deceleren en  met de gewenste snelheid te gaan rijden. Om dit te voorkomen zal de routine door middel van verschillende 'constraints' moeten worden gecontroleerd. (Deze routine kan ook worden toegepast om onder andere de oriëntatie en positie van de robot te beheersen.)




***Figuur 9.11*** *Waardes zoals de snelheid en oriëntatie kunnen door middel van feedback worden gestuurd.*

## 9.4 Real-time aspecten

Waar het bij het optische triangulatie systeem van belang is dat een frequentie van een paar hertz kan worden behaald gaat het bij de driver van de RP6 om microseconden. Bij het berekenen van de onder andere de snelheid, locatie en oriëntatie van de RP6 is precisie essentieel. Met hele kleine afwijkingen in de timers kunnen namelijk over een langere periode hele grote afwijkingen in de locatie en oriëntatie accumuleren. Door de timer zo nauwkeurig mogelijk af te stellen kan de grootste precisie worden behaald. De beste frequentie die met de algemene timer van de Atmega kan worden behaald in 10 kHz. Er kan gesteld worden dat hierdoor al precisie verloren gaat aangezien alle stappen tussen de 10 kHz 'ticks' verloren gaan. Deze afwijking is echter verwaarloosbaar vergelijken met de andere aspecten die goed moeten worden afgehandeld. Een voorbeeld hiervan is de wijze waarop de snelheid, locatie en oriëntatie worden berekend. In een van de eerdere prototypes werden deze waardes berekend door over bepaalde stukken tijd te kijken hoeveel de encoder zich verplaatste. Het probleem hierbij is dat het systeem niet in staat is om het verschil te zien tussen drie stappen en 'bijna vier' stappen. Hierdoor ontstaat er al gauw een afwijking die het systeem zelf niet meer kan corrigeren. Een betere aanpak is om niet over een bepaalde tijd, maar voor iedere individuele verplaatsing de waardes te berekenen. Zowel de snelheid als de locatie en oriëntatie waren aanzienlijk meer accuraat door deze techniek te hanteren.

Een ander aspect waar rekening mee gehouden moet worden is de frequentie waarmee de routines worden uitgevoerd. Hoewel een hoge frequentie wenselijk is voor een hoge nauwkeurigheid, kan de frequentie niet alleen worden gekoppeld aan interrupt van de encoders. Als dit wel zou worden gedaan dan zou de frequentie waarmee de routines worden uitgevoerd volkomen afhankelijk zijn van de snelheid van de robot. Dit zou er toe kunnen leiden dat het programma overbelast raakt bij een te hoge snelheid. Hierdoor is het dus noodzakelijk om de verplaatsing van de encoders te controleren in een 'loop' met een begrensde frequentie. Het resultaat is helaas wel dat de nauwkeurigheid van de snelheid, locatie en oriëntatie af zullen nemen als de snelheid te veel toeneemt.

Hierdoor is het dan weer wel interessant om de verschillende technieken met een zo groot mogelijk efficiëntie te implementeren. Wiskundige functies zoals de tangus, sinus en cosinus kunnen het systeem aardig belasten. Door hier slim mee om te gaan kan er een hoop processortijd worden bespaard. Er kan dus gesteld worden dat bij alle onderdelen van de driver van de RP6, de real-time aspecten een aanzienlijke rol spelen.

# 10. Ontwerpfase

*21 april 2014  t/m 25 april 2014 (1 week)*

In de tweede onderzoeksfase wordt de basis van de RP6 ontworpen. De RP6 is een autonome robot met rupsbanden. De ontworpen basis  de RP6 in staat stellen om autonoom door een kamer te rijden en zijn eigen snelheid, locatie en oriëntatie te bepalen. De software-module zal worden geschreven in de programmeertaal C.

In de ontwerpfase wordt met name aandacht besteed aan:

Welke eisen worden er aan het systeem gesteld? (**10.1**)

Op welke wijze kunnen de technieken uit de onderzoeksfase het beste worden samengebracht? (**10.2**)

# 10.1 Eisen

De eisen die aan RP6 worden gesteld moeten de RP6 in staat stellen om gebruik te maken van de informatie die door middel van optische triangulatie kan worden verkregen. Deze eisen zijn het resultaat van twee onderzoeksfasen en gesprekken met de werknemers van Berkelaar Meet- en Regeltechniek. Er wordt onderscheid gemaakt tussen functionele eisen en niet-functionele eisen. Functionele eisen geven de kerntaken van het systeem weer en niet-functionele eisen geven de kwaliteit en eigenschappen van het systeem weer.

**Functionele eisen**:

*Het systeem moet zijn eigen snelheid kunnen bepalen.*

*Het systeem moet zijn eigen locatie kunnen bepalen ten opzichte van zijn startpositie.*

*Het systeem moet zijn eigen oriëntatie kunnen bepalen ten opzichte van zijn startpositie.*

*Het systeem moet een specifiek aangegeven snelheid kunnen rijden.*

**Niet-functionele eisen**:

*De maximale frequentie waarmee snelheid, locatie en oriëntatie bepaald wordt moet gekoppeld zijn aan tijd.*
*Door processen te koppelen aan interrupts (oftewel de snelheid van de wielen) kan bij een te hoge snelheid het systeem overbelast raken. Het is daarom essentieel dat de processen onder controle gehouden kunnen worden.*

*De robot moet bochten maken door een wiel harder te laten rijden dan het andere.*
*Een systeem met rupsbanden kan op de plaats roteren om van richting te veranderen. Het is echter de bedoeling dat het systeem soepel kan rondrijden. Van richting veranderen door te roteren is daarom onwenselijk.*

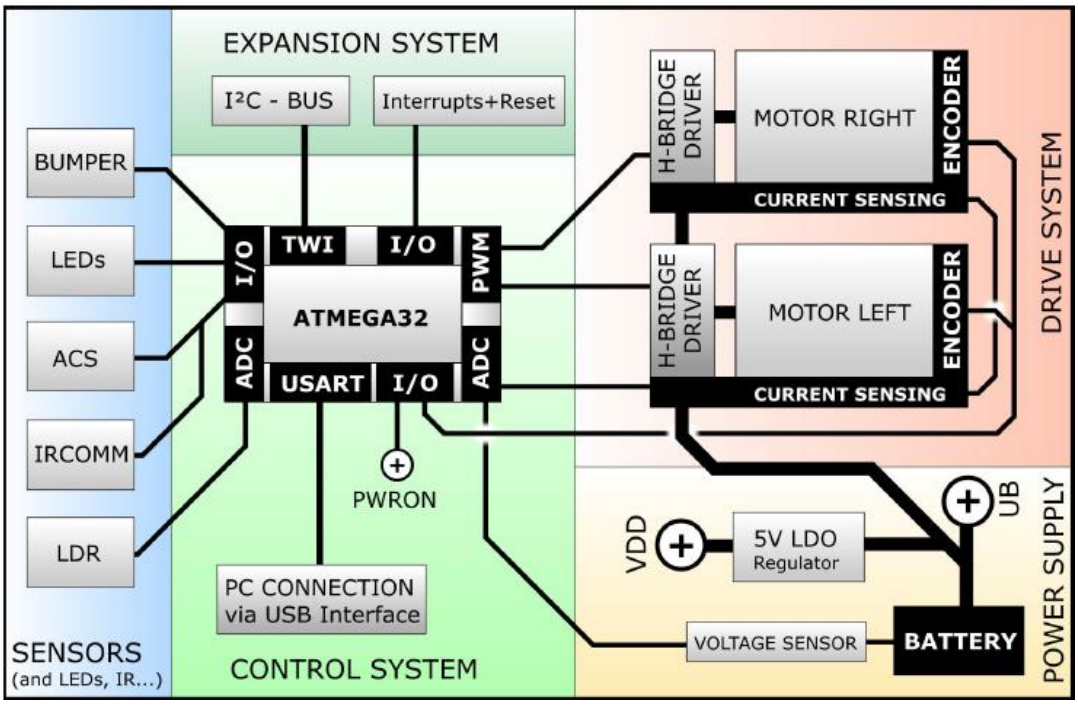
*Als de snelheid moet veranderen moet dit geleidelijk gebeuren.*
*Het systeem kan heel snel optrekken, van snelheid veranderen en tot stilstand komen. Hoewel het in sommige situaties nut kan hebben moet het systeem soepel kunnen rondrijden en niet op deze schokkende wijze.*

## 10.2 Technische specificaties

De software-module van de RP6 zal worden geïmplementeerd op een ATmega32 microcontroller. Deze microcontroller ondersteunt alleen de programmeertaal C. De software-module bestaat daarom niet uit klassen, maar uit functies. Deze functies maken gebruik van verschillende globale variabelen. De interactie tussen de functies is daarom niet overzichtelijk weer te geven door middel van modelleertalen zoals UML. Om deze reden is de onderlinge interactie tussen de functies weergegeven door middel van een abstract functiediagram.

