

Ontwikkeling van een autonomoos lijn-volgend racevoertuig

Jorik DE BRUYCKER
Jonas BOLLE

Begeleiders:
Leenders Guus
Crul Stijn
Naessens Carine
Van der Perre Liesbet

Industriële ingenieurswetenschappen
Elektronica-ICT: Elektronica

Coach: Cox Bert

©Copyright KU Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor(en) als de auteur(s) is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, kan u zich richten tot KU Leuven Technologiecampus Gent, Gebroeders De Smetstraat 1, B-9000 Gent, +32 92 65 86 10 of via e-mail iiw.gent@kuleuven.be.

Voorafgaande schriftelijke toestemming van de promotor(en) is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Inhoudsopgave

1 Inleiding	3
1.1 Opdrachtbeschrijving	3
1.1.1 Hardware	4
1.1.2 Software	4
1.2 Doelstellingen	4
1.3 Structuur	4
2 Taakverdeling	5
3 Onkosten	6
4 Toelichting hardware	7
4.1 PCB-ontwerp	7
4.1.1 Eagle-ontwerp	7
4.1.2 Maken van de PCB	12
4.2 Sensoren en toebehoren	15
4.2.1 Infrarood-sensoren	15
4.2.2 Multiplexers	17
4.2.3 Hall-sensor en magneten	17
4.2.4 RFID-reader	18
4.2.5 Bluetooth-module	19
4.2.6 Pintoewijzing	19
4.3 Raspberry Pi	20
5 Toelichting software	21
5.1 IR-sensoren inlezen	21
5.2 Rijden en PID-regeling	22
5.2.1 Bijsturen van de motoren	22
5.2.2 Berekening van foutwaarde	22
5.2.3 PID-regeling	23
5.3 Snelheid meten	23
5.4 RFID-tags inlezen	24

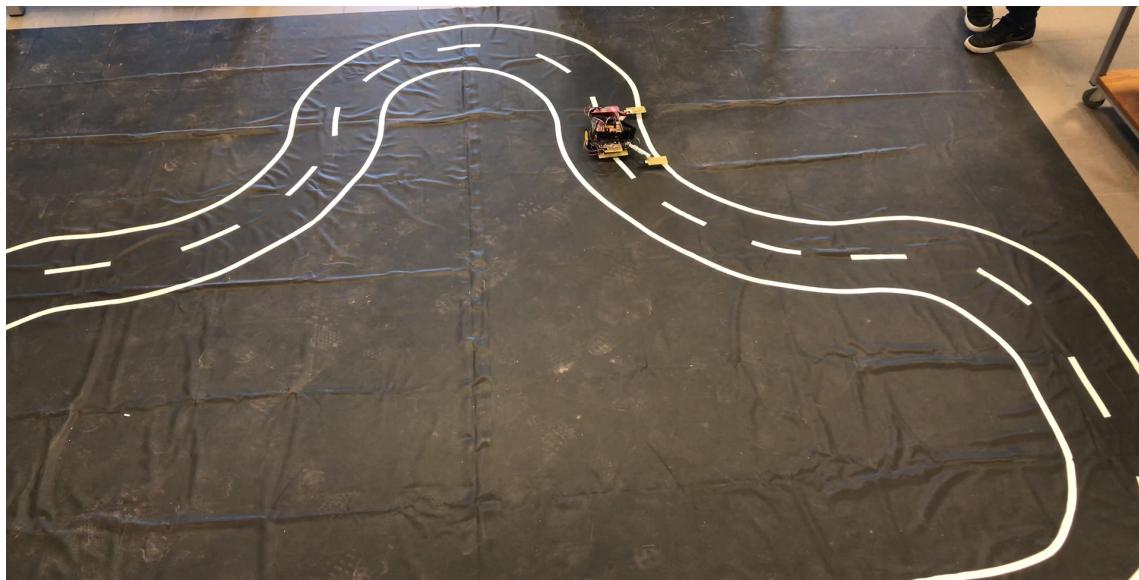
5.5	Bluetooth-communicatie naar Raspberry Pi	24
5.5.1	Arduino met HC05-module als Slave	24
5.5.2	Raspberry Pi 3 met ingebouwde Bluetooth-adapter als Master	24
5.6	Custom board progammeren	25
6	Problemen en moeilijkheden	27
6.1	Rijden aan de hand van PID-regeling	27
6.1.1	Positionering van de sensoren	27
6.1.2	Bepalen van de PID-constanten	27
6.2	Bluetooth-communicatie	28
6.3	RFID-lezer	28
6.3.1	Libraries	28
6.3.2	Duur van het inlezen	28
6.4	Arduino PCB	28
7	Evaluatie coach	29
8	Besluit	30

Hoofdstuk 1

Inleiding

1.1 Opdrachtbeschrijving

In het kader van ons bachelorproject ontwikkelen we een wagentje dat autonoom zo snel mogelijk een raceparcours kan afleggen. Dit parcours bestaat uit een zwarte ondergrond aangebakend door volle witte lijnen en een witte stippellijn in het midden, een voorbeeld hiervan ziet u in figuur 1.1. Ons wagentje mag niet buiten de volle witte lijnen van het parcours treden. Tevens zullen er op het traject verschillende RFID-tags geplaatst worden die door het voertuig ingelezen moeten worden. Eveneens moet er een manier voorzien worden om de snelheid van het voertuig te meten. De ingelezen tags en opgemeten snelheid moet ten slotte nog verzonden worden naar een Raspberry Pi. Voor het aanschaffen van nodige hardware krijgen we 50 euro ter beschikking.



Figuur 1.1: Voorbeeld parcours dat ons wagentje dient af te leggen

1.1.1 Hardware

We krijgen reeds een wagentje met twee motoren en een batterij om te beginnen. Voor het prototypen en testen mogen we gebruik maken van een Arduino Uno met motor shield, een belangrijk deel van de opdracht bestaat er uit deze Arduino én motor shield te vervangen door één custom board. Qua hardware moeten we dus nog de custom board ontwikkelen, sensoren voorzien en uiteraard ook modules aanschaffen voor (Bluetooth) communicatie met de RPi en het inlezen van RFID-tags.

1.1.2 Software

De besproken hardware moet natuurlijk ook correct aangestuurd worden door software. Dit zal gebeuren aan de hand van Arduino-code ondersteund door C/C++ libraries. In deze software moeten voorzieningen getroffen worden voor het aansturen van de motoren, het inlezen van sensoren, communicatie met Raspberry Pi,... Aan de kant van de Raspberry Pi zullen we verbinding moeten maken met ons wagentje en de data die doorgestuurd wordt af printen op het scherm.

1.2 Doelstellingen

De belangrijkste doelstellingen van ons project zijn dus de volgende:

- Voertuig autonoom en zo snel mogelijk een parcours laten afleggen
- Custom board ontwerpen om Arduino en motorshield te vervangen
- Snelheid van het wagentje meten
- RFID-lezer voorzien voor het uitlezen van tags op het parcours
- Communicatie (Bluetooth) tussen Arduino en Raspberry Pi 3 realiseren

1.3 Structuur

We beginnen dit verslag met een bespreking van onze planning, taakverdeling en gemaakte onkosten. Daarna behandelen we de gebruikte hardware en software om deze aan te sturen in meer detail. Vervolgens overleggen we de problemen en moeilijkheden die we doorheen het project ondervonden. Hierna volgt nog een evaluatie van onze coach, Bert Cox. Ten slotte eindigen we met een besluit omtrent het project waarin we samenvatten wat we gerealiseerd hebben, wat we hebben bijgeleerd, wat er beter kan/kon,...

Hoofdstuk 2

Taakverdeling

Groepslid Periode		Bolle Jonas	De Bruycker Jorik
nov	jan	Oriënteren omtrent opdracht, overleggen van ideeën, voorbereidend opzoekwerk	
1/feb	11/feb	Bestellen benodigheden, concrete afspraken, aanvang project	
12/feb	25/feb	Custom Arduino: research	Ontwerp IR-sensorarrays
26/feb	11/mrt	Custom Arduino: Eagle schematic	Start software sensoren en bijsturen
12/mrt	25/mrt	Custom Arduino: Eagle routing	Software PID-regeling
21/mrt		Eerste statusupdate	
26/mrt	1/apr	Custom Arduino: etsen	Snelheidsmeter voorzien
2/apr	15/apr	Custom Arduino: bestukken, solderen en testen	Bluetooth-communicatie met RPi3, RFID-lezer voorzien
16/apr	27/apr	Verslag schrijven	
18/apr		Demonstratie/race	
27/apr		Verslag indienen	
28/apr	8/mei	Presentatie voorbereiden	
9/mei		Presentatie	

Tabel 2.1: Planning en taakverdeling

Hoofdstuk 3

Onkosten

We kregen voor dit project een bedrag van 50 euro ter beschikking. Onze uitgaven van dit budget kan u vinden in tabel 3.1. In totaal komt dit neer op een bedrag van 21,72 euro.

order nr. + beschrijving	prijs/stuk (euro)	aantal	totaal (euro)
1703519: TCRT5000L infrarood sensor	0.741	10	7.41
1470593: SS41 hall sensor	0.88	6	5.28
1106109: CD4051BE 8:1 analog multiplexer	0.415	5	2.08
1703519: TCRT5000L infrarood sensor (later extra bijbesteld)	0.695	10	6.95

Tabel 3.1: Onkosten

Andere vereisten zoals de Bluetooth-module en RFID-lezer werden voorzien zonder zelf te bestellen.

