Solution Architectuur

Team: 4

Wouter van den Heuvel	1564952
Wilco Louwerse	1655993
~~Robin Noten~~	~~1671668~~
Daniel Klomp	1661521

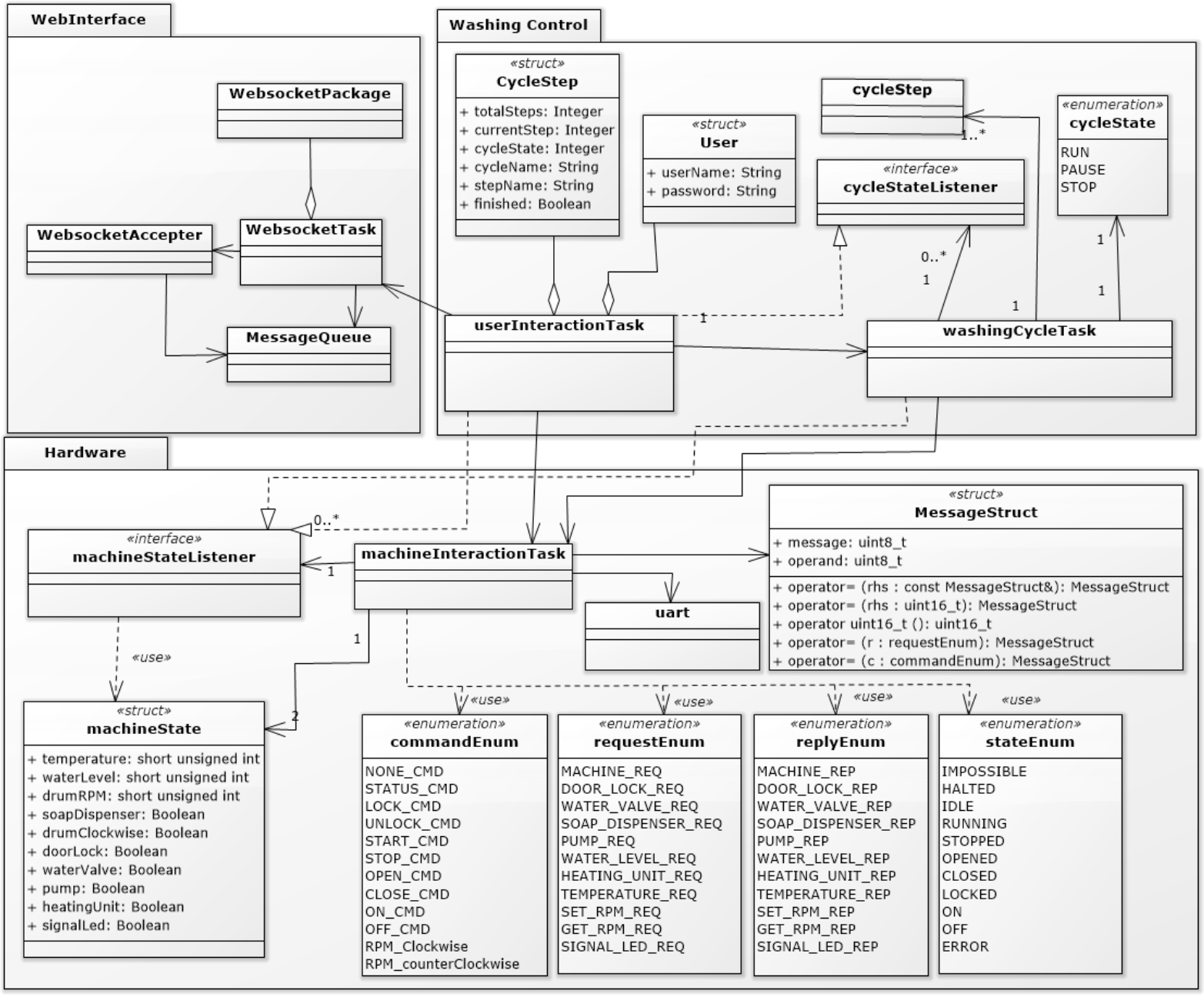
Datum: 10-12-2015

Versie: 0.1

Index:

Inhoud

1 Klassen Diagram



Het klassendiagram brengt in beeld wat de connecties zijn tussen klassen binnen het systeem. Door middel van dit diagram wordt ook duidelijk hoe en waarom bepaalde klassen met elkaar communiceren. Voor dit project staat het klassendiagram op de vorige pagina. Dit is niet het gehele klassendiagram, omdat de methodes en atributten van de klassen zijn weggelaten. Het volledige uitgewerkte klassendiagram staat in de de bijlagen(Hoofdstuk 9,4).

