Robot arm controller

David Hendrik Nusselder Maurits Muijsert

17-04-2019

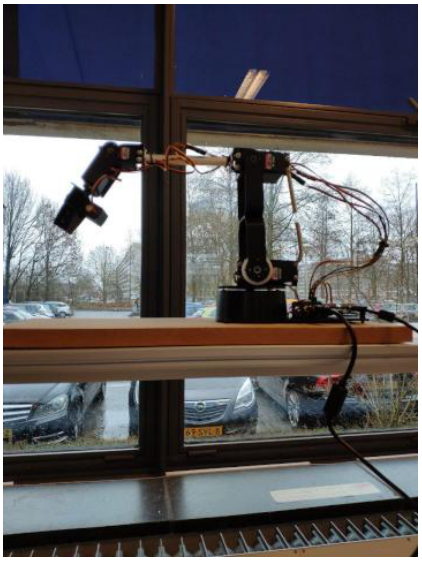

Figure 1: AL5D robot arm

```
Versie      : 1
Datum       : 17-04-2019
Klas        : ITA-WoR-A-f
Vak         : I-WOR World 18/19 S2
Docent      : Jorg Visch
Studenten   :
    David Hendrik Nusselder (584310)
    Maurits Muijsert (585468)
```

Contents

1 Robot arm controller

1.1 Inleiding

In dit verslag wordt de documentatie beschreven van de aansturing interface van de LynxMotion AL5D robotarm. Er wordt in dit document geïnformeerd hoe het systeem in elkaar. Dit met behulp van: een Use Case Diagram, Component, Behavioral State Machine, Protocol State Machine, Timing Diagram en een Sequence Diagram. In dit document zullen wij nader toelichting geven over de bruikbaarheid van deze arm waarbij we kijken naar de gestelde eisen in de opdracht.

1.2 Use Cases

In het Use Case Diagram worden alle eigenschappen beschreven waarmee de gebruiker kan interacteren. In dit diagram hebben we twee systemen beschreven die elk een applicatie representeren. De onderlinge contacten vinden plaats via Ros Messages, dit kan via drie ontvanger nodes. De eerste node is voor de emergency stop, de tweede node voor een preset position en de derde voor een custom position.

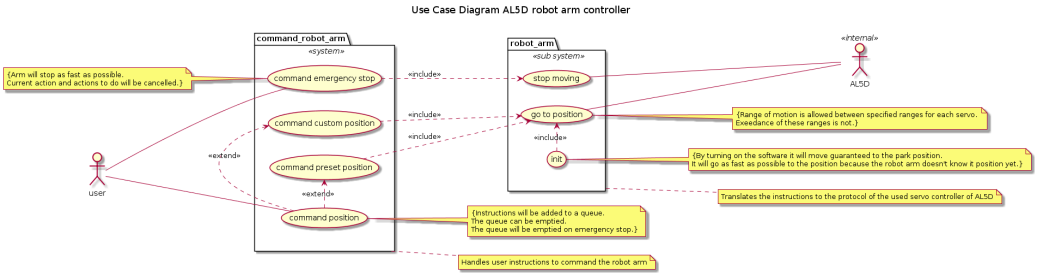

Figure 2: UCD

1.3 Componenten

In het Component Diagram beschrijven wij de verschillende componenten in het systeem. Hierbij wordt onderscheid gemaakt tussen de verschillende drivers en de subsystemen. De “getCurrenPosition()” in de High Level Driver is gemaakt, omdat de robotarm zelf geen feedback verzorgt. De functie maakt een schatting van de verwachte positie als de robotarm geen problemen ondervindt. Deze functionaliteit is essentieel om erachter te kunnen komen of de arm klaar is met zijn taak, hierdoor kan de Queue geïmplementeerd worden.

