Dit document beschrijft het solution architecture van de interface en de software van de nieuwste generatie wasmachines van Swirl Industries B.V.

Solution Architecture

*Easy interface, easy life.*

Joost Wagensveld 1664713 Zehna van den Berg 1662506 Jessy Visch 1661709 Koen de Groot 1638079

Team 11

Versie 0.1

11-1-2016

Inhoud

[1. Inleiding 2](#_Toc437959327)

[2. Klassendiagram 3](#_Toc437959328)

[2.1. Het diagram 3](#_Toc437959329)

[2.2. Beschrijving klassen 3](#_Toc437959330)

[2.2.1. //klasse 1 3](#_Toc437959331)

[2.2.2. //klasse 2 3](#_Toc437959332)

[1.1.1. //klasse 3 3](#_Toc437959333)

[2.2.3. //klasse 4 3](#_Toc437959334)

[2.2.4. //klasse rest 3](#_Toc437959335)

[2. Concurrency Diagram 4](#_Toc437959336)

[Protocol communicatie van webserver naar websocket 5](#_Toc437959337)

# Inleiding

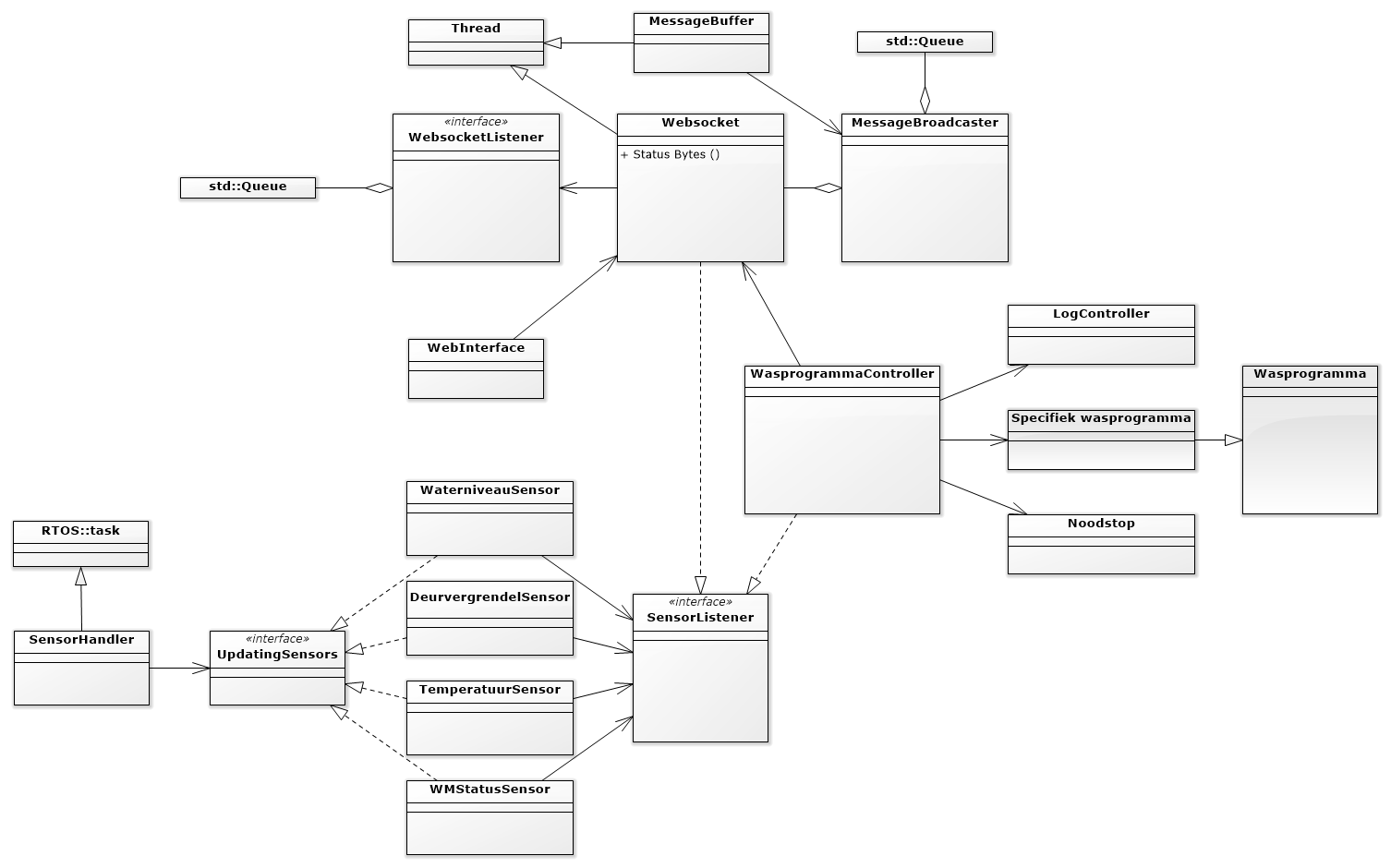
In dit document kunt u alle diagrammen terug vinden die behoren tot het requirements architecture. Een requirements architecture helpt bij het vinden van de antwoorden op de vraag: “Wat moet de software van het systeem doen?” Ook wordt vastgelegd hoe goed het systeem moet functioneren. Daarvoor hebben we het constraints model opgenomen in hoofdstuk 4.