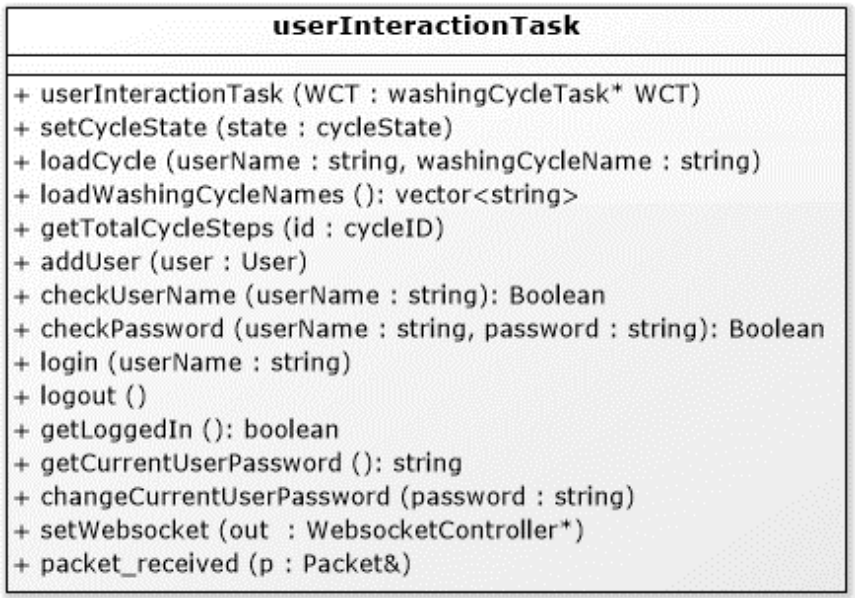
Voor zowel de washingCycleTask als de machineInteractionTask is er voor gekozen om het listener pattern te gebruiken, zodat andere klassen relatief eenvoudig statusberichten kunnen ontvangen met betrekking tot de voortgang van een wasprogramma, of de huidige status van het systeem.

De keuze om de machineInteractionTask verantwoordelijk te maken voor het beheren van de temperatuur en waterniveau komt voort uit het feit dat deze klasse in alle de statistieken nauwlettend moet observeren , en dat het verbergen van de hardware er voor zorgt dat kennis van het fysieke systeem op één enkele plaats bestaat.

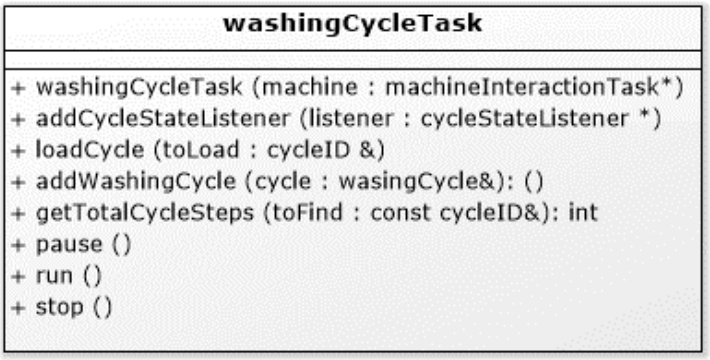
In plaats van aparte boundary klassen voor alle onderdelen van de wasmachine is er voor gekozen om deze details te verbergen in de machineInteractionTask, allereerst om de complexiteit van klassen die bij hardware details moeten kunnen komen te beperken, en omdat de hardware is afgeschermd door middel van de UART.

UserInteractionTask:
De UserInteractiontask is verantwoordelijk voor het communiceren tussen de WebsocketTask en de WashingCycleTask. De taak leest uit de CycleStateListener en de MachineStateListener de status van het wasprogramma en de status van de wasmachine om die weer te kunnen geven op de website.

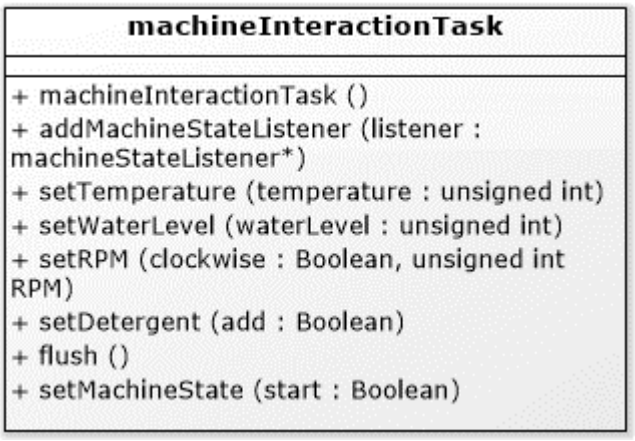
Verder stuurt de UserInteraction Task berichten door vanaf de WebsocketTask naar bijvoorbeeld de WashingCycleTask. Voorbeeld: als de gebruiker het wasprogramma wilt pauzeren, dan stuurt de WebsocketTask dat bericht naar de UserInteractionTask, zodat de taak dit vervolgens door kan sturen naar de washingCycleTask.



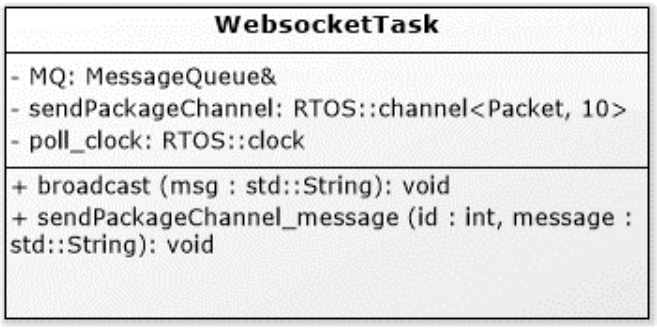
WashingCycleTask:
De WashingCycleTask is verantwoordelijk voor het draaien van het wasprogramma. De taak communiceert met de wasmachine via de MachineInteractionTask om het wasprogramma uit te voeren. Ook stuurt de washingCycleTask de status van het wasprogramma naar alle CycleStateListeners, zodat andere taken kunnen weten hoever het wasprogramma is.



