KU LEUVEN kulak

Wetenschap & Technologie Kulak

Bachelor Ingenieurswetenschappen
Probleemoplossen en ontwerpen, deel 2
Martijn Boussé, Benjamin Maveau en Kevin Truyaert

Eindverslag Smart City: zelfrijdende robotauto

Team 8

Robbe Decapmaker, Emile Dejans, Simon Desimpelaere, Hannes Ingelaere, Lander Van Hevel en Emily Van Schependom

KU Leuven Kulak, Wetenschap & Technologie

Academiejaar 2020 – 2021

Inhoudsopgave

Inleiding

Het belangrijkste aspect van het ingenieurswezen is het kunnen *oplossen* van moderne *problemen* en het *ontwerpen* van gepaste oplossingen. Het vak *Probleemoplossen en ontwerpen* leert ons, bachelorstudenten ingenieurswetenschappen, de kennis uit andere vakken te bundelen en toe te passen op ingenieursproblemen. In het vak *P&O* werkten we dit semester aan een zelfrijdende robotwagen.

Ingenieurs en wetenschappers zijn al sinds het begin van de 20ste eeuw bezig met het ontwikkelen van systemen die de mens moet helpen bij de besturing van voertuigen. In 1912 ontwikkelde Sperry Corporation een autopiloot die in vliegtuigen werd gebruikt om de zware last van lange vluchten op piloten te verminderen. Autopiloot wordt tot op vandaag nog steeds verder ontwikkeld om piloten meer vrijheid te geven tijdens lange vluchten, maar werkt nog steeds niet volledig autonoom. Bedrijven zoals Google en Tesla zagen het succes en de bruikbaarheid van de autopiloot en proberen sinds het begin van de 21ste eeuw een volledig autonome tegenhanger binnen de auto-industrie te ontwikkelen.

In het dagelijks leven moet een zelfrijdende auto in staat zijn om onder andere verkeersborden correct te interpreteren en te stoppen voor rode lichten en andere weggebruikers. Voor dit vak is de opdracht gereduceerd tot het detecteren van en reageren op rode lichten, kruispunten, een lijn volgen en rekening houden met andere robotwagentjes op de baan.

Net zoals in de realiteit worden er financiële restricties opgelegd, er wordt een budget vastgelegd dat niet mag overschreden worden. Bepaalde onderdelen zijn niet voor alle groepen beschikbaar, waardoor het belangrijk is om bij de eersten te zijn die mogen aankopen doen. Deze volgorde wordt bepaald door een bieding. Het is dus belangrijk om op voorhand te berekenen welk budget er maximaal kan geboden worden.

Hoofdstuk 1

Klantenvereisten

Om te mogen rijden door de modelstad moet de robotwagen aan drie basisvoorwaarden voldoen. Om daarbovenop een geslaagde test te voltooien moet de robotwagen nog bijkomende eisen naleven. Hieronder worden de voorwaarden opgelijst.

1.1 Basisvoorwaarden

Manual override

De robotwagen moet over een werkende manual override beschikken, dit wil zeggen dat hij volledig bestuurbaar moet zijn door een groepslid via zijn/haar laptop. Hierbij voert de robotwagen zelf geen beslissingen uit.

Lijn volgen

Op de grond wordt een lijn getrokken met plakband die moet kunnen waargenomen en volledig autonoom gevolgd worden. De lijn is niet noodzakelijk recht, er kunnen ook bochten in zitten.

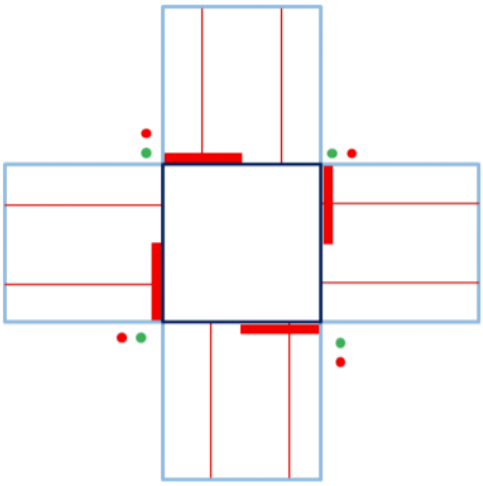
Stoppen bij stopstreep

Ter hoogte van elk kruispunt wordt loodrecht op de volglijn een stoplijn getrokken met plakband (dikke rode lijn, zie figuur 1.1). Wanneer de robotwagen aankomt bij een stopstreep moet hij automatisch stoppen.

1.2 Bijkomende eisen

Voorliggers detecteren & snelheid

Obstakels op de baan moeten gedetecteerd kunnen worden om aanrijdingen te vermijden. De snelheid moet worden aangepast aan die van eventuele voorliggers. Om zelf geen obstakel te zijn tegenover de andere robotwagens moet een voldoende hoge snelheid worden aangehouden.



Figuur 1.1: Tekening van het kruispunt dat op het parcours komt [1]

Verkeerslicht interpreteren

De robotauto mag niet zomaar afslaan aan een kruispunt. Er zijn verkeerslichten aangebracht (zie Figuur 1.1) en er moet gewacht worden tot het verkeerslicht op groen staat.

Kruispunten oversteken

Het robotautootje moet in staat zijn om te navigeren over een kruispunt. Elk kruispunt heeft dezelfde opbouw die bestaat uit vier straten die samenkomen (zie Figuur 1.1). De robotauto moet dus in staat zijn bochten te nemen van 90° en rechtdoor te rijden, ook wanneer er geen lijn is om te volgen.

