KU Leuven
Departement Computerwetenschappen

P&O: Computerwetenschappen

Eindverslag

Team:
Geel

Robin Goossens
Samuël Lannoy
Ruben Pieters
Eline Vanrykel
Dries Verreydt

Academiejaar 2012 – 2013

Samenvatting

De robot moet autonoom een doolhof verkennen en het kortste pad berekenen doorheen dit doolhof, met enkel het gebruik van vier sensoren. Verder moet hij ook in staat zijn om barcodes te herkennen en hij moet de actie uitvoeren die bij een bepaalde barcode hoort. Allereerst moet de robot dus heel nauwkeurig kunnen rijden en draaien. Deze nauwkeurigheid wordt bekomen door de robot uitvoerig te testen en zo de parameters voor deze functies te bepalen. Ook de lichtsensor en ultrasone sensor zijn getest. De robot moet immers weten hoe hij de waardes die deze sensoren teruggeven moet interpreteren om vervolgens de aangewezen acties uit te voeren. Om zo snel mogelijk een doolhof te verkennen, is het wenselijk niet te botsen met muren. Daarom hebben we een algoritme gemaakt dat de robot in het midden van een vakje zet en evenwijdig oriënteert met de muren. Dit, gepaard met een minimale afwijking bij het rijden, zorgt ervoor dat botsingen vermeden worden. Het algoritme maakt gebruik van een ander algoritme dat de robot loodrecht op een witte lijn zet.
Terwijl de robot het doolhof verkent, loopt er op de robot een aparte thread waar de robot met de lichtsensor constant op zoek is naar ofwel een witte lijn ofwel een barcode. Een barcode begint altijd met een zwarte waarde en een witte lijn met een witte waarde. Het algoritme maakt hiervan gebruik om een onderscheid te maken tussen de twee. Wanneer er sprake is van een barcode, wordt de gelezen data verder geanalyseerd om te bepalen over welke barcode het gaat. Vervolgens wordt de bijbehorende actie uitgevoerd. Het verkennen van het doolhof zelf gebeurt door herhaaldelijk aangrenzende tegels te zoeken, deze toe te voegen aan een lijst en deze lijst af te werken met een diepte-eerst zoekalgoritme. Wanneer de layout van het doolhof bekend is, wordt het kortste pad berekend tussen een begin- en eindpunt met behulp van het A*-zoekalgoritme.
Verder is er een simulator ontworpen die dezelfde functionaliteit bezit als de echte robot. Deze simulator staat toe om veel algoritmes makkelijker en sneller te testen dan wanneer dit zou moeten gebeuren met de echte robot. Zowel de robot, als de simulator worden bediend met behulp van een grafische user interface (GUI) op PC. Deze GUI is in staat om commando's te sturen naar zowel de robot, via bluetooth, als rechtstreeks naar de simulator. Ook de robot en simulator sturen gegevens naar de PC die worden weergegeven in de GUI. Zo is het duidelijk wat de robot op elk moment waarneemt en dus waarom hij de acties onderneemt.

Inhoudsopgave

1 Inleiding

We hebben een robot gebouwd die bestuurbaar is vanuit een grafische gebruikersinterface (GUI) op PC. De PC communiceert met deze robot met behulp van bluetooth. Daarnaast hebben we een simulator gebouwd die het gedrag van deze robot virtueel nabootst. De robot heeft vier sensoren: een lichtsensor, een ultrasone sensor en twee druksensoren. Met behulp van deze sensoren zijn de robot en de simulator in staat autonoom een doolhof te verkennen en vervolgens het kortste pad doorheen dit doolhof te berekenen. Een goed begrip van de werking en limieten van deze sensoren is dus van cruciaal belang. In dit verslag wordt onder meer het proces beschreven om tot dit begrip te komen. Eerst beschrijven we hoe we de robot hebben gebouwd en de daarmee gepaard gaande testen hebben uitgevoerd. Deze dienden om de parameters voor de robot te bepalen en tot een begrip te komen van de werking van de ultrasone sensor en de lichtsensor. Vervolgens bespreken we de verschillende algoritmes die we hebben geschreven om aan de gestelde opgave te voldoen. In de daarop volgende sectie bespreken we de software die we hebben geschreven om tot een succesvol geïntegreerd project te komen, waaronder de communicatie via bluetooth tussen de PC en de robot. Verder lichten we ook toe hoe we de simulator en zijn verschillende onderdelen hebben ontworpen. Eindigen doen we met een beschrijving van de grafische gebruikersinterface.

2 Bouw robot

2.1 Fysieke bouw

De basis van het ontwerp van onze robot is gebaseerd op de zogenaamde Express-bot[3]. Bij dit ontwerp van de robot ligt de NXT brick helemaal horizontaal en is hij omringd door de hulpstukken van Lego. De twee voorste wielen worden aangedreven door motoren en achteraan is er een vast wieltje. Dat vaste wieltje heeft geen rubberen velgen om zich heen, zodat de wrijving met de grond bij het draaien minimaal is. De robot is tevens redelijk laag gelegen want een laag zwaartepunt zorgt immers voor een groter evenwicht. Onze robot, zoals die was bij demo 1 is te zien in figuur 1(a).

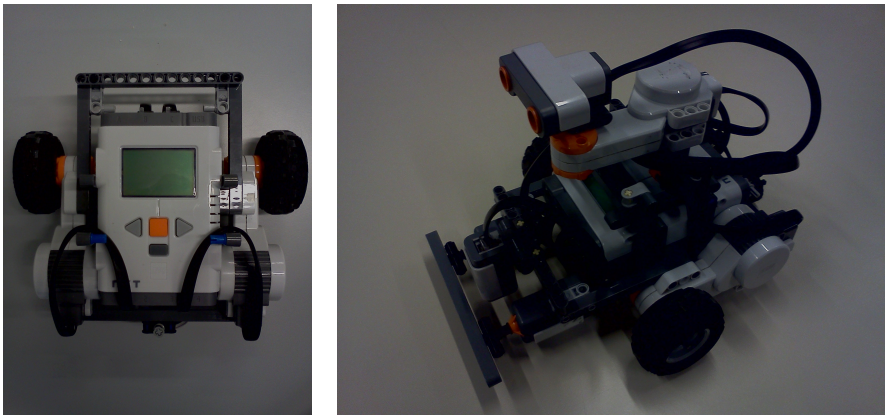
Voor demo 2 en 3 hebben we op dit model vier sensoren bevestigd: een ultrasone sensor, een lichtsensor en twee druksensoren. De sensoren zijn zodanig geplaatst zodat ze de stabiliteit en de symmetrie van de robot bewaren.
De ultrasone sensor is bovenop de robot geplaatst en wordt aangedreven met een derde motor. Zo kan de ultrasone sensor metingen uitvoeren in verschillende richtingen, zonder dat de robot hiervoor moet ronddraaien.
De twee druksensoren werden vooraan op de robot geplaatst en zijn allebei vastgemaakt aan eenzelfde staaf die als bumper fungeert. De bumper heeft onder meer als voordeel dat hij de NXT brick van de robot beschermt tegen botsingen. Bovendien hebben we door de twee druksensoren (één links en één rechts ten opzichte van de middellijn van de robot) te verbinden, een veel betere gevoeligheid bij niet-loodrechte botsingen van de robot op een muur.
De lichtsensor is ook vooraan de robot geplaatst, maar bevindt zich achter de bumper. Dit heeft als voordeel dat de sensor tijdig een witte lijn of barcode op de grond opmerkt en dat de sensor niet geraakt wordt bij botsingen. De sensor is loodrecht naar beneden gericht om de metingen zo correct mogelijk te laten verlopen.

Later hebben we het vaste wieltje vervangen door twee ronddraaiende wieltjes met een rubberen velg. Het vaste wieltje bood immers nog steeds vrij veel weerstand.
Het uiteindelijke resultaat is te zien op figuur 1(b).

2.2 Proefopstellingen en meetresultaten

2.2.1 Afwijking op de gereden afstand

Om de ingebouwde *DifferentialPilot* van LeJOS te gebruiken, moet de wieldiameter van beide wielen worden ingegeven. Deze wieldiameter werd bepaald door de robot telkens een vooraf ge-



Figuur 1: **(a)** Eerste ontwerp van de robot. Vooraan zijn de aandrijvende wielen te zien, achteraan het vaste wieltje. **(b)** Finaal ontwerp van de robot met alle sensoren. Vooraan is de bumper die de robot beschermt tegen botsingen, daar net achter de lichtsensor die witte lijnen en barcodes scant. Bovenaan staat de ultrasone sensor die de afstanden tot obstakels meet.