MachineInteractionTask:
De MachineInteractionTask is verantwoordelijk voor de communicatie tussen de wasmachine en de washingCycleTask. Dit doet hij door middel van de uart klassen. Ook stuurt de MachineInteractionTask de status van de wasmachine (temperatuur, waterlevel, etc.) naar alle MachineStateListeners zodat deze taken weten in welke status de wasmachine verkeerd.



WebsocketTask:
De WebsocketTask is verantwoordelijk voor de communicatie tussen de Websocket en de UserInteractionTask. De WebsocketTask krijgt van de UserInteractionTask de status van de wasmachine en van het huidig draaiende wasprogramma. Dit wordt vervolgens doorgestuurd naar de Websocket. Daarnaast kan de WebsocketTask de commando's RUN, PAUSE en STOP krijgen van de Websocket als deze commando's worden gegeven vanaf de website.



2 Taakstructurering

De taakstructurering is gemaakt als vervolg op het klassendiagram. In dit diagram zijn bepaalde klassen samengevoegd tot groepjes om nog meer duidelijkheid te krijgen van de communicatie binnen het systeem en om te voorkomen dat meerdere gelijksoortige taken vergelijkbare dingen gaan doen.

2.1 Objecten

Object	ID	Object omschrijving	Type taak	Periode	Deadline	Prioriteit
WashingMachine	1	Interface voor fysieke wasmachine	Asynchroon I/O	-	250ms	0
WebSocket	2	Internet interface	Asynchroon I/O	-	500ms	2
UART	3	Comminucatie tussen de wasmachine en het Rtos	Asynchroon I/O	-	50ms	0
DisplayController	4	Use case “Display machine state” Bepaalt wat wordt laten zien op het display	Asynchroon	-	500ms	2
MachineRead Controller	5	Use case “Read machine state” Laat de WashingMachine periodiek de status updaten.	Periodiek	500ms	250ms	0
WashingCycle Controller	6	Use case “Control washing cycle”	Periodiek	500ms	250ms	1
UserProfile Controller	7	Use case “Manage user profile”	Asynchroon	-	500 ms	3
LoadCycle Controller	8	Use case “Load saved washing cycle” Laadt de WashingCycle en slaat dit ook op.	Asynchroon	-	500 ms	3

In de opsomming van de objecten op de vorige pagina zijn alle losstaande Boundary objecten van de wasmachine samengevoegd tot een object, de Washingmachine. In dit object zitten alle bijbehorende objecten van de wasmachine, zoals de deurvergrendeling, de motor en de noodknop. Een compleet overzicht van de objecten staan vermeld in de wasmachine emulator beschrijving(Wensink, M.*“Beschrijving wasmachine-emulator”*).

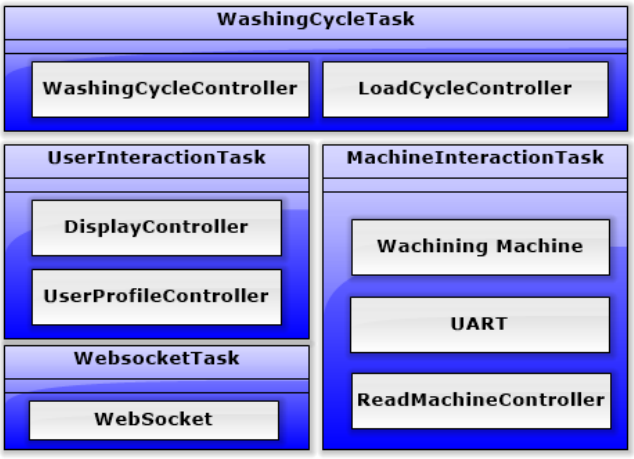
2.2 Taken

Om de objecten goed met elkaar te kunnen laten communiceren, en om het systeem snel en efficiënt te maken, worden de objecten samengevoegd tot taken. Deze taken lopen(“Runnen”) constant in het systeem.

De MachineInteractionTask bestaat uit de Washing Machine, de UART en de ReadMachineController. Deze taak heeft de verantwoordelijkheid om met de objecten van de wasmachine te communiceren. Dat houdt in berichten versturen en de status van de wasmachine ontvangen. Deze taak wordt om de 500ms aangeroepen (periodiek) om de status van de wasmachine op te vragen en nieuwe berichten naar de wasmachine te versturen. Dit is 500ms omdat de status van de wasmachine niet drastisch zal veranderen in een kortere periode dan 500ms, maar langer dan 500ms is te lang. De deadline voor deze taak is 250ms omdat de taak snel uitgevoerd moet worden, echter, de UART heeft een vertraging van 10ms per bericht, wat de taak zal ophouden.
Deze taak heeft de hoogste prioriteit omdat een bericht naar de wasmachine, bijvoorbeeld een noodgeval om te stoppen, altijd voor gaat.

