Solution architecture

Groep: 13

Leden:

Bouke Stam 1664653

Thijs Hendrickx 1660936

Mathijs van Bremen 1665553

Yorick Schellevis 1657667

Waila Woe 1615856

Klas: V2B

Datum: 15-12-2015

Versie: 0.1 (feedback)

Klassendiagram

**Hardware controllers**

De HardwareController class houdt bij wat zijn status op het moment is en wat de status moet zijn. Elk HardwareComponent kent de UARTInterface en heeft methodes voor alle acties die het kan uitvoeren. Via de UARTInterface kan elk onderdeel berichten sturen en ontvangen. Elke controller is een RTOS::Task.

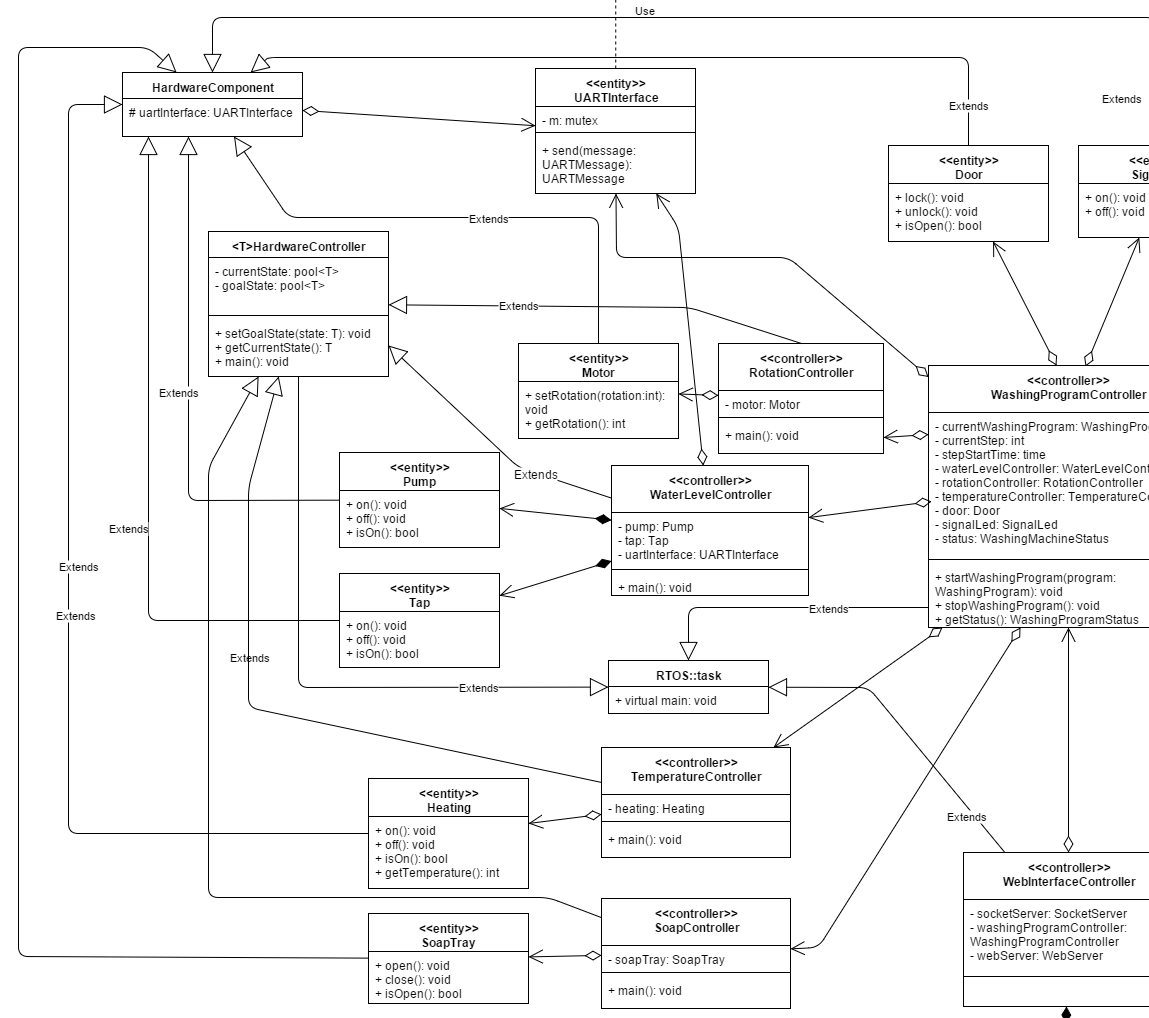
De RotationController beheert de Motor en moet de rotatiesnelheid daarvan geleidelijk veranderen richting de doel snelheid. De Motor heeft methodes om de rotatiesnelheid in te stellen en op te vragen.

