INTERFACE OPDRACHT

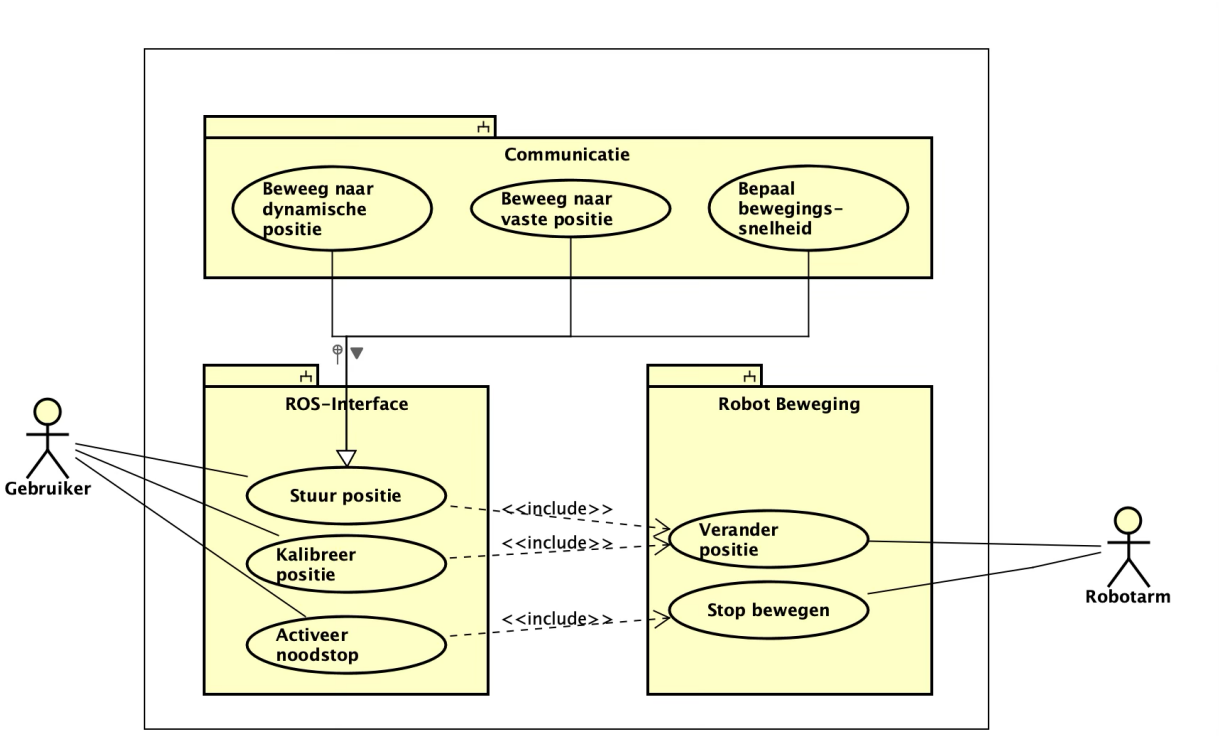
Docent: Bram Knippenberg

Kars Visscher 1657531, Mitchel van de Pest 1653354

Table of Contents

Beschrijving van het systeem

Usecase Diagram



Beschrijvingen

Ros Interface Subsysteem

Het eerste subsysteem is het ROS-interface-subsysteem Dit subsysteem handelt alle usecases af waarbij de gebruiker direct betrokken is. Vanuit dit subsysteem worden de usecases "Stuur positie", "Kalibreer positie" en "Activeer noodstop" uitgevoerd. Het ROS-interface-subsysteem bevindt zich aan de clientzijde van de action server. De client maakt gebruik van het "Communicatie"-sub-systeem om de commando's van de gebruiker, ingevoerd via de terminal, te vertalen naar opdrachten voor de serverzijde van de action server. Verder maakt hij hierdoor gebruik van de iHumanInterface interface.

Robot beweging sub-system

Het robot beweging sub systeem bevat alle usecases die worden afgehandeld door de fysieke robotarm in de vorm van bewegingen, weergegeven als actor robotarm. Dit sub-systeem bevindt zich aan de server kant van het programma dat uiteindelijk de seriele commando's stuurt naar de robotarm over de beschikbare seriele poort. De gebruiker maakt gebruik van dit systeem doormiddel van de usecases Stuur positie, Kalibreer positie en activeer noodstop. Deze usecases ontvangen de commando's van het sub-systeem 'Communicatie'.

