KU Leuven
Departement Computerwetenschappen

P&O: Computerwetenschappen

Tussentijds verslag

Team:
Paars

Arno Biesmans
(Coördinator)
Arne Agten
(Secretaris)
Pieter Apeltans
Wouter Beert
Frederik Bode
Kristof Arron

Academiejaar 2015-2016

Samenvatting

Dit verslag beschrijft onze uitwerking van de opdracht voor P&O Computerwetenschappen van het academiejaar 2015-2016. De opdracht bestaat er in een robot te bouwen met behulp van een Raspberry Pi, een Brickpi, Lego Mindstorm sensoren en motoren en Lego Technics. Deze robot moet autonoom doorheen een parcours kunnen rijden en er moet een webinterface worden voorzien om de robot te kunnen aansturen. Voor het eerste deel van de opdracht moet de robot enkel een aantal basisbewegingen - rechtdoor, cirkel, vierkant- kunnen. In een eerste deel worden de ontwerpkeuzes en het design van de robot uitgebreid toegelicht. Verder worden een aantal tests en algoritmen beschreven die een correcte werking van de robot garanderen. In een volgend deel wordt alle ondersteunende software besproken, gaande van aansturing van de robot en zijn sensoren via Python scripts tot beeldanalyse en server-client communicatie. Tot slot wordt er aandacht besteed aan het ontwerp van de grafische user-interface (GUI) en besluiten.

Inhoudsopgave

1 Inleiding

Arne Agten

Het autonoom rijden van wagens is vandaag de dag één van de meest spraakmakende (en opkomende) technologieën. Onder andere Google, Tesla en andere autoconstructeurs steken veel tijd en geld in het ontwikkelen van dit soort wagens. Dit probleem is echter zeer complex, zowel de gezonde menselijke agressiviteit als het functioneren bij slecht weer moet opgevangen kunnen worden door de soft- en hardware van de auto.

In dit verslag wordt het model van de autonome wagen vereenvoudigd, en trachten we een robot te bouwen en programmeren die door een parcours kan rijden door een gekleurde lijn te volgen met zijn sensoren. Hierbij komen alle aspecten van het probleem aan bod: van materiaalkeuze tot finetunen van bepaalde parameters in de software.

Allereerst moet de robot in staat zijn om bepaalde figuren te kunnen rijden en om manueel aangestuurd te worden. De manuele aansturing is via een webpagina op allerhande toestellen te gebruiken en ook de figuren kunnen via deze webpagina door de robot gereden worden. Dit is waar het tussentijds verslag over handelt en dieper op in tracht te gaan.

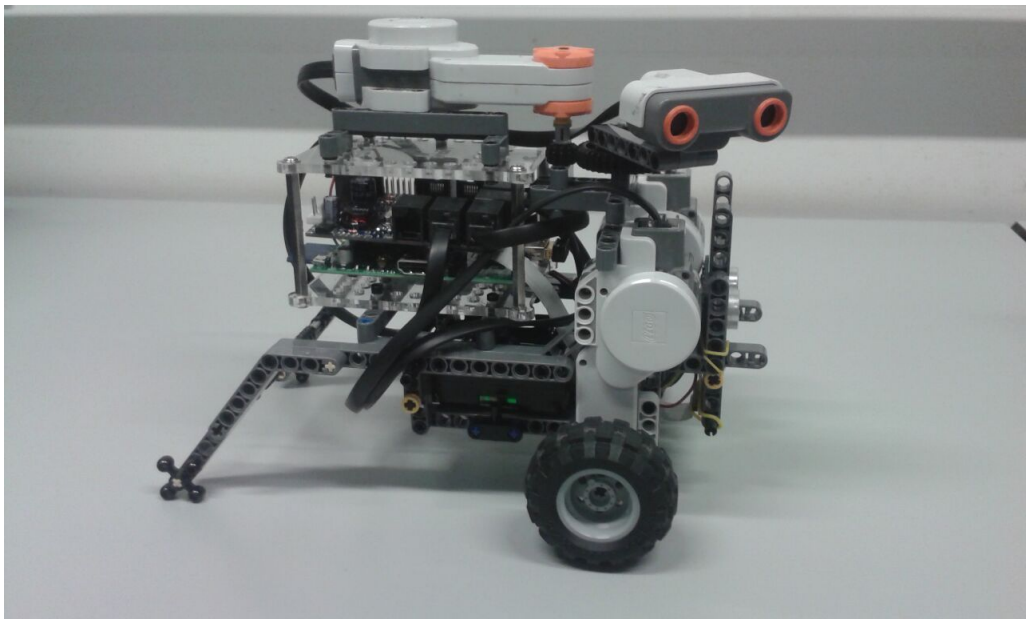
In een later deel van de opdracht wordt de complexiteit van de opdracht verhoogd door een lijn doorheen een parcours te kunnen volgen. Sommige delen van de latere opdracht zijn reeds verwerkt in dit verslag (o.a. afstandssensoren en beeldherkenning), andere delen zullen pas in het finale verslag verwerkt worden.

