TH06 Team 12

Plan van Aanpak

Datum

5 december 2015

Auteurs

Christiaan van den Berg 1660475
Aydin Biber 1666849
Martijn van Dijk 1660713
Chiel Douwes 1666311

Docenten

Wouter van Ooijen
Joost Schalken
Marten Wensink
Jan Zuurbier

Hogeschool Utrecht Faculteit Natuur en Techniek

Inhoudsopgave

1 Inleiding

Er is door de Hogeschool Utrecht opdracht gegeven voor de ontwikkeling van een wasmachine die via een netwerkverbinding te bedienen is. Steeds meer apparaten hebben een internetverbinding, en zijn verbonden met andere apparaten. (Blaauboer, 2014)

De opdracht betreft het ontwikkelen van een prototype om in een kleine groep testers te evalueren wat de meerwaarde is van een wasmachine met internetverbinding is.

2 Producten

2.1 Inleiding

Hier worden alle op te leveren producten benoemd die van toepassing zijn tot het project.

- Projectdocumentatie
  - Teamcontract
  - Plan van Aanpak
  - Requirements Document
  - Requirements Architechture
  - Solution Architechture
  - Technisch Verslag
- HTML, CSS en JavaScript code van de webpagina's
- C++-code van de wasmachine
- Demonstratie

2.2 Projectdocumentatie

Teamcontract

De afspraken die onderling in het team gemaakt zijn, uiteenlopend van afspraken over beschikbaarheid

Plan van Aanpak

Dit document. Het bevat de basisopzet en -planning van het project.

Requirements Document

Hierin worden de functionele en niet-functionele eisen vermeld die op basis van een interview met de klant zijn vastgesteld, met prioriteiten volgens de MoSCoW methode.

Requirements Architecture

De requirements architecture bevat de functionele systeemeisen: Wat moet de software van het systeem doen? Dit wordt vastgelegd met een of meerdere use case diagrammen. Bij iedere usec case hoort een activity diagram om deze toe te lichten. Verder bevat de Requirements Architecture een Constraints Model. Hierin worden de niet-functionele eisen vastgelegd.

Solution Architecture

De solution architecture bevat een klassenmodel, een concurrency model en een dynamisch model. Het klassenmodel beschrijft de soorten objecten in het systeem, en welke attributen methodes de klassen hebben. Verder beschrijft het klassenmodel de relatie tussen de verschillende klassen in het systeem. Het concurrency model beschrijft hoe de objecten in het systeem samenwerken en data uitwisselen. Het dynamische model bevat state transition diagrammen die de werking van controller-klassen beschrijven.

Technisch Verslag

Het technisch verslag wordt in dit document op SharePoint beschreven. Het informatie over de volgende onderwerpen:

- Onderzoek
- Requirements Architecture
- Solution Architecture
- Realisatie
- Evaluatie

- Conclusies en aanbevelingen

2.3 HTML, CSS en JavaScript code

De front-end code van de webinterface die geladen wordt in de webbrowser. Deze code wordt opgeleverd via GitHub.

2.4 C++-code van de wasmachine

De back-end code die de verantwoordelijk is voor het uitvoeren van wasprogramma's. Verder wordt vanuit deze software gecommuniceerd met de webbrowser voor het starten van wasprogramma's en om de status van een draaiend wasprogramma te bekijken. Deze code wordt opgeleverd via GitHub

2.5 Demonstratie

Een live demonstratie van het eindproduct aan de opdrachtgevers.

3 MoSCoW

3.1 Inleiding

In dit onderdeel bespreken wij de lijst van functionaliteiten die tijdens het interview met de klant zijn achterhaald. Wij maken een lijst van deze prioriteiten aan de hand van de MoSCoW methode. Deze methode houdt in dat functionaliteiten een letter krijgen toegewezen die aanduid wat hun prioriteit is binnen het systeem. De anduiding is als volgt:

3.2 Legenda

M	Must	Deze eis is verplicht voor een goede afronding van het project.
S	Should	Deze eis moet er in komen maar het project komt niet in gevaar wanneer dit niet mogelijk is.
C	Could	Deze eis zou kunnen worden toegevoegd indien de Must en Should eisen behaald zijn.
W	Would	Indien er tijd extra over is kunnen deze eisen worden geïmplementeerd.

