Wasjes  
Themaopdracht 6

short line

Hendrik Conelisse 1630375

Marianne Delmaar 1648452

Matthijs Vos 1651350

V2.2 - 30/01/17

Marten Wensink en Jan Zuurbier

Inhoudsopgave

[1. Inleiding 4](#_Toc473619697)

[1.1 Achtergrondinformatie 4](#_Toc473619698)

[2. Onderzoek 5](#_Toc473619699)

[2.1 Resultaten 5](#_Toc473619700)

[2.1.1 Verschillende Wasprogramma's 5](#_Toc473619701)

[2.1.2 Meest gebruikte wasprogramma's 5](#_Toc473619702)

[2.1.3 Temperatuurinstellingen 6](#_Toc473619703)

[2.1.4 Toerental Centrifuge 6](#_Toc473619704)

[2.1.5 Speciale wasprogramma's 6](#_Toc473619705)

[2.1.6 Verloop van het wasprogramma 7](#_Toc473619706)

[2.1.7 Watertoevoer 7](#_Toc473619707)

[2.1.8 Wassen 7](#_Toc473619708)

[2.1.9 Water afpompen 7](#_Toc473619709)

[2.1.10 Spoelen 7](#_Toc473619710)

[2.1.11 Centrifugeren 7](#_Toc473619711)

[2.1.12 Verloop na het uitvallen van de spanning 8](#_Toc473619712)

[2.1.13 Gegevens die waardevol zijn om te loggen 8](#_Toc473619713)

[3. Requirements Architecture 9](#_Toc473619714)

[3.1 Use Case 9](#_Toc473619715)

[3.2 Constraints 11](#_Toc473619716)

[4. Solution Architecture 13](#_Toc473619717)

[4.1 Klassendiagram 13](#_Toc473619718)

[4.1.1 WasprogrammaBeHeerContoller & WebSocket 15](#_Toc473619719)

[4.1.2 WasProgrUitvoerHandler & WasProgrammaUitvoerController 17](#_Toc473619720)

[4.2 Taakstructurering 18](#_Toc473619721)

[4.2.1 De PeriodiekeIOTask 18](#_Toc473619722)

[4.2.2 PassiveIOTask 18](#_Toc473619723)

[4.2.3 WasProgUitVoerTask 18](#_Toc473619724)

[4.2.4 WasProgBeheerTask 19](#_Toc473619725)

[4.2.5 WebSocketTask 19](#_Toc473619726)

[4.2.6 LogTask 19](#_Toc473619727)

[4.2.7 Overzicht alle tasks 19](#_Toc473619728)

[4.3 Concurrency model 20](#_Toc473619729)

[4.3.1 WasProgUitvoerTaks en PassiveIOTask 20](#_Toc473619730)

[4.3.2 WasProgUitvoerTask en PeriodiekeIOTask 21](#_Toc473619731)

[4.3.3 WasProgUitvoerTaks en LogTask 22](#_Toc473619733)

[4.3.4 WasProgUitvoerTask en WasProgBeheerTask 23](#_Toc473619735)

[4.3.5 WasProgBeheerTask en WebSockedTask 25](#_Toc473619737)

[4.4 Dynamic model 26](#_Toc473619738)

[4.4.1 PeriodiekeIOHandler 26](#_Toc473619739)

[4.4.2 PassiveIOHandler 28](#_Toc473619741)

[4.4.3 WebSocket 28](#_Toc473619742)

[4.4.4 WasProgrammaBeheerController 29](#_Toc473619744)

[4.4.5 WasProgUitVoerHandler 30](#_Toc473619746)

[4.4.6 LogTask 31](#_Toc473619747)

[5. Communicatie tussen de webserver en de wasmachine 32](#_Toc473619748)

[6. Tests 33](#_Toc473619749)

[6.1 Wasmachine Applicatie 33](#_Toc473619750)

[6.1.1 Task Deadlines Testen : 33](#_Toc473619751)

[6.1.2 Error Management : 33](#_Toc473619752)

[6.1.3 Noodknop: 33](#_Toc473619753)

[6.1.4 Ram en Rom: 33](#_Toc473619754)

[6.1.5 Webapplicatie 33](#_Toc473619755)

[6.1.6 Weergave Website 33](#_Toc473619756)

[6.1.7 Gebruiksgemak: 33](#_Toc473619757)

[6.1.8 Communicatie Website 33](#_Toc473619758)

[7. Realisatie 34](#_Toc473619759)

[8. Evaluatie 35](#_Toc473619760)

[9. Conclusies en Aanbevelingen 36](#_Toc473619761)

[10. Bibliografie 37](#_Toc473619762)

[11. Bijlagen 38](#_Toc473619763)

[11.1 Klassendiagram 38](#_Toc473619764)

[11.2 Concurency 39](#_Toc473619765)

# Inleiding

Het doel van dit document is om dit project op een zo gestructureerd mogelijke manier af te ronden. Het project waar dit document betrekking op heeft is de Swirl T2015 wasmachine. Voor deze Internet of Things wasmachine moet een proof of concept ontwikkeld worden. De wasmachine is te besturen via een webinterface waar een gebruiker met gemak het wasprogramma kan selecteren en de status van het wassen kan bekijken.

## 1.1 Achtergrondinformatie

Dit is de herkansing van Themaopdracht 6 uit Jaar 2 van de opleiding Technische Informatica. Veel in dit plan van aanpak is gebaseerd op de informatie die verkregen is tijdens het eerste moment van deze opdracht. De leden van het team zaten ieder bij een ander projectgroep. De informatie van elk teamlid wordt naast elkaar gelegd en verbeterd in deze uitvoering geleverd.

De opdracht is voortgekomen uit het bedrijf Swirl. Swirl is een fabrikant van huishoudelijke apparatuur. Swirl wil graag op The Internet of Things markt inspelen. Om dit te realiseren wil Swirl een proof of concept van de Swirl T2015 wasmachine. De besturing Swirl T2015 moet via de webbrowser gaan waardoor er geen mechanische besturing meer nodig is. Zo kunnen de eindgebruikers de wasmachine starten en status zien zonder naar de wasmachine te lopen.

De eindgebruikers van de Swirl is T2015 zijn gebruikers die dit product hebben gekocht en er dagelijks hun was er mee doen.

# Onderzoek

Om het project tot een goed einde te brengen zal er onderzoek moeten worden gedaan naar de algemene en gedetailleerde werking van een wasmachine.

Onderwerpen waar onderzoek naar moet worden gedaan zijn:

- Verschillende soorten wasprogramma’s;

    - Meest gebruikte wasprogramma’s;

    - Temperatuurinstellingen;

    - Toerental centrifuge;

    - Speciale wasprogramma's;

- Verloop van het wasprogramma;

- Verloop na het uitvallen van de spanning;

- Gegevens die waardevol zijn om te loggen.

Er wordt onderzoek gedaan door het team om deze vragen te kunnen beantwoorden. Dit wordt gedaan door middel van bronnen zoals internet, readers en boeken maar ook door het onderzoeken van een al op de markt zijnde wasmachine.

## 2.1 Resultaten

### 2.1.1 Verschillende Wasprogramma's

Er zitten verschillende soorten wasprogrammaś op elke wasmachine.