De WaterLevelController moet het waterniveau naar het doel niveau zien te krijgen. Hiervoor kan het de Pump of de Kraan aanzetten. Wordt de Pump aangezet dan daalt het waterniveau, wordt de Tap aangezet dan verhoogt het. Op deze manier kan de WaterLevelController het waterniveau reguleren.

De TemperatureController moet de temperatuur van het water op de doel temperatuur krijgen. Om dit te doen kan het de Heating aan of uitzetten. Ook kan het de huidige temperatuur opvragen via het UARTInterface.

De SoapController beheert de SoapTray, het moet het bakje openen en na een tijd weer sluiten. De SoapTray kan geopent en gesloten worden.

Op de volgende pagina staat dit onderdeel van het diagram.



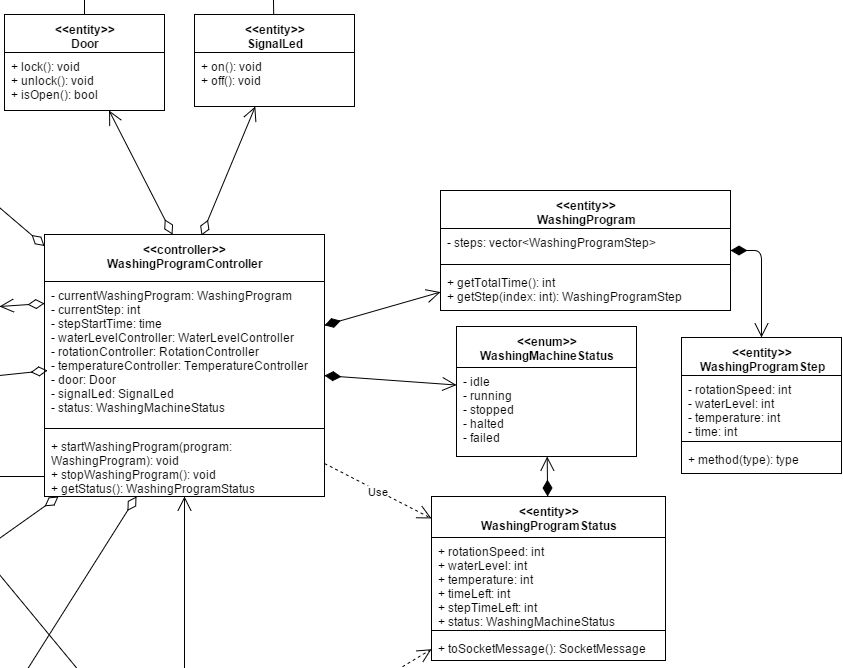
**Wasprogramma’s**

De WashingProgramController heeft directe toegang tot de Door en de SignalLed. Verder kent het alle andere hardware controllers. De taak van de WashingProgramController is om een WashingProgram uit te voeren door te zorgen dat alle hardware wordt ingesteld volgens het programma.

Een WashingProgram bestaat uit meerdere stappen. Elke stap heeft een rotatiesnelheid, waterniveau, temperatuur en tijd. De WashingProgramController heeft methodes om een WashingProgram te starten en te stoppen.

Verder kan de WashingProgramController de status van alle componenten opvragen en teruggeven via de methode getStatus(). Het geeft de status vervolgens terug als een WashingProgramStatus object, dat weer kan worden omgevormd tot SocketMessage om verzonden te worden naar de browser.

