KU Leuven
Departement Computerwetenschappen

P&O: Computerwetenschappen

Eindverslag

Team:
Geel

Robin Goossens
Samuël Lannoy
Ruben Pieters
Eline Vanrykel
Dries Verreydt

Academiejaar 2012 – 2013

Samenvatting

Dit verslag handelt over het ontwerpen van een robot die in staat is het spel 'Team Treasure Trek' te spelen met drie andere robots. Dit betekent dat hij een voorwerp moet zoeken in een onbekend doolhof en daarna naar zijn teamgenoot moet toe rijden.

Terwijl de robot het doolhof verkent, gaat hij met de lichtsensor op zoek naar witte lijnen en barcodes. Met behulp van de ultrasone sensor, die muren kan detecteren, wordt het doolhof in kaart gebracht. Het doolhof bevat wippen die, afhankelijk van hun stand, infraroodsignalen uitzenden. Wanneer de robot een barcode behorend bij een wip detecteert, gaat hij zijn infrarode sensor gebruiken om te controleren of de wip berijdbaar is. Indien een barcode aangeeft dat het te zoeken object op het volgende vakje ligt, gaat de robot dit object oppikken. Wanneer zijn teamgenoot bekend is, gaan ze beide hun doolhoven samenvoegen en naar elkaar toe bewegen.

Om zich zo nauwkeurig mogelijk door het doolhof te kunnen bewegen, moet de robot compact gebouwd en goed gekalibreerd zijn. Bij een eventuele afwijking moet hij zichzelf kunnen corrigeren. Bovendien moeten botsingen met andere robots in het doolhof vermeden worden. Hiervoor is er een virtuele sensor aanwezig die gebruik maakt van informatie over het hele doolhof en de posities van de medespelers. Om het spel te kunnen spelen, moet de robot dus in staat zijn te communiceren met de andere robots. Deze communicatie verloopt met behulp van een communicatieprotocol dat gebruik maakt van RabbitMQ.

Verder is er een simulator ontworpen die dezelfde functionaliteit bezit als de echte robot. Zowel de robot als de simulator worden bediend en gevisualiseerd met behulp van een grafische gebruikersinterface (GUI) op een PC.

Inhoudsopgave

1 Inleiding

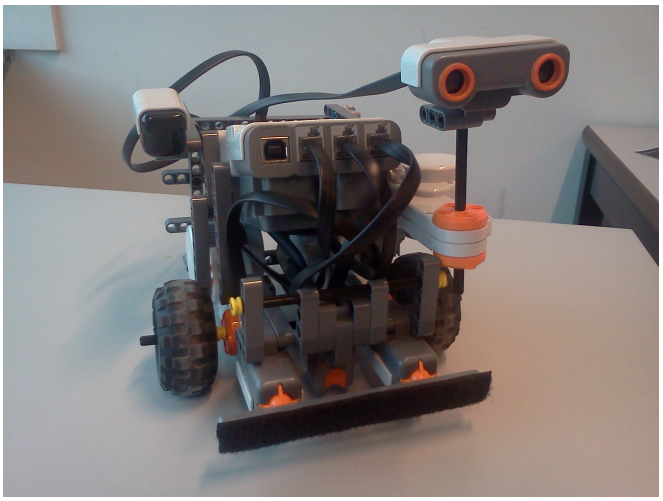
In dit project moet een autonome robot ontworpen worden die in staat is het spel 'Team Treasure Trek' te spelen met drie andere robots. De robot moet op zoek gaan naar een voorwerp in een onbekend doolhof. Na het voorwerp te hebben opgepikt, zal hij moeten communiceren met zijn teamgenoot om het spel te winnen. Hierbij moeten ze hun doolhoven met elkaar delen en samenvoegen. De winnaars zijn de robots die het eerst hun voorwerpen samenbrengen.

Eerst beschrijven we hoe we de robot hebben gebouwd en lichten we de bijhorende testen toe. Deze dienen om de robot te kalibreren en de verschillende sensoren correct te kunnen gebruiken. Vervolgens bespreken we de algoritmes die gebruikt worden om het spel te spelen en daarbij om te gaan met obstakels en andere robots. In de daarop volgende sectie bespreken we de verschillende software-onderdelen, waaronder de communicatie tussen de verschillende robots via RabbitMQ. Verder lichten we ook kort de werking van de simulator, de NXT en de grafische gebruikersinterface toe. Eindigen doen we met een bespreking van het software design.

2 Robot

2.1 Ontwerp

Het ontwerp van de robot is geïnspireerd op de Express-bot [1]. Bij de Express-bot is de brick helemaal omgeven door de wielen en bevindt er zich achteraan een vast, ondersteunend wieltje. In ons ontwerp is de brick verhoogd en schuin naar achter gericht. Deze aanpassingen zorgen ervoor dat de robot nog maar $17,4$ cm breed en $22,3$ cm lang is. Het uiteindelijke resultaat is te zien op figuur 1.



Figuur 1: Ontwerp van de robot. Vooraan is de bumper met velcrostrip te zien. Daarachter bevindt zich de lichtsensor. Aan de linkerkant hangt de motor voor de ultrasone sensor. De ultrasone sensor steekt met een staaf tot boven de brick uit, zodat hij correct afstanden kan meten. Aan de rechterkant hangt de infraroodsensor. Achteraan is een hoge constructie gebouwd.

Vooraan de robot is er een bumper bevestigd aan twee druksensoren. Aan de bumper is een velcrostrip bevestigd. Dit maakt het mogelijk om de wc-rol, bekleed met velcro, op te nemen. De lichtsensor is vooraan in het midden en zo ver mogelijk naar achter geplaatst. Hierdoor kon de bumper ook naar achter geplaatst worden en heeft de robot geen problemen om over de wip te rijden.

De ultrasone sensor is samen met zijn motor aan de linkerkant van de robot bevestigd. Het gebruik van een motor heeft als voordeel dat de robot metingen kan uitvoeren in verschillende richtingen

zonder dat hij hiervoor moet ronddraaien. Aan de andere kant van de robot bevindt zich de infraroodsensor.

Achteraan de robot is een zware constructie gebouwd zodat de robot niet naar voren omslaat bij het omslaan van de wip. Een bijkomend voordeel hiervan is dat de kabels gemakkelijk kunnen worden samengehouden. Daarnaast kan hieraan ook, in tegenstelling tot bij de Express-bot, een ronddraaiend wieltje met meer grip worden bevestigd.

2.2 Kalibratie

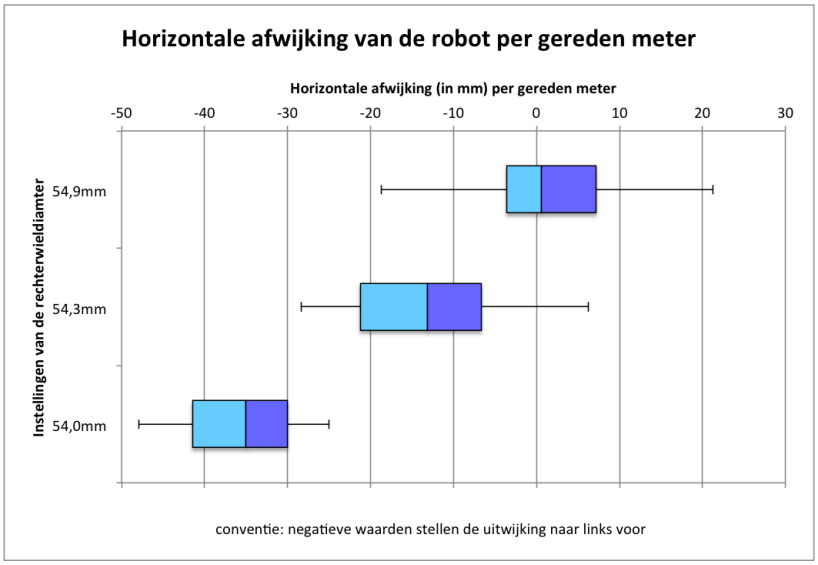
De robot wordt aangestuurd met behulp van de *DifferentialPilot* uit de LeJOS-API [3]. Hiervoor moeten een aantal parameters gekalibreerd worden die afhankelijk zijn van de bouw van de robot. Dit zijn met name de wieldiameters en de lengte van de wielas.

2.2.1 Afwijking op de gereden afstand

De diameter van de wielen is niet veranderd ten opzichte van vorig semester. Tijdens deze testen [6] zagen we dat de robot bij een instelling van 54,3 mm een gemiddelde afwijking van 0 mm per te rijden meter heeft.

