

## PROFIELWERKSTUK

---

# Het bouwen van een autonome robot

---

*Auteurs:*

Erik KOOISTRA  
Koen WOLTERS

*Begeleider:*

Stefan CARREE

22 januari 2015



Dit werk valt onder een Creative Commons Naamsvermelding-NietCommercieel-GelijkDelen 4.0 Internationaal-licentie. Ga naar <http://creativecommons.org/licenses/by-nc-sa/4.0/> om een kopie van de licentie te kunnen lezen.



## Samenvatting

Wij zijn al enkele jaren geïnteresseerd in robotica. Voor dit profielwerkstuk besloten we daarom om zelf een robot te bouwen. De voornaamste eisen waaraan onze robot moest voldoen waren de volgende:

- Zelfgebouwd: de robot moet zo ver mogelijk zelfgebouwd zijn
- Modulair: de robot moet modulair worden opgezet, zodat er makkelijk onderdelen toegevoegd en verwijderd kunnen worden.
- Betaalbaar: de robot moet binnen een budget van 200 euro blijven.
- Leerzaam: we wilden een uitdaging hebben aan het bouwen van de robot

Uiteindelijk kozen we ervoor om een ‘gewone’ rijdende robot te bouwen. Dit type robot is namelijk relatief eenvoudig zelf te bouwen, is daarnaast betaalbaar, maar kan wel zover je wilt worden uitgebreid. Wij wilden graag dat de robot uiteindelijk zelfstandig kon opereren, maar daarvoor was het noodzakelijk dat de robot nauwkeurig kon bewegen. Wij besloten daarom een robot te ontwerpen die een kaart kon maken van de omgeving door obstakels te vinden en met behulp van die kaart objecten later kon terugvinden.

Wij hebben uitgebreid literatuuronderzoek gedaan naar de verschillende mogelijkheden aan sensoren, actuatoren en controllers. Uit dit onderzoek hebben we enkele onderdelen geselecteerd die wij nodig dachten te hebben om aan de eisen te voldoen. Belangrijk was daarbij dat de prijs binnen de perken bleef.

Vervolgens hebben wij een prototype in elkaar gezet en zijn begonnen we met het programmeren van de robot. Al snel bleek dat we heel veel grote en kleine problemen tegenkwamen bij het aansturen van de robot. Zo waren er bijvoorbeeld enkele ontwerpfouten in het prototype en waren er daarnaast problemen met de elektronica. Ook het communiceren met de gekochte onderdelen bleek nog niet eenvoudig. Het kostte ons daarom veel tijd om de robot stabiel te krijgen en de besturing van de onderdelen goed te laten werken.

Nadat we eindelijk zover waren, was er nog maar zeer korte tijd om de onderdelen via de software te combineren. Uiteindelijk zijn we er in geslaagd om de robot op afstand via WiFi te laten rijden en de sensordata via een grafische interface uit te lezen. Uiteraard was dit onvoldoende om het door ons gestelde probleem op te lossen.

Direct gezien is het project dus niet geslaagd, voornamelijk door gebrek aan tijd. Wij zijn desondanks tevreden, omdat wij er veel van hebben geleerd en omdat er vooruitgang was in het project. In de toekomst hopen wij de doelstelling alsnog te halen en wij zijn ervan overtuigd dat wij dit met meer tijd ook zeker kunnen.



# Inhoudsopgave

<b>I Inleiding</b>	<b>7</b>
1 Probleemstelling	8
2 Doelstelling	8
3 Probleemanalyse	8
4 Indeling	9
<b>II Programma van eisen</b>	<b>10</b>
5 Algemeen	10
6 Robot	11
7 Software	13
<b>III Ontwerpvoorstel en prototype</b>	<b>14</b>
8 Ontwerpvoorstel	14
9 Prototype	18
10 Software	18
<b>IV Problemen en oplossingen</b>	<b>19</b>
11 Bouw	19
12 Elektronica	19
13 Software	20
<b>V Evaluatie</b>	<b>22</b>
14 Eisen	22
14.1 Algemeen . . . . .	22
14.2 Robot . . . . .	22
14.3 Software . . . . .	24
15 Conclusie	24
16 Persoonlijk	24

<b>VI Verantwoording</b>	<b>25</b>
<b>17 Logboek</b>	<b>25</b>
<b>VII Bijlage: Theoretische achtergrond</b>	<b>32</b>
<b>A Onderdelen</b>	<b>32</b>
A.1 Hardware . . . . .	32
A.1.1 Microcontrollers . . . . .	32
A.1.2 Motoren . . . . .	35
A.1.3 Motorcontrollers . . . . .	37
A.1.4 Wielen . . . . .	37
A.1.5 Communicatie . . . . .	39
A.2 Sensoren . . . . .	40
A.2.1 Terugkoppeling . . . . .	40
A.2.2 Anti-botsingssystemen . . . . .	41
A.2.3 Plaatsbepaling . . . . .	41
A.3 Software . . . . .	42
A.3.1 Besturingssysteem . . . . .	42
<b>B Motorkracht en -snelheid</b>	<b>43</b>
<b>VIII Bijlage: Software</b>	<b>45</b>
<b>C Protocol</b>	<b>45</b>
C.1 Algemeen . . . . .	45
C.2 Beweging . . . . .	45
C.3 Sensoren . . . . .	46
<b>D GUI</b>	<b>46</b>
<b>E Code</b>	<b>46</b>
<b>IX Bijlage: Overigen</b>	<b>47</b>
<b>F Opzet project</b>	<b>47</b>
F.1 L <sup>A</sup> T <sub>E</sub> X . . . . .	47
F.2 GIT . . . . .	47
<b>G Schetsen</b>	<b>48</b>
<b>H Foto's</b>	<b>54</b>
<b>I Rekening materialen</b>	<b>55</b>
<b>J Bibliografie</b>	<b>56</b>

# Deel I

## Inleiding

Robots zijn de laatste paar jaren aan een opmars bezig. Op allerlei plekken vervangen zij taken die eerst door mensen werden uitgevoerd. Robots kunnen dingen doen die nooit eerder voor mogelijk werden gehouden. Ze kunnen bijvoorbeeld steeds meer onafhankelijk doen, maar ook opdrachten in het algemeen veel beter uitvoeren.

Wij zijn al enkele jaren erg geïnteresseerd in vakken als informatica, natuurkunde en robotica. Wij waren er dan ook al vrij snel uit dat we iets wilden doen met robots. Aangezien onze interesse voor robots vrij divers was, maar wij tegelijkertijd ook beginners waren op dit vlak, hebben we gekozen om een vrij simpele robot te maken.

Het zelfstandig bouwen van zowel de hardware als de software was voor ons erg belangrijk. Dat dit niet geheel mogelijk zou zijn was eigenlijk al vrij snel duidelijk, maar we wilden alle losse onderdelen wel zo simpel mogelijk houden. Door de individuele onderdelen simpel te houden, konden we door het koppelen van verschillende onderdelen toch een complexe robot maken.

Een tweede belangrijk eis was om alle onderdelen op software- en hardwarematig gebied zo modulair mogelijk op te zetten. Dit bood verschillende voordelen. Ten eerste was het daardoor eenvoudiger om later onderdelen te vervangen of aan te passen, als zou blijken dat ze niet aan onze eisen voldeden. Daarnaast zou het dan ook veel eenvoudiger zijn om later nieuwe onderdelen toe te voegen als we de robot lastigere taken wilden laten uitvoeren. Als laatste was dit ook voordelig voor de samenwerking, omdat we allebei aan losse modules konden werken.

Door de robot simpel te houden was het ook mogelijk om het budget beperkt te houden. Dit was noodzakelijk, omdat wij alle onderdelen zelf moesten betalen. We besloten daarom om maximaal 200 euro te gaan besteden, wat er automatisch voor zou moeten zorgen dat de robot niet te ingewikkeld zou worden.

Een robot die aan deze eisen voldeed en daarnaast mogelijkheid zou bieden voor latere uitbreiding was een ‘standaard’ rijdende robot. Aangezien er wel voldoende uitdaging moest zijn besloten we dat de robot autonoom moest gaan bewegen en taken uitvoeren. Het belangrijkste eerste doel was om een robot te maken die nauwkeurig kon voortbewegen, obstakels kon detecteren en daarmee een kaart van de omgeving te maken. Met behulp van deze kaart zou de robot dan autonoom taken moeten gaan uitvoeren.



**Figuur 1:** Een speelgoed robot

# 1 Probleemstelling

Het zelfstandig bouwen van een robot die zich op een autonome manier door een ruimte kan voortbewegen.

## 2 Doelstelling

In ons profielwerkstuk willen wij een grotendeels autonome robot ontwerpen. Dit is een proces dat bestaat uit verschillende onderdelen. Wij wilden al deze onderdelen van dit proces zoveel mogelijk zelfstandig uitvoeren. De robot en de software moeten modulair worden opgezet zodat wij makkelijk de mogelijkheden van de robot kunnen aanpassen.

Ons einddoel is een robot die ten minste twee basale dingen moet kunnen. Als eerste moet de robot zelfstandig kunnen voortbewegen door de ruimte en hierbij obstakels opslaan op een kaart. Daarnaast moet het mogelijk zijn om later met behulp van deze kaart terug te kunnen rijden naar het obstakel.

## 3 Probleemanalyse

De eerste stap van onze probleemstelling is het maken van een robot die kan bewegen. Voor dat bewegen zijn natuurlijk motoren nodig. Er zijn verschillende manieren om een robot te laten bewegen. We hebben echter vastgesteld dat de robot autonoom moet werken en aangezien dat niet eenvoudig te maken is, leek het ons handig om de manier van bewegen zo simpel mogelijk te houden. Daarom hebben we er dan ook voor gekozen om de robot enkel over relatief vlak land te laten bewegen.

Daarnaast moet de robot zelfstandig kunnen voortbewegen en zijn omgeving in een kaart kunnen opslaan. Dit onderdeel van de doelstelling vraagt om twee belangrijke dingen: ten eerste een onderdeel dat het autonome ‘denken’ kan regelen en daarnaast verschillende sensoren die de plaats moeten kunnen bepalen. Voor dat autonome denken hebben we dus een simpele computer nodig, oftewel een ‘microcontroller’. Voor het bepalen van de plaats is grote nauwkeurigheid vereist. Afwijkingen in de bepaling van de plaats kunnen op latere momenten namelijk grote gevolgen hebben. Als de robot bijvoorbeeld na het opslaan van een locatie tien meter verder rijdt, terwijl hij denkt vijftig meter te hebben gereden, dan gaat hij de locatie natuurlijk nooit meer terugvinden. Daarom is het belangrijk dat er goede sensoren komen voor de plaatsbepaling.

Voor het ontvangen van de taken (om bijvoorbeeld naar een bepaalde locatie te rijden) moet de robot in staat zijn om commando’s te ontvangen en is er dus een manier nodig om te communiceren met de robot. Aangezien de robot beweegt moet het ook mogelijk zijn om deze communicatie binnen een bepaald bereik in stand te kunnen houden. De snelheid van de verbinding is iets minder van belang, omdat we de robot toch voornamelijk vrij willen laten opereren.

Als laatste moet de robot in staat zijn om verschillende obstakels te detecteren, anders is het niet mogelijk om een kaart te maken. Onder obstakels kunnen verschillende dingen worden verstaan, maar wij hebben gekozen voor een duidelijk waarneembaar on-

doorzichtig object. Dit soort obstakels zijn namelijk relatief eenvoudig te detecteren door een sensor.

De basale benodigheden zijn dus: een microcontroller, sensoren voor de plaatsbepaling, een module voor de communicatie en een sensor voor het detecteren van objecten. Deze benodigheden (en de eisen waaraan ze moeten voldoen) kunnen we nu verder uitwerken in het plan van eisen. Hier stellen we nauwkeurig vast waar de robot aan moet kunnen voldoen om de probleemstelling op te lossen.

## 4 Indeling

Het bouwen van een robot gaat in een hele ketting van stappen. Als eerste heb je een idee, dat idee werk je voorzichtig uit en vervolgens blijkt dat idee helemaal niet zo goed te zijn, waarbij je het idee weer aanpast. Deze voortdurende ring van stappen is lastig in een verslag weer te geven. Wij hebben er voor gekozen om in dit verslag dan ook voornamelijk globaal in te gaan op het proces van idee naar uiteindelijke robot.

Het idee is hierboven al vastgesteld, maar zal in de komende paragraaf schematisch worden vastgelegd in een rijtje van eisen (zie pagina 10), die vastleggen wat onze robot moet kunnen, om ons idee te kunnen uitvoeren. Hierna gaan wij verschillende onderdelen benoemen die we kunnen gebruiken om de eisen uit te voeren. Deze eisen hebben we vervolgens omgezet in een prototype (zie pagina 14). In dit prototype kwamen we diverse problemen tegen. Wij gaan kort op enkele problemen in en leggen uit hoe wij deze problemen hebben aangepakt en hebben opgelost (zie pagina 19). Uiteindelijk evalueren we ons uiteindelijke product en stellen vast in hoeverre het voldoet aan de eisen. In dit onderdeel bekijken we dan ook in hoeverre ons product geslaagd is (zie pagina 22).

Voor ons onderzoek was natuurlijk ook onderzoek nodig naar de verschillende onderdelen. Een uitwerking van dit onderzoek is terug te vinden in de eerste appendix, de theoretische achtergrond. Naast hardware is software natuurlijk ook een erg belangrijk onderdeel van ons project en in de tweede appendix sluiten we dan ook de code bij en uitleg over enkele programma-onderdelen. Daarnaast zijn er nog enkele andere losse bijlagen toegevoegd, waar we onder andere in uitleggen waarom dit verslag gemaakt is in L<sup>A</sup>T<sub>E</sub>X (zie pagina 47).

# Deel II

# Programma van eisen

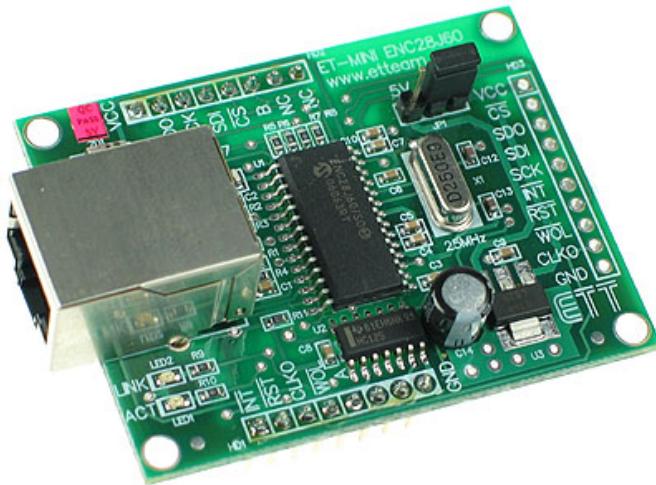
Uit de probleemanalyse kwamen de basale benodigdheden al naar voren (zie pagina 8). Om te bepalen welke uiteindelijke onderdelen we nodig hadden, hebben we een lijst gemaakt van eisen waar het project aan moest voldoen. Met behulp van de gekozen onderdelen kunnen we vervolgens een prototype maken (zie pagina 14).

## 5 Algemeen

**Zelfgebouwd** De robot moet zo veel mogelijk zelfgebouwd zijn. Het in elkaar zetten van de robot moet door ons zelf gebeuren. Alle onderdelen die (met onze kennis) binnen redelijke tijd te bouwen zijn, willen we zelf bouwen.

**Gebaseerd op elektronica platform** De aansturing van de robot gaat met een microcontroller en een bijbehorend platform. Het zelf maken van zo'n controller en het platform kost teveel tijd en daarom wordt er gebruik gemaakt van een al bestaand elektronisch platform. Het platform moet minstens aan de volgende eisen voldoen:

1. Het project moet open zijn, zodat we het naar wens kunnen aanpassen.
2. De controller kan in ieder geval werken met de spanningen 0 tot 5 volt.
3. De controller moet data kunnen ontvangen en verwerken van alle benodigde sensoren (zie verderop).
4. De controller moet alle benodigde actuatoren (toestellen die een actie kunnen uitvoeren) aan kunnen sturen (zie verderop).
5. De processor moet voldoende snelheid hebben om de benodigde beslissingen binnen redelijke tijd te kunnen maken.
6. De controller moet voldoende vaste opslag bieden om de te software op te slaan.
7. De controller moet genoeg variabele opslag bieden, zodat we de benodigde data kunnen opslaan.
8. Het platform moet zijn energie kunnen halen uit batterijen (zie pagina 12).



**Figuur 2:** Voorbeeld microcontroller

9. Het platform moet mogelijkheden ondersteunen voor zowel bedrade als draadloze communicatie (eventueel door extensies) met een computer.

