

Lectoraat Mechatronica, Saxion

Afstudeerproject ROS 2 NeNa Navigation & Security

Afstudeerverslag

Tom Kleine
16-1-2022

ROS 2 NeNa Navigation & Security

Auteur: Tom Kleine

Studentnummer: 448153

E-mailadres: 448153@student.saxion.nl

Opleiding: HBO-ICT SE

Onderwijsinstelling: Saxion Enschede, academie ACT

Bedrijf/organisatie: Lectoraat Mechatronica, Saxion

Bedrijfsbegeleider: Wilco Bonestroo

Afstudeerbegeleider: Wouter van Kleunen

Afstudeerperiode: September 2021 tot en met februari 2022

Voorwoord

Dit afstudeerverslag beschrijft het afstudeerproject dat bij het lectoraat Mechatronica is afgerond. Dit afstudeerproject is het tweede project dat ik bij het lectoraat Mechatronica heb uitgevoerd. Het eerste project is na het onderzoek helaas gestopt omdat duidelijk werd dat de competenties niet gerealiseerd konden worden. Dit project richtte zich volledig op de beveiliging van robots die middels ROS 2 ontwikkeld worden. In het huidige project is hier een aspect van overgenomen. De focus van de opdracht is echter volledig verschoven naar het uitbreiden en afronden van bepaalde functionaliteit van een robot die het lectoraat recent heeft ontwikkeld.

Voorafgaand wil ik graag Wilco Bonestroo bedanken voor de begeleiding vanuit het lectoraat Mechatronica tijdens dit afstudeerproject. De feedback en het sparren tijdens de wekelijkse meetings heb ik als zeer prettig ervaren. De getoonde betrokkenheid liet zien dat er baat was bij een goede afronding van het afstudeerproject.

Ten slotte wil ik Wouter van Kleunen bedanken voor de begeleiding vanuit Saxion. De kritische blik, feedback en het bijsturen om deze scriptie leesbaar en begrijpelijk op te stellen, heeft geresulteerd in een kwalitatief betere afronding van het afstudeerproject.

Versiebeheer

Versie	Wijzigingen/opmerkingen
1.0	Opzet structuur
1.1	Concept inhoud
1.2	Feedback verwerkt Conceptversie
1.3	Definitieve versie

Samenvatting

Een van de robots die het lectoraat Mechatronica recent heeft ontwikkeld, is de NeNa robot, zie afbeelding 1. De NeNa robot is een prototype dat is ontwikkeld tijdens het NeNa project. In dit project werd onderzocht hoe een nauwkeurige, robuuste, betrouwbare en veilige industriële grondrobot ontwikkeld kon worden middels open-source oplossingen. Het doel van de NeNa robot is om autonoom te navigeren en kratten op te pakken.



Afbeelding 1, NeNa robot [1]

Het autonoom navigeren naar een bepaald doel, zoals een oplaadstation, is een steeds vaker voorkomend vereiste voor robots. Dit concept wordt ook wel docking genoemd en is ook van toepassing op de NeNa robot met betrekking tot het oppakken van kratten. De huidige NeNa robot is niet in staat dit consistent te verrichten. Zodoende is er bij het lectoraat nog baat bij deze implementatie die tevens ook in andere toekomstige projecten hergebruikt kan worden.

Ook het belang van security dringt ook steeds verder door in de robotica. Zo zijn robots in productiecellen een interessant doelwit voor ransomware aanvallen. Individuele robots kunnen ook als doelwit genomen worden in gerichte aanvallen om schade aan te richten. Verder blijft er een inherent risico met alle apparaten die met het internet verbonden zijn, zo ook robots. Er is bij het lectoraat Mechatronica interesse naar de implementatie van beveiliging in de robots die zij ontwikkelen.

Tijdens het afstudeerproject heeft de afstudeerder onderzoek gedaan naar mogelijkheden om kratten te detecteren en navigatieoplossingen die passend zijn binnen de huidige NeNa robot. Daarnaast is er onderzocht hoe NeNa en soortgelijke robots te beveiligen zijn. Deze onderzoeksvraag is overgehouden van de eerste afstudeeropdracht. De hoofdvraag voor het onderzoek is als volgt: “Hoe kan de NeNa robot consistent zijn doel detecteren, er overheen rijden en beveiligd worden?”.

Op basis van het onderzoek is gebleken hoe de detectiefunctie het beste geïmplementeerd kan worden. Hiervoor worden de karakteristieken van het krat en goniometrie gebruikt om de locatie ervan te berekenen. Uit het onderzoek met betrekking tot de navigatiefunctie is de keuze gevallen op het gebruikmaken van de Nav2 library die reeds in de NeNa robot gebruikt wordt.

Het eindproduct van het afstudeerproject is de NeNa robot, voorzien van deze nieuwe functionaliteit. De NeNa robot is in staat om consistent en autonoom zijn doel te detecteren en hier naartoe en overheen te navigeren. De ontwikkelde functionaliteit is door het lectoraat Mechatronica in volgende projecten te hergebruiken.

Inhoud

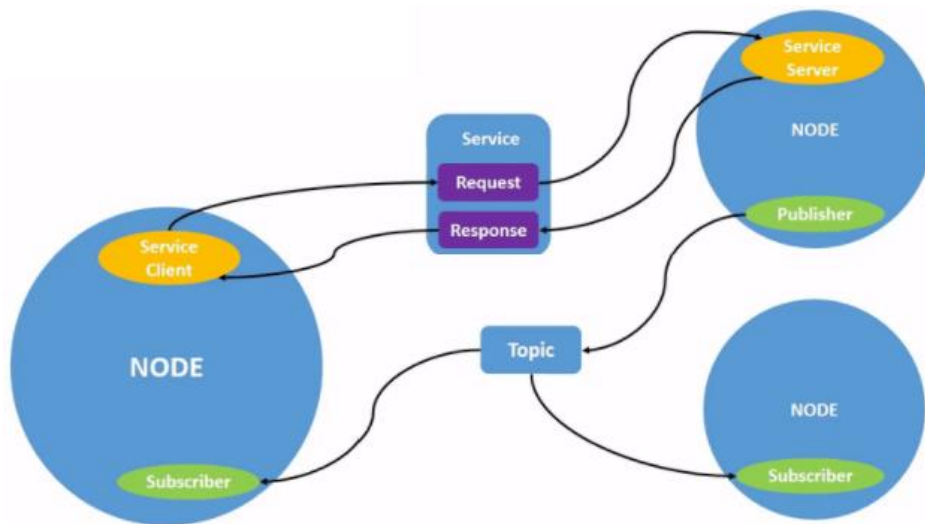
ROS 2 NeNa Navigation & Security	1
Voorwoord	2
Versiebeheer	3
Samenvatting.....	4
1. Inleiding	7
2. Opdracht.....	10
2.1 Probleemstelling.....	10
2.2 Doelstelling.....	10
2.2.1 Onderzoeksvragen.....	11
2.3 Projectgrenzen	11
2.4 Beroepsproducten.....	12
2.4.1 Onderzoeksrapport	12
2.4.2 Ontwerp.....	12
2.4.3 Prototype.....	12
2.4.4 Adviesrapport	12
3. Projectaanpak.....	13
3.1 Projectmanagementmethode	13
3.2 Ontwikkeling.....	13
3.3 Fases	13
3.3.1 Voorbereidingsfase.....	13
3.3.2 Onderzoeksfase	13
3.3.3 Ontwikkelfase	14
3.3.4 Afrondingsfase.....	14
3.3.5 Planning.....	14
4. Onderzoek	15
4.1 Onderzoeksvragen.....	15
4.1.1 Wat zijn bestaande methoden om de positie en oriëntatie van kratten te detecteren op basis van LiDAR data?.....	15
4.1.2 Hoe kan de NeNa robot correct over het krat heen rijden om deze op te pakken?.....	20
4.1.3 Hoe kunnen de ROS 2 beveiligingsopties toegepast worden?.....	23
4.1.4 Hoe kan bovenstaande kennis samengevoegd worden tot een demonstreerbaar prototype?.....	26
4.2 Conclusie: Hoe kan de NeNa robot consistent zijn doel detecteren, er overheen rijden en beveiligd worden?	27
5. Ontwerp & realisatie	28

5.1	Functioneel ontwerp	28
5.2	Technisch ontwerp	30
5.2.1	Functionaliteit	30
5.2.2	Requirements	33
5.2.3	User stories	33
5.3	Realisatie	34
5.3.1	Detectiefunctieiteit	34
5.3.2	Navigatiefunctieiteit	36
5.3.3	Adviesrapport security	36
5.4	Testplan	37
6.	Reflectie.....	39
6.1	Thuiswerken	39
6.2	Onderzoek	39
6.3	Ontwerp.....	39
6.4	Implementatie	40
7.	Adviesrapport security	41
8.	Conclusie	42
9.	Bronnen	43
10.	Bijlagen	45

1. Inleiding

Dit afstudeerproject is uitgevoerd bij het lectoraat Mechatronica, onder begeleiding van W. Bonestroo als bedrijfsbegeleider vanuit het lectoraat en W. van Kleunen vanuit Saxion als afstudeerbegeleider.

De huidige NeNa robot is ontwikkeld met ROS 2 (Robot Operating System) [2]. ROS is een framework waarmee relatief eenvoudig robotische prototypes en eindproducten gerealiseerd kunnen worden. Een ROS project bestaat uit een verzameling nodes. Een node is een klein programma dat verantwoordelijk is voor een stuk functionaliteit. Nodes kunnen informatie uitwisselen met andere nodes middels twee communicatiemodellen, publish-subscribe en request-reply. Het publish-subscribe model kan worden gebruikt om bijvoorbeeld sensordata te delen. In dit model zijn er publishers die informatie sturen en subscribers die de informatie ontvangen. Het kanaal waarover deze informatie wordt verstuurd wordt geïdentificeerd door een topic, een unieke naam. In het request-reply model zijn er servers en clients. Een client kan een request sturen naar een server om een bepaalde handeling uit te voeren. De server stuurt het resultaat terug in een reply. In afbeelding 2 is een overzicht te zien van het publish-subscribe en request-reply model.



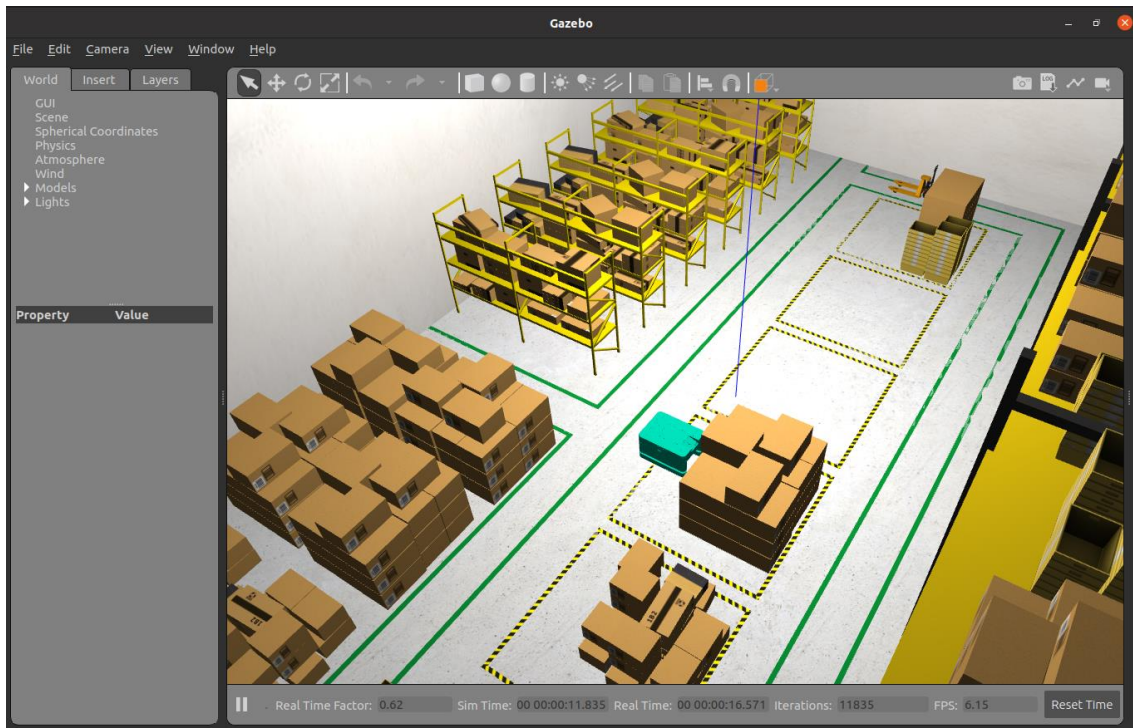
Afbeelding 2, bron: [3]

De functionaliteit in een ROS project is opgedeeld in verschillende packages. Elke package is verantwoordelijk voor een deel van de functionaliteit. In een package worden ROS nodes, configuraties en andere bestanden opgeslagen.

Het grootste verschil tussen ROS 1 en ROS 2 is op het gebied van beveiliging. ROS 2 maakt gebruik van DDS (Data Distribution Service) [4]. DDS is een open standaard die publish-subscribe communicatie voorschrijft. DDS-Security is een uitbreiding op de DDS-standaard [5]. Deze uitbreiding biedt standaardfunctionaliteit voor authenticatie, access control, cryptografie, logging en data tagging voor. Daarnaast biedt het de mogelijkheid voor ontwikkelaars om implementaties te schrijven voor deze functionaliteit. Dit wordt gefaciliteerd middels plugins.

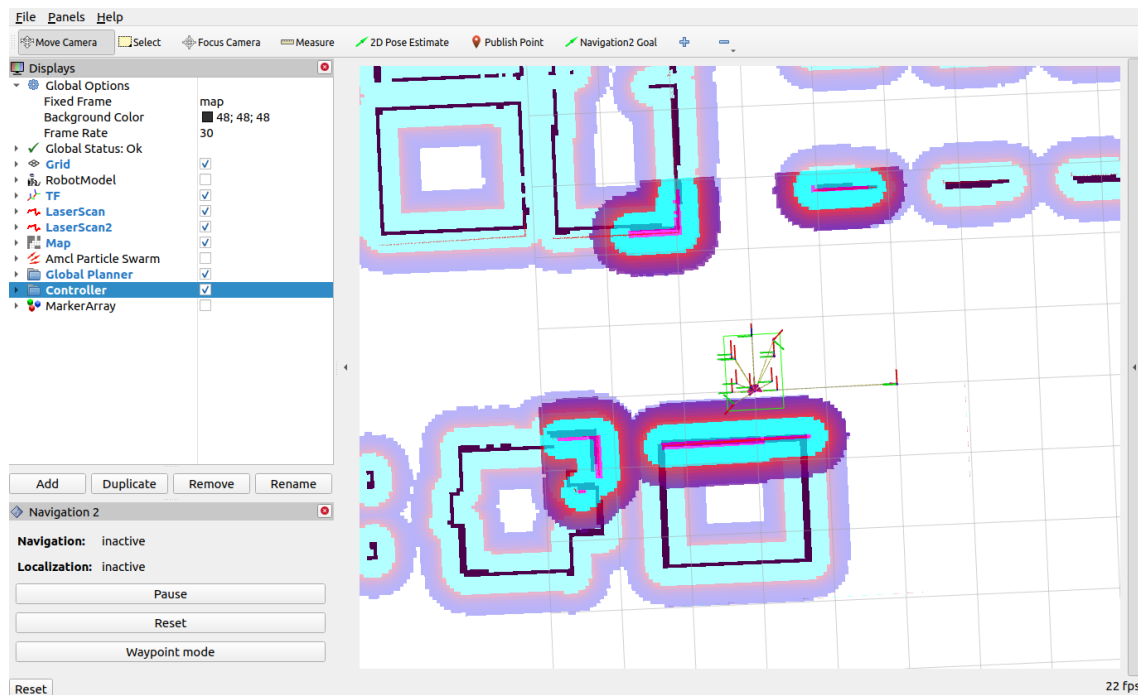
In het begin van het NeNa project is er met ROS 1 gewerkt. Later is tijdens de ontwikkeling overgegaan op ROS 2. Dat maakt de NeNa robot het eerste prototype dat door het lectoraat Mechatronica is ontwikkeld middels ROS 2. Bij het lectoraat bestaat daarom nog geen duidelijk beeld welke beveiligingsopties met DDS geboden worden en hoe deze te gebruiken zijn.

