Ontwerpdocument Robotarm Interface

WoR World

Inhoudsopgave

[Inleiding 3](#__RefHeading___Toc110_2075580451)

[Ontwerpbeslissingen 4](#__RefHeading___Toc112_2075580451)

[Initialisatie programma 4](#__RefHeading___Toc114_2075580451)

[Illegale commando’s 4](#__RefHeading___Toc116_2075580451)

[Usecase diagram 5](#__RefHeading___Toc118_2075580451)

[Component diagram 6](#__RefHeading___Toc120_2075580451)

[Sequence diagram 7](#__RefHeading___Toc122_2075580451)

[Protocol State Diagram 8](#__RefHeading___Toc124_2075580451)

# Inleiding

In dit document is het ontwerp voor de hardware interface opdracht van WoR World terug te vinden. Dit is opgedeelde in de verschillende diagrammen die gemaakt zijn met begeleidende tekst. Ook zijn hier de gemaakte ontwerpbeslissingen terug te vinden.

# Ontwerpbeslissingen

In dit hoofdstuk zijn de gemaakt ontwerpbeslissingen terug te vinden.

#### Initialisatie programma

Volgens de opdracht is het de bedoeling dat de robotarm bij het opstarten van het programma naar de “park” stand gaat. Echter is dit niet mogelijk omdat er in de [documentatie](http://www.lynxmotion.com/images/data/lynxmotion_ssc-32u_usb_user_guide.pdf) wordt aangegeven dat er aan het eerste commando geen snelheid meegegeven kan worden. Dit omdat de robotarm dan nog niet weet in welke stand hij staat.

“*Because the controller doesn't know where the servo is positioned on power-up, it will ignore speed and time commands until the first normal command has been received.*”

#### Illegale commando’s

Zoals in de opdracht aangegeven mogen er alleen waardes worden afgehandeld binnen de toegestane range of motion vallen. Als de waardes niet binnen de range valt zijn er echter twee opties, namelijk het limiteren op de hoogste/laagste toegestane waarde en die uitvoeren of het negeren van het commando. Wij hebben ervoor gekozen om in dit geval het commando te negeren.

# QoS Constraints

#### Bewijs SA03

In dit hoofdstuk wordst bewezen dat de constraint die in SA03 staat gehaald wordt door ons programamma. De constraint is als volgt gedefinieerd : “*Als de gripper van AL5D-robotarm naar een locatie (een set van samen-gestelde servohoeken 1 ) wordt gestuurd moet deze binnen 2,3 seconden worden bereikt.*”.

Om dit aan te tonen hebben we een aantal metingen gedaan waarin te zien is hoelang het duurt tussen het sturen van het commando tot het daadwerkelijk bereiken van de gestelde posititie.

Hierbij houden we rekening met de volgende aspecten:

* Tijd voor het versturen van een ROS message.
* Tijd voor het omzetten van de message naar een commando.
* Tijd voor het verzenden van het commando over de seriele poort.
* Tijd voor het versturen van de gewensde pulsbreedte
* Tijd voor het bewegen van de servo’s naar de gestelde positie.

Hierbij laten we een aspect buiten beschouwing omdat deze niet te meten is, dit is het vertalen van het seriele commando door de SSC-32U. Hierover is in de documentatie ook geen informatie te vinden. We gaan er vanuit dat dit geen tijd kost.

Voor de rest van de aspecten gaan we uit van de worst-case scenario.

Uit testen is gebleken dan deze aspecten zo lang duren:

ROS message : 175166 nanoseconden → 175.166 microseconden.

Omzetten message : ~30.000 nanoseconden → 0.03 microseconden.