De WashingCycleTask bestaat uit de washingCycleController. Deze taak heeft de verantwoordelijkheid om de veranderingen van de wasmachine te analyseren en te vergelijken met het huidige wasprogramma. Verder wordt deze taak periodiek aangeroepen om de 500ms omdat de MachineInteractionTask om de 500ms een nieuwe status levert aan de WachingCycleTask. De deadline voor deze taak is 250ms omdat het systeem niet teveel vertraagd, maar wel druk zet achter de taak.
Deze taak heeft een prioriteit lager dan de MachineInteractionTask omdat de washingCycleTask wel snel berichten moet versturen naar de MachineInteractionTask, maar de berichten naar de wasmachine zelf belangrijker zijn.

De UserInteractionTask is verantwoordelijk voor het communiceren tussen het RTOS en de WebsocketTask. De UserInteractionTask ontvangt de berichten van de WebsocketTask en stuurt de status van de wasmachine (MachineInteractionTask) door naar de WebsocketTask. Deze taak wordt asynchrone aangeroepen door de WebsocketTask of de MachineInteractionTask. De deadline voor deze taak is 500ms, omdat het doorsturen en ontvangen van berichten van de WebsocketTask niet erg tijdsgebonden is.

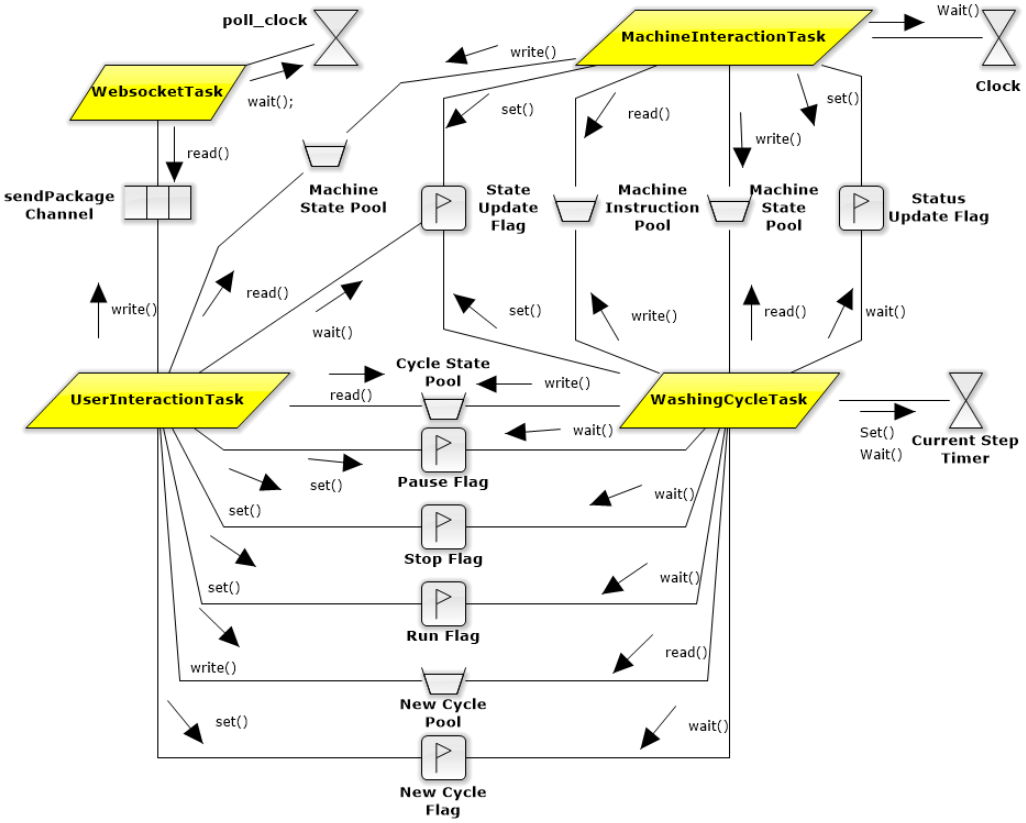


De WebsocketTask is verantwoordelijk voor het ontvangen en doorsturen van informatie tussen het RTOS en de websocket. De WebsocketTask ontvangt de berichten van de UserInteractionTask en stuurt de ontvangen status van de wasmachine door naar de Websocket. Deze taak wordt asynchrone aangeroepen door de Websocket of de UserInteractionTask. De deadline voor deze taak is 500ms, omdat het doorsturen en ontvangen van berichten van de Websocket niet erg tijdsgebonden is.

Taak	Onderdelen	Type taak	Periode	Deadline	Prioriteit
MachineInteractionTask	1,3,5	Periodiek	500ms	250ms	1
WashingCycleTask	6, 8	Periodiek	500ms	250ms	2
UserInteractionTask	4,7	Asynchrone I/O-taak	-	500ms	3
WebsocketTask	2	Asynchrone I/O-taak	-	500ms	3

3 Concurrency Diagram

In het Concurrency diagram wordt de interactie tussen de verschillende taken duidelijk gemaakt. De taken binnen het systeem communiceren met elkaar via synchronisatie methodes.



