

11/30/2010



RAMON H
&
YORAN S

EEN ROBOT ALS MENS

Inhoud

Inleiding.....	3
De technische middelen.....	4
De programmeertaal NXC.....	7
Eigen keuzes van de robot.	11
Hoe mensen zouden reageren op de proefopstelling	13
Hoe de robot zal reageren op de proefstelling.....	15
Gedrag van de robot ten opzichte van de mens.....	19
Conclusie.....	22
Bronnen.....	23
Logboek.....	24

Inleiding

Toen wij, Ramon Houtsma en Yoran Sturkenboom, bij de presentatie waren over het maken van het profielwerkstuk dachten we al dat we iets met informatica wilden gaan doen. We kwamen op het idee om iets te gaan doen met robots. Maar wat we precies wilden doen wisten we nog niet. Toen we eenmaal wat onderzoek hadden gedaan op het internet kwamen we op het idee om iets met kunstmatige intelligentie te gaan doen.

Robots met menselijke eigenschappen. Dat is waar wij aan dachten toen wij begonnen na te denken over de hoofdvraag. Wij zaten met vragen in ons hoofd als “Wat is artificial intelligence?” en “Welke rol speelt artificial intelligence in ons leven?” , maar dit was allemaal veel te breed. Toen wij eenmaal met dhr. Krol gepraat hadden wisten we het. We gaan een eigen robot bouwen en programmeren met menselijke eigenschappen. Uiteindelijk zijn wij op de hoofdvraag “Hoe kunnen wij een robot menselijke intelligentie laten nabootsen?”. Onze hypothese hierbij is: “Door middel van gebruik van de ultrasone sensor, de druksensor, de lichtsensoren, de geluidssensoren en de programmeertaal die wij geleerd hebben, kunnen wij onze robot zo op obstakels laten reageren als een mens zou doen.”. Zo zijn wij ons werkstuk gestart.

We hebben het werk verdeeld maar we hebben ook een groot deel samen gedaan. We hebben eerst een aantal uren besteed aan het in elkaar zetten van de robot, en samen wat tests gedaan. Ook hebben we onze eerste programma's samen geschreven. De zes deelvragen hebben we verdeeld. Yoran zou drie vragen gaan uitwerken en Ramon zou de andere drie vragen gaan uitwerken.

Uiteindelijk hebben we een hoop kunnen doen met de robot die dhr. Krol voor ons besteld heeft. We hebben goede testresultaten gehad en hebben die in dit werkstuk uitgewerkt. Veel plezier met het lezen van dit profielwerkstuk!

De technische middelen

Allereerst zullen we een robot nodig hebben om menselijke intelligentie na te bootsen. De robot die wij voor het onderzoek hebben gebruikt is de NXT van LEGO Mindstorms. Hieronder is de versie te zien die wij hebben gebouwd.



De robot beschikt onder andere over drie motoren. De eerste motor wordt gebruikt om de 'arm' van de robot aan te sturen, links in beeld op de foto. Met behulp van de motor zal deze arm bijvoorbeeld in staat zijn om een balletje te slaan. De tweede en derde motor worden gebruikt om de wielen aan te drijven.

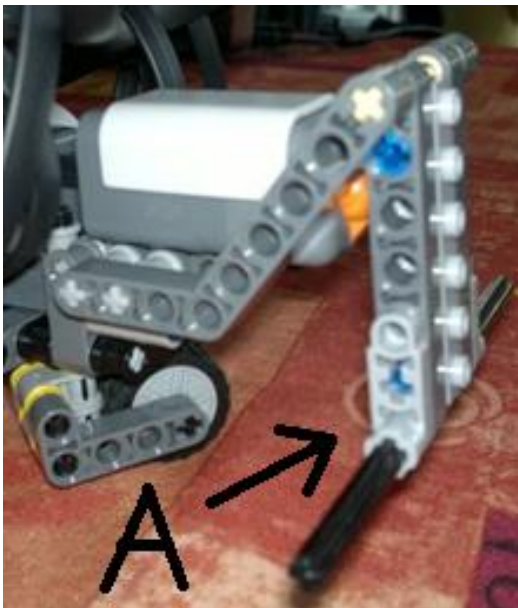
Maar voordat een robot menselijk gedrag zal kunnen nabootsen, zal hij eerst, net als een mens, op verschillende manieren zijn omgeving moeten kunnen waarnemen; we zullen hem dus 'zintuigen' moeten geven. Hij zal moeten kunnen reageren op wat hij waarneemt. Om de robot de omgeving te kunnen laten waarnemen, gebruiken we sensoren. Wij gebruiken voor onze robot vier soorten sensoren, namelijk:

- De geluidssensor
- De druksensor
- De ultrasone sensor
- De lichtsensor

Hoe deze sensoren werken zal op de volgende pagina's worden uitgelegd.

Geluidssensor

Om de robot te kunnen laten horen, gebruiken we een geluidssensor, die is te zien op de foto hiernaast. De geluidssensor kan de geluidsintensiteit meten, geluidspatronen herkennen en is daarom dus het oor van de robot. De geluidsintensiteit kan worden gemeten op een schaal van 0 tot 100, zodat we er in een programma ook daadwerkelijk wat mee kunnen. Als de waarde 0 is meet de sensor geen geluid, en als de waarde 100 is, dan is de waarde van het gemeten geluid maximaal. De informatie die de geluidssensor verzamelt, worden via een zogeheten 'connector cable' doorgegeven aan de NXT. Een connector cable is de kabel die een sensor of motor kan verbinden met de NXT. Doordat de robot eigenlijk maar één oor heeft, is het niet mogelijk om te horen waar het geluid vandaag komt.



Druksensor

Op de foto links is de druksensor te zien. Het oranje knopje kan worden ingedrukt door de beweegbare onderdeel A. Op het moment dat de robot dus ergens tegenaan rijdt, wordt de druksensor via onderdeel A ingedrukt. Dit signaal wordt vervolgens weer doorgestuurd via de connector cable naar de NXT. De druksensor geeft altijd een waarde 0 of 1; 0 betekent dat hij niet is ingedrukt en 1 betekent dat hij wel wordt ingedrukt.

Ten slotte willen we nog dat de robot kan zien. Daar hebben we echter twee sensoren voor nodig. We willen immers dat hij niet alleen licht kan zien, maar ook hoe de omgeving eruit ziet. Om de robot dus te kunnen laten zien combineren we twee sensoren met elkaar; de lichtsensor en de ultrasone sensor.