2 Beschrijving materiaal en bouw robot

Arno Biesmans

2.1 Materiaal

Het frame van de robot is helemaal gemaakt uit Lego Technic, met uitzondering van de twee ingebouwde Lego NXT motoren en de BrickPi case (zie Figuur 1). Verder is alleen gebruik gemaakt van een paar elastiekjes om de Pi camera en afstandsensor op zijn plaats te houden.



Figuur 1: Het zijaanzicht van de robot.

Lengte	Breedte	Hoogte
25cm	16,3cm	18,5cm

Tabel 1: De afmetingen van de robot in centimeter.

2.2 Bouw

De robot bestaat uit een frame waarin twee motoren zijn ingebouwd. Op dit frame wordt, achter de motoren, de BrickPi gezet. Voor de batterijen is er een rekje voorzien onder de BrickPi. Boven op de BrickPi staat nog een derde motor die, via twee tandwielen, de draaiende afstandssensor bedient. Voorop de motoren is er nog een klein platform voorzien om de Pi camera en afstandssensor te zetten. De afmetingen van de robot zijn beschikbaar in Tabel 1.

2.3 Ontwerp

Voor de aandrijving hebben we gekozen voor twee wielen, elk aangedreven door hun eigen motor om de robot makkelijk te kunnen laten draaien. Deze motoren staan verticaal gemonteerd vooraan in het frame zodanig dat er meer gewicht op de wielen komt te staan, wat de kans op slippen vermindert. De wielen staan rechtstreeks op de motoren gemonteerd. Een eerdere versie van de robot gebruikte tandwielen om enerzijds meer precisie of anderzijds een hogere maximum snelheid te bereiken. Tijdens het rijden werd echter duidelijk dat deze tandwielen elkaar scheef trokken, waardoor de wielen niet recht stonden en dit het heel moeilijk maakte om op een rechte lijn te rijden.

Als steunpunt aan de achterkant is uiteindelijk de keuze gemaakt voor twee bolvormige steunpunten. Na ook een enkel steunpunt, vrij wiel en bolvormig steunpunt getest te hebben werkte voor ons de huidige constructie het beste. De andere opties bleven nogal vaak vasthangen achter de gebruikte tape en zorgden er zo voor dat de robot slipte.

De Pi camera is vooraan beneden op de robot geplaatst en staat onder een kleine hoek naar beneden. Dit geeft een beeld van vlak voor de robot tot aan de horizon. De draaiende afstandssensor is toegevoegd om de robot in de toekomst (tweedee semester) makkelijk de mogelijkheid te geven om obstakels in zijn omgeving te detecteren. Op deze manier heeft de robot ook informatie over wat er zich voor en langs hem bevindt, in plaats van alleen de mogelijkheid te hebben om vooruit te kijken. Vooraan staat nog steeds een andere afstandssensor die vast naar voor kijkt. Voor de motor bovenop is er een tandwiel overbrenging aangebracht om de afstandssensor te draaien.Hierdoor ontstaat de mogelijkheid om deze sensor met een veel grotere precisie te roteren in het horizontale vlak. De BrickPi is op het frame gemonteerd. Dit maakt het makkelijk om deze te demonteren tijdens het testen.