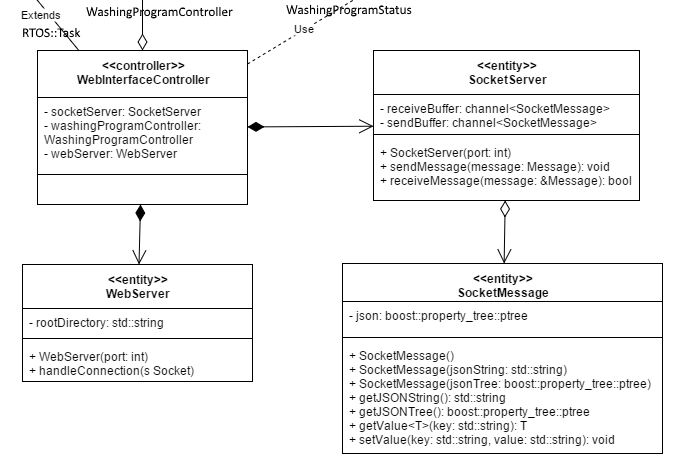
De WashingProgramController kent de UARTInterface waardoor het de status van de wasmachine kan opvragen. Ook kan het de wasmachine starten en stoppen.



**Webinterface**

Het Webinterface bestaat uit de Webserver die bestanden host en de SocketServer die de communicatie met de browser na het laden van de webpagina regelt. De SocketServer verstuurt en ontvangt SocketMessages. Een SocketMessage bestaat uit een JSON structuur en kan geparsed worden vanuit een string en geconverteerd worden naar een string.

De SocketServer heeft twee buffers waarin respectievelijk alle ontvangen en verzonden berichten worden opgeslagen. De WebinterfaceController hoeft hierdoor niet te wachten op de SocketServer.



Taakstructurering

**I/O taakstructurering**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | Type taak | Periode | Deadline | Prioriteit |
| Web interface | periodiek | 1000 ms | 100 ms | 3 |
| Heating | asynchroon |  | 20 ms | 0 |
| Motor | asynchroon |  | 20 ms | 1 |
| Soap | asynchroon |  | 20 ms | 2 |
| Signal led | geen |  |  |  |
| Door lock | geen |  |  |  |
| Pomp | geen |  |  |  |
| Kraan | geen |  |  |  |

De pomp en kraan worden samengevoegd in de WaterLevelTask omdat ze beide effect hebben op het waterniveau en dus altijd samen worden aangeroepen bij het reguleren van het waterniveau.

**Interne taakstructurering**

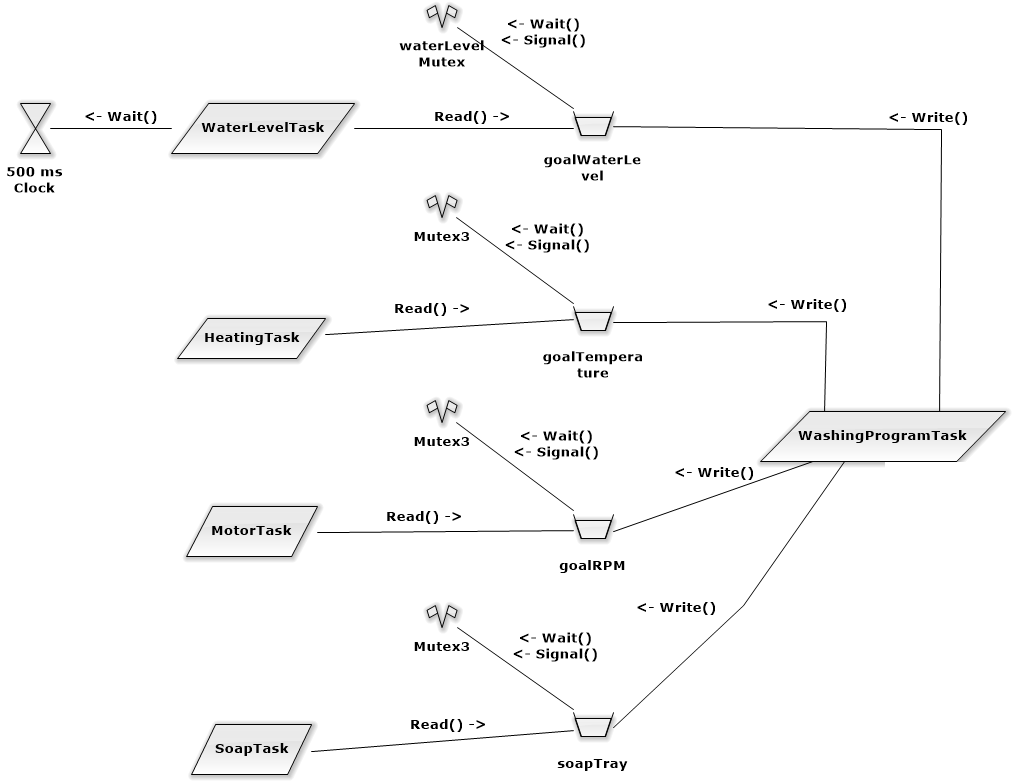
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | Type taak | Periode | Deadline | Prioriteit |
| WashingProgramTask | periodiek | 1000 ms | 10 ms | 0 |
| WaterLevelTask | periodiek | 500 ms | 20 ms | 1 |
| TemperatureTask | asynchroon |  | 20 ms | 2 |
| MotorTask | asynchroon |  | 20 ms | 3 |
| SoapTask | asynchroon |  | 20 ms | 4 |
| WebinterfaceTask | periodiek | 1000 ms | 100 ms | 5 |

