KU LEUVEN kulak

Groep Wetenschappen & Technologie

X0B54a – Probleemoplossen en ontwerpen, deel 2

Ontwerpproces van een zelf-rijdende wagen

Tussentijds verslag

Team 4: Team R2D2

Matthijs Deforche, Karl Van Holder, Thomas Varheust, Kobe De Weerdt, Yaron Verhulst

Academiejaar 2020 – 2021

Abstract

Inhoud

Lijsten van tabellen en figuren

Figuren

Tabellen

Inleiding

Wij zijn vijf studenten ingenieurswetenschappen in het eerste jaar aan de Katholieke Universiteit Leuven Campus Kulak Kortrijk. Het vak P&O 1 en de andere vakken van het eerste semester hebben ons de nodige kennis gegeven om het vak P&O 2 aan te vatten. De opzet van de teamopdracht dit semester is een zelfrijdend wagentje te ontwerpen en te bouwen dat uiteindelijk zelfstandig door een modelstad zal moeten rijden. Het wagentje moet niet alleen kunnen rijden, ook moet het obstakels zoals tegenliggers en voorgangers kunnen detecteren. Het wagentje zal kruispunten en verkeerslichten correct moeten interpreteren en zal er gepast op reageren. Het ontwerpen en het bouwen van het wagentje kan met behulp van een virtueel budget, een interface LABVIEW om met het wagentje te kunnen communiceren, een CAD-programma om onze wagen te 3D-modelleren enz. Een extra toevoeging aan de opdracht is om onze wagen een persoonlijke touch te geven.

Hoofdstuk 1

Methodologie

1.1 Planning

Voor de planning van onze teamopdracht hebben we enkele tools gebruikt die in de volgende secties uitgelegd zullen worden.

Klantenvereisten

Eerst en vooral hebben we de klantenvereisten opgesteld. De opdracht die we kregen bevat alle informatie van wat er tegen het eind van het project van ons verwacht wordt. Om hier het overzicht in te bewaren, hebben we de klantenvereisten opgesteld. Per item hebben we een klantenvereiste opgesteld die een algemene beschrijving hiervan is. De klantenvereisten zijn terug te vinden in bijlage 4.3.

Ontwerpspecificaties

Aan de hand van deze klantenvereisten hebben we de ontwerpspecificaties samengesteld. Een klantenvereiste is eerder algemeen terwijl de ontwerpspecificaties veel gedetailleerder zijn. Door de ontwerpspecificaties op te stellen, hebben we nu een beter en duidelijker beeld van hoe een klantenvereiste ineen zit.De ontwerpspecificaties zijn terug te vinden in bijlage 4.2.

Teamverantwoordelijkheden

De teamopdracht vergt een goede planning en een goede werkverdeling. Daarom hebben we per categorie een tweetal studenten verantwoordelijk gesteld. Met een categorie bedoelen we bijvoorbeeld programmeren, Solid Edge enzovoverder. De teamverantwoordelijkheden zijn terug te vinden in bijlage 4.4.

Takenstructuur

Om het werk te verdelen onder elkaar hebben we een takenstructuur gemaakt. Deze bevat alle deeltaken die nodig zijn om ons project tot een goed einde te brengen. De deeltaken zijn verdeeld onder een samenvattende hoofdcategorie.

Bijvoorbeeld bij de categorie Solid Edge is er een deeltaak 'wiel modelleren'. De takenstructuur is terug te vinden in bijlage 4.5.

Gantt-grafiek & Teamkalender

Nu we de verschillende deeltaken hebben opgesteld is het mogelijk om deze in te plannen. We hebben de deeltaken genummerd en een teamkalender opgesteld om te plannen wanneer een deeltaak afgewerkt moet zijn. Het nummer van de deeltaak wordt bij de week/dag geplaatst wanneer het af moet zijn in de teamkalender. Een teamkalender vertelt echter niet hoeveel tijd een deeltaak in beslag zal nemen. Daarom hebben we een bijhorende Gantt-grafiek gemaakt. Deze toont per deeltaak hoeveel tijd we eraan zullen besteden. Wanneer een deeltaak pas kan aangevat worden als een andere deeltaak is afgewerkt, zullen we dit in de Gantt-grafiek aanduiden met een pijl. De Gantt-grafiek en Teamkalender zijn terug te vinden in bijlage 4.6.

1.2 Ontwerp

1.2.1 Onderdelen

We kozen voor de NI Myrio-microcontroller omdat deze het simpelste is en voor de aard van de opdracht voor ons het efficiëntste leek. Bovendien hebben we via de NI Myrio altijd een handige visualisatie van het programma en kunnen we gebruik maken van de analoge afstandssensor die met een bereik van 10 to 80 centimeter de meest geschikte afstandssensor is. Door deze keuze kunnen we voor de andere sensoren enkel de analoge versie gebruiken. Aangezien de NI Myrio een power input nodig heeft van 6-16 Volt, kan de powerbank niet gebruikt worden. Door 2 Lithium-ion batterijen van $3,6$ Volt te kiezen en deze in serie te schakelen is het wel mogelijk om een geschikte power-input te verkrijgen. Als persoonlijke touch kozen we om geen makerbeams te gebruiken. Hierdoor leek het ons het beste om een groot chassis te hebben. We kozen het rechthoekig zwart boven het universeel chassis omdat ze redelijk gelijkaardig zijn, maar het universeel chassis 50 eenheden meer kost. Om dit chassis stabiel te maken kozen we ervoor om de ball caster te gebruiken samen met 2 aangedreven wielen. Als wiel kozen we voor de wielen met een breedte van 8 millimeter en diameter van 60 millimeter, deze zijn de grootste en kunnen samen met de spacers van de ball caster ervoor zorgen dat het chassis perfect horizontaal ligt. Als aandrijving hadden we de voorkeur voor de "Micro Metal Gear Motor 50:1 HP". Deze heeft een 3 millimeter as, wat compatibel is met onze gekozen wielen en heeft een gemiddeld vermogen om de wagen te doen stoppen en opnieuw te versnellen. Omdat deze motoren niet meer beschikbaar waren wanneer wij onze bestelling konden plaatsen kozen we voor de "Micro Metal Gear Motor 100:1 HP", deze is redelijk gelijkaardig maar heeft een ander vermogen. Samen met deze motoren kozen we 2 motorbeugels om de motoren aan het chassis te kunnen bevestigen en de "Dual Drive DRV8833"zodat we beide motoren tegelijk kunnen aansturen. Om eerst de wagen te testen voor we gaan solderen op de printplaat kozen we nog een breadbord, aangezien we maar 3 sensoren en 2 motoren hadden kozen we voor het kleinere formaat omdat dit ons minder budget kost.

