DE HAAGSE HOGESCHOOL

Afstudeerverslag

CGI

Raspberry Pi Solar Challenge

Definitief V1.0

Student:	Albert Haaksema
Begeleider:	Wesley Bosvelt
Team Lead:	Jeroen de Bekker
Begeleidend Examinator:	Nanny Jacobs
Tweede Examinator:	Tim Cocx

1. Versielijst en aanpassingen

In dit hoofdstuk is het versiebeheer van dit document te volgen (Tabel 1). De versies worden aangepast bij elke wijziging met de veranderingen van die update. Zo ontstaat een lijst van alle versies met de daarbij aangepaste punten.

Versie	Datum	Naam	Aanpassing
0.1.1	1 mei 2015	Albert Haaksema	Opzet document
0.1.2	4 mei 2015	Albert Haaksema	Aanvulling inleidende hoofdstukken
0.1.3	6 mei 2015	Albert Haaksema	Update aan de inleiding
0.1.4	8 mei 2015	Albert Haaksema	Aanvulling aanpak
0.1.5	15 mei 2015	Albert Haaksema	Invulling van Sprint 0 en de Ik vorm uit alle hoofdstukken verwijderd (met uitzondering van het voorwoord)
0.2.1	1 juni 2015	Albert Haaksema	Invulling van Sprint 1
0.2.2	5 juni 2015	Albert Haaksema	Verbeteren inleidende hoofdstukken, sprint 0 en sprint 1. Toevoeging Requirements Analyse en Bijlagen
0.3.1	15 juni 2015	Albert Haaksema	Verbeteren en toevoeging sprint 2
0.4.1	17 juni 2015	Albert Haaksema	Verder verbeteren sprint 2 en update bijlages
0.4.2	30 juni 2015	Albert Haaksema	Verbeteren lay-out verslag en verwijderen van onnodige informatie / onderdelen.
0.5.1	13 juli 2015	Albert Haaksema	Sprints 0 en 1 samengevoegd tot één sprint 0. Afkortingenlijst samengevoegd met de woordenlijst. Aanpassingen in Opdracht en Aanpak hoofdstukken.
0.5.2	14 juli 2015	Albert Haaksema	Hoofdstukken in de sprints vernieuwd en beter verdeeld.
0.6.1	22 juli 2015	Albert Haaksema	Verbeteren scriptie met feedback van de 60% afspraak
0.6.2	24 juli 2015	Albert Haaksema	Invulling sprint 4
0.7.1	6 augustus 2015	Albert Haaksema	Begin invullen sprint 5
0.7.2	7 augustus 2015	Albert Haaksema	Verbeteren sprint 4, aanvullen sprint 5, update woordenlijst
0.7.3	10 augustus 2015	Albert Haaksema	Nederlands verbeteren
0.8.1	21 augustus 2015	Albert Haaksema	Sprint 6 toevoegen
0.8.2	24 augustus 2015	Albert Haaksema	Bijlages toevoegen, Sprint 6 afronden, Nederlands Controleren
0.8.3	25 augustus 2015	Albert Haaksema	Woordenlijst updaten, Verwijzingen Controleren, Resultaten en Evaluatie uitgebreid.
0.8.4	26 augustus 2015	Albert Haaksema	Referaat en Voorwoord uitgebreid.
0.8.5	27 augustus 2015	Albert Haaksema	Requirements gesynchroniseerd vanuit het requirementsrapport.
0.9.1	3 september 2015	Albert Haaksema	Invulling sprint 7
0.9.2	7 september 2015	Albert Haaksema	Verbeteren sprint 7, woordenlijst en referentielijst updaten. Eindproducten en competenties uitgebreid.

Versie	Datum	Naam	Aanpassing
0.10.1	23 september 2015	Albert Haaksema	Verder verbeteren sprint 7, invullen sprint 8, begin invullen sprint 9.
0.10.2	24 september 2015	Albert Haaksema	Snippet SPI beter toegelicht, onderzoek sprint 0 onderscheid gemaakt tussen vooronderzoek en het werkelijke onderzoek. Paar hoofdstukken gewijzigd van naam en kern toegevoegd aan inleiding.
0.10.3	28 september 2015	Albert Haaksema	Verwijzingen goed gezet, Nederlandse controle, Woordenlijst geüpdatet, 80% feedback afgerond
1.0	28 september	Albert Haaksema	

Tabel 1: Versielijst Afstudeerverslag

2. Voorwoord

In het voorwoord leg ik kort uit voor wie het document bedoeld is en maak ik graag van de gelegenheid gebruik een groep mensen te bedanken.

Dit document is bedoeld voor personen met een voorkennis van analyseren, ontwerpen, bouwen en testen van software ontwikkelprojecten. Er komt een hoeveelheid van stof langs die moeilijk te begrijpen of te volgens is als er geen tot weinig kennis van deze onderdelen is. De onderdelen gerelateerd worden deels uitgelegd in de scriptie en anders in de bijlagen. Er zal verwezen worden naar de juiste sectie van de bijlagen als dit van sprake is.

Hier volgen allemaal personen die ik graag wil bedanken:

- Wesley Bosvelt van CGI, omdat hij mij gedurende het project begeleid en beoordeeld heeft. Dankzij hem heb ik bepaalde onderdelen, zoals de hardware opstellingen, kunnen maken. Dit was een vaardigheid waar ik geringe ervaring mee had opgedaan.
- Jeroen de Bekker van CGI, omdat hij mijn team lead was en mij geholpen heeft bij procesmatige vragen.
- Diana van der Kraan van CGI, omdat zei ronduit vriendelijk was en mij naar de juiste personen kon verwijzen of van spullen kon voorzien om mijn opdracht beter uit te voeren.
- Sander Kapsenberg van CGI, omdat hij mij geïntroduceerd heeft aan het bedrijf en uiteindelijk ook heeft aangenomen voor het afstuderen.
- Nanny Jacobs van Haagse Hogeschool, omdat zei mij goed advies heeft gegeven over de opdracht-aanpak. Daarnaast heeft ze goede feedback gegeven op mijn proces en hoe deze te documenteren.
- Tim Cocx van Haagse Hogeschool, omdat hij mij goede kritische feedback heeft gegeven over mijn documenten en opleveringen.
- Arno Nederend van Haagse Hogeschool, omdat hij mij regelmatig de juiste weg opstuurde tijdens mijn studie.
- Lucia Molenkamp van Thuis, omdat ze mij veel gesteund heeft in het bij elkaar halen van componenten en gesteund heeft bij moeilijke keuzes.
- Mart Ruijgt van Thuis, omdat hij mij geholpen heeft de hardware componenten bij elkaar te halen en in elkaar te zetten

3. Inhoudsopgave

4. Inleiding

Deze scriptie beschrijft alle onderdelen en keuzes die gemaakt zijn tijdens de afstudeeropdracht genaamd: Raspberry Pi Solar Challenge. De opdracht is gestart op 1 mei 2015 en eindigt op 5 oktober 2015. De opdracht is uitgevoerd bij CGI, een bedrijf leidend in informatietechnologie (IT) en zakelijke dienstverlening, gevestigd in Rotterdam.

Het doel van de Raspberry Pi Solar Challenge is uiteindelijk een workshop te realiseren waarin deelnemers een algoritme zullen maken om een zonnepaneel te doen bewegen. Hierbij zal de afstudeerder de omgeving ontwerpen en bouwen op zowel de hardware als software kant. De deelnemers van de workshop zullen met behulp van een API de hardware kunnen aansturen.

Deze API maakt het mogelijk om gelijk aan de slag te kunnen zonder enige configuraties of instelling klaar te moeten zetten. De deelnemers kunnen gebruik maken van een interface om een beter beeld te krijgen over onder andere de batterij stand en de energieopbrengst van de opstelling.

Om de algoritmes te testen en te beoordelen wordt een testsysteem met een scoresysteem meegeleverd. Deze zullen bewegingen van de zon ten opzichte van het zonnepaneel simuleren. Na een voor ingestelde cyclus zal het algoritme een score ontvangen die bepaald hoe efficiënt deze is.

De uitwerking start met een korte bedrijfsbeschrijving. Daarna volgt een toelichting van de opdracht, de aanpak hier van. Daarna is een uitleg van elke sprint uitgewerkt, inclusief de processen en taken die plaats hebben gevonden en de resultaten die de betreffende sprint heeft opgeleverd. Vervolgens wordt het eindresultaat van de opdracht beschreven en hoe CGI verder kan met het opgeleverde resultaat.

Na het eindresultaat volgt een eigen evaluatie, bestaande uit de ervaringen die zijn opgedaan tijdens het project, bijvoorbeeld wat goed en wat fout is gegaan, inclusief een toelichting van de verbetermogelijkheden. Het verslag sluit af met een woordenlijst en een aantal bijlagen die te vinden zijn in het tussenproducten lijst.

5. Samenvatting

Het referaat is een samenvatting van het gehele document. Hiermee kan in een vrij verkorte versie een overzicht verkregen worden over wat er allemaal behandelt wordt en wat de eindresultaten ervan zijn. Dit hoofdstuk zal geen uitgebreide uitleg hebben en alleen kort samenvatten het verloop van het project met de daarin kritische stappen die zijn voorgekomen.

In het begin was er begonnen met de opstart sprint, deze dient om het begin van het project voor te bereiden. In de opstartsprint komen bijvoorbeeld het plan van aanpak voor, de requirementsanalyse en een vooronderzoek. Het bedrijfsleven vergeleken bij het studentenleven was een grote switch en de eerste resultaten waren vandaar minder op orde.

De eerste paar sprints zijn veel beter gegaan. In deze sprints zijn voornamelijk de ontwerpen gemaakt voor zowel hardware als software. Ook is een kleiner onderzoek gedaan naar het opstellen van de ontwikkelomgeving en hoe deze geconfigureerd kan worden om te ontwikkelen en deployen naar de Raspberry Pi. Uiteindelijk is hier Eclipse voor gekozen op een virtuele Fedora machine. Fedora is een Linux operating systeem die gratis beschikbaar is. De software wordt met cross compiler en een toolchain gebuild en gedeployed via een SSH connectie vanuit Eclipse.

De hardwareopstelling was een proof of concept van hoe de hardware opstellingen er in de workshop uit zullen zien, deze is in latere sprints gedemonteerd, omdat de onderdelen hergebruikt moest worden in de testopstelling want er was maar 1 Raspberry Pi beschikbaar. Voor de testopstelling is wederom een onderzoek gedaan evenals de Analoge naar Digitaal converter die gebruikt wordt om het zonnepaneel af te lezen.

Het onderzoek voor de testopstelling was erg uitgebreid, omdat deze aan zeer strenge eisen moest voldaan zoals consistent en controleerbaar testen. Heel veel opties die goedkoop of makkelijk zijn vallen hierdoor weg en met een beperkt budget moest alles stevig onderbouwd worden. Om te garanderen dat de juiste opstelling gebouwd zou worden zijn er berekeningen gedaan voor prijzen, controleerbaarheid, de betrouwbaarheid en de consistentie. Hiervoor zijn berekeningen gedaan of aannames op basis van feiten van de componenten en waarvoor deze gemaakt zijn. Bijvoorbeeld servomotoren die gemaakt zijn om altijd dezelfde positie aan te nemen bij dezelfde commando's.

De testopstelling zal gebruikt worden om de algoritmes te testen gedurende de workshop. De eerdere hardware opstelling zal gebruikt worden als een opstelling waarop de deelnemers hun algoritme kunnen testen. De hardware opstelling zal alleen het deel zijn wat de algoritme moet aansturen zonder een uitgebreide opstelling om alle licht buiten te sluiten of te simuleren van bewegingen. Verwacht wordt dat deelnemers met een aansteker of zaklamp dit kan na bootsen om hun algoritme te verfijnen.

Het eindresultaat is de testopstelling met een hardware opstelling of twee die het beeld van de workshop geven. Daarnaast zijn handleidingen beschikbaar om alles te reproduceren en voor de deelnemers is een handleiding die beschrijft welke functies wat doen en hoe deze functies bereikt kunnen worden.

6. Bedrijf

6.1 Organisatie

CGI is van oorsprong een Canadees bedrijf, gericht op IT en zakelijke dienstverlening, en opgericht met de gedachte: “Een omgeving te maken waarin we gezamenlijk tevreden kunnen werken en als eigenaren contributie leveren om een bedrijf op te bouwen waar we trots op kunnen zijn”. CGI streeft na wereldwijd de leiding te houden in IT services en business processen. Daartoe investeert het bedrijf veel in innovaties; nieuwe oplossingen voor bestaande problemen.

6.2 Geschiedenis

CGI is gesticht door meneer Godin en Imbeau in 1976. Toen stond CGI voor “Consellers en gestion et informatique”. Dit vertaalt grofweg naar consultants in management en informatie technologie. Het bedrijf is gestart in een kelder met een klant, een telefoon en veel ambitie.

De eerste 10 jaar zijn er strategieën, modellen en principes vast gesteld die leidde tot groei in het bedrijf. Verder zijn er ook visies uitgekomen waarnaar het bedrijf en alle leden toe streven. Gedurende deze periode is CGI geëvolueerd van alleen consultancy naar consultancy en integratie van systemen.

Begin 1986 is outsourcing erg populair geworden bij bedrijven. CGI speelde hier op in en heeft een paar bedrijven gekregen voor outsourcing. CGI werd hierdoor ook bekend als een bedrijf die investeerde in outsourcing. Naast de consultancy en integratie van systemen, kwamen de end-to-end it services bij en IT outsourcing services. Als laatste ging CGI zich in een positie zetten om snel te reageren op de stijgende trend voor globalization.

In de periode van 1996-2006 is de IT industrie erg snel gegroeid. De groei strategie “build and buy” werd geprioriteerd om uit te voeren. Hierdoor kan het bedrijf groeien door organische groei maar ook door acquisities. Deze strategie wordt tot op deze dag nog steeds toegepast.

Op heden dag heeft CGI ruim 68.000 werknemers over de hele wereld verspreid. CGI is gevestigd in 40 landen met een sterke aanwezigheid in IT markten en consultancy. CGI streeft er om wereldwijd bekend te zijn bij haar klanten en leden als een leider in business process services en IT services.

6.3 Essentie van CGI

CGI bevat een constitutie die definieert wat de essentie van CGI is. Wie CGI is en hoe de cultuur van CGI werkt. Het is de DNA van CGI als het ware. De constitutie beschrijft de droom, visie, missie en de waarden waarop CGI gezamenlijk draait.

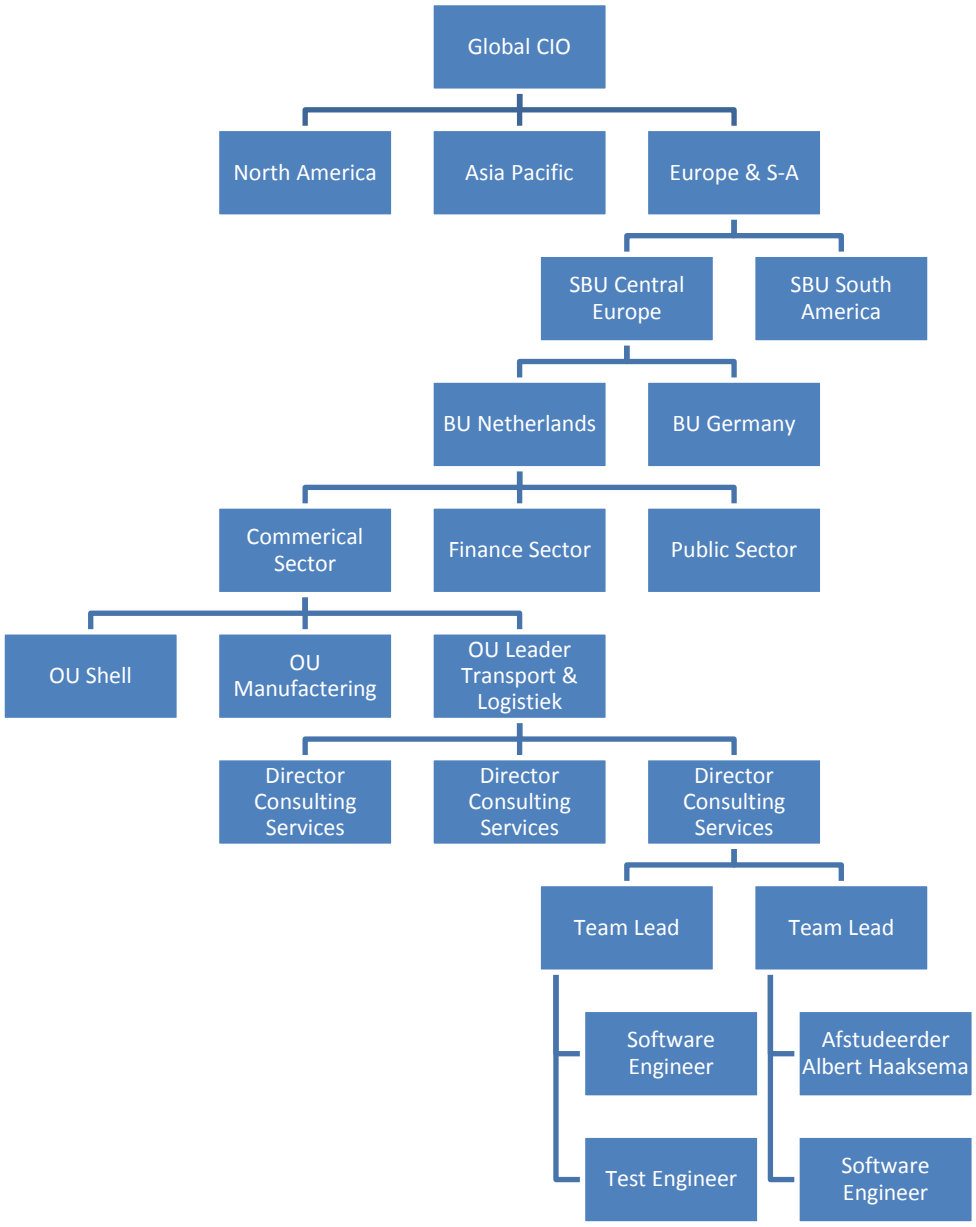
De droom van CGI is een omgeving te maken waarin er gezamenlijk als eigenaren contributie leveren aan het opbouwen van een bedrijf waar we trots op kunnen zijn. Deze droom is door de oprichters gedefinieerd en het is het principe waarop CGI draait. Alles in CGI wordt ook gezamenlijk gedaan en iedereen kan ook bij anderen vragen voor hulp.