In het Concurrency diagram voor de wasmachine praten/communiceren de meeste taken via Pools. Dit is omdat op de informatie die wordt doorgestuurd gewacht moet kunnen worden en dit de meest effectieve manier is om gegevens door te geven.

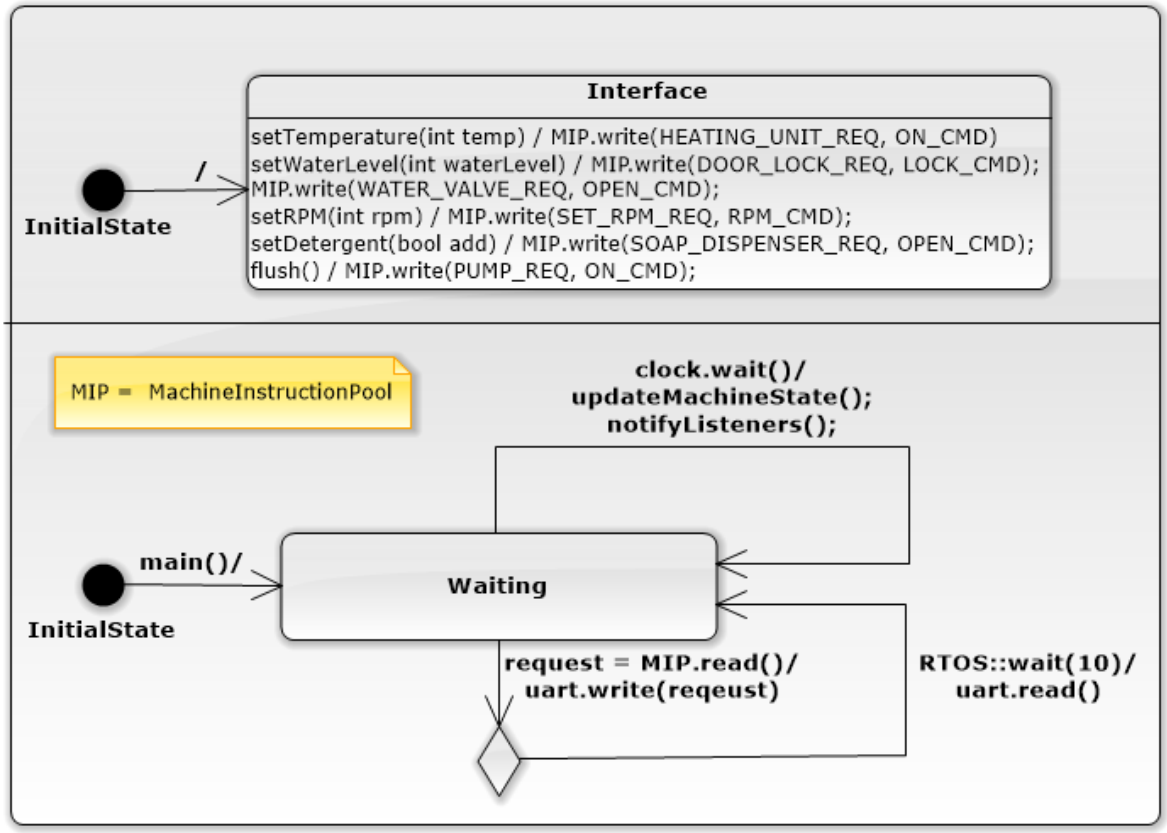
Een goed voorbeeld is de communicatie vanaf de MachineInteractionTask naar de UserInteractionTask en de WashingCycleTask door middel van de MachineStatePool. In deze communicatielijn schrijft de MachineInteractionTask de status van de wasmachine in de MachineStatePool. Dit had ook een Channel kunnen zijn, een mechanisme dat vrijwel het zelfde effect heeft als een Pool met een Flag. Het gebruik van een Pool met een flag is echter handiger omdat daarbij altijd maar een (actuele) status word doorgegeven en wanneer er een nieuwe status word doorgegeven die de oude overschrijft. Zodat niet zoals het bij een channel werkt oude statussen nog worden verwerkt terwijl je eigenlijk alleen de allerlaatst gestuurde status wilt gebruiken.

4 State Transition Diagrams

In de State Transition Diagrams(STD) wordt de volledige werking van het systeem uitgewerkt. Elke taak die het systeem uitvoert wordt volledig uitgewerkt. Verder is ook te zien hoe de taken met elkaar communiceren.