Hier valt op te merken dat eisen 5 t/m 7 deels door een aparte computer kunnen worden uitgevoerd, als deze in verbinding staat met de robot (zie eis 9).

Daarnaast is het platform bij voorkeur simpel in gebruik en biedt het standaardbibliotheeken voor de aansturing. Als laatste zou het mooi zijn als het platform makkelijk mogelijkheden biedt tot uitbreiding.

**Modulair** De robot moet uit een kern bestaan waar losse sensor- en actuatormodules vrij makkelijk aan moeten kunnen worden bevestigd, hierdoor moet de robot relatief makkelijk tussen verschillende typen actuatoren en sensoren kunnen wisselen. Voor elke losse module geld dat deze specifiek gericht moet zijn op een bepaalde taak. De software moet eveneens modulair worden opgebouwd (zie pagina 13).

**Betaalbaar** Het hele project gaat geld kosten, omdat we aan onderdelen moeten komen. Aangezien we een eenvoudige robot bouwen en we zelf de onderdelen moeten betalen, moet het project betaalbaar blijven. Het budget voor alle onderdelen is dan ook maximaal tweehonderd euro.

**Leerzaam** Dit project is bedoeld om een werkende robot te leveren, maar het proces moet wel voornamelijk een leerzaam karakter hebben. Dit houdt in dat we bij voorkeur werkt met technieken die we nog niet eerder hebben toegepast.

## 6 Robot

**Autonom** De robot moet autonom kunnen voortbewegen. Dit houdt in dat nadat de robot van te voren een bepaalde locatie heeft ontvangen, hij deze locatie vervolgens moet kunnen vinden zonder verdere hulp. Van te voren mag wel de robot wel worden geholpen bij het opbouwen van de benodigde kaart.

**Verplaatsbaar** De robot moet in staat zijn om zich zonder moeite over een vlakke ondergrond te kunnen voortbewegen. Vlak wordt hierbij gezien als een ondergrond die hoogstens 5 mm hoogteverschil heeft. Daarnaast moet de robot in staat zijn om over kleine drempels van ca. 2 cm te komen en omhoog te kunnen over hellingen tot ca. 20 graden (op een lagere snelheid).

**Wendbaar** De robot moet in staat zijn om volledig te kunnen draaien tot 360 graden zonder van zijn plaats te raken. De draai moet dus op de huidige locatie kunnen worden uitgevoerd zonder extra ruimte te gebruiken.

**Nauwkeurigheid** De robot moet in staat zijn om objecten van minimaal één kubieke decimeter te herkennen. Voor het herkennen en later terugvinden van deze objecten is nauwkeurige beweging nodig. De volgende eisen zien wij als richtlijn voor een nauwkeurige beweging:

- De robot legt een afstand van 10m af met een maximale afwijking van ongeveer 2% (dus ca. 20 cm).
- De robot legt een volledig cirkel af met een maximale afwijking van ongeveer 2% (dus ca. 7 graden).

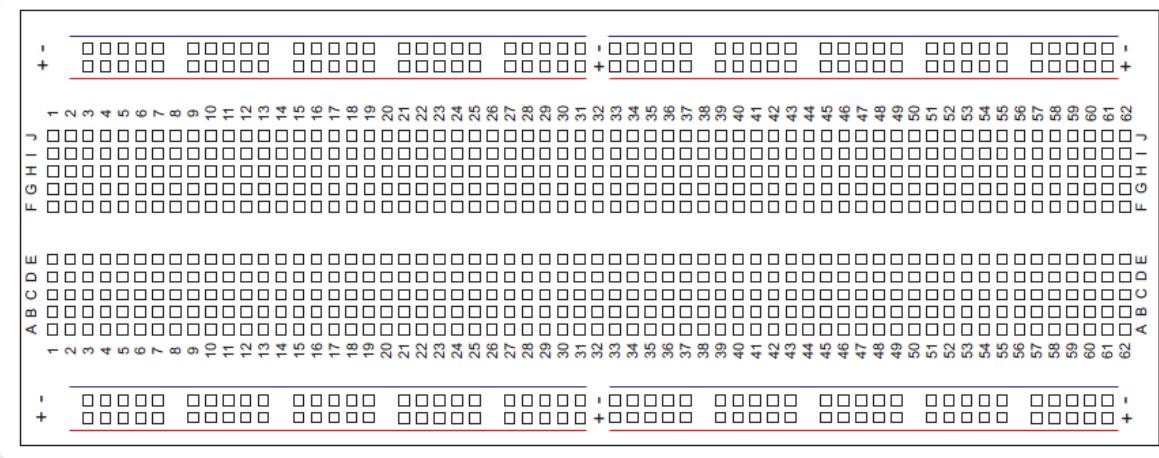
**Kracht en snelheid** De motoren van de robot kunnen genoeg kracht leveren om de robot, met alle onderdelen geladen, zonder moeite te laten rijden op een helling van 20 graden. De robot moet daarnaast een snelheid van minstens 5 km/h kunnen bereiken.

**Verwerking** De robot moet een gegeven opdracht binnen drie seconden ontvangen en deze omzetten in een actie (en deze actie vervolgens zelfstandig uitvoeren).

**Modules** De robot moet minstens twee plekken hebben om een sensor/actuator module te monteren, die respectievelijk moet kunnen worden uitgelezen of moet kunnen worden aangestuurd door de controller. Deze plekken zijn aanwezig in de beweegrichting van de robot (bij een robot die geen standaard beweegrichting heeft, is dit uiteraard niet noodzakelijk).

**Communicatie** De robot moet bedraad kunnen worden aangestuurd en moet draadloos kunnen communiceren met externe aansturingsapparatuur. De draadloze communicatie moet tot een afstand van vijf meter zonder obstakels en drie meter met obstakels goed verlopen.

**Breadboard** Op de robot moet ruimte zijn voor een breadboard om ter plekke schakelingen te bouwen met de bijbehorende modules. Een breadboard is een bord waarin kolommen onderling gekoppeld zijn en pinnen kunnen worden gestoken voor het maken van simpele schakelingen



**Figuur 3:** Breadboard

**Stroomvoorziening** De robot moet zijn stroom halen uit batterijen en minstens een uur lang op deze batterijen kunnen werken (als de robot intensief bezig is). Op de robot moet uiteraard ook ruimte zijn om de batterijen kwijt te kunnen.

**Detectie** Objecten die zich op minder dan een halve meter van de robot bevinden moeten kunnen worden gedetecteerd.

## 7 Software

**Modulair** Alle verschillende onderdelen moeten in losse modules worden opgebouwd, die onafhankelijk zijn. Deze modules moeten andere methoden alleen aanspreken via een globale interface (ook wel een API genoemd). Met deze structuur moet het vrij eenvoudig mogelijk zijn om aanpassingen te maken.

**Gebaseerd op opensource platform** Het code-platform behorende bij de controller (zie pagina 10) moet opensource zijn. Als het opensource is kunnen wij namelijk de code inzien en eventueel zelf aanpassen.

# Deel III

## Ontwerpvoorstel en prototype

Met behulp van het programma van eisen zijn we literatuuronderzoek gaan doen naar de verschillende onderdelen en hebben we verschillende schetsen gemaakt. We hebben met behulp van dit onderzoek een indeling gemaakt in verschillende deeltaken die de robot moet uitvoeren en onderdelen die robot moet hebben. Uit ons literatuuronderzoek zijn verschillende oplossingen naar voren gekomen (zie pagina 32). Hieruit zullen we beargumenteerd kiezen en met de gekozen onderdelen een prototype bouwen. De prijs van de onderdelen is verderop te vinden (zie pagina 55)

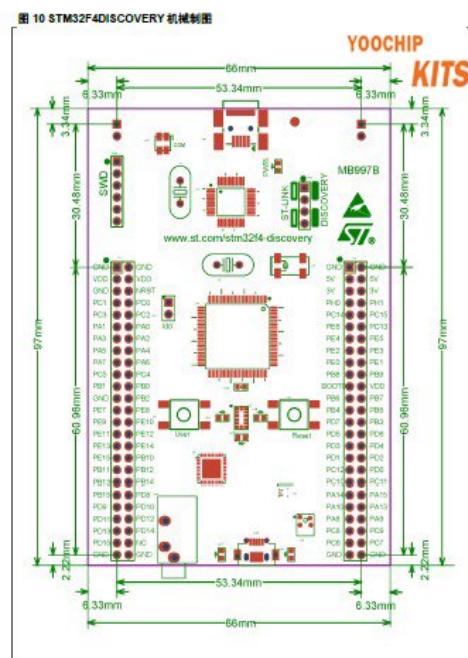
### 8 Ontwerpvoorstel

Arduino	NXP	STM32 F4DISCOVERY	Raspberry Pi
---------	-----	-------------------	--------------

Tabel 1: Controllers

**Controller** Van alle onderzochte controllers viel de Arduino als eerste af. De Arduino bood te weinig rekenkracht en was niet in staat om met alle benodigde sensoren te communiceren. Naast de Arduino bleek ook de Raspberry Pi al snel niet echt een geschikte keuze. Hoewel de Raspberry Pi een stuk krachtiger is, was deze niet afgestemd op de robot die wij wilden maken: het ondersteunde een hoop mogelijkheden die wij nooit nodig zouden hebben en had aan de andere kant nogal weinig pinnen, die wel erg belangrijk waren.

Uiteindelijk hebben we een keuze gemaakt uit de NXP en de STM32F4 DISCOVERY. Hoewel de eerste wat kleiner was, hebben we toch gekozen voor de STM32F4 DISCOVERY (vanaf nu STM genoemd). Onze keuze was hierbij gebaseerd op drie voorname punten. Ten eerste is deze microcontroller 'sterker' dan NXP, dit is voornamelijk terug te zien in de hoeveelheid geheugen en de snelheid van de processor. Daarnaast biedt deze controller veel meer mogelijkheden dan de NXP: de STM heeft veel meer pinnen, waardoor we veel konden aansluiten, en bied toegang tot veel verschillende protocollen, die de communicatie met de sensoren vereenvoudigden. De doorslaggevende reden was dat de STM al een ingebouwde magnetometer had, die we dan niet los meer hoefden te kopen.



Figuur 4: Technische tekening van STM32F4 DISCOVERY

Voor deze controller zijn diverse bibliotheken geschreven die toegang bieden tot de verschillende functies, waarvan ChibiOS de bekendste is (zie pagina 17). Hier komt echter wel direct het grootste nadeel van deze controller naar boven: er is niet erg eenvoudig mee te werken. De documentatie is veel minder overzichtelijk dan bij voorbeeld de Arduino, mede omdat er zoveel mogelijke opties zijn. Voor ons was dit echter voornamelijk een leuke uitdaging, die paste in onze eisen (zie pagina 11).

Servo's	DC-motor	Geared DC-motor	Stappen-motor
---------	----------	-----------------	---------------

**Tabel 2:** Motoren

**Motoren** Servo's bleken niet de benodigde nauwkeurigheid, kracht en configuratiemogelijkheden te geven die we wilden en deze vielen dan ook snel af. Over de andere mogelijkheden hebben we langer getwijfeld. Uiteindelijk hebben we gekozen voor geared DC-motoren. De reden was simpel: omdat we een set vonden die meerdere ratio's van de tandwielen ondersteunde. Dit was voornamelijk erg handig omdat we dan later de verhouding tussen kracht en snelheid konden aanpassen afhankelijk van onze wensen. We hadden van te voren de verwachte benodigde kracht berekend (zie pagina 43), maar mogelijk zou later in de praktijk blijken anders te zijn. Een ander voordeel van deze motoren is dat de oplossing behoorlijk goedkoop is, de losse motoren en de motorcontroller kosten samen relatief weinig. Ondanks de relatief lage kosten kunnen deze motoren in combinatie met de gearbox toch zeer redelijk kracht en snelheid leveren. Door de modulaire opzet van de motoren, de gearbox en de motorcontroller kunnen we ook relatief simpel onderdelen vervangen en aanpassen als we dat willen.

Normaal	Mecanum	Omni-directioneel	Ball-caster
---------	---------	-------------------	-------------

**Tabel 3:** Wielen

**Wielen** Onze keuze hier viel op twee wielen en een ball-caster. Dit biedt een grotere wendbaarheid dan vier wielen, door de grote wendbaarheid van de ball-caster. Er zijn nog wendbaardere wielen: omnidirectionele wielen en mecanumwielen, die beiden ook zijdelings kunnen bewegen. Deze twee soorten wielen zijn echter veel duurder, omdat ze beiden extra motoren nodig hebben (en een extra motorcontroller voor de aansturing). Deze extra kosten wogen niet op tegen nog betere wendbaarheid van deze wielen.

Kabel	SD-kaart	433 MHZ	XBee	Wifi	Bluetooth
-------	----------	---------	------	------	-----------

**Tabel 4:** Communicatie-modules

**Communicatie** De kabel en de SD-kaart vielen vrijwel direct af, omdat het instandhouden van de communicatie over langere afstanden toch al snel een probleem wordt. De draadloze communicatie-mogelijkheden bleven dan over. De goedkoopste oplossing met de 433 MHZ ontvangers viel vervolgens ook af, omdat het veel tijd zou vragen om deze

werkend te krijgen.

Uiteindelijk hebben we gekozen voor een WiFi-module. Deze heeft een vrij grote range vergeleken met Bluetooth. Daarnaast was de oplossing goedkoper dan de vaak erg dure XBee's, vooral omdat we maar een module nodig hadden, omdat elke computer en smartphone al met Wifi is uitgerust. Aangezien deze module daarnaast voor een zeer redelijke prijs ook veel uitgebreide mogelijkheden bood die later nog van pas konden komen (waaronder beveiliging, mogelijkheid tot het ophalen van webpagina's en tijdsynchronisatie) was dit volgens ons duidelijk de beste keuze.

Mechanische terugkoppeling	Optische terugkoppeling
----------------------------	-------------------------

**Tabel 5:** Terugkoppelingssensoren

**Motor-terugkoppeling** Uit financiële en praktische overwegingen, hebben we gekozen om geen motorterugkoppeling in te bouwen. Het ontbreken van deze sensoren heeft wel negatieve invloed op de nauwkeurigheid van de motoren, maar dit leek toch het beste onderdeel om op te besparen.

Sonar	IR
-------	----

**Tabel 6:** Detectiesensoren

**Botsingsdetectie** Hier hebben we gekozen voor een wat duurdere optie, een sonar. Een sonar werkt op grotere afstanden ook nog goed in tegenstelling tot IR. Vooral op grote afstanden is het voor ons interessant om objecten goed te kunnen identificeren, omdat dit het creëren van een kaart eenvoudiger maakt. Daarnaast kan de sonar (door de verre afstandsbeperking) ook gebruikt worden als hulpmiddel voor plaatsbepaling door aangelegde afstanden te meten.

Lichtsensor	CMOS	Sonar	Accelerometer
Kompas	Magnetometer + Accelerometer	Terugkoppeling	GPS

**Tabel 7:** Plaatsbepalingssensoren

**Plaatsbepaling** De plaatsbepaling was, zoals al uit de probleemanalyse bleek, erg belangrijk. Daarom hebben we hier ook meerdere onderdelen voor. Volledige controle over de plaats doormiddel van Mini-GPS viel direct af, aangezien deze ruim buiten ons budget viel. Ook gebruik van een lichtsensor of een CMOS vonden we niet passen binnen de eisen, omdat de robot dan losse bakens nodig zou hebben. Hiermee zou de robot dan minder autonoom zijn. Uit de overige sensoren hebben we een aantal gekozen. De sensoren zijn ingericht in twee categoriën: richting en afstand.

Voor de afstand hebben we als eerste een sonar. Deze kan helpen bij het bepalen van de plaats, door middel van detectie van afstanden. Verder hebben we een accelerometer, waarmee het mogelijk is om versnelling te meten. Een accelerometer kan met behulp van deze versnelling bewegen detecteren. De accelerometer gaat echter ook voornamelijk gebruik worden voor het kompas. Voor het berekenen van de hoek hebben we namelijk enkel een magnetometer. Dit is geen volledig kompas, die waren namelijk te duur. Voor het berekenen van de hoek is daarom een correctie nodig. Een magnetometer kan wel de richting van het aardmagnetisch veld berekenen, maar kan hier geen goede conclusies uit trekken zodra de sensor niet evenwijdig aan het aardmagnetisch veld ligt. Om deze afwijking te corrigeren kan een accelerometer gebruikt worden, deze meet namelijk de versnelling veroorzaakt door de zwaartekracht en weet daardoor ook de richting van het aardmagnetisch veld (die staat daar namelijk loodrecht op).



