Computer Vision

De link tussen VR en de fysieke wereld

Bachelorproef voorgedragen tot het behalen van
de graad van Bachelor in de elektromechanica

Afstudeerrichting: automatisering

Promotor: Mathias Vermeulen
Tweede beoordelaar: Lode Poelman

Academiejaar 2018-2019

Pieter-Jan Cassiman
Jochen Sarrazyn

HO
GENT

Deze bachelorproef/scriptie is gemaakt door Cassiman Pieter-Jan en Sarrazyn Jochen, student aan de Hogeschool Gent, ter voltooiing van de bacheloropleiding, bachelor in de elektromechanica. De standpunten die in deze bachelorproef zijn verwoord, zijn louter het persoonlijke standpunt van de individuele auteur en reflecteren niet noodzakelijkerwijs de mening, het officiële standpunt of het beleid van de Hogeschool Gent.

Woord vooraf

Hoe we tot deze stage en bachelorproef zijn gekomen is een ietwat lang verhaal.

Nog voordat er sprake was van een stage, hadden we het idee om een shield te maken voor Arduino. Hiermee wouden we het gemakkelijker en toegankelijker maken om Arduino te leren, door de meest gebruikte ingangen, uitgangen en sensoren op dit shield te plaatsen. Voor wat feedback op ons idee stapten we naar Mr. Sierens, aangezien zijn vak het dichts aanleunde bij ons idee, en doelpubliek.

Bij het uitwisselen waren er ook andere docenten aanwezig, waaronder Mr. Lievens. Na enkele ideeën uitgewisseld te hebben kwam het naar boven dat er een mogelijkheid was om dit uit te werken als onze stage.

Dit bleef hangen in ons achterhoofd en als het tijd werd voor onze stage zijn we er dan ook op ingegaan om onze stage op school te doen. Ondertussen was het idee voor het shield al heel wat verder geëvolueerd. We waren erop uitgekomen om een autootje te maken met verschillende mogelijkheden. Dit had de mogelijkheid om vele aspecten aan te leren, maar ook leuk te blijven voor de studenten.

Om deze reden willen we als eerste Mr. Sierens bedanken voor alle hulp, feedback en begeleiding tijdens de stage, maar ook nog ervoor. Hij stelde zijn lokaal open, en het materiaal ter beschikking, waarvoor we zeer dankbaar zijn. En zeker om een heel weekend met ons op de Maker Faire te staan.

We willen Mr. Vermeulen bedanken voor de begeleiding en hulp tijdens de stage.

Verder willen we ook HoGent bedanken voor de opleiding in de mogelijkheid om ons idee uit te werken als onze stage. We hopen dat we op deze manier ook iets kunnen teruggeven aan de hogeschool en de opleiding.

Ook hadden we tijdens onze stage verschillende docenten gecontacteerd, voor ideeën en feedback. We willen al deze docenten bedanken voor hun ideeën en inspiratie.

Als laatste willen we ook de crew en de organisatie van de mini Maker Faire Gent 2019 bedanken, voor het leuk evenement en weekend.

Inhoudsopgave

Samenvatting

Mr. Sierens heeft plannen om een virtual-reality systeem te gebruiken in zijn (praktijk-)lessen, zodat de studenten meer voeling krijgen met hun opleiding en het werkveld. Wij hebben gewerkt aan een link tussen de fysieke wereld en dit virtual-reality systeem. Voor deze link is een positioneringssysteem nodig. Echter bleek zeer snel dat de huidige opties enerzijds niet nauwkeurig genoeg waren (zoals GPS) en anderzijds enorm duur (zoals RF).

We kregen het idee om een webcam, met bijhorend programma, te gebruiken als positioneringssysteem. De ontwikkeling van het programma en bijhorende hardware is dan ook de kern en opzet van deze bachelorproef.

Om aan te tonen dat een computer-vision-systeem gebruikt kan worden voor het nauwkeurig positioneren van objecten op een beperkte oppervlakte werd een ***proof-of-concept*** uitgedacht. Hierbij worden autootjes langsheen een parcours gevolgd en de rondetijden opgemeten. Eén van de autootjes rijdt volledig autonoom, als linefollower. Het andere autootje wordt bestuurd door een persoon.

Dit ***proof-of-concept*** werd uitgewerkt als een spel, aangezien het ging voorgesteld worden op de Maker Faire te Gent. Hierdoor konden alle aspecten van de ***proof-of-concept*** gemakkelijk getest en voorgesteld worden.

Er werd gebruik gemaakt van Python voor het computer-vision-systeem, via de OpenCV bibliotheek. De autootjes werden geprogrammeerd met Arduino.

Een computer-vision-systeem blijkt een zeer nauwkeurige en betrouwbare manier voor het positioneren van objecten. De totale oppervlakte waarop het vision systeem getest werd, was een A0 blad. Door de modulaire opbouw van het programma is het ook zeer gemakkelijk uit te breiden en te implementeren in andere projecten.

Achteraf werd het volledige programma en alle bijhorende ontwerpen uitgebracht als open-source. Het volledige project is terug te vinden op: https://github.com/pcassima/thesis_hogent

1. Inleiding

In het kader van de verbetering van het onderwijs, en meer specifiek het praktijkonderwijs, is mr. Sierens momenteel bezig met het ontwikkelen van een virtual reality systeem. Dit zou hem toelaten om de opstellingen, die de studenten moeten programmeren, volledig te virtualiseren.

De studenten leren hierdoor werken met de echte machines en kunnen veel sneller inzien hoe ze bepaalde problemen moeten aanpakken waardoor er kan afgestapt worden van klassieke opgaves, waar er enkele LEDs en knoppen moeten geprogrammeerd worden. Nu kunnen ze werken met een realistischer beeld waardoor de studenten meer voeling krijgen met het daadwerkelijke werkveld.

Een bijkomend voordeel van een volledige virtuele opstelling, is het feit dat er geen enkel gevaar aan gekoppeld is. De student kan de opstelling volledig verkeerd programmeren en kapot maken. Door de simulatie te resetten is alles opgelost. Ook is er geen nood meer aan grote investeringen om de verschillende opstellingen aan te kopen, te plaatsen en te onderhouden.

Echter hebben de studenten nog steeds nood aan een soort van fysieke feedback. Er is nood aan een schaalmodel van de opstelling of een fysieke weergave. Om nauwkeurig objecten te kunnen overbrengen van de fysieke wereld naar virtual-reality, is het nodig om constant te weten waar deze objecten zich bevinden.

Hiervoor zijn verschillende manieren en technieken. De meeste zijn industriële technieken, met een industrieel prijskaartje. Voor de oppervlakte van een doorsnee tafel in een klaslokaal zijn de meeste van deze systemen ofwel niet nauwkeurig genoeg, zoals GPS, ofwel veel te nauwkeurig, wat de prijs dan ook navenant maakt.

Uiteindelijk werd geopteerd voor een computer vision systeem. Dit is snel en goedkoop te implementeren en kan zeer snel en gemakkelijk uitgebreid worden.

Er werd een ***proof-of-concept*** uitgewerkt om aan te tonen dat een computer-vision systeem wel degelijk bruikbaar is voor deze toepassing. Voor deze ***proof-of-concept*** werden autootjes langsheen een parcours gevolgd en de rondetijden opgemeten. Dit liet toe om alle mogelijkheden van een dergelijk systeem te testen en aan te tonen.

De autootjes zelf werden gestuurd via MQTT. Mr. Sierens wou dit protocol gebruiken voor het communiceren met de VR. Door dit te gebruiken kon dit ook getest en geïmplementeerd worden. Voor het gebruiksgemak en duidelijkheid werd ook een protocol uitgewerkt bovenop MQTT.

2. Opzet

Zoals eerder vermeld willen we met deze ***proof-of-concept*** aantonen dat een computer-vision systeem bruikbaar is om objecten te positioneren op een beperkte oppervlakte.

De ***proof-of-concept*** werd uitgewerkt met als einddoel de Maker Faire te Gent.

Om dit aan te tonen werd een spel uitgedacht, dit was ook met de Maker Faire in het achterhoofd. Het spel hield in dat er een linefollower autonoom een zwarte lijn volgt. Deze linefollower werkt en rijdt volledig autonoom, hij staat dus volledig los van het vision systeem. De linefollower wordt gevolgd door de camera met behulp van een rode driehoek. De rondetijden worden gemeten en bijgehouden.

De bezoekers kunnen dan een tweede autootje besturen en zelf proberen om zo goed mogelijk de lijn te volgen en de rondetijden te verbeteren. Het tweede autootje werd op dezelfde manier gevolgd als de linefollower.

Door het project voor te stellen als een spel was er veel interesse vanuit het publiek. Ook was er niet alleen iets te zien, maar ook iets te doen. Door deze hands-on aanpak konden alle aspecten van het systeem zeer gemakkelijk overgebracht worden.

Op deze manier konden we ook een tweede aspect van de opleiding ten toon stellen: de linefollower, het syntheseproject van de opleiding automatisering.

2.1. Opstelling

Voor het spel was er een spelbord en opstelling nodig. Er werd een opvouwbaar spelbord gemaakt. Het spelbord op zich was te groot om door een deur te passen en zeer onhandig om te dragen. Daarom werd een scharnier voorzien in het midden van het spelbord, zodat de hele opstelling kon gedragen worden als een koffer.

Het spelbord spel bestond uit een MDF-plaat met allemaal gaatjes in. Oorspronkelijk was het de bedoeling om te werken met ge-3D-printe muurtjes. Op deze manier kon zelf een parcours gemaakt worden waar de autootjes moeten doorrijden. Hier zijn we niet aan toe geraakt.

Verder had de opstelling een frame, dat vergeleken kan worden met de constructie voor een dak. Dit was zodat we de webcam voldoende hoog konden monteren. Op deze constructie waren ook twee LED-lampen gemonteerd, zodat de invloeden van omgevingslicht verkleind konden worden.

De volledige constructie kan gedemonteerd worden en opgeslagen worden in de ruimte tussen de twee helften van het spelbord. De volledige opstelling kon dus opgeslagen en getransporteerd worden als een koffer.

De gehele constructie was niet geheel professioneel gebouwd. Deels door het gebrek aan tijd. Echter geeft dit een zeer toegankelijke indruk. Dit was ook een deel van deze ***proof-of-concept***: aantonen dat het mogelijk is, en aantonen dat het mogelijk is met alledaagse middelen, die ofwel reeds aanwezig zijn ofwel zeer gemakkelijk (en goedkoop) te vinden zijn.

3. Vision

> *"For both biological systems and machines, vision begins with a large and unwieldy array of measurements of the amount of light reflected from surfaces in the environment. The goal of vision is to recover physical properties of objects in the scene, such as the location of object boundaries and the structure, color and texture of object surfaces, from the two-dimensional image that is projected onto the eye or camera. This goal is not achieved in a single step; vision proceeds in stages, with each stage producing increasingly more useful descriptions of the image and then the scene. The first clues about the physical properties of the scene are provided by the changes of intensity in the image." (Hildreth, 1985)*
>
> *Voor zowel biologische systemen als machines begint het zicht met een onoverzichtelijke en grote reeks metingen van de hoeveelheid licht die wordt gereflecteerd door oppervlakken in de omgeving. Het doel van zicht is om fysieke eigenschappen van objecten in de scène te achterhalen, zoals de locatie van objectgrenzen en de structuur, kleur en textuur van objectoppervlakken, van het tweedimensionale beeld dat op het oog of de camera wordt geprojecteerd. Dit doel wordt niet in één stap bereikt; zicht verloopt in fasen, waarbij elke fase steeds nuttigere beschrijvingen van het beeld en vervolgens de scène produceert. De eerste aanwijzingen over de fysieke eigenschappen van de scène worden geleverd door de intensiteitsveranderingen in de afbeelding.*

Met dit citaat wordt een artikel van de AI-afdeling van MIT geopend (***Artificial Laboratory of the Massachusetts Institute of Technology***). Dit citaat geeft zeer goed weer waar het om draait bij computer-vision en is dus perfect om mee te openen.

Het artikel, waar dit citaat uit komt, gaat over de verschillende manieren die bestaan voor het detecteren van randen in afbeelding en foto's. Het artikel dateert van 1985, en geeft dus zeer goed de origine van de hedendaags gebruikte methodes in computer-vision weer. Verder wordt er verwezen naar computer-vision als CV.

3.1. Camera

Voor elk CV-systeem is uiteraard een camera nodig. Die het centrale element vormt van elk vision-systeem.

De gebruikte bibliotheek zet afbeelding om naar een multidimensionale matrix. Op deze matrix kunnen dan alle verdere bewerkingen uitgevoerd worden. De bibliotheek zelf kan ook een hele reeks camera's uitlezen, zowel voor video als foto's. In de meeste gevallen kan dit zelfs zonder drivers te moeten installeren.

Door het stuk code aan te passen dat de webcam start, kunnen verschillende camera's gebruikt worden. Voor de rest van het programma verandert er niets. Hier wordt verder op ingegaan in **Frame inlezen** pagina **31**.

3.1.1. USB-Camera

Zo goed als elke (digitale) camera kan gebruikt worden voor een CV-systeem, zolang er een manier is om de beelden om te zetten naar een reeks van bytes die door de computer verwerkt kunnen worden.

Het CV- systeem werd ontwikkeld en getest met een Logitech USB-webcam op een laptop, dit kan gemakkelijk overgezet worden naar de Raspberry Pi. Na het downloaden en installeren van de benodigde bibliotheken kan het programma zonder problemen overgezet worden naar de Raspberry Pi. Op de benodigde bibliotheken wordt verder teruggekomen.

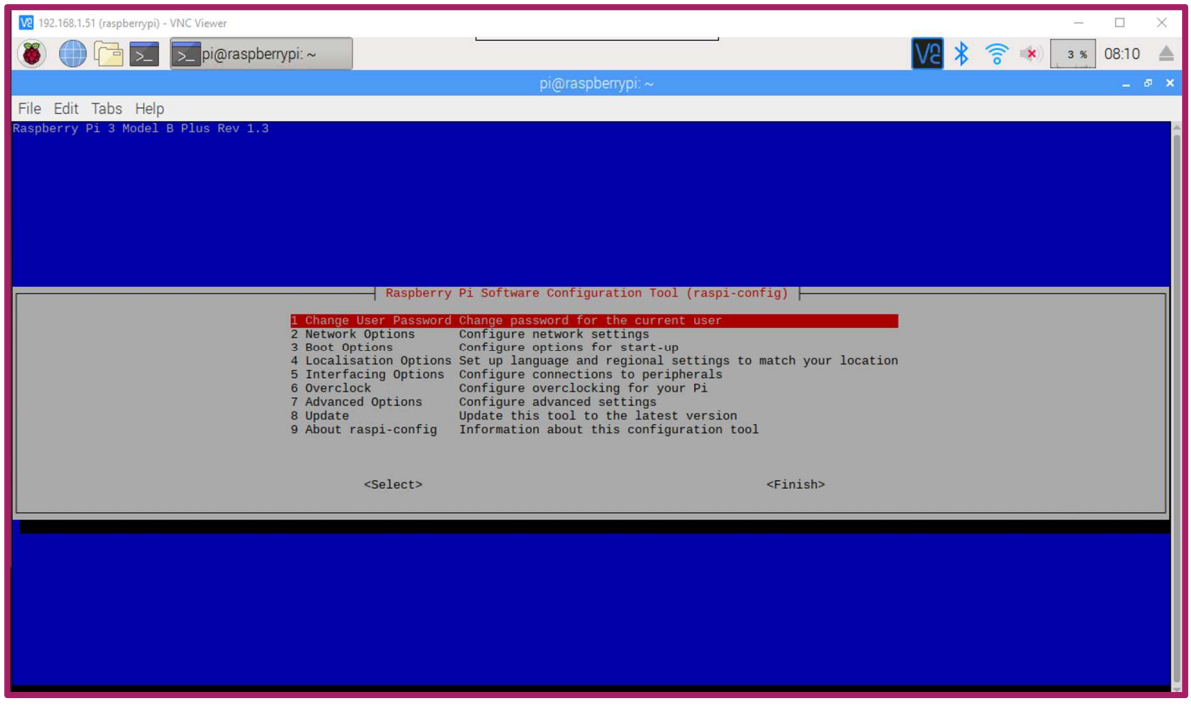
3.1.2. Raspberry Pi camera module

Voor het gebruik van de Raspberry Pi camera module, moet via het configuratiescherm de camera-interface aangezet worden. Het configuratiescherm kan opgeroepen worden via volgend commando:

```
sudo raspi-config
```

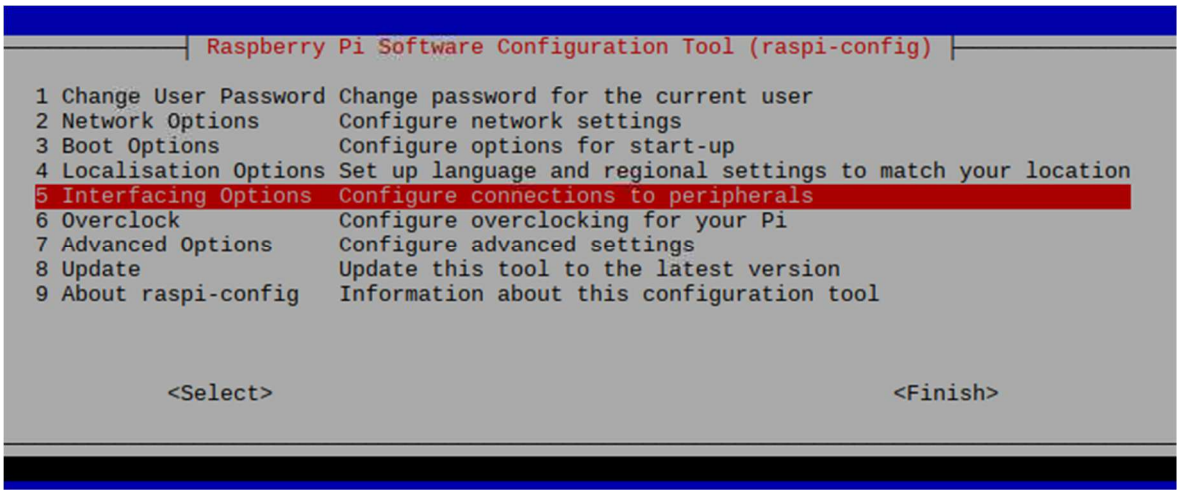
Code 3.1: Commando voor het oproepen van Raspberry Pi configuratiescherm

Na het uitvoeren van dit commando wordt volgend scherm verkregen:



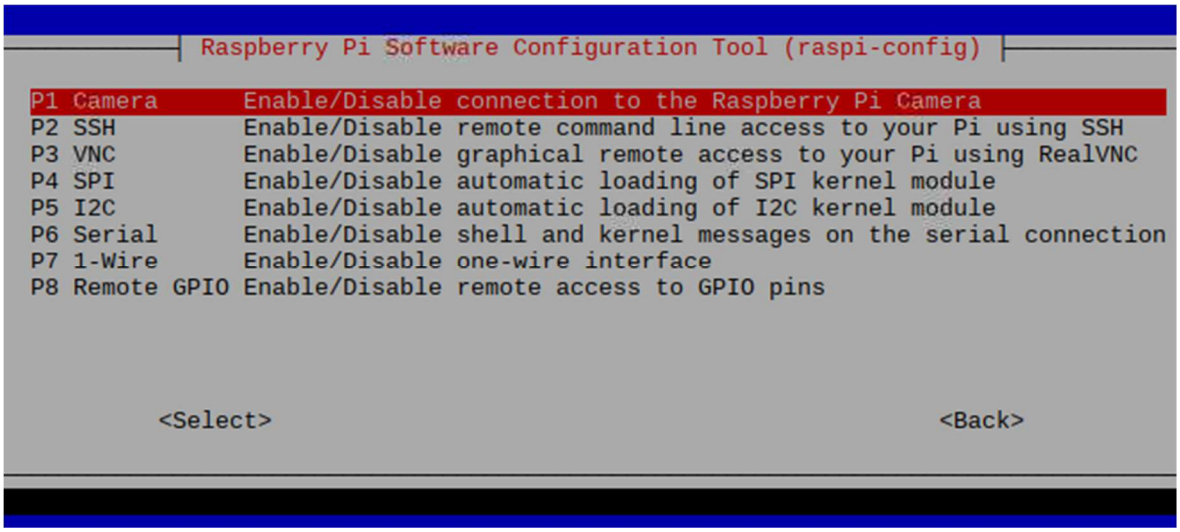
Figuur 3.1: Configuratiescherm Raspberry Pi

Onder de ***interfacing-options*** (nummer 5) kan de camera interface aangezet worden.



Figuur 3.2: Interfacing optie in het configuratiescherm

De optie om de camera interface aan te zetten is de eerste optie onder ***interfacing options***.



Figuur 3.3: Camera optie in het configuratiescherm

Onder dit menu kunnen ook andere instellingen en interfaces geactiveerd worden. Voor dit project werd onder meer ***SSH*** en ***VNC*** aangezet. Onder het hoofdstuk “extra” is een stuk terug te vinden over hoe ***SSH*** kan aangezet worden nog voordat de Raspberry Pi opgestart wordt (evenals op voorhand een Wi-Fi netwerk instellen). Dit stuk is te vinden onder **Raspbian - headless** op pagina **88**.

Eveneens werd de optie ***expand file system***, onder ***advanced options*** geactiveerd. Hierdoor zal Raspbian de volledige geheugenkaart innemen. Dit is zeer handig aangezien de installatie van de benodigde bibliotheken redelijk wat ruimte vraagt.

3.2. Systeem

Voor het programmeren van het CV-systeem werd Python gebruikt. Hierdoor kan het vision-systeem gestart worden en draaien op elk systeem waarop Python geïnstalleerd kan worden. Kortom zo goed als elk systeem.

Echter moet dit genuanceerd worden. Elk programma dat beeldherkenning of beeldverwerking doet, is een zwaar programma. Bijgevolg zijn er wel enkele minimumvereisten voor het systeem:

Specificatie	Vereiste
RAM	Minstens 1 GB (2 GB+ aangeraden)
CPU-kernen	Minstens 2
CPU-snelheid	Minstens 1.2 GHz
Opslag	Minstens 8 GB

Tabel 3.1: Minimumvereisten voor het vision systeem

De Raspberry Pi 3B valt nog net binnen deze vereisten. Het deel van het programma dat verantwoordelijk is voor het vision-systeem werd getest op een Raspberry Pi 3B. Het programma draaide en het vision-systeem werkte, maar de Raspberry Pi werd zeer warm. Doordat de visualisatie een beetje te wensen overliet, werd er verder gewerkt met een laptop.

Het programma draaien op een moderne desktop of laptop zou geen enkel probleem mogen vormen, aangezien deze meer dan genoeg middelen hebben. De laptop waarop het programma ontwikkeld werd was een laptop met 8 GB RAM en een Intel dual-core i5.

3.3. Framework

3.3.1. Programmeertaal

> *"Python is powerful... and fast;*
> *plays well with others;*
> *runs everywhere;*
> *is friendly & easy to learn;*
> *is Open." (The Python Software Foundation, sd)*

Python is een enorm populaire programmeertaal, zeker in de wetenschappelijke wereld. Bijgevolg zijn er enorm veel modules en uitbreiding beschikbaar voor Python. Voor bijna elke toepassing is er minstens één module beschikbaar voor Python.

