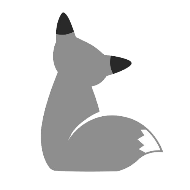
MOTORS CAPS EN ROCK’N ROLL

 Versie 0.9

Student: Matthijs Vos (1651350)

Stagedocent: Jorn Bunk

Stagebegeleider: Sjoerd Radstok

# Managementsamenvatting

* De samenvatting moet een op zichzelf staand onderdeel zijn
* Ga er in de samenvatting niet vanuit dat de lezer bekend is met de rest van het rapport. Of omgekeerd
* Richtlijn omvang: 1 a 1.5 A

Inhoud

[1 Managementsamenvatting 1](#_Toc21989932)

[2 Inhoud 2](#_Toc21989933)

[3 Inleiding 5](#_Toc21989934)

[4 Organisatorische Context 6](#_Toc21989935)

[4.1 Gibas Automation 7](#_Toc21989936)

[4.2 Bedrijfs- en Informatiesystemen 7](#_Toc21989937)

[4.2.1 Universal Robots 7](#_Toc21989938)

[4.2.2 Product Transportation System 7](#_Toc21989939)

[4.2.3 Festo Drive 7](#_Toc21989940)

[4.3 Bedrijfscultuur 8](#_Toc21989941)

[4.4 Taken en Positie van Stagiair 8](#_Toc21989942)

[5 De Opdracht 9](#_Toc21989943)

[5.1 Stageopdracht 9](#_Toc21989944)

[5.2 Businessdoel 9](#_Toc21989945)

[5.3 Doelstelling Opdracht 9](#_Toc21989946)

[5.4 Eindproduct 9](#_Toc21989947)

[5.5 Tussenproducten 9](#_Toc21989948)

[5.5.1 Gemaakte keuzes en onderbouwing 9](#_Toc21989949)

[5.6 Projectmethode 10](#_Toc21989950)

[5.7 Hoofd en Deelvragen 11](#_Toc21989951)

[6 Theoretisch Kader 12](#_Toc21989952)

[6.1 Universal Robots 12](#_Toc21989953)

[6.1.1 PolyScope 13](#_Toc21989954)

[6.1.2 URScript 13](#_Toc21989955)

[6.1.3 URCaps 13](#_Toc21989956)

[6.1.4 URControl 13](#_Toc21989957)

[6.2 Festo Drive 14](#_Toc21989958)

[6.3 Modbus 14](#_Toc21989959)

[6.4 Festo Configuratie Tool 14](#_Toc21989960)

[7 Requirements FestoCap 15](#_Toc21989961)

[7.1 Requirements FestoCap 15](#_Toc21989962)

[7.2 FestoCap Program Interface 16](#_Toc21989963)

[7.3 Communicatie tussen de FestoCap en de Festo motor 17](#_Toc21989964)

[7.4 User Interface 17](#_Toc21989965)

[7.5 Toolbar URCap 18](#_Toc21989966)

[7.6 Ondersteuning meerdere Festo motoren 18](#_Toc21989967)

[7.7 Requirements Ontwikkelomgeving 19](#_Toc21989968)

[7.8 Communicatie 19](#_Toc21989969)

[7.9 User Interface 19](#_Toc21989970)

[7.10 Conclusie 20](#_Toc21989971)

[8 Analyse 21](#_Toc21989972)

[8.1 URCaps API 22](#_Toc21989973)

[8.1.1 UR-Scriptic 23](#_Toc21989974)

[8.1.2 Models 23](#_Toc21989975)

[8.1.3 Communicatie 24](#_Toc21989976)

[8.1.4 User Interface 24](#_Toc21989977)

[8.2 Deployment 24](#_Toc21989978)

[8.3 Testopstelling 25](#_Toc21989979)

[8.3.1 Virtuele Testopstelling 25](#_Toc21989980)

[8.3.2 Fysieke Opstelling 25](#_Toc21989981)

[8.4 Conclusie 26](#_Toc21989982)

[9 Communication is key 27](#_Toc21989983)

[9.1 Modbus en Festo 27](#_Toc21989984)

[Home 27](#_Toc21989985)

[9.2 Modbus Software 28](#_Toc21989986)

[9.3 Demons 29](#_Toc21989987)

[9.4 Communicatie FestoCap naar Modbus 31](#_Toc21989988)

[9.5 Conclusie 31](#_Toc21989989)

[10 Making a FestoCap 32](#_Toc21989990)

[10.1 Sequence FestoCap 32](#_Toc21989991)

[10.2 URCap 33](#_Toc21989992)

[10.2.1 Installation Node 33](#_Toc21989993)

[10.2.2 Program Node 33](#_Toc21989994)

[10.3 Communicatie 34](#_Toc21989995)

[11 Conclusie 35](#_Toc21989996)

[12 Aanbevelingen 36](#_Toc21989997)

[13 Bronvermelding 37](#_Toc21989998)

[14 Bijlage 38](#_Toc21989999)

[14.1 Plan van Aanpak 38](#_Toc21990000)

[14.2 Festo motor DataSheet ( Temp want geen online versie beschikbaar) 38](#_Toc21990001)

[14.3 Requirements FestoCap 39](#_Toc21990002)

[14.4 API Functionaliteit 41](#_Toc21990003)

[14.5 Klassendiagrammen 42](#_Toc21990004)

[14.6 FestoCap ProgramNode Sequence UML 45](#_Toc21990005)

# Inleiding

# Organisatorische Context

Gibas is een bedrijf dat meerdere dienstenen oplossingen voor bedrijven aanbiedt. Van Draai en freesbanken tot Industriële robots. Daarom is het ook opgedeeld in vier verschillende branches; Numeriek, Automation, Tools en Logistiek. De stageopdracht wordt uitgevoerd bij Gibas Automation. Gibas Automation beschrijft haarzelf als:

De system integrator, gevestigd in Nijkerk, heeft een solide positie in de productie industrie. Bouwend vanuit een jarenlange expertise in de metaalverwerkende sector is een brede know-how ontwikkeld in meerdere branches. Gibas verstevigt zo haar positie als uw leverancier van totaal oplossingen.

Van gerobotiseerde machinebeladingen tot aan assemblage automatiseringen wordt maatwerk equipment vanuit een krachtige engineering geleverd. De interne service organisatie met meer dan 20 technici draagt verder zorg voor een jarenlange gegarandeerde support.

De klant stellen wij centraal en de oplossingen zijn breed gedragen. De integratie met topmerken als Fanuc Robotics, Universal Robots, Siemens ...... maken het geheel betrouwbaar en stabiel. (Gibas Automation, sd)

## Gibas Automation

Gibas Automation biedt verschillende diensten aan, zoals robotoplossingen, machinebeladingen, engineering, speciaal machinebouw en productieautomatisering. Het verspreidt zich dan ook over verschillende branches, van automotive & aerospace tot medical & laboratoria.

## Bedrijfs- en Informatiesystemen

Binnen Gibas komen een aantal systemen voor die betrekking hebben tot de stageopdracht. De Universal Robots(UR) producten worden gebruikt, deze worden in 3.1.1 beschreven. Ook wordt er gebruik gemaakt van URCaps, een systeem dat het mogelijk maakt randapparatuur en accessoires toe te voegen aan een UR product. Communicatie tussen de systemen is een belangrijk aspect in de automatisering, dit wordt vooral afgehandeld door Gibas het eigen Product Transportation System(PTS).

### Universal Robots

Een groot deel van de robotoplossingen van Gibas levert maakt gebruik van Universal Robots. Universal Robots levert robotarmen die toepasbaar zijn op alle productieniveaus met als doel de productiviteit te verhogen, letsel te verminderen en de werksfeer positief te houden. De Universal Robots zijn multi-inzetbaar en ook zeer geschikt voor het beladen van pallets. Vooruitstrevend door haar mobiliteit, de gemakkelijke programmering en interactie tussen mens en robot maakt dat er nieuwe mogelijkheden ontstaan met deze robots.

### Product Transportation System

Het PTS is software ontwikkeld door Gibas, wat het mogelijk maakt via een interface opdrachten in te voeren en deze opdrachten door het programma uit te laten voeren. De software wordt modulair opgebouwd zodat verschillende versies van het programma mogelijk zijn afhankelijk van de wensen van de klant.

Het PTS kan gebruikt worden om met verschillende type robotarmen, mobile robots of opslagsystemen te communiceren. Echter blijft het hoofdprogramma die de opdrachten verwerkt hetzelfde. Het PTS bevindt zich nog in bèta maar zal in de loop van de tijd uitgebreid worden met nieuwe functies.

### Festo Drive

Festo pneumatische actuatoren zijn normaal gesproken cilinders die bedoeld zijn voor lineaire bewegingen. Deze kunnen worden geleverd als standaard cilinder met zuigerstang (ISO 6432, ISO 15552, ISO 21287 of industriële standaard zoals bijvoorbeeld UNITOP) of met/ zonder zuigerstang volgens de specificaties van de fabrikant. De meeste cilinders kunnen eenvoudig worden geïnstalleerd in een bestaande machine met geschikt bevestigingsmateriaal / bevestigingsonderdelen. Ook worden deze geleverd met een motorcontroller en de benodigde documentatie voor de besturing van deze motoren.

## Bedrijfscultuur

Een organisatiecultuur kan beschreven worden met verschillende modellen. Het model van Harrison onderscheidt vier types: de machtsgerichte cultuur, de rollencultuur, de taakcultuur en de personeelscultuur:



Figuur 1 Model van Harrison

Binnen Gibas heerst er een Taakcultuur. Er is sprake van hoge machtsspreiding en een hoge samenwerkingsgraad, dit is terug te zien in het organogram (Bijlage 10.1) van het bedrijf maar ook op de werkvloer. De klanten die aangenomen worden hebben specifieke verzoeken en worden in aparte opdrachten aangenomen. Er zijn meerdere specialisten nodig om een opdracht voor een klant te voltooien. Tevens is er veel contact met de klant nodig om een op maat gemaakte oplossingen te leveren. (Toolshero, sd)

## Taken en Positie van Stagiair

Binnen de organisatie heeft de student een van tevoren gedefinieerde opdracht om te voltooien. Deze opdracht is een uitbreiding op een systeem dat al toegepast wordt op de producten die Gibas levert. De student wordt als software engineer aangenomen onder begeleiding van een software engineer.

# De Opdracht

## Stageopdracht

De stageopdracht is een productopdracht. De opdracht omvat het toevoegen van een festo drive aan de workflow van een Universal Robot(UR) Robotarm. Deze festo drive is in essentie een motor die het mogelijk maakt om een product op zijn platform te bewegen door de UR, zodat de UR niet verplaatst hoeft te worden of rare bewegingen moet maken om bij een bepaald punt te komen. Het UR systeem maakt gebruik van URCaps, een platform om randapparatuur en accessoires toe te voegen. Gibas heeft hier al software geproduceerd alleen vertoond deze een aantal kuren volgens de opdrachtgever.

## Businessdoel

Gibas levert op maat gemaakte oplossingen voor het automatiseren van de productie bij bedrijven. Het resultaat van de opdracht zal meerdere mogelijkheden toevoegen aan het assortiment en diensten van Gibas. De toevoeging van het product is een volgende stap in de automatisering en optimalisering van de UR producten die Gibas levert.

## Doelstelling Opdracht

De opdracht is op te delen in een onderzoek, analyse en productie. Allereerst zal er onderzoek gedaan worden naar de UR, URCaps ,festo drive en de communicatie tussen deze systemen, gevolgd door het analyseren van de geleverde code. Er zal een keuze gemaakt worden of er verder aan de bestaande code gewerkt wordt of dat er een nieuwe codebase wordt opgezet. Vervolgens zal er aan de productie van een systeem gewerkt worden, afhankelijk van het resultaat van het onderzoek en de analyse van de code zal er gewerkt worden aan de bestaande code of aan een geheel nieuwe codebase.

## Eindproduct

Het uiteindelijke eindproduct betreft een werkend systeem dat op een UR product gebruik maakt van de URCaps om een festo drive te kunnen besturen.

