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-van Veen 1664713 Zehna van den Berg 1662506 Jessy Visch 1661709 Koen de Groot 1638079

Team 11

Versie 0.1

22-1-2016

Inhoud

[1. Inleiding 2](#_Toc440447995)

[2. Klassendiagram 0](#_Toc440447996)

[2.1. Het diagram 0](#_Toc440447997)

[2.2. Beschrijving klassen 0](#_Toc440447998)

[2.2.1. Wasprogrammacontroller 0](#_Toc440447999)

[2.2.2. SensorHandler 0](#_Toc440448000)

[2.2.3. UpdatingSensor 0](#_Toc440448001)

[2.2.4. WaterniveauSensor 0](#_Toc440448002)

[2.2.5. TemperatuurSensor 0](#_Toc440448003)

[2.2.6. DeurvergrendelSensor 0](#_Toc440448004)

[2.2.7. WMStatusSensor 0](#_Toc440448005)

[2.2.8. Sensor 0](#_Toc440448006)

[2.2.9. SensorListener 0](#_Toc440448007)

[2.2.10. StatusWeergaveController 0](#_Toc440448008)

[2.2.11. NoodstopController 0](#_Toc440448009)

[2.2.12. UART 1](#_Toc440448010)

[2.2.13. Wasprogramma 1](#_Toc440448011)

[2.2.14. Fase 1](#_Toc440448012)

[2.2.15. LogController 1](#_Toc440448013)

[2.2.16. ActivityLogItem 1](#_Toc440448014)

[2.2.17. SystemLogItem 1](#_Toc440448015)

[2.2.18. MessageBuffer 1](#_Toc440448016)

[2.2.19. MessageBroadcaster 1](#_Toc440448017)

[2.2.20. Websocket 1](#_Toc440448018)

[2.2.21. WebsocketListener 1](#_Toc440448019)

[2.2.22. MessageQueue 1](#_Toc440448020)

[3. Concurrency Diagram 2](#_Toc440448021)

[4. Protocol communicatie van webserver naar websocket 3](#_Toc440448022)