**Figuur 5:** Sonar

Frame	Hout	Staal	Plastic (3D geprint)
-------	------	-------	----------------------

**Tabel 8:** Frames

**Frame** In ons testmodel hebben we gekozen voor een houten frame. Dit is sterk en goedkoop en daarnaast makkelijk in simpele vorm te zagen. Ook kan op dit bord makkelijk het benodigde breadboard worden gezet. Een nadeel is wel dat het frame niet in ingewikkeldere vormen gemaakt kan worden en dat het frame vrij zwaar is. Een stalen model was echter duurder en ingewikkelder om te fabriceren, en een plastic model bleek ook niet eenvoudig mogelijk. Daarom hebben we in eerste instantie gekozen voor een houten model.

Fabrikantsomgeving	ChibiOS	Linux
--------------------	---------	-------

**Tabel 9:** Code-platform

**Besturingssysteem** Het besturingssysteem is afhankelijk van het gebruikte microcontroller-platform. Aangezien wij gekozen hebben voor de STM is het nodig om een apart besturingssysteem te gebruiken, in de fabrikantsomgeving zijn namelijk de benodigde standaardbibliotheeken met code niet bijgeleverd. Onze keuze voor het besturingssysteem viel vrijwel direct op ChibiOS, een realtime besturingssysteem, speciaal afgestemd voor onder andere de STM. De keuze voor zo'n realtime-OS was erg praktisch om nauwkeurigheids- en snelheidsredenen. Dit besturingssysteem bevat een uitgebreide bibliotheek met alles wat we nodig hadden, die tevens relatief goed gedocumenteerd is (al is het wel veel lastiger mee te werken dan met bijvoorbeeld een Arduino). De ondersteuning is niet zo uitgebreid als bij bijv. Linux, maar dit besturingssysteem was veel te groot voor onze doeleinden en daarnaast niet afgestemd op een STM.

## 9 Prototype

Met behulp van de schetsen hebben we een eerste prototype gemaakt (zie pagina 54). Alhoewel dit model nog verre van perfect was, konden we het gebruiken om eerste tests mee uit te voeren. We liepen tijdens en na het bouwen wel tegen diverse problemen aan. In het volgende deel behandelen we enkele van die problemen. Hieronder een beschrijving van enkele onderdelen.

**Houten frame en gearbox** Alle onderdelen zitten vast op een groot houten frame, met voldoende lengte om alles tijdelijk op te zetten. De breedte van het frame is zo gezaagd dat het motorblok er precies op kon. De gearbox is volgens de handleiding in de ratio gezet die het dichtste kwam bij de waardes uit onze berekeningen: 38:1 (zie pagina 43).

**Wielen en ballcaster** De wielen konden vrij eenvoudig worden gekoppeld, omdat deze speciaal hoorden bij de gearbox. Voor de verbinding was enkel een speciaal bijgeleverd verbindingsstuk nodig. De ballcaster kon tevens vrij eenvoudig met twee schroeven (nadat we gaten hadden geboord) aan het houten plankje worden bevestigd.

**Microcontroller, breadboard en batterijhouder** De STM, het breadboard en de batterijhouder hebben we met plakband aan het hout vastgemaakt.

**Sensoren en motorcontroller** De sensoren en de motorcontroller hebben we nog geen vast plek gegeven, en hebben we alleen via het breadboard gekoppeld.

## 10 Software

Zoals gezegd maken wij gebruik van ChibiOS. Dit platform is geschreven in de programmeertaal C++ en de libraries zijn daarom ook aanspreekbaar in C++. Voor de omzetting naar machine-code gebruiken we een aangepaste versie van de meestgebruikte opensource compiler GCC. Om ons te houden aan de software eisen (zie pagina 13) hebben we enkele afspraken gemaakt over de layout. Voor het schrijven van de code maken we zoveel mogelijk gebruik van functies (kleine stukken code). Deze functies zijn gebundeld in zogenaamde services en scripts. De services zorgen voor de communicatie met de onderlinge onderdelen en de scripts gebruiken deze services om een bepaalde taak uit te voeren. Services kunnen onderling gekoppeld zijn, en dus van elkaar afhankelijk zijn, maar dit moet zoveel mogelijk worden vermeden. Het samenvoegen van de services wordt dus voornamelijk gedaan in de scripts.

Functies in een service die openbaar zijn voor scripts en andere services worden geschreven in zogenaamde camelcase op de volgende manier: `serviceAction` (bijv `wiflyInit`). Functies die alleen in de service zelf gebruikt dienen te worden gebruiken de volgende syntax: `service_action` (bijv. `wifly_loop`). Het gebruik van deze afspraken is terug te zien in de code (zie pagina 46).

# Deel IV

# Problemen en oplossingen

## 11 Bouw

Bij de bouw kwamen al snel enkele tekortkomingen in onze ontwerpen naar voren.

**Frame** We hebben ervoor gekozen om uiteindelijk dunner hout te halen, omdat het frame van het prototype te groot en te zwaar was en daarmee de haalbaarheid van enkele eisen in de weg zat.

**Ruimte** Op het frame bleek te weinig ruimte te zijn om alles goed kwijt te kunnen. Het gevolg was dat alles een beetje op elkaar lag. Voor een prototype aanvaardbaar, maar voor een eindproduct niet. We hebben daarom gekozen om in het eindproduct een dubbele laag te gebruiken, waarbij de STM en de elektronica bovenin zaten en de gearbox en de batterijhouder onderin.

## 12 Elektronica

**Verbindingen** Het breadboard deed in eerste instantie prima dienst, maar na een tijdje kregen we last van losschietende verbindingen en niet goed verbonden draden. Hierdoor leek het alsof er een onderdeel kapot was gegaan, terwijl er enkel iets was fout gegaan met de verbinding. Een bijkomend probleem was dat onderdelen verbinden met de STM niet zo eenvoudig was, omdat de STM uitstekende pinnen had, terwijl onze draadjes die ook hadden (we moesten daardoor enkele headers gebruiken, die een hoop ruimte innamen). Om deze redenen hebben we besloten om in het uiteindelijke ontwerp gebruik te maken van een prototyping board, waar je alles op kon solderen.

**Documentatie pinnen** De pinnen van de STM bleken meerdere functies te hebben, die konden worden ingesteld. Zo kon een pin tegelijk ondersteuning bieden voor analoge uitvoer (uitvoer van in te stellen spanning), maar ook voor tekstcommunicatie (zoals met de wifly). Welke nummer een functie had bleek echter vrij lastig te vinden. Uiteindelijk vonden we een excel-sheet,<sup>1</sup> maar deze bleek niet volledig te kloppen: alle functies stond er wel per pin in, maar niet met het correcte nummer erbij. Uiteindelijk bleek dat de benodigde informatie stond in de handleiding van de STM rond bladzijde 58,<sup>2</sup> wat we in eerste instantie over het hoofd zagen.

**Sonar** Om de sonar nuttig te laten zijn was het noodzakelijk dat deze loodrecht stond op het frame (anders was de sensor niet op de muur gericht, maar op het plafond). Om dit goed vast te maken was waren er echter gekromde headers nodig, die we daarom ook hebben gehaald.

---

<sup>1</sup>Pinfuncties. URL: <http://kornakprotoblog.blogspot.nl/2012/01/pinout-spreadsheet-for-stm32f4.html>.

<sup>2</sup>Datasheet van STM32 F4 Discovery. URL: [http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data\\_brief/DM00037955.pdf](http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00037955.pdf).

**Spanning** Aangezien in onze batterijhouder vier batterijen konden en elke batterij die we gebruikten 1.2V gaf, leverde dit samen 4.8V aan spanning op. De microcontroller moest echter 3.3V hebben, terwijl de motoren 3 V nodig hadden (maar een veel hogere stroomsterkte). We moesten dus een manier hebben om de spanning omlaag te krijgen. Hiervoor hebben we een speciale onderdeel gekocht dat de spanning van de batterij naar 3.3V kan omzetten, zodat de microcontroller op de gewenste spanning kan opereren. De motor driver kon vervolgens gewoon 4.8 volt worden toegediend, alleen het moest worden voorkomen dat die volledige spanning ook naar de motoren zouden gaan (die zouden bij die spanning veel eerder kapotgaan). Na het onderzoeken van verschillende methoden, hebben we uiteindelijk een hele simpele oplossing doorgevoerd. Door het gebruiken van analoge communicatie (met instelbare spanningen) kan je de snelheid van de motoren veranderen en dit werkt doormiddel van een verschil in spanning! Door de spanning bij de microcontroller laag te houden, werd er via de motor driver niet meer dan 3V naar de motoren aangevoerd.

## 13 Software

Bij het schrijven van de software kwamen we enkele problemen tegen die we nog niet eerder waren tegengekomen. Hieronder de belangrijkste problemen die we tegenkwamen.

**Synchronisatie** Bij een project met meerdere sensoren is het belangrijk dat er meerdere dingen tegelijk worden gedaan. Dit kan erg makkelijk bij een realtime-OS door gebruik te maken van zogenaamde threads. Dit brengt echter wel synchronisatie problemen met zich mee. Hieronder vallen diverse problemen, zoals onverklaarbare acties als twee threads tegelijk zich met hetzelfde bezighouden of het oneindig wachten van twee threads op elkaar. Om te zorgen dat threads goed met elkaar kunnen communiceren en elkaar niet in de weg zitten is een goed software-ontwerp nodig. Deze manier van multithreaded programmeren was voor onze beiden nieuw en verliep daarom (zeker in het begin) niet erg goed.

**Debugging** Het debuggen van een microcontroller is heel anders dan het debuggen van een computer-programma. Bij een computer-programma kun je meestal vrij nauwkeurig zoeken naar een fout en (een hoop) debug-informatie meegeven om je fout te zoeken. Bij een microcontroller werkt dit niet, want er is geen standaarduitvoer. In simpele gevallen voldoet het om wat lampjes te laten branden, maar bij ingewikkeldere problemen is er meer nodig. Deze problemen treden veel sneller op bij een microcontroller dan een computer, omdat deze laatste vaak meer beveiliging tegen ze heeft ingebouwd. Gelukkig ondersteunt deze STM geadvanceerdere debugmanieren om ingewikkeldere fouten op te lossen door de uitvoer naar de computer door te sturen, wat erg handig bleek. Het kostte echter enige tijd om deze constructie op te zetten (mede omdat de handleiding vaak niet erg duidelijk was).

**Communicatie robot** De communicatie met de robot bleek nog aardig ingewikkeld om op te zetten. Voor de communicatie met de robot hebben wij gekozen om een eigen protocol te gebruiken. De robot kan dus dan reageren op gegeven opdrachten (zoals set\\_speed 70 of forward 20). Aangezien het onbekend is wanneer er informatie ontvangen wordt (en hoeveel informatie er ontvangen wordt) hadden we eerst wat moeilijkheden met

het opzetten van goede communicatie. Uiteindelijk hebben we gekozen om één losse taak alle inkomende informatie te laten lezen en schrijven en deze informatie door te spelen naar een andere taak als deze daarom vroeg, of (als geen van de taken een lopende vraag had) een functie aan te roepen (een zogenaamde callback). Op deze manier was het vrij goed mogelijk om een protocol te schrijven (zie pagina 45) en tegelijk ook te zorgen dat er losse informatie gelezen en geschreven kon worden.



Figuur 6: Wifly

**Uitlezen sonar** Het uitlezen van de gegevens van de sonar ging vrij makkelijk over seriële communicatie, maar dat was niet goed genoeg om aan onze eisen te voldoen: de uitleessnelheid was namelijk te laag. Het uitlezen via een ander protocol leverde echter een hoop problemen op die tijd hebben gekost.

**Berekenen hoek** Het berekenen van de hoek met een niet-gekompenseerde kompas met behulp van een accelerometer leverde ook moeilijkheden op. Ten eerste moest de data van beide onderdelen worden binnengehaald en daarnaast moest dit worden omgezet in een hoek in graden. Met behulp van eerder uitgezochte informatie<sup>3</sup> hebben we dit geïmplementeerd in de code. Door enkele verschillen in uitlezen was ook dit tijdrovend.

---

<sup>3</sup> Tilt-compensatie. 2012. URL: <http://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf>.

# Deel V

# Evaluatie

Nadat we het prototype van de robot hebben verbeterd (zie pagina 54) en de benodigde software hebben geschreven is het nu tijd om deze te testen en te evalueren: voldoet de robot aan de gestelde eisen? Tot slot trekken we hier een conclusie uit en geven we onze persoonlijke evaluatie op het project.

## 14 Eisen

Als eerste testen we hier ons ontwerp aan de eerder gestelde eisen (zie pagina 10).

### 14.1 Algemeen

**Zelfgebouwd** De robot is zover als mogelijk zelfgebouwd. Veel onderdelen zijn echter toch ingekocht, omdat bijv. een eigen gearbox maken te ingewikkeld bleek te zijn.

**Gebaseerd op elektronica platform** Ons platform was gebaseerd op de STM32F4 DISCOVERY. Deze voldeed ruim aan onze eisen, zodat we ruimte hadden voor wat spelling. De benodigde bibliotheken konden we halen uit het gebruikte realtime-OS ChibiOS.

**Modulair** Door in eerste instantie gebruik te maken van een vrij losse layout met een breadboard konden we vrij eenvoudig wijzigingen aanbrengen in de verschillende sensoren. Het bleek aan het eind echter wel praktischer om een wat minder modulair prototyping board te gebruiken, omdat dit zorgde voor stabielere communicatie tussen de onderdelen. Met dit bord ook nog goed mogelijk om nieuwe modules toe te voegen, maar daar moest echter wel eerst gesoldeerd voor worden.

**Betaalbaar** Uiteindelijk hebben we ongeveer 180 euro besteed aan de robot (zie pagina 55), waardoor onze robot binnen de prijsklasse viel. We hebben door deze limiet wel meerdere keren moeten besparen op onderdelen, waardoor het halen van andere eisen, zoals bijvoorbeeld de nauwkeurigheid, ingewikkelder werd.

**Leerzaam** Het bouwen van de robot is voor ons uitermate leerzaam geweest. Omdat we veel moesten uitzoeken viel de voortgang soms tegen (zie pagina 24).

### 14.2 Robot

**Autonom** Deze eigenlijk vrij primaire eis is niet volledig geslaagd, de robot is uiteindelijk minder autonom dan we zelf zouden willen. De robot is in staat om zeer simpele opdrachten te verwerken (zoals twee seconden lang vooruitgaan), maar niet om complexere opdrachten zelfstandig uit te voeren. Dit kwam niet omdat het voor ons niet mogelijk was, maar meer door een gebrek aan tijd. Het uitzoeken en het correct werkend krijgen van vele onderdelen bleek nog aardig ingewikkeld te zijn. Wij hopen in een later stadium de robot nog autonom te kunnen laten zijn.

**Verplaatsbaar** Uit testen bleek dat de robot zonder problemen kon voortbewegen over een vlakke ondergrond en dat hellingen ook weinig problemen vormden. Het beklimmen van kleine hellingen kan echter alleen achteruit, omdat de ballcaster veel te klein is voor een helling. Het voordeel van de ballcaster was echter wel een betere wendbaarheid (zie hieronder).

**Wendbaar** De robot had een grote wendbaarheid door het gebruik van twee wielen en een ballcaster. De lengte van de robot bleek in eerste instantie een probleem om een stabiele draaiing uit te voeren, maar dit probleem werd grotendeels opgelost door later een smaller dubbel frame te gebruiken. Verdere wendbaarheid was enkel mogelijk met ander type wielen, maar dit was een te dure investering.

**Nauwkeurigheid** De nauwkeurigheids-richtlijnen zijn in onze eerste versie helaas nog niet helemaal zoals we willen. De bewegingen van de robot zijn nog veel te onnauwkeurig om binnen de 2% norm te vallen, uit simpele testen is gebleken dat het momenteel nog op de 10-15% ligt. Deze eis is de belangrijkste eis om te verbeteren bij een volgende versie.

**Kracht en snelheid** Een snelheid van 5km/h haalde ons robot zonder veel moeite, omdat we de robot volgens berekeningen zo hadden afgesteld. Uiteindelijk bleek echter dat 5km/h een vrij grote snelheid is, die het lastiger maakte om ook rustigere bewegingen uit te voeren. Mogelijk is het in een latere versie beter om een lagere maximumsnelheid in te stellen, waardoor de nauwkeurigheid kon worden verbeterd.

**Verwerking** De communicatie met de wifi bleek in de meeste gevallen prima aan deze eis te voldoen (reactietijd minder dan een halve seconde), enkele keren schoot de reactietijd echter uit naar ongeveer 2-3 seconden, die maar net binnen de eisen lag. Waardoor deze uitschieters veroorzaakt werden, hebben we niet kunnen achterhalen.