**Taak structuur diagram**



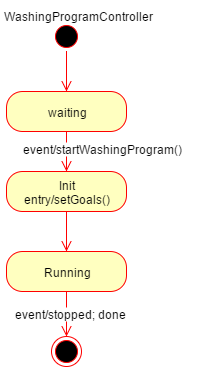
Concurrency model

Dit model beschrijft de communicatie middelen tussen de verschillende taken. Bij de goal waarden hebben we gekozen voor een pool omdat er vanuit een taak wordt gelezen en vanuit de andere naar wordt geschreven. De waarde is niet atomair en kan daardoor niet in één operatie worden geschreven.

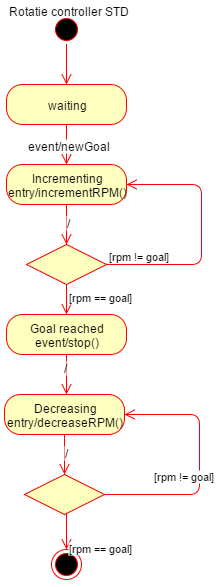


Dynamic model

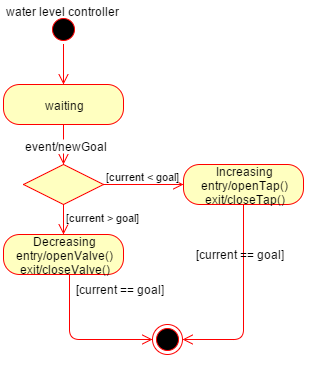
**WashingProgramController**



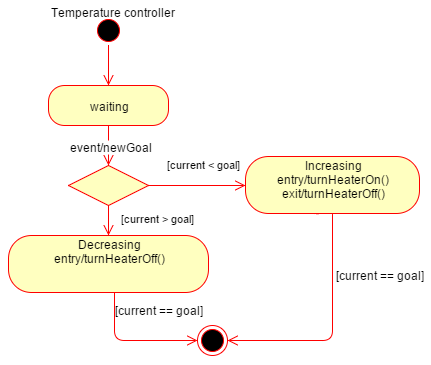
**RotationController**



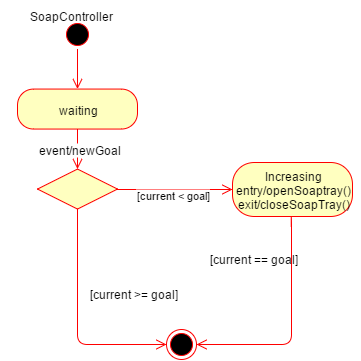
**WaterLevelController**



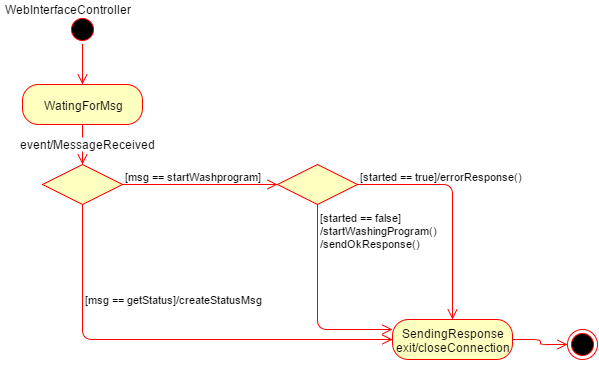
**TemperatureController**



**SoapController**



**WebinterfaceController**



Communicatie protocol

De communicatie tussen de browser en het besturingsprogramma zal met JSON werken. Dit wil zeggen dat data in een JSON string wordt gestopt. Deze string kan bij de browser met behulp van javascript verwerkt worden. Bij het besturingsprogramma zal deze JSON string met een JSON parser/generator verwerkt moeten worden.

Voor het verkrijgen van de beschikbare wasprogramma’s zal de onderstaande JSON string naar de websocket verstuurd worden.

{"**event**":"getWashingPrograms"}

Na het ontvangen van de bovenstaande JSON string zal de websocket de wasprogramma’s terug sturen. Hieronder staat een voorbeeld van het versturen van deze data. Bepaalde opties zullen standaardwaarden zijn die door de gebruiker aangepast kunnen worden.

{

"**event**":"washingProgramList",

"**washingPrograms**":[

{"**desc**":"Washing program 1",

"**steps**":[

{"**step**":"0",

"**settings**":[

{"**name**":"degrees", "**value**":"60"},

{"**name**":"rpm", "**value**":"1600"},