Veel van de functionaliteit van de software-module is afhankelijk van de onderliggende hardware. De kern van deze hardware is een ATmega32 8-bit microcontroller met een 8MHz kloksnelheid. De microcontroller bevat 32KB flash ROM, 2KB SRAM en 1KB EEPROM. De microcontroller ondersteunt de programmeertaal C (WinAVR/AVR-GCC). De microcontroller bevat een USB-PC interface (USART) die aangesloten kan worden op onder andere de computer en de Raspberry Pi. De robot kan zich verplaatsen door middel van twee rupsbanden die worden aangestuurd door twee 7.2V DC-motoren. Hiermee kan de RP6 snelheden behalen tot op 25 centimer per seconde. De RP6 kan informatie zoals zijn eigen snelheid, locatie en oriëntatie bepalen door middel van twee 625 CPR (counts per revolution) wiel-encoders. Hiermee is de RP6 in staat om afstanden waar te nemen van 0.25 milimeter.

-



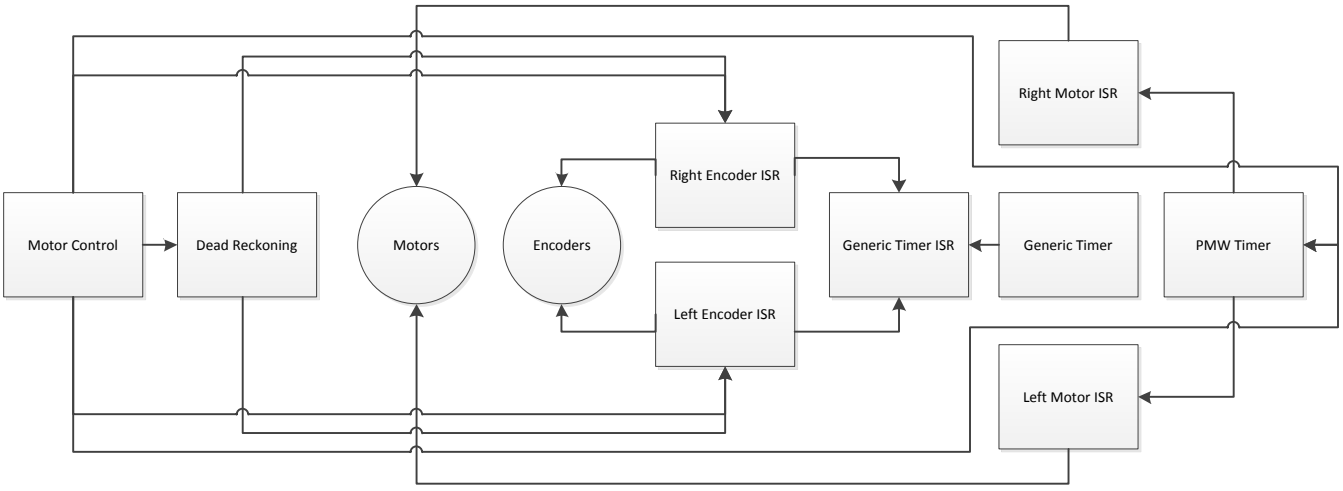

***Figuur 10.1*** *Een overzicht van de componenten waar de ATmega32/RP6 beschikking over heeft.*

In figuur 10.2 is de basis van het ontwerp van de RP6 weergegeven. Het ontwerp bevat de functionaliteit om de *snelheid*, *locatie* en *oriëntatie* van de RP6 te bepalen en te veranderen. De functies **Motor Control** en **Dead Reckoning** bevat het meerendeel van de functionaliteit. De functie **Motor Control** is verantwoordelijk om de huidige *snelheid*, *locatie* en *oriëntatie* te wijzigen naar de gewenste *snelheid*, *locatie* en *oriëntatie*. De gewenste *snelheid*, *locatie* en *oriëntatie* zijn waardes die in de derde iteratie zullen worden bepaald aan de hand van optische triangulatie. De huidige *snelheid* is een waarde die wordt bepaald in de **ISR**'s (Interrupt Service Routines)

van de encoders. Het bepalen van de huidige *locatie* en *oriëntatie* kost te veel processortijd om in een **ISR** uit te laten voeren. Om deze reden worden de huidige *locatie* en *oriëntatie* in de functie **Dead Reckoning** bepaald. De functie **Dead Reckoning** maakt net zoals de functie **Motor Control** gebruik van de **ISR**'s van de encoders om de *locatie* en *oriëntatie* te bepalen. Als **Motor Control** vervolgens op de hoogte is van de huidige *snelheid*, *locatie* en *oriëntatie* kan **Motor Control** door de **PMW Timer** te veranderen de **ISR**'s die de motoren aansturen beïnvloeden. Tot slot is er de **Generic Timer**. Deze timer fungeert als algemene klokfunctie en wordt in nagenoeg alle andere functies en interrupts gebruikt om *frequenties* in te stellen en *snelheden* te bepalen.

De meest voorkomende datatypes in de software-module zijn 'structs' en 'defines'. Struct maken een groot deel van de structuur van de software-module uit. Omdat de software-module niet in een objectgeoriënteerde taal is ontwikkeld kan er geen gebruik worden gemaakt van objecten. Toch zijn er van sommige variabelen (zoals de motoren en encoders) meerdere instanties. Door structs te gebruiken kan het aantal variabelen overzichtelijk worden gehouden. Nagenoeg alle andere datatypes zijn defines. De software-module maakt gebruik van een aanzienlijk aantal gegevens dat gedurende de uitvoer van het programma niet zal veranderen (maten, constanten). Door gebruik te maken van defines kan overzicht worden behouden en de code leesbaar worden gemaakt.

De software-module bestaat uit een grote loop (repetitie). De frequentie waarmee de functies worden aangeroepen wordt echter bepaald door de **GenericTimer**. Zonder de frequentie zelf te definiëren zou de frequentie afhankelijk zijn van de snelheid van de processor en de snelheid van de RP6. Een te hoge snelheid of een wijziging in de code zou anders grote gevolgen kunnen hebben voor de uitvoer van de software-module.