**Modules** Aangezien elke sensor anders was, bleek het niet mogelijk om vaste plekken te gebruiken waarop sensoren konden worden bevestigd. Onze robot is echter ruim groot genoeg om voldoende modules om te kunnen monteren. De sonar is volgens de eis in de beweegrichting van de robot vooraan gezet.

**Communicatie** De communicatie bleek over relatief grote afstanden nog uitstekend te verlopen, waarbij alleen de verwerkingsnelheid (zie hierboven) af en toe om onbekende reden tegenviel.

**Breadboard** Hoewel we in het prototype aan deze eis hebben voldaan, bleek het later in ons eindproduct toch logischer om over te stappen op een prototyping board, omdat deze zorgde voor een stabielere verbinding. Deze eis bleek uiteindelijk te specifiek te zijn en niet nodig voor het maken van een robot.

**Stroomvoorziening** De robot bleek enkele uren te kunnen rijden, waarbij aan deze eis werd voldaan. In de toekomst denken we echter het stroomgebruik nog verder omlaag te kunnen brengen.

**Detectie** Door gebruik te maken van een sonar bleek het mogelijk om goed te kunnen detecteren of een object dichtbij was of ver weg. Met de sonar zou het ook mogelijk moeten zijn om objecten te detecteren die zich op een verdere afstand bevonden (in een bereik van 10 centimeter tot 8 meter), maar hiervoor hebben wij nog geen tijd gehad.

### 14.3 Software

**Modulair** Door gebruik te maken van losse services en scripts, hebben we een vrij gestructureerde code-base gecreeerd. Er is veel gebruik gemaakt van functies voor de communicatie tussen services en scripts waarbij we een goede modulaire structuur hebben opgebouwd.

**Gebaseerd op opensource platform** Het software-platform waarop onze robot is gebaseerd is ChibiOS, dat een opensource licentie heeft en daarom ook vrij kan worden aangepast.

## 15 Conclusie

Algemeen kan worden geconcludeerd dat onze robot nog niet voldoet aan de gestelde eisen en dus ook nog niet in staat is om het gestelde probleem op te lossen. Het niet voldoen aan de eisen ligt niet zozeer aan het kiezen van de verkeerde oplossingen, maar meer aan het gebrek aan tijd. Onder andere de belangrijke eisen op het gebied van autonomie en nauwkeurigheid bleken nog niet eenvoudig te realiseren als aanvankelijk gedacht.

Wij denken dat onze robot wel in staat is om aan deze eisen te gaan voldoen, als deze verder wordt ontwikkeld. Wel zou het kunnen dat de nauwkeurigheid met de huidige onderdelen nog niet voldoende kan worden gegarandeerd. In dat geval hebben we het budget toch te scherp afgesteld en blijkt dat er voor het halen van ons doel toch meer nodig was. Door het optimaliseren van de software kan de nauwkeurigheid verbeterd worden.

## 16 Persoonlijk

Het eindproduct is niet geworden is wat we aanvankelijk wilden, toch zijn we allebei tevreden over het bereikte resultaat. We hebben veel tegenslagen moeten overwinnen om alles werkend te krijgen, maar uiteindelijk hebben we toch een stabiele basis gemaakt, waarbij we in een later stadium de overgebleven eisen kunnen gaan vervullen. De komende maanden zullen we doorgaan met het verbeteren van de robot, zeker omdat de basis nu werkt en het alleen nog maar interessanter kan worden.

Het project is voor ons zeer leerzaam geweest. Hoewel het in het begin allemaal erg onduidelijk was en we moeite hadden met grote hoeveelheid informatie, zijn we uiteindelijk erg tevreden over wat we geleerd hebben. De kennis die we nu hebben op het elektronisch en mechanisch vlak gaat ons later zeker van pas komen als we op de universiteit aan ingewikkeldere projecten gaan werken.

# Deel VI

## Verantwoording

### 17 Logboek

Naam	Datum	Plaats	Duur	Omschrijving
K + E	t/m 2-12-11	Thuis	2 uur	Zoeken onderwerpen • Iets met robots
K + E	2-12-11	School	-	Inleveren eerste onderwerp: Maken robot die een tekening kan maken
K + E	t/m 31-01-12	Thuis	2 uur	<ul style="list-style-type: none"> <li>• Maken programma van eisen en doelstelling.</li> <li>• Doel van een tekening maken gedeeltelijk laten vallen en vervangen door algemene taak uitvoeren. Dit doen we omdat het ons vooral gaat op het doorlopen van het hele proces van het maken van een robot (doel is minder belangrijk).</li> </ul>
K + E	21-11	School	1/2 uur	Eerste gesprek begeleider: <ul style="list-style-type: none"> <li>• Concreter maken plan</li> <li>• Rondje (maken -&gt; verbeteren -&gt; maken)</li> <li>• Opstellen probleem- en doelstelling</li> <li>• Opstellen eerste programma van eisen</li> <li>• Maken eerste ontwerp</li> <li>• Mogelijkheden materialen</li> </ul>

Naam	Datum	Plaats	Duur	Omschrijving
K+E	2-3-12	Skype	1 uur	<p>Aantal afspraken + plannen gemaakt:</p> <ul style="list-style-type: none"> <li>• Afspraak wie wat gaat uitzoeken.             <ul style="list-style-type: none"> <li>– Verschillende sensoren en uitzoeken hardware platform</li> <li>– Model van het frame met actuatoren</li> </ul> </li> <li>• Opzetten mappen-indeling project</li> <li>• PWS waarschijnlijk in Latex omdat het voor meer platformen is en simpeler te onderhouden is voor grote projecten.</li> </ul>
K	3-3-12	Thuis	3 uur	<p>Keuze model-software:</p> <ul style="list-style-type: none"> <li>• Sketchup, omdat het relatief makkelijk in gebruik is en voorlopig voldoende mogelijkheden geeft.</li> </ul> <p>Maken modellen:</p> <ul style="list-style-type: none"> <li>• Robot met vier wielen: stabieler en simpeler</li> <li>• Robot met twee wielen: waarschijnlijk wendbaarder en ook nauwkeuriger</li> </ul>
E	3-3-12	Thuis	1 uur	<p>Uitgezocht wat voor pakketen we allemaal nodig hebben voor het verslag gemaakt in latex</p> <ul style="list-style-type: none"> <li>• Het geometry pakket voor de margins en de pagina groottes</li> <li>• GIT-repo gemaakt voor backup en versiecontrole zodat we altijd terug kunnen.</li> </ul>

Naam	Datum	Plaats	Duur	Omschrijving
K	17-3-12	Thuis	4 uur	<p>Verschillende dingen gedaan:</p> <ul style="list-style-type: none"> <li>• Begin gemaakt uitzoeken materialen</li> <li>• Verder gewerkt aan modellen</li> <li>• Verder gewerkt aan opzetten verslag</li> </ul>
E	19-03-12 en eerder	Thuis	4 uur	<p>Meerdere dingen gedaan</p> <ul style="list-style-type: none"> <li>• Verschillende printer protocollen opgezocht en gekeken welke we kunnen gebruiken</li> <li>• Nagedacht over de verschillende sensoren die nodig zijn</li> <li>• Nog wat andere informatie opgezocht,</li> </ul>
K	01-04-12	Thuis	2 uur	Voorgestelde wijzigingen van begeleider doorgevoerd en op diverse stukken de leesbaarheid van de tekst verbeterd.
K	t/m 01-05-12	Thuis	3 uur	Nagedacht over diverse oplossingen voor onze robot en diverse andere modellen (moeten nog worden uitgewerkt in het verslag).
K	02-05-12	Thuis	5 uur	Diverse dingen opgezocht, wijzigingen aangebracht aan de indeling en opmaak en nieuwe stukken tekst geschreven, voornamelijk over de hardware die in de robot gebruikt kan worden.
E	12-05-12 en 13-05-12	Thuis	3,5 uur	Informatie verzameld en stuk gemaakt over verschillende motoren, motor en micro controllers, wielen en encoders.
E	15-05-12	Thuis	3 uur	Informatie opgezocht, verder uitgewerkt en nieuwe stukken toegevoegd.
K	18-06-12	Thuis	2 uur	<p>Verschillende dingen uitgevoerd:</p> <ul style="list-style-type: none"> <li>• Wijzigingen van begeleider doorgevoerd</li> <li>• Verder specificiëren van de taak die de robot moet uitvoeren.</li> </ul>

Naam	Datum	Plaats	Duur	Omschrijving
K + E	t/m 18-06-12	School	1 uur	Diverse korte gesprekken met begeleider gevoerd over de voortgang van het project.
K + E	27-07-12	Huis Koen	2 uur	Afspraken gemaakt over minimale vooruitgang in de zomervakantie. Daarnaast diverse schetsen gemaakt en enkele besluiten genomenen over onderdelen die we willen gebruiken. We hebben vervolgens een indeling gemaakt over de verdeling van de verschillende individuele taken, waaronder het kiezen van de definitieve onderdelen voor de robot.
K	1-8-12	Thuis	3,5 uur	Verder gewerkt aan verslag en diverse stukken aangepast en verder uitgebreid.
E	31-8-12	Thuis	4,5 uur	Schetsen gemaakt, informatie over motoren gezocht, en berekeningen gemaakt voor de nauwkeurigheid. Begonnen met het testen van de software en opzetten toolchain. Eerste tests gerund met ChibiOS.
E	2-9-12	Thuis	4 uur	Gewerkt aan de OS en IR detectie, benchmarks gerund, serial interface
K	2-9-12	Thuis	1,5 uur	Kleine wijzigingen gemaakt aan verslag en begin gemaakt aan ordenen van in de vakantie bedachte dingen voor de robot.
E	3-8-12	Thuis	5 uur	Verder gewerkt aan OS, IR detectie werkend gekregen. Voor de rest gekeken naar mogelijk sensoren.
K	1-9-12	Thuis	2,5 uur	Doorlezen profielwerkstuk, kleine verbeteringen gemaakt en bepaald wat er moest gebeuren
K	2-9-12 t/m 3-9-12	Thuis	12 uur	Onderdelen en ontwerpen bestudeerd en theorie opgezocht die we konden gebruiken om de uiteindelijke onderdelen te kiezen.
K	t/m 4-10-12	Thuis	6 uur	Meer theorie opgezocht over de benodigde onderdelen en de uiteindelijke onderdelen opgezocht en besteld.
K	6-10-12	Thuis	1 uur	Begin gemaakt aan verwerking van door ons gekozen ontwerp en onderdelen in het verslag.

Naam	Datum	Plaats	Duur	Omschrijving
K + E	25-10-12	Thuis	4,5 uur	Bouwen en testen simpele versie robot en afspraken gemaakt over vervolgstappen.
K	28-10-12	Thuis	1,5 uur	Verder testen robot.
E	4-11-12	Hackerspace	4 uur	Solderen nieuwe onderdelen robot
E	12-11-12	Thuis	4 uur	Draadloze communicatie opgezet en probleem opgelost van voltage van de motor met PWM.
K	12-11-12	Thuis	4,5 uur	Aanpassen verslag en afmaken onderdelen ruwe versie. Verbetering, inkorting en uitbreiding van verschillende onderdelen moet later nog volgen.
K	t/m 21-12-12	Thuis	8 uur	Kleine verbeteringen aan het verslag, installeren benodigde software voor de robot, lezen handleidingen van onderdelen en diverse basale testen
E	t/m 23-12-12	Thuis + Hackerspace	15 uur	Verbeteren van de robot (in de hackerspace), meer uitgezocht over de STM, installeren benodigde software en configureren microcontroller, uitvoeren diverse testen voor de robot.
E	26-12-12 t/m 27-12-12	Thuis	5 uur	Verder inlezen werking ChibiOS (specifieke onderdelen voor de STM) en werking onderdelen. Begin gemaakt uiteindelijke software.
K	3-1-13 t/m 4-1-13	Thuis	10 uur	Verder uitzoeken werking wifly en begin gemaakt aan API voor de wifly (ondertussen veel getest aan verschillende functies ChibiOS).
E	8-1-13	Hackerspace	3 uur	Solderen onderdelen robot
E	5-1-13 t/m 12-1-12	Thuis	10 uur	Opzetten USB debugging, interface maken voor sonar en voor magnetometer.
K	13-1-12 t/m 25-1-12	Thuis	12 uur	Aanpassen interface wifly (opzetten volgens stabiever ontwerp) en afschrijven van alle functies voor communicatie met de wifly. Begin gemaakt aan de communicatie met de motoren.

Naam	Datum	Plaats	Duur	Omschrijving
E	26-1-12 t/m 2-4-12	Thuis	6 uur	Verder werken aan interface sonar en magnetometer (stabiel uitlezen), testen software Koen,
K + E	5-2-12	School	1/2 uur	Voortganggesprek met begeleider, globaal vaststellen haalbare eisen voor eindproduct
K + E	10-12-12	Skype	2 uur	Vaststellen haalbare eisen voor eindproduct en benodigdheden vaststellen voor deze laatste eisen. Met behulp van deze informatie een laatste planning gemaakt voor het afbouwen van de robot (laatste versie), het verslag en de presentatie.
E	18-2-13	Hackerspace	5 uur	Begin solderen en ondergrond gemaakt
E	19-2-13	Hackerspace	9 uur	Solderen afmaken en powersupply ontworpen en getest
E	20-2-13	Hackerspace	3 uur	Alles samengevoegd
K	21-2-13	Thuis	1 uur	Begin gemaakt met eisen verslag
K	23-2-13	Thuis	4 uur	Verder gegaan met afmaken verslag
K	24-2-13	Thuis	2 uur	Ruwe versie eindverslag gemaakt
K + E	28-2-13	School	1 uur	Gesprek begeleider over verslag
K	3-3-13	Thuis	3 uur	Begin gemaakt aan GUI
K	t/m 8-3-13	Thuis	4 uur	Wijzigingen aan verslag doorgevoerd, GUI verbeterd en nieuwe code toegevoegd voor de robot.
K + E	9-3-13	Huis Erik en Hackerspace	7 uur	Verbeteringen aan verslag doorgevoerd, begin gemaakt aan presentatie. Verder motor-library uitgebreid om communicatie met de motoren te versimpelen.
K	10-3-13	Thuis	6 uur	Verbeteren motor-library en fixen verschillende kleine bugs.
K + E	11-3-13	School	1/2 uur	Presentatie
K	t/m 14-3-13	Thuis	4 uur	Afmaken verslag (laatste verbeteringen doorgevoerd)
E	t/m 14-3-13	Thuis en Hackerspace	7 uur	Afmaken verslag, meerdere keren robot gefixed en code ageschreven voor sonar en mag/acc meter

Naam	Datum	Plaats	Duur	Omschrijving
<b>K + E</b>	<b>totaal</b>	-	<b>225 uur</b>	<b>Totaal aantal uren (bij elkaar opgeteld)</b>

## Deel VII

# Bijlage: Theoretische achtergrond

## A Onderdelen

Voor onze robot hebben we verschillende minimale eisen vastgesteld. Voor de uitwerking van deze eisen zijn verschillende onderdelen mogelijk. In dit literatuuronderzoek onderzoeken we de verschillende onderdelen. We bekijken verschillende oplossingen voor sensoren, actuatoren en de onderliggende hardware. Deze onderdelen zijn te ingewikkeld om zelf te produceren (zie pagina 10). We bekijken per onderdeel zowel de voor- als de nadelen. De vergelijking van de onderdelen en de uiteindelijke keuze is terug te vinden in het ontwerpvoorstel (zie pagina 14)

### A.1 Hardware

#### A.1.1 Microcontrollers

Er zijn meerdere microcontrollers op de markt die wij kunnen gaan gebruiken, teveel om allemaal te onderzoeken. Daarom hebben we besloten om in eerste instantie alleen gebruik te maken van micro-controllers die we al reeds in bezit hebben. Deze voldoen allemaal aan de minimale eisen zoals we in die hebben gesteld (zie pagina 10). We bekijken daarnaast nog een nieuwe micro-controller (die we zelf graag in de toekomst wilden aanschaffen) en nemen deze microcontrollers dus alvast mee in onderstaande vergelijking.

