



Drone meshnetwerk simulatie

Afstudeerverslag

Versie 2.0

Alten Nederland B.V.

Hogeschool van Arnhem en Nijmegen

HBO Technische Informatica - Embedded Software Developement

MWJ.Berentsen@student.han.nl

Studentnummer: 561399

Docent: J. Visch, MSc

Assessor: ir. C.G.R. van Uffelen

M.W.J. Berentsen

12 juni 2019

Managementsamenvatting

Het doel van “Drone meshnetwerk simulatie” is het zetten van de eerste stap in de ontwikkeling van het dronenetwerk. Hiervoor is de volgende opdracht opgesteld:

Maak een netwerkmodule voor een drone die het mogelijk maakt om meerdere drones onderling te voorzien van een zelf Herstellend meshnetwerk. Maak deze module na in een simulatie die geschikt is om meer dan 100 drones te simuleren, waarvan het netwerk zich gelijk gedraagt zodat er verdeelalgoritmes getest kunnen worden.

Eerst is er gekeken welke simulatiesoftware gebruikt moet worden voor het simuleren van het dronenetwerk. Er is gekozen om dit te doen in de softwarecombinatie Ros + Gazebo omdat dit het hoge aantal drones kan verwerken en ook toegang heeft tot een physics engine.

Om een betere keuze te kunnen maken welke hardware gebruikt zou moeten worden voor onderlinge communicatie, is er eerst gekeken wat de grootte van de payload is die verstuurd zou moeten worden om nuttig te zijn in een netwerk.

Vervolgens is gekeken naar welke hardware gebruikt zou moeten worden voor de onderlinge communicatie. Hier is gekozen voor een Raspberry Pi model 2B+ met een daarop aangesloten radio module de nRF24l01+.

Hierna is er een keuze gemaakt welke routingstechniek gebruikt moet worden waarop de conclusie was om een hybride routingstechniek te gebruiken. Deze techniek heeft als basis een reactief protocol met als toevoeging dat richtingen opgeslagen worden in plaats van routes. De student heeft een aangepaste versie van de Lightweight Mobile Routing Protocol(LMR) toegepast.

Er is een conclusie getrokken dat het voor dit project niet nodig is om een drone volledig te implementeren. Er zijn regels opgesteld waar een drone minimaal aan zou moeten voldoen om een abstracte drone te representeren.

Om de opdracht uit te kunnen voeren heeft de student een simulatie gemaakt waar drones zich kunnen voortbewegen en zich conform de casus kunnen verdelen. De drones volgen het gedrag van een meshnetwerk en er is applicatie aanwezig om op realistische wijze verzoeken te sturen naar de drones voor verplaatsing.

De opgeleverde software wordt onderbouwd in het software design document en de software requirements specificatie.

Concluderend is er een simulator opgeleverd waarin drones verdeeld kunnen worden. Deze simulator kan gebruikt worden om te experimenteren met verdelingen en netwerkgedrag. Daarnaast is er een fysiek prototype opgeleverd van een netwerkmodule die zich gelijk gedraagt als de gesimuleerde netwerkmodule en via een REST service aangestuurd kan worden.

Inhoudsopgave

1	Inleiding	6
2	Bedrijfsbeschrijving	8
2.1	Het team	8
3	Opdrachtoomschrijving	10
3.1	Probleem	10
3.2	Doelstelling	10
3.3	Opdracht	11
3.3.1	Casus	11
3.4	Op te leveren resultaten	12
4	Methoden, technieken, platforms, tools	13
4.1	Github naast Alten haar eigen Git server	13
4.2	Game engine	13
4.3	Redmine	14
4.4	L ^A T _E X	14
4.5	Template formulieren gebruik	15
4.6	Template Vector3 class	15
4.7	CppCheck	16
4.8	GoogleTest	16
4.9	CMake	16
5	Uitvoer projectplan	18
5.1	Het plan	18
5.2	Reflectie	18
6	Proces en aanpak	20
6.1	Inceptiefase	20
6.2	Elaboratiefase	20
6.3	Constructiefase	21

7 Resultaten	29
7.1 Plan van aanpak	29
7.2 Broncode	29
7.3 Onderzoeksrapport	30
7.4 Simulatiesoftware	30
7.5 Hardwareprototype	31
7.6 Software requirements specificatie	31
7.7 Software design document	31
8 Evaluatie	32
8.1 80% versie	32
8.2 Afstudeereiland	32
8.3 Presentatie oefening voor verdediging	33
8.3.1 Feedback op de presentatie	34
8.3.2 Conclusie	35
9 Conclusie	36
9.1 Advies vervolg project	36
Literatuur	38
A Documenten	39
A.1 Onderzoeksrapport Drone meshnetwork simulatie	39
A.2 Plan van aanpak	39
A.3 SoftwareDesignDocument	39
A.4 SoftwareRequirementSpecification	39
B Broncode	40
C Resultaat CppCheck	41
D Resultaten GoogleTest	42
D.1 gtest-drone_meshnetwork_simulation-GatewayDrone-test.xml	42
D.2 gtest-drone_meshnetwork_simulation-HybridLMRoutingProtocol-test.xml	42
D.3 gtest-drone_meshnetwork_simulation-Vector3-test.xml	42
E Templates	43
E.1 IterationAssessment.pdf	43
E.2 Iteratieplan.pdf	43
E.3 Usecase.pdf	43

F	Videos simulatie netwerkherstel door drone verplaatsing	44
F.1	enkele verloren drone situatie 1.mp4	44
F.2	enkele verloren drone situatie 2.mp4	44
F.3	groep verloren drones situatie 1.mp4	44
F.4	groep verloren drones situatie 2.mp4	44
F.5	groep verloren drones situatie 3.mp4	44
G	Videos drone simulatie	45
G.1	GatewayWissel.mp4	45
H	Video fysiek hardware prototype	46
H.1	eerstecontact.mp4	46
H.2	Fysieke router en drone.mp4	46

Begrippenlijst

Term	Beschrijving
Broadcast	Het rondsturen van een bericht die iedereen mag ontvangen.
Debug	Term die slaat op debugger, vaak informatie die gebruikt wordt om gedrag te vinden binnen een applicatie.
Gateway	Een toegangspunt tot een netwerk.
Gazebo	Simulatiesoftware met ondersteuning voor physics.
nRF24l01+	Radio transceiver module werkzaam op de 2,4Ghz band.
Physics engine	Software die natuurwetten toepast op virtuele objecten.
Raspberry Pi model 2B+	Micro computer geschikt voor prototyping.
Ros	Robot operating system. Wordt gebruikt voor de transportlaag naar zowel virtuele als gesimuleerde robots.
Rosservice	Een request response mechanisme gebruikt door Ros nodes.
Rostopic	Publicatie voor het gebruik van een subscriber patroon binnen Ros.
Routeringstechniek	Techniek die gebruikt voor het opbouwen van een pad binnen een netwerk.
RQT	Programma aangeboden door Ros voor aansluiting op Ros nodes.
SDF	XML format gebruikt voor het inladen van objecten en werelden in Gazebo
Seriële verbinding	Een communicatieverbinding het meest bekend van USB.
Simulatie software	Software die fysieke objecten nabootst in software.

Tabel 1: Begrippenlijst.

1 — Inleiding

Dit document is het eindverslag van de afstudeeropdracht genaamd “Drone meshnetwerk simulatie”. De simulatie simuleert een netwerk van drones die onderling met elkaar verbonden zijn door het gebruik van meshnetwerk technieken. De simulatie dient een tweevoudig doel: Allereerst kunnen er met de simulatie verdeelalgoritmes getest worden. Het tweede doel is het simuleren van het netwerk om zo de stabiliteit hiervan te kunnen optimaliseren.

Het doel van dit document is het aantonen van de volwassenheid van het project en de opgeleverde deelproducten conform de vijf beroepscompetenties. Dit wordt bewezen aan de hand van de producten die zijn gerealiseerd of handelingen die zijn verricht tijdens de afstudeerstage. In dit document wordt regelmatig verwezen naar andere documenten zoals het plan van aanpak, onderzoeksrapport, software design document en de software requirements. Dit wordt gedaan omdat deze documenten aantonen dat de student zich HBO waardig heeft gedragen tijdens dit project.

Het eindverslag is zo opgebouwd dat dit de opdracht, context en eindproducten beschrijft en hoe de student bezig is geweest conform de vijf beoordelingscriteria.

- **Bedrijfsbeschrijving.**

Hier wordt het bedrijf omschreven waar de opdracht wordt uitgevoerd en het team wordt besproken.

- **Opdrachtoomschrijving.**

Dit hoofdstuk wordt gebruikt om de aanleiding, probleemstelling, doel, opdracht en resultaten toe te lichten.

- **Methoden, technieken, platforms, tools.**

De gebruikte methoden, technieken, platforms en tools worden in dit hoofdstuk toegelicht. Hier gaat het met name om waarom er een keuze is gemaakt en wat de alternatieven waren. De projectmanagement methodiek blijft hierbuiten aangezien deze al wordt toegelicht in het plan van aanpak.

- **Uitvoer projectplan.**

Hier wordt het plan van het project gereflecteerd. Er wordt gekeken wat er volgens planning ging maar wat ook niet en waarom dat dan zo was.

- **Proces en aanpak**

In het hoofdstuk proces en aanpak wordt er gekeken welke stappen zijn genomen en wat hun verband is met betrekking tot het proces

- **Resultaten**

Wat is er opgeleverd en voldoet het aan de opgestelde kwaliteitseisen gesteld in het plan van aanpak. Als er afwijkingen zijn worden die ook in dit hoofdstuk besproken.

- Evaluatie

Dit hoofdstuk geeft inzicht met betrekking tot het proces, producten en resultaten en het functioneren van de student.

- Conclusie

Het afsluitende hoofdstuk behandelt wat de student heeft opgeleverd en of dit voldoet aan de doelstelling van het project. Als er afwijkingen zijn wordt dit ook beschreven. Tenslotte worden er een advies gegeven aan het bedrijf in welke vervolgstappen zij zou kunnen nemen.

2 — Bedrijfsbeschrijving

Alten is een multinational met +- 25.000 medewerkers wereldwijd en is van origine begonnen in 1988 en in de eerste tien jaar organisch gegroeid totdat ze in 1999 een beursgenoteerd bedrijf werden. Vanaf het jaar 2000 hebben ze de eerste internationale dochterondernemingen gestart waarvan de eerste in Nederland in 2002 was (dit was een overname van een ander bedrijf). In 2005 is Alten NL tot leven geroepen die zich focust op het geven van technische consultancy. Op dit moment heeft Alten NL ongeveer 650 medewerkers, waarvan elk persoon minimaal een bachelors diploma heeft, 30-40% een masters diploma, en 10-15% een PhD heeft. Dit maakt Alten een vooruitstrevend bedrijf die zich altijd bezig houdt met innoverende technologieën en probeert vooraan te staan wanneer er een nieuwe technologie doorbreekt.

Anno 2019 heeft Alten vier filialen in Nederland: Amstelveen, Capelle a/d IJssel, Eindhoven en Apeldoorn. De student is geplaatst bij het filiaal in Apeldoorn.

Zoals gezegd focust Alten zich vooral op technologie en dat is niet enkel beperkt tot software want zo hebben ze namelijk ook: Technische Software, mechatronica, IT test services, Business Intelligence & Analytics en Digital Enterprise. Alten faciliteert voornamelijk in consultancy (85% van alle werkzaamheden) maar doen ook ‘in huis’ projecten voor haar klanten.

Het bedrijf heeft zes kernwaarden die als volgt luiden: kennisnetwerkers, technologie, streef hoog, open en betrokken, mensen en plezier/lol. Alten benadrukt dat het belangrijk is om je kennis te delen met anderen (want samen sta je sterker), dit proberen zij te bereiken met de zogeheten kennisgroepen.

2.1 Het team

Het team betrokken bij dit project wordt hier kort omschreven.

