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

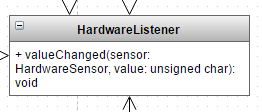
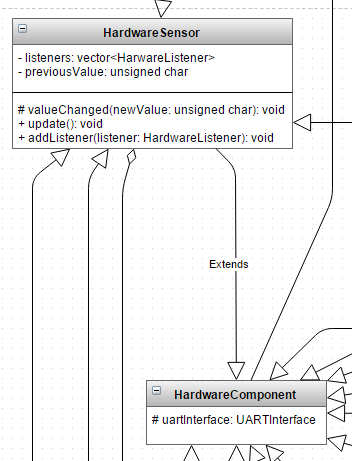
Versie: 0.1 (feedback)

Klassendiagram

**Hardware**

Er zijn twee soorten hardware, de actoren en de sensoren. De actoren vallen onder de HardwareComponent interface. De sensoren vallen onder de HardwareSensor interface, een sensor is ook een HardwareComponent.

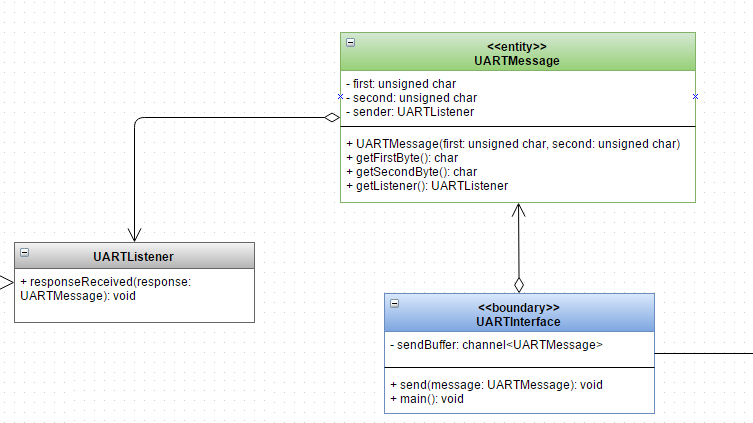
Elk HardwareComponent heeft toegang tot het UARTInterface om berichten te kunnen versturen. De HardwareSensor houdt bij wie zijn listeners zijn en stuurt updates wanneer zijn waarde veranderd. Dit gebeurt via het HardwareListener interface. Elke class die het HardwareListener interface implementeerd kan zich aanmelden bij een HardwareSensor.



**UART**

De UARTInterface is een rtos::task en kan asynchroon berichten versturen via de UART verbinding. Een UARTMessage bestaat uit twee bytes en een sender. De sender is een UARTListener. Het UARTListener interface implementeerd de methode responseReceived, zo kan de UARTInterface het antwoord terug sturen naar degene die het verstuurd had wanneer het beschikbaar is.

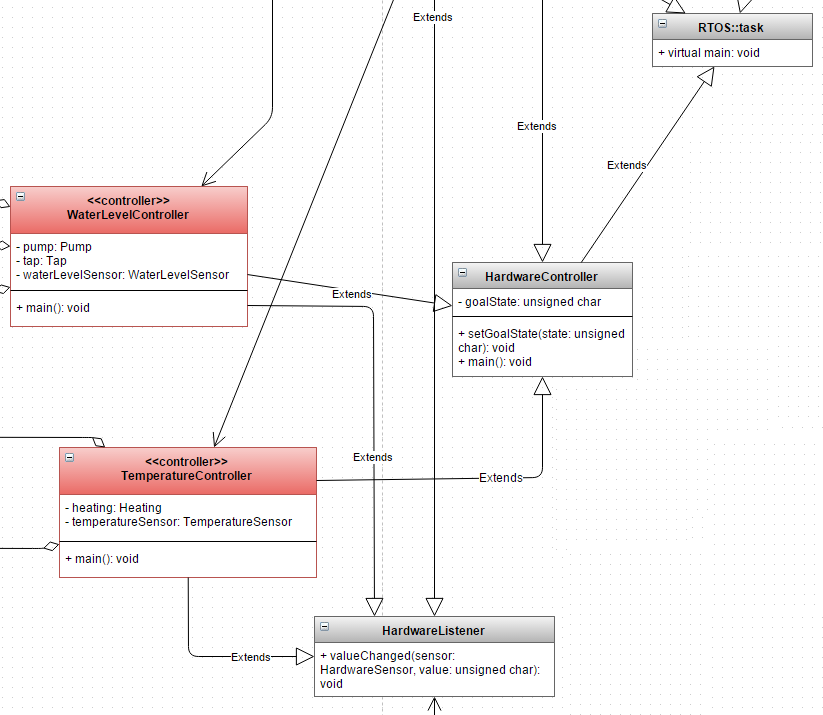
De UARTInterface heeft een channel waar het alle berichten in op slaat, zo hoeft niemand te wachten tot het vorige bericht verzonden is.