1.2.2 Assemblage

Om de poorten van de myRIO-microcontroller vrij te houden, hebben we de myRIO boven de wielen geplaatst. Omdat de myRIO relatief zwaar is ten opzichte van de andere onderdelen, zorgt deze keuze ervoor dat het zwaartepunt van de wagen hoger komt te liggen. Om te voorkomen dat het zwaartepunt te hoog zou liggen, plaatsten we de batterijen centraal op het chassis. De reflectie sensor wordt vooraan in het verlengde van het chassis bevestigd via de "reflectie sensor houder", een onderdeel dat we zelf ge-3D-print hebben omdat we ervoor gekozen hebben om geen makerbeams te gebruiken. De technische tekening van de *reflectie sensor houder* is terug te vinden in bijlage 4.7. Omdat de afstandssensor maar objecten waarneemt vanaf 80mm hebben we besloten deze op 35mm van de voorkant van het chassis te plaatsen, zo kan deze efficiënter gebruikt worden. De kleurensensor wordt op 75mm boven de grond, 25mm horizontale afstand van de reflectie sensor en op 65mm van de symmetrieas van het chassis omdat het zich zo op de ideale positie bevindt om het verkeerslicht uit te lezen wanneer de auto stilstaat. De afstandssensor en kleurensensor worden geïntegreerd in de wagen via het *Skelet van de wagen*, dit is een ge-3D-print onderdeel die ook de myRIO ondersteunt en waarvan een afbeelding te zien is in figuur 4.4 in bijlage 4.7. De sensoren maken verbinding met de myRIO via de printplaat, deze is via plakband bevestigd op de myRIO zodat er voldoende ruimte is om overzichtelijk alles te bevestigen. Het elektrisch circuit is te zien in bijlage 4.8. Het 3D-model van de wagen is te zien in figuur 4.3 in bijlage 4.7.

1.3 Implementatie

1.3.1 Implementatie software

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

1.3.2 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.

1.3.3 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 bijlage 4.8. 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 1.1. 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.

1.3.4 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.

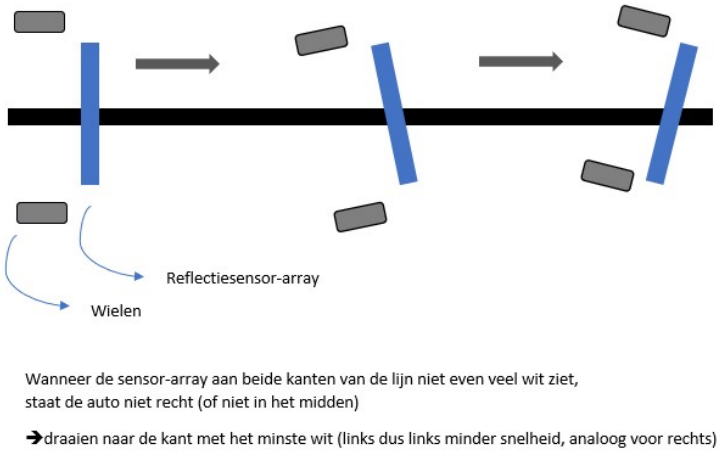
1.3.5 Andere wagen herkennen en versnellen/vertragen

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.

1.3.6 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



Figuur 1.1: Visualisatie van het principe van ons lijnvolgalgoritme.

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)

1.3.7 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 LabVIEW. Later zullen we aan die sliders concreet een invulling voor geven om op de test te gebruiken.

1.3.8 Implementatie traject

Vanaf het moment dat we ons traject kennen, kunnen we aan de hand van de vorige algoritmes ons parcours samenstellen. De flowchart van ons programma is terug te vinden in bijlage 4.9.

1.3.9 Concrete implementatie

Het hele LabVIEW programma, gebruikmakend van de voorgenoemde functies 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 kruispunt 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 LabVIEW-functies om deze taken te vervullen en moet slechts geïmplementeerd worden hoe we de inputs van de sensoren bij de LabVIEW-programma's kijgen en daarna ook de outputs van onze programma's bij de motoren krijgen.

Hoofdstuk 2

Resultaten

2.1 Budget

Een financieel verslag is terug te vinden in bijlage 4.10.

Hoofdstuk 3

Discussie

Hoofdstuk 4

Bijlagen

4.1 Planning

4.2 Overzicht ontwerpspecificaties

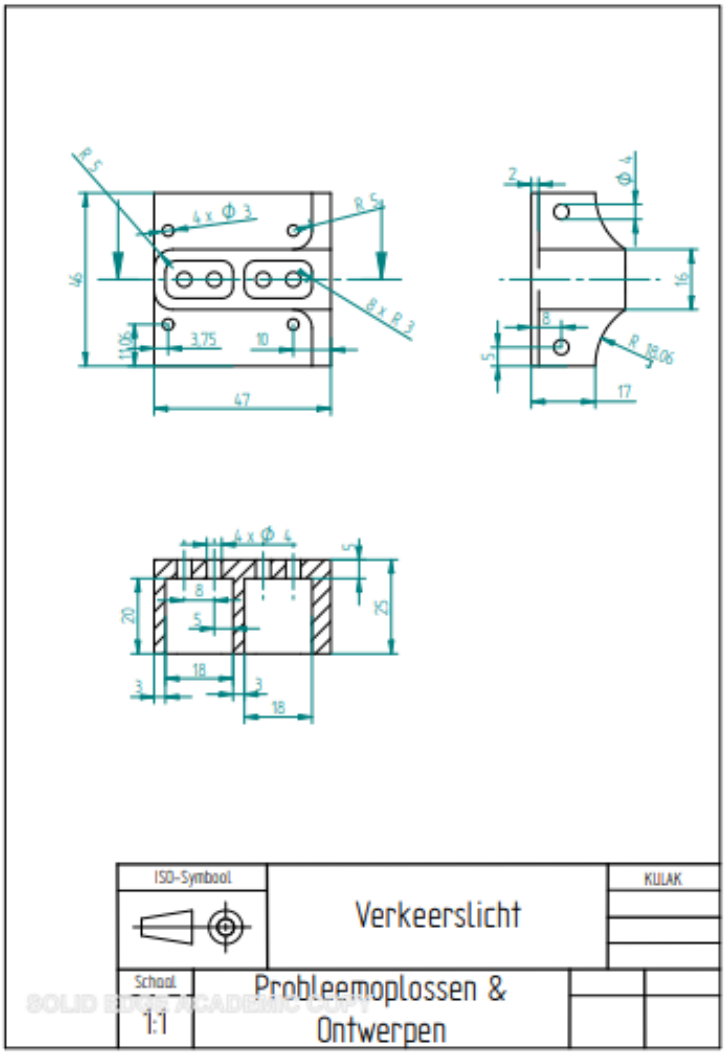
De verkeerslichten kunnen interpreteren: stoppen bij een rood licht, doorrijden bij een groen licht:
De hoogte van het verkeerslicht gemeten vanaf de grond tot aan het middelpunt van het verkeerslicht is $75mm$. Het verkeerslicht heeft een knipperfrequentie van $1Hz$. Een technische tekening van het kruispunt is te zien op 4.1.

