Universiteit van Amsterdam
Faculteit Natuurwetenschappen, Wiskunde en Informatica

Een 3D-pak voor een sphero
ZSB

19 juni 2015

Studenten:
Mark Kooijman 10369074
Rick de Reus 10765042
Jelle Rots 10645349
Max Crous 10771085
Groep: E
Open in pdfviewer
voor correcte referentie hyperlinks

Inhoudsopgave

1 Inleiding

Het doel van het project is om een robot te creëren. Wij wilden een robot die out-of-the-box zijn omgeving niet waar kan nemen in staat stellen dit wel te doen. Dit wilde wij bewerkstelligen door de robot te modificeren. We konden ook de omgeving van sensoren en een controller voorzien om de robot extern te begeleiden, echter wilde wij de autonomie van de robot bevorderen.
Ons idee was om een sensor op een sphero zelf te bevestigen, zodat de sphero met behulp van informatie van de sensor objecten kan vermijden.
Een sphero is een bol vormige robot die door middel van low-power bluetooth aangestuurd kan worden. De sphero is een bol en heeft dus geen locatie om een sensor te bevestigen. Er moest dus een *suit* om de bol gebouwd worden, wilde wij sensorische data gebruiken.
De suit is opgebouwd uit een koepel die, met een paar millimeter speling, om de sphero zit en een kar die bevestigd is aan die koepel. Op de kar zit een raspberry pi om sensorische data te verwerken. De sensor is op de koepel bevestigd. Specificaties van de gebruikte hardware en software staan in de appendix.

1.1 Onderzoeksvraag

Om ons doel te bereiken van het autonoom navigeren door een ruimte hebben wij deze uitdaging opgesplitst in deelvragen.
De eerste vraag luidt als volgt: "hoe kunnen wij de sphero in beweging brengen?:
Wij maken gebruik van de sphero software development kit om via bluetooth commando's te sturen naar de sphero. Van deze SDK is er een API gemaakt voor de programmertaal javascript. Deze javascript API is weer opgenomen in een node.js API. Een node.js compiler heeft als voordeel dat code buiten een browser kan worden uitgevoerd. Om de geschreven code op alle machines bij te houden hebben we gebruik gemaakt van de gitserver bitbucket. Alle code werd "gepulld"naar de raspberry pi om vervolgens daar uitgevoerd te worden. We maakten tijdens het testen gebruik van een ethernet SSH connectie met de raspberry pi om commando's te sturen. Tijdens de demonstratie zal dit via het wifi netwerk gebeuren, zodat de robot losgekoppeld is.
Dit zijn de stappen die wij denken nodig te hebben om langs obstakels te navigeren:

1. Calibreer
2. Waarnemen uit 0°→ Check.py
3. Draai de sphero 90°↘
4. Rij 2 sec vooruit → orb.roll(snelheid,richting in degree)
5. Waarnemen
6. Rij in vrije richting
7. Waarnemen tijdens rijden → Movecheck.py→ sneller dan check.py
8. Als er obstakel is, draai 90°/-90°afwisselend