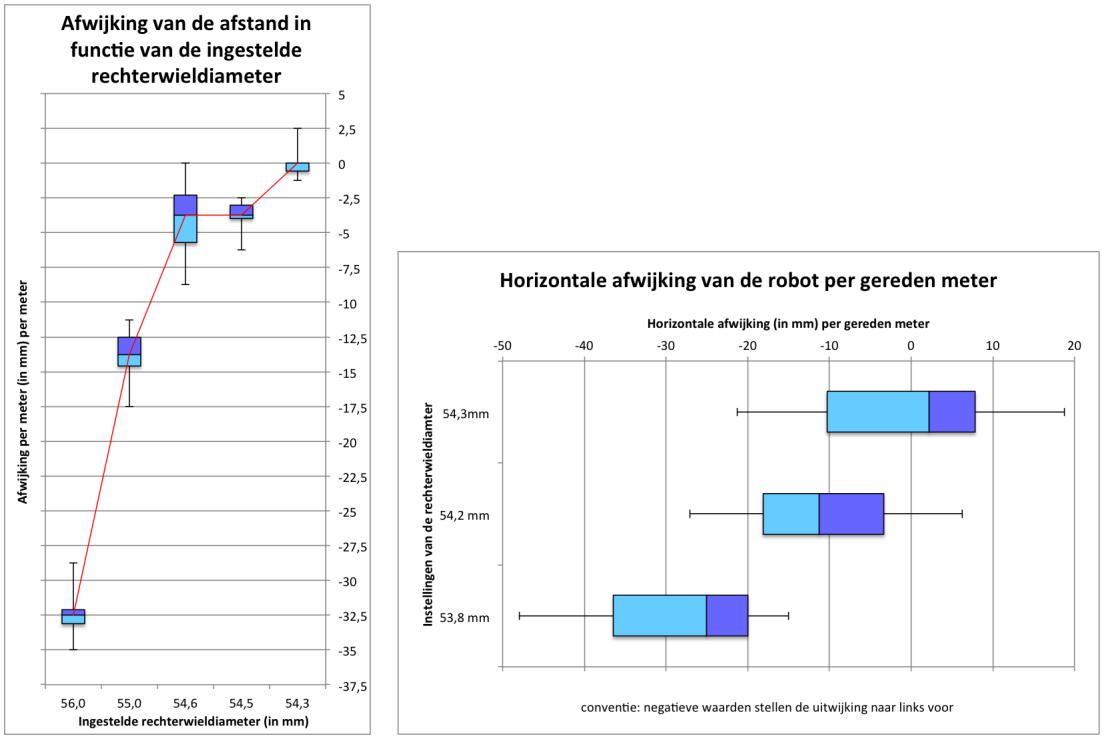
definieerde afstand te laten rijden en dan de werkelijke afstand te meten. Dit werd herhaald voor een aantal verschillende waarden van de wieldiameter. Uit de meetresultaten berekenden we de gemiddelde afwijking die de robot had ten opzichte van de geplande afstand. De robot heeft telkens drie verschillende afstanden gereden, namelijk 80 cm, 160 cm en 240 cm. Door dit meerdere keren te herhalen konden we een vrij nauwkeurig beeld vormen van de afwijking die de robot had per meter en per wieldiameter. Figuur 2(a) toont onze vaststellingen bij verschillende metingen per instelling van de wieldiameter. Het exact aantal testen per instelling is gespecificeerd onder de figuur. We zien dat bij een instelling van 54,3 millimeter de robot een gemiddelde afwijking heeft van nul millimeter per te rijden meter.

2.2.2 Afwijking op de gedraaide hoek

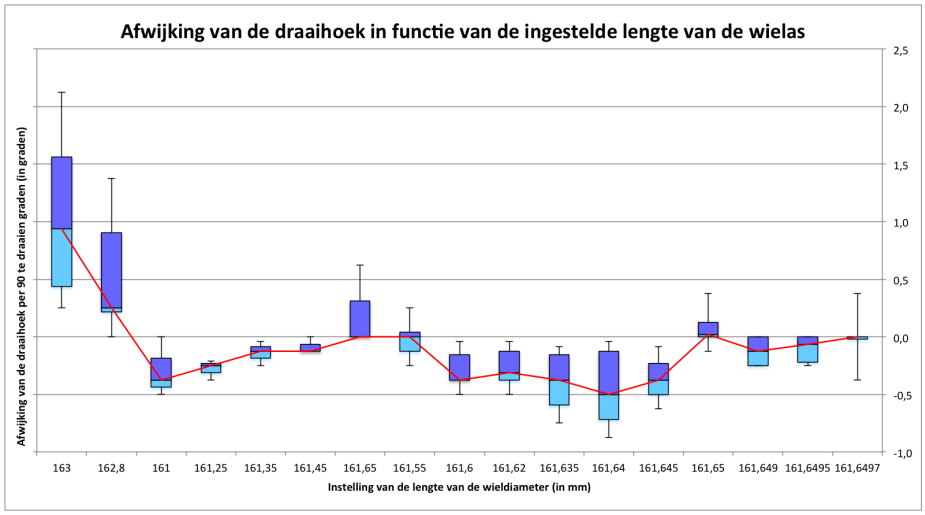
De *DifferentialPilot* heeft ook de afstand tussen de twee wielen op de wielas nodig. Deze parameter heeft vooral invloed op het draaigedrag van de robot. Om de parameter te bepalen lieten we de robot telkens een aantal vooraf gedefinieerde hoeken draaien en werd de werkelijk gedraaide hoek genoteerd. Ook dit werd herhaald voor een aantal verschillende afstanden tussen de twee wielcentra. Hier draaide de robot per waarde drie verschillende hoeken en dit opnieuw verschillende malen na elkaar. Zo kon dan de afwijking per te draaien graad berekend worden. Figuur 3 toont deze meetresultaten uitgezet *per 90 graden* met gemiddeld 6 metingen per instelling. De gegevens zijn uitgezet per 90 te draaien graden omdat de robot telkens een veelvoud van 90 graden draait in het doolhof. Hieruit blijkt dat bij een instelling van 161,6497 millimeter gemiddeld een afwijking van minder dan 0 graden per 90 graden wordt bekomen, met een maximale afwijking van 0,5 graden per 90 te draaien graden. Dit is voldoende nauwkeurig voor onze doeleinden.

2.2.3 Horizontale afwijking

Omdat de robot bij de eerste demo een redelijke afwijking naar rechts vertoonde bij het rechtdoor rijden, hebben we nog extra metingen gedaan om deze afwijking te minimaliseren. We gebruiken nu een andere constructor voor de initialisatie van *DifferentialPilot*, waarbij de wieldiameters van de twee wielen apart moeten worden opgegeven. Hoewel de twee wielen in realiteit even groot zijn, bepalen we toch twee verschillende wieldiameters. Dit komt doordat de twee motoren, die de robot aandrijven, niet exact even snel draaien. Hierdoor ontstaat er een afwijking naar links of rechts bij het rijden op een rechte lijn. Door de wieldiameter aan één kant te vergroten of verkleinen kunnen we de afwijkingen compenseren, waardoor de robot uiteindelijk rechtdoor zal rijden.



Figuur 2: **(a)** De afwijking per meter die de robot vertoonde per ingestelde wieldiameter. Positieve waarden betekenen dat de robot te ver reed ten opzichte van de geplande afstand. Er zijn $n_{56,0mm} = 9$, $n_{55,0mm} = 9$, $n_{54,6mm} = 15$, $n_{54,5mm} = 15$ en $n_{54,3mm} = 30$ metingen opgetekend. **(b)** Resultaten van horizontale afwijking van de robot bij $n_{53,8mm}$=3, $n_{54,2mm}$=9 en $n_{54,3mm}$=18 testen.



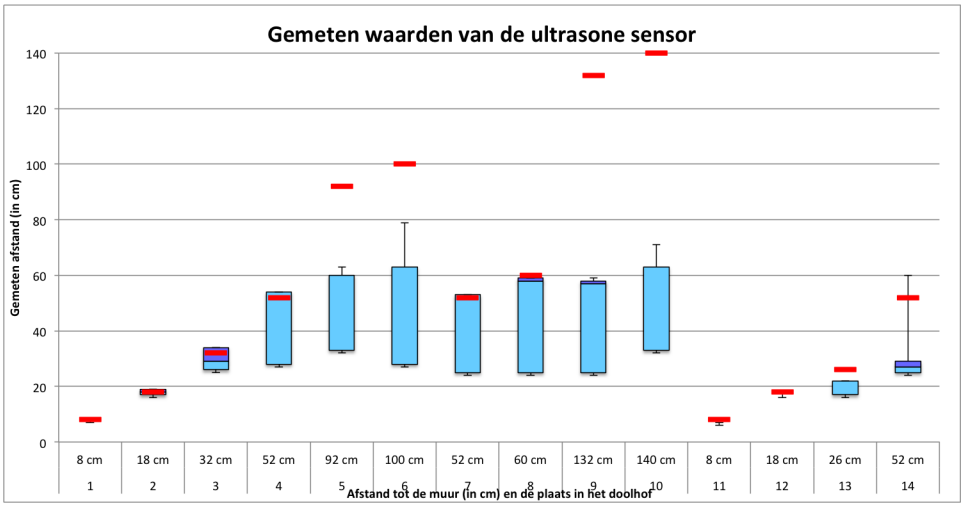
Figuur 3: Resultaten van de afwijkingen van de draaitesten met n=6, 9, 12 of 15 testen per instelling naar gelang de nauwkeurigheid van de draaiing

De proefopstelling was als volgt: we maakten de rechtse wieldiameter variabel en de linkse wieldiameter constant. Per instelling lieten we de robot drie verschillende afstanden rechtdoor rijden, namelijk 80 cm, 160 cm en 240 cm. Dan maten we herhaaldelijk de afwijking naar links of naar rechts op eenzelfde gereden afstand. Omdat de afwijking lineair toeneemt met de gereden afstand hebben we de afwijkingen per gereden meter geplot. Als na een bepaald aantal herhalingen van de proef de afwijking zo significant was, gingen we meteen verder met een nieuwe instelling voor de

wieldiameter. In Figuur 2(b) worden de resultaten van deze proef getoond, met 3 metingen voor een instelling van 53,8 mm, 9 metingen voor 54,2 mm en 15 metingen voor een instelling van 54,0 mm. Bij conventie stellen de negatieve waarden een afwijking naar links voor. Op de figuur is te zien dat bij toenemende waarde voor de rechtse wieldiameter, de gemiddelde afwijking convergeert naar nul. Bij een instelling voor de rechtse wieldiameter van 54,3 mm en een instelling van 54,4 mm voor de linkerdiameter, vinden we een gemiddelde afwijking van 2 mm naar rechts per gereden meter. De afwijking is maximaal 18 mm naar rechts of 21 mm naar links per gereden meter.