# Inleiding

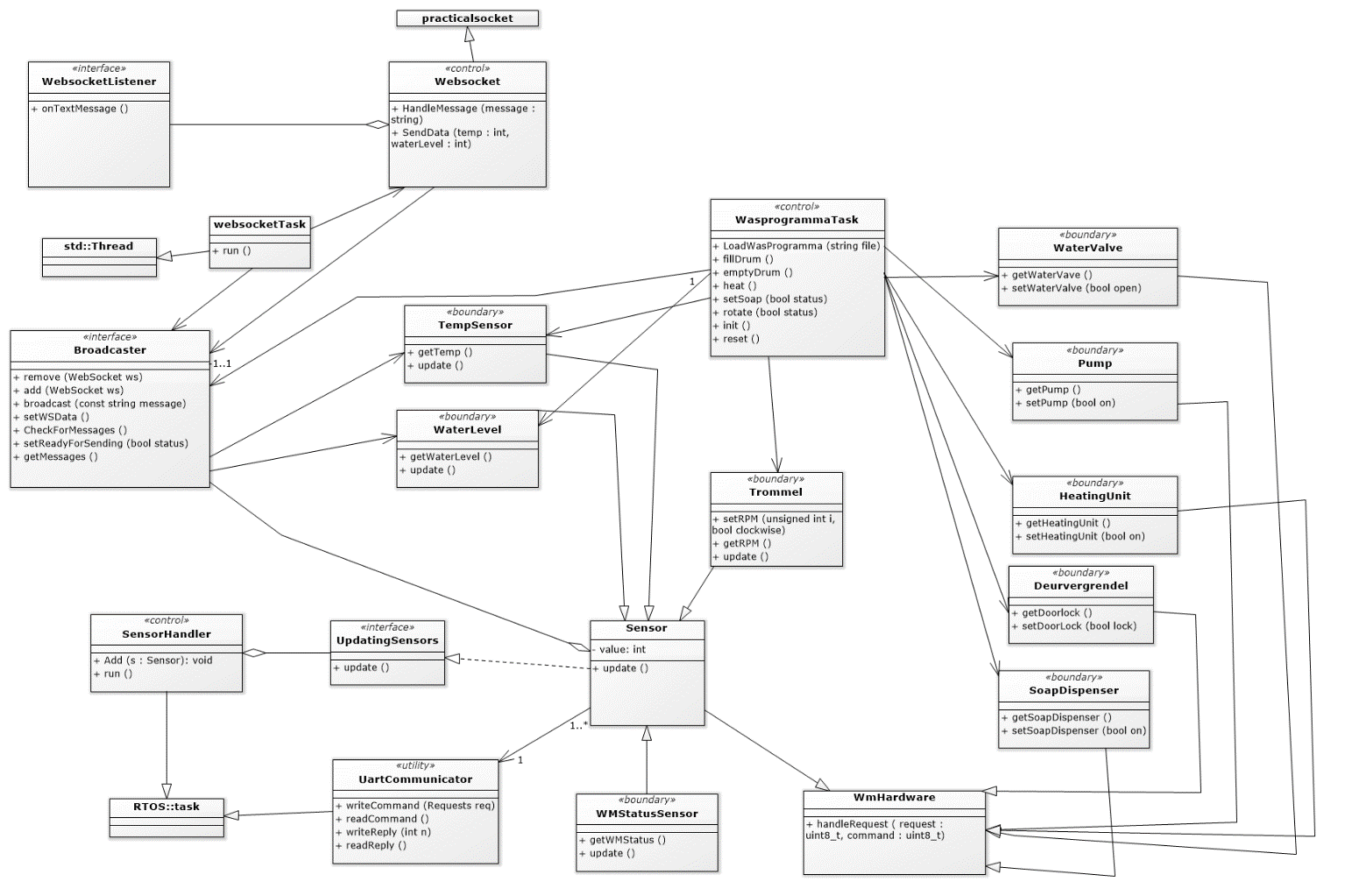
In dit document kunt u alle diagrammen terug vinden die behoren tot het *solution architecture*. Een *solution architecture* helpt bij het vinden van de antwoorden op de vraag: “Hoe kan de software van het systeem het besten in elkaar gezet worden?” Ook word er in meer detail gekeken naar hoe de interne communicatie van het systeem er uit moet zien. Er zal ook kort wat gezegd worden over het protocol dat gebruikt zal worden voor de communicatie tussen de websocket en webserver.

Verder is de *requirements* *architecture* de basis geweest voor het maken van deze *solution* *architecture*.

# Klassendiagram

Dit hoofdstuk beschrijft het klassen diagram. Hier zullen wij in gaan op de verschillende klassen en de functie die deze zullen hebben binnen de software. In paragraaf 2.1 wordt het diagram in zijn geheel getoond. Daarop volgend worden in paragraaf 2.2 de belangrijkste onderdelen van dit diagram verder uit gelicht in wat deze klassen precies doen.

## Het diagram



## Beschrijving klassen

Zoals eerder gezegd zal hier meer uitleg gegeven worden over de klasse getoond in het boven staande diagram. Hier zullen we dan ook waar nodig duidelijk maken hoe verschillende klassen met elkaar samen werken.

### WasprogrammaTask

De klasse wasprogrammatask 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.

### waterLevelSensor

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

### TempSensor

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

### 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.

### WMStatusSensor

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

### Trommel

Deze klasse is verantwoordelijk voor het sturen van de commando’s voor het aanzetten van de motor.

### WaterValve

Deze klasse is verantwoordelijk voor het sturen van de commando’s voor het openzetten van de waterklep

### Pump

Deze klasse is verantwoordelijk voor het sturen van de commando’s voor het leegpompen van de wasmachine.

### HeatingUnit

Deze klasse is verantwoordelijk voor het sturen van de commando’s voor het opwaren van het water van de wasmachine

### Deurvergrendel

Deze klasse is verantwoordelijk voor het sturen van de commando’s voor het op slot zetten van de deur.

### SoapDispenser

Deze klasse is verantwoordelijk voor het sturen van de commando’s voor het opzetten van de soapdispenser.