## Tussenproducten

Omdat het project een agile methode toepast zal het resulteren in meerdere tussenproducten gedurende het ontwikkeltraject. De volgende tussenproducten zijn vastgesteld om op te leveren tijdens de stage.

### Gemaakte keuzes en onderbouwing

Voordat de daadwerkelijke ontwikkeling van het systeem zal plaatvinden moet eerst het benodigde onderzoek gedaan worden naar de het bestaande systeem en alle randsystemen die er mee communiceren. Dit resulteert in een document met het onderzoek naar de verschillende systemen. Uiteindelijk zal er een onderbouwde keuze voortgebracht worden om of de code uit te breiden, of opnieuw te beginnen.

## Projectmethode

Er is gekozen voor een agile aanpak door de lastig te bepalen projectplanning en doordat er veel wijzigingen kunnen komen in de scope. Ook is het waardevol om net werkbare tussenresultaat te kunnen tonen aan de opdrachtgever. Agile software ontwikkeling heeft verschillende kenmerken die van toepassing zijn aan de eisen van zowel het project, de opdrachtgevers en de student (met betrekking tot het persoonlijk ontwikkelplan). Binnen agile development wordt er gebruik gemaakt van iteraties om risico’s te verminderen. Het doel van elke iteratie is om iets bruikbaars op te leveren. Hierdoor is goed vast te stellen met de stagebegeleider of het project correct verloopt en of er mogelijke veranderingen nodig zijn.

De stageperiode wordt opgedeeld in verschillende sprints, waarin elke sprint een afgebakende periode is met vaste onderdelen. Elke sprint duurt een week, en de sprints volgen elkaar steeds direct op. Aan het begin van de sprint is er een sprint planning waarin wordt vastgesteld wat er komende sprint wordt gedaan. Verder is er dagelijks sprake van een Daily Stand-up waarin de stagiaires voorleggen wat er gedaan is, wat er goed ging, wat er beter kan en wat er die dag gedaan wordt. Aan het eind van de sprint zal er een review en een retrospective plaatsvinden waar wordt teruggekeken op de sprint.

## Hoofd en Deelvragen

De hoofdvraag voor deze stageopdracht is als volgt geformuleerd:

Hoe kan doormiddel van een URCap randapparatuur aangestuurd worden voor het gebruik met een UR systeem **voor Gibas**?

Om deze vraag te kunnen beantwoorden zijn de volgende deelvragen opgesteld :

1. Wat zijn de requirements van de FestoCap?
2. Is de huidige ontwikkelomgeving geschikt voor het produceren van de FestoCap?
3. Hoe werkt het communicatieprotocol van het UR systeem met het Festo drive systeem?
4. Hoe zijn de eerder gevonden resultaten omgezet naar een werkende FestoCap met in acht genomen de gevonden requirements?

# Theoretisch Kader

Het doel van het theoretisch kader, vanuit de stageleidraad genomen, is (voor)onderzoek doen. Het opstellen van een onderzoek om de benodigde theorie, begrippen en concepten te kunnen definiëren en onderbouwen voor de voltooiing van de stageopdracht. Dit wordt gebaseerd op betrouwbare literatuur met correcte literatuurverwijzingen. Ook wordt de belangrijkste literatuur die nog onderzocht moet worden voor de uitwerking en beantwoording van de deelvragen vastgesteld. Er worden verschillende systemen gebruikt bij de stageopdracht. Om een duidelijk beeld te creëren van het UR-systeem en het onderbouwen hiervan zullen deze omschreven worden

## Universal Robots

Bij de stageopdracht wordt er gewerkt met de systemen van Universal Robots. Het onderzoek van deze systemen benut het ontwikkelproces van een URCap. Een Universal Robot kan op twee niveaus bestuurd worden op GUI niveau, door middel van de PolyScope (of eigen software) en op Script niveau, door middel van URScript.

### PolyScope

PolyScope is de grafische programmeer interface van Universal Robots. Het is de interface die de programmeur gebruikt om een UR-programma te bouwen en besturen. URCaps zijn plug-ins binnen PolyScope. Er zijn momenteel twee versies van PolyScope die geleverd worden door Gibas. Welke versie wordt gebruikt zal geen verschil zijn in de backend van het systeem. Alleen de user interface is wel anders per versie, er zal tijdens het project onderzocht worden hoe deze verschillen en er zal uiteindelijk een user interface voor beide versies van het systeem komen. Een groot deel van het systeem wordt uitgevoerd in PolyScope, er zal genoeg vooronderzoek gedaan moeten worden naar het optimaal gebruik van deze software. Universal Robots stelt veel hulpmiddelen beschikbaar voor programmeurs , ‘Getting Started’-tutorials en fora voor verdere vragen.

### URScript

Op Script niveau wordt URScript gebruikt om de UR te besturen. URScript is een programmeertaal die gebruik maakt van variabelen, types en flow control statemens zoals “if” en “else”. Verder zijn er ook variabelen en functies beschikbaar voor het monitoren en controleren van de I/O en de bewegingen van de robot. Dit is van toepassing bij het testen het systeem en de communicatie vanuit het UR systeem (Universal Robots, 2018)

### URCaps

Een URCap is een Java-based plugin voor PolyScope. Deze kan gebruikt worden om functionaliteit toe te voegen aan PolyScope, bijvoorbeeld specifieke hardware configuraties of nieuwe templates voor operaties of communicatie. De URCaps komen in het algemeen in vier soorten : Installation Node, Program node, Toolbar en Daemon (Universal Robots)

#### Installation Node

Een Installation Node is een extensie op het installatie domein van het UR systeem. Hierin kan een URCap aanpassingen aan de user interface of extra functionaliteiten toevoegen aan de configuratie van het PolyScope systeem.

#### Program Node

Een Program Node is een extensie op het programmeer domein van het UR systeem. Hier kan de URCap nieuwe programma’s toevoegen aan PolyScope .

#### Toolbar

Een Toolbar is een toevoeging aan de user interface in de vorm van een pop-up menu dat vanuit het gehele programma beschikbaar is. Deze is vaak beschikbaar als er sprake is van randapparatuur als mogelijke manual override die altijd te bereiken is.

### URControl

URControl is de real-time controller, deze bestuurt de robot en voert de programma’s uit. In essentie krijgt URControl een programma van PolyScope en zet deze om naar individuele machine instructies. Het is een low-level robot controller op een PC in de UR. Wanneer de UR opstart zal deze als een service(daemon) opstarten en verbinden met PolyScope via TCP/IP connectie (Universal Robots, 2018)

## Festo Drive

Het project moet uiteindelijk een motor van Festo aansturen. Deze worden geleverd met motor controller en een Festo Handling and Positioning Profile(FHPP). In dit FHPP wordt grondig uitgelegd hoe met de motor controller gecommuniceerd wordt. Om de functionaliteiten van de festo drive beter vast te stellen voor een bruikbare interface zal er testsoftware geschreven worden voor het aanroepen van alle benodigde functionaliteit. (Festo)

## Modbus

Modbus is een serieel communicatie protocol ontworpen door Modicon in 1979 voor het aansturen van programmable logic controllers(PLCs). Het is een methode die gebruikt wordt voor het oversturen van informatie over seriële lijnen naar elektronische apparaten. Het is een open protocol en kan dus vrij gebruikt worden. Het is zelfs in de huidige tijd een veelgebruikte standaard in de industrie.

**(MEER UITLEG OVER MODBUS MET REGISTERS)**

## Festo Configuratie Tool

Voor de geleverde motor was geen duidelijke documentatie beschikbaar omdat hier geen toegang tot was en er binnen het bedrijf niet binnen een korte tijd toegang te regelen was. Als alternatief is de Festo Configuratie tool voorgesteld, deze tool wordt gebruikt door de PLC programmeurs binnen Gibas om de motoren in te stellen. Vanuit hier is de Festo drive aan te spreken en daarbij de motor aan te sturen. In combinatie met de documentatie van andere motoren en de FTC

# Requirements FestoCap

“Weeks of programming can save you hours of planning”

Voordat er een “Hello World” wordt geschreven zal er eerst vastgesteld moeten worden wat een FestoCap nou precies moet doen. In dit hoofdstuk worden de voorbereidende maatregelen getroffen voor het produceren van de FestoCap. Om een product te maken die aan de eisen van Gibas voldoet worden er requirements aan de FestoCap gesteld. Deze, vaak technische, requirements schetsen een duidelijk beeld van de FestoCap en wat er nodig is om deze te maken. Requirements kunnen gebruikt worden om de geleverde ontwikkelomgeving te toetsen, of deze voldoende is om een FestoCap de produceren. Wanneer dit niet het geval is, stelt het mij tijdig in staat om met een oplossing of alternatief te komen.

In dit hoofdstuk worden de deelvraag; “*Wat zijn de requirements van de FestoCap*” beantwoordt.

## Requirements FestoCap

Om vast te kunnen stellen of de ontwikkelomgeving geschikt is voor het produceren van een URCap naar de wens van Gibas is het nodig om requirements vast te stellen. Een groot deel van deze requirements zijn al vastgesteld in het plan van aanpak **(Verwijzing naar PVA 8.2.1)** , maar zijn verder uitgewerkt tijdens de ontwikkeling van de FestoCap. In het plan van aanpak zijn de volgende generieke requirements opgesteld :

* FestoCap program Interface
* Daemon URCap
* Festo Modbus Controller
* Communicatie tussen de FestoCap en de Festo motor
* Toolbar URCap
* User interface URCap
* Ondersteuning meerdere Festo Motoren

Deze requirements zijn samen met de stagebegeleider in gesprek vastgesteld, toen was er nog niet genoeg kennis om de specifieke requirements en mogelijkheden van een URCap en van de Festo Controller vast te stellen. In deze deelvraag worden deze generieke requirements omgezet kunnen worden naar functionele requirements voor de FestoCap.

Tijdens de stageopdracht is onderzocht of de requirements voor de ontwikkelomgeving daadwerkelijk haalbaar zijn met de huidige ontwikkelomgeving. Als een requirement niet te verwezenlijken is binnen de huidige ontwikkelomgeving dan wordt er onderzocht of er een programma geschreven kan worden of externe library dit wel mogelijk maakt.

## FestoCap Program Interface

Tijdens de bespreking van de functionaliteiten van de FestoCap kwamen de basisfunctionaliteiten al snel aan bod. Met een Festo Program Interface wordt de besturen van een Festo motor vanuit het programma beduidt, in dit geval vanuit een URCap. Hier komen al functionele requirements naar voren voor de FestoCap. De verbinding tussen de FestoCap en de Festo motor wordt behandeld in (JUISTE KOP NUMMER). Het moet mogelijk zijn de Festo motor volledig te kunnen besturen vanuit de FestoCap, om deze besturing goed vast te kunnen stellen wordt er naar de datasheet van de Festo motor gekeken (BRON NAAR DATASHEET). In deze datasheet staan de functionaliteiten van de Festo motor, deze zijn gelijk te vertalen naar functionele requirements voor de FestoCap. De functionaliteiten van de Festo motor zijn op te deze in de control functionaliteiten en de status functionaliteiten. De controle functionaliteiten zijn vanzelfsprekend gericht op het aansturen van de Festo motor, waarin de status functionaliteiten het mogelijk maken de status op te vragen van de Festo motor. Vanuit deze datasheet zijn de volgende functionaliteiten gehaald(Bijlage 13.2-Controle). Deze functionaliteiten zijn als volgt omgezet naar eisen voor de FestoCap.

|  |  |
| --- | --- |
| FR1.1 | De FestoCap moet een Festo motor kunnen activeren |
| FR1.2 | De FestoCap moet een Festo motor kunnen stoppen tijdens een beweging |
| FR1.3 | De FestoCap moet een Festo motor naar een absolute positie kunnen verplaatsen |
| FR1.4 | De FestoCap moet een Festo motor naar een relatieve positie kunnen verplaatsen |
| FR1.5 | De FestoCap moet een Festo motor continu in beide richtingen kunnen bewegen |
| FR1.6 | De FestoCap moet een error in de motor kunnen herkennen en verwerken |