Deze taal werd gekozen om verschillende redenen. De grootste reden was de uitgebreide community en documentatie rond Python. Verder woog onze uitgebreide voorkennis van Python ook zwaar door. Python programmeert en leest zeer gemakkelijk. Hierdoor kunnen derden achteraf ook zeer gemakkelijk wijs geraken uit de code en hun eigen aanpassingen aanbrengen.

Door de object-georiënteerde aanpak van Python kan de code ook zeer gemakkelijk opgesplitst worden in verschillende delen die elk verantwoordelijk zijn voor een klein onderdeel van het geheel. Deze opsplitsing vereenvoudigt het lezen en werken verder. Door deze opsplitsing is het ook gemakkelijker om fouten op te sporen. Losse stukken kunnen ook gemakkelijker hergebruikt worden in eigen toepassingen achteraf.

PyPi (***Python Package Index***) is de module manager van python, deze bevat veel van de modules die beschikbaar zijn. Via PIP (***Package Installer for Python***) zijn deze modules zeer gemakkelijk te installeren.

De gebruikte versie van Python is Python 3, specifiek Python 3.7.2 64bit. Dit was op het moment dat we het vision systeem uitbundig gingen testen, de laatst nieuwe versie van Python.

3.3.2. Bibliotheken

3.3.2.a. NumPy

NumPy is een zeer krachtige module, die veel gebruik wordt in wetenschappelijke context voor het verwerken van grote hoeveelheden data. Het meest gekende aspect van NumPy zijn de N-dimensionele arrays (matrices). Deze arrays en bijhorende operaties zijn enorm efficiënt geïmplementeerd waardoor deze bewerkingen zeer snel uitgevoerd worden. Ook met grotere arrays en meerdere dimensies.

Er zijn veel andere Python modules die NumPy gebruiken voor het implementeren van hun datastructuren. Hier wordt NumPy gebruikt om de pixelwaardes van het frame op te slaan. De resultaten zijn telkens ook een NumPy array, evenals de gedetecteerde contouren. Dit laat toe om zeer snel en gemakkelijk bepaalde delen van het beeld te isoleren en verder te verwerken.

Om NumPy te installeren kan het volgende commando uitgevoerd worden:

```
pip install numpy
```

Code 3.2: PIP-commando voor het installeren van NumPy

3.3.2.b. OpenCV

Voor het inlezen en daadwerkelijk verwerken van de beelden wordt OpenCV (***Open Source Computer Vision Library***) gebruikt. Dit is een bibliotheek voor beeldherkenning en –verwerking. OpenCV is een populaire bibliotheek, die gebruikt wordt door bedrijven zoals Google, Microsoft, Intel, IBM en Sony. Enkele van de toepassingen van OpenCV zijn: het samenvoegen en verwerken van de afbeeldingen voor Google Streetview, het monitoren van mijnbouwgereedschap en -machines in China, het detecteren van verdrinkingen in zwembaden, startbanen controleren op puin, etc.

OpenCV kan gebruikt worden in C++, Python, Java en MATLAB op verschillende platformen, waaronder Windows, Linux en Android.

OpenCV is geschreven in C++, en moet gecompileerd worden om te kunnen gebruiken in Python. Op PyPi is een voor-gecompileerde versie te vinden die via PIP zeer gemakkelijk te installeren is. Deze versie is geen officiële versie van de OpenCV ontwikkelaars, maar deze versie vergemakkelijkt het gebruik. Er is minder controle over de installatie van OpenCV omdat er geen aanpassingen kunnen gemaakt worden bij het compileren. In de meeste gevallen is dit ook niet nodig en is het dus veel gemakkelijker om de voor gecompileerde versie te installeren. Dit verkleint ook de kans op fouten bij de installatie. In het programma werd versie op PyPi 4.0.0 gebruikt

Voor het installeren van OpenCV kan in een terminalvenster het volgende commando ingevoerd worden:

```
pip install opencv-python
```

Code 3.3: PIP-commando voor het installeren van OpenCV

Dit zal de laatst nieuwe versie installeren. Om een specifieke versie te installeren kan een commando ingegeven worden als volgt:

```
pip install opencv-python == 4.0.0.21
```

Code 3.4: PIP-commando voor het installeren van OpenCV 4.0.0

Dit commando zal dezelfde versie van OpenCV installeren waarmee het programma ontwikkeld werd. Normaal gezien zou het programma ook moeten werken met nieuwere (en oudere) versies van OpenCV aangezien er geen speciale of specifieke functionaliteit wordt gebruikt.

Er zijn verschillende voor-gecompileerde versies van OpenCV beschikbaar voor Python:

- opencv-python
- opencv-contrib-python
- opencv-python-headless
- opencv-contrib-python-headless

De headless versies bevatten enkel de algoritmes voor de verschillende bewerkingen die uitgevoerd kunnen worden op beelden. Bij deze versies is er dus geen enkele mogelijkheid om de resultaten vanuit het programma, via OpenCV, weer te geven. Deze versies zijn geschikt om op machines te draaien zonder scherm aanhangt en zijn bijgevolg ook de kleinste versies om te installeren.

In deze uitwerking wordt OpenCV gebruikt voor het visualiseren van de resultaten en enkele tussenstappen. Deze versies zijn dus te vermijden als het programma *as-is* gedraaid wordt.

De contrib versies bevatten, naast de standaard functionaliteit, ook uitbreidingen en verbeteringen geschreven en aangebracht door de community, geheel volgens de Open Source visie. Deze versies zijn veel groter om te installeren. De bijkomende functionaliteit kan handig zijn in bepaalde toepassingen.

Voor deze toepassing is de gewone versie voldoende (opencv-python). Ze bevat alle functionaliteit die nodig is voor deze uitwerking. De gewone versie kan geïnstalleerd worden met de eerder vermelde commando's.

3.3.2.c. Imutils

Imutils is een relatief kleine module voor Python. Met deze module is het gemakkelijker om afbeeldingen te verplaatsen, draaien, vergroten en te verkleinen. Ze heeft ook een functie voor het omvormen en sorteren van contouren, gegeven door OpenCV. Het is deze functionaliteit die gebruikt wordt in dit programma.

```
pip install imutils
```

Code 3.5: PIP-commando voor het installeren van Imutils

3.3.2.d. Raspberry Pi camera

Voor het gebruiken van de Raspberry Pi camera module, op de Raspberry Pi, zijn er ook een aantal bibliotheken nodig. Deze bibliotheken zijn standaard geïnstalleerd op Raspbian en zorgen ervoor dat de camera gebruikt kan worden, aangezien deze niet via een USB-aansluiting werkt. Een tweede bibliotheek, zorgt ervoor dat de output van de cameramodule omgezet wordt naar een NumPy array, en dus kan gebruikt worden door OpenCV.

Voor het gebruik en aansturen van de Raspberry Pi camera module is de ***PiCamera*** bibliotheek nodig met de optionele *array* bibliotheek. Beide bibliotheken kunnen geïnstalleerd worden via volgend commando:

```
pip install "picamera[array]"
```

Code 3.6: PIP-commando voor het installeren van PiCamera

Met de eerste bibliotheek, ***PiCamera***, kan de camera module gebruikt en bestuurd worden op de Raspberry Pi via Python. Deze kan aanzien worden als een driver.

De tweede bibliotheek, ***PiRGBArray***, dient enkel voor het correct "formateren" van de data die van de camera module komt. Door de data correct te formateren kan de camera module gebruikt worden in OpenCV.

3.3.2.e. The Python Standard Library

Er zijn nog een aantal andere bibliotheken nodig, maar deze maken allemaal deel uit van de Python Standard Library en worden dus automatisch geïnstalleerd als Python geïnstalleerd wordt. Enkele van deze bibliotheken zijn: *time*, *sys* en *math*.

3.3.2.f. Paho MQTT library

Als laatste is er de MQTT-bibliotheek. Met deze bibliotheek kunnen berichten verzonden en ontvangen worden via het MQTT-protocol. Hier wordt verder op ingegaan onder: **Protocol** op pagina **69**.

Deze bibliotheek werd gedownload van Github en bijgevoegd in de *lib* folder in de code. Op deze manier was deze bibliotheek lokaal opgeslagen en altijd aanwezig. Dit werd gedaan omdat, bij de start van de ontwikkeling van het programma, de installatie via PIP, problemen gaf.

Paho is terug te vinden via volgende link:
https://github.com/eclipse/paho.mqtt.python

Indien er geen problemen ondervonden worden, kan Paho geïnstalleerd worden via volgend commando[1]:

```
pip install paho-mqtt
```

Code 3.7: PIP-commando voor het installeren van de MQTT-client

3.4. Programma

Het volledige programma is terug te vinden in bijlage, hier wordt specifiek ingegaan op het deel verantwoordelijk voor de vision.

3.4.1. Structuur

Het programma is geschreven en georganiseerd om een logische volgorde te hebben en leesbaar te zijn. Het programma werd, tijdens de ontwikkeling, regelmatig getoetst aan de PEP8-standaard (***Python Enhancement Proposal***). Deze standaard bevat de regels om python code leesbaarder te maken, maar ook een aantal do's en dont's voor het programmeren. PEP8 zorgt ervoor dat er een goed onderscheid kan gemaakt worden tussen de verschillende stukken van het programma, zoals klassen, functies etc.

Zo begint elk programma (en Python bestand) met een docstring. Hierin staat beschreven wat het bestand en programma moet doen. Ook wordt er een beperkte

[1] Dit werd ondertussen getest op een andere computer, met een volledig nieuwe installatie van Python, en werkte volledig zonder problemen. Vermoedelijk was er een conflict tussen deze bibliotheek en een andere.

uitleg over variabelen, return-types, en exceptions gegeven. Deze laatste zijn meer van toepassing voor klassen en functies.

Vervolgens komen de import statements voor alle benodigde bibliotheken en klassen. Dit kunnen zowel lokaal als globaal geïnstalleerde bibliotheken zijn. Omdat de Python interpreter niet veel waarde hecht aan de volgorde van code. Functies en klassen kunnen bovenaan of onderaan in het programma, of door elkaar en doorheen het programma, moeten import statements als eerste “echte” code voorkomen in een bestand. De docstring is commentaar en wordt dus niet aanzien als code.

Indien import statements niet bovenaan in een bestand staan, kunnen er fouten optreden waar een functionaliteit niet beschikbaar is. Er kunnen dus bijgevolg fouten optreden die het programma doen crashen.

Volgens PEP8 is het aangeraden om de import statements uit de Python Standard Library te plaatsen vóór de andere import statements. PEP8 raadt ook aan om de import statements uit de Python Standard Library te scheiden van de andere import statements, door een blanco lijn.

Na de import statements komen de globale variabelen. Globale variabelen zijn, in het algemeen maar in Python zeker, zo veel mogelijk te vermijden. Omwille van mogelijke problemen met multithreading en parallelle processen. OpenCV maakt gebruikt van multithreading voor het versnellen van de bewerkingen. Tot nu toe zijn er geen problemen ontstaan, omdat alle andere operaties uitgevoerd worden in één en dezelfde thread. Globale variabelen kunnen vermeden worden door klassen te gebruiken.

Read-only variabelen, zoals metadata, zijn hier wel toegestaan. Hierdoor kunnen modules beter geïdentificeerd worden. Volgens de PEP8-standaard krijgen deze variabelen een naam die begint en eindigt met “__” (twee underscores).

Vervolgens komen de klassen, elk met hun eigen docstring en eventuele metadata. Elke klasse heeft minstens één methode nodig; de constructor. Deze wordt aangeduid door ***__init__***.

```
class Classname(object):
    """
    docstring
    """
    def __init__(self):

        ...
```

Code 3.8: Standaard structuur voor een klasse in Python

Zonder deze constructor, kan een klasse niet gebruikt worden als een object. Deze klasse kan wel gebruikt worden als een eigen structuur om data op te slaan, maar dit is te vermijden. Het is beter om een klasse aan te maken met een constructor, en dan een object te instantiëren. Dit object kan dan gebruikt worden voor het opslaan van de data.

Als laatste komt het deel van het script dat uitgevoerd wordt, aangeduid door:

```
if __name__ == "__main__":

    ...

    while True:

        ...
```

Code 3.9: Aanduiding hoofdprogramma in Python

Dit stuk van de code zal enkel uitgevoerd worden als het bestand hetgeen is welke uitgevoerd wordt. Met andere woorden, als het bestand geïmporteerd wordt in een ander bestand, blijven de functies en klassen beschikbaar, maar wordt dit stuk code niet uitgevoerd.

Onder dit ***if-statement*** is de ***while-lus*** terug te vinden. Alles wat buiten deze lus staat, is vergelijkbaar met de setup functie in Arduino. Alles wat onder deze while-lus staat is vergelijkbaar met de loop functie in Arduino.

3.4.2. Bibliotheken importeren

Als eerste moeten de benodigde bibliotheken geïmporteerd worden. Er zijn drie bibliotheken die in ieder geval nodig zijn: OpenCV, Imutils en NumPy. Deze bibliotheken werden toegelicht in het vorige hoofdstuk.

```
import cv2
import imutils
import numpy as np
```

Code 3.10: Importeren van de bibliotheken

Als er gewerkt wordt met de Raspberry Pi camera, zijn er nog enkele bibliotheken die extra nodig zijn. Deze voorzien de correcte werking en interface om de camera module te kunnen gebruiken met OpenCV.

```
import cv2
import imutils
import numpy as np

# The Raspberry Pi camera module requires these:
from picamera.array import PiRGBArray
from picamera import PiCamera
```

Code 3.11: Importeren van de bibliotheken voor de Raspberry Pi camera

Voor het importeren van Paho uit de lokale ***lib*** folder moet het volgende import-statement gebruikt worden:

```
import lib.paho.mqtt.client as mqtt
```

Code 3.12: Importeren van de MQTT-bibliotheek

3.4.3. Klassen

3.4.3.a. Webcam

Om het programma overzichtelijker te maken en het gebruik van de webcam te vergemakkelijken werd een klasse geschreven. Deze klasse breidt de VideoCapture klasse van OpenCV uit. Hierdoor worden alle nodige instellingen, voor het gebruik van de webcam ingesteld bij het aanmaken van het object. Door deze code af te zonderen naar een klasse werd het hoofdprogramma een stuk leesbaarder.

Ook werden de optimale waarden voor enkele instellingen van de webcam, ingegeven als standaardwaarden, waardoor het aanmaken van het webcam object nog gemakkelijker wordt. De nodige instellingen moeten enkel meegegeven worden indien ze verschillen van de standaardwaarden.

Door de manier waarop klassen en ***inheritance*** geïmplementeerd zijn in Python, wordt de constructor van de parent-klasse overschreven van zodra er een constructor geschreven wordt in de klasse die erft van de parent (dit geldt voor elke twee methodes met dezelfde naam). In deze toepassing is het belangrijk om de originele constructor te behouden en ook aan te roepen.

Dit gebeurt via het ***super*** sleutelwoord. Op deze manier wordt de constructor van de parent-klasse aangeroepen en worden alle eigenschappen, functies en variabelen overgeërfd naar de huidige klasse en object.

Zo wordt het gebruik van het VideoCapture object vereenvoudigd en ook uitgebreid met de eigen methodes, zoals de auto-exposure functie.

```
class WebCam(cv2.VideoCapture):
    """
    WebCam class to act and deal with usb webcams.
    This class is an extension on the OpenCV VideoCapture
    class.
    """

    def __init__(self,
                 channel: int = 0,
                 resolution: tuple = (1440, 1080),
                 exposure: int = -2,
                 framerate: int = 40) -> cv2.VideoCapture:
        # Call the super class to generate a VideoCapture
        # object.
        super(WebCam, self).__init__(channel)
        # Set the capture resolution, works best with 4:3
        # aspect-ratio.
        self.set(cv2.CAP_PROP_FRAME_WIDTH, resolution[0])
        self.set(cv2.CAP_PROP_FRAME_HEIGHT, resolution[1])
        # Set the exposure of the camera
        self.set(cv2.CAP_PROP_EXPOSURE, exposure)
        # Set the framerate on the webcam.
        self.set(cv2.CAP_PROP_FPS, framerate)
```

Code 3.13: WebCam klasse met constructor

Op de Maker Faire waren er problemen met de belichting (hier wordt later op teruggekomen) waardoor de vision niet altijd even betrouwbaar werkte. Hierdoor kwam de nood voor een functie om de gevoeligheid van de camera bij te regelen, naar boven.

Daarom werd de ***adjust_exposure***-methode geschreven onder de Webcam-klasse. Deze functie leest 5 frames van de webcam en berekent de gemiddelde belichting van de vijf frames. Op basis van deze gemiddelde waarde wordt de gevoeligheid van de camera bij geregeld.

```
def adjust_exposure(self, target: int = 127):
        while True:
            brightness = 0
            for i in range(5):
                # Read 5 frames and take the brightness
                # from those.
                _, frame = self.read()
                frame = cv2.cvtColor(frame,
                                     cv2.COLOR_BGR2GRAY)
                brightness += cv2.mean(frame)[0]
            # Average out the brightness
            brightness /= 5
```

Code 3.14: Het inlezen van de frames voor de auto-exposure

Omdat de gevoeligheid van de webcam slechts te regelen is in beperkte stappen, is er een marge aangebracht rond de gewenste belichtingswaarde. De functie blijft zichzelf herhalen tot de belichtingswaarde binnen de marge is, of totdat één van de grenzen van de gevoeligheid bereikt is. Van zodra één van de voorwaarden voldaan is wordt de lus doorbroken en keert de methode terug naar de rest van het programma.

```
            # If the brightness is not within a certain
            # margin from the target adjust the exposure.
            if not (target - 32) <
                    brightness <
                    (target + 32):

                if brightness < target:

                    exposure = self.get(
                                    cv2.CAP_PROP_EXPOSURE)
                    if exposure == -1:
                        break
                    self.set(cv2.CAP_PROP_EXPOSURE,
                             exposure + 1)

                elif target < brightness:

                    exposure = self.get(
                                    cv2.CAP_PROP_EXPOSURE)
                    if exposure == -15:
                        break
                    self.set(cv2.CAP_PROP_EXPOSURE,
                             exposure - 1)
                else:
                    break
            else:
                break

        # Return to the main program
        return
```

Code 3.15: Het bijregelen van de exposure op basis van de belichting

Deze methode werd oorspronkelijk geschreven met het gemiddelde van slechts één frame, maar dit gaf zeer onbetrouwbare resultaten. De eerste keer dat de functie aangeroepen werd, was de belichting te donker en werd de gevoeligheid van de webcam verhoogd. De volgende keer was de belichting te fel, waardoor de gevoeligheid verlaagd werd. De oorzaak hiervoor was te vinden bij de webcam. De webcam had niet altijd een frame klaar op het moment dat het programma het volgende frame wou inlezen. De webcam wordt op de computer, door het operating system, voorgesteld als een buffer. Als er een frame ingelezen wordt in het programma, wordt dit gelezen vanuit de buffer. De webcam zet steeds nieuwe frames aan het begin van de buffer.

Op het moment dat de webcam achter komt wordt dus tweemaal hetzelfde frame ingelezen. Wat de oorzaak van dit probleem bleek te zijn. Door het gemiddelde van vijf frames te nemen werd dit probleem opgelost en verdween het bijhorende jojo-effect.

3.4.4. Functies

3.4.4.a. Mask_sides

Deze functie is meer een gemaks-functie. Ze overschrijft de randen van het frame met een witte rechthoek. Dit werd gedaan omdat het zichtveld van de webcam groter was dan het spelbord, waardoor er aan de randen van het frame objecten gedetecteerd werden die buiten het spelbord waren. Door deze te overschrijven met witte rechthoeken voor enige operaties worden deze volledig genegeerd.

Deze operaties werden in een functie geschreven om het programma overzichtelijk te houden.

3.4.4.b. Order_66

```
# making the array to send
arr = bytearray([20, 61, 0, 0, 11, 0, 0, 0])

# stop the car
arr[0] = 20

mqtt_client.publish("vroom", arr)

# reset the lights
arr[0] = 21

mqtt_client.publish("vroom", arr)

# disable the controls
arr[0] = 60

mqtt_client.publish("vroom", arr)
```

Code 3.16: Order_66 functie

Wanneer het autootje 3 toeren heeft gedaan zullen de rondetijden worden stopgezet en het gemiddelde berekend worden. Maar de besturing is nog actief en de lichten staan nog aan. Hiervoor is dit stukje code. Niet alleen voor een bewijs dat het vision systeem een apparaat kan besturen maar ook voor alles te stoppen.

Als eerste wordt er een bericht opgemaakt en verzonden naar het autootje. Deze zal ervoor zorgen dat de motoren stoppen met draaien. Aangezien alles van het

bericht hetzelfde is voor de volgende twee verzendingen. Wat er wel elke keer moet veranderen is het doel, voor wie het bericht bestemd is. Deze berichten worden dan een voor een verzonden.

3.4.4.c. Finished

Deze functie is verantwoordelijk voor alles wat te maken heeft met de finishlijn en wordt aangeroepen telkens een autootje over de finishlijn rijdt. Door tijdsgebrek is deze functie minder goed geschreven. De Maker Faire stond voor de deur en de finishlijn moest op zijn minst werken. De hele functie zou beter herschreven worden als een klasse zodat ook het gebruik van de globale variabelen vermeden kan worden. Desalniettemin doet de functie wat ze moet doen.

Als eerste leest de functie de globale variabelen in en wordt het maximale aantal rondes vastgelegd.

```
def finished():
    """
    Function triggered when the car crosses the finish line.
    """
    # !Should not be here
    # Access the global variables.
    global LAP_TIMES
    global START_TIME
    global CURRENT_LAP
    # Set the maximum amount of laps.
    max_laps = 3
```

Code 3.17: Aanroepen van de finished() functie

Vervolgens wordt de huidige ronde met één vermeerderd. Op basis van de waarde van de huidige ronde wordt er beslist welke actie genomen moet worden.

Zolang het niet de laatste ronde is wordt dit stuk code uitgevoerd. Als eerste wordt er gekeken of het de eerste ronde is. Indien dit het geval is wordt de lijst met rondetijden leeg gemaakt en de tijd gestart.

Als het niet de eerste ronde is, wordt de rondetijd opgenomen en opgeslagen. Om verdere storingen door het vision-systeem te voorkomen, moet een ronde minsten vijf seconden bedragen.