De NeNa robot gebruikt de open-source Nav2 library [6][7]. Nav2 is een navigatiestack voor ROS 2. Nav2 biedt een verzameling standaardfunctionaliteit om navigatie voor robots te faciliteren. Het doel van de Nav2 library is het vereenvoudigen van complexe navigatiefunctie, zodat ontwikkelaars niet voor elke robot het wiel opnieuw uit hoeft te vinden. De Nav2 library stelt ontwikkelaars ook in staat om de standaardfunctionaliteit uit te breiden of nieuwe functionaliteit toe te voegen. De exacte werking van de Nav2 library wordt in het onderzoek in hoofdstuk 4.1.2 uitgebreid toegelicht. De NeNa robot is volledig te simuleren met Gazebo [8][9]. Gazebo is een onafhankelijke applicatie die ontwikkelaars in staat stelt robots te simuleren. In onderstaande afbeelding is de simulatieomgeving te zien. De testomgeving die in Gazebo geladen wordt is opgezet door AWS Robotics [10].



Afbeelding 3, de Gazebo simulatieomgeving

Om de NeNa robot te visualiseren wordt RViz gebruikt [11]. RViz is een 3D visualisatieapplicatie voor ROS robots. In onderstaande afbeelding is de visualisatieapplicatie te zien. Met RViz is het ook mogelijk een Goal voor de Nav2 stack door te geven.



Afbeelding 4, de RViz visualisatieomgeving

In hoofdstuk 2 wordt de opdracht beschreven. De probleemstelling wordt toegelicht met de bijhorende onderzoeksvragen, de grenzen van het afstudeerproject worden aangegeven en er wordt aangegeven welke beroepsproducten aan het eind het afstudeerproject worden opgeleverd.

In hoofdstuk 3 wordt de procesmatige aanpak van het afstudeerproject toegelicht.

In hoofdstuk 4 wordt het verrichte onderzoek toegelicht. Er wordt besproken hoe het onderzoek is uitgevoerd, welke vragen onderzocht zijn, wat de resultaten van het onderzoek zijn en de daaruit volgende conclusies.

In hoofdstuk 5 wordt het ontwerp en de realisatie hiervan besproken.

In hoofdstuk 6 wordt het adviesrapport beschreven. Het adviesrapport is gebaseerd op de bevindingen van het onderzoek naar de beveiligingsopties die met DDS in ROS 2 geboden worden.

In hoofdstuk 7 wordt gereflecteerd op het afstudeerproject.

In hoofdstuk 8 worden de conclusies van het afstudeerproject toegelicht en eventuele aanbevelingen besproken.

2. Opdracht

In dit hoofdstuk worden de problemen met de huidige NeNa robot toegelicht, wat het doel is van de afstudeeropdracht, welke grenzen er zijn gesteld aan de opdracht en welke beroepsproducten bij afronding van de opdracht opgeleverd worden.

2.1 Probleemstelling

De huidige versie van de NeNa robot is niet in staat om in ongecontroleerde omgevingen autonoom over een krat heen te rijden zodat deze opgepakt kan worden. De huidige implementatie is gedemonstreerd in een gecontroleerde omgeving waarin een krat op een vooraf bepaalde afstand recht voor de tunnel van de NeNa robot is geplaatst en de NeNa robot recht vooruit te laten rijden.

Om de NeNa robot volledig autonoom zijn taak te laten verrichten is het noodzakelijk dat deze in staat is om de kratten nauwkeurig te detecteren en daarna nauwkeurig hier overheen kan rijden. De tunnel van de NeNa robot waarin het mechanisme zit om een krat op te pakken is in totaal 45 cm breed, zie bijlage 1. De kratten zijn 40 cm breed, zie bijlage 2. Dit houdt in dat de marge aan weerszijde van de tunnel van de NeNa robot 2,5 cm bedraagt. De te implementeren functionaliteit dient de nauwkeurigheid van de detectie en navigatie onder deze marge te houden.

Daarnaast is er bij het lectoraat nog geen onderzoek gedaan naar de beveiligingsopties die met DDS in ROS 2 geboden worden. Hierdoor is het niet volledig duidelijk welke mogelijkheden er zijn en in welke situaties het verantwoord is om er gebruik van te maken om de beveiliging te bevorderen. Zonder beveiliging is het voor kwaadwillenden mogelijk om data te onderscheppen en aan te passen van robots die met een netwerk communiceren. Zo is het mogelijk om de normale werking van de robot te onderbreken of negatief te beïnvloeden. Zodoende is het voor het lectoraat van belang om een goed beeld te krijgen van de beveiligingsmogelijkheden. Zo kunnen toekomstige ROS 2 prototypes en eindproducten hiermee uitgerust worden om een veilige werking te garanderen.

2.2 Doelstelling

Het doel van dit afstudeerproject is om, met een onderzoek en prototype, te ontdekken of het mogelijk is de NeNa robot in staat te stellen om consistent en autonoom over kratten heen te rijden. Hiervoor wordt onderzoek gedaan naar passende detectie- en navigatieoplossingen. Daarnaast dient onderzocht te worden hoe de NeNa robot en ROS 2 robots in het algemeen beveiligd kunnen worden.

Tijdens het afstudeerproject wordt er een prototype gebouwd. De basis voor dit prototype is de huidige NeNa robot en zal na afronding van dit project nieuwe detectie- en navigatiefunctiealiteit bevatten.

2.2.1 Onderzoeksvragen

Om de doelstelling te realiseren is de volgende hoofdvraag opgesteld:

- *Hoe kan de NeNa robot consistent zijn doel detecteren, er overheen rijden en beveiligd worden?*

De bovenstaande hoofdvraag is opgedeeld in vier deelvragen:

- *Wat zijn bestaande methoden om de positie en oriëntatie van kratten te detecteren op basis van LiDAR data?*

Met deze onderzoeksvraag is er gekeken naar methoden die eerder binnen het lectoraat zijn gebruikt om dit vraagstuk op te lossen. Er is gebruik gemaakt van literatuuronderzoek om tot een antwoord te komen op deze deelvraag.

- *Hoe kan de NeNa robot correct over het krat heen rijden om deze op te pakken?*

Om tot een antwoord te komen op deze onderzoeksvraag is er wederom gebruik gemaakt van literatuuronderzoek. Er is onderzocht wat de mogelijkheden zijn die goed passen in de huidige NeNa robot.

- *Hoe kunnen de ROS 2 beveiligingsopties toegepast worden?*

Voor deze onderzoeksvraag is er gebruikt gemaakt van literatuur- en werkplaatsonderzoek. De bronnen die hiervoor zijn gebruikt zijn documentatiepagina's van ROS 2 en DDS. De bevindingen zijn getest in een eenvoudige ROS 2 omgeving om een indruk te krijgen van de werking van de beveiligingsopties.

- *Hoe kan bovenstaande kennis samengevoegd worden tot een demonstreerbaar prototype?*

Met deze onderzoeksvraag is onderzocht hoe er een werkend prototype ontwikkeld kan worden op basis van de bovenstaande onderzoeksvragen.

De resultaten van deze onderzoeksvragen worden in hoofdstuk 4 Onderzoek uitgebreid besproken.

2.3 Projectgrenzen

Voor de implementatie van de detectie- en navigatiefunctie wordt een onnauwkeurige positie van de locatie van het op te pakken krat verwacht. Tijdens het ontwikkelen is dit te simuleren door in RViz een navigatiedoel op te geven. Dit wordt gezien als de startsituatie waarin de NeNa robot zich begeeft in de context van deze opdracht.

Het toepassen van de resultaten uit het onderzoek naar de beveiliging valt in eerste instantie buiten de scope van de afstudeeropdracht. Mocht er tijdens het afstudeerproject tijd zijn wordt dit alsnog geïmplementeerd voor de functionaliteit die tijdens de implementatiefase ontwikkeld is. De implementatie wordt gedocumenteerd in een adviesrapport. Dit adviesrapport kan door ontwikkelaars binnen het lectoraat geraadpleegd worden tijdens het instellen van de beveiligingsopties.

2.4 Beroepsproducten

De volgende beroepsproducten worden aan het eind van het afstudeerproject opgeleverd.

2.4.1 Onderzoeksrapport

Het onderzoeksrapport bevat een overzicht van de opgestelde onderzoeksvragen en een uitgebreide uitwerking van de deelvragen en een conclusie als antwoord op de hoofdvraag.

2.4.2 Ontwerp

Het ontwerp voor de ontwikkelde functionaliteit met betrekking tot de detectie en navigatie.

2.4.3 Prototype

Het prototype bestaat uit de ontwikkelde functionaliteit. Deze code is opgeslagen in een Bitbucket Git repository.

2.4.4 Adviesrapport

Het adviesrapport beschrijft de ervaring met het implementeren van de beveiligingsopties en geeft advies over hoe deze beveiligingsopties toegepast kunnen worden in toekomstige ROS 2 projecten.

Het adviesrapport is optioneel en geen vereiste om het afstudeerproject af te ronden.

3. Projectaanpak

3.1 Projectmanagementmethode

De gekozen projectmanagementmethode is Scrum. Deze methode is gekozen omdat de te implementeren functionaliteit makkelijk in verschillende sprints opgedeeld kon worden. Elke week is er een vaste meeting om de voortgang, eventuele problemen en andere zaken te bespreken met de bedrijfsbegeleider. Elke vier weken is er een gezamenlijke meeting met de bedrijfsbegeleider en de stagebegeleider om de voortgang te bespreken en de huidige documenten te bespreken en van feedback te voorzien.

3.2 Ontwikkeling

Het prototype is ontwikkeld in C++. Hiervoor is gekozen omdat het grootste deel van de NeNa robot in C++ is geschreven. Als IDE is CLion gebruikt van JetBrains. Hoewel hier in het lectoraat weinig tot geen gebruik van wordt gemaakt voor de ontwikkeling van ROS 2 producten, is hier toch voor gekozen omdat de student al veel ervaring heeft met de IDE's die door JetBrains geleverd worden en Saxion voor deze producten licenties levert aan studenten. Daarnaast zijn ROS projecten relatief eenvoudig in CLion in te laden [12].

De code wordt bijgehouden met Git in Bitbucket. De broncode voor de NeNa robot staat in twee Git repositories. Er is een repository voor de code voor de NeNa robot en een repository met de bestanden voor de simulatieomgeving.

Omdat thuisgewerkt wordt op een desktop en buitenshuis op een laptop is er gekozen om de ontwikkelomgeving en de documenten met betrekking tot het afstudeerproject op een externe SSD-schijf te zetten. Voor de ontwikkelomgeving is een virtuele machine opgezet, voorzien van Ubuntu 20.04.

3.3 Fases

Het project is opgedeeld in vier fases. In de volgende paragrafen wordt elke fase toegelicht.

3.3.1 Voorbereidingsfase

Tijdens de voorbereidingsfase is het plan van aanpak opgesteld. Hierin staat de aanpak van het project beschreven, de probleemstelling en bijhorende onderzoeksvragen. Ook is de onderzoeksoptzet geformuleerd en is er een planning opgesteld waarin alle fases en belangrijke momenten zijn opgenomen.

In deze fase is er ook hands-on ervaring opgedaan met ROS 2, waaronder het opzetten van nodes en publish-subscribe communicatie tussen de nodes.

De voorbereidingsfase heeft twee weken geduurd.

3.3.2 Onderzoeksfase

In de onderzoeksfase zijn de opgestelde onderzoeksvragen uit het plan van aanpak onderzocht en beantwoord. Deze resultaten zijn opgenomen in het onderzoeksrapport. Tijdens het onderzoeken is er ook gelijktijdig hands-on gewerkt met de resultaten om een idee te krijgen van de tijd die nodig is om bepaalde functionaliteit te implementeren in de ontwikkelfase. Zo zijn bijvoorbeeld een aantal beveiligingsopties al getest om de werking ervan te beoordelen.

De onderzoeksfase heeft vijf weken geduurd.

3.3.3 Ontwikkelfase

De ontwikkelfase begon met het opstellen van een ontwerp aan de hand van de resultaten uit de onderzoeksfase. Na het opstellen van het ontwerp en het verwerken van de feedback van de bedrijfsbegeleider en de stagebegeleider zijn de sprints begonnen. In sprint 1 werd er gewerkt aan de detectiefunctie. In sprint 2 werd er gewerkt aan de navigatiefunctie. In sprint 3 werd er gewerkt aan het adviesrapport en implementatie van de beveiligingsopties. Sprint 4 is in overleg met de bedrijfsbegeleider als buffer sprint ingedeeld. Voorgaande sprints kunnen opgerekt worden als er door onvoorziene omstandigheden meer tijd nodig blijkt te zijn om bepaalde functionaliteit af te ronden. Verder is er de mogelijkheid om eventuele could-have requirements te implementeren.

In de ontwikkelfase is het concept van dit document, het afstudeerverslag, ingeleverd.

De ontwikkelfase heeft in totaal negen weken geduurd.

3.3.4 Afrondingsfase

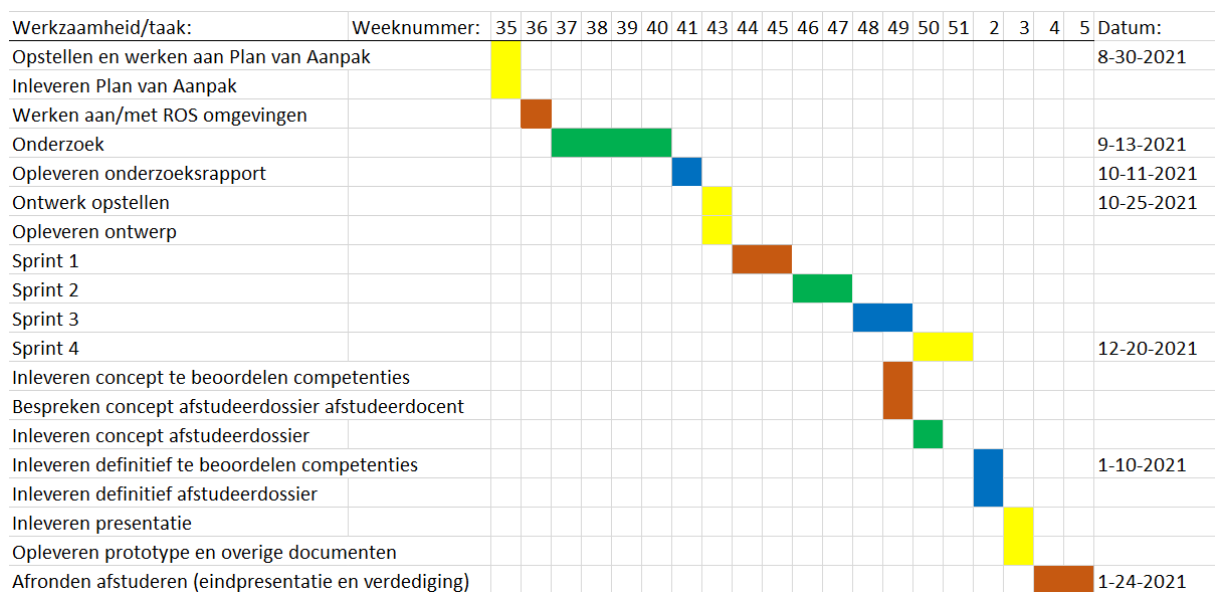
In deze fase zijn de definitieve versie van het afstudeerdossier en de afstudeerpresentatie ingeleverd. Ook zijn de in hoofdstuk 2 beschreven beroepsproducten opgeleverd. In deze fase bevindt zich ook de eindpresentatie en de verdediging van het afstuderen.

Tijdens de afrondingsfase is er waar mogelijk gewerkt aan de overgebleven

De afrondingsfase heeft twee weken geduurd.