### 2.1.2 Meest gebruikte wasprogramma's

De meest gebruikte wasprogramma’s volgens de consumentenbond zijn:

* Verkort programma: het programma is sneller klaar, maar verbruikt wel meer energie.
* Eco-programma: wast zuiniger maar duurt langer.
* Gemengde was/mix: voor een mix van katoen en synthetisch wasgoed of gemengde weefsels; vaak geschikt voor max. 4 kilo was en sneller dan een standaard programma.
* Extra water/extra spoelen: voor een beter spoelresultaat, bijv. wanneer je een gevoelige huid hebt, of er regelmatig wasmiddel restjes achterblijven.
* Los spoelen en centrifugeren: handig na een handwas.
* Voorwas: voor wasgoed met veel stof of zand.
* Zeer kort/opfrissen: een programma van 15 tot 30 minuten voor een kleine hoeveelheid niet zo vuile was. Soms wordt hierbij gebruik gemaakt van stoom om luchtjes te verwijderen.
* Antikreuk: wasgoed wordt voorzichtiger gewassen en/of gecentrifugeerd. Soms wordt er stoom gebruikt om kreuk te verminderen.
* Overhemden: dit programma wast en centrifugeert extra voorzichtig.
* Jeans/donkere was: er wordt meer water gebruikt en/of extra gespoeld om te voorkomen dat er zichtbare wasmiddelresten achterblijven.
* Vlekken: het programma blijft langer op hoge temperatuur en/of spoelt extra.[2]

Uit deze wasprogramma’s wordt een keuze gemaakt om in het concept te gaan gebruiken.

### 2.1.3 Temperatuurinstellingen

15-20 graden

30 graden

40 graden

60 graden

90-95 graden[3]

### 2.1.4 Toerental Centrifuge

Het toerental is het aantal rondjes dat de trommel draait in een minuut tijdens het centrifugeren. Hoe hoger het aantal rondjes per minuut is, des te sneller de wasmachine en zo kan er theoretisch dus meer vocht uit het wasgoed worden geslingerd. Doorgaans is het zo dat een hoger toerental een drogere was oplevert. Ons onderzoek laat zien dat een machine met een toerental van 1200 meestal redelijk droge was oplevert. Een wasmachine met een toerental van 1400 of 1600 doet het meestal iets beter. Het verschil tussen 1400 en 1600 vrij klein.[4]

De consumentenbond heeft het over toerentallen van:

- 1200 toeren/pm

- 1400 toeren/pm

- 1600 toeren/pm

### 2.1.5 Speciale wasprogramma's

#### **Wasprogramma voor wol.**

De naam zegt het al. Deze was is alleen bestemd voor wol.

#### **Wasprogramma voor katoen.**

Het katoen wasprogramma is het energiezuinigste wasprogramma. Dit wasprogramma verbruikt namelijk het minste water. Als je een katoen wasprogramma draait is het bedoeld voor dingen als beddengoed en handdoeken.

#### **Wasprogramma voor jeans.**

Spijkerbroeken zijn gemaakt van een stugge en stijve stof: denim. Het is belangrijk dat de stof zo blijft en de broek niet dun wordt. Ook vervaagd de kleur van spijkerbroeken snel waar je dus mee moet uitkijken. Spijkerbroeken moeten daarom gewassen worden op het fijnwasprogramma.

#### **Wasprogramma voor fijne was.**

Voor overhemden geldt hetzelfde als voor spijkerbroeken. Overhemden zijn erg kwetsbaar en het behoort daarom ook tot de fijne was. Panty’s moeten ook tot de fijne was gerekend worden.

#### **Wasprogramma voor sportkleding.**

Sportkleding was je net als overhemden, bij de fijne was. Het is hierbij belangrijk om goed uit te kijken op welke temperatuur je de kleding wast omdat het vaak om dunne stoffen gaat.[5]

### 2.1.6 Verloop van het wasprogramma

Het verloop van een wasprogramma gaat volgens de website monteur op afstand als volgt.

Alle wasmachines werken grofweg op dezelfde manier. Je kan de werking verdelen in:

* watertoevoer
* wassen
* afpompen
* spoelen
* centrifugeren

### 2.1.7 Watertoevoer

Nadat je op de aan knop drukt, gaat de wasmachine aan de slag. De ingebouwde computer (de printplaat) weet precies wat, wanneer moet gebeuren. Het deurslot wordt aangezet, om te voorkomen dat je een met water gevulde machine opent. Het inlaatventiel in de wasmachine wordt geopend, waardoor het water uit de geopende kraan de wasmachine in kan stromen. Maar voordat het water de wasmachine instroomt, passeert het eerst de aquastop. Dit is een waterbeveiliging die vaak op de slang aan de achterkant van de wasmachine is bevestigd. Als de wasmachine lekt, gaat de waterbeveiliging aan en wordt de watertoevoer gestopt. Via een slang gaat het water naar het zeepbakje en via een andere slang komt het in de wastrommel terecht. Het waterniveau in de trommel stijgt. De lucht in de trommel wordt via een slangetje naar de drukschakelaar gedrukt. Bij voldoende druk sluit de drukschakelaar het inlaatventiel. Er wordt geen water meer ingenomen en de volgende stap in het wasproces kan gestart worden.

### 2.1.8 Wassen

De draaiende trommel mengt het wasgoed goed met het zeepwater. Het water wordt verwarmd. De trommel draait, stopt en draait de andere kant op. Hierdoor tuimelt het wasgoed over elkaar en kan het water en wasmiddel goed doordringen. Via de motor loopt een band naar de as van de trommel. De trommel wordt in beweging gezet via een motor, die via een band verbonden is met de as van de wasmachine trommel.

### 2.1.9 Water afpompen

Het water uit de kuip verlaat de wasmachine via de afvoerslang naar de afvoerpijp. De pomp brengt het water in beweging. De pomp wordt beschermd door een pluizenfilter.

### 2.1.10 Spoelen

Om al het wasmiddel uit het wasgoed te krijgen, wordt het wasgoed uitvoerig gespoeld. Er wordt schoon water ingenomen. Het wasgoed wordt vergelijkbaar met de wascyclus heen en weer gerold. De spoelgang gebruikt koud water. Moderne apparaten gebruiken sensoren om te bepalen of al het sop uit het wasgoed is. Indien dit niet het geval is wordt er een extra spoelgang gestart. Als wasverzachter in het zeepbakje zit, wordt deze tijdens de laatste spoelbeurt toegevoegd. Vergelijkbaar met de watertoevoer komt het water via het inlaatventiel en het zeepbakje in de kuip. Het water wordt weer afgepompt.

### 2.1.11 Centrifugeren

Door het wasgoed heel snel rond te draaien, wordt al het water uit het wasgoed gedrukt. Eerst wordt het wasgoed gelijkmatig over de trommel verdeeld, om de trommel in balans te houden. Bij het centrifugeren komen enorme krachten vrij. De wasmachine is verzwaard met beton. Hierdoor blijft de wasmachine op zijn plek staan. Het water uit de kleding wordt opgevangen en weer afgepompt. Als laatste wordt het wasgoed langzaam losgedraaid.[6]

### 2.1.12 Verloop na het uitvallen van de spanning

Wat moet er gebeuren als de stroom uitvalt? De mogelijke opties zijn:

* Wasmachine opstarten en klaar voor gebruik maken;
* Verdergaan met het wasprogramma dat de wasmachine aan het draaien was toen de stroom uitviel.

Het team heeft ervoor gekozen de wasmachine startklaar te maken na stroomuitval.