Hoofdstuk 4

Toelichting hardware

In dit hoofdstuk bekijken we de gebruikte hardware meer in detail. We bespreken eerst het ontwerp van ons custom Arduino board. Vervolgens gaan we dieper in op de gebruikte sensoren en modules. Ten slotte halen we kort enkele eigenschappen van de Raspberry Pi aan die voor ons relevant zijn.

4.1 PCB-ontwerp

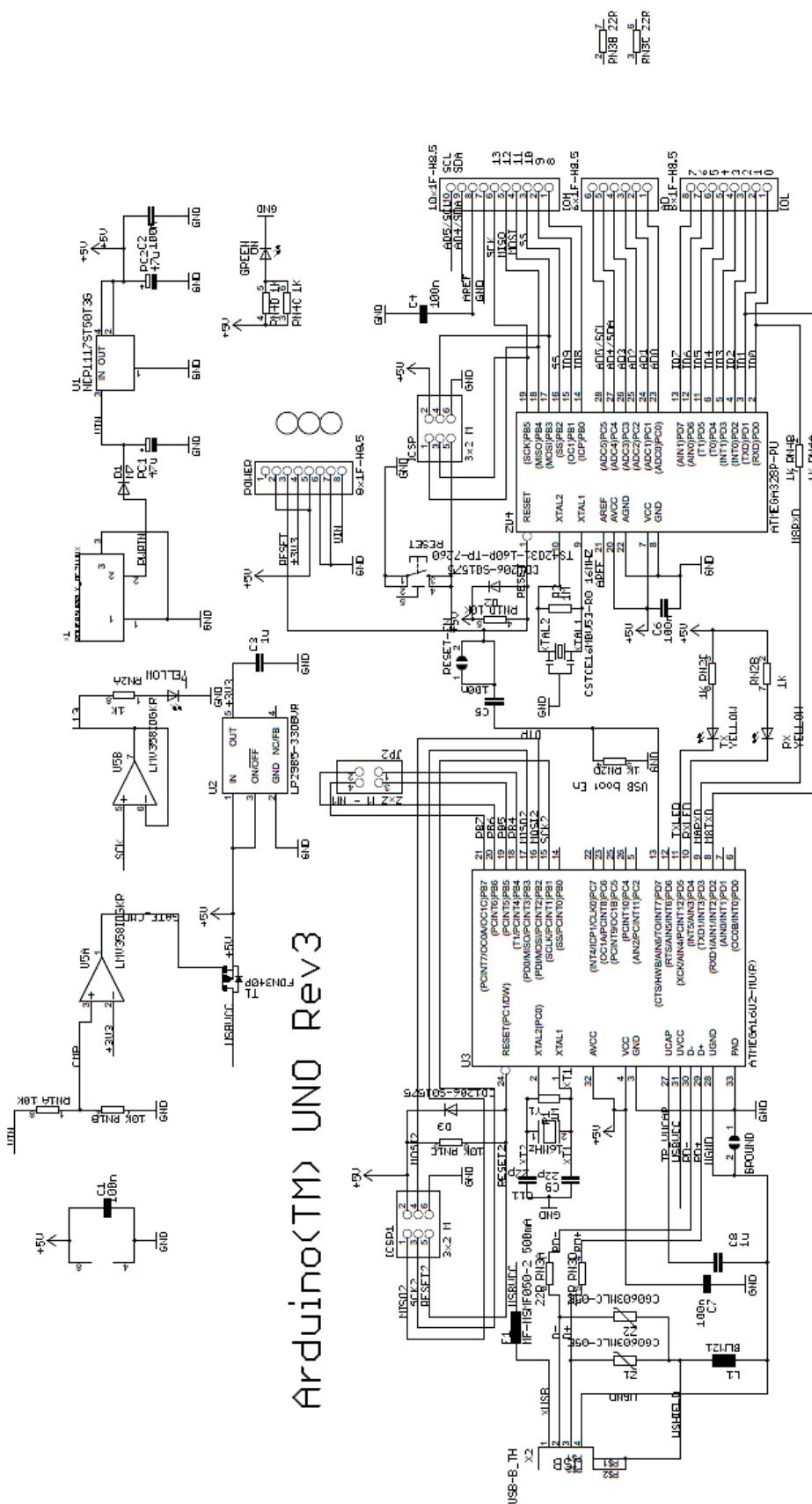
Voor het ontwerp van de custom Arduino PCB hebben we gewerkt met Eagle. Vertrekkende van de officiële schematic van de Arduino Uno maken we onze eigen versie hiervan. Na de ontwerp fase kunnen we de PCB etsen, boren, bestukken en solderen. Als dit allemaal achter de rug is, testen we onze custom Arduino en lossen we nog eventuele problemen op.

4.1.1 Eagle-ontwerp

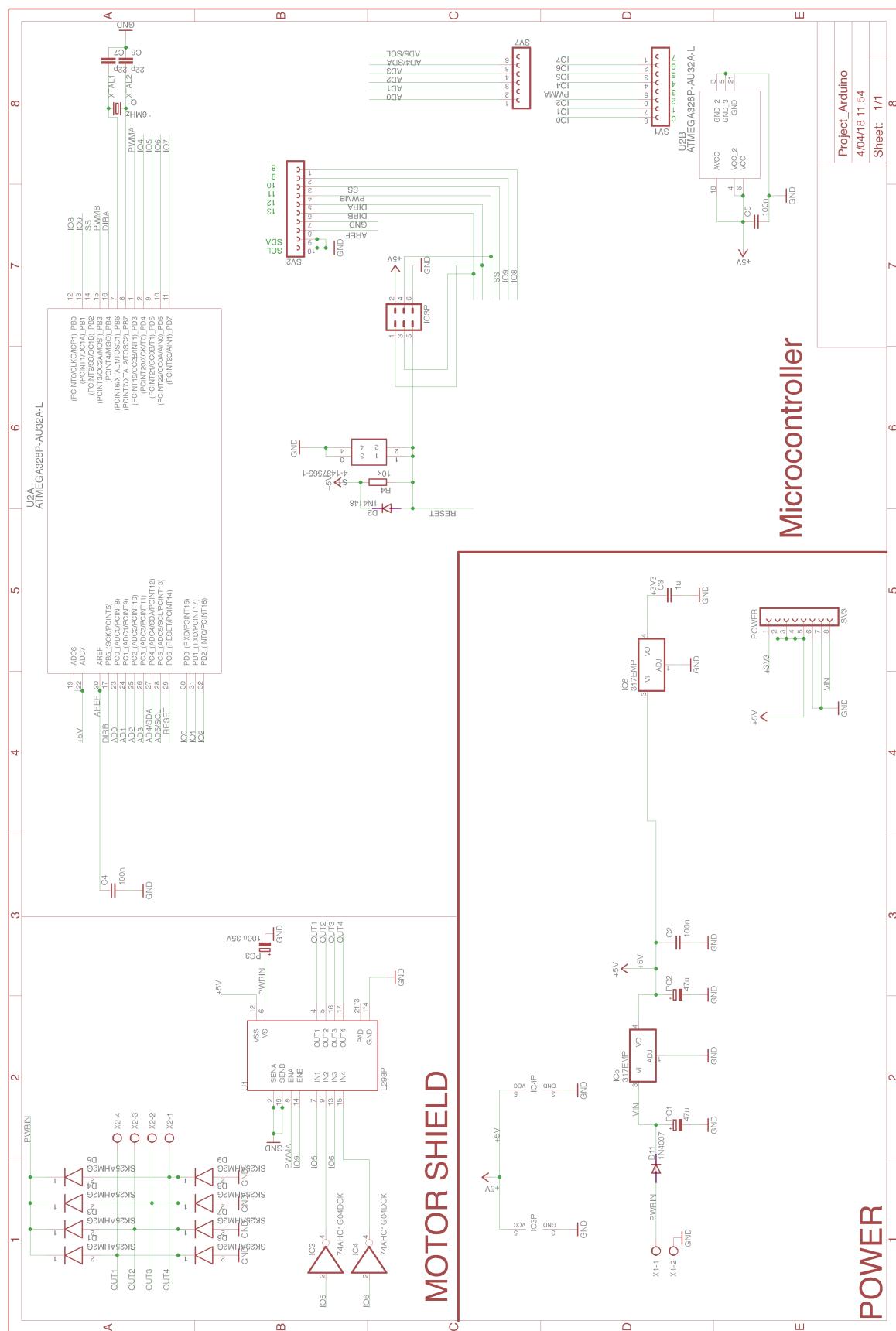
Schematic

Om onze custom board te ontwerpen moeten we uiteraard niet helemaal vanaf nul beginnen. We vertrokken van het officiële schema dat makkelijk online te vinden is (zie figuur 4.1 op pagina 9). We bestudeerden dit schema grondig om te bepalen welke componenten we effectief nodig hebben en welke we weg kunnen laten. Hieruit bleek dat er veel onderdelen uit het originele ontwerp niet vereist zijn voor onze doeleinden. Zoals bijvoorbeeld de USB to serial interface, aangezien we besloten om onze custom Arduino te programmeren via de ICSP-pinnen, is de aanwezigheid van een USB-connectie overbodig. Dit zorgt ervoor dat de ATMEGA16U2 chip en al zijn connecties niet nodig zijn op onze eigen PCB. Ook de voeding van de microcontroller wordt hierdoor minder complex: we hoeven ons namelijk geen zorgen te maken om het gedeelte die de batterijspanning afkoppelt als er een USB-connectie gemaakt wordt. Vervolgens zijn ook alle LED's uit het officiële ontwerp weggehaald, zodat nog een IC (dat twee opamps bevat), enkele weerstanden en een ont-koppelcondensator kunnen weggeleggen worden. Ten slotte integreren we de Arduino Motorshield in ons ontwerp. De hoofdcomponent van dit gedeelte bestaat uit de L298P dual full-bridge driver en bevat de nodige afschermdiodes voor de motor outputs. Het resultaat is onze vereenvoudigde versie van de Arduino Uno (zie figuur 4.2 op pagina 10) dat we vervolgens kunnen gaan boarden. We lichten ook even de dual full-bridge driver toe. Deze chip is verantwoordelijk voor het aansturen

