|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ontwerpdocumentatie\_  WoR WORLDS Simulatie   |  |  |  |  | | --- | --- | --- | --- | | **Auteur** | Mike Ackerschott | **Docent** | Jorg Visch | | **Student nr** | 1652342 | **Plaats** | Nijmegen | | **Opleiding** | HBO-ICT | **Datum** | 26-10-2023 | | **Profiel** | Embedded Software Development | **Versie** | 1.0 | | **Studiejaar** | Jaar 3 |  |  | |

26 oktober 2023

INHOUDSOPGAVE

[1 Ros Structuur 3](#_Toc149233228)

[1.1 Nodes 3](#_Toc149233229)

[1.2 Topics 3](#_Toc149233230)

[1.3 Services 3](#_Toc149233231)

[2 Broncode structuur 3](#_Toc149233232)

# Ontwerp simulatie

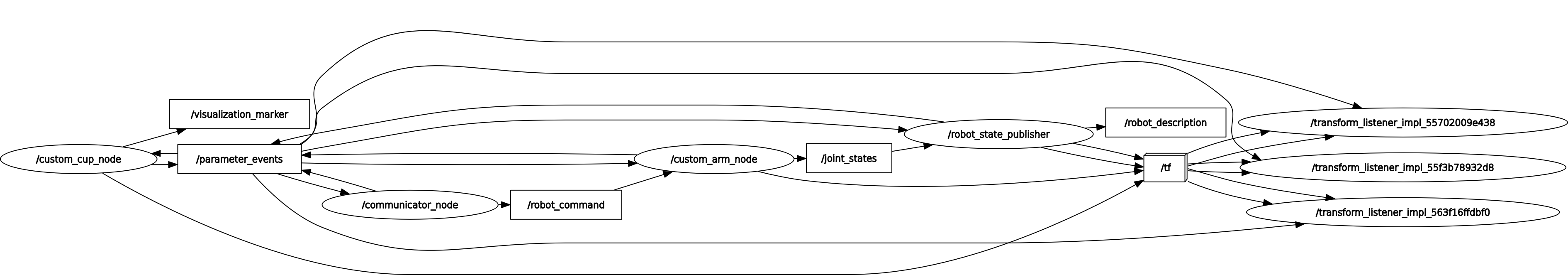
In dit hoofdstuk zal er in meer detail mijn uitwerking van de opdracht beschreven worden. Eerst zal de structuur en samenhang van alle ROS-onderdelen uitgelegd worden aan de hand van een diagram en extra toelichting.

Verder zal dit document ook nog dieper ingaan op de code zelf aan de hand van een klasse diagram.

## ROS-structuur

Het programma bestaat in totaal uit drie verschillende nodes, namelijk /communicator\_node, /custom\_arm\_node en /custom\_cup\_node.

Het bewegen van de robotarm start bij de /communicator\_node. Dit is een node die een programma draait waar er wordt gewacht op user input via een commandline interface. Als dit ontvangen is, wordt de input over de topic /robot\_command als string verstuurt naar de node /custom\_arm\_node. Hier wordt eerst geverifieerd of het commando in het formaat van het AL5D protocol is. Als dat niet zo is wordt het commando genegeerd.

Diagram 1: ROS structuur

Als het commando is geverifieerd, wordt er een beweging geplant gebaseerd op de huidige positie van de robotarm en de meegegeven waarden. Als het plannen klaar is, wordt er elk 10 milliseconde voor de algehele duratie van de beweging een nieuwe positie gepost via de /joint\_states topic naar de node /robot\_state\_publisher. Hierdoor beweegt de robotarm geleidelijk.

Wanneer de robotarm zijn gripper volledig dicht is na het uitvoeren van een beweging, wordt er vanuit de /custom\_arm\_node node een client request gedaan naar de /pickup\_cup service op de /custom\_cup\_node node. Hierbij wordt in de request de variabele pickup op true gezet. De /custom\_cup\_node node checkt dan of de hand van de robotarm dicht bij de beker is. Als dat zo is, wordt de beker opgepakt en bindt de beker zich aan de hand van de robotarm via TF2. De service antwoord dan met een pickup\_success = true als het bekertje is opgepakt.

Wanneer een beker is opgepakt en de gripper volledig wordt geopend, wordt er vanuit de /custom\_arm\_node over dezelfde service (namelijk /pickup\_cup) een request gestuurd, waarbij de variabele pickup op false wordt gezet. Dit ontvangt de service op de /custom\_cup\_node node en weet dan dat het bekertje zich moet losbinden van de hand van de robotarm en zwaartekracht toegepast moet worden. Hierbij bindt het bekertje via TF2 zich weer aan de sim\_link frame, wat betekend dat het op dezelfde plek in de wereld blijft.

## Node interfaces

Om het duidelijk te maken over hoe er gecommuniceerd kan worden met behulp van dit programma, is hieronder voor elke node de topics, services en actions gedocumenteerd en welke berichttypes ze gebruiken

### /communicator\_node

**Topics (Publishers):**

1. Topicnaam: "robot\_command"
   * Type: msg\_srv::msg::RobotCommand
   * Naam in sourcecode: robotCommandPub\_
   * Beschrijving: Dit topic wordt gepubliceerd om opdrachten naar de hoog-niveau driver te sturen. Een commando ziet er als volgt uit: #0 P1500 #1 P1500 #2 P1500 T3000

**Services (N/A):**

### /custom\_arm\_node

**Topics (Publishers):**

1. Topicnaam: "joint\_states"
   * Type: sensor\_msgs::msg::JointState
   * Naam in sourcecode: joint\_state\_pub\_
   * Beschrijving: Dit topic wordt gepubliceerd om de toestand van de gewrichten ("joint states") van de robot door te geven.

**Topics (Subscribers):**

1. Onderwerp: "robot\_command"
   * Type: msg\_srv::msg::RobotCommand
   * Naam in sourcecode: robot\_command\_sub\_
   * Beschrijving: Deze subscriber ontvangt opdrachten ("robot command") van externe bronnen om acties op de robot uit te voeren.

**Services (Clients):**

1. Servicenaam: "pickup\_cup"
   * Type: msg\_srv::srv::PickupCup
   * Naam in sourcecode: pickupCupClient\_
   * Beschrijving: Deze serviceclient wordt gebruikt om een verzoek te sturen om een beker op te pakken of los te laten, afhankelijk van de meegegeven parameters.

### /custom\_cup\_node

**Topics (Publishers):**

1. Topicnaam: "visualization\_marker"
   * Type: visualization\_msgs::msg::Marker
   * Naam in sourcecode: markerPub
   * Beschrijving: Dit topic wordt gepubliceerd om markerinformatie (waaronder het 3D .stl bestand) voor de "cup" naar visualisatietools zoals RViz te sturen.
2. Topicnaam: "cup\_pos"
   * Type: msg\_srv::msg::Pos
   * sourcecode: posPub
   * Beschrijving: Dit topic wordt gepubliceerd om de positie van de "cup" te verzenden.

**Services (Server):**

1. Service: "pickup\_cup"
   * Type: msg\_srv::srv::PickupCup
   * sourcecode: pickupCupService
   * Beschrijving: Dit is een service die kan worden aangeroepen om een "pickup" -actie uit te voeren voor de "cup." Het wordt gebruikt om de "cup" op te tillen of neer te zetten, afhankelijk van het verzoek.

# Broncode structuur