Ultrasone sensor

De ultrasone sensor staat rechts afgebeeld. Met behulp van de ultrasone sensor is de robot in staat om te bepalen hoever een voorwerp van de sensor verwijderd is. De sensor doet dat op dezelfde manier als de manier waarmee vleermuizen kunnen zien in het donker. Hij genereert geluidsgolven van een hoge frequentie, die worden vervolgens weerkaatst door een voorwerp, en die weerkaatste golven vangt hij weer op. Met behulp van de snelheid van het geluid en de tijd die de golven er over doen om weer terug in de sensor te komen, kan worden uitgerekend hoever een voorwerp van de sensor verwijderd is. Ook deze sensor zit via een connector aangesloten op de NXT.



Lichtsensoren

Dan is er ten slotte nog de lichtsensor, zoals hieronder in twee verschillende standen is afgebeeld. De lichtsensor kan worden gebruikt om het verschil te zien tussen licht en donker en het kan de lichtintensiteit in de kamer of van verschillende kleuren meten. De licht intensiteit wordt gemeten op een schaal van 0 tot 100, dus als de lichtsensor een waarde 0 geeft, meet hij geen licht. De lichtsensor kan bijvoorbeeld gebruikt worden om twee balletjes met verschillende kleuren te kunnen onderscheiden, zoals in het linkerplaatje zou kunnen, of hij kan bijvoorbeeld gebruikt worden om de robot een lijn te laten volgen, zoals in het rechterplaatje mogelijk zou zijn.



De programmeertaal NXC

De taal waar de NXT op draait heet NXC. Vol uit is dit "Not Exactly C". Dit is afgeleid van de programmeertaal C. C is een imperatieve programmeertaal die ontworpen is door Dennis Ritchie. Imperatief houdt in dat het programma wordt opgesteld in de vorm van opdrachten die gelijk uitgevoerd kunnen worden.

De werking van de taal is in principe simpel. Je schrijft een opdracht en de robot voert die uit. De taal heeft honderden verschillende commando's die je kunt gebruiken om de robot dingen te laten waarnemen of doen. Deze commando's staan allemaal in de API van NXC. API staat voor "application programming interface".

Er zijn verschillende soorten commando's in de taal. Zo zijn er een aantal standaard definities die in meerdere programmeertalen terug komen zoals "if" en "else", maar ook zijn er definities die specifiek voor deze programmeertaal zijn gemaakt.

Commentaar

In NXC kan je tussen de geschreven stukken programmeertaal commentaar plaatsen. Dit doet men om toelichting te geven op het geschreven programma. Dit maakt het voor andere mensen die het moeten lezen makelijker om het te volgen. Commentaar kan op twee verschillende manieren worden toegepast. De eerste manier is de manier die in de "C" programmeertaal wordt toegepast. Dit doe je door het commentaar tussen "/*" en "*/" te plaatsen. Deze manier wordt gebruikt om commentaar over meerdere regels te plaatsen.

```
01 /* Dit is de eerste vorm van commentaar */
02 /* Maar het kan ook
03 Over 2 regels */
```

De tweede manier is de manier die in "C++" wordt gebruikt. Hierbij wordt het commentaar geplaatst na "//". Bij deze manier moet voor elke regels "//" worden geplaatst en kan je dus niet doorgaan op de volgende regel zonder weer eerst "//" te plaatsen.

```
01 //Dit is de tweede vorm commentaar
```

Witruimtes

In NXC kan je net zoals in C en C++ ook spaties en tabs gebruiken om het geschreven programma overzichtelijker te maken. Deze witruimtes zullen niets veranderen aan de werking van het geschreven programma tenzij ze 2 karakters van een operator zoals ">>" scheiden.

```
01 x=5; //Dit zet de waarde van x naar 5
02 x = 5; //Dit zet eveneens de waarde van x naar 5
03 x>>5; //Dit betekent x is groter dan 5.
04 x> >5; //Dit zal resulteren in een fout tijdens het compileren.
```

Numerieke constanten

Numerieke constanten kunnen in NXC op verschillende manieren worden geschreven. Ze kunnen in het decimale en hexadecimale stelsel geschreven worden. Als cijfers in het decimale stelsel geschreven worden wordt het gewone getal uitgeschreven en kan er een punt worden gebruikt om getallen achter de komma aan te duiden. Hexadecimale getallen beginnen met "0x" en worden gevolgd door het hexadecimale getal.

```
01 x = 10; //Dit zet de waarde van x naar 10
02 x = 0x10; //Dit zet de waarde van x naar 16
03 x = 10.10; //Dit zet de waarde van x naar 10,10
```

String constanten

String constanten worden in NXC eenvoudig weergegeven. Alle strings moeten tussen dubbele apostrofs geschreven worden.

```
01 TextOut(0, LCD_LINE1, "testing"); //Dit laat de text "testing" zien
```

Karakter constanten

Karakter constanten lijken erg op string constanten. Dit zijn constanten van slechts één karakter zoals "x". Deze moeten tussen enkele apostrofs geschreven worden

```
01 x = 'a' //Dit zet de waarde van x naar a
```

Sleutelwoorden

In NXC worden verschillende sleutelwoorden gebruikt. Dit zijn door de programmeur gekozen woorden waaraan gerefereerd kan worden. Het wordt ook gebruikt om functies en routines te maken die later in het programma weer terug moeten komen. Een aantal sleutelwoorden in NXC zijn hetzelfde als die uit "C" en "C++". Maar ook zijn er een aantal nieuwe sleutelwoorden in NXC Een aantal bekende sleutelwoorden zijn het if-statement en het while-statement.

```
01 if x==10 //Hier wordt gecontroleerd of x 10 is.
02 {
03 OnFwd(OUT_BC, 50); //Als dat zo is wordt deze actie gestart.
04 }
```

```
01 while (true) //Zolang de waarde true is (in dit geval altijd)
02 {
03 OnFwd(OUT_BC, 50); //Wordt deze actie uitgevoerd.
04 }
```

Structuur

Een NXC programma is samengesteld uit codeblokken en variabelen. Er zijn 2 verschillende soorten codeblokken. Dit zijn taken en functies. Het maximum aantal codeblokken is 256. Er wordt hierbij geen verschil gemaakt tussen codeblokken als taak of codeblokken als functie.

Elk NXC programma moet ten minste één taak hebben die main heet. Dit wordt zo geschreven.

```
01  task main ()
02  {
03  //Code voor in de taak.
04  }
```

Deze taak zal altijd als eerste uitgevoerd worden en vanuit deze taak kan je ook weer andere taken en functies starten. Als er meerdere taken tegelijkertijd uitgevoerd worden, en een bepaalde taak wil een functie uitvoeren die al uitgevoerd wordt kan dit leiden tot fouten en onvoorspelbaar gedrag van de robot.