### 2.1.13 Gegevens die waardevol zijn om te loggen

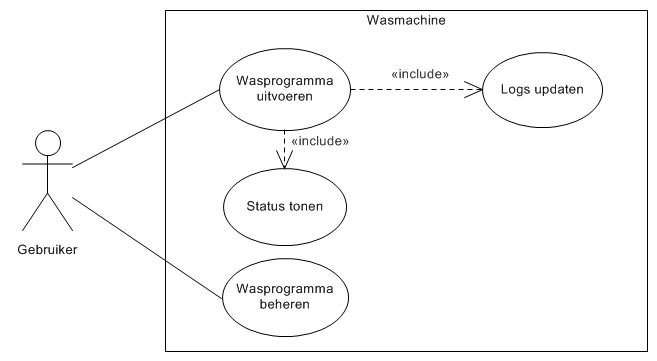
De gegevens die uit het interview naar voren kwamen om te loggen zijn:

* Energieverbruik
* Waterverbruik
* Welke wasprogramma’s worden gebruikt
* Stroomuitval

# Requirements Architecture

## Use Case

Een use case is voor het aanbrengen van structuur in de diverse acties die het systeem moet kunnen uitvoeren. Het beschrijft een reeks samenhangende acties die door het systeem wordt uitgevoerd om aan een bepaald doel van de actor te voldoen. Om een goed beeld te krijgen van het systeem zijn de volgende use cases opgezet :



**Usecase 1**

|  |  |
| --- | --- |
| Use Case Naam | Wasprogramma Uitvoeren |
| Doel | Actor User een wasprogramma uit laten kiezen en uitvoeren |
| Pre-Condities | Alle hardware onderdelen zijn functioneel |
| Post-condities | Alle fasen van het wasprogramma uitgevoerd en de deur is ontgrendeld |
| Uitzonderingen | Als de deur niet vergrendeld kan worden het wasprogramma niet uitvoeren |

**Usecase 2**

|  |  |
| --- | --- |
| Use Case Naam | Wasprogramma Beheren |
| Doel | Actor User een nieuw wasprogramma laten creëren of een bestaand wasprogramma aanpassen |
| Pre-Condities | Verbinding met de websocket |
| Post-condities | Een wasprogramma is aangepast of aangemaakt en vervolgens opgeslagen |
| Uitzonderingen | Als het aanpassen niet voltooid wordt zal het programma de aanpassingen of het nieuwe wasprogramma niet opslaan |

**Usecase 3**

|  |  |
| --- | --- |
| Use Case Naam | Status tonen |
| Doel | Actor User naar een weergave van de status van de wasmachine brengen |
| Pre-Condities | Er is een verbinding met de wasmachine |
| Post-condities | Weergave van de huidige was-status van de wasmachine |
| Uitzonderingen | Als er geen contact gemaakt kan worden met de wasmachine zal de status dit weergeven |

**Usecase 4**

|  |  |
| --- | --- |
| Use Case Naam | Log updaten |
| Doel | De vooruitgang van een wasprogramma opslaan |
| Pre-Condities | Er wordt een wasprogramma gestart |
| Post-condities | Elke voltooide fase van het wasprogramma is opgeslagen in de log |
| Uitzonderingen | Geen uitzonderingen |

## 3.2 Constraints

Het constraints model definieert de niet-functionele eisen die worden gesteld aan het systeem, z.g. constraints. Een constraint is een eis die wordt gesteld aan een waarneembare (vaak meetbare) eigenschap van het systeem. Deze zijn verdeeld in de volgende constraint typen : Accuracy , Learnability, Liability , Performance , Resource use , Usability. In de constraints table hierin staan de verschillende Constraints weergegeven met de Identificatie, beschrijving, eisen die aan de constraint zitten en verificatie ervan.

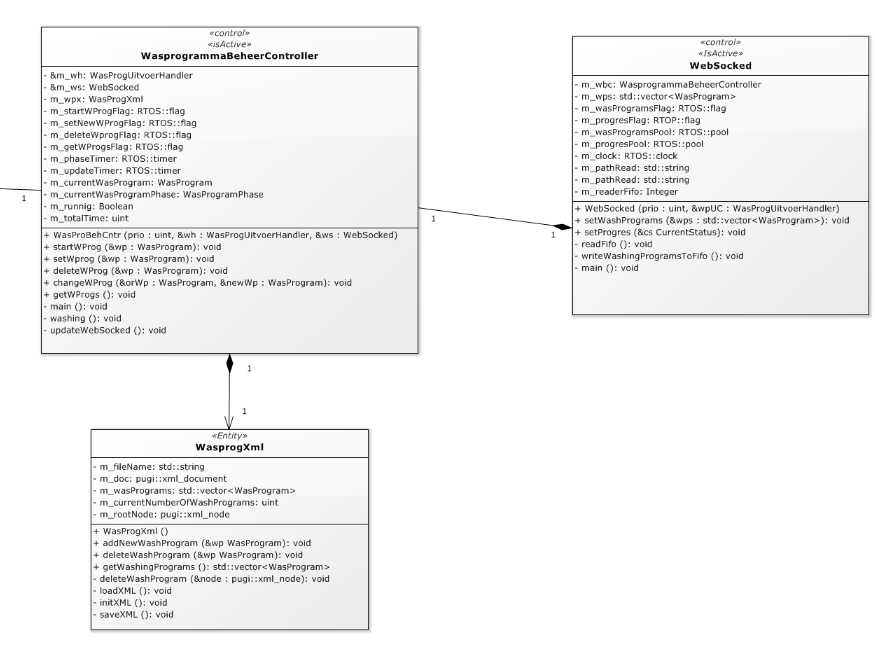
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Constraint Type** | **Identificatie** | **Beschrijving** | **Eis** | **Verificatie** |
| **Accuracy** | Weergave website | Behouden van voortgang van het wasprogramma op de website ten opzichte van de wasmachine. | Afwijking maximaal 10 seconden. | Het timen van de voortgang van de wasmachine en dit vergelijken met de voortgang op de website. |
| **Learnability** | Gebruiksgemak website | Hoe eenvoudig het is voor de gebruiker om de website te leren gebruiken. | Een nieuwe gebruiker moet zonder hulp de wasmachine kunnen aansturen. | Testpanel van Huisvrouwen. |
| **Liability** | Error management | Het programma moet verschillende errors herkennen en hier de juiste feedback op geven. | Problemen met de wasmachine moeten overeenkomen met de gegeven feedback van het programma. | Errors simuleren en kijken of het programma deze errors juist opvangt. |
| **Performance** | Noodknop | De tijd die tussen het drukken op de noodknop en het daadwerkelijk stoppen van de wasmachine zit. | Maximaal 1 seconde/ | De werkelijke tijd meten met behulp van een program analyser. |
|  | Communicatie tussen webapplicatie en wasmachine | De tijdsduur van het verzenden en ontvangen van data tussen de twee instanties. | Maximaal 2 seconden. | De werkelijke tijd meten met behulp van een program analyser. |
| **Resource use** | ROM en RAM gebruik Raspberry Pi. | ROM en RAM gebruik vastleggen. | ROM en RAM gebruik onder maximale capaciteit van de Raspberry Pi. | ROM en RAM gebruik in beeld brengen door een benchmark test uit te voeren. |
| **Usability** | Gebruikersinterface | Hoe makkelijk de gebruiker een wasprogramma kan kiezen en de wasmachine kan starten. | Binnen 3 stappen moet de gebruiker bij zijn of haar doel zijn. | Testpanel van Huisvrouwen. |

# Solution Architecture

In dit hoofdstuk worden de gemaakte modellen en keuzen voor de solution architecture toegelicht.