**Controllers**

De HarwareComponent worden gereguleerd door HardwareControllers, elke HardwareController heeft een goalState waarin die terecht wil komen. Iedere controller is een rtos::task waardoor ze los van elkaar werken.

De HarwareControllers krijgen nieuwe waardes van de sensoren binnen via het HardwareListener interface.

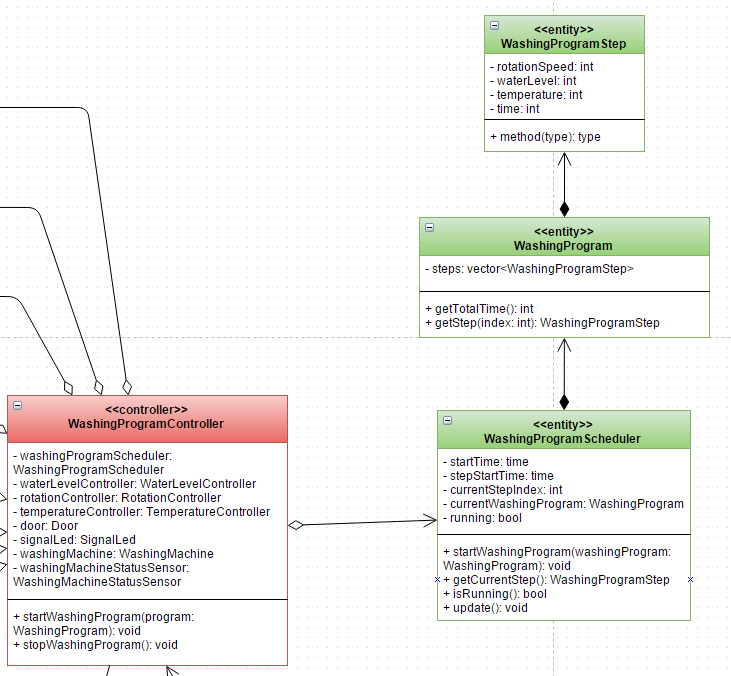


**Wasprogramma’s**

De WashingProgramController heeft directe toegang tot\ de Door, SignalLed en SoapTray. Verder kent WashingProgramController 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.

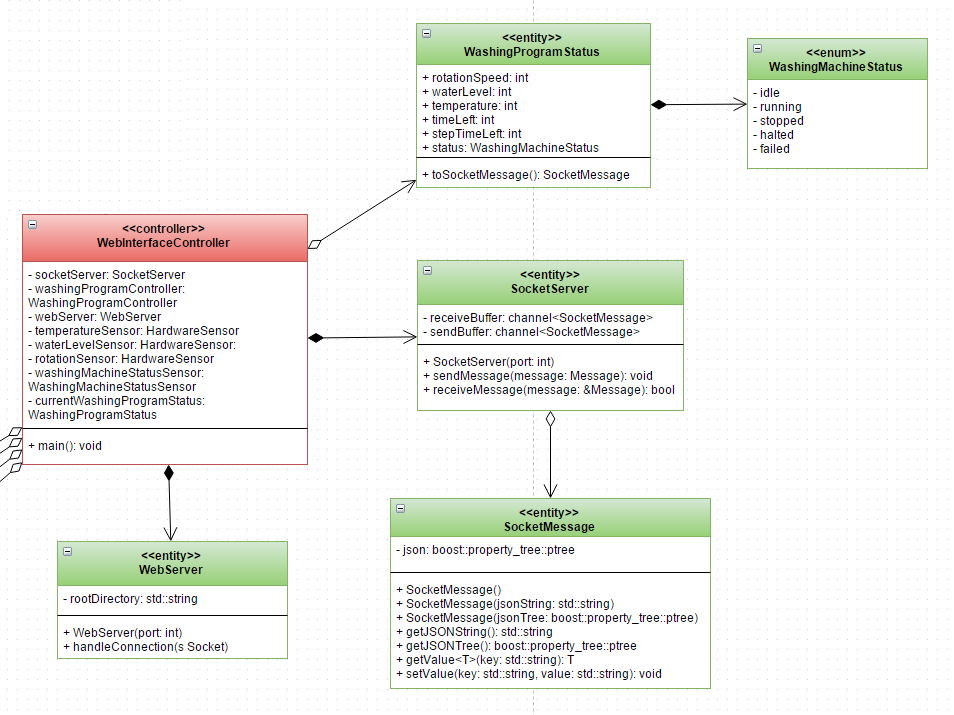
De WashingProgramController kent de UARTInterface waardoor het de status van de wasmachine kan opvragen. Ook kan het de wasmachine starten en stoppen.