Vanuit de taak “main” kun je dus andere taken uitvoeren. Dit kan door ze in de main taak aan te roepen. Zo zou je bijvoorbeeld een compleet programma kunnen draaien met meerdere taken zonder dat er ook maar 1 specifieke opdracht in de main taak staat. Het maakt hierbij niet uit in welke volgorde de taken staan geschreven. Het programma zal altijd starten bij de main taak en zal van daar alle andere taken kunnen vinden. Taken kunnen vanuit andere taken worden gestart door middel van “precedes” en “follows”. Precedes betekent voorafgaand en follows betekent opvolgend. Hieronder volgt een voorbeeld.

```
01  task rij() {
02  while(true)
03  {
04  OnFwd(OUT_BC, 50);
05  }
06  }
07
08  task main () {
09  precedes(rij,muziek);
10  }
11
12  task muziek() {
13  while (true)
14  {
15  PlayTone(TONE_A4, MS_500);
16  Wait(MS_600);
17  }
18  }
```

In dit programma zijn 3 taken geschreven. Taak “main” start taak “rijd” en taak “muziek”. Taak “rijd” zorgt ervoor dat de robot constant vooruit blijft rijden en taak “muziek” zorgt er voor dat hij een deuntje speelt.

Variabelen

In NXC zijn er verschillende typen variabelen. Een aantal bekende hiervan zijn integers, en strings. Hieronder staan enkele voorbeelden.

```
01 int x;      /*Dit definieert de variabele "x". Het kan alleen een
02 heel getal zijn. */
03 string ss = "Spongebob Squarepants";
```

Bovenstaande code laat van x alleen weten dat de variabele er is maar heeft hem nog geen waarde gegeven. Om x gelijk een waarde te geven kan er gewoon "=" teken na de variabele geplaatst worden om hem een waarde te geven.

Er zijn 2 soorten variabelen, globale variabelen en locale variabelen. Globale variabelen zijn variabelen die door het hele programma gebruikt kunnen worden. Deze variabelen dienen buiten een codeblok gedefinieerd te worden. Locale variabelen zijn variabelen die alleen door bepaalde taken gebruikt kunnen worden. Deze variabelen moeten binnen een codeblok gedefinieerd worden.

Operators

Er zijn verschillende operators in NXC. Deze worden gebruikt om bijvoorbeeld waarden te vergelijken of om waarden aan te passen. Hieronder staan een aantal manieren waarop deze operators gebruikt worden.

```
01 if x = y {...} //Hier wordt gekeken of x gelijk is aan y.
02 if x != y {...} //Hier wordt gekeken of x niet gelijk is aan y.
03 if x > y {...} //Hier wordt gekeken of x groter is dan y.
04 if x <= y {...} //Hier wordt gekeken of x kleiner of gelijk is aan y.
05 x += 2 //Hier wordt er 2 bij de variabele x opgeteld.
06 x *= 2 //Hier wordt x vermenigvuldigt met 2.
```

Er zijn een groot aantal verschillende operators. Dit is slechts een klein deel.

Eigen keuzes van de robot.

Je kunt de robot op verschillende manieren zelf keuzes laten maken. Dit doe je met behulp van verschillende subroutines in NXC. Het is voor ons de bedoeling om een programma te schrijven waar bij de robot constant rijdt maar ook constant controleert of zijn omgeving niet die tekenen vertoont waarbij hij zijn gedrag aan moet passen.

Keuze door ons bepaald

Dit kan je dus maken door verschillende subroutines te schrijven. Hierbij is het while statement erg handig en dit zal dus ook zeker gebruikt worden. Wij willen dus de onderstaande code gebruiken om te zorgen dat de robot constant rijdt.

```
01 task main()
02
03 {
04 OnFwd(OUT_A, 20);
05 }
```

Hierna willen wij invoegen dat de robot constant controleert of hij niet over de rand van de tafel rijdt. De robot mag niet van de tafel vallen. Dit doen we door de onderstaande code in te voegen.

```
01 #define THRESHOLD 40
02 task main()
03 {
04 SetSensorLight(IN_3);
05 OnFwd(OUT_BC, 20);
06 while(true)
07 {
08 if (Sensor(IN_3) < THRESHOLD)
09 {
10 OnRev(OUT_BC, 20);
11 Wait(100);
12 until(Sensor(IN_3) >= THRESHOLD);
13 OnFwd(OUT_BC, 20);
14 }
15 }
16 }
```

Door deze code toe te voegen zullen we de robot dus constant laten controleren of hij niet over de zwarte lijn rijdt op de rand van de tafel. Eerst wordt de variabele THRESHOLD een waarde gegeven. Daarna wordt de lichtsensor geactiveerd. De robot zal nu gewoon gaan rijden zoals altijd maar zal ondertussen ook constant controleren op wat er tussen de haken van het while-statement geschreven is. Hier wordt constant gecontroleerd of de lichtintensiteit kleiner is dan de minimale waarde. Als de robot over de zwarte lijn komt zal de lichtintensiteit onder de minimale waarde komen en zal hij dus doen wat er tussen de haken van het if statement staat. Hierin gaat hij voor 0.1 seconde achteruit en dit zal hij blijven doen totdat de lichtintensiteit gelijk aan of groter is dan de minimale waarde. Daarna gaat de robot weer gewoon verder waarmee hij bezig was, namelijk vooruit rijden.

In feite maakt de robot dus niet echt een “eigen keuze” maar hebben wij de robot mogelijkheden gegeven om uit te kiezen. Door middel van gebruik van de sensoren moet de robot zelf de beslissing kunnen maken om een bepaalde actie uit te voeren.

De lichtsensor is niet de enige sensor waar we gebruik van kunnen maken. We kunnen van alle sensoren tegelijk gebruik maken binnen 1 programma. Dit wordt gedaan met behulp van het if en het while-statement.

Door middel van het while-statement zal de robot constant controles blijven doen die met het if-statement gedefinieerd zijn.

Eigen keuze

We kunnen de robot wel een soort van eigen keuze laten maken met behulp van het “random” sleutelwoord. Met behulp van dit sleutelwoord kunnen we de robot (uit een beperkt) aantal opties laten kiezen om te doen. Zo kunnen we de robot dus bijvoorbeeld in een ruimte laten rijden waar die om de zoveel seconden een bepaalde bocht neemt. Hieronder staat een voorbeeld

```
01 task main() {
02 while (true) {
03   OnFwd(OUT_BC, Random(100));
04   Wait(1000);
05 }
06 }
```

Met deze code laten we de robot constant voor één seconde met een bepaalde snelheid rijden. De snelheid zal na elke seconde willekeurig opnieuw gekozen worden op een schaal van 0 tot 100. Op deze manier kan de robot zelf een soort van keuze maken wat hij wil doen. Het random sleutelwoord is op veel meer manieren toe te passen en kan dus gebruikt worden om een robot een persoonlijkheid te geven die niet constant hetzelfde doet.