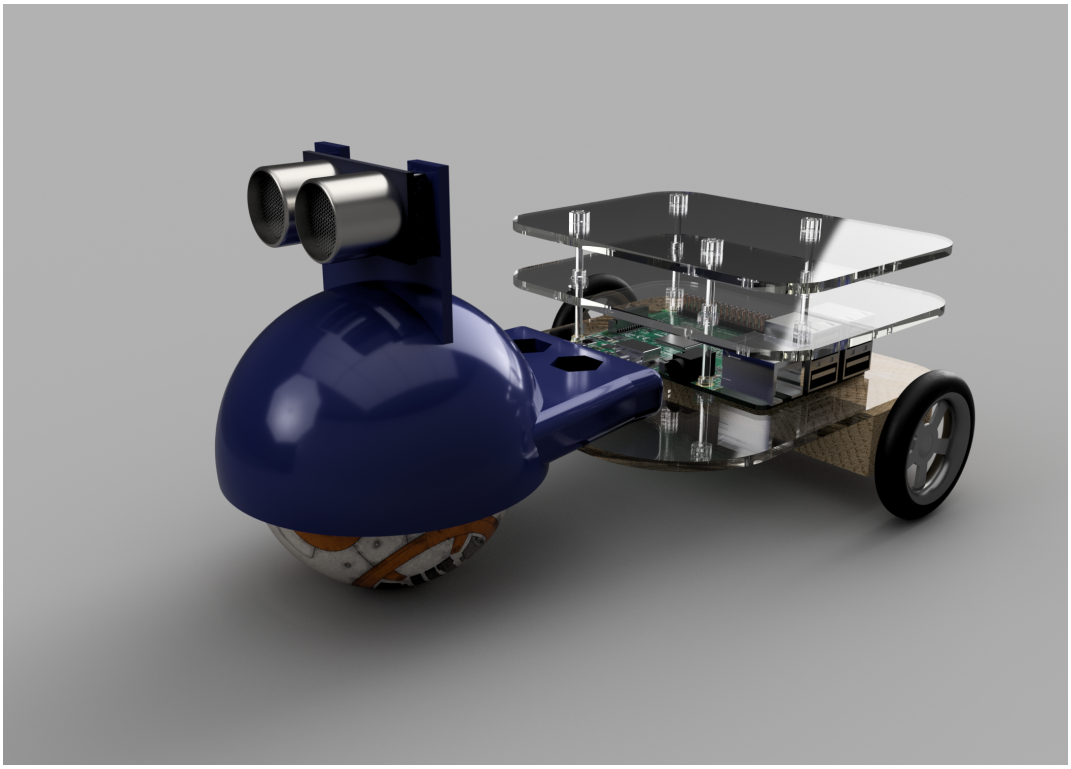
De tweede deelvraag is: "hoe maken wij de suit voor de sphero?"
De suit bestaat uit een koepel, een kar, een sensor en wielen. De koepel hebben wij gemaakt met een 3D printer. Het virtuele model hebben wij in het programma Autodesk Fusion 3D gemodelleerd. Om de sensor aan de koepel te kunnen bevestigen hebben we bovenop de koepel een plaatje bevestigd met vrije ruimte. We hebben ervoor gekozen deze boven de koepel te plaatsen, opdat de

sensor daar veilig is van eventuele botsingen. De bevestigingspunten van de koepel (aan de plaat met wielen) bestaan uit twee 6.5mm gaten op een vlak achter de koepel,loodrecht op de sensor plaat. Met behulp van twee bouten en twee moeren kan de koepel aan een wagen bevestigd worden.
Nadat de koepel geprint was hebben we de kar ontworpen. Om de kar zo klein mogelijk te houden hebben we in Autodesk Fusion 3D met een 3D model van de raspberry pi de minimale dimensies van de basisplaten bepaald, zodanig dat er bij alle hoeken nog ruimte overbleef voor de schroefdraden. Deze schroefdraden houden de hoogste platen op hun plaats. We hebben gekozen voor schroefdraden zodat de afstand tussen de twee platen met behulp van moeren aangepast kon worden. Het ontwerp is volledig modulair, wat betekent dat er nieuwe delen aan toegevoegd kunnen worden, en oude verwijderd. De wielen van de kar zijn aan de achterkant geplaatst, opdat de koepel volledig op de sphero rust.