Afwerking robotautootje

Een breadboard mag enkel gebruikt worden bij het prototype van het wagentje, bij de finale test moet alles gesoldeerd zijn op een printplaat en mag er geen breadboard meer gebruikt worden.

Individuele touch

Uit vijf opgegeven extra voorwaarden [1] kozen we ervoor om geen makerbeams te gebruiken bij het ontwerp en de constructie.

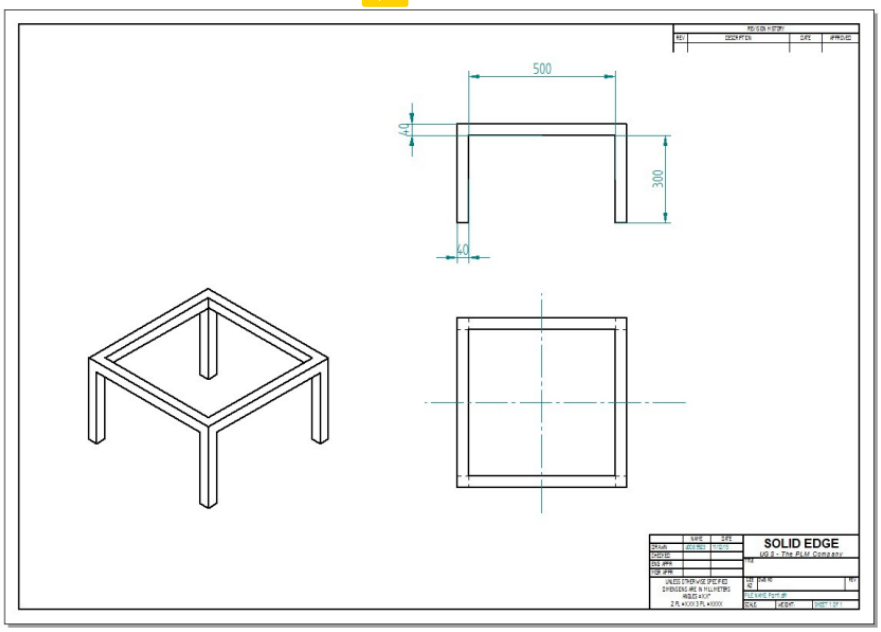
Hoofdstuk 2

Ontwerpspecificaties

Praktisch gezien moet er met bepaalde zaken rekening gehouden worden om het robotwagentje in de modelstad te kunnen laten rijden. Straten zijn niet oneindig breed, kruispunten hebben een maximumhoogte en sensoren moeten op de correcte plaats bevestigd worden om optimaal de gewenste signalen op te kunnen vangen.

2.1 Specificaties van een kruispunt

Figuur 2.1 toont een technische tekening van een kruispunt. Een kruispunt bestaat uit vier poorten. Een poort heeft een breedte van ~~500mm~~ en hoogte van ~~300mm~~. Aangezien er tweerichtingsverkeer is wilt dit zeggen dat één baan een breedte van ~~250mm~~ heeft.



Figuur 2.1: Technische tekening van een kruispunt [1]

Zoals te zien op Figuur 1.1 zijn er geen volglijnen aangebracht op de kruispunten, dit wil zeggen dat er algoritmes moeten geschreven worden die ervoor zorgen dat het robotwagentje bochten van 90° kan maken zonder gebruik te maken van een lijn.

Aan elk kruispunt zijn er voor de aankomende robotwagens aan de rechterkant verkeerslichten bevestigd. Deze verkeerslichten zijn opgehangen ~~7,5cm~~ boven het wegdek en knipperen aan 1 Hertz.

2.2 Dimensies van de lijnen

De volglijnen tussen kruispunten zijn 1m lang en hebben een breedte van ~~25mm.~~ De stoplijnen loodrecht op de volglijnen ter hoogte van de kruispunten hebben een breedte van ~~50mm~~. De volg- en stoplijnen zijn donker gekleurd ten opzichte van de ondergrond.

2.3 Dimensies van de wagen

2.3.1 Hoogte

De poorten die op de kruispunten bevestigd zijn hebben een hoogte van ~~300mm~~ en een breedte van ~~500mm~~. Met een marge van 10% is de maximumhoogte van de robotauto dus ~~270mm.~~

2.3.2 Breedte

Een rijvak is ~~250mm~~ breed, hier nemen we een iets grotere marge van 20% omdat de robotauto moet kunnen oscilleren rond de middellijn door de manier waarop hij de volglijn volgt. De maximumbreedte van de robot is dus ~~200mm~~.

2.3.3 Lengte

De robotauto mag niet te lang zijn omdat hij anders voor hinder op de kruispunten en de baan kan zorgen, daarom nemen we een maximale lengte van ~~350mm.~~ Dit geeft ons genoeg vrijheid in ons ontwerp zonder te moeten zorgen maken over het hinderen van andere robotwagens.

2.4 Hoogte van de sensoren

2.4.1 Kleursensor

Aangezien de verkeerslichten ~~75mm~~ boven het wegdek bevestigd zijn, zorgen we dat het midden van de kleursensor ook ~~75mm~~ boven het wegdek bevestigd is aan onze robotwagen zodat het licht van de verkeerslichten optimaal op de kleursensor invalt.