Hoe mensen zouden reageren op de proefopstelling

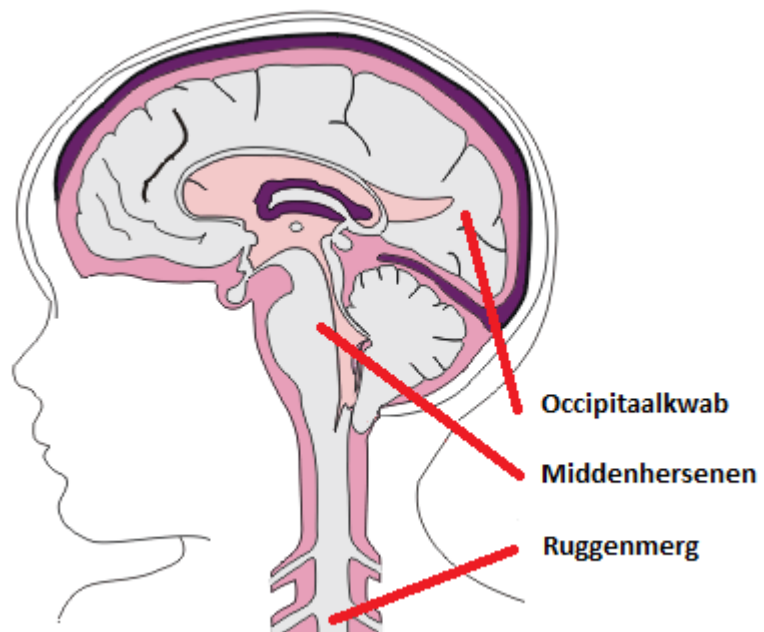
Om te kunnen kijken of het mogelijk hoe we menselijke intelligentie kunnen nabootsen met onze robot, hebben we een proefopstelling gemaakt. Nadat we hebben onderzocht hoe mensen zouden reageren op de proefopstelling, en hoe we onze robot kunnen laten reageren op de proefopstelling, kunnen we die resultaten gaan vergelijken.

De proefopstelling die we gebouwd hebben voor de robot bestaat uit een plaat van drie meter lang en anderhalve meter breed, waar we de robot op zullen zetten en over zullen laten rijden. De ondergrond is wit, en langs de rand loopt een zwarte streep. Op die plaat liggen of staan een aantal obstakels, waar de robot slim mee om zal moeten gaan.

Maar eerst willen we weten hoe een mens op de omstandigheden zou reageren, anders hebben we straks geen materiaal om de resultaten van de robot mee te vergelijken. Stel dat we een mens op een vergrootte versie van de proefopstelling zouden laten lopen, hoe zou hij dan reageren op bepaalde situaties?

Afgrond

Het eerste obstakel dat de mens terwijl hij loopt wellicht tegen zal komen is de afgrond. Het beeld dat in het oog op het netvlies valt, worden door twee soorten fotoreceptoren omgezet in elektrische signalen. Er zijn twee soorten fotoreceptoren: kegels en staafjes. Met de kegels kunnen we bij daglicht zien, details zien en kleuren onderscheiden, terwijl we met de staafjes 's nachts kunnen zien, minder details zien en geen kleuren kunnen zien. De fotoreceptoren zetten het beeld om in elektrische signalen. Deze elektrische signalen worden vervolgens via de optische zenuw, die de verbinding vormt tussen het oog en de hersenen, naar de hersenen verzonden, waar de signalen ten slotte zullen worden omgezet in beeldinterpretaties. Dit gebeurt in de occipitaalkwab. In de mesencephalon, oftewel de middenhersenen, wordt wat gedaan met die informatie. De middenhersenen zijn namelijk betrokken bij de zintuiglijke en motorische functies. Die zullen vervolgens via het ruggenmerg signalen sturen naar het lichaam, en wel het signaal dat het lichaam moet stoppen met lopen, anders zou de persoon de afgrond in lopen. Op het moment dat een mens dus de afgrond in dreigt te lopen, zorgen de hersenen ervoor dat dat niet gebeurt.



Obstakel voor de mens

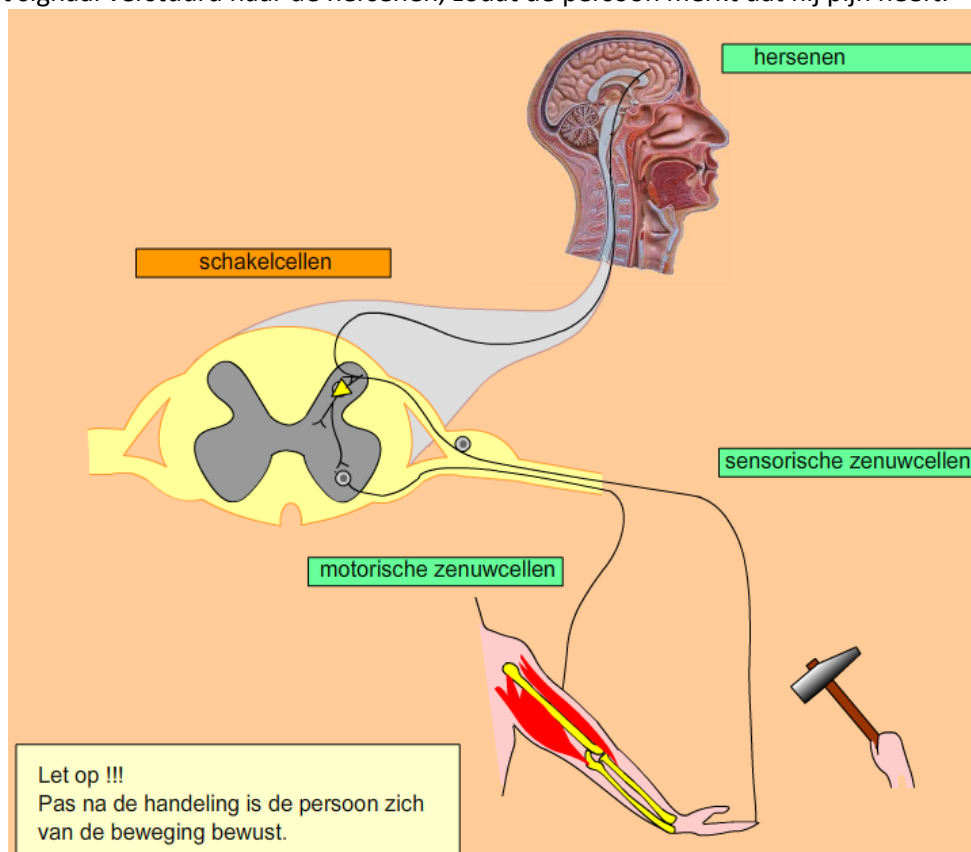
In onze proefopstelling voor de robot zullen we ook obstakels plaatsen op de plaat. Wat zou er gebeuren als een mens op een obstakel stuit terwijl hij vooruit loopt? Stel dat hij vooruit loopt en er staat een tafel in zijn pad. Op het moment dat hij de tafel in zicht krijgt, zal hij de afstand tot de tafel inschatten, zodat hij weet hoever en hoelang hij nog door kan lopen. Daarna zal hij natuurlijk niet doorlopen, want dat zal niet gaan. Hij zal daarentegen om de tafel heen gaan lopen. Al deze handelingen worden, ook al heeft hij het zelf niet door, eigenlijk door de hersenen bepaald.