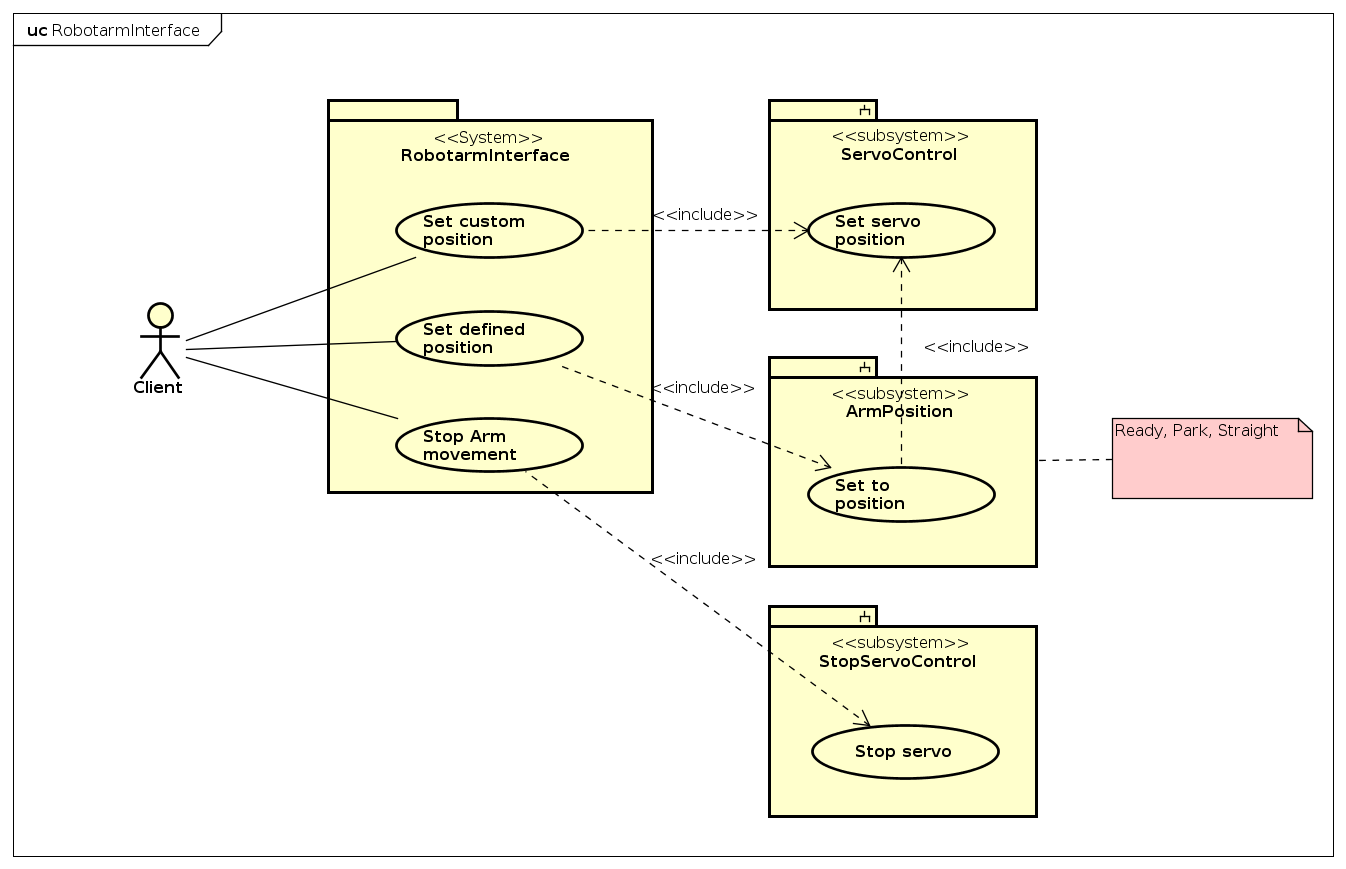
Verzenden commando : ~20.000 nanoseconden → 0.02 microseconden.

Pulsbreedte : 2.5ms.

Beweging servo (180°) : 0.19sec/60° → 3.16666666667ms/1° → 570.0000000006ms/180°.