Als de ronde minder dan vijf seconden heeft geduurd wordt de tijd genegeerd en wordt de huidige ronde met één verminderd. Hierdoor blijft alles werken en worden er geen vals-positieven gedetecteerd.

```
if CURRENT_LAP <= max_laps + 1:
    # If it's the first lap, start the timer.
    if CURRENT_LAP == 1:
        LAP_TIMES = []
        START_TIME = time.time()
        print()
        print("Starting times")
    else:
        # During the other laps, calculate the lap time.
        lap_time = time.time() - START_TIME
        # If the lap time is longer than 5 seconds we can
        # proceed. This is done to filter out glitches in
        # the vision system.
        if lap_time >= 5:
            # Set a new start time.
            START_TIME = time.time()
            # print the time to the terminal
            # TODO: add functionality to add lap time to
            # the screen.
            print("lap: {}, lap time: {}"
                  .format(CURRENT_LAP - 1, lap_time))
            # Add the lap time to the list of lap times
            # for later reference.
            LAP_TIMES.append(lap_time)
        else:
            # If the lap isn't longer than five seconds,
            # undo the lap increment.
            CURRENT_LAP -= 1
```

Code 3.18: Code voor de finishlijn

Als het de laatste ronde is wordt de tijd gestopt en de controle over het autootje wordt stopgezet. Na de laatste ronde wordt ook de snelste en gemiddelde rondetijd uitgebracht naar het scherm. Ook wordt de huidige ronde gereset naar nul, zodat de volgende speler kan beginnen.

```
# If the current lap is the maximum (or last) lap.
if CURRENT_LAP == max_laps + 1:
    # Disable controls to the car.
    order_66()
    # Print finished to the terminal.
    print('Finished')
    # Calculate the average and fastest lap times.
    average = sum(LAP_TIMES) / len(LAP_TIMES)
    fastest = min(LAP_TIMES)
    # Print the results to the terminal.
    print("Average time: {}, record:{}"
            .format(average, fastest))
    print()
    # Reset the current lap
    CURRENT_LAP = 0
```

Code 3.19: Code voor de laatste ronde

3.4.4.d. Name_colour

De functie vraagt als argument een tuple met drie waardes. Deze tuple stelt de kleur voor van de pixel. Als eerste wordt de tuple opgesplitst in drie afzonderlijke waarden.

```
def name_colour(pixel_colour: tuple = (0, 0, 0)) -> str:
    # Split the pixel into three separate variables.
    blue = pixel_colour[0]
    green = pixel_colour[1]
    red = pixel_colour[2]
```

Code 3.20: Opsplitsen van een tuple in afzonderlijke waardes

Deze drie waarden worden dan met elkaar vergeleken en aan de hand van de vergelijking wordt de kleur bepaald.

Als alle drie de kanalen hoger zijn dan 250, is de kleur wit. Gelijkaardig zal de kleur zwart zijn als alle drie de kanalen een waarde hebben lager dan 50.

Voor het detecteren van de daadwerkelijke kleuren wordt gekeken welk kanaal de hoogste waarde heeft. Op deze manier kunnen er slechts drie kleuren gedetecteerd worden, naast zwart en wit.

```
    # Start with a blank name.
    c_name = ''

    # If all colours are above 250 the colour is white.
    if blue > 250 and green > 250 and red > 250:
        c_name = 'white'
    # If all colours are below 50 the colour is black.
    elif blue < 50 and green < 50 and red < 50:
        c_name = 'black'

    # Extract the other primary colours.
    # TODO: improve the "algorithm" for checking colours.
    elif blue > green and blue > red:
        c_name = 'blue'
    elif green > blue and green > red:
        c_name = 'green'
    elif red > blue and red > green:
        c_name = 'red'

    return c_name
```

Code 3.21: Bepalen van de kleur

Als laatste wordt de naam van de kleur als een string teruggegeven. Er wordt gestart met een lege string voor de naam, zodat als er iets fout loopt en er geen kleur wordt herkend, er een lege string wordt teruggegeven. Op deze manier zal de rest van het programma niet crashen, maar deze kleur (en vorm) gewoon negeren.

3.4.5. Main

3.4.5.a. Frame inlezen

Aangezien er gewerkt wordt met ***live-video***, moet er aan het begin van elke programmalus een nieuw frame ingelezen worden. Samen met de opstart van de camera zijn dit de enige stukken code die (kunnen) verschillen afhankelijk van de camera. Ze zullen hier dus ook samen bekeken worden.

```
# Creating a webcam object.
CAP = WebCam()

# Running auto exposure-adjustment
CAP.adjust_exposure(150)

while True:
    # Read a frame from the webcam
    _, FRAME = cap.read()
```

Code 3.22: Code voor het gebruik van een USB-camera

Doordat de WebCam-klasse werd geschreven is het gebruik van de webcam zeer eenvoudig, zeker als alle standaardinstellingen behouden worden. Er kan gewoon een webcam object aangemaakt worden en direct gebruikt worden. Doordat de WebCam-klasse de VideoCapture klasse van OpenCV uitbreidt is het gebruik volledig hetzelfde.

Indien de standaardwaarden verschillen van de gewenste waarden, kunnen deze meegegeven worden aan de constructor als argumenten. Eén van de meest voorkomende afwijkingen is het kanaal waarop de camera te vinden is. Op deze manier kunnen meerdere camera's verbonden worden met dezelfde computer en gebruikt worden in OpenCV. OpenCV kan dus ook gebruikt worden voor 3D toepassingen met twee camera's.

Op een desktop met slechts één aangesloten camera zal het kanaal altijd 0 zijn. Op een laptop, met een ingebouwde camera, kan het zijn dat deze camera als kanaal 0 wordt aanzien en dat een webcam dus kanaal 1 zal krijgen. De laptop waarop het programma getest werd had een ingebouwde camera, maar de USB-camera was op kanaal 0 te vinden.

Voor het gebruik van de USB-camera moest geen enkele driver geïnstalleerd worden. Er werd slechts één USB-camera getest, dus het is mogelijk dat voor andere camera's wel drivers geïnstalleerd moeten worden.

De webcam wordt nu in een oneindige lus uitgelezen en het programma is dus enkel te stoppen via een ***KeyboardInterrupt*** (ctrl + c), wat is een geforceerde manier inhoudt om het programma te stoppen.

Het scherm, dat door OpenCV getoond wordt, kan gebruikt worden om het programma gecontroleerd af te sluiten. Zolang het venster open is, kan de webcam verder uitgelezen worden. Op het moment dat het visualisatie-scherm gesloten wordt, door bv. het windows-kruisje in te duwen, moet het programma stoppen. Dit wordt geïmplementeerd door de while-lus te vervangen door het volgende fragment:

```
# As long as the window is opened keep running. This allows
# the windows close button ('X') to be functional.
while cv2.getWindowProperty('frame', 0) >= 0:

    ...
```

Code 3.23: While-lus voor het kunnen sluiten van het programma

Het programma kan op een andere, gecontroleerde, manier afgesloten worden, door een toets in te duwen. Dit wordt verkregen door het volgende fragment:

```
# As long as the window is opened keep running. This allows
# the windows close button ('X') to be functional.
while cv2.getWindowProperty('frame', 0) >= 0:

    ...

    if cv2.waitKey(1) & 0xff == ord('q'):
        # Terminate the loop when the 'q' key is pressed.
        break
```

Code 3.24: Code voor het stoppen van OpenCV met een toets

Voor het gebruik van de Raspberry Pi camera, zijn twee bijkomende bibliotheken nodig.

```
import cv2

from picamera import PiCamera
from picamera.array import PiRGBArray

if __name__ == "__main__":

    # Initialize the camera.
    camera = PiCamera()
    # Set the resolution.
    camera.resolution = (640, 480)
    # Set the framerate.
    camera.framerate = 32
    # Create an object for the raw camera video stream.
    rawCapture = PiRGBArray(camera, size=(640, 480))

    for image in (
        camera.capture_continuous(rawCapture,
                                  format="bgr",
                                  use_video_port=True)):

        # Grab the image coming from the camera and get the
        # NumPy-array representing this image.
        FRAME = image.array

        ...
```

Code 3.25: Code voor het gebruik van de Raspberry Pi Camera module

De twee benodigde bibliotheken zijn PiCamera en PiRGBArray. Deze werden reeds toegelicht onder het hoofdstuk **Raspberry Pi camera** op pagina **19**.

Net als bij de USB-webcam moet een camera-object aangemaakt worden. Voor de Raspberry Pi cameramodule werd nog geen klasse geschreven zoals voor de webcam, hierdoor is het aanmaken van het object ietwat omslachtiger.

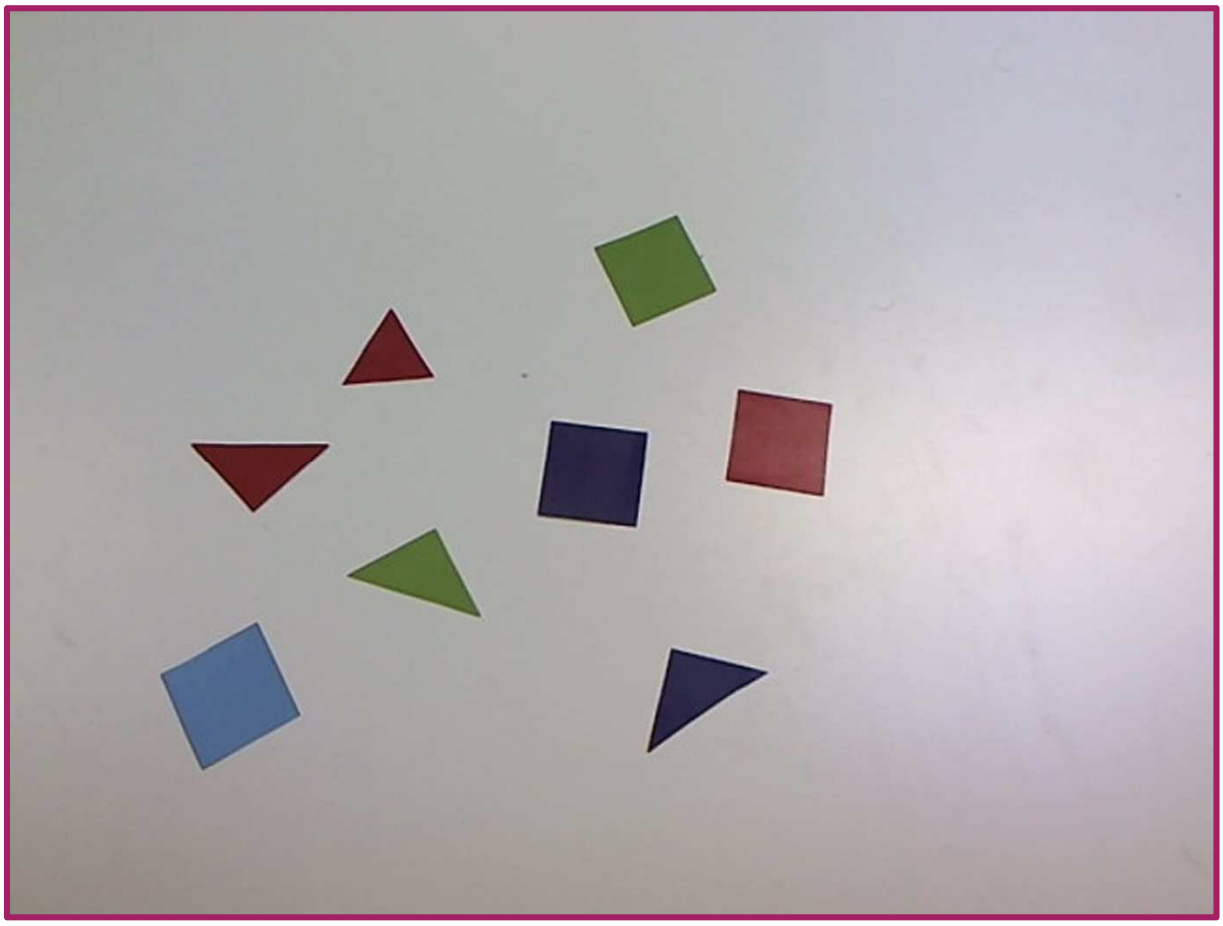
Dit object kan dan gebruikt worden voor het instellen van de vereiste parameters, zoals de resolutie[2] en de framerate. Vervolgens kan een object aangemaakt worden dat de video-stream van de cameramodule beschikbaar maakt voor de rest van het programma.

[2] Het instellen van de resolutie vereist wat finesse; de Raspberry Pi camera module ondersteunt slechts een beperkt aantal resoluties. Deze resoluties zijn ook niet beschikbaar bij elke framerate. De USB-webcam werkte het beste bij een resolutie verhouding van 4:3. Ook hier waren de resoluties beperkt, een resolutie van 1440 op 1080 leek het beste te werken.

De while-lus is nu vervangen door een for-lus. Beide hebben hetzelfde effect: bij de start van elke lus wordt een frame ingelezen en verwerkt.

Op de Raspberry Pi is het programma enkel te stoppen via een toets.

Als alles gelukt is, wordt het volgende beeld verkregen, dit wordt verder verwerkt in het programma.



Figuur 3.4: Het ingelezen frame

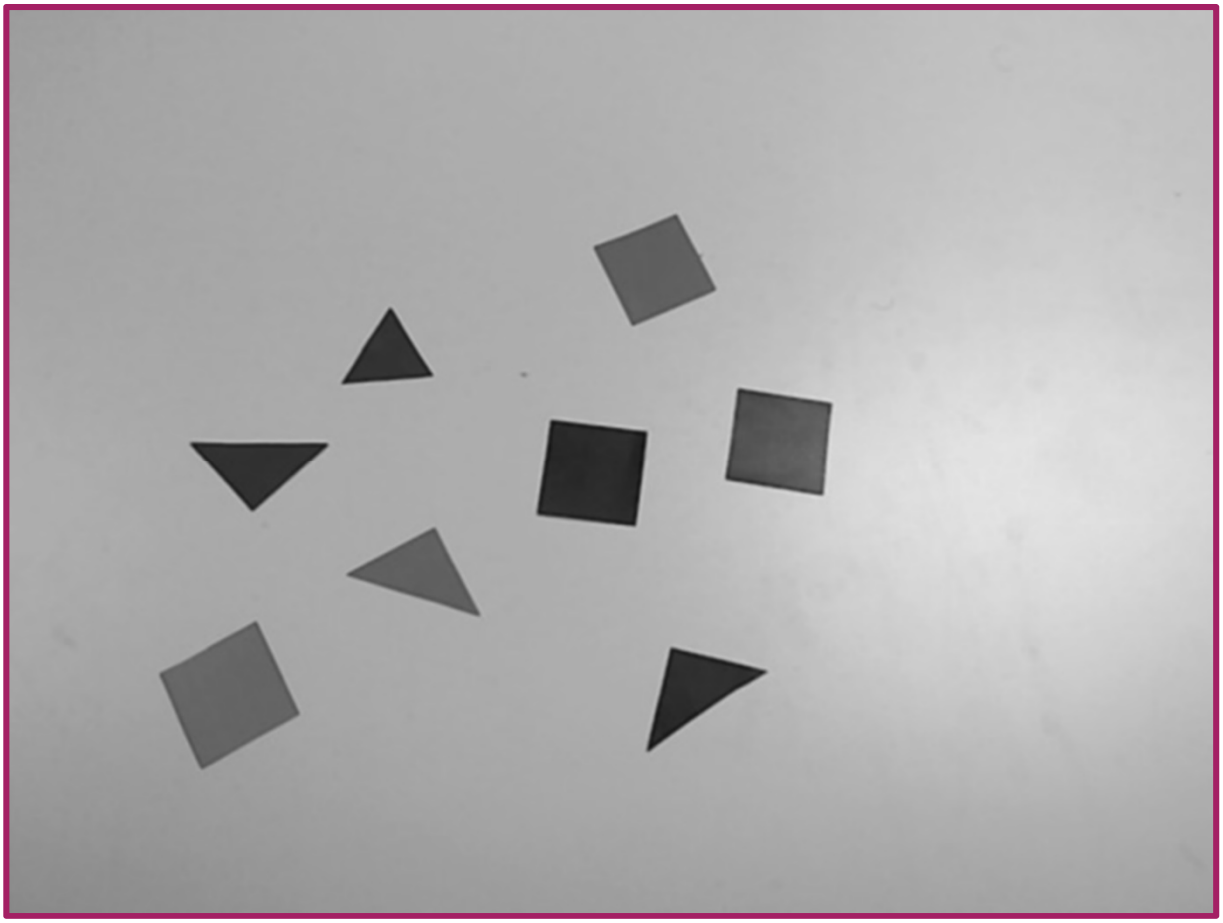
3.4.5.b. Omzetten naar grijstinten

De allereerste bewerking die wordt uitgevoerd op het ingelezen frame, is een omzetting naar grijstinten. Een kleurenbeeld bevat drie kanalen per pixel (rood, groen en blauw). Voor de bewerkingen die volgen worden alle pixels in het hele frame overlopen en worden de bewerkingen uitgevoerd op alle kanalen van alle pixels.

Zoals het citaat in het begin van dit hoofdstuk reeds vermelde, draait het bij computer-vision voornamelijk om intensiteit en de verandering ervan. Bijgevolg hebben we voor de meeste operaties geen nood aan de extra informatie die kleur met zich meebrengt.

Door het omzetten naar grijstinten worden de drie kanalen herleidt tot één kanaal dat de intensiteit weergeeft. Hierdoor bedraagt het aantal benodigde berekeningen slechts een derde, in vergelijking met een kleurenbeeld., waardoor de verwerking ook, ruwweg, drie keer zo snel gebeurt.

Er wordt wel een kopie bewaard van het kleurenbeeld zodat dit later kan weergegeven worden. Het kleurenbeeld wordt ook gebruikt om later de kleur te bepalen van de gedetecteerde vormen.



Figuur 3.5: Frame omgezet naar grijstinten

Het omzetten naar grijstinten wordt als volgt verkregen in OpenCV:

```
# convert the frame to greyscale.
GRAY = cv2.cvtColor(FRAME, cv2.COLOR_BGR2GRAY)
```

Code 3.26: Code fragment kleur omzetten naar grijstinten

3.4.5.c. Gaussian blur

De volgende operatie is een ***Gaussian blur*** om de hoeveelheid ruis te verminderen, de grootste boosdoener bij computer-vision. Een aantal latere bewerkingen zijn enorm gevoelig voor ruis. Ruis, in alle vormen, wordt in alle stappen zo veel mogelijk onderdrukt op verschillende manieren. De gebruikte ***kernel*** werd

proefondervindelijk bepaald. De grootte is een balans tussen het onderdrukken van ruis en het behouden van detail.

In OpenCV wordt een Gaussian blur als volgt verkregen:

```
# Use a gaussian blur to reduce noise in the image.
KERNEL = (9, 9)
GRAY = cv2.GaussianBlur(GRAY, KERNEL, 0)
```

Code 3.27: Code fragment Gaussian blur en kernel

De parameters die aan deze functie moeten worden meegegeven zijn de volgende:

Het beeld waarop de operatie moet worden uitgevoerd, de ***kernel*** en de standaardafwijking. Als de standaardafwijking nul is, wordt deze berekend uit de ***kernel***.

Meer detail en uitleg over ***Gaussian blur*** en bijhorende parameters kan gevonden worden onder **Gaussian blur** op pagina **86**.

3.4.5.d. Threshold

Aangezien het hier gaat over het detecteren van vormen, is een zo hoog mogelijk contrast wenselijk. Het contrast kan verhoogd worden tot er slechts twee mogelijke waardes over zijn voor een pixel. Dit kan bekomen worden op verschillende manieren. De meest efficiënte manier is een binaire drempel (***binary threshold***).

Hierbij wordt elke pixel vergeleken met een drempelwaarde: wanneer de pixelwaarde lager is dan de drempelwaarde, wordt hij zwart gezet. Wanneer de pixelwaarde hoger is dan de drempelwaarde, dan wordt hij wit gezet.

Op die manier wordt een zeer hoog contrast verkregen, wat zeer geschikt is voor verdere operaties. De drempelwaarde werd ook proefondervindelijk bepaald en moet aangepast worden aan de hoeveelheid omgevingslicht. Een bijkomend voordeel is dat dit ook verder ruis onderdrukt en dat een zeer egaal beeld verkregen wordt.

In OpenCV kan een ***threshold*** verkregen worden als volgt:

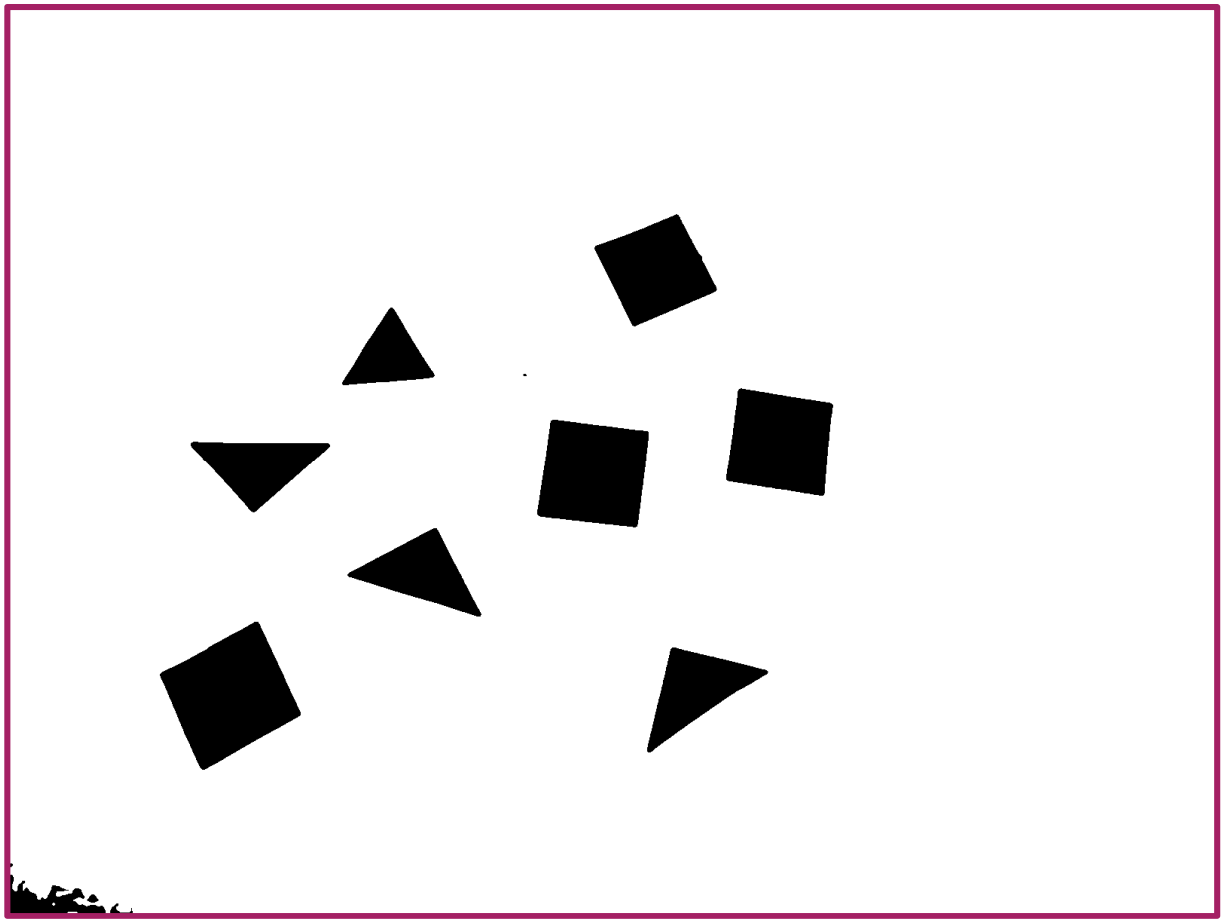
```
# use a binary threshold on the image.
_, THRESHOLD = cv2.threshold(GRAY, 140, 255,
                             cv2.THRESH_BINARY)
```

Code 3.28: Code fragment threshold

De functie geeft twee waarden terug. De eerste waarde die teruggegeven wordt is de drempelwaarde. Deze is echter van toepassing voor andere drempel-technieken. De teruggegeven drempelwaarde moet dus niet bijgehouden worden

(voorgesteld door de “_”). De tweede waarde die teruggegeven wordt is het resultaat van de ***threshold*** operatie. Deze wordt opgeslagen als een nieuwe afbeelding.