Obstakel achter de mens

Maar wat nou als de mens achteruit aan het lopen is? Omdat de ogen aan de voorkant van het hoofd zitten, zal hij niet in staat zijn om te zien hoe de omgeving achter hem eruit ziet. Als er nu wederom een tafel in zijn pad staat, zal hij niet in staat zijn om van te voren te zien dat die tafel er ook echt staat. Hij zal er dus in ieder geval tegenaan lopen. Op het moment dat hij tegen het obstakel aan loopt, zal hij natuurlijk niet proberen om alsnog door te blijven lopen. Hij zal dan stoppen met lopen, stilstaan dus. Dit is mogelijk doordat bij de aanraking van de tafel, er een signaal naar de hersenen wordt gestuurd. Dat is de rede dat hij merkt dat hij ergens tegenaan loopt. De hersenen verwerken vervolgens dit signaal en sturen dan signalen naar het lichaam dat het moet stoppen met lopen.

Schrik!

Dan is er ten slotte nog het laatste obstakel, en dat is een hele harde knal. Omdat de mens niet op dit geluid is voorbereid, zal hij schrikken. Omdat een schrikreactie een reflex is, wordt er dit keer niet via de hersenen gewerkt. Bij een reflex wordt er eerst gehandeld, voordat de hersenen doorhebben wat er aan de hand is. Als de persoon bijvoorbeeld op zijn hand geslagen wordt met een hamer, dan wordt eerst door de motorische zenuwcellen bepaald dat hij zijn hand terug trekt, en daarna pas wordt het signaal verstuurd naar de hersenen, zodat de persoon merkt dat hij pijn heeft.



Hoe de robot zal reageren op de proefstelling

Afgrond

Het belangrijkste obstakel dat de robot zal moeten ontwijken is de rand. Als we de robot geen enkele vorm van intelligentie zouden geven, zou hij gewoon rechtdoor blijven rijden, zo de afgrond in. Het resultaat: robot kapot. Dat willen we dus vermijden. Gelukkig is dat mogelijk, dankzij de lichtsensor. Omdat de lichtsensor in staat is de lichtintensiteit te meten, zal hij dus ook in staat zijn het verschil te zien tussen wit en zwart, want die kleuren geven verschillende lichtintensiteiten af. Dat komt mooi uit, want aangezien de rand zwart is en de rest van de ondergrond wit, zullen we in staat zijn de robot de zwarte lijn te laten herkennen, en vervolgens de robot de lijn te laten vermijden. We zullen hiervoor natuurlijk wel de lichtsensor op de grond moeten richten (zie pagina 6).

Om de robot de rand te laten herkennen en te laten vermijden, hebben we de volgende code geschreven:

```
01 #define THRESHOLD 40
02 task main()
03 {
04     SetSensorLight(IN_3);
05     OnFwd(OUT_BC, 20);
06     while (true)
07     {
08         if (Sensor(IN_3) < THRESHOLD)
09         {
10             OnRev(OUT_B, 20);
11             until (Sensor(IN_3) >= THRESHOLD);
12             OnFwd(OUT_BC, 20);
13         }
14     }
15 }
```

Eerst wordt in regel 01 de aan de variabele THRESHOLD de waarde 40 toegekend. Daarna vertellen we de NXT in regel 04 dat de lichtsensor aangesloten zit op poort 3. In regel 05 laten we de robot vooruit rijden. In regel 08 laten we de NXT continu checken of de lichtsensor een waarde lager dan 40 geeft, wat zou betekenen dat hij de zwarte lijn waarneemt. In regels 10 en 11 laten we de robot draaien door het rechterwiel achteruit te laten rijden totdat de lichtsensor weer boven de witte ondergrond staat. In regel 12 zeggen we vervolgens dat hij weer rechtdoor moet gaan rijden, en deze cyclus zal oneindig worden herhaald, omdat het in een while(true) statement staat (regel 06 t/m 14).

Dankzij dit programmaatje zal de robot dus nooit meer over de rand heen rijden, en zal hij dus wat er ook gebeurt op de plaat blijven.

Obstakels voor de robot

We hebben ook obstakels op de plaat geplaatst. Nu willen we er natuurlijk voor zorgen dat de robot niet tegen een dergelijk obstakel aan zal rijden. De robot is in staat om het voorwerp te zien met de ultrasone sensor die op de NXT geplaatst is, zoals beschreven op pagina 6. Dus zal het ook mogelijk zijn om het voorwerp te ontwijken. Dat laten we de robot doen met het volgende programma:

```
01 #define NEAR 25 //cm
02 #define AWAY 30 //cm
03 task main()
04 {
05     SetSensorLowspeed(IN_4);
06     while(true)
07     {
08         OnFwd(OUT_BC,25);
09         if (SensorUS(IN_4)<NEAR)
10         {
11             OnRev(OUT_C,25);
12             until (SensorUS(IN_4)>AWAY);
13             Wait(200);
14         }
15     }
16 }
```

Eerst wordt in regel 01 aan de variabele NEAR de waarde 25 gegeven en in regel 02 de waarde 30 aan de variabele AWAY (in cm in deze gevallen, omdat het om de ultrasone sensor gaat). Vervolgens wordt in regel 05 aan de NXT verteld dat de ultrasone sensor aangesloten zit op poort 4, dus dat daar de gegevens vandaan gehaald moeten worden. Daarna laten we de robot in regel 08 oneindig vooruit rijden, en controleren wanneer de waarde die door de ultrasone sensor gemeten wordt groter is dan 25. Als de waarde groter is dan 25, betekent dat dat de ultrasone sensor een voorwerp ziet dat minder dan 25 cm verwijderd is van de sensor zelf. Op het moment dat dat zo is, zal de langzaam naar links blijven draaien totdat het voorwerp uit beeld is (regel 11 en 12). Daarna draait de robot nog 1/5^e seconde door. Als hij dat niet zou doen, dan zou hij dus weer vooruit gaan rijden omdat de ultrasone sensor geen voorwerp meer ziet, maar omdat de robot breder is dan de sensor zelf, zou de robot alsnog tegen het voorwerp aan rijden. Dankzij dit programma zal de robot dus nooit meer tegen een bordje of iets dergelijks aanrijden.