Uitvoerend student: Maurice Berentsen Het afstudeerproject is uitgevoerd door de student Maurice Berentsen. Hij is een student aan de Hogeschool van Arnhem en Nijmegen. Zijn belang bij dit project is tweezijdig. Enerzijds voert Maurice dit project uit om aan te tonen hij de vijf beroepscompetenties beheerst en daarom waardig is voor het ontvangen van het Bachelor diploma Technische Informatica. Anderzijds voert hij dit project uit omdat hij een interesse heeft naar mobiele meshnetwerken en dit na het behalen van zijn diploma in de praktijk zou willen brengen bij een toekomstig opdrachtgever.

Technisch manager: Hugo Logmans Hugo Logmans is de technische manager voor Alten oost. Hij begeleidt de consultants in hun carrière en werk. Sinds 1999 is Hugo werkzaam als ontwikkelaar en heeft hier dus bijna 20 jaar ervaring in. Zijn belang in

dit project is het onderzoeken wat er toegevoegd kan worden in een meshnetwerk om het ondervinden van uitgevallen punten snel te achterhalen.

Business manager: Jeroen Nijenhuis Jeroen Nijenhuis is business manager van de locatie oost. Hij zorgt voor contactafhandeling naar bijvoorbeeld p&o. Bij hem kan de student terecht voor vragen buiten het vakgebied. Zijn belang bij het project is het zorgen dat de overgang van de schoolomgeving naar een bedrijfsomgeving goed verloopt

Bedrijfsbegeleider: Hugo Heutinck Hugo Heutinck is een werknemer van Alten. Hij heeft in 2003 een master behaald in Computer Technics en is sindsdien aan het werk als (embedded) software engineer. Hij is een specialist in Linux en heeft voert de rol uit van scrum master. Zijn belang bij dit project is het begeleiden van de student.

Begeleidende leraar: Jorg Visch Jorg Visch is een leraar op de Hogeschool van Arnhem en Nijmegen werkzaam binnen het lerarenteam van de uitstroomrichting Embedded Software Developer. Hij is leraar van Maurice geweest tijdens het semester Internet of Things. Zijn belang bij dit project is begeleiden van Maurice tijdens het afstuderen.

Assessor: Chris van Uffelen Chris van Uffelen is een collega van Jorg Visch en werkzaam in hetzelfde team. Hij heeft Maurice begeleidt tijdens het Object-oriented Software and Modelling project. Zijn belang bij dit project is het beoordelen of het project waardig is aan het verschaffen van het Bachelor diploma Technische Informatica. Hij doet dit in samenwerking met Jorg Visch.

3 — Opdrachtschrijving

Het doel van Alten is om onderling verbonden drones te kunnen verdelen om zo een netwerk op te kunnen bouwen over een gebied. Alten wil dat dit netwerk zichzelf kan onderhouden door te reageren op uitval of een slechte verbinding door de één of meerdere drones te herverdelen.

3.1 Probleem

Op dit moment is Alten in het bezit van drones maar deze kunnen niet met elkaar communiceren. Om te kunnen communiceren moet er een netwerkmodule toegevoegd worden aan elke drone. Alten wil grote netwerken kunnen opbouwen die robuust zijn en daarom wil zij gebruik maken van een zelf herstellend meshnetwerk. Hoewel er al meerdere oplossingen bestaan in het gebruik van meshnetwerken wil Alten het zelf herstellend vermogen vergroten. Daarom wil Alten dat de focus van de netwerkmodule ligt op het snel detecteren van uitval van punten in het netwerk zodat daar adequaat op gereageerd kan worden. Adequaat reageren kan op twee manieren volgens Alten. De netwerkmodule kan een ander netwerkpunt zoeken om via dat punt te communiceren. Wanneer deze niet beschikbaar is het alternatief om de drone opnieuw te verdelen. Het autonoom fysiek kunnen verplaatsen van de netwerkpunten is dan ook de uitbreiding die Alten wil toevoegen aan het zelf herstellend vermogen. Alleen wat is een efficiënte manier van herverdelen? Het makkelijkste is om alle drones terug naar hun start punt laten vliegen maar dit zou betekenen dat het netwerk op dat moment niet beschikbaar is en het zou ook nog eens onnodig veel stroom verbruiken.

Om algoritmes te testen voor het verdelen van de drones is een simulatie de oplossing. Hierbij is het dus ook van belang dat een simulatieomgeving zich realistisch gedraagt. Om realistisch gedrag na te bootsen moeten de virtuele netwerkmodules zich in de simulatie zich net zo gedragen als in het echt.

3.2 Doelstelling

Het doel van dit project is het zetten van de eerste stap in de ontwikkeling van het dronenetwerk. De eerste stap is ontwikkelen van een netwerkmodule voor het onderling verbinden van drones. Het is van belang dat het meshnetwerk van de drones snel kan reageren op uitval van netwerkpunten. Deze netwerkmodule moet zowel virtueel als fysiek gerealiseerd worden in dit project.

Voor het virtueel realiseren van de netwerkmodule moet een simulatie gebruikt worden. Omdat er nog geen simulatiesoftware beschikbaar is wil Alten dat de student uitzoekt welke

simulatiesoftware geschikt is voor het simuleren van meer dan 100 drones. Vervolgens kan de student deze software gebruiken om de virtuele netwerkmodules te testen.

3.3 Opdracht

De opdracht van dit project is de ontwikkeling van meerdere producten die bijdragen aan het behalen van het doel.

De student moet een prototype maken van een netwerkmodule die het onderlinge mesh-netwerk verzorgt en zichzelf kan herstellen bij uitval of slecht signaal. Wanneer er geen alternatieve communicatie route mogelijk is moet de module de drone een instructie uitsenden om zich te herpositioneren.

De student moet een simulatie opzetten waarin de netwerkmodules verplaatst kunnen worden in een vrije ruimte. In de simulatie moet het mogelijk zijn om de netwerkmodules te verplaatsen aan de hand van hun X , Y en Z as zodat er verdeelalgoritmes getest kunnen worden. De netwerkmodules moeten in de simulatie kunnen uitvallen en de onderlinge afstand heeft effect op de virtuele signaalsterkte. De simulatie moet aan de hand van een script herhaalbaar zijn met eenzelfde resultaat.

Alten wil graag dat de simulatiesoftware gebruik maakt van Robot Operating System (Ros) als middleware. Dit wil Alten omdat zij al kennis heeft in Ros maar nog niet in combinatie met drones en meshnetwerken en door het begeleiden hier ervaring in opdoet. Het ervaring opdoen in Ros is geen doel van dit project en de student hoeft zich hier niet op te concentreren.

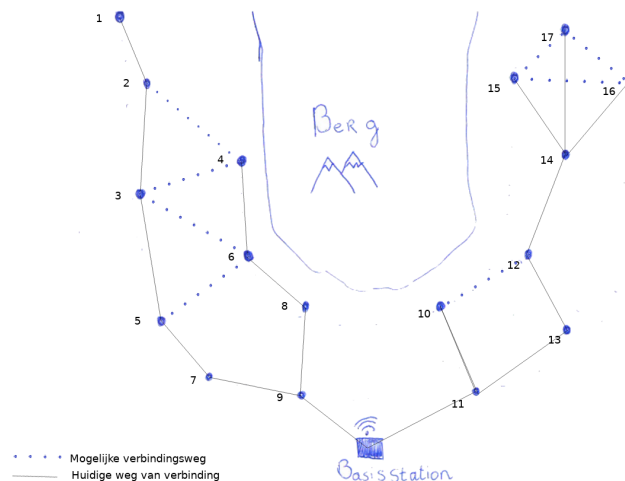
Hoewel de keuze van de simulatiesoftware zich concentreert op het simuleren van een dronenetwerk zal de student alleen de netwerkmodule toevoegen aan de simulatie. De student zal zich dus niet specialiseren in het realistisch simuleren van drones.

Het project is geslaagd als een prototype van het meshnetwerk is opgeleverd samen met een simulatie waarin het meshnetwerk getest kan worden met verschillende verdelingsalgoritmes.

3.3.1 Casus

Om het doel waar naartoe gewerkt moet worden te voorzien van een richting heeft Alten een casus bedacht. Deze casus wordt hieronder toegelicht in een schets en is voorzien van individuele situaties. Er wordt gedrag omschreven voor een simpel algoritme van het herstel van het meshnetwerk om aan te tonen dat dit getest kan worden in de simulatie. In de casus blijft het vlieggedrag buiten de scope.

In de schets van [Figuur 3.1](#) wordt er vanuit gegaan dat elke drone een directe vluchtroute heeft met zijn verbonden netwerkpunt. De verbindingen zijn gemaakt om de casus uit te leggen en volgen dus niet de realistische verbindingen die ze zouden hebben op basis van locatie.



Figuur 3.1: Schets van casus netwerk.

In de casus komen de volgende situaties apart voor:

- Punt 2 valt uit waardoor punt 1 geen verbinding meer heeft met het basisstation. Het verwachte resultaat is dat drone 1 zich verplaatst naar de positie van drone 2 en zo het netwerk herstelt.
- De bovenstaande situatie gebeurt. Tijdens het verplaatsen van punt 1 herstelt punt 2 zich weer. Het verwachte resultaat is dat punt 1 weer terug keert naar zijn oude positie.
- Punt 3 valt uit waardoor punt 1 en 2 geen verbinding meer hebben met het basisstation. Het verwachte resultaat is dat drone 2 een nieuwe verbinding aangaat met drone 4.
- Punt 14 valt uit waardoor punt 15, 16 en 17 geen verbinding meer hebben met het basisstation. Het verwachte resultaat is dat de drones 15, 16 en 17 eerst met elkaar verbinden, vervolgens zal één van de drones zich verplaatsen naar punt 14.

3.4 Op te leveren resultaten voor het bedrijf en school

Aan het einde van het project levert de student de volgende producten op aan het bedrijf:

- Simulatiesoftware met netwerksimulatie
- Prototype meshnetwerk module.
- Broncode van de bovenstaande producten.
- SRS (Software Requirement Specification)
- SDD (Software Design Document)
- Onderzoeksverslag

Aan het einde van het project levert de student het bovenstaande plus een projectverslag op aan de Hogeschool van Arnhem en Nijmegen.

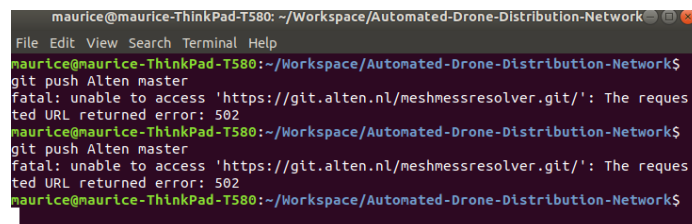
4 — Methoden, technieken, platforms, tools

Dit hoofdstuk wordt gebruikt om de methoden, technieken, platforms en tools toe te lichten.

4.1 Github naast Alten haar eigen Git server

Gedurende het project is gebruik gemaakt van een dubbele git server voor versiebeheer. Hiervoor is een private repository op github.com gebruikt en de aangeboden server van Alten.

Deze keuze is gemaakt omdat het afstudeerproject een belangrijk project is. Daarom is het belangrijk dat het aantal single points of failure zo laag mogelijk blijft. Om een werknemer van Alten te citeren: “Je kan gerust alleen gebruikt maken van onze server daar is nooit wat mee” bleek toch het tegendeel zie [Figuur 4.1](#)



```
maurice@maurice-ThinkPad-T580: ~/Workspace/Automated-Drone-Distribution-Network
File Edit View Search Terminal Help
maurice@maurice-ThinkPad-T580:~/Workspace/Automated-Drone-Distribution-Network$
git push Alten master
fatal: unable to access 'https://git.alten.nl/meshmessresolver.git/': The request URL returned error: 502
maurice@maurice-ThinkPad-T580:~/Workspace/Automated-Drone-Distribution-Network$
git push Alten master
fatal: unable to access 'https://git.alten.nl/meshmessresolver.git/': The request URL returned error: 502
maurice@maurice-ThinkPad-T580:~/Workspace/Automated-Drone-Distribution-Network$
```

Figuur 4.1: Alten server niet beschikbaar.