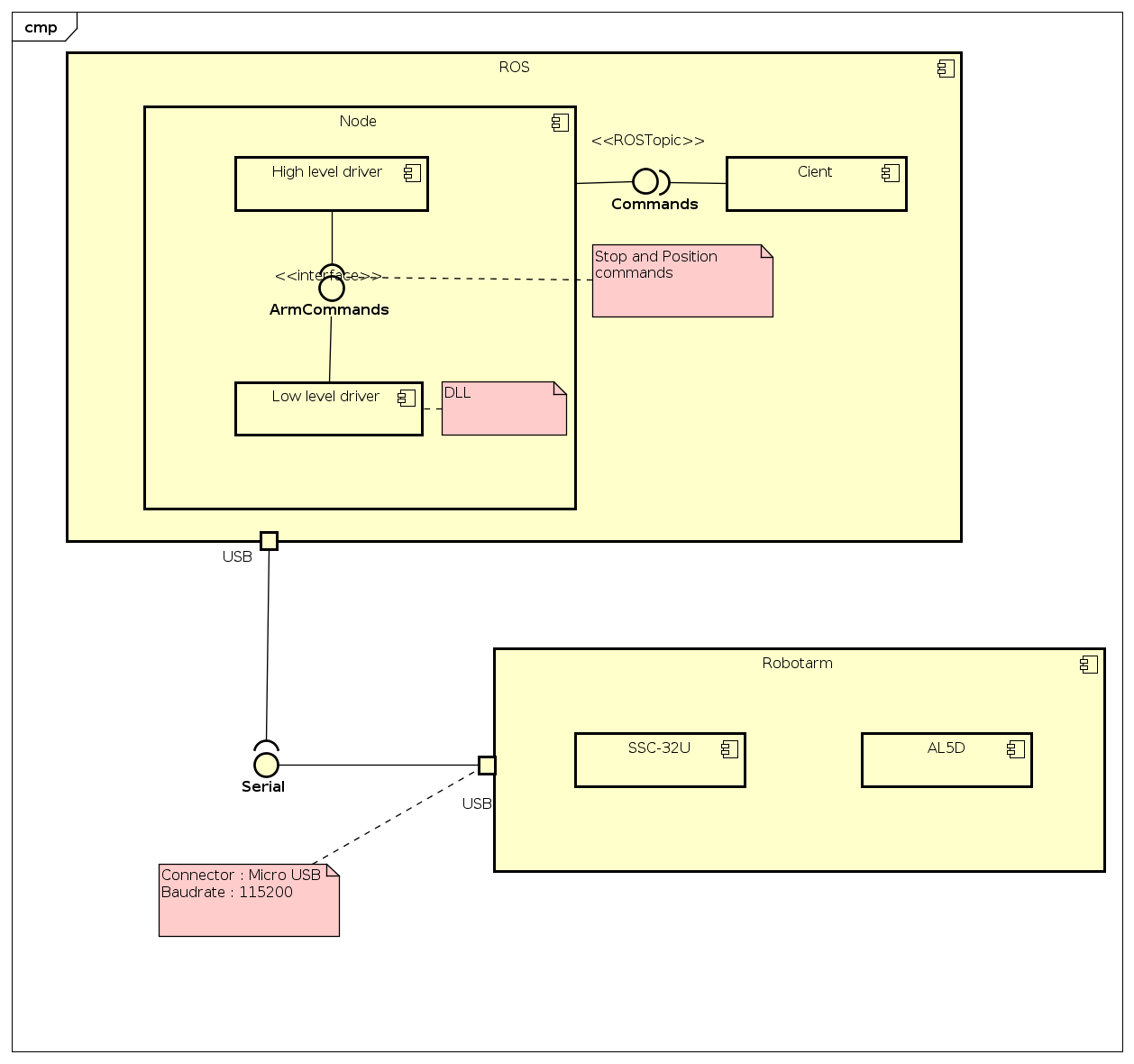
Totaal : 572.675ms.