Communicatie Subsysteem

Het communicatie-subsysteem fungeert als een bibliotheek van functies voor de clientzijde van de action server. In dit subsysteem worden de commando's opgebouwd die vervolgens naar de serverzijde van de action server kunnen worden verstuurd. Hier worden de usecases "Beweeg naar dynamische positie", "Beweeg naar vaste positie" en "Bepaal bewegingssnelheid" uitgevoerd. Net als het ROS-interface-subsysteem bevindt dit subsysteem zich aan de clientzijde.

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.

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.

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.

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.

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.

Stop bewegen

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

Communicatie

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

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.

Beweeg naar vaste positie

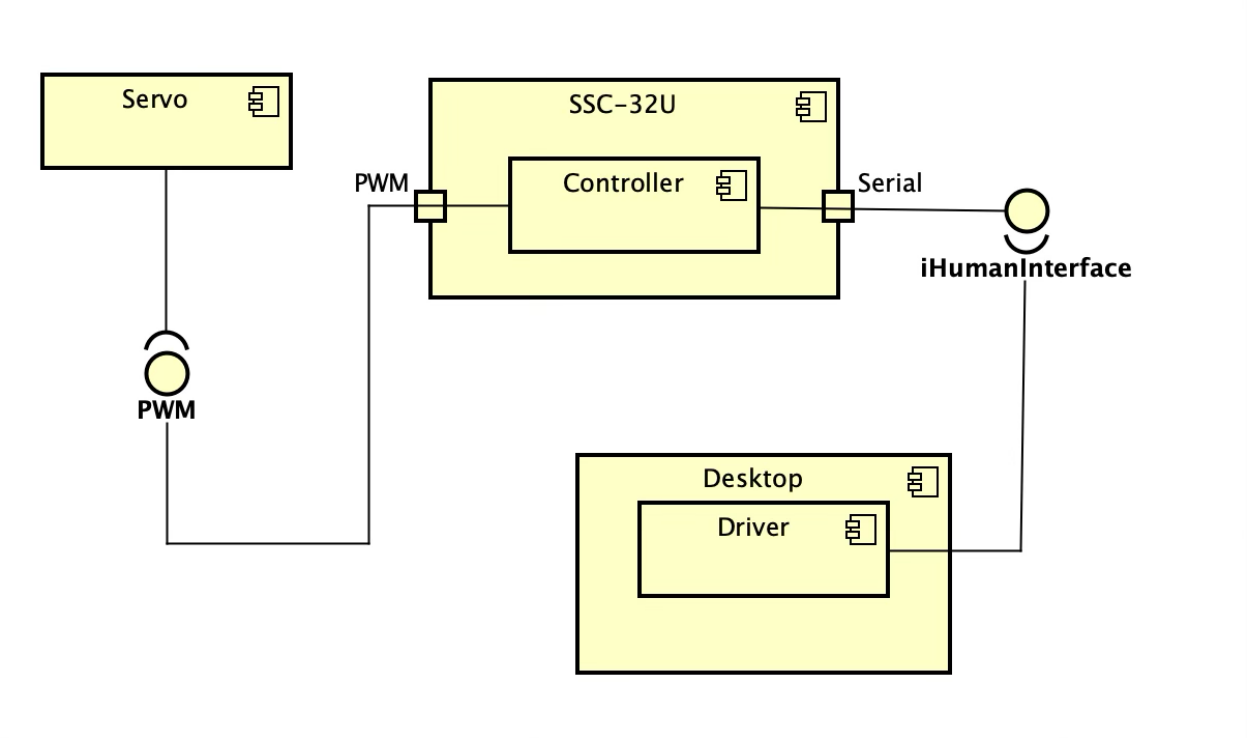
Het bewegen naar een vaste positie kan gedaan worden vanuit de ROS-Interface. Deze posities zijn van te voren aangegeven. Wanneer deze posities worden gekozen, wordt dit doorgestuurd naar het Robot Beweging Sub-Systeem.

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.

Component diagram

Diagram



Servo

In de robotarm bevinden zich meerdere servomotoren die samen verantwoordelijk zijn voor het bewegen van de individuele gewrichten van de robotarm. Deze motoren worden aangestuurd via het PWM-protocol. Het PWM-signaal wordt gegenereerd door het Controller-component.

SSC-32U

De SSC-32U is een servo-controllerboard waarop alle servomotoren zijn aangesloten. Dit component bevat een ander subcomponent, namelijk de Controller. De Controller zet seriële berichten om in PWM-signalen die geschikt zijn voor de servomotoren.

Controller

De Controller is onderdeel van het SSC-32U-component. Het vertaalt de seriële commando's, afkomstig van de Driver, naar PWM-signalen die de servomotoren kunnen gebruiken.