4.1 MachineInteractionTask

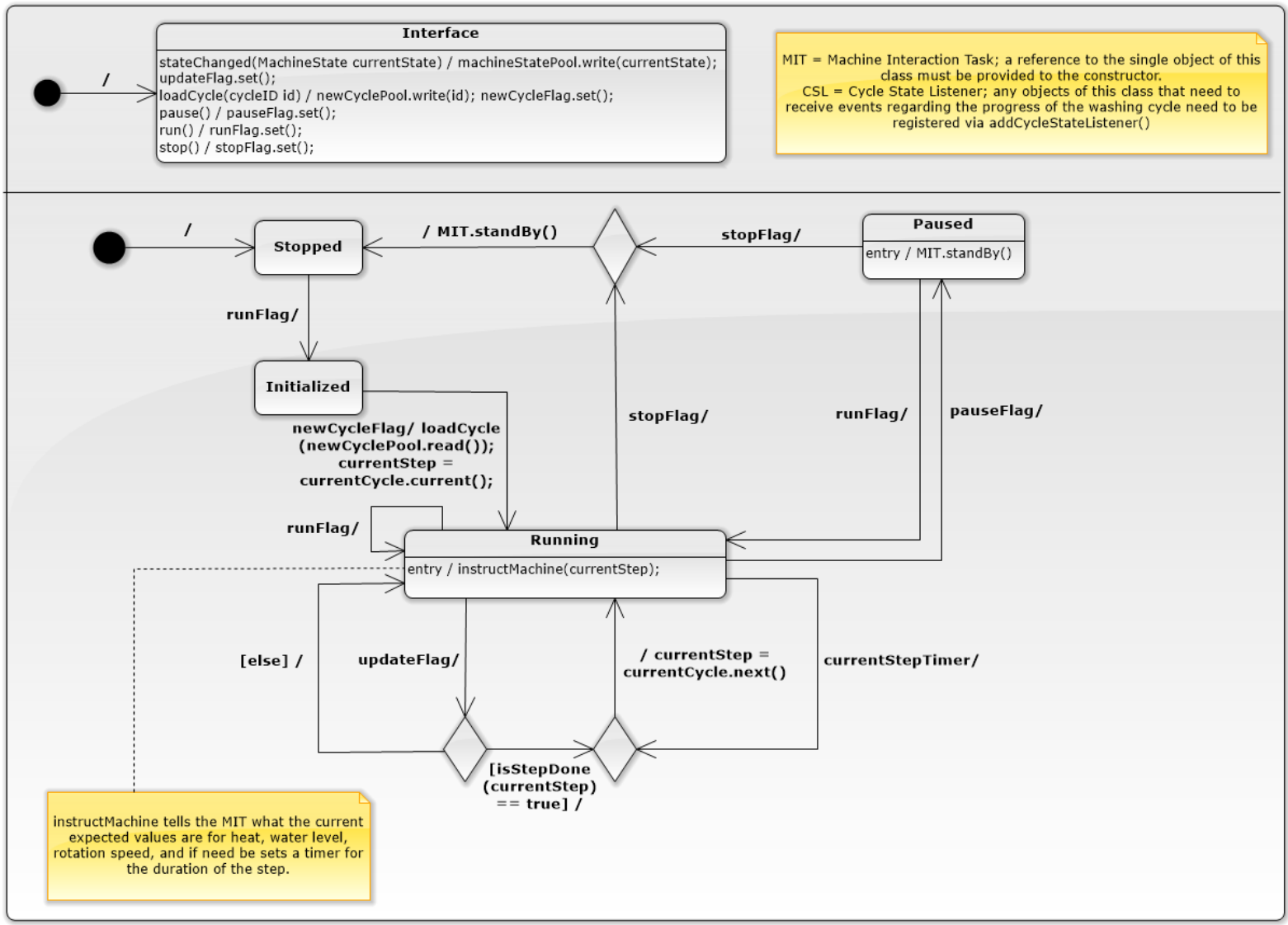
In de machineInteractionTask wordt alle communicatie met de tastbare wasmachine geregeld door bytes naar de Uart te sturen en de byte die terug wordt gegeven uit te lezen.



Bij het starten van het programma wacht de machineInteractionTask eerst totdat er ofwel een klokslag plaatsvind, of tot er een stuuropdracht in de daarvoor toegewezen pool wordt gezet. Bij een klokslag worden de verschillende onderdelen van de wasmachine ondervraagt, en vind er een event plaats voor alle aangemelde listeners met de meest recente informatie. Als er een stuuropdracht in de pool staat wordt deze aan de machine doorgegeven, waarna de MIT eerst de response uitleest, voordat er wordt teruggekeerd naar de wachttoestand.

4.2 WashingCycleTask

In de washingCycleTask wordt het huidige wasprogramma uitgevoerd/bijgehouden. De status van het huidige wasprogramma kan van buiten af aangepast worden (pauzeren of stoppen), dit wordt ook in deze taak verwerkt. Naast dit stuurt de washingCycleTask ook naar alle geregistreerde cycleStateListeners wat de status is van het huidige wasprogramma en de fase waarin deze verkeerd.

