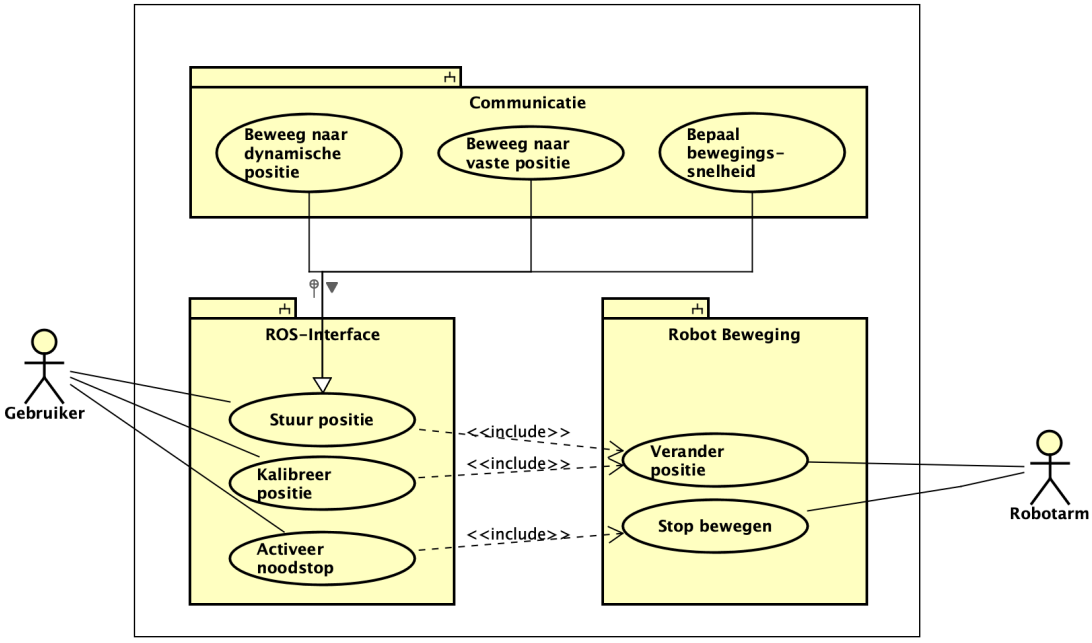
Robotarm Interface Opdracht

Kars Visscher, Mitchel van de Pest

26 June 2024

1. Usecases en Subsystemen

1.1 Diagram



1.2 Beschrijvingen

1.2.1 ROS-Interface

De ROS-Interface is het subsysteem waar de gebruiker een interactie mee heeft om de robotarm een actie te laten uitvoeren.

1.2.1.1 Stuur Positie

De gebruiker kan via de CLI poses of standen doorgeven volgens een vast protocol. Dit protocol staat aangegeven in de CLI. Wanneer de gebruiker een foutieve waarde invoert, dan krijgt de gebruiker een foutmelding. De ingevoerde waarden bestaan uit 3 soorten berichten, het bewegen naar zelf ingevoerde “dynamische” positie, het bewegen naar een vaste pose en het bepalen van de bewegingssnelheid. Hierbinnen wordt gebruik gemaakt van het sub-systeem Robot Beweging om de arm daadwerkelijk te laten bewegen.

1.2.1.2 Kalibreer Positie

De gebruiker kan de robotarm kalibreren via een .JSON bestand in de code. Hierbij staat gedefinieerd hoeveel graden aan vrijheid een gewricht heeft en hoeveel deze afwijkt in de realiteit. Hierbinnen wordt gebruik gemaakt van het sub-systeem Robot Beweging om de arm daadwerkelijk te laten bewegen.

1.2.1.3 Activeer Noodstop

De gebruiker kan de robotarm tot stoppen brengen met een noodstop. Bij een noodstop wordt er een commando gestuurd waarbij alle gewrichten van de robotarm tot stoppen worden gebracht. Hierna verkeerd de robot zich in een niet geinitaliseerde staat. Hierbinnen wordt gebruik gemaakt van het sub-systeem Robot Beweging om de arm daadwerkelijk te laten stoppen.

1.2.2 Robot Beweging

Het Robot Beweging Sub-Systeem heeft de directe interactie met de fysieke robotarm. Dit Sub-Systeem ontvangt commandos vanuit de Ros-Interface en vertaalt deze door naar commando's voor de controller van de robotarm.

1.2.2.1 Verander Positie

In verander positie wordt de positie van de robotarm fysiek veranderd. De commando's hiervoor zijn afkomstig uit het Ros-Interface subsysteem. Verander Positie houd ook rekening met de ingestelde kalibratie.

1.2.2.2 Stop Bewegen

In Stop Bewegen wordt de robot fysiek stilgezet. Dit commando is afkomstig vanuit het Ros-Interface Sub-Systeem

1.2.3 Communicatie

In het communicatie Sub-Systeem staan use-cases gedefinieerd naar in welke positie de robot kan bewegen.