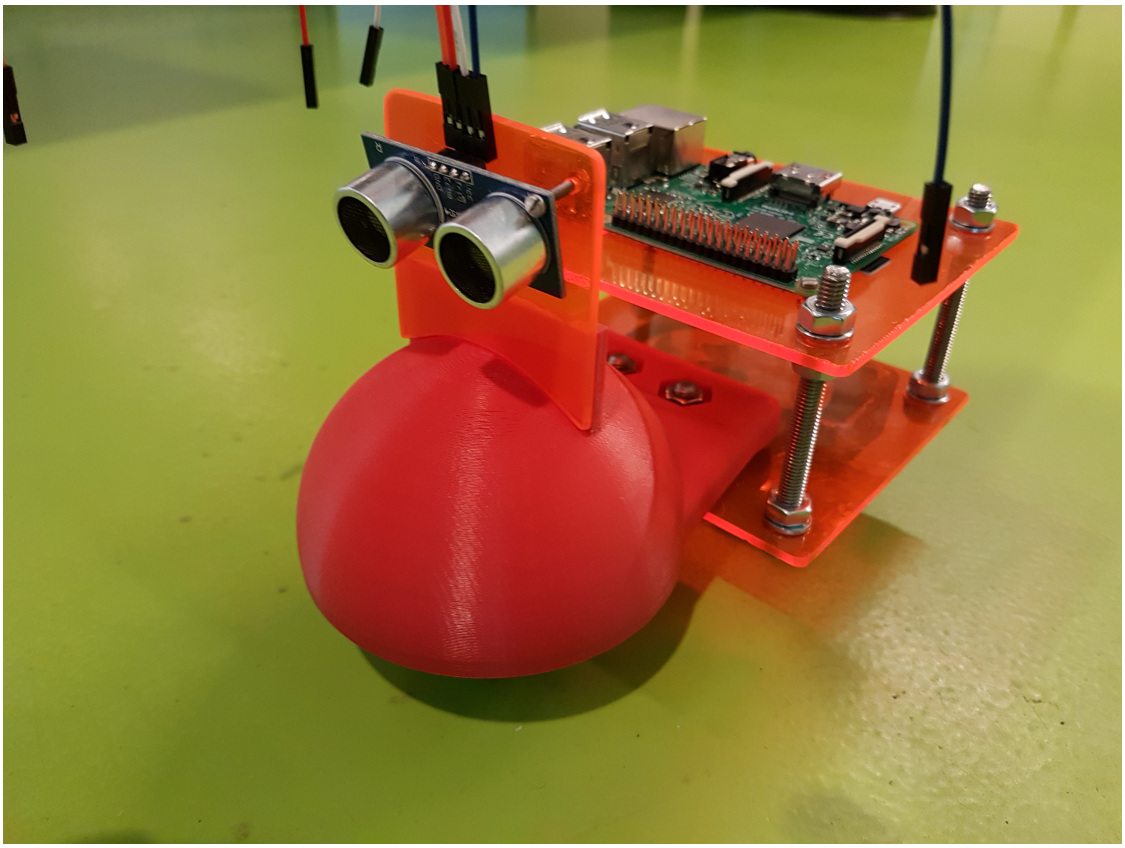
De derde deelvraag is: "hoe verkrijgen we de data van de ultrasonische sensor?"
Om de sensor elektriciteit te geven hebben we het aangesloten op de GPIO pins van de raspberry. Toen hebben we gebruik gemaakt van een python code verkregen van de website https://goo.gl/3ncips. Deze code roept de transmitter aan om ultrasonische signalen (40k Hz) uit te zenden. Na het uitzenden worden de echo's van de geluidsgolven weer opgevangen. Aan de hand van de tijdsduur tussen het uitzenden en de ontvangst kunnen we met behulp van de snelheid van het geluid de afstand uitrekenen.

```
while GPIO.input(Of_The_ECHO)==0: #Check whether the ECHO is empty
    pulse_start = time.time() #Start sending a pulse and
                              save the time when the pulse started

while GPIO.input(Of_The_ECHO)==1: #Check whether the ECHO is active
    pulse_end = time.time() #Saves the time that the pulse
                            was received back (the actual echo)

pulse_duration = pulse_end - pulse_start #Get pulse duration
                                         to a variable

distance = pulse_duration * 17150 # Multiply pulse duration by 17150
                                  to get distance in centimeters

                                  # The 17150 is the factor needed to
                                    incorporate the speed of sound
                                    into the equation
```