Desktop

Op de desktop draait de software waarmee de gebruiker interactie kan hebben en de benodigde code kan uitvoeren.

Driver

Het Driver-component draait op de desktop en zorgt ervoor dat seriële berichten naar de Controller worden gestuurd. Dit component bevat een CLI (Command Line Interface) waarmee de gebruiker commando's kan invoeren.

Interface beschrijvingen

PWM

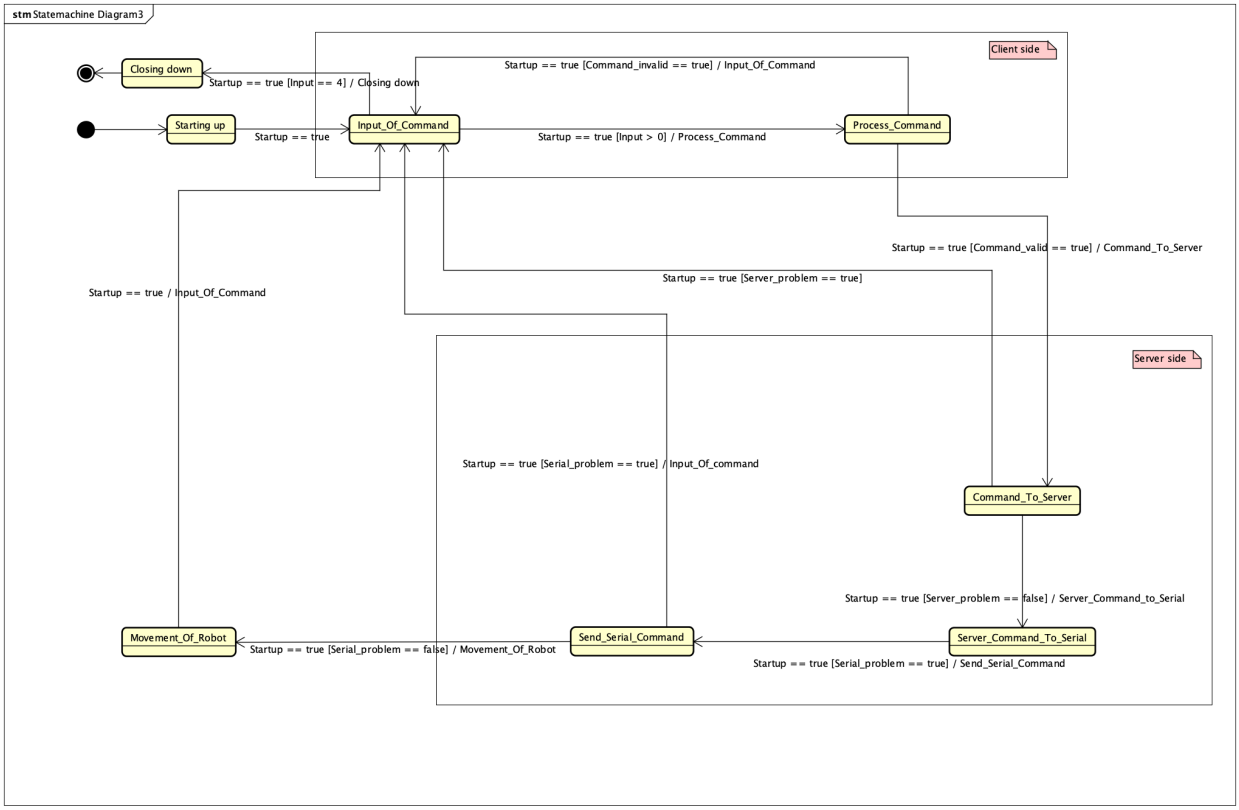
De PWM interface wordt gebruikt door de Servo en de Controller. Dit stelt de controller en de servo in staat om over dezelfde interface te communiceren. De interface wordt aangeboden door de controller. PWM is een elektrisch signaal dat gemodificeerd wordt naar een duty cycle, waarbij op bepaalde intervallen het signaal hoog is en op andere intervallen laag. Dit wordt gestuurd op een stroom van 5v. Een servo motor kan de dalen en pieken in het signaal vertalen naar een positie ergens tussen de -90 en 90 graden. Deze vorm moet elke 20ms herhaald worden om een integer signaal te zijn. Hierna zal de servo succesvol van beweging veranderen. De gebruiker van het systeem heeft hier geen directe interactie mee. Deze interactie wordt afgehandeld en gegenereerd door de processor op Lynxmotion printplaat. De gestuurde commando's vanuit de driver het desktop component vormen de basis voor alle PWM signalen en bewegingen.