## Klassendiagram

In het hoofdstuk klassendiagram worden de controllers in het klasse diagram besproken. Het hele klasse diagram is als bijlage 1 toegevoegd.



Figuur 1

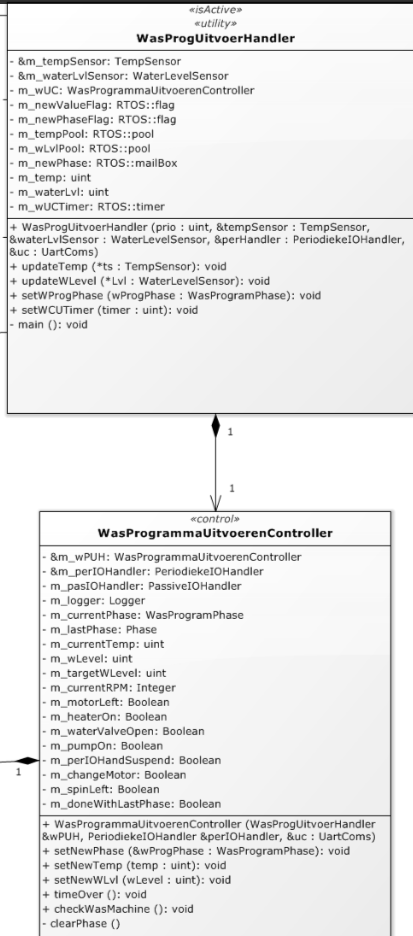
### 4.1.1 WasprogrammaBeHeerContoller & WebSocket

In figuur 1 is de websocket met de WasprogrammaBeheerController te zien. Allebei deze klassen zijn controllers.

De WebSocket is verantwoordelijk voor de communicatie met de webserver. De communicatie tussen de webserver en de wasmachine loopt via named pipes (Kerrisk, 2016). Via named pipes kunnen processen communiceren zonder dat ze elkaar kennen.

De WasprogrammaBeheerController is verantwoordelijk voor het beheren van de wasprogramma’s De wasprogramma’s worden opgeslagen in een XML file. Het afhandelen van de XML doet de WasprogXml. WasprogXml is onderdeel van WasprogrammaBeheerController.

De WebSocket en WasprogrammaBeheerController communiceren met elkaar. De WebSocket stuurt door wat de gebruiker wilt doen met een wasprogramma en de WasprogrammaBeheerController stuurt de status van hoe lang het wasprogramma nog duurt door naar de WebSocket.

Als een wasprogramma gestart wordt is de WasprogrammaBeheerController verantwoordelijk dat de juiste phase gestart wordt. De volgende phases zijn aanwezig: NONE, WASSEN, SPOELEN en CENTRIFUGEREN. Welke phase de wasmachine zich op dat moment bevindt stuurt WasprogrammaBeheerController door naar de WasProgUitvoerHandler. De WasProgUitvoerHandler wordt later in dit hoofdstuk behandelt. De WasprogrammaBeheerController houd bij hoe lang een phase al bezig is. Als de tijd van een phase er op zit stuurt de WasprogrammaBeheerController de volgende phase door naar de WasProgUitvoerHandler tot alle phases zijn geweest.

Figuur 2

### 4.1.2 WasProgrUitvoerHandler & WasProgrammaUitvoerController

In figuur 2 staan de WasProgUitvoerHandler en de WasProgrammaUitvoerenController.

De WasProgrammaUitvoerenContoller wordt aangestuurd door de WasProgUitvoerHandler. Deze onderdelen zijn los van elkaar omdat de WasProgUitvoerHandler eerst ook nog verantwoordelijk was voor het aansturen van de StatusToonController. De StatusToonController is komen te vervallen doordat het alleen een doorschuif luik was naar de WebSocket. In plaats van dat de status van de WasprogrammaBeheerController naar de StatusToonController naar de WebSocket gaat, stuurt de WasProgrammaBeheerController de status gelijk naar de WebSocket.

De WasProgUitvoerHandler krijgt de huidige temperatuur en waterlevel door via het listenerpattern. Welke was phase er uitgevoerd moet worden krijgt de WasProgUitvoerHandler doorgestuurd van de WasprogrammaBeheerController. De WasProgUitvoerHandler stuurt de temperatuur, waterlevel en phase door naar de WasProgrammaUitvoerenController.

De verantwoordelijkheid van WasProgrammaUitvoerenController is het aansturen van de wasmachine onderdelen. Aan de hand van welke phase en de input waardes van de wasmachine stuurt de WasProgrammaUitvoerenController de componenten van de wasmachine aan. De componenten aansturen gaat via de PassiveIOHandler.

## 4.2 Taakstructurering

In dit hoofdstuk wordt besproken welke taken het systeem uitvoert. Voor elke taak wordt besproken wat hun verantwoordelijkheid is, wat de deadline van de taak is en hoe hij hier aan gekomen is. Voor de periodieke taken wordt onderbouwt hoe ze aan hun periode komen.

### 4.2.1 De PeriodiekeIOTask

De PeriodiekeIOTask is I/O periodiek omdat hardwarecomponenten gepollt worden. Deze taak is verantwoordelijk voor het pollen van de Waterlevelsensor en de Temperatuursensor.

De periode dat de taak aangeroepen wordt is 200 ms. Het duurt een paar seconde voordat de temperatuur of waterlevel gestegen is, dus daarom hoeven deze waardes niet heel vaak bijgewerkt worden.

De deadline van PeriodiekeIOTask is 50 ms. Het afhandelen van het pollen is een intensieve taak (Wensink, 2015). Aangezien er 2 waardes gepollt worden zit er een pauze van 20 ms tussen de polls. Zo heeft elke poll 15 ms om afgehandeld te worden.

De prioriteit van de task is 1. Het systeem moet snel kunnen reageren als de juiste temperatuur of waterlevel is bereikt.

### 4.2.2 PassiveIOTask

De PassiveIOTaks is een task die het aan en uitzetten van hardware componenten regelt. De task is Asynchroon, want het reageert op een andere task en kan gegevens wegschrijven naar een andere task.

De deadline is 100 ms. Het wegschrijven en controleren via de Uart kost tijd, maar mag niet te lang duren anders staan componenten te lang aan.

De prioriteit van deze task is 2. Omdat het hardware moet aansturen is het vrij belangrijk als er een component uit wordt gezet dat dit niet te lang duurt.

### 4.2.3 WasProgUitVoerTask

De WasProgUitVoerTask is verantwoordelijk voor het aansturen van de hardware componenten aan de hand van input waardes. Als de WasProgBeheerTask een phase doorgeeft aan de WasProgUitvoerTask, zal deze de PeriodiekeIOTask activeren en berichten van de PeriodiekeIOTask ontvangen. Aan de hand van deze berichten van de PeriodiekeIOTask en de huidige phase zal de WasProgUitvoerTask aan de PassiveIOTask doorgeven wat er met de hardware moet gebeuren.

De deadline van deze task is 50 ms. De task moet kleine controles voeren en hardware aansturen. Omdat het nog een task duurt voor het component echt reageert heeft deze task een kleine deadline.

De prioriteit van deze task is 0. Omdat deze task met alle hardware componenten, input en output, te maken heeft is het belangrijk dat deze task hier snel op reageert.