2.2.4 Ultrasone sensor

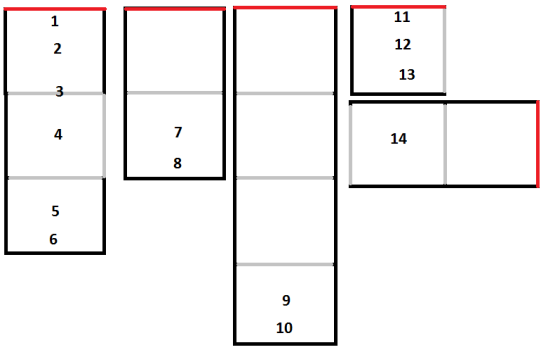
Om de ultrasone sensor te kunnen gebruiken is een begrip van de werking en limieten van deze sensor noodzakelijk. Daarom hebben we deze uitvoerig getest. In de boxplot van Figuur 4 staan de resultaten van deze testen. We zien dat de gemeten afstanden kleiner dan 20 cm heel erg betrouwbaar zijn. Dit betekent dat muren naast de robot met een grote zekerheid juist worden gedetecteerd. De afwijking bij afstanden kleiner of gelijk aan 32 cm zijn ook relatief klein, maar er zijn enkele uitschieters. Om deze fouten op te vangen wordt er bij een gemeten waarde tussen de 25 cm en 40 cm bijgehouden dat het niet zeker is of er nu een witte lijn of muur in die richting ligt. Het algoritme kan dan met deze informatie gepast reageren om het doolhof succesvol te verkennen. Verder zien we dat de gescande waarden van een muur die verder ligt dan 50 cm sterk afwijken van de echte waarden. Deze waarden kunnen echter wel nog gebruikt worden. We scannen immers enkel muren voor en langs de huidige tegel waar de robot zich bevindt. Als we dus een waarde krijgen die iets groter is dan 50 cm, is het een witte lijn. In figuur 5 is een representatie gemaakt van de locatie van de robot en welke muur we hebben gescand. Locatie 13 is een locatie waar we achterwaarts hebben gemeten, met andere woorden de sensor gedraaid naar achter. Zoals we zien in de boxplot van de vorige figuur, levert deze test foute, te kleine waarden op. Om dit op te lossen zullen we de robot zelf altijd naar achter laten draaien in plaats van de ultrasone sensor te laten draaien wanneer we in die richting willen scannen. Zoals we later zullen zien in het algoritme om het doolhof te verkennen, moeten we enkel in deze richting scannen bij het begin van dit algoritme. Het laten draaien van de robot zelf zorgt dus niet voor een groot verlies.



Figuur 4: Gemeten waarden van de ultrasone sensor bij een stilstaande robot geplot in een boxplot. De reële waarden staan aangeduid met een rode streep. We zien de duidelijke afwijkingen voor muren met een afstand van 50+ cm. Er is wel een goede nauwkeurigheid bij een afstand onder de 20 cm. Het aantal gemeten waarden ligt voor de verschillende testen rond de 150.

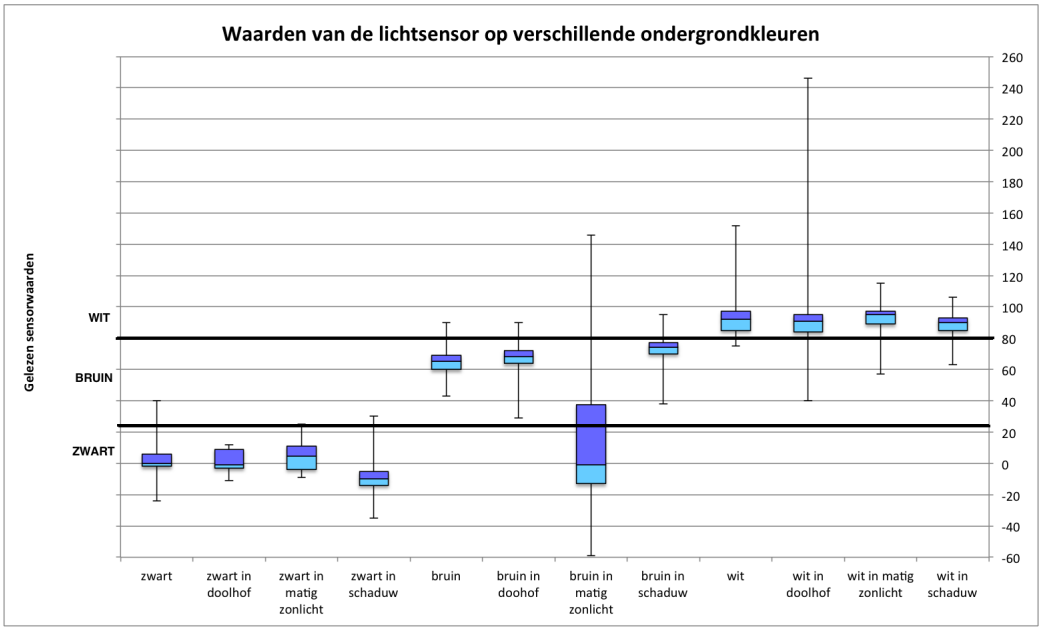
2.2.5 Lichtsensor

Om de robot zichzelf loodrecht op een witte lijn te laten oriënteren, een operatie die vaak wordt gebruikt bij het verkennen van het doolhof, is een goed begrip van de werking van de lichtsensor on-



Figuur 5: De locaties die de robot had bij de metingen in figuur 4. De rode muur duidt de gemeten muur aan. De plaats van het nummer duidt de locatie van de robot aan. Bij elke test waren de robot en de sensor naar de muur gedraaid, met uitzondering van de test in locatie 14. Hier was de sensor naar achter gedraaid.

misbaar. We hebben de lichtsensor daarom ook uitvoerig getest. In omgevingen met verschillende lichtinvallen hebben we de robot laten rijden over bruine, witte en zwarte stukken van verschillende panelen.
In de boxplots van Figuur 6 staan de resultaten van deze testen per kleur: wit, zwart en bruin. Er zijn ongeveer 260 meetwaarden opgetekend per boxplot van kleur en lichtinval. Het is duidelijk dat enkel in omgevingen met zonlicht de lichtsensor niet meer bruikbaar is. De waarden zijn immers inconsistent voor elke kleur die gescand wordt. In omgevingen met normale belichting of schaduw vallen de waarden altijd binnen dezelfde intervallen. Deze intervallen kunnen dus gebruikt worden om de waarden die de lichtsensor teruggeeft te associëren met de gepaste kleur.



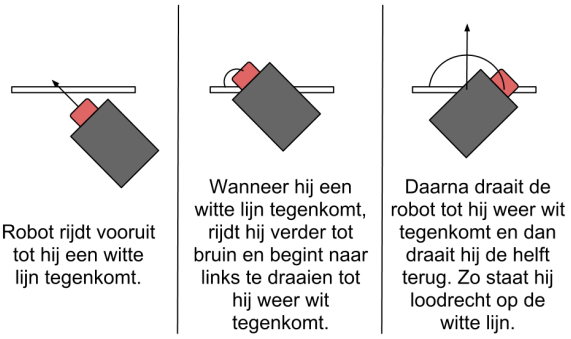
Figuur 6: De resultaten van de gelezen waarden van de lichtsensor op verschillende kleuren ondergrond en bij verschillende mogelijke lichtinvallen. Er zijn gemiddeld 260 gelezen waarden per boxplot.

3 Algoritmes

3.1 Robot

3.1.1 Rechtzetten van robot op een witte lijn

De lichtsensor scant eerst totdat vier keer een waarde overeenkomstig met een witte kleur is gevonden. Deze buffer is toegevoegd om eventuele ruis, bijvoorbeeld bij het verlichten van het paneel, tegen te gaan. Wanneer vier keer wit is gescand, gaat de robot weer op zoek naar bruine waarden. Nadat vier keer een bruine kleur is gevonden, wordt het doorsturen van positiedata naar de PC tijdelijk onderbroken. Indien dit niet zou gebeuren, kunnen er foute gegevens worden verwerkt. De robot draait vervolgens naar rechts totdat de lichtsensor weer vier keer de witte kleur heeft gescand. Daarna draait de robot naar links en houdt hij de hoek bij die gedraaid werd, om ten slotte de helft van deze hoek naar rechts te draaien en zo loodrecht geöriënteerd te staan op de witte lijn. Daarna rijdt de robot achterwaarts totdat weer vier keer een witte waarde wordt gescand door de lichtsensor. Als laatste rijdt hij nog 7,5 cm voorwaarts, zodat hij niet alleen loodrecht ten opzichte van de witte lijn is georiënteerd, maar dat zijn wielas ook op die witte lijn staat. Een grafische voorstelling van dit algoritme volgt in figuur 7.



Figuur 7: Grafische weergave van het algoritme om loodrecht op een witte lijn te oriënteren met uitleg.

3.1.2 Robot in het midden van een tegel plaatsen

Voor de robot kan beginnen met het verkennen van het doolhof, is het belangrijk dat hij perfect in het midden van een tegel staat. Hiervoor is een algoritme ontworpen. Bij de aanvang van het algoritme zal de robot zich loodrecht oriënteren op een witte lijn. Om deze witte lijn te vinden scant hij in het rond en draait hij in de richting met de maximaal gemeten afstand tot een muur. In deze richting is het namelijk het meest waarschijnlijk dat de robot een witte lijn zal tegen komen. Nadat hij zich heeft georiënteerd op de gevonden witte lijn zal de robot 20 cm naar achter rijden, om zo in de richting loodrecht op de wielas van de robot in het midden van de tegel te belanden. Daarna scant de robot links en rechts van zich en kijkt hij of de kleinste afstand kleiner is dan 20 cm. Indien dit het geval is, kan er met zekerheid besloten worden naar er in die richting een muur is in de huidige tegel. De robot draait naar deze richting en rijdt totdat de druksensor aangeeft dat de robot zich tegen de muur bevindt. Dan rijdt de robot 10 cm naar achter en draait terug naar de richting die de robot had na het voltooien van het oriënteren op de witte lijn. Als de linkse en rechtse afstand van de robot beide groter zijn dan 20 cm, is er waarschijnlijk zowel links als rechts van de robot geen muur in de huidige tegel. De robot zal dan draaien naar de richting van de maximale gemeten afstand en dan een witte lijn zoeken en zich daar dan vervolgens loodrecht op plaatsen. De reden waarom de maximale afstand wordt gebruikt en niet de minimale, is dat er met grotere zekerheid besloten kan worden dat er in die richting geen muur is ten gevolge van onnauwkeurigheden van de ultrasone sensor. Tenslotte rijdt de robot nog 20 cm naar achter en draait hij naar de gewenste richting. De robot staat dan in het midden van een tegel.