iHumanInterface

De iHumanInterface is een interface die zoals de naam al beschrijft, gebruikt wordt door de gebruiker (mens) in het systeem. Deze interface is te bedienen doormiddel van een CLI interface. In deze interface kan de gebruiker kiezen tussen verschillende commando's die gebruikt worden om de robotarm in een bepaalde pose of exacte positie te zetten. Hierbij is het ook mogelijk om een tijdsinterval mee te geven aan een commando. Hierbij wordt bij een interval groter dan 2300ms aangegeven dat de QoS van het systeem niet meer behaald kan worden, ook biedt deze interface ruimte tot kalibratie en een noodstop. Bij kalibratie kan de basis rust positie in een pose aangepast. Dit is relevant omdat elke fysieke robotarm een net iets ander commando nodig heeft om dezelfde positie te behalen. Dit kan komen door degradatie van de servo motoren of een net andere aansluiting van een gedeelte van de arm op een servo motor. Bij een noodstop worden alle servo's direct stilgezet. Deze input methode wordt bevindt zich op de desktopcomponent, waarin zich een driver component bevindt. De driver component zet de ingevoerde commando's om tot seriële berichten die ingelezen worden door de controller in het SSC-32U component.

De seriele bus wordt gebruikt doormiddel van een USB kabel die fysiek is verbonden aan de desktop en aan het SSC-32U component.

Systeem gedrag



Starting up

De starting up state is de eerste state waarin het systeem zich kan bevinden. Hierin wordt het client side programma opgestart. Dit is het CLI programma waarin commando's kunnen worden ingegeven om de robot arm te laten bewegen. Dit triggerd het begin van de state waarin het systeem zich kan begeven.

Input of command

In de input of command state kunnen commando's worden ingegeven. Met deze commando's is het mogelijk om de robotarm naar poses te laten bewegen, naar een specifieke plek te bewegen, een noodstop te sturen of om het programma te eindigen. Dit gebeurt aan de client side van het programma en heeft relevantie tot alle sub-systemen die toebehoren aan dit systeem. Dit is de kern van het aansturen van de robotarm. Wanneer

een commando wordt ingegeven verandert de state naar Process command of wanneer er een exit input wordt gegeven naar closing down.

Process Command

In de process command state wordt het gegeven commando van de verbruiker verwerkt. Hierbij wordt gekeken naar de syntax van het commando en naar de inhoud van het commando. Het gegeven commando moet valide zijn volgens de gegeven input regels in de CLI omgeving om door te gaan naar de server. Wanneer dit commando valide is veranderd het systeem van staat naar Command to server. Wanneer dit niet valide is, gaat deze terug naar Input of command.

Command to Server

Zodra het commando inhoudelijk en syntactisch correct is, komt het systeem in de command to server state terecht. In deze staat wordt het commando klaargemaakt om naar de server toegestuurd te worden. Zodra dit bericht succesvol naar de server is verstuurd veranderd het programma naar de state Server command to serial. Hierbij wordt de client side van het systeem verlaten en ingeruild voor de server side van het systeem. Wanneer een bericht niet succesvol verstuurd kan worden om wat voor reden dan ook, wordt er teruggevallen naar de Input of command state en wordt het versturen van een commando afgebroken.

Server command to serial

Wanneer de server een bericht ontvangt van de client, verandert het systeem naar de state van server command to serial. Deze staat het gegeven commando om naar een bruikbaar serieel bericht voor de robotarm. Zodra dit is gebeurd veranderd de state naar Send serial command. Het gekregen commando is van te voren gefilterd op eventuele fouten die kunnen voorkomen.

Send serial command

In de send serial command state wordt het opgestelde seriële bericht verstuurd naar het actuele seriële apparaat dat is aangesloten aan de desktop. Dit bericht bestaat uit gegeven waarden van gebruiker, in de vorm van een pose of van een handmatige beweging. Wanneer er een probleem optreed met de seriële poort, verandert het systeem terug naar de input of command state met daarin feedback over wat er fout is gegaan. Wanneer er een succesvol bericht is verstuurd wordt er overgeschakeld naar de movement of robot state. Hierbij wordt de server side van het programma verlaten.