3.3.5 Planning

In onderstaande afbeelding is de planning te zien



Afbeelding 5, de planning

4 Onderzoek

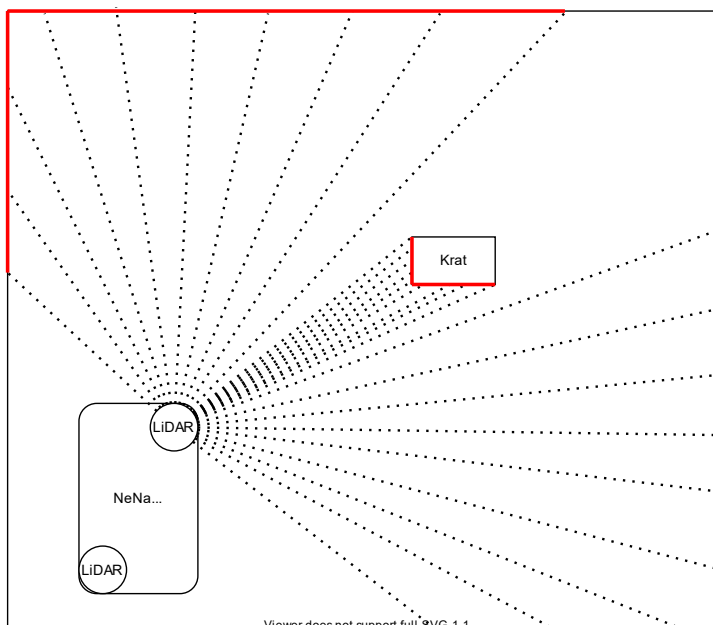
Het verrichte onderzoek is kwalitatief van aard geweest. Om de onderzoeksvragen te beantwoorden is er gebruik gemaakt van literatuur- en werkplaatsonderzoek. Voor het literatuuronderzoek zijn wetenschappelijke papers, documentatiepagina's en discussieforums gebruikt. Voor het werkplaatsonderzoek zijn bevindingen tijdens het onderzoek getest om een beter beeld te krijgen hoeveel tijd de te implementeren functionaliteit gaat kosten tijdens de ontwikkelfase.

4.1 Onderzoeksvragen

In deze paragraaf worden de resultaten per onderzoeksvraag toegelicht.

4.1.1 Wat zijn bestaande methoden om de positie en oriëntatie van kratten te detecteren op basis van LiDAR data?

LiDAR data bestaat uit een serie afstandsmetingen. Een LiDAR scanner doet een groot aantal afstandsmetingen vanaf een bepaalde beginhoek. In de context van de NeNa robot zijn dit er ruim 1000. In de onderstaande afbeelding is een visualisatie te vinden van de werking van een LiDAR scanner.



Afbeelding 6, visualisatie van een LiDAR scan, in rood het 'zicht' van de scanner (frequentie van metingen zijn illustratief)

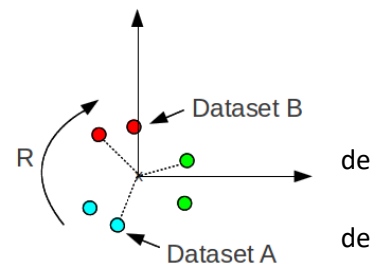
Detectiemethoden

Er zijn twee methoden onderzocht om kratten te detecteren op basis van LiDAR data. De eerste methode is het ICP (Iterative Closest Point) algoritme [13]. In het lectoraat is al vaker gewerkt met het ICP-algoritme. De tweede methode die is onderzocht is het detecteren van het krat op basis van goniometrie.

ICP-algoritme

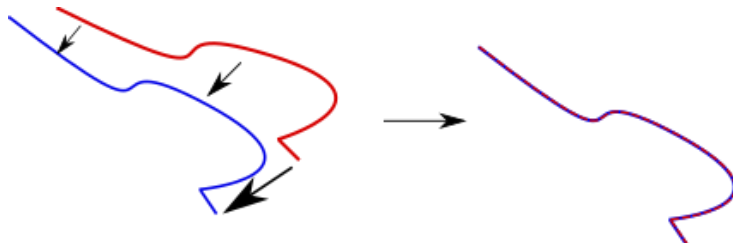
Het ICP-algoritme werkt met twee point clouds. Een point cloud is een verzameling punten in twee of meer dimensies. De eerste point cloud in het ICP-algoritme is de referentie point cloud, het doel. In de context van de NeNa robot zal dit een point cloud van het krat zijn dat eerder is ingescand of opgesteld. De tweede point cloud is de bron, in dit geval de LiDAR data. Het doel van het ICP-algoritme is het verschil tussen deze twee point clouds minimaliseren, oftewel de twee point clouds elkaar zo goed mogelijk laten overlappen.

De implementatie van het ICP-algoritme berust op twee stappen die een bepaald aantal iteraties worden uitgevoerd, waarbij elke iteratie een verbetering oplevert. De eerste stap is om voor elk punt van de referentie point cloud de afstand te bepalen tot het dichtstbijzijnde punt in de bron point cloud. Deze afstand kan berekend worden met de formule $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Voor al deze punten in beide point clouds wordt het middelpunt berekend. Vervolgens wordt de bron point cloud getransleerd, zodat de middelpunten van beide point clouds overeenkomen. Stap twee is het bepalen van de rotatie die de afstand tussen de punten in beide point clouds minimaliseert. Een zeer bekende en veelgebruikte methode is Singular Value Decomposition (SVD). Singular Value Decomposition kan gebruikt worden om in één berekening de rotatie tussen twee point clouds te berekenen. Dit gebeurt door matrixen van de referentie point cloud en de bron point cloud te reduceren tot één matrix. Deze matrix beschrijft de rotatie tussen referentie en bron point clouds.



Afbeelding 7, bron: [14]

De transformaties van elke iteratie kunnen bij elkaar opgeteld worden om een definitieve transformatie van het referentie point cloud naar het bron point cloud te krijgen. Dit is de locatie van het krat ten opzichte van de LiDAR scanner. In onderstaande afbeelding is een eenvoudige visualisatie van de werking van het ICP-algoritme gevisualiseerd.



Afbeelding 8, eenvoudige visualisatie van het ICP-algoritme, Bron: [13]

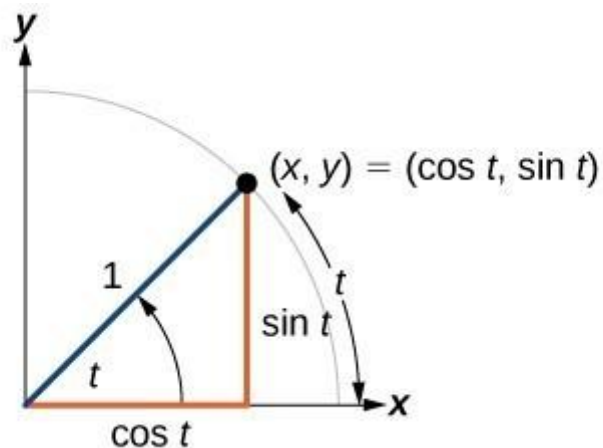
Detectie op basis van goniometrie

De tweede onderzochte methode is het detecteren van kratten op basis van goniometrie. In het wetenschappelijke paper 'Automatic precision docking for autonomous mobile robots' is onderzocht hoe een robot autonoom kan docken met een oplaadstation middels een marker met een unieke vorm [14]. Dit concept kan ook toegepast worden voor de detectiefunctie en het docking-vraagstuk. Het krat is een eenvoudig object om te detecteren op basis van de hoek die twee zijden van het krat maken.



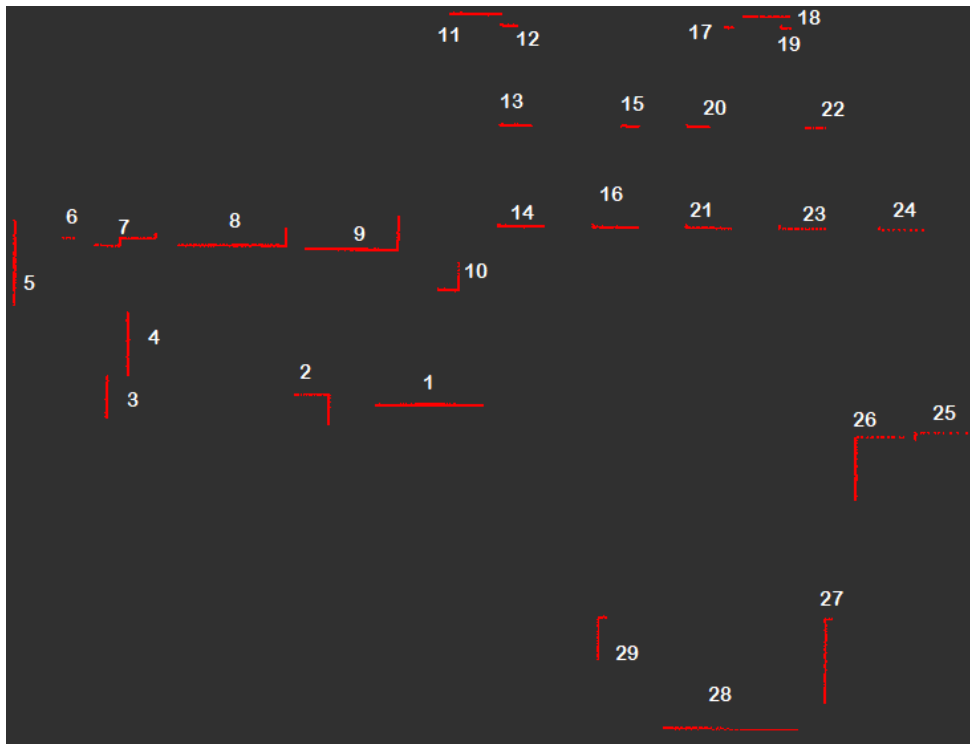
Afbeelding 9, LiDAR detecties op het krat

Zoals in bovenstaande afbeelding te zien is, heeft de LiDAR scanner meerdere detecties op het krat. Voor deze detecties zijn de relatieve coördinaten ten opzichte van de LiDAR scanner te berekenen. De x-coördinaat is gelijk aan $afstand * \cos(\theta)$ en de y-coördinaat is gelijk aan $afstand * \sin(\theta)$. De hoek is θ (theta). De afstand en hoek zijn bekend in de LiDAR data.



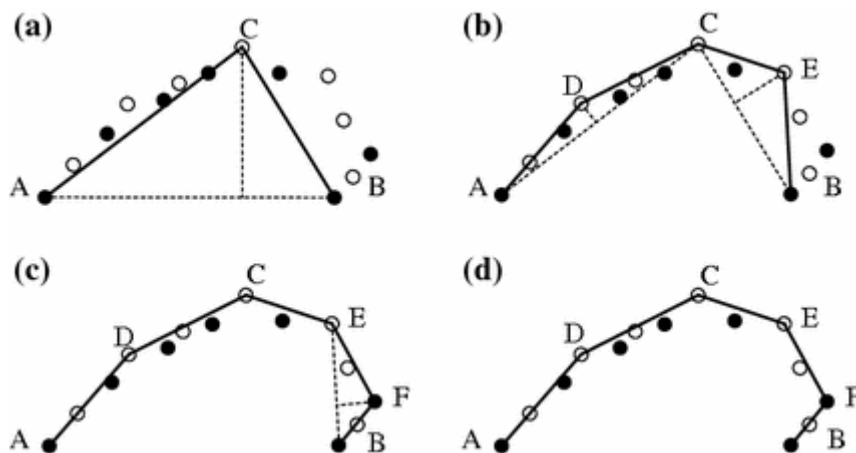
Afbeelding 10, bron: [16]

Met de berekende punten zijn lijnen op te stellen. Voor elk punt wordt bepaald of deze bij een bestaande of nieuwe lijn hoort op basis van de afstand van dit punt tot het vorige punt. In onderstaande afbeelding is een LiDAR scan van de virtuele omgeving waar de NeNa robot zich in bevindt te zien.



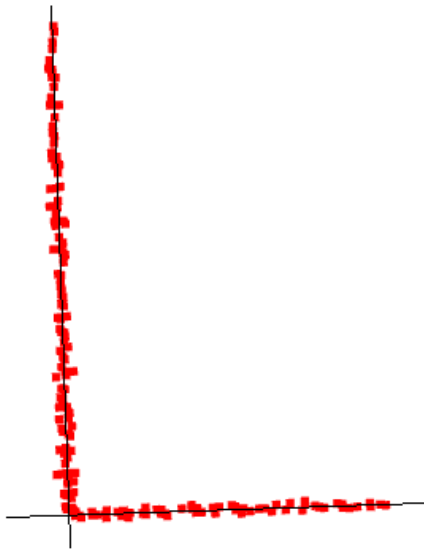
Afbeelding 11

Deze lijnen kunnen echter ook samengestelde lijnen zijn, zoals in bovenstaande afbeelding te zien is. Om de lijnen waaruit deze samengestelde lijnen bestaan te verkrijgen, kan het Iterative End-Point Fit algoritme gebruikt worden [16]. Het Iterative End-Point Fit algoritme is een algoritme dat recursief een gegeven lijn op probeert te delen in afzonderlijke lijnen. Dit werkt door de afstand van elk punt tussen het begin en einde van de lijn te berekenen. Als de grootste afstand groter is dan een bepaalde waarde, ook wel de epsilon (ϵ) genoemd, dan wordt de huidige lijn opgesplitst in twee lijnen, waarvan het eindpunt van de eerste lijn en het beginpunt van de tweede lijn het huidige punt is. In onderstaande afbeelding is de werking van het Iterative End-Point Fit algoritme gevisualiseerd.



Afbeelding 12, bron: [18]

Uit het Iterative End-Point Fit algoritme volgt een verzameling met verzamelingen van lijnen. Alleen een verzameling met exact twee lijnen zijn relevant voor een detectie, aangezien er twee zijden van het krat nodig zijn om de locatie ervan te bepalen. Van deze zijn vervolgens lineaire regressielijnen op te stellen. Met lineaire regressie kan er een lineair verband aangetoond worden tussen twee variabelen. In de context van de detectiecoördinaten kan lineaire regressie ook uitstekend gebruikt worden om een gemiddelde lijn door een set coördinaten te berekenen. Deze lineaire regressielijn heeft de standaard lineaire formule $y = ax + b$, met a als richtingscoëfficiënt en b als beginwaarde. De a waarde wordt gegeven door de formule $a = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$ met n als aantal coördinaten. De b waarde wordt gegeven middels de formule $b = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}$ met n als aantal coördinaten.



Afbeelding 13, LiDAR detecties op het krat en de lineaire regressielijnen

Met de opgestelde lineaire regressielijnen is het mogelijk om de hoek tussen beide lijnen op te stellen. De hoek θ tussen twee lijnen wordt gegeven door de formule $\theta = \left| \tan^{-1} \left(\frac{a_1 - a_2}{1 + a_1 a_2} \right) \right|$. Deze hoek zal tussen 0 en 90 graden liggen.

Vervolgens is het snijpunt te berekenen op basis van de opgestelde lineaire regressielijnen. Dit snijpunt is de hoek van het krat. De x-coördinaat wordt gegeven volgens de formule $x = \frac{b_2 - b_1}{a_1 - a_2}$. De y-coördinaat is te vinden door de berekende x coördinaat te substitueren in een van de twee formules van de lineaire regressielijnen.

Tenslotte kan het middelpunt van het krat berekend worden met de afstand van het snijpunt naar het middelpunt en de hoek naar het middelpunt vanaf het snijpunt. De afstand is te bereken met de formule $d = \sqrt{\left(\frac{\text{breedte krat}}{2}\right)^2 + \left(\frac{\text{lengte krat}}{2}\right)^2}$. De hoek naar het middelpunt wordt gegeven door de formule $\theta_{\text{middelpunt}} = \theta_{\text{krat}} \pm \tan^{-1} \left(\left(\frac{1}{2} \text{ breedte krat}\right) / \left(\frac{1}{2} \text{ lengte krat}\right) \right)$ waar θ_{krat} gelijk is aan de inverse tangens van de richtingscoëfficiënt van de lineaire regressielijn van de langste zijde. Deze hoek is ook te beschrijven als de oriëntatie van het krat ten opzichte van de LiDAR scanner. Voor de x-coördinaat van het middelpunt geldt $x_{\text{middelpunt}} = x_{\text{snijpunt}} + d * \cos(\theta_{\text{middelpunt}})$. Voor de y-coördinaat geldt $y_{\text{middelpunt}} = y_{\text{snijpunt}} + d * \sin(\theta_{\text{middelpunt}})$.

4.1.2 Hoe kan de NeNa robot correct over het krat heen rijden om deze op te pakken?

Er zijn twee methoden onderzocht die passen in de huidige NeNa robot.

Handmatige aansturing

De eerste methode is het handmatig aansturen van de robot op basis van de LiDAR data. Een robot kan in principe altijd handmatig aangestuurd worden door een commando te sturen naar het 'cmd_vel' topic. De ongeschreven regel binnen ROS is dat dit topic wordt gebruikt om de robot aan te sturen. De commando's die dit topic accepteert bestaan uit een lineaire snelheid en rotatiesnelheid. De wielen worden op basis van deze commando's aangestuurd.

Om dit te implementeren dient er ook functionaliteit geschreven worden om obstakels te detecteren. Verder moet een pad gepland worden dat de robot kan afleggen om bij het doel te komen. Dit pad moet vertaald worden naar commando's voor de aansturing.