1.2.3.1 Beweeg Naar Dynamische Positie

Het bewegen naar een dynamische positie kan worden gekozen door de gebruiker in de ROS-Interface. Hierbij wordt er rekening gehouden met kalibratie en de maximale vrijheidsgraden.Wanneer deze posities worden gekozen, wordt dit doorgestuurd naar het Robot Beweging Sub-Systeem.

1.2.3.2 Beweeg Naar Vaste Positie

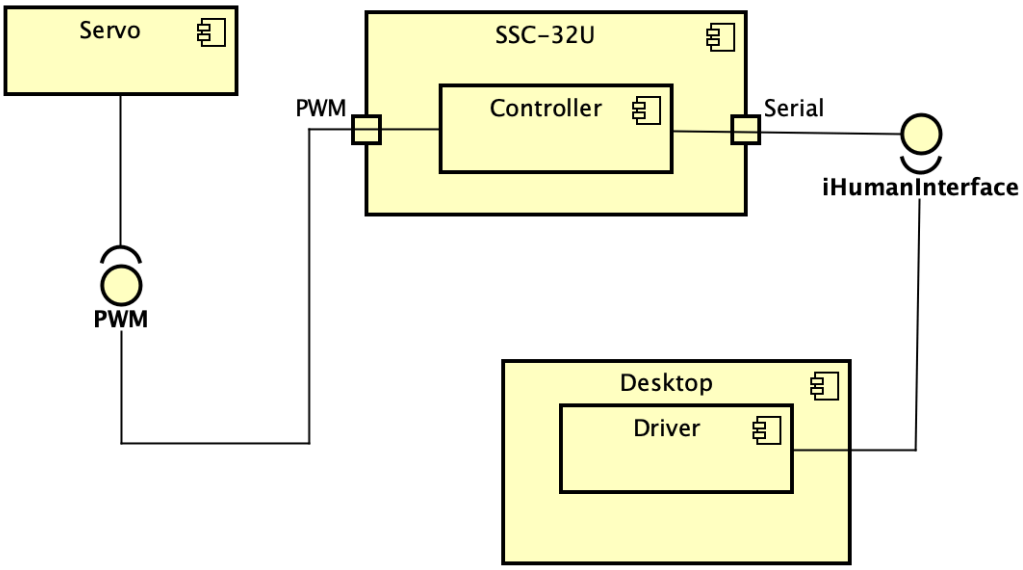
Het bewegen naar een vaste positie kan gedaan worden vanuit de ROS-Interface. Deze posities zijn van te voren aangegeven. Wanneer deze posities worden gekozen, wordt dit doorgestuurd naar het Robot Beweging Sub-Systeem.

1.2.3.3 Bepaal Bewegingssnelheid

Het bewegen naar een positie gaat met een bepaalde snelheid. Er is een standaard snelheid, maar voor speciale snelheden is er ook een optie. Wanneer deze snelheden worden gekozen, wordt dit doorgestuurd naar het Robot Beweging Sub-Systeem.

2. Component Diagram

2.1 Diagram



2.2 Component Beschrijvingen

2.2.1 Servo

In de robotarm bevinden zich meerdere servo motoren die samen verantwoordelijk zijn voor het bewegen van de individuele gewrichten van de robotarm. Deze motoren worden aangestuurd over het PWM protocol. Dit PWM signaal is afkomstig van het Controller component.

2.2.2 SSC-32U

De SSC-32U is een servo-controller board. Op dit bord zijn alle servo motoren aangesloten. Dit component bevat nog een ander component, namelijk de controller. De controller zet seriele berichten om naar PWM signalen voor de servo motoren.

2.2.3 Controller

De controller is onderdeel van het SSC-32U component. Deze vertaalt de seriele commandos afkomstig van de Driver naar daadwerkelijke PWM signalen die de Servo motoren kunnen gebruiken.

2.2.4 Desktop

Op de desktop draait de software die de gebruiker instaat stelt om de code te draaien en er een interactie mee te hebben.

2.2.5 Driver

Het driver component draait op de desktop en zorgt er voor dat er seriele berichten verstuurd kunnen worden naar de controller. Hierbij is er een CLI voor de gebruiker aanwezig.

2.3 Interface beschrijvingen

2.3.1 PWM

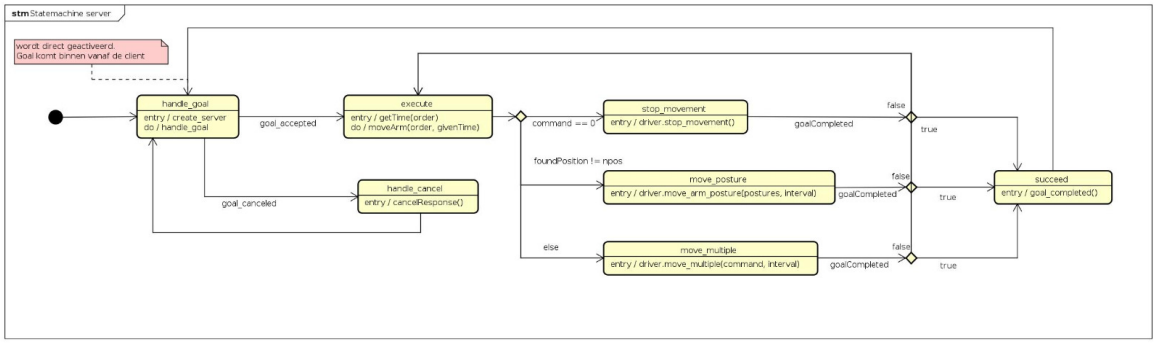
De PWM interface wordt gebruikt door de Servo en de Controller. Dit stelt de controller en de servo in staat om over dezelfde interface te communiceren. De interface wordt aangeboden door de controller.