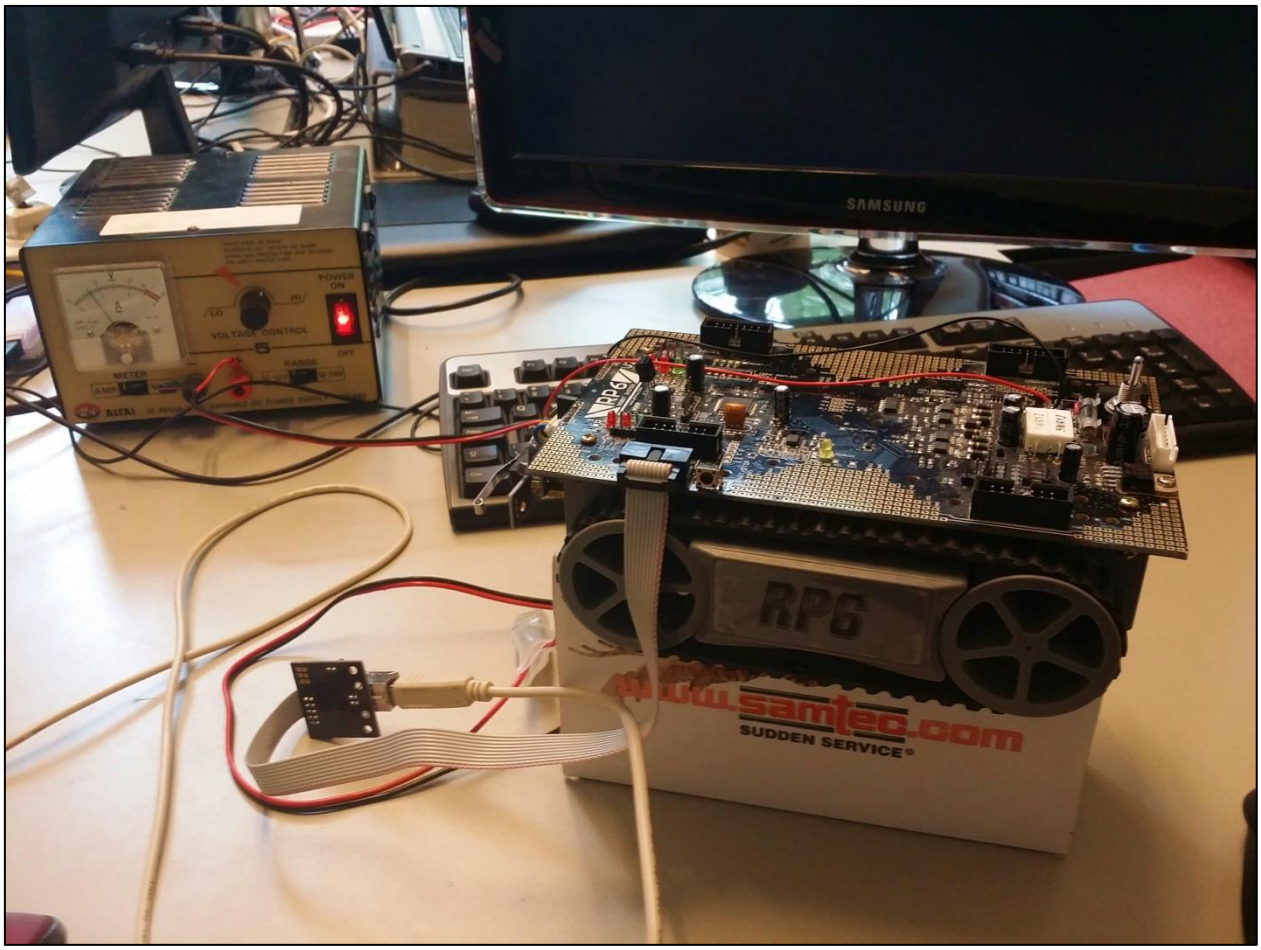



***Figuur 10.2*** *Abstract ontwerp van de software-module van de RP6*

# 11. Implementatiefase

*28 april 2014 t/m 9 mei 25 2014 (2 week)*

In de tweede implementatiefase wordt het ontwerp van de software-module voor de RP6 geïmplementeerd. Gedurende de implementatiefase wordt er onderscheid gemaakt tussen twee niveaus van functionaliteit. Het eerste niveau bevat de 'low-level' interactie met de hardware. Hier toe behoren met name de timers en de interrupts. Het tweede niveau bevat de implementatie van de technieken waarmee de snelheid, locatie en oriëntatie kunnen worden bepaald. De hele software-module is ontwikkeld en gecompileerd op een notebook (MSI-GP60) en vervolgens door middel van een USB-interface geüpload naar de RP6. De code is gecompileerd door middel van een gcc compiler en een 'makefile'. De code is geüpload door middel van een tool genaamd 'Robotloader'. Deze tool is ontwikkeld door de makers van de RP6 en is in staat om .hex-bestanden over te zetten van een notebook naar het flash-geheugen van de RP6. Deze tool heeft tevens de beschikking over onder andere een terminal, debug-fasciliteiten en de mogelijkheid om instellingen van de ATmega32 te configureren. Om te voorkomen dat de batterijen van de RP6 tijdens het ontwikkelen leeglopen is de RP6 aangesloten op een externe accu. In figuur 11.1 zijn de RP6, de externe accu en de USB-interface weergegeven.




***Figuur 11.1*** *Een foto van de RP6, externe accu en USB-interface waarmee tijdens de tweede iteratie ontwikkeld is.*

```
/*****************************************************************************************\
|                                                                                         |
|   TODO: The relative orientation has to be doubled to fit the model and calculations?   |
|                                                                                         |
\*****************************************************************************************/

void dead_reckoning()
{
    // Calculates the new location and orientation in the case that the left wheel travelled further than the right
    if(left_encoder.encoder_value_increase > right_encoder.encoder_value_increase)
    {
        int mutual_distance = right_encoder.encoder_value_increase;
        int additional_distance = left_encoder.encoder_value_increase - right_encoder.encoder_value_increase;

        current_position.x += (sin(current_position.orientation * (PI / 180)) * (mutual_distance * ENCODER_STEP));
        current_position.y += (cos(current_position.orientation * (PI / 180)) * (mutual_distance * ENCODER_STEP));

        float relative_orientation = (180 - (acos(pow((additional_distance * ENCODER_STEP), 2) / (2 * (additional_distance * ENCODER_STEP) * AXLE_SIZE)) * (180 / PI)) * 2);

        current_position.x += ((cos((180 - ((180 - relative_orientation)/2) - current_position.orientation) * (PI / 180)) * (additional_distance * ENCODER_STEP)) / 2);
        current_position.y += ((sin((180 - ((180 - relative_orientation)/2) - current_position.orientation) * (PI / 180)) * (additional_distance * ENCODER_STEP)) / 2);

        current_position.orientation += (relative_orientation / 2); // Requires attention ---> ( / 2) <---
    }

    // Calculates the new location and orientation in the case that the right wheel travelled further than the left
    if(left_encoder.encoder_value_increase < right_encoder.encoder_value_increase)
    {
        int mutual_distance = left_encoder.encoder_value_increase;
        int additional_distance = right_encoder.encoder_value_increase - left_encoder.encoder_value_increase;

        current_position.x += (sin(current_position.orientation * (PI / 180)) * (mutual_distance * ENCODER_STEP));
        current_position.y += (cos(current_position.orientation * (PI / 180)) * (mutual_distance * ENCODER_STEP));

        float relative_orientation = -(180 - (acos(pow((additional_distance * ENCODER_STEP), 2) / (2 * (additional_distance * ENCODER_STEP) * AXLE_SIZE)) * 180 / PI) * 2);

        current_position.x += ((sin((90 - ((180 - relative_orientation)/2) + current_position.orientation) * (PI / 180)) * (additional_distance * ENCODER_STEP)) / 2);
        current_position.y += ((cos((90 - ((180 - relative_orientation)/2) + current_position.orientation) * (PI / 180)) * (additional_distance * ENCODER_STEP)) / 2);

        current_position.orientation += (relative_orientation / 2); // Requires attention ---> ( / 2) <---
    }

    // Calculates the new location in the case that both wheels travelled the same distance
    if(left_encoder.encoder_value_increase == right_encoder.encoder_value_increase)
    {
        current_position.x += (sin(current_position.orientation * (PI / 180)) * (left_encoder.encoder_value_increase * ENCODER_STEP));
        current_position.y += (cos(current_position.orientation * (PI / 180)) * (left_encoder.encoder_value_increase * ENCODER_STEP));
    }
}
```

***Figuur 11.2*** *Voorbeeld-code: de implementatie van de functie dead_reckoning().*

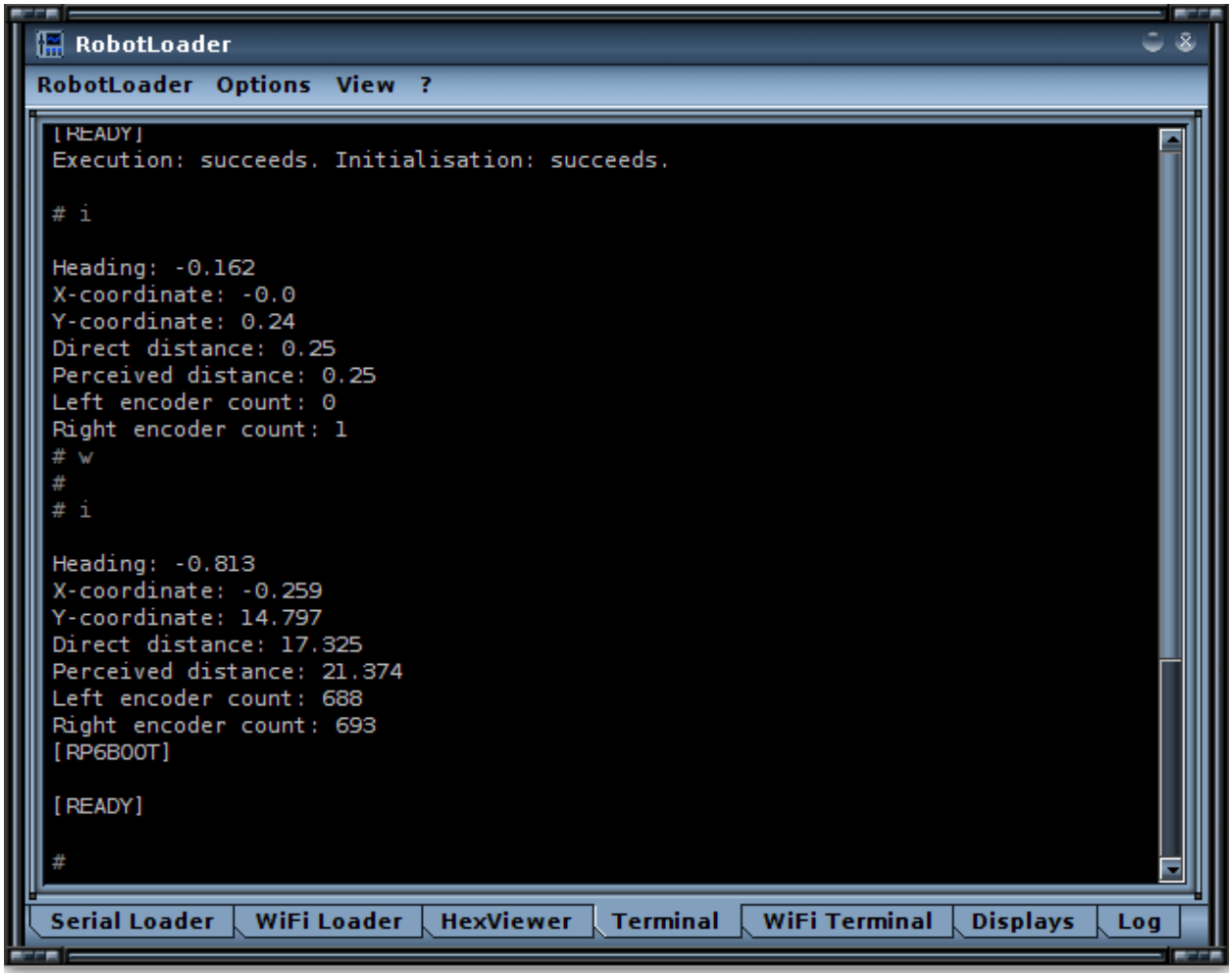
```
/*****************************************************************************************************\
|                                                                                                     |
|   TODO: Find the loopholes in the calculations which allow the current speed to be incorrect (0.00)   |
|                                                                                                     |
\*****************************************************************************************************/

// Calculates and controls the current speed and direction of the wheels based on the encoders and passed time
void motor_control()
{
    // Calculates the current speed of the left motor and improves it towards the desired speed
    if(left_encoder.encoder_value - left_encoder.previous_encoder_value)
    {
        left_encoder.encoder_value_increase = left_encoder.encoder_value - left_encoder.previous_encoder_value;
        left_encoder.previous_encoder_value = left_encoder.encoder_value;

        // Calculates the average speed (cm/s) since the last calculation
        left_motor.current_speed = ((left_encoder.encoder_value_increase * ENCODER_STEP) / left_encoder.timer_counter1) * REFACTOR_VALUE;

        left_encoder.timer_counter1 = 0;

        // Increases the speed of the left motor
        if(left_motor.current_speed < left_motor.desired_speed)
        {
            if(LEFT_MOTOR_SPEED < MAXIMUM_SPEED) LEFT_MOTOR_SPEED += SPEED_GAIN;
        }

        // Decreases the speed of the left motor
        if(left_motor.current_speed > left_motor.desired_speed)
        {
            if(LEFT_MOTOR_SPEED > MINIMUM_SPEED) LEFT_MOTOR_SPEED -= SPEED_GAIN;
        }
    }
```