Nav2

De tweede methode is het gebruikmaken van de Nav2 library die al in de NeNa robot wordt gebruikt. Nav2 biedt standaard navigatiefunctieiteit waarmee algemene navigatietaken volbracht kunnen worden. Deze functionaliteit is geschreven op basis van een aantal plugins die Nav2 beschikbaar stelt. Met deze plugins kunnen ontwikkelaars ook zelf implementaties schrijven die in de navigatie stack gebruikt kunnen worden. Hierdoor is Nav2 relatief eenvoudig uit te breiden naar wens om een robot te voorzien van speciale navigatiemogelijkheden. De Nav2 library maakt gebruik van verschillende plugins [19].

- Costmap plugin:
De costmap plugin houdt een 2D representatie bij van de omgeving waarin de robot zich bevindt. De costmap kan door andere plugins geraadpleegd worden om informatie met betrekking tot mogelijke obstakels te verkrijgen. In de standaardfunctionaliteit wordt de costmap gebruikt voor het plannen en volgen van een pad naar een bepaald doel. Het is ook mogelijk om in de costmap lagen aan te brengen die no-go zones specificeren of zones waarin een bepaalde (maximale) snelheid aangehouden dient te worden.
- Planner plugin:
De planner plugin berekent een pad naar een bepaald doel. Met het plannen van het pad kan ook de costmap plugin geraadpleegd worden om zo obstakels te vermijden bij het plannen. De planner blijft herhaaldelijk een pad plannen naar het huidige doel.
Op het moment van schrijven kent de Nav2 library vijf standaardimplementaties van de planner plugin.
- Controller plugin:
De controller plugin volgt het berekende pad van een planner. De controller plugin stuurt de robot aan en kan met de aansturing ook rekening houden met dynamische obstakels, zoals voorbijlopende mensen of een object dat op het pad is terechtgekomen door de costmap plugin te raadplegen. De controller controleert de huidige positie en het berekende pad van de planner meerdere malen per seconde en stuurt commando's om de pad te blijven volgen.
Op het moment van schrijven zijn er vier standaardimplementaties van de controller plugin.
- Recovery plugin:
De recovery plugins hebben als doel om de robot van logica te voorzien waarmee met onbekende situaties of fouten om kan worden gegaan. Nav2 biedt standaardimplementaties om een bepaalde tijd te wachten, een bepaalde afstand achteruit te rijden of rond te draaien.

- BehaviorTree plugin:

De BehaviorTree plugins verzorgt het gedrag van een robot. Een BehaviorTree bestaat uit BehaviorTree nodes. Deze nodes stellen stukken gedrag voor. Enkele voorbeelden van hiervan zijn het omschakelen naar een andere planner of controller in een bepaalde situatie. Ook kan de frequentie van de planner geconfigureerd worden en het aantal navigatiepogingen dat gemaakt moet worden voordat alternatief gedrag, zoals een Recovery plugin, geactiveerd wordt. De BehaviorTree plugin biedt ontwikkelaars de mogelijkheid om het gedrag van de robot te definiëren in een menselijk leesbare manier met een XML-bestand. Daarnaast is het mogelijk zelf BehaviorTree nodes toe te voegen.

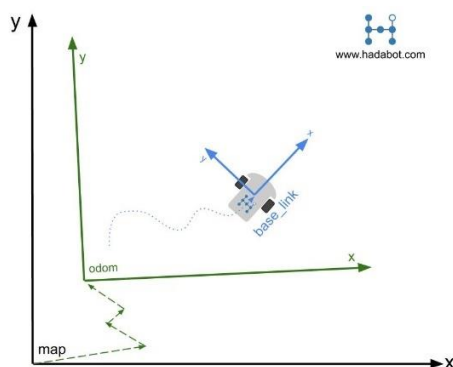
Nav2 maakt gebruik van een configuratiebestand waarin de functionaliteit geconfigureerd dient te worden. In dit configuratiebestand zijn de beschikbare plugins gedefinieerd en zijn andere instellingen met betrekking tot bepaalde aspecten van Nav2 te configureren.

Om de positie van de robot bij te houden wordt er gebruik gemaakt van de tf2 library in Nav2. De tf2 library houdt coordinate frames bij. Een coordinate frame is een driedimensionaal coördinatensysteem. Coordinate frames en de positie van objecten kunnen in andere coordinate frames worden uitgedrukt. Een coordinate frame kan meerdere 'child' frames hebben, maar elk coordinate frame kan slechts één 'parent' frame hebben. Zo kan een hiërarchie opgesteld worden van meerdere coordinate frames. De positie van een bepaald object dat is uitgedrukt in een coordinate frame kan met tf2 eenvoudig opgevraagd worden in een ander coordinate frame in dezelfde hiërarchie.

Neem bijvoorbeeld een robotarm met meerdere segmenten, gewrichten en een grijper. Om een object op te pakken is het noodzakelijk om te weten waar elk segment zich bevindt in verhouding tot het aansluitende segment. Deze verhoudingen worden beschreven als transformaties, een combinatie van translatie en rotatie. Om al deze verhoudingen zelfstandig bij te houden en te berekenen is niet eenvoudig en foutgevoelig. Bovendien is het niet nodig om het wiel opnieuw uit te vinden, aangezien tf2 exact ontwikkeld is om in dit soort situaties toegepast te worden.

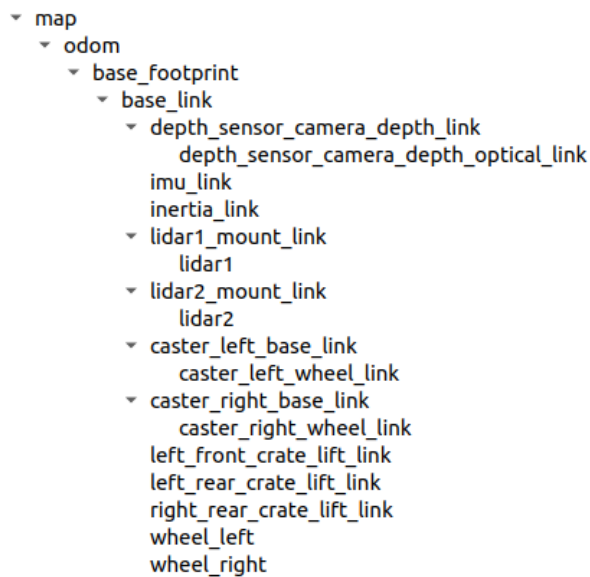
Tf2 ondersteunt statische en dynamische transformaties tussen coordinate frames. Statische transformaties kunnen in bestanden gedefinieerd worden. Voorbeelden hiervan zijn scanners die gemonteerd zijn op de robot, zoals de LiDAR scanners op de NeNa robot. Dynamische transformaties kunnen toegevoegd worden als de robot in bedrijf is. Voorbeelden hiervan zijn objecten die de robot tegenkomt terwijl deze in bedrijf is, zoals een krat in de context van deze opdracht.

De REP (Ros Enhancement Proposal) 105 schrijft voor dat er voor grondrobots een aantal standaard coordinate frames gedefinieerd moeten zijn voor het 'map', 'odom' en 'base_link frame'. In onderstaande afbeelding is er een overzicht te zien van deze frames.



Afbeelding 14, visualisatie van het map, odom en base_link frame conform REP 105, bron: [20]

Het base_link frame is de basis van de robot, vaak wordt hiervoor het middelpunt gekozen maar dat is niet noodzakelijk of verplicht. In het base_link frame kunnen zich onderdelen van de robot bevinden. In onderstaande afbeelding wordt de tf2 hiërarchie van de NeNa robot weergegeven.



Afbeelding 15, tf2 hiërarchie van de NeNa robot

Het is overigens niet verplicht om het map, odom en base_link frame direct onder elkaar te definiëren. Het is toegestaan om gebruik te maken van tussenliggende coordinate frames, zoals in bovenstaande afbeelding te zien is, mits het odom frame niet boven het map frame staat. Ditzelfde geldt voor het base_link en odom frame.

Het odom frame houdt de positie van de robot bij, het base_link frame. Het odom frame heeft als garantie dat de positie van de robot altijd geleidelijk wordt bijgewerkt. Hiervoor wordt odometrie gebruikt en daar heeft het odom frame ook zijn naam aan te danken. Bij odometrie wordt gebruik gemaakt van bewegingssensoren om de verandering in positie door tijd bij te houden. Het odom frame wordt in het map frame periodiek bijgewerkt. Dit kan echter in sprongen gaan. Normaal gesproken zijn deze sprongen klein genoeg om geen duidelijke impact te hebben op de navigatiefunctie. Echter is dit niet wenselijk bij toepassingen waarin de navigatie nauwkeurig uitgevoerd dient te worden.

4.1.3 Hoe kunnen de ROS 2 beveiligingsopties toegepast worden?

Zoals als in de inleiding beschreven, is ROS 2 gebouwd op DDS met de DDS-Security uitbreiding. ROS 2 maakt gebruik van de authenticatie, access control en cryptografie plugins. De logging en data tagging zijn niet vereist om te voldoen aan de DDS-Security specificatie. DDS werkt op basis van domeinen (DDS-domains). Een DDS-domein is een omgeving die bestaat uit deelnemers (DDS-participants). Een ROS node is rechtstreeks te vertalen naar een domeindeelnemer. In de volgende paragrafen wordt de werking van de authenticatie, access control en cryptografie plugins toegelicht.

Authenticatie plugin

De authenticatie plugin maakt gebruik van het veelgebruikte Public Key Infrastructure (PKI). PKI werkt op basis van een autoriteit die door meerdere partijen vertrouwd wordt, een CA (Certificate Authority). Deze autoriteit verleent certificaten waarmee de identiteit van de aanvrager aangetoond kan worden. Zo kan er door twee partijen gecontroleerd worden dat de andere partij is welke deze claimt te zijn. PKI is ook het fundament van beveiligde HTTP-verbindingen met websites op het internet.

Access control plugin

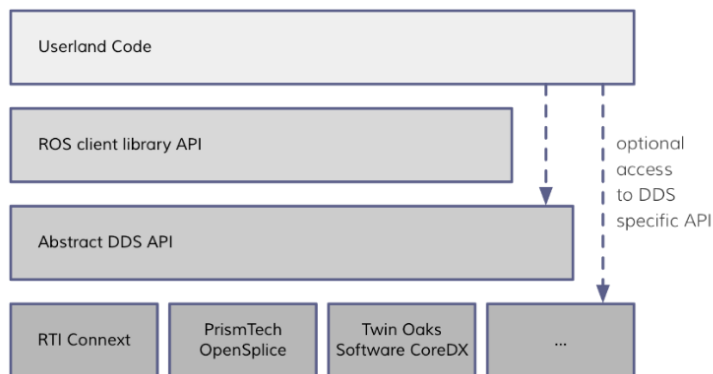
Middels de access control plugin zijn permissies te specificeren voor de deelnemers aan DDS-domeinen. Voorbeelden hiervan zijn het beperken van de mogelijke DDS-domeinen waarin een ROS node mag deelnemen of het beperken van de lees- en/of schrijfrechten van een ROS node op bepaalde ROS topics. Zo kan het principe van het laagste privilege toegepast worden [21]. Dit principe beschrijft dat een proces, gebruiker of applicatie alleen bij de nodige informatie kan die nodig is om legitieme doeleinden te behalen. Stel dat er een kwetsbaarheid zit in een bepaalde ROS node, dan blijft de schade beperkt tot de informatie waar deze node toegang tot heeft.

Cryptografie plugin

De cryptografie plugin wordt gebruikt voor versleuteling en het authenticeren van berichten. In de standaardfunctionaliteit worden de AES-GCM (Galois-Counter-Mode) en AES-GMAC (Galois-Message-Authentication-Code) algoritmen geïmplementeerd. Het AES-GCM algoritme is in staat berichten geauthentiseerd te versleutelen. Dit houdt in dat naast dat de informatie onleesbaar wordt gemaakt, de versleutelde informatie ook niet aangepast kan worden middels een bit-flipping attack [22] door het bericht te ondertekenen met een authenticatiecode. In een bit-flipping attack wordt de versleutelde data aangepast. Hoewel een aanvaller niet precies weet wat voor informatie versleuteld is, is deze alsnog in staat om aanpassingen te doen. De aangepaste versleutelde data wordt geïnterpreteerd als geldig versleutelde informatie. De impact kan verschillen van bijvoorbeeld veranderen van een te versturen geldbedrag tot een Denial-of-Service (DoS). Het AES-GMAC algoritme versleutelt de informatie niet maar voorziet alleen in authenticatie van de informatie.

DDS in ROS 2

Er zijn meerdere implementaties van de DDS standaard beschikbaar. Om DDS in ROS 2 te gebruiken is er naast de DDS-implementatie ook een DDS-middleware nodig. Deze middleware is op maat gemaakt voor een bepaalde DDS-implementatie. Het is eenvoudig om te wisselen van DDS en middleware implementatie zonder code opnieuw te moeten compileren, zie onderstaande afbeelding.



Afbeelding 16, overzicht van de verschillende lagen van gebruikerscode tot de DDS-middleware (Abstract DDS API) en DDS-implementatie

Zoals in afbeelding 16 te zien is, heeft de code alleen rechtstreeks te maken met de ROS client library API. De 'Abstract DDS API' is de DDS-middleware die bij een bepaalde DDS-implementatie hoort. ROS 2 ondersteunt officieel drie DDS-implementaties volledig:

- eProsima Fast DDS (voorheen eProsima Fast RTPS)
- Eclipse Cyclone DDS
- RTI Connext

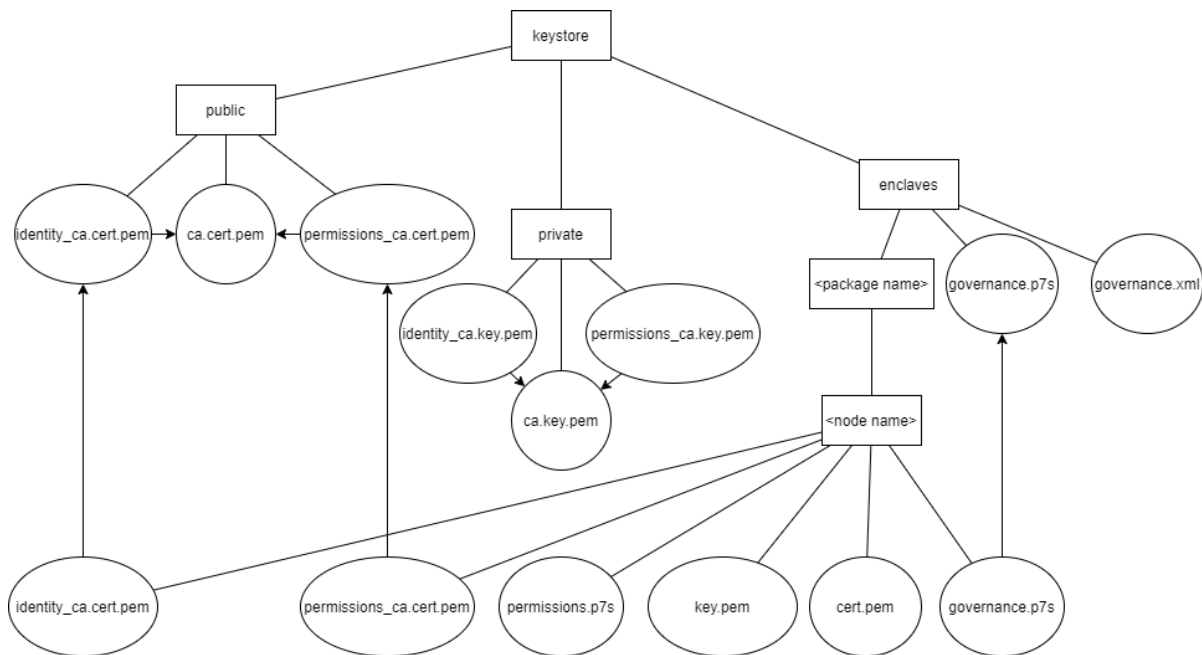
Theoretisch gezien kan elke DDS-implementatie in ROS 2 gebruikt worden, mits er een ROS middleware beschikbaar voor is. Zo is er bijvoorbeeld ook OpenDDS [23], een open source DDS-implementatie waar ook voor ROS 2 geschikte middleware beschikbaar voor is [24].

Beveiligingsopties

De beveiligingsopties staan standaard niet aan in ROS 2. Om dit aan te zetten dienen er security artifacts aanwezig te zijn en moeten er een aantal omgevingsvariabelen gezet worden.