**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.

De WebInterfaceController is registreerd als listener bij de sensoren. Deze waardes slaat die op in de WashingProgramStatus en geeft die door aan de SocketServer wanneer er iets veranderd.



Taakstructurering

**I/O taakstructurering**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | Type taak | Periode | Deadline | Prioriteit |
| Web interface | periodiek | 60000 ms | 100 ms | 6 |
| RotationSensor | periodiek | 500 ms | 10 ms | 0 |
| TemperatureSensor | periodiek | 500 ms | 10 ms | 1 |
| WaterLevelSensor | periodiek | 500 ms | 10 ms | 2 |
| Heating | asynchroon |  | 10 ms | 3 |
| Motor | asynchroon |  | 10 ms | 4 |
| Soap | asynchroon |  | 10 ms | 5 |
| UART interface | asynchroon |  | 10 ms | 6 |
| Signal led | geen |  |  |  |
| Door lock | geen |  |  |  |
| Pump | geen |  |  |  |
| Tap | geen |  |  |  |

De UARTInterfaceTask verstuurd asynchroon berichten via de UART en geeft de antwoorden terug via een listener pattern. Dit kost 10 ms per bericht. Deze taak hoeft alleen actief te worden wanneer er een bericht wordt verstuurd en is daarom niet periodiek.

De WashingProgramTask vraagt de status van de wasmachine op en start eventueel de wasmachine op. Ook kijkt het of er een nieuw wasprogramma moet worden opgestart.

De TemperatureTask stuurt de Heating aan en update de TemperatureSensor, dit gebeurt namelijk altijd tegelijkertijd. Het updaten van de sensor wordt asynchroon gedaan door de UART interface en kost daarom bijna geen tijd voor de TemperatureTask.

De MotorTask polled de RotationSensor en bepaald de te zetten snelheid van de Motor.

De WaterLevelTask update de WaterLevelSensor en stuurt aan de hand daarvan de Pump en Tap aan. In het ergste geval moet hij de Pump uitzetten en de Tap aanzetten.

De WebInterfaceTask stuurt elke minuut een update naar de browser met de resterende tijd van het wasprogramma. Ook stuurt het een bericht wanneer het een update krijgt van 1 van de sensoren via de listener interface.

Deze samenvoegingen leiden tot de volgende taken:

**Interne taakstructurering**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Object | Type taak | Periode | Deadline | Prioriteit |
| UARTInterfaceTask | asynchroon |  | 10 ms | 0 |
| TemperatureTask | periodiek | 500 ms | 10 ms | 1 |
| MotorTask | periodiek | 500 ms | 10 ms | 2 |
| WashingProgramTask | periodiek | 1000 ms | 10 ms | 3 |
| WaterLevelTask | periodiek | 500 ms | 10 ms | 4 |
| WebInterfaceTask | periodiek | 60000 ms | 100 ms | 5 |