3.1.3 Barcodes lezen

De lijst van gescande kleuren wordt bepaald uit de methode *lightSensorVigilante()* (zie sectie 4.2.1). Aangezien barcodes altijd in een recht stuk liggen is het onmogelijk om een barcode onvolledig te hebben ingelezen door bijvoorbeeld te draaien. Bovendien beginnen en eindigen de barcodes altijd met een zwarte strook en zijn ze altijd 8 stroken lang. Met deze gegevens kan dan gemakkelijk de lijst van gescande kleuren worden geconverteerd naar lijnen of barcodes. Zoals aangeven in sectie 4.2.1, wordt de lijst verwerkt in de *Barcode*-klasse. Daar wordt de lijst in 8 delen opgesplitst. Elk deel van de lijst stelt nu een strook van de barcode voor. Vervolgens wordt in elk deel geteld hoeveel keer er een witte of zwarte kleur voorkomt. De kleur die het meest voorkomt, is de kleur van de strook en deze wordt dan omgezet naar het bijbehorend binair equivalent (0 voor zwart en 1 voor wit). Er wordt gecontroleerd of de eerste en de laatste strook zwart zijn. Indien dit niet het geval is, wordt de barcode opnieuw gelezen. Daarna worden de eerste en de laatste bits van de barcode eraf geknipt, zodat er een barcode van 6-bit overblijft. Ten slotte wordt er nog gecontroleerd of de barcode gekend is en dus gekoppeld is aan een bepaalde actie. Indien dit niet het geval is wordt deze gespiegeld. Als de barcode gekend is, wordt de gekoppelde actie uitgevoerd. Vervolgens wordt de barcode doorgestuurd naar de PC. Als de barcode ook na spiegeling niet gekend is, wordt dit aan de PC gemeld en zal deze gepaste acties ondernemen. Dit wordt verder uitgediept in sectie 3.2.3. Deze procedure geïntegreerd met de lichtsensor is te zien in figuur 12.

3.2 PC

3.2.1 Rijden van een veelhoek

De lengte van de zijde en het aantal hoeken moeten worden ingegeven in de GUI. Vervolgens wordt de hoek die moet gedraaid worden berekend door 360 graden te delen door het aantal hoeken. Met behulp van een for-lus wordt n keer de gekozen afstand voorwaarts gereden, telkens gevolgd door het draaien van de berekende hoek.

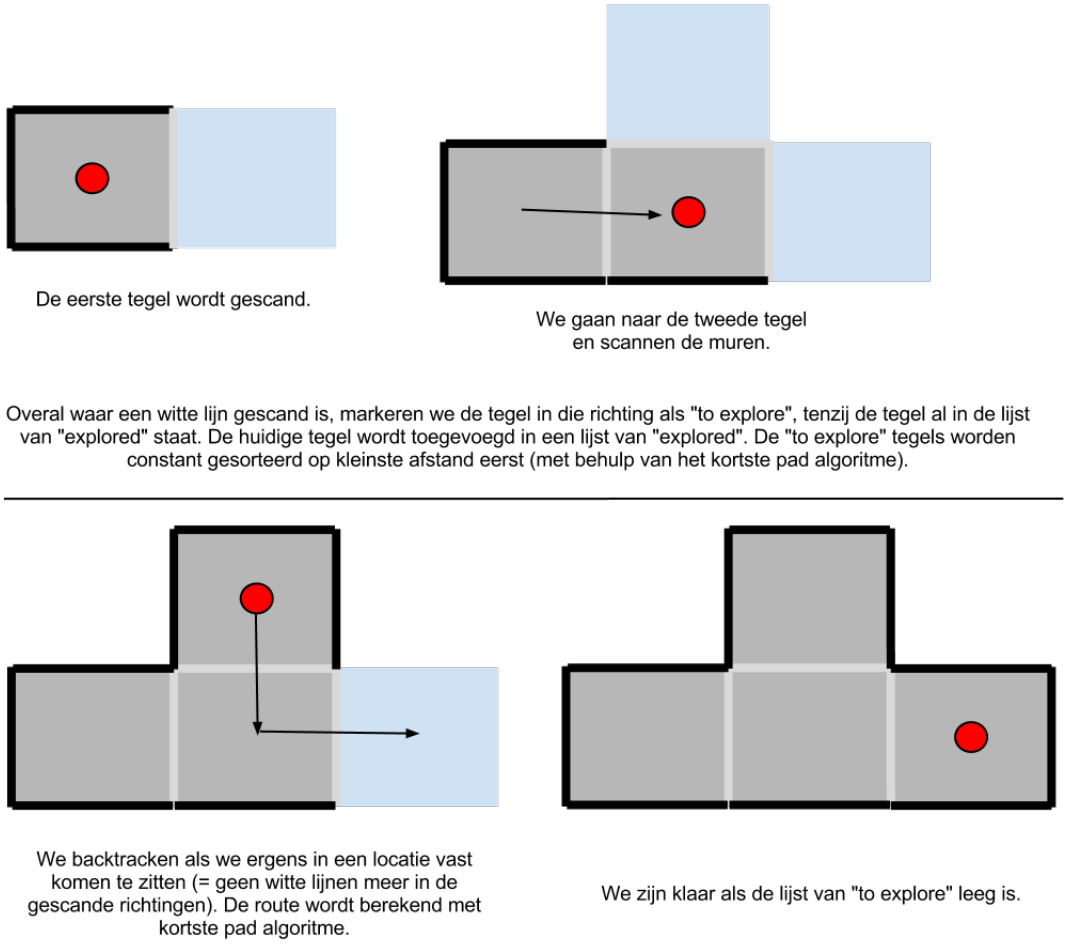
3.2.2 Kortste Pad Algoritme

Ons kortste pad algoritme is volledig identiek aan het A* algoritme[8]. De conversie van de doolhof-tegels naar de zoeknodes voor de A* zoekboom is heel simpel. Dit is omdat de tegels van het doolhof een grid gebaseerde positie hebben. Zo kunnen we in dit grid een simpel A* zoekproces opstarten, met de Manhattan afstand als heuristiek. Er zijn eventueel nog overgangen die illegaal zijn, dit is bijvoorbeeld het geval wanneer er een muur staat tussen twee tegels.

3.2.3 Doolhof verkennen

Voor het verkennen van het doolhof gebruiken we een variatie van het diepte-eerst zoekalgoritme. We beginnen met het scannen van de huidige tegel. Overal waar deze tegel een witte boord heeft, voegt het algoritme de tegel in die richting toe aan de lijst van 'te verkennen tegels'. Dan wordt de lijst gesorteerd op de af te leggen afstand vanaf de locatie van de robot. Vervolgens neemt de robot de eerste tegel uit de lijst, gaat daar naar toe met het kortste pad berekend door het A* algoritme en scant de tegel. Dit blijft doorgaan tot de lijst van 'te verkennen tegels' leeg is, wat betekent dat het volledige doolhof is verkend.

Ook zijn er enkele vangnetten in het algoritme aanwezig omdat onder andere de sensoren niet 100% betrouwbaar zijn. Er is bijvoorbeeld iets voorzien voor de onzekere boorden bij gemeten waarden tussen 25cm en 40cm van de ultrasone sensor. Indien er een onzekere boord ontdekt is, krijgt de robot de instructie om dichterbij te gaan staan. Vervolgens wordt er gescand tot er een zinvolle waarde teruggegeven is en het dus zeker is of er ofwel een muur of een witte lijn voor de robot ligt. Ook is er een vangnet voorzien voor het lezen van ongekende barcodes. Wanneer dit gebeurt, krijgt de robot de instructie om er nogmaals over te rijden, zodat hij de barcode opnieuw leest.
Ook zijn er enkele voorzieningen die het verkennen van het doolhof wat sneller laten verlopen. Bijvoorbeeld bij het ontdekken van een barcode wordt er niet gestopt om de muren te scannen, omdat we toch weten dat er enkel twee muren aan de zijkanten zijn. Een grafische representatie van de werking van dit algoritme is te zien in figuur 8.



Figuur 8: Verduidelijking hoe het doolhof verkend wordt. De grijze tegels stellen de al verkende tegels voor, de blauwe zijn de tegels die nog verkend moeten worden. De figuur linksboven stelt de initiële positie voor, de figuur rechts onder stelt een volledig verkend doolhof voor. Meer uitleg over de navigatie staat onder de figuur zelf.

4 Software

4.1 Software design

Het project is opgedeeld in twee Java-projecten: het ene wordt uitgevoerd op de fysieke robot en het andere op de PC.

- Het Java-project op de fysieke robot maakt enkel gebruik van de LeJOS-bibliotheken. Dit project bevat de decodering van de commando's die via de PC worden doorgestuurd en de verschillende klassen van sensoren.
- Het PC-project bevat alle andere onderdelen: de gebruikersinterface, de simulator, de gemeenschappelijke denklaag en de benodigdheden voor het verbinden met de fysieke robot via bluetooth.

We hebben voor twee aparte projecten gekozen om te vermijden dat het bestand op de echte robot interfereert met de standaard Java-bibliotheken. De LeJOS bibliotheken bevatten namelijk bepaalde klassen die dezelfde naam hebben als de standaard Java-bibliotheken.

