1 Programmeren

Implementatie software

Als microcontroller hebben wij voor de MyRio gekozen, omdat het ons eenerzijds handiger leek om direct alles in LabVIEUW te programmeren dan enkel de interface in LabVIEUW te maken. Anderzijds ook omdat we de analoge afstandssensor liever wilden gebruiken dan de digitale (ze hebben een andere range).

Algoritmen

Voordat we begonnen met programmeren bedachten we eerste welke functies de auto zou moeten kunnen uitvoeren.
We kwamen tot volgende functies:

- De auto moet een zwarte lijn kunnen volgen en zich corrigeren als de auto wat afwijkt van de baan.
- Terwijl de auto de lijn volgt moet hij ook voorliggers kunnen detecteren en vertragen wanneer de afstand te klein wordt.
- De auto moet kunnen stoppen aan een stoplijn en daarvoor moet hij een gematigde snelheid hebben.
- Wanneer de auto stopt voor een kruispunt moet hij het verkeerslicht herkennen en hier gepast op reageren.
- De auto moet kunnen inslaan op een kruispunt en bij een bocht.
- De auto moet op ieder moment extern bestuurd kunnen worden.

1.1 Lijnvolgalgoritme

Op de baan die de robot moet volgen zijn allemaal donkere lijnen getrokken. Bij het volgen van deze lijnen maken we gebruik van de "QTR-8A analoge reflectie sensor array". De aansluiting hiervan op de microcontroller kan je zien in(ref1). De sensor heeft 8 lichtgevoelige sensoren. We hebben ons lijnvolgalgoritme gebaseerd op het feit dat men aan de ene kant van de sensor meer zwart ziet dan aan de andere wanneer die niet evenwijdig met de lijn rijdt. Wanneer men meer zwart ziet aan de linkerkant van de sensor, moet men naar links draaien (dit door de linkermotor trager te doen draaien). Wanneer men echter meer zwart ziet aan de rechterkant van de sensor, moet men naar rechts draaien (dit door de rechtermotor trager te doen draaien). Als men aan beide zijden even veel zwart detecteert moet men rechtdoor blijven rijden. Dit alles kan men ook zien in (ref2). Dit algortime komt tot een einde wanneer beide kanten even veel zwart ervaren en deze waardes tesamen veel groter zijn. Dit betekent dat er een stopstreep is, en men dus moet stoppen.

1.2 Licht herkennen

Dit algoritme treedt pas in werking als men een stopstreep detecteert. Dit is zo wanneer de 8 sensoren van de reflectiesensor zwart detecteren (zoals in vorige paragraaf beschreven). Met behulp van een while-loop gaan we kijken of het licht groen is dat de kleurensensor

detecteert. We gaan gebruik maken van een polling rate van 1 seconde zodat we de myrio niet overbelasten. Zolang het rood is blijft de while-loop lopen, vanaf de kleurensensor groen ziet, stopt die while-loop en kan het traject verdergezet worden.

1.3 Andere wagen herkennen en versnellen/vertragen

Met de afstandssensor kunnen we de afstand tot een voorliggende wagen detecteren. Wanneer de afstand te klein is gaat de wagen vertragen. Om dit te implementeren maken we gebruik van een while-loop. Als echter bij het vertragen de kritieke minimale snelheid overschreden wordt(dus nog minder dan de minimale), dan stopt onze wagen(dit is wanneer een andere wagen voor ons stilstaan). Versnellen gebeurt ook met een while loop, wanneer de snelheid boven een bepaalde waarde is, neemt hij de 'ideale snelheid' aan.

1.4 Draaien op kruispunt

We maken een ondersheid tussen 3 gevallen; linksaf, rechtsaf en rechtdoor. Bij linksaf moeten we eerst 375mm rechtdoor rijden, dan 90graden naar links draaien om de as van de wagen en tenslotte meer dan 375 mm rechtdoorrijden om dan weer de lijn te volgen. Bij rechtsaf moet men eerst 125mm rechtdoorrijden, dan 90graden naar rechts draaien om de as van de wagen en tenslotte meer dan 125mm rechtdoorrijden om dan weer de lijn te volgen. Bij rechtdoor moet men meer dan 500mm rechtdoorrijden om dan vervolgens de lijn te volgen. bij deze drie gevallen moet men tijdens het uitvoeren ervan rekening houden of er al dan niet een voorligger zich binnen de zeer kritische gevarenzone bevindt (dit wil zeggen dat de ander wagen voor de een of andere reden stilstaan op het kruispunt)

1.5 Manual override

Om de controle manueel over te nemen sturen we de motoren rechtstreeks aan. Dit stellen we voor met behulp van twee sliders in LabVIEUW. Later zullen we aan die sliders concreet een invulling voor geven om op de test te gebruiken.

1.6 Implementatie traject

Vanaf het moment dat we ons traject kennen, kunnen we aan de hand van de vorige algoritmes ons parkour samenstellen.

1.7 Concrete implementatie

Het hele LabVIEUW programma, gebruikmakend van de voorgenoemde funcites in de vorm van subVI's, splitsen we op in twee stappen:

De eerste stap is het rijden. Hier moet de auto tegelijk de lijn op de grond volgen, een goede afstand tot zijn voorligger behouden (wat dus inhoudt dat die de voorligger moet detecteren en aan de hand van de afstand tot die voorligger zijn snelheid moet aanpassen) en de auto moet ook nog eens comfortabel kunnen stoppen wanneer die een stopstreep detecteert. Al deze inputs komen van de sensoren aan boord van de auto.

De tweede stap hier begint bij die stopstreep. Wanneer de auto gestopt is aan de stopstreep moet die een verkeerslicht detecteren en daar gepast op reageren, nadat het groen licht geworden is moet die dus over het krijspunt draaien. Daarna moet die een bocht nemen

of rechtdoor rijden aan de hand van een vooraf opgesteld parcours. Na deze stappen is het algoritme klaar en begint hij weer met stap 1.

Op het moment van dit verslag zijn we zo goed als klaar met het schrijven van LabVIEUW-functies om deze taken te vervullen en moet slechts geïmplementeerd worden hoe we de inputs van de sensoren bij de labview-programmas kijgen en daarna ook de outputs van onze programma's bij de motoren krijgen.