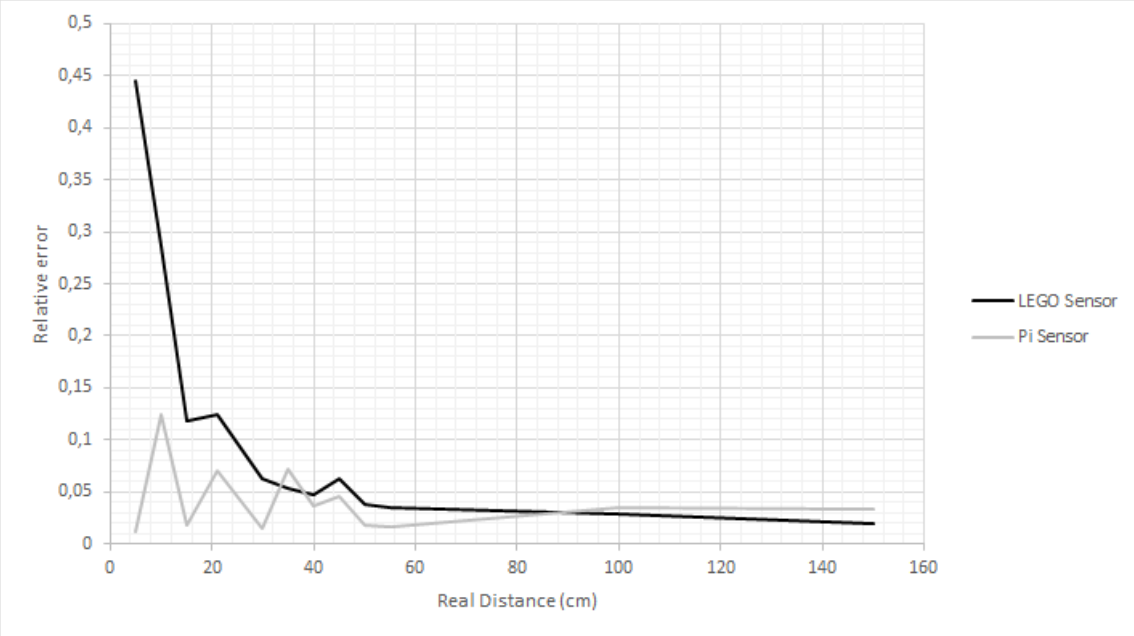
3 Testen

3.1 Afstandssensoren

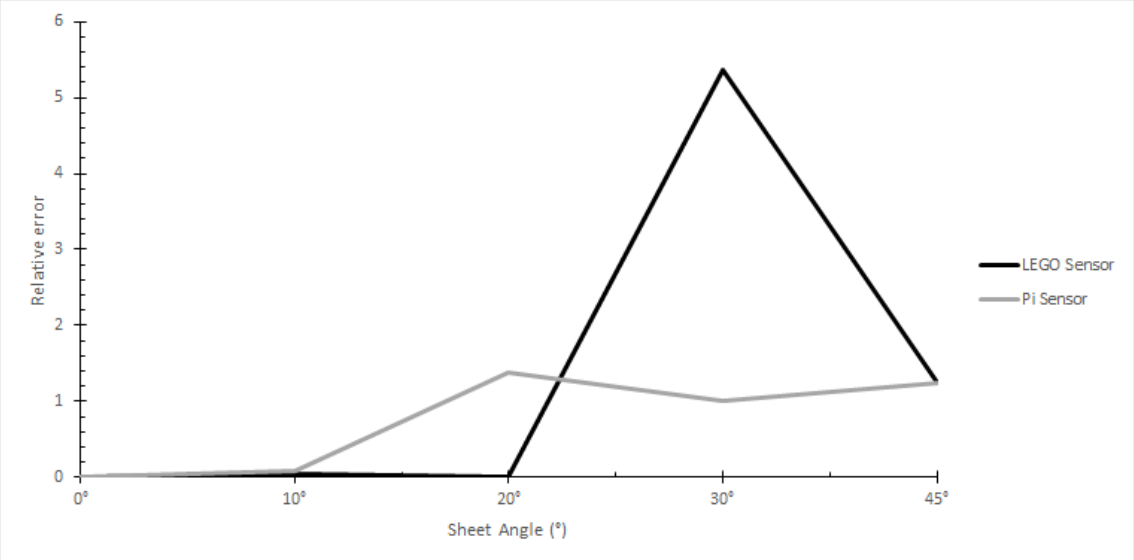
Arne Agten
Bij het rijden doorheen het parcours, moet de robot objecten kunnen detecteren. Hiervoor heeft hij twee afstandssensoren, die door het principe van ultrasonische pulsen de afstand tot het object bepalen. De afstandssensor van LEGO Mindstorms kan in centimeters en inches meten. Zijn precisie is ongeveer 3cm en kan afstanden van 0 tot 255 centimeter opmeten [3]. De afstandssensor van de Raspberry Pi heeft een meetafstand van 2 tot 400 centimeter en heeft een precisie van ongeveer 3mm [2].