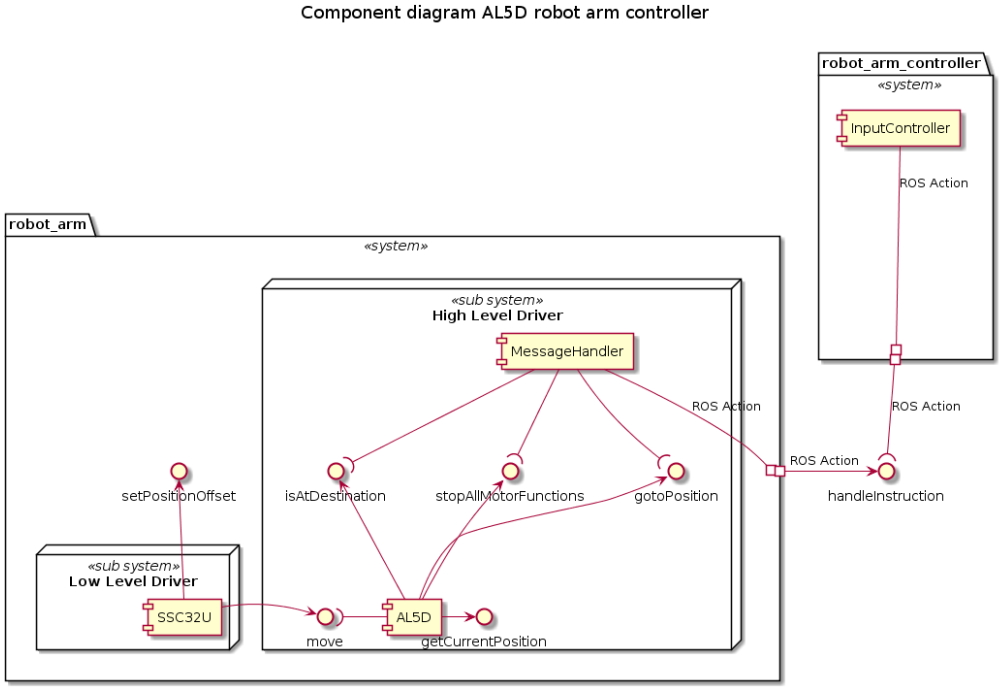

Figure 3: Component diagram

1.3.1 robot_arm_controller

De **robot_arm_controller** package is een ROS Action client applicatie. Die publiceert een action specification. Die bestaat uit een goal, feedback en result. De mogelijke goals zijn *pose*, *custom pose* en *emergency*.

1.3.2 robot_arm

De robot_arm package is een ROS action server applicatie. Die voert een action specification uit. Die reageert op *pose*, *custom pose* en *emergency*. Bij pose gaat de robot arm als het mogelijk is naar een *preset pose*. Dat houdt in dat de robot arm naar een vooraf ingestelde positie gaat. De vooraf ingestelde posities zijn **park**, **ready** en **straight_up**. Park is de positie waarin de robot begint.

1.3.2.1 High Level Driver

De **High Level Driver** bestaat uit twee onderdelen. Namelijk het afhandelen van goals (met *MessageHandler*) en het aansturen van de **Low Level Driver** (met *AL5D*). AL5D biedt de volgende functionaliteiten aan:

- **isAtDestination** : Geeft aan of de arm op de positie van de huidige instructie is.
- **stopAllMotorFunctions** : Zorgt ervoor dat all servo's stoppen met bewegen. Dit betekent dat de robot arm zo snel mogelijk stil staat en dus niet meer beweegt.
- **getCurrentPosition** : Geeft de huidige positie terug van de AL5D robot arm waarin die zich op dat moment bevindt.
- **goToPosition** : Zorgt ervoor dat de AL5D robot arm naar een gespecificeerde positie gaat. Positie in graden kan voor elke gewricht en optioneel tijd is te specificeren. De gespecificeerde positie moet binnen de gespecificeerde range liggen van de gewrichten. Als dat niet zo is beweegt de robot arm niet en wordt dit teruggekoppeld dat de opgeven positie niet mogelijk is.

1.3.2.2 Low Level Driver

De **Low Level Driver** handelt communicatie af met de servo controller SCC32U van de AL5D robot arm. De driver biedt de volgende functionaliteit aan door middel van commando's te sturen over serial naar het servo controller bord:

- **move** : Stuurt een commando om een servo aan te sturen. Hoek in graden, optionele snelheid in ms en optionele tijd in ms kan

gespecificeerd worden. Bij het niet specificeren van snelheid of tijd gaat de servo zo snel mogelijk naar de opgeven positie.