Toen dit gebeurde was de keuze om niet afhankelijk te zijn van een enkele service terwijl dit niet hoeft een bewezen goede keuze.

4.2 Game engine

Tijdens het project is gebruik gemaakt van de simulatiesoftware Gazebo Deze software-keuze is gebaseerd op het onderzoek die van te voren is uitgevoerd en wordt daar ook ruim onderbouwt. Een alternatief die ik niet in het onderzoek heb meegenomen maar nog wel hier wil aankaarten is de overweging om een game engine te gebruiken. Deze keuze heb ik voor het onderzoek overwogen omdat game engine geavanceerde physics bevatten en gemakkelijk te gebruiken zouden zijn voor prototyping. Daarnaast heb ik voor het afstuderen begon de minor GAME gedaan waardoor ik ook al de nodige ervaring heb in het werken met game engines. De reden dat ik uiteindelijk de keuze niet meegenomen heb

in het onderzoek is omdat ik wist dat game engines niet geschikt zijn om code te verwerken die zonder al te veel moeite ook op een microcontroller werken. Daarnaast zou het gebruiken van een game engine ook niet passen in mijn uitstroomrichting als embedded software developer.

4.3 Redmine

Voor het bijhouden en plannen van het project is gebruik gemaakt van een Redmine omgeving. De keuze om deze te gebruiken is gekomen door de positieve ervaring opgedaan met Redmine tijdens het Internet of Things semester. Een overwogen alternatief was het gebruiken van Github projects omdat deze een gemakkelijkere koppeling zou hebben naar de commits op git dan Redmine heeft. Ook zou de service van Github projects betrouwbaarder zijn dan de door Alten gehoste Redmine server.

De keuze om toch de Redmine server van Alten te gebruiken is gebaseerd op het volgende:

- Inzichtelijker voor mijn begeleiders vanuit Alten.
- Standaard geïntegreerde Gantt planning.
- Mogelijkheid voor sprint planningen.
- Bij uitval is de planning tijdelijk niet inzichtelijk dit wordt niet als kritisch beschouwd.

4.4 L^AT_EX

De keuze om Latex (LaTeX3 Team, z. j.) te overwegen in het project is al bijna twee jaar voor het afstuderen begon gemaakt. Toen de leraar Joost Kraaijeveld tijdens het semester World of Robots de mogelijkheden liet zien van Latex is er in eigen tijd meer aandacht gestoken in de markup taal. Het werd duidelijk dat Latex de tool was voor het schrijven van wetenschappelijke artikelen of technische documentatie. Hier kwam uit dat Latex veel voor je kan automatiseren zoals referenties, bibliografie, betiteling en het opstellen van formules maar dat er wel een leercurve bij zit. Om te voorkomen dat ik tijdens het afstuderen nog door die leercurve heen zou moeten gaan ben ik toen al begonnen met kleine documenten te schrijven in Latex voor opdrachten van semesters. Toen ik dit meer onder knie had heb ik besloten ook mijn verslag voor de meewerkstage te schrijven in Latex. Tijdens de stage werd door alle begeleiders aangegeven dat het verslag er zeer netjes en verzorgd uitziet. Ook heb ik tijdens de meewerkstage ondervonden dat deze tool veel geschikter is voor versiebeheer dan de concurrent Microsoft Word.

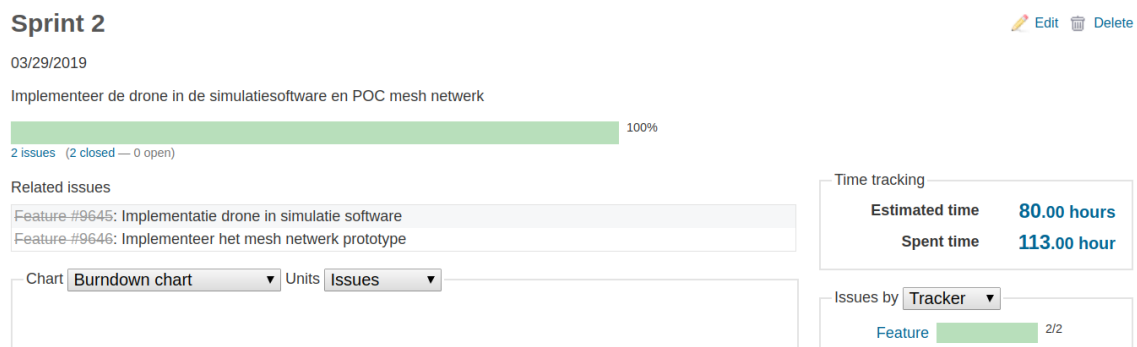
Tijdens het afstuderen is Word ook geen praktisch alternatief omdat het project uitgevoerd wordt met Ubuntu. Daarop kan geen Word geïnstalleerd worden waarbij de linux variant van LibreOffice overblijft of de online tool van Word.

Omdat ik deze beide geen waardig alternatief vond ten opzichte van Latex, waar ik ondertussen aardig bekwaam in ben, heb ik definitief besloten tijdens het afstuderen om Latex te gebruiken

4.5 Template formulieren gebruik

Tijdens de eerste fase van het project is er het plan geweest om templates te gebruiken gedurende het project. Hiervoor was gekozen omdat het tijdens het WoR project veel gemak opleverde.

Met het afstuderen kwam ik er al snel achter dat bepaalde templates voor dit project niet nodig waren. Zo is er een template waar de planning van de komende iteratie in opgeschreven moest worden te vinden in bijlage E.2. Dit template is uiteindelijk niet gebruikt. Er is besloten om de planning in Redmine bij te houden zoals te zien in [Figuur 4.2](#).



Figuur 4.2: Redmine sprint 2 planning.

Mijn begeleiders waren niet altijd op locatie en door het registeren van de planning in Redmine was er snel en gemakkelijk overzicht in de sprintplanning.

Een template die voortvloeit uit de hiervoor beschreven template is het IterationAssessment uit bijlage E.1. De sprints zijn veel informeler afgesloten door het er gewoon mondeling over te hebben met de begeleider tijdens de lunch of de koffietafel. Achteraf was het niet zo slim om het op deze manier af te sluiten zonder dat ook op schrift vast te leggen. Ik denk dat dit een valkuil is voor studenten die afstuderen omdat ze toch vaker alleen aan een opdracht werken. Hoewel er stakeholders waren tijdens het project die natuurlijk ook op een later moment besproken stof willen kunnen teruglezen is het er niet gekomen. Dit is een fout geweest van mij die er ingeslopen is. In een volgende project wil ik daarom dus ook absoluut hier verandering in brengen en daar de tijd nemen om meer vast te leggen.

Een ander template die wel gebruikt is, is de template voor een usecase te vinden in bijlage E.3. Dit template heeft veel gemak opgeleverd tijdens opstellen van het SRS (bijlage A.4). Omdat er al een template aanwezig was heb ik mij volop kunnen concentreren op wat er toe doet en dat is het opstellen van de usecase.

4.6 Template Vector3 class

Tijdens het bouwen van de software van het netwerkcomponent is er begonnen met het maken van software voor de simulatie. Vanuit gemak is er op dat moment gebruik gemaakt van de bestaande Vector3 die meegeleverd wordt met Ros (Open Source Robotics Foundation, 2014). Op dat moment was het een makkelijke oplossing zodat er snel verder gewerkt kon worden.

Later toen de software ook moest draaien op een Raspberry Pi kwam de afhankelijkheid terug naar voren die gecreëerd was. Er is gekozen om de vector3 zelf te implementeren in een template klasse aangezien niet alle functionaliteit van de vector3 uit de ignition library (Open Source Robotics Foundation, 2014) gebruikt wordt. Om specifiek te zijn wordt er zelfs maar een enkele functie gebruikt voor het bepalen van een afstand tussen twee vectoren. Die is als volgt in de code geïmplementeerd.

```
template <typename T>
float Vector3<T>::Distance(const Vector3<T>& rhs) const
{
    Vector3<T> difference(*this);
    difference -= rhs;
    return std::sqrt( std::pow( difference.X(), 2 ) + std::pow( difference.Y(), 2 )
        + std::pow( difference.Z(), 2 ) );
}
```

In deze functie wordt de stelling van Pythagoras gebruikt om de afstand tussen twee punten in een drie dimensionale ruimte te berekenen.

Door het maken van deze klasse is de code van het netwerkcomponent onafhankelijk geworden van Ros libraries.

4.7 CppCheck

Tijdens het project is gebruik gemaakt van een statische code checker. De keuze was gevallen op de CppCheck vanwege het gebruikersgemak. Bij oplevering toont de tool geen melding meer, deze zijn onderdrukt of opgelost. De resultaten van CppCheck zijn te vinden in bijlage C.

4.8 GoogleTest

Gedurende het project is er gebruik gemaakt van unit testen. Deze testen zijn geschreven via het framework GoogleTest. Dit leverde toen er op een later moment een template class van een 3D vector (bijlage B/drone_meshnetwork_simulation/src/Vector/) gemaakt werd het voordeel dat deze test driven ontwikkeld kon worden. Verder is de mogelijkheid van testen gebruikt om de opslag van routerichtingen te verifiëren en het coderen van berichten te testen. Een uitdraai van de de unittesten is terug te vinden in bijlage D

4.9 CMake

Aan het einde van het project heeft de gesimuleerde software de overstap gemaakt naar het fysieke prototype. Hiervoor moest de code gecompileerd worden op een Raspberry Pi. In de simulator werd ik als ontwikkelaar natuurlijk al een beetje verwend door het standaard aanwezige catkin in Ros. Voor het compileren van de code op de Raspberry Pi kwam ik eerst nog weg door telkens een lang commando in te voeren waarmee het router programma werd aangemaakt. Later toen er ook nog een gateway programma aangemaakt moest worden heb ik besloten om een overstap te maken naar CMake (CMake, z. j.).

Met CMake kon ik met een enkel commando beide programma's simpel compileren op de Raspberry Pi. Dit bewees zijn nut helemaal op het moment toen er getest moest worden met meerdere Raspberry Pi's.

5 — Uitvoer projectplan

In de dit hoofdstuk wordt het plan eerst kort toegelicht, vervolgens wordt gekeken naar waar de afwijkingen zitten en waarom deze er waren.

5.1 Het plan

Het plan was om dit project volgens de RUP ontwikkelmethode uit te voeren. Omdat er nog niet voldoende voorkennis aanwezig was voor het bouwen van de software geeft dit de ruimte om voldoende onderzoek uit te voeren en daar de kennis uit te halen. Er was gepland dat de fases inception, elaboration, construction en transition een duur hadden van vijf, acht, vier en twee weken. Dit geeft de student dusdanig veel tijd voor onderzoek en het opzetten van losse prototype componenten dat er tijdens een constructiefase van vier weken weinig problemen konden voorkomen.

In de eerste fase, de inception fase, was gepland dat er een plan van aanpak geschreven werd en dat er ruimte was voor kleine onderzoeken. De resultaten van de onderzoeken konden meteen meegenomen worden in de planning van het te komen onderzoek. Vervolgens tijdens de elaboratie fase stond het onderzoek naar de te gebruiken hardware voorop omdat hier de te bouwen simulatie op gebaseerd moest worden. Op het moment dat de hardware bekend was kon deze wanneer nodig besteld worden en werd er gekeken naar welke simulatiesoftware gebruikt moest worden. Toen ook dit bekend was, was de volgende stap het bouwen van losse componenten in de simulatie. Zodra de hardware beschikbaar was kon de software die voor simulatie gemaakt was gebruikt worden op de hardware. In de constructiefase kon de gebouwde software en hardware gebruikt worden voor het beantwoorden van onderzoeksvragen. Er zou er in deze fase van het project voldoende tijd moeten overblijven voor het afronden van de documentatie. Tenslotte zou er in de transitiefase ruimte zijn voor het maken en geven van presentaties voor school en het bedrijf

5.2 Reflectie

Zoals bij de meeste projecten is dit project anders verlopen dan gepland is. Zo begon het met de tegenvaller dat er veel meer aandacht naar het plan van aanpak moest dan verwacht was. Hierdoor was er minder tijd om kleine onderzoeksvragen uit te voeren die wel van belang waren in voor de onderzoeken in de elaboratiefase. Hierdoor moesten deze doorgeschoven worden.

