Ontwerpdocument RobotarmSimulatie

WoR World

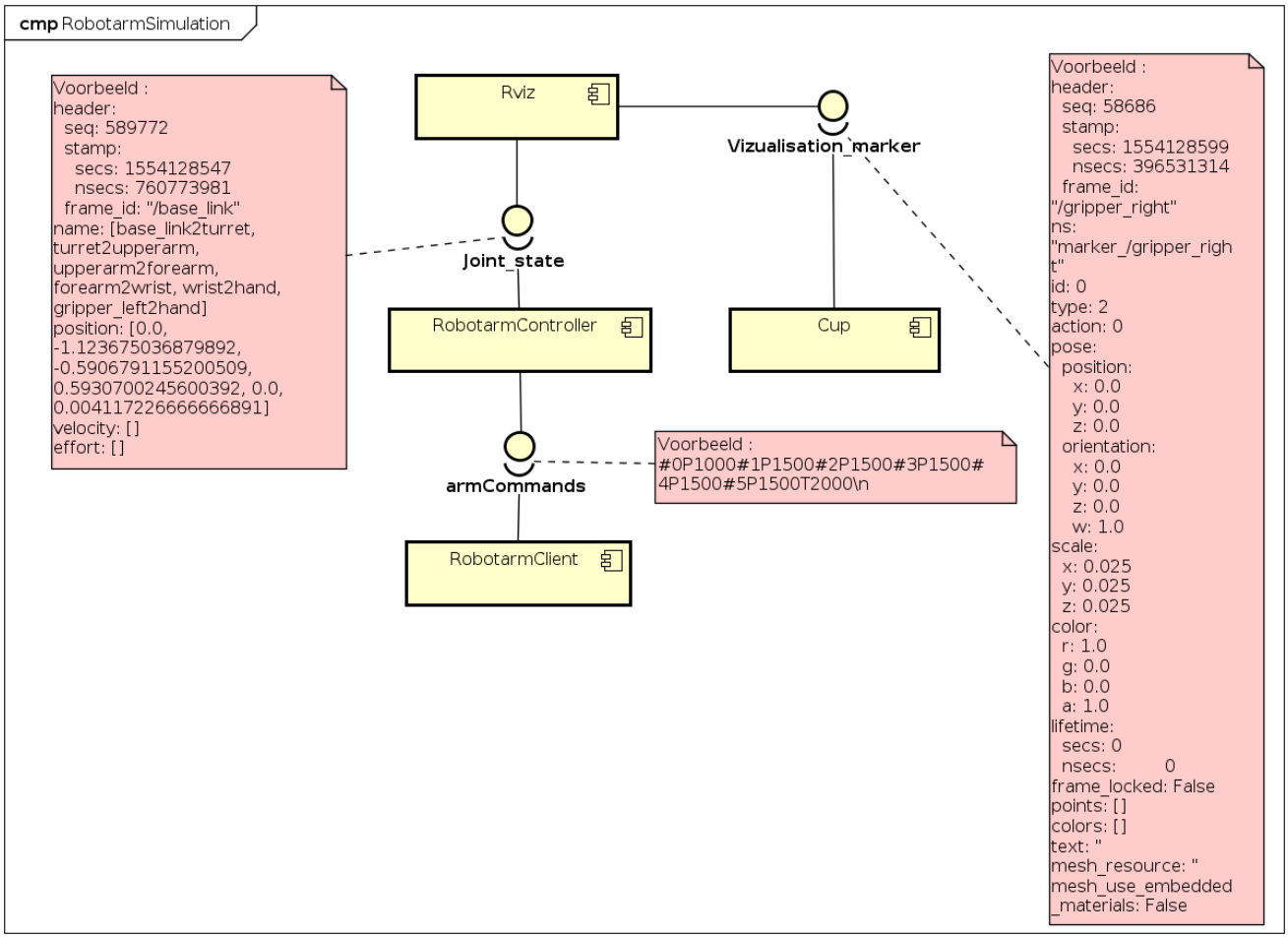
Dibran Dokter & Hendrik Nusselder

Inhoudsopgave

1. Inleiding

In dit document is de ontwerpdocumentatie voor de opdracht RobotarmSimulatie van WoR World terug te vinden. Hierin zijn de volgende dingen terug te vinden. Er is een overzicht te zien van de verschillende packages en nodes. Dan volgt er een overzicht van de code door middel van een klasse diagram. Uiteindelijk volgt er een beschrijving van de belangrijkste implementatie delen. Als laatste volgt er een API beschrijving van de publieke interfaces van de nodes.