Autodesk Fusion 3D model



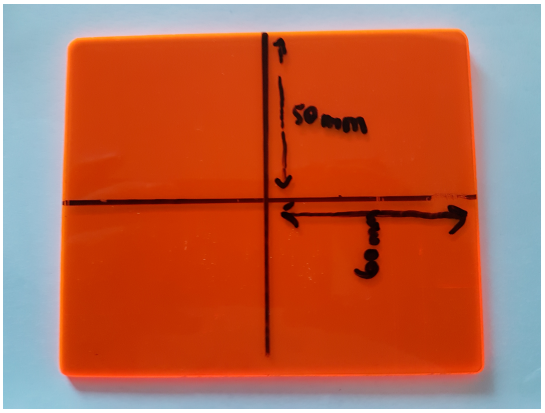
In realiteit



2 Voortgangsrapportage

2.1 Maandag, 20 juni 2016

De eerste stap naar ons doel is het samenstellen van de robot. In het weekend is de koepel geprint door de 3D printer. Maandag hebben we de koepel uit de printer gehaald en in een bak met aceton gelegd. Een chemische reactie met het aceton verwijderde alle steunpilaren die door de printer waar toegevoegd aan het model. Vervolgens hebben we werkplaats medewerkers gevraagd of zij ons een stuk plexiglas konden verschaffen. Zij gaven ons een prachtige oranje fluoriserende plaat. Deze hebben wij gezaagd in vier platen van 50 bij 60 mm. Twee hiervan zouden dienen als ondergrond voor de onderdelen. De andere twee dienden als reserves. Om de platen geschikt te maken voor gebruik hebben wij gaten geboord van 6mm in alle vier de hoeken. De plaat die de koepel direct raakt hebben we voorzien van een kleine inkeping. Om te voorkomen dat de robot geradbraakt gaat onder het gewicht van de suit hebben we de schroefdraden afgezaagd tot een lengte van 7 cm. Eerst waren deze 14 cm. Later zal blijken dat dit tevergeefs was, aangezien het gewicht van de suit alsnog veels te zwaar was voor de sphero.



2.2 Dinsdag,21 juni 2016

Deze dag hebben wij ons gericht op het werkend krijgen van de sensor. Door het gebruik te maken van deze website. Via de website zijn wij tot de conclusie gekomen dat wij een breadboard en weerstanden nodig hebben om de voltage aan te passen. Dit is nodig omdat het Echo kanaal van de sensor een constante stroom van 5V doorgeeft. In de stroom zit het signaal verwerkt. De raspberry pi kan 5V niet aan dus hebben we met weerstanden dit signaal terug gebracht naar 3.3V. We ontdekte bij het monteren van de sensor op het plaatje van de koepel, dat de sensor niet perfect horizontaal gericht was. Om de sensor horizontaal te krijgen hebben we filters van een sigaret van een medestudent gebruikt. De filters zijn geplaatst tussen de bevestigingsplaat en de sensor.
Verder hebben wij de sensor werkend gekregen. De sensor geeft met een constante interval aan wat de afstand is tussen de receptoren en het dichtstbijzijnde object, tot een afstand van 400cm.

