

Ontwikkeling van een self- driving voertuig

Jorik DE BRUYCKER
Jonas BOLLE

Begeleiders:
Leenders Guus
Crul Stijn
Naessens Carine
Van Der Perre Liesbeth

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
1.2 Doelstellingen	3
1.3 Structuur	3
2 Taakverdeling	4
3 Onkosten	5
4 Toelichting hardware	6
4.1 PCB-ontwerp	6
4.1.1 Eagle-ontwerp	6
4.1.2 Maken van de PCB	8
4.1.3 Pintoewijzing	11
4.2 Sensoren en toebehoren	11
4.2.1 Infrarood-sensoren	11
4.2.2 Multiplexers	13
4.2.3 Hall-sensor en magneten	14
4.2.4 RFID-reader	14
4.2.5 Bluetooth-module	15
5 Toelichting software	16
5.1 IR-sensoren inlezen	16
5.2 Rijden en PID-regeling	16
5.2.1 Bijsturen van de motoren	16
5.2.2 Berekening van foutwaarde	17
5.2.3 PID-regeling	18
5.3 Snelheid meten	18
5.4 RFID-tags inlezen	19
5.5 Bluetooth-communicatie naar Raspberry Pi	19
5.5.1 Arduino met HC05-module als Slave	19
5.5.2 Raspberry Pi 3 met ingebouwde Bluetooth-adapter als Master	19

5.6 Custom board progammeren	20
6 Problemen en moeilijkheden	21
6.1 RFID-lezer en libraries	21
7 Evaluatie coach	22
8 Besluit	23

Hoofdstuk 1

Inleiding

1.1 Opdrachtbeschrijving

Voor dit project ontwikkelen we een robot/voertuig dat autonoom zo snel mogelijk een raceparcours kan afleggen, dit parcours bestaat uit een zwarte ondergrond afgebakend door volle witte lijnen en een witte stippellijn in het midden. Het prototypen van ons voertuig voeren we uit met behulp van Arduino met een motor shield, tevens ontwikkelen we een custom board om deze Arduino in een later stadium van ons project te vervangen. We voorzien ons voertuig ook nog van enkele andere functionaliteiten. Zo zullen we een snelheidsmeter voorzien. Ten tweede zorgen we ook voor een RFID-reader waarmee ons voertuig de ID's van tags die verspreid over het parcours liggen kan inlezen. Ten slotte moeten we deze data ook allemaal draadloos kunnen verzenden, hiervoor voorzien we Bluetooth-communicatie van onze custom Arduino naar een Raspberry Pi 3.

1.2 Doelstellingen

- Voertuig autonoom laten het parcours afleggen
- Custom board ontwerpen om Arduino en motorshield te vervangen
- Voorzien van een snelheidsmeter
- Voorzien van een RFID-lezer om tags uit te lezen
- Bluetooth-communicatie tussen Arduino en Raspberry Pi 3 realiseren

1.3 Structuur

...

...

...

Hoofdstuk 2

Taakverdeling

PeriodeGroepslid		Bolle Jonas	De Bruycker Jorik
1/feb	11/feb	Voorbereidend opzoekwerk	
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