De security artifacts zijn bestanden die nodig zijn om van de beveiligingsopties gebruik te maken en deze te configureren. ROS 2 biedt een CLI (Command Line Interface) tool waarmee deze bestanden eenvoudig aangemaakt kunnen worden [25].

De locatie waar deze security artifacts zich in dienen te begeven wordt de keystore genoemd. Deze map bevat de certificaten en private keys van de Identity en Permissions CA's. Elke ROS 2 node kan opgestart worden met een security enclave. Deze enclaves worden ook in de keystore opgeslagen in de 'enclaves' map. In de 'enclaves' map staan ook de configuratiebestanden die specificeren hoe het DDS-domein beveiligd dient te worden. In elke security enclave moeten er een aantal bestanden aanwezig zijn: een door de Identity CA ondertekend certificaat, een bijhorende private key en een xml-bestand waarin de permissies beschreven zijn, ondertekend door de Permissions CA. Verder zijn er een aantal symbolische links naar (of kopieën van) het certificaat van de Identity CA, de Permissions CA en het configuratiebestand dat de DDS-domein beveiliging specificeert. In afbeelding 17 is een duidelijk overzicht te zien van de structuur van de keystore.



Afbeelding 17, de structuur van de keystore (pijlen zijn symbolische links)

Zoals in Afbeelding 17 te zien is, is het ook mogelijk om de Identity en Permissions CA naar eenzelfde CA te laten verwijzen middels symbolische links. Dit is hoe de ROS 2 security CLI-tool de keystore inricht.

De volgende omgevingsvariabelen dienen gezet te worden om de beveiligingsopties te gebruiken:

- ROS_SECURITY_KEYSTORE
De waarde van deze omgevingsvariabele moet verwijzen naar de keystore.
- ROS_SECURITY_ENABLE
De waarde van deze omgevingsvariabele dient 'True' te zijn. Met deze omgevingsvariabele zijn de beveiligingsopties eenvoudig aan of uit te zetten, bijvoorbeeld voor debug-doeleinden.
- ROS_SECURITY_STRATEGY
De waarde van deze omgevingsvariabele dient 'Enforce' te zijn. Een andere waarde resulteert in een scenario waarin de beveiligingsopties alsnog worden toegepast, maar waar het ook mogelijk is dat nodes zonder beveiligingsopties gestart kunnen worden. In de praktijk is het zeer streng aan te raden deze omgevingsvariabele altijd op 'Enforce' te zetten en van de ROS_SECURITY_ENABLE omgevingsvariabele gebruik te maken.

Er zijn naast de bovenstaande omgevingsvariabelen nog een aantal optionele omgevingsvariabelen:

- RMW_IMPLEMENTATION
De waarde van deze omgevingsvariabele verwijst naar de naam van de DDS-middleware implementatie die gebruikt dient te worden. In ROS 2 Foxy is dit standaard 'rmw_fastdds_cpp'.
- ROS_SECURITY_LOG_FILE
De waarde van deze omgevingsvariabele dient te verwijzen naar een bestand dat gebruikt kan worden als logbestand
- ROS_SECURITY_LOG_VERBOSITY
De waarde voor deze omgevingsvariabele kan één van de volgende waarden aannemen: 'DEBUG', 'INFO', 'WARN', 'ERROR', 'FATAL'. Alleen logging met het niveau van de waarde van deze omgevingsvariabele en hoger wordt in het logbestand opgeslagen.

4.1.4 Hoe kan bovenstaande kennis samengevoegd worden tot een demonstreerbaar prototype?

Om tot een demonstreerbaar prototype te komen tijdens de ontwikkelfase is er een ROS 2 node nodig om het krat te detecteren. Deze node wordt gestart samen met de andere functionaliteit van de robot en dient het krat zo nauwkeurig mogelijk te detecteren. Er is geen concrete waarde te noemen voor de nauwkeurigheid die de detectie node dient te bereiken. Uiteraard is het wenselijk dat de detectie zo nauwkeurig mogelijk is. De uiteindelijke nauwkeurigheid van de detectie- en navigatiefunctie dienen samen de marge van 2,5 cm aan weerszijde van het krat niet te overschrijden zodat de tunnel van de NeNa robot het krat niet raakt tijdens het docken.

De navigatiefunctie is op te splitsen in meerdere stappen. Zodra de NeNa robot de onnauwkeurige locatie van het krat krijgt als doel, kan deze de standaardnavigatiefunctie in Nav2 gebruiken om richting dit doel te rijden. Als de detectie node het krat heeft gedetecteerd, schakelt deze over naar een nieuwe planner met de BehaviorTree plugin. Deze planner is gebaseerd op een standaardimplementatie maar zorgt ervoor dat de NeNa robot zichzelf op een afstand recht tegenover het krat positioneert. Deze afstand is te configureren in het configuratiebestand dat Nav2 gebruikt. Deze positie voor het krat dient dusdanig nauwkeurig te zijn dat de NeNa robot recht vooruit kan rijden zonder het krat te raken. De laatste stap is de NeNa robot vooruit laten rijden zodat deze over het krat heen rijdt.

De planners in Nav2 plannen standaard in het map frame. Zoals in hoofdstuk 4.1.2 is beschreven, is het map frame niet nauwkeurig genoeg om precies te kunnen navigeren. De nieuwe planner dient in het odom frame te plannen in plaats van het map frame.

Voor het docken met het krat zal de standaardfunctionaliteit niet voldoende zijn. Nav2 zal het krat als obstakel detecteren en vast komen te zitten. Dit dient in de costmap aangepast worden door de costmap rondom de positie van het krat te legen.

De toepassing van de beveiligingsopties kan gedemonstreerd worden in de nieuwe detectie- en navigatiefunctie. Zoals eerder is toegelicht, wordt dit alleen uitgewerkt als de detectie- en navigatiefunctie volledig is afgerond.

4.2 Conclusie: Hoe kan de NeNa robot consistent zijn doel detecteren, er overheen rijden en beveiligd worden?

In de volgende paragrafen wordt de conclusie op basis van het verrichtte onderzoek beschreven voor de verschillende onderdelen.

Detectie

Om de NeNa robot zijn doel consistent te kunnen laten detecteren is er gekozen om voor de detectiefunctie functionaliteit gebruik te maken van detectie op basis van goniometrie. In de huidige NeNa robot wordt het ICP-algoritme toegepast. Echter is het met die functionaliteit niet makkelijk om wijzigingen aan te brengen in de afmetingen van het te detecteren object. De huidige functionaliteit is afhankelijk van reeds ingescande point clouds van het krat. Dit dient voor verschillende afstanden gedaan te worden voor een consistente detectie. Voor elke afstand zijn honderd tot enkele honderden datapunten nodig. Dit zorgt ervoor dat de huidige implementatie niet eenvoudig voor andere objecten te hergebruiken is. Daarnaast is het detecteren van het krat in een relatief grote point cloud van de omgeving niet makkelijk. Er moet een initiële positie meegegeven worden om het ICP-algoritme in de juiste richting te wijzen. Het voordeel van het gebruik van het ICP-algoritme is dat complexere objecten gedetecteerd kunnen worden of een robot in een omgeving gelokaliseerd kan worden. In de context van de detectiefunctie functionaliteit in de NeNa robot is dit echter niet nodig en is detectie op basis van goniometrie beter geschikt. Het voordeel van de detectie op basis van goniometrie is dat de code eenvoudig aangepast kan worden en zo ook voor de detectie van andere objecten gebruikt kan worden.

Navigatie

Voor de navigatiefunctie functionaliteit wordt verder van de Nav2 library gebruikt. In alle functionaliteit die bij de handmatige aanstuuringsmethode geïmplementeerd dient te worden, voorziet Nav2. Indien nodig is het altijd mogelijk om een controller te schrijven welke de robot handmatig aanstuurt. Hier zit wel een nadeel aan. De controller werkt namelijk alleen zolang de robot nog niet op zijn doel is aangekomen en de Nav2 library merkt dat de robot nog wel voortgang maakt. Om de voortgang te detecteren wordt gemeten hoeveel een robot zich verplaatst over een bepaalde tijd. Dit is in het configuratiebestand voor Nav2 aan te passen.

Gezien het feit dat er meerdere implementaties zijn voor de verschillende plugins die Nav2 biedt, maakt dit het een zeer geschikte keuze. De standaardfunctionaliteit voorziet in de meeste gevallen van een implementatie waar geen aanpassingen aan gemaakt hoeven te worden. De implementaties zijn eenvoudig te tweaken in het configuratiebestand van Nav2. Daar komt bij dat Nav2 met plugin interfaces ontwikkelaars in staat stelt zelf een implementatie te schrijven.

Al deze voordelen over het handmatig aansturen van de robot en het feit dat de NeNa robot het reeds gebruikt, maakt het een eenvoudige keuze om de navigatiefunctie functionaliteit te ontwikkelen met gebruik van de Nav2 library.

Security

Voor de beveiliging wordt de te ontwikkelen functionaliteit waar mogelijk voorzien van beveiligingsopties. In eerste instantie zal dit de ROS 2 node voor de krat detectie betreffen. Verder wordt er een adviesrapport geschreven dat ook dient als handleiding. In dit adviesrapport wordt beschreven hoe de beveiligingsopties in toekomstige ROS 2 projecten geïntegreerd kunnen worden.

De implementatie van de beveiligingsopties blijft zoals eerder vermeld optioneel.

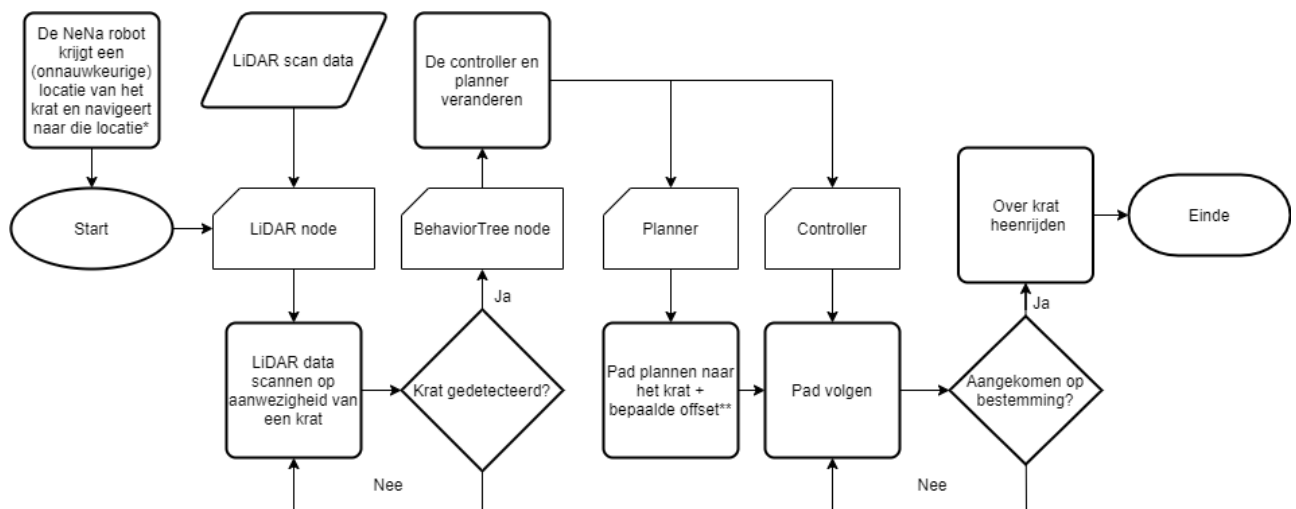
5. Ontwerp & realisatie

In dit hoofdstuk wordt het ontwerp en de realisatie van het ontwerp besproken.

5.1 Functioneel ontwerp

De te ontwikkelen functionaliteit stelt de NeNa robot in staat om zelfstandig en consistent een krat te detecteren en er overheen kan rijden om deze op te pakken. Tenslotte wordt de nieuwe functionaliteit beveiligd door gebruik te maken van beveiligingsopties welke geboden worden door DDS in ROS 2.

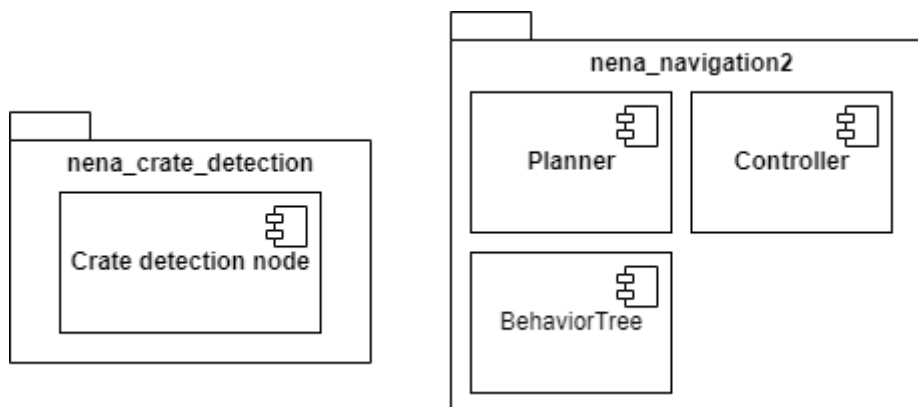
In onderstaande afbeelding is een overzicht te zien van de werking van de te ontwikkelen functionaliteit.



Afbeelding 18, overzicht van de werking van de te ontwikkelen functionaliteit

*Deze functionaliteit bestaat al in de NeNa robot, vandaar dat de startsituatie hiervan uitgaat

**Met positie + offset wordt de positie bedoelt waarop de robot dient te staan als deze recht tegenover het krat staat



Afbeelding 19, structuur van de packages

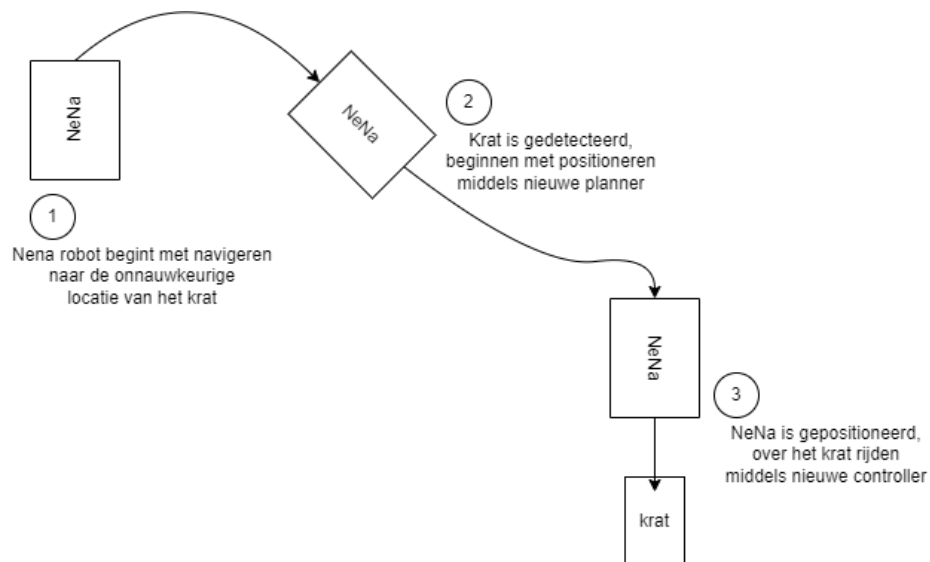
De nieuwe functionaliteit wordt in de bestaande packages geïmplementeerd. Zo is de nieuwe functionaliteit met Git eenvoudig te mergen op de main branch. De oude functionaliteit die niet meer gebruikt wordt, wordt verwijderd uit de packages. In bovenstaande afbeelding is een overzicht te zien van de packages waar tijdens de ontwikkeling aan gewerkt wordt, inclusief de onderdelen.

Detectiefunctionaliteit

Voor de detectiefunctionaliteit wordt er een nieuwe ROS 2 node geschreven. Deze node verwerkt de LiDAR data met als doel het krat te detecteren. Bij een detectie publiceert de node de locatie van het krat. De node wordt gestart met het 'ros run' commando.

Navigatiefunctionaliteit

De navigatiefunctie bestaat uit meerdere fasen. In onderstaande afbeelding zijn deze fasen gevisualiseerd.



Afbeelding 20, overzicht van de navigatiefasen

Voor de eerste navigatiefase wordt de standaardfunctionaliteit van Nav2 gebruikt. Het initiële doel wordt via RViz gegeven aan Nav2. Als de detectie node het krat detecteert, wordt naar de volgende fase overgegaan.