3.3 Indeling Prioriteiten

Functionaliteit	Beschrijving	Prioriteit
Starten & Stoppen machine	De machine mag er tot een half uur over doen om het wasprogramma te starten. Stoppen moet vrijwel direct zijn	M
Aanpassen temperatuur	De gebruiker moet de temperatuur van een wasprogramma moeten kunnen aanpassen aan de hand van beschikbare opties	M

Wasprogramma's	Er moeten standaard 3 wasprogramma's aanwezig zijn: Witte was, Fijne was en Bonte was	M
Toevoegen wasprogramma's	Iemand die in dienst is bij de klant kan indien gewenst een nieuw wasprogramma maken en toevoegen aan de lijst van beschikbare wasprogramma's	M
Inhoud wasprogramma's	Wasprogramma's bevatten de volgende gegevens: Duur, Temperatuur, Voorkeurstemperatuur en Centrifugesnelheid	M
Updaten wasprogramma's	Het updaten van de lijst van wasprogramma's moet automatisch gaan	M
Inloggen webinterface	De gebruiker logt in op de webinterface door middel van een pincode	M
Inplannen wastaken	Wastaken moeten kunnen worden ingepland (wasprogramma uitvoeren naar X aantal uren)	M
Logbestanden	Er moeten logs worden bijgehouden van wat de gebruiker heeft gedaan	M
Inhoud logbestanden	De logbestanden moeten de volgende informatie tonen: Uren motor gedraaid, waterverbruik en stroomverbruik	M
Crashbeveiliging	Het systeem moet beveiligd zijn tegen crashes	M
Acties stroomuitval	Wanneer na een stroomuitval de stroom weer terug is moet het systeem automatisch verder gaan met het wasprogramma	M
Accepteren updates	De gebruiker moet per update aangeven of hij/zij deze update wilt ontvangen	S
Aanpassen pincode	De pincode moet door de gebruiker aanpasbaar zijn	S

Herstelcode pincode	Er moet een herstelcode beschikbaar zijn die bij het systeem wordt geleverd om de huidige pincode op te vragen	S
Kiezen pincode	De gebruiker kiest zelf een pincode bij het voor het eerst opstarten van de webapplicatie	S
Opties stroomuitval	De gebruiker kan via een optie in de webinterface kiezen of de machine na een stroomuitval automatisch verder gaat of dat het water wordt weggepompt	S
Meldingen webinterface	De huidige tijd en temperatuur van het systeem moet worden getoond aan de gebruiker	S

4 Onderzoek

4.1 Inleiding

Om dit project te kunnen maken is er onderzoek naar diversen onderwerpen nodig. Zonder dit onderzoek mist er veel informatie die er later in het project nodig zijn om problemen te verhelpen of het project sneller te laten lopen. Hiervoor zijn de volgende onderwerpen die wij gaan onderzoeken.

4.2 Read/Write met RTOS

Het doel van dit onderzoek is om een systeem te maken dat bijhoud waar het systeem was na stroomuitval of een systeem crash. Hiervoor is het nodig om de informatie op te slaan in een log bestand, maar omdat het maken en schrijven in een bestand een actie van het OS is kan dit het rest van het systeem in een sleep zetten. Om te onderzoeken of dit een probleem wordt met deadlines van het realtime systeem gaan wij onderzoeken hoe lang het duurt om een log file te maken, en dan meerdere keren het bestand openen, schrijven en daarna sluiten om te kijken hoe lang het duurt. Als deze periode de deadlines niet overschrijdt dan kunnen wij deze methode gebruiken in het systeem.

4.3 Wasprogrammas

Het doel van dit onderzoek is om beter verstand krijgen van de wasmachine en wasprogrammas om de hardware beter aan te sturen en zodat specialisten nieuwe wasprogrammas kunnen toevoegen. Dit kunnen wij doen door ons meer te verdiepen in de geleverde hardware, andere wasmachines bestuderen en onderzoeken hoe wasprogrammas precies in elkaar zitten.

