MJTL: PROJECT TURTLE

Door: Leo Jenneskens, Tim IJntema, Monisha Wielkens en Jip Galema.

De opdracht

Voor dit project was het de bedoeling om een zelfrijdende auto te maken. Dit zou gebeuren in de vorm van een lego mindstorm robot. Deze robot 'moest' een aantal dingen kunnen. Daarnaast was er een lijstje dingen die eigenlijk ook gedaan moesten worden. Als laatste was er een lijstje punten die je kon doen als er tijd over was. Onze robot (zie afbeelding hieronder) kan alles wat we moesten hebben, dit hield in dat het voertuig: Een lijn kan volgen, stopt als er een obstakel aanwezig is, tijdens het rijden geluid maakt, stopt op commando van de mobiele telefoon en weer rijdt op commando. Daarnaast hebben we ook een groot deel van de andere punten voor de casus voltooid. Zo kan onze robot bijvoorbeeld kruispunten herkennen en links, rechts of rechtdoor gaan op het kruispunt.
(voorbeeld van de structuur van task main staat in bijlage)

Voordelen en nadelen van ons ontwerp

De voordelen van ons ontwerp hebben vooral te maken met de lichtsensoren. We hebben de twee sensoren naast elkaar gezet. Hierdoor kunnen we heel goed kruispunten herkennen (als immers de twee sensoren allebei zwart geven dan staat de robot op een kruispunt). Daarnaast hebben we ook een kapje om de sensoren heen gebouwd, hierdoor komt er minder licht van buitenaf binnen. Een nadeel van ons ontwerp is het achterste draaiwieltje, deze wordt namelijk best onder druk gezet door het gewicht van de robot, het sturen lukt goed maar dit kan over tijd misschien voor problemen zorgen

Problemen waar we tegen aan liepen

Er waren bij dit project een aantal problemen waar we tegenaan liepen. Zo kwamen we erachter dat het lastig is om t-splitsingen te herkennen met de robot, gelukkig zitten er in het parkour geen t-splitsingen. Daarnaast was er een probleem met de licenties van RobotC. Hierdoor konden we een hele tijd in de derde week niet werken. Ook werkt de bluetooth app vrij slecht. Het verbindt maar heel kort en het blijft de hele tijd dingen sturen. Daarnaast was het lastig om robot codes te testen, omdat we allemaal robot codes wilden testen.

Hoe werkt het?

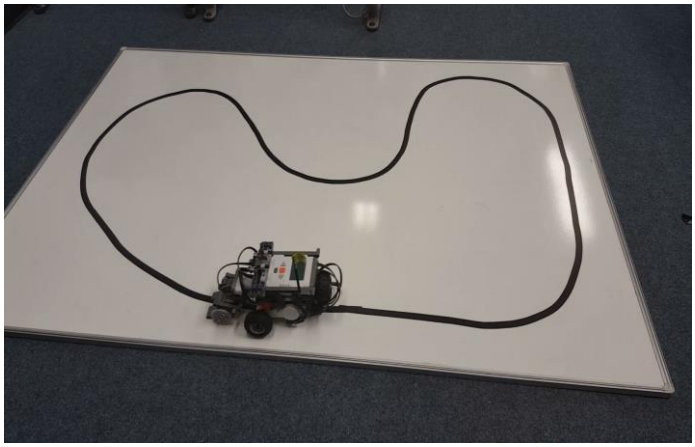
Lijn volgen

Eén van de hoofd opdrachten het zorgen dat de robot een lijn volgt. De robot moet over een lijn rijden en hierbij op de lijn blijven en niet van deze lijn afgaan. Hoe wordt op de lijn gebleven? Door aan elke kant van de lijn een sensor(*light sensor en color sensor die wordt gebruikt en functioneert als een light sensor*) te hebben die vele malen per seconde de licht waarde uit meet. Deze waardes die de sensoren meten, geven aan welk deel van het onderliggende vlak zwart is en dus hoever de robot over de lijn zit. Door de gemeten waardes als een variabele in een wiskundige formule te plaatsen. De formule bestaat uit een aantal verschillende variabele. De formule =

Speed+(speed / (maxlight-minlight*(maxlight-light)

(andere motor is speed-……)

de formule is gevormd door middel van de variabele van de standaard snelheid genaamd: **speed.** De maximale en de minimale waarde van de sensoren **maxlight / maxcolor** en **minlight / mincolor**. En de waarde die de sensor op het moment heeft ingelezen **color / light**. De formule bestaat uit drie delen eerst de standaardsnelheid. Het tweede deel is onderstreept en dit berekent hoeveel er per verandering moet worden bij geteld. Het laatste deel geeft aan hoeveel verandering stappen er moet worden gedaan. Samen vormt het de snelheid voor de wielen. Om soepel door de bochten te kunnen komen. Voor de meest scherpste bochten worden ondervangen door één motor op 0 te zetten en de andere motor op 2*standaardsensor. Dit samen vormt het lijn volg systeem.



Bluetooth

De onderdelen om bluetooth werkend te krijgen waren door de docenten geleverd. Dit hebben wij gekopieerd en gewijzigd zodat het zou werken in onze code. We hebben nu 2 bluetooth functies in een eigen header file De eerste bluetooth functie is een hele simpele functie die bij aanroep de bluetooth input uitleest. Wanneer deze is gegeven, stopt hij de input in een string die , door de pointer verwijzing die meegegeven is bij de aanroep, ook buiten de bluetooth functie beschikbaar is. Deze functie heet *check_bluetooth()*. De tweede is wat uitgebreider. Hij wordt aangeroepen vanuit *task main()* als de input van de bluetooth check die daar wordt gedaan gelijk is aan “B”. Dit is de knop voor een noodstop aangezien de motoren in een keer op 0 worden gezet. In de functie zelf, genaamd *bluetooth_control(),* wordt er opnieuw gewacht op een bluetooth input. Aan de hand van deze bluetooth input wordt er een bepaalde handeling uitgevoerd. Behalve rechts, links, enz., is er ook nog de optie om de code af te sluiten of verder te gaan met de normale code (de knoppen “C” en “A”).