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

Tabel 3.1: Onkosten

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

Hoofdstuk 4

Toelichting hardware

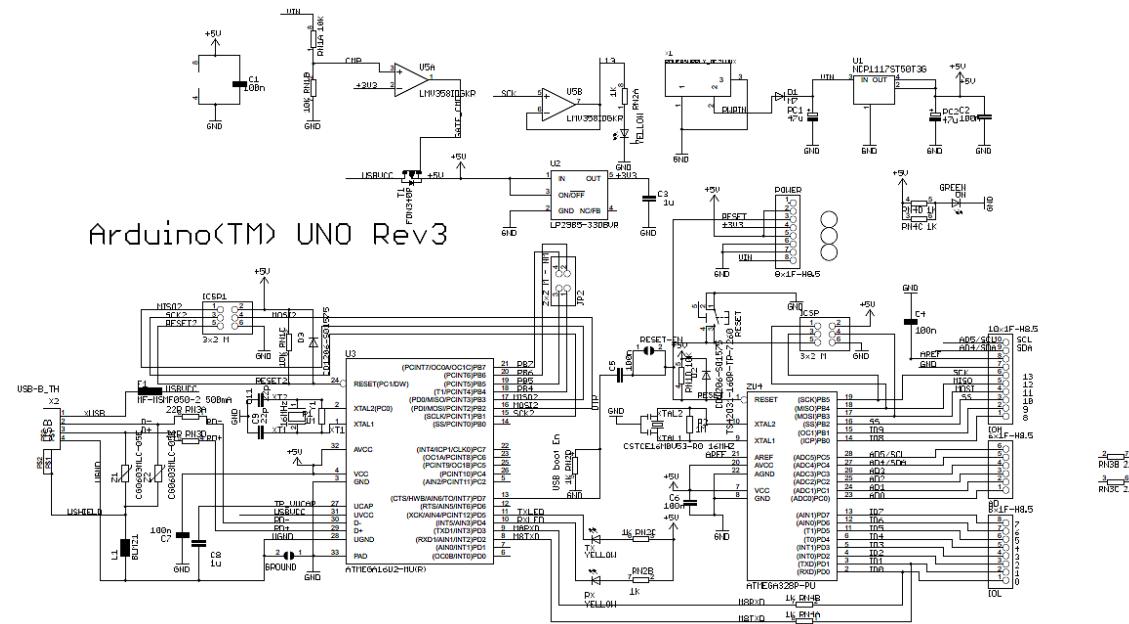
4.1 PCB-ontwerp

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.