Er moeten een aantal parameters meegegeven worden aan de functie; van links naar rechts: het originele beeld waarop de operatie wordt uitgevoerd, de drempelwaarde, de maximumwaarde en het type ***threshold*** dat gebruikt wordt. De maximumwaarde is de “witte” waarde voor de pixel, hier ingesteld op 255.



Figuur 3.6: Resultaat van de threshold operatie

Op het resultaat is te zien hoe alle figuren omgevormd zijn tot volledig zwarte vlakken op een witte achtergrond, door de drempel-operatie. Er is echter ook nog een restant van ruis gedetecteerd. Deze ruis vormt geen direct probleem, en wordt bij latere operaties verder uitgefilterd.

Voor de edge detectie is normaal het inverse vereist: witte vormen op een zwarte achtergrond. In OpenCV is een afbeelding zeer gemakkelijk te inverteren. De gewenste resultaten werden echter niet verkregen met een geïnverteerd beeld. Vandaar dat er werd verder gewerkt met het resultaat van de drempel-operatie, zonder deze te inverteren. Dit gaf wel de gewenste resultaten.

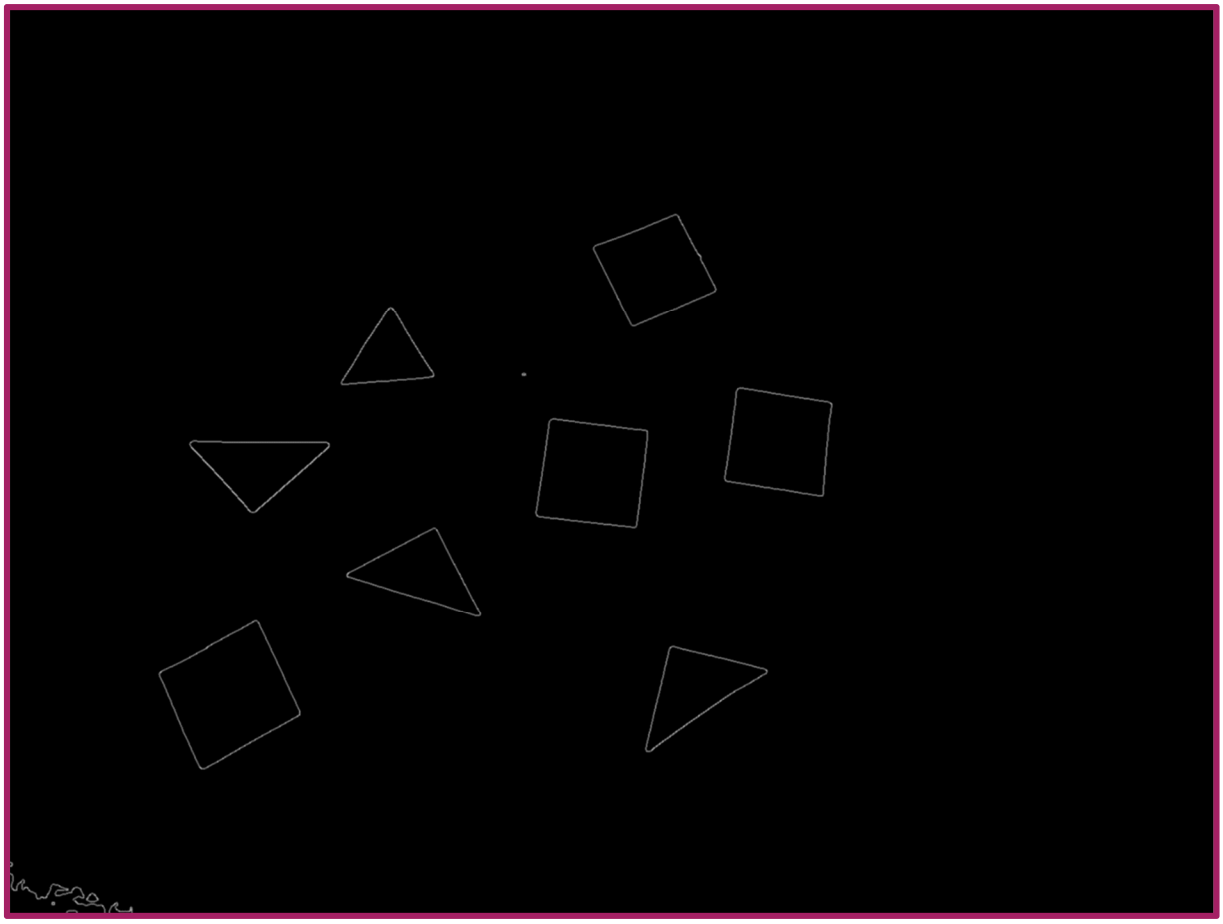
3.4.5.e. Canny edge detection

De figuren zijn herleid tot volle vlakken, maar kunnen nog altijd niet herkend worden door het programma. Het menselijk brein kan onmiddellijk zien welke figuur overeenkomt met welke vorm. Maar hoe wordt een vorm herkend?

Een vorm met vier zijdes betekent een vierkant (of rechthoek), één met drie zijdes betekent een driehoek, enzoverder. Dit is exact dezelfde manier waarop de computer de vormen herkent.

De volgende stap is het detecteren van de zijden. Dit wordt verkregen door het detecteren van randen in het beeld. Een rand is een scherpe overgang van één intensiteit naar een andere. Door de threshold operatie zijn deze overgangen zo scherp als ze maar kunnen zijn. Voor het detecteren van de randen wordt ***Canny edge detection*** gebruikt.

Door alle eerder uitgevoerde operaties is het beeld ondertussen veel vereenvoudigd waardoor enkel de randen (en omtrek) van de figuren bekomen worden. Er is opnieuw ruis dat gedetecteerd wordt.



Figuur 3.7: Resultaat van edge-detectie

Daarom wordt op de gedetecteerde randen nogmaals een Gaussian blur gebruikt. Hierdoor worden ook kleine imperfecties in de randen onderdrukt waardoor het

detecteren van de contouren veel betere resultaten geeft en de vormen nauwkeuriger worden gedetecteerd. In de volgende stap wordt hier verder op ingegaan.

Het detecteren van de randen in OpenCV gebeurt als volgt:

```
# Detect the edges in the image.
EDGED = cv2.Canny(THRESHOLD, 25, 145)
# Reduce noise on the detected edges.
EDGED = cv2.GaussianBlur(EDGED, (3, 3), 0)
```

Code 3.29: Canny edge detection in OpenCV

Meer detail over de edge detectie kan gevonden worden onder **Edge detection** op pagina **86**.

3.4.5.f. Contouren

Via volgend codefragment worden de contouren uit het resultaat van de edge detection gehaald en geformatteerd.

```
# Find the contours in the image.
CONTOURS = cv2.findContours(EDGED.copy(),
                            cv2.RETR_TREE,
                            cv2.CHAIN_APPROX_SIMPLE)
# Get the contours,
# to deal with different versions of OpenCV
CNTS = imutils.grab_contours(CONTOURS)
```

Code 3.30: Contouren detecteren en isoleren in OpenCV

De laatste lijn van dit fragment formatteert de gevonden contouren als een lijst. Hierdoor kunnen de gevonden contouren gemakkelijk overlopen worden in een for-lus.

```
for c in CNTS:
    # Approximate the contour.
    PERIMETER = cv2.arcLength(c, True)
    # Filter the perimeters on the length.
    if 80 <= PERIMETER <= 400:
        # Approximate a polygon based on the contour
        approx = cv2.approxPolyDP(c, 0.03 * PERIMETER, True)
```

Code 3.31: For-lus om over de contouren te lopen en te filteren

Voor iedere gedetecteerde contour wordt de omtrek berekend (***arclength***). Die gebruikt wordt om verder ruis weg te filteren. Te kleine en te grote contouren worden genegeerd.

Als laatste wordt een veelhoek benaderd op basis van de contour en de omtrek. Deze laatste wordt dan gebruikt voor het identificeren en lokaliseren van figuren. In deze functie is een factor terug te vinden (0,03), die is een indicatie is van de de mate waarin de vorm mag afwijken ten opzichte van de ideale vorm.

De functie gebruikt deze waarde om te bepalen wanneer hoekpunten dicht genoeg bij elkaar liggen om aanzien te worden als één hoekpunt. Hier worden hoekpunten aanzien als één hoekpunt als de afstand tussen de twee hoekpunten minder is dan 3% van de omtrek.

Deze factor kan aangepast worden als vormen niet (correct) gedetecteerd worden, bijvoorbeeld door ruis op het beeld. Dit wordt reeds deels onderdrukt door de tweede Gaussian blur die wordt uitgevoerd na het detecteren van de randen.

Deze tweede Gaussian blur kan niet te sterk gemaakt worden, omdat anders de randen zullen verdwijnen.

Het ook zijn dat er sleet is op of schade aan de fysieke objecten, waardoor ze geen perfecte vorm meer hebben. Als deze factor vergroot wordt, is er een grotere tolerantie op hoeveel de gedetecteerde vormen moeten overeenkomen met een perfecte vorm.

3.4.5.g. Hoekpunten tellen in de contour & vorm bepalen

De vorm wordt bepaald door het tellen van het aantal hoekpunten in de contouren. Door de vorige operaties is het aantal hoekpunten al verminderd, waardoor de vormen nu zeer gemakkelijk en goed kunnen gedetecteerd worden. De contouren worden teruggegeven als een lijst met coördinaten van de hoekpunten. Door te kijken hoe “lang” iedere contour is, kan gekeken worden om welke vorm het gaat. In Python kan dit bekomen worden als volgt:

```
if len(approx) == 3:
    # Triangles.
    ...

elif len(approx) == 4:
    # squares and rectangles.
    ...

else:
    # A different shape
    pass
```

Code 3.32: Hoekpunten differentiëren in de contouren

Python beschikt niet over een ***switch*** functie, maar dit resultaat kan ook bekomen worden via een ***if-else*** structuur. Er kan een onderscheid gemaakt worden tussen elk nummer van hoekpunten, maar op een bepaald punt is het onderscheid tussen de vormen te klein. Vandaar dat het aantal hoekpunten best beperkt wordt tot een maximum van acht. Alles boven acht kan dan worden aanzien als een cirkel.

3.4.5.h. Zwaartepunt van de vorm bepalen

Nadat de vorm gedetecteerd is, wordt het zwaartepunt bepaald, in de meeste gevallen is dit het center. Voor driehoeken kan dit problemen geven, vandaar dat het zwaartepunt gebruikt wordt.

```
M = cv2.moments(c)
if M['m00'] != 0:
    # Extract the centre of the triangle coordinates.
    cX = int(M['m10'] / M['m00'])
    cY = int(M['m01'] / M['m00'])
```

Code 3.33: Het zwaartepunt berekenen van een gedetecteerde vorm

3.4.5.i. Kleur bepalen

Het herkennen van de kleur gebeurt op dit moment op een vrij rudimentaire manier, maar werkt goed genoeg voor deze toepassing. Omdat er enkel gekeken wordt naar een rode driehoek.

```
# Name the colour of the triangle.
colour = FRAME[cY, cX]
colour_name = name_colour(colour)
```

Code 3.34: Pixel isoleren en kleuren-functie aanroepen

Nadat de coördinaten van het zwaartepunt van de gedetecteerde vorm zijn berekend, worden deze gebruikt om de desbetreffende pixel uit het frame te halen.

De uitleg over deze functie is te vinden onder **Name_colour** op pagina **30**.

3.4.5.j. Hoekpunten en kleur doorgeven naar de rest van het programma

Nu dat alles gedetecteerd is en correct herkend is, kan dit doorgegeven aan de rest van het programma. Momenteel wordt nog niets met deze informatie gedaan. Alle acties worden genomen vanuit de functie die de vormen herkend. Het is maar een kleintje om dit door te geven aan een andere functie, of zelfs over het netwerk door te sturen naar een andere computer en programma.

3.4.6. Resultaat

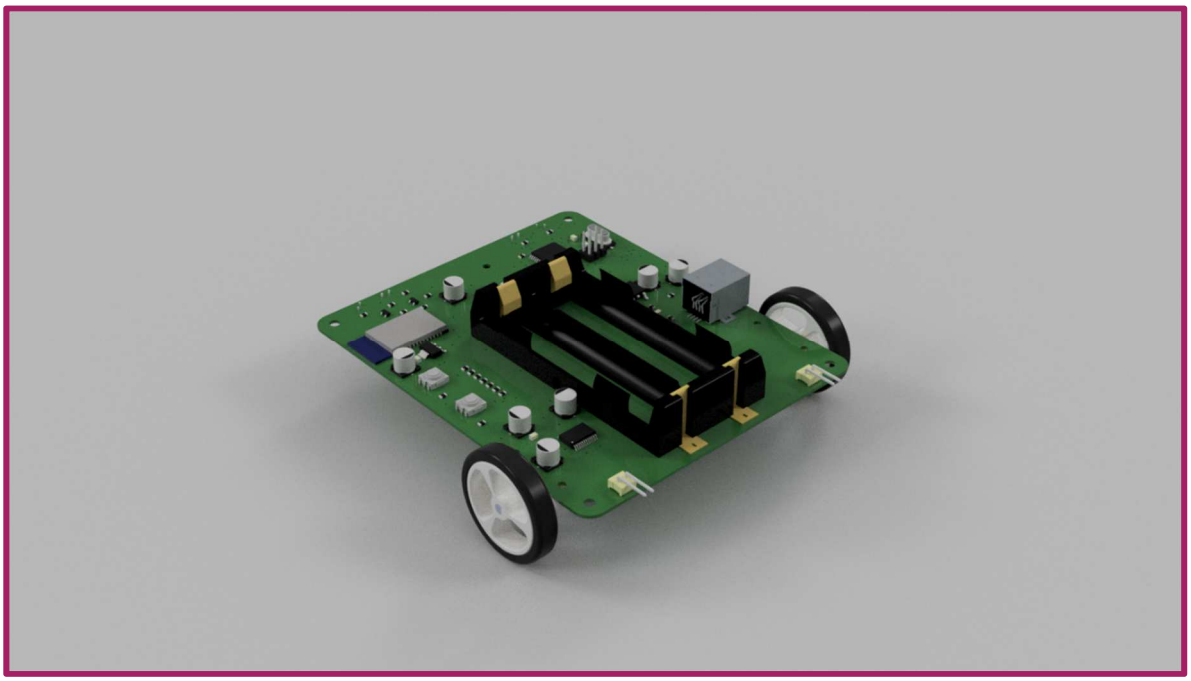
Figuur 3.8: Resultaat van het vision-programma

Voor het gedeelte van het programma dat verantwoordelijk is voor de vision te illustreren werd een kleine demo geschreven.

Deze demo herkent de verschillende vormen en kleuren. Nadat de detectie voltooid is, worden alle vormen voorzien van een label. Deze demo slaat ook alle tussen stappen op.

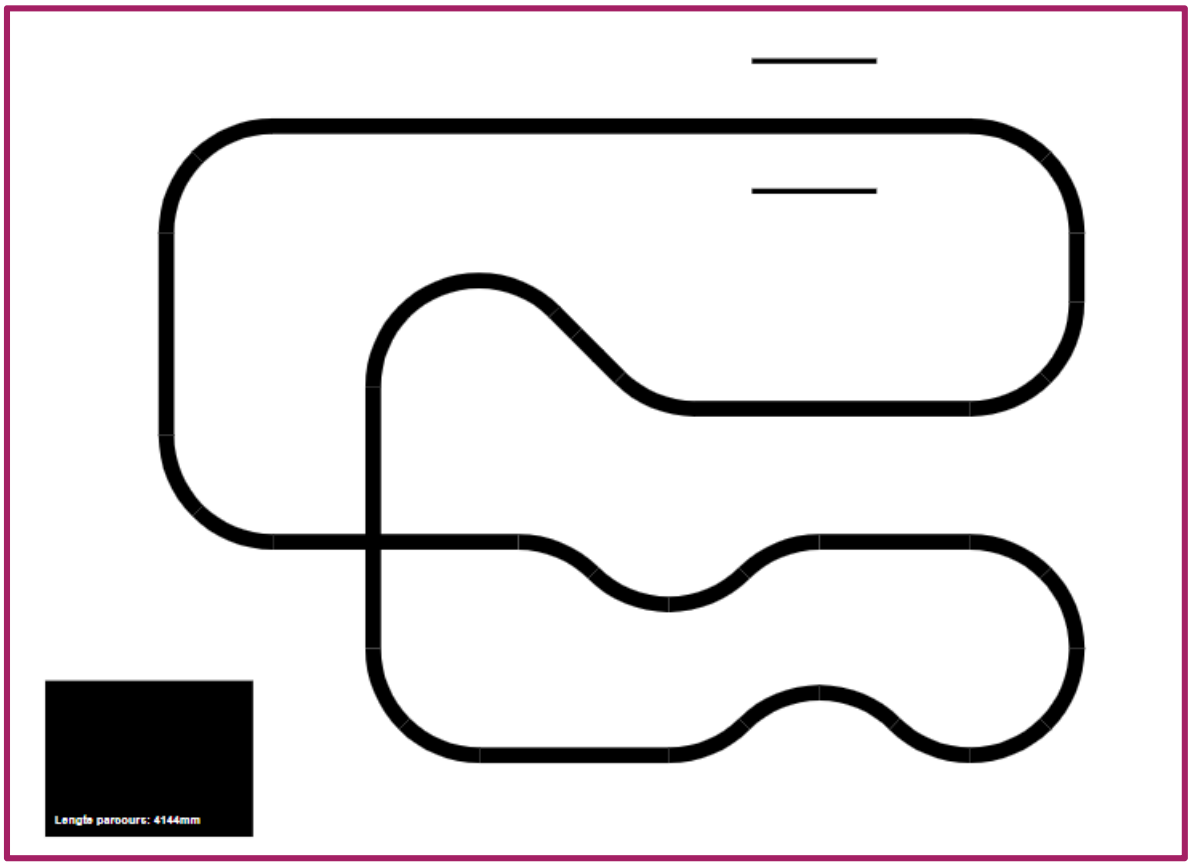
4. Linefollower

Voor een duidelijk beeld te geven van het vision systeem is er een linefollower. Dit autootje werkt volledig autonoom (staat los van de camera). Hij zal een lijn volgen die afgedrukt is op een blad papier (deze lijn is ongeveer 1 cm dik). De bedoeling is dat het vision systeem dit robotje kan volgen. Hiervoor is er een soort deksel ge-3D-print die er bovenop geplaatst wordt. Op dit deksel bevindt zich een driehoek die gevolgd wordt door het vision-systeem. Dit is terug te vinden onder **Vision** op pagina **13**.



Figuur 4.1: Liniefollower concept render

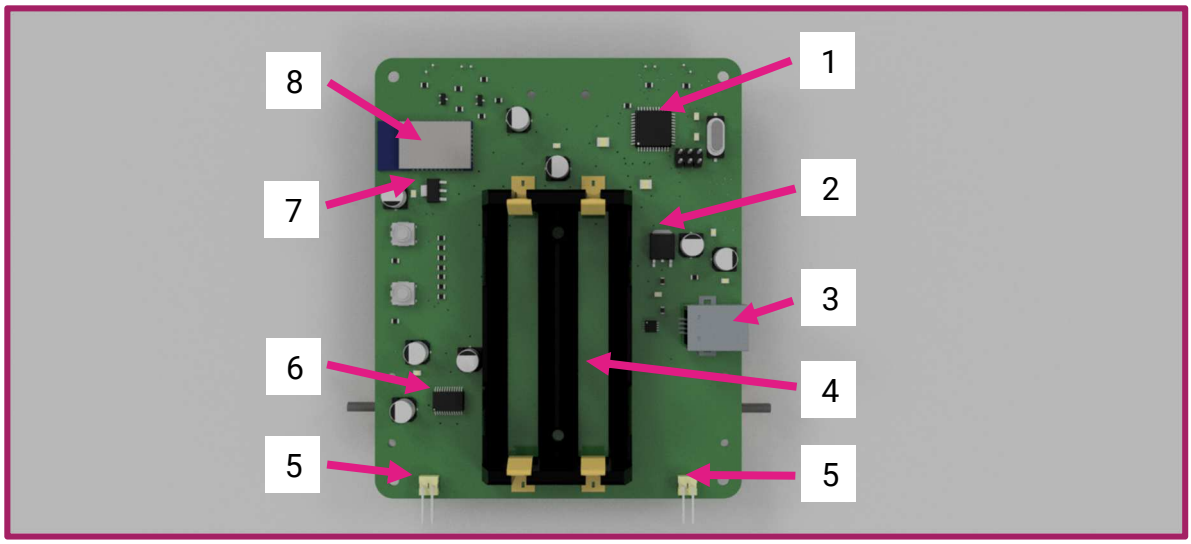
Het parcours die gevolgd wordt is hieronder afgebeeld. Er is een finishlijn voorzien die zal in het vision systeem gebruikt worden voor het opnemen van de rondetijden. De route is afgedrukt op een A0 formaat en aangezien de lengte van de lijn geweten is kan ook de snelheid van het autootje gemeten worden.



Figuur 4.2: Het gebruikte parcours

4.1. Hardware

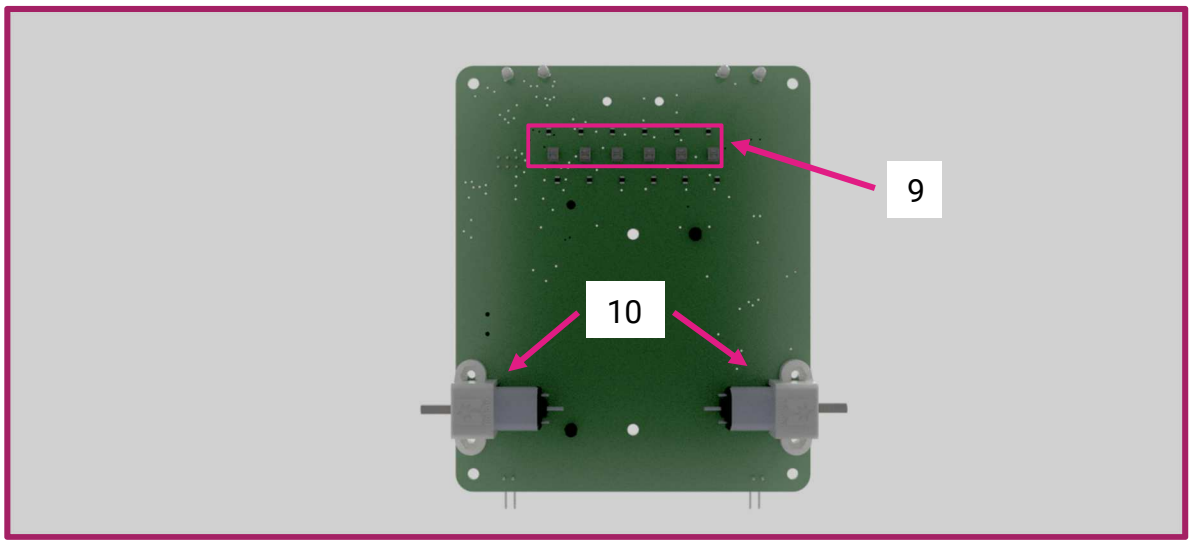
De hardware is een printplaat die een uitgebreide Arduino Leonardo is. Hierbij is een ATMEGA 32U4 chip aanwezig met alle benodigdheden om deze te doen werken als een Arduino. Daarbovenop is er voeding voorzien via twee 18650 batterijen. Deze spanning wordt omgevormd naar 5V en 3,3V. Er is een bluetooth module (RN42 van Microchip) voorzien voor het instellen van de waarden voor de PID-regelaar en andere communicatie met de computer. De motoren worden rechtstreeks vastgemaakt aan de PCB. Deze worden aangestuurd via een motor-driver (TB6612 van Toshiba). Alsook is er een start/stop knop voorzien zodat het niet nodig is om te verbinden met de pc voor het robotje te starten of te stoppen. Als laatste zijn er aan de onderkant zes analoge sensoren voorzien deze zullen de intensiteit van het gereflecteerde licht uitlezen.