In de tweede navigatiefase wordt via de BehaviorTree overgeschakeld op de nieuwe planner. Deze planner is gebaseerd op een standaardimplementatie van Nav2. Deze implementatie wordt aangepast zodat de NeNa robot zichzelf op een configureerbare afstand recht voor het krat positioneert. Als de NeNa robot dit doel heeft bereikt, begint de derde en laatste navigatiefase.

Tijdens de laatste navigatiefase rijdt de NeNa robot over het krat heen. De NeNa robot is in de vorige fase recht voor het krat gepositioneerd waardoor deze alleen nog vooruit hoeft te rijden. Hiervoor wordt een al beschikbare planner gebruikt die een recht pad plant naar het krat. Daarnaast wordt een standaard controller van Nav2 gebruikt om dit pad te volgen. Om te voorkomen dat de planner en controller het krat als obstakel zien, wordt de costmap plugin aangeroepen om de representatie van het krat in de costmap te legen.

Security

Om de onderzochte beveiligingsopties te demonstreren, worden deze toegepast in de nieuwe functionaliteit. In het adviesrapport wordt de ervaring beschreven die hiermee is opgedaan en wordt de implementatie toegelicht. De verschillende beveiligingsopties voor authenticatie en geauthentiseerde versleuteling van berichten, de access control opties en inrichting van de PKI-infrastructuur worden getest.

5.2 Technisch ontwerp

In deze paragraaf wordt het technische ontwerp van het te ontwikkelen prototype toegelicht.

5.2.1 Functionaliteit

Detectiefunctionaliteit

Het krat wordt op basis van goniometrie gedetecteerd. Om tot een detectie te komen zijn een aantal stappen nodig.

De eerste stap is het omzetten van de afstandsmetingen naar punten ten opzichte van de LiDAR scanner. In elke meting wordt de afstand, de beginhoek van de scan en de stap in radialen voor elke meting ten opzichte van de LiDAR scanner gegeven. Zo is voor elke meting een punt te berekenen. De coördinaten van een punt worden gegeven met de formules $x = afstand * \cos(\theta)$ en $y = afstand * \sin(\theta)$.

Op basis van de afstand tussen elk punt is te bepalen of deze aan een huidige of nieuwe lijn toebehoort. Zolang de afstand van een punt tot een vorig punt kleiner is dan een bepaalde waarde, wordt deze beschouwd als een lijn. Dit houdt in dat er ook een samengestelde lijn kan worden gevormd die uit meerdere lijnen bestaat, zie onderstaande afbeelding.



Afbeelding 21, visualisatie van het krat als een samengestelde lijn

Om een samengestelde lijn te ontleden in lijnen wordt het Iterative End-Point Fit algoritme gebruikt. Dit algoritme geeft een verzameling met verzamelingen van lijnen. De verzamelingen waar twee lijnen in voorkomen, zijn alleen relevant om een detectie op te proberen te maken.

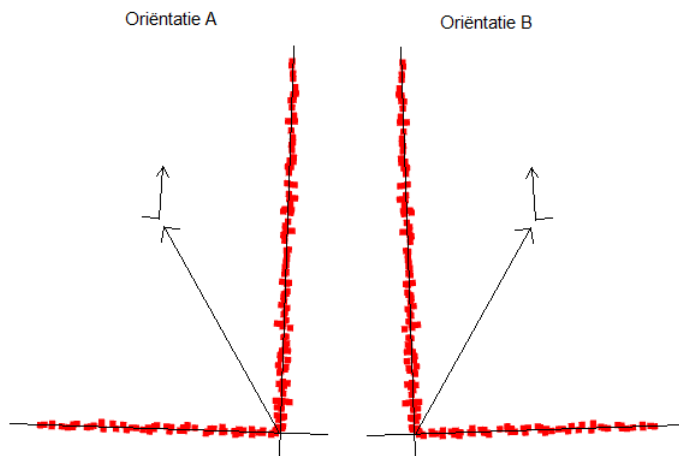
Voor alle verzamelingen van twee lijnen wordt de lengte van beide lijnen berekend met de formule $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Door de lengte van beide lijnen te berekenen kan bepaald worden of deze overeenkomen met de zijden van het krat.

Van twee lijnen die overeenkomen met de zijden van het krat worden lineaire regressielijnen opgesteld. Deze lineaire regressielijnen volgen de standaard lineaire formule $y = ax + b$, met a als richtingscoëfficiënt en b als beginwaarde. De a waarde wordt gegeven door de formule

$$a = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$
 met n als aantal coördinaten. De b waarde wordt gegeven middels de formule
$$b = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}$$
 met n als aantal coördinaten.

De hoek van deze twee lineaire regressielijnen dient +/- 90 graden te zijn. De hoek wordt gegeven met de formule $\theta = \left| \tan^{-1} \left(\frac{a_1 - a_2}{1 + a_1 * a_2} \right) \right|$. Als de berekende hoek binnen een marge van 90 graden valt, is er definitief vast te stellen dat er een krat is gedetecteerd. Vervolgens dient het middelpunt van het krat, de locatie, berekend te worden.

Het middelpunt van het krat is te berekenen op basis van het snijpunt van de twee lineaire regressielijnen, de hoek, oftewel oriëntatie, van het krat ten opzichte van de LiDAR scanner en de hoek van het snijpunt naar het middelpunt. De x-coördinaat van het snijpunt wordt gegeven door de formule $x = \frac{b_2 - b_1}{a_1 - a_2}$. De y-coördinaat is te verkrijgen door het substitueren van de berekende x-coördinaat in de formule van een van de lineaire regressielijnen. De afstand van het snijpunt tot het middelpunt wordt gegeven door de formule $d = \sqrt{\left(\frac{\text{breedte krat}}{2}\right)^2 + \left(\frac{\text{lengte krat}}{2}\right)^2}$. De hoek van het krat ten opzichte van de LiDAR scanner is gelijk aan de inverse tangens van de richtingscoëfficiënt van de langste lijn of zijde, oftewel $\theta = \tan^{-1}(a_{\text{langste zijde}})$. De formule voor het berekenen van de hoek van het snijpunt naar het middelpunt is afhankelijk van de oriëntatie van het krat, zoals in onderstaande afbeelding te zien is.



Afbeelding 22, de mogelijke oriëntaties van het krat

Voor oriëntatie A geldt de formule

$\theta_{\text{middelpunt}} = \theta_{\text{krat}} - \tan^{-1} \left(\left(\frac{1}{2} \text{ breedte krat} \right) / \left(\frac{1}{2} \text{ lengte krat} \right) \right)$. Voor oriëntatie B geldt de formule $\theta_{\text{middelpunt}} = \theta_{\text{krat}} + \tan^{-1} \left(\left(\frac{1}{2} \text{ breedte krat} \right) / \left(\frac{1}{2} \text{ lengte krat} \right) \right)$. De coördinaten voor het middelpunt zijn daarna te berekenen met de formules $x_{\text{middelpunt}} = x_{\text{snijpunt}} + d * \cos(\theta_{\text{middelpunt}})$ en $y_{\text{middelpunt}} = y_{\text{snijpunt}} + d * \sin(\theta_{\text{middelpunt}})$.

Aangezien deze coördinaten relatief zijn aan de LiDAR scanner, worden deze coördinaten gepubliceerd in het tf2 frame van de LiDAR scanner. Zo is de locatie van het krat te visualiseren in RViz. De locatie van het krat ten opzichte van het odom frame wordt berekend middels de tf2 library en als doel gepubliceerd voor de navigatie. Hiervoor wordt het 'goal_pose' topic gebruikt.

Navigatiefunctie

De nieuwe planner wordt op de NavFn planner gebaseerd [26]. Dit is een standaardplanner van Nav2 die ook voor de eerste navigatiefase gebruikt wordt om naar de onnauwkeurige locatie van het krat te rijden. Deze planner wordt aangepast om het doel te verplaatsen naar een configureerbare afstand ten opzichte van het doel dat van het 'goal_pose' topic komt. Zo plant deze controller een pad naar een positie recht voor het krat. Daarnaast wordt het coordinate frame waarin genavigeerd wordt, aangepast naar het odom frame. Dit zorgt ervoor dat de navigatie zijn nauwkeurigheid behoudt door de vertaalslag van het odom frame naar het map frame te voorkomen.

Voor het docken in de volgende en tevens laatste navigatiefase wordt een planner gebruikt die reeds in de 'nena_navigation2' package beschikbaar is. Deze planner plant een rechtstreeks pad naar het huidige doel. Een standaardcontroller van Nav2 volgt dit pad om zo over het krat heen te rijden. Ook in deze planner wordt het coordinate frame gewijzigd naar het odom frame, in plaats van het map frame. De costmap wordt aangeroepen om de representatie van het krat te verwijderen uit de costmap, zodat de planner en controller het krat niet als obstakel zien.

Het omschakelen van de planners in de verschillende navigatiefasen wordt verzorgd door de BehaviorTree. In de BehaviorTree wordt middels XML-componenten het gedrag voor de verschillende navigatiefasen gedefinieerd.

Security

Naast de ontwikkelen functionaliteit wordt er aandacht besteedt aan het beveiligen van de te ontwikkelen node(s) door gebruik te maken van de beveiligingsopties die middels DDS geboden worden. Hiervoor worden de verschillende mogelijkheden voor authenticatie en geauthentiseerde versleuteling gebruikt. Voor de access control functionaliteit wordt de configuratie zo ingesteld dat de te ontwikkelen nodes alleen toegang hebben tot de benodigde topics. Op basis van de ervaring die hiermee wordt opgedaan wordt een adviesrapport opgesteld. In dit adviesrapport wordt beschreven hoe deze beveiligingsopties in toekomstige ROS 2 projecten gebruikt kunnen worden.

5.2.2 Requirements

De volgende requirements zijn opgesteld:

Tabel 1, requirements

#	Beschrijving	Functioneel/ Niet functioneel	MoSCoW
1	De NeNa robot moet een krat kunnen herkennen uit LiDAR data en van dit krat exact de positie en oriëntatie kunnen bepalen.	F	M
2	De NeNa robot moet zichzelf recht tegenover het krat kunnen positioneren	F	M
3	De collision detection functionaliteit in de NeNa robot moet het krat niet als obstakel zien tijdens het oppakken	F	M
4	De NeNa robot moet over het krat heen kunnen rijden	F	M
5	De NeNa robot kan het krat oppakken	F	W

5.2.3 User stories

Op basis van deze requirements zijn de volgende user stories geschreven:

Tabel 2, user stories

#	Beschrijving	Story points	MoSCoW
1	Als gebruiker wil ik dat de ROS 2 node de LiDAR data uit kan lezen, zodat deze data verwerkt kan worden	2	M
2	Als gebruiker wil ik dat de ROS 2 node een krat kan detecteren uit de LiDAR data, zodat van het krat de positie en oriëntatie berekend kan worden	6	M
3	Als gebruiker wil ik dat de ROS 2 node de positie en oriëntatie van het krat kan berekenen, zodat er naar deze positie (plus een bepaalde offset) genavigeerd kan worden	8	M
4	Als gebruiker wil ik dat de planner een route kan plannen naar de Pose* van het krat met een bepaalde offset, zodat de NeNa robot recht tegenover het krat komt te staan	3	M
5	Als gebruiker wil ik dat de controller de geplande route volgt, zodat de NeNa robot op de juiste positie (en met de juiste oriëntatie) uitkomt	2	M
6	Als gebruiker wil ik dat de navigatie overschakelt naar de planner (US.4) zodra er een krat is gedetecteerd, zodat de NeNa robot hier uiteindelijk naartoe kan navigeren	4	M
8	Als gebruiker wil ik dat de NeNa robot over het krat heenrijdt zodra deze er recht tegenover staat, zodat deze het krat uiteindelijk op kan pakken	3	M
9	Als gebruiker wil ik dat de navigatie het krat niet als obstakel ziet, zodat de NeNa robot het krat op kan pakken zonder dat de collision detection dit voorkomt	4	M
10	Als gebruiker wil ik dat de NeNa robot het krat oppakt zodra deze recht boven het krat staat, zodat de NeNa zijn taak kan uitvoeren	3	W

*Pose is de combinatie van positie en oriëntatie

Eén story point staat gelijk aan een halve dag.

5.3 Realisatie

In deze paragraaf wordt de implementatie van het ontwerp toegelicht. De broncode is als bijlage bij dit afstudeerverslag toegevoegd.

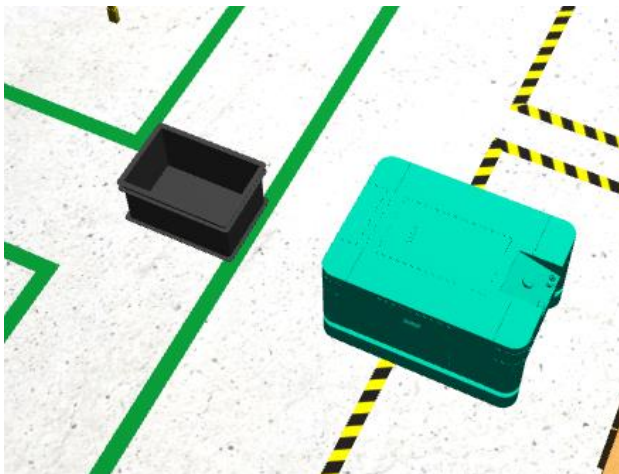
5.3.1 Detectiefunctionaliteit

De detectiefunctionaliteit is volledig volgens het ontwerp geïmplementeerd.

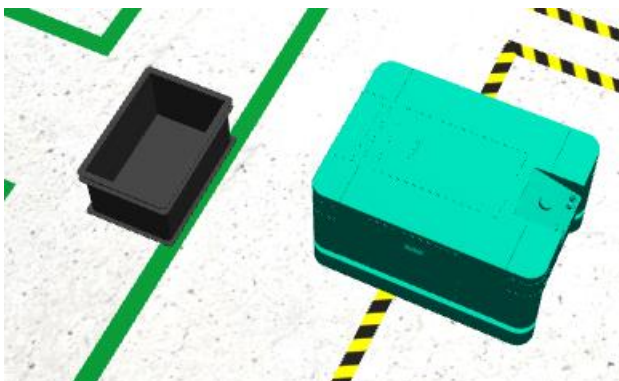
Voor de detectiefunctionaliteit is er een nieuwe ROS node ontwikkelt. Deze node probeert een krat uit de binnenkomende LiDAR data te detecteren. De detectiefunctionaliteit kan eenvoudig gewijzigd worden om andere eenvoudige objecten te detecteren. De variabelen die in de detectiefunctionaliteit gebruikt worden, kunnen aangepast worden om deze strikter in te stellen of juist meer tolerantie te geven.

Om de detectiefunctionaliteit efficiënter te maken is er tijdens de ontwikkeling in overleg met de bedrijfsbegeleider besloten om niet de gehele LiDAR data te verwerken. In de implementatie is het mogelijk een hoek en bereik te geven. Het opgegeven bereik wordt verwerkt, met de opgegeven hoek als midden.

Voor de nauwkeurigheid van de detectie zijn er twee metingen gedaan met twee oriëntaties, zoals in de onderstaande afbeeldingen te zien is.