2.2.2 Horizontale afwijking

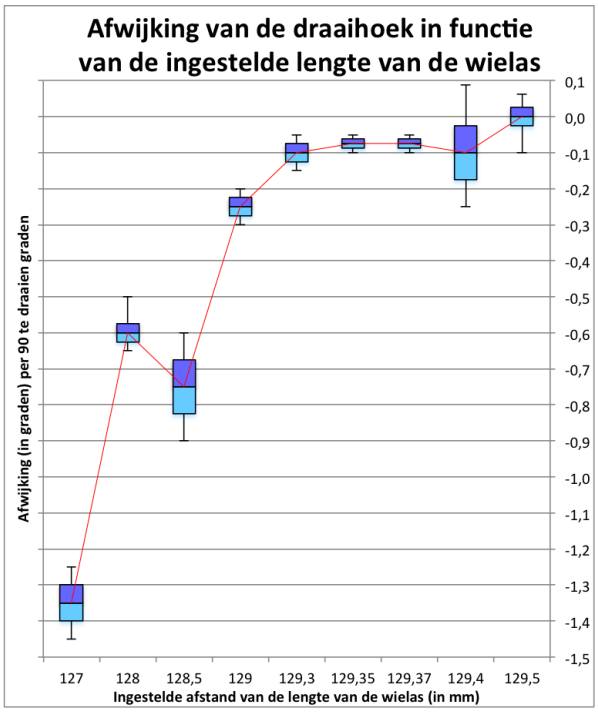
De twee wielen zijn in werkelijkheid even groot, maar de twee motoren draaien niet aan exact dezelfde snelheid. Er ontstaat dus een afwijking naar links of naar rechts bij het rijden op een rechte lijn. Deze afwijking kan gecompenseerd worden door de wieldiameter aan één kant te vergroten of verkleinen. We houden tijdens deze testen de linkerwieldiameter constant op 54,3 mm terwijl we de rechterwieldiameter laten variëren. De robot rijdt telkens 80, 160 en 240 cm. Daarna meten we de afwijking naar links of rechts tot op één millimeter nauwkeurig. Hieruit kunnen we de horizontale afwijking per gereden meter bepalen. De resultaten van de testen zijn te vinden in figuur 2. We besluiten hieruit dat bij een rechterwieldiameter-instelling van 54,9 mm de robot gemiddeld 0,6 mm naar rechts rijdt per afgelegde meter. Doordat de horizontale afwijking zeer veel variatie vertoont, kon deze niet kleiner worden gekregen. We nemen dus aan dat dit voldoende nauwkeurig is voor onze doeleinden.



Figuur 2: Resultaten van de horizontale afwijking van de robot met gemiddeld tien testen per instelling. Bij goede prestaties werden er meer testen gedaan dan bij slechte prestaties. Bij een instelling van 54,9 mm voor de rechterwieldiameter is een gemiddelde van 0,6 mm te zien.

2.2.3 Afwijking op de gedraaide hoek

De opgegeven lengte van de wielas zorgt voor het nauwkeurig draaien van de robot. Om deze parameter te bepalen laten we de robot telkens zestien keer 90 graden draaien. Daarna wordt de afwijking opgemeten tot op één graad nauwkeurig. Dit wordt herhaald voor een aantal instellingen voor de lengte van de wielas. De resultaten zijn te vinden in figuur 3. Per instelling worden er gemiddeld zes testen weergegeven. In de figuur zien we dat bij een instelling van 129,5 mm, er een fout van gemiddeld nul graden per 90 te draaien graden is. De robot draait maximaal 0,05 graden te veel of 0,1 graden te weinig op 90 graden. Dit is een te verwaarlozen afwijking.



Figuur 3: Resultaten van de draaitesten met gemiddeld zes testen per instelling. Bij goede prestaties werden er meer testen gedaan dan bij slechte prestaties. Bij 129,5 mm is een gemiddelde van nul graden te zien.

2.3 Sensoren

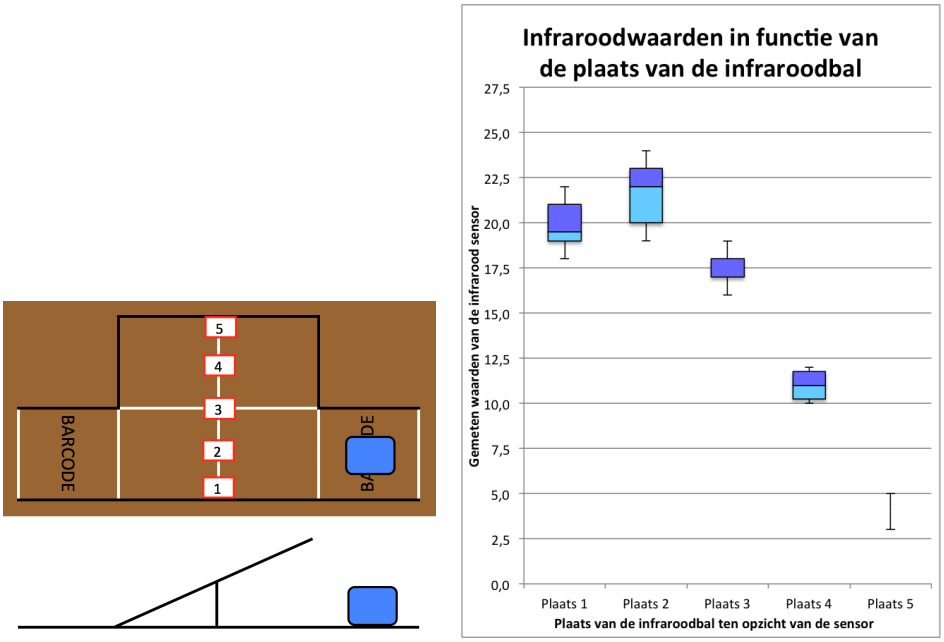
De prestaties van de robot zijn sterk afhankelijk van de informatie die hij verkrijgt van de sensoren. Om te weten hoe hij met deze gegevens moet omspringen, zijn de sensoren uitvoerig getest.

2.3.1 Infraroodsensor

De infraroodsensor bevindt zich aan de rechterzijde van de robot en is naar voren gericht. Deze sensor wordt gebruikt om informatie te verkrijgen over het al dan niet berijdbaar zijn van de wip. Onder de wip zal een bal geplaatst worden die infraroodsignalen uitzendt. De robot moet weten welke waarden hij kan verwachten bij een berijdbare en niet-berijdbare wip, zodat hij in beide gevallen de gepaste actie kan ondernemen.

Voor elke balpositie hebben we bij een berijdbare en een niet-berijdbare wip twintig waarden van de infraroodsensor uitgelezen. De balposities worden weergegeven in figuur 4(a). Bij een berijdbare wip wordt altijd de waarde 0 ontvangen. Deze gegevens zijn dan ook niet uitgezet in een grafiek. In figuur 4(b) zien we de resultaten voor een niet-berijdbare wip. Hieruit blijkt dat de minimale waarde die hiervoor gelezen wordt, gelijk is aan 3. Bovendien wordt deze waarde enkel gelezen

indien de bal op de slechtst mogelijke positie is geplaatst. We hebben besloten om deze waarde te gebruiken als grenswaarde voor het al dan niet berijdbaar zijn van de wip.



Figuur 4: **Links:** Locatie van de infraroodbal tijdens de metingen. Het blauwe symbool stelt de robot voor. Bovenaan staat het bovenaanzicht, onderaan het zijaanzicht. **Rechts:** Resultaten van de testen met de infraroodsensor. Op plaats 5 is de kleinst gemeten waarde 3, wat als grenswaarde gekozen werd.

2.3.2 Ultrasone sensor

Het is wenselijk dat de ultrasone sensor in het midden van de robot staat, zodat de metingen in alle richtingen op dezelfde manier geïnterpreteerd worden. In ons ontwerp staat deze echter links vooraan. Om het toch te laten lijken alsof hij in het midden staat, wordt er per richting een correctieterm opgeteld bij de metingen.

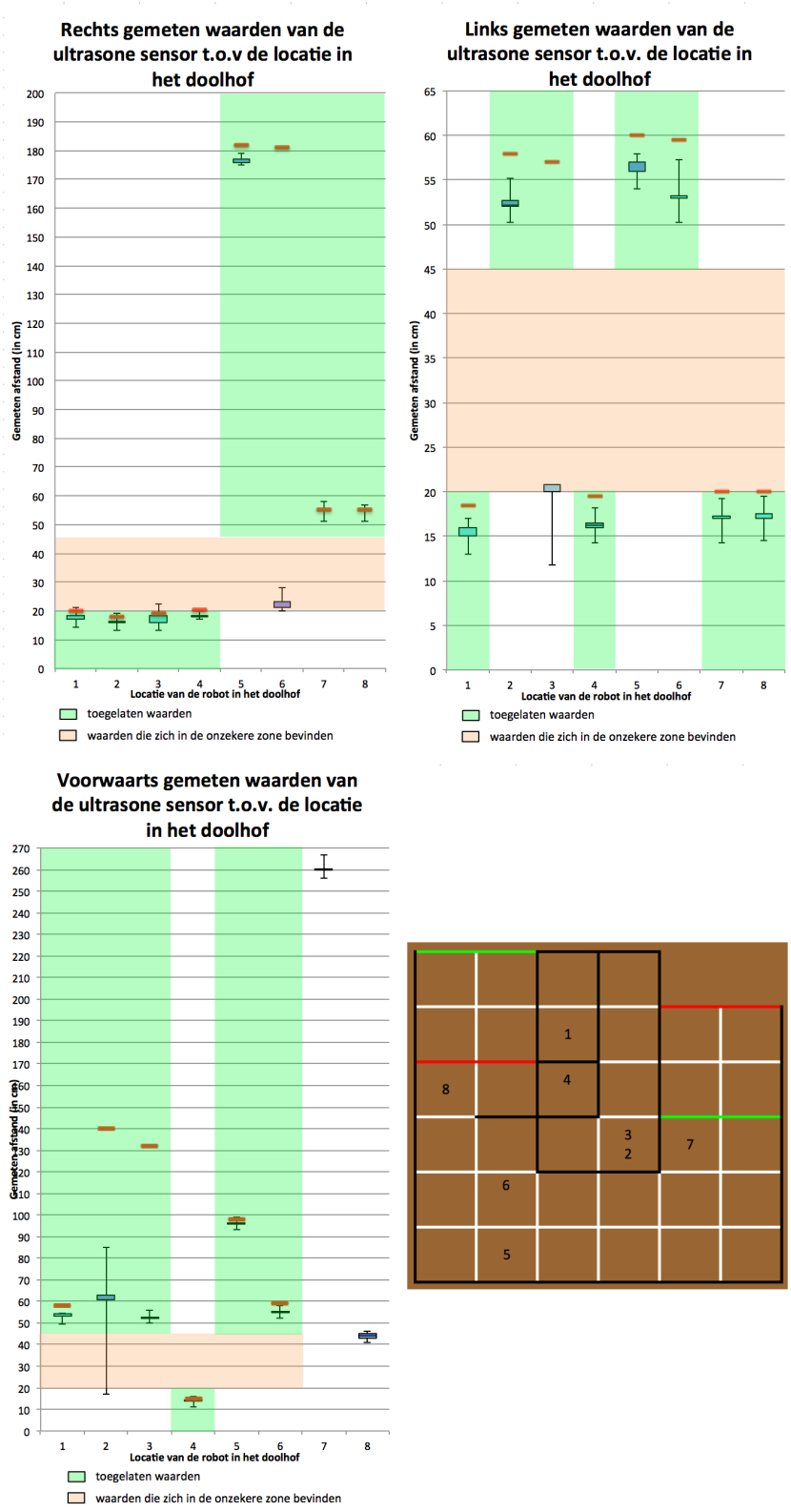
In figuur 5 zijn er verschillende metingen uitgezet op verschillende posities van de robot in het doolhof. De robot detecteert een muur bij een waarde kleiner dan 20 cm en een witte lijn bij een waarde groter dan 45 cm. Uit de figuur blijkt dat deze intervallen samen met de correctietermen bijna altijd een juiste detectie van muren opleveren. De enige uitzondering is een lezing naar links op locatie 3. Hier stond de robot echter niet in het midden van de tegel, maar te ver naar voor en stond er links een pilaar. Deze situatie doet zich normaal gezien niet voor tijdens het spel. Bovendien blijkt uit de metingen op locaties 7 en 8 dat de ultrasone sensor niet gebruikt kan worden voor het controleren of een wip berijdbaar is.

