Inhoudsopgave

[Inleiding 2](#_Toc462918204)

[1.0 Introductie 2](#_Toc462918205)

[1.1 Aanleiding 2](#_Toc462918206)

[1.2 De context 2](#_Toc462918207)

[Context en Methodologie 3](#_Toc462918208)

[2.0 Inleiding 3](#_Toc462918209)

[2.1 Doelstelling van het onderzoek 3](#_Toc462918210)

[2.2 Verzameling van data 3](#_Toc462918211)

[2.3 Betrouwbaarheid onderzoek 3](#_Toc462918212)

[2.4 Onderzoeks ontwerp 3](#_Toc462918213)

[Theorie 4](#_Toc462918214)

[Verloop van het onderzoek 5](#_Toc462918215)

[Resultaten 6](#_Toc462918216)

[Reflectie 7](#_Toc462918217)

[Aanbevelingen 8](#_Toc462918218)

[BIJLAGEN 9](#_Toc462918219)

[Bijlage 1 Literatuurlijst 9](#_Toc462918220)

# Inleiding

## 1.0 Introductie

Het uiteindelijke doel van het Cooperative Arduino Maze Solver Project is om robots te programmeren die informatie van het doolhof lezen en deze informatie vervolgens met elkaar te delen. Deze gedeelde informatie moet er vervolgens toe leiden dat meerdere rijdende robots de oplossing van het doolhof zo snel mogelijk kunnen vinden.

## 1.1 Aanleiding

Het voorgaande onderzoek voegde een radio en een aantal vlakken toe aan het project. Deze gaven een opzet voor de communicatie en het opzetten van het doolhof. Het resultaat van het voorgaande onderzoek was dus nog niet voldoende om het gewenste resultaat van het project te behalen.

## 1.2 De context

Aan het begin van het onderzoek werden de white paper en de code van het voorgaande onderzoek geleverd. Ook kregen we alle onderdelen die de vorige onderzoeksgroep had achtergelaten.  
Volgens deze white paper was er code ontwikkeld die twee rijdende Arduino's met elkaar liet communiceren zodat ze samen het doolhof konden oplossen.

# Context en Methodologie

## 2.0 Inleiding

De minor Business Innovation van de Haagse Hogeschool biedt een scala aan opdrachten waar studenten aan kunnen werken. Tijdens deze opdrachten kunnen ze onderzoek doen naar nieuwe technieken en technologieën en deze in een praktische omgeving testen. De opdracht die wij gekozen hebben heette het Cooperative Arduino Maze Solver Project.

## 2.1 Doelstelling van het onderzoek

De doelstelling van het onderzoek is om de code van de Arduino's verder te ontwikkelen zodat deze informatie van het doolhof met elkaar delen en een kaart van het doolhof kunnen tekenen.

## 2.2 Verzameling van data

Om de data van de functionaliteit en de betrouwbaarheid van de robots te verzamelen, hebben we verschillende tests uitgevoerd. Bij deze tests werd telkens bijgehouden hoeveel fouten de robot maakte en hoelang hij er over deed om door het hele testdoolhof te rijden.

## 2.3 Onderzoeks ontwerp

De opzet logica bood nog niet heel betrouwbare resultaten. De code van het vorige onderzoek liet zien hoe de radio's werkten en deed al een poging om informatie van de rijdende Arduino naar een luisterende Arduino te versturen. De informatie die verstuurt werd was echter lang niet altijd de juiste informatie waardoor de luisteraar er niet veel mee kon. Deze situatie kon verbeterd worden door het ontwerp van de rijdende robots aan te passen. Het ontwerp dat de vorige groep gebruikte bestond uit vier batterijen als stroomtoevoer en drie sensoren om de lijn te volgen. Met dit ontwerp was de robot erg traag.