Om de status van de Festo motor te monitoren beschikt de Festo motor over de volgende functionaliteiten volgens de datasheet(Bijlage 13.2-Status). Deze functionaliteiten zijn als volgt omgezet naar eisen voor de FestoCap.

|  |  |
| --- | --- |
| FR1.7 | De FestoCap moet de status van de Drive op kunnen vragen |
| FR1.8 | De FestoCap moet de status van de huidige operatie op kunnen vragen |
| FR1.9 | De FestoCap moet een warning of malfunction in de Festo motor kunnen herkennen |
| FR1.10 | De FestoCap moet op kunnen vragen of de Festo motor enabled kan worden. |
| FR1.11 | De FestoCap moet op kunnen vragen of Halt geactiveerd is |
| FR1.12 | De FestoCap moet op kunnen vragen of de Festo motor klaar is om te start |
| FR1.13 | De FestoCap moet op kunnen vragen of de bewegingsopdracht voltooid is |
| FR1.14 | De FestoCap moet op kunnen vragen of de bewegingsopdracht een absolute of relatieve positie is |
| FR1.15 | De FestoCap moet de huidige positie van de Festo motor op kunnen vragen |
| FR1.16 | De FestoCap moet de ingestelde snelheid van de Festo motor op kunnen vragen |

## Communicatie tussen de FestoCap en de Festo motor

Voor het aansturen van de Festo motor moet er de mogelijkheid zijn om berichten te kunnen versturen en ontvangen van Festo motor. Om dit te kunnen realiseren moet er naar de beschikbare software aan zowel de kant van de FestoCap als die van de Festo Motor. Vervolgens moet er ook vastgesteld worden hoe deze berichten er uit komen te zien. Ten slotte is het ook belangrijk om een verbinding tussen de programma’s te onderhouden, natuurlijk is een stabiele verbinding vereist maar de FestoCap moet ook functioneren als de verbinding wegvalt. Zo zijn er de volgende requirements aan de FestoCap gesteld betreffende connectie en communicatie :

|  |  |
| --- | --- |
| FR2.1 | De FestoCap moet kunnen verbinden met een Festo motor |
| FR2.2 | De FestoCap moet de verbinding kunnen verbreken met een Festo motor |
| FR2.3 | De FestoCap moet verbindingsfouten kunnen herkennen en weergeven aan de gebruiker |
| FR2.4 | De FestoCap moet berichten kunnen versturen naar een Festo motor |
| FR2.5 | De FestoCap moet berichten kunnen ontvangen van een Festo motor |

## User Interface

Omdat de FestoCap een product is dat ingezet wordt door Gibas zal er ook een uiteindelijk een duidelijk user interface moeten komen. Grafisch Design valt niet onder de stageopdracht, maar het is wel tot een bepaalde mate van toepassing om de functies van de FestoCap duidelijk weer te geven. Na meerdere besprekingen met de stagebegeleider zijn de volgende requirements voor de user interface naar voren gekomen:

|  |  |
| --- | --- |
| FR3.1 | De gebruiker moet vanuit de user interface een positie kunnen invoeren voor de Festo motor |
| FR3.2 | De gebruiker moet vanuit de user interface een variabele in PolyScope als positie  voor de Festo motor invoeren |
| FR3.3 | De gebruiker moet vanuit de user interface een home commando uit kunnen voeren |
| FR3.4 | De gebruiker moet vanuit de user interface aan kunnen geven of de positie relatief is |
| FR3.5 | De gebruiker moet vanuit de user interface aan kunnen geven of de positie absoluut is |
| FR3.6 | De gebruiker moet vanuit de user interface aan kunnen geven of PolyScope moet  wachten tot het commando uitgevoerd is |
| FR3.7 | De gebruiker moet vanuit de user interface een IP in kunnen voeren om mee te verbinden |
| FR3.8 | De gebruiker moet vanuit de user interface een Port in kunnen voeren om mee te verbinden |
| FR3.9 | De gebruiker moet vanuit de user interface een enable commando uit kunnen voeren |
| FR3.10 | De gebruiker moet vanuit de user interface kunnen verbinden met een Festo motor |
| FR3.11 | De gebruiker moet vanuit de user interface de verbinding met een Festo motor kunnen verbreken |
| FR3.12 | De gebruiker moet vanuit de user interface kunnen selecteren of er verbonden moet  worden bij het opstarten van PolyScope |
| FR3.13 | De gebruiker moet vanuit de user interface de huidige positie van een Festo motor kunnen zien |
| FR3.14 | De gebruiker moet vanuit de user interface de status van de verbinding kunnen zien |

## Toolbar URCap

Vanuit versie 5.x van PolyScope is het mogelijk een Toolbar toe te voegen aan PolyScope. Een Toolbar is een drop-down menu dat ten alle tijden binnen PolyScope geopend kan worden. De Toolbar is een aparte node vergelijkbaar met de Program en Installation Nodes, alleen heeft deze een versimpelde API en is niet gebonden aan URCommand. Deze Toolbar kan toegepast worden bij de FestoCap om directe commando’s naar de motor te geven zonder er een volledig programma binnen PolyScope voor uit te voeren, bijvoorbeeld om de positie van de motor de verstellen of home opnieuw in te stellen. Omdat de Toolbar een Node op zich is komen er zowel functionele eisen als eisen voor de user interface voor in de requirements van de Toolbar. Uit overleg met de stagebegeleider en collega’s zijn de volgende requirements voor de Toolbar naar voren gekomen :

|  |  |
| --- | --- |
| FR4.1 | De gebruiker moet vanuit de Toolbar interface de huidige positie van een Festo motor kunnen zien |
| FR4.2 | De gebruiker moet vanuit de Toolbar interface de status van de verbinding kunnen zien |
| FR4.3 | De gebruiker moet vanuit de Toolbar interface een stap commando uit kunnen voeren |
| FR4.4 | De gebruiker moet vanuit de Toolbar interface de grootte van de stap in kunnen voeren |
| FR4.5 | De gebruiker moet vanuit de Toolbar interface de Festo motor positief laten Joggen door een knop in te houden |
| FR4.6 | De gebruiker moet vanuit de Toolbar interface de Festo motor negatief laten Joggen door een knop in te houden |
| FR4.7 | De gebruiker moet vanuit de Toolbar interface een home commando uit kunnen voeren |
| FR4.8 | De Toolbar moet duidelijk te identificeren zijn als de Toolbar voor een Festo motor |

## Ondersteuning meerdere Festo motoren

Om meerdere Festo motoren te kunnen ondersteunen moet de FestoCap dynamisch opgesteld worden. Dit betekend dat de FestoCap moet kunnen functioneren ongeacht van de verbonden motor, en het moet mogelijk zijn voor de gebruiker om verschillende motoren te kunnen selecteren. Ook komt de uitdaging om de verschillende functionaliteiten van de motoren vast te stellen, er is natuurlijk een duidelijk verschil in ene lineaire motor en roterende. Bepaalde commando’s zouden ook anders zijn afhankelijk van de motor. Het is belangrijk dat de FestoCap deze afwijkingen in motoren kan verwerken zodat er geen nieuwe URCap aangemaakt hoeft te worden voor elke nieuwe motor. De volgende requirements voor de ondersteuning van meerdere Festo motoren zijn vastgesteld :

|  |  |
| --- | --- |
| FR5.1 | De engineer moet, zonder de FestoCap aan te passen ,een nieuwe Festo motor klasse toe kunnen voegen |
| FR5.2 | De gebruiker moet een Festo motor kunnen selecteren in de user interface |
| FR5.3 | De FestoCap moet een interface klasse hebben die het mogelijk maakt nieuwe Festo motoren toe te voegen |
| FR5.4 | De FestoCap moet onderscheid kunnen maken tussen de verschillende Festo motoren |

## Requirements Ontwikkelomgeving

De requirements aan de ontwikkelomgeving zijn eisen die de ontwikkeling van de FestoCap mogelijk maakt. Om de requirements van de ontwikkelomgeving vast te stellen kunnen de functionele requirements voor de FestoCap gebruikt worden. Veel van de requirements van de FestoCap kunnen direct vertaald naar requirements voor de ontwikkelomgeving, bijvoorbeeld :“*Het moet mogelijk zijn om FR.3.1 te kunnen implementeren met de geleverde SDK*”. Sommige requirements aan de FestoCap kunnen vertalen naar dezelfde requirement aan de Ontwikkelomgeving, deze zullen ook samen behandeld worden, in bijlage(x) is een volledige lijst van requirements te vinden.

## Communicatie

Een cruciaal deel van dit project is de communicatie. Er moet uiteindelijk een motor bestuurd worden vanaf de FestoCap, dit is niet mogelijk zonder goede communicatie tussen de twee systemen. In de requirements voor de FestoCap is al vastgesteld dat deze verbinding mogelijk moet zijn en te onderhouden. De requirements beschrijven een simpele stabiele connectie als eis. Er worden berichten verzonden en ontvangen, vanzelfsprekend moet er ook gehandeld worden wanneer de connectie verbroken wordt, of deze vertraagd. De volgende eisen aan de Communicatiemogelijkheden van de Ontwikkelomgeving zijn vastgesteld.

* Er moet de mogelijkheid zijn om een URCap te produceren die kan verbinden met een webserver via een socketverbinding
* De Festo motor moet via een, bij voorkeur TCP, verbinding aangesloten kunnen worden aan het UR Systeem
* Er moet de mogelijkheid zijn om commando berichten te versturen naar de webserver en statusberichten op de vragen van andere sockets.
* Er moet ondersteuning zijn voor het Modbus protocol dat gebruikt wordt in de Festo motor.

## User Interface

Omdat de FestoCap uiteindelijk geleverd wordt in met de producten van Gibas, is een functionele user interface van groot belang. Zoals in het PvA besproken zal er minimale aandacht besteed worden aan de weergave maar wordt er vooral aan de functionaliteit van de interface gewerkt. Uiteindelijk moet het mogelijk zijn de om met een Festo motor te kunnen verbinden en alle functionaliteiten van de motor aan te kunnen sturen vanuit deze interface. Omdat er gewerkt wordt met de SDK van UR, die zijn eigen interface library heeft, is het van belang van tevoren vast te stellen welke eisen er aan de interface van de FestoCap zitten. Wanneer deze eisen niet voldaan kunnen worden zal er gekeken worden of er een alternatief mogelijk is binnen of buiten de SDK.

Vanuit de eisen van de FestoCap komen al basisfunctionaliteiten naar voren waar , na een kort vooronderzoek, vanuit wordt gegaan dat deze mogelijk zijn binnen de interface van de UR. Dit betreft functionaliteiten als; aanpassen van de lay-out, vernieuwen van de lay-out at run-time , eigen variabelen weergeven, waardes real-time updaten. Er zijn ook eisen waar onderzoek gedaan moet worden naar de mogelijkheden, bijvoorbeeld de variabelen binnen PolyScope.

## Conclusie

De FestoCap is een stap verder van de gebruikelijke URCap. De FestoCap moet een duidelijke interface bevatten voor het vertalen van alle mogelijke commando’s van de motoren. Berichten die door de FestoCap verzonden en ontvangen worden moeten juist afgehandeld worden binnen PolyScope. Op de communicatie na zijn de overige eisen redelijke basisfunctionaliteiten waar de meeste programmeertalen al ondersteuning voor bieden.