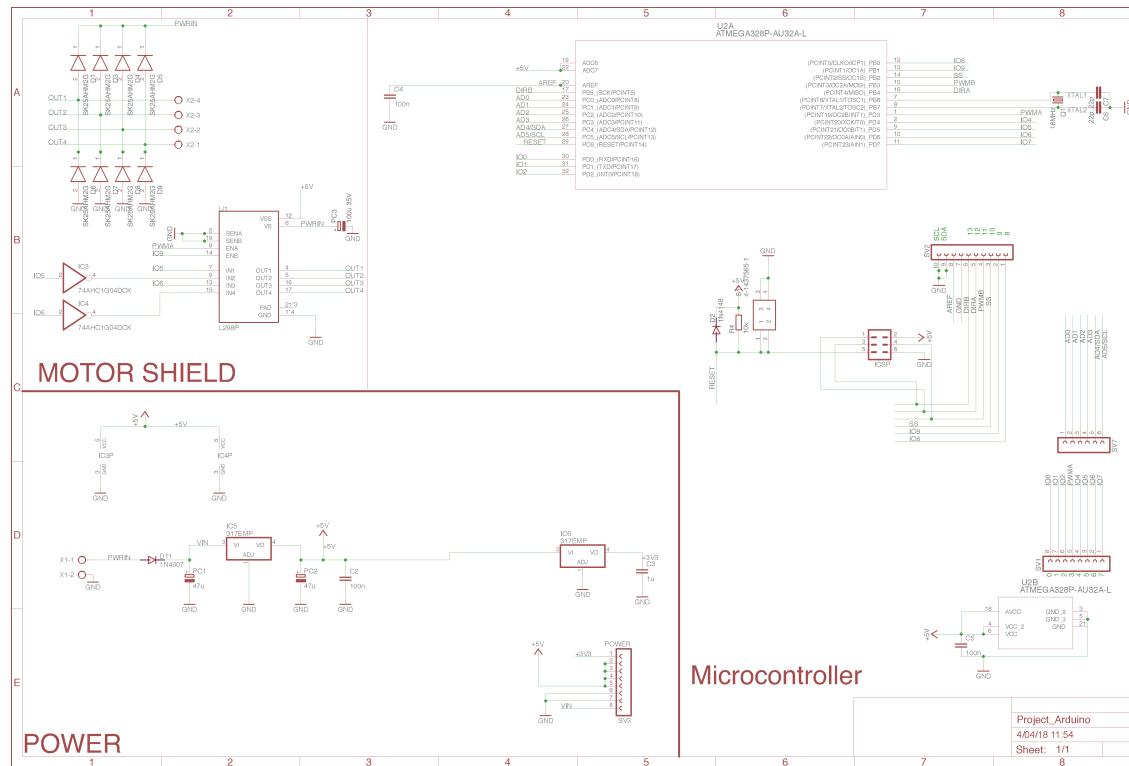
4.1.1 Eagle-ontwerp

Schematic

Om te beginnen hebben we het officiële schema nodig van de Arduino Uno, welke makkelijk online te vinden is (zie figuur 4.1 op de pagina hierna). Om te weten wat we effectief nodig hebben en wat weggelegd mag worden, bestuderen we het schema grondig. Hierdoor zien we in dat er veel weg kan vallen uit het originele ontwerp, zoals de USB to serial interface. Aangezien we beslissen 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 onze voedingslijn wordt hierdoor minder complex: we hoeven ons geen zorgen te maken om de batterijspanning af te koppelen eens we een USB-connectie zouden gebruiken van onze computer. Vervolgens zijn ook alle LED's uit het officiële ontwerp gehaald, zodat nog een IC (die twee opamps bevat), enkele weerstanden en een ontkoppelcondensator kunnen weggelegd 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 de volgende pagina) dat we vervolgens kunnen gaan boarden.



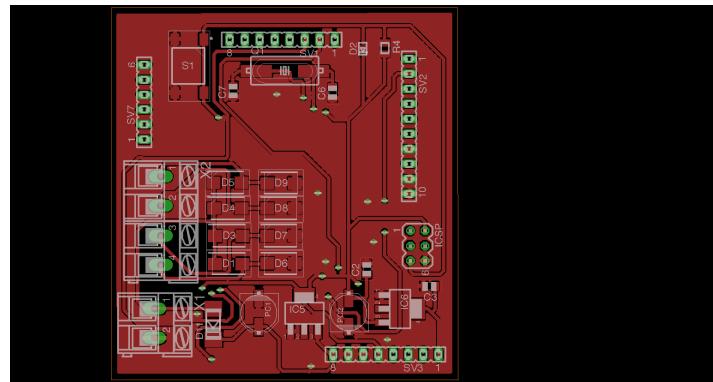
Figuur 4.1: Officiële schema van de Arduino Uno



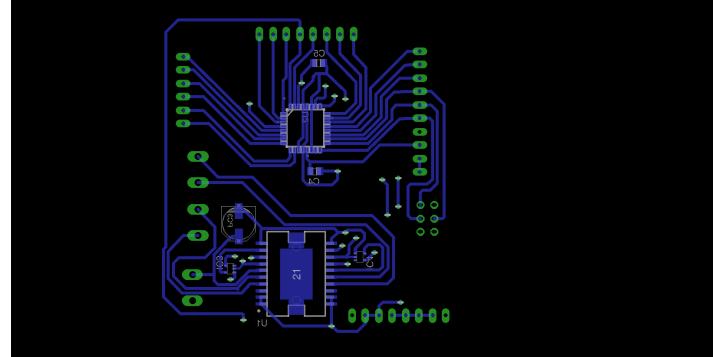
Figuur 4.2: Eigen schema van de Arduino Uno

Board

Nu onze eigen schematic afgerekend 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 massa vlak voorzien aan de bovenkant. De top- en bottom-layer van de PCB kan u vinden in respectievelijk figuur 4.3 en figuur 4.4.