2.4.2 Afstandssensor

De afstandssensor mag niet te hoog hangen aangezien te lage obstakels dan niet gedetecteerd zullen worden. We schatten dat de andere robotwagens ~~50-250mm~~ hoog zullen zijn. We zorgen dat de afstandssensor tussen ~~10mm en 50mm~~ hoog bevestigd wordt zodat er zeker geen obstakels gemist worden.

2.4.3 Reflectiesensor

De reflectiesensor moet zo dicht mogelijk bij de grond hangen om optimale lezing van de data te verzekeren. Daarom leggen we de maximumhoogte van de reflectiesensor ten opzichte van de grond vast op ~~15mm~~.

Hoofdstuk 3

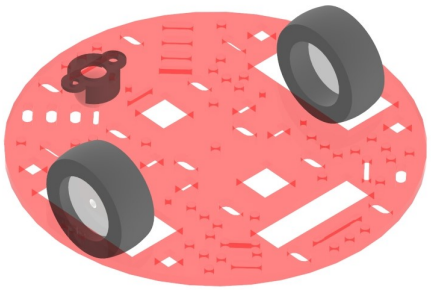
Hardware

In deze sectie wordt de opbouw van de robotwagen stap voor stap uitgelegd en wordt er bij elk onderdeel verklaard waarom het werd gekozen.

3.1 Chassis & wielen

Als chassis kozen we het *Pololu 5"Robot Chassis RRC04A*. Dit was de duurste optie maar gaf ons de vrijheid om met twee wielen en een ballcaster te werken. Dit zorgt ervoor dat we maar twee motoren moeten aankopen en besturen in tegenstelling tot vier bij een groter chassis. Het is een klein chassis wat de mobiliteit van de robotwagen bevordert. Het chassis heeft een diameter van 127mm wat goed binnen de maximale lengte en breedte ligt.

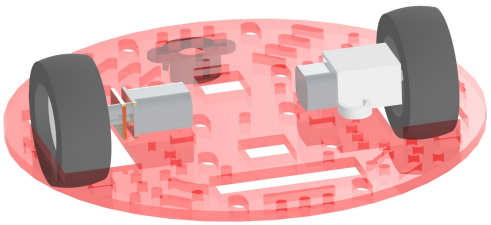
De *Pololu 42x19 wielen* zijn net als het chassis gemaakt door *Pololu*. We kozen deze wielen omdat ze perfect in het chassis passen zonder extra modificaties, zoals te zien op Figuur 3.1. Aangezien de robotwagen niet stabiel staat op twee wielen hebben we nog een derde ondersteuning nodig. We kozen voor een ballcaster aangezien die geen motor nodig heeft en, net zoals de wielen, een voorziene plaats heeft op het chassis, zoals te zien op Figuur 3.1 (onderkant achteraan).



Figuur 3.1: Render van het chassis met wielen en ballcaster

3.2 Motoren

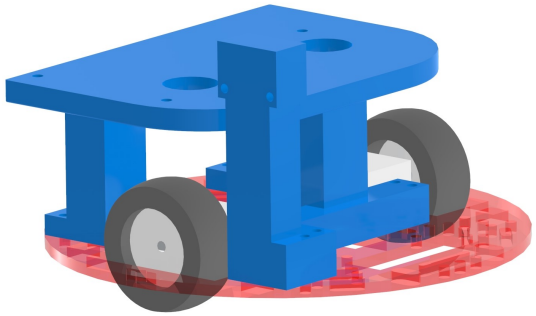
Voor de aandrijving van de robotwagen hebben we twee motoren nodig. De keuze bestond uit verschillende *Pololu Micro Metal Gearmotor* motoren die elk 6 Volt nodig hebben. De motoren verschillen onderling in hun maximumsnelheid en de torsie die ze kunnen uitoefenen. Wij kozen voor de *50:1 Micro Metal Gearmotor HP 6V* omdat een 50:1 gear-ratio voor aanvaardbare snelheden kan zorgen. Verder kozen we de 'High Power' variant zodat we wat meer vrijheid hebben voor het finale gewicht van de robotwagen. De motoren werden bevestigd aan het chassis met een motorbeugel zoals te zien is op 3.2, waar je één motor kan zien die niet bevestigd (links) is en één die wel bevestigd is (rechts).



Figuur 3.2: Render van het chassis met wielen, ballcaster en motoren

3.3 3D-prints

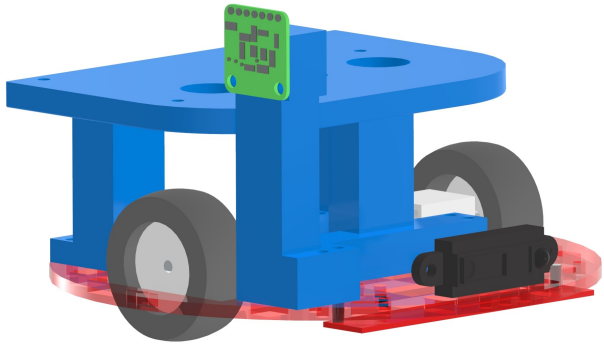
Aangezien we geen makerbeams gebruikten in ons ontwerp 1.2 hebben we gebruikgemaakt van zelf ge-3D-printe ondersteuningsonderdelen. De kleursensor moest op een hoogte van 75mm komen, daarvoor werd een onderdeel ontworpen waaraan de sensor bevestigd kon worden. Aangezien onze microcontroller te groot was om op het chassis te passen, werd een tafel ge-3D-print waarop hij kon bevestigd worden via drie gaatjes voor M3 schroeven. Op Figuur 3.3 zie je de kleursensor-ondersteuning vooraan en de tafel achteraan op het chassis.