# Analyse

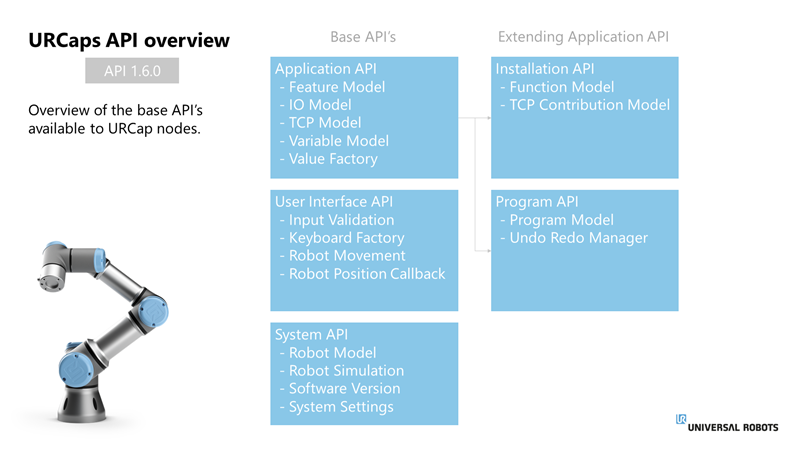
In deze analyse worden de requirements die eerder opgesteld zijn naast de mogelijkheden van de ontwikkelomgeving gezet en wordt er gekeken of deze voldoen. Ook wordt er naar de andere mogelijkheden van de ontwikkelomgeving gekeken en hoe een URCap gegenereerd wordt. Ten slotte word de geleverde testopstelling onder de loep genomen.

Om te kunnen ontwikkelen voor de PolyScope is door Universal Robots een ontwikkelomgeving opgesteld. Deze omgeving bestaat uit een Ubuntu image met een van te voren ingestelde programmeeromgeving met de URCap SDK, een simulator en documentatie over het ontwikkelen van URCaps. De URCap SDK bevat een shellscript genaamd ‘newURCap.sh” die een wizard opstart om een nieuwe URCap te creëren. Ook bevat de image de tutorials voor het ontwikkelen van de verschillende soorten URCaps, in deze tutorials wordt er grondig ingegaan op het gehele proces en de werking van de individuele klassen die gebruikt worden.

In dit hoofdstuk worden de deelvraag; “Is de huidige ontwikkelomgeving geschikt voor het produceren van de FestoCap?” beantwoord.

## URCaps API

De URCaps API geeft een programmeur toegang tot de functionaliteiten binnen PolyScope. Zoals eerder beschreven ( verwijzing naar theoretisch kader ) maakt de API gebruik van contributions. Deze contributions zijn services die ieder zijn eigen functionaliteit bieden. De vier types contributions zijn: Installation Node, Program Node, Toolbar en Daemon. De API die meegegeven wordt aan deze nodes is het Domain API. Deze bestaat weer uit drie aparte API lagen; Application, User Interface en System API’s. In de afbeelding hieronder is te zien hoe deze lagen zijn opgebouwd in de URCaps API.



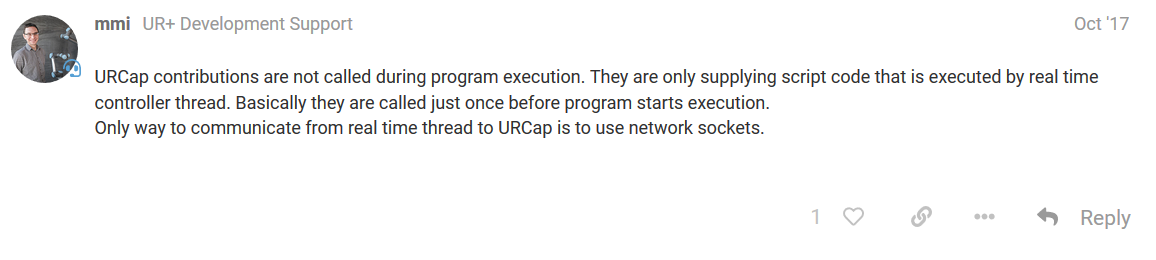
Figuur 2 URCaps API overview, <https://plus.universal-robots.com/getting-started/urcap-api-overview/Deployment>

De toegang tot deze API’s wordt gedaan via een “Provider”-object van de node. Bijvoorbeeld een Installation Node krijg een “InstallationAPIProvider”-object, die toegang geeft aan alle relevante functies voor die node. En deze wordt meegegeven aan de Installation node wanneer deze geïnstantieerd wordt.

De eisen die eerder gesteld zijn aan de ontwikkelomgeving zijn vanzelfsprekend vaak van toepassing op de API zelf. Universal Robots heeft zijn URCaps API duidelijk in kaart gebracht met een simpele overview van de URCaps API (zie Bijlage, 13.3). Deze overview is gebruikt om vast te stellen of de API voldoet.

### UR-Scriptic

Het URCommand maakt gebruik van de taal URScript, dit is makkelijk gezegd een gestripte versie van Python met toegevoegde embedded functionaliteiten. Zoals code wordt omgezet naar machine instructies, wordt de code die een URCap levert omgezet naar URScript. Wanneer er een programma wordt gestart binnen PolyScope , start de zogenaamde “Real Time Thread”(RTT). Deze RTT is een thread dat bestaat uit URScript code dat gegenereerd wordt door PolyScope, waaronder ook de URCap. Wanneer deze RTT gestart wordt kunnen de waardes die meegegeven waren voor de URCap ook niet meer aangepast worden. Kortom, er is geen directe communicatie mogelijk tussen een URCap en de RTT. Zoals bevestigd op het forum van Universal Robots :



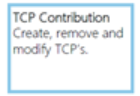
Figuur 3 <https://forum.universal-robots.com/t/urcap-program-node-execution/595>

### Models

(Nog een stuk over models schrijven, wat de beperkingen hiervan zijn)

### ****Communicatie****

Als er naar de eisen gekeken wordt in 7.3.1 en naar de URCaps API, blijkt dat de URCaps API beschikt over de functionaliteiten om deze te volbrengen. De Application API beschikt over de functionaliteiten betreffende IO’s en TCP en de Installation API maakt gebruik van een TCP Contribution.

Deze API’s beantwoorden de requirements, Er is de mogelijkheid om TCP’s te creëren, verwijderen en aan te passen. Ook heeft het de mogelijkheid om te communiceren via Modbus ,en zo het protocol van de Festo motor kan ondersteunen. De specifieke requirements en hoe de API hier aan voldoet wordt behandeld in Bijlage 13.4.

### User Interface

Elke node in PolyScope krijgt een GUI contribution , deze bevat functies om de interface aan te passen met behulp van HTML of Swing. Universal Robots raad Swing aan omdat het de meeste toegepaste toolkit is voor Java user interfaces onder Java developers en URCap developers. Ook worden er geen updates meer geleverd betreft HTML. Alle objecten die ondersteund worden in JDK6 zijn geïmplementeerd bij de URCaps, dit bevat o.a. : Text Input Fields, Number Input Fields, Special Input Fields ( IP of Passwords ) Buttons, Checkboxes, Dropdowns, Lists, Images en nog veel meer. De Swing toolkit is meer dan geschikt om een bruikbare user interface te creëren voor de FestoCap

## Deployment

Een URCap wordt gecompileerd en gebuild door middel van een Maven build script Maven is een “Build Automation Tool” voor Java projecten gebaseerd op een Project Object Model(POM). Het wordt gebruik voor het builden van projecten ,dependencies en documentatie. Het levert een simpel build proces, vergelijkbaar aan ANT maar met veel meer mogelijkheden. Maven maakt gebruik van een POM file waar alle informatie in staat gerelateerd aan het project, zoals configuratie informatie en dependencies, source directory, plug-ins en goals. Wanneer een Maven project gebuild wordt leest deze het pom.xml bestand uit en voert deze commando’s out.

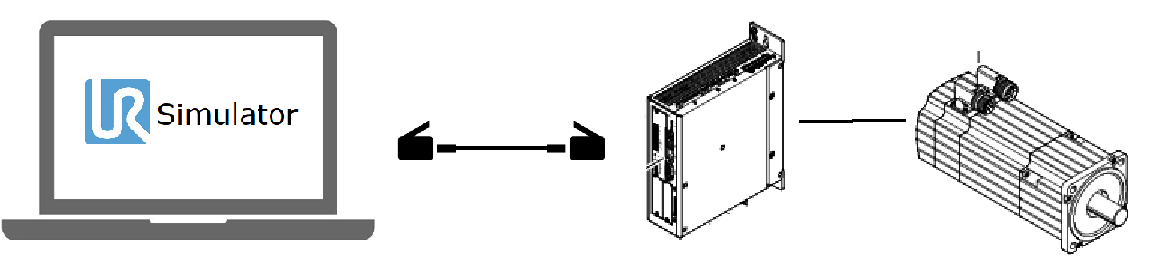
Vervolgens moet het bestand handmatig toegevoegd worden en daarna binnen de simulator toegepast worden. Hierna dient de simulator opnieuw opgestart te worden. Het kost veel tijd om een simpelere URCap te deployen . Dit neemt vooral veel tijd in beslag tijdens de ontwikkeling van een URCap.

## Testopstelling

Om een URCap te kunnen produceren volgens de wensen van de opdrachtgever is een testomgeving opgezet. Deze opstelling moet de toepassing van het product voldoende simuleren tijdens de ontwikkeling van het product.

### Virtuele Testopstelling

Om in de beginfasen van de ontwikkeling van het project wordt er gebruik gemaakt van de virtuele omgeving geleverd door Universal Robots. Deze is verbonden via netwerkkabel met een Festo motorcontroller. Deze testopstelling maakt het mogelijk snel een project te testen zonder een UR Robotarm.



### Fysieke Opstelling

Voor de tests wordt er gebruikt gemaakt van een UR5 robotarm opstelling met twee Festo motorcontrollers en motors. Ook beschikt de opstelling over een Pick-IT Camera voor objectdetectie en grijper voor het pakken van objecten. Het doel van deze opstelling is om een bal te pakken uit een bakje en deze vervolgende bovenaan een baan te plaatsen die eindigt in het bakje. De robotarm wordt over een as verplaatst naar het drop-off punt met behulp van de eerste motor, boven het drop-off punt draait een schijf met een gat, deze schijf wordt met behulp van de tweede motor rondgedraaid. De robotarm moet op het juiste moment de bal loslaten zodat deze door het gat valt en de baan afrolt. Voor de motor in kwestie is beperkte documentatie beschikbaar, maar er is vanuit Gibas voorgesteld de documentatie van andere motoren te gebruiken in combinatie met het FTC programma.



## Conclusie

De ontwikkelomgeving is in zijn huidige staat **niet** geschikt om de FestoCap te kunnen produceren die aan alle opgestelde requirements voldoet. Om te kunnen communiceren met de Festo Drive moet er gebruik worden gemaakt van een specifiek Modbus protocol, PolyScope beschikt niet over dit protocol. Verder is het niet mogelijk op een makkelijke manier te communiceren vanuit de real time thread naar een URCap en met randapparatuur. Hier moet extra aandacht aan besteed worden bij de ontwikkeling van de FestoCap. Ook beperkt de manier waarop variabelen aangepakt worden een paar van de gestelde requirements. Het is vereist om alternatiever voor zowel de communicatie en het gebruikt van variabelen binnen PolyScope te vinden.

# Communication is key

Tijdens de analyse van de URCaps API kwam naar voren dat communicatie binnen en buiten de FestoCap een uitdaging zal zijn. In dit hoofdstuk worden de soorten communicatie behandeld waarmee de FestoCap te maken zal krijgen en wordt er dieper op de communicatie ingegaan met de resultaten gevonden in de analyse. Dit beperkt zich niet alleen tot de verbinding met de Festo motor, maar ook de communicatie tussen de Nodes en de real time thread.

In dit hoofdstuk worden de deelvraag; “Hoe werkt het communicatieprotocol van het UR systeem met het Festo drive systeem?” beantwoord.

## Modbus en Festo

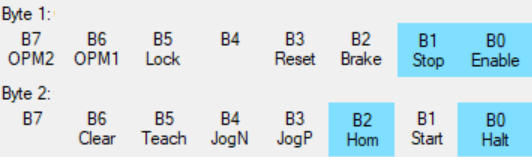
Om de communicatie tussen de FestoCap en de motor te maken is het eerst van belang vast te stellen hoe er berichten naar de motor verstuurd moeten worden. In de requirements is gekeken naar welke commando’s beschikbaar zijn voor de motor, maar er is niet op in gegaan hoe deze verstuurd worden.

Zoals in het Theoretisch kader (7.3 Modbus) is beschreven is het een kwestie van de juiste registers vinden en hier de waardes in te veranderen. In de geleverde documentatie is niet beschreven welke registers gebruikt worden door de Festo drive, hiervoor is het programma ModRSsim2 gebruikt. Een Modbus simulator die zowel de Master als de Slave kan simuleren.