De gemiddelde spanning die 4 oplaadbare batterijen kunnen leveren is 4.8V terwijl de aanbevolen spanning voor een Arduino Uno bord 7-12V is[1]. Het gevolg was dat de robot weinig stroom over hield om zijn motoren aan te sturen. De robot kon zichzelf nauwelijks voortbewegen. Dit maakte het moeilijk om de robots goed te testen aangezien ze telkens tot stilstand kwamen. Daarom zijn we als eerst gaan kijken naar het verbeteren van de stroomtoevoer. We zijn eerst gaan kijken naar de mogelijkheid om een accu aan de robot toe te voegen. Een accu zou ook een oplaadbare stroombron vormen die al snel een veel hogere spanning zou kunnen leveren. Maar het bleek dat een accu al snel een grote investering zou zijn. Aangezien er geen budget voor het onderzoek bestond was een accu geen realistische mogelijkheid.

Wat wel een vrij simpele oplossing zou leveren was om twee batterijhouders in serie te schakelen aangezien we wel beschikking hadden over een groot aantal batterijhouders. Twee gevulde batterijhouders zouden acht batterijen bevatten die samen een spanning van 9,6V zouden leveren. Dit voltage lag ruim binnen de 7-12V die de Arduino nodig had.

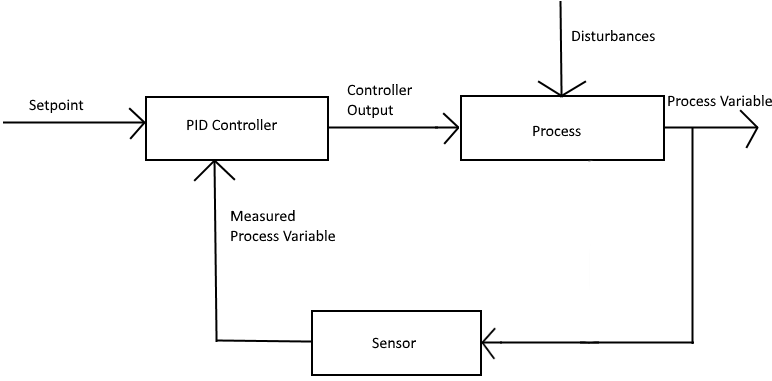
# Theorie

Het uiteindelijk doel van het project is om een aantal rijdende robots in een doolhof te plaatsen die vervolgens met elkaar samenwerken om zo snel mogelijk het doolhof op te lossen. Om dit mogelijk te maken zijn er verschillende methodes om de robots te laten communiceren.

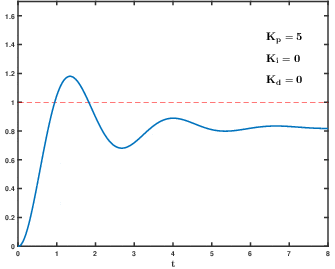
Wij hebben gekeken naar de mogelijkheid om meerdere zendende robots in het doolhof te hebben die informatie verzamelen en doorsturen en een centrale ontvanger te hebben die de informatie verwerkt. Het voordeel van deze methode was dat de centrale ontvanger vastgemaakt kon worden aan een laptop. Daardoor hadden we veel meer geheugen tot onze beschikking dan de hoeveelheid die op een Arduino bordje paste. Hierdoor zou de centrale ontvanger gemakkelijk alle informatie van de zenders op kunnen vangen en tot een kaart kunnen verwerken voor elke individuele zender.

### PID Controller

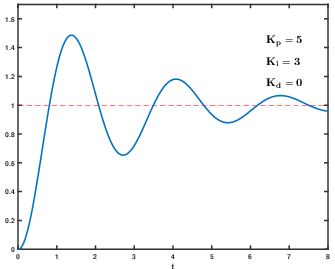
Aangezien we tijdens de vroege testen erachter kwamen dat het volgende van de lijn zelf ook niet erg goed werkte, hebben we ook een Proportional Intergral Derivative Controller (PID Controller) in de code geïmplementeerd. Met behulp van deze controller bleef de robot een stuk beter de lijn volgen.