Voor de eerste test staat de robot op een variërende afstand voor een witte muur. Op de volgende afstanden (in centimeter) worden er metingen gedaan: 5 - 10 - 15 - 20 - 25 - 30 - 45 - 55 - 100 - 150. Het testprogramma neemt voor elke meting ongeveer honderd waarden en pakt van deze waarden de mediaan. Beide afstandssensoren ondergaan de test. Voor de tweede test staat de robot op een vaste afstand van 40cm van de muur af. De hoek van de robot ten opzichte van de muur varieert voor volgende waarden (in graden): 0 - 10 - 20 - 30 - 45. Dezelfde meetmethode als bij de eerste test geldt.

Resultaat De uitkomsten van het eerste experiment zijn afgebeeld in Figuur 2: De relatieve fout van de mediaan van honderd opgemeten waarden in functie van de afstand tot de muur. Op de verticale as bevindt zich de relatieve fout, als volgt berekend: rel. err. = (real dist. - output dist.)/real dist.. Op de horizontale as is de reële afstand van de robot tot de muur uitgezet. De uitkomsten van het tweede experiment zijn afgebeeld in Figuur 3: De relatieve fout van de mediaan van honderd waarden in functie van de hoek tussen de robot en de muur. Op de verticale as bevindt zich opnieuw de relatieve fout. Op de horizontale as de hoek tussen de robot en de muur.



Figuur 2: De relatieve fout van de mediaan van honderd opgemeten waarden in functie van de afstand tot de muur in centimeter.



Figuur 3: De relatieve fout van de mediaan van honderd waarden in functie van de hoek tussen de robot en de muur in graden.

Conclusie We kunnen bemerken dat de Pi sensor een hogere precisie heeft op kleine afstanden dan de LEGO sensor. Dit is in lijn van de verwachtingen, aangezien de LEGO sensor een veel

kleinere accuraatheid heeft, die bij kleine afstanden harder doorweegt. Bij afstanden van meer als 100cm hebben beide sensoren een kleine relatieve fout, dus kan de LEGO sensor op de derde motor roteren om op een kruispunt de juiste richting te kiezen. Uit Figuur 3 kunnen we ook bemerken dat de accuraatheid van de LEGO sensor ten opzichte van de grootte van de hoek groter is dan die van de Pi. Vanaf een hoek van 15°stopt de Pi sensor met werken, de LEGO sensor kan tot 20°gaan. De LEGO sensor kan dus kleinere oppervlakten waarnemen.

4 Algoritmes

Pieter Appeltans

PID-regelaar: Een PID-regelaar heeft een uitgangssignaal dat evenredig is met de grootte van de fout (Proportional), de integraal (Integral) van de fout en de afgeleide van de fout (Derivate). Ieder van deze factoren krijgt een zeker gewichtscoëfficiënt: Kp, Ki en Kd respectievelijk. De integrerende actie treed in werking als er een kleine accumulerende fout blijft openstaan, de differentiëren actie zal inwerken op een plotse verandering van de fout. De robot gebruikt een PID-regelaar om correct de opgelegde bewegingen (rechtdoor rijden over een zekere afstand, een cirkel rijden met een zekere straal en een vierkant met een gegeven lengte voor de zijdes) uit te voeren [1].

4.1 Rechte lijn met variabele lengte

De robot gebruikt twee PID-regelaars om een zo recht mogelijke lijn te rijden. De eerste PID-regelaar heeft als ingang de nog af te leggen afstand en geeft als uitgang een snelheid voor beide motoren. Aan de ingang van tweede PID-regelaar wordt het afstandsverschil tussen het linker en het rechter wiel aangelegd en zijn uitgang geeft het snelheidsverschil tussen beide wielen aan. Zie Code 1 voor de pseudocode van dit algoritme.

4.2 Cirkel met variabele straal

Om een cirkel met variabele[1] straal te rijden zijn er voorlopig nog twee algoritmes. Verdere testen moeten uitwijzen welk algoritme het beste is.