4.4 Snelheid van de UI

Het doal van dit onderzoek is om de acties van de UI zo snel naar het systeem te brengen, zoals stoppen, starten en de temperatuur te laten zien in de UI. Dit kunnen wij doen door een debug functie toe te voegen die laat zien wanneer er een actie wordt uigevoerd in de UI en wanneer er een actie wordt uitgevoerd door het systeem. Met deze informatie kunnen wij uitzoeken hoe wij het systeem sneller kunnen laten reageren.

5 Kwaliteitseisen

5.1 Inleiding

Om er voor te zorgen dat de op te leveren producten goed leesbaar zijn worden er een aantal eisen gesteld aan deze producten.

5.2 Software

- De webinterface moet voldoen aan de door de MoSCoW beschreven eisen.
- De C++ code zal getest worden door middel van unit tests. Voordat de code opgeleverd wordt moeten alle unit tests slagen.
- De C++ code moet voldoen aan de deadlines.
- De C++ code moet gedocumenteerd worden, zowel de public als de private methods.
- De C++ code mag geen geheugenlekken hebben.
- De C++ code moet voldoen aan de code eisen zoals beschreven in het vak V2CPSE1.

5.3 Documentatie

- De documentatie mag maximaal drie spelfouten bevatten per pagina. In dien dit wel het geval is wordt het document niet opgeleverd verklaard.
- De diagrammen van de diverse modellen moeten goed leesbaar zijn.
- De diagrammen van de diverse modellen moeten overzichtelijk zijn. Om dit te bereiken mogen diagrammen niet te groot worden, en indien nodig opgesplitst worden in subdiagrammen.

6 Methode van kwaliteitsbewaking

Om de kwaliteit van het eindproduct te verzekeren zullen er een aantal maatregelen worden genomen:

6.1 Kwaliteit van de code

Geautomatizeerde tests

Er zullen geautomatischeerde tests gemaakt worden om er zeker van te zijn dat de code in werkende staat is. Deze tests zullen automatisch verifiëren of de code de bedoelde functionaliteit heeft, en als dit niet zo is dan zullen de tests dit aangeven.

Code reviews

Er zullen code reviews gedaan worden door de verschillende leden van het team om te verzekeren dat de geschreven code aan alle standaarden voldoet. Op deze manier kunnen wij er allemaal zeker van zijn dat de code goed leesbaar is, en het door iedereen begrepen en aangepast kan worden in het geval van bijvoorbeeld een bug.

Code reviews van docenten

Omdat wij er zelf niet zeker van kunnen zijn dat onze code reviews goed uitgevoerd worden, zullen wij waar mogelijk onze code door docenten laten overzien. Door het gebruikmaken van een (meer ervaren) derde partij om de code na te kijken kunnen wij zeker zijn dat onze code geen onopgemerkte problemen heeft, en de kwaliteit van de code zal in het algemeen verbeterd worden.

6.2 Gebruik van standaarden

LaTeX

Wij gebruiken LaTeXvoor alle documenten rondom het project. De reden hiervoor is om de layout van de documenten te standardisseren, en het makkelijker te maken om veranderingen in de documenten in git te kunnen synchronizeren. Zo kunnen wij effectiever samenwerken, hoeven we minder tijd te besteden aan het formatteren en opstellen van de documentatie en wordt de kwaliteit van de documenten verbeterd.

Code formatting

Wij zullen standaarden opstellen met betrekking tot hoe de code geformatteerd moet worden, en deze standaarden zullen in de code review op gebracht worden.