De visie van CGI is een globale wereld klasse informatie technologie en een business process services leider te zijn waarbij de klanten geholpen worden om te slagen.

De missie waarmee de visie gerealiseerd wordt is de klanten te helpen met uitstekende kwaliteit, objectiviteit, competentie, beste services leveren en de klanten volledige tevreden maken bij opgeleverde oplossingen in informatie technologie en business processen en management. In alles wat CGI doet wordt een cultuur voor partnerschap, teamwork en integriteit opgebouwd.

6.4 Plek afstudeerder

De stagiair is in contact gekomen met het bedrijf door te solliciteren via de bedrijfswebsite. De opdrachtverstrekking was het resultaat. Figuur 1 schetst het organogram. De stagiair bevind zich in een team van de Operating Unit (OU) Transport en Logistiek. De stagiair zal zelfstandig als software developer werken aan de afstudeeropdracht.



Figuur 1: Organogram

6.5 Werkwijze bedrijf

Het bedrijf als geheel werkt met meerdere methodieken en aanpakwijzen. Dit verschilt niet alleen per afdeling, maar ook per project. De beschrijving in deze paragraaf focust zich met name op de werkwijze van de afdeling waar de student onder valt.

De afdeling Transport en Logistiek bevat veel projecten rondom ProRail, NS en Rijkswaterstaat. Het grootste deel van deze projecten gebruiken scrum als methodiek voor het project. Enkele project op de afdeling gebruikt de waterval methode. Elk project heeft zijn eigen indeling voor de methodiek, maar in grote lijnen werken ze allen hetzelfde.

Bijvoorbeeld elk project dat scrum gebruikt heeft een scrumboard met alle stories in de nabije omgeving. Verder gebruikt elk project een scrummaster, een product owner en een verzameling van leden die het team vormen. De leden zijn vaak verdeeld in test engineers en software engineers met aanvulling van rollen als architecten.

In de projecten wordt gebruik gemaakt van de milsim standaarden voor producten en documenten. Enkele voorbeelden zijn Software Requirements Specification, System-Subsystem Specification en het Software Design Description. Deze documenten bevatten een vaste layout en worden regelmatig gebruikt binnen in het bedrijf.

7. Opdracht

7.1 Probleemstelling

Het bedrijf CGI wil een workshop opzetten die zowel voor recruitment als interne doeleinden gebruikt zou kunnen worden. Deze workshop valt onder de naam Raspberry Pi Solar Challenge. In deze workshop moeten deelnemers met behulp van elektronica een algoritme ontwikkelen om een zonnepaneel te doen bewegen. Het zonnepaneel zit vast aan een satelliet die om de Aarde heen draait. De workshop moet deelnemers uitdagen om efficiënte algoritmes te schrijven om zoveel mogelijk energie opbrengsten te krijgen. Naast de inrichting en ontwikkeling van de workshop moet er ook een beoordelingssysteem komen. Het beoordelingssysteem zal de algoritmes beoordelen op efficiëntie.

7.2 Doelstelling

Het doel van de opdracht is een omgeving te realiseren waarbij CGI collega's en externen direct aan de slag kunnen met het ontwikkelen van algoritmes om de motoren zo efficiënt mogelijk te bewegen. Daarbij wordt een score en testsysteem meegeleverd die de algoritmes test en beoordeelt. Dit systeem zal zo consistent mogelijk moeten werken om betrouwbare vergelijkingen te kunnen maken tussen algoritmes.

7.3 Resultaat

Het resultaat van de opdracht is een omgeving waarbij deelnemers direct aan de slag kunnen met het ontwikkelen van hun algoritmes voor de Raspberry Pi Solar Challenge. Hierbij zullen de deelnemers gebruik maken van de API die mee geleverd wordt die alle onderdelen aan zal sturen. De API zal in C++ ontwikkeld worden, omdat deze lage taal gemakkelijk de hardware kan aanspreken. Aan het einde van de challenge zal een scoresysteem beoordelen welk team het meest efficiënte algoritme geschreven heeft. Het geheel kan gebruikt worden voor recruitmentdoeleinden of als uitdaging voor collega's.

7.4 Op te leveren producten

Gedurende dit project zullen meerdere producten worden opgeleverd. Hier volgt een lijst van deze producten:

- Plan van Aanpak: Het plan van aanpak beschrijft hoe het project aangepakt gaat worden en welke middelen daarvoor gebruikt gaan worden.

- Requirementsrapport: Het requirementsrapport beschrijft alle requirements ook wel eisen en wensen van de opdrachtgever en andere stakeholders.

- Onderzoeksrapport: Het onderzoeksrapport bevat het servomotor onderzoek, onderzoek naar ontwikkelen voor de Raspberry Pi en het onderzoek voor de testopstelling die plaatsvinden in dit project. Elk onderzoek is onderverdeeld in een hoofdvraag en deelvragen.

- Interface tussen Raspberry Pi en servomotor aansturing: De servomotoren gebruiken specifieke commando's om te bewegen. Deze commando's zijn niet veel sprekend en worden met behulp van een interface duidelijker gemaakt.

- API ten behoeve van de workshopdeelnemers: De deelnemers zullen gebruik maken van een API om alle functionaliteiten te gebruiken van het systeem. Meer hoeven ze ook niet te gebruiken.

- Testopstelling voor scoresysteem: De algoritmes zullen getest worden met een apart testsysteem. Het testsysteem moet consistent zijn om vergelijkbare resultaten op te leveren.

- Scoresysteem: De algoritmes worden beoordeeld door een scoresysteem die samen met het testsysteem een eindresultaat bepaalt.

-Handleiding: Voor dit project worden meerdere handleidingen opgeleverd. De deelnemers krijgen een uitwerking van de API documentatie met de functies die ze nodig hebben om hun de servomotoren aan te sturen. Er zal nog een API documentatie komen van het gehele product die voor CGI bestemd is om het eindresultaat te kunnen onderhouden of uitbreiden. CGI zal ook een handleiding ontvangen waarin beschreven staat hoe de ontwikkelomgeving geïnstalleerd en geconfigureerd moet worden om software / algoritmes voor de Raspberry Pi te kunnen maken. Het doel hiervan is dat er meerdere laptops klaar worden gezet voor de deelnemers zodat deze niets hoeven te configureren. Als laatste wordt een handleiding opgeleverd over de werking van de testopstelling in combinatie met het scoresysteem.

- Afstudeerdossier: Dit dossier die het gehele proces en keuzes zal beschrijven.

8. Aanpak

In dit hoofdstuk wordt de aanpak van het project beschreven. Een project wordt aangepakt met een ontwikkelmethodiek om duidelijke richtlijnen en deadlines te kunnen vaststellen. In de volgende subhoofdstukken kan gevonden worden welke methodieken op de tafel lagen om uit te kiezen en vervolgens wordt geschreven hoe de gekozen methodiek toegepast gaat worden voor dit project.

8.1 Mogelijke opties

Bij dit project waren de requirements van te voren bekend en welke werkzaamheden op zijn minst moesten plaatsvinden waren ook bekend. Het onbekende deel lag voornamelijk bij onderzoeken, ontwerpen en maken van de hardware, omdat dit een vrij onbekend gebied is voor de student. Het bekende deel lag bij de software kant en de daarbij behorende onderdelen zoals requirements opstellen en testen. Het gehele traject zou vrij lineair verlopen met vooronderzoek en requirements als eerste gevolgd met ontwerpen, bouwen en testen. Tussendoor zouden eventueel wijzigingen kunnen plaatsvinden in verband met moeilijkheden bij bijvoorbeeld de hardware onderdelen. Een bonus was de mogelijkheid om meerdere talen te gebruiken bij het ontwikkelen van de algoritmes door de deelnemers. Met deze informatie werd gedacht aan een Waterval methode, een RUP aanpak of Scrum.

Waterval is een stapsgewijze methode die alles afmaakt van de eerste stap voordat er door wordt gegaan naar de tweede. Waterval is moeilijk aan te sturen en er kan weinig veranderen tussendoor, omdat het de hele planning in de war kan brengen. Dit project had een deel van de requirements al bekend.

RUP is een iteratieve methode die een project verdeeld in 4 fasen. Deze 4 fasen richten zich elk op onderdelen als analyseren of bouwen, maar elk onderdeel komt voor in elke fase. Het idee is dat aan het einde van de eerste fase ongeveer 80% van alle analyses en requirements af zijn. In de volgende fase moet ongeveer 80% van alle modellen af zijn en de analyse en requirments moeten af zijn. Dit herhaalt zich tot en met de laatste fase waarin voornamelijk deployment zit. Dit project had een paar grote risico's en het was bekend dat de verschillende systemen in volgorde gemaakt zouden worden. Hierdoor zou RUP niet ideaal zijn in een project waarin meerdere systemen opvolgend gemaakt gaan worden.

Scrum is ook een iteratieve methode, maar is veel meer flexibel. Scrum werkt met behulp van sprints op regelmatig opleveringen te doen. Per oplevering wordt feedback gegeven waardoor het goed aangestuurd kan worden. Elke sprint richt zich op een onderdeel en probeert deze af te krijgen voor de oplevering, zodat een werkende versie klaar staat met de nieuwe functionaliteiten. Het nadeel aan Scrum is dat de productowner of eventueel andere stakeholders voldoende tijd beschikbaar moeten hebben om het scrumteam aan te sturen. Voor dit project is Scrum erg voordelig, omdat de bedrijfsbegeleider regelmatig feedback kan geven. Op deze manier kan er regelmatig bijgestuurd worden bij problemen, tevens komen problemen ook eerder naar voren waardoor sneller ingegrepen kan worden. Dit geheel heeft als resultaat dat risico's als de complexiteit van de opdracht een stuk minder impact krijgen.

Uiteindelijk is gekozen om het project op een waterval manier op te bouwen met een Scrum implementatie. Het lineaire proces van waterval wordt aangehouden, maar het wordt met sprints ingevuld zodat de mogelijkheid voor bijsturing veel actiever en groter is.

8.2 Inrichting methodiek

Voor dit project is gekozen voor een Waterval methode met een Scrum implementatie. Hierbij is het lineaire proces van waterval aangehouden, maar is deze aangevuld met onderdelen van Scrum zoals sprints om de mogelijkheid voor flexibiliteit en bijsturing hoog te houden. Tevens worden enkele risico's zoals project falen verlaagd, omdat er regelmatig bijsturing kan plaatsvinden vanuit de product owner.

De stagiair werkt zelfstandig zoals hiervoor genoemd en zal het proces vullen en gebruiken om het project uit te voeren. De produc t owner wordt door de bedrijfsbegeleider uitgevoerd. De stagiair is zelf verantwoordelijk voor de sprintindeling en het behalen van de resultaten in de sprint. Bij complicaties of tijdsgebrek wordt met de product owner overlegd welke stories af moeten zijn voor de sprint en welke stories achterwege gelaten kunnen worden voor opvolgende sprints.

De sprints zijn twee weken lang en de stories bij elkaar zullen 72 uur bevatten, inclusief tijd voor het updaten van de scriptie. De andere 8 uur is voor het indelen van de sprints, het voorbereiden en uitvoeren van de sprintdemo's en de sprintwissel. Bij sprintdemo's wordt rond de tafel gezeten met de product owner ook wel de bedrijfsgbegeleider. De stagiair bespreekt met hem hoe de sprint verlopen is en welke resultaten behaald zijn. Daarbij wordt ook besproken welke resultaten niet behaald zijn en waar moeilijkheden lagen. Uiteindelijk zal besproken worden wat in de volgende sprint moet gebeuren. Na afloop zal de sprint ingedeeld worden met aanvulling van andere stories als er nog uren openstaan.

De allereerste sprint is de opstartsprint. Deze wordt ook sprint 0 genoemd. Sprint 0 is vaak wat langer, omdat het onderzoek daarin plaatsvindt. Het onderzoek neemt vaak meer tijd in beslag dan 1 sprint toestaat. Naast het onderzoek vindt het maken van het plan van aanpak en de requirements-validatie en analyse plaats in sprint 0. Sprint 0 voor dit project zal 4 weken lang zijn en dat is 2 sprints bij elkaar. Halverwege sprint 0 zal een demo plaatsvinden om de progressie te bespreken met de bedrijfsbegeleider.

9. Sprint 0: Opstartsprint

Dit hoofdstuk beschrijft het proces van sprint 0. Het proces houdt in alle activiteiten in en de keuzes die gemaakt zijn met bijbehorende toelichting. Sprint 0 is de opstart sprint en het doel van de sprint is vooral inwerken en onderzoeken. Tijdens deze sprint is gewerkt aan het plan van aanpak, requirementsanalyse en het onderzoek naar servomotoren. Daarnaast is het ook een inwerkperiode met de Raspberry Pi. De Raspberry Pi is voor de stagiair een nieuw onderdeel en speelt een grote rol in het project. Daarom is twee dagen ingewerkt om te begrijpen hoe het functioneert. Deze onderdelen passeren de revue in de volgende hoofdstukken. Vervolgens vindt een beschrijving van het resultaat plaats met de daarbij gekregen feedback bij de sprint demo.

9.1 Plan van aanpak

Het plan van aanpak beschrijft hoe het project aangepakt gaat worden. Bij dit project wordt gebruik gemaakt van Scrum. Hoe Scrum toegepast gaat worden op dit project wordt verwezen naar het hoofdstuk Aanpak. Naast de ontwikkelmethodiek zijn er ook andere aspecten in een plan van aanpak zoals de risicoanalyse en een productbacklog.

De risicoanalyse beschrijft de risico´s die zich kunnen voordoen in het project. Hiermee kan een eigenaar van het product keuzes maken of het de investering waard is of dat bepaalde punten meer aandacht moeten krijgen tijdens de productontwikkeling. De risicoanalyse is een SWOT analyse om alle sterktes, zwaktes, kansen en bedreigingen weer te geven. Gekozen is voor de SWOT analyse, omdat deze een duidelijk overzicht van de risico's geeft, maar ook hoe die in verhouding staan met de positieve punten. Uit de analyse komt naar voren dat de risico's niet zeer groot zijn en dat er een aantal goede kansen zich bevinden in dit project. In Tabel 2 volgen een paar risico's uit dit project.

Nummer	Risico	Impact / Gevolg	Prioriteit	Oplossing
1	Onbekend met elektronica	Hardware ontwerpen en bouwen zal veel langer duren	Gemiddeld	Begeleider / Collega's om hulp vragen bij ingewikkelde problemen.
2	Uitgebreide opdracht: Software ontwikkelen, bouwen, testen Hardware ontwikkelen, bouwen Testopstelling ontwerpen, bouwen Onderzoek	Onderdelen kunnen incompleet zijn bij de eindoplevering -> Tijdsdruk	Hoog	Goede aanpak kiezen om zoveel mogelijk feedback en begeleiding te kunnen krijgen. Daarbij goed plannen en houden aan deadlines
3	Onbekend met Raspberry Pi	Veel besluiten betreft de hardware hebben meer tijd nodig	Laag	Inwerken Raspberry Pi voor Onderzoeksresultaten definitief zijn.

Tabel 2: Risico's in het project

Naast de risicoanalyse is gewerkt aan de product backlog. Deze staat uitgewerkt in het plan van aanpak en ook online, zodat de product owner alle prioriteiten kan wijzigen. Daarnaast kan de product owner stories toevoegen, aanpassen of verwijderen als dat nodig is. Dit was voorheen niet mogelijk, maar wel gewenst.

De laatste stap om het proces van Scrum compleet te maken, is naast de sprint demo´s ook standups organiseren en een burndown chart bijhouden. Zo is duidelijk te zien hoeveel werk gedaan is en hoeveel werk resteert. In sprint 0 zijn eerste versies in gebruik genomen. Vanaf sprint 1 moeten de versies up to date gehouden worden en een goed overzicht beschikbaar zijn van het werk dat afgerond is en het werk dat nog open staat.