### 4.2.4 WasProgBeheerTask

De WasProgBeheerTask is verantwoordelijk voor het beheren van de wasprogramma’s. WasProgBeheerTask werkt met een XML file waar alle wasprogramma’s instaan. Via de WebSocketTask kunnen wasprogramma’s worden toegevoegd, verwijdert en gestart. Deze zijn afkomstig van de website.

Als een wasprogramma gestart moet worden stuurt de WasProgBeheerTask een phase naar de WasProgUitvoertask. De WasProgUitvoerTask zal deze phase net zolang blijven uitvoeren totdat de WasProgBeheerTask een nieuwe phase stuurt, dus de WasProgBeheerTask is verantwoordelijk voor de duur van de phases.

De deadline is 250 ms. Het is niet heel belangrijk als het iets langer duurt voordat er iets gebeurt met het wasprogramma wat opgegeven is.

De prioriteit van deze task is 4. Omdat deze task niet met tijdskritische onderdelen te maken heeft krijgt deze task een lage prioriteit.

### 4.2.5 WebSocketTask

De webSocketTask is de task die communiceert. Deze task vertaalt berichten die van de webserver komen naar bruikbare objecten voor de wasmachine en vertaalt objecten naar een bericht van de webserver.

Deze task communiceert alleen met de WasProgBeheerTask. Zo vertaalt deze task start, delete en nieuwe wasprogramma berichten vanaf de website naar wasprogramma objecten waar de WasProgBeheerTask mee overweg kan.

Verder ontvangt het berichten van de WasProgBeheerTask van de status van het huidige wasprogramma wat uitgevoerd wordt. Deze status zal doorgestuurd worden naar de webserver.

De deadline van deze task is 200 ms. Deze task heeft niet te maken met tijdskritische onderdelen, dus is de deadline wat hoger.

De prioriteit van deze task is 3. Omdat deze task niet met tijdskritische onderdelen te maken heeft krijgt deze task een lage prioriteit.

### 4.2.6 LogTask

De LogTask schrijft de string die in zijn channel worden gezet weg naar een log bestand.

Deze task heeft een deadline van 500 ms. Deze task mag andere tasks niet ophouden, dus als het nodig is mag deze task lang over zijn taak doen.

De prioriteit van deze task is 5. Dit is de laagste prioriteit die het systeem heeft.

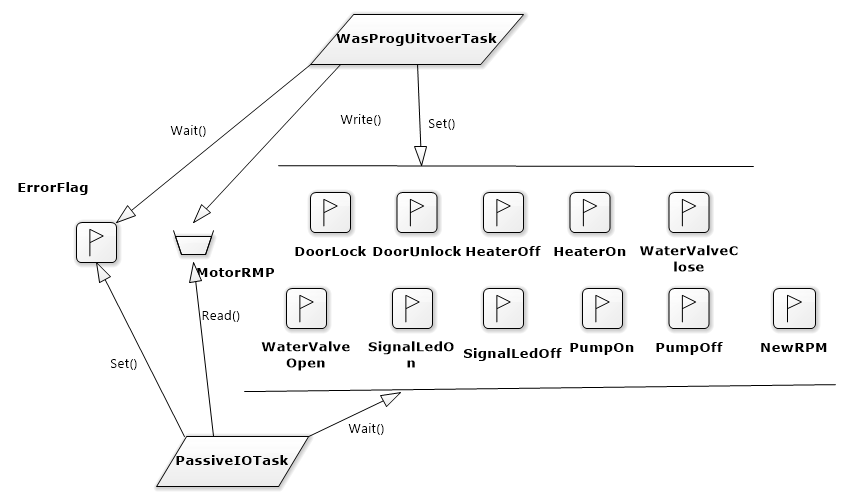
### 4.2.7 Overzicht alle tasks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Type taak | Periode | Deadline | Prioriteit |
| PeriodiekIOTask | I/O periodiek | 200 ms | 50 ms | 1 |
| PassiefIOTask | Asynchroon |  | 100 ms | 2 |
| WasProgUitvoerTask | Asynchroon |  | 50 ms | 0 |
| WasProgBeheerTask | Asynchroon |  | 250 ms | 4 |
| WebSockedTask | Asynchroon |  | 250 ms | 3 |
| LogTask | Passief |  | 500 ms | 5 |

## 4.3 Concurrency model

In dit hoofdstuk wordt er gekeken naar de synchronisatiemiddelen tussen tasks. Per 2 task wordt er gekeken welke keuzes zijn gemaakt en wat de alternatieven zijn. In Bijlage 2 is het hele concurrency diagram te zien.

### 4.3.1 WasProgUitvoerTaks en PassiveIOTask



Figuur 3

In figuur 3 is de communicatie tussen de WasProgUitvoerTask en de PassiveIOTask te zien. Voor het aansturen van de hardwarecomponenten is gekozen om gebruik te maken van vlaggen en een pool voor de motorsnelheid.

Als er een error optreed tijdens het aansturen van de hardwarecomponenten wordt er een error vlag gezet. De WasProgUitvoerTask zal hier op reageren en de wasmachine stil leggen. Wat een mooier alternatief zou zijn, is dat er aan de WasProgUitvoerTask meegegeven wordt welk component de error reageert. Dit zou kunnen aan de hand van een error nummer in een mailbox te schrijven.

### 4.3.2 WasProgUitvoerTask en PeriodiekeIOTask

## D:\Dropbox\School\THO6\Solution\img\Concurrencydiagram3.png

Figuur 4

In figuur 4 is de communicatie tussen de WasProgUitvoerTask en de PeriodiekeIOTask te zien. De PeriodiekIOTask schrijft de temperatuur en het waterlevel in de pools van WasProgUitvoerTask. En als allebei de waarden zijn geschreven zet de PeriodiekeIOTask de newValueFlag. Hier is gekozen om gebruik te maken van pools, want als de WasProgUitvoerTask geen tijd heeft om de pools uit te lezen worden de oude waardes overschreven met de nieuwe.

Wat hier anders had gekund is, dat er gebruik gemaakt wordt van 2 vlaggen 1 per poo.

### 4.3.3 WasProgUitvoerTaks en LogTask

## D:\Dropbox\School\THO6\Solution\img\Concurrencydiagram4.png

Figuur 5

In figuur 5 is de communicatie tussen de WasProgUitvoerTask en de LogTask te zien. De communicatie is heel simpel. De WasProgUitvoerTask schrijft in de logChannel en verder is er geen communicatie.