Omdat er een onduidelijkheid zat wat er opgeleverd moet worden bij de 80% versie, wat beter omschreven wordt in [Paragraaf 8.1](#). Is er in week drie van de elaboratie fase besloten dat er nieuwe planning gemaakt moest worden om aan de vraag van school te voldoen.

Hier is besloten dat het beter zou zijn dat het hardware prototype pas gebouwd zou worden na het inleveren van de 80% versie als de beschikbare tijd het toestaat. De focus is verschoven naar het bouwen van een correct werkende simulatie die af is voor de 80% versie. Het deel van het onderzoek dat al uitgevoerd is, bevatte genoeg handvatten voor het bouwen van alle software. Daarom is deze ontwikkeling ook meteen gestart en is de software stapsgewijs opgebouwd. Hoe dit gedaan is wordt omschreven in [Paragraaf 6.3](#).

Dit wijkt af van de RUP methodiek waarbij componenten los van elkaar gebouwd worden en waar de construction fase gebruikt wordt om deze componenten op elkaar aan te sluiten. Dit besluit is genomen wegens de harde deadlines uit het project. Het direct op elkaar aansluiten van de componenten tijdens het schrijven van de software kan wanneer correct uitgevoerd tijd schelen maar brengt het risico met zich mee dat componenten gaan afwijken in hun interface. Deze keuze kon alleen gemaakt worden door het feit dat het afstudeerproject alleen wordt uitgevoerd waardoor er altijd genoeg kennis is van alle componenten en interfaces.

Concluderend kan gesteld worden dat voor dit project RUP de juiste ontwikkelmethode is ondanks dat het verkeerd is uitgevoerd. Het onderzoek van tevoren naar de te gebruiken technieken heeft de kennis opgeleverd welke nodig is voor het project. Tijdens de elaboratiefase en de constructiefase zijn de regels van RUP wellicht geschonden, maar net als elke methode zijn de regels niet in steen gehouwen. Door het flexibele optreden is wel een werkend product neergezet die het doel verzadigd van Alten terwijl de student ook zijn doel heeft behaald in het bewijzen dat hij bekwaam is als embedded software engineer.

Uiteindelijk is ook gebleken dat de keuze om de hardware pas op te leveren na de 80% versie zijn vruchten heeft afgeworpen. Nadat de 80% versie is opgeleverd is er nog een oplevering geweest van het hardware prototype. Hiermee is alles wat beloofd is opgeleverd. Dit bewijst dat als de focus op het juiste ligt, je kan opleveren wat beloofd is.

6 — Proces en aanpak

In dit hoofdstuk wordt het proces en de aanpak van het afstudeerproject omschreven. In het project is gewerkt met de RUP ontwikkelmethodiek, aan de hand van deze fases worden de verschillende stappen van het project doorgelopen.

6.1 Inceptiefase

Aan het begin van het afstudeerproject is een plan van aanpak (projectplan) opgesteld met input vanuit Alten. Dit plan is toegevoegd als bijlage [A.2](#) Om achter de eisen en wensen van Alten te komen zijn middels meerdere gesprekken het probleem en doel van de opdracht geanalyseerd om deze concreet te beschrijven. Voor dit onderdeel waren meerdere gesprekken nodig, omdat de achtergrond van het project in eerste instantie niet concreet omschreven was. Door de meerdere gesprekken met Alten is de opdracht geconcretiseerd. De geconcretiseerde opdracht is te lezen bij [Hoofdstuk 3 Opdrachtomschrijving](#).

Voor het schrijven van het plan van aanpak was vijf weken ingepland. In eerste instantie werd geschat dat die dusdanig veel tijd was dat er tijdens deze fase al begonnen kon worden aan kleine onderzoeken. De reden hierachter was het idee dat de uitkomst van deze kleine onderzoeken zou helpen in het inschatten van een planning.

Doordat het concept plan nog niet concreet genoeg was heeft dit meer aandacht en tijd nodig gehad dan verwacht. Hierdoor zijn de volle vijf weken van de inceptiefase gebruikt om tot een kwalitatief plan te komen voor het project.

6.2 Elaboratiefase

De elaboratie fase is begonnen met het opstellen van een software requirements specificatie(SRS) te vinden in bijlage [A.4](#). Dit is gedaan met de requirements van de opdrachtgever uit het plan van aanpak en het doorvragen aan de opdrachtgever. Door het vooraf maken van dit document is de opdracht verder geconcretiseerd. Deze requirements konden weer meegenomen worden in het onderzoeksverslag als criteria in keuzes. Er is rekening mee gehouden dat er bepaalde eisen aan de software pas duidelijk zouden worden gedurende het onderzoek. Vandaar dat het schrijven van de onderzoeken en het schrijven van een SRS enige overlap zou hebben.

Het schrijven van het onderzoek (bijlage [A.1](#)) is begonnen met het opdelen van de hoofdvraag in meerdere deelvragen. Deze deelvragen zijn vervolgens ingepland als taken op het scrum bord in Redmine. Deze taken zijn vervolgens in sprints afgerond. Dit betekend dat er in iteraties van twee weken gewerkt is. De onderzoekstaken zijn van te voren al

toegewezen aan een specifieke sprint. Bij het begin van elke sprint zijn de taken geconcretiseerd. Deze keuze is gemaakt omdat veel taken in het project de kennis vereisen van de taak ervoor.

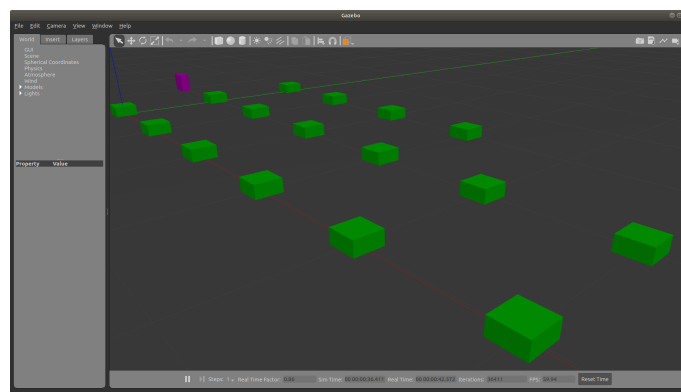
Er is het besluit gemaakt om de elaboratiefase met vier weken te verkorten. Dit besluit was gemaakt omdat er bij de zogeheten 80% versie van het project een product opgeleverd moet worden die al bijna voldoende is om op af te studeren. Er was op dat moment voldoende kennis uit het onderzoek aanwezig om een goede simulatie te kunnen schrijven die zich zou houden aan de limieten van de te gebruiken hardware. Meegenomen in het besluit was dat de HAN zelf ook een hogere prioriteit heeft op softwareontwikkeling dan het onderzoek ook al moeten ze wel beiden aanwezig zijn.

Daarom heeft de laatste week van de elaboratie fase in het teken gestaan van het op orde maken van het Software Design Document te vinden in bijlage A.4. Deze week was ook de eerste week dat er code geschreven is. Dit was vaak code die bestond uit lege classes of kleine stukjes proef code zoals het uitproberen hoe een drone geplaatst kon worden in Gazebo.

6.3 Constructiefase

Het voorgaande besluit hield in dat er nog zes weken de tijd was om de simulatie te realiseren voor de 80% versie ingeleverd moest worden.

Abstracte gesimuleerde drone Er was begonnen met het ontwikkelen van een abstracte drone die voorzien is van een interface om zich te verplaatsen naar verschillende locaties. Een abstracte drone houdt in dat alles wat niet van belang is weglaten mag worden. Van de drone bleef alleen blokje over zoals zichtbaar in [Figuur 6.1](#).



Figuur 6.1: Groep van drones in de simulatie.

Wat hierin een uitdaging was, was het injecteren van de drone informatie in Gazebo en het vervolgens aansturen. Dit is gedaan via een SDF beschrijving (Open Source Robotics Foundation, 2019) die een plugin heeft die de drone voorziet van een motor zoals in het onderstaand voorbeeld te zien is. Deze SDF beschrijving wordt volledig opgebouwd en geïnjecteerd in de simulatie door het aanmaken van een drone class. Deze heeft een statische variabele waardoor er nooit een dubbel ID optreedt, met uitzondering van een integer overflow.

```
<plugin name='DroneEngine' filename='libDroneEngine.so'>
  <DroneID>0</DroneID>
</plugin>
```

Deze motor kon vervolgens via een Ros service aangesproken worden. Zo kon er vanuit een aanroep in de console of het Ros programma RQT een drone verplaatst worden van positie.

Directe communicatie Nu er drones in de simulatie mogelijk waren betekende dat er meshnetwork componenten gekoppeld konden worden aan de drones in de simulatie.

Er is begonnen met het schrijven van een plugin die het netwerkcomponent representeerde in de simulatie. Vervolgens konden deze op twee drones 'aangesloten' worden om vervolgens deze met elkaar te laten communiceren. Het communiceren was nog een simpele applicatie waarbij een component een 32 byte array naar de andere moest sturen waarop het andere component weer response moest geven. Dit lukte al vrij snel waarop de volgende stap was het limiteren van de communicatie op basis van de onderlinge afstand.

Hiervoor is de draadloze communicatie simulator tot stand gekomen. Deze applicatie moet draaien om onderlinge communicatie mogelijk te maken omdat de netwerkcomponenten niet meer direct met elkaar mogen praten. In deze simulator zijn regels gesteld of het op basis van afstand het toegestaan is om een bericht te versturen. Een uitdaging hierin was het op de hoogte blijven van de locatie van een node. Er is besloten dat dit aan de hand van een Rostopic zou gebeuren zodat een drone, als die zich verplaatst, continue zijn nieuwe positie door kan geven. Een netwerkcomponent kan op zijn beurt op dit topic aangeven op welke drone hij aangesloten zit en op welk topic hij berichten wil ontvangen. De werking hiervan kon getest worden door drones met een directe aansturing te verplaatsen.

Broadcast Met een fysieke NRF24 antenne is het mogelijk om op een kanaal een bericht te versturen naar iedere andere NRF24 antenne binnen bereik. Zo komt er een bericht aan via een zogeheten broadcast. In een simulatie werkt dit niet omdat er niet iets alle kanten opgestuurd kan worden. Daarnaast is het ook zo dat er geen limiet zal zijn in het bereik van een broadcast. Daarom is er besloten een functie toe te voegen aan de draadloze communicatie simulator waarmee opgevraagd kan worden wie er binnen bereik is. Zo kon er toch een implementatie gemaakt worden waar een broadcast uitgevoerd wordt.

```
bool WirelessSignalSimulator::getNodesInRange(
drone_meshnetwork_simulation::AreaScan::Request &req,
drone_meshnetwork_simulation::AreaScan::Response &res )
{
  auto from = Network.find( req.id );
  res.near.clear( ); //Clear the response before filling.
  if ( from != Network.end( ) ) {
    for ( std::pair< uint8_t, Node * > other : Network ) {
      if ( other.first == from->first ) continue; //Not interested in ourself
      if ( !other.second->getOn( ) ) continue; //Ignore nodes that are turned off
      // using pythagoras in the function Vector3 to get the distance between the
      // nodes
      float distance =
        from->second->getPosition( ).Distance( other.second->getPosition( ) );
      if ( distance < maxComDistance ) {
```

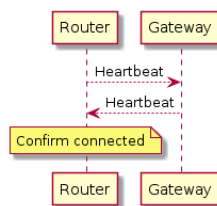
```

    res.near.push_back( ( int )other.first );
  }
}
return true;
} else {return false;}
}

```

Gateway router verbinding Nu er een implementatie was in de simulatie waarmee eigenlijk alle mogelijkheden nagebootst konden worden die een fysieke antenne zou kunnen aanbieden, was het tijd om te beginnen aan het implementeren van de routingstechniek die uit het onderzoek gekozen was.