Een PID Controller gaat uit van een bepaalde "Setpoint (SP)". Dit is de doelwaarde van de controller. Deze waarde vergelijkt de controller met de "Process Variable (PV)" ook wel de werkelijk gemeten waarde van het proces waar de controller aan vast zit. Als de waardes overeen komen hoeft de controller niets te doen, maar als de waardes niet overeen komen moet de controller in actie komen.   
De controller gaat proberen om het verschil tussen de SP en de PV te verkleinen. Dit doet hij door de PV aan te passen aan de hand van het verschil, als de SP hoger is dan de PV dan moet de waarde van de PV groter worden en andersom.

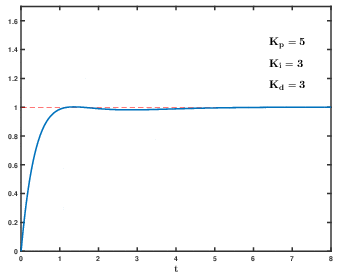
Hoeveel de PV per iteratie van het proces wordt aangepast gebeurt aan de hand van drie andere waardes, namelijk de Proportional, Intergral en Derivative. De controller gebruikt de waarde van de Proportional om het gemeten verschil te vermenigvuldigen. Het vermenigvuldigde verschil wordt vervolgens opgeteld bij de PV. Dit zorgt ervoor dat de PV rond de SP gaat oscilleren zodat het proces minder gaat afwijken van de gewenste SP.



Vervolgens wordt voor de Integral het totale verschil opgeteld van alle iteraties die het proces doorlopen heeft. Dit totaal wordt vervolgens met de waarde van de Integral vermenigvuldigd en daarna ook opgeteld bij de PV. Dit leidt ertoe dat de oscillaties groter worden, waardoor de PV rond de SP blijft oscilleren.



Ten slotte is er nog de Derivative. Met de Derivative probeert de controller het verschil van zijn volgende iteratie te voorspellen. Voor de Derivative wordt het vorige verschil afgetrokken van het huidige verschil. Vervolgens wordt het resultaat vermenigvuldigd met de waarde van de Derivative en daarna opgeteld bij de PV. Hierdoor worden de oscillaties van de PV kleiner zodat de PV dichter in de buurt blijft van de SP.



In het geval van ons onderzoek was de SP dat de lijn precies onder de robot doorliep en de PV de hoeveelheid stroom die naar de motoren geleid werd.

# Verloop van het onderzoek

Het ontwerp van de rijdende robot in het voorgaande onderzoek gebruikte drie Analog Line Sensors om de lijn te blijven volgen. De code ging ervan uit dat als de robot een bocht tegenkwam, dat oriëntatie van de robot ten opzichte van de lijn bijna parallel was. Tijdens het oriënteren op de code van het voorgaande onderzoek werd al snel duidelijk dat de opstelling van de robot en de code niet kon garanderen dat de robot parallel aan de lijn reed. Hierdoor gaf de robot al snel de verkeerde informatie door aan de luisteraar en gedroeg de rijdende robot zich vaak als een simpele lijn volgende robot die bochten naar rechts prefererende.

We zijn daarom gaan brainstormen om het lijn volgen van de robots betrouwbaarder te maken. We zijn eerst gaan nakijken of de onderdelen van de robot goed werkten en hun functie vervulden. Dit hebben we gedaan door de bedrading en de waardes van de sensoren te controleren. Het bleek dat de sensoren wel goed functioneerden, maar dat ze maar een kleine marge hadden waarin ze konden werken nadat ze gekalibreerd waren door de code.   
Na het natrekken van de specificaties van de sensoren bleek ook dat de optimale detectie afstand van een Analog Line Sensor 3mm was. De sensoren op onze robot zaten op 5mm hoogte. Hierdoor kregen de sensoren veel extra licht binnen en werden de waardes van het wit van de vlakken een stuk hoger. We hebben daarom de bevestiging van de sensoren aan de car aangepast zodat de sensoren dichter bij de optimale 3mm kwamen. Hierdoor werden de witwaardes van de sensoren een stuk lager en werd dus het verschil tussen het wit van een vlak en de zwarte lijn een stuk groter.