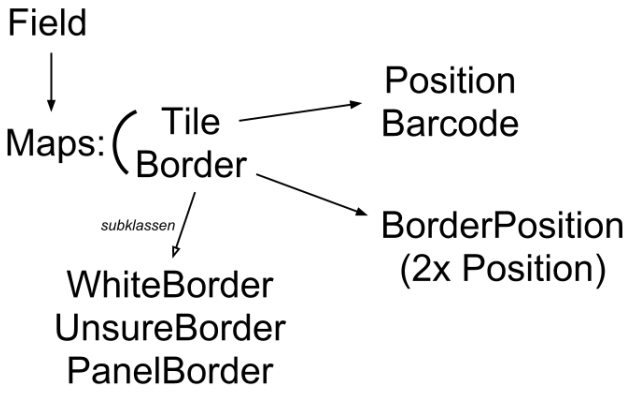
In het PC-project hebben we een denklaag gemaakt, die gemeenschappelijk is aan robot en simulator. Deze denklaag bevat bijvoorbeeld de logica om zowel robot als simulator het doolhof te laten

verkennen. Een groot voordeel van deze laag is dat verschillende algoritmes maar één keer moeten geïmplementeerd worden en toch gebruikt kunnen worden door zowel de robot als de simulator. In de denklaag wordt tevens een positie bijgehouden ten opzichte van het centrum van de huidige tegel waar de robot zich bevindt. Uiteraard staat deze laag in directe verbinding met de GUI.

4.1.1 Doolhof architectuur

Voor de voorstelling van het veld hebben we onze eigen architectuur opgebouwd. De reden hiervoor is dat we deze zo veel mogelijk aan onze noden wilden laten voldoen. Dit om het berekenen van het kortste pad zo makkelijk mogelijk te maken. De structuur is ook gemaakt op een manier die dicht aanleunt bij de probleemstelling.

Er is een centrale veldklasse. Deze houdt een mapping van posities naar tegels en van boordposities (= twee posities) naar boorden bij. Zo kunnen tegels op bepaalde posities en aangrenzende/tussenliggende boorden heel makkelijk opgevraagd worden. In de veldklasse zijn enkele methodes aanwezig om tegels/boorden toe te voegen en zaken op te vragen. De tegelklasse houdt zijn positie bij en een barcode, indien die aanwezig is. Met de positie van een tegel wordt de positie op het rooster van het doolhof bedoeld. De boordklasse is een abstracte klasse waar er twee subklassen van zijn: witte boord (*WhiteBorder*) en boord met paneel (*PanelBorder*). De boordpositie van een boord wordt voorgesteld door de twee posities waartussen die boord ligt. In de boordpositie is er voorziening bijvoorbeeld om symmetrische boorden correct te testen op gelijkheid. De barcode wordt ook voorgesteld door een aparte klasse en houdt een array bij van zes gehele getallen om de waarde van de barcode op te slaan. Als laatste is er nog een richtingsklasse die enkele methodes bevat die richtingsgerelateerd zijn.
Een visuele representatie van deze veld-package is te zien in figuur 9.



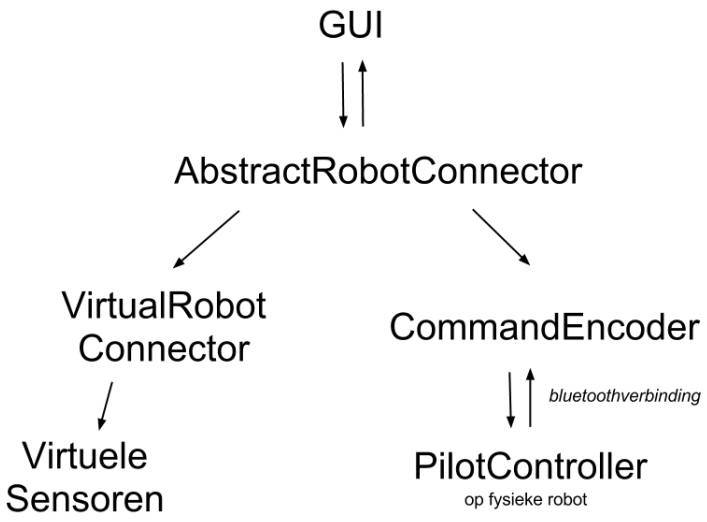
Figuur 9: Schema van de werking van de *veldklasse*.

4.1.2 Simulator architectuur

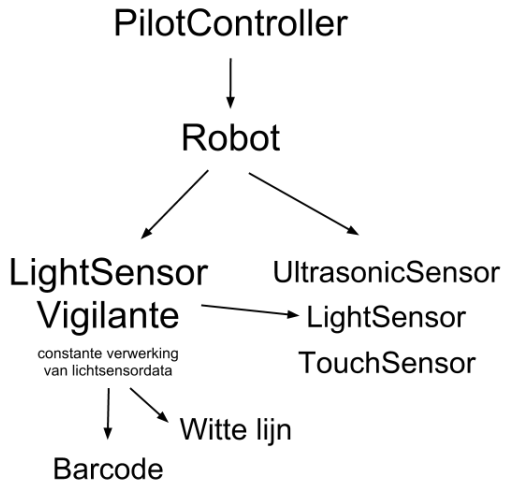
Voor de simulator is er een centrale klasse *VirtualRobotConnector*. Deze erft over van de *AbstractRobotConnector* en symboliseert de connectie tussen de PC en de virtuele robot. Deze centrale klasse bevat de interne werking die de beweging van de fysieke robot simuleert. Daarnaast zijn er de sensoren die de werking van de fysieke sensoren simuleren. Voor uitgebreide uitleg van de werking van deze architectuur zie sectie 4.4. Een voorstelling van de klassen staat in figuur 10.

4.1.3 Robot architectuur

De robot heeft een centrale klasse *Robot*. Deze onderhoudt communicatie met de verschillende sensoren. De communicatie van de robot naar de PC is de verantwoordelijkheid van de klasse *PilotController*. Een uitgebreider overzicht wordt gegeven in figuur 11.



Figuur 10: Overzicht hoe en welke klassen met elkaar gelinkt zijn op de PC. De twee onderste elementen zijn voor de simulator, de rechtse voor de reële robot. De figuur toont duidelijk de parallelle structuur van virtuele en reële robot.



Figuur 11: Grafische weergave van de structuur op de robot zelf en welke sensoren deze gebruikt.

4.2 Robot

Bij de aanvang van het programma op de robot wordt de methode om de lichtsensor te calibreren opgeroepen. Nadat de calibratie gebeurd is, wordt er gewacht totdat er een verbinding is gemaakt met de PC.

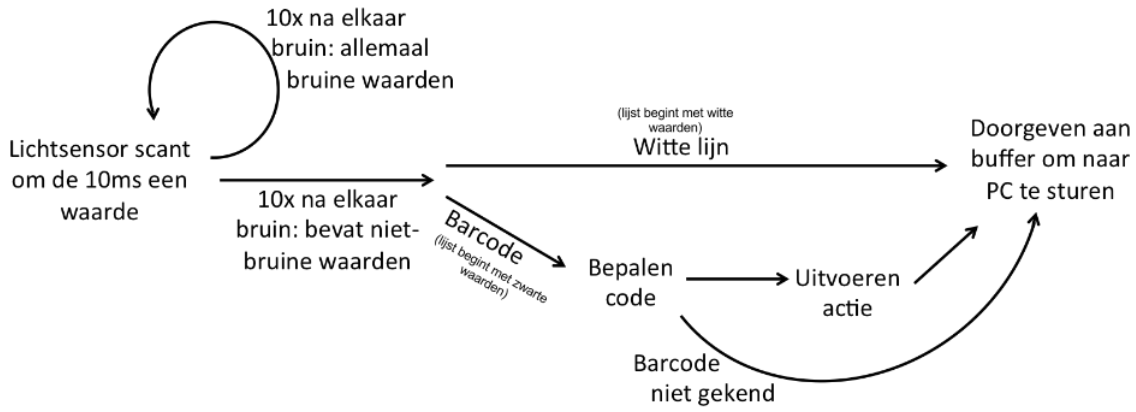
4.2.1 Lichtsensor

Deze klasse bevat methodes om te lichtsensor te calibreren, een kleur uit te lezen met de lichtsensor en een methode *lightSensorVigilante()*. Deze laatste is in staat lijnen en barcodes te herkennen en terug te geven. Ze wordt constant aangeroepen in een aparte *thread* om te controleren of de robot een lijn of een barcode gepasseerd is.

De lichtsensor voert om de 10 milliseconden een meting uit en voegt de kleur, behorende bij deze gemeten waarde, toe in een lijst van kleuren. Als er dan een aantal zwarte waarden na elkaar

wordt gedetecteerd met hiervoor enkel bruine waarden, weten we met zekerheid dat de robot over een barcode aan het rijden is. Aangezien het onmogelijk is om aan hoge snelheid over een barcode te rijden en deze correct te lezen, zal de robot een beetje naar achter rijden aan het begin van een barcode. Dan verlagen we de snelheid van de robot en rijden we over de barcode. Vervolgens zetten we de snelheid van de robot terug naar zijn oorspronkelijke snelheid. Om te voorkomen dat bij stilstand van de robot op een barcode er een onevenwicht wordt gecreëerd door het continu toevoegen van de huidige kleur, wordt de lijst enkel aangevuld als de robot in beweging is. Bovendien worden enkel witte en zwarte kleuren in deze lijst bewaard.
De methode eindigt bij het lezen van 10 bruine waarden na elkaar of indien de lijst leeg is. Het is nodig dat deze methode regelmatig eindigt om bepaalde instellingen te kunnen aanpassen. Het is namelijk soms wenselijk om geen barcodes door te geven aan de PC, bijvoorbeeld als de robot de huidige tegel al heeft gescand.

Als de robot dan 10 keer na elkaar een bruine kleur heeft gedetecteerd, weet hij dat de lijn of de barcode volledig achter zich ligt. Is de lijst leeg, dan moet er niets ondernomen worden. In het andere geval moet de lijst van gelezen kleuren geconverteerd worden naar een barcode of een witte lijn. Indien de gelezen lijst van kleuren begint met een aantal witte waarden, weten we zeker dat de robot over een witte lijn heeft gereden. De methode geeft terug dat er een lijn is gepasseerd. In het andere geval zijn we zeker van een barcode en moet de code hiervan bepaald worden. Dit wordt gedaan de *Barcode* klasse. Voor verdere uitleg hierover wordt er verwezen naar sectie 3.1.3. Een schema waarin deze hele procedure is uitgelegd, is gegeven in figuur 12.



Figuur 12: Een visuele representatie om barcodes een witte lijnen te detecteren. Het omvat de methode *lightSensorVigilante()* en andere methodes die deze methode aanroept bij de bepaling van de ondergrond van de robot.