van de snelheid en richting van onze dc-motoren. Het schema van deze chip kan u vinden in figuur 4.3 op pagina 11. Er zijn drie belangrijke aansluitingen aanwezig: de IN-, EN- en OUT-pinnen. Aan de IN-pinnen sluiten we een signaal aan dat de draairichting van de motoren bepaalt. Hiervoor gebruiken we de signalen dirA en dirB in ons schema. Concreet wordt dirA op IN1 en zijn inverse op IN2 aangesloten (zelfde principe voor dirB op IN3 en IN4). Dit zorgt ervoor dat de AND-poorten in de chip de transistoren die dienst doen als schakelaar, in cut-off of in saturatie te sturen, op deze manier kan de polarisatie van de spanning over de OUT-klemmen gewijzigd worden. De snelheid van de motoren wordt vervolgens aangestuurd door een PWM-signal. Dit signaal komt aan de enable-ingangen van alle vier de AND-poorten. Op deze manier kunnen we aan de hand van de duty-cycle van het PWM-signaal de snelheid van de motoren bepalen. Dit signaal komt dan terecht op de OUT-klemmen, met een amplitude gelijk aan de Vs-ingangsspanning.

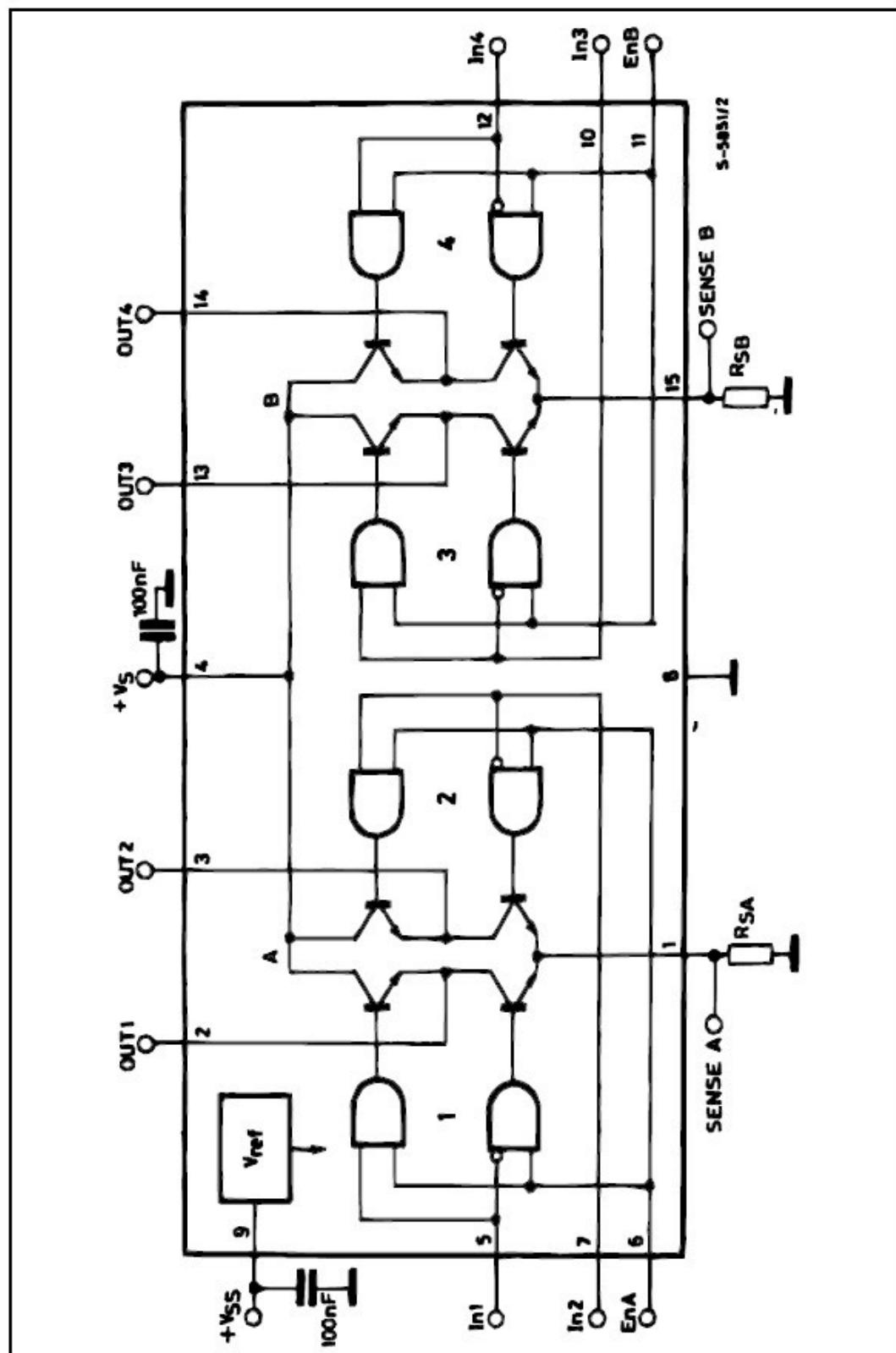


Figuur 4.1: Officiële schema van de Arduino Uno



Figuur 4.2: Eigen schema van de Arduino Uno

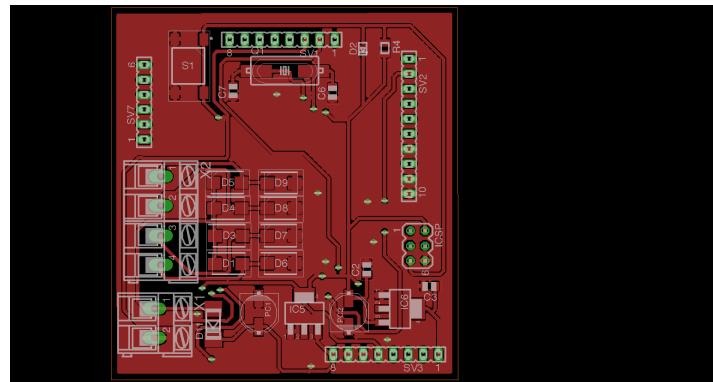
BLOCK DIAGRAM



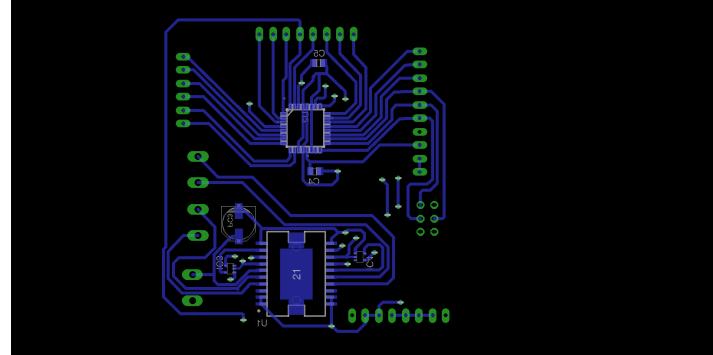
Figuur 4.3: Block diagram van de dual full-bridge driver

Board

Nu onze eigen schematic afgerond is, kunnen we beginnen boarden. We trachten de PCB zo klein mogelijk te maken door de componenten zowel op de top- als de bottom-layer te plaatsen. We hebben ervoor gekozen de twee chips (de ATMEGA328P en de L298P) op de bottom-layer te plaatsen samen met de nodige ontkoppelcondensatoren. De andere componenten staan op de top-layer. Na het routing proces bestaat onze PCB uit 20 via's. We hebben ook een massavlak voorzien aan de bovenkant. De top- en bottom-layer van de PCB kan u vinden in respectievelijk figuur 4.4 en figuur 4.5.