2.3.1 iHumanInterface

De iHumanInterface is een interface die aangeboden wordt door de Controller. Deze wordt gebruikt door de Driver. Deze interface werkt over een seriele verbinding en stelt de controller en driver in staat om met elkaar te kunnen communiceren.

3. Protocol State Machine

3.1 Diagram Server



3.1.1 Toelichtingen

3.1.1.1 handle_goal

De handle_goal functie wordt geloopt om te luisteren naar opdrachten van de client. De server wordt hier de eerste keer op gestart. Hier wordt bekeken of het goal geaccepteerd wordt of wordt gecanceld.

3.1.1.2 execute

Als het goal wordt geaccepteerd wordt de getTime functie aangeroepen. Hierin wordt de tijd in milliseconde uit de opdracht gehaald om te controleren of het commando binnen de QoS behaalt kan worden(2300 ms). Vervolgens wordt de moveArm functie aangeroepen om de opdracht uit te voeren.

3.1.1.3 handle_cancel

Als de opdracht ongeldig is of als er een fout zit in de communicatie tussen de server en de client wordt de opdracht gecanceld en gaat de server weer wachten.

3.1.1.4 stop_movement

Op het moment dat het commando 0 is wordt de stop_movement functie uit de driver aangeroepen om het stop commando uit te voeren.

3.1.1.5 move_posture

Op het moment dat de foundposition een posture bevat wordt die bijbehoren posture vanuit een ENUM doorgegeven aan een functie in de driver die hem uitvoert.

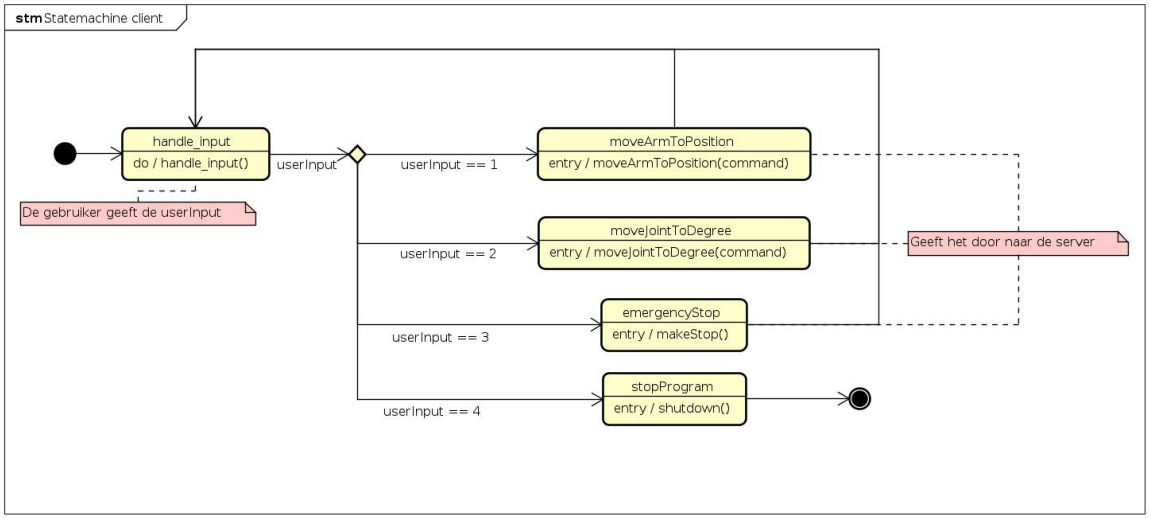
3.1.1.6 move_multiple

Als het commando geen stop of posture is wordt het beschouwt als een move_multiple commando. Dus wordt de bijbehorende driver functie aangeroepen.

3.1.1.7 succeed

Als het commando is afgerond gaat de server opnieuw wachten tot er een nieuw commando is.

3.2 Diagram Client



3.2.1 Toelichtingen

3.2.1.1 handle_input

Hierin wordt de client aangemaakt en wordt er gewacht op input van de gebruiker.

3.2.1.2 moveArmToPosition

Op het moment dat de input van de gebruiker 1 is wordt er gewacht op een moveArmToPosition commando om die vervolgens door te geven naar de server.

3.2.1.3 moveJointToDegree

Op het moment dat de input van de gebruiker 2 is wordt er gewacht op een moveJointToDegree commando om die vervolgens door te geven naar de server.

3.2.1.4 emergencyStop

Op het moment dat de input van de gebruiker 3 is wordt er een noodstop commando doorgegeven naar de server.

3.2.1.5 stopProgram

Als de input van de gebruiker 4 is wordt de client afgesloten.