4.2.2 Ultrasone sensor

De klasse van de ultrasone sensor bevat methodes om de sensor te richten in een bepaalde richting en metingen uit te voeren. Ook is er een basis methode voorzien om in alle vier richtingen direct na elkaar te scannen. De sensor stuurt een korte geluidsimpuls en wacht vervolgens 100 ms. Dan wordt de afstand tot het obstakel gevraagd. Vervolgens draait de sensor 90 graden naar links en doet hij een nieuwe meting. Dit doet hij tot er in alle richtingen is gescand. Aangezien deze sensor veel ruis heeft (zie sectie 2.2.4), is er een robuust algoritme geschreven om deze gegevens juist te kunnen interpreteren.

4.2.3 Druksensor

De implementatie van de druksensor is vrij triviaal. Er worden waarden doorgegeven die opgevraagd worden aan de LeJOS-bibliotheken.

4.3 Bluetooth

4.3.1 Connectie maken

De connectie met de robot wordt altijd opgestart vanaf de PC. Deze zoekt naar de robot met behulp van zijn naam en MAC-adres. De specificatie van het MAC-adres zorgt ervoor dat de connectie op een directe en snelle manier gemaakt kan worden.

4.3.2 Communicatie van PC naar Robot

Aan elke opdracht die moet gegeven worden aan de robot is er een nummer gekoppeld. Nadat dit nummer is doorgegeven kunnen er ook nog twee numerieke parameters doorgegeven worden en een booleaanse waarde. Op de robot wordt dit nummer opnieuw gedecodeerd en wordt de actie die ermee verbonden is uitgevoerd.

4.3.3 Communicatie van Robot naar PC

In de andere richting is er uiteraard ook communicatie. Informatie over de positie van de robot, of die aan het bewegen is en of die aan het scannen is, wordt constant doorgegeven.

De meeste informatie die afkomstig is van sensoren wordt op de robot zelf verwerkt tot bruikbare informatie en daarna doorgestuurd naar de PC.
Voor het teruggeven van afstanden tot de muren is er niet zo'n groot probleem. De metingen worden doorgestuurd samen met de bijhorende positie (in vier richtingen) ten opzichte van de robot. Op de PC kunnen deze dan geïnterpreteerd worden.
Als de robot in meer dan vier richtingen scant, worden deze ruwe waarden ook doorgestuurd naar de PC. Dit is vooral handig om te kunnen debuggen, bijvoorbeeld om te weten op wat de robot zijn richtingskeuze bij het oriënteren op een witte lijn baseert.

De lichtsensor moet constant informatie verzamelen, want hij moet kunnen detecteren wanneer de robot in een nieuw vakje van het doolhof terechtkomt om de positiebepaling niet in het gedrang te brengen. Die constante stroom aan informatie wordt op de robot verwerkt en wanneer de robot een witte lijn of een barcode tegenkomt, wordt enkel WHITELINE of BARCODE doorgestuurd naar de PC via een buffer. Via een andere buffer stuurt de robot dan nog de desbetreffende barcode door wanneer hij klaar is met die volledig te lezen.

Naast deze verwerkte vorm van informatie, geeft de lichtsensor ook nog ruwe gegevens door aan de PC. Deze gegevens dienen vooral om te debuggen en de nauwkeurigheid van de sensoren na te gaan. Deze gegevens worden doorgestuurd aan een iets lagere frequentie dan hij effectief meet, maar deze is echter voldoende om de gegevens te kunnen interpreteren.

De druksensoren geven elke 100 milliseconden (frequentie voor het doorsturen van informatie) terug of één van hen ingedrukt is.

4.4 Simulator

Onze visie op de simulator is dat hij zich nagenoeg identiek gedraagt als de robot. Daarom hebben we een timer in de simulator-klasse toegevoegd. Deze timer staat onder andere toe het continue rijgedrag van de robot te simuleren. De positie van de robot varieert immers continu. De robot heeft een bepaalde bewegings- en draaisnelheid per clocktick van de timer. Met behulp van deze informatie kunnen we de robot een bepaalde afstand laten rijden of een bepaalde hoek laten draaien. Dit gebeurt door bij het initialiseren het aantal clockticks dat nodig is om de totale afstand te rijden, te berekenen en vervolgens bij elke clocktick dat aantal te decrementeren. De methodes hebben hier ook een volledig identieke structuur als bij de echte robot. Hierdoor kunnen we heel simpel zowel de robot als de simulator aansturen op eenzelfde manier. Dit vergemakkelijkt de opbouw van de klassen die robot en simulator gemeenschappelijk hebben.

4.4.1 Doolhof in simulator

Het virtuele doolhof dat verkend moet worden door de simulator wordt intern bijgehouden. Hierdoor kan de simulator over een witte lijn rijden en muren detecteren. Deze boorden hebben een vaste dikte die uitsteekt bij de rand van de tegel. De simulator houdt ook nog zijn werkelijke positie intern bij. Dit is nodig om correct tegen muren te kunnen botsen en witte lijnen te detecteren. Dit is parallel aan de echte robot omdat er ook kleine fouten kunnen optreden tussen de positie die we op de PC bijhouden van de robot en de positie die de robot in werkelijkheid heeft.

4.4.2 Lichtsensor in simulator

De lichtsensor in de simulator leest de waarden in op basis van de positie van de gesimuleerde robot. Als de robot zich bevindt op een witte boord, dan geeft hij de waarde wit terug. Wanneer hij zich niet bevindt op een witte boord betekent dat dus dat de robot op een paneel staat en wordt bijgevolg de waarde bruin teruggegeven. Ook worden de lichtwaarden van barcodes ingelezen. Het doolhof berekent de locaties van de stroken van de barcode. Wanneer de robot dan op één van die stroken terechtkomt wordt de juiste lichtwaarde teruggegeven. Barcodes inlezen in de simulator is niet geïmplementeerd omdat we onze algoritmes rechtstreeks met echte barcodes hebben getest.

4.4.3 Ultrasone sensor in simulator

De ultrasone sensor wordt gesimuleerd door de afstand tot de muren in de vier richtingen (voor, achter, links en rechts) terug te geven. Dit is dezelfde soort informatie als deze die de robot naar de PC doorstuurt. Aangezien de sensor in de simulator een analoog gedrag moet vertonen als op de robot, werd er ruis geïntroduceerd. Deze ruis wordt berekend op basis van de gemeten werkelijke afwijking van de ultrasone sensor.

4.4.4 Druksensor in simulator

Voor de druksensor wordt per clocktick aan de hand van zijn positie in de simulator gecontroleerd of de robot een muur raakt. Hier is het niet nodig om ruis te introduceren, aangezien de druksensor zowat de meest betrouwbare sensor is.

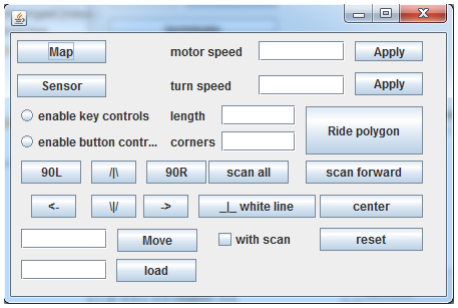
4.5 Grafische gebruikersinterface (GUI)

4.5.1 Structuur en vorm

De grafische gebruikersinterface is ontworpen met behulp van Windowbuilder Pro. Ze is erg gestructureerd met verschillende deelgroepen. Dit zorgt ervoor dat alles gemakkelijk uitbreidbaar is en dat het scherm overzichtelijk blijft.

4.5.2 Input

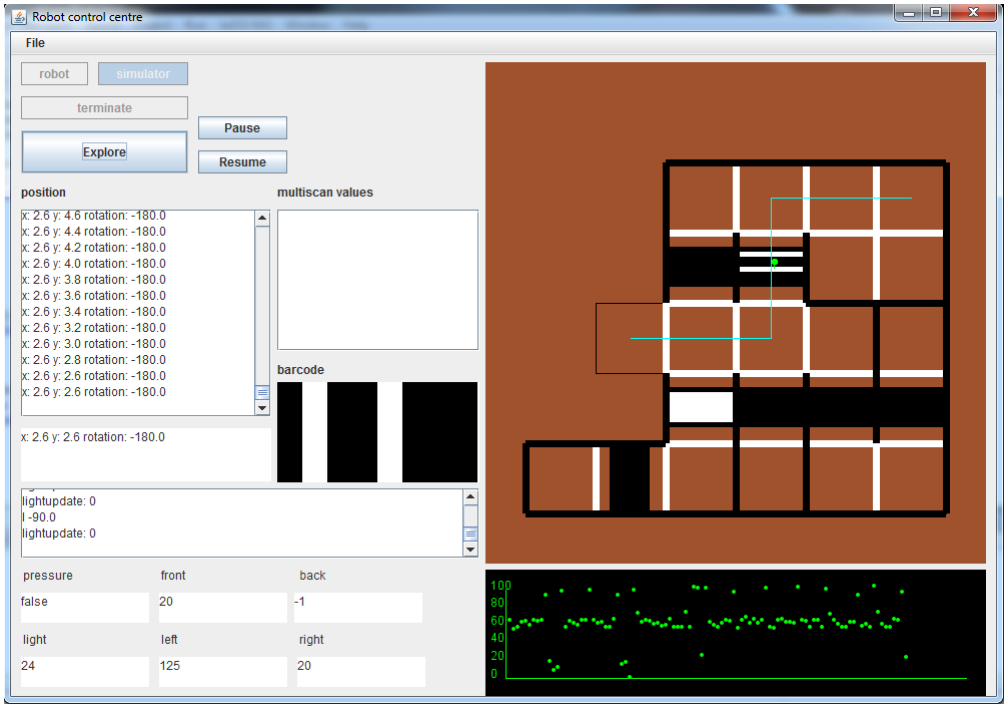
Om de GUI ordelijk te houden zijn de meeste invoer opties weggestoken onder het menu File > *Advanced*, enkel de belangrijkste staan nog op het hoofdscherm. Allereerst zijn er de instellingen. De keuze of er met de robot gewerkt wordt of met de simulator. Hierbij hoort ook de mogelijkheid om deze keuze en elke huidige opdracht die bezig is te termineren. In het *Advanced* venster zitten knoppen om bijvoorbeeld de snelheid van de motoren in te stellen, of een knop om over te gaan op handmatige besturing. Met andere woorden bevat het *Advanced* venster alle functionaliteit die nodig was voor de vorige demo´s. Veel van deze functies maken het testen van de verschillende algoritmes namelijk een stuk eenvoudiger. Op het hoofdscherm is dus enkel de functionaliteit die nodig is voor demo 3 overgebleven. Namelijk een knop *Explore* voor de autonome verkenning van het doolhof. Hiernaast is er ook nog een pauzeer en hervat knop indien dit nodig zou blijken tijdens de demo. In figuur 13 en figuur 14 staan respectievelijk het *Advanced* venster en het hoofdscherm van de GUI. De andere onderdelen op het hoofdscherm zijn de outputkaders. Deze worden uitvoerig besproken in de volgende sectie.