Een geïmplementeerd traject kunnen volgen / stoplijn detecteren:
De ondergrond van het traject zal ofwel donker ofwel helder zijn. De kleur van de lijnen die de auto moet volgen zal afhangen van wat de kleur van de ondergrond is. Het zal het omgekeerde zijn waardoor het verschil duidelijk is. Er zijn ook nog verschillen in de soorten lijnen die op de ondergrond aangebracht zullen worden. De breedte van een volglijn is $25mm$ en de breedte van een stoplijn is $50mm$. De kruispunten liggen op $1000mm$ van elkaar en figuur 4.2 toont het bovenaanzicht van een kruispunt.

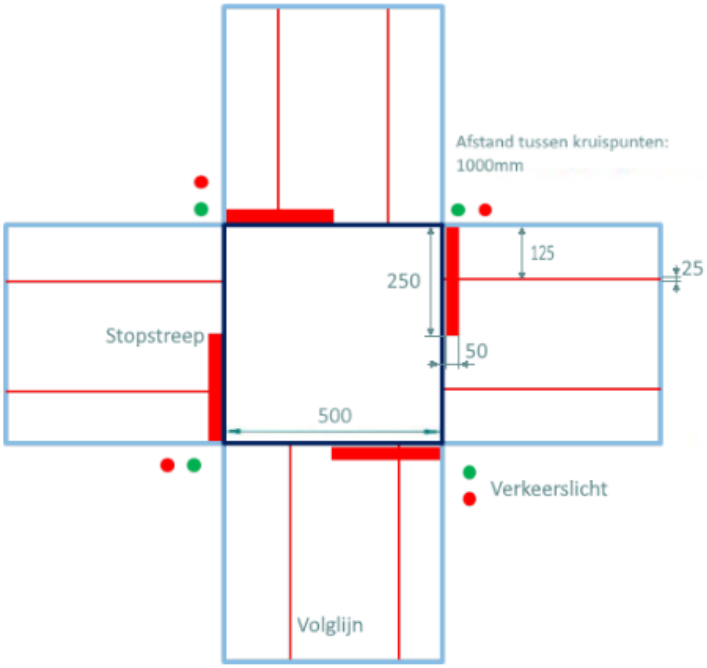
Commando's van een computer kunnen volgen:
Vooraleer onze auto het traject mag oprijden, moeten we eerst op afstand een noodstop kunnen uitvoeren.

De auto moet aan een bepaald budget voldoen:
Er is een budget van 3500 eenheden beschikbaar. Dit budget kunnen we gebruiken om materiaal aan te kopen op de site: http://www.irkulak.be/po2/. Verder kunnen we het budget ook nog gebruiken om onze eigen ontwerpen te 3D-printen.

Het volledige traject foutloos kunnen afleggen met een aanvaardbaar tempo:
De wagen heeft een maximumbreedte van $250mm$, een maximumhoogte van $300mm$ en een minimumhoogte van $75mm$.



Figuur 4.1: Technische tekening verkeerslicht, opgehaald van [1].



Figuur 4.2: Bovenaanzicht kruispunt met relevante maten en items, aangepast vanuit [2].

4.3 Samenvatting klantenvereisten

Voor de bouw van ons wagentje zijn er meerdere vereisten waar we rekening mee moeten houden. Dit document is een samenvatting hiervan.

De constructie van onze wagen moet aan een bepaald budget voldoen van 3500 eenheden. Er is een materiaallijst beschikbaar die we kunnen raadplegen om onderdelen te bestellen.

Onze auto moet een geïmplementeerd traject kunnen volgen via een lijn die op de ondergrond is aangebracht. Deze verschilt qua intensiteit met de ondergrond zodat de auto de lijn kan herkennen. Het traject bevat ook meerdere kruispunten met verkeerslichten. Onze auto moet deze kunnen interpreteren en gepast reageren op het rode of groene licht. Om te kunnen stoppen bij het rode licht zal onze auto een dikke stoplijn moeten kunnen detecteren.

Om het traject te volgen zal onze auto dus moeten kunnen rijden en draaien. Ook stoppen is belangrijk. Onze auto moet namelijk voorliggers kunnen detecteren en op tijd stoppen om een botsing te vermijden. Er wordt verwacht dat het volledige traject foutloos uitgereden kan worden op een aanvaardbaar tempo.

Voor onze auto het traject op kan, zullen we ook een noodstop moeten uitvoeren. Dit zal via commando's van een computer moeten gebeuren.