Een eis was dat het netwerk altijd bestaat uit minimaal één gateway en nul tot 100 routers. Een logische stap was dus het scheiden van de logica van het netwerkcomponent in een gateway en in een router. Een andere eis van het netwerk stelt dat een router altijd direct of indirect verbinding moet hebben met een gateway. De volgende implementatie stap die genomen was, is het registreren in een router of deze daadwerkelijk een verbinding heeft met een gateway. Er is voor gekozen om een pessimistische implementatie te maken waarbij een router alleen mag claimen dat hij een verbinding heeft met een gateway op het moment dat hij een bericht ontvangt van de gateway, die voor hem geadresseerd is. Deze berichten worden onderling uitgewisseld via een heartbeat.



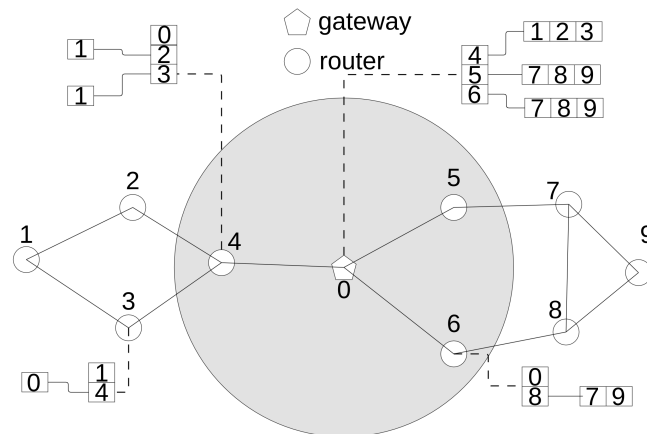
Figuur 6.2: Communicatie voor bevestiging verbinding.

Verbinding router naar een gateway Na het implementeren van een heartbeat voor een enkele router binnen bereik was de volgende stap om een heartbeat te implementeren voor een router buiten het bereik van een gateway. Dit hield eigenlijk in dat er een implementatie moest komen voor het doorsturen van berichten. Er is eerst simpel begonnen met twee routers en een gateway. Het eerste deel van de route was het makkelijkste, de gateway had namelijk altijd hetzelfde adres 0. Daarom kon er vrij snel al een implementatie gemaakt worden. In de applicatie, als de tussen node een bericht ontving voor 0 moet deze doorgegeven worden aan de gateway. De weg terug wordt een stuk complexer voor de gateway. De gateway kan namelijk niet simpelweg het bericht terugsturen want dat is buiten het bereik.

Routingstechniek Er moet dus een routingstechniek gemaakt worden om bij de node te komen. In het geval van de gateway is er de volgende kennis aanwezig:

- Er bestaat een router met ID 2 buiten bereik
- De router met ID 1 die heeft direct of indirect een connectie met die router

Met die kennis is er de theorie dat als het bericht dat naar router ID 2 gestuurd moet worden aan router ID 1 gegeven wordt dat deze dan wel weet hoe het bij router 2 moet aankomen. Deze theorie wordt toegepast in het lightweight mobile routing protocol (S. A. Hamatta, Bokhari & Siddiqui, 2014). Aan het nu abstracte netwerkcomponent is een routingstabel toegevoegd waar in opgeslagen worden welke routers een directe verbinding hebben en bij elke van die routers wordt bijgehouden met welke routers zij weer een verbinding hebben.



Figuur 6.3: Toelichting routingstabel.

In [Figuur 6.3](#) wordt geïllustreerd hoe de routingstabellen van de gateway en verschillende routers er uitzien. Een nadeel van de huidige implementatie is dat er geen rekening wordt gehouden met het kortste pad door het netwerk. Als in [Figuur 6.3](#) bijvoorbeeld de gateway een bericht wil versturen naar router nummer 8 zal dit gebeuren via de weg 0-5-7-8 in plaats van de kortste weg 0-6-8. Dit komt omdat er nu eerst gekeken wordt of het bericht voor de geadresseerde een direct aangesloten punt is. Wanneer dat niet zo is worden de tabellen van de indirecte punten nagelopen. De tabellen zijn gesorteerd op numerieke volgorde. Wanneer het algoritme de tabellen van externe punten nazoeft zal de tabel van punt 5 als eerste een routepunt geven.

Wat opvalt aan de techniek is dat een router altijd op een manier een route weet naar een gateway toe. Dit maakt het dus altijd mogelijk om de gateway als punt aan te houden als er een communicatieweg gevonden moet worden naar een onbekend punt met als regel dat een bericht niet in de richting teruggestuurd mag worden waar die een vorige stap ook was.

Als laatste punt over LMR moet benoemd worden dat het netwerk zich event driven onderhoudt. Dit houdt in dat er alleen informatie wordt rondgestuurd over nieuwe of verbroken verbindingen op het moment dat er een wijziging optreedt in het netwerk. Volgens de LMR techniek wordt een verbroken verbinding pas gecommuniceerd wanneer het vinden van een pad naar een aangesloten buur niet lukt. Omdat de opdracht was dat het netwerk snel moet doorhebben dat een verbinding verloren is, is er toegevoegd dat elk punt periodiek de verbinding van direct aangesloten punten naloopt. Deze routingstechniek is voorzien van unittesten die testen of de routes daadwerkelijk gevonden worden en de tabellen juist worden bijgehouden. Een voorbeeld van een test is hieronder te zien waar een richting van een niet direct aangesloten punt gezocht wordt. Hier zijn twee routes naar punt 5 mogelijk, maar wordt 2 verwacht omdat dit een kleiner getal is dan 3.

```

TEST( TestSuiteHybridLMRoutingProtocol, findRouteToGrandChild )
{
    HybridLMRoutingProtocol CTT;
    CTT.OtherCanCommunicateWithNode( 1, 4 );
    CTT.OtherCanCommunicateWithNode( 2, 5 );
    CTT.OtherCanCommunicateWithNode( 3, 5 );

    EXPECT_TRUE( CTT.getDirectionToNode( 5 ) == 2 );
}

```

Berichten Binnen het netwerk worden compacte berichten gebruikt die een vaste volgorde in een 32 byte array hebben. Deze worden per byte in de array geplaatst en aan de andere kan er weer uitgehaald. De code die gebruikt wordt voor het plaatsen van data in en uit een array zijn twee functies gemaakt die er als volgt uitzien:

```

void Message::CopyToCharArray( uint8_t *value, uint16_t size, uint8_t *arr,
    uint16_t start )
{
    for ( uint16_t i = 0; i < size; i++ ) {
        arr[i + start] = value[i];
    }
}

void Message::CopyFromArray( uint8_t *value, uint16_t size,
    const uint8_t *arr, uint16_t start )
{
    for ( uint16_t i = 0; i < size; i++ ) {
        value[i] = arr[i + start];
    }
}

```

In de eerste functie wordt een waarde per byte gekopieerd naar een array van byte. De tweede functie doet juist het omgekeerde. Een belangrijke voorwaarde is dat de programmeur precies weet hoe groot het datatype is en op welke plek de waarde in een array staat.

Om te zorgen dat deze berichten juist in een array gezet worden, en er ook weer juist uitgehaald worden, is elke type bericht voorzien een meerdere unittesten. Naast de standaard unit testen zijn ook bijzondere locaties getest zoals de locatie van Lelystad airport waar de landingsbaan onder het zeeniveau ligt. Dit te zien in de volgende test:

```

TEST( TestMessages, locationLelystadAirportMessageToPayload )
{
    float lat = 52.455702;
    float lon = 5.519210;
    int16_t hei = -5;
    uint32_t time = 1557435698;

    LocationMessage amsg( 66, 55, 44, 33, lat, lon, hei, time );
    uint8_t buffer[MAX_PAYLOAD];
    amsg.toPayload( buffer );

    LocationMessage msg( buffer );
}

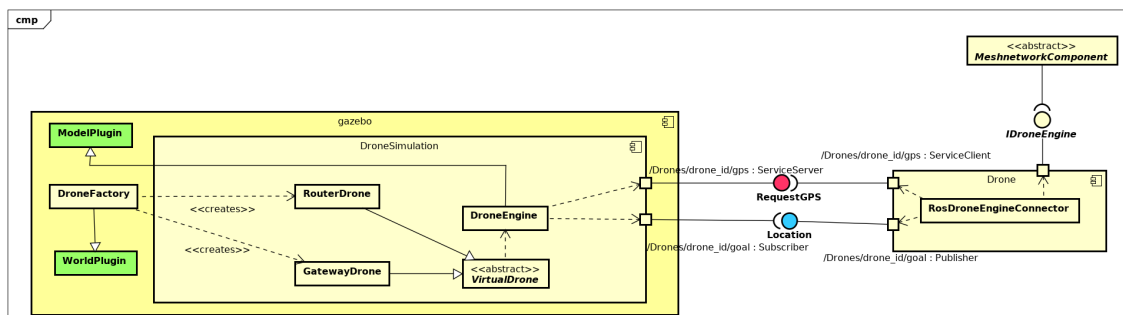
```

```

EXPECT_EQ( LOCATION, msg.getMessageType( ) );
EXPECT_EQ( lat, msg.getLatitude( ) );
EXPECT_EQ( lon, msg.getLongitude( ) );
EXPECT_EQ( hei, msg.getHeight( ) );
EXPECT_EQ( time, msg.getTimeSincePosix( ) );
}

```

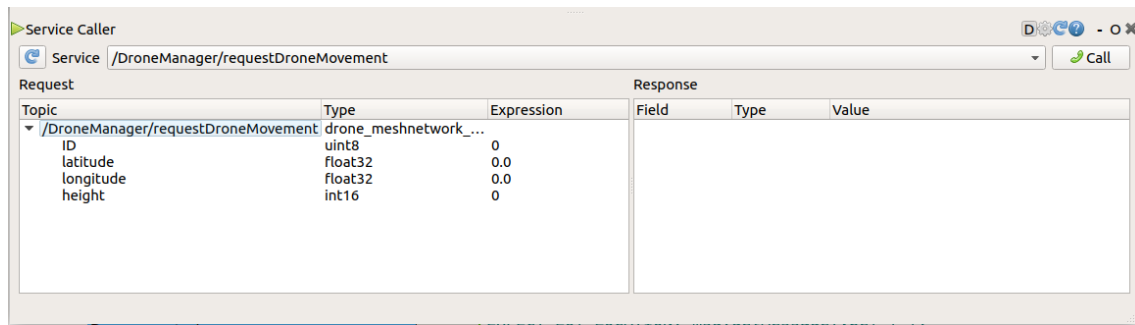
Drone aansturing Nu er communicatie mogelijk was tussen de netwerkcomponenten, en deze in staat zijn correct te kunnen detecteren of er een verbinding naar een gateway mogelijk is, kunnen de componenten aangesloten worden op de interface van de drone voor aansturing. Omdat de abstracte drone niet via een enkele interface aangestuurd kan worden is er gebruik gemaakt van een adapter (Gamma, Helm, Johnson & Vlissides, 1994) tussen het netwerkcomponent en de drone. De adapter is te zien in [Figuur 6.4](#).



Figuur 6.4: Uitwerking adapter tussen de DroneEninge en het MeshnetworkComponent.

Hier is begonnen door met het ontvangen van een bericht op het netwerkcomponent een doel te sturen naar een dronemotor. Vervolgens kon er getest worden of dit werkte door zelf een bericht op te stellen en dit naar de antenne te sturen van het netwerkcomponent. Bij het ontvangen van een juist bericht zal de drone zich verplaatsen naar de locatie die opgegeven is.