2.3.3 Lichtsensor

De werking van de lichtsensor is niet gewijzigd ten opzichte van vorig semester. We verwijzen naar het finale verslag [6] voor meer details over de nauwkeurigheid van deze sensor.

2.3.4 Druksensoren

Beide druksensoren zijn getest en werken naar behoren. Dit is echter triviaal en behoeft dus geen verdere uitleg. Er is wel maar één druksensor aangesloten, wegens een gebrek aan beschikbare poorten op de robot.



Figuur 5: Gelezen waarden van de ultrasone sensor met gemiddeld twintig metingen per positie en per kijkrichting. De rode lijnen duiden de werkelijke afstand aan. De locatienamen zijn verduidelijkt in het doolhof rechtsonder.

3 Algoritmes

Gedurende het spel zijn er twee fases. In de eerste fase gaat de robot op zoek naar zijn voorwerp terwijl hij het doolhof verkent. Dit wordt beschreven door het algoritme in sectie 3.1. Hierbij moet onder andere rekening worden gehouden met de wip. In de tweede fase weten de robots wie hun teamgenoot is en moeten ze elkaar vinden. Eerst worden de doolhoven van de verschillende spelers op elkaar afgestemd. Dit wordt beschreven door het algoritme in sectie 3.3. Hierna bewegen de robots van hetzelfde team naar elkaar toe, wat wordt uitgelegd in sectie 3.4. Hoe met andere robots in het doolhof wordt omgegaan, wordt uitgelegd in sectie 3.5.

3.1 Verkennen van het doolhof

De robot verkent het doolhof met het verkenalgoritme dat in het eerste semester geschreven werd [6]. Dit algoritme heeft enkele kleine wijzigingen ondergaan. Aangezien er nu rekening moet worden gehouden met wippen en andere robots, zal de robot op elke nieuwe tegel nagaan of het geplande pad berijdbaar is. Indien dit niet het geval is, zal er een nieuw pad berekend moeten worden waarbij rekening wordt gehouden met de obstakels. Dit zou er kunnen voor zorgen dat er andere te verkennen tegels dichterbij liggen dan voorheen. De lijst met te verkennen tegels wordt dus ook opnieuw gesorteerd.

3.1.1 Positie en oriëntatie corrigeren

Om te vermijden dat de robot na een bepaalde tijd heel scheef gaat rijden, corrigeerde hij zich door telkens te oriënteren op een witte lijn. Ondanks de compactheid van de robot, gaf dit bij een smalle doorgang soms problemen. Aangezien de robot traag afremt na het detecteren van een witte lijn, stond hij te ver voorbij de witte lijn. Bijgevolg draaide hij een te grote hoek, waardoor zijn achterkant soms tegen de muur botste. Dit werd opgelost door de robot bij het oriënteren een kleine afstand naar achter te laten rijden. Hierdoor wordt de totale gedraaide hoek aanzienlijk verkleind. Telkens oriënteren op een witte lijn kost echter nog steeds veel tijd. Daarom wordt dit nu nog maar om de drie tegels gedaan. Dit is nog net genoeg om ervoor te zorgen dat de robot niet tegen muren botst als hij afwijkt. Meer uitleg over hoe het oriënteren op een witte lijn gebeurt, is terug te vinden in het finale verslag van het eerste semester [6].

Daarnaast gaat de robot zich nu telkens in het midden van een tegel proberen te plaatsen vooraleer hij zich gaat oriënteren op de witte lijn. Dit gebeurt door tegen een zijmuur te botsen indien er één aanwezig is en daarna een vaste afstand naar achter te rijden. Hierbij wordt er minder ver achteruit gereden indien de robot zijn voorwerp vasthoudt. Deze methode zorgt ervoor dat de robot minder snel in de problemen zal geraken, maar vereist wel dat alle muren in het doolhof vast staan. Het botsen tegen muren werd boven het gebruik van de ultrasone sensor verkozen, omdat de waarden van de ultrasone sensor niet volledig betrouwbaar zijn.

3.1.2 Barcodes lezen

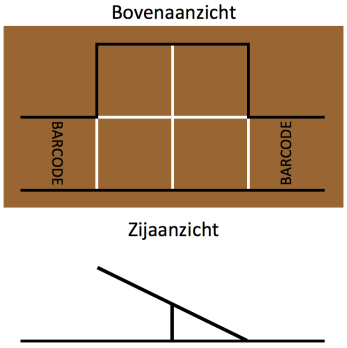
Een uitgebreide bespreking van het algoritme om barcodes te lezen is terug te vinden in het finale verslag van het eerste semester [6]. De barcodeacties worden nu wel op een andere manier uitgevoerd. Vroeger gebeurde dit rechtstreeks op de NXT. Nu worden de barcodes altijd doorgestuurd naar de PC en zal deze beslissen wat voor actie er ondernomen moet worden. Deze verandering was noodzakelijk, omdat het bijvoorbeeld niet altijd voordelig is om meteen over de wip te rijden bij het lezen van zo'n barcode. Er kunnen namelijk nog tegels dichterbij liggen die beter eerst verkend worden.

3.1.3 Wip

Een wip is een paneel van 2×2 tegels. Er zullen muren worden geplaatst aan de linker- of rechterkant van de wip, zodat er telkens maar één robot tegelijk over de wip kan rijden. Bovendien zal de wip telkens vooraf worden gegaan door op voorhand afgesproken barcodes. Zo weet de robot dat de volgende twee tegels deel uitmaken van een wip en zal hij het verkenalgoritme hier niet op

uitvoeren. De configuratie van een wip in het doolhof is te zien in figuur 6.

Om de stand van de wip te kunnen detecteren, wordt er gebruik gemaakt van een infraroodbal en een infraroodsensor. Als de robot een barcode scant die een wip bepaalt, zal hij de infraroodbal proberen te detecteren. Indien de bal gedetecteerd wordt, is het niet mogelijk om over de wip te rijden. We behandelen de wip op dat moment als een muur. Als de robot bij het scannen de infraroodbal niet detecteert, dan kan hij wel over de wip rijden. Bij het rijden over de wip, moet hij aan het einde even stoppen om zeker te zijn dat de wip overgeheld is. Vervolgens rijdt hij verder tot aan de volgende barcode.

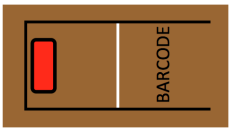


Figuur 6: Configuratie van de wip. Aan de linker- of rechterkant worden er muren geplaatst. De wip wordt telkens voorafgegaan door een barcode.

3.2 Opnemen voorwerp

Alle voorwerpen bevinden zich in een doodlopend stuk en worden voorafgegaan door een tegel met een barcode, zoals getoond in figuur 7. Aan elke barcode is één van de vier spelers gekoppeld. De robot komt op die manier te weten of het voorwerp achter de barcode voor hem is of niet. Als het een voorwerp van een andere speler blijkt te zijn, dan gaat de robot verder met het verkennen van het doolhof zonder de volgende tegel te beschouwen. Hij weet dat dit een doodlopend stuk is met een voorwerp.

Indien de robot de barcode van zijn voorwerp gevonden heeft, rijdt hij een bepaalde tijd vooruit zodat het voorwerp aan de velcrostrip van zijn bumper hangt. Dit zal hij daarna laten weten aan de andere robots.