Vervolgens zijn we gaan kijken hoe we de werkelijke beweging en de berichtgeving van de sensoren konden verbeteren. Hiervoor zijn we gaan kijken naar het aantal sensoren die op de Arduino car zaten. Omdat drie sensoren niet voldoende bleken te zijn om betrouwbare berichtgeving te garanderen, kwamen we tot de conclusie dat er minstens vijf sensoren nodig zouden zijn om en de lijn te kunnen volgen en bochten te herkennen. De binnenste drie sensoren zouden dan het volgen van de lijn regelen en de buitenste twee het herkennen van bochten.



De robot reed alleen nog steeds niet stabiel genoeg om goed gebruik te kunnen maken van de lijn volgende sensoren. De robot zwiepte veel heen en weer over de lijn waardoor hij vaak scheef aankwam bij een bocht. Om die stabiliteit te verhogen zijn we op zoek gegaan naar voorbeelden van andere "line followers". Het bleek dat veel van deze voorbeelden in de code gebruik maakten van een Proportional Integral Derivative Controller (zie hfdst Theorie PID Controller). Op basis van de voorbeelden hebben we onze eigen controller ontwikkeld.

De Setpoint van onze PID Controller was dat de lijn precies onder het midden van de robot liep. We hadden vier sensoren beschikbaar om de lijn te volgen. Deze nummerden we van 1 tot 4. Als een sensor aangaf dat hij op de lijn zat, gaf hij zijn nummer door. De lijn was dik genoeg dat maximaal 2 sensoren tegelijk aangaven dat ze de lijn zagen. Vervolgens telden we de Nadat we een PID Controller geïmplementeerd hadden in de loop kwam de gewenste situatie veel vaker voor dan voorheen.

### Informatie verzamelen

Nu konden we gaan kijken naar het regelen van het verzenden van informatie. Hierbij konden we dezelfde logica gebruiken die het voorgaande onderzoek ook toepaste. In dat onderzoek werd al onderscheid gemaakt tussen bochten, kruispunten en doodlopende lijnen. Hier moest echter nog aan toegevoegd worden wat een recht vlak was. Om dat op te lossen zijn we eerst gaan kijken naar de mogelijkheid om de robot te laten herkennen wanneer hij een nieuw vlak tegenkwam, want als de robot doorgaf dat hij een nieuw vlak tegengekomen was en dat het bericht dat hij daarna verzond ook aangaf dat hij een nieuw vlak was tegengekomen moest dat betekenen dat hij over een rechte lijn gereden had.

Om een nieuw vlak te kunnen herkennen hadden we een uniek patroon ontwikkeld. Dit patroon was wit op de middelste sensoren en zwart op de buitenste. Als dit patroon gesignaleerd werd, stuurde de robot dat hij een nieuw vlak gevonden had. Tijdens het testen bleek echter dat het moeilijk was om in het programma af te vangen wanneer het patroon herkent moest worden. De robot dacht te snel dat hij een bocht moest maken waardoor het voorkwam dat hij vast kwam te zitten in een loop van bochten maken.

Uiteindelijk hebben we afgezien van het nieuwe patroon en aan het rechte vlak zelf een aanpassing gemaakt. De zwarte lijn van het rechte vlak werd nu in het midden van het vlak afgebroken door drie witte strepen. De robot moest de strepen in een bepaalde tijd tellen en bericht geven van een recht vlak zodra hij alle drie de strepen geteld had. Dit patroon bleek wel effectief en correct bericht te geven en had geen invloed meer op de andere vlakken.

# Resultaten

# Reflectie

# Aanbevelingen

# BIJLAGEN

## Bijlage 1 Literatuurlijst

Maniacbug, Arduino driver for nRF24L01. From, https://github.com/maniacbug/RF24

iPrototype.nl, Line Tracking sensor – Analog. From, https://iprototype.nl/products/components/sensors/linetracking-analog

PID Line Following Robot http://waihung.net/pid-line-following-robot/

https://en.wikipedia.org/wiki/PID\_controller

https://github.com/pololu/libpololu-avr/blob/master/src/PololuQTRSensors/PololuQTRSensors.cpp

http://letsmakerobots.com/node/38550

[1]https://www.arduino.cc/en/Main/ArduinoBoardUno