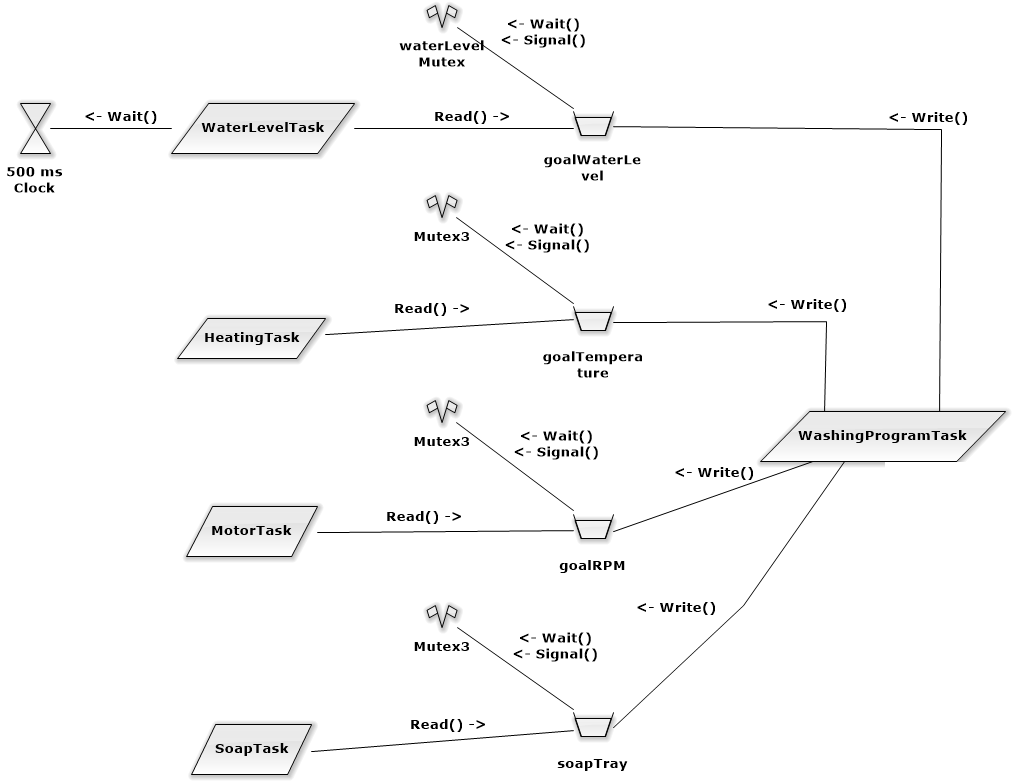
Op de volgende pagina staan alle objecten ingedeeld per taak in het taak structuur diagram.

**Taak structuur diagram**



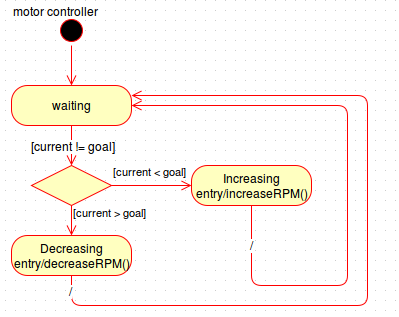
Concurrency model

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.

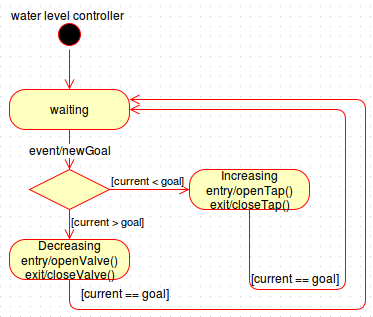


Dynamic model

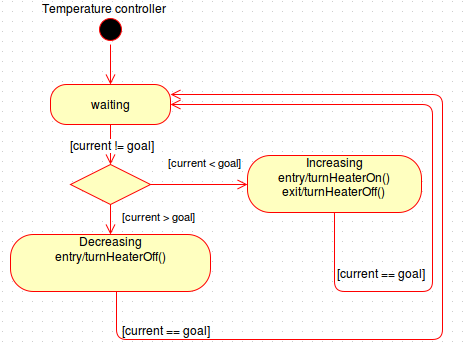
Motor Controller



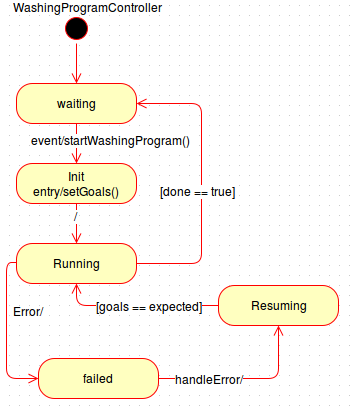
De motor controller heeft 3 states. De waiting state. Daar wacht de controller als de current RPS niet gelijk is aan het doel. Als het niet gelijk is aan het doel dan gaat de controller naar de volgende state. Als huidige RPS lager is dan het doel dan gaat de controller naar een increasing state. Als de RPS al hoger is dan gaat die naar decreasing state. hij blijft deze circle maken tot current gelijk is aan het doel. Dan komt die weer in een wacht staat terecht tot het doel aangepast word.