***Figuur 11.3*** *Voorbeeld-code: een deel van de implementatie van de functie motor_control() die de snelheid berekent.*

# 12. Testfase

*van 12 mei 2014 t/m 16 mei 2014  (1 week)*

In de tweede testfase staat het testen van de autonome robot centraal. De tests zijn gebaseerd op onder andere de functionele eisen  en niet-functionele eisen (P.10.1) en bekende problemen (P. 9.2.3). Hoewel alle functies zijn doorlopen is er met name aandacht besteed aan de resultaten van odometrie. Er zijn veel tests uitgevoerd om te testen hoe goed de RP6 in staat is om zijn eigen snelheid, locatie en oriëntatie te bepalen en aan te sturen.  Deze tests zijn uitgevoerd door de RP6 verschillende afstanden te laten rijden en rotaties te laten maken. Door de daadwerkelijke verplaatsingen te meten met een meetlint en geodriehoek en de waargenomen verplaatsingen te printen naar het scherm konden de afwijkingen worden bepaald. Uit deze testen kwam naar voren dat nagenoeg alle functies naar behoren werken behalve het bepalen en aansturen van de oriëntatie. Zodra de RP6 moet rijden zal de robot slippen. Hierdoor zal een oriëntatie worden verwacht die niet overeenkomt met de realiteit. Vervolgens zullen ook de x- en y- waarde van de locatie van de robot niet meer kloppen. Na dit fenomeen vele malen te testen, zowel op papier als in de praktijk kwam naar voren dat deze afwijking niet in kaart te brengen is.

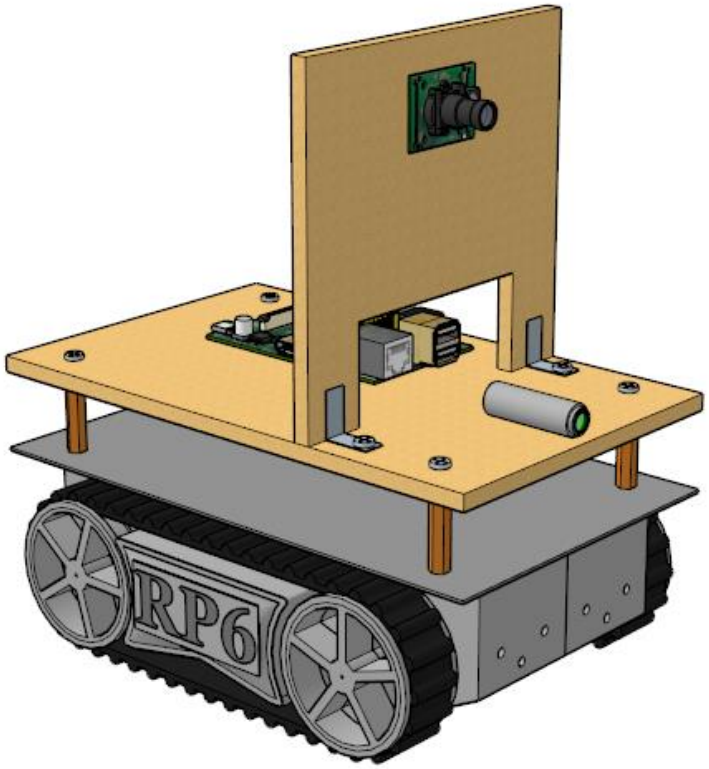



***Figuur 12.1*** *Een foto van de RobotLoader tool waarmee de code geüpload en gedebugged kon worden.*

# **De derde iteratie:** Navigatie

*19 mei 2014 t/m  30 mei 2014 (2 weken (+1))*

In de derde iteratie wordt onderzoek gedaan naar de wijze waarop optische triangulatie en de plaatsbepalings-technieken van de RP6 kunnen worden samengebracht om de RP6 zelfstandig te laten navigeren. Er wordt gekeken naar de stappen die nodig zijn om de door de optische triangulatie geleverde informatie om te zetten in concrete commando's voor de RP6. Daarnaast wordt er gekeken naar de voor- en nadelen van de verschillende interpretaties van de door de optische triangulatie geleverde informatie,  de beste manier om het werk te verdelen tussen de Raspberry Pi en de ATmega32 en een aantal praktische problemen waar de RP6 tegen aan zal lopen en hoe deze het beste kunnen worden benaderd. De derde iteratie is de laatste iteratie van het ontwikkelingsproces en bestaat alleen uit een onderzoeksfase. Het doel van deze iteratie is het verzamelen van kennis over de wijze waarop optische triangulatie kan worden ingezet om een systeem autonoom te laten navigeren.




***F. 1*** *Een model van het systeem*

| Aantal weken: | 2 | 1 |
|---|---|---|
| Activititeit: | Onderzoek | De laatste week |
| Communicatie | | |
| Navigatie | | |
| Opleveren afstudeerverslag | | |
| Demonstratie | | |

***Figuur P.3*** *De planning van de derde iteratie. Het onderzoek doen naar navigatie.*

# 13. Onderzoeksfase

*19 mei 2014 t/m 30 mei 2014 (2 weken)*

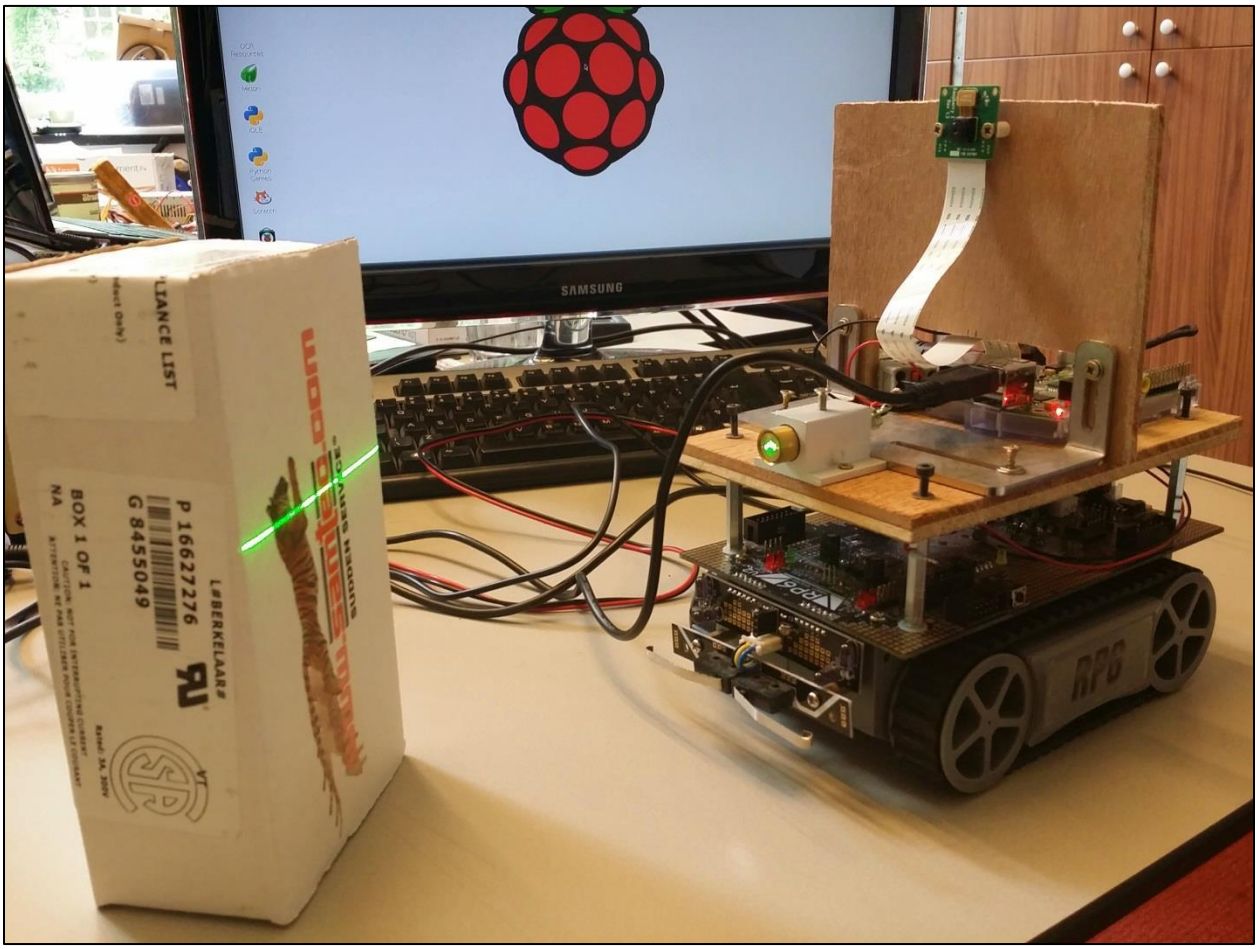
In de derde onderzoeksfase wordt onderzoek gedaan naar de wijze waarop de RP6 door middel van odometrie en optische triangulatie kan navigeren. Het is een onderzoek naar de combinatie van technieken uit de eerste en de tweede iteratie. In figuur 13.1 is het resultaat weergegeven van de combinatie van de systemen uit de eerste en tweede iteratie. De robot is functioneel maar niet in staat om erg vrij te bewegen vanwege de vele kabels waaraan hij is verbonden. Het onderzoek is met name gebaseerd op praktijktesten en eerder verzamelde informatie.