Dronemanager Omdat er nog geen manier was om op een simpele manier berichten het netwerk in te kunnen sturen is de dronemanager ontwikkeld. Deze applicatie start een Rossservice die communiceert op een Rostopic waar alle gateways naar luisteren. Omdat er alleen interesse is naar het versturen van locaties naar drones kan ook alleen dit gepubliceerd worden op het topic.



Figuur 6.5: Het versturen van een verplaatsingsverzoek via de dronemanager.

Zodra een gateway een nieuw bericht ontvangt op het topic maakt hij hier een bericht van. Dit bericht wordt het netwerk ingestuurd met een verzoek voor verplaatsing. Op deze manier kon er makkelijker getest worden of de drone aansturing goed werkte en of een punt ver weg in het netwerk in staat was berichten van een gateway te ontvangen.

Protocol bij een verloren verbinding met de gateway De opdracht stelt dat drones zich moeten verplaatsen bij een verloren verbinding naar een alternatieve locatie om zo te proberen de verbinding te herstellen. Hier was de eerste stap om eerst te ontdekken of een netwerkmodule alleen is of dat er nog andere punten in de buurt zijn die ook een verbinding verloren zijn. Hiervoor kon de broadcast functie goed gebruikt worden. De meeste simpele implementatie kon vervolgens gemaakt worden voor de drone die in zijn eentje de verbinding met de gateway verloren is. Na overleg met Alten is besloten dat het voor nu voldoende was als de drone eerst terug vliegt naar het punt dat verloren is gegaan om daar te zoeken naar een nieuwe verbinding met de gateway. Op het moment dat daar nog steeds geen verbinding gevonden is, moet de drone terug keren naar de locatie waar de gateway staat.

Op het moment dat er groep is die verloren gaat moet er een overleg gepleegd worden tussen de drones, het heeft immers geen zin om allemaal naar een zelfde locatie te bewegen of om twee werkende netwerkmodules op een plek te parkeren. Er is nu geïmplementeerd dat elke drone berekend wat zijn afstand is tot de gateway, de drone met de grootste afstand die wil verplaatsen naar een plek waar nog geen andere drone staat zal een beweging uitvoeren.

Een demonstratie hiervan is te zien in de video's van bijlage [F](#).

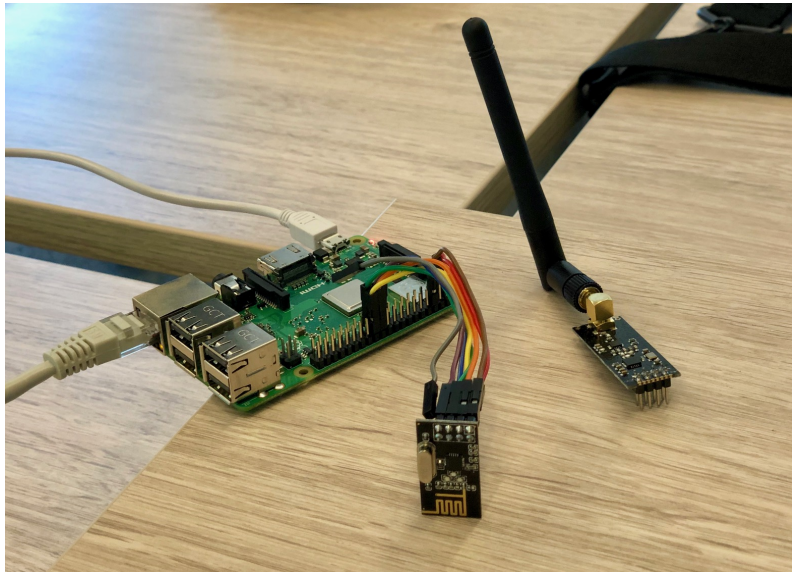
Meerdere gateways in een netwerk Een wens van Alten is de mogelijkheid om meerdere gatewaydrones te gebruiken in een netwerk. Hiervoor is logica toegevoegd aan de routers dat er een voorkeur voor een gateway bestaat. Op het moment dat een router een bericht krijgt van een ander punt in het netwerk die minder hops tot aan een gateway heeft dan neemt de router het id over van de voorkeur gateway van dat punt.

Een video waar de werking van twee gateways in het netwerk gedemonstreerd wordt is te vinden in bijlage [G.1](#)

Fysiek prototype router Na de oplevering van de 80 procent versie is er gewerkt aan de oplevering van een fysieke netwerkmodule. Hiervoor was de makkelijkste stap om de router software draaiende te krijgen op de Raspberry Pi. Dit omdat alleen het virtuele

NRF24 component vervangen hoefde te worden voor een fysieke. In eerste instantie verliep dit goed, vooraf waren er al ontwerpen gemaakt op basis van de beschikbare NRF24 library (nRF24, 2019).

Wat een tegenvaller was, was dat ik vergeten was dat er een Vector3 gebruikt werd uit de library die meegeleverd wordt met Ros. Omdat er in de software alleen de Vector3 een afhankelijk heeft naar de library is er besloten de Vector3 zelf te implementeren in een template class. Na het implementeren van deze template class kon er een router programma gecompileerd worden voor de Raspberry Pi.



Figuur 6.6: Raspberry Pi met de nRF24l01+ antenne voor korte afstand.

Door twee Raspberry Pi's op het netwerk aan te sluiten van Alten (Figuur 6.6) konden ze via een ssh verbinding gestart worden om met elkaar te communiceren wat het eerst contact via de fysieke NRF opleverde te zien in bijlage H.1.

Fysiek prototype gateway Tenslotte was er nog net tijd voor een laatste implementatie die de oplevering compleet zou maken naar wat beloofd is in het begin van het project. Kort overleg gaf de goedkeuring om zonder uitgebreid onderzoek een C++ library te pakken voor de internetgateway. Deze goedkeuring was gebaseerd op het feit dat het toch om een prototype ging en het gewoon heel cool zou zijn om alles op te kunnen leveren. Na kort zoekwerk op het internet kwam Pistache (Stefani, z. j.) naar voren vanwege het gemak voor het opzetten van een REST framework.

Wat een mooi moment was voor mij als programmeur was dat ik de service eerst ontwikkeld heb voor de gesimuleerde gateway. Vervolgens heb ik de Pistache library geïnstalleerd op de Raspberry Pi en kon daar dezelfde software draaien als in de simulatie. Dit bewees voor mij het nut van de simulatie als ontwikkelaar, omdat de simulatie mij dus de mogelijkheid biedt software te ontwikkelen op mijn pc voor mijn pc. Vervolgens kon deze code op de Raspberry Pi worden gezet en daar met gelijk gedrag gedraaid worden.

Een video met audio is te vinden in bijlage H.2 waar de gateway getoond wordt in samen-spel met een router.

7 — Resultaten

In de dit hoofdstuk worden de opgeleverde producten behandeld en beoordeeld op hun kwaliteit.

7.1 Plan van aanpak

Het eerste product die is opgeleverd was het plan van aanpak. Het product kreeg zijn echte waarde na de feedback op het conceptplan.

Het eerste plan was nog niet concreet genoeg in wat opgeleverd zou ging worden. Dit was dan ook in grote lijnen wat de feedback was vanuit de begeleiders van school.

Na het verwerken van de feedback was het plan veel gericht in wat opgeleverd zou moeten worden. Het plan dat er nu ligt zou uitgevoerd kunnen worden door elke ICT'er met de juiste vaardigheden. De kwaliteitseis die ik gesteld heb aan het plan van aanpak was dat het moest omschrijven wat het plan van de student is en dat het voldoet aan het feedbackformulier projectplan van het ICA Praktijkbureau (2017). Deze kwaliteitseisen zijn voldaan wat dus betekend dat het op papier voldoet aan alle kwaliteit.

Wat in mijn ogen nog beter bewijst dat het plan van aanpak een document van kwaliteit is, is dat ik gedurende het project regelmatig het plan erbij heb gepakt om beslissingen op te baseren en kwaliteit van andere producten mee te meten.

7.2 Broncode

De code geschreven voor dit project heeft veel aandacht van mij gekregen. Er heeft veel moeite gezeten in het gebruik van Doxygen om dit volledig compleet te houden. Als resultaat geeft dit dan een compleet resultaat bij het generen van de documentatie.

Daarnaast wat wel is uitgevoerd maar naar mijn mening te weinig is toegepast is het gebruiken van unittesten. Dit is weinig gebruikt omdat veel code een communicatie uitvoert naar een andere netwerkcomponent. Er kunnen wel unittesten voor geschreven worden maar het opzetten van een basis voor zulke testen kost veel tijd. Toch zou ik Alten wel adviseren als ze dit project voortzetten om hier in te investeren. Het schrijven van een routingstechniek is een complex proces die snel problemen kan geven. Het breder toepassen van unittesten zou de kwaliteit van het meshnetwerk zeker ten goede komen.

De componenten zijn op elkaar aangesloten door interfaces aan elkaar aan te bieden. Dit is gedaan om meerdere redenen. Door elkaar interfaces aan te bieden bewaakt een component zijn toegang tot de publieke functies. Daarnaast bieden de interfaces een potentie voor het schrijven van testen omdat deze gemockt kunnen worden.

Wat jammer is maar niet mogelijk was in het project, was dat er geen buildserver beschikbaar was waar een geavanceerde codeanalyse tool aan gekoppeld zat zoals coverity of sonarqube. Wel is er in het project gebruik gemaakt van de statische codechecker CPPcheck. Hoewel het geen maatstaaf is hier nul meldingen op te krijgen in verband met false positives. Toch is elke melding die CPPcheck heeft gegeven afgehandeld. Dit is gedaan door het aanpassen van code of door het bewust onderdrukken van de functie. Zo zijn er op dit moment alleen meldingen onderdrukt over ongebruikte debugging functies of een het gedeelte waar een float wordt opgeknipt in bytes.

7.3 Onderzoeksrapport

Het onderzoeksverslag is een verslag die naar mijn mening inhoudelijk niet voldoende gespreide aandacht heeft gekregen. Hoewel het misschien de onderzoeksvraag beantwoord is er duidelijk te merken dat het literatuur onderzoek naar de te gebruiken hardware en simulatiesoftware veel meer tijd heeft gekregen dan de rest van het document.

De hoofdvraag gesteld in het onderzoek is relevant omdat het helpt met het uitvoeren met de specifieke opdracht. De deelvragen daarentegen zijn niet allemaal even sterk uitgewerkt. Zo is er de vraag wat nodig is om een abstracte drone te representeren. Hier had ik veel onderzoek naar kunnen doen maar dit heeft geen meerwaarde. Een kort overleg voldoet in dit onderwerp ook waarin besproken is van wat moet de gesimuleerde drone allemaal kunnen.

Op zich is dit gezien de beschikbare tijd een prima keuze maar het haalt de kwaliteit van het onderzoek wel naar beneden.

Wat goed is gedaan in het onderzoek is het gebruik van bronnen, zo zijn ze voornamelijk gebaseerd op wetenschappelijke artikelen en zijn ze ruim aanwezig.

Verder merkte ik dat het fijn is voor het bedrijf dat de onderzoeksmethode geleerd op de HAN nog toegelicht is aangezien niet iedereen hiermee bekend is. Ook het toelichten hoe onderzoek op de HAN werkt is goed onderbouwd en toegelicht.

7.4 Simulatiesoftware

Tegen het einde van het project werd de simulatiesoftware opgeleverd. Dit was het product waar de meeste focus en ontwikkeling in heeft gezeten. Dit was initieel niet het plan omdat er meer focus op het hardware prototype zou liggen. Gedurende het project werd de urgentie van de simulatiesoftware steeds duidelijker.

Door het gebrek aan voldoende hardware kon er niet goed getest worden hoe het netwerk zich zou gedragen bij een hoog aantal nodes. Daarom is er gekozen om de simulatie een hogere prioriteit te geven zodra de gesimuleerde communicatie realistisch genoeg was.