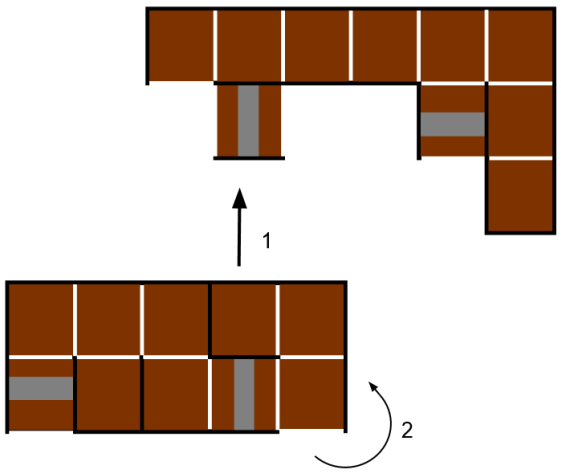
Figuur 7: Ligging van voorwerpen in het doolhof. Het voorwerp ligt altijd in een doodlopend stuk en wordt voorafgegaan door een barcode in de vorige tegel.

3.3 Samenvoegen van doolhoven

De robots van hetzelfde team verkennen het doolhof tot ze allebei twee gemeenschappelijke barcodes hebben gevonden. Om een gemeenschappelijk doolhof te bekomen, vertrekken we van deze gemeenschappelijke barcodes. We verschuiven eerst het doolhof van de teamgenoot, zodat één barcode correct overlapt. Daarna roteren we het doolhof tot de twee barcodes correct overlappen. Dit mechanisme wordt getoond in figuur 8. Ten slotte worden alle tegels van het andere doolhof, die niet in het eigen doolhof voorkomen, aan het doolhof toegevoegd. Alle boorden die niet

overeenkomen, worden vervangen door onzekere boorden. In sectie 4.5.2 wordt meer informatie gegeven over wat onzekere boorden inhouden.

Nadat de doolhoven samengevoegd zijn, houden we de translatie en rotatie bij om de startpositie van de teamgenoot in het eigen doolhof te kunnen berekenen. Dit is nodig om de doorgestuurde relatieve posities van de teamgenoot te kunnen vertalen naar coördinaten in het eigen doolhof.



Figuur 8: Het samenvoegen van doolhoven. (1) Het doolhof wordt eerst verschoven zodat één barcode correct overlapt. (2) Het doolhof wordt gedraaid zodat de andere barcode nu ook overlapt.

3.4 Teamgenoot bereiken

Eens de teams bekend zijn en de doolhoven samengevoegd zijn, moeten de teamgenoten naar elkaar toe bewegen. Dit doen ze door beide het kortste pad naar de andere te berekenen met behulp van hun eigen samengevoegd doolhof. In ons geval gebeurt dit met het A*-algoritme [9]. De heuristiek die hiervoor gebruikt wordt, is de Manhattan-afstand [4]. Als de teamgenoten zich beide bevinden op aangrenzende tegels zonder muur ertussen, zijn ze klaar.

Tijdens het berekenen van het kortste pad wordt er geen rekening gehouden met onzekere boorden. Als een robot op zijn pad onzekere boorden tegenkomt, zal hij deze onderzoeken. Indien deze boord een muur blijkt te zijn, zal het pad herberekend worden.

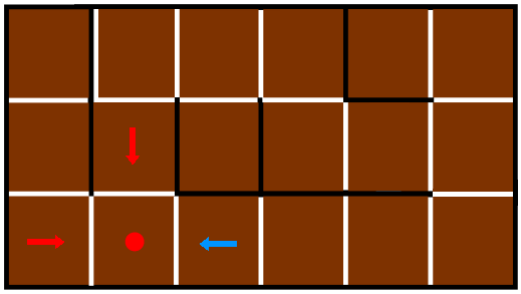
3.5 Andere robots

3.5.1 Robotdetectie

De robot moet andere robots in zijn nabije omgeving kunnen detecteren. Hiervoor beschikt hij over een virtuele robotdetectiesensor. Deze sensor heeft toegang tot informatie over het hele doolhof en alle spelers. Tegels die niet rechtstreeks bereikbaar zijn doordat er een muur tussen staat, worden bij robotdetectie telkens buiten beschouwing gelaten.

Om botsingen te voorkomen zal de robot voor het bewegen naar een nieuwe tegel controleren of daar geen robot staat. Daarnaast wordt ook gecontroleerd of er op de tegels die grenzen aan de nieuwe tegel, geen robot staat met de oriëntatie naar de bewuste tegel gericht. Indien dit wel zo is, betekent dit dat er in de nabije toekomst een robot op deze tegel kan komen te staan. De kans is dus groot dat dit tot een botsing leidt. Dit wordt getoond in figuur 9.

Daarnaast staat de robotdetectiesensor de ultrasone sensor bij, bij het detecteren van muren. Indien er een robot staat in de richting waarin de ultrasone sensor aan het scannen is, is het namelijk



Figuur 9: Het detecteren van andere robots. De eigen robot is in het blauw gemarkeerd. Mogelijke andere robots waarmee kan gebotst worden, staan in het rood aangegeven. De pijlen geven de richting van de robots aan en de bollen duiden op eender welke richting.

mogelijk dat er onterecht een muur wordt gedetecteerd. Stel dat de ultrasone sensor een muur detecteert. Als de robotdetectiesensor een robot kan detecteren in die richting, betekent dit dat er eigenlijk geen muur is. In het andere geval staat er wel degelijk een muur in die richting.

3.5.2 Voorrangsregels

Het is mogelijk dat er voor twee robots geen ander pad is naar een bepaalde tegel en ze elkaar dus blokkeren. In dit geval ontstaat er een deadlock-situatie. Om dit op te lossen gaat de robot een willekeurige tijd wachten en hopen dat de andere robot tegen dan al een actie heeft ondernomen en niet meer in de weg staat. Indien dit niet het geval is, zal hij willekeurig door het doolhof beginnen bewegen tot na een punt waarop hij meerdere richtingen kan uitgaan. Hij zal één van de richtingen willekeurig kiezen en daar een willekeurige tijd blijven wachten. In tussentijd kan de andere robot misschien zijn doel al bereiken. Dit wordt telkens opnieuw gedaan tot beide robots niet meer in elkaars weg zitten.

Nadat één robot het doolhof volledig heeft verkend, maar zijn teamgenoot nog niet bekend is of de doolhoven nog niet zijn samengevoegd, zal hij wachten. Tijdens dit wachten controleert hij of hij de doorgang voor andere robots niet blokkeert. Indien dit wel het geval is, zal hij opnieuw willekeurig door het doolhof bewegen tot na een keuzepunt. Daar zal hij dan opnieuw blijven wachten tot zijn teamgenoot bekend is en de doolhoven samengevoegd zijn.

4 Software

4.1 Communicatie tussen robots

Om de robots te kunnen laten communiceren wordt er gebruik gemaakt van RabbitMQ. RabbitMQ is een stuk software dat in staat is om berichten uit te wisselen tussen verschillende clients, met behulp van een server. Het programma implementeert het Advanced Message Queuing Protocol. Meer informatie hierover is te vinden in de documentatie van RabbitMQ [8].

De praktische implementatie van het communicatieprotocol gebeurt met een bibliotheek die door alle teams samen geschreven werd: Het Team Treasure Trek Protocol (HTTTP) [2]. Hoe deze geïntegreerd werd in ons project, wordt besproken in secties 4.5.3 en 4.5.4.

4.1.1 Connectie maken

Om een verbinding te kunnen maken met de andere robots moet elke speler een PlayerClient aanmaken. Hierbij wordt samen met andere robots deelgenomen aan een spel. Toeschouwers kunnen ook verbinding maken met alle robots door een SpectatorClient aan te maken.

4.1.2 Verzenden van berichten

De PlayerClient voorziet methodes om verschillende soorten berichten te versturen naar de andere robots. Hij voorziet methodes om posities te updaten, andere robots te laten weten dat hij zijn voorwerp gevonden heeft, zich bij een team te voegen, toeschouwers te laten weten dat hij over de wip is gereden en om doolhoven te versturen. De posities worden verstuurd als coördinaten van tegels en dit telkens wanneer de robot in het midden van een nieuwe tegel komt te staan. Doolhoven worden tegel per tegel verstuurd als tekst volgens een overeengekomen specificatie [5].

4.1.3 Ontvangen van berichten

Het ontvangen van berichten gebeurt met de implementatie van de PlayerHandler en de SpectatorHandler. De PlayerHandler wordt gebruikt door de effectieve deelnemers van het spel. De SpectatorHandler wordt gebruikt door toeschouwers die het spel enkel overzien, maar er zelf niet aan deelnemen. Beide handlers voorzien methodes om te weten te komen of het spel bezig is, of een speler aan het spel deelneemt, of een speler klaar is om te starten, of een speler gewonnen is, of de andere robots hun voorwerp gevonden hebben en om doolhoven te ontvangen. De SpectatorHandler bevat bovendien methodes om te weten te komen welke robot over welke wip is gereden en wat de positie is van elke robot. De PlayerHandler kan enkel de positie van zijn teamgenoot te weten komen.

4.2 Simulator

De simulator moet zich nagenoeg identiek gedragen als de robot. Dit is nodig om de algoritmen te testen die gemeenschappelijk zijn aan de fysieke robot en de simulator. Om het continue gedrag van de robot te simuleren, wordt de staat van de gesimuleerde robot elke milliseconde vernieuwd. De applicatie-interface van de simulator heeft ook een volledig identieke structuur als bij de echte robot. Hierdoor kunnen we heel gemakkelijk zowel de robot als de simulator aansturen op eenzelfde manier.

De sensoren lezen waarden in aan de hand van de positie van de gesimuleerde robot. Als de robot zich bevindt op een witte boord, bijvoorbeeld, dan geeft de lichtsensor een mogelijke waarde voor wit terug. Als hij muren scant, dan geeft de ultrasone sensor realistische waarden voor de afstand tot de muren terug. Aan deze sensoren is ruis toegevoegd die wordt berekend aan de hand van de gemeten afwijking van de sensoren. Meer informatie hierover is terug te vinden in het finale verslag van het eerste semester [6].