Movement of robot

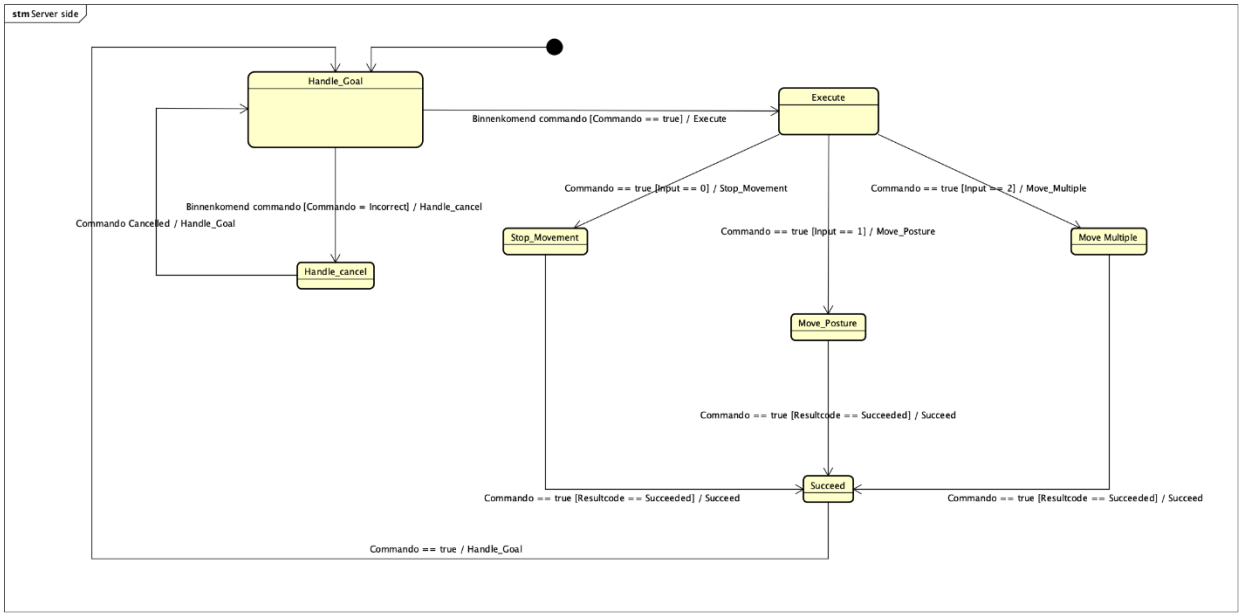
De movement of robot duid op het daadwerkelijk bewegen van de robotarm. Nadat het bewegen op gegeven interval is afgerond, treed het systeem terug in de Input of command state en wordt de client side weer betreden.

Closing down

De closing down state treed op wanneer er in de input of command state een commando wordt gegeven voor het afsluiten van het programma. Zodra deze state is aangeroepen, betekent dit dat het programma sluit en er geen interactie meer mogelijk is tot dat het programma weer opnieuw wordt opgestart.

Inzicht in gebruik

Protocol State Machine Server



States Toelichtingen

Handle_goal

De handle_goal-functie wordt continu uitgevoerd om te luisteren naar inkomende opdrachten van de client. Dit is de eerste state waarin de server zich bevindt na het opstarten. In deze state wordt gecontroleerd of een binnengekomen commando geldig is en geaccepteerd kan worden. Indien het commando niet voldoet aan de voorwaarden, zoals ontbrekende of onjuiste gegevens, wordt het geannuleerd en wordt er feedback terug gestuurd. Nadat de opdracht is geaccepteerd, wordt de server doorgestuurd naar de volgende state om de opdracht uit te voeren.

Execute

Wanneer een goal wordt geaccepteerd, roept de server de getTime-functie aan. In deze functie wordt de opgegeven tijd (in milliseconden) uit het commando gehaald en gecontroleerd of deze binnen de gestelde QoS-normen (2300 ms) valt. Hierna schakelt de server over naar de moveArm-functie om het opgegeven commando uit te voeren. Hierbij worden de instructies vertaald naar bewegingen van de robotarm. Als de opdracht niet binnen de QoS-normen valt door er langer dan 2300 ms over te doen zal het systeem dit via de terminal laten weten. De robotarm zal nogsteeds het commando uitvoeren maar het zal langer dan 2300 ms duren. Hiermee kan de beoogde 2300ms niet behaald worden.

Handle_cancel

Indien een opdracht ongeldig blijkt te zijn of als er een fout optreedt in de communicatie tussen de client en de server, wordt de opdracht geannuleerd in de handle_cancel-state. Dit kan bijvoorbeeld gebeuren als er onjuiste gegevens worden verstuurd of als de verbinding wordt verbroken. In deze state zorgt de server ervoor dat de annulering correct wordt afgehandeld en eventuele resterende processen worden beëindigd. Zodra de annulering is voltooid, keert de server terug naar de wachtstand om te luisteren naar nieuwe opdrachten.