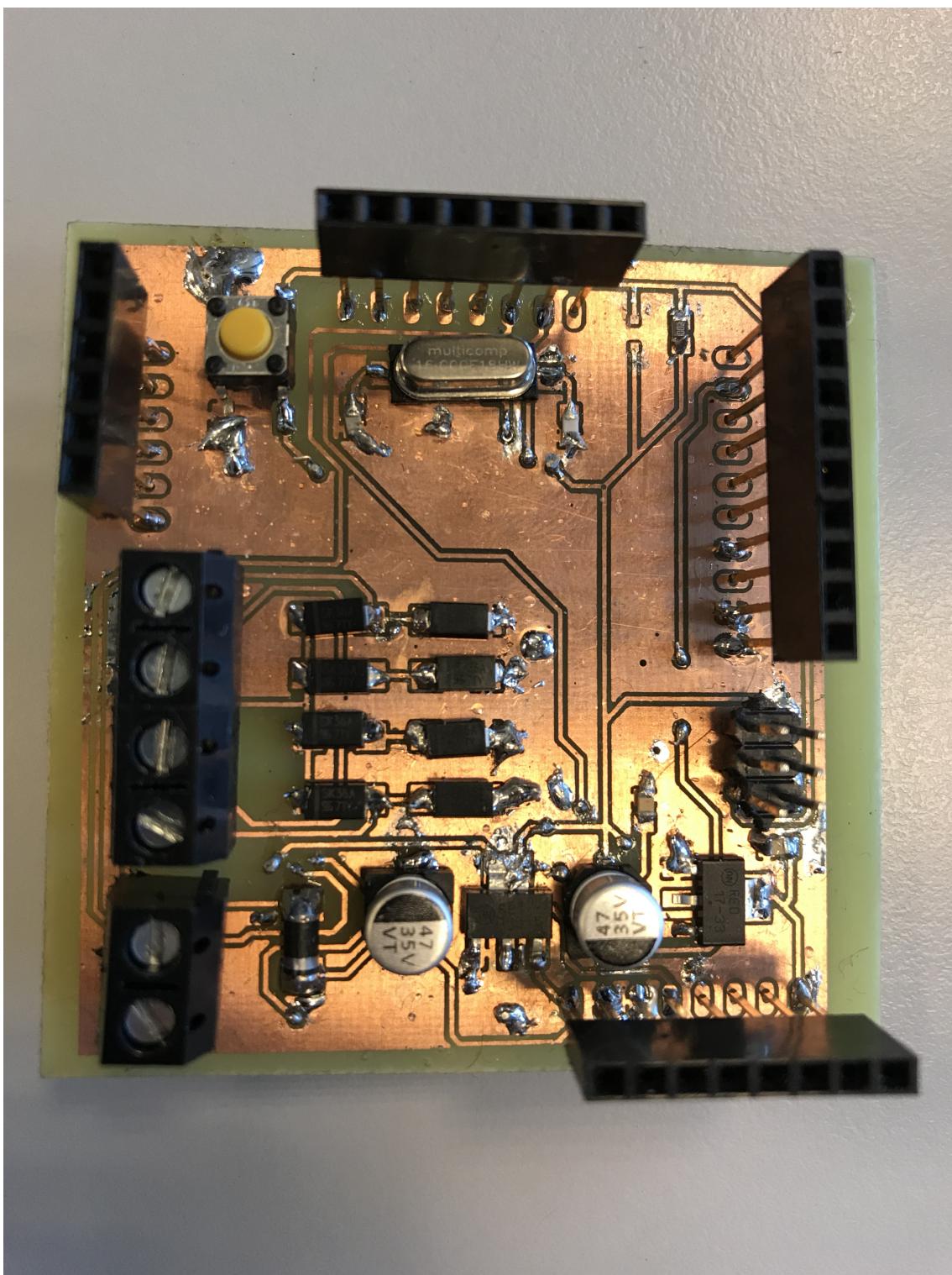
Figuur 4.3: Eigen board van de Arduino Uno