Momenteel bezitten we de volgende microcontrollers:

1. Arduino Duemilanove (2x)
2. NXP LPCXpresso LPC1114 REV A
3. STM32F4DISCOVERY

Deze microcontroller was er verder nog die na kort vooronderzoek interessant leek:

1. Raspberry Pi

**Arduino Duemilanove** De Arduino Duemilanove is een vrij recente microcontroller van het Arduino<sup>4</sup> platform. Dit platform is open-source en speciaal gericht op prototypes. Het is bedoeld voor designers, hobbyisten en alle anderen die graag werken met interactieve objecten en nieuwe dingen willen ontwikkelen. Het maakt gebruik van een speciale Arduino programmeertaal (gebaseerd op C/C++) en levert een simpele IDE bij, die het werken met de Arduino gemakkelijk maakt. Bij de programmeertaal is een redelijke uitgebreide standaard bibliotheek aan stukken code meegeleverd, die makkelijk is uit te breiden. De Arduino Duemilanove kan overweg met analoge en digitale invoer en uitvoer. De Arduino kan daarbij makkelijk worden uitgebreid met zogenaamde ‘shields’. Deze kun je op een Arduino zetten waardoor extra functionaliteit (zoals een motor

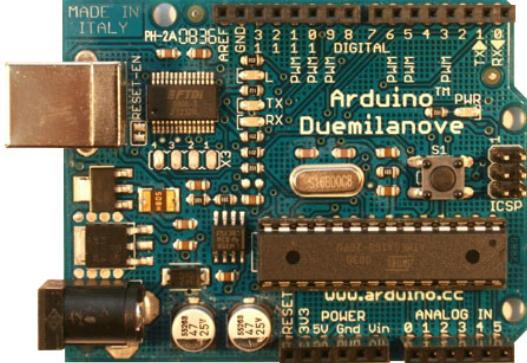
---

<sup>4</sup>Arduino. 2012. URL: <http://arduino.cc/en/>.

controller of een Ethernet poort) wordt toegevoegd, zonder dat daarvoor de elektronica zelf moet worden gesoldeerd.

Voltage	5V
Digital I/O	14
Analoge output	6
Analoge input	6
Kloksnelheid	16 MHz
Flash Memory	32 KB
SRAM	2 KB

Tabel 10: Specificaties Arduino Duemilanove



Figuur 7: Arduino Duemilanove

Het grote voordeel van een Arduino boven alle andere controllers is dat we allebei goed bekend zijn met deze controller. We hebben al diverse kleine projecten gedaan met deze microcontroller en kunnen er daardoor goed mee overweg. De Arduino is daarnaast waarschijnlijk de bekendste microcontroller die er is voor kleine projecten en heeft daarom ook een grote community. Om deze reden zijn er ook aardig wat bibliotheken en tutorials voor geschreven. Dit is een groot voordeel, omdat wij op zich vrij nieuw zijn met robotica en elektronica en veel informatie is daarom een groot pluspunt.

De standaard bibliotheek van de Arduino heeft ook al aardig wat mogelijkheden en is makkelijk in gebruik. De Arduino Duemilanove heeft echter ook een belangrijk nadeel: het is niet echt een snelle microcontroller en hij heeft ook niet veel geheugenruimte. Dit kan vooral later een probleem worden als de code groter wordt en we het project steeds verder gaan uitbreiden. Een ander probleem van de Arduino is dat het werkt met een eigen programmeertaal. Dit maakt het programmeren weliswaar makkelijk, maar maakt het minder uitwisselbaar en dus minder modulair, zoals we graag willen (zie pagina 11).

**NXP LPCXpresso LPC1114** De LPCXpresso<sup>5</sup> is het laag-geprijsde elektronica platform van NXP, gebouwd rondom een ARM processor. Het is speciaal gemaakt voor ontwikkeling van hardware, zonder veel geld te betalen aan allerlei tools. Er wordt een versimpelde op Eclipse gebaseerde IDE bijgeleverd die gebruikt maakt van de programmeertaal C. Met de IDE worden enkele eenvoudige en weinig ruimte vergende bibliotheken meegeleverd. De LPCXpresso heeft een ingebouwde debugger, om makkelijk problemen vast te kunnen stellen. Ook heeft deze microcontroller ingebouwde ondersteuning voor usb 2.0 wat communicatie met veel uiteenlopende apparaten relatief makkelijk maakt (al moeten er dan uiteraard wel drivers beschikbaar zijn). De LPCXpresso biedt upgrade-opties naar grotere systemen en kits.

---

<sup>5</sup> NXP LPCXpresso. 2012. URL: <http://ics.nxp.com/lpcxpresso/>.

Voltage	3.3V
Digital I/O	54
Analoge output	12
Analoge input	6
Kloksnelheid	12 MHz
Flash Memory	32 kB
SRAM	8 kB

**Tabel 11:** Specificaties van LPCXpresso



**Figuur 8:** LPCXpresso

De LPCXpresso is wat professioneler dan de Arduino en is op lange termijn misschien een betere keuze. Er zit een betere processor in en de software wordt in C geschreven wat uiteindelijk prettiger is dan een ‘eigen’ programmeertaal. De LPCXpresso is echter wat ingewikkelder om mee te werken dan de Arduino en is ook een stuk groter, wat mogelijke redenen kunnen zijn om niet voor deze controller te kiezen. Daarnaast is de snelheid van de processor en de hoeveelheid geheugen nog steeds aan de lage kant.

**STM32F4 DISCOVERY** De STM32F4 DISCOVERY is de snelste van de tot nu toe genoemde microcontrollers en heeft ook een stuk meer opslag wat hem veel geschikter maakt voor ingewikkeldere software. Deze controller biedt de meeste aantal pinnen en heeft ook het grootste aantal ingebouwde functies. De benodigde functies hangen natuurlijk af van de gebruikte sensoren en actuatoren, maar een groot scala aan mogelijkheden is altijd gunstig. De STM32F4 DISCOVERY heeft daarnaast een accelerometer ingebouwd, wat losse kosten scheelt. Een groot nadeel van al deze functies is dat hij behoorlijk wat ruimte inneemt, nog meer dan de LPCXpresso. Een ander probleem is dat er standaard geen software wordt meegeleverd door de fabrikant waardoor er extra onderzoek vereist is naar de benodigde software. Voor eerste prototypes is dit type misschien minder geschikt. Voor latere versies is de extra snelheid, ruimte en mogelijkheden echter wel een groot voordeel.

Voltage	3,3V (5v tolerant)
Digital I/O	82(+)
Analoge output	20
Analoge input	8
Kloksnelheid	168 MHz
Flash Memory	192kb
SRAM	1 MB

**Raspberry Pi** De Raspberry Pi<sup>6</sup> is de bekendste kleine controller van dit moment, die dit jaar veel in het nieuws is geweest. De naam microcontroller doet de Raspberry Pi eigenlijk tekort, want het is eigenlijk gewoon een mini-computer met als doel om mensen weer meer echt met elektronica te laten omgaan. De Raspberry Pi is een computer die bijna in een luciferdoosje past waar gewoon een volledig besturingssysteem op draait: Linux. Bij deze volledige software horen ook andere zeer redelijke specificaties, terwijl de controller met 35 euro nog wel zeer goed betaalbaar is.

Voltage	5V
Digital I/O	6
Analoge output	0
Analoge input	0
Kloksnelheid	700 MHz
Flash Memory	- (SD-kaart)
SRAM	256 MB

De Raspberry Pi heeft allemaal extra's die normaal gesproken nooit op een microcontroller zal terug vinden, zoals bijvoorbeeld een HDMI-uitvoer om een monitor op aan te sluiten en een jack om geluid af te spelen. Uit het aantal analoge pins (0) en het aantal digitale I/O pins is ook duidelijk af te lezen dat deze Pi eigenlijk niet bedoeld is voor simpele robots, maar veel meer om grotere componenten aan te sturen. De aanzienlijke hogere kloksnelheid, de veel grotere hoeveelheid geheugen en de opslag op een SD-kaart maken het wel mogelijk om veel complexere programma's uit te laten voeren. vergeleken met de andere uitgebreide controllers die hierboven zijn behandeld is de Raspberry Pi ook relatief klein.

De extra mogelijkheden zijn punten die deze Raspberry Pi toch zeker tot de mogelijkheden blijft laten behoren. Er moet echter dan wel een oplossing worden gevonden om analoge invoer en uitvoer te lezen, oftewel er moet een aparte AD-converter worden gekocht. Dit hoeft niet echt een probleem, omdat deze controller ook kosten bespaard met de al ingebouwde mogelijkheden. Het kleine aantal digital I/O poorten blijft dan echter nog steeds een belangrijk nadeel, evenals de relatief grote hoeveelheid stroom die al deze extra's bij elkaar natuurlijk vragen. Wat betreft software is de Raspberry Pi wel weer erg praktisch, omdat er gewoon een besturingssysteem als Arch Linux op kan draaien en er dus ook gewoon met 'standaard' C++ gewerkt kan worden, wat een modulaire oplossing mogelijk maakt. Wel dient hier natuurlijk bij opgemerkt te worden dat het een beetje overdreven is om een volledig besturingssysteem te gebruiken voor zo'n soort project.

### A.1.2 Motoren

Voor de aandrijving van de robot zijn verschillende mogelijkheden. Voor ons is voornamelijk de nauwkeurigheid belangrijk (zie pagina 11), maar de motoren moeten natuurlijk ook voldoende kracht en een aanvaardbare snelheid kunnen leveren (zie pagina 12). Voor de aandrijving zijn diverse onderdelen van belang: de motoren, de motor-controllers en

---

<sup>6</sup>Raspberry Pi. 2012. URL: <http://www.raspberrypi.org/>.

de wielen. Daarnaast is het voor ons bij enkele motoren noodzakelijk om een systeem toe te voegen dat terugkoppelt wat de daadwerkelijk afgelegde afstand is (als deze afstand niet in de motoren zelf nauwkeurig is in te stellen).

Er zijn vier typen elektromotoren die wij mogelijk kunnen gebruiken. Hierbij is voornamelijk onze nauwkeurigheids-eis een belangrijke factor (zie pagina 11). Na deze eis hebben we al verschillende typen weggestreept, waarbij de volgende typen overblijven: DC motoren, geared DC motoren, servo-motoren en stammotoren

**DC-motor** Dit is het meest eenvoudige type motor, simpel gezegd zet je er stroom op en het de motor gaat draaien. Een microcontroller kan bij deze motoren echter niet zorgen voor voldoende vermogen en daarom hebben deze type motoren een motor-controller nodig die deze taak kan uitvoeren. Verder hebben deze motoren geen mechanisme om te kijken hoeveel rotaties er daadwerkelijk zijn afgelegd na het toevoegen van stroom. Daarom is bij deze motoren een terugkoppelings-systeem of een andere wijze van plaatsbepaling gewenst om de nauwkeurigheid te bieden die wij in dit project willen.



**Figuur 9:** DC motor

**Geared DC-motor** Een geared DC motor lijkt erg op een normale DC motor. Het verschil tussen een DC motor en de geared variant is dat hier nog een set tandwielen achter zit waardoor de motor minder hard gaat en meer kracht kan leveren. Het is dus een beter afgestelde versie van de DC motor. Door de verminderde ratio, is de nauwkeurigheid van deze motoren vaak wat hoger.

**Servo-motor** Een servo motor is een verder verbeterde versie van een geared DC motor: deze heeft dus nog meer ingebouwde mogelijkheden die je niet meer zelf hoeft toe te voegen. Zo heeft een servomotor al een ingebouwde motorcontroller, zodat je hem rechtstreeks op je microcontroller kan aansluiten. Een nadeel van deze aanpak is dat je minder controle kan uitoefenen, hoewel het voordeel is dat de configuratie minder tijd kost. Een voordeel van een servo-motor is dat deze vaak compacter zijn en relatief goedkoop door de massaproductie.

Servo motoren hebben naast een motor-controller vaak ook een ingebouwde feedback loop waardoor er nauwkeurige controle is van de beweging van de motoren. Bij de meeste servo's werkt deze feedback loop met een potentiometer, die echter maar maximaal ca. 270 graden te draaien is. Voor onze robot is dat natuurlijk niet handig, omdat onze motoren moeten blijven doordraaien. Daarom hebben we continue servo's nodig die een andere terugkoppelingsssysteem gebruiken (zie pagina 40). Dit systeem is meestal vrij simpel en niet zeer accuraat, maar werkt wel grotendeels automatisch. Opnieuw is de geringere controle een nadeel, maar is het prettig dat alles alvast is geconfigureerd.

Een nadeel is wel dat het aanbod aan servo's met aangepaste feedback-loop niet erg groot is en dat de prijzen daardoor minder gunstig zijn.

**Stap-motor** Een stamotor biedt nauwkeurige aansturing, maar op een heel andere manier dan de hierboven genoemde combinatie van een DC motor met een terugkoppelingsysteem. De stamotor kan namelijk van tevoren al in stappen worden aangestuurd. Aan een stamotor wordt door het toevoegen van stroom aan telkens een andere spoel, het draaiende midden steeds een stapje verder 'getrokken'. Hierdoor kun je dus van te voren al aangeven hoeveel stappen de motor moet zetten. Dit is erg praktisch voor systemen, waar correctie achteraf niet gewenst is. Bij onze robot is het echter niet echt een probleem om achteraf een correctie uit te voeren. Een nadeel van het van te voren sturen van de motor is dat je geen precieze controle hebt over welke stappen daadwerkelijk worden uitgevoerd. Als er namelijk teveel wrijving is, lukt het de spoel niet om de as een stap verder te draaien. Of dit wel of niet gelukt is, is echter achteraf niet te detecteren. Voor ons is dat wel nadelig, omdat er voor onze robot momenten kunnen komen waar er meer kracht noodzakelijk is, maar naar onze inschatting zou dit niet vaak moeten voorkomen (mede omdat voortbeweging over ruwe grond niet nodig is).

Een stap-motor moet nauwkeurig worden aangestuurd om precies het gevraagde aantal stappen te bereiken. Hier is een ingewikkeldere motor-controller bij nodig, dan die een DC motor aanstuurt. Dit is nadelig, omdat deze motor-controllers duurder zijn. Het komt ook vaak voor dat motor controllers twee DC motoren kunnen aansturen of een stap-motor. Wij zouden dan twee motor controllers moeten aanschaffen in plaats van een. Omdat de motor-controller van te voren al de stappen berekent scheelt dat wel weer werk voor de microcontroller. Deze hoeft dan namelijk niet meer achteraf met behulp van een terugkoppelings-sensoren berekeningen te maken over de daadwerkelijk afgelegde rotatie.

### A.1.3 Motorcontrollers

Een micro-controller heeft zelf standaard voor de meeste motoren niet genoeg stroom beschikbaar en kan ook niet de vereiste precisie leveren om een motor aan te sturen. Daarom heb je een zogenaamde motorcontroller nodig. Hiervan zijn vele verschillende versies, die je moet gebruiken voor verschillende motoren. Ten eerste zijn er normale controllers die zijn gemaakt om een of meer (geared) DC-motoren aan te sturen. Daarnaast zijn er ook nog speciale controllers voor stamotoren. Vaak bieden controllers ook ruimte voor allebei, maar dan voor minder stamotoren dan DC-motoren. Voor servo- motoren is geen controller nodig omdat die al zelf een controller ingebouwd hebben. Bij de Arduino is een motor controller beschikbaar in de vorm van een shield, waardoor deze zeer eenvoudig met de Arduino kan communiceren (zie pagina 32).

### A.1.4 Wielen

Voor de wielen zijn er vier opties: normale wielen, omni-directionele wielen, mecanum wielen en ball-casters (al zijn deze laatste strict gezien niet helemaal wielen).

**Gewone wielen** Deze zijn meestal van rubber en zijn beschikbaar in vele verschillende groottes en profielen. De grootte maakt uit voor de verhouding van de snelheid en de

kracht van de wielen. Voor ons zijn relatief kleine wielen het gunstigst, omdat deze het meeste controle geven. Een nadeel is dat de snelheid dan wel minder is, maar deze is voor ons minder belangrijk dan de nauwkeurigheid.

**Omni-directionele wielen** Omni-directionele wielen zijn eigenlijk gewone wielen met kleine, in het grote wiel verwerkte, wieljes die loodrecht op het grote wiel staan. Hierdoor kunnen deze omni-directionele wielen ook bewegen in een richting die loodrecht staat op het grote wiel, zonder veel wrijving te ondervinden. Deze wielen kunnen echter niet zelf de kracht leveren om deze zijdelinge beweging te maken. Daarom wordt er meestal een vierkante opstelling gebruikt bij omni-directionele wielen, waarbij twee wielen zorgen voor de horizontale voortbeweging en twee andere voor de verticale.

Een groot voordeel van deze wielen is, dat er een zeer goede wendbaarheid mee bereikt kan worden, wat we graag willen (zie pagina 11). Een groot nadeel is echter dat de nauwkeurigheid en dus de positiebepaling minder wordt. Dit komt omdat de weerstand van de omni-directionele wielen wel afneemt, maar nog altijd stukken groter is dan bij een wiel dat gewoon recht staat.