4.2.1 Algoritme 1

Het eerste algoritme maakt gebruik van twee PID-regelaars. Het berekent daarvoor de omtrek van de binnen- en buitencirkel. Aan de ingang van iedere PID-regelaar wordt het verschil van de al afgelegde afstand van het binnen-, buitenwiel en respectievelijk de binnen en buiten straal aangelegd. Dit algoritme is vrij eenvoudig. Het nadeel is dat deze methode er mogelijks toe leidt dat de robot een ellips beschrijft in plaats van een cirkel, doordat de robot vertraagt naarmate hij zijn doel nadert, zie Code 2 voor de pseudocode van dit algoritme.

4.2.2 Algoritme 2

Ook het tweede algoritme maakt gebruik van twee PID-regelaars. Deze zal de cirkel echter benaderen door een veelhoek. Iedere tijdsperiode moet ieder wiel een zekere afstand afleggen. Deze afstand wordt als ingang van de PID-regelaar aangeboden. Het voordeel van deze methode is dat de robot in elke periode dezelfde afstand probeert af te leggen, waardoor de beweging gelijkmatiger is.

4.3 Rotatie om as

Ook voor deze beweging is er een algoritme met twee PID-regelaars voorzien. Ieder wiel moet een gelijke afstand afleggen, maar in de tegengestelde richting. Het verschil tussen de af te leggen afstand en de werkelijk afgelegde afstand wordt aan de ingang van iedere PID-regelaar aangelegd.

[1] De gegeven straal is de straal die door het binnenste wiel wordt afgelegd.

5 Software

Arne Agten, Pieter Appeltans, Kristof Arron, Frederik Bode

5.1 Aansturing

De code voor aansturing van de verschillende componenten (motoren, sensoren en de camera) is geschreven in Python (versie 2.7) gebruik makend van een object gerichte methode. Voor elk van deze componenten is een aparte klasse voorzien. Daarboven is er voor elk type input/output nog een aparte thread (zie volgende paragraaf) voorzien. Tot slot is er nog een interface nodig die alle commando's uitvoert. Figuur 4 toont het klassediagram in verband met de aansturing van de robot.

Interface zorgt voor de aansturing van de robot en maakt daarvoor gebruik van twee threads en drie andere zelfgemaakte klassen: DistanceSensor, MindstormSensor en Motor.

5.2 Threading

Threading (of ook Multithreading) is een techniek waarbij een proces wordt opgesplitst in meerdere threads. Als gevolg van deze opsplitsing is het mogelijk om verschillende taken tegelijk uit te voeren binnen één proces. Wisselen tussen threads is sneller dan het toewijzen van een compleet nieuw proces aan de processor.

Deze eigenschappen zijn in de praktijk zeer nuttig bij de aansturing en monitoring van de robot. Door gebruik te maken van threads voor verschillende componenten (motoren, sensoren, camera) kunnen al deze componenten tegelijk gecontroleerd worden. Zo kan de robot bijvoorbeeld rijden en op hetzelfde moment de sensorwaarden en camerabeelden versturen.