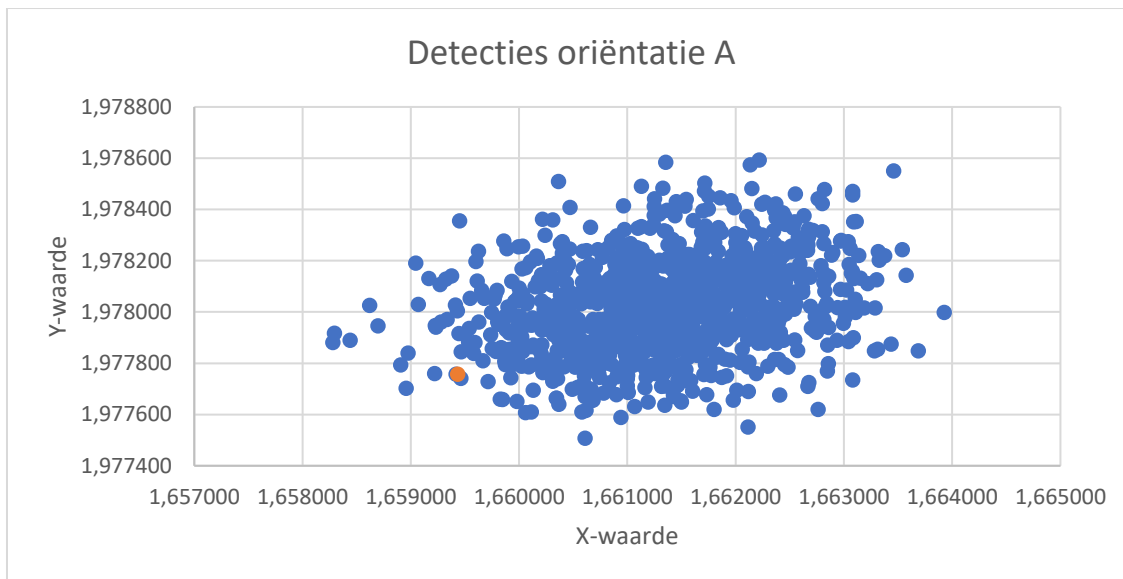


Afbeelding 23, oriëntatie A

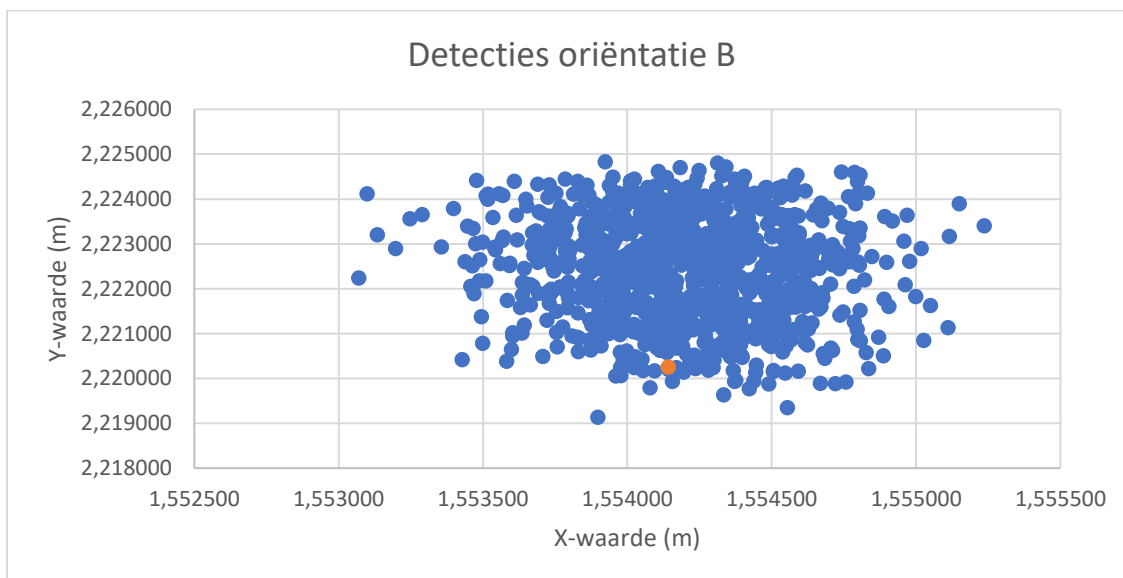


Afbeelding 24, oriëntatie B

De resultaten van de metingen zijn te zien in de volgende grafieken.



Grafiek 1



Grafiek 2

Voor de metingen, zie bijlage 3. Uit deze metingen blijkt dat de detecties dusdanig nauwkeurig zijn dat dit om minder dan centimeters gaat. Voor de detectiefunctie wordt dit als meer dan voldoende geacht. De afwijkingen in de metingen zijn te wijten aan het feit dat de samengestelde lijnen aan het uiteinde van de eerste en het begin van de tweede lijn niet helemaal recht zijn. Dit komt doordat er op de hoek van het krat een grote verzameling detecties zijn. Dit kan verholpen worden door de laatste punten van de eerste lijn en de eerste punten van de tweede lijn weg te laten bij het opstellen van de lineaire regressielijnen.

5.3.2 Navigatiefunctiefunctionaliteit

Voor de tweede navigatiefase, het positioneren van de NeNa robot recht voor het krat, is voor de implementatie het ontwerp gevolgd. De standaard NavFn planner is gebruikt. Hierin is het navigatie coordinate frame aangepast naar het odom frame. Daarnaast plant de planner het doel op een configureerbare afstand van het middelpunt van het krat.

Bij het ontwikkelen van de functionaliteit voor de tweede navigatiefase, het positioneren voor het krat, is bij het testen gebleken dat de te configureren nauwkeurigheid van de standaardcontroller niet erg hoog in te stellen is. Na veel testen en tweaken is de hoogste nauwkeurigheid met betrekking tot de positie die bereikt kon worden 1,8 cm. Met lagere waardes was de NeNa robot niet in staat consistent het doel te bereiken en liep deze vast. De oriëntatie was daarentegen zeer nauwkeurig in te stellen op 0,01 radiaal ($\approx 0,6$ graden). Dit valt nog wel binnen de marge van 2,5 cm aan weerszijde van het krat. Voor het lectoraat Mechatronica is het voor volgende projecten waar Nav2 in wordt gebruikt, waardevol om zich daarvan bewust te zijn.

De standaardcontroller was in de laatste navigatiefase, het docken, niet in staat om rechtdoor over het krat te rijden zonder deze aan te raken. In overleg met de bedrijfsbegeleider is er besloten om een nieuwe controller te ontwikkelen die stuurt op basis van de LiDAR data van het krat. Hiervoor worden de detecties recht voor het krat omgezet naar punten. De detectiefunctiefunctionaliteit wordt gebruikt om de punten om te zetten naar samengestelde lijnen. De robot zal in eerste instantie bij het begin van het docken de twee lijnen van het krat zien die de zijden voorstellen. De tweede lijn is de referentielijn die gevolgd wordt. Door van deze lijn een lineaire regressielijn te maken, is er met de b -waarde van de lineaire regressielijn te bepalen of er bijgestuurd moet worden. Door middel van testen is gevonden dat de optimale b -waarde, -0.14 bedraagt. Als de gedetecteerde b -waarde buiten een marge van 0,25 cm ligt, wordt er bijgestuurd. Zo is de NeNa robot in staat om zelfs met een onnauwkeurige eindpositie bij de tweede navigatiefase toch in staat om autonoom en consistent over het krat heen te rijden. Zodoende is ervoor gekozen om de nauwkeurigheid in de controllerconfiguratie iets lager te zetten naar 2,0 cm, in plaats van 1,8 cm. Hierdoor bereikt de NeNa robot eerder het positioneringsdoel. De nieuwe controller maakt geen gebruik van de costmap. Daardoor is het niet nodig geweest om de costmap aan te roepen om de representatie van het krat te verwijderen.

5.3.3 Adviesrapport security

Doordat het implementeren van de detectie- en navigatiefunctiefunctionaliteit meer tijd heeft gekost dan gepland, is het niet gelukt om een adviesrapport op te stellen waarin wordt toegelicht hoe de beveiligingsopties in te stellen zijn. Het onderzoek naar de beveiliging in ROS 2 middels DDS biedt een goed overzicht van de mogelijkheden op dit gebied.

5.4 Testplan

Het testen van de functionaliteit is gedaan middels RViz. Hiervoor is gekozen omdat het duidelijk te zien is dat de functionaliteit naar behoren werkt. Als alle functionaliteit met betrekking tot het detecteren van het krat en de navigatie naar en over het krat, zijn de testen te verifiëren door alle functionaliteit samen te laten werken, oftewel door middel van een systeemtest. Als er functionaliteit mist, wordt de geïmplementeerde functionaliteit geverifieerd door een afzonderlijke test van deze functionaliteit.

De uitkomsten van de testen zijn bij dit afstudeerverslag gevoegd als bijlage.

Test Case	1
Requirement	1
Beschrijving	De node kan een krat herkennen in een (virtuele) omgeving waar een krat staat
Input	LiDAR data van de NeNa robot
Verwachting	De node detecteert het krat
Uitkomst	Werkend, zie bijlage 4
Geslaagd	Ja

Test Case	2
Requirement	1
Beschrijving	De node geeft geen vals positieven als deze rondrijdt in een (virtuele) omgeving waar geen krat aanwezig is
Input	LiDAR data van de NeNa robot
Verwachting	De node detecteert geen krat
Uitkomst	Werkend, zie bijlage 4
Geslaagd	Ja

Test Case	3
Requirement	1
Beschrijving	De node kan de exacte positie van een gedetecteerd krat berekenen
Input	LiDAR data van de NeNa robot
Verwachting	De exacte positie kan berekend worden
Uitkomst	Geïmplementeerd, zie bijlage 4
Geslaagd	Ja

Test Case	4
Requirement	2
Beschrijving	De navigatie planner kan een route plannen naar de juiste positie (+ offset) en oriëntatie
Input	Positie van het krat
Verwachting	De navigatie planner plant een geldige route
Uitkomst	Geïmplementeerd, zie bijlage 4
Geslaagd	Ja

Test Case	5
Requirement	2
Beschrijving	Zodra de BehaviorTree het signaal krijgt dat er een krat is gedetecteerd, wordt de planner geactiveerd
Input	Krat detectie
Verwachting	De planner wordt ingeschakeld bij een detectie op het krat
Uitkomst	Geïmplementeerd, inclusief de nieuwe controller, zie bijlage 4
Geslaagd	Ja

Test Case	6
Requirement	3
Beschrijving	Voor het oppakken van het krat is het noodzakelijk dat het krat niet als obstakel wordt gezien
Input	-
Verwachting	De NeNa robot ziet het gedetecteerde krat niet als obstakel tijdens het docken met het krat
Uitkomst	Overbodig doordat er een nieuwe controller is geschreven voor het docken
Geslaagd	Ja

Test Case	7
Requirement	4
Beschrijving	De NeNa robot kan, wanneer deze op de juiste positie staat en goed georiënteerd is, over het krat heenrijden en deze oppakken
Input	-
Verwachting	De NeNa robot kan het krat oppakken
Uitkomst	Niet geïmplementeerd
Geslaagd	Nee

6. Reflectie

In dit hoofdstuk wordt gereflecteerd op het afstudeerproject.

6.1 Thuiswerken

Gedurende het afstudeerproject is er voornamelijk thuis gewerkt. Hoewel het mogelijk was om op het lectoraat te werken, is hier geen gebruik van gemaakt. De communicatie met mijn bedrijfsbegeleider was naar mijn mening dusdanig goed, dat ik hier ook geen behoefte aan heb gehad. De betrokkenheid van mijn bedrijfsbegeleider was een motiverende factor voor het thuiswerken.

Met het thuiswerken ben ik in staat geweest efficiënter te werken door het gebruik van meerdere beeldschermen en een veel snellere werkomgeving in vergelijking met mijn laptop. In het begin van het afstudeerproject is gekozen om voor het onderzoek en de ontwikkeling gebruik te maken van virtuele machines, is dit achteraf een minder handige keuze geweest. Doordat VirtualBox niet optimaal werkt met dezelfde virtuele machines op verschillende fysieke machines (desktop en laptop) gaf dit een aantal problemen. De virtuele machines die ik heb gebruikt hadden snapshots. Een voor de schone Ubuntu 20.04 installatie, een daarop gebaseerde snapshot met de volledige ROS 2 installatie en ontwikkelomgeving. Echter worden deze snapshots in de standaardmap geplaatst door VirtualBox. Hierdoor stonden deze niet op de externe SSD die ik gebruikte voor het afstuderen. Dit corrigeren was enigszins omslachtig en heeft ertoe geleid dat ik mijn voortgang van de implementatie niet op locatie heb kunnen laten zien aan mijn afstudeer- en bedrijfsbegeleider.

Verder werkte de hardware versnellingsoptie niet goed samen met de simulatietool Gazebo. Het aanzetten van deze optie resulteerde in onzichtbare textures in Gazebo. Door deze complicaties was het werken in een virtuele machine achteraf niet ideaal. Dit zorgde voor een lage frame rate, waardoor het programma niet zo soepel te bedienen was.

6.2 Onderzoek

Tijdens het onderzoek heb ik veel geleerd over de werking van ROS. Robots worden steeds breder en vaker ingezet voor allerlei taken. Daarmee wordt het ook interessanter voor cybercriminelen, om industriële productielijnen als doelwit te nemen. Na het afronden van mijn studie kan ik beginnen als Junior Cybersecurity Consultant bij Capgemini, in een vakgebied waar de opgedane kennis zeker van meerwaarde kan zijn, met oog op het onderzoek met betrekking tot DDS in ROS 2.

6.3 Ontwerp

Het maken van het ontwerp is goed gelukt. Op basis van het verrichte onderzoek ben ik in staat geweest om een duidelijk ontwerp te ontwikkelen waarvan in de implementatiefase nauwelijks van is af hoeven te wijken. Voor de navigatiefunctie is er meer functionaliteit toegevoegd dan oorspronkelijk in het ontwerp is opgenomen. De functionaliteit die in het ontwerp is beschreven hoefde daarvoor aangepast te worden.

6.4 Implementatie

De detectiefunctie heeft een week langer geduurd dan gepland. Hiervoor is een week van de buffer sprint gebruikt. Omdat er in de planningsfase al rekening is gehouden met onverwachte vertragingen door een buffer sprint in te plannen, heeft dit niet geleid tot ernstige vertraging.

De functionaliteit met betrekking tot de navigatie bleek veel meer tijd te kosten dan hiervoor gepland was. Het nauwkeurig positioneren van de robot in de tweede navigatiefase bleek een grotere opgave dan gedacht. De meeste tijd is in het tweaken van de controller gaan zitten om te testen hoe nauwkeurig deze kan zijn en het definiëren van de BehaviorTree. Met een nauwkeurigheid van 1,8 cm is het mogelijk om consistent de NeNa robot voor het krat te positioneren. De oriëntatie van de eindpositie was nauwkeurig op ongeveer 0,6 graden. Echter het rijden over het krat heen met de huidige planner was geen succes. Deze bleef, ondanks dat deze theoretisch gezien alleen rechtdoor hoefde te rijden, toch het krat raken. In overleg is besloten een nieuwe controller te ontwikkelen voor de derde en laatste navigatiefase. Door de controller gebruik te laten maken van een stuk van de detectiefunctie is het uiteindelijk gelukt om de NeNa robot over het krat te laten rijden. Hierdoor is het niet meer nodig geweest om de costmap te legen zodat het krat niet als obstakel wordt gezien.

Het werkend krijgen van de BehaviorTree kostte enige moeite. Uiteindelijk is een goed werkende BehaviorTree opgezet, maar de documentatie was niet zo duidelijk als gewenst. Er is tijdens de ontwikkeling met de bedrijfsbegeleider afgesproken om hier geen focus meer op te leggen en de aandacht te richten op het implementeren van de afzonderlijke stukken navigatiefunctie voor de verschillende fasen. In mijn vrije tijd heb ik dit alsnog werkend gekregen waardoor het eindresultaat als compleet systeem gedemonstreerd kon worden.

Daar kwam bij dat het testen van de functionaliteit niet zonder problemen verliep. De volgende problemen kwamen regelmatig voor:

1. Gazebo start niet volledig op, zie bijlage 5
2. Gazebo start wel, maar initialiseert niet goed, zie bijlage 6. Hierdoor werkt de simulatie niet.
3. Het lifecycle manager proces van Nav2 crasht, zie bijlage 7. Het lifecycle proces zorgt voor het opstarten en afsluiten van nodes die binnen Nav2 gebruikt worden, waaronder planners, controllers en BehaviorTree functionaliteit.
4. RViz kan geen doel naar Nav2 versturen doordat het 'bt_navigator' proces is gecrasht, zie bijlage 8.

Deze complicaties waren te verhelpen door Gazebo of RViz opnieuw te starten. Dit toont aan dat het probleem bij Gazebo/RViz/Nav2 lag. Helaas is er geen tijd geweest om dit beter te onderzoeken.

Al met al heeft dit wel voor enige vertraging gezorgd in de implementatiefase.

7. Adviesrapport security

Zoals in de implementatie en reflectie is toegelicht, is het niet gelukt om een adviesrapport over de implementatie van de beveiligingsopties op te stellen in de geplande tijd.

8. Conclusie

In dit afstudeerproject is onderzocht hoe de NeNa robot consistent zijn doel kan detecteren, hier overheen kan rijden en beveiligd kan worden. Uit het onderzoek is gebleken dat voor het detecteren van het krat het best een detectie op basis van goniometrie gemaakt kan worden. Het onderzochte ICP-algoritme is een geschikte kandidaat voor het lokaliseren van de robot in een omgeving of voor het detecteren van complexere objecten dan het krat. Voor de navigatie is de beste keuze gebleken om van de Nav2 library gebruik te maken.