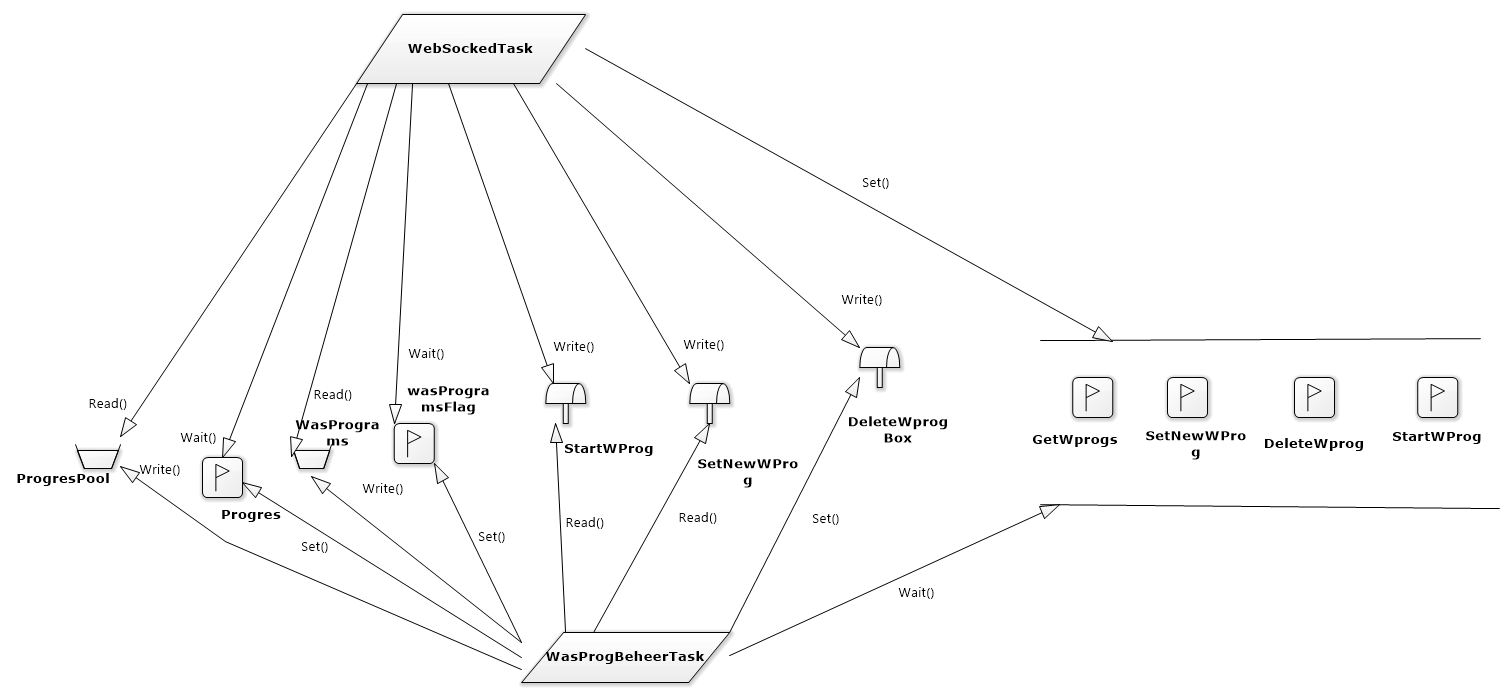
### 4.3.4 WasProgUitvoerTask en WasProgBeheerTask

## D:\Dropbox\School\THO6\Solution\img\Concurrencydiagram5.png

Figuur 6

In figuur 6 is de communicatie tussen de WasProgBeheerTask en de WasProgUitvoerTask te zien. De communicatie is maar 1 kant op van de WasProgBeheerTask naar de WasProgUitvoerTask. Dit gebeurt aan de hand van een mailbox. Er is gekozen voor een mailbox zodat als er nog een was phase in staat de WasProgBeheerTask weet dat er iets mis is met het systeem.

Wat een verbeterpunt aan dit concept zou zijn, is als er ook communicatie was van de WasProgUitvoerTask naar de WasProgBeheerTask. Zo zou het dan mogelijk zijn om error meldingen door te geven. Als de WasProgUitvoerTask nu een error ontvangt van de PassiveIOTask krijgt de WasProgBeheerTask dit niet te weten en gaat door met zijn werk.



Figuur 7

## 

### 4.3.5 WasProgBeheerTask en WebSockedTask

In figuur 7 is de communicatie met de WasProgBeheerTask en de WebSocketTask te zien. Voor het ontvangen van start, nieuw en delete wasprogramma’s is gebruik gemaakt van mailboxen. Dit hadden ook pools kunnen zijn want gebruikers zullen niet sneller wasprogramma’s kunnen toevoegen dan het systeem ze kan verwerken.

Voor het aangeven dat de WebSocket wasprogramma’s wilt ontvangen is gebruik gemaakt van een vlag. De wasprogramma’s zal in een pool worden geschreven.

De status wordt ook in een pool geschreven het is niet erg als deze overschreven zou worden als de WebSocket deze nog niet gelezen heeft.

## 4.4 Dynamic model

In dit hoofdstuk staan de STD’s van de controllers met toelichting er op.

### 4.4.1 PeriodiekeIOHandler

## D:\Dropbox\School\THO6\Solution\img\Periodiekeiohandler.png

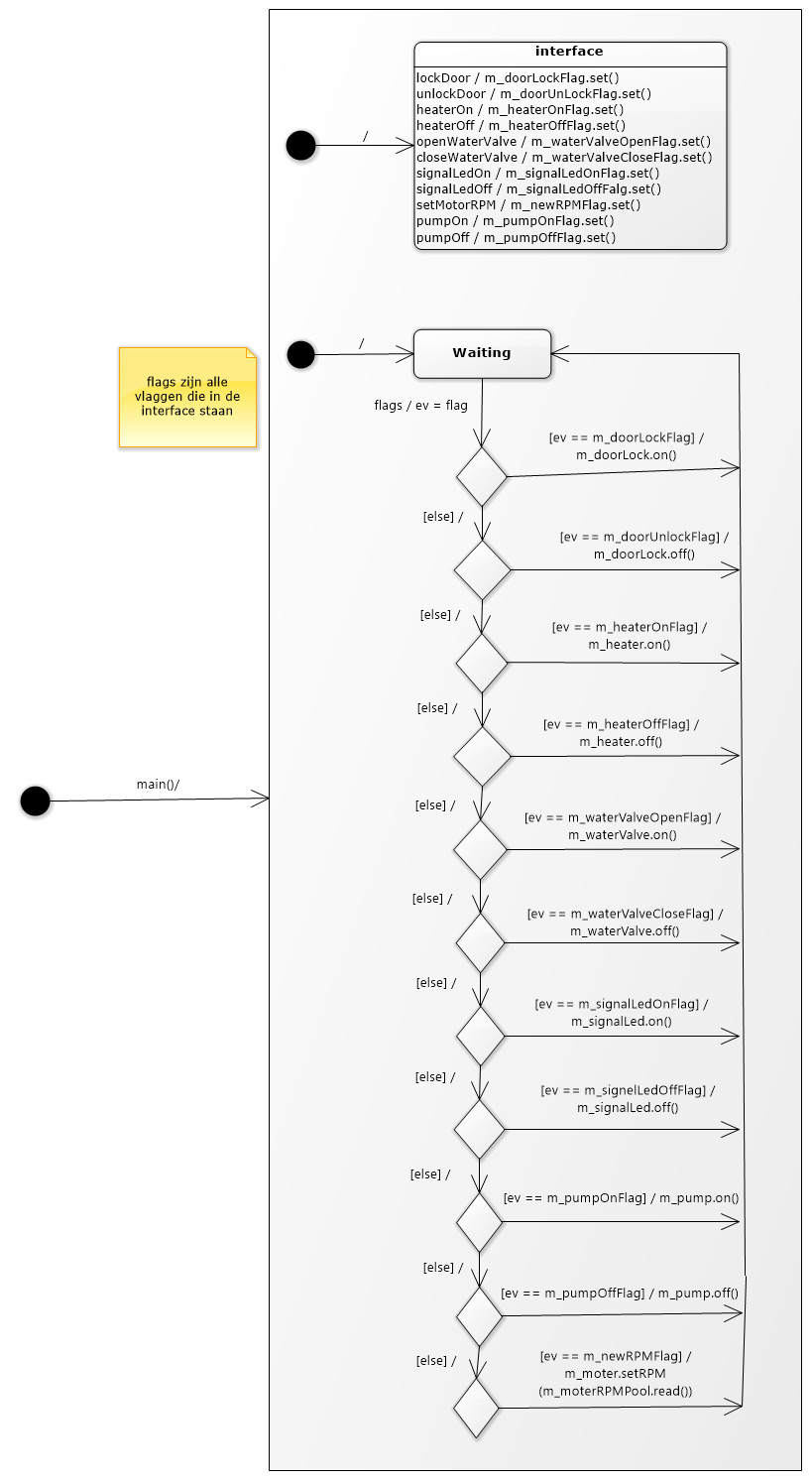
Figuur 8

De PeriodiekeIOHandler suspend na het starten van de main. Dit wordt gedaan omdat er geen wasprogramma loopt, dus het ophalen van de temperatuur en water level is overbodig.