Figuur 3.3: Render van de robotwagen met de ge-3D-printe onderdelen

3.4 Sensoren

Om aan alle klantenvoorwaarden te voldoen hebben we verschillende sensoren nodig. Om een lijn te volgen gebruiken we de *QTR-8A Reflectance Sensor Array*. We verkiezen de analoge variant boven de digitale aangezien we voor de analoge reflectiesensor al programma's om ~~de sensor uit te lezen ter beschikking hebben gekregen van de assistenten.~~ Om voorliggende wagens te detecteren gebruiken we de *Adafruit GP2Y0A21YK0F IR* afstandssensor. Ook hier kozen we voor de analoge variant omdat we gemakkelijker kunnen aflezen wat de afstand bedraagt aan de hand van voltages dan met binaire waarden. Om de verkeerslichten te kunnen inlezen gebruiken we de *Adafruit TCS34725 RGB* kleursensor aangezien data inlezen met deze kleursensor veel gemakkelijker is dan met een webcam. Op Figuur 3.4 zie je de kleursensor (groen), de reflectiesensor (rood) en de afstandssensor (zwart).



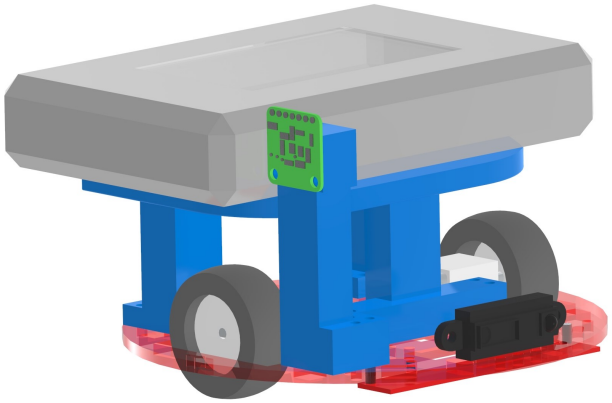
Figuur 3.4: Render van de robotwagen met de sensoren op de juiste plaats

3.5 Micro-controller

Als micro-controller gingen we voor de MyRio omdat die specifiek ontworpen is voor kleine projecten zoals onze zelfrijdende robotauto. In LabVIEW is een aparte database voorzien specifiek om de MyRio aan te sturen. We wisten dat de Raspberry Pi geen analoge sensoren kon ontvangen, dus wilden we onszelf ook de vrijheid geven te kunnen kiezen, zonder veel problemen te ondervinden. De MyRio kan namelijk analoge en digitale sensoren inlezen. Een derde reden was eenvoud, we wisten dat als we een Raspberry Pi gebruikten in Python gingen moeten werken en dit verbinden aan LabVIEW om daar een GUI te ontwerpen. Dit moedigde ons af om een Raspberry Pi te kopen aangezien we het gemakkelijker vonden in eenzelfde programmeertaal te werken. Op Figuur 3.5 is de MyRio bovenop de tafel bevestigd.

3.6 Batterijen & printplaat

Om stroom te leveren aan onze robotwagen gebruiken we batterijen aangezien een powerbank niet genoeg voltage kan leveren. We gebruiken een printplaat omdat in het finale ontwerp geen breadboard meer mag gebruikt worden 1.2. De



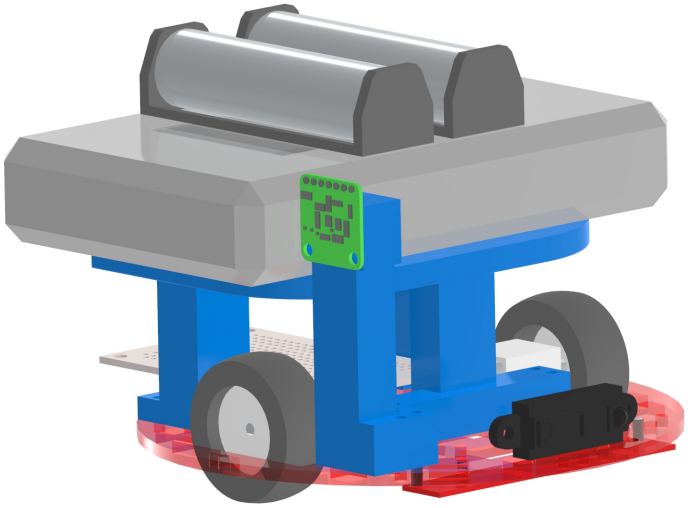
Figuur 3.5: Render van de robotwagen met de MyRio

batterijen worden bovenop de MyRio bevestigd en de printplaat eronder zoals te zien op Figuur 3.6.

Figuur 3.6 toont hoe onze robotwagen er zou uitzien zonder bekabeling, bouten, etc. Door lage kwaliteit van de 3D-prints werd ook nog gebruik gemaakt van kabelbinders.

3.7 Financieel rapport

Het virtueel budget bedroeg bij de start 3500 eenheden. Na een grondige planning van het ontwerp werd besloten dat er maximaal 1600 eenheden konden geboden worden en dat we bij de eerste vier moesten zijn om zeker te zijn dat we alle gewenste onderdelen konden aankopen. Uiteindelijk boden we 1500 eenheden wat goed was voor een eerste plaats. Na de eerste aankoop en de 3D-modellering hadden we nog 202 eenheden over. Dit moest instaan om alle niet voorziene kosten in te dekken.