Tijdens de implementatiefase is de NeNa voorzien van de onderzochte functionaliteit. Hiermee is de volledige functionaliteit van de NeNa robot zo goed als afgerond. Er rest alleen nog het integreren van de functionaliteit voor het oppakken van het krat. Hier is de afstudeerder niet aan toegekomen. Dit was geen vereiste functionaliteit.

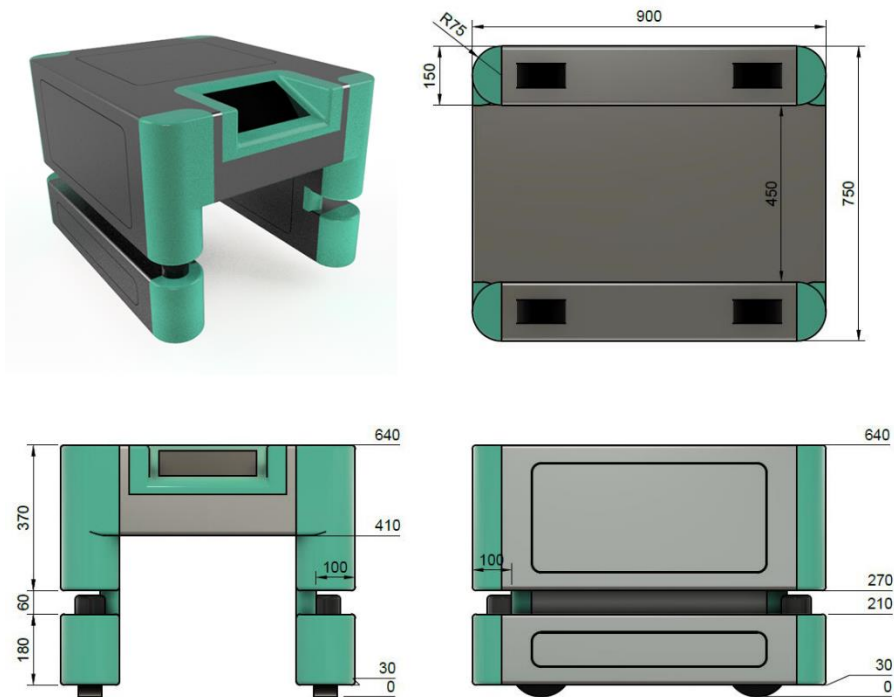
De detectie- en navigatiefunctie is door het lectoraat te hergebruiken in ROS projecten. Deze functionaliteit is zodanig ontwikkeld dat de werking eenvoudig aangepast kan worden door gebruik te maken van het configuratiebestand van Nav2 voor de navigatiefunctie en de variabelen in de detectiefunctie.

9. Bronnen

1. Saxion. (z.d.). Next Generation Navigation (NeNa). Hogeschool Saxion. Geraadpleegd op 16 januari 2022, van <https://www.saxion.nl/onderzoek/smart-industry/mechatronica/next-generation-navigation>
2. Robot Operating System. (2022, 7 januari). In Wikipedia. https://en.wikipedia.org/wiki/Robot_Operating_System
3. Open Robotics. (2021, 26 oktober). Understanding ROS 2 nodes — ROS 2 Documentation: Foxy documentation. ROS 2 Documentation. Geraadpleegd op 16 januari 2022, van <https://docs.ros.org/en/foxy/Tutorials/Understanding-ROS2-Nodes.html>
4. Data Distribution Service. (2021, 14 december). In Wikipedia. https://en.wikipedia.org/wiki/Data_Distribution_Service
5. Object Management Group. (z.d.). About the DDS Security Specification Version 1.1. OMG. Geraadpleegd op 16 januari 2022, van <https://www.omg.org/spec/DDS-SECURITY/1.1/About-DDS-SECURITY/>
6. Nav2 — Navigation 2 1.0.0 documentation. (2021, 7 december). Nav2. Geraadpleegd op 16 januari 2022, van <https://navigation.ros.org/>
7. S. Macenski, F. Martín, R. White, J. Clavero. The Marathon 2: A Navigation System. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.
8. Open Source Robotics Foundation. (z.d.). Gazebo. Gazebo. Geraadpleegd op 16 januari 2022, van <http://gazeboosim.org/>
9. Open Source Robotics Foundation. (2018, 14 december). Gazebo : Tutorial : ROS 2 overview. Gazebo. Geraadpleegd op 16 januari 2022, van http://gazeboosim.org/tutorials?tut=ros2_overview
10. AWS Robotics. (2021, 29 juli). GitHub - aws-robotics/aws-robomaker-small-warehouse-world. GitHub. Geraadpleegd op 16 januari 2022, van <https://github.com/aws-robotics/aws-robomaker-small-warehouse-world>
11. Hershberger, D., Gossow, D., Faust, J., & Woodall, W. (2018, 16 mei). rviz - ROS Wiki. ROS Wiki. Geraadpleegd op 16 januari 2022, van <http://wiki.ros.org/rviz>
12. JetBrains. (2021, 29 augustus). ROS2 setup tutorial | CLion. JetBrains CLion. Geraadpleegd op 19 december 2021, van <https://www.jetbrains.com/help/clion/ros2-tutorial.html#build-ws>
13. Iterative closest point. (2021, 22 mei). In Wikipedia. https://en.wikipedia.org/wiki/Iterative_closest_point
14. Ho, N. (2011, 11 september). Finding optimal rotation and translation between corresponding 3D points | Nghia Ho. Nghiaho. Geraadpleegd op 16 januari 2022, van http://nghiaho.com/?page_id=671
15. Vongbunyong, S., Thamrongaphichartkul, K., Worrasittichai, N., & Takutrueta, A. (2021). Automatic precision docking for autonomous mobile robot in hospital logistics - case-study:battery charging. IOP Conference Series: Materials Science and Engineering, 1137(1). <https://doi.org/10.1088/1757-899x/1137/1/012060>
16. Lumen Learning, OpenStax. (z.d.). Unit Circle: Sine and Cosine Functions | Precalculus II. Lumen Learning. Geraadpleegd op 16 januari 2022, van <https://courses.lumenlearning.com/precalc2two/chapter/unit-circle-sine-and-cosine-functions/>
17. Ramer–Douglas–Peucker algorithm. (2022, 4 januari). In Wikipedia. https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

18. Shi, Z., & Wu, Z. (2013). Maneuver Target Detection Method with Iterative Endpoint Fitting Assisted. Lecture Notes in Electrical Engineering, 839–846.
https://doi.org/10.1007/978-3-642-38466-0_93
19. Navigation Plugins — Nav2. (z.d.). ROS Nav2. Geraadpleegd op 19 december 2021, van <https://navigation.ros.org/plugins/index.html>
20. Hadabot Blog - A ROS2 Nav2 navigation tf2 tutorial using turtlesim. (2020, 27 september). Blog for Hadabot. Geraadpleegd op 16 januari 2022, van <https://blog.hadabot.com/ros2-navigation-tf2-tutorial-using-turtlesim.html>
21. Principle of least privilege. (2021, 28 december). In Wikipedia.
https://en.wikipedia.org/wiki/Principle_of_least_privilege
22. Bit-flipping attack. (2021, 30 maart). In Wikipedia. https://en.wikipedia.org/wiki/Bit-flipping_attack
23. Object Computing. (z.d.). OpenDDS. OpenDDS. Geraadpleegd op 19 december 2021, van <https://opendds.org/>
24. oci-labs (Object Computing). (2021, 16 maart). GitHub - oci-labs/rmw_opendds. GitHub. Geraadpleegd op 19 december 2021, van https://github.com/oci-labs/rmw_opendds
25. GitHub - ros2/sros2. (2021, 6 oktober). GitHub. Geraadpleegd op 19 december 2021, van <https://github.com/ros2/sros2>
26. NavFn Planner — Navigation 2 documentation. (2021, 2 september). Nav2. Geraadpleegd op 16 januari 2022, van <https://navigation.ros.org/configuration/packages/configuring-navfn.html>

10. Bijlagen



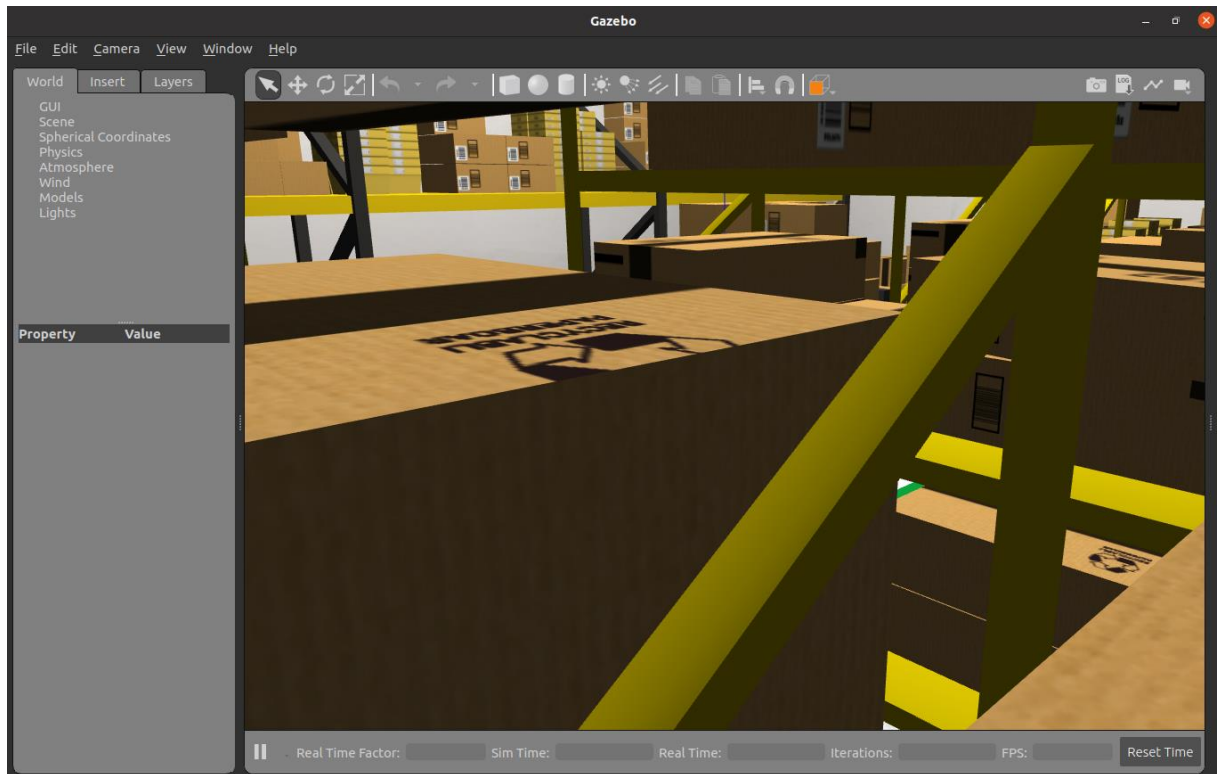
- 1.
2. Zie bijgevoegd bestand 'crate_datasheet.pdf'.
3. Zie bijgevoegd bestand 'Metingen.xlsx'.
4. Zie bijgevoegd bestand 'systeemtest.mp4'.

```

ros@ros: ~/nena_ws
g/crate_controller.yaml
gzserver-1 [INFO] [1642354069.189794030] [gazebo_ros2_control]: connected to service!! robot_state_publisher
gzserver-1 [INFO] [1642354069.191647383] [gazebo_ros2_control]: Rectieved urdf from param server, parsing...
gzserver-1 [INFO] [1642354069.200091124] [gazebo_ros2_control]: Loading joint: left_rear_crate_lift_joint
gzserver-1 [INFO] [1642354069.200154315] [gazebo_ros2_control]: Command:
gzserver-1 [INFO] [1642354069.200161208] [gazebo_ros2_control]: position
gzserver-1 [INFO] [1642354069.200241191] [gazebo_ros2_control]: State:
gzserver-1 [INFO] [1642354069.200249647] [gazebo_ros2_control]: position
gzserver-1 [INFO] [1642354069.200283851] [gazebo_ros2_control]: velocity
gzserver-1 [INFO] [1642354069.2003097542354066.642575334] [robot_state_publisher]:
gzserver-1 [INFO] [1642354069.200348821111] [robot_state_publisher]: joint
gzserver-1 [INFO] [1642354069.2003563242354066.642578961] [robot_state_publisher]: id be in front of the
gzserver-1 [INFO] [1642354069.200380021111] [robot_state_publisher]: :KingNavigation">
gzserver-1 [INFO] [1642354069.2003866242354066.642582408] [robot_state_publisher]:
gzserver-1 [INFO] [1642354069.20039131] [robot_state_publisher]:
gzserver-1 [INFO] [1642354069.2003960842354066.642585894] [robot_state_publisher]: .retries="2" name="Cr
gzserver-1 [INFO] [1642354069.20044739] [robot_state_publisher]: use goal="{goal}" pat
gzserver-1 [INFO] [1642354069.2004553066.679056986] [spawn_entity]: Spawn Entity s goal="{goal}" path="oint
gzserver-1 [INFO] [1642354069.20046021] [spawn_entity]: Loading entity name="ClearGlobalCos
gzserver-1 [INFO] [1642354069.20046946n] [spawn_entity]: Waiting for en
gzserver-1 [INFO] [1642354069.2004741866.680525974] [spawn_entity]: Waiting for se
gzserver-1 [INFO] [1642354069.20047879] [spawn_entity]: Waiting for se
gzserver-1 [INFO] [1642354069.2004833266.695150460] [spawn_entity]: Waiting for se
gzserver-1 [INFO] [1642354069.20048945] [spawn_entity]: Waiting for se
gzserver-1 [INFO] [1642354069.2004940566.695416879] [spawn_entity]: Waiting for se
gzserver-1 [INFO] [1642354069.200498474] [gazebo_ros2_control]: State:
gzserver-1 [INFO] [1642354069.200503412] [gazebo_ros2_control]: position
gzserver-1 [INFO] [1642354069.200508000] [gazebo_ros2_control]: position
gzserver-1 [INFO] [1642354069.200512539] [gazebo_ros2_control]: velocity
gzserver-1 [INFO] [1642354069.200517038] [gazebo_ros2_control]: effort
gzserver-1 [INFO] [1642354069.200689788] [gazebo_ros2_control]: Loading controller_manager
[INFO] [spawn_entity.py-5]: process has finished cleanly [pid 133269]
gzserver-1 [WARN] [1642354069.213892603] [gazebo_ros2_control]: Desired controller update period (0.01 s) is slower than the gazebo simulation per
(0.001 s).
gzserver-1 [INFO] [1642354069.214005118] [gazebo_ros2_control]: Loaded gazebo_ros2_control.
gzserver-1 [INFO] [1642354069.232256812] [turtlebot3_diff_drive]: Wheel pair 1 separation set to [0.633000m]
gzserver-1 [INFO] [1642354069.232336023] [turtlebot3_diff_drive]: Wheel pair 1 diameter set to [0.125000m]
gzserver-1 [INFO] [1642354069.233661350] [turtlebot3_diff_drive]: Subscribed to [/cmd_vel]
gzserver-1 [INFO] [1642354069.234889442] [turtlebot3_diff_drive]: Advertise odometry on [/odom]
gzserver-1 [INFO] [1642354069.235999404] [turtlebot3_diff_drive]: Publishing odom transforms between [odom] and [base_footprint]

```

- 5.



6.

```
[bt_navigator-8] [INFO] [1642359588.842892216] [bt_navigator]: Begin navigating from current location to (-0.56, 0.43)
[bt_navigator-8] [ERROR] [1642359588.873038506] [bt_navigator]: Action server failed while executing action callback: "send_goal failed"
[bt_navigator-8] [WARN] [1642359588.873916072] [bt_navigator]: [navigate_to_pose] [ActionServer] Aborting handle.
```

7.

```
[bt_navigator-8] [ERROR] [1642360139.034746217] [bt_navigator]: Action server failed while executing action callback: "send_goal failed"
[bt_navigator-8] [WARN] [1642360139.034827562] [bt_navigator]: [navigate_to_pose] [ActionServer] Aborting handle.
[bt_navigator-8] [INFO] [1642360139.831533478] [bt_navigator]: Begin navigating from current location to (-2.79, -0.31)
[bt_navigator-8] [ERROR] [1642360139.831533478] [bt_navigator]: process has died [pid 196058, exit code -11, cmd '/opt/ros/foxy/lib/nav2_bt_navigator/bt_navigator --ros-args -r __node:=bt_navigator --params-file /tmp/tmp1n6hacs -r /tf:=tf -r /tf_static:=tf_static'].
```

8.