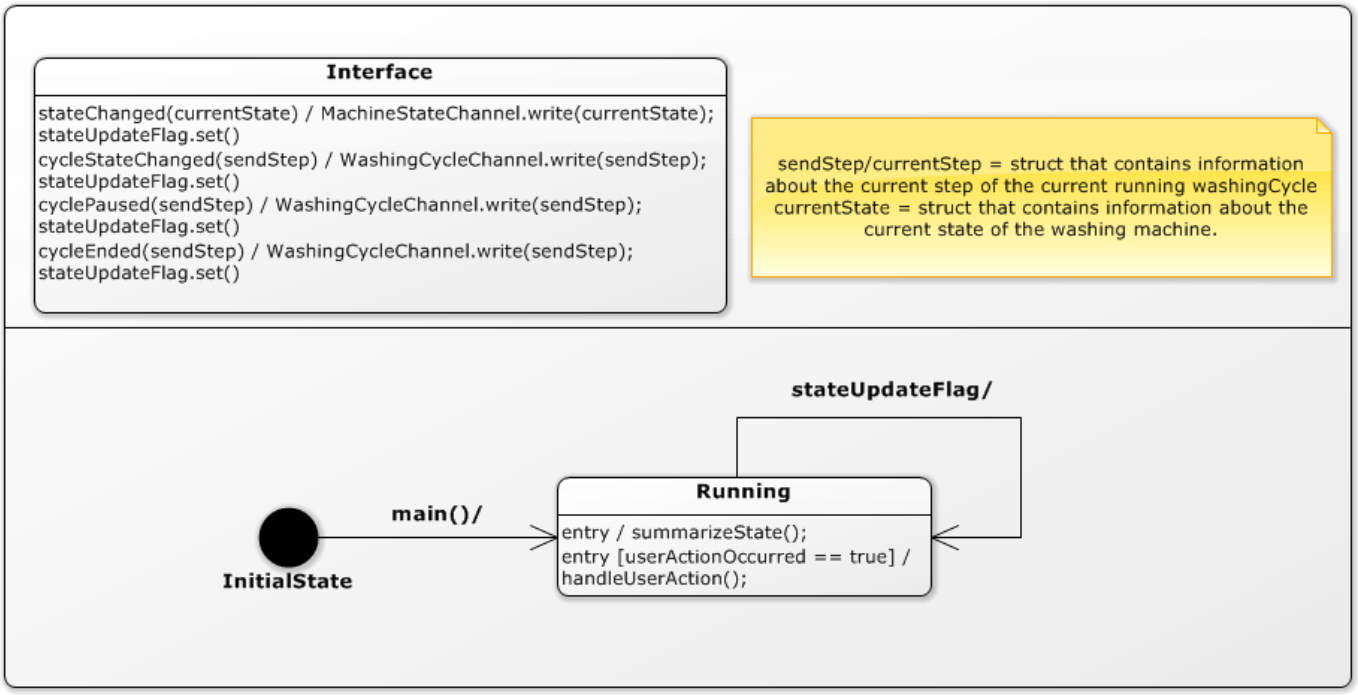


Bij het starten van het programma komt de washingCycleTask eerst in de ‘Stopped’ status terecht. In deze status wordt gewacht tot dat de run flag geset wordt, zodra dat gebeurd gaat deze task door naar zijn ‘Initialized’ state. Zodra de washingCycleTask van zijn ‘Stopped’ status naar ‘Initialized’ status gaat wordt de loadCyclePool uitgelezen, hierin wordt gezet welk wasprogramma wordt gedraaid. Het uitgelezen wasprogramma wordt opgeslagen in de “ongoing” ‘washingCycle’ zodat de Task onthoud welk wasprogramma hij aan het draaien is.

Na het vaststellen van het wasprogramma die moet draaien komt de washingCycleTask in een nieuwe staat (‘Running’)terecht. In deze grote loop wordt gekeken of de status van het wasprogamma moet veranderen van “RUN” naar “PAUSE” of “STOP”. Als dit niet het geval is blijf de status van deze taak ‘Running’ en zal hij alle fases van het wasprogramma een voor een uitvoeren. Wordt het wasprogramma gestopt of is het klaar, dan zal de status weer terug gaan naar ‘Stopped’. Ook is er nog een ‘Paused’ status, dit is ook een loop waarin de taak blijft vanaf wanneer de status van het wasprogramma “PAUSE” wordt tot dat het weer veranderd naar “RUN” of “STOP”. Waarbij in het geval van “RUN” de taak weer verder gaat in zijn ‘Running’ status en bij het geval van “STOP” terug komt in zijn ‘Stopped’ status.