2. Structuur packages



Afbeelding 1: Component Diagram

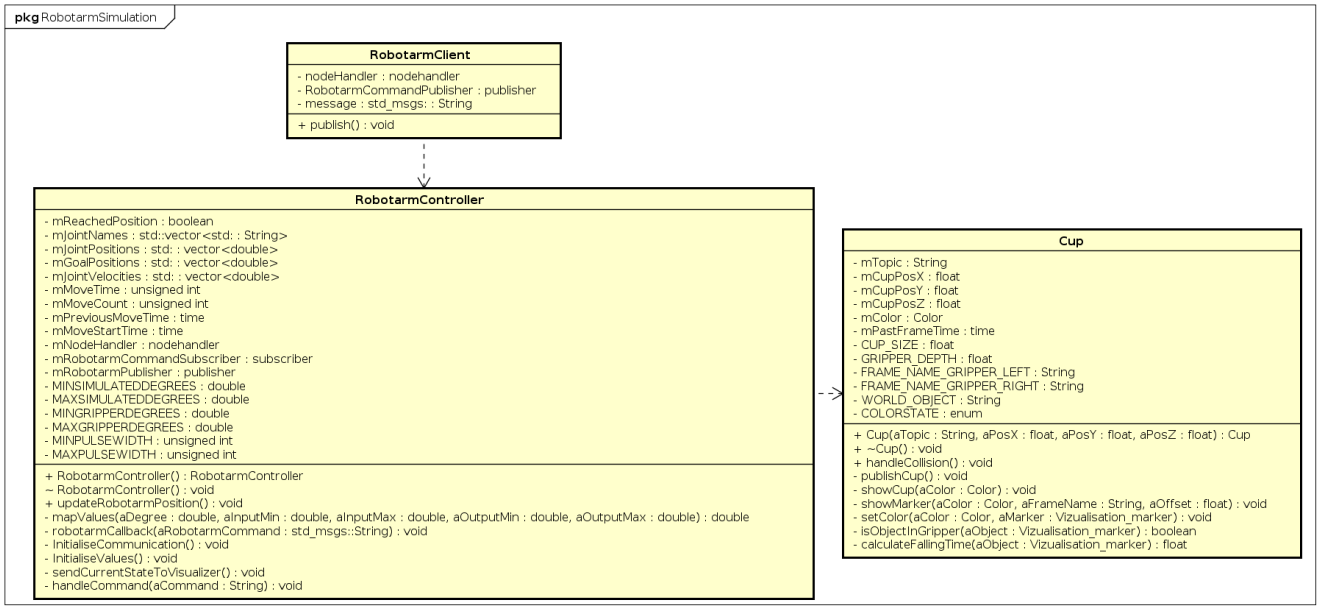
Zoals hierboven in ons component diagram te zien is hebben we vier componenten waarvan we er drie zelf gemaakt hebben. Het eerste component is de robotarm client. Dit component published robotarm commando's naar de robotarm controller via een ROS topic. Een voorbeeld van dit commando is te zien in het diagram. De syntax komt overeen met de commando's die naar de SSC32-U worden verstuurd.

Het robotarm controller component verwerkt de robotarm commando's die via het ROS topic binnenkomen en vertaald deze naar de juiste servo's en graden. Na het vertalen stuurt hij deze beweging over de gegeven tijd naar de robotarm simulatie in Rviz. Dit wordt gedaan door de huidige status van de robotarm te publishen op het ROS topic "/joint_state". Een voorbeeld van een bericht met de status van de robotarm is te zien in het diagram. Hierin worden de verschillende servo's en de graden van de hoeken gedefinieerd.

Naast deze twee componenten welke van elkaar afhankelijk zijn hebben we nog het cup component. Dit component visualiseert de cup en zorgt ervoor dat deze opgepakt kan worden en dat deze valt wanneer hij wordt losgelaten. Dit component stuurt de cup in de simulatie in Rviz aan door de status

van de cup naar het ROS topic “/vizualisation_marker” te sturen. Ook is hierbij een voorbeeld van het visualisatie bericht te zien. Hierin staat de huidige positie, richting en kleur van de cup.

3. Structuur code



Afbeelding 2: Klasse Diagram

Zoals hierboven in het klasse diagram te zien hebben we ook hier weer drie verschillende onderdelen. Dit zijn dezelfde componenten als die in het component diagram staan.

De robotarmClient is relatief simpel. Deze klasse heeft alleen een publisher en een bericht welke verstuurd wordt. Deze klasse is afhankelijk van de robotarmController omdat deze de berichten moet verwerken.

De robotarmController heeft een functie die elke keer vanuit een while loop wordt aangeroepen. Dit is de updateRobotarmPosition functie. Deze functie handelt alle bewegingen van de robotarm af en stuurt de robot naar de huidige positie in de beweging. Om de commando’s te kunnen parseren en versturen zijn er een aantal private hulp functies gemaakt welke gebruikt worden door de robotarmCallback. Een van deze hulp functies is de functie mapValues welke de waardes van het ene stelsel naar het andere mapt.