Het onderzoek zal vorm worden gegeven door antwoorden te zoeken op de volgende vragen:

*Hoe kunnen afstanden worden omgezet in commando's voor de RP6? (**13.1**)*

*Kan odometrie nog wel worden ingezet met de afwijkingen die ontstaan door rupsbanden? (**13.2**)*

*Waar licht de werkverdeling tussen de Raspberry Pi en de ATmega32 bij het bepalen van de koers? (**13.3**)*

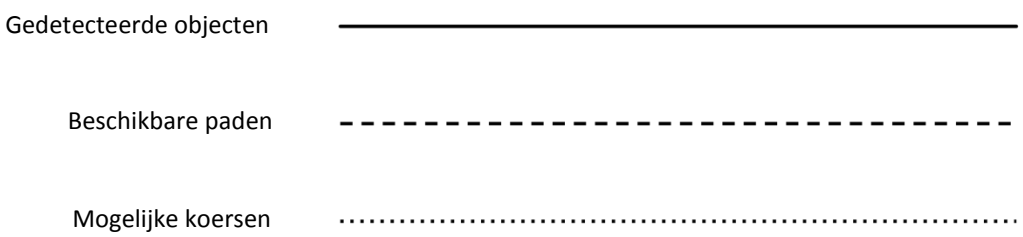



***Figuur 13.1** Een foto van de hardware-module waarin de RP6 en optische triangulatie zijn gecombineerd.*

# 13.1 Paden detecteren

Door middel van optische triangulatie kan de locatie van nabije objecten worden bepaald. Het systeem is zo ingesteld dat alleen de locaties van het meest vooraanstaande object in iedere kolom van pixels wordt berekend. Het resultaat is een soort dieptekaart. Omdat de locaties van alle vooraanstaande objecten bekend zijn, kan ook de locatie worden bepaald van plaatsen waar zich geen objecten bevinden. Deze informatie kan worden ingezet om een systeem autonoom te laten navigeren. De informatie kan echter op verschillende manieren worden benut.

Deze verschillende manieren kunnen het beste worden weergegeven door middel van een aantal afbeeldingen. In figuur 13.2 is een legenda weergegeven van de objecten die in de afbeeldingen voorkomen.

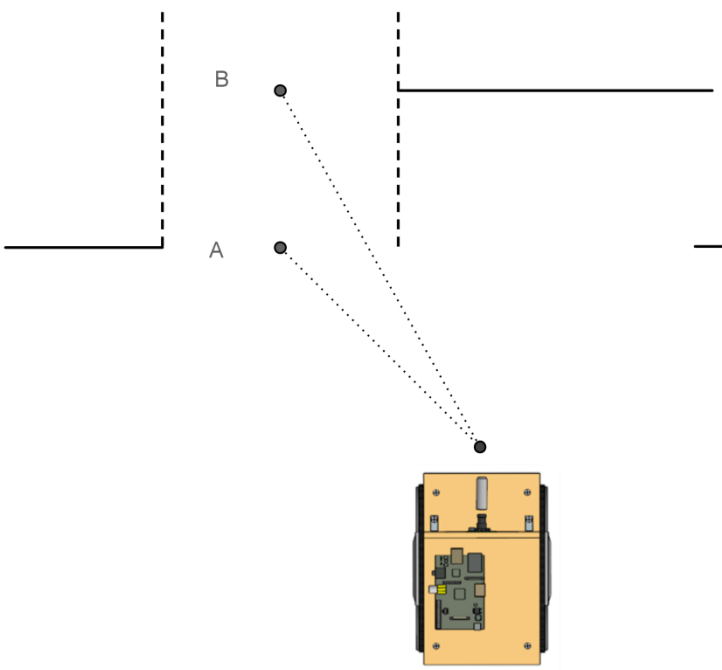



***Figuur 13.2*** *Een legenda van de objecten waar figuren 13.3, 13.4 en 13.5 uit bestaan.*

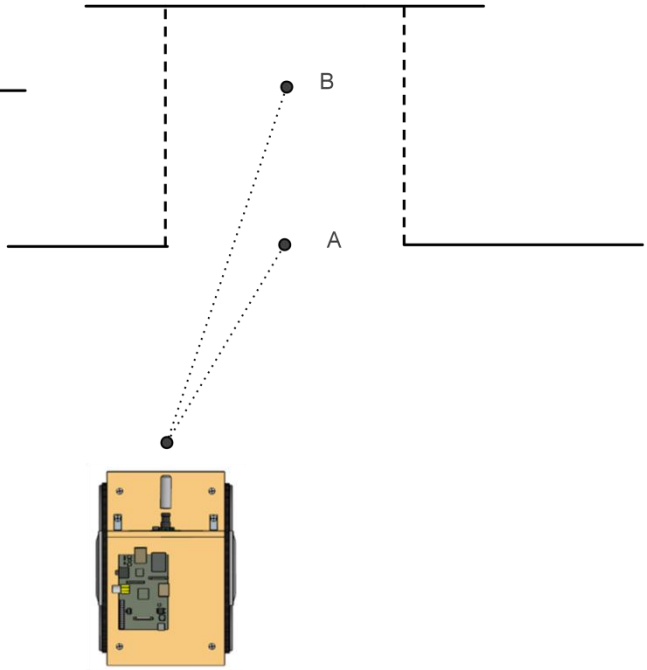
Aan de hand van de gedetecteerde objecten kunnen twee soorten paden worden waargenomen. Rechte paden en gebroken paden. In figuur 13.3 is een recht pad weergegeven. Een recht pad ontstaat wanneer er in een reeks aansluitende pixelkolommen geen objecten worden waargenomen. Als deze reeks samen breeder is dan de RP6 kan het worden gezien als een pad. Hoewel de objecten in figuur 13.3 zich niet op dezelfde afstand van de RP6 bevinden heeft dit geen invloed op bepalen van de locatie van het pad. De RP6 weet echter niet wat paden zijn. De RP6 weet alleen wat locaties en oriëntaties zijn. De meest eenvoudige manier om de RP6 te sturen is door hem een oriëntatie (koers) mee te geven. De RP6 zal vervolgens een van zijn twee wielen harder laten rijden dan het andere tot de gewenste oriëntatie is behaald. Voordat een koers mee gegeven kan worden moet deze eerst echter worden bepaald. De koers kan aan de hand van de locatie van het pad worden bepaald. De koers kan echter op verschillende manieren worden gedefinieerd.

In figuur 13.3 zijn twee verschillende opties weergegeven. Koers A leidt tot een punt ter hoogte van het meest dichtbij zijnde object. Koers B leidt tot een punt ter hoogte van het meest verre punt. Een voordeel van optie A is dat de RP6 kleine stappen neemt. Hierdoor is de RP6 in staat om snel te corrigeren wanneer paden wijzigen (door bijvoorbeeld omstanders). Het nadeel van optie A is dat de robot zich scheef voor het pad zal positioneren. De RP6 zal bijna niet meer in staat zijn om op door het pad te rijden vanwege zijn oriëntatie en de beperkte ruimte voor zich. Daarnaast zou de RP6 dicht tegen een muur aan kunnen staan waardoor zijn eigen optische triangulatie nutteloos wordt. Optie B is een betere optie. De RP6 zal minder scheef in het pad gepositioneerd worden. Niet alleen zal de RP6 hierdoor in staat zijn daadwerkelijk door het pad een te rijden, de RP6 zal ook in staat zijn om optische triangulatie uit te blijven voeren. Waar in optie A de optische triangulatie wordt belemmerd door te dicht bij de muur te komen is dat bij optie B geen probleem.