Ten eerste is er een communicatie opgezet met de Festo Drive, er zijn er vervolgens handmatig waardes ingevuld die volgens de documentatie bepaalde commando’s uit moeten voeren. Deze commando’s zijn decimale waardes die twee bytes voor moeten stellen.

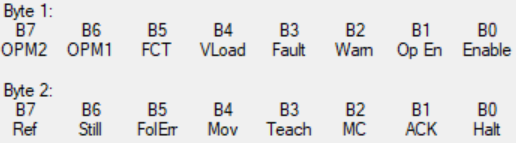
### Home

Om het home commando uit te voeren zijn bits B0,B1 van byte 1 en bits B0 en B2 van byte 2 op waarde 1 gezet terwijl de andere bitjes de waarde 0 hebben. Deze twee bytes worden vervolgens naast elkaar geplaatst en omgezet naar een decimale waarde. In het geval van het Home commando komt er de binaire waarde **[0000 0011 0000 0101]** uit, wat gelijk is aan de decimale waarde **773**.



Figuur 4, Visualisatie van de commando Bytes in de Festo Motor

Wanneer dit bericht in het juiste register[1] gezet is zal de Festo motor reageren met een statusbericht in andere registers. Wanneer het commando uitgevoerd is wordt de Acknowledge bit omgezet.



Figuur 5 Visualisatie van de status bytes van de Festo motor

Zo zijn alle commando’s die de motor kan ontvangen omgezet naar decimale waardes. Vervolgens zijn de mogelijke statussen van de motor vastgesteld. Het rest alleen nog deze berichten in code om te zetten naar functies die aangeroepen kunnen worden door de FestoCap.

## Modbus Software

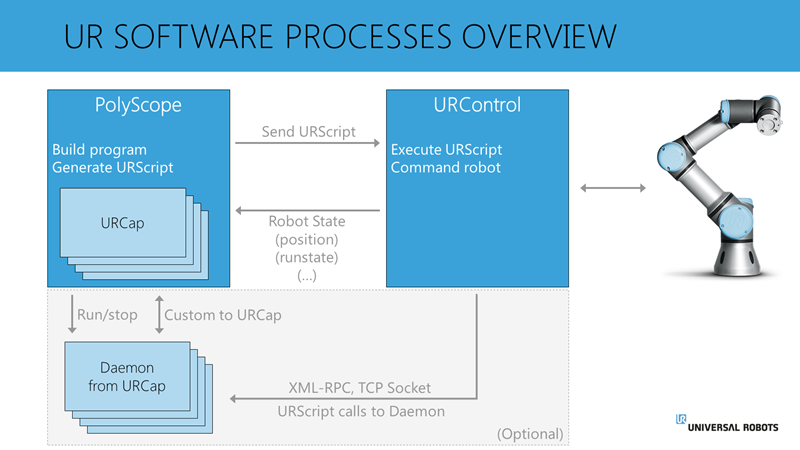
Om de Festo motor aan te kunnen sturen vanuit de FestoCap is alleen een decimaal getal niet voldoende. Er moet een programma geschreven worden die de volgende acties kan uitvoeren :

* Een commando voor de motor omzetten naar bytes
* Bytes omzetten naar decimale waardes
* Decimale waardes in specifieke registers zetten
* Specifieke registers uitlezen
* Decimale waardes omzetten naar bytes
* Bytes omzetten naar een status van de motor

Om naar de juiste registers[[1]](#footnote-1) te schrijven moet er gebruik gemaakt worden van een externe library. Modbus is een bekend protocol en er is een groot aanbod aan software dat dit mogelijk maakt, de voorkeur gaat uit naar duidelijke library zodat er niet veel onbekende acties op de achtergrond uitgevoerd worden.

## Demons

In de analyse is vastgesteld dat er alleen communicatie mogelijk is tussen de FestoCap en RTT in real-time met een socket verbinding. Dit betekend dat bijvoorbeeld het updaten van de status via deze verbinding moet, of wanneer er een nieuwe variabele wordt aangeroepen. Om dit , volgens UR, te versimpelen wordt er gebruik gemaakt van een zogenaamde daemon. Deze daemons zijn headless extensions die het mogelijk maken real-time met een extern apparaat te communiceren, wat in dit geval dus ook de FestoCap zelf is. In de “UR Software Processes Overview” is goed te zien hoe dit verloopt.



Figuur 6 UR Software Processes Overview , https://plus.universal-robots.com/getting-started/ur-software-architecture/

Een URScript gebruikt XML-RPC via een TCP socket om functies aan te roepen van de daemon. Universal Robots levert een voorbeeld[[2]](#footnote-2) van het gebruik van een daemon om een waarde in de RTT aan te passen. Dit voorbeeld bestaat uit een simpele URCap waarin een waarde aangepast kan worden in real-time. Alleen op de achtergrond bestaat dit meerder verwarrende files in meerdere programmeertalen[[3]](#footnote-3), terwijl eigenlijk alleen om simpele socket communicatie moet gaan.

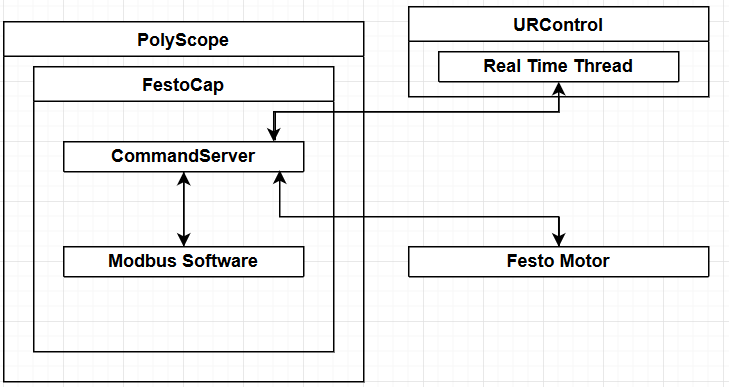
Tijdens de oriëntatie over het gebruik van deze daemons kwam al snel het idee van versimpeling naar voren. Waarom maakt men niet een Java webserver aan en laat deze communiceren met de RTT? Dit idee werd geopperd op het forum van Universal Robots en werd afgeslagen met de reactie :

*“If you need to interact with the program executing at real-time, you should use a daemon process, and not the Java layer(Contributions, etc.) Take a look at our software architecture. “ [[4]](#footnote-4)*

Waar na verzoek om verduidelijking geen reactie op gegeven werd. Om de code van de FestoCap overzichtelijk en duidelijk te houden gaat de voorkeur uit om de software binnen Java te houden indien mogelijk, zolang dit geen onnodige complicaties met zich mee brengt. Na de bespreking met Sjoerd Radstok is besloten Java te gebruiken voor de communicatie, waarbij hij het eens was voor het behoud van overzichtelijkheid en duidelijkheid in de software.

## Communicatie FestoCap naar Modbus

Wanneer er een interface geschreven is voor de motor communicatie moet deze berichten nog op de juiste manier kunnen versturen. Tijdens de analyse is naar voren gekomen dat de enige communicatie die mogelijk is vanuit de real time thread netwerk sockets zijn. Netwerk sockets maken gebruik van een client server protocol waarbij in dit geval de FestoCap de server is en de motor een client. Tevens omdat de real time thread een aparte thread is die onbereikbaar is voor de FestoCap, zal deze ook als client moeten fungeren om te kunnen communiceren met de FestoCap. Dit zal ongeveer resulteren in de volgende netwerkopstelling(Figuur 7)



Figuur 7 Voorbeeld Netwerkopstelling

In dit voorbeeld is de CommandServer een webserver waarmee de RTT en Festo motor als clients mee verbinden. Vanuit de RTT worden commando’s verstuurd, deze worden omgezet naar berichten voor de festo motor m.b.v. de Modbus Software en vervolgens doorgestuurd naar de Festo Motor. Vanzelfsprekend wordt het tegenovergestelde proces gebruikt voor het uitlezen van de status

## Conclusie

Na de analyse van de URCap API, blijkt de communicatie een uitdaging te zijn bij de ontwikkeling van de FestoCap. Vanuit de analyse is verder gewerkt naar een opstelling van hoe de communicatie rondom de FestoCap vorm zal krijgen. Hierbij is het maken van de Modbus interface is een tijdrovend maar duidelijk proces, maar moet grondig aangepakt worden.

De communicatie tussen de real-time thread en de FestoCap/Motor is te realiseren door middel van een Java Webserver waarin de berichten vertaald worden naar commando’s voor de Motor. Dit proces is simpeler en overzichtelijker dan de voorgestelde daemon methode vanuit Universal Robots.

# Making a FestoCap

Een groot deel van de stage bestond uit het programmeren van de FestoCap. Tijdens de ontwikkeling van de FestoCap zijn er meerdere obstakels naar voren gekomen die het moeilijk maakten om sommige requirements te voltooien. In dit hoofdstuk worden deze obstakels besproken en de methodes waarmee toch aan de requirements voldaan is.

## Sequence FestoCap

Wanneer in PolyScope een programma gestart wordt, start de real-time thread. In deze thread staan alle stappen die uitgevoerd moeten worden, bijvoorbeeld het bewegen van de robotarm of een commando vanuit de FestoCap. Wanneer het commando voor de FestoCap aan bod komt wordt de ScriptWriter van de Program node in de FestoCap aangeroepen. Vanuit deze ScriptWriter start een sequence voor het uitvoeren van het commando, dit proces is vastgelegd in een Sequence UML[[5]](#footnote-5). In dit Sequence UML staan de verschillende entiteiten, en interacties tussen deze entiteiten, die worden gebruikt om een commando uit te voeren. Bij het uitvoeren van een commando in de FestoCap komen de volgende stappen aan bod:

1. De ScriptWriter opent een socket verbinding met de CommandServer, deze reageert met een accept bericht waardoor de ScriptWriter weet dat er verbinding is
2. De ScriptWriter genereerd een bericht vanuit de waardes in de models en stuurt deze als een string naar de CommandServer
3. De CommandServer leest dit bericht en zet deze om naar een MotorCommand object
4. De CommandServer voegt de MotorCommand toe aan een queue in de MotorCom
5. In MotorCom wordt het MotorCommand uit de queue gehaald en gelezen welke functie uitgevoerd moet worden en met welke waardes.
6. Vervolgens worden deze geschreven naar de Modbus registers van de Festo Drive
7. De MotorCom leest de Modbus registers uit totdat er sprake is van een Acknowledge en zet de Motor Status op “*Movement Done*”
8. Wanneer de CommandServer de “*Movement Done*” leest uit de motor status stuurt deze een bericht over de socketverbinding terug naar de ScriptWriter
9. De ScriptWriter sluit te socketverbinding met de Commandserver

Bij dit voorbeeld is er sprake van een *“wait for Movement”,* wanneer deze optie niet ingeschakeld is zal de ScriptWriter het bericht sturen en vervolgens afsluiten terwijl op de achtergrond het commando uitgevoerd wordt.

## URCap

Het URCap onderdeel van de FestoCap bestaat uit een installation node en een program node. Deze nodes met ieder respectievelijk eigen verantwoordelijkheden. Er is vastgehouden aan de structuur van UR, waarin de installation node de functionaliteiten van de URCap bevat en de program node alles bevat om de juiste URScript code te genereren.

### Installation Node

De installation node brengt de functionaliteit aan de FestoCap en initialiseert de klassen die dit mogelijk maken. Het is terug te zien in het klassendiagram(Figuur 11) dat deze node de meeste verantwoordelijkheid met zich mee brengt. De installation node start MotorCom en verbindt, met de verbindingsdetails die worden gelezen uit een configuratiefile, met de Festo Drive. Vervolgens wordt ook de CommandServer gestart waar de berichten vanuit de real-time thread ontvangen worden ontvangen, omgezet worden naar MotorCommand en in een queue gezet worden voor de MotorCom klasse.