4.4 Teamverantwoordelijkheden

Verantwoordelijkheidsstructuur

Penningmeester: Kobe De Weerdt

Teamleider: Karl Van Holder

Notulist: Thomas Varheust

3D-modelleren + technische tekeningen: Yaron Verhulst + Thomas Varheust

Programmeren: Matthijs Deforche + Karl Van Holder

Verantwoordelijke bouwen: Yaron Verhulst + Karl Van Holder

Verantwoordelijke aankopen: Kobe De Weerdt

Presentatie: Matthijs Deforche + Kobe De Weerdt

4.5 Takenstructuur

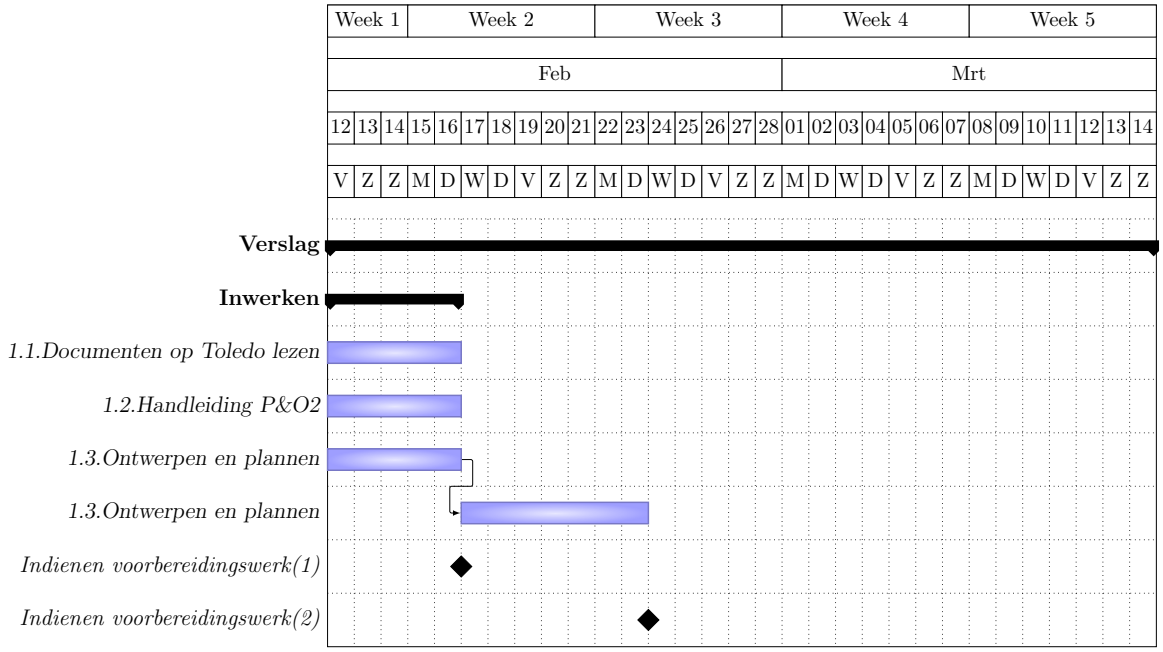
Tabel 1: Takenstructuur

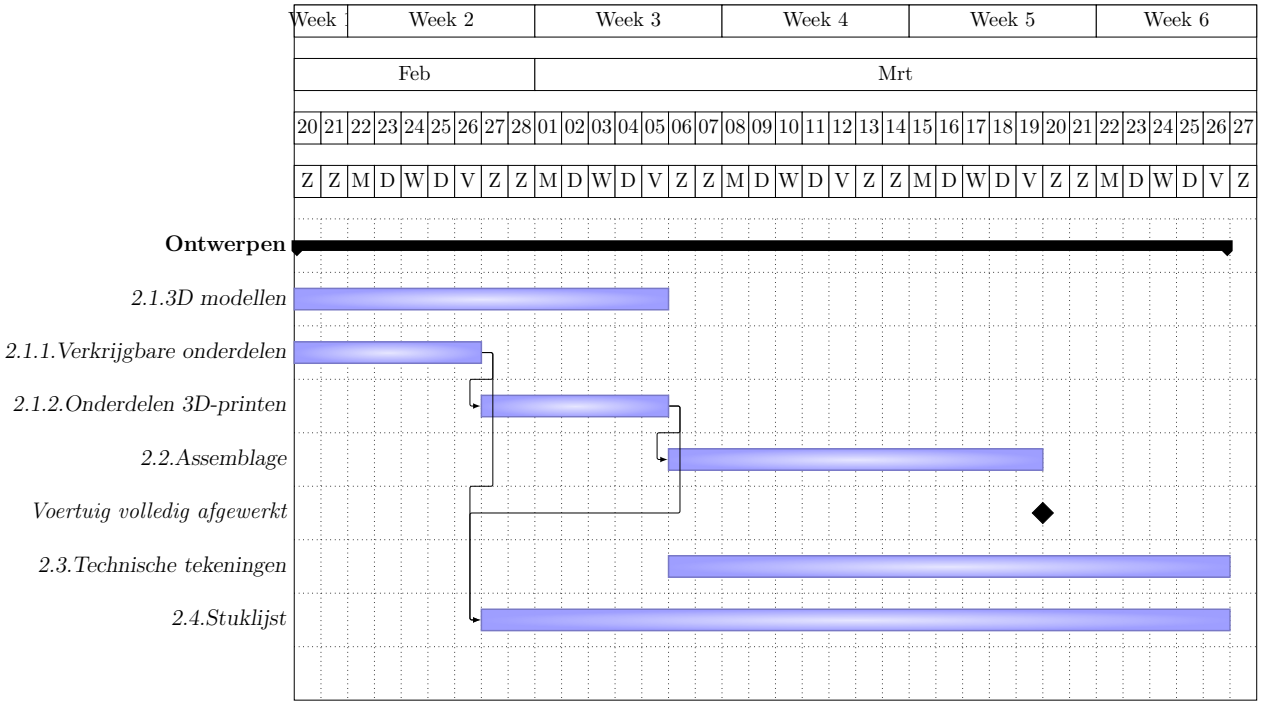
Code	Taak	Status
1	Inwerken	OK
1.1	Documenten op Toledo lezen	OK
1.2	Ontwerpen en plannen	OK
1.2.1	Materiaallijst maken	OK
1.2.2	Teamkalender maken	OK
1.2.3	Klantenvereisten opstellen	OK
1.2.4	Overzicht ontwerpspecificaties	OK
1.2.5	Takenstructuur	OK
1.2.6	Gantt-grafiek	OK
2	Ontwerpen met behulp van de computer	niet OK
2.1	3D modellen (Solid parts)	OK
2.1.1	Wiel	OK
2.1.2	Motor	OK
2.1.3	Motorbeugel	OK
2.1.4	Chassis	OK
2.1.5	Ballcaster	OK
2.1.6	Printplaat	OK
2.1.7	Kleursensor	OK
2.1.8	Afstandssensor	OK
2.2	Assemblage (Assembly)	OK
2.3	Technische tekeningen (Drawing)	niet OK
2.4	Stuklijst	niet OK
3	Software	niet OK
3.1	Sturen/snelheid regelen	OK
3.2	Lijnvolgalgoritme	OK
3.3	Verkeerslichtinterpretatie	OK
3.4	Voorliggerdetectie	OK
3.5	Stoppen	OK
3.6	Vertragen	OK
3.7	Handmatig besturen	OK
3.8	Inputs vormgeven	niet OK
3.9	Outputs vormgeven	niet OK
3.10	Testen	niet OK
4	Rapportering	niet OK
4.1	Tussentijds verslag	OK
4.1.1	Nalezen	OK
4.2	Tussentijdse presentatie	niet OK
4.2.1	Structuur	OK
4.2.2	Presentatie maken	niet OK
4.2.3	Nalezen	niet OK
4.2.4	Inoefenen	niet OK
4.3	Eindverslag	niet OK
4.3.1	Structuur	niet OK
4.3.2	Nalezen	niet OK
4.4	Eindpresentatie	niet OK
4.4.1	Structuur	niet OK
4.4.2	Presentatie maken	niet OK
4.4.3	Nalezen	niet OK
4.4.4	Inoefenen	niet OK