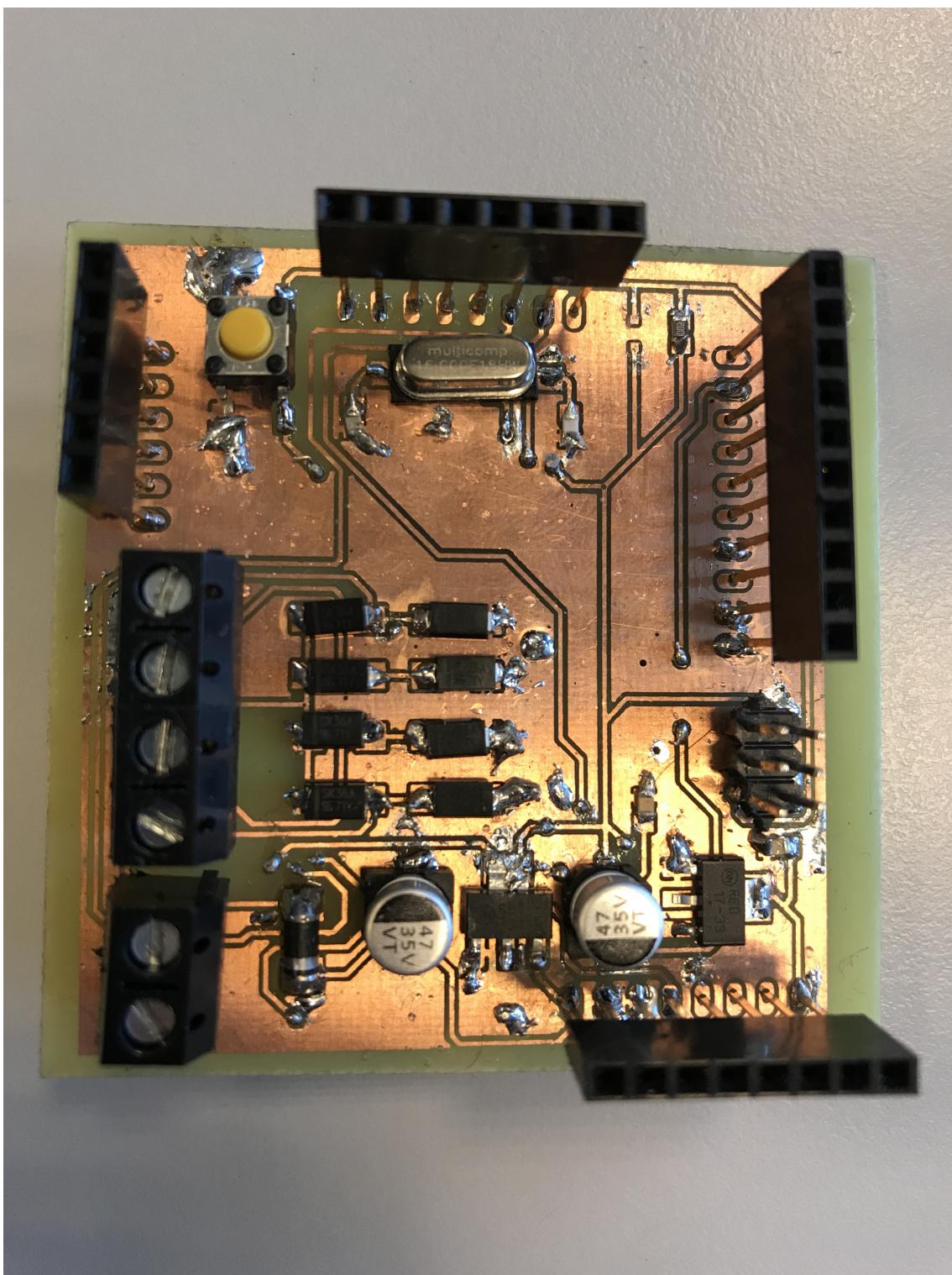
Figuur 4.4: Eigen board van de Arduino Uno



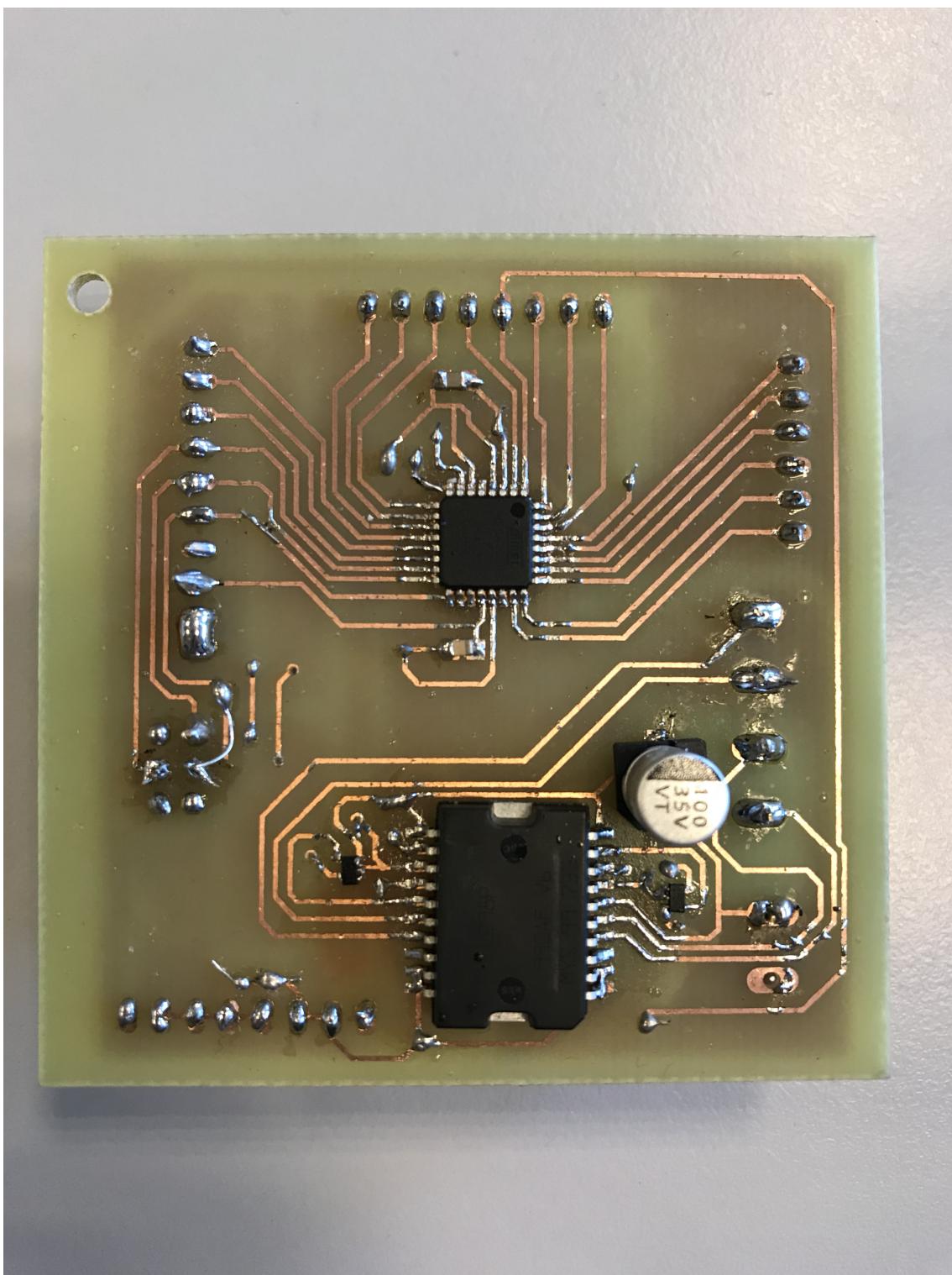
Figuur 4.5: Eigen board van de Arduino Uno

4.1.2 Maken van de PCB

Het ontwerp is volledig af en we kunnen onze custom Arduino beginnen maken. Na het etsen, boren, bestukken en solderen ziet onze eigen Arduino er als volgt uit: figuur 4.6 op de pagina hierna is de bovenkant en figuur 4.7 op pagina 14 de onderkant.



Figuur 4.6: Bovenkant van onze custom Arduino



Figuur 4.7: Onderkant van onze custom Arduino

4.2 Sensoren en toebehoren

4.2.1 Infrarood-sensoren

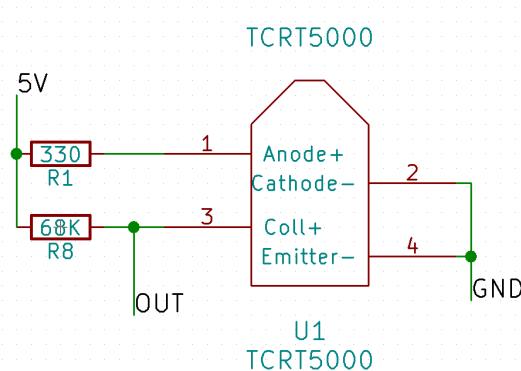
Zoals we reeds vermeldden is het de bedoeling dat ons voertuig een parcours autonoom kan afleggen dat afgebakend wordt door twee volle witte lijnen op een zwarte ondergrond met een witte stippellijn tussen beide. Om dergelijk parcours te kunnen navigeren zullen we onderscheid moeten kunnen maken tussen de witte en zwarte ondergrond. Voor dergelijke toepassing kiezen we voor infrarood-sensoren. We zullen hiervoor gebruik maken van de TCRT-5000-sensor, zoals te zien is in figuur 4.8.