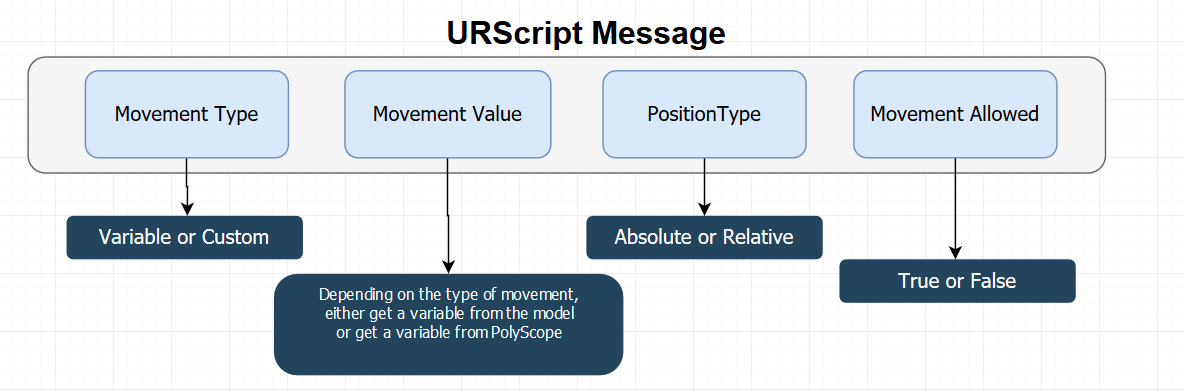
Vanuit PolyScope is de installation node ook te gebruiken om de configuratie aan de passen of om handmatig te verbinden of de verbinding te verbreken. Ook is het mogelijk om de optie “Connect on Start” te veranderen.

### Program Node

De taak van de program node is de juiste URScript code doorgeven aan de real-time thread in URScript. Dit is echter niet zo simpel als een variabele declareren en weer aanroepen. Wanneer de real-time thread start is de enige code die beschikbaar is, de code uit de GenerateScript functie en het DataModel van de program node. Dit betekend dat wanneer er bijvoorbeeld een waarde als *“position”* wordt veranderd in de URCap, dat er een model.set(POSITION,”*position*”) functie aangeroepen wordt. Dit is ook het geval wanneer er de huidige positie opgevraagd wordt vanuit de program node, dat deze vanuit de GenerateScript aangeroepen moet worden , vervolgens in het datamodel gezet en ten slotte een model.get(CURRENT\_POSITION*,”current\_position”*). Om vervolgens de juiste URScript code uit te voeren, moeten er strings verstuurd worden met de functie writer.appendLine(). Hiermee inject het programma een line in URScript die dan uitgevoerd kan worden in de real-time thread.

Figuur 8, voorbeeld van de appendLine functie

writer.appendLine("socket\_open(\"localhost\",59898)");



Figuur 9, URScript Message Format

Hierin is socket\_open() een standaardfunctie van URScript. Het formatten van de strings is een uitdaging geweest omdat er op meerdere niveaus strings doorgestuurd worden. Eerst wordt er de inhoud voor de append Line() als string gestuurd, vervolgens moeten de meeste functies van URScript ook als attribuut een string. Er is veel tijd verloren gegaan om tot een correcte format te komen zodat er uiteinde een bericht verstuurd wordt als in figuur 9.

## Communicatie

Wanneer er een bericht verstuurd is vanuit URScript komt deze aan bij de CommandServer. De CommandServer accepteert alle lokale socketverbindingen en maakt een aparte thread aan per socketverbinding. In deze thread wordt gewacht op een bericht van de client om deze vervolgens te verwerken naar een MotorCommand object. Vervolgens word dit object in een queue gezet voor de MotorCom om af te handelen. Wanneer de queue leeg is en de status van de MotorCom op “Movement Done” staat, stuurt de MotorCommand een bericht terug naar de real-time-thread en sluit deze vervolgens de socket thread.

Tijdens de ontwikkeling van het communicatiesysteem is gekozen voor een simpel communicatieprotocol en zo min mogelijk socketcommunicatie om overzichtelijkheid en simpeler te houden. In vergelijking tot het proces dat UR gebruikt, voldoet de huidige oplossing naar de wensen van de opdrachtgever.

## Lay-out

Zoals eerder vermeld, in hoofdstuk 7.9, heeft de lay-out van de FestoCap geen hoge prioriteit behalve als doel voor het testen van de functionaliteiten van de FestoCap. Om deze reden is gekozen voor een minimalistische aanpak zonder teveel grafische aanpassingen(Figuur 16,Figuur 17,Figuur 18). Een groot deel van de lay-out is wel het juist visualiseren van de waarden en mogelijkheden.

### Variabelen weergeven

Zo bleek het een uitdaging te zijn om de variabelen vanuit PolyScope weer te geven in een lijst in de program node. Deze variabelen bestaan pas voor PolyScope wanneer de real-time thread gestart wordt, om toch een dynamische lijst mogelijk te maken is er een workaround geïmplementeerd. Het is echt wel mogelijk de namen van deze waarden op te halen zonder enige inhoud. Wat betekend dat, wanneer er een variabele geselecteerd wordt, de program node alleen een naam heeft van de variabele. Deze wordt vervolgens in een model gezet en wanneer de real-time thread gestart wordt. Vanuit URScript wordt deze naam uit het model gehaald en vergeleken met de geïnitialiseerde waardes waar vervolgende de daadwerkelijke variabele gebruikt wordt voor het bericht.

### Titles en titles

Binnen de URCap API heeft elke node een titel, deze titel is de naam van de URCap, in dit geval de FestoCap. Wanneer de program node uit de lijst met mogelijke nodes gekozen wordt(Figuur 12), heeft deelt deze de titel ook in de lijst van het “Robot Program”. Het was de wens van de opdrachtgever om in deze titel de actie weer te geven, bijvoorbeeld “Move(300)” of in het geval van een variabele “Move(Var1)”. Om dit te bereiken moest ten eerste weer gebruik gemaakt worden van de Models, wat in 10.2.2 al besproken is, maar wanneer men vervolgens de titel verandert beide instanties van de titels veranderen naar “Move(300)”. De gevonden oplossing hiervoor is het gebruik van HTML, de GUI van PolyScope maakt nog gebruik van HTML, alleen kunnen sommige attributen bijvoorbeeld geen padding of style bevatten en worden deze genegeerd door het systeem.

Figuur 10, Code Snippit FestoCap Title

return "<html><table style=\"padding-left: 130px;\"><tr><td width=\"99%\"><h1> ='"FestoCap"'</h1></td></tr></table></html>"+"Move :" + value;

Door de titel in een header te stoppen en de actie als string er achter te doen wordt binnen PolyScope in de ProgramNode en in de lijst van mogelijke URCaps de HTML titel weergegeven, echter omdat in de Robot Program lijst dit niet ondersteund wordt word dit genegeerd en wordt de string aan het einde van de string weergegeven.

Deze manier is ook toegepast bij de afbeelding in de installation node(Figuur 17). Hierin worden ook de titles gedeeld, het is echter niet mogelijk een afbeelding te gebruiken in de lijst met URCaps, dus deze wordt automatisch genegeerd. Maar in de UI van de installation node zelf is het wel mogelijk deze afbeelding te gebruiken.

## Toolbar

De Toolbar Node binnen de FestoCap wijkt af van de andere nodes omdat deze juist een dynamische opbouw heeft. Het doel van de Toolbar is ook dat deze overal beschikbaar is binnen PolyScope, hierdoor is de Toolbar niet gebonden aan statische Models of de real-time thread. De Toolbar is te vergelijken met de Program node, maar kan dezelfde functies uitvoeren in real-time. Er wordt ook via een socketverbinding contact gelegd met de Commandserver, alleen dan vanuit de standaard Java Library in plaats van URScript(Figuur 14, Toolbar Klassendiagram). Er is weinig op te merken aan de Toolbar, alleen dat deze de structuur heeft die ook gewenst zou zijn in de andere nodes.

# Conclusie

Het maken van een URCap is een onderneming, zeker wanneer er geen ervaring aanwezig is. De tools die UR biedt zijn waardevol maar forceren ook een ,niet altijd even logische, werkwijze. Wanneer dit botst met de denkwijze en redenatie van de programmeur doet het meer kwaad dan goed. Voor een bedrijf dat veel focus heeft op de vrijheid van developers, sluit het zich zowel in de software als in communicatie af voor innovatie.

Bij de ontwikkeling van de FestoCap kwam naar voren dat de omvang van het project al snel buiten de vaardigheden van een enkele programmeur binnen gestelde tijdsnormen. Desalniettemin is er een functioneel product geleverd. Door de nadruk op overzichtelijkheid en duidelijkheid in de structuur en functionaliteit van de FestoCap is er goede fundering gelegd voor wat in de toekomst een waardevol product kan zijn voor Gibas.

# Aanbevelingen

# Bronvermelding

# Bijlage

## Plan van Aanpak

## Festo motor DataSheet ( Temp want geen online versie beschikbaar)

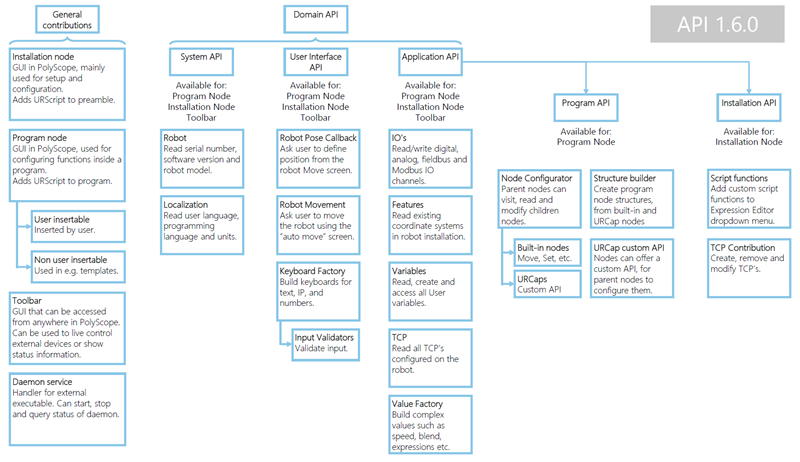
|  |  |
| --- | --- |
| Controle Functie | Beschrijving vanuit Datasheet |
| Enable Drive | Enables the Festo Drive(controller) |
| Stop | The drive stops with maximum braking ramp, the positioning job is reset. |
| Start Positioning Task | A rising edge\*transfers the current nominal data and starts a positioning process |
| Start Homing | A rising edge\* starts homing with the set parameters |
| Jog positive | The drive moves at a specified velocity or rotational speed in the direction of larger actual values, as long as the bit is set. |
| Jog negative | The drive moves at a specified velocity or rotational speed in the direction of smaller actual values, as long as the bit is set. |
| Reset Fault | A malfunction is acknowledged wit ha rising edge and the malfunction value is deleted |
| Velocity | Velocity ramp as % of base value |
| Position | Position in positioning unit |
| Status Functie | **Beschrijving vanuit Datasheet** |
| Drive Enabled | =1: Drive(controller) is enabled  =0: Drive blocked, controller not active. |
| Operation Enabled | =1 Operation enabled, positioning possible.  =0: Stop active. |
| Warning | =1: Warning applied.  =0: No warning present |
| Ready Enable | =1: Ready for Enable.  =0: Not ready for Enable. |
| Halt | =1: Halt is not active; axis can be moved.  =0: Halt is active. |
| Acknowledge Start | =1 : Start executed(homing,jogging,positioning)  =0: Ready for start(homing,jogging,positioning) |
| Motion Complete | =1 : Positioning job completed, where applicable with error  =0 : Positioning job active |
| Absolute / Relative | =1: Nominal value is relative to the last nominal value.  =0: Nominal value is absolute. |
| Control Mode Feedback | Return the Control mode according to the bits set(Position control, Force mode, Velocity control) |
| Feedback depending on Control Mode | Returns feedback depending on Control mode :  Position control : return Velocity as % of base value.  Force mode : return Torque as % of the rated torque  Velocity control : no function, =0 |