In figuur 13.4 is een pad met bochten weergegeven. Het pad bevat geen rechte doorgang maar een ruimte waar de hele RP6 in past en paden naar links en rechts. Het optische triangulatie systeem registreert echter alleen de eerste objecten in iedere pixelkolom. Hierdoor weet het systeem niet of er een doorgang naar links of rechts is.



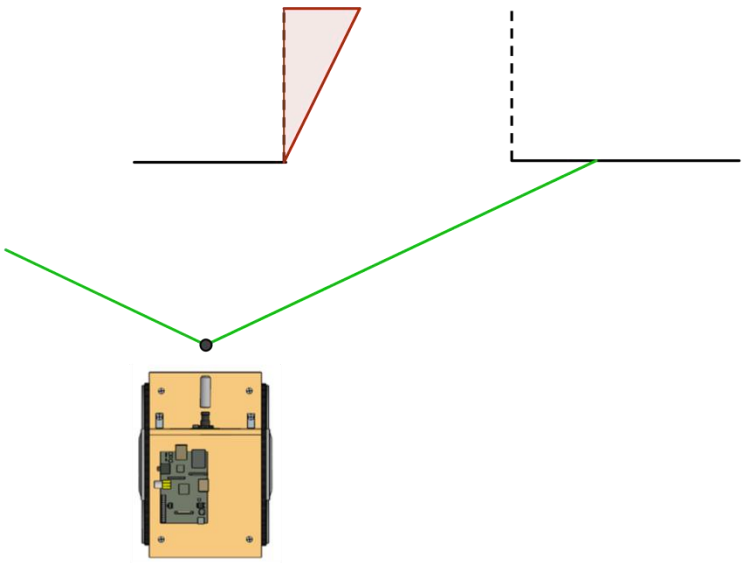

***Figuur 13.3*** *Een recht pad voor de RP6.*




***Figuur 13.4*** *Een gebroken pad voor de RP6.*

Toch zou deze informatie kunnen worden geleverd. De pixellocaties zijn bekend en de extra berekeningen zullen voor de processor van de microcontroller geen probleem zijn. Het probleem heeft meer te maken met de situatie van koers A uit figuur 13.3. Als de RP6 in figuur 13.4 koers B zou willen aanhouden zou de robot uiteindelijk op een punt uitkomen waar hij vanwege zijn afstand tot de muur geen optische triangulatie meer kan uitvoeren. In tegenstelling tot figuur 13.3 is koers A hier de beste keuze. Het systeem zal vanaf punt A nog in staat zijn om optische triangulatie uit te voeren en vanwege zijn scheve positie daadwerkelijk in staat zijn om mogelijke paden te detecteren. De wijze waarop een optimale koers kan worden gedefinieerd is dus afhankelijk van het soort pad.

Een ander probleem dat zich kan voordoen bij het detecteren van paden is schaduw van andere objecten. In figuur 13.5 is weergegeven dat de optische triangulatie een pad detecteert. Het is een recht stuk waar geen objecten in zijn waargenomen. Het in rood aangegeven valt echter buiten het bereik van de laser omdat het object ervoor het licht blokkeert. Toch zal er worden aangenomen de robot er kan rijden omdat er geen objecten zijn waargenomen. Dit probleem zal zich voordoen bij alle paden die zich niet direct in het midden van het beeld bevinden.



***Figuur 13.4*** *Het ontstaan van dode hoeken.*

## 13.2 Odometrie met rupsbanden

Odometrie is een techniek waarmee informatie van een robot met wielen kan worden afgeleid aan de hand van de rotaties van de wielen. Door middel van odometrie kan onder andere de snelheid, locatie en oriëntatie van een robot worden bepaald. Hoewel rupsbanden lijken op wielen en er odometrie mee kan worden uitgevoerd werken rupsbanden anders. Rupsbanden moeten namelijk slippen om te kunnen rijden. Elke keer dat de rupsbanden slippen zal odometrie foute informatie leveren. Odometrie gaat er namelijk vanuit dat de iedere rotatie verbonden is aan een bepaalde snelheid, verplaatsing van locatie en verandering van oriëntatie.

In paragraaf 13.1 wordt ingegaan hoe een robot op eenvoudige wijze aangestuurd kan worden om naar een bepaalde locatie te rijden. In figuur 13.1 wordt uitgegaan van een robot met wielen. Dead reckoning zou werken bij robots met rupsbanden zou de robot namelijk op een veel eenvoudigere wijze kunnen worden aangestuurd. Als dead reckoning zou werken zou de robot gewoon op zijn plek kunnen draaien en zo met haakse hoeken naar zijn doel toe rijden. Robots met wielen zijn niet in staat om op hun plek te draaien, hierdoor is de aansturing veel complexer. Dead reckoning werkt echter niet goed bij rupsbanden dus hoeveel er op de plek gedraaid kan worden, kan er niet worden bijgehouden hoe veel de robot al is gedraaid. Hoewel deze waardes zijn te kalibreren, is de afwijking sterk afhankelijk van factoren zoals onder andere de snelheid van de robot, het gewicht van de robot, het materiaal van de vloer. Door het uitvoeren van verschillende tests kwam naar voren dat de afwijking een even groot probleem vormt wanneer de robot draait door een wiel harder te laten rijden dan het andere als wanneer de robot op zijn plek draait. De theorie dat beide wielen even veel afwijking vertronen en de afwijking dus opheffen wanneer er op de plek wordt gedraaid was verre van correct.

De robot kan zonder odometrie echter nog steeds navigeren op de wijze die in paragraaf 13.1 is beschreven. Hoewel het systeem niet in staat is om door middel van odometrie zijn oriëntatie te bepalen kan hij uit gaan van de oriëntatie die door de optische triangulatie wordt berekend. De optische triangulatie zal namelijk continu een koers doorgegeven. Aan de hand van deze koers kan de robot achterhalen of hij zijn koers moet aanpassen of dat hij op de goede weg is. Dit is echter een hele riskante manier van navigeren omdat de robot dan volkomen afhankelijk is van de resultaten van de optische triangulatie.

Een goed alternatief is de robot te voorzien van een kompas. De meest essentiële informatie blijft namelijk de oriëntatie van de robot. Door gebruik te maken van een extern kompas zou de robot op de hoogte kunnen blijven van zijn eigen oriëntatie. Op deze wijze is hij veel minder afhankelijk van de informatie van optische triangulatie.

Hoewel er mogelijkheden zijn om odometrie en rupsbanden te combineren is er echter weinig aanleiding voor. De enige reden dat er gedurende dit ontwikkelingsproces gebruik van is gemaakt is omdat het de enige beschikbare robot was. De systemen waar de optische triangulatie voor bedoeld is maken gewoon gebruik van wielen.

## 13.3 Communicatie

Het optische triangulatie systeem uit de eerste iteratie levert een tweedimensionale array met locaties van objecten. De autonome robot uit de tweede iteratie verwacht een commando dat bestaat uit een bepaalde koers. Ergens moeten de afstanden worden omgezet in een bepaalde koers. Dit zou zowel kunnen gebeuren op de microcontroller van de optische triangulatie als de microcontroller van de autonome robot.

Het voordeel van het bepalen de koers op de microcontroller van de optische triangulatie (Raspberry Pi) is dat de Raspberry Pi aanzienlijk meer rekenkracht ter beschikking heeft dan de ATmega32. De Raspberry Pi heeft niet alleen een betere processor, maar maakt ook nog eens voornamelijk gebruik van zijn grafische processor voor het merendeel van zijn taken. Hierdoor kan hij zonder veel frequentie te verliezen de koersen berekenen. Daarnaast is het sterk aan te raden om deze koersen te berekenen op de Raspberry Pi omdat anders continu grote tweedimensionale arrays over een seriële verbinding naar de ATmega32 gestuurd zou moeten worden. Deze communicatie is zeer belastend voor de ATmega32 en deze microcontroller is daardoor niet meer in staat om zijn gewenste frequenties bij te houden. Het nadeel van de koersen berekenen op de Raspberry Pi is dat de Raspberry Pi niet weet waar de robot heen wil. Oftewel, als er meerdere doorgangen zijn zal de Raspberry Pi een keuze moeten gaan maken zonder dat hij weet waar de ATmega32 heen wil gaan. Dit zou verholpen kunnen worden door een tweezijdige communicatie op te zetten. De ATmega32 zou dan continu de Raspberry Pi op de hoogte moeten snellen van zijn gewenste locatie. Toch blijven deze problemen ondergeschikt aan de geringe processorkracht van de ATmega32. In deze situatie zal het daarom bijna wel op de Raspberry Pi moeten worden gedaan. Als de ATmega32 het aan zou kunnen om de berekeningen zelf te doen zou dat echter mooier zijn. Het optische triangulatie systeem zou daardoor minder afhankelijk worden van de systemen die er gebruik van maken.