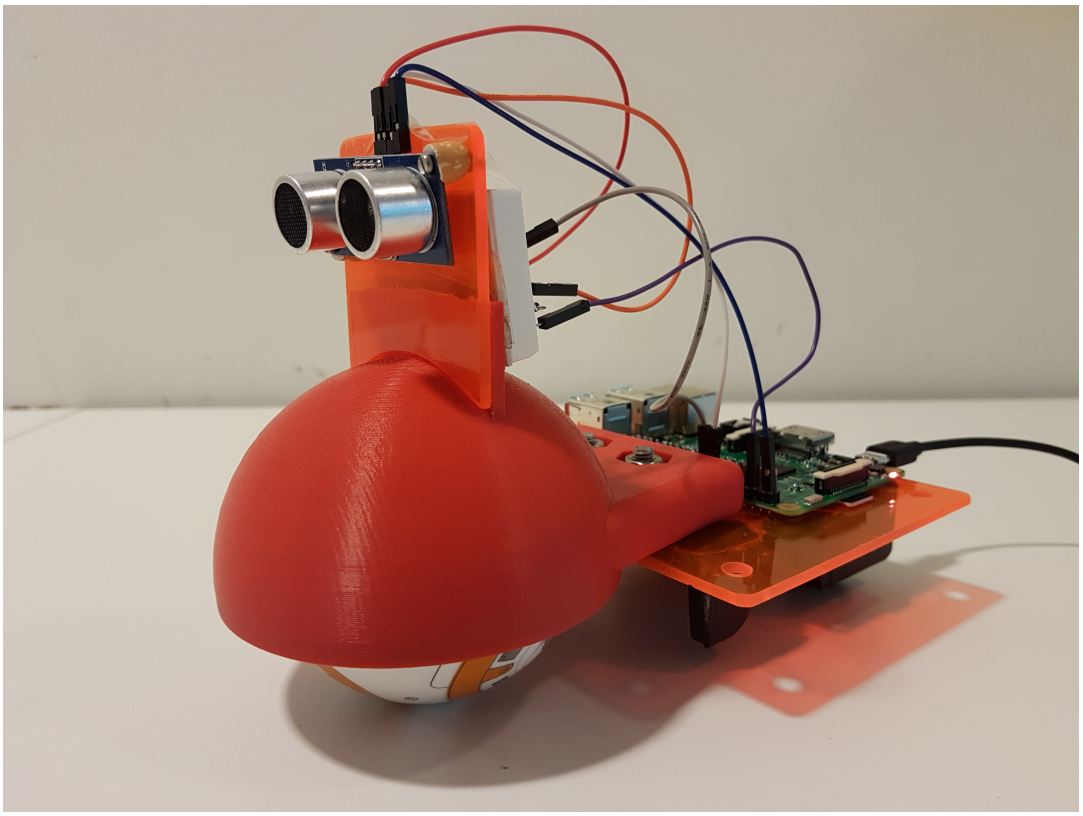
2.3 Woensdag,22 juni 2016

Er waren problemen met het laten rijden van de kar. In het begin zaten er 4 wielen op een locatie aan de achterkant van de kar. Als de batterij van de raspberry pi niet op de kar geplaatst was reed de kar met genoeg snelheid om recht vooruit te gaan. Als de batterij eenmaal op de kar zat had de sphero niet genoeg kracht om de kar voort te trekken.

In een poging om de kar sneller te laten gaan hadden we extra wielen onder de koepel geplaatst. Hierdoor viel de koepel echter niet volledig op de sphero waardoor deze tijdens het draaien grip op de vloer verloor. Om deze reden hebben we deze extra toegevoegde wielen weer verwijderd.
Eerst hebben wij het probleem proberen op te lossen door een betere ondergrond te gebruiken. Namelijk een hout platform. We besloten dat wij niet wilde dat onze robot afhankelijk zou zijn van het soort vloer. Om deze reden hebben wij het gewicht van de kar vermindert. Dit hebben wij gedaan door het tweede platform van de kar eraf te halen. Dit zorgde ervoor dat er minder gewicht op de.

