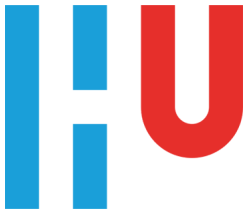
Hogeschool Utrecht



Group Project

Course code

Het besturingsalgoritme

Deelnemers:
Nick Snel
Ramon van Bemmel
Nathan Hoekstra
Cris van der Nolle
Marc Dirven

Docent:
Adrie Doesburg

April 13, 2018

Inhoudsopgave

1 Inleiding

In dit document wordt ingegaan op het door ons ontwikkelde algoritme. Onze robot (gebouwd op LEGO mindstorms) wordt bestuurd door een programma dat draait op een Raspberry Pi en input geeft aan de robot via een BrickPi shield.
De bedoeling van de robot is dat hij volledig autonoom een zwarte lijn volgt op een wit veld. Hij moet dus ook bochten kunnen draaien, objecten ontwijken en de lijn terug kunnen vinden.
De BrickPi dient voor ons als interface met sensoren en actuatoren op de robot. De robot zelf is een lego mindstorms robot, maar i.p.v. de meegeleverde computer van mindstorms gebruiken we dus bovengenoemde configuratie van Raspberry Pi plus een BrickPi shield. Op het BrickPi shield zitten MMJ aansluitingen om via een serieel protocol met de motoren en sensoren te kunnen communiceren.

2 Doelen

Het algoritme moet zo werken dat de lijnvolger:

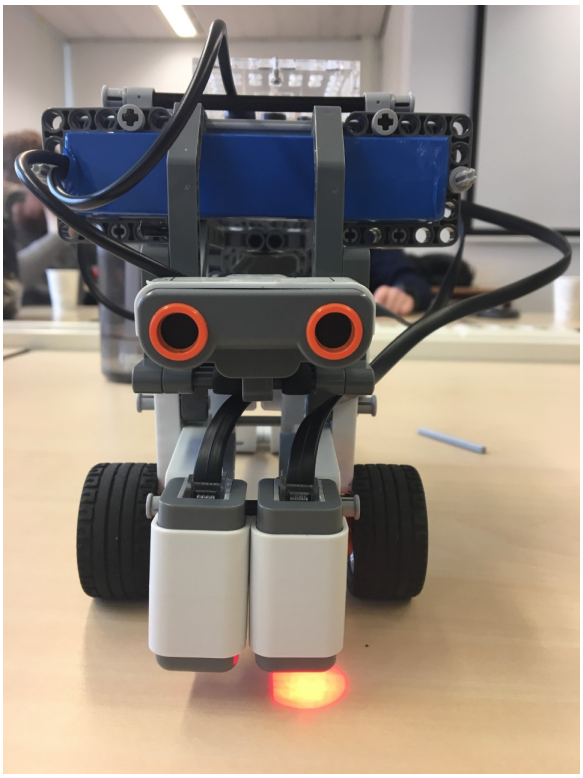
- Een lijn kan zien door middel van een zwart-wit contrast.
- Op een lijn kan blijven door zijn rijrichting aan te passen.
- Bij een bocht kan afslaan.
- Obstakels op de lijn kan ontwijken.
- Zijn weg terug kan vinden naar de lijn.
- Kan navigeren door een grid.

3 Ontwerp

In dit hoofdstuk zullen we aan de hand van een aantal diagrammen het ontwerp omschrijven.

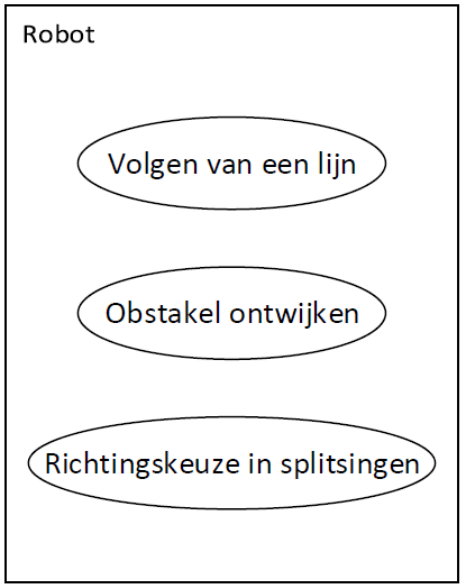
3.1 De Robot

De robot is net als alle andere robots gebouwd in Lego Mindstorms. Zie hieronder voor onze robot. Hij heet "Tesla Model S".



Figuur 1: "Model S"

Hierbij is de rechter sensor (t.o.v. de rijrichting) de lichtsensor en de linker sensor de kleurensensor. De brede lange sensor aan de bovenkant met de twee oogjes is de ultrasoonsensor.



Figuur 2: In deze figuur is te zien welke activiteiten de robot kan.