Figuur 4.3: Bovenkant van de linefollower

1	ATMEGA 32U4 Arduino Leonardo chip
2	Batterij spanning naar 5V omvormen
3	USB-poort (type B) voor te programmeren
4	Batterij houder
5	Connectoren voor de motoren
6	TB6612 motor-driver
7	Batterij spanning naar 3,3V omvormen
8	RN42 Bluetooth module
9	6 analoge sensoren
10	Motoren

Tabel 4.1: Onderdelen bovenkant linefollower



Figuur 4.4: Onderkant van de linefollower

4.2. Software

Het programma zal de analoge waarde van die hij inleest van de sensoren in procenten omzetten (hoe wit of zwart de ingelezen waarde is). Hiervan nemen we dan het gewogen gemiddelde door de lengte van de sensor tot het middelpunt van de linefollower te gebruiken. Zodat de positie van de lijn gekend is. Deze waarde wordt dan gebruikt in de PID-regelaar voor het aansturen van de motoren. Meer informatie over de PID-regelaar kan gevonden worden onder **PID** op pagina **80**. Het volledige programma is real-time gemaakt dit is nodig voor de I-term in de PID.

Het programma kan pas goed werken als de uitgelezen sensorwaarden kunnen vergeleken worden met een gekalibreerde waarde. Deze wordt verkregen door de sensoren een volledig wit en een volledig zwart stuk uit te lezen, deze worden dan ook opgeslagen. Zo kan ervoor gezorgd worden dat zelf met een verschillende lichtbron, mits herkalibreren, de lijn correct wordt gedetecteerd.

```
void onCalibrate()
{
  char* arg = sCmd.next();

  if (strcmp(arg, "white") == 0){
    for (int i = 0; i < 6; i++) {
        params.white[i] = analogRead (channel[i]);
    }
    SerialPort.println("Calibrated white values");
  }
  else if (strcmp(arg, "black") == 0){
    for (int i = 0; i < 6; i++) {
        params.black[i] = analogRead (channel[i]);
    }
    SerialPort.println("Calibrated black values");
  }
  else {
    SerialPort.println("calibrate <arg>");
  }
}
```

Code 4.1: Kalibreren van de sensoren

De sensoren worden dan een voor een uitgelezen en vergeleken met de opgeslagen waarden. Maar de sensorwaarde wordt in procent uitgedrukt (hierbij is 0% volledig gekalibreerd wit en 100% volledig gekalibreerd zwart). Dit is normaliseren van de sensorwaarden.

```
// reading out all the sensors
for (byte i = 0; i < 6; i++){
    value[i] = analogRead(channel[i]);
}

// comparing the values read by the saved values and
// setting them in a %
for (byte i = 0; i < 6; i++){
    normalised[i] = map(value[i],
                        params.black[i],
                        params.white[i], 100, 0);
}
```

Code 4.2: Uitlezen en normaliseren

Na het normaliseren zal er een gewogen gemiddelde genomen worden. Dit door de genormaliseerde waarden te vermenigvuldigen met hun respectievelijke lengte weg van het middelpunt. Waarna ze allemaal opgeteld worden met elkaar en gedeeld door de som van de genormaliseerde waarde. Hierdoor verkrijg je de afstand van de lijn naar het middelpunt van het autootje. Deze kan dan gebruikt worden voor het sturen van de motoren.

```
//calculating the weighted average
average = ((-225 * normalised[5]) +
           (-135 * normalised[4]) +
           (-45 * normalised[3]) +
           (45 * normalised[2]) +
           (134 * normalised[1]) +
           (225 * normalised[0]));
average /= (normalised[5] +
            normalised[4] +
            normalised[3] +
            normalised[2] +
            normalised[1] +
            normalised[0]);
```

Code 4.3: Gewogen gemiddelde

Het berekenen van de output waarde van de PID kan ingedeeld worden door P, I en D. Voor de foutwaarden kunnen berekent kunnen worden gaan we opzoek naar de error. Deze wordt berekend door de setpoint van de linefollower te nemen en het gewogen gemiddelde ervan af te trekken. In dit geval is het middelpunt nul

aangezien de afstanden van de sensoren tot de setpoint naar de ene kant positief zijn en naar de andere kant negatief.

```
// saving the last error for use in the D-term
prevError = error;

// calculating the error
error = 0 - average;
```

Code 4.4: Error berekenen

Voor de P-term te berekenen is het zeer gemakkelijk, we nemen de error waarde en vermenigvuldigen hem met de ingestelde K_p-waarde. Dit betekent dat hoe groter de K_p is hoe meer impact deze zal hebben op het draaien.

```
//P of the PID
Pterm = error * params.kp;
```

Code 4.5: P-term berekenen in Arduino

De I wordt berekend door de ki-waarde op te tellen met de *error* vermenigvuldigen met de cycle-time gedeeld door 1000. Dit kijkt dus hoe lang het autootje zich weg van de lijn bevindt. Ook hier zal de K_i meer invloed hebben als deze groter is. De I-term van de PID is vooral een hoognodige waarde als er lange rechte stukken zijn. Bij veel bochten kan er zelf zonder I-waarde gewerkt worden. Er wordt ook gezorgd dat de I-term niet groter dan 255 of kleiner dan -255 zal worden.

```
/I of the PID
Iterm += error * (params.cycleTime / 1000.0);

//Iterm ceiling
if (Iterm > 255){
Iterm = 255;
}
if (Iterm < -255){
Iterm = -255;
}
```

Code 4.6: I-term berekenen in Arduino

Als laatste heb je de D-waarde. Deze maakt gebruik van de vorige error waarde. Deze zal dus kijken of er verbetering is. Deze zal berekend worden door de oude

error van de nieuwe te trekken en daarna alles te vermenigvuldigen met de ingestelde K_d-waarde. En ook hier zal de invloed vergroten als de K_d vergroot.

```
///D of the PID
Dterm = (error - prevError) * params.kd;
```

Code 4.7: D-term berekenen in Arduino

De uitgang van de PID-regelaar wordt verkregen door alle termen op te tellen. Na het berekenen zal er nog gekeken worden als de output niet groter is dan 255 (de maximum PWM-waarde) en -255 (de maximum PWM-waarde in de andere draairichting).

```
// Full PID
output = Pterm + (Iterm * params.ki) + Dterm;

// making sure the output value doesn't go over (under)
// 255 (-255)
if(output > 255){
output = 255;
}
else if(output < -255){
output = -255;
}
```

Code 4.8: Volledige PID-uitgang

Deze nieuwe output waarde wordt voor de ene motor afgetrokken van de ingestelde maximumwaarde en opgeteld voor de andere. Deze snelheden mogen natuurlijk niet sneller zijn dan de maximumsnelheid die het autootje mag rijden dus zal er nog gekeken worden of het al dan niet zo is. Als het groter is dan zal de snelheid gelijkgezet worden aan de maximum PWM.

```
// calculating the left and right speed
SpeedLeft = params.speedPWM - output;
SpeedRight = params.speedPWM + output;

// PWM values can't be greater than the set PWM (max)
// or smaller than -set PWM (min)
if (SpeedLeft > params.speedPWM) {
    speedLeft = params.speedPWM;
}
if (SpeedLeft < -params.speedPWM) {
    SpeedLeft = -params.speedPWM;
}
if (SpeedRight > params.speedPWM) {
    SpeedRight = params.speedPWM;
}
if (SpeedRight < -params.speedPWM) {
    SpeedRight = -params.speedPWM;
}
```

Code 4.9: Snelheid motoren beperken

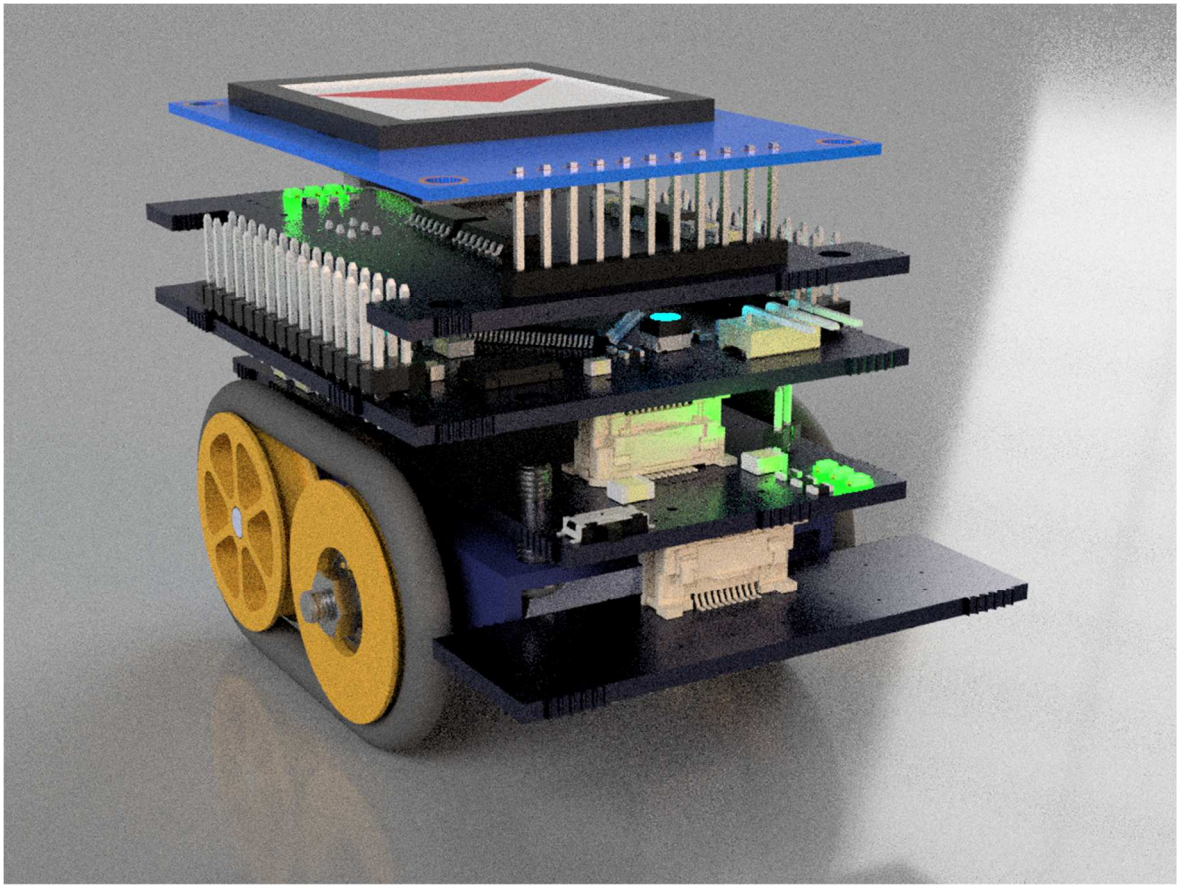
Er wordt dan gekeken in welke richting de motoren moeten draaien. En als laatste worden de waarden weggeschreven naar de motoren.

```
// checking if the motors need to turn back or forward
if (SpeedLeft > 0) {
    digitalWrite(AIN1, LOW);
    digitalWrite(AIN2, HIGH);
}
else {
    digitalWrite(AIN1, HIGH);
    digitalWrite(AIN2, LOW);
}

if (SpeedRight > 0) {
    digitalWrite(BIN1, HIGH);
    digitalWrite(BIN2, LOW);
}
else {
    digitalWrite(BIN1, LOW);
    digitalWrite(BIN2, HIGH);
}

// Writing the analog values to  the motors
// (need to be absolute)
analogWrite(PWMA, abs(SpeedLeft));
analogWrite(PWMB, abs(SpeedRight));
```

Code 4.10: Richting en snelheid wegschrijven

De K_p-, K_i- en K_d-waarde kunnen ingesteld worden via een programma op de pc. Deze zal via de bluetooth de nieuwe waarden doorsturen. Via dit programma kan ook de maximum PWM en cyclustijd ingesteld woorden. Ook de sensoren worden via hier gekalibreerd.

5. Bestuurbaar autootje



Figuur 5.1: Concept render voor het autootje

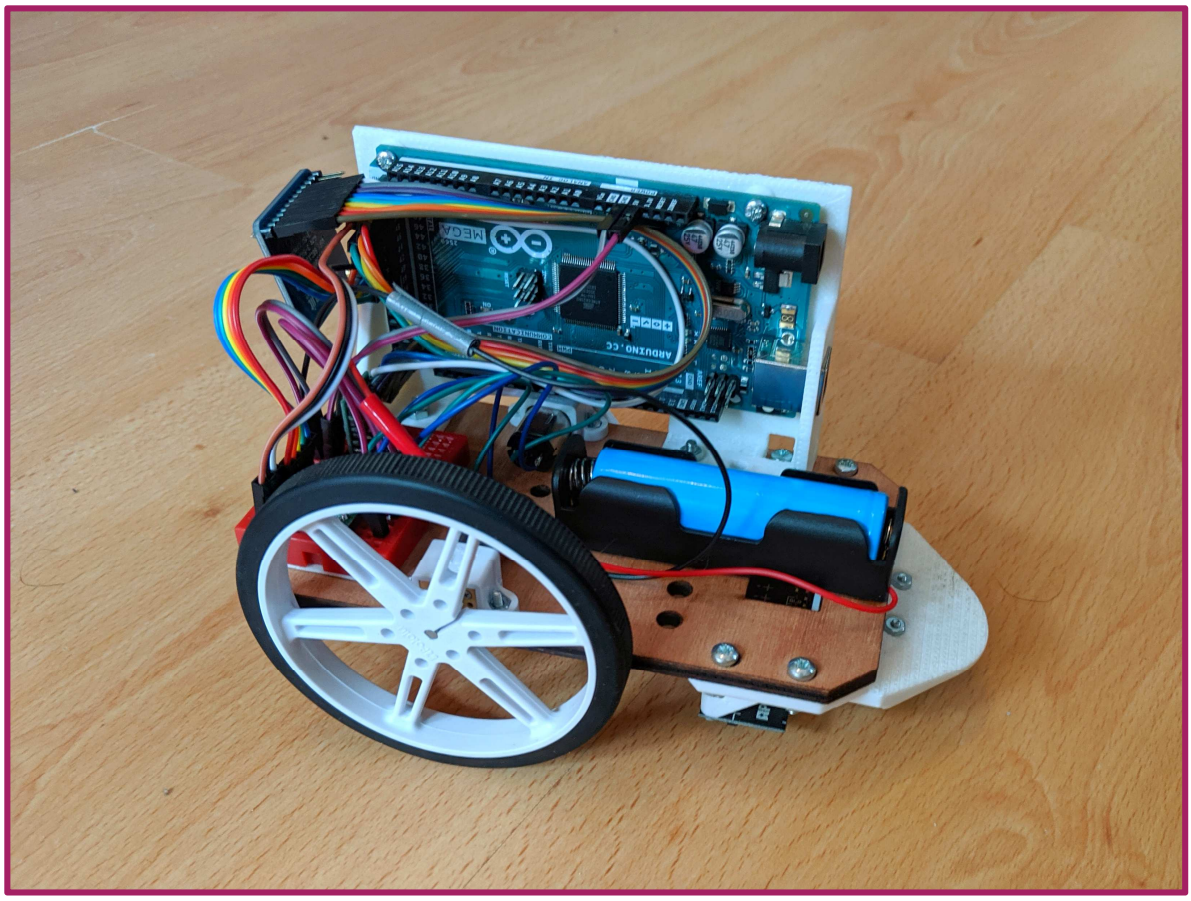
5.1. Prototype

Om vlot van start te kunnen gaan met de ontwikkeling van het autootje, werd een prototype gemaakt. Op deze manier kon reeds van start gegaan worden met de ontwikkeling van de software, zonder te hoeven wachten op de printplaat en onderdelen. Dit bleek een goede beslissing aangezien er grote vertragingen waren op de bestelling[3].

Het prototype was zeer eenvoudig opgebouwd en bestond voornamelijk uit ge-3D-printe onderdelen. De basisplaat, van het chassis, werd uitgesneden op een laser-cutter. De elektronica bestond uit een Arduino MEGA en modules van Adafruit en Pololu.

De Arduino werd verticaal gemonteerd, met behulp van een ge-3D-printe houder. Op deze manier was het mogelijk om de grote Arduino MEGA te monteren en te gebruiken, maar ook de grootte van het autootje te beperken.

[3] Op het moment van schrijven (23 mei 2019) is de bestelling nog niet toegekomen.



Figuur 5.2: Prototype van het autootje

Achteraan werd een klein ***breadbord*** bevestigd. Dit werd gebruikt voor de DC/DC-converter en de motor-driver. Dit liet toe om zeer gemakkelijk de modules en de motoren te verbinden.

Als batterij werd een 18650-ceel gebruikt van een oude laptop-batterij. Een DC/DC-converter module werd gebruikt om de nodige 5-volt te verkrijgen. Deze 5-volt wordt gebruikt om alle modules en motoren te voeden.

De Wi-Fi module werd rechtstreeks aangesloten op de Arduino. Dit was mogelijk aangezien deze ingebouwde ***level-shifters*** en 3,3-volt regulator had.

Dit prototype werd ook gebruikt voor verschillende andere onderdelen van het systeem te testen, zoals de vision en de MQTT.

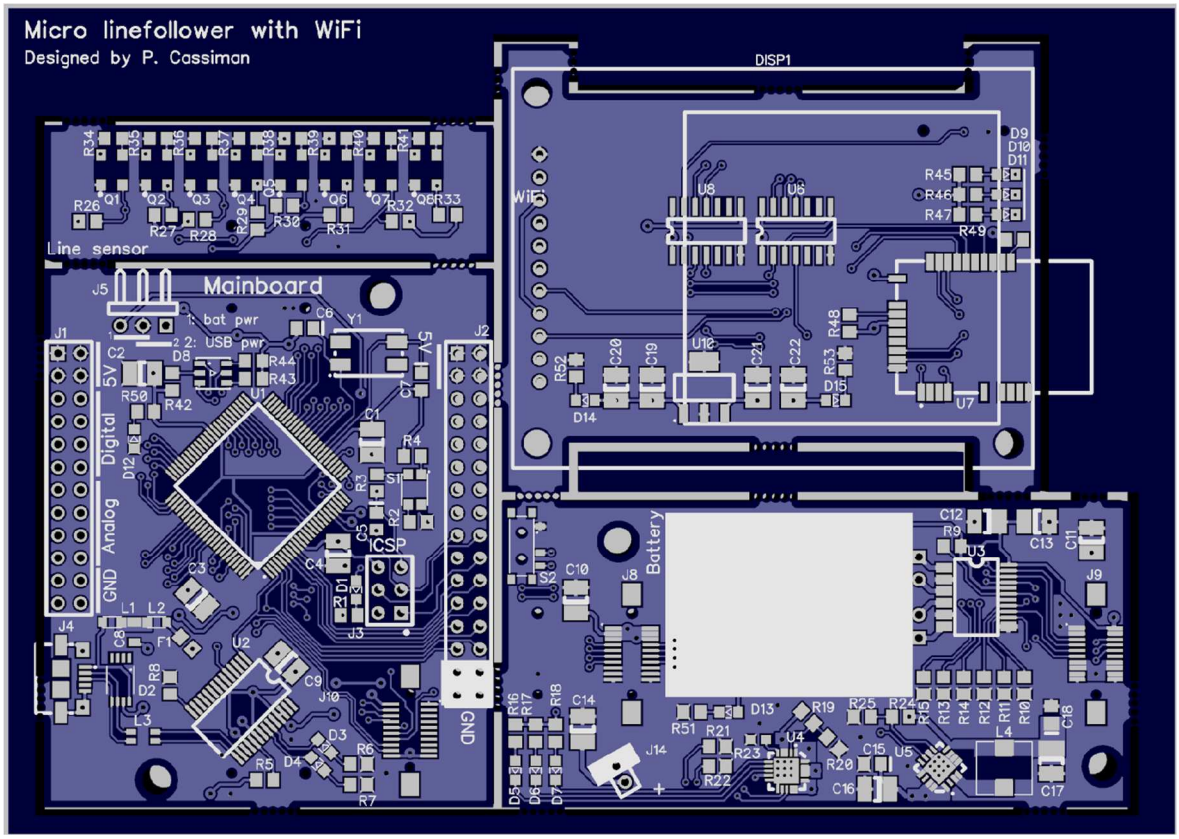
Aan de voorkant van het autootje was ook een sensor-module gemonteerd voor het volgen van een lijn. Deze werd nooit aangesloten en dus ook niet gebruikt.

Op de Maker Faire werd deze sensor-module wel gebruikt om te illustreren dat de autootjes zowel op een volledige printplaat kunnen gemaakt worden, als modulair, zodat de autootjes ook kunnen gebouwd worden met en door kinderen.

5.2. Hardware

Voor het autootje werd een PCB ontworpen. Zoals te zien is in de concept-render, in het begin van dit hoofdstuk, wordt de pcb gestapeld. Door te werken met “verdiepingen” kunnen de autootjes kleiner gemaakt worden. Dit heeft als grootste voordeel dat er meer autootjes op dezelfde oppervlakte kunnen rijden.

Voor het gemak tijdens de assemblage werden de verschillende verdiepingen op één pcb ontworpen. Deze kunnen achteraf los gemaakt en gestapeld worden.



Figuur 5.3: PCB van het autootje

De PCB is 130 op 92 mm. De meeste fabrikanten van PCB’s hebben een minimumprijs voor het maken van printplaten. Deze prijs is voor een printplaat van 100 op 100 mm. Door de verschillende verdiepingen op één printplaat te plaatsen werd ook een grote kost gespaard.

Net zoals het prototype is dit autootje gebaseerd op een Arduino MEGA. Deze werd geïmplementeerd op het ***mainboard***. Alle niet gebruikte pinnen werden uitgebracht naar de twee headers aan de zijkant.

De onderste verdieping bevat de motor-driver en de batterij-lader. Hier is ook een DC/DC-converter terug te vinden, voor het genereren van 5-volt. Hier is ook een aansluiting te vinden voor de module met de analoge sensoren (voor het volgen van een lijn).

De bovenste verdieping bevat de Wi-Fi module. Met de bijhorende ***level-shifters*** en 3,3v-volt regelaar. Op deze verdieping is ook een aansluiting te vinden voor het OLED-scherm.

Het autootje gebruikt hetzelfde principe als een tank voor het sturen. Dit liet toe om de motor gekruist te plaatsen. Eén motor werd vooraan geïnstalleerd, de andere achteraan. Hierdoor kon het chassis veel smaller gemaakt worden.

Door het achterblijven van de bestelling (met de onderdelen), was het niet mogelijk om het autootje af te werken.

Het schema voor deze printplaat is terug te vinden op Github via de volgende link:

https://github.com/pcassima/thesis_hogent/tree/master/design/minicar/electronics/schematics

Alle schema's en printplaten werden ontworpen in ***Diptrace***.