**Mecanum wielen** Mecanum wielen hebben grofweg hetzelfde doel als omni-directionele wielen, een grote wendbaarheid verkrijgen. De mecanum wielen bereiken dat door de wielen een beetje schuin te laten lopen. Hierdoor hebben alle wielen wel de voorkeur om naar voren te gaan, maar als bepaalde wielen worden stopgezet dan gaat het mecanum wiel automatisch een zijdelinge beweging maken. Met dit systeem is het dan mogelijk om een zeer grote wendbaarheid te verkrijgen zonder dat er wielen zijn die gedraaid hoeven te worden, wat veel praktischer kan zijn.

Een groot nadeel is echter nog steeds dat de nauwkeurigheid door meer wrijving afneemt. Er is ook nog een ander probleem: voor een goede besturing is er onafhankelijke besturing nodig van vier verschillende wielen. En zouden dan dus vier motoren nodig zijn en dat is uiteraard duurder dan twee motoren. Hierdoor is er ook niet de mogelijkheid om drie wielen (een voor en twee achter) te gebruiken in plaats van vier.



**Figuur 10:** Omni-directioneel wiel



**Figuur 11:** Mecanumwheel

**Ball-casters** Dit zijn eigenlijk geen wielen en ze kunnen ook niet worden aangestuurd door motoren. Ball-casters bestaan uit een balletje die goed kan draaien. Dit zorgt ervoor dat er grote wendbaarheid is, als je deze ball-casters als voorwiel gebruikt. Deze ball-casters kunnen dan zelf niet bewegen, maar bewegen zeer makkelijk mee met de beweging die door de achterwielen wordt ingezet. Het grote voordeel van het gebruik van een ball-casters is dat deze relatief goedkoop zijn en geen extra onderdelen nodig hebben.

### A.1.5 Communicatie

Voor de overdracht van de opdrachten naar de robot zijn er meerdere mogelijkheden. Deze zijn grofweg in te delen in draadloze en niet-draadloze (of semi-draadloos als je een SD-kaart gebruikt, die je steeds handmatig naar de robot moet verplaatsen).

#### Niet-draadloos

**Kabel** Het gebruik van een kabel is gebruiksvriendelijk en zeer eenvoudig te gebruiken: er is dan een constante verbinding met een andere apparaat (meestal een computer). Een voordeel is dat je destroomvoorziening via de computer kan leveren en ook een deel van de berekeningen door de computer kan laten uitvoeren, wat de robot eenvoudiger maakt. Deze manier is ook het makkelijkst om te implementeren, omdat de meeste controllers dit standaard ondersteunen. Een erg groot nadeel is natuurlijk dat de robot dan niet over een grote afstand kan bewegen en de robot daarnaast niet echt zelfstandig is.

**SD-kaart** Wanneer we gebruik maken van een SD kaart is de robot juist wel geheel onafhankelijk. Deze methode is daarnaast vrij eenvoudig te gebruiken, omdat er geen externe apparatuur voor nodig is. Het grote nadeel is dat er geen real-time communicatie mogelijk is, je moet de SD-kaart namelijk zelf in en uit de robot halen. Hier lever je iets mee in bij het gebruiksgemak, maar vooral op de snelheid van de communicatie. Een andere nadeel van deze methode is dat er voor vrijwel elke microcontroller extra bordjes nodig zijn om deze SD-kaarten te lezen.

**Draadloos** Wanneer je gebruik maakt van een draadloos protocol is het veel eenvoudiger om opdrachten naar de robot te sturen en je hebt ook nog eens realtime feedback van de robot. Draadloze communicatie is wel veel lastiger te implementeren, wat een nadeel is.

Er zijn verschillende mogelijkheden voor draadloze communicatie, waarbij de volgende de belangrijkste zijn:

**434 MHZ** Dit is de goedkoopste methode, die werkt via radio transmitters. Deze methode is voornamelijk zo goedkoop, omdat er geen achterliggende hardware en software voor is geschreven: als we deze manier van communicatie gebruiken moeten we die dus ook helemaal zelf maken. Behalve de tijd om een goed protocol te implementeren kost dit ook rekentijd van de processor (omdat er geen losse controller is meegeleverd die dit doet). Een ander nadeel van deze methode is dat je altijd minimaal twee transmitters nodig hebt, omdat deze techniek niet gebruikt wordt in bijv. mobieltjes en computers. Een voordeel van deze methode is wel dat het een behoorlijk grote afstand biedt, terwijl de prijs toch erg laag is.

**Bluetooth** Bluetooth is een standaardprotocol voor de communicatie op korte afstand, die voor allerlei soorten toepassingen gebruikt wordt. Bij deze manier van communiceren is alles in hardware en software zo afgesteld dat communicatie zo makkelijk mogelijk is. De totale oplossing is daarom ook vrij duur. De afstand die kan worden afgelegd met Bluetooth is ook relatief kort en meestal korter dan bij de 434 MHZ. Een voordeel van deze methode is dat dit protocol ook in mobieljes en sommige computers wordt geïmplementeerd waardoor een tweede module niet noodzakelijk is.

**XBee** Dit is een simpelere en wat goedkopere variant van Bluetooth die meer voor robotica is bedoeld. Deze zijn dus ook beter aangepast om in robotica gebruikt te worden en bieden ook eenvoudige communicatie met computers maar ook tussen meerdere robots. Het protocol is al geïmplementeerd beschikbaar voor veel verschillende platformen en programmeertalen. Ondanks dat het goedkoper is dan Bluetooth is deze optie meestal alsnog vrij duur, zeker omdat er weer twee modules nodig zijn.

**WiFi** WiFi is het meest gebruikte draadloze protocol, maar het wordt standaard niet gebruikt voor communicatie tussen robots. Het heeft wel enkele voordeelen. Er is bijvoorbeeld maar één module nodig, omdat vrijwel alle computers en mobieljes in staat zijn om draadloze verbinding te maken via WiFi. Daarnaast biedt WiFi een redelijke verre afstand, zeker als je een sterke ontvanger gebruikt (zoals de meeste routers). Verder biedt het ook mogelijkheden om de robot via internet overal waar je wilt aan te sturen. Deze methode heeft bij elkaar gezien zeker de meeste mogelijkheden, maar is helaas niet de goedkoopste. Meestal is het echter wel goedkoper dan Bluetooth en XBee's omdat er maar een module benodigd is. Een nadeel is wel dat je wel zelf de configuratie en het protocol moet implementeren om WiFi geschikt te laten maken voor robotica, omdat dit niet standaard is.

## A.2 Sensoren

### A.2.1 Terugkoppeling

Voor het goed werken van de motoren hebben we vaak terugkoppelingsystemen nodig (zie pagina 35), omdat deze meestal niet zijn ingebouwd en we ze wel nodig hebben om voldoende nauwkeurigheid te bieden. Bij een stappenmotor is deze feedback niet nodig (omdat de stappen van te voren nauwkeurig kunnen worden bepaald) en bij een servo-motor is dit al ingebouwd. Je hebt verschillende typen feedback-systemen die we hieronder kort behandelen.

**Mechanische terugkoppelings** Mechanische terugkoppeling werkt doormiddel van een schijf met gaten op vaste afstand, waarin pinnetjes vallen. Elk zet gaten staat voor een binair code die kan worden omgezet naar een bepaalde mate van rotatie. Deze methode is vrij simpel, maar mist wel wat nauwkeurigheid omdat er maar een beperkt aantal gaten te maken is.

**Optische terugkoppeling** Een optische terugkoppelingsysteem werkt doormiddel van een stuk doorschijnend materiaal met gedeeltes die niet doorschijnend zijn gemaakt. Met behulp van een lichtbron en een lichtontvanger meet hij hoe ver hij gedraaid is. In

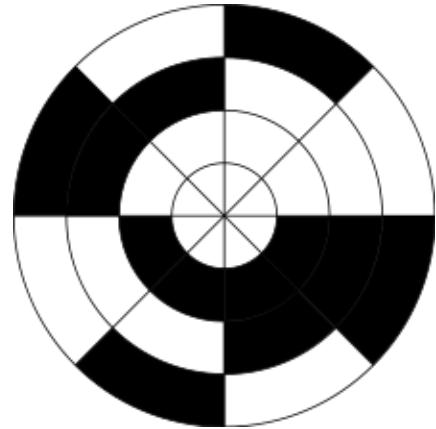
deze optische encoder zijn vaak diverse vlakken aangebracht zodat er ook een soort binaire code ontstaat net als bij een mechanische encoder. Afhankelijk van de gelezen code kan deze dan door de software worden vertaald in een bepaald vlak van de encoder en dit vlak correspondeert dan weer met een bepaalde mate van draaiing. Deze methode is nauwkeuriger, omdat er meerdere vlakken gemaakt kunnen worden en er tevens minder wrijving is. Deze manier is wel wat duurder, omdat er nauwkeurige lichtsensoren benodigd zijn voor allebei de motoren.

### A.2.2 Anti-botsingssystemen

Om geen botsingen te veroorzaken en obstakels van te voren te kunnen detecteren zijn er verschillende systemen mogelijk. De twee belangrijkste systemen zijn :

**Sonar** Een sonar maakt gebruik van ultrasoon geluid om de afstand tussen objecten te meten. Deze methode kan met vrij hoge nauwkeurigheid ook langere afstanden detecteren (tot rond de acht meter). Het nadeel wel is dat een sonar minder goed is in het detecteren van zeer korte afstanden en deze sensor ook vrij prijzig is.

**IR** Infraroodstraling gebruiken om de afstand te meten. Deze methode werkt zeer goed voor de korte afstand, maar is op iets langere afstanden wel vele malen onnauwkeuriger dan een sonar, waardoor deze bij een afstand van meer dan een meter eigenlijk niet meer gebruikt kan worden. Het grote voordeel ten opzichte van een sonar, behalve de goede detectie op korte afstand, is de prijs van een IR-sensor, die een stuk lager is.



**Figuur 12:** Tekening van een optische encoder schijf

### A.2.3 Plaatsbepaling

Om de plaats te bepalen kunnen we gebruik maken van verschillende mogelijkheden. Als eerste kun je natuurlijk gebruik maken van een terugkoppeling-systeem en met behulp van de afstanden en rotaties dan een locatie uitrekenen. Hoewel deze methode wel relatief eenvoudig te implementeren is, biedt het niet de benodigde hoge nauwkeurigheid op langere termijn. Daarom hebben we onderzoek gedaan naar diverse andere methoden om de plaats te bepalen.

De plaats kan ten eerste worden bepaald met behulp van speciale bakens. Deze methode is vaak zeer nauwkeurig, maar vereist wel dat de bakens van te voren worden geplaatst. De volgende methoden zijn hierbij mogelijk:

1. Gebruik maken van een gekleurde grond, waarbij een lichtsensor vlakken kan herkennen.
2. Gebruik maken van een CMOS sensor (een camera). Dit biedt een hoge nauwkeurigheid, maar is wel een stuk duurder en veel ingewikkelder om te programmeren.

In plaats daarvan kunnen we ook gebruik maken van systemen die de plaats kunnen bepalen zonder waarneembare punten:

1. Positie bepalen door de beweging via accelerometers. Dit is niet zeer nauwkeurig, omdat een accelerometers enkel versnelling kan meten en houdt daarnaast geen rekening met wrijving. De methode is wel vrij eenvoudig te implementeren en is redelijk goedkoop.
2. Draaing bepalen doormiddel van een kompas, wat meestal zeer nauwkeurig is. Kompassen zijn meestal niet erg goedkoop, al is het mogelijk om zelf een magnetometer (die het aardmagnetisch veld meet) en een accelerometers te combineren om een vrij nauwkeurig kompas te krijgen.
3. Mini-GPS, een systeem dat werkt met een frame dat werkt als een GPS maar dan op een veel kleinere schaal. Deze methode is zeer nauwkeurig, maar is ook meestal zeer duur.

## A.3 Software

De software is het andere belangrijke deel naast de totale hardware (aansturing, sensoren en actuatoren) en moet ervoor zorgen dat alle onderdelen goed samenwerken. In ons geval bestaat de software uit verschillende onderdelen: als eerste heb je het besturingssysteem en de verschillende onderdelen van dit besturingssysteem die de basis van de software vormen. Daarnaast heb je het door onszelf geschreven deel.

Het eigen deel van de software is voor ons natuurlijk het belangrijkst en moet ervoor gaan zorgen dat de robot de taken gaat uitvoeren die wij graag willen dat de robot gaat uitvoeren. Hiervoor is voornamelijk de gebruikte programmeertaal belangrijk, maar deze hangt erg af van het besturingssysteem

Het besturingssysteem hangt op zijn beurt weer erg af van de gebruikte microcontroller (zie pagina 32). De microcontroller moet namelijk overweg kunnen met de software, want anders kunnen we deze uiteraard niet gebruiken. Daarom hangt het onderdeel software sterk samen met de te kiezen microcontroller.

### A.3.1 Besturingssysteem

Het besturingssysteem is het hart van de software en biedt de basale operaties om alle andere software te kunnen laten werken. Op sommige microcontrollers draaien alleen hele eenvoudige besturingssystemen, zoals bij de Arduino, op andere, zoals de Raspberry Pi kunnen volledige OS'en draaien die ook op computers werken, zoals Linux, draaien. Hieronder gaan we kort in op de belangrijkste besturingssystemen bij onze onderzochte microncontrollers:

**Fabrikantsomgeving** De fabrikant levert meestal een standaardbesturingssysteem bij. Bij de Arduino is deze weliswaar vrij eenvoudig, maar voorzien van grote en goed gedocumenteerde bibliotheken aan code, waardoor er met de Arduino zeer eenvoudig te werken valt. Bij andere microcontroller zoals de NXP en de STM32 F4DISCOVERY levert de fabrikant geen standaard-bibliotheken mee en is het gebruik van een ander besturingssysteem dus eigenlijk noodzakelijk.

**Linux** Linux is het grote OS op de wereld naast Windows en Mac OSX. In tegenstelling tot deze is het echter gratis en opensource. Dit OS is niet bedoeld voor microcontrollers, maar voor gewone computers. Enkel de Raspberry Pi kan goed omgaan met Linux, voor de andere onderzochte controllers is het eigenlijk onmogelijk om dit OS te gebruiken (omdat het OS te groot en te uitgebreid is). Het grote voordeel van Linux als OS, is dat er heel veel verschillende bibliotheken en programmeertalen voor beschikbaar zijn, waardoor het maken van code relatief eenvoudig is. Omdat Linux echter niet standaard bedoeld is voor microcontrollers is het daarvoor ook niet afgestemd, wat nadelig is voor het maken van de software.

**ChibiOS** ChibiOS is een redelijk nieuwe besturingssysteem voor microcontroller. Het is een zogenaamde ‘realtime operating system’ (RTOS), dat taken op een andere manier indeelt dan standaard besturingssystemen (zoals Linux). In plaats van met prioriteit te werken, maakt de indeling van de taken gebruik van tijdsduur. Als voorbeeld taak X elke seconde iets moet doen kan er worden gegarandeerd dat dit ook elke seconde gebeurt. Een RTOS is voor het gebruik een groot pluspunt, omdat taken inderdaad vaak tijdsafhankelijk zijn. Een ander groot voordeel is dat ChibiOS HAL heeft. HAL staat voor Hardware Abstraction Layer en dat is een manier om het makkelijker te maken om allerlei soorten verschillende apparaten te ondersteunen. Waar HAL voor zorgt is dat wanneer je bijv. een lampje aan wil zetten, je alleen hoeft te ‘zeggen’ dat je een lampje aan wil hebben. De driver, die per device verschilt, regelt dan dat het lampje aangaat op de manier die specifiek is voor dat apparaat. Deze opzet in modules past zeer goed bij onze modulaire eis. (zie pagina 11).

## B Motorkracht en -snelheid

Voor het aansturen van de robot hebben we uiteraard motoren nodig. Motoren hebben grofweg vier belangrijke variabelen waarop ze verschillen

1. Kracht(moment)
2. Snelheid
3. Prijs
4. Onderliggende aansturing

De laatste twee items zijn hier eigenlijk het minst interessant, omdat we deze kunnen aanpassen naar de eerste twee. We wilden echter wel graag weten hoeveel kracht en snelheid benodigd waren om aan onze eisen te voldoen (zie pagina 10). De snelheid van een motor wordt meestal gegeven in RPM (rotations per minute) en de kracht in newton-meter. Volgens onze eisen wilden we dat de robot minstens 5 km/h kon rijden en dat er voldoende kracht is om een helling van 20 graden op te komen. Met voldoende kracht bedoelen we dat de robot op deze helling zijn maximale snelheid in ongeveer 1-2 seconden kan bereiken.