De task wordt op running gezet door de WasProgUitvoerTask wanneer de wasmachine een opdracht krijgt. Na de opdracht wordt de PeriodiekeIOHandler weer op suspend gezet.

Als de Task op running wordt gezet komt het in een wait terecht. Zodra er 200 ms voorbij zijn geeft de m\_clock een signaal en gaat de task in een ranged for loop de EventSource\* langs. Elke event source heeft te maken met Uart communicatie, dus na elke UpdateState gaat de task in sleep van 20 ms om de Uart niet teveel te belasten.

Als alle EventSoruce\* zijn geweest gaat de task weer in wait.



Figuur 9

### 4.4.2 PassiveIOHandler

In figuur 9 is de STD van de PassiveIOHandler te zien. Het lijkt aardig wat, maar is eigenlijk simpel. De Task start standaard in waiting en wordt komt uit waiting als een van de vlaggen is gezet. Na het controleren welke vlag het is geweest en een commando naar de Uart te hebben geschreven gaat de task weer naar waiting.

### 4.4.3 WebSocket

## D:\Dropbox\School\THO6\Solution\img\Websocket.png

Figuur 10

In figuur 10 is de STD van WebSocket te zien. De WebSocket start standaard in waiting en wacht tot er een vlag wordt gezet of de klok voorbij is. Als de klok voorbij is wordt de named pipe gecontroleerd. Als er in geschreven wordt handelt de WebSocket het bericht af en gaat weer naar waiting. Als een val de vlaggen gezet is handelt die het onderdeel af wat bij die vlag hoort.

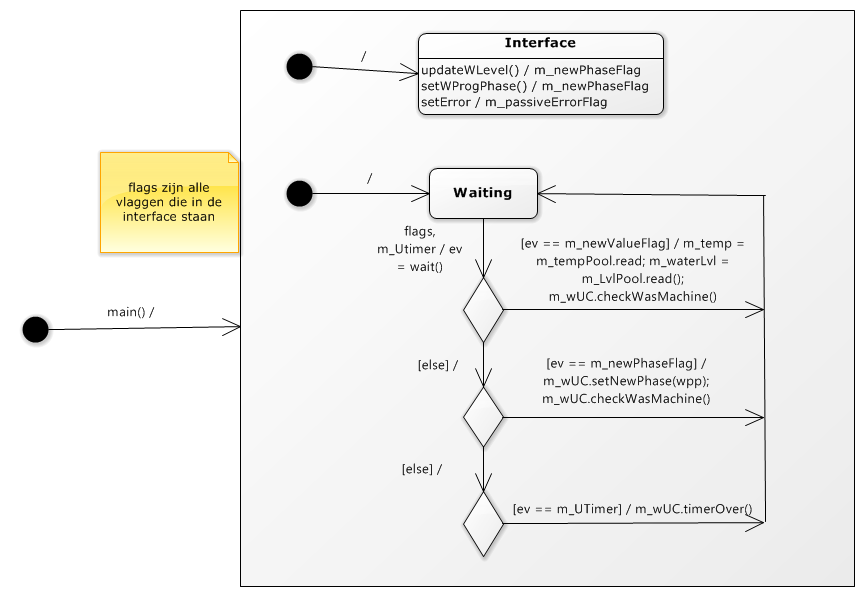
### 4.4.4 WasProgrammaBeheerController

## D:\Dropbox\School\THO6\Solution\img\Wasprogrammabeheercontroller.png

Figuur 11

In figuur 11 is de STD van WasProgrammaBeheerController te zien. De task start standaard in waiting en wacht tot er een vlag gezet is. Alleen de m\_startWProgFlag start een wasprogramma, de rest heeft te maken met het ophalen, toevoegen en verwijderen van wasprogramma’s. Als de m\_startWprogFlag gezet is wordt de functie washing aangeroepen. In deze functie geeft de phase door aan WasProgUitvoerHandler en zet de m\_phaseTimer. Als deze timer afgaat wordt er een nieuwe phase gestuurd naar WasProgUitvoerHandler en wordt de timer met een nieuwe waarde gezet. Ondertussen wordt ook de m\_updateTimer gezet om de WebSocket van updates te voorzien en wordt gecanceld als het wasprogramma klaar is. Ondertussen kunnen ook nog wasprogramma’s worden toegevoegd en verwijdert. Als er opnieuw een wasprogramma gestart wordt, wordt dit genegeerd.

### 4.4.5 WasProgUitVoerHandler



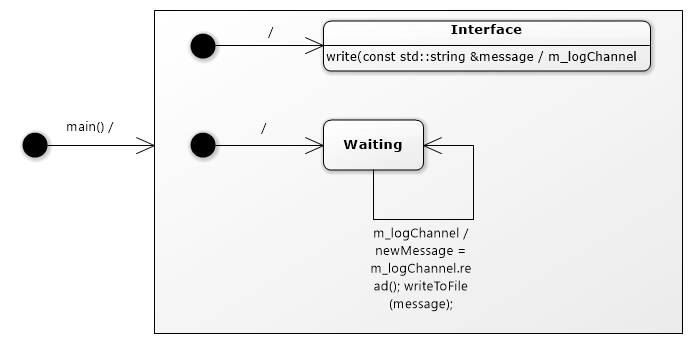
Figuur 12

In figuur 12 is de WasProgUitvoerHandler te zien. Deze task start standaard in waiting. Deze task wacht eerst alleen op de setWProgPhase flag, want de timer is nog niet gezet en de PeriodiekeIOHandler die de m\_newValueFlag zet is suspended.

Als er een nieuw phase binnenkomt wordt de PeriodiekeIOHandler op resumed gezet en wordt er nu ook gewacht op de m\_newValueFlag.

De m\_UTimer is bedoeld om de motor de andere kant op te laten draaien, dus zodra de motor gestart wordt zal de timer geset worden..

### 4.4.6 LogTask



Figuur 13

In figuur 13 is de STD van de LogTask te zien. De LogTask start in waiting en wacht tot er in zijn channel geschreven wordt. Zodra er in zijn channel geschreven is gaat de task weer naar waiting.

# Communicatie tussen de webserver en de wasmachine

Als webserver is gekozen voor een simpele c++ webserver waar de source code van aan te passen is. Doordat de source code aan te passen is kan er anders omgegaan worden bij bepaalde HTML files. Zo is het weergeven van de wasprogramma’s pagina een voorbeeld van een zo’n file. In de HTML code van wasprogrammas.html file staat een afgesproken teken waar de wasprogramma’s moeten komen te staan. De webserver weet niet welke wasprogramma’s er zijn, dus stuurt een  “GET” bericht naar de wasmachine via de afgesproken named pipe (Kerrisk, 2016). De wasmachine stuurt via een andere afgesproken named pipe de wasprogramma’s naar de webserver. De webserver vult op de plek van het afgesproken teken de wasprogramma’s in en stuurt deze naar de browser.