9.2 Requirementsanalyse

Tijdens de requirementsanalyse worden alle requirements verzameld en gedocumenteerd. Om de requirements te prioriteren is gebruik gemaakt van MoSCoW. Om ze te prioriteren is samen met de product owner overlegt welke stories belangrijker waren dan andere. Op deze manier is de lijst opgebouwd met een lijst van prioriteiten. Een deel van de requirements was van te voren bekend, zoals de voorwaarde dat de deelnemers direct aan de slag moeten kunnen bij de opstart van de workshop.

Een deel van de requirements is gehaald uit de opdrachtomschrijving, maar deze moesten eerst gecontroleerd worden of deze goed gedocumenteerd waren en of ze compleet waren. Om de requirements aan te vullen is gebruik gemaakt van interviews met de opdrachtgever en de product owner om de requirements aan te vullen. De interviews hebben ervoor gezorgd dat de onduidelijkheden werden weggewerkt. Met een complete lijst van requirements kan goed gecontroleerd worden of alles voldoet aan de eisen en wensen van de eindgebruikers.

Requirements worden op een specifieke manier geformuleerd om duidelijk weer te geven wat de eis is en wie die eis stelt. Deze manier is bekend als: Als (Rol) wil ik graag (Actie) kunnen uitvoeren. Enkele voorbeelden van requirements in dit project zijn in Tabel 3 te zien.

Nummer	Requirement	Prioriteit
F2	Als workshopgever wil ik dat de testopstelling de algoritmes consistent getest kunnen worden.	Must have
F9	Als software developer wil ik dat de algoritmes blackbox met het systeem communiceert	Should have
NF15	Als workshopgever wil ik dat de hardware opstelling onder de 5kg weegt.	Could have

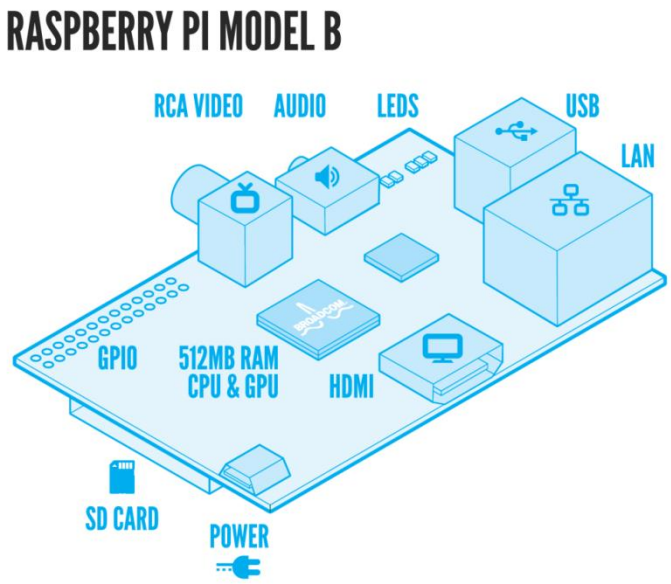
Tabel 3: Paar requirements uit het requirementsrapport

De requirements zijn onderverdeeld in functionele en niet functionele requirements. Functionele requirements beschrijven hoe het eindproduct zich moet gaan gedragen en niet functionele requirements beschrijven criteria waaraan het systeem moet voldoen, bijvoorbeeld hoe snel het moet zijn of hoeveel gebruikers het moet aankunnen.

Deze requirements zijn verder onderverdeeld per product van het project. Zo zijn alle requirements in een mapping gekomen naar de producten waarin de requirement afgevangen wordt. Als een requirement afgevangen is dan voldoet het product aan de eis. Alle requirements staan in tussenproducten lijst hoofdstuk 4.

9.3 Inwerken Raspberry Pi

De Raspberry Pi is een mini computer met educatie als doeleinde. De Raspberry Pi (Figuur 2) is een goedkoop alternatief op een normale computer waarmee iedereen kan leren programmeren. Het is ook in staat om bijvoorbeeld LED lampen of motoren aan te sturen met de general pins for input and output ook wel GPIO. Gedurende de studieperiode heeft de onderzoeker geen ervaringen opgedaan met de Raspberry Pi. Hierdoor heeft de student twee dagen de tijd genomen om in te werken met de Raspberry Pi. Hierbij valt te denken aan het lezen van de websites tot het installeren en remote aansluiten van de Raspberry Pi.



Figuur 2: Raspberry Pi met een abstracte weergave van alle hardware componenten

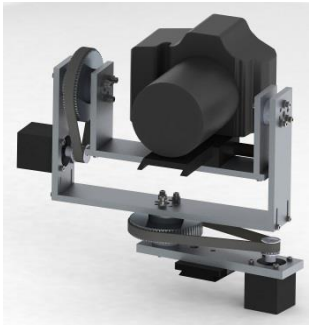
De Raspberry Pi bevat alle hardware om een draaiende computer te maken, zonder een bijgeleverd muis, keyboard of scherm. Door USB aansluitingen vallen muis en keyboard aan te sluiten. Het scherm kan met een HDMI kabel verbonden worden. Het geheugen en operating systeem staan op een SD kaart en niet op een vaste drive op de Raspberry Pi. De Raspberry Pi heeft ook een ethernet verbinding voor het aansluiten op internet. Een draadloze aansluiting is niet ingebouwd, maar het is wel mogelijk om draadloze Wi-Fi te krijgen door een USB Wi-Fi connector aan te sluiten. Door al deze losse onderdelen is de Raspberry Pi zelf erg licht en makkelijk op bijvoorbeeld een robot aan te sluiten. Om alsnog de Raspberry te kunnen gebruiken kan bijvoorbeeld door Virtual Networking Computing (VNC), Secure Shell (SSH) of andere manieren van Remote Acces een connectie gemaakt worden.

Het configureren van de Raspberry Pi om met remote acces te werken was vrij ingewikkeld, omdat niet duidelijk was waar de ontwikkelomgeving uiteindelijk op zou komen. Er is wel VNC geïnstalleerd zodat remote bij de Raspberry Pi kon zonder dat een extra scherm, muis en keyboard nodig zijn. Als laatste was het wennen aan de nieuwe omgeving, omdat de student weinig werkt met Linux. Het doel van deze sprint was dan ook uitzoeken hoe de Raspberry Pi werkt en nog niet direct de ontwikkelomgeving klaar zetten.

9.4 Onderzoek Servomotoren

Voor de opdracht is een hardware opstelling nodig die gebruik maakt van onder meer servomotoren en een zonnepaneel. De servomotoren moeten bevestigd worden aan het zonnepaneel om het zonnepaneel naar de zon te kunnen draaien. Doel van het onderzoek is het vinden van en adviseren over de servomotoren die nodig zijn om het zonnepaneel te kunnen draaien en het gewicht moeten kunnen dragen tijdens het stilstaan. Het is de bedoeling dat het zonnepaneel naar de zon blijft kijken en niet naar de grond zakt.

Om het zonnepaneel naar de zon te blijven richten en maximaal zon kan ontvangen, moet het niet alleen kunnen draaien, maar ook kunnen kantelen. Om te kunnen draaien en te kantelen, is een hardware opstelling nodig. Deze hardware opstellingen heten pan en tilt opstellingen. Pan en tilt opstellingen zijn vrij bekend in het gebruik van camera's, omdat met een pan en tilt set de camera heel vloeiend kan draaien en kantelen om een specifiek punt te volgen (Figuur 3). Voor de opdracht is de keuze op een pan en tilt set gevallen om het zonnepaneel te draaien en te kantelen.



Figuur 3: Abstracte weergave van een pan en tilt systeem met camera

Om de resultaten van het onderzoek te achterhalen wordt een probleemstelling geformuleerd met daarbij een doelstelling. Vervolgens wordt de hoofdvraag en de deelvragen gemaakt. De probleemstelling voor dit onderzoek is: Met welke servomotoren kan er op goedkope wijze een zonnepaneel bewogen worden met behulp van een pan en tilt systeem en hoe kunnen deze motoren geen elektronische schade leveren aan de Raspberry Pi.

Het doel van dit onderzoek is een selectie van servomotoren kiezen die goedkoop (in euro totaal) zijn, genoeg kracht bevatten om het zonnepaneel te bewegen en te dragen en geen kortsluitingen veroorzaakt bij de Raspberry Pi.

Om een goede hoofdvraag en deelvragen te formuleren is er een vooronderzoek nodig, omdat de student vrij weinig kennis heeft van servomotoren.

9.4.1 Vooronderzoek

Om dit onderzoek goed te kunnen doen is kennis nodig van het onderwerp namelijk servomotoren. Na het vooronderzoek zal het werkelijke onderzoek plaatsvinden waarmee de hoofdvraag en deelvragen beantwoord zullen worden.

Om voldoende kennis te krijgen voor het onderzoek spelen de volgende vragen een rol:
- Wat voor type servomotoren zijn er?
- Hoe werken servomotoren?

Servomotoren bevatten verschillende types. Er zijn kernloze, borstelloze en standaard motoren. Servomotoren bevatten een elektrische motor die met behulp van elektriciteit een klein magnetisch veld maken. Dit stuurt de motor aan en zorgt voor een roterende beweging. De standaard servomotoren zijn zoals de naam stelt het standaard model zonder enige bijzonderheden of eigenaardigheden.

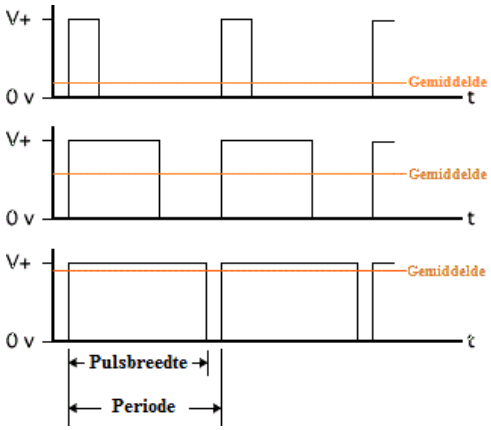
De borstelloze motoren bevatten geen borstels aan de uiteindes die de stroom doorgeven aan de roterende kern om de magnetische velden te geven. Deze zijn een stuk duurder, maar zijn vaak sneller omdat ze minder weerstand hebben. Tevens is de levensduur ook een stukje langer, omdat er minder slijtage kan plaatsvinden tussen de onderdelen.

De kernloze servomotor bevat geen kern. De motor werkt op hetzelfde principe als de standaard motor, maar in plaats daarvan is er een draadwerk van draden die rondom de magneten zitten. Het is veel lichter en kan daardoor sneller accelereren en afremmen.

Onder de type servomotoren is er ook een onderscheid in de aansluitingen naar de buitenwereld toe. Veel servomotoren bevatten plastic tandwielen waarop onderdelen zoals wielen of armen aangesloten worden. Deze zijn ideaal voor miniatuur werken als radio grafische auto's of kleine applicaties. Voor doeleinden die iets meer kracht nodig hebben zoals vliegtuig aerobatiek zijn er servomotoren met metalen tandwielen. Deze zijn sterker en zijn meer geschikt voor zwaardere taken.

Als laatste zijn er ook verschillende formaten servomotoren. De meest gebruikte formaten zijn standaard en micro servomotoren. Naast de grote is het verschil voornamelijk in hoeveel kracht ze kunnen leveren. De standaard formaat servomotoren zijn bedoeld voor applicaties vrij belangrijk is en micro servomotoren zijn met name bedoelt voor het fijn werk zoals een wissel schuiven op een miniatuur treinbaan.

Servomotoren hebben 3 aansluitingen: Voeding, Aarding en Controle signaal. De voeding en aarding zijn vanzelfsprekend waar de servomotor zijn stroom vandaan haalt, maar het controle signaal speelt een belangrijke rol voor de servomotor. Het controle signaal geeft aan in welke positie de servomotor moet staan. Servomotor werken met behulp van pulse width modulation (PWM). PWM is een herhalend signaal die een bepaalde up en downtime heeft om een soort golf effect te krijgen.



Figuur 4: Voorbeeld pulse width modulation

Hier naast in Figuur 4 een voorbeeld van een PWM signaal. Het signaal herhaalt zich over een periode en de tijd dat het signaal aan staat is de puls breedte of duty cycle. De duty cycle is weer gegeven in procenten om aan te geven hoeveel tijd van de periode het signaal hoog staat ten opzichte van de periode. Servomotoren kunnen 180 graden draaien, maar dit verschilt per servomotor. Hierdoor is een afspraak gemaakt dat elke servomotor op de neutrale positie (90 graden) staat als het pwm signaal 1.5ms aan staat over een periode van 20ms. De meeste servo's hebben een 0 graden stand op 1.0ms over een periode van 20ms en een 180 graden stand bij 2.0ms over 20ms. Het pwm signaal hoeft niet 20ms te zijn, want het draait om de duty cycle. De 180 graden stand heeft een duty cycle van 10% bijvoorbeeld. Bij een periode van 100ms en een pulsbreedte van 10ms zou de servo nog steeds op 180 graden staan. Zolang het signaal actief is zal de servomotor constant proberen op die positie te blijven tot een nieuw signaal gegeven wordt of de motor uitgezet wordt.

9.4.2 Onderzoek

De hoofdvraag bij dit onderzoek luidt als volgt: Welke servomotoren kunnen met behulp van een pan en tilt systeem een 300 gram zonnepaneel bewegen zonder schade te leveren aan de Raspberry Pi?

Uit deze hoofdvraag kunnen de volgende deelvragen opgenomen worden:
Hoe kunnen servomotoren schade veroorzaken aan de Raspberry Pi?
Hoe kunnen servomotoren aangesloten worden op de Raspberry Pi?
Wat voor externe hardware is er beschikbaar om de connectie tussen Raspberry Pi en servomotoren te beveiligen?
Welke servomotoren zijn het meest geschikt voor de benodigde opstelling?

Om de resultaten te krijgen op deze vragen is er geïnterviewd met een paar hardware specialisten op de werkvloer. Daarnaast is er ook veel literatuur onderzoek gedaan met name op het internet. De collega's heb ik gevraagd over het aansluiten van apparatuur op de Raspberry Pi en gevraagd wat voor invloed hogere voltages kunnen hebben op apparatuur die het niet aankunnen.

Voor het literatuur onderzoek is voornamelijk met behulp van de zoekmachine google gebruik gemaakt om diverse informatie te vinden over Servomotoren en uitbreidingsborden voor de Raspberry Pi. Zoektermen bevatten dan ook worden als 'Servomotor', 'Raspberry Pi', 'Expansion boards for Raspberry Pi' en 'Servo control with Raspberry Pi'.

De General Purpose Input and Output(GPIO) pinnen van de Raspberry Pi zijn erg gevoelig en niet beveiligd. Om ervoor te zorgen dat geen kortsluitingen ontstaan die schade toebrengen aan de Raspberry Pi, wordt onderzoek gedaan naar de mogelijkheden om een servomotor veilig aan te sluiten. Het onderzoek is nodig om te garanderen dat de beste mogelijkheid gekozen wordt om de motor veilig aan te sluiten op de Raspberry Pi. Hierbij moet het risico dat de Rasberry Pi onnodige schade krijgt zoveel mogelijk weggedrongen worden.

De Raspberry GPIO zal gebruikt worden om de servomotoren aan te sturen, maar om dit te kunnen doen moet de werking ervan bekend zijn. De Raspberry Pi maakt gebruik van 3,3 Volt logica: bij een HIGH signaal staat 3,3 Volt op het draad/pin en bij een LOW signaal staat er 0 Volt op het draad/pin. Servomotoren gebruiken vaak voltages van 5 V en als deze aangesloten worden op de GPIO van de Raspberry Pi kan kortsluiting ontstaan. In het ergste geval zal de Raspberry Pi blijvende schade oplopen. De Raspberry Pi heeft geen beveiliging hiertegen. Kortsluiting moet daarom zo veel mogelijk vermeden worden door servomotoren veilig aan te sluiten op de Raspberry Pi.