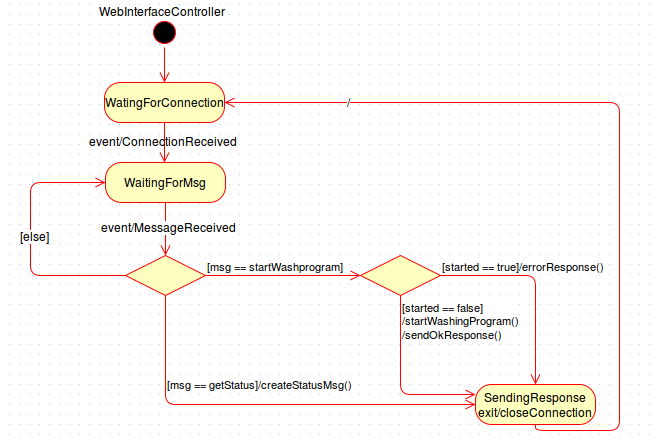
De water level controller heeft 3 states. Hij begint in de waiting state en wacht op een nieuw doel. Als het doel veranderd dan gaat de controller naar een nieuwe state afhankelijk of het doel lager of hoger is dan het huidige water niveau. Als het huidige water niveau lager is dan gaat de controller naar de increasing state. In die state zet die de kraan open, als het current gelijk is aan het doel dan word de kraan dicht gedaan. Als het doel lager is dan het huidige water niveau dan word de pomp aan gezet. Die gaat uit zodra het water op het goeie niveau zit.



De temperatuur controller heet ook 3 states. De wait state, daarin wacht de controller op een nieuw doel. Als die een nieuw doel krijgt dan gaat die naar de Increasing of Decreasing state afhankelijk of huidige temperatuur lager of hoger ligt dan het doel. Als de temperatuur hoger ligt dan het doel dan word de verwarming uitgezet. En dan moet er gewacht worden tot het water afkoelt. Als de temperatuur lager ligt dan het doel dan word de verwarming aan gezet tot het doel gehaald is. Als het doel gehaald is word de verwarming uitgezet en dan gaat de controller weer terug naar de waiting stand.



Washing program controller die begint in de waiting state. Daar blijft die in staan tot er een start wasprogramma event ontvagen word. Dan in de init state worden alle doelen aan hun eigen controller door gegeven. Daarna komt die in de running state. Als er in de running state een error voor komt dan gaat die naar de failed state. Als de error afgehandled word dan gaat de controller in de resuming state. Daar word alles weer naar het verwachte peil gebracht. Daarna gaat de controller verder in de running state. Als het programma klaar is dan gaat de controller terug naar de waiting state.



De webinterface controller begint met het wachten op een verbinding. Zodra er een verbinding is word er gewacht op het bericht van de gene die verbind. Hier zijn er 3 mogelijkheden.

1. het bericht is het start wasprogramma bericht
2. het bericht is het get status bericht
3. het bericht is geen van de 2

Bij de 1e mogelijkheid word er gekeken of het was programma al gestart is of niet. Als het programma al gestart is dan word er een error gestuurd. Als er nog geen programma gestart is dan word het was programma gestart en een OK response terug gestuurd. Daarna word de de connectie gesloten en wacht de controller op de volgende verbinding.

Bij de 2e mogelijkheid word er een status bericht gemaakt. Die word door gestuurd naar de client waarna de verbinding gesloten word en er gewacht word op de volgende verbinding.

Bij de 3e mogelijkheid word het ontvangen bericht niet herkend en verder genegeerd. De controller gaat dan wachten op het volgende bericht en kijken of die wel klopt.

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 was programma 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"}