5.3. Software

Het programma die op de autootjes staat is zeer simpel. Het enige dat het doet is zich verbinden met de Wi-Fi en met het MQTT-protocol. Waarna hij zal "abonneren" op het juiste onderwerp. Als er een bericht verschijnt onder dit onderwerp dan zal hij het ook binnen krijgen. Daarna kijkt hij of het wel voor hem bedoeld was. Zo ja dan staat er in de rest van het bericht wat er moet gebeuren. Dit houdt in: vooruit, achteruit, links, rechts en stoppen. De PWM-waarde, snelheid van de motoren, wordt in de payload meegegeven.

Meer informatie over het protocol kan gevonden worden onder **Protocol** op pagina **69**.

In de tabel hieronder is er een samenvatting van welk doel elk functie nummer heeft. Deze kunnen altijd worden uitgebreid als er noot voor is. Het maximumnummer die mogelijk is voor de functie is 255. Als er meer nodig zij kan er gebruik gemaakt worden van de userdata op het einde van de header. Deze is opnieuw een byte en kan dus ook van 0 tot 255 opslaan.

Functie nummer	Functie
0	Stoppen
1	Vooruit
2	Achteruit
3	Links draaien
4	Rechts draaien
5	Links draaien stoppen
6	Rechts draaien stoppen

Tabel 5.1: Doel van de functie nummers

Wanneer er een bericht binnenkomt zal er gekeken worden of het bericht voor dit autootje is. Als dit het geval is dan zal ervoor gezorgd worden dat het niet meerdere keren zal gelezen worden. Daarna kijkt hij wat het doel is van het bericht en indien nodig zal hij de payload uitlezen en gebruiken. In dit geval staat de snelheid in de payload. In het onderstaande voorbeeld is de vooruit actie getoond.

```
// Check if the message is for this car
if (MessageByte[0] == thisCar) {

// Making sure the message is only read ones
MessageByte[0] = 0;

// Checking what the function number is
if (MessageByte[2] == 1) {

    // Set speed to received speed en go forward
    currentSpeed = MessageByte[6];
    Forward(currentSpeed);
}
```

Code 5.1: Bericht vooruit

Hierna wordt een volgende functie aangeroepen. Deze zal ervoor zorgen dat de motoren met de juiste snelheid naar de juiste kant zullen draaien.

```
void Forward(byte PWM) {

  // Left motor turn forward
  digitalWrite(left1, HIGH);
  digitalWrite(left2, LOW);

  // Right motor turn forward
  digitalWrite(right1, HIGH);
  digitalWrite(right2, LOW);

  // Set speed left and right
  setPWM(PWM, PWM);
}
```

Code 5.2: Functie vooruitrijden

5.4. Scherm

Als het vision systeem een autootje wil volgen dan is er op het autootje een herkenningspunt nodig. In dit geval is er een driehoek aanwezig. Deze driehoek

wordt via een blad papier op het dag geplakt. Voor een betere implementatie is er een mogelijkheid voor een scherm te gebruiken. Hier wordt er op elk display een driehoek gezet met een verschillend kleur. Deze kleuren zullen ervoor zorgen dat de autootjes uit elkaar kunnen gehouden worden. Het volgende programma is gebaseerd op het NHD-1.5-AU-SHIELD.ino programma van ***Newhaven Display*** (N.D). Dit is dan ook bedoeld voor het 128 x 128 display van N.D.

Het zal hier enkel gaan over het maken van een driehoek in een bepaalde kleur en niet over de stukken geschreven door N.D.

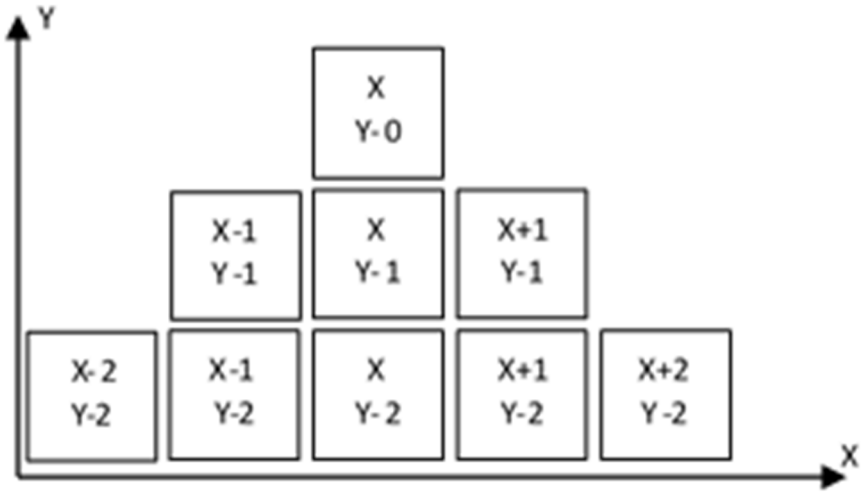
Eerst zullen de kleuren moeten vastgelegd worden dit is hier in BGR en niet in het gebruikelijke RGB. Ook moet de achtergrond wit zijn zodat de camera goed het verschil zal zien.

```
#define    RED  0x0000FF
#define  WHITE  0xFFFFFF

//deze staat in de setup
OLED_FillScreen(WHITE);
```

Code 5.3: Kleuren

Voor een klein programmatje te kunnen schrijven en niet elke pixel te ***hard-coden***[4] wordt er gebruik gemaakt van twee for-loops. De eerste zal kijken voor de X-coördinaat en de tweede kijkt voor de Y-coördinaat maar ook voor de lengte van het lijntje. Dus het werkt door een eerste pixel te kleuren en daarna één pixel naar beneden te gaan en links en rechts ook een bij te kleuren. De lengte zal dus met elke stap naar beneden twee pixels vergroten. Hieronder een verduidelijking.

[4] Hard-coded is de praktijk waarbij bepaalde dingen volledig worden geprogrammeerd, in plaats van bijvoorbeeld in lus te gebruiken



Code 5.4: Driehoek verduidelijking

Voor de driehoek te tekenen worden een aantal waarden doorgegeven. Deze zullen ervoor zorgen dat de driehoek op de juiste plaats en met de juiste kleur wordt getekend. Hiervoor is het bovenste coördinaat, onderste coördinaat, onder welke hoek en de kleur nodig. Hieronder is enkel 0° weergegeven.

```
void MakeTriangle(unsigned long color,
                  byte upperCoor,
                  byte lowerCoor,
                  byte flip)
{
  // flip 1: 0°
  if (flip == 1) {
    //coor of the vertical starting place
    byte lengthHor1 = 0;

    for (int y = lowerCoor; y >= upperCoor; y = y - 1) {

      //vertical
      OLED_SetRowAddress( y, y);

      //length of the horizontal line
      for (int i = (midPoint - lengthHor1);
           i <= (midPoint + lengthHor1);
           i++) {

        OLED_SetColumnAddress( i, i); //horizontal
        OLED_WriteMemoryStart();
        OLED_Pixel(color);
      }
      lengthHor1 += 1;
    }
  }
}
```

Code 5.5: Driehoek tekenen

6. Controllers

6.1. Toetsenbord

Voor het besturen van het autootje zijn er verschillende mogelijkheden zo is er een mogelijkheid om via een gewone pc de juiste commando's door te sturen (het moet enkel python kunnen draaien). Het programma werkt door de toetsen die worden ingedrukt uit te lezen. De snelheid staat ook in dit programma en kan door de gebruiker aangepast worden.

Er wordt hier gebruik gemaakt door de Paho bibliotheek. Deze zal alles met de MQTT onder handen nemen. Dit is het stuk code verantwoordelijk voor het verbinden met de MQTT-broker en ***subscriben*** op de juiste topic.

```
# set mqttc as the client
mqttc = mqtt.Client()

# setting the mqtt broker adress and port
mqttc.connect("192.168.1.5", 1881)

# subscibing to the topic
mqttc.subscribe("vroom")

# starting the mqtt loop
mqttc.loop_start()

# show that the program is ready
print("ready")
```

Code 6.1: Verbinden met de MQTT-broker

Er wordt voor gezorgd dat het programma in een ***loop*** terecht komt zodat het niet eenmalig wordt doorlopen en dan stopt. Er wordt dan continue gekeken of er een toets wordt ingedrukt of losgelaten. Onderstaande tabel toont aan wat de toetsen doen.

Z	Vooruit
S	Achteruit
Q	Links
D	Rechts
U	Start de aftelling
K	Lichten uit (zie *lights*)
L	Blokkeer besturing en lichten uit

Tabel 6.1: Functie van de toetsen

Als er een toets tussen zit die een betekenis heeft (zie tabel) dan zal er gekeken worden of de toets voor de eerste keer wordt ingedrukt. Dit om te voorkomen dat het bericht meerdere keren zal worden doorgestuurd. Maar als vooruit ingedrukt is kan achteruit niet ingedrukt worden en omgekeerd. Dit is ook zo bij links en rechts.

```
# button z (forward)
if letter == "z":

    # checking if z or s is already pressed
    if z_pressed == False:
        if s_pressed == False:

            # setting the destination and the speed
            arr[2] = 1
            arr[6] = 200

            # send the array
            mqttc.publish("vroom",  arr)
```

Code 6.2: Vooruit sturen

Als een knop wordt losgelaten (ook hier wordt gecontroleerd of het de eerste keer is dat hij losgelaten wordt) dan zal er een stop bericht moeten gestuurd worden. Deze zijn verschillend van welke toets er wordt losgelaten.

```
# check if z is pressed
if z_pressed == True:

    # button z released
    if letter == "z":

        # setting the destination and the speed
        arr[2] = 0
        arr[6] = 0

        # send the array
        mqttc.publish("vroom",  arr)
```

Code 6.3: Vooruit lossen

De besturing kan ook geblokkeerd worden en via buitenaf terug gedeblokkeerd worden. Dit gebeurt via het ***lights***-programma. Dit programma wordt in detail besproken onder **Lights** op pagina **73**.

Als er een bericht toekomt op de MQTT-broker die voor de besturing bedoeld is dan zal dit uitgelezen worden en indien nodig wordt de besturing geblokkeerd of gedeblokkeerd. Dit moet gestart worden via de toets “U”.

```
# button u (start lights)
if letter == "u":

    # check if already pressed
    if u_pressed == False:

        # setting the target, function and delay
        arr[0] = 21
        arr[2] = 1
        arr[6] = 225

        # send the array
        mqttc.publish("vroom",  arr)
```

Code 6.4: Zenden naar lights

```
# if the function is 1
elif message.payload[2] == 1:

    # enable driving
    drive_enabled = True
```

Code 6.5: Besturing deblokkeren bij het juiste bericht

Als de lichten uit gezet worden, via de toets “K”, dan zal er nog altijd gereden kunnen worden. Maar wanneer er op de toets L gedrukt wordt dan zullen de lichten uitgaan (indien deze nog aan waren) en zal ook het autootje stoppen met rijden waarbij ook de besturing weg valt.

```
# button k (lights out)
elif letter == "k":

    # check if already pressed
    if k_pressed == False:

        # setting the target, function and delay
        arr[0] = 21
        arr[2] = 0
        arr[6] = 0

        # send the array
        mqttc.publish("vroom",  arr)
        k_pressed = True

# button l (lights out and disable driving and stop car)
elif letter == "l":

    # check if already pressed
    if l_pressed == False:

        # setting the target, function and delay
        arr[0] = 21
        arr[2] = 0
        arr[6] = 0

        # send the array
        mqttc.publish("vroom",  arr)

        # setting the target, function and speed
        arr[0] = 20
        arr[2] = 0
        arr[6] = 0

        # send the array
        mqttc.publish("vroom",  arr)

        # disable driving input
        l_pressed = True
```

Code 6.6: Lichten uit en blokkeren besturing

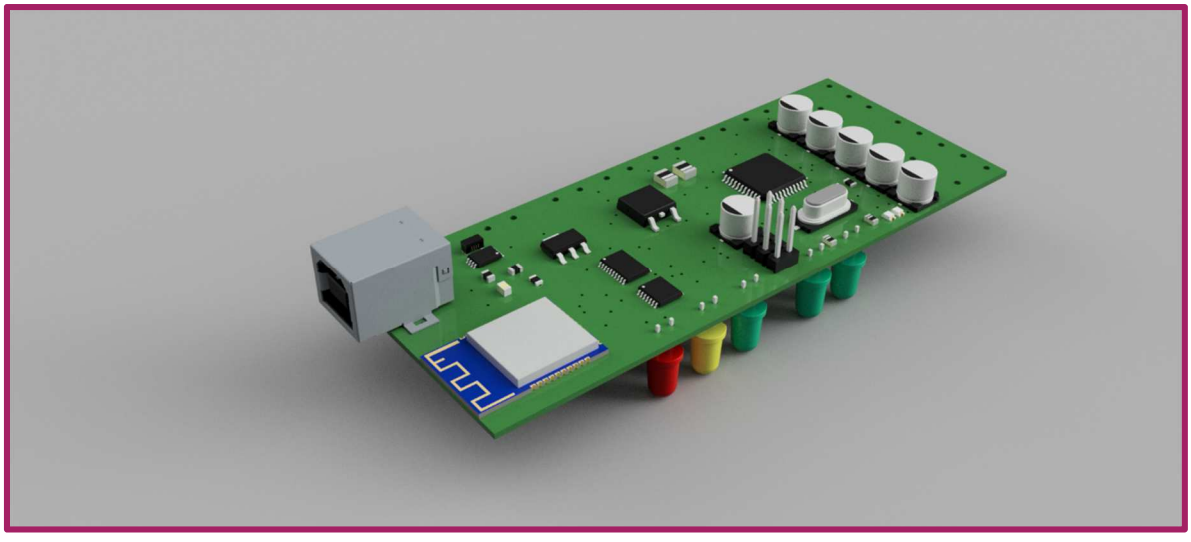
Als laatste is er ook een mogelijkheid nodig voor het programma te stoppen. Dit gebeurt door op de toets “P” te drukken. Deze zal voor het programma stopt eerst een bericht sturen naar het autootje. Deze zal door dit bericht stoppen met rijden voor dat de connectie met de besturing weg valt.

```
# button p (to kill the program)
if letter == "p":

    # setting the target, function and speed
    arr[0] = 20
    arr[2] = 0
    arr[6] = 0

    # send the array
    mqttc.publish("vroom",  arr)

    # print out exiting and then leave the program
    print('exiting')
    exit()
```

Code 6.7: Het programma stoppen

6.2. Joystick

Een tweede optie voor het besturen van het autootje is een joystick. Dit maakt niet uit welke soort het is. Voor de analoge waarden van de joystick uit te lezen is er een stuk hardware aanwezig. Dit is een smalle printplaat met een aantal extra’s.

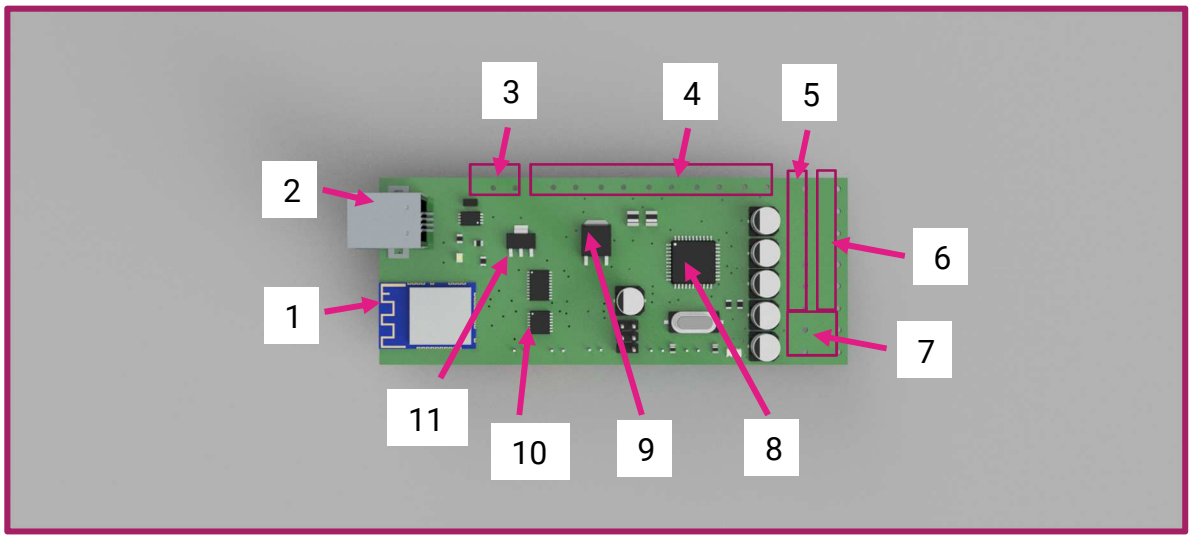


Figuur 6.1: Concept render voor de joystick PCB

6.2.1. Hardware

Aangezien het een smalle PCB is moet er ook een kleine microcontroller aanwezig zijn deze is de ATMEGA 32U4 (Leonardo chip). Ook is er verbinding met USB of Wi-Fi aanwezig. Maar vooral een hoop aansluitingen voor knoppen en analoge ingangen. Er zijn ook indicatie ledjes aanwezig.

Door het achterblijven van de bestelling (met de onderdelen), was het niet mogelijk om de joystick af te werken.



Figuur 6.2: PCB voor de joystick

1	ATWINC1500 Wi-Fi-module
2	USB-poort (type-B)
3	Ingang voor de spanning
4	Aansluiting voor de knoppen
5	X en Y richting voor de joystick
6	Extra analoge ingangen
7	UART van de Wi-Fi-module en Leonardo
8	ATMEGA 32U4
9	5-volt spanningsregelaar
10	Level-shifters voor de Wi-Fi-module
11	3,3-volt spanningsregelaar

Tabel 6.2: Onderdelen joystick PCB

6.2.2. Software

De software is ontworpen voor een Arduino MEGA. Deze heeft meer plaats voor een groter programma. Op de PCB is er geen plaats voor de ATMEGA 2560 (Arduino MEGA) vandaar dat een ATMEGA 32U4 werd gebruikt. Hierdoor is het programma (inclusief de Wi-Fi) verkleint. Wanneer het volledige programma nodig

is kan er altijd een Arduino MEGA met een Wi-Fi-module gebruikt worden in plaats van deze PCB.

Er is ook een mogelijkheid om alles van de MQTT en Wi-Fi via de pc door te sturen. Dit kan door de printplaat te verbinden met een computer via een usb kabel. In het programma moet dan alles via de seriële-poort gestuurd worden. Dit wordt dan uitgelezen en op de MQTT-server gezet.

Nadat er connectie is gemaakt met het Wi-Fi-netwerk en met de MQTT-broker zullen de analoge ingangen continu uitgelezen worden. Deze waarden worden dan omgevormd naar een bepaalde snelheid. Wanneer de snelheid naar boven gaat zal hij later schakelen dan naar het beneden gaan dit om te voorkomen dat er continu zal wissel van snelheid.

```
int val = analogRead(A4);
//to Speed -255
if (val < 50 && val < analog_Y_val && analog_Y_val != 50) {
//Serial.println("Speed: -255");
    analog_Y_val = 50;
    speed_val = -255;
}
```

Code 6.8: Uitlezen en omvormen van de joystick positie

De teruggekregen waarde ligt dan tussen -255 en 255 (achteruit en vooruit). Deze worden dan absoluut verzonden met het juiste functie nummer erbij. Wanneer de snelheid nul wordt zal er een stop commando doorgestuurd worden. Deze code zal enkel worden doorgelopen bij verandering.

```
if (speed_val != prev_speed_val) {
    if (speed_val == 0) {

      // Stopping speed
      sendDataValue[2] = 0;

      // Publish
      client.publish("vroom", sendDataValue, 8);
    if (speed_val < 0) {

      // Set action to back and set pos speed
      sendDataValue[2] = 2;
      sendDataValue[6] = abs(speed_val);

      // Publish
      client.publish("vroom", sendDataValue, 8);
    }
}
```

Code 6.9: Snelheid doorsturen

Voor het draaien zal hetzelfde gebeuren. Het autootje zal dan aan de hand van het functie getal moeten kijken wat er moet gebeuren.

7. Link

7.1. Protocol

MQTT werkt al met een eigen protocol maar aangezien er meerdere apparaten het netwerk zitten is er een protocol nodig die ervoor zorgt dat het juiste bericht door het juiste apparaat wordt uitgelezen.

7.1.1. Pegasus

Dit protocol werkt door alle data die verzonden moet worden om te vormen naar bytes deze worden samen met een header en een ***tail*** verzonden. De minimumlengte van een bericht is 8 bytes en het maximum is 255 bytes. Men kan ook langere berichten zenden maar deze zullen dan worden gesplitst.

Bij ontvangst van het volledige bericht zal eerst de header worden gelezen waarin zal vermeld staan of het bericht bedoeld is voor dit apparaat of niet. Dan zal het de beslissing nemen of het verder gelezen kan worden of als het bericht mag vergeten worden.

Zoals eerder vermeld wordt alle data die zal worden doorgestuurd in bytes gezet. Dit wordt gedaan omdat de verwerkingstijd van bytes veel kleiner is dan bijvoorbeeld strings. Als je bijvoorbeeld het cijfer 200 neemt dan kan in een byte het volledige cijfer (bytes slaan cijfers op van 0 tot 255) maar als we ditzelfde cijfer opslaan als string dan gaat deze drie bytes innemen. Dit kan er dus voor zorgen dat de lengte van het doorgestuurde bericht kan ver driedubbelen door het datatype aan te passen.

7.1.1.a. Header

De header is het begin van het bericht, deze geeft info door naar het ontvangende apparaat. Deze bevat een aantal belangrijke en een aantal minder belangrijke gegevens.

Hieronder ziet u hoe de header wordt doorgestuurd en wat deze bytes zijn in gewone cijfers. Ook is de plaats waar de byte zich bevindt in het volledige bericht weergegeven.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
0001 0100	0011 1100	0000 0001	0000 0000	0000 1011	0000 0000
20	60	1	0	11	0

Tabel 7.1: Structuur van de header

Elke byte in de header heeft een doel deze worden in volgend tabel verduidelijkt.

Byte 0	Het doel, voor wie het bericht bestemd is.
Byte 1	De zender, van wie het komt.
Byte 2	De functie, wat wil het bericht doen.
Byte 3	Het kanaal, dit is een aanvullen van de functie.
Byte 4	Het datatype, wat is het datatype is van de payload.
Byte 5	Userdata, een lege byte die kan ingevuld worden door de gebruiker.

Tabel 7.2: Detail van de header

Sommige van deze bytes kan men vrij invullen als de waarde maar tussen 0 en 255 ligt. Maar een aantal hebben een bepaalde vast gelegde waarde.

Byte 0 en byte 1 kunnen nooit dezelfde waarde hebben anders wil dit zeggen dat er een bericht naar hetzelfde apparaat wordt gestuurd. De namen die kunnen aangenomen worden staan ook vast zodat men direct kan zien voor welk soort apparaat het bericht bedoeld is.

0 – 9	Staan vast geen gebruik mogelijk door de gebruiker
10 – 19	Systeem
20 – 59	Bestuurde elementen (vb. Autootjes)
60 – 99	Besturingselementen (vb. Toetsenbord)
100 - 255	Vrij voor gebruik

Tabel 7.3: Adres ruimte voor het Pegasus protocol

Byte 4 moet ook een bepaalde waarde hebben. Deze gaat afhangen van wat men doorstuurt aangezien hij het datatype van de payload aanduid. Als men met een ander soort datatype werkt die niet in de volgende lijst te vinden is dan zijn er plaatsen vrij die vrij ingevuld kunnen worden.