- **setPositionOffset**: Stuurt een commando naar het servo controller bord aan om de offset van een servo aan te passen. Dit gaat doormiddel van een pulse breedte afwijking wat kan liggen tussen -100 en 100 uS. Dit staat ongeveer gelijk aan 15 graden.

1.4 Behavioral state machine

In het Behavioral State Machine Diagram hebben we een abstracte weergaven van de werkelijkheid beschreven. Bij dit gedrag hebben we onderscheid gemaakt tussen “MOVING” en “STAND_BY”. Het systeem zal terug keren naar “STAND_BY” als hij zijn positie heeft bereikt en of de Emergency stop wordt geactiveerd. Pas als het systeem in “STAND_BY” stand staat, kan hij een nieuwe opdracht uitvoeren uit de Queue.

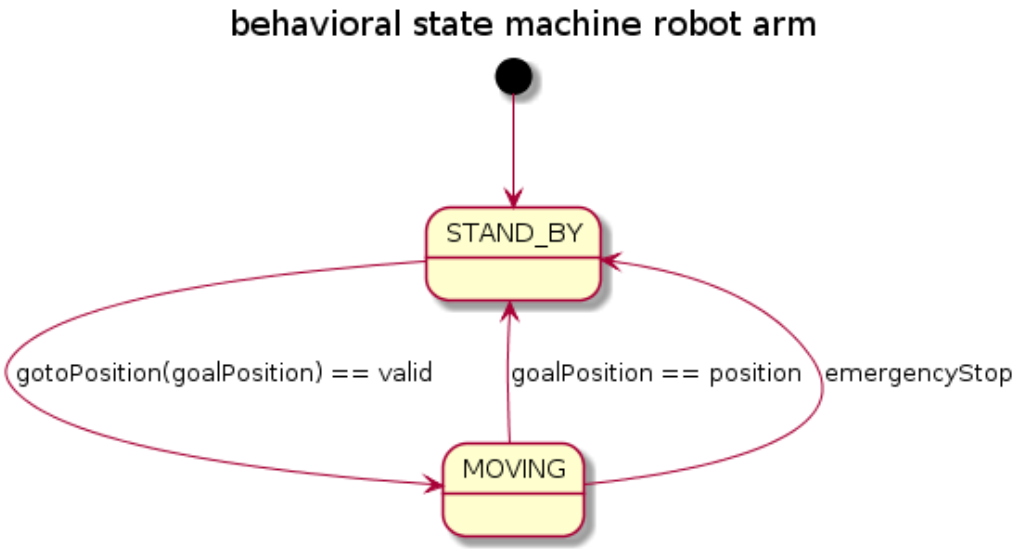

Figure 4: Behavioral state machine

1.5 Protocol State Machine

In het Protocol State Machine diagram hieronder wordt beschreven wat de pre- en postcondities zijn van elke state verandering van de applicatie. Bij elke transitie beschrijven wij wat de actie is die de transitie activeert. Beiden applicaties hebben onderling contact met de ROS Action messages, de gebruikte topic zijn “pose action”, “costum_pose_action” en “emergency”. De Robotarm controller stuurt een bericht naar de gepaste note met de gebruikers voorkeur, dit gebeurt in “Send_message”.