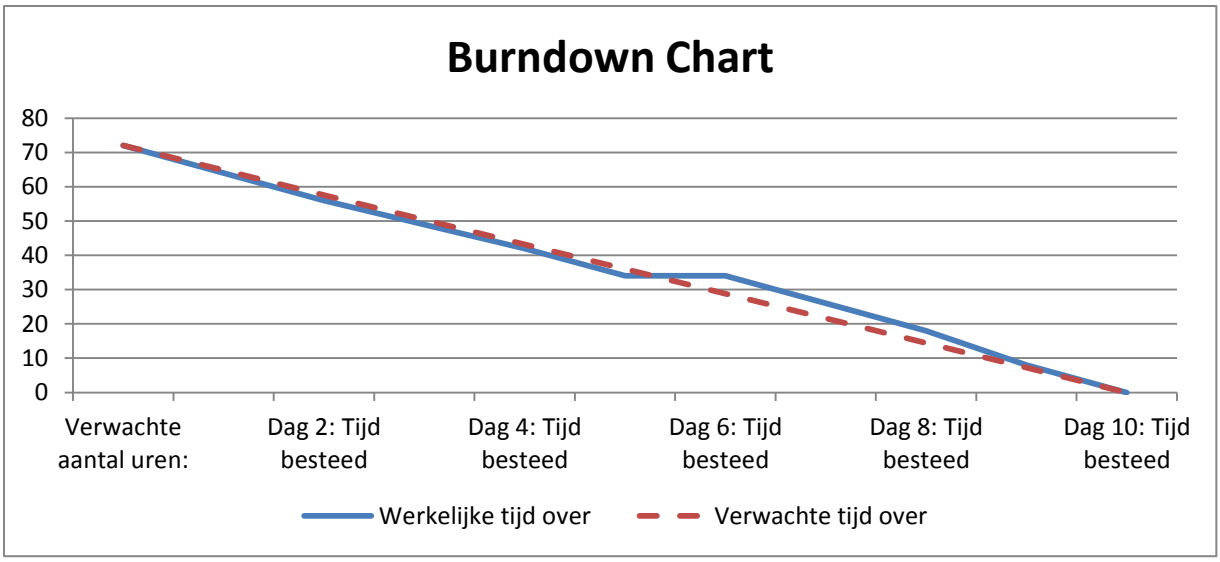
Servomotoren kunnen op twee manieren aangesloten worden op de Raspberry Pi: direct en indirect. Direct is veel goedkoper, maar gevaarlijker en indirect is veiliger, maar duurder. In het onderzoek is gekeken naar de directe aansluiting en twee manieren van indirect aansluiten. Beide indirecte manieren gebruiken een PCB om de pinnen veilig aan te sluiten. De eerste indirecte manier is alleen een schild gebruiken om de pinnen te beveiligen en meer niet. De andere manier is een PCB die de hele aansturing van de servomotoren verzorgd (Figuur 5is een voorbeeld daarvan). Er is niet naar meer oplossingen gezocht, omdat deze te uitgebreid zijn of geen mogelijkheid bieden om de servomotoren aan te sturen.

Figuur 5: Servo Pi aangesloten op de Raspberry Pi

De resultaten zijn uitgewerkt in een tabel en met een adviesrapport is er besproken welke servomotoren het beste waren en hoe deze het veiligst aangesloten konden worden, De servomotoren die de beste potentie hadden waren de servomotoren die meegeleverd werden bij de Adafruit pan en tilt kit. De servomotoren zijn Tower Pro SG90 microservomotoren. Deze zijn erg goedkoop en leveren ongeveer evenveel kracht als standaard formaat servomotoren. De pan en tilt kit is vrij klein en heeft wel modificatie werk nodig om een zonnepaneel er op te kunnen monteren. Om de servomotoren veilig aan te sluiten is gekozen om de Servo Pi aan te schaffen. De Servo Pi is een uitbreidingsbord voor de Raspberry Pi. De Servo Pi is speciaal ontworpen om servomotoren aan te sturen en kan direct op de Raspberry Pi aangesloten worden. Alle resultaten staan met meer detail in de tussenproductenlijst onder hoofdstuk 5.

9.5 Resultaat en Feedback

Het resultaat van sprint 0 is de opstart van het project. Het doel was om de documentatie als plan van aanpak, requirements analyse en het onderzoek af te ronden. Het enige dat niet voltooid is, is het aanmaken van de repository en het updaten van de scriptie. Tijdens deze sprint zijn de onderzoeken afgerond, zijn de requirements afgerond en geprioriteerd, de Raspberry Pi is klaar gemaakt voor het eerste gebruik en als laatste is het Plan van Aanpak afgerond. Het verloop van deze sprint is in de burndown chart in Figuur 6 te zien. De gegevens kloppen niet helemaal, omdat de burndown chart op ongeveer 60% van de sprint werd aangemaakt. Hierdoor is het niet duidelijk wanneer het werk precies af was en hoeveel werk er open ligt op welk moment.



Figuur 6: Burndown chart sprint 0

Gedurende de sprintdemo is het onderzoek en de resultaten het meest besproken onderdeel. Het plan van aanpak en de requirements zijn niet vaak onderwerp van gesprek geweest. Het onderzoek heeft veel tijd in beslag genomen, omdat de kennis van de stagiair over elektronica vrij beperkt was, voordat de stagiair aan de opdracht begon. Zodoende heeft de stagiair veel informatie moeten verkrijgen om goede vergelijkingen te kunnen maken in het onderzoek.

Uiteindelijk zijn knopen doorgehakt en zijn beide partijen het eens over welke aankopen gedaan moeten worden. Gekozen is de pan en tilt kit van Adafruit aan te schaffen, omdat deze goedkoop is en over voldoende kracht beschikt om de zonnepanelen te dragen en sturen. Daarnaast is de Servo Pi aangeschaft omdat deze ook goedkoop is, genoeg aansluitingen heeft en veiligheid biedt voor de Raspberry Pi GPIO pinnen. De onderzoeksresultaten relevant bij deze onderzoeken staan in Bijlage 2 voor de servomotoren met pan en tilt en in Bijlage 3 voor het veilig aansluiten van de servomotoren.

9.6 Afwijking van het afstudeerplan

Deze sprint heeft weinig afgeweken van het afstudeerplan. Volgens het afstudeerplan zouden er 4 weken beschikbaar zijn voor sprint 0 om het plan van aanpak, het onderzoek en de requirementsanalyse uit te voeren. Deze zijn op tijd afgerond en er kan volgens de planning verder gewerkt worden.

10. Sprint 1

Sprint 1 sluit de onderdelen van sprint 0 af en gaat verder met ontwerpen van de interfaces. In sprint 1 zijn de volgende onderdelen en taken uitgevoerd: De hardware onderdelen zijn besteld. Er is uitgezocht hoe de ontwikkelomgeving eruit moet zien om software werkend op de Raspberry Pi te krijgen. Zijn de interfaces ontworpen om een beeld te scheppen over hoe het geheel eruit zal gaan zien. Is de workbreakdown structure gemaakt. Er is een aansluitschema gemaakt en als laatste is de repository aangemaakt.

Voor de bovengenoemde taken worden een aantal requirements afgevangen. De requirements beschrijven wensen of eisen waaraan het product moet voldoen. In Tabel 4 staan een paar van de requirements die afgevangen worden in deze sprint.

Nummer	Requirement	Prioriteit
NF5	Als workshopgever wil ik dat er een duidelijke handleiding is die beschrijft hoe gebruik gemaakt kan worden van de op te leveren interface	Must have
F8	Als software developer wil ik dat alle objecten / klassen uitbreidbaar zijn om toekomstig nieuwe klassen toe te voegen.	Should have
F9	Als software developer wil ik dat de algoritmes blackbox met het systeem communiceert	Should have
F11	Als workshopgever wil ik dat er een virtuele batterij gebruikt wordt om een werkelijke batterij te simuleren	Should have
NF11	Als software developer wil ik dat er design patterns toegepast worden om de software uitbreidbaar te maken.	Should have
NF12	Als software developer wil ik dat elke klasse een duidelijke rol heeft	Should have

Tabel 4: Requirements voor Sprint 1

10.1 Bestellen hardware

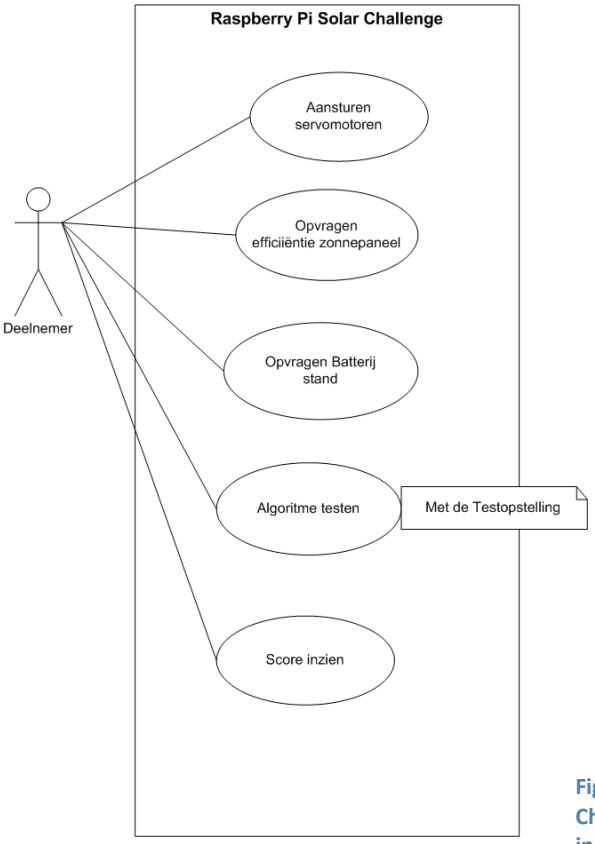
In sprint 0 zijn de onderzoeken naar de hardware afgerond. Tijdens deze sprint zijn de onderzoeksresultaten benut en is de hardware aangeschaft. In overleg met de teamlead, die het budget beheerst, is een keuze gemaakt van de hardware: de pan en tilt mini servo kit en de ServoPi zijn aangeschaft. De pan en tilt servo kit van Adafruit beweegt het zonnepaneel naar de zon toe. De ServoPi stuurt de servomotoren van de pan en tilt kit aan, zodat de Raspberry Pi GPIO pinnen veilig zijn. Tevens heeft de ServoPi een stabieler PWM signaal, waardoor de motoren geen onverwachte bewegingen maken. In totaliteit kosten de producten ongeveer 40 euro, exclusief verzendkosten.

10.2 Ontwerpen

Voor de software zijn ontwerpen nodig om een beeld te scheppen hoe het product er uit moet zien en hoe het zich moet gedragen. Hiervoor wordt Unified Modeling Language (UML) gebruikt. UML gebruikt meerdere modellen om vanuit verschillende perspectieven een beeld te krijgen van het gehele product. De modellen die in dit project gebruikt worden zijn: Use case diagrammen, klassendiagrammen, sequencediagrammen en activitydiagrammen. Een state diagram wordt niet gemaakt, omdat het systeem twee states heeft waar tussen geen transities bevinden gedurende het draaien van het programma.

Use case diagrammen

Use case diagrammen beschrijven hoe actoren interactie hebben met het systeem, bijvoorbeeld een klant die moet kunnen betalen bij een winkelsysteem. De use case diagram voor de Raspberry Pi Solar Challenge staat in Figuur 7.



Een use case diagram geeft de interactie weer tussen actoren en systeem, maar het geeft niet weer hoe en wanneer dat gebeurt. Hiervoor worden use case scenario's gebruikt. Een use case scenario beschrijft een scenario die plaats kan vinden bij de interactie tussen actor een use case.

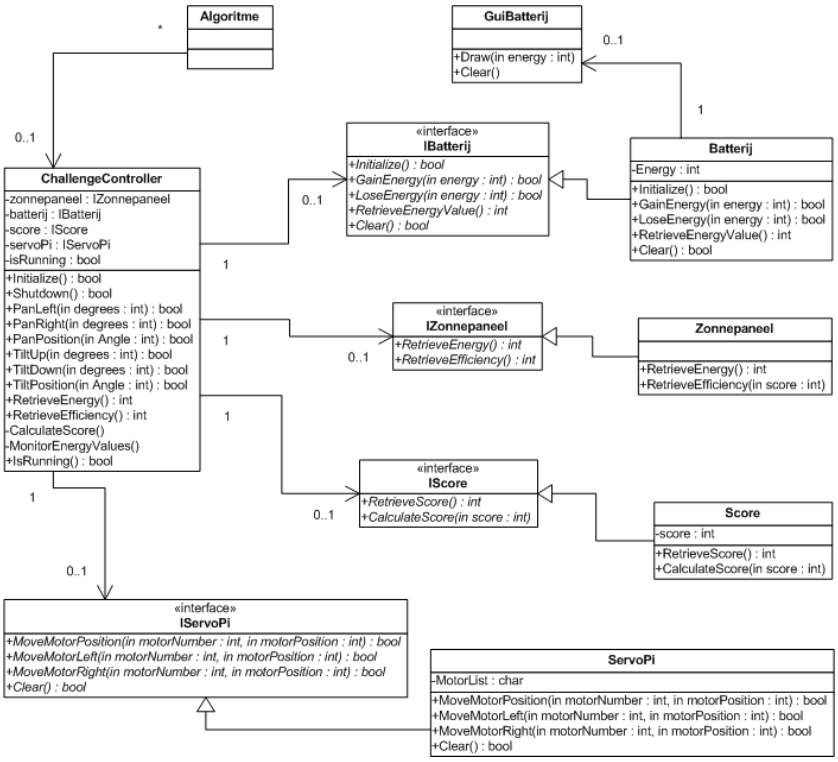
Een voorbeeld voor een scenario voor Aansturen servomotoren zou zijn. Deelnemer voert een commando in om de ServoMotor te draaien. Het systeem draait de ServoMotor op basis van het commando. Deelnemer ziet de motor in zijn nieuwe stand staan.

De use case diagram bevindt zich in de tussen producten lijst onder hoofdstuk 6.1

Figuur 7: Use case diagram Raspberry Pi Solar Challenge. (Let op: Deze use case diagram is in sprint 1 gemaakt, de use case diagram in de bijlage is de meest recente versie)

Klassendiagrammen

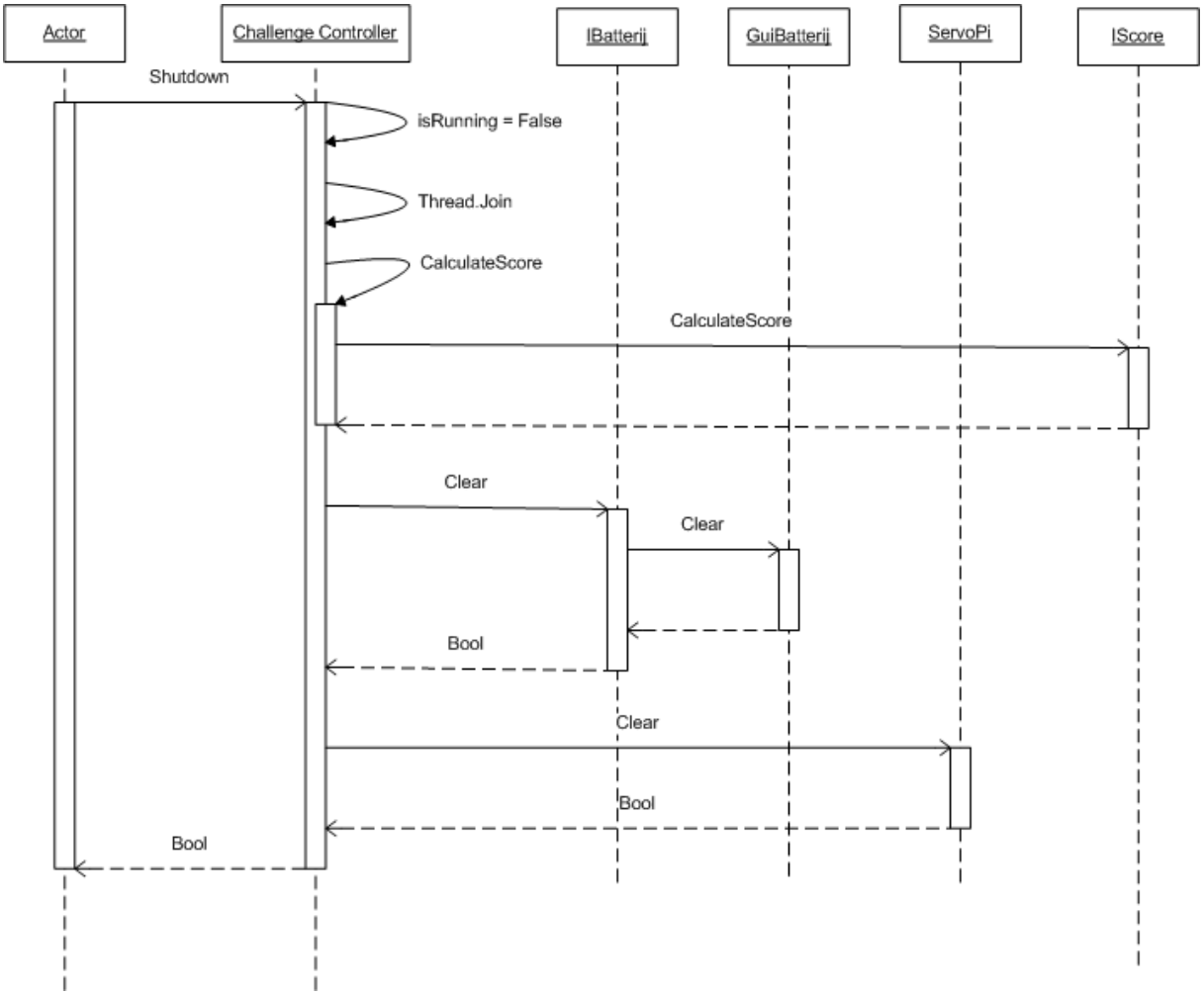
Het klassendiagram beschrijft de klassen en de relaties tussen de klassen. Een klasse is een sjabloon voor objecten en wordt gebruikt bij object-georiënteerd programmeren. Het klassendiagram is een manier om een beeld te scheppen over hoe het software systeem eruit zal zien. Hieronder in Figuur 8 is het klassendiagram voor dit project te zien. Het algoritme communiceert met de ChallengeController en de ChallengeController communiceert met alle andere componenten via interfaces. Hierdoor zijn bijna alle onderdelen uitbreidbaar met andere onderdelen. De Batterij is verantwoordelijk voor de GUI, omdat in de huidige situatie de energiestand de belangrijkste informatie is voor de gebruiker. Het gehele klassendiagram voor de Raspberry Pi Solar Challenge is in de tussenproducten lijst onder hoofdstuk 6.2 te vinden.