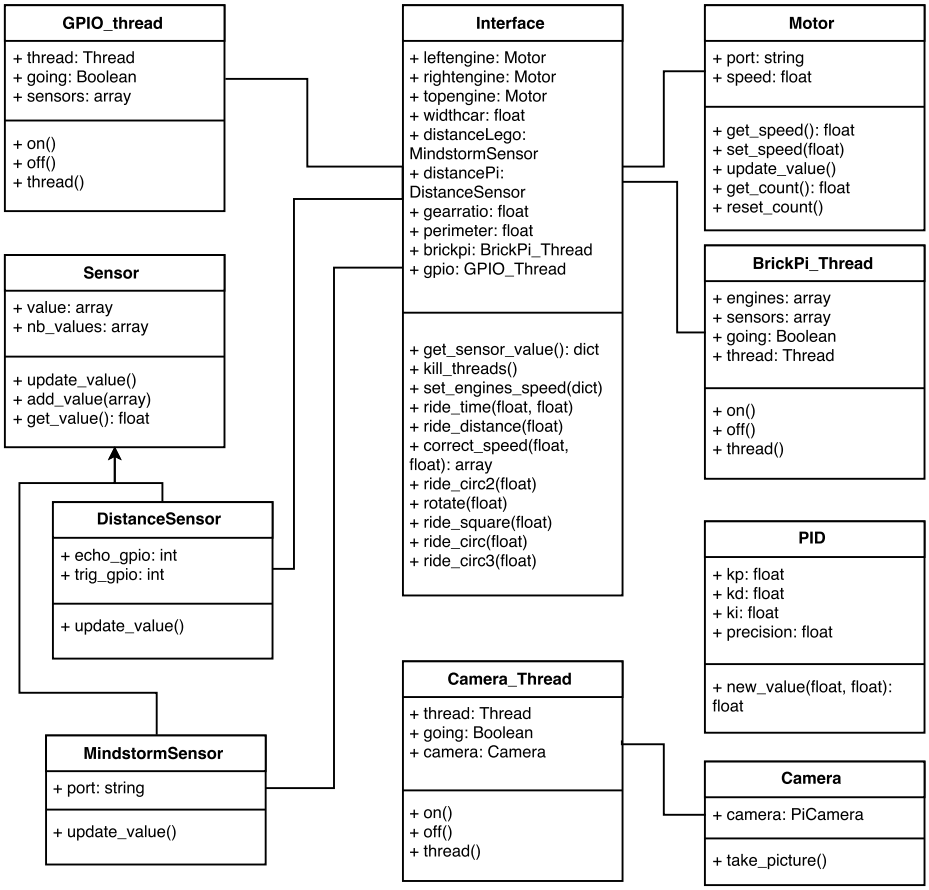
De code maakt gebruik van drie threads, één voor de aansturing van de BrickPi onderdelen, één voor de aansturing van de GPIO-pinnen en nog een laatste voor de camera. Deze threads kunnen gestart en gestopt worden (on en off). Als een thread gestart is, zal het ieder tijdsinterval een actie - sensorwaarde uitlezen, motorsnelheid aanpassen, een foto nemen, ... - ondernemen.

5.3 Sensoren

De gebruikte sensoren hebben maar een beperkte nauwkeurigheid. Daarom geeft de software niet de ogenblikkelijke waarde van de sensoren terug maar geeft hij de mediaan van de laatste x metingen terug. Door het aantal bijgehouden metingen en de tijd tussen opeenvolgende metingen beperkt te houden zal er een goede schatting van de werkelijke waarde worden teruggegeven.

5.4 Communicatie externe server

De communicatie tussen de Raspberry Pi en de computer maakt gebruik van sockets. Het server script draait op de Pi en het client script draait op de computer. Elke transmissie bestaat uit twee delen. Eerst stuurt de client de lengte en het volgnummer van de boodschap door naar de server. Het volgnummer bepaald de volgorde waarin de Pi de boodschap moet afhandelen. Het tweede deel is de boodschap zelf. Dit is een commando dat op de Pi moet worden uitgevoerd. Alle verzonden commando's worden door middel van de module cPickle getransformeerd tot een bytestream vooraleer ze verzonden worden. De ontvanger converteert deze bytes dan weer terug naar de oorspronkelijke boodschap. De verstuurde grootte van de commando's en de volgnummers worden niet omgezet naar een bytestream. Indien de client dit vereist stuurt de server na het uitvoeren van het commando een respons terug. Ook hier stuurt de zender eerst de lengte en dan pas de echte boodschap. Het versturen van de lengte heeft als voordeel dat de ontvanger weet hoe groot de boodschap is die hij zal ontvangen van de zender.



Figuur 4: Klassediagram voor de robot

Bij het opstarten van de server wordt er gewacht tot een client verbinding maakt. Eens de verbinding tot stand is gekomen luistert de server actief naar binnenkomende data. Zoals eerder vermeld zal die data uit twee delen bestaan. De ontvangen bytestream wordt getransformeerd tot een commando dat uitgevoerd wordt op de interface van de Pi.

Aan de clientzijde gebeurt er iets meer. Allereerst maakt de client verbinding met de server, vanaf dan kan de gebruiker commando's versturen. Elk instructie die de gebruiker opgeeft wordt eerst getransformeerd naar een commando dat de server kan interpreteren. Daarna krijgt het commando een volgnummer. Dit nummer is het eerste beschikbare nummer binnen een vooraf bepaalde range. Het koppel (volgnummer, commando) komt op een queue terecht tot een apart thread het kan versturen naar de server. Een tweede thread luistert naar de ontvangen respons die het verstuurde commando kan genereren. Die respons wordt dan lokaal opgeslagen en kan gebruikt worden om de robot bij te sturen.