De robots in de simulator hebben afmetingen, zodat er rekening kan worden gehouden met de reële grootte van de robot. Dit heeft als gevolg dat de robot gemakkelijker zal botsen met muren of andere robots, zoals met de fysieke robot ook het geval is. Verder kunnen andere robots en voorwerpen gesimuleerd worden. Wanneer de gesimuleerde robot tegen de muur botst van het vakje waarin het voorwerp ligt, wordt het voorwerp automatisch opgepikt.

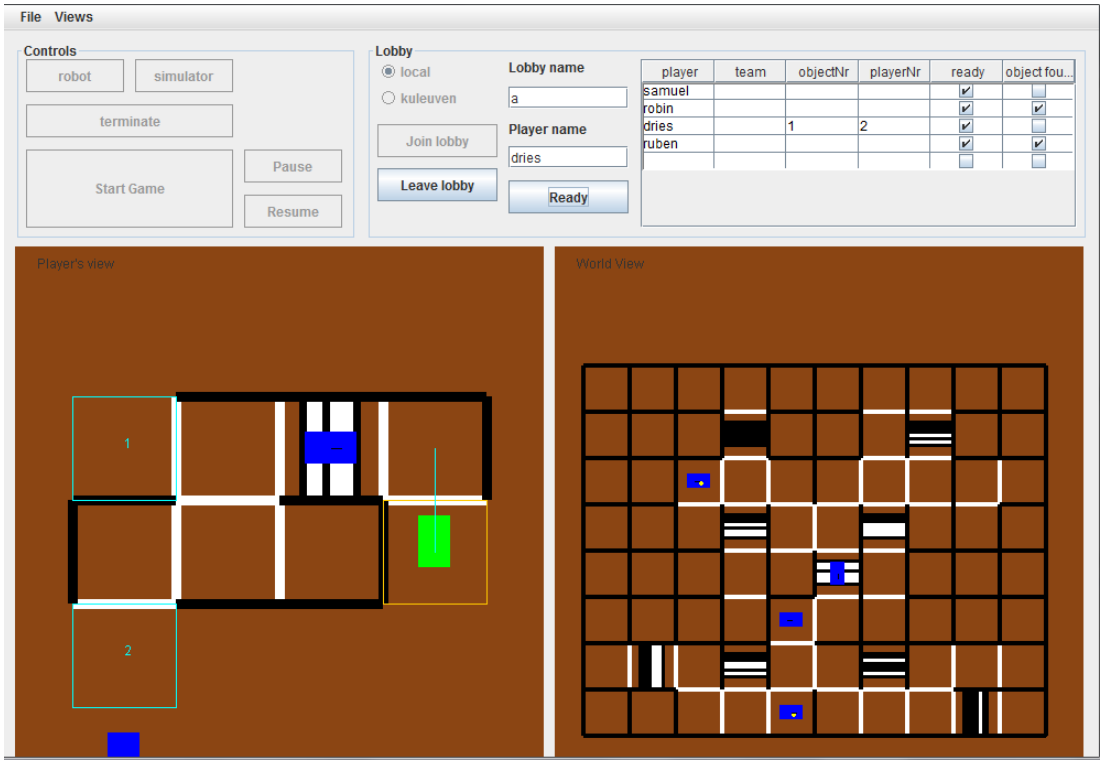
4.3 Software op de NXT

De software uit het eerste semester is bijna volledig behouden gebleven [6]. Het is nu echter niet meer nodig om de lichtsensor bij het opstarten van de robot telkens te kalibreren. Er worden standaardwaarden opgeslagen op de NXT, maar er kan ook nog steeds voor gekozen worden om manueel te kalibreren. De barcodes worden nu ook meteen doorgestuurd naar de PC in plaats van ze uit te voeren op de robot. Meer informatie hierover is te vinden in sectie 3.1.2.

4.4 Grafische gebruikersinterface (GUI)

De GUI werd volledig herontworpen om het spelgevoel te versterken. Op het hoofdvenster staan nu enkel nog de knoppen om het spel te starten, een tabel die lobby-informatie bevat en twee kaarten. De linkerkaart toont informatie vanuit het standpunt van de eigen speler. De rechtse kaart is een overzichtskaart van het doolhof en de posities van de verschillende robots. Op het moment

dat de speler een teamgenoot heeft, maar de kaarten nog niet kunnen worden samengevoegd, zal deze overzichtskaart worden vervangen door de kaart van de teamgenoot. Wanneer de kaarten zijn samengevoegd, wordt de kaart terug de overzichtskaart. Hoe het hoofdvenster er op dit moment uitziet, is te zien op figuur 10.



Figuur 10: Het hoofdscherm van de grafische gebruikersinterface. Linksboven staan de knoppen om het spel te starten. Rechtsboven staat de lobby-tabel. Linksonder staat de kaart vanuit het standpunt van de eigen robot en rechtsonder staat de overzichtskaart van het doolhof met de posities van alle robots. De eigen robot is in het groen aangeduid; de andere robots zijn in het blauw aangeduid. De blauwe cijfers duiden op de volgorde waarin de volgende tegels verkend zullen worden.

Op de kaart wordt een wip voorgesteld als twee vakjes die naast elkaar liggen. Op de tegels voor en na de wip zullen de barcodes liggen die aangeven dat de wip aanwezig is. De boorden aan de zijkant zullen altijd muren zijn, getekend door zwarte lijnen. De boorden aan de voor- en achterkant van de wip zijn afhankelijk van de helling van de wip. Robots worden, in tegenstelling tot vorig semester, als rechthoeken getoond om aan te tonen dat ze afmetingen hebben.

De andere onderdelen van de GUI bevinden in andere vensters, die geopend kunnen worden vanuit het menu in het hoofdvenster. Het geavanceerde venster bevat knoppen om de functionaliteit uit de vorige demo's aan te roepen. Het sensorvenster bevat alle informatie uit de sensoren van de robot. De debuginformatie is terug te vinden in het debugvenster en de precieze positie van de robot kan afgelezen worden in het positievenster.

De lobby-functionaliteit bestaat uit knoppen om te verbinden met een opgegeven lobby en de verbinding te verbreken. De gebruiker moet kiezen uit een lokale lobby of een lobby via KU Leuven. Daarnaast geeft hij de naam van de lobby en de naam van zijn robot in om te kunnen verbinden met de gekozen lobby. Eenmaal verbonden kan de gebruiker klikken op *ready* als hij klaar is om het spel te starten. Wanneer alle gebruikers klaar zijn, begint het spel automatisch.

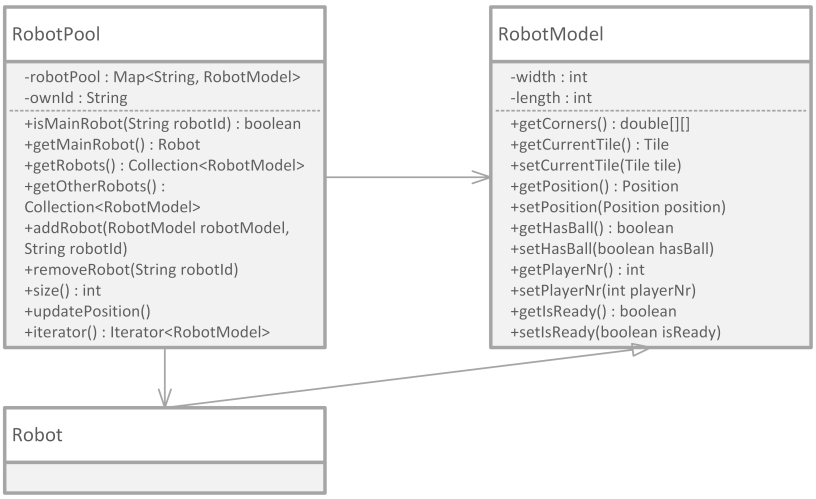
In de lobby-tabel is alle informatie te vinden over de spelers die deelnemen aan het spel. Deze tabel

geeft de naam van elke speler en de naam van zijn teamgenoot weer, samen met zijn objectnummer en spelersnummer. Daarnaast is uit deze tabel ook af te leiden wie klaar is om het spel te starten en wie zijn object al gevonden heeft.

4.5 Software design

4.5.1 Kennis van meerdere robots

De RobotPool-klasse houdt de verschillende robots in het spel bij. De hoofdrobot in de RobotPool is degene die bestuurd wordt in de lokale applicatie en wordt bijgehouden als Robot. Daarnaast zijn er nog drie andere robots die bestuurd worden door andere applicaties en waarvan informatie ontvangen wordt via RabbitMQ. Deze robots worden bijgehouden als RobotModels. Een RobotModel houdt enkel zijn positie, status en afmetingen bij. Met behulp van deze informatie kunnen de verschillende robots op de GUI getekend worden. In de Robot-klasse wordt de eigen teamgenoot ook nog apart bijgehouden als Robot om de gecommuniceerde informatie, zoals het veld van de teamgenoot, op te slaan.