## Requirements FestoCap

Voor de ontwikkeling van de FestoCap zijn requirements aan de functionaliteiten vastgesteld. De volgende lijst bevat requirements die tijdens de stageperiode zijn vastgesteld.

|  |  |
| --- | --- |
| FR1.1 | De FestoCap moet een Festo motor kunnen activeren |
| FR1.2 | De FestoCap moet een Festo motor kunnen stoppen tijdens een beweging |
| FR1.3 | De FestoCap moet een Festo motor naar een absolute positie kunnen verplaatsen |
| FR1.4 | De FestoCap moet een Festo motor naar een relatieve positie kunnen verplaatsen |
| FR1.5 | De FestoCap moet een Festo motor continu in beide richtingen kunnen bewegen |
| FR1.6 | De FestoCap moet een error in de motor kunnen herkennen en verwerken |
| FR1.7 | De FestoCap moet de status van de Drive op kunnen vragen |
| FR1.8 | De FestoCap moet de status van de huidige operatie op kunnen vragen |
| FR1.9 | De FestoCap moet een warning of malfunction in de Festo motor kunnen herkennen |
| FR1.10 | De FestoCap moet op kunnen vragen of de Festo motor enabled kan worden. |
| FR1.11 | De FestoCap moet op kunnen vragen of Halt geactiveerd is |
| FR1.12 | De FestoCap moet op kunnen vragen of de Festo motor klaar is om te start |
| FR1.13 | De FestoCap moet op kunnen vragen of de bewegingsopdracht voltooid is |
| FR1.14 | De FestoCap moet op kunnen vragen of de bewegingsopdracht een absolute of relatieve positie is |
| FR1.15 | De FestoCap moet de huidige positie van de Festo motor op kunnen vragen |
| FR1.16 | De FestoCap moet de ingestelde snelheid van de Festo motor op kunnen vragen |
| FR2.1 | De FestoCap moet kunnen verbinden met een Festo motor |
| FR2.2 | De FestoCap moet de verbinding kunnen verbreken met een Festo motor |
| FR2.3 | De FestoCap moet verbindingsfouten kunnen herkennen en weergeven aan de gebruiker |
| FR2.4 | De FestoCap moet berichten kunnen versturen naar een Festo motor |
| FR2.5 | De FestoCap moet berichten kunnen ontvangen van een Festo motor |
| FR3.1 | De gebruiker moet vanuit de user interface een positie kunnen invoeren voor de Festo motor |
| FR3.2 | De gebruiker moet vanuit de user interface een variabele in PolyScope als positie  voor de Festo motor invoeren |
| FR3.3 | De gebruiker moet vanuit de user interface een home commando uit kunnen voeren |
| FR3.4 | De gebruiker moet vanuit de user interface aan kunnen geven of de positie relatief is |
| FR3.5 | De gebruiker moet vanuit de user interface aan kunnen geven of de positie absoluut is |
| FR3.6 | De gebruiker moet vanuit de user interface aan kunnen geven of PolyScope moet  wachten tot het commando uitgevoerd is |
| FR3.7 | De gebruiker moet vanuit de user interface een IP in kunnen voeren om mee te verbinden |
| FR3.8 | De gebruiker moet vanuit de user interface een Port in kunnen voeren om mee te verbinden |
| FR3.9 | De gebruiker moet vanuit de user interface een enable commando uit kunnen voeren |
| FR3.10 | De gebruiker moet vanuit de user interface kunnen verbinden met een Festo motor |
| FR3.11 | De gebruiker moet vanuit de user interface de verbinding met een Festo motor kunnen verbreken |
| FR3.12 | De gebruiker moet vanuit de user interface kunnen selecteren of er verbonden moet  worden bij het opstarten van PolyScope |
| FR3.13 | De gebruiker moet vanuit de user interface de huidige positie van een Festo motor kunnen zien |
| FR3.14 | De gebruiker moet vanuit de user interface de status van de verbinding kunnen zien |
| FR4.1 | De gebruiker moet vanuit de Toolbar interface de huidige positie van een Festo motor kunnen zien |
| FR4.2 | De gebruiker moet vanuit de Toolbar interface de status van de verbinding kunnen zien |
| FR4.3 | De gebruiker moet vanuit de Toolbar interface een stap commando uit kunnen voeren |
| FR4.4 | De gebruiker moet vanuit de Toolbar interface de grootte van de stap in kunnen voeren |
| FR4.5 | De gebruiker moet vanuit de Toolbar interface de Festo motor positief laten Joggen door een knop in te houden |
| FR4.6 | De gebruiker moet vanuit de Toolbar interface de Festo motor negatief laten Joggen door een knop in te houden |
| FR4.7 | De gebruiker moet vanuit de Toolbar interface een home commando uit kunnen voeren |
| FR4.8 | De Toolbar moet duidelijk te identificeren zijn als de Toolbar voor een Festo motor |
| FR5.1 | De engineer moet, zonder de FestoCap aan te passen ,een nieuwe Festo motor klasse toe kunnen voegen |
| FR5.2 | De gebruiker moet een Festo motor kunnen selecteren in de user interface |
| FR5.3 | De FestoCap moet een interface klasse hebben die het mogelijk maakt nieuwe Festo motoren toe te voegen |
| FR5.4 | De FestoCap moet onderscheid kunnen maken tussen de verschillende Festo motoren |

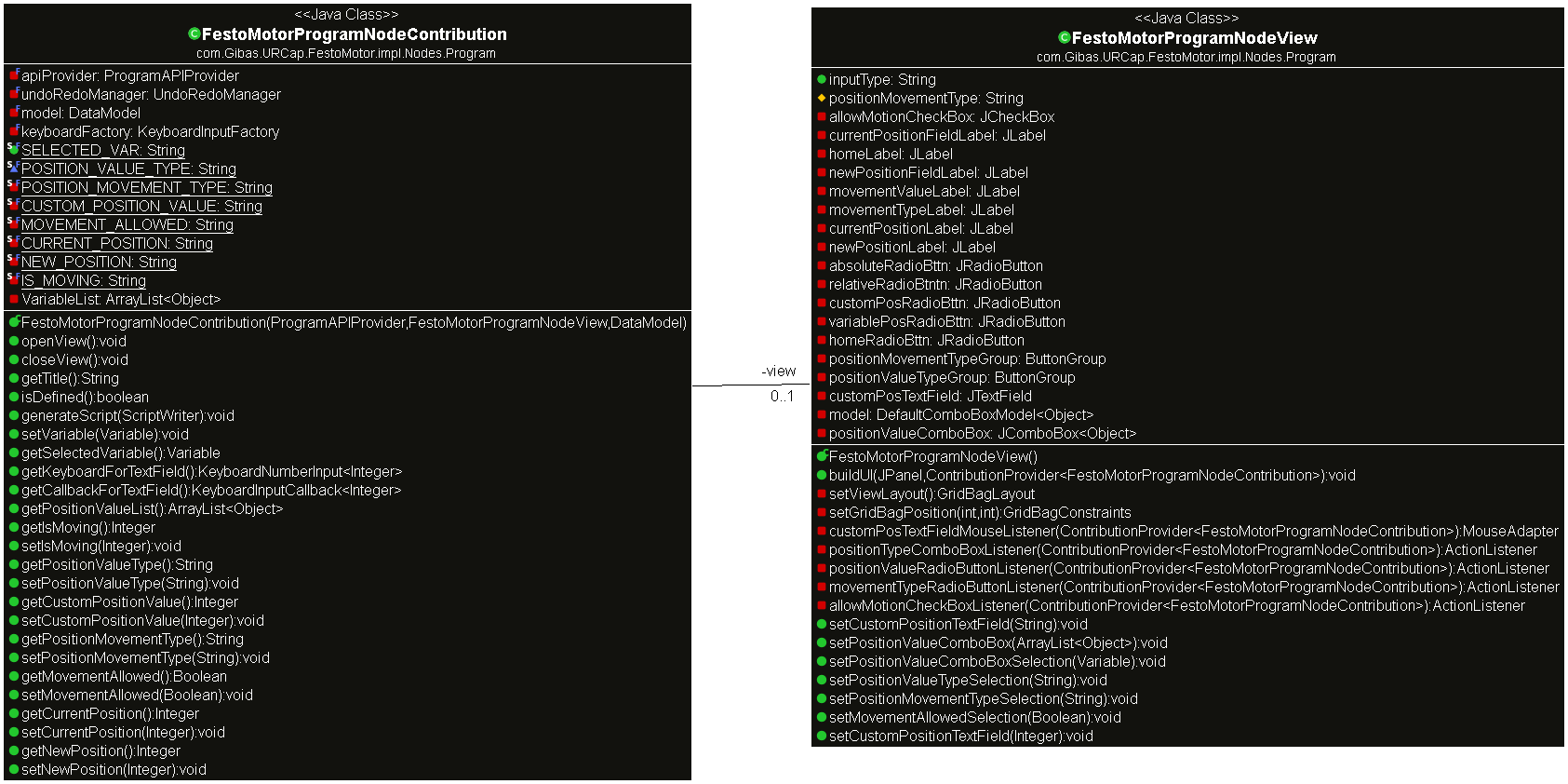
## API Functionaliteit

(Robots, URCap API Overview, sd)