{"**name**":"setting3", "**value**":"val"}

]},

{"**step**":"1",

"**settings**":[

{"**name**":"degrees", "**value**":"80"},

{"**name**":"rpm", "**value**":"1800"},

{"**name**":"setting3", "**value**":"val"}

]}

]}

]}

Op zo’n zelfde manier kan het gekozen wasprogramma naar het besturingsprogramma gestuurd worden. Dit bericht bevat, naast het startcommando, alle opties die bij het wasprogramma horen. Het bericht om het wasprogramma te starten ziet er als volgt uit:

{"**event**":"startWashingProgram",  
"**washingProgram**":{  
 "**steps**":[

{"**step**":"0",

"**settings**":[

{"**name**":"degrees", "**value**":"60"},

{"**name**":"rpm", "**value**":"1600"},

{"**name**":"setting3", "**value**":"val"}

]},

{"**step**":"1",

"**settings**":[

{"**name**":"degrees", "**value**":"80"},

{"**name**":"rpm", "**value**":"1800"},

{"**name**":"setting3", "**value**":"val"}

]}

]},

}

Om het wasprogramma te stoppen wordt het volgende bericht vanaf de browser naar de server gestuurd:  
  
{"**event**":"stopWashingProgram"}

Als de wasmachine al bezig is met het wasprogramma moet de website bepaalde data laten zien aan de gebruiker. De websocket stuurt deze data wanneer er nieuwe data beschikbaar is. Dit bericht ziet er als volgt uit:

{"**event**":"statusUpdate",

"**washingProgram**":{  
 "**currentStep**":"0",

"**data**":[

{"**name**":"status", "**value**":"1"}, // 0 = off, 1 = on, -1 = error

{"**name**":"currentDegrees", "**value**":"60"},

{"**name**":"currentRpm", "**value**":"1500"},

{"**name**":"currentWaterLevel", "**value**":"60"},

{"**name**":"timeLeft", "**value**":"1200"}

]

}

Mocht de browser de pagina openen als het wasprogramma al gestart is, dan kan met het onderstaande commando de data alsnog opgevraagd worden:

{"**event**":"getStatus"}