2.4 Donderdag,23 juni 2016

Op donderdag hebben wij de code voor de sphero geschreven. De talen die wij hiervoor gebruikt hebben zijn python en node.js. Python gebruiken wij voor de sensor en node.js voor het besturen van de sphero. Tijdens het voortbewegen van de sphero vraagt het continue de waardes op van de sensor. Hiermee kan de sphero bepalen of er een obstakel voor hem bevindt. Met deze informatie wordt er in node.js bepaald welke richting de sphero op moet rollen om niet tegen een obstakel aan te komen. Hiervoor hebben wij meerdere bestanden gemaakt. In totaal zijn het drie bestanden. Respectievelijk één voor het ontwijken van obstakels door middel van een manoeuvre naar rechts. Één voor het afwisselen van de manoeuvres, eerst een naar rechts gevolgd door een manoeuvre naar links. En als laatste één die tijdens het manoeuvreren naar rechts checkt of er genoeg ruimte is, wanneer dit niet het geval is zal hij manoeuvreren naar links.
Aan het einde van de dag ziet de eindversie van de robot er dusdanig uit:



3 Discussie

Het is ons gelukt om een sphero semi-autonoom te maken. Het feit dat de robot door een gang kan manoeuvreren en objecten anticipeerd van een redelijke afstand, geeft ons hoop dat het ontwerp potentie heeft. Idealiter zouden we de autonomie van de robot nog kunnen bevorderen door de terminal commando's te sturen met een afstands bediening.
Wij hebben ons niet verdiept in de innerlijke werking van de sphero (Accelerometer and Gyrometer). Deze kennis zou eventueel hebben bijgedragen aan de mogelijke manoeuvres van de sphero. Wij zijn ook erg geïnteresseerd in het idee van automatische calibratie. Als we de koepel zouden voorzien van een lichtgevoel element, zoals een diode, en de calibratie lamp konden laten richting op dat element, dan zou de robot in staat kunnen zijn altijd te weten wat "vooruit" is.
Het vermeiden van objecten gebeurt op een vrij primitieve manier. Als de robot een object signaleerd, dan rijdt hij voor een bepaalde tijd een andere kant op. Met complexere algoritmes en meerdere sensoren zouden we veel meer "states" kunnen herkennen. Denk hierbij aan de situatie waarbij de robot zich in de hoek van de kamer begeeft.
Betreffende het ontwerp van de kar hebben we veel geleerd. Er is veel fout gegaan waardoor we gedwongen werden om creatief na te denken. Dit valt ook te zien aan het uiteindelijke concept. Door het gewicht van de kar kan de sphero de kar niet voortbewegen. Hierdoor moesten we het originele design van twee platformen reduceren tot een platform.
Het bouwen van de suit zorgde voor veel problemen en daardoor hebben wij niet genoeg tijd kunnen besteden aan het programmeren zelf. In het vervolg is het noodzakelijk meer te testen om te zeker te weten of het mogelijk is dat de robot zich kan voortbewegen.
In de derde versie van de software voor het ontwijken van obstakels (runrun3.js) zit ergens een bug die we wegens tijdsgebrek niet meer op konden sporen. Wanneer er een obstakel aan de voorkant van de kar is, draait de sphero correct naar rechts. Wanneer de sphero een beweging naar rechts gemaakt heeft en er een obstakel gedetecteerd is bevindt de sphero zich een een hoek. Om hier uit te kunnen komen zou de sphero naar links moeten bewegen in plaats van naar rechts. Dit gebeurd echter niet. Deze versie van de object ontwijk software hadden we eerder moeten schrijven zodat we meer tijd over zouden hebben voor het debuggen. Wegens onverwachte complicaties betreffende het ontwerp van de kar was dit helaas niet mogelijk.
Een demonstratie van onze robot

4 Appendix:

BB-8 by sphero.
OS X El Capitan.
Javascript NODE.JS compiler versie 4.45.
Python 2.7.0 .
Autodesk fusion 360 .
Weerstanden van 4.7k Ohm en 10k.
Breadboard
sensor
sensor connectie naar raspberry pi
raspberry pi 3 model B.
3D printer van de universiteit van amsterdam
sphero software development kit
node.js API
Code op gitbucket die gebruikt is om de sphero te bewegen

Een demonstratie van onze robot