Een tegenvaller van de simulatie in Gazebo was dat voor het simuleren van 100 drones en dit visueel te maken een computer met een dedicated GPU nodig is. Dit was vooraf niet verwacht omdat de drones in de simulatie alleen maar vliegende blokjes zijn.

Het opgeleverde product voldoet aan de wens van Alten. Alten wil namelijk verdeelalgoritmes kunnen testen waarbij de drones op een realistische wijze worden aangestuurd, en dat is opgeleverd. Verder is de simulatie voorzien van voldoende documentatie.

7.5 Hardwareprototype

Het huidige hardware prototype staat nog echt in de kinderschoenen. Hoewel het routingsprotocol die gebruikt wordt in het prototype zichzelf bewezen heeft in de simulatie is deze summier getest in de fysieke wereld. Daarom kan er onvoldoende garantie gegeven worden op het gedrag van het prototype. Dankzij de opzet van de code kan er wel eerst getest worden in de simulatie waarna het vervolgens mogelijk is om de code direct te draaien op het prototype. Dit zal de doorontwikkeling aanzienlijk versnellen voor het prototype.

7.6 Software requirements specificatie

Voordat er een goed design kan komen is het zaak dat er requirements opgesteld worden. Dit heb is gedaan door het opstellen een software requirements specificatie. Deze heb is opgesteld aan de hand van gesprekken met de opdrachtgever Alten. In deze gesprekken zijn de gesprekstechnieken ANNA (altijd navragen nooit aannemen) en LSD (luisteren samenvatten doorvragen) toegepast zoals dit geleerd is in de professional skills lessen. Vervolgens zijn de requirements opgedeeld in functionele requirements en niet functionele requirements. De functionele requirements zijn door de MoSCoW methode toe te passen geprioriteerd. De niet functionele requirements zijn vervolgens verdeeld in categorieën uit FURPS.

Hoewel de SRS in zijn eerste oplevering ver ondermaats was, is hij in zijn huidige staat van voldoende kwaliteit. Het domeinmodel is opnieuw opgezet zodat de niet te technisch is. Het usecase diagram is nu geplaatst binnen een systeem. De usecases zijn volledig te parafraseren met het domeinmodel. Een extra kop is toegevoegd om de requirements toe te lichten. Tenslotte zijn de usecases herschreven zodat deze alleen het blackbox gedrag omschrijft.

Dit alles heeft er voor gezorgd dat het SRS voldoet aan de eisen van de HAN en de kwaliteit gewaarborgd wordt.

7.7 Software design document

Het software design document is zo compleet mogelijk opgeleverd. Er gekozen om niet per subcomponent het design te behandelen maar dit te doen per package. Deze keuze is gemaakt omdat de componenten die in elke package zitten niet zonder elkaar behandeld kunnen worden. Zo is bijvoorbeeld de virtuele NRF24 een component die alleen communiceert via het component die draadloze communicatie afhandelt. Dit komt door het gebruik van Ros. Omdat er door het toepassen van Ros een aantal subcomponenten van het systeem als aparte applicatie draaien.

Verder als het design document tegen de kwaliteitseisen die in het plan van aanpak zijn gesteld wordt gehouden dan voldoet deze aan de gestelde eisen. Het document komt overeen met de code en alle relevante ontwerpen zijn ruimschoots vastgelegd. Wat afwijkt in het proces ten opzicht van het plan is de stap waar het domeinmodel wordt omgezet tot class diagrammen. Er is gekozen om gebruik te maken component diagrammen die de samenhang van de software overzichtelijker maken dan een class diagram.

8 — Evaluatie

In dit hoofdstuk worden situaties behandeld die zich tijdens de afstuderen hebben voorgedaan. Hiervoor wordt de STARR (Situatie, Taak, Actie, Resultaat, Reflectie) methode toegepast.

8.1 80% versie

Het is vaak verteld door de professional skills leraren en toch heb ik in dit project de fout gemaakt om iets aan te nemen. Deze situatie beschrijft hoe dit project flink is beïnvloed omdat ik een aanname had gedaan ten opzichte van de 80% versie.

Tijdens de eerste fase van het project moet elke student een plan van aanpak schrijven. Hierin is een belangrijk stuk de planning. Tijdens het plannen van het project heb ik de fout gemaakt om zo te plannen dat er bij een 80% versie het project ook 80% voltooid zou zijn. Dit leek ook allemaal logisch tot er de evaluatie kwam van een plan onder begeleiding van Jorg Visch. Pas toen kwam ik erachter dat het beter zou zijn voor de kans van slagen dat het project tijdens de 80% versie tenminste alle elementen al zou bevatten om te kunnen slagen. Op dat moment was de planning hier nog niet naar terwijl dit plan al aardig strak gepland was.

Eerst had ik nog idee dat als ik iets harder zou werken en voor zou gaan lopen op de planning dat ik nog steeds een 80% versie zou kunnen opleveren die zou voldoen aan de wensen vanuit school. Na drie weken in het project werd het duidelijk dat dit niet genoeg zou zijn. Er moest een geschoven worden binnen de planning om het toch allemaal op orde te krijgen.

Tijdens het project was er ook een terugkomdag op school, hier sprak ik met andere studenten onder andere over het 80% inlevermoment. Zij vertelde mij dat ze al wel vroeg tijdens het schrijven van het plan van aanpak wisten dat ze zo moesten plannen dat er bij 80% een project moest staan die waardig is voor slagen.

Hieruit leerde ik dat hoe logisch iets ook klinkt dat ik nooit is zomaar moet aannemen. Het effect van de verkeerde aanname is groot geweest op mijn project terwijl die eigenlijk makkelijk voorkomen had kunnen worden. Ik had in plaats van de aanname een mail naar mijn schoolbegeleider moeten sturen met de vraag naar wat hij verwacht van de inlevermomenten.

8.2 Afstudeereiland

Deze situatie kaart ik aan omdat ik het afstuderen als eenzaam heb ervaren. Natuurlijk zijn er de begeleiders en collega's waaronder ook oud HAN studenten aanwezig die zorgen

voor de nodige gezelligheid en zijn ze allemaal bereid om mij te helpen. Daarnaast is het afstudeerproject ook een proeve die een student alleen moet doorlopen. Toch heb ik mij op een aantal momenten eenzaam gevoeld gedurende het project vandaar dat ik het afstudeereiland noem.

Dit eiland komt vooral omdat de afstudeeropdracht geen directe impact heeft op de productontwikkeling binnen het bedrijf. Hierdoor voelen collega's zich ook niet verplicht om zich te verdiepen in het werk wat ik uitvoer.

Als student van de HAN informatica is dit voor mij voor het eerst dat ik een project alleen uitvoer. Zo werden alle voorgaande projecten met ongeveer 5 mensen uitgevoerd met zelfs een uitzondering in het World of Robots project die met 17 personen werd gedaan. Een project van 20 weken alleen is dus even wennen. Het heeft mij de waarde van samenwerken aan een project veel meer leren waarderen dan ik al deed.

Het werken op een eiland brengt meerdere nadelen met zich mee. Het eerste nadeel is dat het alleen werken ten kostte gaat van de kwaliteit van het product. Omdat er geen dagelijks contact is zoals dat bijvoorbeeld is in een daily stand up loop je een groot risico. Wanneer een project een verkeerde richting in gaat wordt dit pas laat onderkent. Hiernaast zijn er ook geen hoog frequente checks op de technieken toegepast binnen de producten.

Een ander nadeel dat kan optreden maar door Alten is voorkomen is dat ik makkelijk buiten de organisatie kon gaan vallen. Alten heeft dit voorkomen door mij overal actief bij te betrekken en mij dezelfde toegang te geven tot activiteiten die normale collega's ook hebben. Zo heb ik presentaties bijgewoond waar bijvoorbeeld een kijkje in de softwarekeuken werd gegeven van de gigant [takeaway.com](https://www.takeaway.com) ([thuisbezorgd.nl](https://www.thuisbezorgd.nl)), maar heb ik ook met het hele team op de schaatsbaan in Enschede gestaan om te curlen. Dit was allemaal optioneel en kon ik mij zelf voor aanmelden.

Een les die ik hier uit leer is dat ik meer mijn best had moeten doen om collega's inhoudelijk te betrekken bij mijn afstuderen. Dat het afstuderen alleen uitgevoerd moet worden is een feit, maar dit betekend nog niet dat het eenzaam hoeft te zijn.

Ik had mijn collega's meer kunnen betrekken door presentaties te geven over inhoudelijke keuzes in het project. Dit zou hun de kans geven om gericht feedback te geven over mijn werk. Terugkijkend ben ik hier zelf niet proactief genoeg in geweest.

Als ik terugkijk naar wat ik nog meer had kunnen doen denk ik dat alleen maar zou kunnen adviseren aan toekomstige afstudeerders om te gaan werken aan afstudeeropdrachten die invloed hebben op het product van het bedrijf. Dit forceert het bedrijf om zich actief met de student bezig te houden omdat de impact van het resultaat groter is.

8.3 Presentatie oefening voor verdediging

Om te oefenen voor de verdediging en mijn collega's inzicht te geven in mijn product heb ik een presentatie gehouden zoals ik dat zou doen tijdens mijn verdediging. De presentatie is gehouden voor een publiek van negen collega's. Om op de presentatie feedback te krijgen heb ik gebruik gemaakt van een feedbackformulier waar de focus lag op mijn presentatie vaardigheden. Er was ruimte op het formulier om schriftelijk feedback te geven, en er kon met -, + of ++ een waardering gegeven worden. Aan het einde van de presentatie konden er vakinhoudelijke vragen gesteld worden zoals dat bij de echte verdediging ook zou zijn.

8.3.1 Feedback op de presentatie

De feedback is opgedeeld in structuur, inhoud, non-verbaal, spreekstijl, visueel en contact met publiek. Elk stuk wordt hieronder apart behandeld.

Structuur (-[0], +[4], ++[4], leeg[1])

In de feedback wordt aangegeven dat de opbouw goed is. Een terugkerend punt is dat er niet genoeg aandacht wordt gegeven aan de probleemstelling. Verder is er niet ingegaan op de structuur.

Inhoud (-[0], +[3], ++[4], leeg[2])

De feedback over de inhoud was positief. Het publiek was tevreden over mijn kennis en het verhaal werd duidelijk verteld. Ze vonden de reflectie over het project een leuke toevoeging. Punten uit de feedback zijn dat ik niet moet twijfelen over wat ik moet uitleggen. Ik bepaal van te voren wat ik wil toelichten. Tijdens de presentatie gebruik ik het woord 'we' op het moment dat ik beslissingen toelicht. Als feedback wordt aangegeven dat ik niet 'we' maar 'ik' moet gebruiken, het is wellicht met Alten overlegd maar het is mijn project.

Non-verbaal (-[0], +[6], ++[1], leeg[2])

Hierover waren de feedbackgevers positief. Er wordt aangegeven dat ik autoriteit uitstraal, zeker overkom en een rustige sfeer heb. Leerpunten die gegeven worden zijn dat ik moet letten waar ik naartoe kijk dus niet te veel naar hetzelfde persoon of het bord.

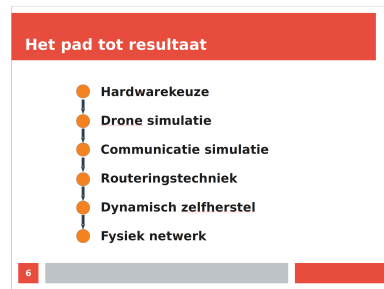
Spreekstijl (-[0], +[3], ++[6], leeg[0])

Over de spreekstijl waren mijn collega's heel enthousiast. De rust die ik tijdens de presentatie bewaarde waren ze niet gewend van stagiaires. Ik praatte duidelijk en articuleerde goed. Wel moest ik oppassen dat de rust die ik had niet ten kostte zou gaan van mijn enthousiasme.