Figuur 8: Klassendiagram Raspberry Pi Solar Challenge (Let op: deze klassendiagram is in Sprint 1 gemaakt. De klassendiagram in de bijlage is de meest recente versie)

Sequencediagrammen

Het sequencediagram beschrijft de interacties tussen verschillende klassen die een bepaalde functionaliteit implementeren. Het sequencediagram geeft de tijdsvolgorde van de interacties tussen de klassen weer. In Figuur 9 volgt een sequencediagram voor de functionaliteit shutdown uit het project. Shutdown heeft als implementatie alles af te ronden en de score weer te geven. Deze en de rest van de sequence diagrammen zijn te vinden in de tussenproducten lijst onder hoofdstuk 6.3.



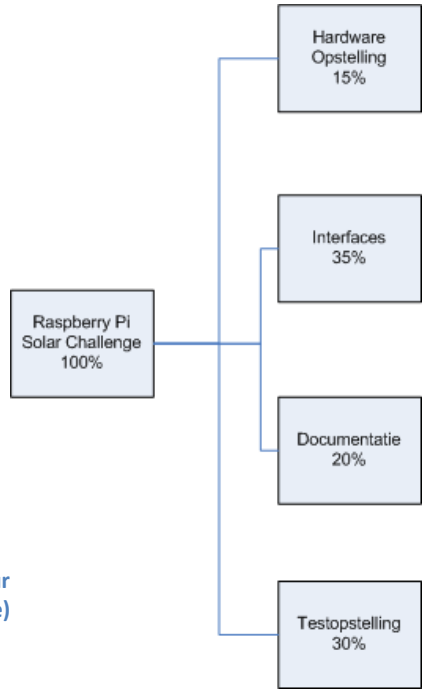
Figuur 9: Sequence diagram van Shutdown (Let op: Deze sequence diagram is in sprint 1 gemaakt, de respectievelijke sequence diagram in de bijlage is de meest recente versie)

Activity diagrammen

Activity diagrammen laat de activiteiten zien die plaatsvinden gedurende het gebruik van het systeem. Deze worden per use case beschreven en is een bredere weergave van de use case scenario. De activity diagram voor dit project is in de tussenproducten lijst onder hoofdstuk 6.4 te vinden en betreft wat er gebeurt bij het draaien van het algoritme gedurende een test.

10.3 Workbreakdown Structure

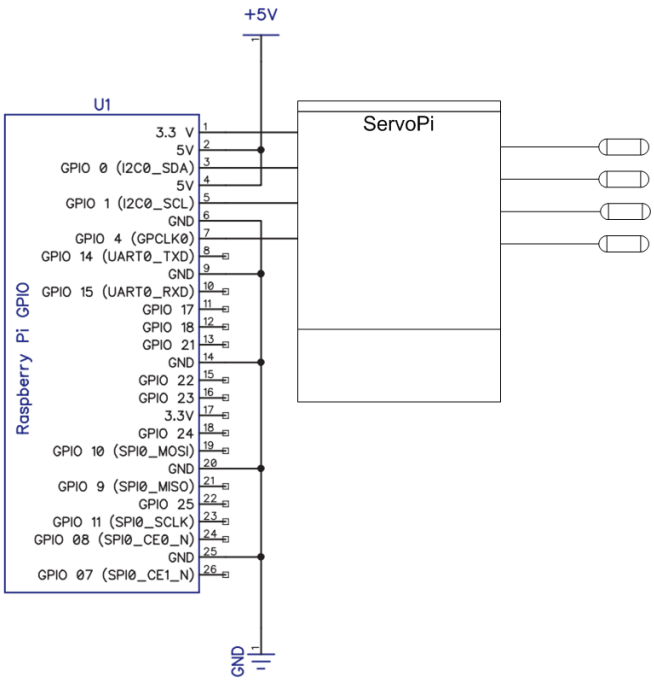
De workbreakdown structure (WBS) is een model waarbij een eindproduct wordt gescheiden in de onderdelen waaruit het geheel bestaat. Deze producten kunnen verder onderverdeeld worden tot het duidelijk is wat er moet gebeuren. In Figuur 10 is een WBS voor dit project te zien. Deze WBS is versimpeld en de volledige WBS is terug te vinden in de tussenproducten lijst onder hoofdstuk 7. De Raspberry Pi Solar Challenge zal 100% van alle tijd kosten, want dat is het eindproduct. 35% van de gehele tijd zal besteed worden aan het maken van de Interfaces. De percentages zijn een schatting om een beeld te scheppen hoeveel tijd in elk onderdeel zal zitten.



Figuur 10: Workbreaddown Structuur (verkleinde versie)

10.4 Aansluitschema

In sprint 0 is onderzocht hoe de servomotoren veilig aan te sluiten zijn aan de Raspberry Pi. Deze aansluitingen vinden plaats met de ServoPi, die over de gehele GPIO van de Raspberry Pi heen gaat. De ServoPi gebruikt een klein aantal van de GPIO pinnen van de Raspberry Pi om de motoren aan te sturen. De rest van de GPIO pinnen zijn beschikbaar via een header boven op de ServoPi. Om te kijken welke pinnen gebruikt worden door de ServoPi is een aansluitschema nodig. Figuur 11 toont dit schema.



Het aansluitschema geeft een abstracte weergave van de ServoPi, omdat het doel van dit schema is te tonen welke GPIO pinnen nodig zijn om de functionaliteiten van de ServoPi te gebruiken. Uit het plaatje valt af te leiden dat GPIO 0, 1 en 4 gebruikt worden. GPIO 0 en 1 zijn I2C pinnen die gebruikt worden voor de communicatie naar de ServoPi. GPIO pin 4 is een optionele instelling op de ServoPi om zelf een PWM signaal aan te leveren. De complete uitleg betreft I2C en het aansluitschema is te vinden in bijlage 6.

Figuur 11: Aansluitschema Servo Pi op Raspberry Pi

10.5 Uitzoeken werking Raspberry Pi en hoe er software op te zetten

De Raspberry Pi is een mini computer en niet krachtig. Het werkt ideaal om leerlingen te introduceren aan programmeren en embedded systems met behulp van Scratch en Python. Helaas is de Raspberry Pi niet sterk genoeg om een C++ ontwikkelomgeving te gebruiken, omdat niet genoeg RAM beschikbaar is. Vandaar is onderzocht hoe er C++ software beschikbaar gemaakt kan worden op de Raspberry Pi. Uit deze probleemstelling kan de volgende hoofdvraag geformuleerd worden: Hoe kan er C++ software ontwikkeld en deployed worden voor de Raspberry Pi? Uit deze hoofdvraag volgen de deelvragen: Hoe kan er direct op de Raspberry Pi code ontwikkeld worden?
Hoe kan er extern code ontwikkeld worden voor de Raspberry Pi?
Hoe kan externe software gemaakt voor de Raspberry Pi deployed worden op de Raspberry Pi?
Hoe kan er extern code debugged worden voor de Raspberry Pi?

Het is mogelijk om direct C++ te compileren op de Raspberry Pi, maar dan moet in acht genomen worden dat er geen ontwikkelomgeving beschikbaar is en dus een texteditor gebruikt moet worden. Dit is zeer ongeschikt bij het bouwen van complexere applicaties. Het onderzoek brengt in beeld welke ontwikkelomgeving en welk operating systeem het meest geschikt is om code te ontwikkelen en te compileren voor de Raspberry Pi.

Het compileren van code voor een platform of architectuur dat anders is dan waar de ontwikkelomgeving op staat heet cross compiling. Cross compiling werkt voor elk operating systeem anders. Het onderzoek richt zich op de methode die het mogelijk maakt om code te compileren en debuggen op de Raspberry Pi. Het debuggen speelt een grote rol, omdat de ontwikkelaren algoritmes moeten maken, waarbij het debuggen een onlosmakelijk onderdeel van het ontwikkelproces is.

Het deployen van programma's op de Raspberry Pi is met behulp van USB te doen of via remote acces. Remote acces is vaak sneller, omdat er geen onnodige kopieën gemaakt hoeven te worden en in en uit pluggen van een USB apparaat. Om te kunnen debuggen kan ook gebruik gemaakt worden van Remote Acces. Onder Linux wordt vaak gebruik gemaakt van GDB. De andere operating systemen is niet opgezocht, omdat de Raspberry Pi op een Linux omgeving draait.

Om resultaten en mogelijkheden te vinden, is gezocht op forums naar informatie over cross compiling in relatie tot de Raspberry Pi, vanuit een Windows of Linux omgeving. De resultaten zijn dat Linux cross compiling voor de Raspberry het beste werkt, vanwege de gelijkenis met het operating systeem, waardoor het makkelijker is om op te zetten. De ontwikkelomgeving die hiervoor gebruikt zal worden is Eclipse, omdat deze gratis beschikbaar is op elke operating systeem. Tevens zijn er verschillende tutorials beschikbaar om Eclipse op te stellen voor cross compiling voor de Raspberry Pi. Er waren vergelijkbare tutorials voor Visual Studio voor Windows, maar de cross compiling opzetten was veel ingewikkelder.

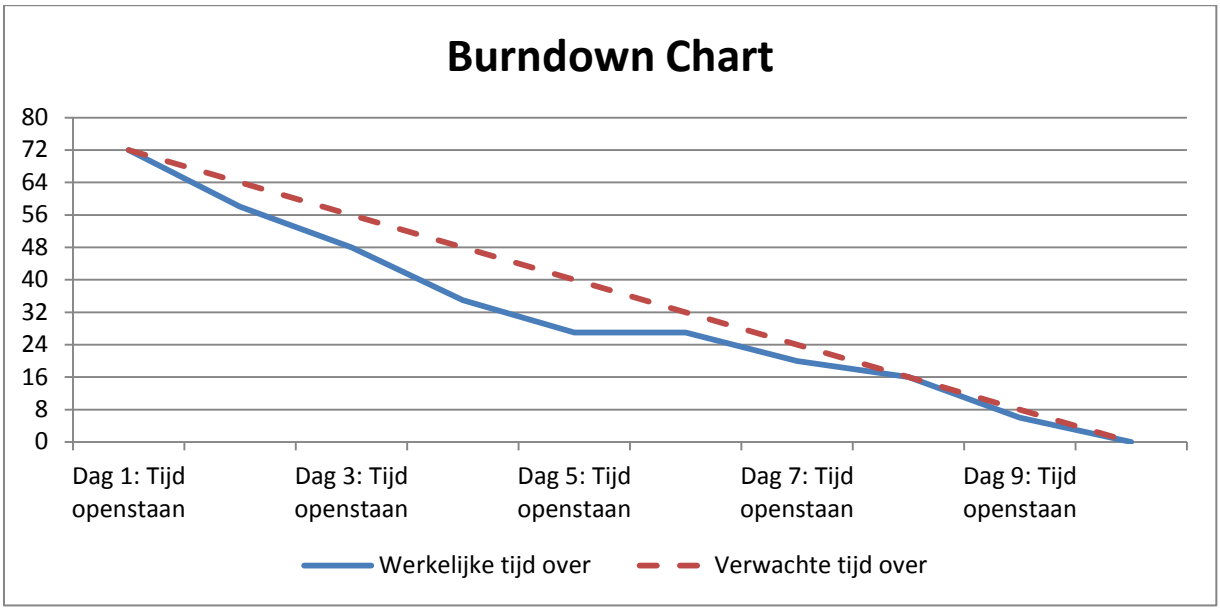
De bevindingen zijn in de tussenproductenlijst onder hoofdstuk 9 te vinden.

10.6 Repository

Een repository versimpelt het proces om versiebeheer toe te passen op het project. Met een repository worden bestanden online gezet in een trunk en valt elke update te synchroniseren. De repository vergelijkt het oude met het nieuwe en merged de bestanden of geeft conflictmeldingen aan. Gezien het feit dat het project zelfstandig is uitgevoerd, is het accent gelegd op het beheren van bestanden en het bieden van de mogelijkheid versies terug te draaien.

10.7 Resultaat en Feedback

Het resultaat voor deze sprint is het ontwerpen van de interfaces en het analyseren hoe software op de Raspberry Pi te zetten valt. Hoe de sprint is verlopen, valt in de burndown chart van Figuur 12 af te lezen. De sprint startte erg goed en liep ruim voor de geschatte tijdwaardes. De tweede helft van de sprint ging minder goed, maar de sprint liep niet uit en bleef binnen de tijdwaardes.



Figuur 12: Burndown chart sprint 1

Een story dat zinvol zou zijn geweest om mee te nemen in deze sprint, was het opstellen van de ontwikkelomgeving om naar de Raspberry Pi te kunnen groeien. Helaas is dit niet gelukt en 'verhuist' deze story naar sprint 3.

De demo verliep goed en het ontwerp is goedgekeurd. Verbeterpunten zijn dat de ontwerpen fraaier zijn vorm te geven om ze beter leesbaar te maken en dat een toevoeging van een interface is om de software blackbox te maken voor de deelnemers. Verder is het van belang dat de aankomende sprint tevens het begin van de bouwperiode betekent en dat de ontwikkelomgeving dus klaar moet zijn voor gebruik. Naast dit is het van belang dat de interface requirements klaar staan bij de ontwerpen, zodat bekend is waaraan deze moeten voldoen.

10.8 Afwijking van het afstudeerplan

Sprint 1 heeft een afwijking met het afstudeerplan, omdat er niet gewerkt is aan het maken van van de aansturing van de servomotoren. Er kon niet gewerkt worden aan de aansturing van de servomotoren, omdat de onderdelen daarvoor nog niet aanwezig waren. De andere opgeleverde producten, zoals het workbreakdownstructure en het aansluitschema zijn aanvullingen op eerder gemaakte documenten. Deze onderdelen waren nodig om deze documenten vollediger te maken.

11. Sprint 2

Sprint 2 gebruikt de ontwerpen en analyses uit sprint 1 om te beginnen met bouwen van de software. Hiervoor is het van belang om eerst de ontwikkelomgeving af te maken, zodat deze gebruikt kan worden om te ontwikkelen voor de Raspberry Pi. Daarnaast zal ook gewerkt moeten worden aan de hardware opstelling, omdat de software deze moet aansturen.

In Tabel 5 staat een deel van de requirements die deze sprint afgevangen worden.

Nummer	Requirement	Prioriteit
F1	Als workshopgever wil ik dat het zonnepaneel op twee assen kan draaien om zo een koepel te maken waarin het zonnepaneel kan draaien.	Must have
F2	Als workshopgever wil ik dat de testopstelling de algoritmes consistent getest kunnen worden.	Must have
F4	Als workshopgever wil ik dat de testopstelling de beweging van satelliet ten opzichte van zon kan simuleren	Must have
F8	Als software developer wil ik dat alle objecten / klassen uitbreidbaar zijn om toekomstig nieuwe klassen toe te voegen.	Should have
NF15	Als workshopgever wil ik dat de hardware opstelling onder de 5kg weegt.	Could have
NF16	Als workshopgever wil ik dat de hardware opstelling niet groter is dan 40 bij 40 cm.	Could have

Tabel 5: Requirements voor Sprint 2

11.1 Ontwikkelomgeving opstellen

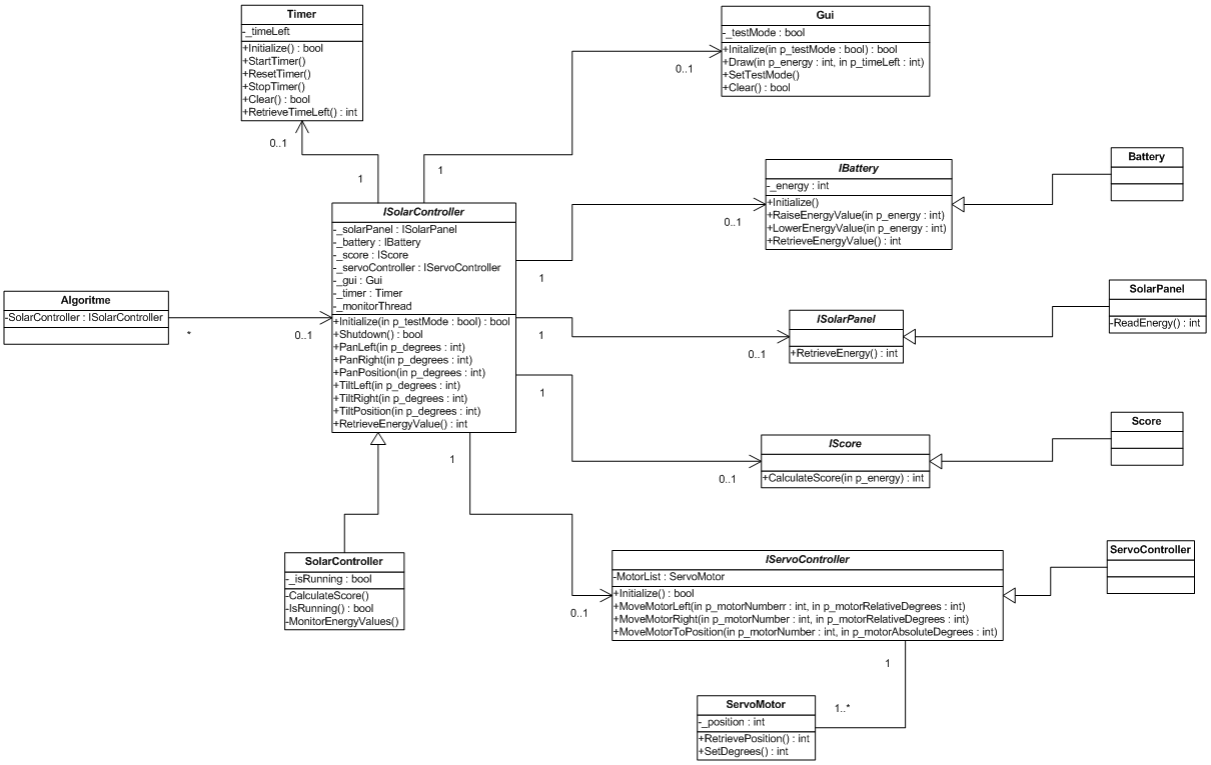
In sprint 1 is onderzoek gedaan naar de mogelijkheden voor de ontwikkelomgeving. Deze sprint past de resultaten toe om de ontwikkelomgeving te maken. Het resultaat van het onderzoek was een Eclipse omgeving te maken in een Linux omgeving. Na overleg met de begeleider is gekozen om het Fedora platform te nemen, omdat het een gratis Linux systeem is die veel geüpdatet wordt met nieuwe features en ondersteuningen.