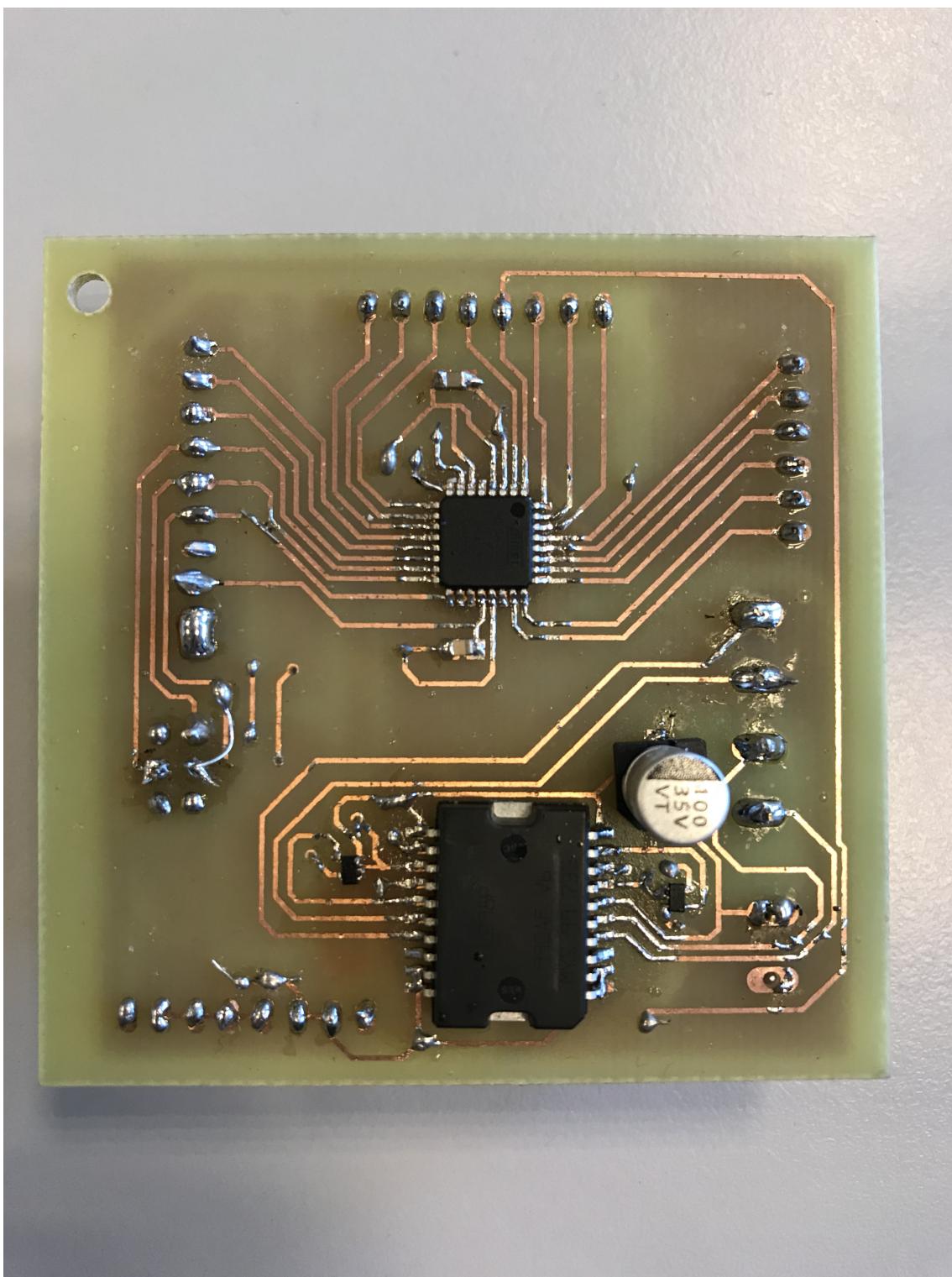
Figuur 4.4: Eigen board van de Arduino Uno

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 ?? op pagina ?? is de bovenkant en figuur ?? op pagina ?? de onderkant.



Figuur 4.5: Bovenkant van onze custom Arduino



Figuur 4.6: Onderkant van onze custom Arduino

4.1.3 Pintoewijzing

In tabel 4.1 ziet u een overzicht van de hardware die aangesloten op onze custom Arduino. Deze hardware wordt nog verder besproken in dit hoofdstuk.

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

4.2 Sensoren en toebehoren

4.2.1 Infrarood-sensoren

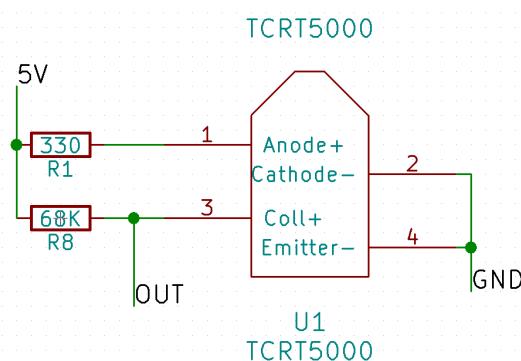
Zoals we reeds vermeldden is het de bedoeling dat ons voertuig een parcours autonoom kan afleggen dat afgebakend wordt door twee volle witte lijnen op een zwarte ondergrond met een witte stippelijn tussen beide. Om dergelijk parcours te kunnen navigeren zullen we onderscheid moeten kunnen maken tussen de witte en zwarte ondergrond. Voor dergelijke toepassing kiezen we voor infrarood-sensoren. We zullen hiervoor gebruik maken van de TCRT-5000-sensor, zoals u ziet in figuur 4.7.