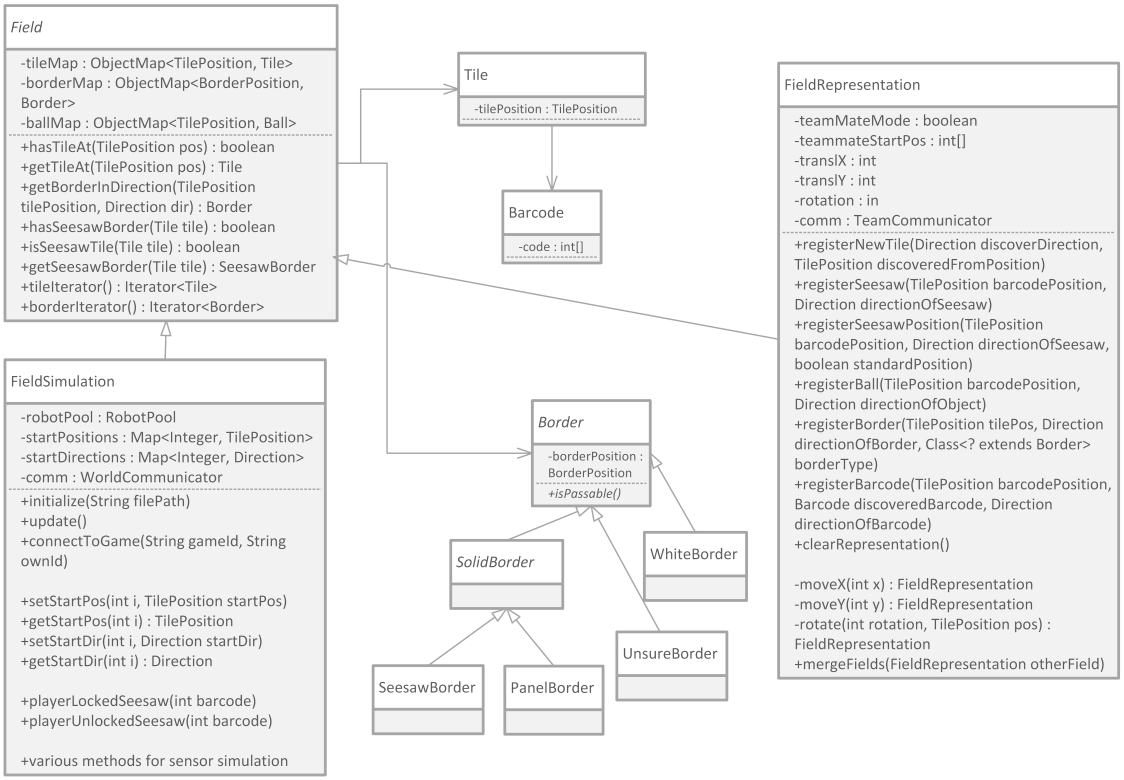


Figuur 11: Klassendiagram voor de RobotPool, die alle robots in het spel bijhoudt. De eigen robot wordt bijgehouden als Robot. De andere robots worden bijgehouden als RobotModel.

4.5.2 Doolhof

Het doolhof wordt voorgesteld door tegels (Tile) en boorden (Border). Deze hebben verschillende subklassen naargelang de verschillende fysische tegels of boorden die voorkomen. In het bijzonder zijn dat witte lijnen (WhiteBorder) en muren (PanelBorder). Daarnaast zijn er nog twee speciale boorden. Een onzekere boord (UnsureBorder) is een boord waarvan niet zeker is of het een muur of een witte lijn is. De SeesawBorder stelt één van de boorden van de wip voor. Deze boord gaat altijd gepaard met een tweede SeesawBorder die de andere kant van de wip voorstelt. De SeesawBorder voorziet methodes waarmee hij 'omlaag' of 'omhoog' kan worden gezet.

De overkoepelende veldklasse (Field) staat in voor het beheren van alle tegels en boorden. Bovendien houdt hij de twee gekoppelde SeesawBorders van eenzelfde wip consistent met elkaar. De veldklasse wordt opgesplitst in een veldrepresentatie (FieldRepresentation) enerzijds en een veldsimulatie (FieldSimulation) anderzijds. De FieldRepresentation stelt de representatie van het veld door de robot voor. Deze voorziet methodes om de representatie uit te breiden en samen te voegen met het verkende veld van een teamgenoot. De FieldSimulation stelt de simulatie van het gehele veld voor. Hij houdt een lijst van alle robots bij met behulp van de RobotPool. De FieldSimulation wordt geïnitialiseerd door een doolhof uit een bestand in te lezen. Hij kan ook informatie ontvangen vanuit de andere wereldsimulatoren om bijvoorbeeld een omgeslagen wip te simuleren. Daarnaast voorziet hij ook methodes om de verschillende sensoren te simuleren.



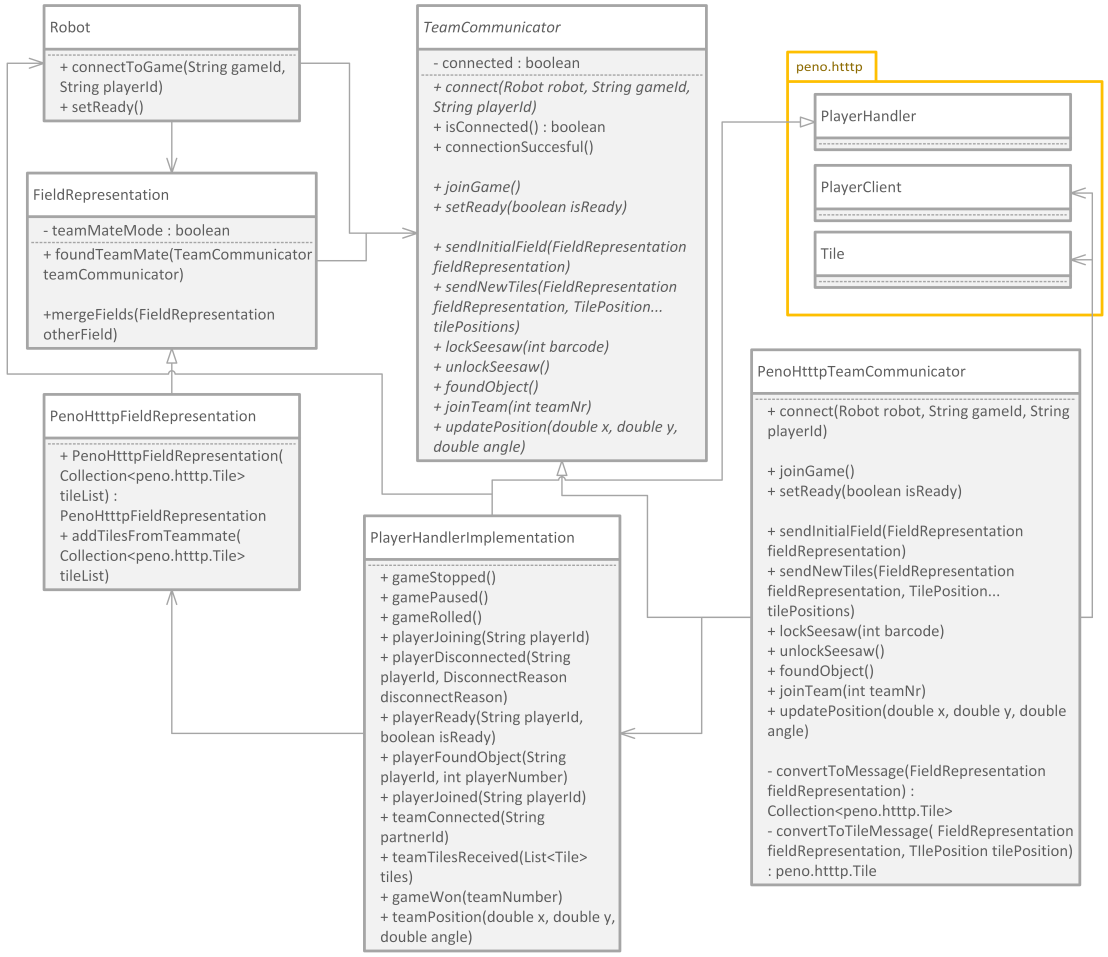
Figuur 12: Klassendiagram van de doolhofarchitectuur. Centraal staat de Field-klasse, die het vertrekpunt is voor boorden (Border) en tegels (Tile) met barcodes (Barcode). Deze klasse wordt overgeërfd door FieldRepresentation, wat de representatie van het veld door de robot is, en FieldSimulation, wat de simulatie van het veld is. De WhiteBorder stelt witte lijnen voor, de PanelBorder stelt muren voor en de UnsureBorder stelt een onzekere boord voor. De SeesawBorder stelt een boord aan een rand van een wip voor.

4.5.3 Teamcommunicatie

Zowel de Robot-klasse als zijn veldrepresentatie (FieldRepresentation) maken gebruik van de TeamCommunicator. Deze klasse staat in voor het verzenden van de verschillende signalen die gedefiniëerd zijn. Hieronder vallen bijvoorbeeld de methode *foundObject* om te signaleren dat het object gevonden is en de methodes *sendInitialField* en *sendNewTiles* om tegels naar de teamgenoot te verzenden. De TeamCommunicator is een abstracte klasse om het specifieke gebruik van de HTTTP-bibliotheek af te schermen, zodat ook andere bibliotheken zouden kunnen worden gebruikt.

De implementatie waarbij de HTTTP-bibliotheek wordt gebruikt, zit in de PenoHtttpTeamCommunicator. Deze implementeert de TeamCommunicator. De PenoHtttpTeamCommunicator maakt rechtstreeks gebruik van de methodes van de PlayerClient om de verschillende signalen te verzenden.

In de *connect*-methode van de PenoHtttpTeamCommunicator wordt er ook een PlayerHandlerImplementation gemaakt. Deze klasse zorgt ervoor dat de communicatie van het spel de Robot en zijn FieldRepresentation kan bereiken. Deze klasse maakt gebruik van de PenoHtttpFieldRepresentation om zo vanuit de verzonden berichten uit de HTTTP-bibliotheek een FieldRepresentation te genereren.