Object detectie

Object detectie is een onderdeel dat in de follow line functie verwerkt zit. Een object dat minder dan 30 cm voor de robot staat, is een object waar de robot voor stopt. Dit doet hij langzaam afremmend om te zorgen dat inzittenden niet door elkaar worden geschud. Vervolgens draait de robot rustig om met een versnelling. Wanneer de robot de lijn weer ziet vertraagt hij met draaien en hierna rijdt hij verder de andere kant op. De sonar op de voorkant van de robot detecteert het object. Dit doet hij continu vanwege de oneindige while loop in de task main functie. Hierdoor weet de robot altijd wanneer hij moet stoppen.

Het geluid

We hebben een lijstje gevonden met verschillende muziek functies gevonden en konden met playSoundFile *();* de gewenste muziek afspelen. Het rso-bestand (muziek bestand omgezet uit een WAV-bestand) was alleen niet boven de motoren te horen. Daarom zijn we overgestapt op de functie playTone(); maar dat resulteerde in een hele lange lijst omdat elke toon individueel met eigen wait-time aangegeven moet worden. Uiteindelijk zijn we op het rmd bestand (omgezet midi-bestand) gestuit, die muziek als een lijst playTone functies afspeelt. De muziek werkt nu volgens onderstaande codes.

```
task music(){
        playSoundFile("supermario.rmd");
        wait1Msec(10000);
}
task main(){
        startTask(music);
        while(1){
                //motorcodes
        }
```

Bitmap grid.

Naast de hiervoor genoemde onderdelen van de autonome auto hebben we ook nog een stuk code die, aan de hand van getallen, over een grid van kruispunten kan rijden. Hij heeft als eerst de volgende standaard grid:

4 1 4 1 4
1 0 1 0 1
4 1 4 1 4
1 0 1 0 1
4 1 4 1 4

Als de grid uitgebreid word dan gaat dat op dezelfde manier. Bij het starten van de code zetten we een 2 en een 5 neer. De 5 markeert de plek waar we naartoe willen en de 2 is de startpositie. Die actie doen we handmatig om te simuleren dat je in een navigatiesysteem een beginpositie aan de hand van GPS verkrijgt en een bestemming invoert. Het programma loopt door deze array heen en zoekt de instructies uit die de robot uit moet gaan voeren en zet die in een variabele. Vervolgens gaat de robot al rijdend deze instructies op de grid uitvoeren. Als hij een object tegen komt rijd hij achteruit terug naar het vorige kruispunt en dan net iets verder zodat hij op een 1 uitkomt in de grid. Deze 1 word naar 2 verandert om de beginplek te wijzigen en het stukje lijn waar het object stond word een 0 omdat hij daar niet langs kan. Daarna word de route opnieuw berekent en kan de weg weer vervolgd worden.

De hardware(LEGO)

Voor de hardware van de robot is lego mindstorm gebruikt dit bestaat uit verschillende onderdelen zoals: de hoofd unit(brein), motoren, sensoren, constructie onderdelen(lego staafjes en dergelijke) en wielen. voor de eerste versie van onze robot hadden we een constructie bedacht die wordt aan gedreven met twee motoren en een bal die aan de achter kant. Deze eerste versie was bedacht met het idee van dat de bal makkelijk meer rolt in alle richtingen, echter dit bleek niet te werken aangezien de bal te vaak vast bleef zitten en hierdoor de robot afremde en verkeerd let rijden. Na dat de eerste versie niet volledig bleek te werken hadden we een tweede plan om drie motoren te gebruiken en hierbij dus twee motoren voor de aandrijving en 1 voor de sturing alleen door te weinig constructie hardware kon het tweede plan niet goed worden uitgevoerd.na het falen van het tweede plan kwamen we met het derde plan. Het derde plan was het gebruiken van twee motoren en het gebruik een zwenk wieltje aan de achterkant. Deze versie is later op door gewerkt tot het eind resultaat. We hebben later nog een schild toegevoegd bij de light en color sensor om te zorgen dat er minimale beïnvloedingen van buiten af zijn. Het schild heeft later gebleken functioneel te zijn. De robot zoals die er uit ziet is te zien op de afbeelding op de eerste pagina links onder. Het werken met lego mindstorm was hardware matig gezien goed te doen omdat het makkelijk in elkaar en uit elkaar te halen is en hierdoor zijn aanpassingen makkelijk te maken aan het ontwerp. Echter doordat bepaalde onderdelen van de hardware niet goed functioneerde zoals de wielen die niet goed draaide en een motor die niet goed functioneerde was het lastig om in het begin er goed mee over weg te gaan.

Slot

Al deze onderdelen maken samen een semi autonoom voertuig. De sonar zorgt dat er geen botsingen komen. Het detecteren van kruisingen zorgt ervoor dat je links of rechts kan. Het ingrijpen via bluetooth zorgt ervoor dat mogelijke problemen minder snel gevaar opleveren en het volgen van de lijn simuleert het volgen van een door het navigatie systeem berekende route.