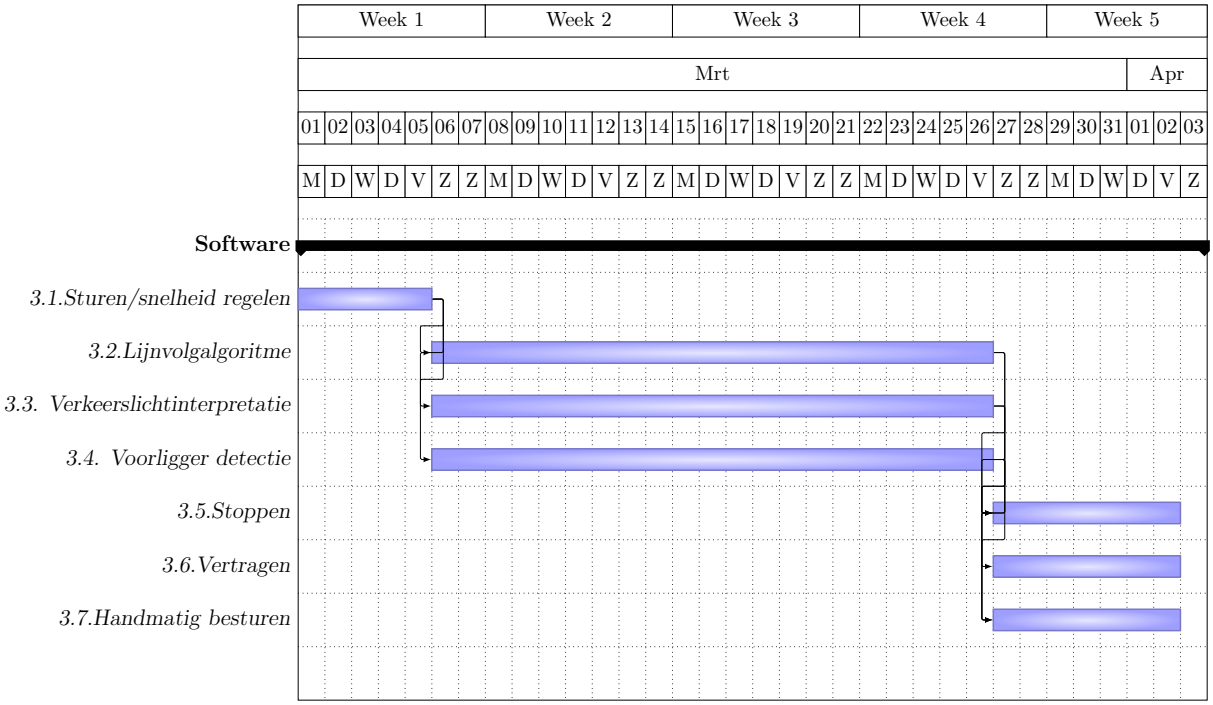
Tabel 4.1: Takenstructuur

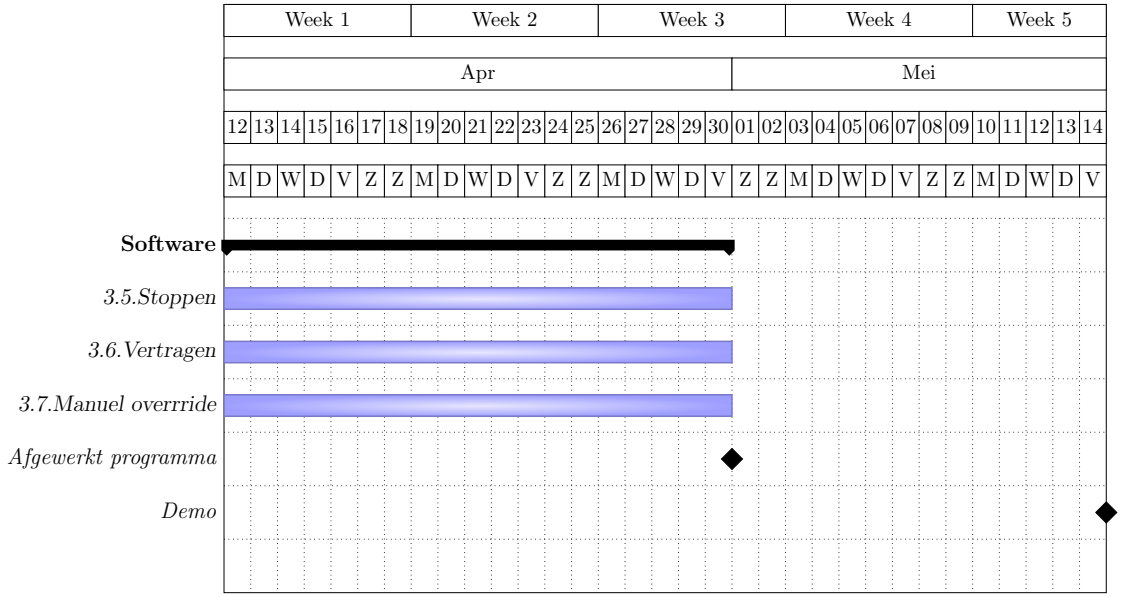
4.6 Gantt-grafiek & Teamkalender

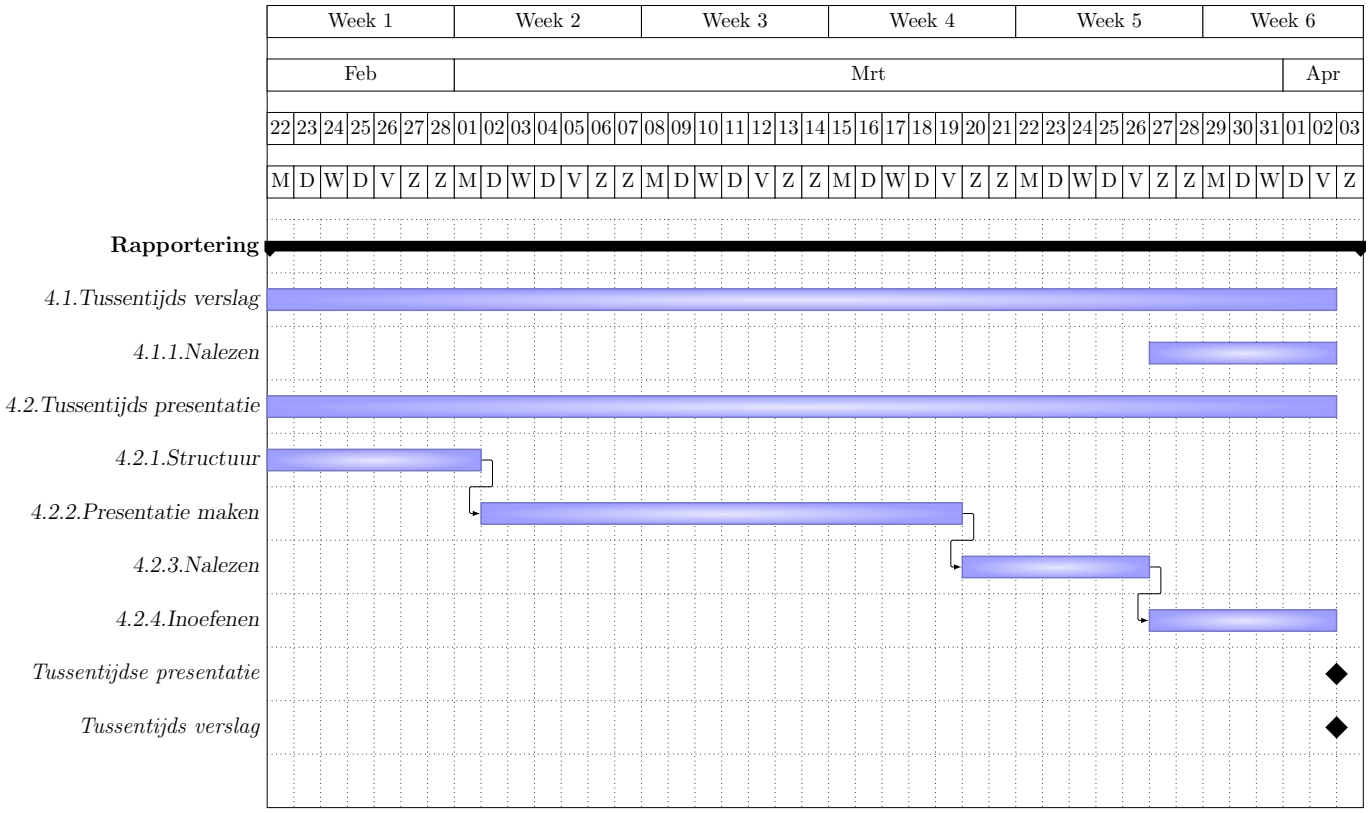
Week	Data	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag
1	08/02-12/02					
2	15/02-19/02		1.3			
3	22/02-26/02		1.4			
4	01/03-05/03	4.2.1				2.1.2/3.1
5	08/03-12/03					
6	15/03-19/03					2.2/4.2.2
7	22/03-26/03					2.3/2.4/4.2.3/3.2/3.3/3.4
8	29/03-02/04					tussentijdse presentatie en verslag
Paasvakantie1	05/04-09/04					
Paasvakantie2	12/04-16/04					
9	19/04-23/04	4.4.1				