Figuur 4.7: TCRT5000 infrarood sensor

Sensor-cel

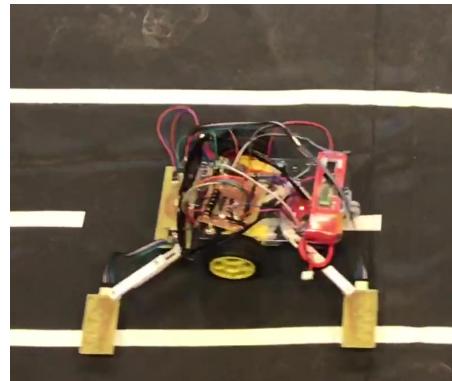
De TCRT5000 bestaat uit 2 belangrijke onderdelen: ten eerste hebben we een IR-led , deze led wordt aangestuurd met een voorschakelweerstand van 330Ω . Ten tweede heeft de TCRT5000 een transistor die zich in essentie als infrarood-gevoelige weerstand gedraagt, samen met een serieweerstand van $68\text{k}\Omega$ vormt dit een spanningsdeler. De IR-led zal infraroodgolven uitstralen, naargelang de ondergrond waar dit op invalt worden deze golven quasi volledig of amper gereflekted. 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 4.8 geeft dus een maat voor de reflectiecoëfficient 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.



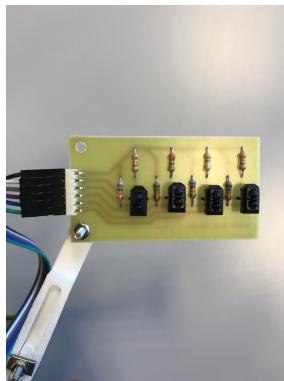
Figuur 4.8: Sensor-cel met TCRT 5000

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. De positionering van deze sensoren ziet u in figuur 4.9 op de pagina hierna. In figuren 4.10 op de volgende pagina en 4.11 op de pagina hierna ziet u de twee verschillende sensorarrays.



Figuur 4.9: Sensorarray op het wagentje ten opzichte van de zijlijn



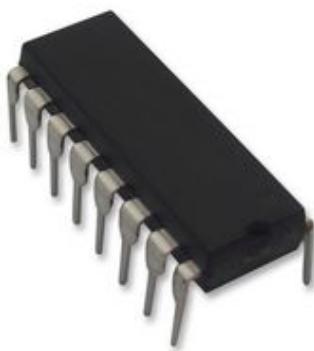
Figuur 4.10: Sensorarray linksvooraan



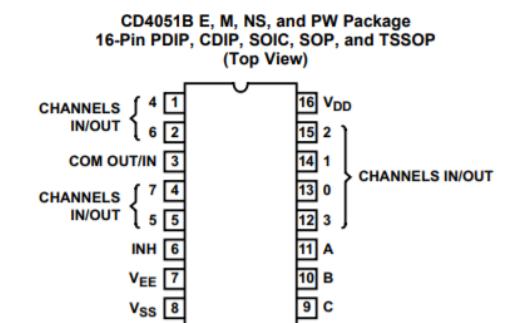
Figuur 4.11: Sensorarray linksachteraan

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 in figuur 4.12, de pinout van deze IC vindt u in figuur 4.13. We gebruiken deze multiplexer om acht sensoren in te kunnen lezen op één analoge pin: A0, hiervoor gebruiken we drie digitale pinnen D0,D1 en D2, die de bit selects van de multiplexer aansturen.



Figuur 4.12: CD4051BE multiplexer



Figuur 4.13: CD4051BE multiplexer pinout

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 4.14 ziet u de magneten die binnen in de wielas gemonteerd werden, merk op dat deze magneten een tegengestelde polarisatie hebben. In figuur 4.15 ziet u hoe de Hall-sensor bij het wiel gemonteerd is.



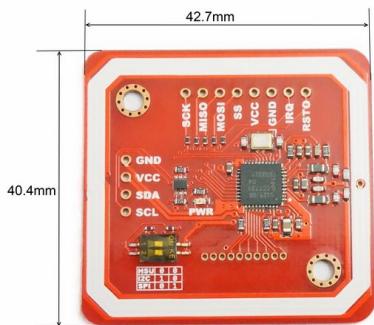
Figuur 4.14: Magneten gemonteerd in wielas

Figuur 4.15: Hall-sensor gemonteerd naast wiel

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 4.16 en 4.17. 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 21. 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 4.16: PN532 NFC-module

