Gebruikershandleiding

Geschreven door: André Slokker

1. Hoe te compileren?

Allereerst zijn de volgende software pakketten nodig:

- ROS Melodic + TF
- Rviz
- Boost 1_65

Ga naar de volgende locatie : Code/SimulationArm en voer hier het commando 'catkin_make' uit.

2. Starten demoscript

Ga naar de volgende locatie in de root map van dit project:

Code/SimulationArm en open in deze folder een terminal window.

Voer het volgende commando uit: source devel/setup.bash

Voer daarna het volgende commando uit:

roslaunch robotarm_description display.launch model:=lynxmotion_arm.urdf

3. Sturen commando’s naar de robotarm

"#0 P1500 T3000 #1 P1500 T3000 #2 P2450 T3000 #3 P800 T4000 #4 P1500 T5000 #5 P2000 T2000 \r\n"

Hierboven staat een voorbeeld van een commando dat naar de arm gestuurd kan worden. Deze commando's moeten volgens het SSC32U protocol zijn.

"#0" betekend servo 0

"P1500" betekend positie 1500 (in PWM)

"T3000" betekend dat hij er 3000 miliseconden over moet doen om naar die positie te gaan.

Belangrijk hierbij is om niet te vergeten de \r\n toe te voegen. Dit zorgt ervoor dat de parser weet dat dit het einde van het commando is.

De servo's hebben een range van 0 tot en met 5

De positie moet in PWM zijn. De posities hebben een range van 500 tot 2500.

De tijd is optioneel. Als er geen tijd in het commando staat, dan wordt er automatisch een tijd van 1000 milliseconden (de minimale waarde) toegekend.

Daarnaast is het ook niet nodig om alle servo's in het commando op te nemen. Dit commando is bijvoorbeeld ook valide:

"#1 P1000 #2 P1500 T1000"

4. Gerealiseerde eisen

PA02: Package is te bouwen met catkin op Melodic.

PA03: De applicatie wordt gebouwd met C++ volgens de Object Oriented principes die je geleerd hebt bij eerdere courses.

VS01: De virtuele controller luistert naar een topic waarop string messages in het formaat van de SSC32U 1 worden geplaatst.

VS02: De virtuele controller reageert op het topic (zie eis VS01) door bijbehorende joint_state messages te publiceren.

VS03: De virtuele robotarm wordt gevisualiseerd in Rviz (een URDFmodel van de arm is beschikbaar op OnderwijsOnline).

VS04: De virtuele robotarm gedraagt zich realistisch m.b.t. tijdgedrag (servo’s roteren kost tijd en gaat geleidelijk)

VC01: Er kan op een willekeurige plek in de virtuele wereld een bekertje geplaatst worden.

VC02: Publiceert een 3Dvisualisatie van het bekertje voor Rviz.

VC05: Visualiseert wanneer de gripper het bekertje vastheeft.

VC06: Het bekertje beweegt mee met de gripper (als hij vastgehouden wordt).

VC07: Bekertje is onderhevig aan zwaartekracht wanneer losgelaten.

VC08: Bekertje bepaalt en publiceert zijn positie.

DI01: Een demoscript stuurt over de tijd een sequentie van commando’s naar de armcontroller.

DM01: Beschrijft hoe de code gebouwd kan worden.

DM02: Beschrijft stap voor stap hoe de arm bewogen kan worden middels enkele voorbeelden.

DM03: Beschrijft welke eisen gerealiseerd zijn.

DD01: Beschrijft de structuur van de package (Nodes, topics, messages, etcetera).

DD02: Beschrijft de structuur en samenhang van de broncode
(classdiagrams, beschrijving, et cetera).