10	26/04-30/04					3.5/3.6/3.7
11	03/05-07/05					demonstratie/4.4.2
12	10/05-14/05					4.4.3
13	17/05-21/05				eindverslag	eindpresentatie

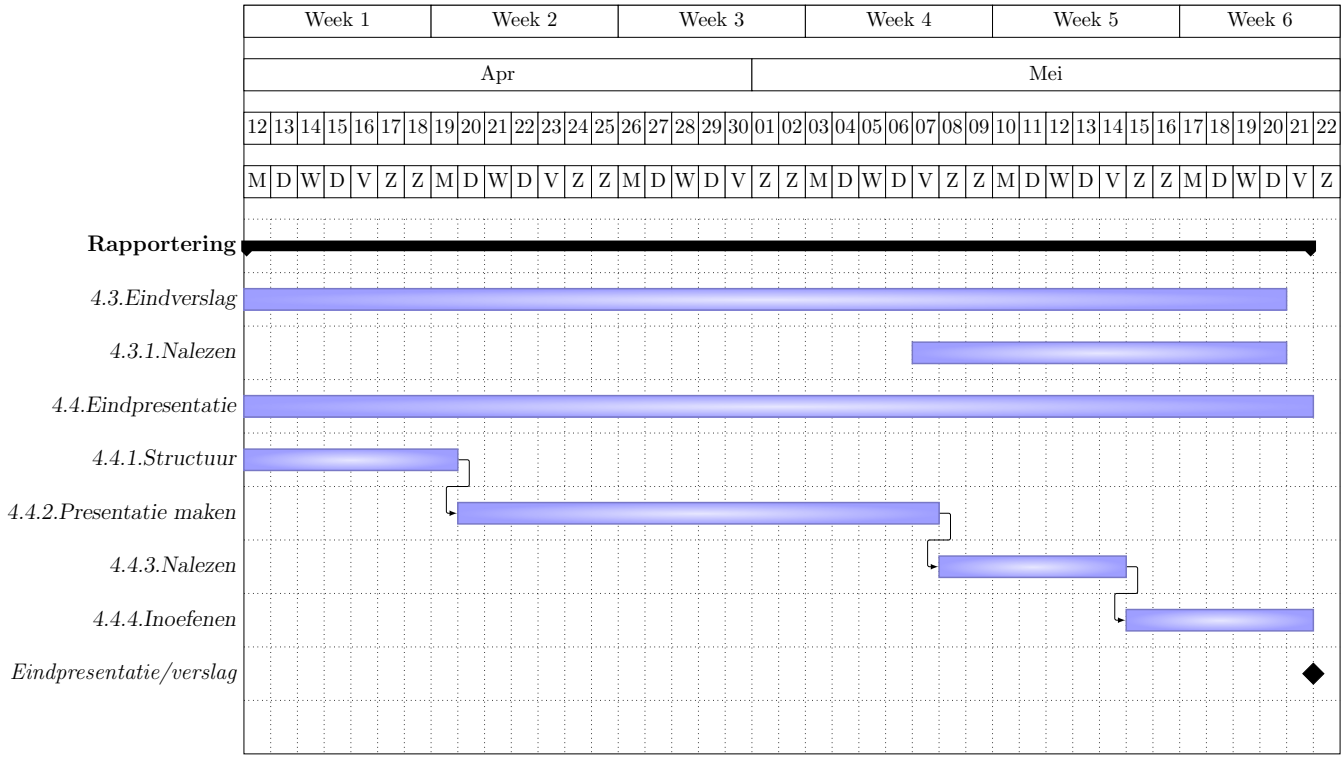

Week 1
Week 2
Week 3
Week 4
Week 5
Feb
Mrt
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 01 02 03 04 05 06 07 08 09 10 11 12 13 14
V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z
Verslag
Inwerken
1.1.Documenten op Toledo lezen
1.2.Handleiding P&O2
1.3.Ontwerpen en plannen
1.3.Ontwerpen en plannen
Indienen voorbereidingswerk(1)
Indienen voorbereidingswerk(2)


Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Feb
Mrt
20 21 22 23 24 25 26 27 28 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z
Ontwerpen
2.1.3D modellen
2.1.1.Verkrijgbare onderdelen
2.1.2.Onderdelen 3D-printen
2.2.Assemblage
Voertuig volledig afgewerkt
2.3.Technische tekeningen
2.4.Stuklijst

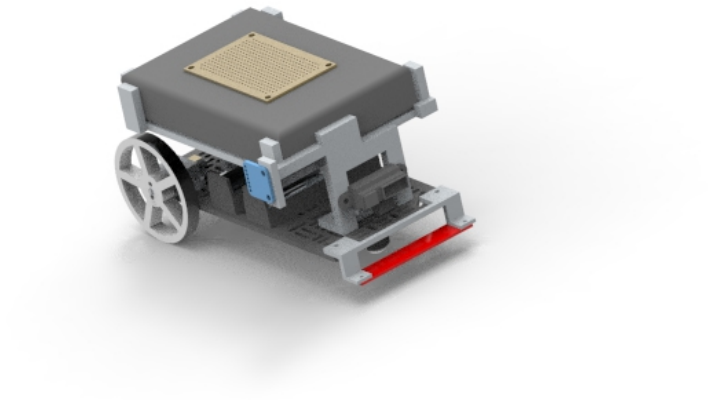

Week 1
Week 2
Week 3
Week 4
Week 5
Mrt
Apr
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 01 02 03
M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z
Software
3.1.Sturen/snelheid regelen
3.2.Lijnvolgalgoritme
3.3. Verkeerslichtinterpretatie
3.4. Voorligger detectie
3.5.Stoppen
3.6.Vertragen
3.7.Handmatig besturen


Week 1
Week 2
Week 3
Week 4
Week 5
Apr
Mei
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 01 02 03 04 05 06 07 08 09 10 11 12 13 14
M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V
Software
3.5.Stoppen
3.6.Vertragen
3.7.Manuel overrride
Afgewerkt programma
Demo

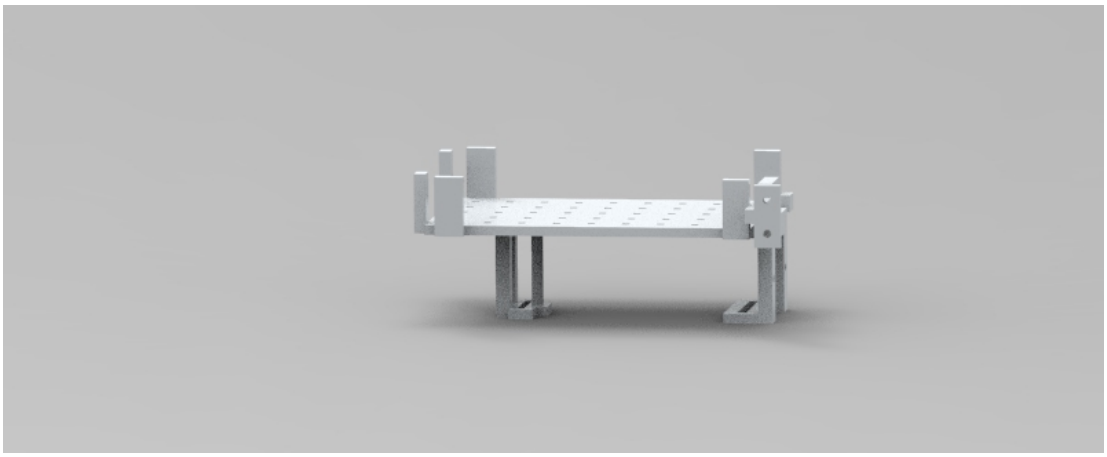

Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Feb
Mrt
Apr
22 23 24 25 26 27 28 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 01 02 03
M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z
Rapportering
4.1.Tussentijds verslag
4.1.1.Nalezen
4.2.Tussentijds presentatie
4.2.1.Structuur
4.2.2.Presentatie maken
4.2.3.Nalezen
4.2.4.Inoefenen
Tussentijdse presentatie
Tussentijds verslag


Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Apr
Mei
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22
M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z M D W D V Z Z
Rapportering
4.3.Eindverslag
4.3.1.Nalezen
4.4.Eindpresentatie
4.4.1.Structuur
4.4.2.Presentatie maken
4.4.3.Nalezen
4.4.4.Inoefenen
Eindpresentatie/verslag

4.7 Visualisatie ontwerp



Figuur 4.3: CAD-ontwerp van de wagen, samengesteld uit *.par-onderdelen*, gezien vanuit de rechterbovenhoek.