# 14. Conclusie

Ingenieursbureau Berkelaar Meet- en Regeltechniek heeft de opdracht gegeven op een systeem te ontwikkelen waarmee robots autonoom nabije objecten kunnen lokaliseren. Optische triangulatie kan Berkelaar Meet- en Regeltechniek hierin voorzien. Er worden echter ook eisen gesteld aan de kosten van het systeem en deze kosten zijn zeer bepalend voor de kwaliteit van het systeem. Door middel van een Raspicam en een Raspberry Pi kunnen objecten tot op ongeveer één meter worden gelokaliseerd. Dit komt niet overeen met de door hun gestelde eisen. Daarnaast is de betrouwbaarheid nog eens sterk afhankelijk van de omgeving, met als uitgangspunt, hoe beter de kwaliteit van de hardware, des te minder de omgeving de kwaliteit zal verstoren. De kwaliteit en de prijs van hardware gaan echter de goede kant op en hiermee ook de kwaliteit van betaalbare optische triangulatie.

Tijdens het ontwikkelingsproces is ook veel tijd besteed aan het doen van onderzoek naar en het ontwikkelen van platformen waarop het systeem uiteindelijk moet gaan functioneren. Bij dit onderzoek zijn technieken naar voren gekomen die op deze platformen gehanteerd kunnen worden om het optische triangulatie systeem te ondersteunen. In theorie sluiten deze technieken perfect aan op de eisen die er aan worden gesteld. In de praktijk ontstonden echter een aantal problemen. Het meest voorname probleem ontstond door de incompatibiliteit van de techniek dead reckoning en de rupsbanden van het testplatform. De platformen waar het systeem voor bedoeld is hebben echter geen rupsbanden. Zonder dit te testen kan er helaas alleen maar gegist worden dat de technieken naar wens functioneren wanneer ze worden geïmplementeerd op een platform met wielen.

Als het systeem toch met rupsbanden zou moeten werken zijn er een aantal technieken die de effectiviteit toch kunnen vergroten. Door verschillende technieken te combineren kunnen de problemen die de rupsbanden veroorzaken worden gecontroleerd. Zo kan bijvoorbeeld een kompas de oriëntatie van het platform corrigeren. Door gebruik te maken van RFID-chips kan de robot zijn locatie corrigeren. Tot slot kan zelfs de optische triangulatie bijgestuurd worden door middel van infrarood sensoren. Geen van deze technieken is namelijk feilloos. In combinatie leveren ze verreweg de beste resultaten.

# 15. Referenties


[1] Math is Fun (z.d.). *Trigonometry Index*. http://www.mathsisfun.com/algebra/trigonometry-index.html Geraadpleegd in april 2014.

[2] K. Maxon (2001). *A Real-time Laser Range Finding Vision System*. http://www.seattlerobotics.org/encoder/200110/vision.htm Geraadpleegd in februari 2014.

[3] Broadcom (2013). *RaspiCam Documentation*. http://www.raspberrypi.org/wp-content/uploads/2013/07/RaspiCam-Documentation.pdf Geraadpleegd in februari 2014.

[4] Dafydd Walters (2000). *Implementing Dead Reckoning by Odometry on a Robot with R/C Servo Differential Drive*. http://www.seattlerobotics.org/encoder/200010/dead_reckoning_article.html Geraadpleegd in april 2014.

[5] Ana Huamán (z.d.). *OpenCV Tutorials*. http://docs.opencv.org/doc/tutorials/tutorials.html Geraadpleegd in februari 2014.

[6] Z. Fan, J. Borenstein, D. Wehe, Y. Koren (1994). *Experimental Evaluation of an Encoder Trailer for Dead-reckoning in Tracked Mobile Robots.* http://deepblue.lib.umich.edu/bitstream/handle/2027.42/4835/bac6476.0001.001.pdf?sequence=5 Geraadpleegd op april 2014.

[7] D. Endo, Y. Okada, K. Nagatani, K. Yoshida. (2007). *Path Following Control for Tracked Vehicles Based on Slip-Compensating Odometry.* http://www.researchgate.net/publication/224296486_Path_following_control_for_tracked_vehicles_based_on_slip-compensating_odometry Geraadpleegd in april 2014.

[8] P. Raufast (2013). *OpenCV and Pi Camera Board*. http://thinkrpi.wordpress.com/2013/05/22/opencv-and-camera-board-csi/ Geraadpleegd op februari 2014.

# 16. Evaluatie

Na zeventiek weken is de afstudeerperiode tot zijn einde gekomen. De afstudeeropdracht heeft geresulteerd in twee prototype-systemen en een afstudeerverslag. Het eerste prototype is een optische triangulatie systeem. Het systeem is in staat om door middel van een (betaalbare) microcontroller, camera-module en lijnlaser de locatie van nabije objecten te bepalen. Het systeem levert de positie's in de vorm van coördinaten in een twee-dimensionaal assenstelsel. Hierin voldoet het aan de eisen van Berkelaar Meet- en Regeltechniek. Het systeem heeft echter maar een (betrouwbaar) bereik tot op één meter. Hierin voldoet het niet aan de eisen (drie meter) van Berkelaar Meet- en Regeltechniek. Dit (ontoereikende) bereik wordt met name bepaald door de kwaliteit van de hardware. Het onvermogen om het licht van de laser te herkennen en het onvermogen om onderscheid te kunnen maken tussen licht uit de omgeving en licht van de lijnlaser zijn voornamelijk toe te kennen aan de kwaliteit van de camera-module. Desondanks er niet aan alle niet-functionele eisen wordt voldaan, heeft het systeem zijn primaire functionele eis (het bepalen van de locatie van nabije objecten) behaald.

Het tweede systeem is een driver voor een autonome robot. De driver is ontwikkeld om technieken te beproeven die naast optische triangulatie nodig zijn om autonome navigatie te kunnen realiseren. In de onderzoeksfase uit de tweede iteratie kwam naar voren dat dead reckoning in staat zou moeten zijn om genoeg informatie te verschaffen om autonome navigatie mogelijk te maken. In de praktijk was dit verre van waar. Dit lag echter niet aan de techniek, maar aan het testplaform. De op wiel-encoders gebaseerde dead reckoning is namelijk bedoeld voor voertuigen met wielen. De robot (RP6) die in de tweede iteratie diende als testplatform heeft rupsbanden. Door de wijze waarop rupsbanden functioneren is het nagenoeg onmogelijk om betrouwbare dead reckoning te realiseren. Dit vormt echter geen probleem. De RP6 is namelijk niet meer dan een testplatform dat bij aanvang van de tweede iteratie bij toeval beschikbaar was. De robots waar de systemen voor bedoeld zijn maken gewoon gebruik van wielen. Om deze reden is de tweede iteratie (ondanks het testplatform) gewoon doorgezet. De vorm en frequentie van de door dead reckoning geleverde informatie bleken uiteindelijk ideaal te zijn voor navigatie.

De prototype-systemen zijn ontwikkeld door middel van een iteratieve aanpak. Deze aanpak maakte geen gebruik van softwareontwikkelmethodes zoals 'Rational Unified Proces' en 'Scrum'. De voornaamste reden hiervoor is dat deze ontwikkelmethode gericht zijn op pure softwareontwikkeling. Bij processen waar onderzoek centraal staat bieden ze minder ondersteuning. Het voordeel van een eigen aanpak is dat er veel ruimte is om het proces zelf in te richten. Het risico is echter ook te veel vrijheid. Tijdens het proces stond het onderzoek namelijk op de eerste plaatst en kwam het ontwerp en de implementatie er altijd achteraan. Dit kan onwenselijke situaties opleveren waarin terug de terugkoppeling naar de klant te wensen over laat of complicaties bij de implementatie pas laat aan het licht komen (rupsbanden). Een strakkere aanpak met meer 'milestones' en tussen-meetpunten of een vaste softwareontwikkelmethode zouden meer vastheid en garantie tijdens het ontwikkelproces kunnnen bieden.

Een andere reden om voor de gekozen aanpak te gaan was de aanvankelijke kennis van het probleemdomein. Het was vóór de uitvoer van de opdracht moeilijk om alle iteraties in te kunnen plannen. Om deze reden moesten er tijdens het ontwikkelingsproces een aantal belangrijke keuzes gemaakt worden. Een van deze keuzes was om in de tweede iteratie de scope te verbreden in plaats van te verdiepen. Het verdiepen zou het bereik en de betrouwbaarheid van de lokalisatie moeten verbeteren. Toch is er gekozen om de scope te verbreden omdat het bereik en de betrouwbaarheid grotendeels worden bepaald door de kwaliteit van de hardware. Daarnaast was veel leerzamer om onderzoek te doen naar de nieuwe onderwerpen in plaats van de overige iteraties te besteeden aan een enkele techniek die het bereik en de betrouwbaarheid op minimale wijze kunnen verbeteren. Het resultaat bestaat hierdoor echter wel uit onderzoek en prototype's in plaats van één operationeel eindproduct.