Obstakels achter de robot

Hoe moet dat nou als de robot achteruit rijdt, en er zit een obstakel in de weg? De ultrasone sensor kijkt alleen vooruit, en zal dus vanzelfsprekend alleen obstakels kunnen zien die voor hem opdoemen. Gelukkig hebben we nog de druksensor aan de achterkant van de robot. Op pagina 5 staat uitgelegd hoe die werkt. Met behulp van de druksensor kunnen we ervoor zorgen dat de robot merkt dat hij ergens tegenaan rijdt, zodat hij daarna een andere kant op kan gaan rijden. Dat doen we met het volgende programma:

```
01 task main()
02 {
03     SetSensorTouch (IN_1) ;
04     while (true)
05     {
06         if (SENSOR_1 == 1)
07         {
08             OnFwd (OUT_BC, 50) ;
09             Wait (250) ;
10             OnRev (OUT_B, 50) ;
11             Wait (1250) ;
12         }
13         else
14             OnRev (OUT_BC, 50) ;
15     }
}
```

Het is iets makkelijker om met de druksensor te programmeren, omdat de druksensor alleen een 1 of een 0 als waarde zal geven. Daardoor is het niet nodig om een variabele aan te maken. In regel 03 vertellen we de NXT dat de druksensor is aangesloten op poort 1. Omdat de druksensor aan de achterkant van de robot zit, kan die uiteraard alleen worden gebruikt als de robot achteruit ergens tegenaan rijdt, en daarom laten we de robot in regel 14 oneindig achteruit rijden, tenzij de druksensor een waarde 1 geeft. In regel 06 laten we de robot oneindig testen of de waarde die de druksensor geeft 0 of 1 is. Als de waarde 0 is, wordt de sensor niet ingedrukt dus kan de robot gewoon achteruit blijven rijden. Als dat niet zo is, dus als de robot wel ergens tegenaan rijdt, dan zal de robot eerst een kwart seconde vooruit rijden, zodat er ruimte is om te draaien (regels 8 en 9). Doordat het linkerwiel daarna gewoon vooruit blijft rijden, terwijl het rechtwiel voor één en een kwart seconde achteruit zal rijden, zal de robot op zijn plaats draaien, waardoor het daarna recht voor het voorwerp staat. Op dat moment zal de ultrasone sensor het voorwerp zien, waardoor het programma voor de ultrasone sensor in werking zal worden gesteld.

Schrik!

Ten slotte kunnen we de robot nog laten 'schrikken'. Ook dat is mogelijk met de programmeertaal NXC:

```
01 #define THRESHOLD_1 80
02 #define THRESHOLD_2 40
03 #define MIC SENSOR_2
04
05 task main()
06 {
07     SetSensorSound(IN_2);
08     while(true)
09     {
10         until(MIC > THRESHOLD_1);
11         OnFwd(OUT_C, 100);
12         OnRev(OUT_B, 100);
13         Wait(250);
14         OnFwd(OUT_B, 100);
15         OnRev(OUT_C, 100);
16         Wait(500);
17         OnFwd(OUT_C, 100);
18         OnRev(OUT_B, 100);
19         Wait(250);
20         OnFwd(OUT_C, 50);
21         OnRev(OUT_B, 50);
22         Wait(500);
23         OnFwd(OUT_B, 50);
24         OnRev(OUT_C, 50);
25         Wait(1000);
26         OnFwd(OUT_C, 50);
27         OnRev(OUT_B, 50);
28         Wait(500);
29         Off(OUT_BC);
30
31         until(MIC > THRESHOLD_2 && MIC <= THRESHOLD_1);
32         OnFwd(OUT_C, 50);
33         OnRev(OUT_B, 50);
34         Wait(500);
35         OnFwd(OUT_B, 50);
36         OnRev(OUT_C, 50);
37         Wait(1000);
38         OnFwd(OUT_C, 50);
39         OnRev(OUT_B, 50);
40         Wait(500);
41         Off(OUT_BC);
42     }
43 }
```

In regel 03 zeggen we dat MIC is SENSOR_2. In regel 07 geven we aan de NXT door dat de geluidssensor op poort 2 zit. In de oneindige cyclus onder de while statement (regels 08 t/m 42) staat dat er niets gebeurt, totdat de geluidssensor een waarde waarneemt van hoger dan 40 op een schaal van 0 tot 100 (regel 08). Op het moment dat dat gebeurt, wordt de hele reeks commando's uitgevoerd die het laten lijken alsof de robot schrikt. Er zijn ook verschillende THRESHOLD levels zodat bij een zachte klap de robot een beetje schrikt en bij een harde klap heel erg.

Gedrag van de robot ten opzichte van de mens

De robot lijkt van buiten hetzelfde op onze proefopstelling te reageren als een mens, maar er zijn wel degelijk verschillen. Zoals eerder is uitgelegd, kan onze robot zien, horen en voelen. Dit doet hij door middel van de sensoren. Deze sensoren zijn enigszins te vergelijken met hoe de mens dingen waarneemt.

De lichtsensor

Het oog van de mens is te vergelijken met de lichtsensor en de ultrasone sensor van de robot. Hierbij is de lichtsensor te vergelijken met het vermogen van de mens om licht en donker van elkaar te scheiden. Valt er veel licht in de sensor van de robot, dan geeft dat een hoge waarde. Valt er weinig licht in de sensor van de robot, dan geeft dat een lage waarde. De robot vangt het licht op en zet de intensiteit om in een digitale waarde op een schaal van 0 tot 100. Als de waarde eenmaal is omgezet, wordt die via de connector cable verstuurd naar de NXT die de waarde zal verwerken en zo zal reageren op de waarde als wij hebben geprogrammeerd.