Stop_movement

De stop_movement-state wordt geactiveerd wanneer het ontvangen commando een waarde van 0 heeft. In deze state roept de server de stop_movement-functie van de Driver aan, die ervoor zorgt dat de robotarm onmiddellijk stopt met bewegen. Dit is een essentiële state om veiligheid te waarborgen, bijvoorbeeld in noodsituaties. Nadat de stopbeweging is uitgevoerd, controleert de server of verdere actie nodig is of dat hij kan terugkeren naar de wachtstand.

Move_posture

Als het commando aangeeft dat een specifieke houding (posture) moet worden aangenomen, gaat de server naar de move_posture-state. Hier wordt de bijbehorende posture opgehaald uit een ENUM-lijst en doorgegeven aan een functie in de Driver. Deze functie vertaalt de opdracht naar een beweging die door de robotarm wordt uitgevoerd. Dit type commando wordt gebruikt voor vaste posities die vooraf zijn gedefinieerd, zoals de park-, ready- en straigt up positie. Nadat de beweging is voltooid, gaat de server verder naar de succeed state.

Move_multiple

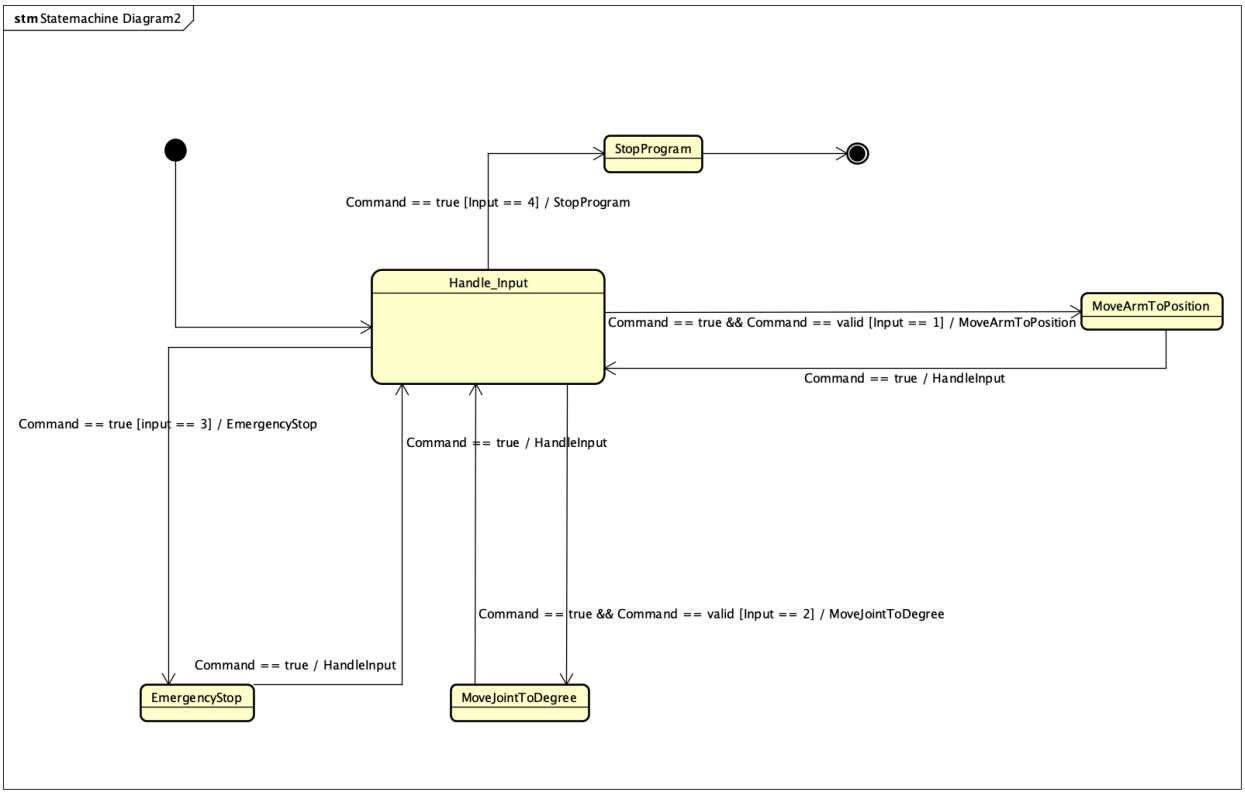
Indien het ontvangen commando noch een stop- noch een posture-opdracht is, wordt het beschouwd als een move_multiple-commando. In deze state roept de server de move functie aan om meerdere servo's tegelijkertijd een opgegeven beweging uit te laten voeren. Dit zorgt ervoor dat de robotarm een volledige vrije beweging kan maken binnen de opgegeven tijd. Nadat deze state is uitgevoerd, verandert het systeem naar de succeed state. Bij het maken van het commando is het aan de gebruiker om een veilig commando te bedenken, hierbij wordt geen rekening gehouden met safety aspecten die de veiligheid van de gebruiker of robotarm kunnen schaden.