De snelheid van de robot is gelijk aan:

$$v = 2 \times \pi \times \frac{RPM}{60} \times radius$$

Deze vergelijking bestaat gewoon uit de omtrek van de cirkel keer het aantal rotaties per seconde: dit is dus ook de snelheid in meter per seconde. Het is te zien dat behalve de RPM van de motor ook de diameter van de wielen hier een rol speelt: de robot gaat sneller bij grotere wielen. Helaas is het niet handig om dan maar hele grote wielen erop te zetten, omdat de kracht die de motor dan kan leveren afneemt.

De kracht die moet worden opgewekt wordt bepaald door de versnelling van de robot volgens de tweede wet van Newton:

$$F = m \times a$$

$$a = \frac{F}{m}$$

Als er dan ook nog een helling is dan is de resulterende kracht groter, omdat de zwaartekracht dan ook nog gecompenseerd moet worden. Dan ontstaat de volgende formule:

$$a = \frac{F - 9,81 \times \sin(hoek)}{m}$$

De formules die we gebruiken voor het berekenen van de motoren zijn dus:

$$v = 2 \times \pi \times (RPM \times 60) \times radius$$

$$a = \frac{F - 9,81 \times \sin(hoek)}{m}$$

Voor deze formules is het handig om eerst even om te rekenen naar meter per seconde:  $5 \text{ km/h} \approx 1,4 \text{ m/s}$ . Om deze snelheid in ongeveer anderhalve seconde vanaf de start te bereiken is dus een versnelling nodig van ongeveer  $1 \text{ m/s}^2$ .

Voorlopig gaan we ervan uit dat de robot maximaal één kilogram gaat wegen en dat de radius van de wielen ongeveer  $25\text{mm} = 0,025\text{m}$  is. Invullen in de formules levert dan de volgende uitkomsten.

$$1,4 = 2 \times \pi \times \frac{RPM}{60} \times 0,025$$

$$RPM = \frac{1,4 \times 60}{2 \times \pi \times 0,025}$$

$$RPM \approx 535$$

$$1 = \frac{F - 9,81 \times \sin(20)}{1}$$

$$F = 1 + 9,81 \times \sin(20)$$

$$F \approx 4,355$$

Deze laatste waarde moet nog worden omgerekend in een krachtmoment, volgens de volgende formule

$$M = F \times 2 \times \pi \times radius$$

$$M = 4,355 \times 0,025$$

$$M \approx 0,11$$

De motoren moeten dus in staat zijn om ongeveer 535 rotaties per minuut te maken (RPM) en 0,11 newton-meter aan kracht kunnen opwekken!<sup>7</sup>

---

<sup>7</sup>Robot-dynamica. 2012. URL: [http://www.societyofrobots.com/mechanics\\_dynamics.shtml](http://www.societyofrobots.com/mechanics_dynamics.shtml).

# Deel VIII

## Bijlage: Software

### C Protocol

Voor de communicatie met de robot wordt gebruik gemaakt van wifi, met een eigen protocol. Hierdoor is het nodig om verbinding te maken met de robot via het verkregen ip van de router op poort 80. De verbinding kan het best worden gemaakt via telnet, volgens het volgende commando:

**telnet IP 80**

De communicatie tussen de robot en de zender verloopt via commando's. Let op dat er geen nieuw commando wordt gestuurd, voordat de vorige is verwerkt. Een commando bestaat uit een regel die eindigt met \r \n. Een regel die begint met drie sterretjes (\*\*\*) is extra informatie (commentaar) en moet worden genegeerd voor basale communicatie. Als de robot zich in API-mode kun je testen door de status \*HELLO\* te zenden, de robot zou met \*HELLO\* antwoorden als dat zo is. Het protocol geeft altijd één van de volgende antwoorden terug, behalve als anders wordt aangegeven:

- **\*HELLO\***: wordt teruggestuurd als er \*HELLO\* als status wordt verzonden.
- **\*OK\***: de opdracht is begrepen en uitgevoerd
- **\*OVERFLOW\***: de opdracht was te lang en wordt overgeslagen
- **\*ERROR\***: de opdracht is om onbekende redenen niet uitgevoerd

De robot ondersteunt de volgende commando's:

#### C.1 Algemeen

- **version**: geeft de protocolversie terug.
- **lights *id***: laad het licht met het gegeven ID branden.
- **close**: de verbinding met de robot wordt gesloten

#### C.2 Beweging

- **forward *sec***: zet de motoren voor *sec* seconden op vooruit.
- **backward *sec***: zet de motoren voor *sec* seconden op achteruit.
- **turnLeft *sec***: zet de rechtermotoren voor *sec* seconden op vooruit.
- **turnRight *sec***: zet de linkermotoren voor *sec* seconden op vooruit.

### C.3 Sensoren

- **sonar 0/1:** zet het uitlezen van de sonar respectievelijk aan of uit.
- **readSonar:** geeft de waarde van de sonar terug.
- **heading 0/1:** zet de sonar respectievelijk aan of uit
- **readHeading:** geeft de waarde van de heading terug.

## D GUI

Voor de communicatie met de robot kan ook gebruik worden gemaakt van een GUI. Deze GUI bevat een implementatie van het protocol en zorgt voor een grafische schil om het protocol heen. De GUI is bij de code hieronder terug te vinden.

## E Code

Onze software bestaat uit verschillende services en scripts. De totale hoeveelheid aan, momenteel gebruikte, zelfgeschreven code ligt zo rond de 1500 regels, die niet goed in dit verslag was in te voegen. De geprogrammeerde software is daarnaast nog verre van af. De momenteel gebruikte code is online terug te vinden.<sup>8</sup>

---

<sup>8</sup>Software robot. URL: <https://github.com/Koensw/Robot-PWS>.

# Deel IX

## Bijlage: Overigen

### F Opzet project

Het profielwerkstuk is het grootste onderzoek dat wij tot nu toe hebben uitgevoerd. Wij wilden beide daarom al van begin af aan een goede opzet hebben. Daarom hebben we eerst gediscussieerd over de manier waarop we het profielwerkstuk gingen aanpakken. We hebben hierbij besloten welke software en andere ‘gereedschappen’ we wilden gebruiken als ondersteuning voor het maken van een profielwerkstuk. Hieronder lichten we deze kort toe:

#### F.1 L<sup>A</sup>T<sub>E</sub>X

Traditioneel gebruikten we o.a. veel Word voor het schrijven van verslagen, maar hier zijn een aantal belangrijke nadelen. Voor grote projecten vonden we het vrij lastig om het document onder controle te houden en ook waren er vrij weinig uitbreidingen beschikbaar. Wij wilden namelijk ook o.a. stukken code invoeren, iets dat standaard niet goed wordt ondersteund door Word. Een ander belangrijk nadeel aan het gebruiken van Word is de compatibiliteit tussen verschillende versies, dit was voornamelijk vervelend omdat we allebei verschillende besturingssystemen gebruikten. Uit eerdere ervaringen wisten we al dat dit vaak tot vervelende problemen kon leiden.

Wij hebben daarom na onderzoek besloten om ons profielwerkstuk te schrijven in L<sup>A</sup>T<sub>E</sub>X.<sup>9</sup> Dit is een markup-taal, net als bijvoorbeeld HTML, die speciaal bedoeld is voor het schrijven van grote wetenschappelijke teksten. Het wordt daarom ook veel gebruikt in deze sector. Het heeft niet de nadelen die hierboven genoemd waren en heeft ook nog enkele andere voordelen. Je kunt bijvoorbeeld de opmaak van bestanden heel makkelijk later wijzigen als het je niet bevalt. Standaard wordt de opmaak echter vaak goed ingeschat door L<sup>A</sup>T<sub>E</sub>X, waardoor je je tijd gewoon kan gebruiken voor het schrijven van teksten. Het enige nadeel was dat we allebei wat tijd moesten stoppen in het leren van de syntax, maar deze tijd hebben we uiteindelijk ruimschoots teruggewonnen.

#### F.2 GIT

We hebben gekozen om gebruikt te maken van een versiebeheer-systeem: GIT.<sup>10</sup> Dit systeem biedt diverse mogelijkheden om het maken van documenten (vooral met meer mensen) te ondersteunen. Zo maakt GIT het mogelijk om apart aan documenten te werken en elkaars wijzigingen later makkelijk te combineren. Dit werkt ook erg makkelijk tussen verschillende computers (met verschillende besturingssystemen).

GIT biedt ook geadvanceerdere mogelijkheden voor verschillende versies van documenten. Hierdoor kunnen teksten uit eerdere versie makkelijk worden teruggelezen en eventueel worden teruggehaald. Dit systeem maakt het opnieuw voor ons mogelijk om

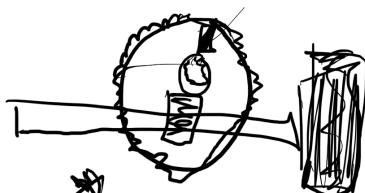
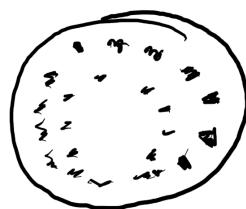
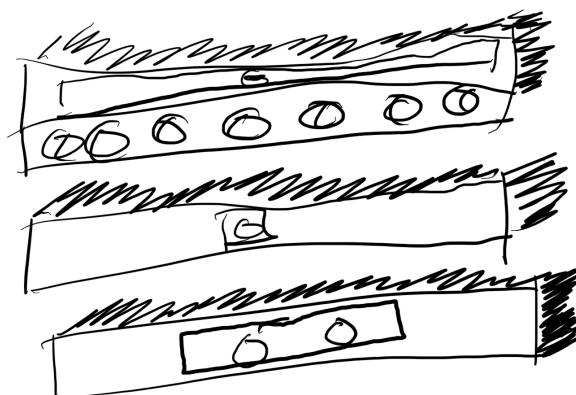
---

<sup>9</sup>Latex. 2010. URL: <http://www.latex-project.org/>.

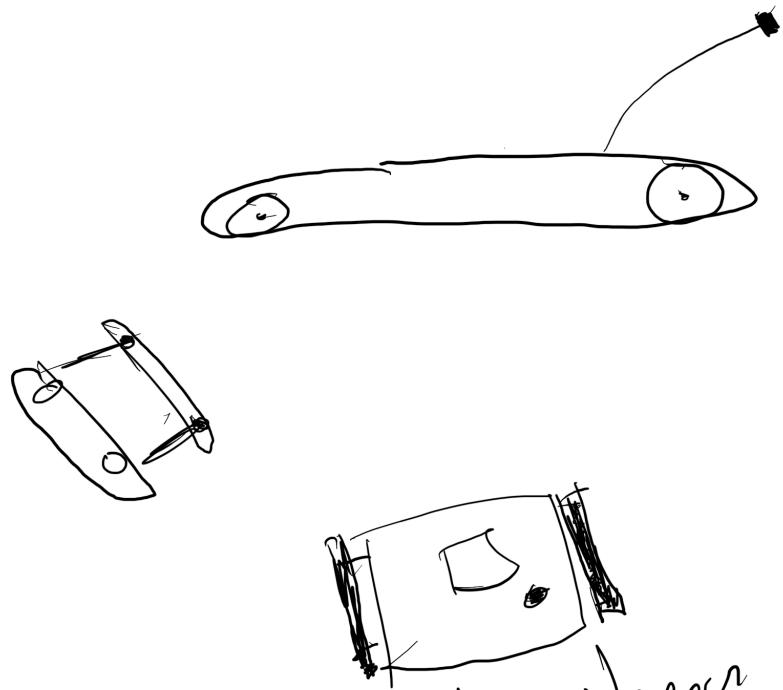
<sup>10</sup>Git. 2012. URL: <http://git-scm.com/>.

ons te richten op het maken van de teksten, zonder tijd te verliezen aan het bijhouden van de verschillende versies.

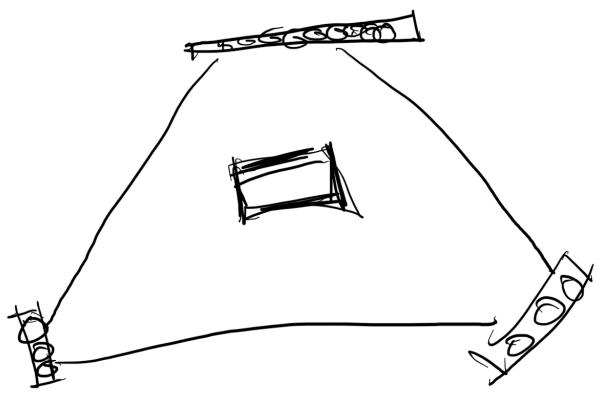
## G Schetsen



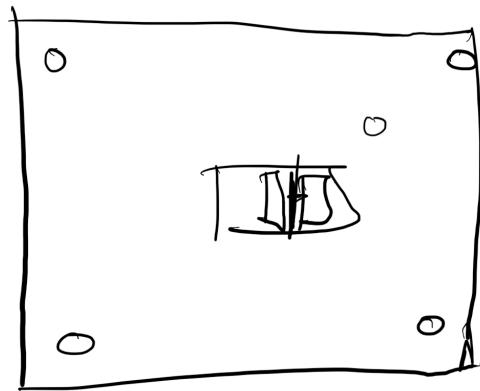
End <sup>van</sup> el. draai per  $\frac{1}{50}$  1 keer  
rond.



- + houtblokje droogdoen.
- + ijzeren en doek
- ijzerdraad (ring)
- ijzerdraad (ring)
- + meer takken.



- + draai voor weerstand
- - inge wiberde aan te stuur en



\* ~~use kleef~~

maken  
draaien

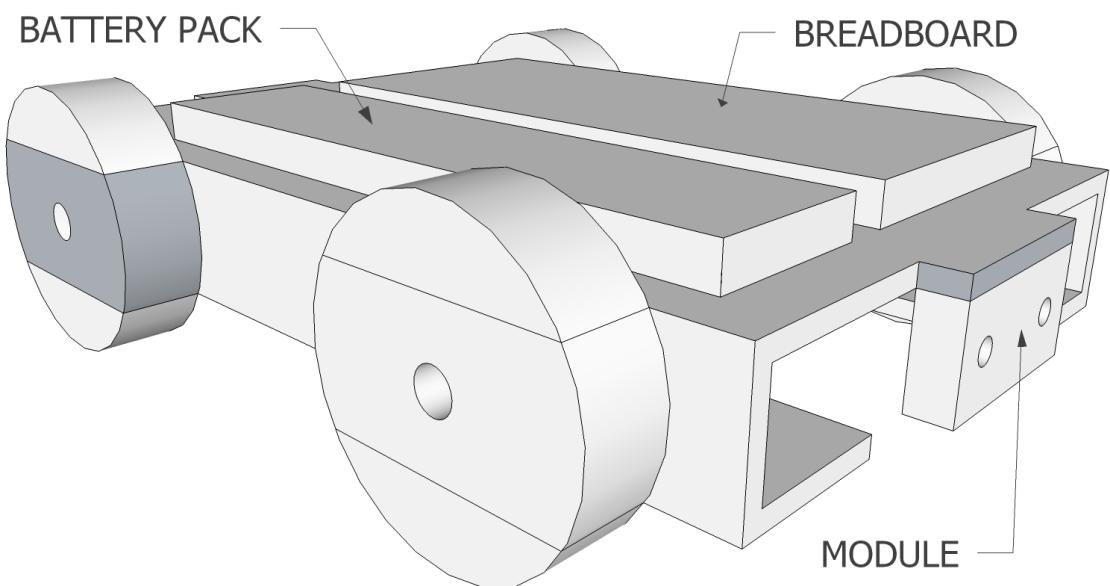
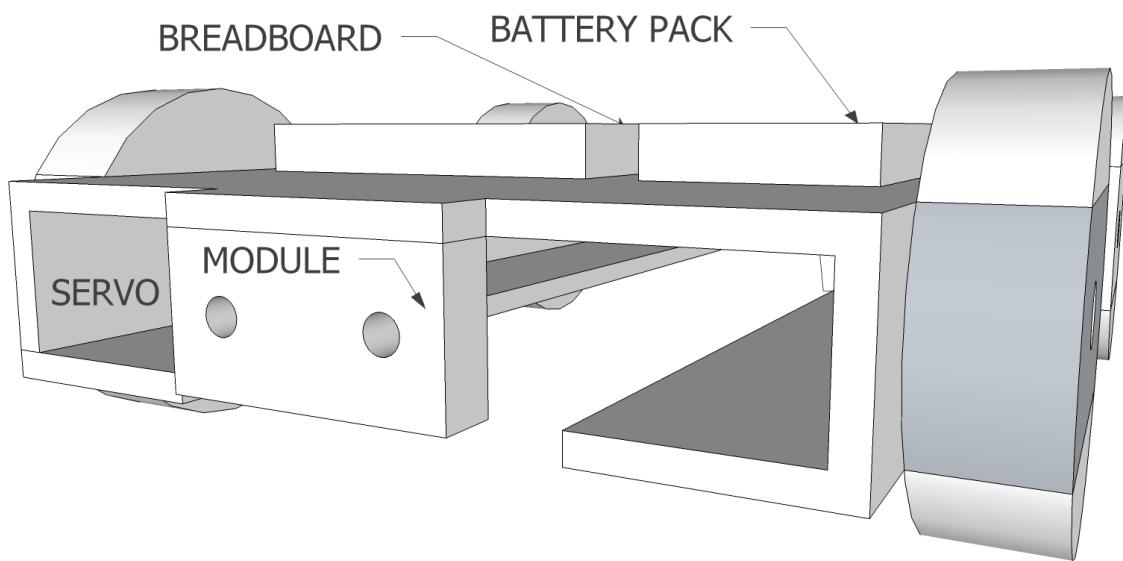
\* kneden  
weer efficiënt