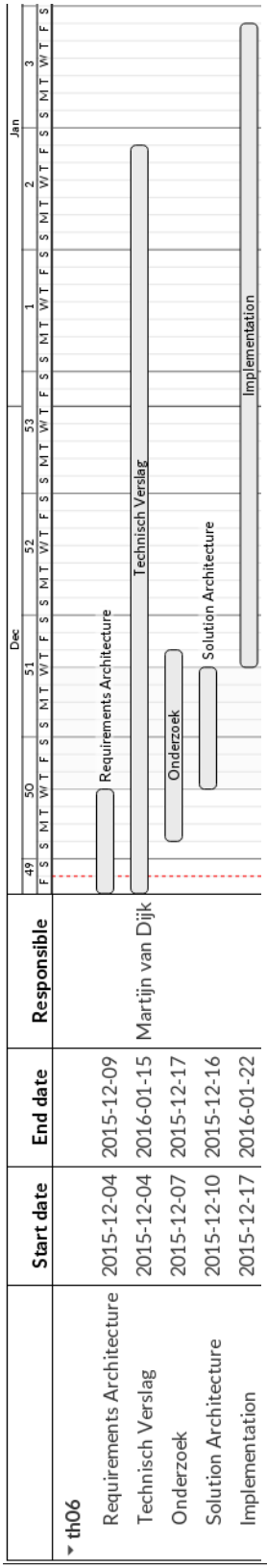
7 Projectactiviteiten

7.1 Mijlpalen

Hieronder bevindt zich de planning voor het gehele project. De genoemde data zijn de deadlines voor oplevering en reflecteren niet de werkelijke inleverdatum.

Teamcontract	11-11-2015 (Week 1)
Github Repo (met bijbehorende mappen)	11-11-2015 (Week 1)
Interview Opdrachtgever	30-11-2015 (Week 3)
Plan van Aanpak	02-12-2015 (Week 4)
Requirements Document (MoSCoW)	01-12-2015 (Week 4)
Requirements Architechture	09-12-2015 (Week 5)
Solution Architechture	16-12-2015 (Week 6)
Technisch Verslag	(Week 7 / Projectweek 1)
Eindproduct (software + hardware)	22-01-2016

7.2 Planning



	Start date	End date	Responsible
th06			
Requirements Architecture	2015-12-04	2015-12-09	Martijn van Dijk
Technisch Verslag	2015-12-04	2016-01-15	
Onderzoek	2015-12-07	2015-12-17	
Solution Architecture	2015-12-10	2015-12-16	
Implementation	2015-12-17	2016-01-22	

De bovenstaande planning is een screenshot van de OpenProject Web interface. De interactieve versie is te vinden op https://project-th06.martijnvandijk.net/projects/th06/timelines/2

7.3 Verantwoordelijkheden

Deelproduct of taak	**Eindverantwoordelijke**
Documentatie	
Onderzoek	Christiaan van den Berg
Plan van Aanpak	Martijn van Dijk
Requirements Document	Chiel Douwes
Requirements Architecture	Christiaan van den Berg
Solution Architecture	Aydin Biber
Technisch Verslag	Martijn van Dijk
Software	
Webinterface	Aydin Biber
Low-level interfacing met hardware	Martijn van Dijk
Demonstratie & Presentatie	Chiel Douwes

8 Risico's

Er zijn verschillende risico's die het project in gevaar kunnen brengen. Hier worden die risico's genoemd. Zie de Kwaliteitsbewaking voor wat de gepaste reactie is op deze risico's.

8.1 Uival van de hardware

Het kan gebeuren dat de hardware niet goed werkt, of compleet uitvalt. Dit vormt een risico voor het project omdat het team dan niet goed de code kan testen. In het geval van falende hardware zal de teamleider zorgen voor vervangende hardware.

8.2 Te hoge eisen

In het geval dat de eisen aan het project te hoog zijn, en niet haalbaar binnen de looptijd van het project zal er door de teamleider contact opgenomen worden met de opdrachtgever om te overleggen welke punten geschrapt of aangepast kunnen worden om de totale werklast te verminderen.

8.3 Hardware niet krachtig genoeg

In het geval dat de hardware van de Raspberry Pi niet krachtig genoeg is om de beschreven functionaliteit te implementeren zal er door de teamleider overlegd worden met de opdrachtgever over het schrappen of aanpassen van functionaliteit, of het ter beschikking stellen van krachtigere hardware.

Bibliografie

Blaauboer, R. (2014, oktober). Wonen tussen 100 sensoren: het internet of things start bij jou thuis. http://www.frankwatching.com/archive/2014/10/07/wonen-tussen-100-sensoren-het-internet-of-things-start-bij-jou-thuis/.