# Usecase diagram

  
Afbeelding 1: Usecase Diagram

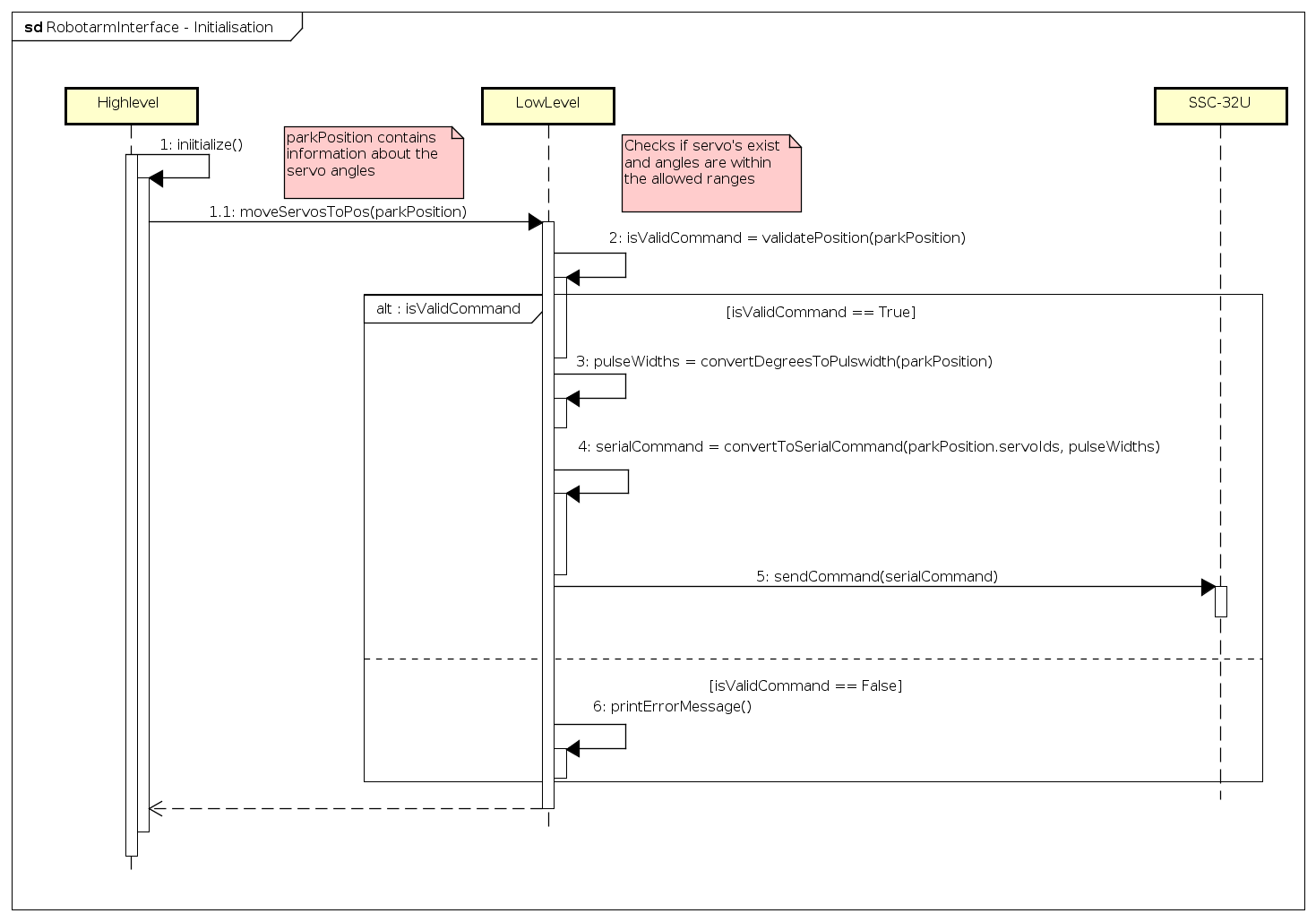
Binnen dit usecase diagram hebben we een client die gebruik wil maken van de robotarm. Deze kan de robotarm naar een van te voren ingestelde positie sturen, een zelfgekozen positie sturen of een lopende beweging stoppen. Bij het uitvoeren van deze usecases worden de benodigde usecases in de subsystemen aangeroepen. Zo bestaat er een subsysteem om een servo naar een bepaalde positie te zetten welke gebruikt wordt door zowel de van te voren ingestelde versie en de zelfgekozen versie. Voor het stoppen is een ander subsysteem gedefinieerd.