Succeed

Zodra een commando met succes is uitgevoerd, schakelt de server over naar de succeed-state. Hier wordt de succesvolle afronding van de opdracht bevestigd aan de client. Vervolgens keert de server terug naar de wachtstand en luistert hij naar nieuwe inkomende

opdrachten. Deze state markeert het einde van de cyclus en zorgt ervoor dat de server klaar is voor de volgende instructies van de gebruiker.

Protocol State Machine Client



Toelichtingen states

Handle_input

In deze state wordt de client aangemaakt en wordt er gewacht op input van de gebruiker. De gebruiker kan verschillende opdrachten invoeren om acties te initiëren. Afhankelijk van de ingevoerde input schakelt de client over naar de bijbehorende state.

MoveArmToPosition

Wanneer de gebruiker een input van 1 invoert, schakelt de client over naar de moveArmToPosition-state. In deze state wordt gewacht op een specifiek moveArmToPosition-commando dat vervolgens wordt doorgestuurd naar de server. Dit commando bepaalt naar welke positie de robotarm moet bewegen.

MoveJointToDegree

Als de gebruiker een input van 2 invoert, wordt de client naar de moveJointToDegree-state geleid. Hier wordt het een moveJointToDegree-commando dat meegegeven wordt uitgelezen. In dit commando staat naar welke hoek een specifiek gewricht moet bewegen. Dit commando wordt daarna doorgestuurd naar de server.

EmergencyStop

Bij een input van 3 door de gebruiker schakelt de client over naar de emergencyStop-state. In deze state wordt een noodstopcommando gegenereerd en onmiddellijk doorgestuurd naar de server. Dit commando zorgt ervoor dat de robotarm alle bewegingen en eerdere commando's afbreekt.

StopProgram

Wanneer de gebruiker een input van 4 invoert, wordt de stopProgram-state geactiveerd. In deze state wordt de client afgesloten en stopt het programma volledig. Dit markeert het einde van de interactie met de client.

Inzicht in bruikbaarheid

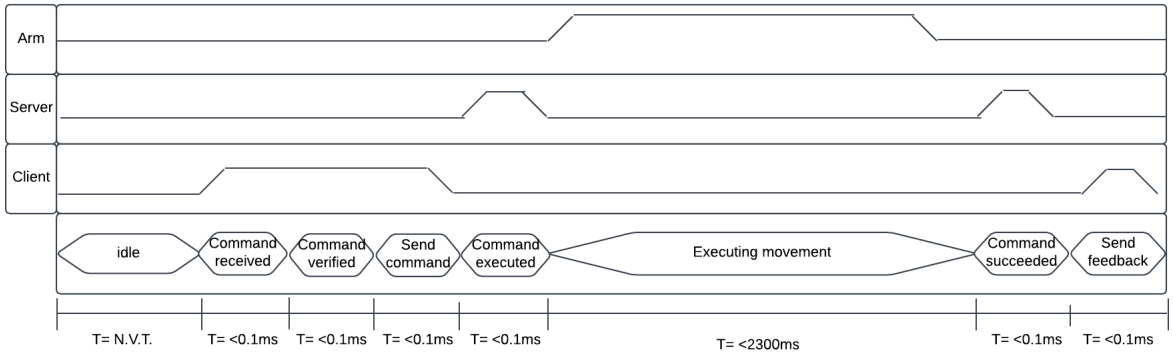
Weergave Quality of Service

Voor de quality of service wordt er gekeken naar de usecases zoals ze in de opdracht staan beschreven. Voor elke usecase wordt aangegeven of het systeem voldoet aan de opgestelde eis.