0 – 9	Niet in gebruik
10	Char
11	Unsigned char
12	Integer
13	Unsigned integer
14	Word
15	Long
16	Unsigned long
17	Float
18	Double
19	String
20 – 255	Niet in gebruik

Tabel 7.4: Datatype waardes

Deze zullen dan gebruikt worden door het programma om te kijken wat de binnengekomen bytes naar moeten gevormd worden.

7.1.1.b. Payload

De payload is het bericht dat door de zender wordt meegegeven. Aangezien het datatype anders kan zijn zal ook de lengte van de payload veranderen. De lengte

zal wel liggen tussen 1 en 248 bytes. Als deze langer is dan dit dan zal het bericht moeten gesplitst worden en zal er moeten gebruik gemaakt worden van de ***tail***.

Aantal bytes	Datatypes
1	Char en Unsigned char
2	Integer, unsigned integer en word
4	Long, unsigned long en float
1 – n	String

Tabel 7.5: Payload lengte voor elk datatype

7.1.1.c. Tail

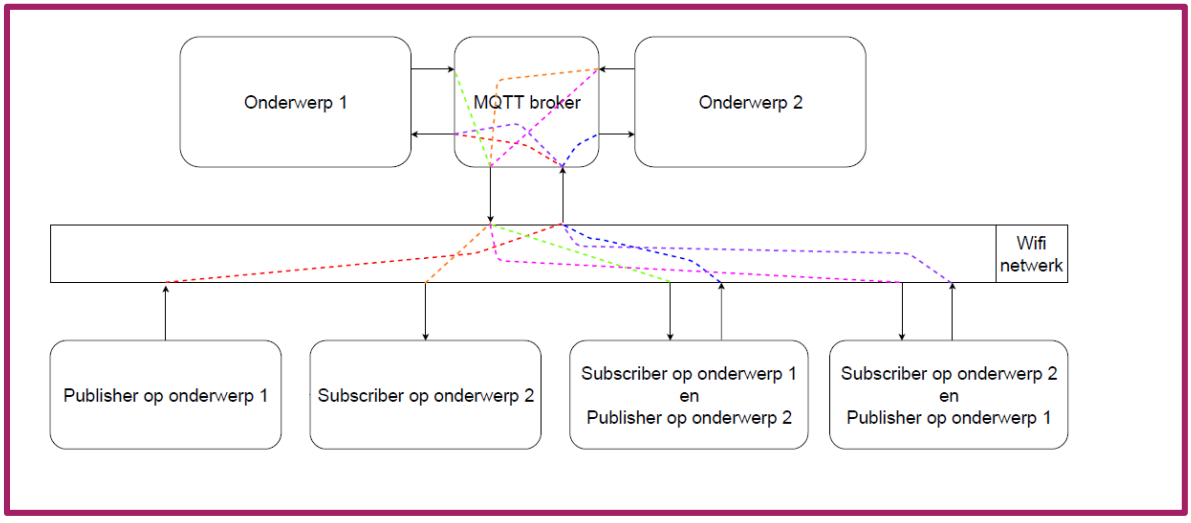
De ***tail*** is de allerlaatste byte van het bericht. Deze zegt tegen het apparaat wat er moet gebeuren na het bericht. Als dit alles is of als er iets anders moet gebeuren dit zorgt ervoor dat men geen twee berichten moet sturen voor nog iets kleins.

0	Niets
1	Ontvangstbewijs sturen
2	Continue: de payload gaat verder in het volgende bericht
3 – 255	Andere mogelijkheden

Tabel 7.6: Waardes voor de tail

7.1.2. MQTT

Message Queuing Telemetry Transport of simpeler MQTT is een bericht protocol die ***publishers*** en clients heeft. Dit is bedoeld voor over een netwerk met elkaar te communiceren. Aan het hoofd van het MQTT-protocol staat een bericht broker. Deze zal elk bericht die binnenkomt onder het juiste onderwerp (topic) zetten die is meegegeven met het bericht. De zenders van deze berichten worden de ***publishers*** genoemd. Hierbij zijn er ook clients deze zullen zich abonneren op een bepaald onderwerp en zo elk bericht die daarop binnenkomt ook lezen. Een ***publisher*** kan ook een client zijn.



Figuur 7.1: Voorbeeld van een MQTT-netwerk

Link

In bovenstaande afbeelding is te zien dat een ***publisher*** of een ***subscriber*** niet samen moeten staan dit mag op een verschillend apparaat staan. Wanneer een apparaat zowel een ***publisher*** als een ***subscriber*** is dan moet dit niet op hetzelfde onderwerp zijn. Ook is het mogelijk om meerdere onderwerpen aan te maken op de broker.

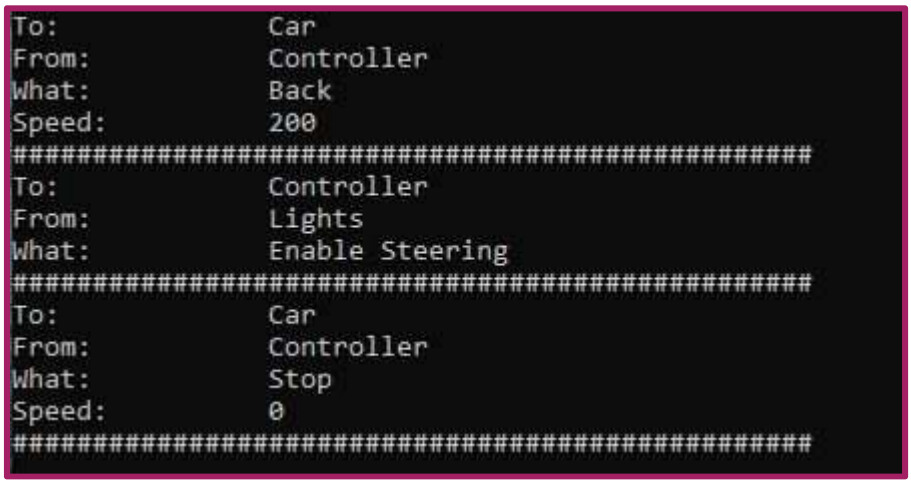
7.1.3. Reader

Voor alle berichten op een bepaalde topic te lezen en te ontcijferen is er een reader. Dit programma, geschreven in python, leest alle berichten op het onderwerp binnen en ontcijferd deze. Het kijkt voor wie het bedoeld is en naar wie het gaat. Maar ook wat de bedoeling is hiervoor.

Eerst zal er verbinding gemaakt worden met het MQTT-netwerk. Hierna zal elk bericht binnen gelezen worden en byte voor byte ontcijfert.

```
# check for who the message is
if message.payload[0] == 20:

    print("To:      Car")

elif message.payload[0] == 21:

    print("To:      Lights")

elif message.payload[0] == 60:

    print("To:      Controller")

elif message.payload[0] == 61:

    print("To:      Vision")
```

Code 7.1: Bestemming bekijken



Figuur 7.2: Voorbeeld van een gedecodeerd stuk

7.2. Lights

Het volledige systeem kan over een netwerk aangestuurd worden. Hiervoor is het Pegasus protocol ontworpen deze wordt samen met MQTT gebruikt in een programma. Het ***lights***-programma is een Arduino-programma waarin het volledige communicatiesysteem wordt uitgelegd.

Eerst en vooral heb je een netwerk nodig (via Wi-Fi of ethernet). Het programma zal er dan voor zorgen dat, in dit geval, de microcontroller verbonden is met het netwerk. Dit is hetzelfde netwerk dat de broker mee is verbonden. Deze zal alle binnen gekomen berichten in een ***topic*** zetten.

```
// Start WiFi connection
status = WiFi.begin(ssid, password);

// Connecting to the WiFi (for debug)
Serial.print("Connecting to WiFi");

// While not connected to WiFi
while (WiFi.status() != WL_CONNECTED) {

// Wait 500 ms
delay(500);

// Connecting (for debug)
Serial.print(".");
}
```

Code 7.2: Verbinden met een Wi-Fi netwerk

Wanneer de microcontroller verbonden is met het netwerk zal hij verbinding moeten maken met de broker. Wanneer deze verbonden is kan hij ***subscriben*** op

een bepaalde ***topic***. Dit betekent dat wanneer er een bericht binnen komt (dit maakt niet uit voor wie het is) de microcontroller deze ook binnen leest via de ***callback***-functie.

```
// Set mqtt server (port) and calback
client.setServer(mqttServer, mqttPort);
client.setCallback(callback);

// While not connected to mqtt
while (!client.connected()) {

    // Wait 500 ms
    delay(500);

    // Connecting (for debug)
        Serial.println("Connecting to MQTT...");

    // If connected to the mqtt
    if (client.connect(thisObject)) {

        // Connected to the mqtt (for debug)
        Serial.println("Connected");

        // Wait 200 ms
        delay(200);
    }
}
```

Code 7.3: Verbinden met een MQTT-broker

```
// Function to read the incomming message
void callback(char* topic,
              byte* payload,
              unsigned int length) {

  // Set every incomming byte to a place in the array
  for (int i = 0; i < length; i++) {
    MessageByte[i] = payload[i];
  }
}
```

Code 7.4: Bericht callback-functie

Deze functie zal elke byte die is binnengekomen opslaan in een array van bytes. Aangezien we werken met het Pegasus-protocol kan men kijken voor wie het

bericht bedoeld was. Als het bericht voor hem is dan zal hij het verder decoderen. Wanneer dit niet het geval is en het bericht bedoeld was voor iemand anders zal het genegeerd worden.

```
if (MessageByte[0] == 21) {
    ...
}
```

Code 7.5: Controlleren van het adres

Het verder decoderen gebeurt door de bytes om te vormen naar cijfers. Deze bekomen cijfers zullen dan ook gebruikt worden voor het aansturen (in dit geval) van de lichtjes. Deze zijn uitgewerkt als een race licht. Dit wil zeggen dat er vier RGB-LEDs zijn verbonden met de microcontroller. Wanneer het start commando gegeven is zullen de ledjes één voor één rood worden, zijn ze allemaal rood dan worden ze groen.

```
Delay = (MessageByte[6] * 4);
```

Code 7.6: Ontvangen van een variabele

Bij het laatste stuurt hij een bericht over het netwerk die gericht is naar de controller (controller toetsen) deze zorgt ervoor dat de stuur toetsen geblokkeerd zijn waardoor er niet kan gestuurd worden. Zodat het bestuurbaar autootje geen valse start kan maken. Het autootje werd in detail besproken onder **Bestuurbaar autootje** op pagina **53**.

```
client.publish(Topic, sendMessage, 8);
```

Code 7.7: Verzenden van berichten over MQTT, in Arduino

Door dit systeem toe te passen op verschillende microcontrollers, pc's of andere besturingselementen die zicht kunnen verbinden met een netwerk kan men op een gemakkelijke en efficiënte manier berichten verzenden naar elkaar.

7.3. Verdere mogelijkheden

Het autootje wordt gedetecteerd door de webcam. Ook is het mogelijk om informatie en commando's naar het autootje terug te sturen.

Op dit moment is het niet zo complex meer om het hele systeem uit te breiden en het autootje aan te sturen gebaseerd op wat de camera ziet. Het is ook zonder problemen mogelijk om het autootje aan te sturen vanuit een spel-omgeving, zoals de virtual-reality omgeving. Als het mogelijk is om berichten te versturen via MQTT is het mogelijk om informatie uit te wisselen met het autootje en het computer-vision systeem. Enkele ideeën worden aangekaart onder het **Besluit** op pagina **79**.

8. Maker Faire Gent 2019

Als deel van onze stage en bachelorproef hebben we ons project gepresenteerd op de Maker Faire te Gent, onder de naam van ***Makers for Education***.

Dit was een goede kans om ons project en de verschillende systemen te demonstreren en te testen. De systemen hebben het zeer goed uitgehouden, maar er kwamen enkele bevindingen naar boven.

8.1. Opstelling

De opstelling voor de Maker Faire bestond uit twee delen. Er waren twee demonstraties die getoond konden worden.

Het eerste deel van de opstelling bestond uit het bord met de webcam en het parcours voor de linefollower.

Het tweede deel waren de computers waarop de software draaide. Er was één computer die de vision bestuurde en een tweede computer die een toetsenbord had dat de bezoekers konden gebruiken voor het besturen van een autootje.

De eerste demonstratie was een linefollower die autonoom de lijn kon volgen. De tweede demonstratie was een autootje dat de bezoekers konden besturen via een toetsenbord dat aan één van de computers hing. De toetsaanslagen werden op de computer ingelezen en vervolgens via ons protocol, over MQTT, doorgestuurd naar het autootje.

Beide autootjes werden langs het hele traject gevolgd door het vision-systeem. Dat de rondetijden bijhield en na drie rondes het gemiddelde en de snelste tijd weergaf.

Op deze manier konden alle aspecten van deze ***proof-of-concept*** getoond en getest worden. Dit was de eerste keer dat het hele systeem samen getest werd. Het was ook de eerste keer dat het systeem gedurende een lange tijd getest werd. Door de Maker Faire ervaring werd duidelijk welke aspecten er goed werken en welke nog moesten verbeterd worden.

8.2. Bevindingen

Dit zijn de bevindingen die naar boven kwamen gedurende de twee dagen van de Maker Faire.

De opstelling kon ten eerste op zeer veel belangstelling rekenen. Voortdurend stonden mensen te kijken en kwamen er vragen onze richting.

De vision zelf bleek zeer stabiel en betrouwbaar maar bleek zeer gevoelig te zijn voor veranderingen in het omgevingslicht. De opstelling stond dan ook net onder

een skylight. Dat weekend was het weer zeer wisselvallig, waardoor de gevoeligheid van de camera constant moest worden aangepast. De vision was echter tijdens momenten van constante belichten zeer stabiel en betrouwbaar. Hij was ook zeer accuraat en gaf consistente resultaten. Het is dus zeer aannemelijk dat deze manier van positionering zeer goed toe te passen is in een gecontroleerde omgeving zoals een klaslokaal of afgesloten ruimte.

Er zou een functie moeten geschreven worden die de gevoeligheid van de camera automatisch (en continu) aanpast aan de hand van de gemiddelde intensiteit van het beeld.

Het vision-systeem draaide zeer vlot en betrouwbaar op een gewone computer. Dit is een verdere indicatie dat CV-systeem een betrouwbare manier is voor lokalisatie op een relatief kleine oppervlakte.

Op de Maker Faire waren er heel veel Wi-Fi netwerken aanwezig waardoor de verschillende banden oververzadigd waren en er storingen en vertragingen werden ondervonden. Het is dus best zo veel mogelijk te verbinden via kabel. De router die gebruikt werd had hier ook enorm veel last van en moest een aantal keren gereset en opnieuw ingesteld worden gedurende het weekend.

Het bestuurbaar autootje was enorm populair bij het jongere publiek. Dit is dus een enorm goede manier om de opleiding voor te stellen en jonge kinderen in contact te brengen met technologie en programmeren en goed te gebruiken in de context van STEM-onderwijs. Met enkele aanpassingen zou deze opstelling een zeer goede kandidaat zijn voor workshops in het lager en middelbaar onderwijs. Het autootje moet zelfs niets speciaals zijn, gewoon de Arduino, twee wielen en enkele LEDs.

Het toetsenbord bleek niet direct de meest geschikte optie om te gebruiken als controller: veel kleinere kinderen hadden immers moeite hadden met het onthouden welke letters de autootjes bestuurden. Dit werd opgelost door het toetsenbord te voorzien van stickers met pijltjes. Ook werd de lay-out van het toetsenbord vaak veranderd, per toeval, doordat er verschillende kinderen willekeurig toetsen aan het induwen waren.

De Arduino MEGA met een Wi-Fi module is misschien niet de meest geschikte keuze voor deze toepassing, aangezien de berichten vrij snel moeten doorgegeven worden en er ook snel op gereageerd moet worden. Het autootjes kwam vaak zichtbaar na op de gegeven commando’s. Een ESP8266 zou hiervoor een betere optie zijn. Dit is een 32-bit microcontroller met een hogere kloksnelheid en ingebouwde Wi-Fi. Een ESP32 zou nog beter zijn aangezien de kloksnelheid nog hoger ligt, en deze module ook meer bruikbare I/O pinnen heeft.

De batterijduur van de autootjes was enorm verrassend. Het bestuurbaar autootje reed de hele dag op één enkele 18650 batterij met een capaciteit van +-2000mAh. In het programma is er geen code aangebracht om het stroom verbruik te verminderen. De linefollower reed de hele dag op zijn twee 18650 batterijen.

Als laatste bevinding kwam naar boven dat het bord best zou beschikken over een boord of rand, zodat de autootjes er niet kunnen afrijden. Aangezien niet alle toeschouwers even gekwalificeerde chauffeurs waren, was het soms zeer nipt om de autootjes op te vangen. Als er een boord geplaatst wordt, moet de hele opstelling wel lager geplaatst worden. Nu konden sommige kinderen nog maar net over het bord kijken en zien wat er gebeurde. Een andere optie is natuurlijk een opstapje voorzien. Dit zou ook duidelijker afbakenen wie het autootjes aan het besturen is. Dit opstapje zou ook kunnen verminderen dat er meerdere kinderen op het toetsenbord aan het duwen zijn.

Al bij al was de Maker Faire een zeer goede field-test voor deze ***proof-of-concept***. Het toonde aan dat een CV-systeem perfect te gebruiken is voor lokalisatie op een beperkte oppervlakte, zoals een tafel. Het bracht ook enkele aspecten aan het licht waar we niet direct aan gedacht hadden.

9. Besluit

Ons doel om aan te tonen dat een computer-vision systeem gebruikt kan worden voor het nauwkeurig positioneren van objecten, hebben we zeker gehaald.

De autootjes worden nauwkeurig en betrouwbaar gevolg door het vision-systeem. Ondertussen werd er al wat verder gewerkt aan de software, en zijn enkele van de vermelde problemen reeds weggewerkt.

Desalniettemin zijn de autootjes niet afgeraakt waardoor er toch een gevoel van “het is niet af” blijft hangen. We hopen dat de bestelling nog toekomt en dat we autootjes alsnog kunnen af werken.

9.1. Verder

Dit zijn enkele ideeën die wij hebben voor het systeem verder te ontwikkelen. Dit zijn ook voornamelijk ideeën voor het hele systeem te demonstreren en zijn dus voornamelijk spellen.

Als eerste is er Pac-Man. Dit was ook ons eerste idee naar de Maker Faire toe. Echter door het achterblijven van de bestelling zijn we hier niet aan te geraakt. Als de bestelling nog toekomt, zouden we hier nog aan verder werken en Pac-Man uitwerken.

Bomber was een ander spel dat leuk leek om uit te werken. Vooral om dat dit zeer goed te spelen is met meerdere personen. Dit spel zou ook zeer goed gebruik kunnen maken van de geplande RGB-matrix[5] in het bord.

Ook zouden we zeer graag een uitwerking zien die gebruik maakt van de virtual-reality omgeving, aangezien dit het scenario is waarvoor deze proof-of-concept werd opgezet.

[5] Er waren plannen om een RGB-matrix in te werken in de basisplaat, door het tijdsgebrek is deze niet uitgewerkt geweest.

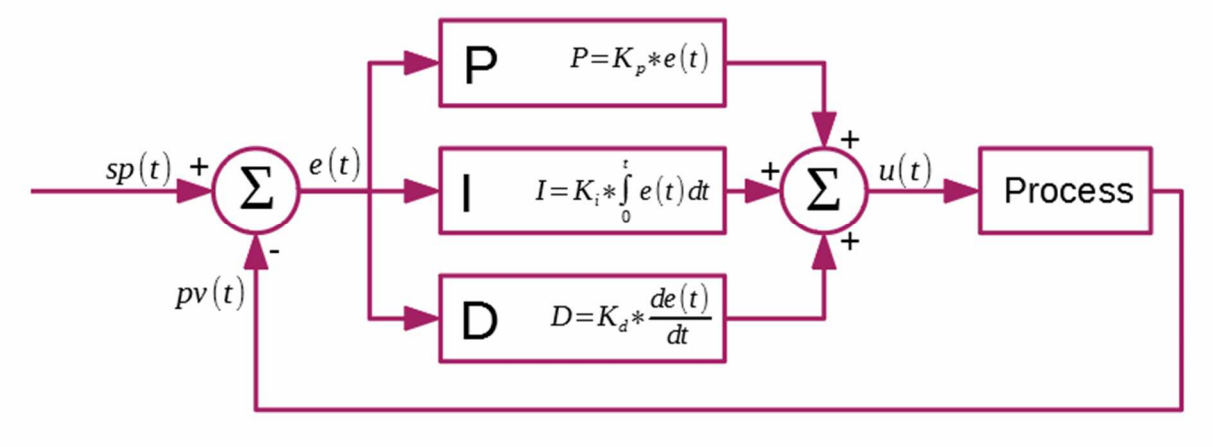
10. Extra

10.1. PID

Een PID-regelaar is de populairste en meest gebruikte ***closed-loop controller***.

Een PID-regelaar krijgt een input (sp), dit is de gewenste waarde. Van deze waarde wordt de huidige sensorwaarde (pv), of proces waarde, afgetrokken. Het resultaat (e), of afwijking, is de input voor het eigenlijke PID-algoritme. Dit algoritme bestaat uit drie parallelle functies: een ***P***roportionele, een ***I***ntegraal en een afgeleide (***D***erivative).

Het resultaat van elk van deze drie functies wordt opgeteld (u), dit is dan ook de uitgang van de PID-regelaar. Deze dient als input voor het proces. De uitgang van het proces (pv) is dan teruggekoppeld aan de PID-regelaar, vandaar dat men spreekt over een ***closed-loop*** systeem.



Figuur 10.1: Schema van een PID-regelaar

Elk van de functies in een PID-regelaar heeft zijn eigen K-factor. Deze factor bepaalt hoeveel elk van de functies doorwegen in het eindresultaat. Deze factoren worden ook gebruikt voor het tunen van de PID-regelaar. Als deze factoren juist gekozen en ingesteld worden, kan een zeer snelle en nauwkeurige PID-regelaar bekomen worden.

10.1.1. Proportioneel – P

De eerste functie is tevens de meest eenvoudige. Zoals de naam het zegt, is de uitkomst van deze functie proportioneel aan de input. De K_p-waarde bepaalt in welke mate de uitgang proportioneel is aan de ingang. Bij een grotere K_p-waarde zal de uitgang sneller stijgen bij een stijging van de ingang, en vice-versa met een kleinere K_p-waarde. De formule is als volgt:

$$P = K_P * e(t)$$

De P-functie is de meest eenvoudige, maar “kijkt” enkel naar de ogenblikkelijke afwijking. Hierdoor treedt ***overshoot*** zeer snel op, waardoor een P-regelaar zal oscilleren. De D-functie is hiervoor een oplossing.

De proportionele uitgang zal bij zeer kleine afwijkingen dan ook niet sterk genoeg zijn om de kleine afwijking weg te werken. De integraal biedt hier een oplossing op.

10.1.2. Integraal – I

De I-functie neemt de integraal van de afwijking. Bijgevolg zal de uitgang steeds groter en groter worden bij een constante kleine afwijking. De I-functie is uitermate geschikt voor het wegwerken van kleine afwijkingen die lang aanwezig zijn. De I-functie “kijkt” dus als het ware naar het verleden.