Figuur 4.8: TCRT5000 infrarood sensor

Sensorcel

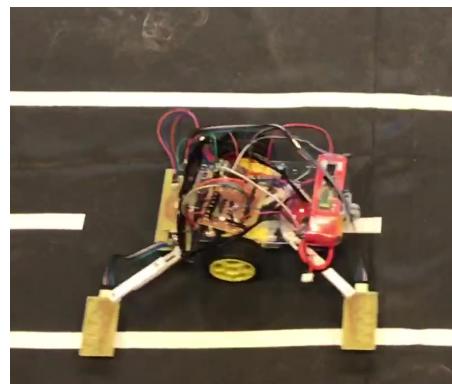
De TCRT5000 bestaat uit 2 belangrijke onderdelen: ten eerste hebben we een IR-led , deze led wordt aangestuurd met een voorschakelweerstand van 330Ω . Ten tweede heeft de TCRT5000 een transistor die zich in essentie als infrarood-gevoelige weerstand gedraagt, samen met een serieweerstand van $68\text{ k}\Omega$ vormt dit een spanningsdeler. De IR-led zal infraroodgolven ustralen, naargelang de ondergrond waar dit op invalt worden deze golven quasi volledig of amper gerefleecteerd. Bij een witte ondergrond zal de reflectie groot zijn en de weerstand tussen collector en emitter bijgevolg zeer klein zijn, omgekeerd hebben we bij een zwarte ondergrond een grote weerstand. De uitgangsspanning bij de spanningsdeler met de weerstand en de TCRT5000-pinnen zoals u ziet in figuur 4.9 op de volgende pagina geeft dus een maat voor de reflectiecoëfficiënt van het oppervlak onder de sensor. De waarde van deze spanning wordt door een analoge pin ingelezen, uit deze inlezing kan dus afgeleid worden of de sensor zich boven een witte of een zwarte ondergrond bevindt. Bij een witte ondergrond bedraagt deze waarde ongeveer 30 tot 50 terwijl een zwarte ondergrond in een waarde tussen de 600 en 800 resulteert.



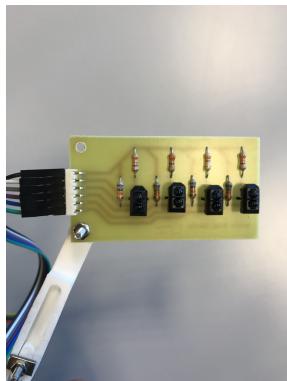
Figuur 4.9: Sensor-cel met TCRT 5000

Sensor-arrays

Om de positie van ons voertuig te detecteren ten opzichte van een witte zijlijn maakten we twee sensor-arrays van elk vier sensoren die via een multiplexer op een aparte printplaat ingelezen worden in één analoge pin. We positioneerden deze linksvooraan en linksachteraan het wagentje met behulp van 3D-geprinte en verstelbare armen, op deze manier kunnen we de zijlijn langs de linkerkant van het parcours detecteren. Het voordeel hiervan is ook dat we in de bochten van de baan vrij stabiel kunnen bijsturen. Na een aantal verschillende positioneringen bleek dit de beste resultaten op te leveren. De positionering van de sensor-arrays is te zien in figuur 4.10. In figuren 4.11 op de volgende pagina en 4.12 op de pagina hierna ziet u de twee verschillende sensorarrays aan de onderzijde.



Figuur 4.10: Sensorarray op het wagentje ten opzichte van de zijlijn



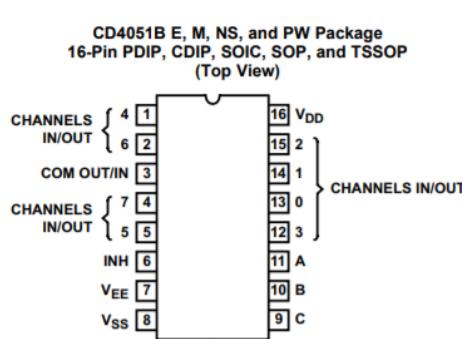
Figuur 4.11: Sensorarray linksvooraan



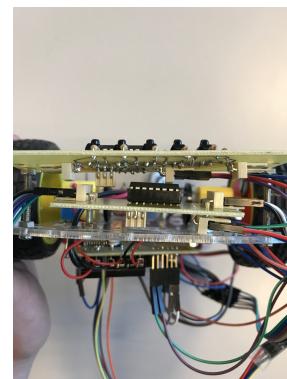
Figuur 4.12: Sensorarray linksachteraan

4.2.2 Multiplexers

Aangezien we slechts over een beperkt aantal analoge ingangen beschikken zullen we de verschillende sensoren moeten multiplexen, hiervoor maken we gebruik van een CD4051BE-multiplexer op een aparte PCB vooraan het wagentje. In figuur 4.13 ziet u de pin-out van dit IC, de PCB met de multiplexer is te zien in figuur 4.14. We gebruiken deze multiplexer om de acht sensoren in de twee sensor-arrays in te kunnen lezen op één analoge pin: A0. We gebruiken hiervoor drie digitale pinnen D0,D1 en D2, die de bit selects van de multiplexer aansturen.



Figuur 4.13: CD4051BE multiplexer pin-out



Figuur 4.14: PCB met CD4051BE multiplexer vooraan

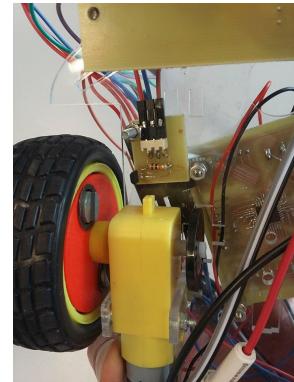
4.2.3 Hall-sensor en magneten

Voor het meten van de snelheid zullen we moeten bepalen hoeveel rotaties de wielen maken gedurende een bepaalde periode. Uit dit aantal rotaties kunnen we vervolgens de afgelegde weg en dus ook de snelheid berekenen. We hebben dus een manier nodig om te detecteren wanneer en hoeveel keer het wiel een rotatie maakt, daarvoor kozen we voor het gebruik van een Hall-sensor die het passeren van magneten gemonteerd in de wielas detecteert. De gebruikte Hall-sensor is de SS41 digitale Hall-sensor van Honeywell die gevoed wordt op 5 V. De output van deze sensor wordt aangesloten aan pin D4. In figuur 4.15 op de volgende pagina ziet u de magneten die binnen in de wielas gemonteerd werden met behulp van een 3D-geprinte houder. Merk op dat deze magneten een tegengestelde polarisatie hebben zodanig dat de output van de Hall-sensor correct

omschakelt als deze magneten beurtelings passeren. In figuur 4.16 ziet u hoe de Hall-sensor bij het wiel gemonteerd is.



Figuur 4.15: Magneten gemonteerd in wielaanpassing

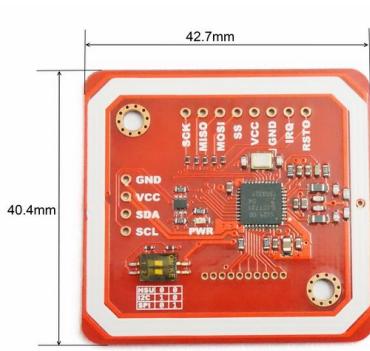


Figuur 4.16: Hall-sensor gemonteerd naast wielen

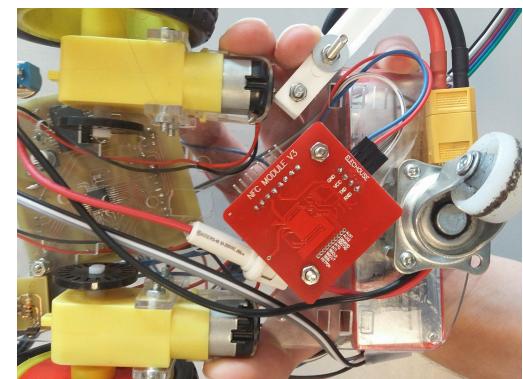
4.2.4 RFID-reader

Het inlezen van de RFID-tags gebeurt met behulp van de PN532 NFC-module v3 van Elechouse die u ziet in figuren 4.17 en 4.18. Deze module is voorzien van aansluitingen voor communicatie over HSU (High Speed UART), I²C en SPI. De gewenste communicatie-interface wordt gekozen aan de hand van de twee SMD-switches op de module. We maken gebruik van I²C om de tags in te lezen, de reden van deze keuze zullen we verder bespreken in hoofdstuk 6 op pagina 27. We sluiten dus volgende pinnen van de PN532-module aan:

- GND: Ground van de module, aangesloten aan de GND van onze Arduino.
- VCC: Voeding van de module, aangesloten aan 5 V van onze Arduino.
- SDA: Seriële data-pin van de module, wordt aangesloten aan pin A4 van onze Arduino.
- SCL: Seriële clock-pin van de module, wordt aangesloten aan pin A5 van onze Arduino.