3.2 Use Case Diagram

De robot moet dus een zwarte lijn kunnen volgen, een obstakel kunnen ontwijken op een parcours en een richtingskeuze moeten kunnen maken in splitsingen. We zullen aan de hand van een activity diagram laten zien hoe dit gerealiseerd wordt.
Er is geen interactie met andere actoren aangezien deze robot geheel zelfstandig is. In principe is het enige dat er door een persoon moet gebeuren, het plaatsen van de robot op het bord. Maar dat hebben we hier buiten beschouwing gelaten.

3.3 Activity Diagram

In dit kader beschrijven we in het kort hoe we globaal het algoritme hebben ontworpen.

4 Algoritme

4.1 Motoren

De motoren zijn een belangrijk onderdeel van de robot. De motoren zorgen voor bochten, maar ook voor de snelheid voorwaarts. In deze paragraaf zullen we kort de werking beschrijven.

4.1.1 Sturing

Het maken van bochten wordt proportioneel gedaan. We kunnen de lichtwaardes meten en kijken wat wit, zwart en gecombineerd wit/zwart is. De x in onderstaande tabel is een gemeten waarde. De bevindingen zijn als volgt:

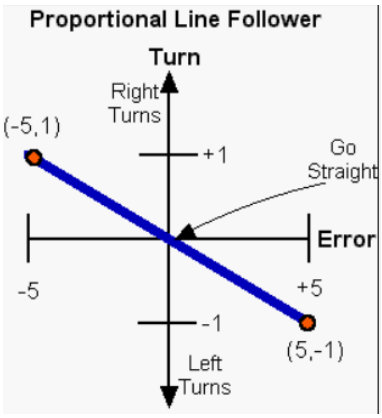
Sensor waardes		
	Donker	**Licht**
Licht sensor	`x > 2000`	`x < 2000`
RGB sensor	`x < 450`	`x > 450`

De volgende formule is gebruikt voor het proportioneel bijsturen van de motoren in de bochten:

$$turn = K_p \times error$$

Hierbij is *turn* de bocht, K_p de proportionele constante en *error* de afwijking ten opzichte van de lijn. Je normaliseert de uitgelezen waarde van een sensor door de waarde af te trekken van de waarde die je wilt. Hierdoor krijg je een error.

Negatief betekent van de lijn af sturen en positief betekent naar de lijn toe sturen. Zie hieronder voor een voorbeeld:



Figuur 3: Proportionele terugkoppeling

4.1.2 Code voorbeeld

Ook zullen we een heel klein eenvoudig stukje code bekijken die beide motoren laat draaien. Het autootje zal vooruit gaat bij het uitvoeren van deze functie.

```
#include "motor_control.hpp"
#include "BrickPi3.h"
#include <stdio.h>
#include <unistd.h>

BrickPi3 MyBP; // new instance of BrickPi3 class
int debug_motor = 0;

void run_motor(int left_power, int right_power)
{
    MyBP.set_motor_power(PORT_C, right_power);
    MyBP.set_motor_power(PORT_B, left_power);
}
```

Bij de functie `void run_motor(int left_power, int right_power)` zal er bij een gegeven `int left_power` en `int right_power` een afstand worden afgelegd.

4.2 Sensoren

Er worden twee sensoren gebruikt voor het meten van de zwart wit waardes zoals te zien in bovenstaande tabel. Tot slot is er nog een sensor voor het meten van een object.

4.2.1 RGB

Een RGB (Rood Groen Blauw) sensor, is een sensor die de drie zojuist genoemde kleuren projecteerd op een oppervlak. De sensor meet vervolgens de gereflecteerde kleuren en meet hieruit wat de waarde is. In dit project gebruiken wij alleen het rode licht. Het is namelijk in te stellen welk van de drie lichten je wilt gebruiken. Hierbij mag het voorzich spreken dat de waardes ook anders zijn als je dus andere lichten op een oppervlak projecteerd. Uiteraard is hier rekening mee gehouden.

4.2.2 Lichtsensor

Een lichtsensor meet de intensiteit van licht. Omdat zwart, licht absorbeert en wit, licht reflecteerd, is hieruit op te maken hoe licht of donker een bepaald oppervlak is. De waarde wordt hoger naarmate een oppervlak donkerder wordt en hoger als een oppervlak lichter wordt.

4.2.3 Ultrasoonsensor

Tot slot is er nog de ultrasoon sensor. Deze sensor meet aan de hand van een uitgezonden geluidssignaal de afstand tussen een potentieel object en zichzelf. Hiermee kunnen we een object dus herkennen. De volgende formule speelt hierbij een rol:

$$v_{geluid} = 340\ m/s$$
$$s_{lengte} = v_{geluid} \times (t/2)$$

Hieronder zullen we een kort stukje pseudocode beschrijven van het ontwijken van een object in een normaal parcours.

```
int pre_defined_value = 10 // set the value to 10 cm
while (true)
{
    if (ultrasonic_sensor > pre_defined_value)
    {
       dodge_cyclus(); // function to dodge the object
    }
}
```