Fedora zoals vermeld is een gratis Linux operating systeem. De werkomgeving is een Windows laptop. Om de Fedora omgeving te draaien is een virtuele machine nodig of een tweede computer of laptop. De virtuele machine is handiger, omdat de image gekopieerd kan worden naar andere machines toe, deze hoeven niet opnieuw geïnstalleerd en geconfigureerd te worden.

Eclipse zal de software compileren voor het ARM architectuur waarop de Raspberry Pi draait. De gecompileerde producten worden met een SSH connectie overgezet naar de Raspberry PI waarop de software zal draaien.

11.2 Interface Requirements

De interface requirements is een uitbreiding op het ontwerpen in sprint 1. De interface requirements beschrijven waaraan het ontwerp en de interfaces moeten voldoen. Hierdoor is het ontwerp lichtelijk aangepast, omdat een van de eisen was dat het systeem blackbox bereikbaar is. Het aangepaste ontwerp toont zich in Figuur 13.



Figuur 13: Klassendiagram Raspberry Pi Solar Challenge

Een aantal klassen hebben een nieuwe naam ontvangen zoals ChallengeController nu SolarController heet. Daarbij zijn er klassen bijgekomen en is de layout verandert voor leesbaarheid. De SolarController heeft nu een interface met de rest van het systeem. Dit creëert een bridge design pattern. De bridge design pattern ontkoppeld abstractie van zijn implementatie, waardoor beide onafhankelijk kunnen veranderen. Zo zou men meerdere controllers, solar panels, servocontrollers, batterijen, score systemen of batterijen kunnen toevoegen zonder dat bestaande code hoeft gewijzigd te worden.

Bovendien spreekt de algoritme nu met de interface en spreekt dus tegen een blackbox systeem aan. De algoritme weet alleen welke functies er zijn, maar wat deze functies in de achtergrond doen is onbekend. Dit is gewenst gedrag, omdat de deelnemers niks van de logica hoeven te weten en zich op hun algoritmes moeten focussen.

Het gevolg van de verandering aan het klassendiagram betekent dat andere UML diagrammen ook gewijzigd moeten worden. Met name de sequencediagrammen, omdat veel benamingen en functionaliteiten zijn verandert of verplaatst.

11.3 Bouwen

In deze paragraaf wordt het proces achter het bouwen van de software uitgelegd. Deze sprint draait om het bouwen en de omliggende taken. Het bouwen wordt stuk voor stuk opgebouwd en elkaar gekoppeld. Het gehele systeem zal een shared object (so)bestand zijn. Dat is een Linux variant voor de dynamic-link library (dll) van Windows. Het systeem zal geen executabel zijn, omdat de deelnemers het systeem moeten gebruiken om hun algoritme mee op te bouwen. Het algoritme zal de executable zijn die gebruik maakt van het shared object bestand.

In deze sprint zal het accent liggen op het maken van de basis voor het systeem. Bij de basis hoort de uitwerking van de klassendiagram en invulling van implementatie voor klassen die al ingevuld kunnen worden. De klassen waar aan gewerkt zijn zijn de SolarController, ServoController, Score, Battery, SolarPanel, ServoMotor, Timer en alle interface klassen. Merendeel van deze klassen hebben basis functionaliteiten gekregen en zijn gekoppeld aan elkaar zodat alleen de uitwerking van de implementaties nodig zijn om het product werkend te krijgen.

Zoals hiervoor genoemd zijn de UML diagrammen gebruikt bij het bouwen van de code. Eerst zijn alle interfaces gebouwd met de basis functies die hun minimaal moesten hebben. Vervolgens zijn de implementatieklassen gebouwd zoals SolarController en ServoController. Deze implementatieklassen hebben eerst alle interface functies overerft en geïmplementeerd. Vervolgens zijn een paar functionaliteiten gescheiden van elkaar, omdat die herhalend zijn of de code helderder maken.

Functionaliteiten die nog niet geïmplementeerd kunnen worden zijn het aflezen van energiewaardes van het zonnepaneel en de besturing van de servomotoren. Deze zijn nog niet gedaan, omdat de apparatuur nog niet compleet is om dit te doen. De Raspberry Pi heeft een sensor nodig om analoge signalen te lezen van het zonnepaneel om energiewaarde af te lezen. Voor de servomotoren is extra voeding nodig om de ServoPi aan te sluiten en te gebruiken.

```
bool SolarController::Initialize(bool p_testMode) {

	/*********************************************************
	Initialisatie van de verschillende klassen die de aansturing verzorgen
	*********************************************************/
	this->_monitorThread = thread(&SolarController::MonitorEnergy, this);

	return true;
}

void SolarController::MonitorEnergy() {
	float energyGain, energyLeft;
	this->_isRunning = true;
	int score = 0;

	while(this->_isRunning == true)
	{
		energyGain = this->_iSolarPanel->RetrieveEnergy();
		this->_iBattery->RaiseEnergyValue(energyGain);
		energyLeft = this->_iBattery->RetrieveEnergyValue();
		this->_graphicalUserInterface->Draw(energyGain, energyLeft);

		if(this->_isRunning == false)
		{
			score = this->_iScore->CalculateScore();
			//cout << "Test Finished score: " << score << endl;
		}
	}
}
```

Figuur 14 Snippet code van SolarController. Het betreft de MonitorEnergy functie

In Figuur 14 is een snipper van SolarController te zien. De monitorenergy functie is een functie die in een aparte thread draait van het hoofdsysteem, omdat deze altijd moet functioneren zonder invloed uit te oefenen op de werking van het algoritme. In de essentie zou het algoritme niet hoeven en moeten wachten op alle logica van het systeem en hiervoor is het programma multi-threaded gemaakt. In de Initialize functie worden alle variabelen en klassen klaar gemaakt voor het draaien van algoritmes. Hier wordt ook de thread aangemaakt die het monitoren van de energie bijhoud. Zoals eerder vermeld zijn veel functies wel gemaakt, maar bevatten nog weinig tot geen implementatie door ontbrekende hardware. Uiteindelijk is wel de kern van de functionaliteit klaar gemaakt voor gebruik en functioneert het goed met de place holder implementaties.

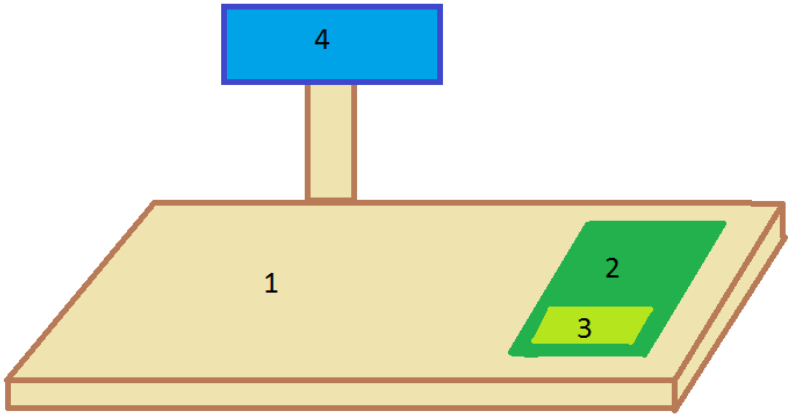
Op dit moment kan het systeem stop gezet worden door _isRunning op false te zetten en wordt ook een score berekend. Dit zit nu verwerkt in een functie die aangeroepen kan worden en ook in de destructor. Het is de bedoeling dat de algoritmes in theorie oneindig kunnen draaien en dit systeem zal op hetzelfde principe werken.

11.4 Hardware opstelling

In dit hoofdstuk wordt beschreven hoe de hardware opstelling tot stand gekomen is. Voor de hardware opstelling is eerst een ontwerp gemaakt en op basis van het ontwerp is een boodschappen lijst gemaakt. De aangeschafte onderdelen zijn vervolgens in elkaar gezet om het geheel te vormen.

11.4.1 Ontwerpen

Voor het ontwerpen van de hardware zijn de requirements gebruikt om te bepalen waar de hardware aan moet voldoen. Als proof of concept is een ontwerp gemaakt die voldoet aan de eisen. Een recreatie van dit ontwerp is in Figuur 15 te vinden, want deze is oorspronkelijk op papier gemaakt.



Figuur 15: Hardware ontwerp

In het plaatje is nummer 1 het bordje waarop het geheel zal gemonteerd worden. De Raspberry Pi is het groene vlak gerepresenteerd met nummer 2. De Raspberry Pi zal bevestigd worden aan de zijkant van het bord zodat het niet in de weg zit, maar hij zal dichtbij genoeg zijn om de bedrading naar de servomotoren zo kort mogelijk te houden. Nummer 3 is de ServoPi uitbreiding die de besturing voor de servomotoren regelt. Als laatste is nummer 4 het zonnepaneel bevestigd aan de pan en tilt opstelling. De neutral positie van de pan en tilt opstelling moest iets gecompenseerd worden zodat deze recht omhoog staat. Vandaar dat deze gemonteerd zit aan een houten staaf.

11.4.2 Bouwen

Met het ontwerp is een boodschappenlijst gemaakt van de spullen die nodig zijn. De onderdelen die al beschikbaar zijn, zijn de: Raspberry Pi, ServoPi, Zonnepaneel en de Pan en Tilt opstelling. Spullen die nog nodig zijn, zijn de: Adapter, pinheader voor ServoPi, Materiaal om informatie van het zonnepaneel te ontvangen en materiaal om de hardware opstelling aan te monteren.

Een groot deel van deze onderdelen zijn elektronica onderdelen en deze zijn verkrijgbaar bij verschillende elektronica winkels. De onderdelen voor dit project zijn aangeschaft bij Radio Service Twenthe in Den Haag. Voor het platform is gekozen om gebruik te maken van hout, omdat hout goedkoop is en gemakkelijk spullen aan te monteren met schroeven. Het hout dat gebruikt is zijn restanten van verbouwingen in de omgeving van het huis van de stagiair.

11.5 Testopstelling

De testopstelling is waarmee de algoritmes van de deelnemers beoordeeld zullen worden. De testopstelling heeft strenge eisen waarvan deze uitgewerkt zijn in de requirements te vinden aan het begin van deze sprint. In deze sprint is vooral aandacht geschonken aan het uitwerken hoe de testopstelling moet gaan functioneren.

De testopstelling is uitermate belangrijk voor de workshop en zal veel tijd nodig hebben om volledig af te ronden. Om niet een lange periode aan één onderdeel te werken is het uitwerken van hoe het zal werken in deze sprint gedaan. De rest van de testopstelling zal volgen in opvolgende sprints.

De testopstelling zal een gesloten donkere ruimte zijn. Dit kan een kamer zijn, maar ook een deel van de kamer. Er zal een lichtbron aanwezig zijn die de rol van de zon neemt. De hardware opstelling zal de 'zon' volgen gedurende één minuut. Na een minuut zal de test afgerond worden en krijgt het algoritme een score op basis van hoe goed het gefunctioneerd heeft. Om de kamer zo donker mogelijk te houden kan gebruik gemaakt worden van rolgordijnen of normale gordijnen. Er zijn stoffen die speciaal gemaakt zijn om licht tegen te houden. Deze stoffen zijn voldoende om buitenstaand licht zoveel mogelijk buiten te sluiten.

Een belangrijk punt is het bewegen van de lichtbron ten opzichte van de hardware opstelling. De algoritmes moeten de zon kunnen volgen, maar als geen van beide beweegt is het statisch object en hoeft er geen veranderingen getroffen te worden om zoveel mogelijk zonne-energie op te wekken. Dit is niet de bedoeling, omdat het een zonnepaneel op een satelliet moet voorstellen en deze beweegt wel degelijk ten opzichte van de zon. Om de beweging na te bootsen kan of de 'zon' bewogen worden of de hardware opstelling.

Om de zon te bewegen zijn er een paar mogelijkheden. De meest effectieve en dynamische is een extra pan en tilt opstellen gebruiken en daaraan een lichtbron monteren, maar deze optie is vrij duur. Een andere optie is gebruik maken van rails waarop het licht verplaatst kan worden, maar dit is handmatig werk en dus inconsistent. De laatste optie is meerdere lampen of looplampen te gebruiken die de beweging van de zon simuleren op een erg abstracte manier.

Naast het bewegen van de zon kan ook de hardware opstelling bewegen en ook hiervoor zijn meerdere opties. De eerste optie betreft wielen te monteren op de hardware opstelling waardoor deze kan bewegen, maar dit is wederom handmatig en dus niet consistent. Een alternatief is gebruik maken van een hellend platform waarop de opstelling schuift naar beneden. Verder zou ook een remote control voertuig gebruikt kunnen worden die de hardware opstelling draagt.

De goedkoopste methode is gebruik maken van een hellend platform waardoor de hardware onderdelen langzaam naar beneden schuiven. Het belangrijkste is dat de onderdelen niet snel naar beneden gaan anders is het te snel voorbij, maar het moet niet te langzaam gaan dat er nauwelijks beweging is en het zonnepaneel niet verschoven hoeft te worden. Deze methode is helaas niet consistent en levert geen vergelijkbare resultaten op. Hierdoor kan onderling geen beoordeling zijn van welk algoritme het beste is.

11.6 Resultaten en Feedback

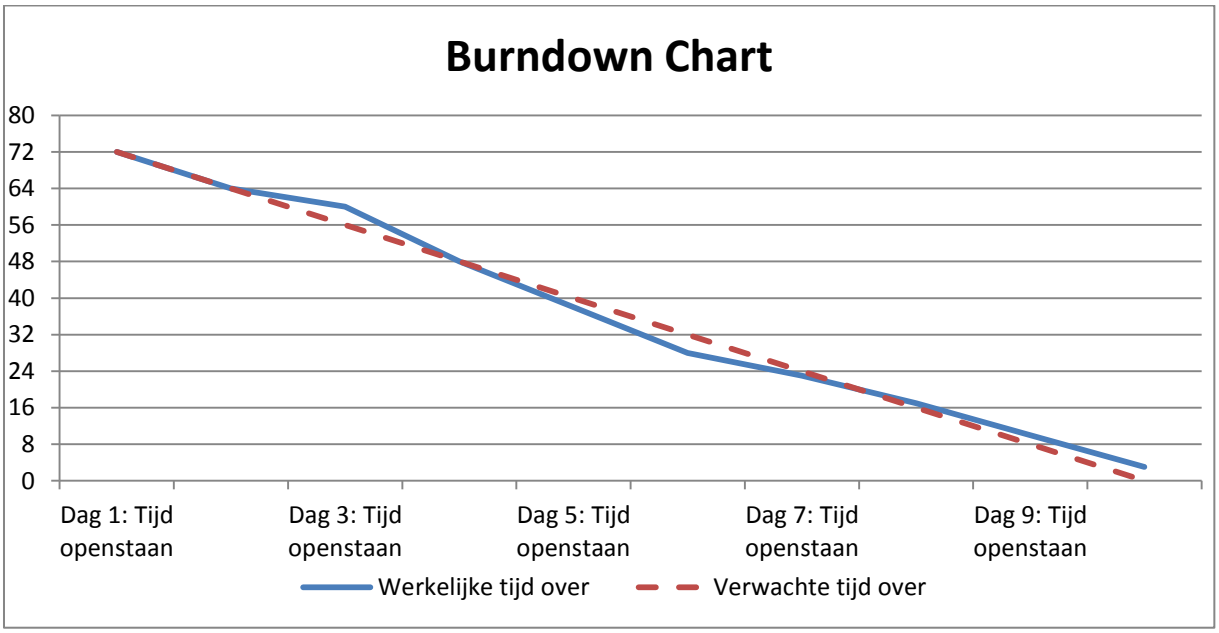
De resultaten van deze sprint zijn het ontwikkelen van de basisstructuur voor de interfaces, het ontwerpen en bouwen van de hardware opstelling en het uitwerken van de werking van de testopstelling.