5.5 Fotoanalyse

Een kort overzicht van de beeldverwerking is samen te vatten in zes stappen. Eerst wordt het beeld omgezet naar grijswaarden. De verdeling van deze waarden in het frame bepaalt hoe ze worden omgezet naar een absoluut zwart wit in de tweede stap. Vervolgens wordt er het algoritme van Canny [4] op toegepast om de randpunten te vinden, waarna er lijnen en curves worden herkend met behulp van de Hough getransformeerde [5] in een vierde stap. Als vijfde wordt er een parallelle curve getrokken in het midden van de witte lijn, en tot slot worden op basis daarvan de snelheden berekend. Al de gemeten paden worden in kaart gebracht.

6 GUI

Arne Agten, Pieter Appeltans

6.1 Bootstrap

Het framework gebruikt voor de website is Bootstrap. Het maakt snel webdesign mogelijk en vereenvoudigt het maken van een website voor verschillende toestellen (PC, tablet, GSM, ...). Door middel van twee CSS- en twee JavaScript files kan men gemakkelijk de lay-out van de website controleren. Ook past de website zich aan aan mobiele interfaces wanneer het scherm kleiner wordt. In 5 kan men de website voor een mobiel toestel vinden. Deze zal naarmate de opdracht vordert uitgebreid worden.

7 Overzicht algoritme's

In dit deel geven we de pseudocode van de beschreven algoritmes.

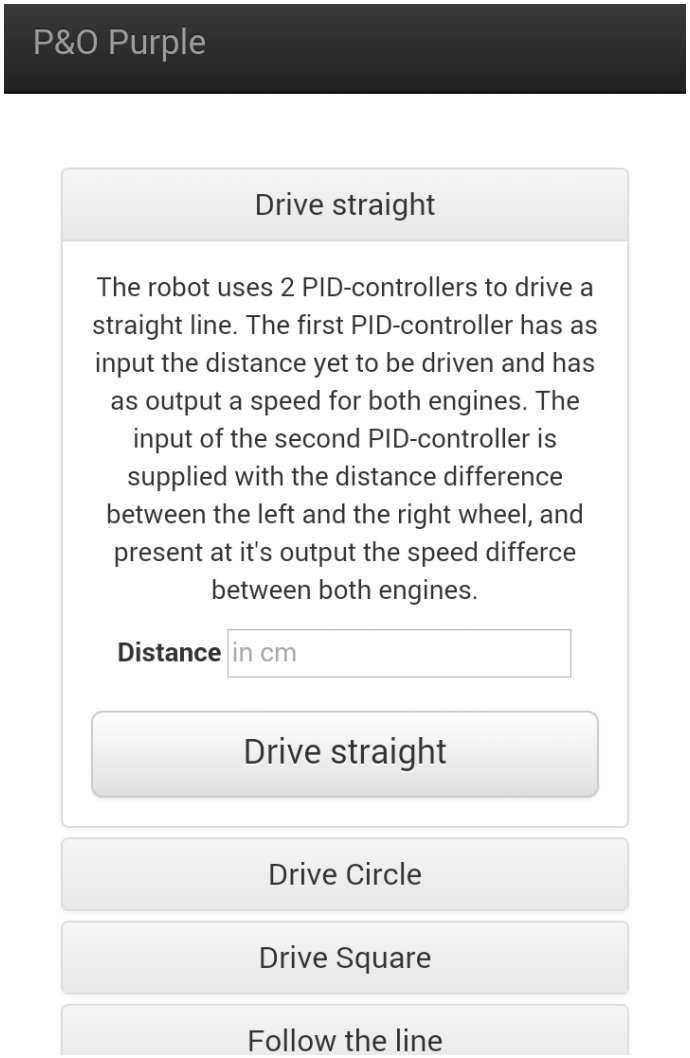
```
# Code for driving a straight line.
ride_distance(distance)
pid1 = PID.PID(5.,1/20.,1/50.,.5)
pid2 = PID.PID(10.,1/2.,1/5.,.5)
speed = MINIMUM_SPEED
leftengine.set_speed(speed)
rightengine.set_speed(speed)
while speed !=0:
    distance1 = leftengine.nb_of_rotations * gearratio * circumference_wheel
    distance2 = rightengine.nb_of_rotations * gearratio * circumference_wheel
    speed = pid1.new_value(distance-distance1,0.01)
    speed_diff = pid2.new_value(distance1-distance2,0.01)
   # Rescales speeds to [-255,255]
    lspeed,rspeed = correct_speed(speed,speed_diff)
    leftengine.set_speed(lspeed)
    rightengine.set_speed(rspeed)
    sleep(0.01)
    leftengine.set_speed(0)
    rightengine.set_speed(0)
```