Figuur 4.17: PN532 NFC-module



Figuur 4.18: PN532 op het wagentje

4.2.5 Bluetooth-module

Een onderdeel van de opdracht bestond er uit om communicatie te voorzien tussen ons wagentje met een Arduino-microcontroller enerzijds, en een Raspberry Pi anderzijds. Om dit te realiseren kozen we er voor om Bluetooth te gebruiken. We maken gebruik van een HC05 Bluetooth-module om deze communicatie mogelijk te maken. Deze module ziet u in figuren 4.19 en 4.20. De HC05 beschikt over zes pinnen:

- EN: Dit is een enable voor de module die actief laag is, deze moet niet aangesloten worden aangezien we de module constant actief willen.
- VCC: Voeding van de module, aangesloten aan 5 V van onze Arduino.
- GND: Ground van de module, aangesloten aan de GND van onze Arduino.
- TXD: Transmit-pin van de module, aangesloten aan RX van de Arduino op pin D7
- RXD: Receive-pin van de module, aangesloten aan TX van de Arduino op pin D8
- STATE: Geeft informatie over de toestand van de module, gaat hoog wanneer deze verbonden is en laag wanneer dat niet het geval is. Deze pin wordt niet gebruikt voor onze toepassing



Figuur 4.19: HC05 Bluetooth-module



Figuur 4.20: HC05 Bluetooth-module op het wagentje

4.2.6 Pintoewijzing

We zagen in voorgaande secties het ontwerp van ons custom Arduino board en de bijkomende hardware. In tabel 4.1 ziet u een overzicht van de aansluitingen tussen deze hardware en de pinnen van het board.

Pin	Input/Output	Aansluiting	Pin	Input/Output	Aansluiting
D0	OUT	MUX Bit select 2	D10 (~)	/	/
D1	OUT	MUX Bit select 1	D11 (~)	/	/
D2	OUT	MUX Bit select 0	D12	/	/
D3 (~)	OUT	PWM Motor A	D13	/	/
D4	IN	Hall Sensor Input	A0	IN	IR-sensoren
D5 (~)	OUT	Richting Motor A	A1	/	/
D6 (~)	OUT	Richting Motor B	A2	/	/
D7	IN	Bluetooth RX	A3	/	/
D8	OUT	Bluetooth TX	A4	IN	RFID-lezer SDA
D9 (~)	OUT	PWM Motor B	A5	OUT	RFID-lezer SCL

Tabel 4.1: Pintoewijzing van custom Arduino

4.3 Raspberry Pi

Ten slotte hebben we nog de Raspberry Pi. We maken gebruik van een Raspberry Pi 3 zoals u ziet in figuur 4.21. Dit is in principe een computer waarop een Linux-distributie draait. We sluiten op de RPi3 via USB nog een muis en toetsenbord aan, de monitor wordt verbonden met een HDMI-kabel. We maken geen gebruik van de voorziene I/O-pinnen. Belangrijk is dat deze Raspberry Pi 3 een ingebouwde Bluetooth-adapter heeft en we dus geen bijkomende Bluetooth-dongle nodig hebben.



Figuur 4.21: Raspberry Pi 3

Hoofdstuk 5

Toelichting software

Dit hoofdstuk behandelt de werking van onze software waarmee we onze hardware aansturen. We bespreken op welke manier we met de Arduino de IR-sensoren inlezen en aan de hand hiervan bijsturen, hoe we de snelheid meten, hoe RFID-tags ingelezen worden en hoe de Bluetooth-communicatie met de Raspberry Pi werkt.

5.1 IR-sensoren inlezen

Voor het inlezen van de sensoren wordt gebruik gemaakt van één analoge pin, met deze pin lezen we de analoge waarden in functie van de uitgangsspanningen van alle acht sensoren in via een multiplexer. Op deze manier kan software-matig onderscheid gemaakt worden tussen wit of zwarte ondergrond. De multiplexer wordt aangestuurd aan de hand van drie digitale pinnen die dienst doen als bit select. Om de tijdsduur voor het inlezen van de infrarood-sensoren via de multiplexer te minimaliseren lezen we deze in aan de hand van Gray-code. De volgorde van inlezen wordt verduidelijkt in tabel 5.1. De ingelezen waarden worden opgeslagen in een array van acht integers. De waarden in de array worden vervolgens gedigitaliseerd aan de hand van een grenswaarde, hiervoor hebben we de waarde 300 gekozen. Indien de sensorwaarde onder deze grens ligt ziet de sensor een witte ondergrond en krijgt dit digitaal een waarde '1', als de waarde boven de grens ligt komt dit overeen met een zwarte ondergrond en een digitale waarde '0'.

#	Bit Select 2	Bit Select 1	Bit Select 0	MUX-pin	Sensor
0	0	0	0	13	Vooraan 2
1	0	0	1	14	Vooraan 3
3	0	1	1	12	Vooraan 1
2	0	1	0	15	Vooraan 4
6	1	1	0	2	Achteraan 3
7	1	1	1	7	Achteraan 2
5	1	0	1	5	Achteraan 1
4	1	0	0	7	Achteraan 4

Tabel 5.1: Inlezen van sensoren aan de hand van Grey-code

5.2 Rijden en PID-regeling

5.2.1 Bijsturen van de motoren

De snelheid van de motoren wordt geregeld aan de hand van Pulse Width Modulation. Softwaramtig gebeurt dit met 8 bits en kunnen we de snelheid dus schalen tussen 0 en 255. Voor het nemen van een bocht maken we gebruik van een factor F die we schalen tussen -100 en 100 . Deze factor drukt uit hoeveel % één van de wielen moet vertragen. Indien $F = -100$ bedraagt zal het wagentje volledig links draaien aangezien het linkerwiel niet meer aangestuurd wordt. Omgekeerd draait het wagentje volledig naar rechts als $F = 100$. Bij $F = 0$ blijven beide wielen aan dezelfde constante snelheid draaien.

5.2.2 Berekening van foutwaarde

Nu de sensoren ingelezen kunnen worden is het mogelijk om aan de hand hiervan de positie van het wagentje ten opzichte van de zijlijn te bepalen. Hiermee kan vervolgens het wagentje correct bijgestuurd worden om deze lijn te volgen. Om dit te realiseren wordt PID-regeling toegepast. Deze PID-regeling gebeurt aan de hand van een foutwaarde die afgeleid wordt uit de ingelezen sensorwaarden, aan elke sensor wordt dus een gewicht toegekend die een maat geeft voor de afwijking ten opzichte van de witte lijn. Deze gewichten vindt u in tabel 5.2.

	Vooraan 1	Vooraan 2	Vooraan 3	Vooraan 4
Foutwaarde	-3	-1	1	3
	Achteraan 1	Achteraan 2	Achteraan 3	Achteraan 4
Foutwaarde	3	1	-1	-3

Tabel 5.2: Gewichten van sensoren

De foutwaarde van de voorste sensoren wordt nu berekend met volgende formule:

$$E_{vooraan} = \frac{\sum_{i=1}^4 S_{vooraan,i} \cdot G_{vooraan,i}}{\sum_{i=1}^4 S_{vooraan,i}}$$

Analoog wordt de foutwaarde voor de achterste sensoren gegeven door:

$$E_{achteraan} = \frac{\sum_{i=1}^4 S_{achteraan,i} \cdot G_{achteraan,i}}{\sum_{i=1}^4 S_{achteraan,i}}$$

Hierin is S_i de digitale waarde in de array, zoals reeds vermeld is deze gelijk aan 1 indien sensor i een witte ondergrond ziet en 0 wanneer de ondergrond zwart is. G_i is het gewicht van de sensor in kwestie.

De totale foutwaarde E_{totaal} wordt dan bepaald door beide foutwaarden op te tellen:

$$E_{totaal} = E_{vooraan} + E_{achteraan}$$

Deze foutwaarde zal negatief zijn wanneer het wagentje teveel naar rechts afwijkt ten opzichte van de zijlijn, omgekeerd is deze fout positief als het wagentje te veel naar links rijdt. Als het wagentje perfect rechtdoor rijdt zullen de foutwaardes elkaar compenseren zodat de foutwaarde 0 wordt.

5.2.3 PID-regeling

Nu we een maat hebben voor de afwijking ten opzichte van de zijlijn kunnen we de PID-regeling toepassen. Hierbij wordt de factor tussen -100 en 100 bepaald die we gebruiken voor het bijsturen van de motoren, deze factor wordt als volgt bepaald:

$$F = K_P \cdot E_P + K_I \cdot E_I + K_D \cdot E_D$$

Hierin is:

- F = factor voor het bijsturen van de motoren
- $E_P = E_{totaal}$ = proportionele term fout
- $E_I = \sum E_{totaal}$ = integrerende term fout
- $E_D = E_{totaal} - E_{totaal,vorig}$ = differentiërende term fout
- K_P = constante waarde om de invloed van de proportionele term te schalen
- K_I = constante waarde om de invloed van de integrerende term te schalen
- K_D = constante waarde om de invloed van de differentiërende term te schalen

De constanten K_P, K_I en K_D bepalen in sterke mate de werking van de PID-regeling. Aangezien het afstellen van deze waarden praktisch gezien via een trial-and-error-proces verloopt worden deze waarden in de setup opgevraagd bij de Raspberry Pi Python Shell, hoe dit precies gebeurt wordt besproken in sectie 5.5 op de volgende pagina. We merken dat we voor de meeste circuits de beste resultaten halen met $K_P = 12$, $K_I = 1$ en $K_D = 25$

In het geval dat de voorste sensorarray teveel van de baan afwijkt wordt afgestapt van werkelijke PID-regeling en wordt er overgeschakeld naar een soort pseudo PID-regeling waarbij de foutwaarde aan de hand van de laatst bepaalde foutwaarde steeds geïncementeerd wordt. Wanneer de voorste sensorarray zich opnieuw boven de lijn bevindt hervat de normale PID-regeling terug.