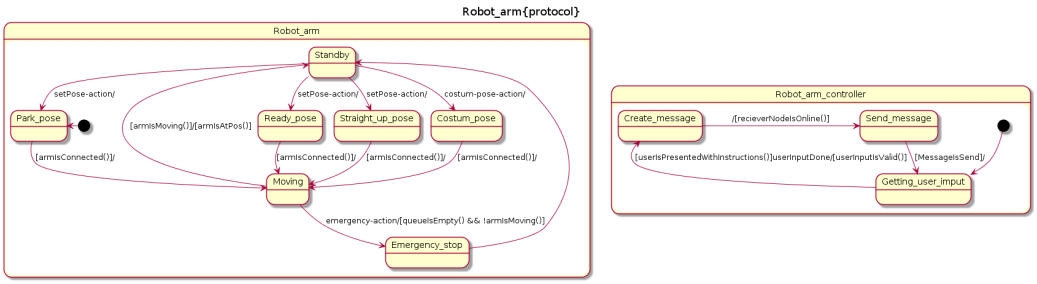

Figure 5: Protocol state machine

1.6 Timing van de Gripper

In het onderstaande Timings Diagram wordt het scenario weergegeven van de aansturing van de gripper. Dit diagram is een weergave van de worst case scenario. Waarbij er geen snelheid of tijd is gespecificeerd. In het diagram is daarnaast ook aangegeven wat het effect is van een andere baud rate en voltage van servo voor dit scenario. De eis van de opdracht was om de gripper naar een bepaalde locatie te sturen. Dat moet gebeuren binnen 2.3 seconden. Dit onderstaande diagram toont aan dat het haalbaar is. Binnen 1.45 seconden heeft de gripper de bestemming bereikt na het invoeren van de instructie bij de robotarm controller.

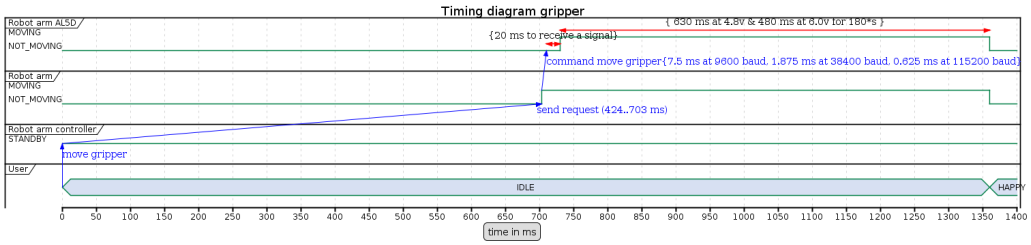

Figure 6: Timing diagram gripper

1.7 Initialisatie

Het Sequence Diagram is gemaakt ter ondersteuning voor het Timing diagram. Bij dit diagram gaan wij dieper in op de verschillende functie aanroepingen en de Alt/if eisen. Daarnaast beschrijven wij met behulp van notities de functionele eisen van de vrijheidsgraden.