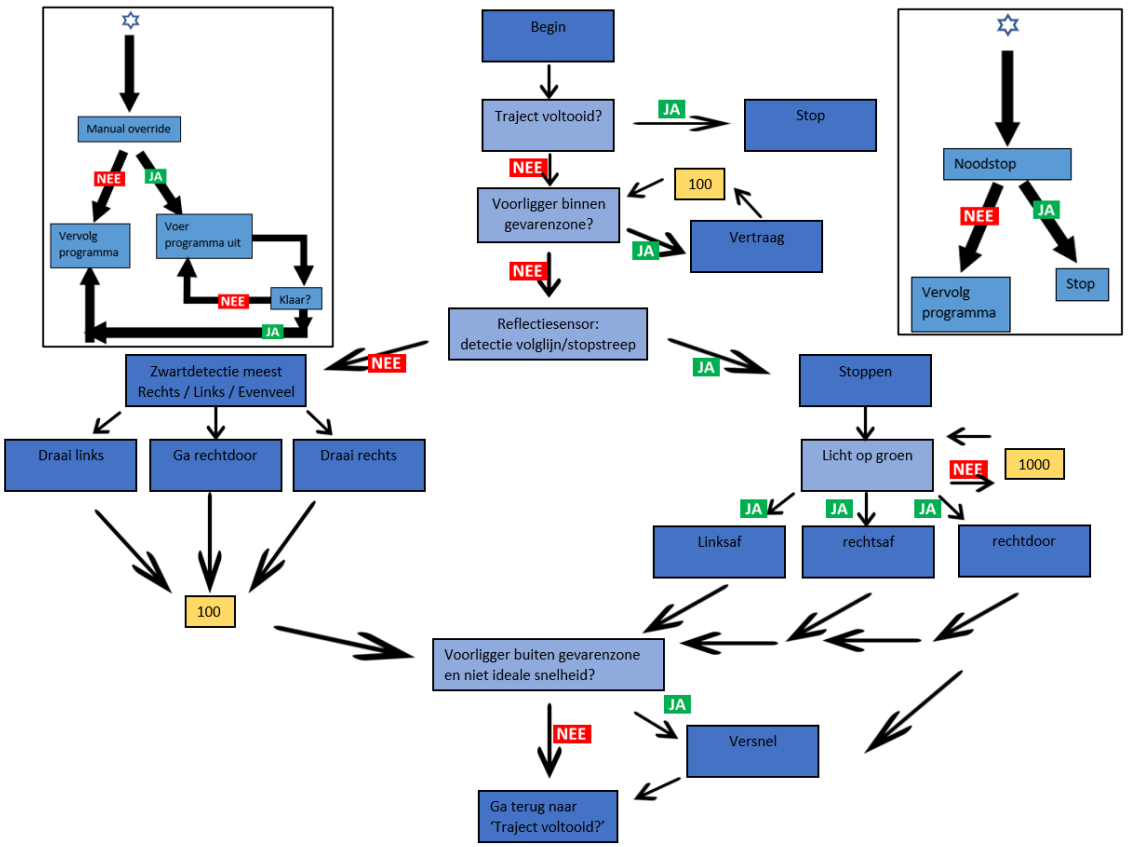


Figuur 4.4: Het 3D-geprinte onderdeel *Skelet van de wagen*

.

4.8 Elektrisch circuit

4.9 Flowcharts



Figuur 4.5: De flowchart van ons algoritme.

4.10 Financieel Rapport

Hardware

Onderdeel	prijs per stuk	aantal	totale prijs
Micro Metal Gear Motor 100:1 HP	160	2	320
Dual Drive DRV8833	70	1	70
Optische afstandsensor (analoog)	160	1	160
TCS34725 Kleur sensor BOB	150	1	150
QTR-8A analoge reflectie sensor array	150	1	150
Oplaadbare LITHIUM-ION	90	2	180
NI MyRio	240	1	240
Breadboard Tiny	40	1	40
Printplaat	50	1	50
Totaal: 1360			

Tabel 4.2: uitgaven hardware

Ontwerp

Onderdeel	prijs per stuk	aantal	totale prijs
Ball Caster	60	1	60
Wiel 60x8mm zwart	35	2	70
Robot Chassis Rechthoekig Zwart	70	1	70
Totaal: 200			

Tabel 4.3: uitgaven ontwerp

Assemblage

Onderdeel	prijs per stuk	aantal	totale prijs
Micro metal gear motor beugel	2	25	50
Skelet van de wagen	N.A	1	N.A
Reflectie sensor houder	N.A	1	N.A
Totaal: 50			

Tabel 4.4: uitgaven assemblage

Onkosten

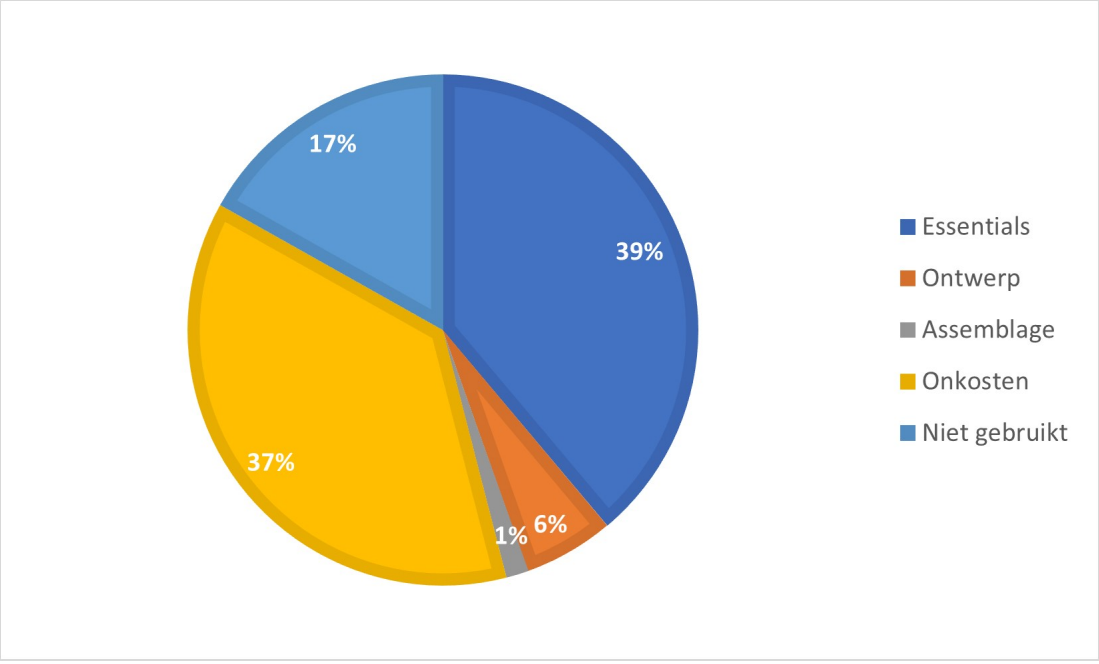
Niet gebruikt

Totaal: 590

kost	waarde
Bieding	1300

Totaal: 1300

Tabel 4.5: uitgaven onkosten



Figuur 4.6: Visualisatie van de besteding van ons budget

Bibliografie

[1] Kevin Truyaert Benjamin Maveau. Opgave teamopdracht probleemoplossen en ontwerpen 2.

[2] Martijn Boussé Benjamin Maveau, Kevin Truyaert. P&o2 : Smart city.

Team 4: Team R2D2
Groep Wetenschap & Technologie
KU Leuven Kulak
Etienne Sabbelaan 53, 8500 Kortrijk