4.3 UserInteractionTask

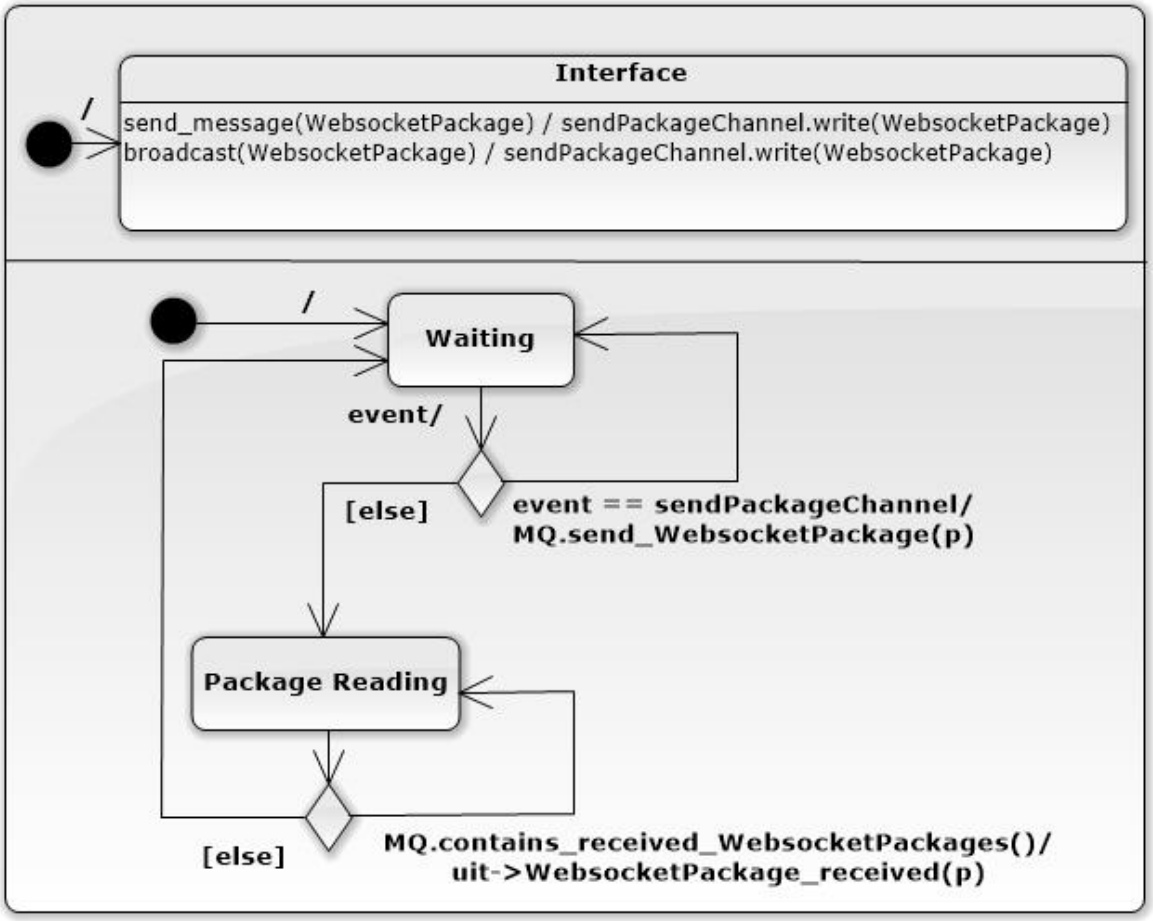
In de UserInteractionTask worden alle berichten die zijn verzonden door de gebruiker verwerkt en wordt de bijbehorende actie uitgevoerd.



Deze taak is constant aan het wachten totdat de stateUpdateFlag wordt geset, dit gebeurd wanneer er van buiten deze task een opdracht wordt gestuurd naar deze task. Als dit gebeurd en de flag wordt geset dan zal deze task de gegeven opdracht uitvoeren. De opdrachten die gestuurd worden naar deze task zijn opdrachten voor de websocket en worden dus daar naar toe gestuurd, maar deze task luisterd ook naar de websocket door middel van handleUserAction(). De uitgelezen opdrachten vanaf de websocket worden doorgestuurd naar hun betreffende task die deze opdracht moet uitvoeren.

4.4 WebsocketTask

In de WebsocketTask worden alle berichten die zijn verzonden door de gebruiker ontvangen en doorgestuurd naar de UIT (UserInteractionTask)



Deze taak is constant aan het wachten totdat er een event plaats vind, als dit event veroorzaakt is doordat er wat is geschreven in de sendPackageChannel (dit gebeurd wanneer er van buiten deze task een opdracht wordt gestuurd naar deze task) dan stuurt hij het verstuurde WebsocketPackage door naar de MQ (MessageQueue). Als het event dat plaats vond niet veroorzaakt is door de sendPackageChannel (maar door de clock) dan komt deze task in een nieuwe status ‘Package Reading’, in deze status worden alle geschreven packages doorgestuurd naar de UIT.

5 Bronvermeldingen

Ooijen, W. "*TI C++ software rules"* (2014) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/Studiemateriaal/2015https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/Studiemateriaal/2015-2016-TI-C++-software-rules-1-0.pdf2016-TI-C++-software-rules-1-0.pdf

Ooijen, W. "*Software Engineering in C++"*(2013) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/50/TCTI-V2CPSE1-15/Studiemateriaal/2015https://cursussen.sharepoint.hu.nl/fnt/50/TCTI-V2CPSE1-15/Studiemateriaal/2015-2016-V2SECP1-reader.docx2016-V2SECP1-reader.docx

Wensink, M. / W.v.Ooijen,"*Realtime System Programming*" (2013) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/8/TCTI-V2RTSP1-10/default.aspx?RootFolder=%2ffnt%2f8%2fTCTI-V2RTSP1-10%2fStudiemateriaal%2freader&FolderCTID=&View=%7b2C0A9E64-0ABB-4B83https://cursussen.sharepoint.hu.nl/fnt/8/TCTI-V2RTSP1-10/default.aspx?RootFolder=%2ffnt%2f8%2fTCTI-V2RTSP1-10%2fStudiemateriaal%2freader&FolderCTID=&View=%7b2C0A9E64-0ABB-4B83-94CA-36695FA8F7CE%7d94CA-36695FA8F7CE%7d

Wensink, M.*"Beschrijving wasmachine-emulator"* (2015) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/Studiemateriaal/Beschrijving%20wasmachine-emulator.pdf

Wensink, M."*2015-2016-V2TH06 notes"*(2015) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/Studiemateriaal/2015-2016https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/Studiemateriaal/2015-2016-V2TH06-notes.pdfV2TH06-notes.pdf