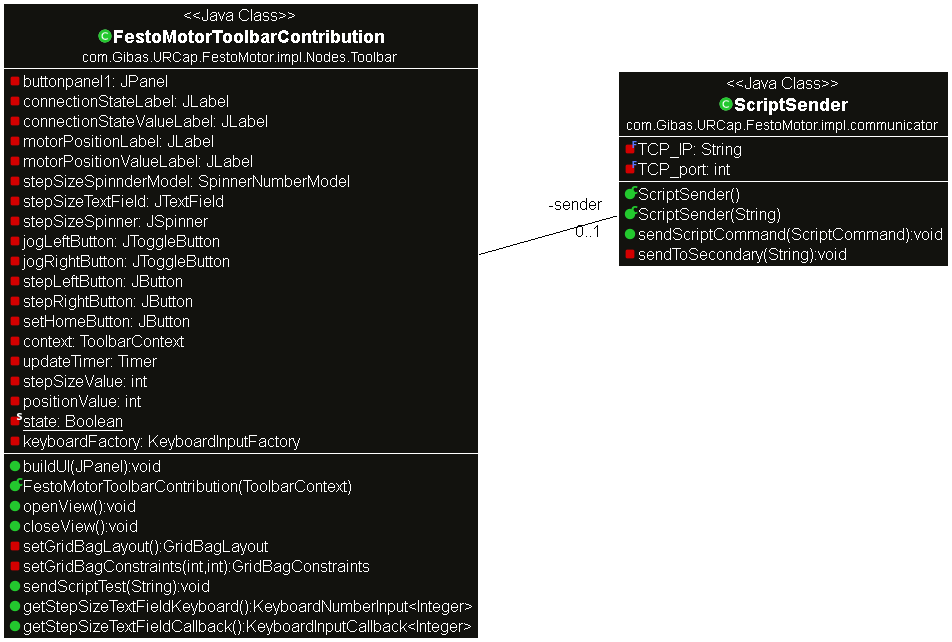
Figuur 11, URCap Functionaliteit

## Klassendiagrammen

Figuur 12, Installation Node klassendiagram

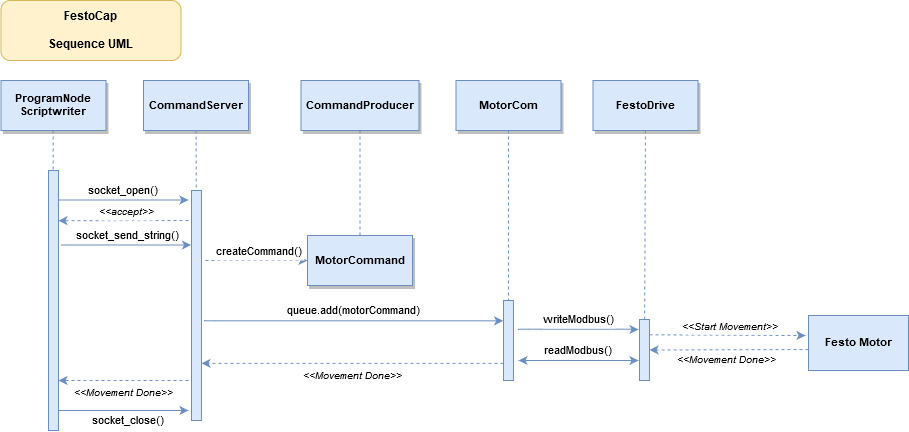


Figuur 13, Program Node Klassendiagram



Figuur 14, Toolbar Klassendiagram

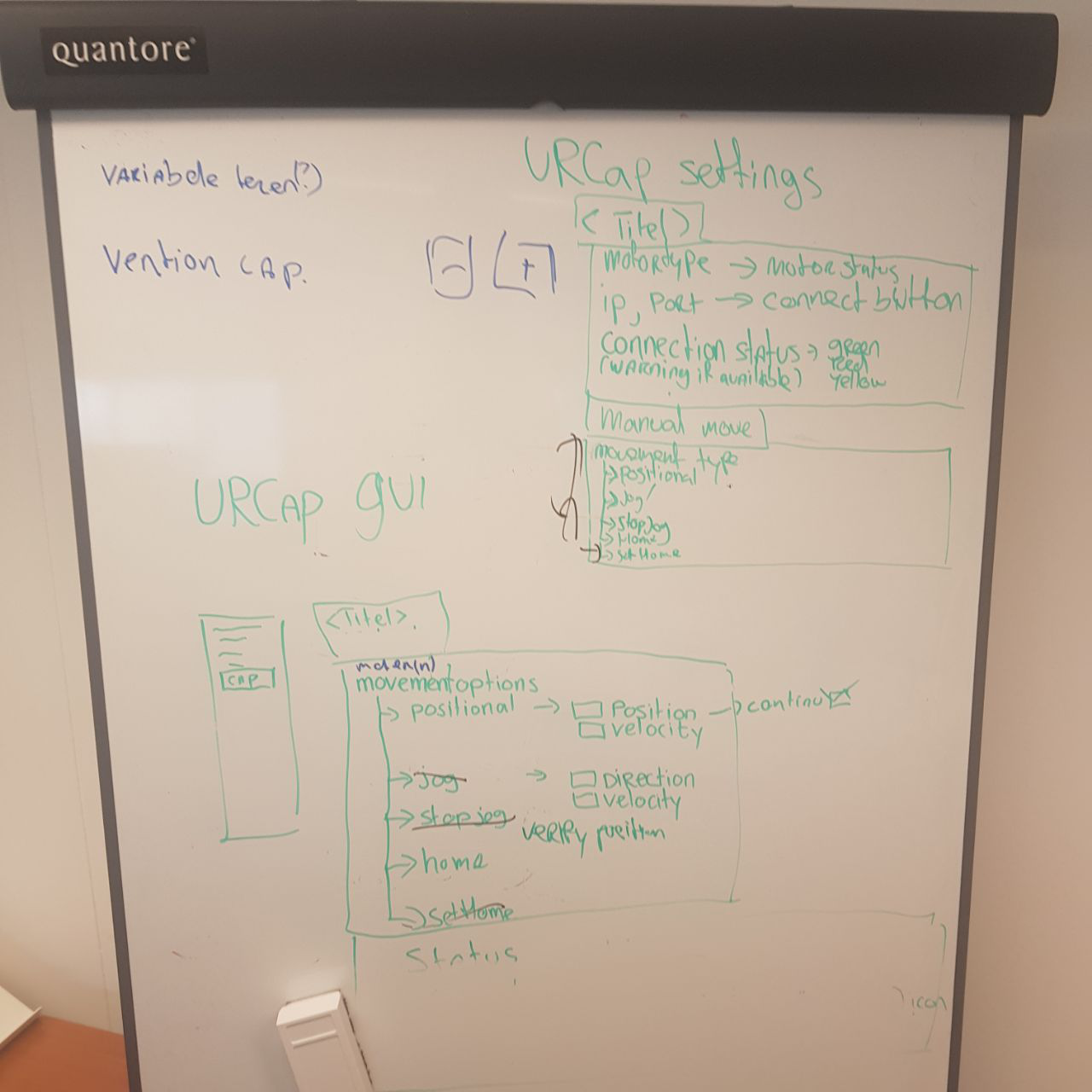
## FestoCap ProgramNode Sequence UML



Figuur 15, Sequence Diagram FestoCap

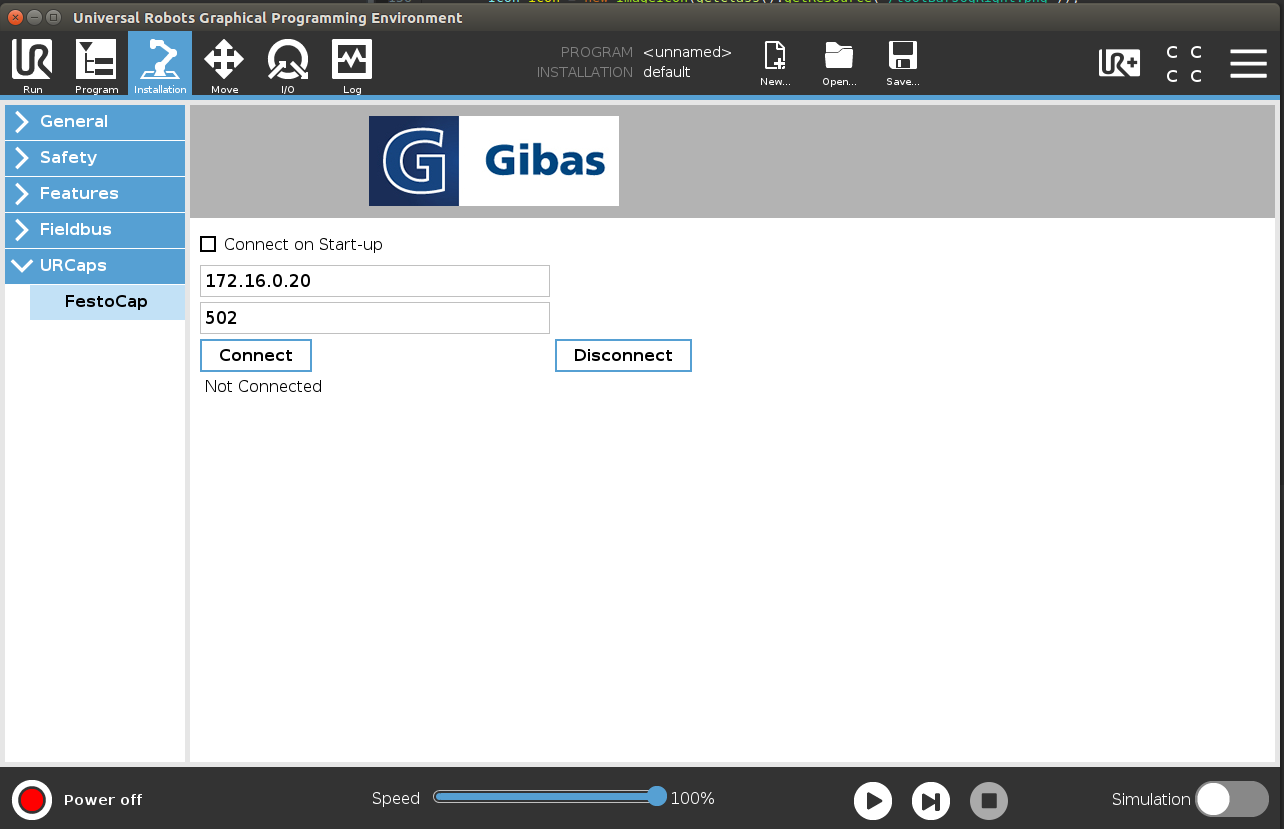
## Lay-Out

### Eerste indruk GUI



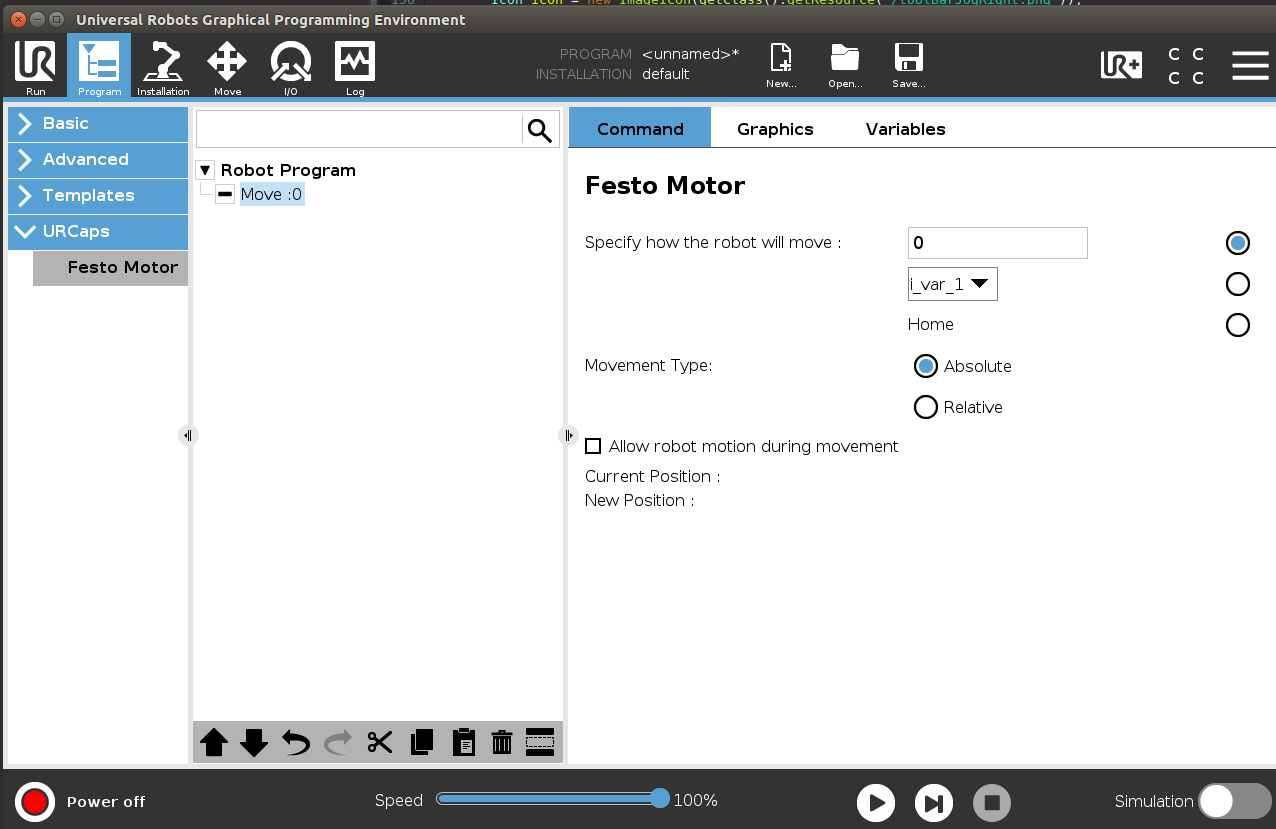
Figuur 16, Eerste indruk FestoCap GUI

### Installation Node



Figuur 17, Installation Node Lay-Out

### Program Node



Figuur 18, Program Node Lay-Out

### Toolbar

## 

Figuur 19, Toolbar Node Lay-Out

1. *De juiste registers zijn gevonden door een methode van trial en error waarbij uiteindelijk vastgesteld is dat de commando registers zich boven de statusregisters bevinden .* [↑](#footnote-ref-1)
2. https://github.com/UniversalRobots/MyDaemonSwing [↑](#footnote-ref-2)
3. https://github.com/UniversalRobots/MyDaemonSwing/find/master [↑](#footnote-ref-3)
4. https://forum.universal-robots.com/t/urcap-program-node-execution/596/9 [↑](#footnote-ref-4)
5. Bijlage FestoCap Sequence UML [↑](#footnote-ref-5)