# Component diagram

  
Afbeelding 2: Component Diagram

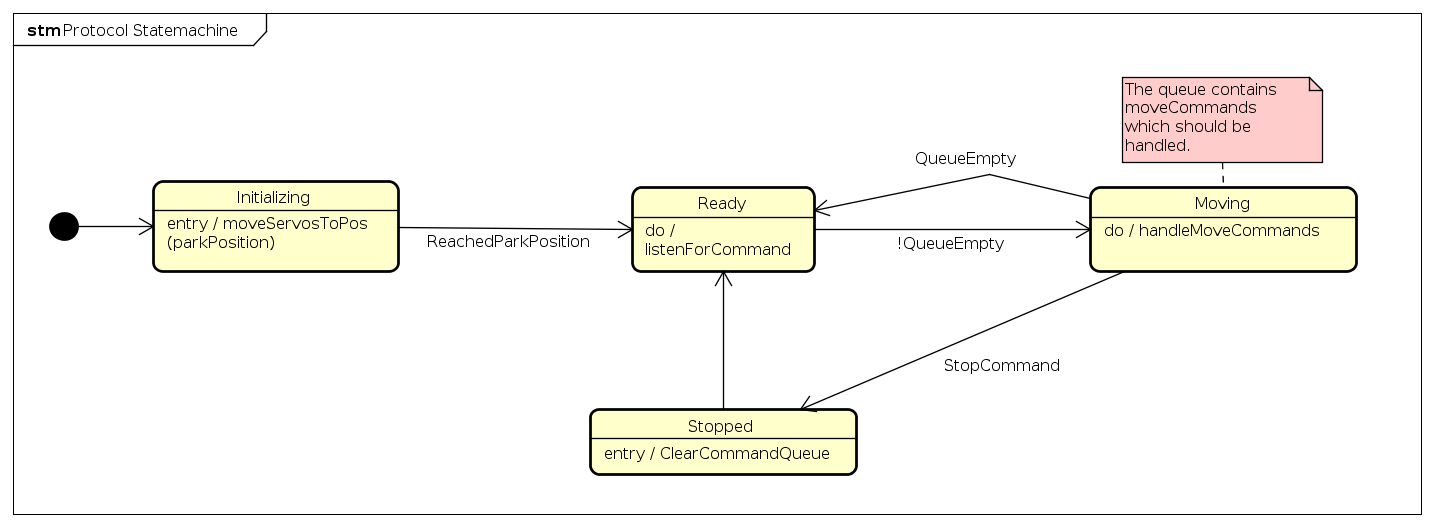
In dit component diagram zijn de verschillende componenten van het systeem te zien. De client kan gebruikt worden om commando’s sturen naar de high level driver sturen via een ROS topic. Binnen de high level driver worden deze commando’s omgezet naar aanroepen naar de low level driver. Deze vertaald de aanroepen naar een commando welke via de seriële poort naar de SSC-32U stuurt. Dit stuk hardware vertaald dit commando en stuurt de daadwerkelijke servo’s in de AL5D aan.

# Sequence diagram

  
Afbeelding 3: Sequence Diagram

In het bovenstaande diagram is de opstartprocedure van de robotarm te zien in de vorm van een sequence diagram. Hierbij wordt de arm bij het opstarten naar de “park” positie gestuurd. Dit commando wordt zoals alle andere commando’s eerst gecontroleerd. Als het commando valide is dan wordt deze vertaald naar een string welke naar de SSC-32U gestuurd kan worden en verstuurd. Wanneer het commando niet valide is wordt er een error gegenereerd en het commando genegeerd.

# Protocol State Diagram



In dit protocol state diagram zijn de verschillende states van het programma te erkennen en de toegestane staat veranderingen. Hier is te zien dat het programma bij het opstarten naar de initialisatie staat gaat om de robotarm in de “park” stand te zetten. Hierna wordt er geluisterd naar commando’s en worden deze toegevoegd aan een lijst welke wordt afgehandeld. Wanneer er tijdens het afhandelen een stop commando binnenkomt gaat het programma naar de stopped staat waarna deze weer gaat wachten op nieuwe commando’s in de ready staat.