5.3 Snelheid meten

In sectie 4.2.3 op pagina 17 bespraken we reeds op welke manier we twee magneten bevestigden in de wielas die passeren bij een SS41 Hall-sensor. Deze twee magneten zorgen door de tegengestelde polarisatie dat de output van de SS41 omschakelt van hoog (5 V) naar laag (0 V) en vice versa bij het passeren van één van de magneten. Binnen een tijdsinterval Δt tellen we het aantal keer C dat deze omschakeling optreedt. Het aantal rotaties in dit interval is dan de helft van het aantal keer dat de sensoroutput omgeschakeld is. Wetende dat de diameter D van het wiel 7 cm bedraagt kunnen we de snelheid v als volgt berekenen:

$$v = \frac{\Delta x}{\Delta t} = \frac{\frac{C}{2} \cdot \pi \cdot D}{\Delta t}$$

De snelheid van ons wagentje wordt voortdurend berekend over periodes van 10 s. Daarna wordt deze ook verzonden over Bluetooth.

5.4 RFID-tags inlezen

De software voor het inlezen van de RFID-tags maakt gebruik van libraries die te vinden zijn op de GitHub-pagina van Elechouse¹. Aangezien het inlezen van een tag over I²C enkele tientallen milliseconden kan duren wordt om de 100 ms gepolled of een nieuwe tag aanwezig is, op deze manier wordt het inlezen van andere sensoren en bijsturen van de motoren in mindere mate onderbroken. Indien er een nieuwe tag aanwezig is zal de ID van deze tag uitgelezen en opgeslagen worden aan de hand van de Elechouse-library. Vervolgens worden deze gegevens via Bluetooth naar onze Raspberry Pi verzonden, deze communicatie wordt besproken in volgend hoofdstuk.

5.5 Bluetooth-communicatie naar Raspberry Pi

5.5.1 Arduino met HC05-module als Slave

De HC05-module laat ons toe om via Bluetooth informatie over ons wagentje te verzenden naar onze Raspberry Pi. We zullen deze module gebruiken als Slave en de verbinding maken vanaf onze Raspberry Pi. Aangezien deze module seriële communicatie voorziet gebruiken we de SoftwareSerial-library van Arduino. We definiëren dus een Transmit- en Receive-pin en maken een SoftwareSerial-object aan waaraan we deze pinnen meegeven. In de setup starten we de seriële Bluetooth-communicatie op met een baudrate van 9600 bits per seconde. Vervolgens kunnen we in de rest van ons programma eenvoudigweg data printen over Bluetooth met *println*-instructies, of integers ontvangen over Bluetooth inlezen met *parseInt()*.

5.5.2 Raspberry Pi 3 met ingebouwde Bluetooth-adapter als Master

Aan de kant van de Raspberry Pi 3, die we als Master gebruiken, dienen we eerst correct verbinding te maken met de HC05-Bluetooth-module. Daarvoor voeren we volgend script uit:

```
#!/bin/bash

sudo bluetoothctl connect 98:D3:32:11:64:0C
sudo rfcomm bind /dev/rfcomm1 98:D3:32:11:64:0C
sudo rfcomm watch hci1
```

Listing 5.1: Script om verbinding te maken met HC05-module

Hierin is 98:D3:32:11:64:0C de hexadecimale voorstelling van het adres van onze Bluetooth-module. Eerst wordt verbinding gemaakt tussen de ingebouwde adapter van de RPi en de HC05. Vervolgens binden we deze verbinding aan rfcomm-poort 1 en monitoren we het verkeer op deze poort.

¹<https://github.com/elechouse/PN532>

Het verzenden en lezen van data wordt nu gedaan aan de hand van een Python-script waarin we gebruik maken van de Python-library serial. Het script leest voortdurend data in die ontvangen wordt van de Arduino. Indien deze data de gemeten snelheid of de gegevens van ingelezen RFID-tags bevat wordt deze op het scherm van onze Raspberry Pi afgeprint. In de setup van de Arduino wordt een token verzonden naar de RPi, als deze token ontvangen wordt kunnen we in de Python Shell de snelheid en de PID-constanten waarmee het wagentje moet vertrekken meegeven. Op deze manier hoeven we niet steeds te herprogrammeren om deze waarden aan te passen. Een voorbeeld van het resultaat in de Python Shell ziet u in figuur 5.1.

```

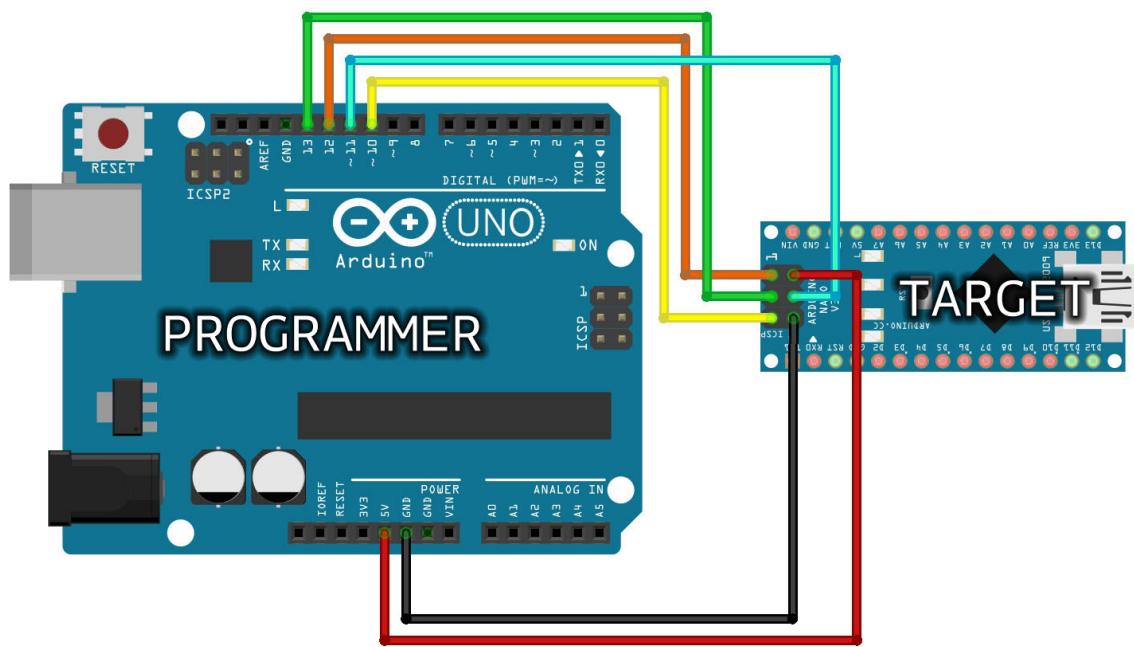
*Python 2.7.9 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
RFID-lezer gevonden: PN532
Firmware ver. 1.6
Wachten op snelheid...
Speed (0-255) = 75
Wachten op KP-waarde...
KP = 12
Wachten op KI-waarde...
KI = 0
Wachten op KD-waarde...
KD = 25
-----
We vertrekken met snelheid = 75; KP = 12; KI = 0 en KD = 25
RFID-tag gevonden!
UID Lengte: 7 bytes
UID waarde: 4 C9 4 82 31 4D 84
Aantal magneten gepasseerd: 12
0.13 m/s
Aantal magneten gepasseerd: 27
0.30 m/s
Aantal magneten gepasseerd: 38
0.42 m/s
Aantal magneten gepasseerd: 21
0.22 m/s
Aantal magneten gepasseerd: 26
0.29 m/s
RFID-tag gevonden!
UID Lengte: 7 bytes
UID waarde: 4 C9 4 82 31 4D 84
Aantal magneten gepasseerd: 20
0.22 m/s
Aantal magneten gepasseerd: 18
0.20 m/s
-----
Bluetooth-communicatie opgestart: setup uitvoeren...
RFID-lezer gevonden: PN532
Firmware ver. 1.6
Wachten op snelheid...
Speed (0-255) = |

```

Figuur 5.1: Voorbeeld van ingeven PID-constanten en ontvangen data in Python Shell

5.6 Custom board programmeren

In het ontwerp hebben we geen voorzieningen getroffen om de ATMega-chip te programmeren via USB-aansluiting, we voorzagen echter wel ICSP-pinnen. Om onze eigen Arduino te programmeren gebruiken we een andere Arduino Uno die dienst doet als programmer. Op deze Arduino uploaden we het ArduinolISP-programma uit de standaard Arduino-libraries. Vervolgens verbinden we ons te programmeren board met deze Arduino gebaseerd op de verbindingen in figuur 5.2 op de pagina hierna. Om de programmer-Arduino via USB-interface verbonden is aan de computer plaatsen we nog een condensator van $10\mu F$ tussen de reset en ground van de programmer. Vervolgens branden we de bootloader naar de ATMega-chip op ons eigen board. Onze custom board kan nu geprogrammeerd worden.