Code 1: Code for driving a straight line.

```
# Code for driving a circle with given radius.
def ride_circ(self,radius):
    if abs(radius) <20:
        raise Exception
    pid1 = PID.PID(10.,1/20.,1/50.,1.)
    pid2 = PID.PID(10.,1/20.,1/50.,1.)
    if radius>0:
        inner_engine = self.__rightengine
        outer_engine = self.__leftengine
```

```
else:
    inner_engine = self.__leftengine
    outer_engine = self.__rightengine
inner_engine.reset_count()
outer_engine.reset_count()
angle_per_loop = 1./(float(radius))
angle = 0.
while angle < 2*math.pi:
    distance1 = inner_engine.get_count()*self.__circumference*self.__gearratio
    distance2 = outer_engine.get_count()*self.__circumference*self.__gearratio
    speed1 = pid1.new_value(angle*abs(radius)-distance1,0.1)
    speed2 = pid2.new_value(angle*(abs(radius)+ self.__widthcar)-distance2,0.1)
    inner_engine.set_speed(speed1)
    outer_engine.set_speed(speed2)
    angle += angle_per_loop
   time.sleep(0.1)
inner_engine.set_speed(0)
outer_engine.set_speed(0)
```

Code 2: Code for driving in a circle with given radius.



Figuur 5: De website zoals gezien van op een mobiele telefoon.

8 Besluit

Arne Agten

Bij de eerste tussentijdse demo moet de autonome robot enkele basis figuren kunnen rijden. Hierbij komen echter wat problemen bij kijken: motoren die niet dezelfde kracht kunnen leveren, wielen die doorslippen, ... Een PID-controller kan deze fouten controleren, maar de parameters die de PID-controller gebruikt hebben telkens een andere ideale waarde. De batterijen kunnen bijvoorbeeld bijna leeg zijn waardoor de wagen onverwachte wendingen neemt tijdens het rijden.

Ook de sensoren zijn niet altijd even betrouwbaar. De afstandssensoren van LEGO hebben een tamelijk slechte accuraatheid op korte afstand en de nauwkeurigheid ervan is ook gering. De afstandssensor van de Raspberry Pi daarentegen is meer betrouwbaar en zal dan ook meer in rekening gebracht worden bij het schrijven van verdere software later in het project.

In het volgende deel van het project zal de robot doorheen een parcours een gekleurde lijn moeten kunnen volgen. Hiervoor is fotoanalyse en lijndetectie nodig. De lichtinval en scherpte van de foto bepalen het niveau van herkenning van lijnen en verdere finetuning van deze code is dan ook nodig. Ook de voorlopig beperkte HTML-interface zal in een later stadium verder uitgebouwd worden om een grotere gebruiksvriendelijkheid toe te laten.

Referenties

[1] AtmelCorporation, *Avr221: Discrete pid controller.* `http://www.atmel.com/images/doc2558.pdf` [raadpleging 3-November-2015].

[2] elecfreaks, *Ultrasonic ranging module hc - sr04.* `http://e-radionica.com/productdata/HCSR04.pdf` [raadpleging 3-November-2015].

[3] M. Moro, *Lego mindstorms nxt (hardware and software).* `http://www.terecop.eu/downloads/appendix_1.pdf` [raadpleging 3-November-2015].

[4] Wikipedia, *Canny edge detector — wikipedia, the free encyclopedia*, 2015. `https://en.wikipedia.org/w/index.php?title=Canny_edge_detector&oldid=683143347` [raadpleging 4-November-2015].

[5] ——, *Hough transform — wikipedia, the free encyclopedia*, 2015. `https://en.wikipedia.org/w/index.php?title=Hough_transform&oldid=679388059` [raadpleging 4-November-2015].