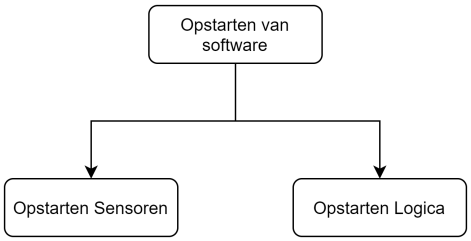
Figuur 3.6: Render van de robotwagen met alle onderdelen

Aankoop	**Onderdeel (aantal)**	**Prijs**	**Kost**
Bod			1.500
Aankoop 1	Breadbord tiny (1)	40	40
	Motor 50:1 (2)	160	320
	Motorshield dualdrive (1)	70	70
	Chassis rond (1)	140	140
	Wiel 42x19 (2)	35	70
	Ball Caster (1)	60	60
	Afstandssensor (1)	160	160
	Kleursensor (1)	150	150
	Motorbeugels (2)	25	50
	Reflectie arry (1)	150	150
	MyRio controller (1)	240	240
	Lithium-ionbatterij (2)	90	180
	Printplaat (1)	50	50
3D-modellering		118	118
Aankoop 2	Motorshield IC (1)	70	70
	Male headers x10 (1)	5	5
totaal			3.373

Hoofdstuk 4

Software

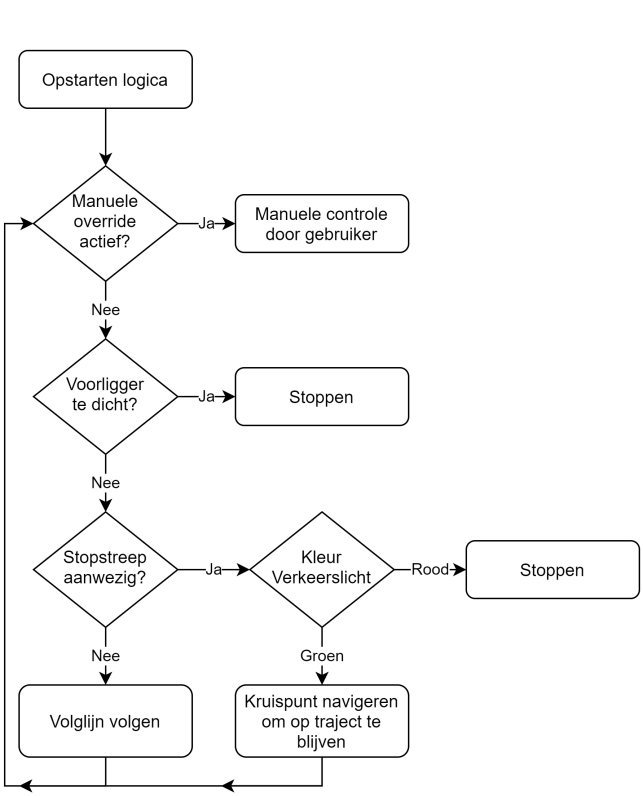
Geschreven programma's worden vanuit LabVIEW [2] opgeladen op de MyRio, zodat informatie kan worden verkregen van en verzonden naar de componenten van de robotwagen. De hoofdfunctie wordt constant doorlopen, zolang we het programma zelf niet manueel besturen. In dit hoofdstuk wordt een overzicht gegeven van alle deelfuncties die doorlopen worden en hoe op bepaalde signalen gereageerd wordt. Dit overzicht wordt schematisch weergegeven in de bijgevoegde stroomdiagrammen (Figuur 4.1, Figuur 4.2). Een gedetailleerd overzicht van alle geprogrammeerde functies is ook te vinden in bijlage.



Figuur 4.1: stroomdiagram dat de structuur van het opstarten weergeeft.

4.1 Data sensoren

- De afstandssensor controleert of een veilige afstand wordt bewaard tot voorliggende obstakels.
  - Zolang geen voorliggers gedetecteerd worden geeft de afstandssensor geen feedback.
  - Als een voorligger gedetecteerd wordt, geeft de afstandssensor feedback aan de hoofdfunctie, waarna die een commando stuurt naar de motoren om de snelheid te minderen.
  - Er wordt ook een veilige afstand meegegeven aan het programma. Vanaf dat het robotwagentje een obstakel nadert en dichter komt



Figuur 4.2: stroomdiagram dat de structuur van de code weergeeft. In deze volgorde worden beslissingen genomen.

dan deze vooraf ingestelde afstand, stopt het autootje met rijden om botsingen te voorkomen.

- De reflectiesensor controleert of de robotwagen nog op de zwarte plakbandlijn rijdt.
  - Als er voldoende gereflecteerd licht wordt ontvangen door de middelste sensoren betekent dit dat de robotwagen mooi de lijn volgt en hoeft er geen feedback verstuurd te worden.
  - Als er hogere waarden worden opgevangen door de sensoren aan de buitenkant, betekent dit dat het autootje aan het afwijken is. De sensor geeft dit door aan de hoofdfunctie en de kracht waarmee de motoren draaien zal veranderen zodat de robotwagen wordt bijgestuurd totdat de lijn terug door de middelste sensoren wordt opgevangen.
- De kleurensensor geeft, indien gevraagd, aan of het licht op rood of groen staat. Dit programma wordt enkel aangesproken als we op een stopstreep staan, want dat is het enige moment dat we geïnteresseerd zijn als het verkeerslicht op groen of op rood knippert.
  - Indien het licht op groen staat, wordt een signaal gegeven aan de motoren om terug te beginnen draaien.