Figuur 13: Klassendiagram van de communicatie-architectuur voor de robot. De PenoHtttp-TeamCommunicator staat in voor het verzenden van signalen met behulp van de PlayerClient. De PlayerHandlerImplementation staat in voor het ontvangen van signalen en updaten van de PenoHtttpFieldRepresentation en de Robot.

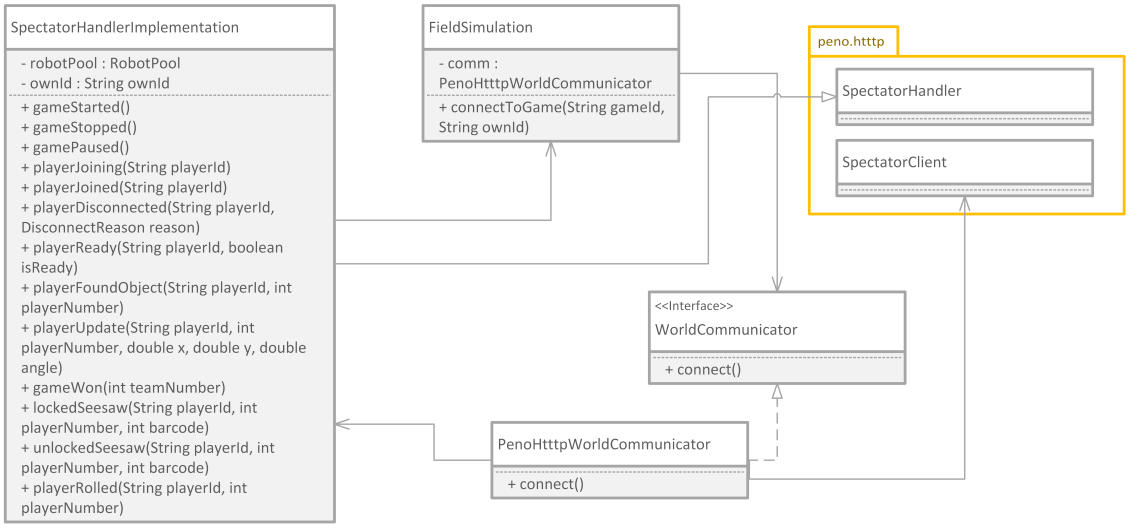
4.5.4 Wereldcommunicatie

Naast de TeamCommunicator is er ook een WorldCommunicator beschikbaar, die op een gelijkaardige manier werkt. Deze WorldCommunicator wordt gebruikt door de veldsimulatie (FieldSimulation) om verbinden met het spel als toeschouwer.

De PenoHtttpWorldCommunicator is een implementatie van de WorldCommunicator. Hij maakt gebruik van de methodes van de SpectatorClient om de verschillende signalen te verzenden. In de *connect*-methode wordt een SpectatorHandlerImplementation aangemaakt. Deze klasse zorgt ervoor dat de communicatie van het spel de FieldSimulation kan bereiken.

5 Besluit

Het naar achter plaatsen van de lichtsensor heeft ervoor gezorgd dat de robot niet vast komt te zitten bij het beklimmen van de wip. Oriënteren op een witte lijn gebeurt nu nog slechts om de drie tegels. Daarvoor zal de robot tegen de muur botsen om zich ook in de andere richting recht te zetten. Dit heeft het verkennen van het doolhof een stuk betrouwbaarder gemaakt. De kans dat de robot niet meer in het midden van een tegel staat, is immers aanzienlijk verkleind. Dit heeft als gevolg dat de kans dat de robot zijn voorwerp effectief opneemt ook veel groter is geworden.



Figuur 14: Klassendiagram van de communicatie-architectuur voor de wereld. De PenoHtttpWorldCommunicator staat in voor het verzenden van signalen met behulp van de SpectatorClient. De SpectatorHandlerImplementation staat in voor het ontvangen van signalen en updaten van de FieldSimulation.

Door gebruik te maken van de virtuele robotdetectiesensor, kunnen botsingen vrij eenvoudig vermeden worden. Door altijd eerst te wachten, kan onze robot zijn weg probleemloos vervolgen als de andere robot een actie ondernomen heeft. De willekeurige wandeling, daarentegen, kan wel veel tijd kosten. Door de wachttijd voldoende lang te nemen, zal deze verstreken tijd in veel gevallen een stuk lager liggen dan de tijd die verloren zou gaan bij het ondernemen van een willekeurige wandeling.

A Demo 1

A.1 Resultaten

De robot kreeg een `OutOfMemoryException` tijdens de demo. In een nieuwe poging probeerde hij het voorwerp op te pakken, maar dat mislukte doordat hij naar een hoek reed. De simulator deed wel wat ervan verwacht werd.

A.2 Conclusies

De robot moet nauwkeuriger afgesteld en compacter gebouwd worden.

A.3 Oplijsting aanpassingen verslag

- Referenties aangepast zodat het 'echte' referenties zijn.
- Testen toegevoegd: nauwkeurig rijden.
- Informatie over druksensoren toegevoegd.
- Referentienotering in de tekst aangepast naar hoe het moet.

B Demo 2

B.1 Resultaten

De robot slaagde er niet in om over de wip te rijden, alhoewel dat tijdens onze testen wel perfect lukte. Hierdoor konden we het spel niet verder spelen en de andere werkende onderdelen niet demonstreren.

B.2 Conclusies

De demo moet opgesplitst worden in verschillende delen die met zekerheid werken. Er moeten ook meer testen gebeuren op het geheel.

B.3 Oplijsting aanpassingen verslag

- Afmetingen van de robot in het verslag vermeld.
- Integratie van RabbitMQ in het softwaresysteem beter beschreven.
- Secties over horizontale afwijking en afwijking op gedraaide hoek van plaats verwisseld, omdat het eerste invloed heeft op het tweede.
- Op verschillende plaatsen voor meer verduidelijking gezorgd.

C Demo 3

C.1 Resultaten

De vier robots, zowel fysiek als simulator, konden succesvol het doolhof verkennen en andere robots ontwijken. Het spel kon volledig worden uitgespeeld met de simulators. Tijdens de gemeenschappelijke run kon het spel met de fysieke robots niet worden uitgespeeld, doordat er voortdurend robots van andere teams crashten.

C.2 Conclusies

De problemen die zich voordeden op vorige demo's zijn opgelost. De robot en simulator voldeden aan de verwachtingen.

C.3 Oplijsting aanpassingen verslag

- Verduidelijken van de figuur en paragraaf over robotdetectie.
- Licht inkorten van samenvatting.

D Beschrijving van het proces

- **Welke moeilijkheden heb je ondervonden tijdens de uitwerking?**
  Het uitwerken van een oplossingsstrategie was in grote mate afhankelijk van de beslissingen van de scheidsrechterscommissie. Deze beslissingen lieten soms te lang op zich wachten waardoor we in tijdsnood kwamen voor sommige demo's.
- **Welke lessen heb je getrokken uit de manier waarop je het project hebt aangepakt?**
  Documenteren van code is essentieel om zo ten alle tijde alle teamleden op de hoogte te houden van de werking van het project. Zo wordt vermeden dat het project te afhankelijk wordt van één persoon die een cruciaal deel van de code heeft geschreven. Verder is het belangrijk om de presentaties van de demo's voor te bereiden. Bij de tweede demo, bijvoorbeeld, hadden we wel een functioneel project, maar de presentatie werd slecht aangepakt. Hierdoor konden we deze functionaliteit in feite niet laten zien.
- **Hoe verliep het werken in team? Op welke manier werd de teamcoördinatie en planning aangepakt?**
  Een specifieke planning is nooit neergeschreven, maar de taken werden wel elke week mondeling en via een Facebook-groep besproken. Iedereen wist telkens voor welke taken hij/zij verantwoordelijk was. Verder waren alle teamleden altijd aanwezig wanneer dat van hen verwacht werd. Kortom, de samenwerking ging vrij vlot.

E Beschrijving van de werkverdeling

Week	**Samuel**	**Ruben**	**Robin**	**Eline**	**Dries**
Week 1	6,5	6,5	6,5	6,5	6,5
Week 2	8,5	7	5,5	5,5	5
Week 3	14,5	8	6,5	14,5	6
Week 4	11,5	5	8	11,5	11
Week 5	6	9	3,5	8	4
Week 6	16,5	15	12	10,5	8
Week 7	16	17	13,5	14	13,5
Week 8	0	14	0	0	0
Week 9	0	15	0	0	0
Week 10	6,5	10	5	9	5
Week 11	8,5	12	7	7	8
Week 12	22,5	18	9,5	11,5	9
Week 13	20	14	10,5	15	9
Demo 1	34,5	26,5	23,5	31,5	22,5
Demo 2	43,5	41	29	39	25,5
Demo 3	55	83	32	41,5	31
Totaal	137	150,5	87,5	113	85

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

E.1 Eline

Als teamleider: een goede samenwerking en communicatie bevorderen.

- Demo 1: Algemene strategie helpen bedenken. Fysieke robot verder afstellen. Barcodes en over de wip rijden.
- Demo 2: Robot herbouwen en afstellen robot. Testen met infrarode sensor.
- Demo 3: Robot nauwkeuriger en robuuster maken. Testen en debuggen van de fysieke robot

E.2 Samuel

Als secretaris: verslagen schrijven en logboek en andere documenten bijhouden.

- Demo 1: Algemene strategie helpen bedenken. Onderzoeken hoe RabbitMQ werkt en nadenken over protocol.
- Demo 2: Herbouwen en afstellen van robot.
- Demo 3: Robotdetectie en ontwijkstrategieën. Debuggen en testen.