### Sensor

Deze klasse is de superklasse van alle sensoren.

### SensorListener

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

### UARTcommunicator

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

### WebsocketTask

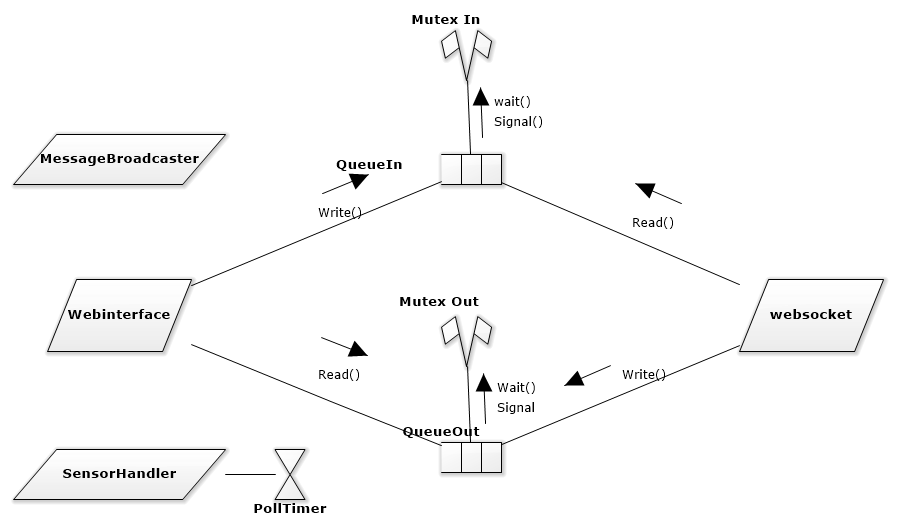
Schrijft naar de pool waar de RTOS opdrachten neer gezet worden zodat deze door de RTOS uitgelezen en uit gevoerd kunnen worden

### WebsocketListener

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

# Concurrency Diagram

In dit hoofdstuk zullen we het *Concurrency diagram* bekijken. Het *Concurrency diagram* wordt voornamelijk gemaakt om datacommunicatie weer te geven. Belangrijk is hoe het systeem omgaat met taken die niet synchroon lopen maar toch elkaar informatie en opdrachten moeten geven. We kijken dus naar wie zijn informatie waar neerzet en wie die informatie dan vervolgens opvraagt.



# Protocol communicatie van webserver naar websocket

|  |  |  |
| --- | --- | --- |
| Request |  | Command |
| Starten van een wasprogramma |  | START\_WAS WASPROGRAMMANAAM TEMPERATUUR DELAY |
| Noodstop |  | STOP\_WAS |

Er zijn bij communicatie tussen webinterface en websocket maar 2 commando’s mogelijk. Deze commando’s zijn start en stop was. Om een was te starten moet er een string worden opgebouwd die bestaat uit het commando START\_WAS gevolgd door de naam van het wasprogramma, de temperatuur, en de wachttijd wanneer deze moet worden uitgevoerd. Deze tijd is in seconden. Alle parameters in de string moeten worden onderscheiden met spaties.

**Uitzonderingen**

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

# State Transition Diagram

De states van de wasmachine worden telkens opnieuw doorlopen aan de hand van de fases die nodig zijn voor het uitvoeren van het wasprogramma. Initiëel begint de wasmachine in de toestand idle waarbij er geen wasprogramma’s in de channel staan. Als er een request tot het starten van een wasprogramma binnenkomt zal dat programma ingeladen worden en de boolean geset worden dat een was bezig is. Op het moment dat de deur gesloten wordt, wordt de deurvergrendeling geactiveerd en de eerste fase gestart. Nadat de betreffende handeling is uitgevoerd wordt er gewacht tot deze handeling zijn doel heeft bereikt en de actie weer uitgeschakeld. Als de laatste fase is geweest wordt er aangegeven dat het wasprogramma klaar en teruggekeerd naar idle. Als er echter nog meer fases zijn wordt er weer naar de state Fase starten gegaan.