De I-functie wordt beschreven door volgende vergelijking:

$$I = K_I * \int_o^t e(t)dt$$

De I-functie is voornamelijk van toepassing op systemen die langzaam veranderen. Bij systemen die snel veranderen zal de afwijking nooit lang genoeg aanwezig zijn om de uitgang van de I-functie noemenswaardig te doen stijgen en dus door te laten wegen.

De I-functie moet altijd begrensd worden. Als de regelaar een constante afwijking heeft, maar niet genoeg kan regelen om de afwijking weg te werken, zal de uitgang van de I-functie constant blijven groeien. In theorie kan deze dus oneindig groot worden.

In de praktijk is de uitkomst meestal beperkt zonder dat er iets voor gedaan moet worden. Als de regelaar analoog werkt, is de uitkomst van de I-functie beperkt tot de voedingsspanning. Als de regelaar digitaal werkt, zal er op een bepaald moment een ***overflow*** optreden. Dit is natuurlijk niet wenselijk aangezien de uitkomst van de I-functie dan van de grootste waarde plots naar de kleinste waarde gaat, of omgekeerd.

Hoe dan ook is de waarde beperkt. Het is echter best om eigen beperkingen in te stellen, zodat ze voorspelbaar zijn en geen onverwacht gedrag met zich meebrengen.

10.1.3. Afgeleide – D

De D-functie neemt de afgeleide van de afwijking. Deze kijkt hoe snel de afwijking verandert. De D-functie “kijkt” naar de toekomst en kan dus beperkt de komende afwijking voorspellen.

De D-functie wordt gegeven door volgende vergelijking:

$$D = K_d * \frac{de(t)}{dt}$$

De D-functie zorgt dat de PID-regelaar sneller reageert op veranderingen in de afwijking.

10.1.4. Continue PID

Continue functies zijn functies die constant en vloeiend veranderen. Ze hebben als het ware een oneindige resolutie. Een continue functie zal ook voor elke ingangswaarde een uitgangswaarde teruggeven. Een continue functie is ook te schrijven als één eindige vergelijking.

Het constant veranderen wijst erop dat als het functieverloop getekend wordt op een grafiek, er geen enkele onderbreking terug te vinden is in het functieverloop. Het functieverloop is één doorlopende lijn, van negatief oneindig tot positief oneindig.

Het vloeiend veranderen wijst erop dat er geen scherpe knikken te vinden zijn in het functieverloop. Als er voldoende “ingezoomd” wordt, zal elke scherpe knik in een continue functie uiteindelijk vloeiend afgerond zijn.

Een continue PID-regelaar is in de praktijk een analoge PID-regelaar. Als deze volledig analoog geïmplementeerd is kan deze op elk tijdstip een uitgangswaarde genereren en doet dat volgens de eerder gedefinieerde regels. Van zodra er iets verandert aan de ingang zal de uitgang aangepast worden.

Als dit in- en uitgangsverloop uitgezet wordt op een grafiek geeft dit een volledig continu verloop.

10.1.5. Discrete PID

Een discrete functie is elke functie waarbij er onderbrekingen zijn in het functieverloop. Bijgevolg is het functieverloop niet één doorlopende lijn. Deze onderbrekingen kunnen voorkomen in de x-richting, de y-richting of beide.

Als een PID-regelaar geïmplementeerd wordt op een microcontroller zijn deze onderbrekingen te vinden in zowel de x- als de y-richting.

De onderbrekingen in de x-richting zijn het gevolg van de tijdsstappen in de microcontroller. Elke microcontroller (en computer) werkt volgens een klok. Bij iedere klokslag wordt de status van het systeem bijgewerkt. Ongeacht de snelheid van deze klok, zullen er altijd tijdsstappen terug te vinden zijn.

Het woord zegt het zelf, doordat er stappen genomen worden, zullen er onderbrekingen te vinden zijn in het tijdsverloop.

Er zijn ook onderbrekingen in de y-richting terug te vinden, welke het gevolg zijn van de resolutie. Het continue verloop in de y-richting wordt herleid tot slechts enkele mogelijkheden.

De stappen in de y-richting zijn afhankelijk van de resolutie. Een grotere resolutie zal resulteren in kleinere stappen. Aangezien een oneindige resolutie niet mogelijk is, zullen er altijd stappen terug te vinden zijn in de y-richting.

Deze stappen in de y-richting vormen geen groot probleem voor de berekeningen en het implementeren van een PID-regelaar. Het zijn de stappen in de tijd die het probleem vormen. Doordat een microcontroller naast het PID-algoritme nog andere zaken uitvoert, zullen deze tijdsstappen nooit even groot zijn.

Bijgevolg moet de tijdsstap meegenomen worden in de berekeningen. De dt-waarde voor de afgeleide is in dit geval niet nul.

10.2. Vision

Om het deel vision in meer detail te kunnen uitleggen is het belangrijk om eerst een vergelijking te maken tussen beelden en een geluidssignaal. Een geluidssignaal voelt veel natuurlijker aan, zeker als het aankomt op de bewerkingen en analyse ervan. Er is echter weinig verschil te bespeuren tussen beelden en een geluidssignaal (of gelijk welk analoog signaal).

10.2.1. Beeld als analoog signaal

Als er vandaag de dag gedacht wordt aan beelden (hetzij foto's of video) en hoe deze worden voorgesteld. Denkt men in eerste instantie altijd aan een matrix of raster van pixels. Beelden zijn echter evengoed een analoog signaal, net als een geluidssignaal.

Bij wijze van voorbeeld, wordt er verwezen naar composiet-video. Dit is een ondertussen bijna antieke standaard voor video. Dit is een volledige analoge manier om een videosignaal door te sturen over slechts één kanaal.

Door een beeld voor te stellen als een in de tijd veranderend analoog signaal, is het gemakkelijk in te zien dat er wiskundige berekeningen op uit te voeren zijn. In essentie is een beeld ook niet meer dan een analoog signaal dat op verschillende tijdstippen wordt gesampled, zoals een geluidssignaal. De uitgevoerde berekeningen zijn op zich niet zo verschillend van de berekeningen die uitgevoerd worden bij het analyseren van een geluidssignaal.

Het grote verschil tussen een beeldsignaal en een geluidssignaal, is het aantal dimensies. Een geluidssignaal verandert met de tijd slechts in één dimensie. Een beeldsignaal daarentegen verandert in twee dimensies. Terwijl een geluidssignaal voorgesteld kan worden door een lijn, zal een beeldsignaal voorgesteld worden

door een vlak. Dit geldt voor een zwart-wit beeld; voor een kleurenbeeld zullen er drie vlakken zijn, één voor elk kleurenkanaal.

Aangezien een beeld voorgesteld kan worden als een analoog signaal, komt er ook een vergelijking of functie mee overeen. Op deze functie kunnen de klassieke wiskundige operaties uitgevoerd worden, zoals integreren en differentiëren, evenals optellen, vermenigvuldigen etc.

Het zijn deze bewerkingen die de basis vormen van beeldherkenning. Bij beeldherkenning wordt er voornamelijk gekeken naar de intensiteit en de verandering ervan. Hierdoor is differentiëren de meest gebruikte bewerking.

Door de eerste en tweede afgeleide te nemen van het beeld, kan rudimentaire rand-detectie gedaan worden. Er wordt gekeken naar de extrema en de nul-doorgangen van deze tweede afgeleide. Uit deze punten kan dan bepaald worden waar de randen zich bevinden binnen het beeld.

Een nul-doorgang wijst op het midden tussen twee lokale pieken in het originele signaal en wijst dus op een mogelijke rand in de originele afbeelding. Door te kijken naar de aanliggende extrema van de tweede afgeleide kan er bepaald worden hoe “scherp” de rand is. Door het verschil te nemen van het extremum voor en na de nul-doorgang, kan bepaald worden hoe “scherp” de rand is, of hoeveel contrast er is tussen beide kanten van de rand. Hoe groter het verschil, hoe groter het contrast.

Door de dt-waarde van de afgeleide te variëren zijn verschillende effecten te bekomen. Door de dt-waarde te vergroten zal de afbeelding aan detail verliezen, maar hierdoor wordt evenwel de ruis verminderd.

In moderne camera’s en protocollen wordt het beeldsignaal niet meer analoog doorgestuurd. Alles wordt digitaal doorgestuurd, als een raster. Het is ook op deze manier dat het wordt opgeslagen in het geheugen van de camera of computer. Er is dus nood aan een operatie die hetzelfde effect heeft als voorgaande bewerkingen, maar die werkt op een raster. Deze bewerking is kernel convolution.

10.2.2. Kernel convolution

Bij kernel convolution wordt exact hetzelfde bekomen als bij de bewerkingen op het analoge signaal. De beelden moeten echter niet eerst omgevormd worden naar een analoog signaal met bijhorende functie. Door de invulling van de kernel juist te kiezen, kunnen bewerkingen, zoals afgeleiden nemen, bekomen worden. Hierbij blijft de afbeelding een tweedimensionale matrix.

Bij kernel convolution wordt een kleine matrix, met het center, over elke pixel in het beeld geplaatst. Alle pixels die onder de kernel liggen, worden vermenigvuldigd met de overeenkomstige waarde in de kernel. De resultaten worden bij elkaar

opgeteld en genormaliseerd. De waarde, die bekomen wordt na de normalisatie, is de nieuwe waarde van de pixel.

Aangezien de kernel gecentreerd wordt rond één pixel, zal deze altijd oneven dimensies hebben. In de meerderheid van de toepassingen zal de kernel ook vierkant zijn, bijgevolg is de kleinst mogelijke kernel drie op drie.

Kernel-convolution kan vergeleken worden met het nemen van het gemiddelde van een klein deel van het beeld. In essentie is kernel-convolution een gewogen gemiddelde.

Door de manier waarop kernel-convolution werkt is het belangrijk om de resultaten op te slaan in een tweede matrix. Indien dit niet gedaan wordt, wordt de originele matrix overschreven met nieuwe waardes tijdens de bewerking. Hierdoor worden verkeerde resultaten bekomen.

Eén van de problemen (of beperkingen) bij kernel-convolution, zijn de randgevallen aangezien dan een deel van de kernel buiten de originele matrix valt. In de meeste implementaties worden de locaties van de kernel die buiten de matrix vallen genegeerd. Hierdoor wordt het effect verkleind, maar worden wel juiste resultaten bekomen.

Een video de dit zeer goed uitlegt is te vinden op: https://youtu.be/C_zFhWdM4ic - (Pound, How Blurs & Filters Work - Computerphile, 2015)

10.2.3. Blur

Het doel van een blur is om de hoeveelheid detail in de afbeelding te verminderen. Een bijkomend resultaat is dat ruis ook verminderd wordt. Dit is dan ook de voornaamste toepassing van een blur bij beeldherkenning.

10.2.3.a. Mean blur

Een mean blur is de eenvoudigste blur. Bij deze blur wordt telkens het gemiddelde genomen van een sectie van het beeld. Dit gemiddelde vormt dan de nieuwe waarde voor de pixel.

Dit kan bekomen worden door alle waardes in de kernel op dezelfde waarde te zetten. In de praktijk worden deze waardes allemaal één. Door de grootte van de kernel aan te passen kan een sterkere of zwakkere blur bekomen worden. Een grotere kernel zal resulteren in een sterkere blur.

Mean blur is het eenvoudigst om te implementeren en vergt ook het minste rekenwerk. Deze vermindert de hoeveelheid detail zeer sterk, waardoor deze niet de voorkeur krijgt wanneer structuren en vormen moet herkend worden.

10.2.3.b. Gaussian blur

Een Gaussian blur zal voor dezelfde vermindering in ruis zorgen en meer detail en randen behouden in vergelijking met de mean blur. Deze blur wordt dan ook het meeste gebruikt in beeldherkenningstoepassingen.

Om een Gaussian blur uit te voeren wordt de kernel ingevuld volgens een (tweedimensionale) normaalverdeling. Vandaar ook de naam. Door dit te doen wordt de kernel-convolution een gewogen gemiddelde. Hoe verder een pixel verwijderd ligt van de center pixel, hoe minder de waarde doorweegt.

1	2	1
2	4	2
1	2	1

Figuur 10.2: 3x3 kernel voor een Gaussian blur

Door de grootte van de kernel aan te passen kan een sterkere blur bekomen worden, net als bij een mean blur. De standaardafwijking van de normaalverdeling kan ook aangepast worden. Op deze manier worden ook verschillende sterktes en effecten bekomen. Bij gevolg is er over een Gaussian blur veel meer controle uit te oefenen.

Dit controleerbare aspect, is wat de Gaussian blur zo populair maakt in beeldherkenningstoepassingen.

10.2.4. Edge detection

Voor het detecteren van randen zijn er meerdere mogelijkheden. Deze zijn nog steeds gebaseerd op kernel convolution. Het detecteren van de randen wordt bekomen door de kernel op een speciale manier in te vullen. De meest voorkomende manieren om randen te detecteren zijn een ***Sobel Operator*** en ***Canny Edge Detection***.

10.2.4.a. Sobel Operator

Eerder werd er gesproken over hoe veel bewerkingen op beelden gebaseerd zijn op afgeleiden. Deze ***Sobel Operator*** benaderd de afgeleide van een beeld. Echter is deze bewerking zeer gevoelig voor ruis en zal deze niet alleen behouden maar ook een deel versterken.

Deze operator zal randen herkennen in slechts een oriëntatie. Als de kernel ingevuld wordt op deze manier zullen randen gedetecteerd worden in de X-richting.

-1	0	1
-2	0	2
-1	0	1

Figuur 10.3: Voorbeeld van een Sobel Operator

Als een kernel convolution wordt uitgevoerd met deze kernel wordt een sterke waarde verkregen als er groot contrast te vinden is.

Als deze kernel 90° wordt gedraaid, wordt de operator bekomen voor het detecteren van randen in de Y-richting.

Door twee kernel convolutions uit te voeren (één met elke kernel) worden twee waardes bekomen voor elke pixel, een voor de X-richting en een voor de Y-richting. Deze waarde is een indicatie voor het contrast op die pixel (de gradiënt). De totale gradiënt kan bekomen worden door de volgende formule:

$$G_t = \sqrt{{G_x}^2 + {G_y}^2}$$

Waar de totale gradiënt een grote waarde heeft, is een rand terug te vinden. Door de manier waarop deze Sobel Operator werkt, is het zeer zeldzaam dat deze gebruikt wordt voor het detecteren van randen in kleurenbeelden.

Door de arctangens te nemen van de gradiënt in de X en Y-richting, kan ook de oriëntatie van de rand bepaald worden. Met andere woorden: aan welke kant van de rand bevindt zich het donkere deel.

Deze video legt de werking van de Sobel Operator in veel meer detail uit: https://youtu.be/uihBwtPIBxM - (Pound, Finding the Edges (Sobel Operator) - Computerphile, 2015)

10.2.4.b. Canny Edge Detection

Canny Edge Detection is een uitbreiding op de Sobel Operator. Het Canny-algoritme heeft als input de output van een Sobel Operator.

Het eerste was het Canny-algoritme doet is de gedetecteerde randen herleiden tot slechts één pixel breed. Door de manier waarop de Sobel Operator werkt, zijn de gedetecteerde randen meerdere pixel breed. Ze zijn zelf ook een gradiënt. Door deze te herleiden tot slechts één pixel breed, wordt het gemakkelijker om de randen te lokaliseren, maar wordt het hele proces ook onafhankelijk van de resolutie.

Hiervoor wordt een hysteresis-treshold uitgevoerd. Deze heeft een gelijkaardige werking aan de binary-threshold, gebruikt in het vision programma. Hier wordt er echter gewerkt met twee drempel waardes.

De eerste stap om deze threshold uit te voeren is bepalen voor elke pixel, of het een lokaal maximum is. Hetgeen vooral van belang is, is kijken of de pixel een lokaal maximum overheen de rand. Langsheen de rand maakt het niet uit, de rand kan scherpen of minder scherp worden, maar het blijft een rand.

Hiervoor is de oriëntatie (of richting) van de rand nodig, gelukkig is deze gemakkelijk te bepalen uit het resultaat van de Sobel Operator.

Dit is de eerste stap van de hysteresis-threshold. De gradiënten van de gedetecteerde randen werden herleid tot slechts één pixel breed.

De hysteresis is te vinden bij het “sorteren” van de randen. Er wordt gewerkt met twee waardes, een bovenste en een onderste drempel. Alle randen die een waarde hebben groter dan de bovenste drempel worden sowieso behouden. Alle randen die een waarde hebben lager dan de onderste drempelwaarde, worden altijd weggelaten.

De randen met een waarde tussen de twee drempelwaardes worden enkel behouden als ze in verbinding staan met een rand die een waarde groter heeft dan de bovenste drempelwaarde. Dit kan vergeleken worden met een dalende of stijgende flank bij elektronica.

Op deze manier worden enkel de belangrijke randen behouden. Het grootste deel van de ruis wordt volledig onderdrukt.

De volgende video legt Canny Edge Detection in veel meer detail uit: https://youtu.be/sRFM5IEqR2w - (Pound, Canny Edge Detector - Computerphile, 2015)

10.3. Raspbian - headless

Op het moment dat de Raspberry Pi geconfigureerd werd, was er noch een HDMI-scherm beschikbaar noch een ethernet kabel. Op zich was dit geen probleem aangezien de Raspberry Pi zou draaien via Wi-Fi en bestuurd werd over SSH. Dit vormde echter wel een probleem bij het instellen en de eerste keer opstarten. Er was geen enkele manier om de Raspberry Pi in te stellen.

Gelukkig bestaat er een manier om SSH aan te zetten en het Wi-Fi netwerk in te stellen, nadat Raspbian naar de geheugenkaart gebrand is. Door deze stappen uit te voeren zal de Raspberry Pi opstarten met de mogelijkheid tot SSH, en verbinden met het ingestelde netwerk. Deze stappen configureren deze instellingen enkel bij de eerste keer opstarten. Als de Raspberry Pi herstart wordt gaan de instellingen verloren (ze worden opnieuw ingesteld naar de standaardwaarden). Deze manier is perfect om een Raspberry Pi in te stellen, zodat deze kan draaien zonder scherm, toetsenbord etc.

Nadat ***Raspbian*** gebrand is naar de geheugenkaart (via bv. Windows), zullen er een aantal partities tevoorschijn komen. Eén van deze partities is de BOOT-partitie. Op deze partitie moeten twee bestanden aangemaakt worden om de gewenste resultaten te bekomen.

Om SSH aan te zetten moet enkel een bestand aangemaakt worden met de naam: SHH. Hierdoor zal de Raspberry Pi opstarten met SSH, bij de opstart wordt dit bestand verwijderd (om veiligheidsredenen) zodat bij de volgende opstart SSH terug uitstaat.

Om het Wi-Fi netwerk in te stellen is iets meer werk nodig. Éérst en vooral moet een tweede bestand aangemaakt worden op de BOOT-partitie. Dit bestand moet de naam wpa_supplicant.conf krijgen.

Dit bestand kan geopend worden met kladblok. In dit bestand moet de volgende tekst komen:

```
country=BE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="[SSID]"
    psk="[PSWD]"
    key_mgmt=WPA-PSK
}
```

Code 10.1: Wi-Fi configuratiebestand voor het headless opstarten van Raspbian

[SSID] moet vervangen worden door de naam van het netwerk waarmee verbinding gemaakt wordt. [PSWD] moet vervangen worden door het wachtwoord van het netwerk.

Hierdoor zal de Raspberry Pi opstarten en verbinding maken met het netwerk. Ook dit bestand wordt verwijderd nadat de Raspberry Pi opgestart is.

Dit is voldoende om de Raspberry Pi op te starten en alles te configureren. Hetgeen nu enkel nog moet gebeuren is de geheugenkaart in de Raspberry Pi steken en de voeding verbinden. Nadat de Raspberry Pi opgestart is, kan er verbinding gemaakt worden via SSH. Het eerste dat dan moet gebeuren is het netwerk en SSH op de “juiste” manier instellen. Dit kan gebeuren door het configuratiescherm op te roepen en de instellingen aan te passen.

Figuren

Tabellen

Codefragmenten

Referenties

Brevig, A. (sd). *Datatype Practices for Arduino*. Opgeroepen op maart 13, 2019, van Arduino Playground: https://playground.arduino.cc/Code/DatatypePractices

Heinisuo, O.-P. (2019, Januari 9). *opencv-python 4.0.0.21*. Retrieved April 25, 2019, from PyPi: https://pypi.org/project/opencv-python/4.0.0.21/

Hildreth, E. C. (1985, September). Edge Detection. *A.I. Memo*(858). Opgeroepen op April 24, 2019, van http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.5715&rep=rep1&type=pdf

Michael. (2019, Februari 12). *How To Setup Raspberry Pi Zero W Headless WiFi*. Opgeroepen op mei 19, 2019, van CORE electronics: https://core-electronics.com.au/tutorials/raspberry-pi-zerow-headless-wifi-setup.html

NumPy developers. (2019). *NumPy - about*. Retrieved April 25, 2019, from numpy.org: https://www.numpy.org

NumPy Developers. (2019, April 22). *numpy 1.16.3*. Retrieved April 25, 2019, from PyPi: https://pypi.org/project/numpy/

OpenCV team. (n.d.). *OpenCV - about*. Retrieved April 25, 2019, from opencv.org: https://opencv.org/about/

Pound, D. M. (2015, November 11). Canny Edge Detector - Computerphile. (S. Riley, Red.) Nottingham, Verenigd Koninkrijk: Computerphile. Opgehaald van https://youtu.be/sRFM5IEqR2w

Pound, D. M. (2015, November 4). Finding the Edges (Sobel Operator) - Computerphile. (S. Riley, Red.) Nottingham, Verenigd Koninkrijk: Computerphile. Opgehaald van https://youtu.be/uihBwtPIBxM

Pound, D. M. (2015, Oktober 2). How Blurs & Filters Work - Computerphile. (S. Riley, Red.) Nottingham, Verenigd Koninkrijk: Computerphile. Opgehaald van https://youtu.be/C_zFhWdM4ic

Rosebrock, A. (2018, December 3). *imutils 0.5.2*. Retrieved April 25, 2019, from PyPi: https://pypi.org/project/imutils/

The Python Software Foundation. (sd). *Python - about*. Opgeroepen op April 25, 2019, van python.org: https://www.python.org/about/

Index

Bijlagen

Alle bijlagen en het volledige programma zijn terug te vinden op Github via volgende link: https://github.com/pcassima/thesis_hogent. Op deze manier is altijd de laatst nieuwe versie zichtbaar. Ook blijft de hele geschiedenis van veranderingen zichtbaar; oudere versies kunnen dus ook opgeroepen worden.

Alle documentatie en gerelateerde documenten zijn terug te vinden onder de **/docs** map. Alle programma's zijn terug te vinden onder de **/src** map. De programma's zijn onderverdeeld in Python en Arduino.

Een kopie van dit document, met eventuele addenda, zal ook terug te vinden zijn op Github.

Het hele project is uitgebracht als open-source onder GNU-GPL V3. Hierdoor kan iedereen de code gebruiken en aanpassen. Meer informatie over open-source kan gevonden worden op https://opensource.org.

Aanpassingen kunnen zeer gemakkelijk gemaakt worden in de browser via https://gitpod.io/#https://github.com/pcassima/thesis_hogent. Via deze link wordt een geïsoleerd systeem aangemaakt in de browser (met alle benodigdheden) en worden alle aanpassing ook direct gepubliceerd naar Github.