– Indien het licht als rood wordt geïnterpreteerd, blijft de kleurensensor data opnemen totdat het kleursensorinterpretatiealgoritme aangeeft dat het licht groen knippert .

4.2 Belangrijkste programma's

4.2.1 Hoofdprogramma

Het hoofdprogramma is het belangrijkste programma en is eigenlijk het brein van de robot. Het ontvangt alle outputs van de subVI's en bepaalt afhankelijk daarvan welke andere programma's moeten worden aangestuurd. Dit programma bestuurt ook de ledjes van de Myrio waar we onze eigen kleurencode aan geven. Op die manier kunnen we volgen met welke programma's de MyRio bezig is te doorlopen.

4.2.2 Verkeerslichten interpreteren

Aangezien de groene ledjes in de verkeerslichten niet fel genoeg brandden, kwamen we tijdens het testen tot de conclusie dat we enkel de data die de kleursensor voor rood licht teruggaf konden gebruiken om het verkeerslicht te interpreteren. Concreet krijgt het programma om de 5 milliseconden een nieuwe waarde binnen van de kleursensor voor rood licht en plaatst die waarde in een 'array'. Deze 'array' bevat 700 meetwaarden voor rood licht waaruit vervolgens een golffunctie wordt opgesteld. Hier wordt dan een fouriertransformatie op toegepast waaruit dan de amplitudes horende bij de frequenties waarmee het rood licht in intensiteit veranderde kunnen gehaald worden. Aangezien we weten dat het verkeerslicht rood knippert aan 1 Hertz kunnen we hieruit een drempelwaarde halen voor de amplitude horende bij die frequentie van 1 Hertz. Ligt de amplitude onder de gekozen drempelwaarde gaan we ervan uit dat het licht groen knippert, ligt de amplitude boven de drempelwaarde gaan we ervan uit dat het licht rood knippert.

4.2.3 besturing van de motor

Dit programma krijgt als input 3 waarden: een snelheid, een richting en een boolean. De snelheid is uitgedrukt in een percentage, een richting is een getal tussen de -90 en 90 en de boolean geeft True als we vooruit willen rijden.

We besturen de motor door middel van PWM signalen, hiervoor is er is een verglijking opgesteld om uit de gegeven snelheid en richting te bepalen welke waarden we moeten geven aan de juiste 'duty cycle'. Verder is alles in een grote 'case structure' gestoken om ervoor te zorgen dat het robotwagentje zeker niet begint te rijden als we snelheid 0 meegeven.

De waarden worden naar de 'duty cycles' verstuurd, die rechtstreeks de motor aansturen. De 'duty cycle' verwacht een waarde tussen de 0 en 1 om te weten aan welke snelheid de motoren moeten draaien.

4.2.4 bochten maken

Het programma om bochten te kunnen maken werkt met een aantal vooraf ingestelde waarden, die gebundeld zitten in een 'cluster'. Door deze waarden konden

we de robotauto gedurende een precies getimed tijdsinterval een draaibeweging laten maken en zo onze robotauto bochten van precies 90° naar links of rechts laten maken.

4.2.5 lijnvolgprogramma

Het lijnvolgprogramma krijgt als input een array met voltwaarden en bestaat uit 3 onderdelen: een deel om de lijn te volgen, een deel om de stopstreep te herkennen en een deel dat beslissingen maakt wanneer we helemaal van de lijn zijn en dus geen lijn meer detecteren.

Het eerste onderdeel van het programma haalt het maximum uit de array en geeft die aan de 'PID', een VI die ernaar streeft om de maximumwaarde naar de middelste sensor te geven. Afhankelijk van van welke sensor het maximum komt, geeft het programma dus het aantal graden dat gedraaid moet worden om de lijn in het midden van de sensoren te krijgen en houden om zo mooi de lijn te kunnen blijven volgen.

Wanneer alle voltwaarden onder 1 gaan, betekent dit dat alle sensoren een lijn detecteren en we dus op een stopstreep staan. Dit programma geeft als output een boolean die op True wordt gezet als we een stopstreep detecteren.

Wanneer er geen enkele waarde meer onder de 1 zit, wil dit zeggen dat de sensoren geen lijn meer detecteren en wordt ook een boolean teruggegeven die de waarde True krijgt op dat moment.