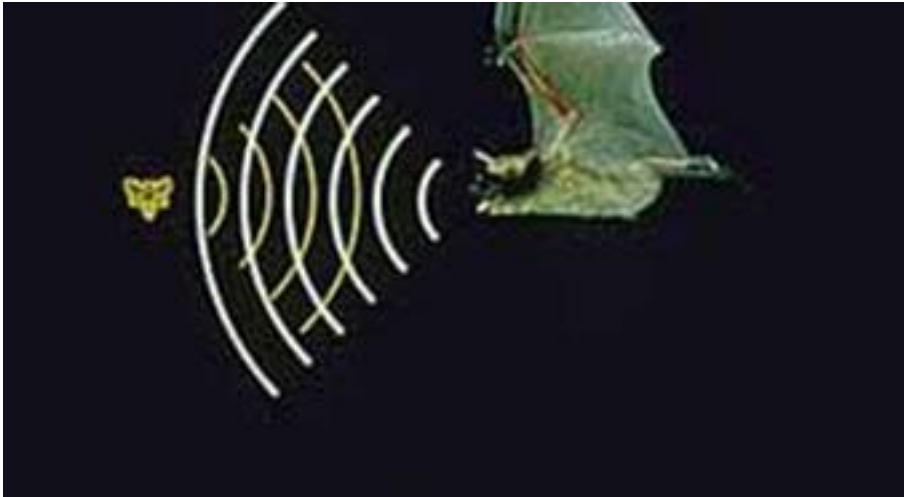


Dit werkt bij de mens vergelijkbaar. In het menselijk oog valt veel of weinig licht. Dit zal zorgen voor een specifiek signaal dat via de zenuwen naar de hersenen van de mens wordt gestuurd. In de hersenen aangekomen zal de mens dan gaan bedenken hoe die hier op reageert. De sensor zelf is te vergelijken met het oog. Beiden nemen ze een bepaalde hoeveelheid licht waar en zetten ze het om in een signaal. De connector cable is te vergelijken met de zenuw van de mens die het signaal verstuurd, en de NXT is te vergelijken met de hersenen van de mens.

De mens heeft ook een soort van geprogrammeerde reactie op een bepaalde hoeveelheid licht. Wanneer de mens bijvoorbeeld vanuit een donkere kamer een lichte kamer in gaat zal die automatisch zijn of haar ogen dichtknijpen. Dit is een reflex omdat de ogen eerst nog moeten wennen aan de hoge hoeveelheid licht in de nieuwe kamer. In wezen hebben wij dus een reflex geprogrammeerd voor de robot. Ook zal een mens als die eerst in een lichte kamer was en daarna in een donkere kamer stapt eerst even een stap terug doen zodat de ogen kunnen wennen aan het licht. Wij kunnen de robot precies hetzelfde laten doen. Als de robot over een zwarte lijn heen rijdt zal de sensor een lage waarde geven en zal de robot een stapje terug doen omdat hij niet de donkere ruimte in wil.

De ultrasone sensor

De ultrasone sensor is te vergelijken met het vermogen van de mens om diepte te zien. Het principe waarmee dit werkt is echter wel verschillend. De robot neemt diepte waar met behulp van ultrasone golven. Dit is hetzelfde principe waarmee vleermuizen kunnen zien. De ultrasone sensor zendt geluidsgolven uit die voor de mens niet hoorbaar zijn. Deze golven zullen uiteindelijk ergens tegenaan botsen en weerkaatst worden. De robot berekent hoe ver hij van een voorwerp is verwijderd door te kijken hoe lang het duurt voordat hij de golf weer opvangt. In feite gebruikt de robot dus een hoor orgaan om te zien.



Bij de mens werkt dit anders. De mens kan diepte zien omdat voorwerpen ook werkelijk diep staan en de mens dat kan zien. Voorwerpen die verder weg staan lijken kleiner en het licht dat ze reflecteren is minder goed waar te nemen voor het menselijk oog. Ook zal de hoeveelheid licht die van verschillende kanten van driedimensionale voorwerpen in het menselijk oog valt aan alle kanten van het voorwerp anders zijn. Hierdoor kan de mens diepte zien.

De manier waarop beide reageren op voorwerpen die voor hen staan is echter wel hetzelfde. Als de ultrasone sensor van de robot een lengte vanaf een voorwerp meet die kleiner is dan de minimale waarde die wij hebben ingesteld, zal die zich niet verder rijden en omkeren. Bij een mens werkt dit hetzelfde. Als een mens voor hem of haar een voorwerp ziet dat te dicht bij is om nog verder te lopen zal hij of haar (gewoonlijk) niet tegen het voorwerp aanlopen. Wij hebben de minimale afstand tot een voorwerp voor de robot moeten instellen waar een mens bij zichzelf kan nadenken of hij of zij nog verder moet lopen. Wij hebben het de robot dus moeten leren. De mens moet dit zichzelf leren gedurende de levensloop.

De druksensor

De druksensor is te vergelijken met het vermogen van de mens om te voelen. Het voelen bij de robot en voelen bij de mens werkt vergelijkbaar. De sensor van de robot is heel simpel. De waarde die hij geeft is 0 als hij niet wordt ingedrukt en 1 als hij wel wordt ingedrukt. Als de druksensor wordt ingedrukt stuurt die een signaal met de waarde 1 via de connector cable naar de NXT. Daar wordt gereageerd op de waarde zoals in het programma wordt beschreven.

Bij de mens is dit iets gecompliceerder. Een mens kan voelen door middel van minuscule zenuwuiteinden in bijvoorbeeld de vingers. Als die iets voelen wordt er een signaal naar de hersenen gestuurd. Dit signaal is bij de mensen niet gewoon een waarde van 1 of 0. Mensen kunnen namelijk hard of zacht voelen.

Wij hebben de robot op vergelijkbare manier laten reageren op voorwerpen die hem aanraken als mensen. De robot zal zodra de druksensor wordt ingedrukt een de andere kant op gaan. Bij mensen is dit vergelijkbaar. Als je als mens bijvoorbeeld je been stoot zal je naar achter lopen omdat je voelt dat het voorwerp van voren kwam en het (waarschijnlijk) pijn deed. De robot doet hetzelfde.

De geluidssensor

De geluidssensor is te vergelijken met de oren van de mens. De geluidssensor vangt geluid op en gebaseerd op de sterkte zet die dit om in een digitaal signaal op een schaal van 0 tot 100. Dit signaal stuurt de sensor naar de NXT en de NXT berekent dan met ons programma of er een actie uitgevoerd moet worden. Wij hebben echter geen 2 geluidssensoren dus wij kunnen de robot niet laten meten waar het geluid vandaan komt. Dit zou namelijk kunnen door te vergelijken hoe groot de waardes van beide geluidssensoren zijn.

Bij de mens werkt het vergelijkbaar. Mensen kunnen met hun oren geluidsgolven opvangen. Deze geluidsgolven komen op het trommelvlies dat hierdoor gaat trillen. De trillingen worden vertaald in signalen en via de zenuwen naar de hersenen geleid waardoor de mens kan horen.

Wanneer een mens een harde klap hoort zal die (waarschijnlijk) schrikken en kan die soms een onverwachte beweging maken als reflex op het geluid. Wij hebben de robot hetzelfde laten doen. De robot registreert het geluid in de ruimte constant en als er ineens een harde klap is zal de robot een soort van "opschrikken" en een rare beweging maken.