ID	Omschrijving	Behaald
SA01	Elke vrijheidsgraad moet worden ontsloten, waarbij de bijbehorende servo's naar de opgegeven positie (hoek, in graden) kunnen worden verplaatst	Ja, De robotarm kan bewegen naar de opgegeven posities.
SA02	De verplaatsing moet met verschillende snelheden uitgevoerd kunnen worden. Hierbij moet de opgegeven positie in de opgegeven tijd (in milliseconden) worden bereikt.	Ja, De commando's van de robotarm hebben de tijd waarin de positie bereikt moet worden.
SA03	Snelheidseis: Als de gripper van AL5D-robotarm naar een locatie (een set van samengestelde servohoeken1) wordt gestuurd moet deze binnen 2,3 seconden kunnen worden bereikt.	Dit is afhankelijk of de gebruiker een tijd van onder de 2300ms aangeeft. Anders geeft het systeem een bericht dat de QoS niet bereikt kan worden.
SA04	De software moet op een eenvoudige manier rekening houden met verschillen tussen robotarmen wat betreft de draaiing. Bijv. bij de optie straightup (eis PO03) staat elke arm precies rechtop. De software houdt dus rekening met verschillen in standen van de servo's.	Ja, De robotarm kan middels config-file worden gekalibreerd.
PO01	Park: Een positie waarin de arm veilig kan worden uitgeschakeld.	Ja, de robotarm beweegt zichzelf na het park commando in de park positie.
PO02	Ready: De "Ready"-stand kan als basispositie voor het uitvoeren van werk worden gebruikt.	Ja, de robotarm beweegt zichzelf na het ready commando in de ready positie.
PO03	Straight up: Een positie waarin de robotarm (gecentreerd) volledig rechtop staat.	Ja, de robotarm beweegt zichzelf na het straigth up commando in de straight up positie.

VE01	Noodstop: De arm wordt zo snel mogelijk gestopt, waarbij de huidige actie wordt afgebroken. Alle openstaande acties van de robotarm worden geannuleerd.	Ja, de robotarm stopt direct elke beweging op het moment dat een stop commando wordt gegeven.
VE02	Beperking range of motion: De software beperkt de beweging van de servo's tot een veilige range. Zie de tabel in appendix A voor de veilige ranges.	Ja, de ingevoerde hoeken van de servo's worden gecontroleerd en een error wordt teruggestuurd naar de gebruiker indien de hoeken niet geldig zijn.
VE03	Opstart-initialisatie: Bij het opstarten van de robotarm gaat deze gegarandeerd naar de Park-positie. De verplaatsingssnelheid is hierbij gelimiteerd tot de helft van de maximumsnelheid.	Ja, de robotarm gaat voordat het eerste commando is gestuurd op gelimiteerde snelheid naar de park positie.
INF01	De interface geeft toegang tot informatie over de operationele toestand van het arm-systeem via rosout op INFO niveau. De weergegeven toestanden moeten overeenkomen met de toestanden van de protocol state machine bij US02. De informatie wordt geboden in het volgende formaat "STATE: {statename}".	Ja, het systeem geeft informatie terug aan de gebruiker over de status van de robot door middel van prints in de terminal. Volgens het gevraagde format.
INF02	Geef event informatie via rosout op DEBUG niveau. Hierbij zijn in ieder geval de events opgenomen zoals zichtbaar zijn in de protocol state machine bij US02. De informatie wordt geboden in het volgende formaat "EVENT: {eventname}".	Ja, het systeem geeft informatie terug aan de gebruiker over de status van de robot door middel van prints in de terminal. Volgens het gevraagde format.
INF03	Geef een WARNING via rosout indien een opdracht aan de robotarm niet binnen de vereiste tijd kan worden uitgevoerd en geef daarbij de wel verwachtte tijd van afhandeling aan. De informatie wordt geboden in het volgende formaat "QoS-Warning: {warning message}".	Ja, in het geval dat de gebruiker een tijd van meer de 2300ms geeft zal het systeem de gebruiker laten weten dat de QoS niet behaald kan worden met deze snelheid.
EX01	Queuing van opdrachten: Er kunnen meerdere opdrachten worden toegevoegd aan een wachtlijst. De queue kan worden geleegd. Bij een noodstop wordt de queue geleegd.	Nee, commando's worden niet achter elkaar in een queue gezet.

Beargumentatie Quality of Service



Het timingdiagram toont de processtappen die plaatsvinden vanaf het ontvangen van een commando tot het succesvol uitvoeren van de beweging door de robotarm. Het systeem bevindt zich aanvankelijk in een Idle-toestand, waarin het wacht op een opdracht van de gebruiker. Zodra een Command received wordt gedetecteerd, wordt dit zo snel mogelijk verwerkt en doorgestuurd door het systeem. Vandaar dat dit aangegeven staat met een tijd van minder dan 0.1 ms. Hierna worden de volgende stappen uitgevoerd Command verified, Send command en Command executed. De langste tijdsduur in het proces is de Executing movement, waarin de robotarm een beweging naar de opgegeven positie uitvoert. Deze tijd is afhankelijk van de opgegeven opdracht, het is de bedoeling dat dit minder dan 2300 miliseconden duurt. Maar als het meer dan dat is zal het systeem dat aangeven. Zodra de beweging is voltooid, bevestigd de server dat het commando is uitgevoerd en worden de stappen command succeeded en send feedback zo snel mogelijk uitgevoerd.