Als laatste hebben we het cup component. Dit component zorgt ervoor dat er een cup zichtbaar is in de simulatie en dat deze opgepakt kan worden door de robotarm. Hiernaast zorgt dit component er ook voor dat de cup weer naar de grond valt wanneer deze wordt losgelaten. Ook hier wordt er weer een functie aangeroepen in een while loop om de positie van de cup constant bij te houden. Dit is de publieke functie handleCollision. Zoals de naam al doet denken rekent deze functie uit of hij in aanraking is met de gripper en past hij de positie van de cup aan. De robotarm controller heeft een losse afhankelijkheid van de cup omdat er anders geen object in de wereld is om op te pakken.

4. Implementatie

Vertalen en updaten robotarm positie

Om de robotarm commando’s te vertalen en de positie te publiceren naar Rviz hebben we een robotarmController gemaakt. Het parseren van het commando wordt gedaan door er altijd vanuit te gaan dat alle servo’s worden aangestuurd en dan de waardes uit de string te lezen. Dit wordt gedaan door de string stukje voor stukje op te knippen en zo de juiste waardes uit te lezen.

Nadat de pulsbreedtes uit het commando uitgelezen zijn zetten we deze om naar de range die de simulatie verwacht door een mapping functie te gebruiken. Nadat we het doel berekend hebben en de tijd voor de beweging hebben kunnen we het verschil per milliseconde uitrekenen.

Nadat we deze waardes hebben kunnen we in een while loop controleren of er een beweging gaande is en als er een milliseconde verlopen is het verschil erbij optellen en de nieuwe status publiceren naar Rviz.

Visualiseren cup en detecteren gripper

Om verschillende cups te kunnen visualiseren in de wereld hebben we voor elke cup een losse executable. Deze verschillende cups maken gebruik van de tf library om zijn locatie tegenover een ander object te bepalen, in dit geval zijn dit de grippers. De cup wordt gestart met een specifiek coördinaat, hij blijft de afstanden controleren van zichzelf tegenover de gripper. Als het er op lijkt dat deze afstand zo dichtbij is dat de cup en de gripper contact hebben wordt hierop gereageerd door middel van een verplaatsing in zijn eigen locatie. Doordat de cup kan waarnemen of er een of twee grippers in contact staan met hem weet hij of hij klem zit of dat hij wordt geduwd.

5. API beschrijving

In dit hoofdstuk is de API beschrijving van de publieke interfaces van de verschillende componenten te vinden. In deze opdracht zijn er niet zoveel dingen die van buiten af aangeroepen kunnen worden. Per component is het een ROS node of een aantal publieke functies.

RobotarmClient

Dit is het programma welke de controller aanstuurt. Deze heeft zelf geen interface omdat deze de messages maakt die naar de controller gepublished worden. Dit bericht is een String welke naar het topic “robotarmCommand” gestuurd wordt. De inhoud van dit bericht is een commando in de syntax van de SSC32-U waarbij alle 6 de servo’s worden aangestuurd en er een tijd wordt meegegeven afgesloten door een newline. Een voorbeeld van zo’n commando is hieronder te vinden.

Voorbeeldcommando:

“#0P1500#1P1500#2P1500#3P1500#4P1500#5P1500T2000\n“

RobotarmController

Topic “/armCommands” met de robotarm commando’s, hier worden commando’s in de vorm van de SSC32-U naar toe gestuurd in een string.

De robotarm controller ondersteund alleen de commando’s in de volgende vorm : “#0P1500#1P1500#2P1500#3P1500#4P1500#5P1500T2000\n“.

De publieke functie updateRobotarmPositions om de robotarm positie te updaten. Deze functie controleert of er een beweging gaande is en als dit zo is wordt de positie veranderd en wordt de nieuwe positie gepublished.

Maakt gebruik van het topic “/joint_states” om de positie van de robotarm te publishen. Het publishen van de robotarm positie wordt gedaan met een jointState bericht.

Hoe dit bericht is opgesteld is hier te vinden.

Cup

De publieke functie handleCollision kan worden gebruikt om de cup te laten controleren of hij opgepakt of verschoven wordt en juist te reageren. Dit handelt ook het vallen van de cup door de zwaartekracht af.

Het programma gebruik van het topic “/vizualisation_marker” om de cup te publishen naar Rviz. Er wordt een vizualisation_msgs::Marker message gebruikt om deze data te publishen.

Hoe dit bericht is opgesteld is hier te vinden.