Conclusie

De mogelijkheden met deze robot zijn eindeloos. Als we alle tijd van de wereld hadden zouden we hem bij wijze van spreken door de hele school kunnen navigeren. Het is met onze middelen mogelijk om de robot zo te programmeren dat hij op bepaalde obstakels zo kan reageren dat dit menselijk lijkt. Het probleem wat wij hier ondervinden is dat de robot alleen weet om te gaan met de obstakels waar wij hem voor programmeren. Het is met onze middelen niet mogelijk om de robot zelf dingen te laten leren. Dit is een cruciale factor voor het menselijk functioneren.

Hoewel wij de robot niet zelf dingen kunnen laten leren, kunnen wij wel zo veel mogelijk zelf programmeren en de robot zo met onze intelligentie intelligent maken. Wij moeten de robot als het ware zelf alle menselijke trekjes leren. Net zoals ouders hun kinderen leren lopen, moeten wij de robot trucjes leren om met obstakels om te gaan. Met behulp van deze informatie en met de programma's die we hebben gemaakt zijn we tot deze conclusie gekomen. "We kunnen een robot menselijke intelligentie laten nabootsen door de 'hersenen' van de robot juist te programmeren, om gebruik te maken van de sensoren en menselijk te reageren."

Bronnen

Auteur	Jaar	Titel	Geraadpleegd op	Website
Essilor	-	Hoe werkt het oog?	29 nov. 10	http://www.essilor.nl/?page=61&lang=nl
Wikipedia	-	Horen	29 nov. 10	http://nl.wikipedia.org/wiki/Horen
Hersenstichting Nederland	-	Anatomie en functies van de hersenen	29 nov. 10	http://www.hersenstichting.nl/dyn_content/files/pdf/anatomie_en_functies_van_de_hersenen.pdf
Hispeed	-	Eye	30 nov. 10	http://t0.gstatic.com/images?q=tbn:1GR4pY-0jQ2veM:http://photocompetition.hispeed.ch/original/1340/mijn_oog/oog_foto_van_mij.jpg&t=1
Syraweb	-	NXT LightSensor	30 nov. 10	http://www.syraweb.org/images/NXTlightsensor.jpg
Bioplek	-	Reflexboog	30 nov. 10	http://www.bioplek.org/animaties/zenuwstelsel/reflexboog.html
Daniele Benedetti	2007	NXC Tutorial	Hele duur van profielwerkstuk.	http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_tutorial.pdf
BriccCC	2010	NXC Guide	Hele duur van profielwerkstuk.	http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_Guide.pdf

Logboek

Datum	Wat ging er goed/ Wat kan beter	Verrichte werkzaamheden	Tijd die we nodig hadden	Plaats	Wie
28-06-2010	Brainstormen ging goed. We hebben een hoofdvraag en deelvragen. Ook hebben we een hypothese opgesteld.	- Hoofd- en deelvragen geformuleerd. - Begin gemaakt aan logboek. - Begin gemaakt aan plan van aanpak.	1,5 uur	School	Ramon en Yoran
28-06-2010	Extra gebrainstormed over de hoofdvraag. We zijn er nog niet helemaal uit.	Gedacht over anders geformuleerde hoofdvraag.	45 minuten	Thuis	Ramon en Yoran
29-06-2010	We hadden dhr. Krol maandagochtend beter gelijk kunnen aanspreken.	- Contact gezocht met begeleider. - Afspraak voor een gesprek gemaakt met dhr. Krol.	10 minuten	Thuis	Ramon en Yoran
Zomervakantie		Handleiding gelezen over NXT Robot	2 uur	Thuis	Ramon en Yoran
15-09-2010	Gesprek is goed verlopen.	Gesprek met dhr. Krol.	10 minuten	School	Ramon en Yoran
17-09-2010	De NXT in elkaar zetten ging goed	De NXT deels in elkaar gezet	1 uur	School	Ramon en Yoran
24-09-2010	De NXT afmaken ging goed	De NXT afgemaakt.	1 uur	School	Ramon en Yoran
1-10-2010	Het verbinden van de NXT met de PC is nog niet gelukt.	Poging gedaan om de NXT met de PC te verbinden.	1 uur	School	Ramon en Yoran

1-10-2010	Ramon heeft de NXT nog niet kunnen laten verbinden	Poging gedaan tot verbinden via bluetooth	3 uur	Thuis	Ramon
4-10-2010	Ramon heeft de NXT eindelijk laten verbinden via USB	NXT verbonden met laptop maar programma's werken nog niet	2 uur	Thuis	Ramon
5-10-2010	Verbinden met PC gelukt en eerste programma werkt!	NXT mee naar huis genomen om mee te oefenen	3 uur	Thuis	Yoran
7-10-2010	We kunnen de NXT nu verbinden met een PC in de mediatheek dankzij de ICT.	We hebben een driver laten installeren en gewerkt aan een programma.	1 uur	School	Ramon en Yoran
8-10-2010	Het oefenen met de NXT ging goed. We hebben meer geleerd over de taal NXC.	Geoefend en een aantal programma's geschreven.	2.5 uur	School	Ramon en Yoran
13-11-2010	Verder gewerkt aan programma's voor de NXT. Vooruitgang geboekt.	Programma's geschreven.	5 uur	Thuis	Ramon en Yoran
13-11-2010	Vooruitgang geboekt	Deelvragen aangepast.	1 uur	Thuis	Ramon en Yoran
20-11-2010	Proefopstelling test ging goed	Proefopstelling gemaakt	8 uur	Thuis	Ramon en Yoran
27-11-2010	Er is een begin gemaakt aan het uiteindelijke werkstuk.	Deelvragen beantwoord	3 uur	Thuis	Ramon
27-11-2010	Vooruitgang	Deelvragen beantwoord	3 uur	Thuis	Yoran
28-11-2010	Vooruitgang	Deelvragen beantwoord	2 uur	Thuis	Yoran

28-11-2010	Vooruitgang	Deelvragen beantwoord en programma's afgemaakt.	5 uur	Thuis	Ramon
29-11-2010	Vooruitgang	Deelvragen beantwoord	3 uur	Thuis	Yoran
29-11-2010	Vooruitgang	Deelvragen beantwoord	2 uur	Thuis	Ramon
30-11-2010	Werkstuk goed voltooid	Het werkstuk afgerond	8 uur	Thuis	Ramon
30-11-2010	Werkstuk goed voltooid	Het werkstuk afgerond	8 uur	Thuis	Yoran
13-01-2011	Programma's aangepast	Getest met de robot	2,5 uur	Thuis	Ramon en Yoran
14-01-2011	Werkstuk afgemaakt	Het werkstuk aangepast	3 uur	Thuis	Ramon en Yoran