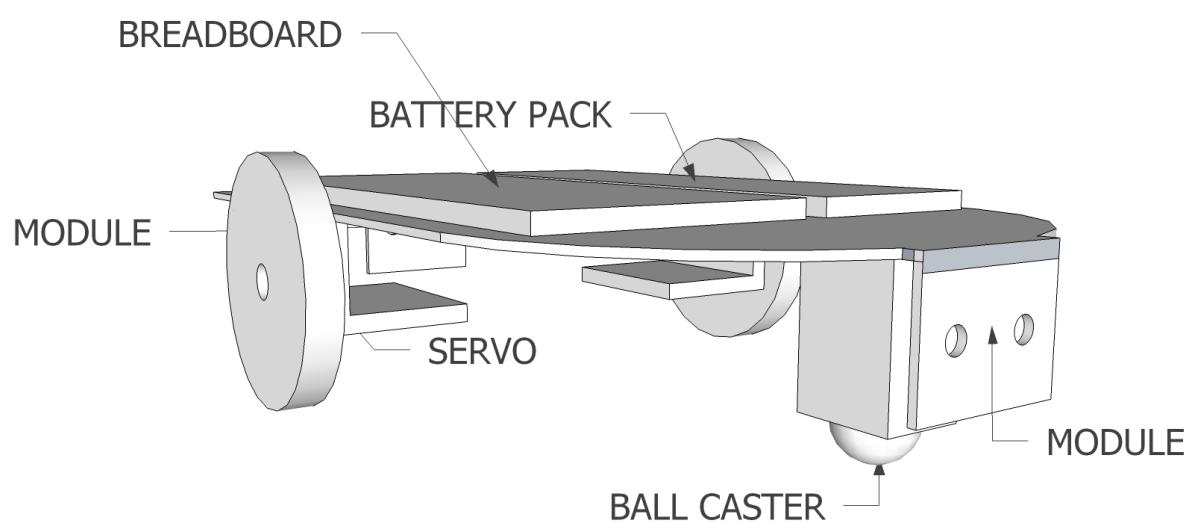
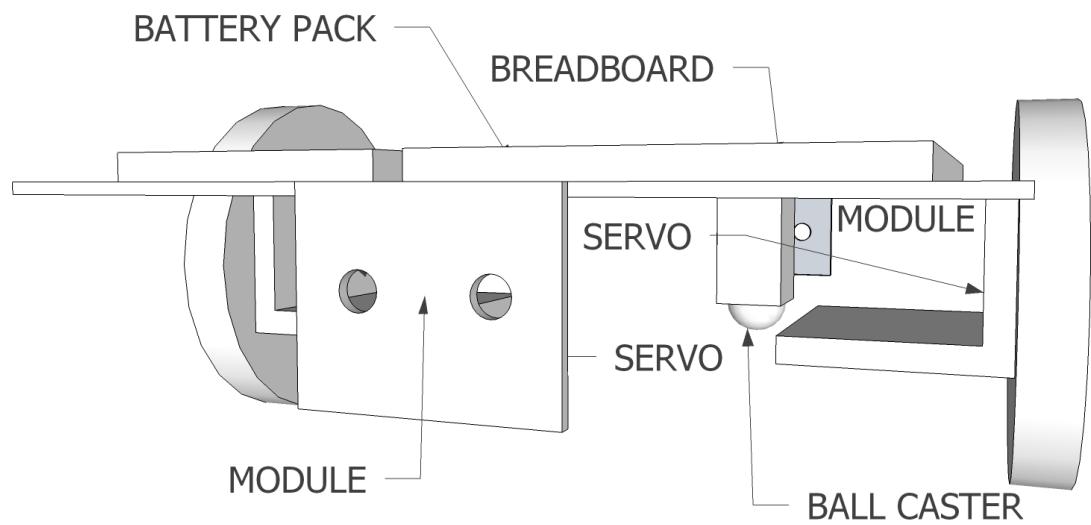


2 wachten in het midden,

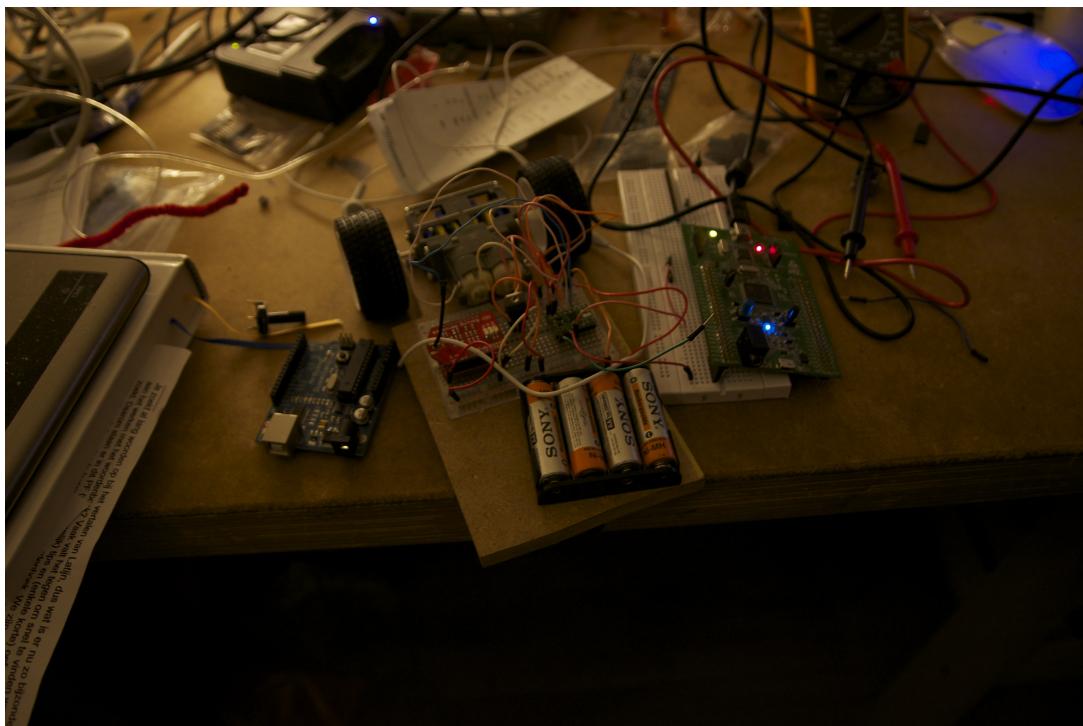
deze kunnen 360

~~360~~ draaien

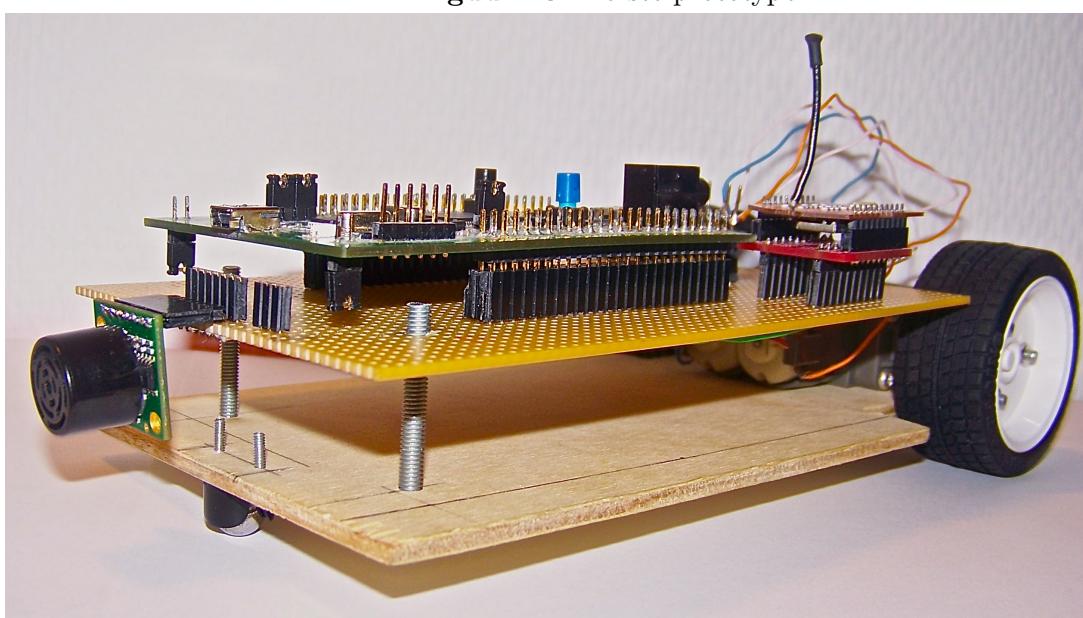




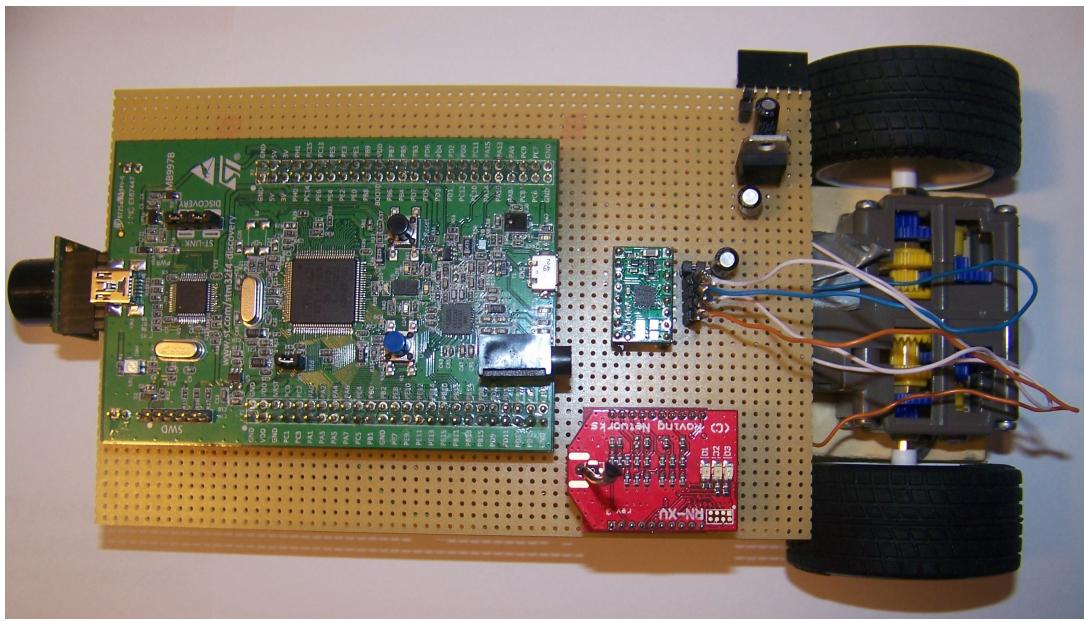
## H Foto's



Figuur 13: Eerste prototype



Figuur 14: Zijkant uiteindelijke robot



**Figuur 15:** Bovenaanzicht uiteindelijke robot

## I Rekening materialen

Onderdeel	Prijs	Aantal	Totaal
Voltage Regulator 7805 (+5V)	1,25	1	1,25
Voltage Regulator - 3.3V	1,95	1	1,95
Tamiya 70111 Sports Tire Set (2 tires)	5,98	1	5,98
Tamiya 70168 Double Gearbox Kit	10,15	1	10,15
Pololu Ball Caster with 3/8" Metal Ball	2,42	1	2,42
DRV8833 Dual Motor Driver Carrier	5,95	1	5,95
Triple Axis Magnetometer Breakout - HMC5883L	13,75	1	13,75
IR Reflectance Sensor - QRE1113	1,95	2	3,90
Ultrasonic Range Finder - Maxbotix LV-EZ1	24,25	1	24,25
400 Tie Points Breadboard - Crystal	4,45	1	4,45
Logic Level Converter	1,75	1	1,75
RN-XV WiFly Module	32,95	1	32,95
Battery Holder - 4xAA Square	1,65	1	1,65
2-AA Battery Holder - Back-to-Back	1,00	1	1,00
FTDI Basic Breakout - 5V	12,95	2	25,90

## Lijst van figuren

1	Een speelgoed robot . . . . .	7
2	Voorbeeld microcontroller . . . . .	10
3	Breadboard . . . . .	12
4	Technische tekening van STM32F4 DISCOVERY . . . . .	14
5	Sonar . . . . .	17
6	Wifly . . . . .	21
7	Arduino Duemilanove . . . . .	33
8	LPCXpresso . . . . .	34
9	DC motor . . . . .	36
10	Omni-directioneel wiel . . . . .	38
11	Mecanumwiel . . . . .	38
12	Tekening van een optische encoder schijf . . . . .	41
13	Eerste prototype . . . . .	54
14	Zijkant uiteindelijke robot . . . . .	54
15	Bovenaanzicht uiteindelijke robot . . . . .	55

## Lijst van tabellen

1	Controllers . . . . .	14
2	Motoren . . . . .	15
3	Wielen . . . . .	15
4	Communicatie-modules . . . . .	15
5	Terugkoppelingsensoren . . . . .	16
6	Detectiesensoren . . . . .	16
7	Plaatsbepalingssensoren . . . . .	16
8	Frames . . . . .	17
9	Code-platform . . . . .	17
10	Specificaties Arduino Duemilanove . . . . .	33
11	Specificaties van LPCXpresso . . . . .	34

## J Bibliografie

- [1] *Arduino*. 2012. URL: <http://arduino.cc/en/>.
- [2] *Chibios HAL functies*. URL: [http://chibios.sourceforge.net/docs/kernel\\_avr\\_gcc\\_rm/index.html](http://chibios.sourceforge.net/docs/kernel_avr_gcc_rm/index.html).
- [3] *Chibios kernel functies*. URL: [http://chibios.sourceforge.net/docs/kernel\\_avr\\_gcc\\_rm/index.html](http://chibios.sourceforge.net/docs/kernel_avr_gcc_rm/index.html).
- [4] *Datasheet van STM32 F4 Discovery*. URL: [http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data\\_brief/DM00037955.pdf](http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00037955.pdf).
- [5] *Datasheet van Wifly*. URL: [http://www.rovingnetworks.com/resources/download/16/RN\\_XV](http://www.rovingnetworks.com/resources/download/16/RN_XV).
- [6] *Datasheet van motordriver*. URL: <http://www.pololu.com/file/0J534/drv8833.pdf>.

- [7] *Datasheet van sonar*. URL: [http://chibios.sourceforge.net/docs/kernel\\_avr\\_gcc\\_rm/index.html](http://chibios.sourceforge.net/docs/kernel_avr_gcc_rm/index.html).
- [8] *Git*. 2012. URL: <http://git-scm.com/>.
- [9] *Handleiding WiFly*. URL: [http://www.rovingnetworks.com/resources/download/93/WiFly\\_User\\_Manual](http://www.rovingnetworks.com/resources/download/93/WiFly_User_Manual).
- [10] *Latex*. 2010. URL: <http://www.latex-project.org/>.
- [11] *NXP LPCXpresso*. 2012. URL: <http://ics.nxp.com/lpcxpresso/>.
- [12] *Pinfuncties*. URL: <http://kornakprotoblog.blogspot.nl/2012/01/pinout-spreadsheet-for-stm32f4.html>.
- [13] *Raspberry Pi*. 2012. URL: <http://www.raspberrypi.org/>.
- [14] *Robot-dynamica*. 2012. URL: [http://www.societyofrobots.com/mechanics\\_dynamics.shtml](http://www.societyofrobots.com/mechanics_dynamics.shtml).
- [15] *Software robot*. URL: <https://github.com/Koensw/Robot-PWS>.
- [16] *Tilt-compensatie*. 2012. URL: <http://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf>.