Het ontwikkelen van de interfaces is geslaagd en er zijn zelfs extra delen bijgekomen die niet op de sprint backlog stonden. Specifieke implementatie zoals de communicatie met de ServoPi zijn nog niet ontwikkeld en het aflezen van het zonnepaneel, omdat de benodigde hardware onderdelen nog niet aanwezig zijn. De ontwikkelde code is nog niet getest en dat moet volgende sprint gebeuren om goede en complete functionaliteiten te hebben. Alle structuur en het bridge pattern zijn wel af en het geheel kan onderling communiceren en functioneren.

De hardware opstelling is bijna compleet. Het bord voor de hardware opstelling is klaar en de pan en tilt opstelling zijn gemonteerd. De rest van de onderdelen die nog gemonteerd moeten worden is de Raspberry Pi met ServoPi en het zonnepaneel met een onderdeel om de stroom af te kunnen lezen en die informatie doorgeven aan de Raspberry Pi.

Voor de testopstelling is uitgewerkt hoe deze moet gaan functioneren en meerdere opties gemaakt om dit gedaan te krijgen. Het goedkoopste is handmatig verplaatsen, maar dit is erg inconsistent. Een meer consistente oplossing is een hellend platform maken en daar de hardware opstelling naar beneden laten schuiven. Het schuiven is niet volledig consistent onderling hardware opstellingen, maar is een goede methode om goedkoop de lichtbron ten opzichte van het zonnepaneel te bewegen.

In Figuur 16 is de burndown chart voor deze sprint te zien. Deze sprint heeft niet alle stories kunnen afronden. De story die niet afgerond is, is de hardware opstelling. Het is niet afgerond, omdat er niet genoeg tijd beschikbaar was om alle benodigde onderdelen vast te stellen, aan te schaffen en te monteren.



Figuur 16: Burndown chart voor sprint 2

11.7 Afwijking van het afstudeerplan

Sprint 2 heeft weinig afgeweken van het afstudeerplan. Volgens de planning zou er gewerkt moeten worden aan de interfaces in het systeem en voor de deelnemers en dat is ook gebeurt. Hierbij is aangevuld dat er uitgezocht is hoe de testopstelling moet gaan functioneren en op welke manieren dat mogelijk gemaakt kan worden. Naast deze onderdelen is er ook gewerkt aan het maken van de hardware opstelling, omdat dit achterloopt en af moet zijn om de functionaliteiten van de interfaces af te kunnen maken.

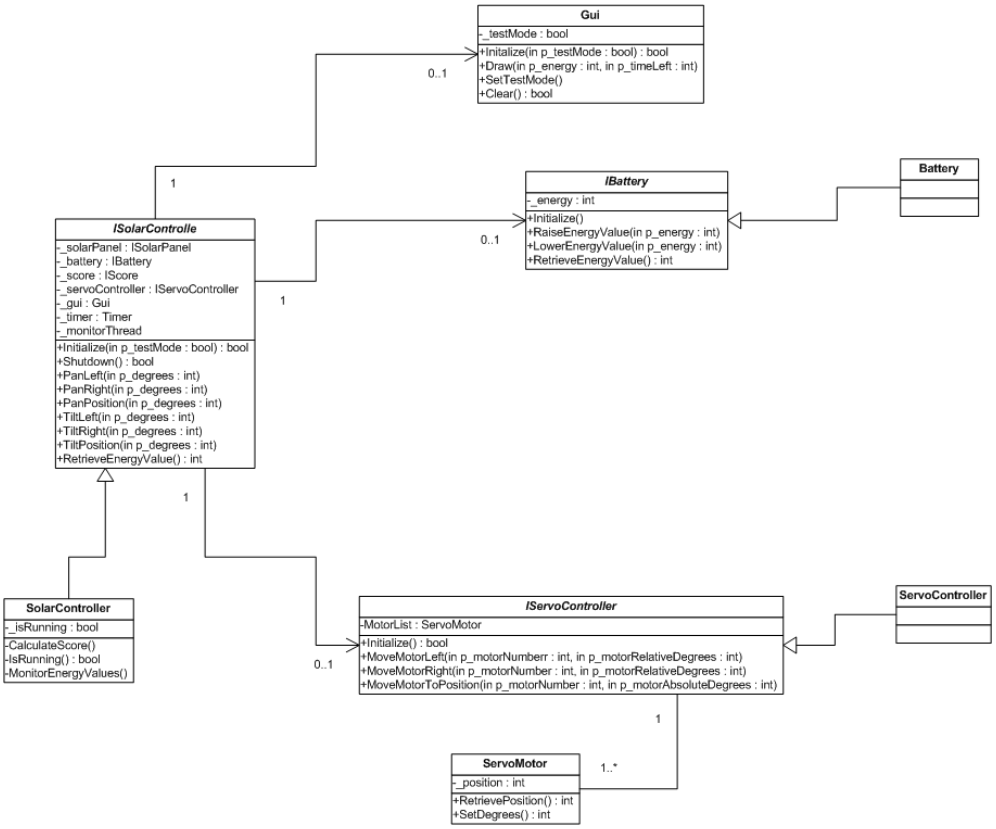
12. Sprint 3

Sprint 3 heeft de focus gelegd op het bouwen en werkend krijgen van de servomotoren. In sprint 2 is de basis klaar gelegd met de SolarController en de interface klassen, maar uiteindelijke implementaties ontbreken voor de ServoController en SolarPanel ontbreken volledig. Omdat de ServoController niet genoeg werk heeft voor de hele sprint is deze aangevuld met een proof of concept voor visuele feedback naar de eindgebruiker. Hieronder in Tabel 6 volgen een paar requirements die deze sprint aan bod zijn gekomen.

Nummer	Requirement	Prioriteit
F5	Als deelnemer wil ik dat het algoritme niet hoeft te wachten op de rest van het systeem	Must have
F11	Als workshopgever wil ik dat er een virtuele batterij gebruikt wordt om een werkelijke batterij te simuleren	Should have
F13	Als workshopgever wil ik dat de virtuele batterij energie verliest op basis van de bewegingen van de servomotoren	Should have
F14	Als workshopgever wil ik dat de deelnemers visuele feedback ontvangen over de stand van de batterij.	Should have
NF12	Als software developer wil ik dat elke klasse een duidelijke rol heeft	Should have

Tabel 6: Requirements voor Sprint 3

De klassen die in deze sprint aanbod komen zijn de SolarController, ServoController, Battery en deels Gui (Figuur 17). De klassen Servomotor is een hulpklasse om de situatie van elke servomotor bij te houden. De interfaces koppelen het geheel aan elkaar waarbij de ISolarController de eerste lijn is voor algoritmes en dus alle onderdelen aanstuurt.



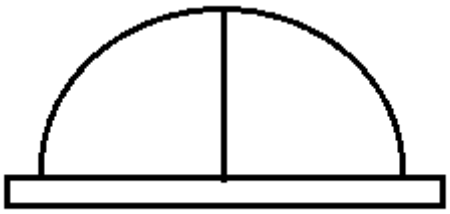
Figuur 17: Klassendiagram voor sprint 3

12.1 Bouwen

12.1.1 Hardware

Met het hardware ontwerp van sprint 2 is eerst de hardware gebouwd. Met hulp van thuis zijn een paar planken gezaagd en geschuurd om de Raspberry Pi en de servomotoren aan te monteren. De Raspberry Pi is gemonteerd met een paar schroeven en een paar knoppen die dienen als distance bus. Een distance bus is een onderdeel die forceert dat onderdelen op zijn minst een paar centimeter van elkaar af staan. Voor de Raspberry Pi is dit van belang zodat het makkelijk blijft kabels aan te sluiten en de SD kaart er in te zetten.

De Adafruit pan en tilt kit heeft een neutraal positie die een stuk scheef staat. Deze moet gecorrigeerd worden zodat de neutraal positie recht omhoog staat. De neutraalpositie moet recht omhoog staan, omdat daarvan uit de servomotoren links en rechtsom kunnen draaien. Dit is gewenst, omdat er dan een soort koepel ontstaat waarin het zonnepaneel zich kan bevinden. Een weergave hiervan is in Figuur 18 te zien.



Figuur 18: Draai-as Tilt servomotor vanuit de neutraal positie

Voor het aansluiten van de servomotoren is een onderzoek gedaan in sprint 0, omdat servomotoren een gevaar kunnen zijn voor de Raspberry Pi als ze direct aangesloten worden. Uit het onderzoek is naar voren gekomen om gebruik te maken van de Servo Pi. De Servo Pi is een klein uitbreidingsbord, speciaal gemaakt voor de Raspberry Pi, met als functionaliteit servomotoren aan te sturen. De Servo Pi heeft een speciale header die direct over de Raspberry Pi GPIO past. Helaas kunnen servomotoren niet direct op de Servo Pi aangesloten worden, omdat de aansluitingen niet overeenkomen. Om dit probleem weg te werken moest een pinheader gesoldeerd worden op de Servo Pi. Dit is met de hulp van meneer Bosvelt uitgevoerd.

12.1.2 Software

Graphical User Interface

Voor eindgebruikers is een directe manier van feedback gewenst om te hebben. Hiervoor waren een paar opties bijvoorbeeld met Led Lampen, Logging of een Grafische User Interface (GUI). De oplossing waarbij direct en duidelijke feedback is, is de grafische interface, omdat logging niet realtime is en ledlampen alleen beperkte mogelijkheden hebben voor feedback. De GUI kan op verschillende manieren opgebouwd worden zoals xwidget, gtk, website of java. Er is geprobeerd met het GTK framework een GUI te maken, omdat deze een goede uitgebreide documentatie heeft om een goede interface op te bouwen. Tevens is er standaard C++ ondersteuning bij met het GTK++ framework. Helaas was het framework erg ingewikkeld te koppelen met de cross compiler voor de Raspberry Pi en begon het framework niet meer te functioneren na toevoegingen van verschillende componenten.

Om alsnog een grafische interface te leveren is de tweede optie genomen namelijke met webbrowsers. Webbrowsers kunnen lokaal draaien en zijn multiplatform. De cross compiler hoeft verder niet verandert of aangepast te worden, want alle functionaliteit komt aan de website zijn kant en het C++ programma hoeft alleen data te versturen. Er was te weinig tijd om te bepalen wat de beste methode voor communicatie met de website is, maar om een begin te maken is er een xml bestand waarin alle data staat die de website nodig heeft, omdat deze snel op te bouwen is en een duidelijke structuur bevat die snel uitgelezen kan worden. In de interface is verder gekozen om de energie in een progressbar weer te geven, omdat dit het idee geeft dat een batterij opgeladen wordt, want het ziet er vergelijkbaar uit.

ServoController

Zoals hiervoor genoemd ligt de focus in deze sprint op de ServoController. De servomotoren zijn gekoppeld aan de Servo Pi. Om de Servo Pi aan te sturen moet er gecommuniceerd worden met Inter-Intergrated Circuit ook wel I2C, omdat dit de enige methode is waarmee de Servo Pi kan communiceren. I2C is een manier van communiceren tussen Integrated Circuits ook wel ICs met behulp van een master en slave systeem. De master stuurt berichten door aan de slaves met data en de slaves reageren op het bericht als die voor hun bedoelt is. I2C is ingebouwd op de Raspberry Pi en kan via de GPIO gebruikt worden om berichten over te sturen. De ServoController zal de rol opzich nemen om met I2C te communiceren met de Servo Pi.

Om de ServoPi te kunnen gebruiken moet er met I2C gecommuniceerd worden. De libraries meegeleverd bij de ServoPi waren geschreven in Python, omdat dit de meest gebruikte taal op de Raspberry Pi is. Daarnaast heeft Python een library om standaard met I2C te communiceren. Deze beide moesten vertaald worden naar C / C++ code om te kunnen gebruiken in de Raspberry Pi Solar Challenge library. Voor de library om met de GPIO te spreken is een 3rd party library gebruikt, omdat deze complete functionaliteit geven over de GPIO pinnen. Daarbij bespaart het veel tijd in het zelf maken van zo een dergelijke library. De I2C library is wel zelf vertaald, omdat hiervoor geen andere bestaande oplossing aanwezig zijn.

In Figuur 19 is een snippet van de I2C code te zien. Hier zijn delen van de functies waarbij de connectie met de ServoPi gemaakt wordt en ook een functie die data verstuurd naar de ServoPi. Het maken van de connectie in I2C via wiringPi is vrij simpel met de functie wiringPiI2CSetup, want deze wil alleen het adres van het device als parameter. Deze is voor de ServoPi standaard 0x40, maar kan veranderd worden met de adres verbindingen op het bord zelf.

```
bool ServoController::Initialize()
{
	int error = 0;
	_device = wiringPiI2CSetup(0x40);
	if(_device < 0)
	{
		return _device;
	}

	error = this->SetPWMFrequency(60);
	if(error < 0) {
		return false;
	}
}

int ServoController::SetPWM(int p_motorNumber, int p_pwm)
{
	int error = 0;

	error = wiringPiI2CWriteReg8(_device, 0x08 + 4 * p_motorNumber, p_pwm & 0xFF);
	if(error<0) {
		return error;
	}

	error = wiringPiI2CWriteReg8(_device, 0x09 + 4 * p_motorNumber, p_pwm >> 8);
	if(error<0) {
		return error;
	}
	Return error;
}
```

Figuur 19: Snippet van I2C code

De SetPWM functie neem als paramaters het motornummer en het pwm signaal dat erop gezet moet worden. Het PWM signaal wordt apart berekend op basis van de begin en eindstanden van de servomotor. Deze worden verstuurd met het wiringPiI2CWriteReg8 functie. Deze neemt een device adres, adres op device en data als argumenten. De device is achterhaald met de I2CSetup functie in Initialize(). De adressen waarnaar geschreven worden zijn 0x08 en 0x09 voor motor op slot 1 (telt vanaf 0). Opvolgende motoren zijn elke 4 adressen verder, dus kan er met motor nummer het adres gevonden worden met de formule 0x08 + 4 * motornummer en 0x09 + 4 * motornummer.

De data die geschreven wordt kan groter dan 1 byte zijn dus wordt er met bitwise operatoren ervoor gezorgd dat de eerste 8 bits in adres 0x08 zitten en de laatste 8 bits in adres 0x09 door het oorspronkelijke getal naar rechts te shiften met 8 plaatsen.

12.2 Testen

In deze sprint was gepland om UnitTests te maken voor de klassen en functionaliteiten die gemaakt zouden worden in deze sprint. Door complicaties in verband met Eclipse, het configureren van de UnitTest omgeving en het moeten leren begrijpen van de syntax van de toolset voor de unittests zijn er nog geen UnitTests gemaakt in deze sprint. In overleg met de begeleider is besloten dat deze story een lagere prioriteit heeft en is doorgeschoven naar de volgende sprint. Wel is er gebruik gemaakt van Error guessing om te bepalen welke onderdelen fouten en bugs bevatten.

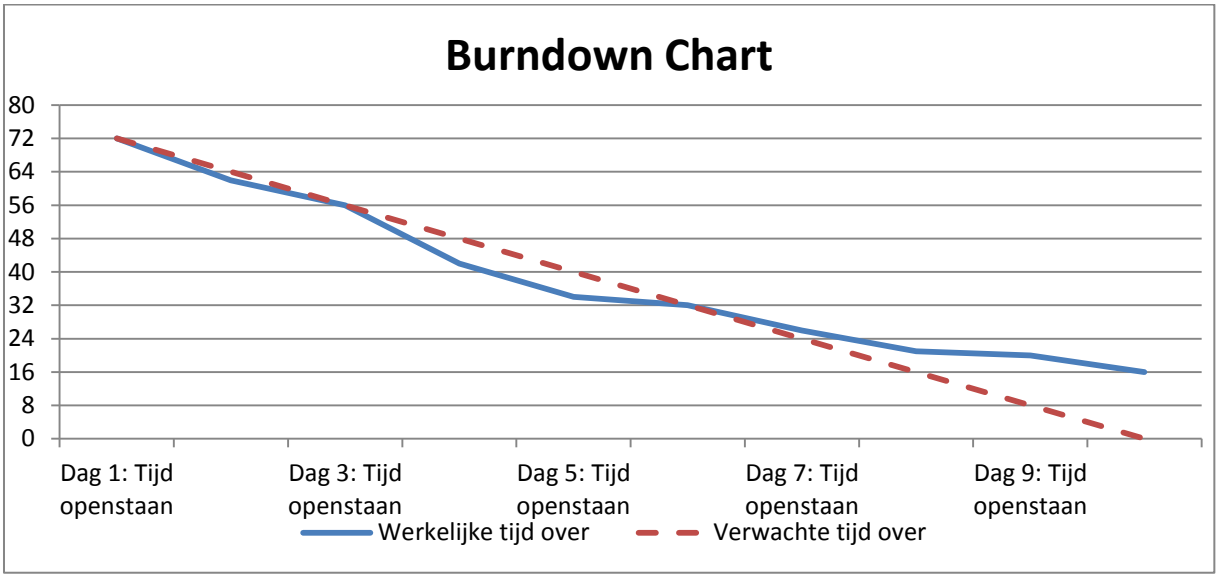
De functionaliteiten die getest zijn met Error guessing zijn de GUI, SolarController, Battery en ServoController. De Gui was het snelste te testen, omdat deze in de browser geopend kon worden en las informatie uit een bestand om weer te geven op het scherm. Als de informatie in het bestand verandert moet het scherm zich er op aan passen en dat gebeurde ook met de gewenste resultaten. Bij onjuiste informatie, zoals karakters waar een getal verwacht wordt, loopt de Progressbar vast. De label toont de informatie ook al is het de onjuiste.