E.3 Ruben

Als afgevaardigde: overleggen met de groep om de vergaderingen van de scheidsrechterscommissie voor te bereiden, aanwezig zijn op elke vergadering en communiceren over de genomen beslissingen.

- Demo 1: Algemene strategie mee bedenken en simulator uitbreiden.
- Demo 2: Verschillende algoritmes voor het spel programmeren (communicatie, samenvoegen van doolhoven, naar teammate gaan).
- Demo 3: Refactoren van de pc-code. Debuggen en uitbreiden van het spelalgoritme. Ondersteuning geven bij nieuwe GUI.

E.4 Robin

- Demo 1: Bedenken algemene strategie. Opnieuw afstellen van de lichtsensor. Oppikken van het voorwerp.
- Demo 2: Herbouwen van de robot. Infraroodsensor en rijden op de wip.
- Demo 3: Corrigeren van afwijkingen bij het rijden. Rijden op de wip verbeteren.

E.5 Dries

- Demo 1: Uitbreiden van de gebruikersinterface. Algemene strategie bedenken en opnieuw afstellen van de lichtsensor.
- Demo 2: Uitbreiden van de gebruikersinterface. Herbouwen van de robot.
- Demo 3: Herontwerpen van gebruikersinterface.

F Evaluatie van de oplossingsstrategie

In de initiële oplossingsstrategie [7] wilden we met een webcam het doolhof overzien om de posities van de robots te weten te komen. Dit hadden we nodig om een centrale server botsingen te laten controleren. Deze informatie is nu ter beschikking doordat alle robots hun positie naar elkaar toesturen, wat uiteraard veel gemakkelijker is. In plaats van botsingen op te lossen, worden die nu gewoon vermeden. Het bouwen van een centrale server die de betrokken robots laat onderhandelen, zou veel tijd in beslag hebben genomen. De huidige aanpak is minder efficiënt, maar werkt wel.

Ons idee om barcodes als checkpoints te gebruiken om doolhoven samen te voegen, werd overgenomen door de scheidsrechterscommissie. Deze aanpak maakt het zeer gemakkelijk om doolhoven samen te voegen. Bij het berekenen van het kortste pad naar de teamgenoot, wordt geen vergelijking gemaakt tussen verschillende mogelijke paden met onzekere boorden. Onzekere boorden worden gezien als doorgangen. Indien dit niet zo blijkt te zijn, dan wordt het pad gewoon herberekend. De kans dat doolhoven niet overeenkomen bleek in verschillende testen zeer klein, waardoor er geen tijd werd gestoken in een uitgebreide strategie hiervoor.

We hebben twee mogelijke strategieën beschreven om de wip aan te pakken. Uiteindelijk is ervoor gekozen om de wip niet te verzwaren. Dit leidt tot extra complicaties in de software, maar leidt tot minder fysieke beperkingen. Mogelijks zou de wip niet gemakkelijk omslaan indien deze verzwaard zou zijn langs één kant. Het idee om met barcodes te werken om de wip aan te duiden, is door de scheidsrechterscommissie overgenomen.

G Kritische analyse

G.1 Sterke punten

- De simulator is goed ontworpen en liet daardoor toe om sneller en gemakkelijker verschillende strategieën uit te testen.
- Het verkenalgoritme is zeer robuust en voert voortdurend correcties uit, zoals bijvoorbeeld het rechtzetten op een witte lijn. Dit verkleint de kans dat de robot ergens tegenaan botst aanzienlijk en brengt het verkennen niet in gedrang.
- De GUI is zeer informatievol, maar toont niet meer dan nodig. De gebruiker wordt niet overladen met informatie, maar kan die wel ten alle tijde opvragen.

G.2 Zwakkere punten

- Het tekenen van het doolhof in de GUI gebeurt niet optimaal, waardoor hij te veel flikkert.
- Botsingen worden niet opgelost, maar enkel vermeden. Dit is een weinig efficiënte aanpak.

H Scheidsrechterscommissie

H.1 Evaluatie

Sommige kritische beslissingen werden te laat genomen en waren onvoldoende doordacht. Er werd bovendien af en toe op een beslissing teruggekomen, wat het moeilijk maakte om erop te rekenen. Bepaalde beslissingen, zoals het afspreken van een punt bij het naar elkaar toegaan of het onderhandelen bij botsingen, werden zeer laks behandeld. Deze beslissingen konden een zeer gunstige invloed hebben gehad op het project, maar weinig teams wilden toegevingen doen. Dit leidt tot spelsituaties die eigenlijk heel inefficiënt zijn. Daarnaast heeft de commissie niemand aangemoedigd om te werken aan de gemeenschappelijke code, waardoor het protocol eigenlijk vooral door één team is gemaakt.

H.2 Eigen inbreng

- Voorwerp in een nieuwe tegel na een barcode
- Barcodes als checkpoints bij het samenvoegen van doolhoven
- Barcodes om wippen aan te duiden
- Invulling van de tweede demo (volledig spel, zonder robotdetectie)
- Ongelimiteerde, virtuele robotdetectiesensor

I Beoordelingen

I.1 Ontvangen beoordelingen

I.1.1 Demo 1, van assistent

Verslag:

- De inleiding is geen inleiding tot het probleem en de mogelijke probleemoplossing.
- De referenties zijn geen referenties.
- Veel te weinig testen. Lijkt een ad-hoc project.
- Er mag verwezen worden naar Semester 1, maar essentiële dingen moeten herhaald worden.
- De beschrijving van de software en samenvoegen van doolhoven is zeer goed. De lezer wordt goed gegidst. Behalve bij sectie 4.4 ontbreekt er wat toelichting op een hoog niveau.
- Het verslag is compact zonder te veel overbodige tekst (er kan nog wel wat geknipt worden hier en daar).

Presentatie:

- De demo was zeer goed gestart, maar verliep uiteindelijk verre van goed.
- Er werd onderling gediscussieerd over de antwoorden op de vragen. Dit is niet zoals het hoort en geeft een slechte indruk naar de toeschouwers toe. Er werd ook gediscussieerd met de vragenstellers, ondanks het feit dat zij het bij het rechte eind hadden.
- De simulator was zeer mooi, maar hier is veel te weinig informatie over terug te vinden in het verslag.

I.1.2 Demo 2, van team Rood

Verslag:

- Algemeen zeer goed verslag en duidelijk leesbaar.
- Parameters worden niet voldoende getest. Ga op zoek naar een minimum!
- Er gaat veel aandacht in het verslag naar de algemene implementatie van RabbitMQ, maar er mist informatie over hoe dit nu interageert met de rest van de software van de robot.
- In vergelijking met de andere groepen is de samenvatting vrij lang, en geeft informatie die niet in het verslag voor komt (bvb. druksensoren).
- Te veel voorwaartse referenties. Probeer deze tot een minimum te beperken. Herstructureer desnoods het verslag.
- Bij het bespreken van de simulator worden een timer en de methodes van de robot vermeld, maar er volgt weinig uitleg over deze termen.

- Er wordt gebruik gemaakt van zeer informatie volle grafieken.

Presentatie:

- In het verslag werd vermeld dat er nagenoeg perfect met de wip rekening gehouden kon worden. Dit werd echter niet bevestigd door de presentatie.
- De simulator was iets te onduidelijk.

I.1.3 Demo 3, van assistent

Verslag:

- Het verslag is goed geschreven en zeer degelijk onderbouwd.
- Figuur 9 en de bijbehorende alinea zijn voor mij niet duidelijk.
- De samenvatting kan een beetje ingekort worden.

I.2 Gegeven beoordelingen

I.2.1 Demo 2, aan team Zilver

Verslag:

- Geen referenties!
- Vage lijn tussen abstract en inleiding. Inleiding geeft geen inleiding tot de opbouw van het verslag.
- Beschrijving van algoritme voor het samenvoegen van doolhoven ontbreekt.
- Robot is helemaal herbouwd, maar geen testen te vinden in het verslag?
- Geen juiste UML-notatie gebruikt; klassen geven verantwoordelijkheden niet weer.
- Verslag lijkt weinig volledig.

Presentatie:

- Goede presentatie, goed uitgelegd, goed taalgebruik.
- Robot reed zeer nauwkeurig en GUI was mooi.
- Eén persoon deed alle uitleg en beantwoordde alle vragen. Dit zorgde ervoor dat het minder chaotisch overkwam.
- Willekeurige lijnen op de GUI: ze wisten niet waarom.

Referenties

[1] Express-bot, http://www.nxtprograms.com/9797/express-bot/steps.html (2013-03-09)

[2] Het Team Treasure Trek Protocol, https://github.com/tgoossens/htttp-peno (2013-05-06)

[3] LeJOS API, http://lejos.sourceforge.net/nxt/nxj/api/ (2013-03-19)

[4] Manhattan Distance, http://xlinux.nist.gov/dads//HTML/manhattanDistance.html (2012-11-19)

[5] Moulin, M., Verspeurt, J. (2013). Maze Specification v2.5.1 and Barcode Specification v2.3.

[6] Pieters, R., Lannoy, S., Goossens, R., Verreydt, D., Vanrykel, E. (2012). P&O Computerwetenschappen, semester 1: finaal verslag.

[7] Pieters, R., Lannoy, S., Goossens, R., Verreydt, D., Vanrykel, E. (2013). P&O Computerwetenschappen, semester 2: oplossingsstrategie.

[8] Rabbit MQ, http://www.rabbitmq.com (2013-02-19)

[9] Russell, S., Norvig, P., Artificial Intelligence: A Modern Approach (3rd edition). Prentice Hall, 2009, 1152 p.