Figuur 13: Het *Advanced* venster van de GUI. Met deze functies zijn er een aantal zeer triviale acties mogelijk. Dit venster is nuttig bij een fout in uitvoering of om de robot handmatig te besturen.

4.5.3 Output

Als tekstuele uitvoer is er een venster dat de posities van de robot weergeeft en een extra venster dat steeds de huidige positie van de robot bevat. Deze positie bestaat uit een x-coördinaat, een y-coördinaat en een richting. Ook de waarden van de sensoren worden tekstueel weergegeven. Het venster met label *pressure* toont de toestand van de druksensor en het venster met label *light* toont de laatst gelezen waarde van de lichtsensor. De ultrasone sensor heeft meerdere uitvoervensters. Wanneer de robot in 10 richtingen scant, zoals bij het rechtzetten in een tegel, worden deze waarden weergegeven in het venster met label *multiple scan values*. In het geval dat de robot enkel in 4 richtingen scant, zoals bij het verkennen van de grenzen van elke nieuwe tegel, worden deze waarden getoond in de vensters met labels: *front*, *back*, *left* en *right*. Als laatste is er nog het debugvenster dat meer informatie geeft over de robot en zijn acties.



Figuur 14: Grafische weergave van de GUI. Links bovenaan staan de knoppen om te verbinden met de robot of simulator en het doolhof te verkennen. Er zijn nog een aantal vensters die extra informatie geven over de positie, de waarden van de sensoren en de gelezen barcode. Rechts is een grafische weergave van het al verkende doolhof, met daaronder de waarden van de lichtsensor weergegeven in een grafiek.

De grafische uitvoer bestaat uit een grafiek van de waarden van de lichtsensor, een barcode canvas en een kaart. De grafiek met lichtwaarden zal steeds de laatste 100 doorgegeven lichtwaarden tonen. Nieuwste waarden worden achteraan toegevoegd en de oudere waarden worden vooraan verwijdert. Het canvas dat barcodes toont zal voor het huidige vakje de bijbehorende barcode tonen met aan beide kanten een zwarte rand. Als de robot op een vakje staat zonder barcode dan is dit canvas volledig wit. Als laatste grafische uitvoer is er de kaart. Dit is een weergave van alle verkende tegels en boorden in het doolhof. De robot zelf wordt voorgesteld als een groen bolletje met een lijn die de richting aanduidt. Alle vakjes worden getekend als vierkanten met dunne rand. Grensobjecten worden getekend als gevulde rechthoeken, witte rechthoeken zijn witte lijnen, grijze rechthoeken zijn grenzen die niet duidelijk gedetecteerd zijn en zwarte rechthoeken zijn muren. Ook barcodes worden op de kaart getekend in de richting waarin ze liggen. Als laatste wordt er een lichtblauwe lijn getekend tussen alle vakjes die onderdeel zijn van een kortste pad berekening. De robot zal steeds deze lijn volgen bij het verkennen van het doolhof.

5 Besluit

De robot is heel nauwkeurig bij het rechtdoor rijden en draaien. Door deze minimale afwijkingen blijft het aantal botsingen met muren beperkt en wordt er niet te veel tijd verspild door te moeten heroriënteren in het midden van de verkenning van het doolhof. De ultrasone sensor is voor kleine afstanden een betrouwbare sensor. Voor grotere afstanden zijn de waarden veel minder bruikbaar. Ook krijgen we storing van de robot als we de sensor zelf draaien. Dit is het meest merkbaar als we met de sensor naar achter kijken terwijl de robot naar voor gericht is. Ook de lichtsensor is in het algemeen een betrouwbare sensor. Tussen twee stroken van zwart en wit geeft de sensor waarden aan die overeenkomen met bruin. Verder zijn de GUI en simulator handige hulpmiddelen geweest om algoritmes te testen. Ten slotte heeft de interne representatie van het doolhof het berekenen van het kortste pad en het verkennen van het doolhof ook bijzonder eenvoudig gemaakt.

A Demo 1

A.1 Resultaten

De robot reed redelijk nauwkeurig een willekeurige veelhoek en was volledig bestuurbaar vanuit de GUI. De omgekeerde communicatie werkte nog niet waardoor op de GUI foutieve debuginformatie kwam te staan.

A.2 Conclusies

Er moet meer getest worden. Het is zeer belangrijk om de uiterste limieten van de robot te verkennen. Er waren onvoldoende gegevens beschikbaar en de beschikbare gegevens moeten beter worden weergegeven in het verslag.

A.3 Oplijsting aanpassingen verslag

- Sectie 2.1: Hier is de zogenaamde "dagboekvorm"van het beschrijven hoe we tot het uiteindelijke ontwerp zijn gekomen veranderd naar een objectieve beschrijving van hoe de robot er momenteel uitziet en welke voordelen dit ontwerp met zich meebrengt.
- Sectie 3.5: Deze subsectie is verplaatst van de 'Software' sectie naar de 'Algoritmes' sectie, deze laatste was oorspronkelijk afwezig.
- Sectie 3.3: Ook hier is de zogenaamde "dagboekvorm"geschrapt en aangepast naar een objectieve beschrijving van hoe de GUI eruit ziet en wat de functionaliteit ervan is.
- Er is een samenvatting van het verslag toegevoegd.

B Demo 2

B.1 Resultaten

Het oriënteren op een witte lijn verliep zonder problemen, maar het blind besturen van de robot vanuit de PC ging moeilijker. De ultrasone sensor is nog niet 100% correct afgesteld waardoor hij enkele keren foute waarden doorgaf. Hierdoor kwam het getekende doolhof niet volledig overeen met het werkelijke doolhof. Ook faalde de bluetooth connectie van robot naar pc in het midden van de demo, waardoor deze herstart moest worden. Ten slotte was de GUI te vol waardoor deze niet overzichtelijk was.

B.2 Conclusies

De bluetooth connectie moet zeker in orde gemaakt worden voor de derde demo. Indien dit niet mogelijk is zal de robot het algoritme om het doolhof te verkennen zelf moeten uitvoeren in plaats van dat dit op de PC gebeurt. Dit is mogelijk, maar zorgt ervoor dat we veel dubbel werk hebben om dezelfde functionaliteit in de simulator te bekomen. Daarom ligt de voorkeur bij het op punt stellen van de bluetooth. Daarnaast moet er opnieuw met de ultrasone sensor getest worden, zodat de werking van deze sensor beter begrepen wordt en de data beter aangewend kan worden om de robot aan te sturen.

B.3 Oplijsting aanpassingen verslag

- Abstract: Niet meer een soort begeleiding doorheen de tekst, maar een korte samenvatting van het hele verslag.
- Sectie 2: verwijzing naar referenties verwijderd
- Captions van figuren zijn uitgebreider.

- Klassendiagramma: kleinere figuren van enkel de belangrijke figuren met een beetje toelichting.
- Algemeen: Geen lappen tekst gevolgd door vele figuren meer, maar figuren geïntegreerd in de tekst.

C Demo 3

C.1 Resultaten

De simulator verkende het doolhof snel en met succes. De fysieke robot heeft ook een aanzienlijk stuk van het doolhof volledig juist verkend. Aan het einde maakte hij echter een fout door een barcode te missen en zich recht te zetten op een witte lijn van deze barcode. Dit probleem bleek ook niet in het verslag aangegeven te zijn. De problemen met bluetooth van vorige keer zijn opgelost en de GUI was een stuk duidelijker.

C.2 Conclusies

Er moeten nog enkele kleine bugs weggewerkt worden zodat het algoritme om het doolhof te verkennen niet meer kan falen. Verder moet er gezorgd worden dat het verslag volledig overeenkomt met wat we laten zien tijdens de demo.

C.3 Oplijsting aanpassingen verslag

- Sectie 3.1.1: de robot zet zich nog steeds recht op witte lijnen van barcodes.

D Beschrijving van het proces

- **Welke moeilijkheden heb je ondervonden tijdens de uitwerking?**
  Verschillende threads maakten gebruik van dezelfde data en zorgden zo voor problemen in bepaalde algoritmes. Deze problemen waren meestal niet evident op te lossen.
  Ons grootste probleem was ongetwijfeld de bluetoothcommunicatie. Deze is pas tegen de derde demo helemaal in orde gebracht. Er is zeer veel tijd gegaan naar het opsporen van die fout, door verschillende personen. Uiteindelijk bleek het probleem te zijn dat er automatisch voortdurend nieuwe threads werden gestart, die allemaal gegevens wilden ontvangen.
  Verder is het niet altijd makkelijk om de tijd optimaal te benutten als drie of meer mensen tegelijkertijd de robot nodig hebben om allemaal verschillende testen uit te voeren.
- **Welke lessen heb je getrokken uit de manier waarop je het project hebt aangepakt?**
  Je kan nooit teveel testen. Duidelijke afspraken moeten gemaakt worden om het werken in team zo optimaal mogelijk te maken. Niet te lang allemaal stilstaan bij één specifiek probleem wanneer andere problemen onafhankelijk van dit probleem opgelost kunnen worden.
- **Hoe verliep het werken in team? Op welke manier werd de teamcoördinatie en planning aangepakt?**
  De werkverdeling kon iets beter verlopen, maar langs de andere kant is het moeilijk om op voorhand in te schatten wat de moeilijkheidsgraad en tijdsbestek gaat zijn van een bepaalde taak. Verder werd er een duidelijke planning gemaakt met behulp van Google Docs. Voor de aanvang van elke sessie werd ook besproken hoe ver iedereen met zijn/haar taak stond. Wanneer bepaalde taken eerder klaar waren dan de voorziene tijd werden personen opnieuw aangewezen om bijvoorbeeld te helpen bij taken die iets moeilijker bleken. Zo wist iedereen op elk moment wat zijn of haar verantwoordelijkheden waren. Deadlines zijn nagenoeg altijd gehaald geweest, zij het meestal nipt.