Besluit

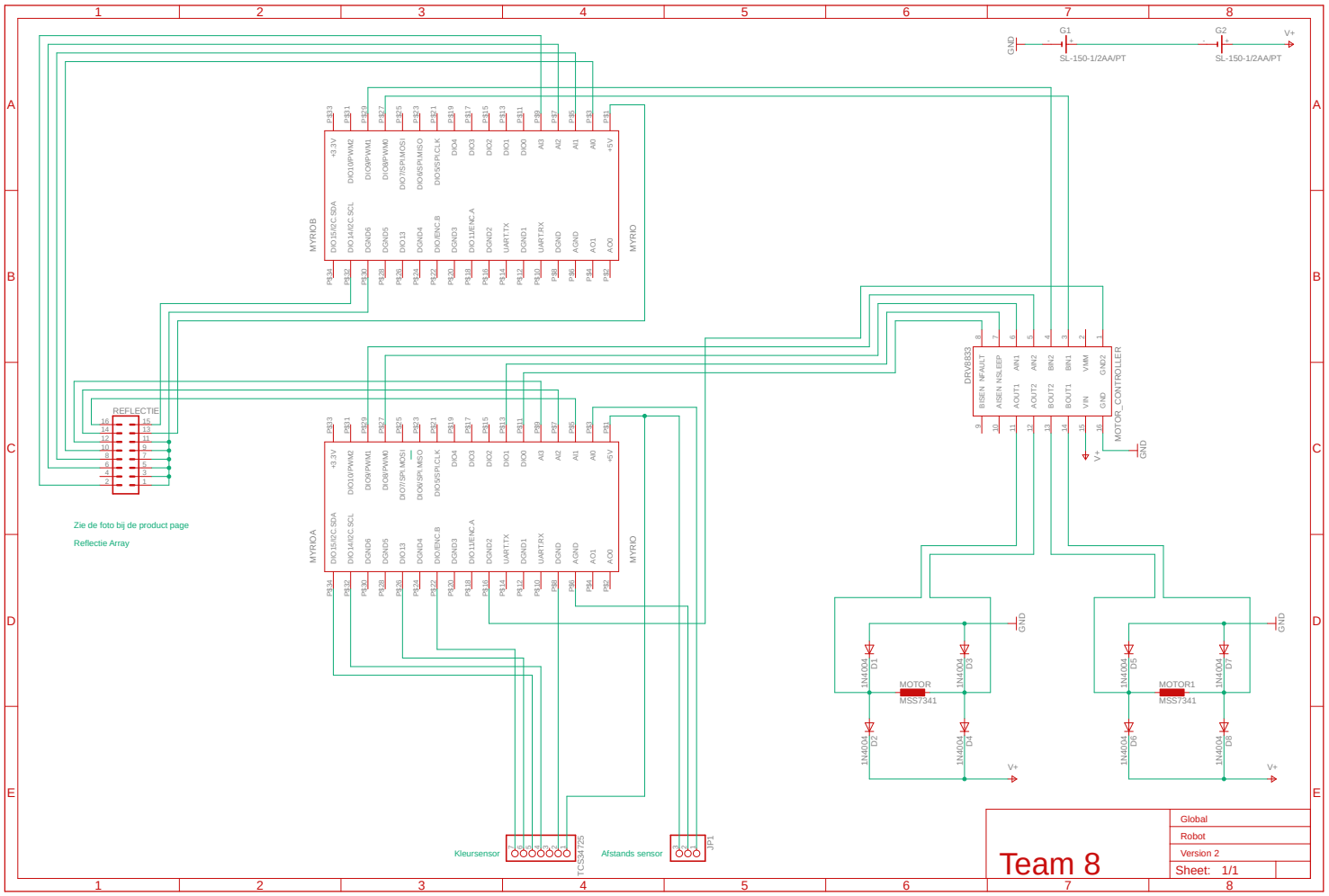
We leerden de grafische programmeerstructuur van Labview beter kennen door deze opdracht. Labview wordt vaak gebruikt bij de besturing van grotere machines in bedrijven. Het belangrijke grafische aspect van Labview leerde ons op een nieuwe manier softwareproblemen oplossen. Het is echter zeer jammer dat we veel tijd zijn verloren aangezien we lange tijd geen verbinding konden leggen met de MyRio, maar nadat dat was opgelost, hebben we ons verder kunnen focussen op het afwerken van de code. Doordat we zo veel tijd zijn verloren zijn we niet in staat geweest een poging te wagen tot functionaliteiten als adap[illegible] cruise control, wat de snelheid van onze robotauto regelt afhankelijk van de snelheid van voorliggende robotauto's.

Het bouwen, programmeren en testen van deze zelfrijdende auto was zeker een leerrijke ervaring. Hoewel onze opdracht klein was in vergelijking met echte zelfrijdende auto's, hebben we toch kunnen proeven van de opdracht die ingenieurs hebben om echte zelfrijdende wagens te ontwerpen.

Bibliografie

[1] K. Truyaert B. Maveau. Opgave teamopdracht probleemoplossen en ontwerpen 2, 2020-2021.