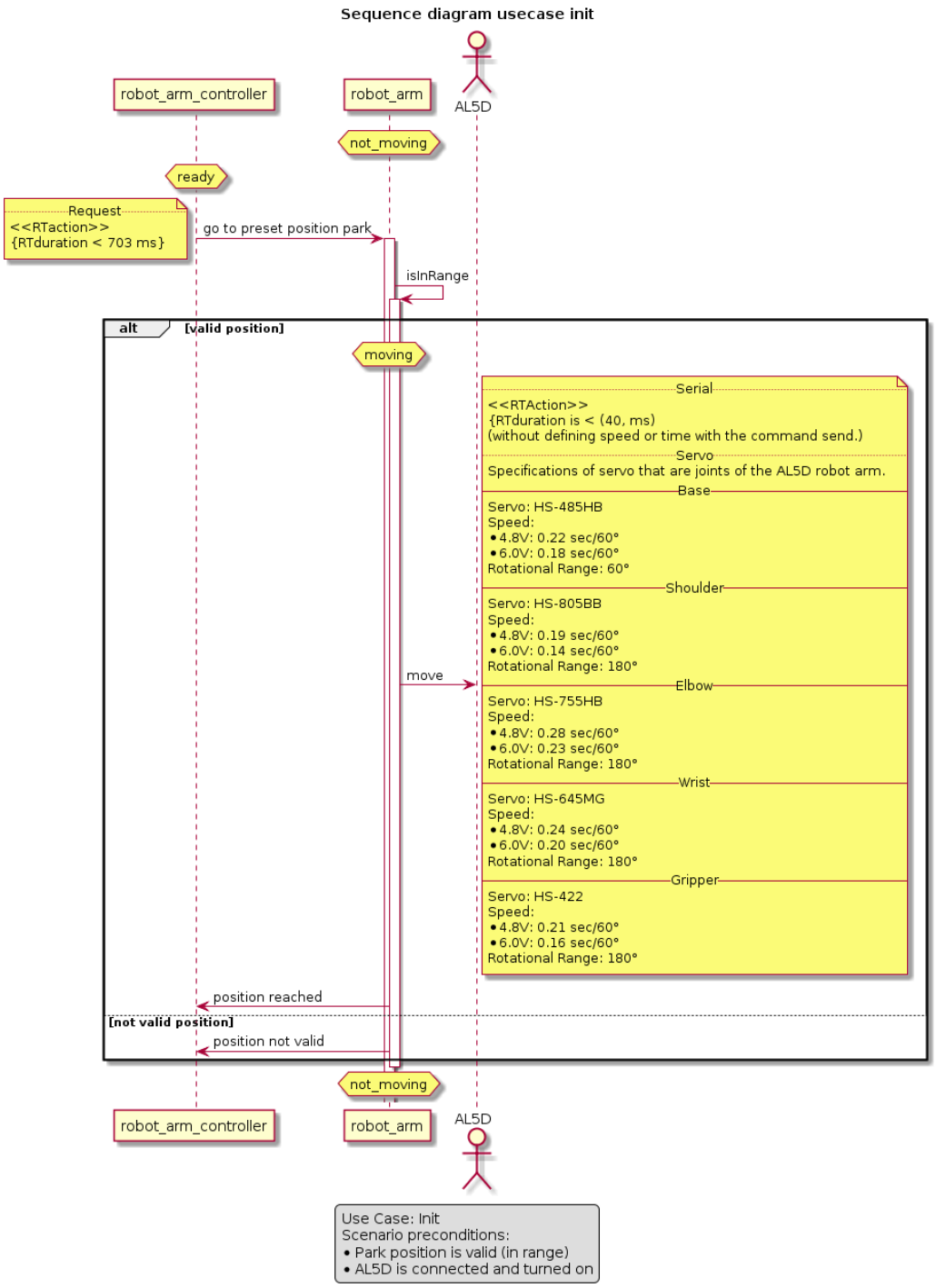

Figure 7: Sequence diagram init

1.8 Bijlage

1.8.1 Range of motion

Beperking van gewrichten:

Nr	Onderdeel	Positie	Hoek (deg)
0	Base	Left	-90
		Middle	0
		Right	90
1	Shoulder	Backwards	-30
		Vertical	0
		Horizontal	90
2	Elbow	Straight	0
		Sharp down	90
		Inwards	135
3	Wrist	Up	90
		Straight	0
		Down	-90
4	Gripper	Fully open	-
		Fully closed	-
5	Wrist rotate	Left	-90
		Middle	0
		Right	90

Figure 8: Range of motion tabel

1.8.2 Posities

Vooraf bepaalde posities:

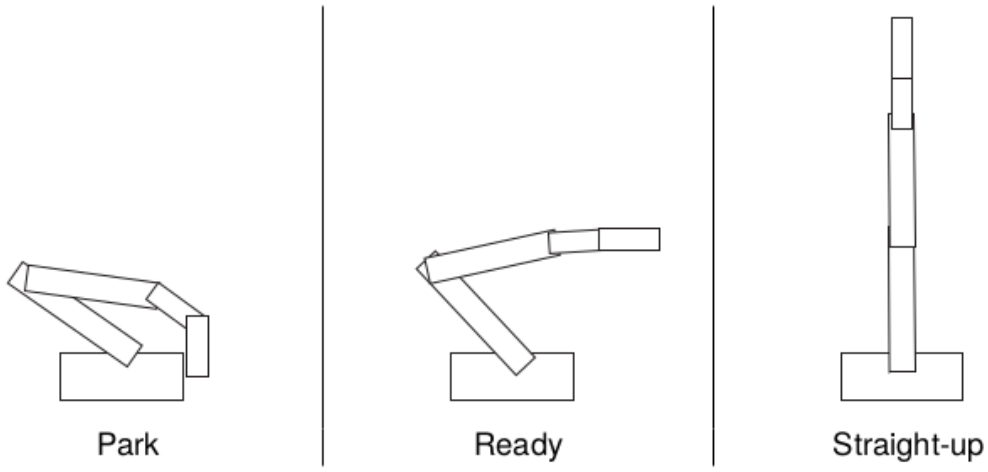

Figure 9: Abstracte afbeelding posities