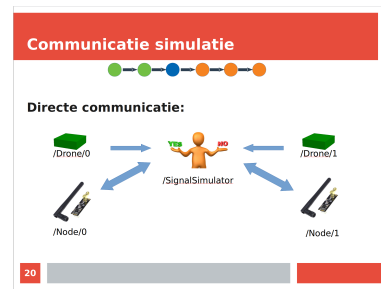
Visueel (-[1], +[6], ++[0], leeg[2])

Het visuele gedeelte was een minder sterk stuk de presentatie. Zo waren bepaalde figuren zoals het domeinmodel onleesbaar voor het publiek. De demo filmpjes hadden last van het zonlicht waardoor het niet goed zichtbaar was voor iedereen. Een collega vond ook dat ik de presentatie te veel als leidraad gebruikte en heeft daarom een - gegeven.

Waar de collega's positief over waren was een zelfbedachte techniek die ik gebruikt heb in mijn presentatie. In mijn presentatie maak ik gebruik van een pad die continue zichtbaar is in de slides. Door middel van kleur gebruik maak ik duidelijk waar ik ben op het pad zoals zichtbaar in [Figuur 8.1](#)



(a) Slide waar het pad geïntroduceerd wordt.



(b) Slide waar de status van het pad bovenin zichtbaar is.

Figuur 8.1: Slides uit de presentatie van de verdediging.

Het continue tonen van het pad bovenin de presentatie heb ik bedacht omdat ik het publiek wil meenemen op het pad die ik de afgelopen maanden heb bewandeld. Een voordeel hieraan is dat het duidelijk is voor zowel mijzelf als het publiek waar we zijn in de presentatie.

Contact met publiek (-[0], +[0], ++[7], leeg[2])

Het contact met het publiek was volgens iedereen goed in orde. Dit komt volgens hun omdat ik veel kennis had om goede inhoudelijke antwoorden te kunnen geven. Een tip die ik meekreeg was om bepaalde vragen voor te bereiden door ze in slides achter de presentaties te stoppen.

8.3.2 Conclusie

Op basis van de feedback kan ik concluderen dat ik het houden van de verdediging beheers. Het is natuurlijk afwachten hoe het straks zal gaan tijdens de echte verdediging. Veel kracht van de presentatie kwam uit de rust die ik kon bewaren. De feedback geeft de laatste tips om de presentatie af te maken, vooral de tip om slides voor mogelijke vragen toe te voegen vond ik handig.

9 — Conclusie

Het project heeft een geavanceerde simulator opgeleverd voor het testen van onderling communicerende drones. Daarmee is het doel van het project behaald omdat Alten wil kunnen experimenteren in het opzetten van grootschalige netwerken door het gebruik van drones. In de opgeleverde simulatie kunnen ze algoritmes testen voor verdelingen en beoordelen waar routers het meest belast zouden worden.

Naast de simulatie is er ook een prototype van het netwerkcomponent opgeleverd. De simulatie is gebouwd op basis van de eisen die de gekozen hardware opgelegd aan de software.

Door het gebruik van een interfaces kan er gemakkelijk geëxperimenteerd worden met de netwerkmodule. In theorie kan er zelf getest worden met een fysieke antenne en een gesimuleerde drone zolang de gebruikte interfaces maar kloppen.

Het doel van Alten is om onderling verbonden drones te kunnen verdelen om zo een netwerk op te kunnen bouwen over een gebied. Van te voren was al bekend dat de resultaten van dit project een eerste stap zijn naar het behalen van dit doel. Het behaalde resultaat om realistisch kunnen testen in een simulatie is het resultaat die Alten nodig had voor deze eerste stap. Deze stap is gezet samen met een oplevering van een hardware prototype die zich gelijk gedraagt als de simulatie. Het is nu aan Alten om deze twee producten door te ontwikkelen om zo haar doel te kunnen behalen.

Het doel van de student om met dit project de vijf beroepscompetenties aan te tonen zijn ook behaald. Het vaststellen van de eisen aan het product in de [Software requirements specificatie](#) maar ook op andere plekken toont aan dat de student voldoet aan beoordelingscriteria 1. Het [Onderzoeksrapport](#) maar ook de keuzes vastgelegd in het [Software design document](#) laten zien dat de student voldoet aan beoordelingscriteria 2. Het [Software design document](#) samen met de opgeleverde [Broncode](#) laat zien dat de student bekwaam is in beoordelingscriteria 3. Het schrijven van een kwalitatief [Plan van aanpak](#) en het effectief kunnen bijsturen van het proces zoals omschreven in [Uitvoer projectplan](#) laat zien dat de student bekwaam is in beoordelingscriteria 4. Dit gehele verslag laat zien dat de studenten bekwaam is in het verantwoorden en reflecteren van zijn werk waarmee hij voldoet aan de laatste beoordelingscriteria 5.

9.1 Advies vervolg project

Wanneer dit project vervolgt wordt het volgende geadviseerd aan Alten:

- Voorzie een volgende ontwikkelaar van een computer die een dedicated GPU aanwezig heeft zodat er grotere simulaties met Gazebo gedraaid kunnen worden.

- Stel een buildserver voorzien van een geavanceerde code checker(Bijv. SonarQube of Coverity) beschikbaar.
- Experimenteer met andere routingsalgoritmes voordat er met fysieke netwerkapparatuur gewerkt gaat worden. Er bestaat niet één oplossing voor een dynamisch netwerk. Wellicht kan er geschakeld worden van protocol zodra de netwerkpunten statisch worden in de wereld.
- Geef een volgende ontwikkelaar de ruimte om een geavanceerdere test omgeving op te zetten voor geautomatiseerde testen. Er kunnen scripts gemaakt worden om scenario's te testen in de verdeling van drones. Het gebruik van interfaces maakt het mogelijk de componenten los te testen door het gebruik van mocking.
- De draadloze communicatie is nu gebaseerd op onderlinge afstand tussen nodes. Het advies luidt om te onderzoeken of het haalbaar is hier een andere techniek voor te gebruiken die bijvoorbeeld ook rekening kan houden met obstakels zodat er simulaties kunnen draaien in omgevingen met bijvoorbeeld bomen of gebouwen. Hier moet rekening gehouden worden met het feit dat deze calculatie ook moet werken voor honderd drones.
- Breidt de simulatie verder uit op basis van het gedrag van het fysieke netwerkcomponent die is opgeleverd. Focus hier op de communicatie van berichten, en hoe snel de netwerkmodule moet concluderen dat er geen verbinding mogelijk is.
- Schrijf de drone engine om naar het software patroon van een state machine en voeg een functie toe in de aangeboden interface die de staat van de drone aangeeft. Zo kan de netwerkmodule nog betere beslissingen nemen of bijvoorbeeld pas timers starten na het landen van de drone.
- Voordat de overstap gemaakt wordt van een abstracte drone naar een volledig gesimuleerde drone in de simulatie adviseer ik een uitbreiding tot een gesimuleerde drone die een batterijduur heeft.

Literatuur

- CMake. (z. j.). *Cmake*. Op 8 juni 2019 verkregen van <https://cmake.org/>
- Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. In (p. 139). Boston, Massachusetts: Addison Wesley.
- ICA Praktijkbureau. (2017, 6 juli). *Feedbackformulier projectplan afstuderen 2017-2018 profielenmodel*. Op 4 februari 2019 verkregen van <https://onderwijsonline.han.nl/elearning/lesson/1NXw0pzq>
- LaTeX3 Team. (z. j.). *Latex - a document preparation system*. Op 8 juni 2019 verkregen van <https://www.latex-project.org>
- nRF24. (2019). *nrf24/rf24: Optimized fork of nrf24l01 for arduino & raspberry pi/linux devices*. Op 10 juni 2019 verkregen van <https://github.com/nRF24/RF24>
- Open Source Robotics Foundation. (2014). *Ignition: ignition::math::vector3 <t>class template reference*. Op 7 juni 2019 verkregen van https://osrf-distributions.s3.amazonaws.com/ign-math/api/1.0.0/classignition_1_1math_1_1Vector3.html#aac4d64d44e9b7b3b810a2cdf530ce8d3
- Open Source Robotics Foundation. (2019). *Sdf specification*. Op 9 juni 2019 verkregen van <http://sdformat.org/spec>
- S. A. Hamatta, H., Bokhari, M. & Siddiqui, S. (2014, oktober). Protocols in mobile ad-hoc networks: A review. *International Journal of Applied Information Systems (IJ AIS)*, 7, 11-14. doi: 10.5120/ijais14-451236
- Stefani, M. (z. j.). *Pistache*. Op 10 juni 2019 verkregen van <http://pistache.io/>

A — Documenten

Ter verantwoording van dit afstudeerverslag zijn de gemaakte documenten meegeleverd.

A.1 Onderzoeksrapport drone meshnetwerksimulatie.pdf

Het onderzoeksrapport.

A.2 Plan van aanpak.pdf

Het plan van aanpak.

A.3 SoftwareDesignDocument.pdf

Het software design document.

A.4 SoftwareRequirementSpecification.pdf

Het software requirement specificatie.

B — **Broncode**

Ter verantwoording van dit afstudeerverslag is de broncode opgeleverd en te vinden in:
`../Code/drone_meshnetwork_simulation/src/`.

C — Resultaat CppCheck

Project Settings

Project:

Paths:

Include paths:

Defines:

Previous Scan

Path selected: /drone_meshnetwork_simulation/src

Number of files scanned: 31

Scan duration: 24 seconds

Statistics

Errors: 0

Warnings: 0

Style warnings: 0

Portability warnings: 0

Performance warnings: 0

Information messages: 0

Het statistics verslag is te vinden in ./Bijlagen/cppcheck-gui.pdf

D — Resultaten GoogleTest

In deze bijlage zijn de resultaten van de Google Test toegevoegd. Deze zijn te vinden in `./Bijlagen/test_results/drone_meshnetwork_simulation/`.

D.1 `gtest-drone_meshnetwork_simulation-GatewayDrone-test.xml`

Bevat de testresultaten van het coderen en decoderen van de berichten die het netwerk-component gebruikt.

D.2 `gtest-drone_meshnetwork_simulation-HybridLMRoutingProtocol-test.xml`

Bevat de testresultaten van het routeringsprotocol.

D.3 `gtest-drone_meshnetwork_simulation-Vector3-test.xml`

Bevat de testresultaten van de Vector3 template class.

E — **Templates**

Bevat template documenten te vinden in ./Templates/

E.1 **IterationAssessment.pdf**

E.2 **Iteratieplan.pdf**

E.3 **Usecase.pdf**

F — Videos simulatie netwerkherstel door drone verplaatsing

De video's van deze bijlagen zijn te vinden door het volgende pad te volgen:
./Bijlagen/Videos simulatie netwerkherstel door drone verplaatsing/

F.1 enkele verloren drone situatie 1.mp4

Video van een enkele drone die verbinding verliest en dit met een enkele stap weer opbouwt.

F.2 enkele verloren drone situatie 2.mp4

Video van een enkele drone die verbinding verliest en nadat een enkele stap niet werkt weer terugkeert naar de gateway.

F.3 groep verloren drones situatie 1.mp4

Video van een groep verloren drones waarbij het terugzetten van een enkele stap van een drone verbinding weer herstelt.

F.4 groep verloren drones situatie 2.mp4

Video van een groep verloren drones waarbij twee drones een verplaatsing moeten uitvoeren voor het herstellen van het netwerk.

F.5 groep verloren drones situatie 3.mp4

Video waarbij alle werkende drones terug keren naar de gateway omdat er geen herstel wordt gevonden.

G — **Videos drone simulatie**

Video's in ./Bijlagen/

G.1 GatewayWissel.mp4

Video van drones die zich verspreiden, hier zijn twee gateways bij aanwezig. Vervolgens wordt een essentieel punt uit gezet. De daarmee verbonden router wisselt van gateway.

H — **Video fysiek hardware prototype**

Video's van het fysieke component. Te vinden in ./Bijlagen/

H.1 eerstecontact.mp4

Video van fysieke routers die met elkaar communiceren.

H.2 Fysieke router en drone.mp4

Video van fysieke router en gateway die verbinden met elkaar