Verder zal de requirements architecture als input dienen voor het maken van de solution architecture.

# Klassendiagram

Dit hoofdstuk beschrijft het klassen diagram. In de eerste paragraaf wordt het diagram getoond. Daarop volgend worden in de tweede paragraaf worden een aantal beschreven.

## Het diagram



## Beschrijving klassen

Dit paragraaf is voornamelijk een opsomming van de verschillende klassen die gebruikt zullen worden. Ook zal er een uitleg zijn van de klassen en bepaalde relaties tussen deze klassen.

### Wasprogrammacontroller

De klasse wasprogrammacontroller is verantwoordelijk voor het uitvoeren van het wasprogramma. Tevens is deze controller verantwoordelijk voor de communicatie met de websocket.

### SensorHandler

Deze klasse is verantwoordelijk voor het periodiek aan roepen van de update functie van alle child klassen van de interface UpdatingSensor. Ook is deze klasse verantwoordelijk voor het bijhouden van alle boundary objecten die hij moet aanroepen.

### UpdatingSensor

UpdatingSensor is de superklasse van alle sensor boundary klassen.

### WaterniveauSensor

Deze klasse is verantwoordelijk voor het pollen van de waterniveau sensor op aangeven van de SensorHandler.

### TemperatuurSensor

Deze klasse is verantwoordelijk voor het pollen van de temperatuur sensor op aangeven van de SensorHandler.

* + 1. DeurvergrendelSensor

Deze klasse is verantwoordelijk voor het pollen van de status van de deurvergrendeling. Tevens kan er via deze klasse de deur worden vergrendelt of ontgrendelt.

* + 1. WMStatusSensor

Deze klasse is verantwoordelijk voor het pollen van de status van de wasmachine op aangeven van de SensorHandler

* + 1. Sensor

Deze klasse is de superklasse van alle sensoren.

* + 1. SensorListener

Deze klasse is de interface die bijhoud of de sensoren nieuwe waarden hebben.

* + 1. StatusWeergaveController

Deze klasse zorgt er voor dat de juiste informatie door gegeven word aan de web interface zodat de informatie die hier word weergegeven up to date blijft.

* + 1. NoodstopController

Deze Klasse is verantwoordelijk voor het afhandelen van de noodstopprocedure. Dit houd in dat het wasprogramma onderbroken word, het water uit de trommel word gepompt en de deur word ontgrendelt.

* + 1. Uart

Deze klasse is verantwoordelijk voor het regelen van de communicatie tussen de sensor klassen en de wasmachine zijn UART interface.

* + 1. Wasprogramma

Houd bij welke fases er gedaan moeten worden betreffende het was programma dat gestart is en zorgt dat deze op de juiste moment gestart word.