Figuur 4.17: PN532 op het wagentje

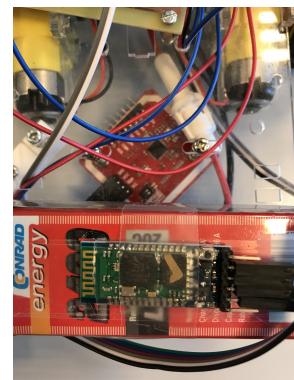
4.2.5 Bluetooth-module

Om Bluetooth-communicatie met een Raspberry Pi3 mogelijk te maken wordt gebruik gemaakt van de HC05 Bluetooth-module. Deze module ziet u in figuren 4.18 en 4.19. 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 4.18: HC05 Bluetooth-module



Figuur 4.19: HC05 Bluetooth-module op het wagentje

Hoofdstuk 5

Toelichting software

5.1 IR-sensoren inlezen

Voor het inlezen van de sensoren wordt gebruik gemaakt van één analoge pin, met deze pin lezen we de waarden van alle acht sensoren in via een multiplexer. Deze 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. Softwiermatig 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_{total} wordt dan bepaald door beide foutwaarden op te tellen:

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

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 foutwaarden 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 pagina hierna. 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 geïncrementeerd 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 14 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 omschakelde. 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:

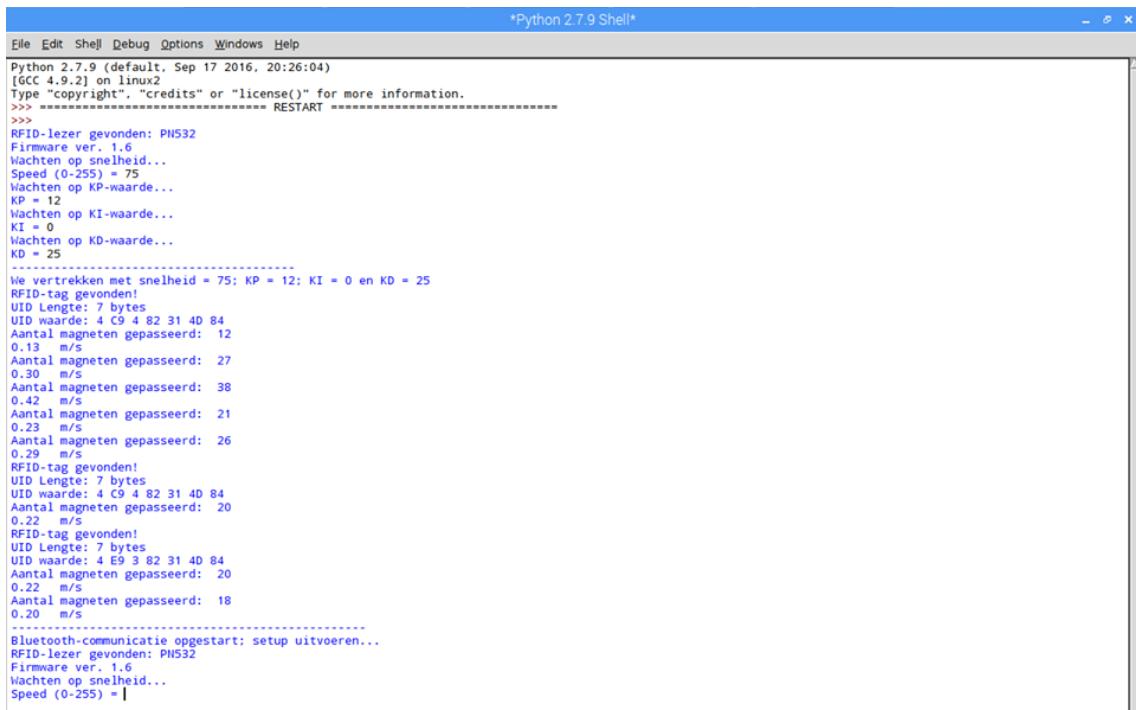
```
#!/bin/bash

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. 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 op de pagina hierna.

¹<https://github.com/elechouse/PN532>



The screenshot shows a Python 2.7.9 Shell window with the title "*Python 2.7.9 Shell*". The window contains the following text:

```
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.
>>> =====
>>> 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 E9 3 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 progammeren

Hoofdstuk 6

Problemen en moeilijkheden

6.1 RFID-lezer en 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, bijgevolgd 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.

Hoofdstuk 7

Evaluatie coach

...

...

...

Hoofdstuk 8

Besluit

...
...
...

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**