# 17. Bewijsvoering

**A1.** Analyseren van het probleemdomein (**niveau 3**)

Bij aanvang van de afstudeerperiode luidde de opdracht: “het ontwikkelen van een systeem dat nabije objecten kan lokaliseren”. Door het probleemdomein te analyseren heeft deze opdracht veel meer vorm gekregen. Het merendeel van de analyse van het probleemdomein heeft plaatsgevonden tijdens het opstellen van het plan van aanpak. Al gauw kwam naar voren hoe, waar en waarom het systeem uiteindelijk moet gaan werken. Dit heeft een grote invloed gehad op de invulling van de tweede en derde iteratie. De tweede en derde iteratie zijn namelijk besteed aan het onderzoek doen naar en het ontwikkelen van het platform waarop het lokalisatiesysteem uiteindelijk moet gaan werken.

**A5.** Opstellen van systeemeisen (requirements) (**niveau 3**)

Tijdens de afstudeerperiode zijn er op drie verschillende momenten systeemeisen verzameld. Tijdens het opstellen van het plan van aanpak zijn voor het eerst de abstracte functionele en in mindere mate niet-functionele eisen van het systeem achterhaald door middel van een gesprek met de medewerkers van Berkelaar Meet- en Regeltechniek. Deze systeemeisen hebben vervolgens bijgedragen aan het selecteren van een aanpak en invullen van de eerste iteratie. Tijdens de ontwerpfase in de eerste iteratie worden de systeemeisen van het optische triangulatiesysteem bepaald. Door eerst uitgebreid onderzoek te doen naar de mogelijkheden van de technieken in de onderzoeksfase konden er door middel van een aantal gesprekken met de medewerkers van Berkelaar Meet- en Regeltechniek functionele en niet-functionele worden geformuleerd. Tijdens de ontwerpfase in de tweede iteratie zijn op dezelfde wijze de systeemeisen voor de robot bepaald.

**C8.** Ontwerpen van een technisch informatie systeem (**niveau 3**)

Zowel het optische triangulatie systeem als de autonome robot zijn in twee stappen ontworpen. De eerste stap vindt plaats in de onderzoeksfasen. In de onderzoeksfasen wordt naast het doen van onderzoek een prototypesysteem ontworpen en ontwikkeld om de verschillende onderzochte technieken te beproeven. Zodra de eigenschappen van de technieken duidelijk zijn wordt er in de ontwerpfasen een concreet ontwerp samengesteld. De kennis die is vergaard in de onderzoeksfasen wordt gecombineerd met de systeemeisen die zijn samengesteld met de werknemers van Berkelaar Meet- en Regeltechniek. Het resultaat is een ontwerp, weergegeven door middel van een modelleertaal (UML). Dit ontwerp vormt het uitgangspunt gedurende de uitvoer van de opdracht maar zal nog waar nodig worden aangepast tijdens de implementatie- en testfasen.

**C13.** Het betrekken van real-time aspecten bij een ontwerp (**niveau 3**)

Real-time aspecten spelen een grote rol bij zowel het optische triangulatie systeem als de autonom robot. Bij het optische triangulatie systeem is de frequentie waarmee de gehele lokalisatieroutine kan worden uitgevoerd essentieel. Een te lage snelheid en het systeem zou de waargenomen obstakels al weer gepaseerd kunnen zijn of nieuwe obstakels niet op tijd kunnen waarnemen. Om dit te kunnen controleren is er veel getest en onderzoek gedaan naar de eigenschappen van de hardware (snelheden) en de verschillende technieken die toegepast kunnen worden om het licht van de laser te kunnen herkennen. De eerste tests vonden al plaats in de onderzoeksfase en de laatste in de testfase. Bij het ontwikkelen van de autonome robot

speelde real-time aspecten een nog grotere rol. Bij het bepalen van de snelheid, locatie en oriëntatie van de robot gaat het om microseconden. Bij het ontwerp van de driver zijn verschillende technieken om deze variabelen te bepalen getest. Als er niet met de allergrootste precisie de verplaatsing van de robot kan worden bepaald zal er over een korte tijd een grote afwijking accumuleren. Het is daarom essentieel dat zowel de frequentie waarmee de variabelen worden bepaald als de nauwkeurigheid ervan zo hoog mogelijk is.

**D16.** Het realiseren van software (**niveau 3**)

Hoewel iteraties aparte implementatiefasen hebben wordt er in de onderzoeksfasen al software gerealiseerd. Het verschil tussen de onderzoeksfasen en de implementatiefasen is dat er in de onderzoeksfase prototypes worden ontwikkeld om de verschillende onderzochte technieken te beproeven. Door zo vroeg in het ontwikkelingsproces al delen van de software te ontwikkelen wordt er in de latere implementatiefase veel werk en risico bespaard. Hoewel alle software ontworpen is om te werken op embedded systemen, is het merendeel van de software eerst voor een laptop ontwikkeld. De reden hiervoor is dat het bewerken en compileren van code op embedded systemen (Raspberry Pi) vaak veel meer moeite en tijd kost doordat de embedded systemen over minder processorkracht en geheugen beschikken. De embedded systemen werden er bij betrokken na het ontwikkelen van onderdelen om de prestaties en compatibiliteit te kunnen testen.

# 18. Reflectie

Na zeventien weken is mijn afstudeerperiode tot een eind gekomen. Tijdens deze periode heb ik voor ingenieursbureau Berkelaar Meet- en Regeltechniek een systeem ontwikkeld dat door middel van optische triangulatie nabije objecten kan lokaliseren. Tot het ontwikkelingsproces van het systeem behoren onder andere: het doen van onderzoek, het ontwerpen van een technisch informatie systeem, het realiseren van software, het testen van software, het communiceren met de klant, het documenteren van de bevindingen en het organiseren van de uitvoer. Hoewel deze competenties allemaal meerdere keren aan bod zijn gekomen gedurende mijn studie, kan een nieuwe situatie een wereld van verschil maken. Het verloop van sommige van deze aspecten overtroffen mijn verwachtingen en andere aspecten konden achteraf beter op een andere wijze worden aangepakt.

Hoewel het doen van onderzoek de meeste tijd heeft gekost heeft het ook het meeste opgeleverd. Door middel van programmatuur zoals Google Sketchup en Geogebra ben ik in staat geweest om op hele visuele wijze de eigenschappen van de verschillende triangulatie- en navigatietechnieken toe te lichten. Ik beschouw het documenteren van onderzoek dan ook als een van de meest positieve competenties uit het ontwikkelingsproces.

Naast het uitvoeren en documenteren van onderzoek verliep het omzetten van theorie naar functionerende code uitstekend. Ik was goed in staat om de door mij bedachte technieken om te zetten in code door middel van de computer vision bibliotheek OpenCV. Mijn eerdere ervaringen met OpenCV kwamen hier goed te pas. Een nadeel van OpenCV was dat het niet kon samenwerken met de standaard drivers van de camera van het embedded systeem. Een workaround hiervoor was echter ook al gauw gevonden en de implementatie verliep soepel.

Bij het realiseren van de code voor de autonome robot was mijn gebrek aan wiskundige kennis helaas een probleem. Bij het optische triangulatie systeem heb ik gebruik kunnen maken van een aanzienlijk aantal functionaliteiten uit een bibliotheek (OpenCV). Bij de implementatie van de code voor de autonome robot heb ik de algoritmes zelf moeten ontwikkelen. De bestaande algoritmes bestonden namelijk grotendeels uit wiskundige formules die mijn kennis te boven gingen. Hoewel mijn eigen implementaties uiteindelijk prima functioneren had ik veel tijd kunnen besparen als ik niet opnieuw “het wiel had moeten uitvinden”.

Een aspect dat ik achteraf op een andere wijze had aangepakt is de aanpak van het ontwikkelingsproces. Omdat het merendeel van het ontwikkelingsproces heeft bestaan uit het doen van onderzoek heb ik ervoor gekozen om niet een standaard ontwikkelmethode zoals ‘RUP’ of ‘Scrum’ te hanteren. Een door mijzelf samengestelde methode zou meer vrijheid bieden om de aanpak zo goed mogelijk aan te laten sluiten aan de aard van de opdracht. Het risicio en uiteindelijk het nadeel hiervan was dat ik mijzelf te veel vrijheid heb gegeven. Waar standaard ontwikkelmethode gebruik maken van een grote hoeveelheid tussenproducten en ‘ceremonie’, was dat bij mijn aanpak aanzienlijk minder. De inzichtelijkheid van het ontwikkelingsproces is er niet vooruit op gegaan.

De communicatie met de opdrachtgever had ook beter kunnen voorlopen door gebruik te maken van een standaardontwikkelmethode of aanpak met meer tussenproducten en ‘ceremonie’. Het aanhouden van officiële deadlines voor onder andere tussenproducten en ‘milestones’ had het ontwikkelingsproces aanzienlijk kunnen verbeteren. Door deze aspecten op een informele wijze in te vullen ontstaat veel eerder het risico dat tussenproducten verlaat worden of dat milestones vooruit geschoven worden.

Uiteindelijk was de meest waardevolle en duurste les die ik geleerd heb dat online versiebeheer een ‘must’ is. Hoewel dit een pijnlijke les was zal ik het nooit meer vergeten. Desondanks deze lessen en leerpunten was het een hele interessante opdracht en heb ik gedurende deze zeventien weken heel veel geleerd en ervaren.