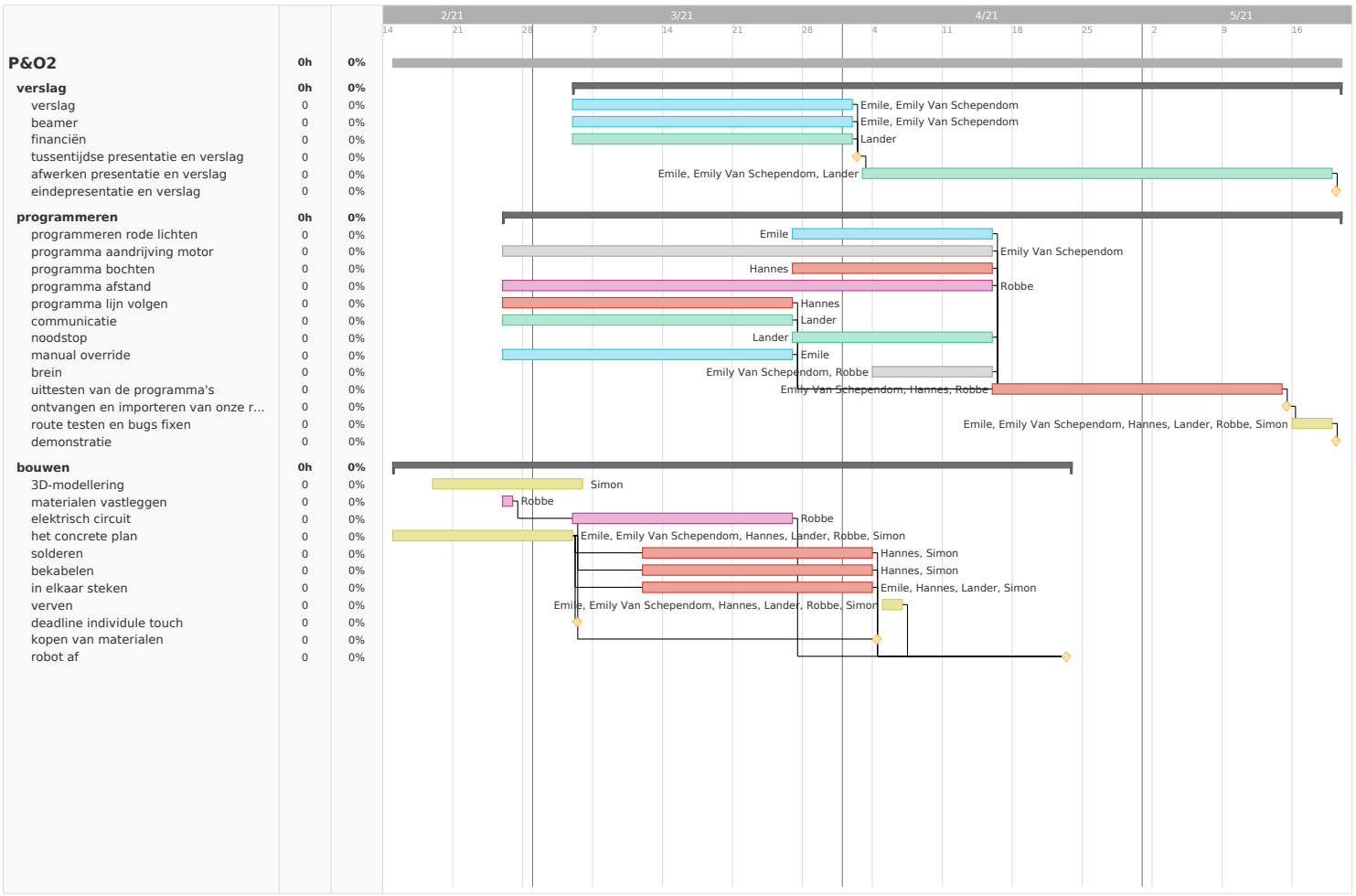
[2] E. Verboven K. Truyaert. Introduction to labview, 2020-2021.


GND
G1
SL-150-1/2AA/PT
G2
SL-150-1/2AA/PT
V+
MYRIO B
MYRIO A
MYRIO
REFLECTIE
Zie de foto bij de product page
Reflectie Array
DRV8833
MOTOR_CONTROLLER
1N4004
D1
D2
D3
D4
D5
D6
D7
D8
MOTOR
MOTOR1
MSS7341
Kleursensor
TCS34725
Afstands sensor
JP1
Team 8
Global
Robot
Version 2
Sheet: 1/1

Tabel 1: Taakstructuur voor het maken van het robotautootje.

Code	Taak	Status
1	Het robotautootje bouwen	OK
1.1	Ontwerpen	OK
1.1.1	Ontwerpplan	OK
1.1.2	3D-modellering	OK
1.1.3	Elektrisch circuit	OK
1.1.4	Benodigde componenten	OK
1.2	Constructie	OK
1.2.1	Componenten en frame	OK
1.2.2	Bekabelen componenten	OK
1.2.3	Solderen	OK
2	Programmeren	OK
2.1	Functies	OK
2.1.1	Main functie	OK
2.1.2	Verkeerslichtinterpretatie-algoritme	OK
2.1.3	Aandrijving motor	OK
2.1.4	Bochten	OK
2.1.5	Afstand	OK
2.1.6	Lijnvolgalgoritme	OK
2.1.7	Communicatie	OK
2.1.8	Noodstop	OK
2.1.9	Manual override	OK
2.2	Testen	OK
2.2.1	Functies testen	OK
2.2.2	Programmeren testroute	OK
3	Verslag	OK
3.1	Tussentijds verslag	OK
3.2	Tussentijdse Beamer-presentatie	OK
3.3	Schriftelijk verslag	OK
3.4	Financieel rapport	OK

teamgantt
Created with Free Edition



P&O2	**0h**	**0%**
verslag	**0h**	**0%**
verslag	0	0%
beamer	0	0%
financiën	0	0%
tussentijdse presentatie en verslag	0	0%
afwerken presentatie en verslag	0	0%
eindepresentatie en verslag	0	0%
programmeren	**0h**	**0%**
programmeren rode lichten	0	0%
programma aandrijving motor	0	0%
programma bochten	0	0%
programma afstand	0	0%
programma lijn volgen	0	0%
communicatie	0	0%
noodstop	0	0%
manual override	0	0%
brein	0	0%
uittesten van de programma's	0	0%
ontvangen en importeren van onze r...	0	0%
route testen en bugs fixen	0	0%
demonstratie	0	0%
bouwen	**0h**	**0%**
3D-modellering	0	0%
materialen vastleggen	0	0%
elektrisch circuit	0	0%
het concrete plan	0	0%
solderen	0	0%
bekabelen	0	0%
in elkaar steken	0	0%
verven	0	0%
deadline individule touch	0	0%
kopen van materialen	0	0%
robot af	0	0%

KU Leuven Kulak
Wetenschap & Technologie
Etienne Sabbelaan 53, 8500 Kortrijk
Tel. +32 56 24 60 20