Figuur 5.2: Aansluitingen ICSP-pinnen om custom board te programmeren

Hoofdstuk 6

Problemen en moeilijkheden

Uiteraard zijn we tijdens het project op enkele problemen gestoten. Deze problemen en eventuele oplossingen ervan worden in dit hoofdstuk besproken.

6.1 Rijden aan de hand van PID-regeling

6.1.1 Positionering van de sensoren

Een correcte PID-regeling toepassen was praktisch nogal een trial-and-error proces dat veel tijd in beslag nam. Oorspronkelijk was het de bedoeling om de stippellijn in het midden van de baan te volgen. Dit bleek echter vrij moeilijk en onbetrouwbaar om toe te passen aangezien het wagentje in de bochten de stippellijn kan kwijtraken. Daarom probeerden we ook om met de 3D-geprinte armen aan beide zijden van het wagentje een sensor bij te hangen die de zijlijn kan detecteren zodat we daarop kunnen bijsturen, maar ook dit bleek niet betrouwbaar aangezien deze sensoren in de bochten soms de middellijn oppikten waardoor het wagentje helemaal van de baan geraakte. Uiteindelijk beslisten we om toch te sturen aan de hand van de buitenste volle lijnen, we probeerden eerst om met één sensor-array voor de linkerzijlijn, en eentje voor de rechterzijlijn, ook hier waren we niet tevreden met de betrouwbaarheid van de opstelling, in sommige bochten pikte één van deze arrays nog steeds de middellijn op waardoor het wagentje opnieuw de weg kwijt raakte. Uiteindelijk positioneerden we dus beide sensor-arrays op de linkerzijlijn. Het voordeel hiervan is dat we betrouwbaarder het parcours kunnen volgen, nadrukkelijk is wel dat we ook iets trager rijden aangezien we de buitenkant van de baan volgen om de betrouwbaarheid te verzekeren.

6.1.2 Bepalen van de PID-constanten

Het bepalen van de PID-constanten K_P, K_I en K_D nam ook veel tijd in beslag aangezien we deze waarden telkens opnieuw moesten uploaden naar onze Arduino. Deze frustratie hebben we uiteindelijk opgelost eenmaal de Bluetooth-communicatie voorzien was, met enkele wijzigingen in onze Arduino-code en Python-code op de Raspberry Pi geven we deze waarde vanaf de RPi in tijdens de setup van ons wagentje. Op deze manier verliep het testen toch vlotter.

6.2 Bluetooth-communicatie

Initieel was het de bedoeling om gebruik te maken van de Adafruit Bluefruit LE SPI Friend, een low-energy Bluetooth-module. Na verschillende pogingen zonder succes met deze module om data te verzenden en ontvangen van en naar de Raspberry Pi zijn we afgestapt van deze module. We kozen als alternatief de bekendere, en beter gedocumenteerde HC05-module. Jammer genoeg verbruikt deze module wel meer vermogen, maar aangezien het vermogenverbruik niet van cruciaal belang is voor ons project is dit niet zo dramatisch.

6.3 RFID-lezer

6.3.1 Libraries

Voor de communicatie tussen de Arduino en de PN532 NFC-module konden we kiezen voor drie verschillende communicatie-interfaces: HSU, I²C en SPI. Bij voorkeur gebruiken we voor deze communicatie SPI, het voordeel hiervan is dat dit over het algemeen veel sneller verloopt ten opzichte van I²C-communicatie. Bij het gebruik van SPI traden er echter problemen op met het detecteren van de PN532 aangezien de gedownloade bibliotheken verouderd waren. HSU gebruikt dan weer pinnen D0 en D1, die we tijdens het prototypen ook gebruikten om met behulp van de seriële monitor te debuggen, bijgevolg konden we deze communicatie-interface ook niet gebruiken. Uiteindelijk kozen we er dus toch voor om gebruik te maken van I²C, wat wel als voordeel heeft dat we minder pinnen hoeven te gebruiken.

6.3.2 Duur van het inlezen

De standaard voorbeeld-programma's die bij de libraries zaten lezen voortdurend data van RFID-tags in, aangezien we onze processor liefst nog voor andere dingen willen gebruiken was dit uiteraard niet ideaal. De beste oplossing hiervoor is werken met interrupts. Wanneer er dan een RFID-tag ingelezen wordt zal een interrupt service routine worden uitgevoerd waarin we de tag kunnen lezen en de data verzenden. Daarna wordt terug gegaan naar het hoofdprogramma. Deze methode was echter iets te omslachtig om aan de hand van deze libraries toe te passen. Daarom opteerden we voor de polling-methode, deze methode is ook niet echt efficiënt, maar zorgde toch voor goede resultaten.

6.4 Arduino PCB

Bij het maken van de custom Arduino is er een kleine opmerking bovengekomen waar we in het vervolg kunnen op letten. We hebben een te klein massavlak voorzien onder de ontkoppelcondensator van onze 5V-regulator. Hierdoor kan de condensator zijn warmte niet goed dissiperen en wordt hij zeer warm. Al bij al heeft dit geen echte problemen opgeleverd, maar hoe warmer de condensator, hoe korter de levensduur van de component. In het vervolg letten we dus op de plaatsing van zo'n componenten in het massa vlak.

Hoofdstuk 7

Evaluatie coach

Op het einde van de projectweek in Nieuwpoort kreeg iedere groep een coach toegewezen. Dit was op basis van enkele positieve en negatieve eigenschappen die we van elk groepslid moesten neerpennen. Het is de eerste keer dat dit gedaan werd voor het project. Er is ons nu ook gevraagd om een evaluatie van onze coach op te stellen.

Onze coach voor het project is Bert Cox. We hebben over het algemeen een positieve ervaring gehad met hem: we konden steeds bij onze coach terecht als we vragen hadden of als we ergens vast zaten in ons project. Zo zaten we als concreet voorbeeld een tijdje in de problemen met onze PID-regeling van ons voertuig en heeft Bert ons op de goede weg gezet door ons documentatie over PID-regeling voor te leggen. Vervolgens heeft hij ook enkele voorstellen gedaan omtrent de plaatsing van onze sensoren om de PID-regeling goed te laten werken. Tijdens de projectweek zelf heeft hij ook met ons samengezeten om enkele mogelijkheden te bespreken. Zo heeft hij het voorstel gedaan om te gaan werken met een multiplexer om zo analoge pinnen op onze Arduino uit te sparen. Het was ook op aanraden van onze coach dat we in onze planning zo snel mogelijk aan het ontwerp van de custom Arduino begonnen zijn.

Langs de andere kant hadden we graag een iets actievere opvolging van ons project gehad van onze coach. Sommige coaches kwamen iedere woensdag eens langs om te kijken hoe hun groep vorderde met hun project. Dit was bij ons minder van toepassing. Desalniettemin konden we wel steeds langs gaan met problemen en vragen, en was Bert ook bereid om feedback te geven over dit verslag.

Hoofdstuk 8

Besluit

In dit project ontwikkelden we een autonoom wagentje dat een raceparcours kan navigeren. We voorzagen ook een aantal andere functionaliteiten zoals het meten van de snelheid, het inlezen van RFID-tags en communiceren over Bluetooth. Tijdens het project hebben we meer ervaring op gedaan met het realiseren van een relatief ruime opdracht en leerden we veel bij over zowel hardware als software. Zo hebben we meer inzicht verworven in de werking van een microcontroller. Tevens leerden we zelf vereiste sensoren en andere hardware uitzoeken, al dan niet aan de hand van datasheets, en deze gebruiken om custom hardware te ontwikkelen voor onze eigen specifieke doeleinden. We raakten meer vertrouwd met de praktische werking van deze hardware en leerden hoe we deze software-matig kunnen laten samenwerken. Eveneens staken we meer op over veelgebruikte communicatie-interfaces zoals I²C, UART en SPI, en de voor- en nadelen eigen aan deze interfaces.

Kortom was dit project dus een aangename en effectieve manier om meer kennis te verwerven en praktische ervaring op te doen omtrent elektronica.

Na realisatie van al onze doelstellingen kunnen we concluderen dat een aantal punten toch anders en/of beter konden. Zo kon het inlezen van de IR-sensoren vlotter verlopen indien we de uitgangsspanningen van deze sensoren hardware-matig digitaliseerden. Ook bij de PID-regeling van het wagentje is er nog ruimte voor verbetering.

FACULTEIT INDUSTRIELE INGENIEURSWETENSCHAPPEN
TECHNOLOGIECAMPUS GENT
Gebroeders De Smetstraat 1
9000 GENT, België
tel. + 32 92 65 86 10
fax + 32 92 25 62 69
iiw.gent@kuleuven.be
www.iiw.kuleuven.be



LID VAN
**ASSOCIATIE
KU LEUVEN**