* + 1. Fase

Word opgesteld en aan geroepen door het wasprogramma. Deze klassen weet wat er stap voor stap gedaan moet worden voor elke fase die binnen het wasprogramma nodig zijn.

* + 1. LogController

Zorgt dat de juiste data weg geschreven word naar de betreffende logs.

* + 1. ActivityLogItem

Krijgt binnen welke activiteit gedaan is en geeft dit door zo dat het weg geschreven kan worden naar het log.

* + 1. SystemLogItem

Zorgt er voor dat elke stap die het systeem door loopt op de juiste manier weg gezet kan worden in het systeem log.

* + 1. MessageBuffer

Dit is een wacht rij met alle berichten die van af het RTOS bij de massageBroadcaster binnen komen.

* + 1. MessageBroadcaster

Roept tegen de klassen die hier naar luisteren dat er en wat voor bericht binnen is gekomen .

* + 1. Websocket

Schrijft naar de pool waar de RTOS opdrachten neer gezet worden zodat deze door de RTOS uitgelezen en uit gevoerd kunnen worden

* + 1. WebsocketListener

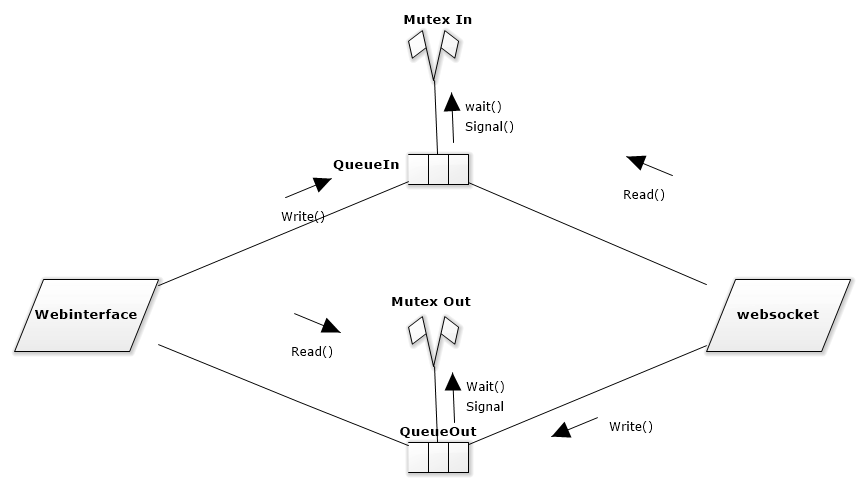
Zorgt er voor dat het verstuurde bericht van de websocket goed neer gezet word in de massageQueue

* + 1. MessageQueue

Een lijst met alle opdrachten (berichten) die naar RTOS door gegeven moeten worden om uitgevoed te kunnen worden.

# Concurrency Diagram

In dit hoofdstuk is het Concurrency diagram te zien. Dit spreekt redelijk voor zich.



# Protocol communicatie van webserver naar websocket

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Request |  | Command | parameter | Antwoord |  |
| WASPROGRAMMA\_START |  | STATUS\_WAS  START\_BONTE\_WAS  START\_KOOK\_WAS  START\_WITTE\_WAS  START\_WOL\_WAS | -  Temp  Temp  Temp  Temp | IDLE  HALTED  RUNNING  STOPPED |  |
| DOOR\_LOCK\_REQ |  | STATUS\_DOOR  LOCK\_DOOR  UNLOCK\_DOOR |  | OPENED  CLOSED  LOCKED |  |
| WATER\_LEVEL\_REQ |  | - |  | Niveau in % |  |
| TEMPERATURE\_REQ |  | GET\_WATER\_LEVEL |  | Temp in celcius |  |
| GET\_RPM\_REQ |  | GET\_RPM |  | RPM |  |
|  |  |  |  |  |  |

**Uitzonderingen**

* Onbekende requests worden beantwoord met “UKNOWN REQUEST”
* Stroomuitval word beantwoord met “POWER ERROR”
* Ineffectieve requests worden beantwoord met “BAD REQUEST”