E Beschrijving van de werkverdeling

E.1 Samuel

Als teamleider: Planningen opstellen, taken verdelen en communicatie met de begeleiders. Als bijkomende opdracht heeft hij telkens de verslagen nagelezen.

- Demo 1: Bluetooth connectie van de robot naar de pc en omgekeerd, parameters voor correct rijgedrag van de robot bepalen en bouw van de robot.
- Demo 2: Niet werkende bluetooth connectie zo goed mogelijk werkend krijgen. Tevens kleine andere problemen uit demo 1 oplossen. Verder nog alles wat met communicatie te maken heeft.
- Demo 3: Barcodes uitlezen en acties die ermee verbonden zijn uitvoeren. Opnieuw bepalen van parameters. Zeer veel testen.

E.2 Robin

Als secretaris: Wekelijkse voortgang bijhouden en verslagen maken.

- Demo 1: Simulator, overkoepelende softwarearchitectuur, GUI.
- Demo 2: Lichtsensor van de robot en simulator, algoritme van rechtzetten op een witte lijn, overkoepelende architectuur, bluetooth.
- Demo 3: Bluetooth, herhalen van nauwkeurigheidstesten, testen van verkennen van doolhof, rechtzetten op witte lijn.

E.3 Ruben

- Demo 1: Simulator, overkoepelende softwarearchitectuur, meehelpen GUI.
- Demo 2: Simulator verder uitbreiden, veldstructuur uitwerken, meehelpen GUI.
- Demo 3: Ultrasone sensor testen en fouten er uithalen, algoritmes autonome robot maken, testen autonome robot (op simulator), samenvoegen verkennen doolhof en barcodes en fysiek testen.

E.4 Eline

- Demo 1: Bluetooth connectie van de robot naar de pc en omgekeerd, parameters voor correct rijgedrag van de robot bepalen en bouw van de robot.
- Demo 2: Bluetooth verbeteringen, optimale metingen van ultrasone sensor, samenwerking ultrasone sensor met lichtsensor voor loodrecht op de witte lijn en begin algoritme voor de robot in het midden van een tegel te plaatsen.
- Demo 3: Barcodes lezen en verwerken, verdere bepaling parameters, verwerking lichtsensor gegevens, afwerken methode midden van de tegel.

E.5 Dries

Dries heeft gedurende heel het semester aan de GUI gewerkt en is ook bijgesprongen bij verschillende andere taken, waar nodig (bijvoorbeeld testen).

Week	Samuel	Ruben	Robin	Eline	Dries
Week 1	15	11	9	13	6,5
Week 2	10	5	6,5	9,5	9
Week 3	20,75	10,5	14	17,25	10,5
Week 4	10,25	14	9	10	10
Week 5	17,5	12	9,5	21	7
Week 6	17	16,5	17	12,5	13
Week 7	11,5	21	7	14,5	14
Week 8	13	6	7,5	10,5	5
Week 9	14	13	5	13,5	5
Week 10	13,5	27,5	19,5	16	14,5
Week 11	8	11	11	11	10
Totaal	147,5	144,5	112	145,75	101,5

Tabel 1: Deze tabel geeft het aantal uren weer die elke persoon elke week gespendeerd heeft aan dit project.

F Kritische analyse

F.1 Sterke punten

- Het algoritme om het doolhof te verkennen is robuust.
- We hebben op een deftige manier rekening gehouden met de (on)nauwkeurigheid van de sensoren.
- De samenwerking binnen het team ging goed.
- Er is op voorhand goed nagedacht over de architectuur waardoor we latere problemen hebben voorkomen.

F.2 Zwakke punten

- Er is onvoldoende documentatie geschreven, waardoor groepsleden soms niet goed konden volgen bij de onderdelen van andere mensen.
- De bluetooth problemen hebben de vooruitgang van het project een beetje belemmerd.
- Het uitvoeren van de barcodes wordt direct op de robot gedaan. Nu zouden we die acties doorsturen naar de PC om alle acties die op de robot uitgevoerd worden op n centrale plaats te kunnen beheren. Dan kan het verkenalgoritme deze uitvoeren wanneer de robot klaar is met iets anders.
- De verdeling van het werk hadden we beter kunnen organiseren.

G Voorstel opgave 2^e semester: Team Treasure Trek

Dit voorstel is gebaseerd op een minigame uit de Nintendo Gamecube Game: Mario Party 4 genaamd Team Treasure Trek[9]. In deze minigame nemen twee teams van elk twee spelers het tegen elkaar op om in een doolhof, waarin zich twee schatkisten en twee sleutels bevinden, om ter eerst hun respectievelijke schatkist en sleutel vinden. Wanneer één teamlid bijvoorbeeld de schatkist heeft gevonden, moet hij zo snel mogelijk het andere teamlid vinden, die dan hopelijk de sleutel heeft. Het team dat het eerste de schatkist opent met de sleutel is de winnaar.

G.1 Doelstellingen

Twee teams van elk twee robots moeten om ter eerst in een doolhof één paar van de twee paar verschillende items in het doolhof vinden. Deze items worden bewaakt door 4 stationaire robots deze enkel het item vrijgeven wanneer deze het juiste wachtwoord doorgestuurd krijgt. Vervolgens moet een teamlid zijn andere teamlid vinden en de items samenbrengen.

G.2 Uitdagingen/probleemstellingen

- **Hoe ziet een doolhof eruit?**
  Mogelijke oplossing: Een doolhof bestaat uit onafhankelijke panelen van afmetingen 80cm x 80cm zodat er zeker plaats genoeg is voor twee robots om elkaar te kruisen.

- **Hoe ziet een doolhof eruit?**
  Mogelijke oplossing: In het doolhof is er voor elk team één balletje en één bakje, die beide bewaakt worden door een bewaker. Deze items moeten licht genoeg zijn om opgepakt te kunnen worden door een grijphaak van Lego. Om te winnen moet een team ervoor zorgen dat het balletje in het bakje terecht komt.

- **Hoe moet een robot het wachtwoord van een bewaker achterhalen?**
  Mogelijke oplossing: Eerst stuurt de robot via bluetooth een vraag naar de bewaker om te weten te komen of hij een item van zijn team bewaakt en of dit item een balletje of een bakje is. Vervolgens speelt de bewaker een van enkele voorgedefinieerde liedjes af. Dit moet de robot herkennen met behulp van de geluidssensor. Elk van deze liedjes heeft een bijbehorend wachtwoord, niet ongelijk aan hoe elke barcode een bijbehorende actie had in het eerste semester

- **Communicatie met teamgenoten en bewakers via bluetooth?**
  Mogelijke oplossing: Overkoepelende communicatie-API.

G.3 Benodigdheden en kostprijs, mogelijke problemen

- Verschillende houten panelen met rechtopstaande randen. Mogelijke problemen: wachttijd bij het bestellen? Kostprijs: moeilijk in te schatten. Minimum aantal panelen: moeilijk in te schatten, genoeg om een uitdagend doolhof te kunnen bouwen.

- Alle andere benodigdheden, grijphaak om balletje/bakje op te kunnen pakken, geluidsensor zitten normaal in de Lego Mindstorms Kit.

G.4 Concrete uitwerking

De sleutel is een bal en de schatkist is een met Lego ontworpen bakje. De vier voorwerpen worden bewaakt door vier stationaire robots. Deze bewakers krijgen ook een speciale plaats binnen het doolhof, zodat deze makkelijk uit de weg kunnen gaan wanneer ze het juiste wachtwoord te horen krijgen. De vier spelende robots racen door het doolhof om deze bewakers te vinden. Wanneer ze een bewaker tegenkomen communiceren ze eerst met deze bewaker om erachter te komen of deze een item bewaakt van zijn team of van het andere team en of dit item een sleutel is of een schatkist. Als het een item van het juiste team is, speelt de bewaker een van enkele voorgedefinieerde liedjes af. De speler moet dit liedje kunnen herkennen en vervolgens een bijbehorend wachtwoord aan de bewaker doorsturen. Dit herkennen van liedjes gebeurt uiteraard met behulp van de geluidssensor, die in het eerste semester nog niet werd gebruikt. Wanneer de robot het correcte wachtwoord gegeven heeft, gaat de bewaker opzij en krijgt de robot toegang tot zijn item. Daarna moet de robot de andere robot van zijn team vinden, die hetzelfde proces heeft moeten doorlopen. Wanneer de robots elkaar gevonden hebben en beide items bezitten, moet de bal in het bakje geplaatst worden en is dit team gewonnen.

Referenties

[1] http://lejos.sourceforge.net/nxt/nxj/api/index.html

[2] http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm

[3] http://www.nxtprograms.com/9797/express-bot/steps.html

[4] http://www.nxtprograms.com/bumper_car/index.html

[5] http://www.youtube.com/watch?v=swqNJW0dhg4

[6] http://cs.unibg.it/scandurra/material/INF3B_1112/windowbuilderTutorial.pdf

[7] http://www.coderanch.com/t/344345/GUI/java/Simple-Graph

[8] http://en.wikipedia.org/wiki/A*_search_algorithm

[9] http://www.mariowiki.com/Team_Treasure_Trek