Op deze manier die hierboven beschreven is loopt alle communicatie tussen de webserver en wasmachine. Er wordt gebruik gemaakt van 2 named pipes. Een maakt de webserver aan en de andere de wasmachine. De Webserver schrijft in de named pipe van de wasmachine en de wasmachine schrijft in die van de webserver.

# Tests

## 6.1 Wasmachine Applicatie

De wasmachine applicatie wordt op verschillende manier getest, sommige in combinatie met de website. De tests komen voort uit de vastgestelde constraints.

### 6.1.1 Task Deadlines Testen :

Bij het uitvoeren van de individuele taken van de wasmachine , zoals het starten van de motor , waterpomp of het opwarmen van water. Deze functies moeten binnen de gegeven deadline uitgevoerd worden.

### 6.1.2 Error Management :

Het programma moet verschillende errors herkennen en hierop de juiste feedback geven. De problemen met de wasmachine moeten overeenkomen met de gegeven feedback van het programma. In de test zullen we Errors simuleren en kijken of het programma deze errors kan herkennen en juist afhandelen.

### 6.1.3 Noodknop:

De noodknop moet functioneren en ook snel genoeg kunnen reageren. De tijd die tussen het drukken op de noodknop en het daadwerkelijk stoppen van de wasmachine moet binnen een seconde gebeuren. Ook moet het programma kunnen vaststellen in welke toestand hij zat toen de knop was ingedrukt.

### 6.1.4 Ram en Rom:

Het gebruik van de de Rom en Ram van de Raspberry pi moet vastgelegd worden om een dit gebruik zo efficiënt mogelijk te houden. Hierbij zal een benchmark test uitgevoerd worden.

### 6.1.5 Webapplicatie

Bij de webapplicatie komen meerdere tests naar voren , gericht op gebruikersgemak en snelheid van de communicatie tussen de webapplicatie en de wasmachine.

### 6.1.6 Weergave Website

De voortgang van het wasprogramma moet accuraat weergegeven worden op de website. Deze afwijking is makkelijk te testen door het timer van de machine en de website

### 6.1.7 Gebruiksgemak:

De interface van de website moet makkelijk zijn voor een nieuwe gebruiker. Een nieuwe gebruiker moet zonder hulp de wasmachine aan kunnen sturen.

### 6.1.8 Communicatie Website

De tijdsduur van het verzenden en ontvangen tussen de webapplicatie en de wasmachine moeten niet te lang duren. Door het simpelweg te timen is de snelheid hiervan vast te leggen.

# Realisatie

Bij het realiseren van het project werd er in eerste instantie gebruik gemaakt van een Apache webserver. Het was de bedoeling dat aan de hand van PHP / Java script gecommuniceerd moest worden met de c++ wasmachine. Dit was veel lastiger dan verwacht en uiteindelijk ook niet gerealiseerd. Om toch communicatie te hebben tussen de webserver en wasmachine is er gekozen voor de oplossing die in hoofdstuk 5 beschreven staat.

Het overschakelen op de c++ webserver is achter veel te laat gebeurt, waardoor er op dit moment weinig mogelijk is vanaf de website. Hier zal na het inleveren van dit document nog aan gewerkt worden.

Er was een probleem dat er niet op meerdere vlaggen gewacht kon worden. Hier is eerst een andere oplossing voor gezocht zodat elke task maar 1 vlag had. Deze manier was echter geen mooie oplossing. Na contact op te hebben genomen met Marten Wensink werd duidelijk hoe er op meerdere vlaggen gewacht kon worden en is de aangegeven manier verwerkt in het systeem.

Nog weinig testen zijn gedaan tijdens het schrijven van dit document. Dit komt i.v.m. tijd gebrek. Er zullen nog zoveel mogelijk testen uitgevoerd worden. Deze testen zullen later toegestuurd worden of tijdens de demo besproken.

# Evaluatie

In retrospect komen er verschillende zaken naar voren die bij nader inzien beter gedaan konden worden. Dit nemen de teamleden van het project mee naar de volgende opdrachten als les.

Door de lineaire taakverdeling was er weinig interactie tussen de leden, er zou meer resultaat geleverd worden als er als een team naar de taken had gekeken op meerdere moment. Dit is een probleem dat op te lossen is door verschillende agile development frameworks toegepast zouden worden zoals de Scrum methode.

Omdat dit project een herkansing is waren er geen ingeplande lessen en moest het team zelf contact zoeken voor feedback, hierin is het team verlegen gebleven. Het zou verstandig zijn om een paar vast feedbackmoment in te plannen in overeenkomst met een docent aan het begin van het project.

Het systeem verwerkt op dit moment de feedback niet als er een fout optreed bij het aansturen van een component. De error vlag wordt alleen gezet als de deur niet gesloten is. Op de Error vlag reageert alleen de WasProgrammaUitvoerenController. Deze zal dan in de huidige phase geen niet meer naar de PassiveIOHandler sturen. Het afvangen en verwerken van error had beter gemoeten, maar is i.v.m tijd problemen niet gerealiseerd.

De meeste taken, met uizondering op PeriodiekeIOHandler, worden gestart in waiting. Veel taken zullen niets doen totdat er een wasprogramma gestart wordt. Dit is niet netjes. Het zou beter zijn geweest als de taken die afhankelijk zijn van een pasprogramma op suspend of standby worden gezet en pas als het nodig is ze in de stand waiting te zetten.

De zeep inlaad is niet meegenomen in de documentatie en programma. Dit werd gedaan als om tijd te besparen, omdat het maar een team van 3 personen is. Uit eindelijk is de zeep inlaad een klein gedeelte en had meegenomen moeten worden.

# Conclusies en Aanbevelingen

Van de wasmachine werken de basis onderdelen. Zo is het mogelijk om een wasprogramma te starten van de website en deze zal dan uitgevoerd worden. Echter het afhandelen van errors in hardware componenten is niet gerealiseerd. Hierdoor kunnen situaties ontstaan dat de wasmachine niet werk, maar geen terug koppeling gegeven wordt aan de gebruiker. Om een veilige wasmachine te hebben moet het afhandelen van errors welgebeuren, dus is het verstandig om dit eerst te realiseren voordat de Wasmachine in productie gaat.

# Bibliografie

Consumentenbond. (sd). *Hoe werkt een wasmachine?* Opgehaald van http://www.consumentenbond.nl/wasmachine/extra/hoe-werkt-een-wasmachine/#item0

Dijk, S. v. (2015). *Handig! Alle wasprogramma’s voor een wasmachine bij elkaar*. Opgehaald van greenem.nl: http://www.greenem.nl/wasprogrammas-voor-een-wasmachine/#1440592735602-1098ee79-928e

Feng Xia, L. T. (2012). The Internet of Things.

Kerrisk, M. (2016, 12 12). *fifo*. Opgehaald van man7.org: http://man7.org/linux/man-pages/man7/fifo.7.html

*Monteur op afstand Hoe werkt een wasmachine*. (sd). Opgehaald van monteuropafstand.nl: http://www.monteuropafstand.nl/wasmachine-reparatie/hoe-werkt-een-wasmachine

Wensink, M. (2015, 11 5). Beschrijving wasmachine.

## Bijlagen

## 11.1 Klassendiagram

## 11.2 Concurency