Zoals in deze sprint benoemd is moest de ServoController gebruik maken van I2C om de servo's aan te sturen. Om dit zorgvuldig te kunnen testen en werkend te krijgen is deze in een aparte solution gemaakt. In deze solution is veelvuldig gebruik gemaakt van Error Guessing om te zien hoe de servomotoren reageren op verschillende commando's. Na het verkrijgen van de gewenste resultaten is de ServoController toegevoegd aan de Raspberry Pi Solar Challenge Library.

De SolarController wordt aangestuurd door een algoritme gemaakt door de deelnemers. Om te zien of alle functionaliteiten werken vanuit de deelnemers kant is een algoritme gemaakt die gebruik maakt van de Solar Challenge library. Alle functies worden gebruikt om te zien welke resultaten opduiken en of deze gewenst zijn. Deze laatste test was een proof of concept en is ook gebruikt bij de sprint demo om te demonstreren hoever het systeem is.

12.3 Resultaat & Feedback

Het resultaat van sprint 3 is een hardware opstelling die bijna compleet is en de software die bijna compleet is. De ontbrekende onderdelen zijn onderdelen gerelateerd met het zonnepaneel en de solar panel klasse. De GUI is een proof of concept en zal verder uitgebreid moeten worden, maar het functioneert wel en geeft het idee weer. Deze sprint is helaas tekort gekomen aan tijd en zijn alle stories niet afgerond, omdat de stagiar veel tijd verloren heeft aan het GTK+ framework en het opzetten van de UnitTest omgeving. De burndown chart is in de onderstaande Figuur 20 te zien.



Figuur 20: Burndown chart Sprint 3

De sprintdemo ging vrij goed en de begeleider is ondanks een story niet afgerond is tevreden met de resultaten. De GUI moet zijn informatie op een andere manier ontvangen, omdat de processor op hoge toeren draait door het vele lezen en schrijven van het bestand.

13. Sprint 4

Omdat sprint 3 uitgelopen is zijn er een paar onderdelen mee genomen naar sprint 4. Dit zijn de stories gerelateerd aan UnitTests maken. De andere activiteiten gepland voor deze sprint zijn het afmaken van het zonnepaneel. Om het zonnepaneel af te maken is er een onderzoek nodig naar het aansluiten van het zonnepaneel op de Raspberry Pi, daarna komt een ontwerp en aansluitingsdiagram en vervolgens kan alles gebouwd worden. Na het hardware deel af is kan de software aangepast worden met de betreffende implementaties om het zonnepaneel af te lezen.

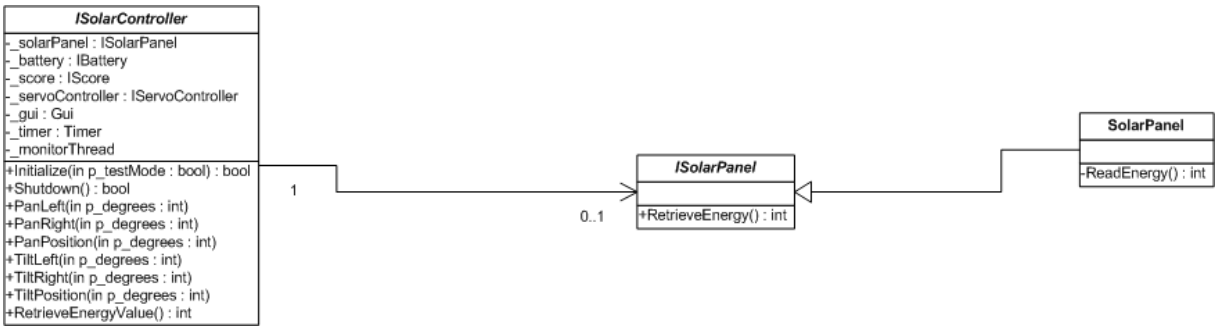
Naast de Unittests en het zonnepaneel is er ook gewerkt aan de testopstelling. Er is niet genoeg tijd beschikbaar in deze sprint om de gehele testopstelling af te ronden, maar de beginselen kunnen gedaan worden. Dit zijn een grondig onderzoek naar de mogelijkheden en opties voor de testopstelling waaronder de methode van testen in valt en hoe externe factoren buitengesloten worden. Met de resultaten van het onderzoek zal een ontwerp gemaakt worden. Uiteindelijk zal deze gebouwd worden, maar er is niet genoeg tijd beschikbaar in deze sprint om dat waar te maken. Omdat de API niet veel meer zal wijzigen is de API documentatie toegevoegd omdat het nu nog vers in het geheugen zit. De API documentatie beschrijft hoe de ontwikkelomgeving geconfigureerd moet worden om programma's te maken voor de Raspberry Pi.

De requirements in deze sprint draaien vooral om het zonnepaneel en de testopstelling. In Tabel 7 staan een paar requirements die afgevangen worden in deze sprint.

Nummer	Requirement	Prioriteit
F2	Als workshopgever wil ik dat de testopstelling de algoritmes consistent getest kunnen worden.	Must have
F4	Als workshopgever wil ik dat de testopstelling de beweging van satelliet ten opzichte van zon kan simuleren	Must have
F15	Als workshopgever wil ik dat de testopstelling minimaal 5V spanning kan generen uit het zonnepaneel.	Could have
NF20	Als workshopgever wil ik een duidelijke handleiding hebben hoe de hardware opstelling opgebouwd is.	Could have

Tabel 7: Requirements voor Sprint 4

Het doel van deze sprint is om de klassen uit de volgende UML diagram in Figuur 21 verder te implementeren. Het gaat vooral om het SolarPanel gedeelte. De overige klassen zijn niet aanwezig omdat deze weinig tot niet gewijzigd zullen worden in deze sprint.



Figuur 21: Klassendiagram sprint 4

13.1 Uitzoeken

13.1.1 Uitzoeken aansluiten van een zonnepaneel

Voor het aansluiten van het zonnepaneel moet een klein onderzoek gedaan worden, omdat het zonnepaneel een te hoge spanning kan opleveren voor de Raspberry Pi. Daarbij geeft het zonnepaneel een analoge signaal terwijl de Raspberry Pi een digitaal signaal verwacht. Een digitaal signaal kan alleen 0 of 1 zijn en een analoog signaal kan alle waarden tussen 0 en de maximale spanning aannemen.

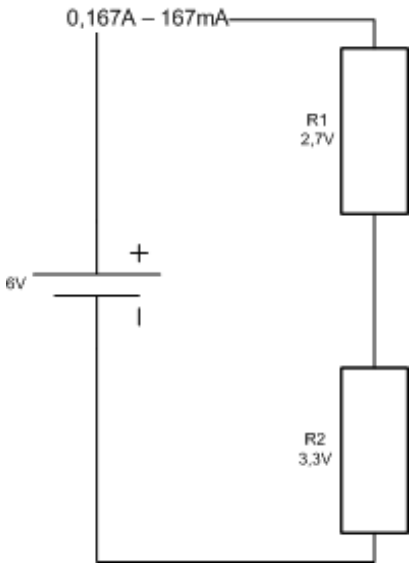
Een manier vinden om het zonnepaneel aan te sluiten aan de Raspberry Pi en dus het analoge signaal om te zetten naar een digitaal signaal is het doel van het onderzoek. De hoofdvraag is: Hoe kan het zonnepaneel aangesloten worden op de Raspberry Pi en hoe kan de Raspberry Pi het signaal van het zonnepaneel lezen?

Voor het onderzoek zijn de volgende deelvragen opgesteld:
- Hoe kan een analoog signaal naar digitaal signaal omgezet worden?
- Hoe kan het zonnepaneel direct aangesloten worden op de Raspberry Pi?
- Is er een expansiebord waarmee die aangesloten kan worden op de Raspberry Pi en het analoge signaal omzet naar een digitaal signaal?

Om een analoog signaal om te zetten naar een digitaal signaal kan een analoog naar digitaal converter (ADC) gebruikt worden. Dit zijn chips die met behulp van een circuit bepalen in welke range het analoge signaal zit en zet dit om naar een digitaal signaal. De Raspberry Pi heeft deze functionaliteit niet ingebouwd en dit moet met een alternatief opgelost worden. Hiervoor zijn twee mogelijkheden, sluit het paneel direct aan en met behulp van software kan bepaald worden hoe hoog het signaal is of met een externe ADC kan bepaald worden hoe hoog het signaal is.

De externe methode is een expansiebord gebruiken voor de Raspberry Pi. Hier zijn een paar mogelijkheden in zoals de ADC-DAC Pi of alleen de chips als MCP3008 en PicAxe. De chips moeten gesoldeerd worden met een circuit om vervolgens aan te sluiten op de Raspberry Pi. De ADC-DAC Pi is een kant en klare Analoge naar Digitaal converter en vice versa van ABElektronics. Dit is hetzelfde bedrijf waar de Servo Pi is vandaan gehaald. Uit deze oplossingen is de ADC-DAC Pi het beste, omdat deze net als de Servo Pi ontworpen is om gestapeld te worden met andere ABElektronics expansieborden. Hierdoor blijven de GPIO pinnen beschikbaar en kan er eventueel nog een bord ook bijkomen. Daarbij is de ADC-DAC Pi kan en klaar en hoeft dus geen extra soldeer werk te krijgen om te functioneren. De ADC-DAC Pi gebruikt stroom van de Raspberry Pi om te functioneren en analoge signalen die binnen komen moeten ook rond diezelfde waarde zitten om schade aan de borden te voorkomen. Hierdoor moet de stroom verlaagt worden met een Spanningsdeler en daar kom ik in de volgende alinea op terug. Al deze externe oplossingen als ADC gebruiken SPI om te communiceren met de Raspberry Pi.

Het zonnepaneel kan direct aangesloten worden, maar dat levert schade op aan de Raspberry Pi omdat het Voltage te hoog is. Deze moet dus verlaagd worden met behulp van een spanningsdeler. Dit moet ook gebeuren bij het gebruik van de ADC-DAC Pi uit de vorige alinea. Hiernaast in Figuur 22 is het ontwerp van een spanningsdeler in de maximale stand. Bekende waardes hier zijn het zonnepaneel ook wel voeding want die is 6V en de stroom in Ampère is 1/6A of 167mA (Gehaald uit de datasheet). De spanning die nodig is bij de Pi is 3,3 Volt en dit is in dit voorbeeld weerstand 2 of R2. Weerstand 1 of R1 is de resterende spanning en dit is 2,7V. Om doorbranding van de weerstanden te voorkomen moet de stroom iets lager zijn dan maximaal. Hier is voor 0,1A gekozen, omdat het ruim in de veilige zone zit en makkelijk mee te rekenen is.



Figuur 22: Spanningsdeler nodig bij het aansluiten van het zonnepaneel

Met de wet van Ohm en het formule van vermogen kan berekent worden hoe groot deze weerstanden minimaal moeten zijn om deze gewenste resultaten te krijgen.

U = I * R

P = U * I

Rv = R1 + R2 ,omdat het een serie schakeling is.

Umax = 6V, U2 = 3,3V, U1 = 2,7V

Imax = 1/6A = +-0,167A , I = 0,100A

U1 = I * R1 en U2 = I * R2. Hieruit volgt dat R1 = U1/ I en R2 = U2/ I. Na invulling komt het resultaat R1 = 27 Ohm en R2 = 33 Ohm. Uit de standaard selectie weerstanden kunnen deze niet gekozen worden, maar het gaat om de verhouding tussen de weerstanden. De verhouding is 1 op 1,2222222, maar deze is te laag om een paar weerstanden te vinden uit de standaardwaarden die hieraan voldoen. Om deze reden is gekozen om een klein verlies te accepteren en het inkomende signaal met 0,3 Volt te verlagen. Hierdoor worden R1 en R2 gelijk aan elkaar op 3 Volt.

Nu deze informatie berekend is kunnen we met het vermogen bepalen wat de weerstanden echt moeten zijn. Het maximale vermogen van het zonnepaneel is 1 Watt (Gehaald uit de datasheet). Het vermogen over elke weerstand is dan P = U * I -> P = 3 * 1/6 = 0.5 W. Met het ombouwen van de wet van Ohm en het formule voor vermogen krijgen we de volgende formules: R = U / I en I = P / U
Deze kunnen gecombineerd worden tot één formule namelijk : R = U / (P / U) = 3 / (0,5 / 3) waaruit 18 Ohm komt. Dit is de minimale waarde die de weerstanden moeten hebben zodat alles draait op de maximale toegestane waardes. Er is gekozen om de eerst volgende waarde uit de standaardseries te nemen en dit is 100Ohm.

Het verschil bij het direct aansluiten en aansluiten via een expansiebord is dat bij het direct aansluiten een condensator nodig is. De condensator laad langzaam op en bij een bepaald treshold zal het signaal hoog genoeg zijn voor de Raspberry Pi om een hoog signaal te lezen of een digitale 1. In software kan een timer bijgehouden worden over hoe lang dat proces duurt en daaruit kan een schatting gedaan worden over hoe groot de spanning is. De vertaling is vrij inaccuraat en is afhankelijk van onder andere het operating systeem en het vertaalproces. Het operating systeem onderbreekt het programma en er is weinig controle over wanneer dat gebeurt, dus het zou kunnen voorkomen dat de timer de ene keer heel snel een hoog signaal krijgt omdat die niet geüpdatet werd vanwege de onderbreking. Naast het operating systeem is het vertaalproces ook inaccuraat, omdat de software een tijd geeft over hoe lang de condensator nodig heeft gehad voordat een hoog signaal gegeneerd werd. Dit is geen spanningswaarde en moet vertaald worden, maar dit kan een paar tiende tot enkele volt afwijken, omdat het handgemaakt moet worden en het veel aanname werk is over welke waarde welk voltage is. Dit komt voort uit wederom de inconsistente onderbrekingen van het operating systeem.

Om de inconsistentie te meten is een testprogramma gemaakt met daarin een aantal timers die naast elkaar lopen met behulp van threads. Dit programma is herhaaldelijk gedraaid en ook onder verschillende workloads op de processor. Hieronder in Tabel 8 zijn de resultaten te zien van één run zonder workload op het systeem.

10 Seconde Run	Zonder workload
Run 3	Thread 1: 0 milli en 83 micro Thread 2: 2 milli en 125 micro Thread 3: 3 milli en 291 micro Thread 4: 2 milli en 719 micro Thread 5: 5 milli en 718 micro

Tabel 8: Resultaten van Run 3 van de 10 seconde run. Deze test is uitgevoerd om uit te zoeken hoe consistent het operating systeem functioneert

Uit de bovenstaande tabel kan gehaald worden dat Thread 0 bijna niet onderbroken werd. Threads 2 en 4 hebben een uitweiking van 2 bijna 3 milliseconde. Thread 3 is een paar keer meer onderbroken met een score van bijna 3,3 milli seconde. Maar dan komt thread 5 met een bijna 6 milliseconde vertraging. Het verschil hierin is niet consistent, maar het is te klein om een grote impact te hebben op het systeem. Uiteindelijk zou er misschien een paar millivolt verschil kunnen zijn per run en dat is na overleg met de product owner niet genoeg om uit te maken.

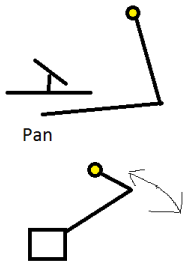
13.1.2 Uitzoeken ontwerp van de testopstelling

De testopstelling heeft strenge eisen en daarom moet er een uitbundig onderzoek gebeuren om alle mogelijkheden langs te gaan. De testopstelling moet consistent de algoritmes testen en het moet ook de hardware opstelling of de lichtbron ten opzichte van de ander bewegen. Dit is het probleem dat opgelost moet worden. Het doel is een methode vinden waarmee de testopstelling consistent dezelfde test zal uitvoeren. Daarbij moet de lichtbron of de hardware opstelling bewegen ten opzichte van de ander waardoor het zonnepaneel verplicht wordt de lichtbron te volgen.

De hoofdvraag voor dit onderzoek is: Op welke manier kan er consistent getest worden en hoe kunnen externe lichtbronnen zoals zonlicht buitengesloten worden? Bij deze deelvraag ontstaan de volgende deelvragen:
Welke factoren spelen een rol voor de testopstelling?
Hoe kan de lichtbron ten op zichtte van de hardware opstelling bewegen?
Hoe kan de hardware opstelling ten op zichtte van de lichtbron bewegen?
Hoe houden deze methoden rekening met de factoren die belangrijk zijn voor de testopstelling?
Hoe sterk moet de lamp zijn om energie op te wekken met het zonnepaneel?
Hoe kan licht van externe bronnen buitengesloten worden?

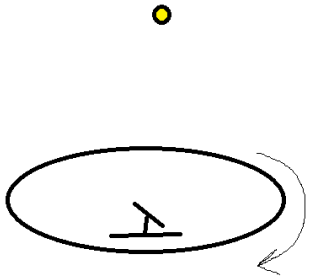
De belangrijkste factoren voor de testopstelling zijn consistentie en prijs. De grote en het gewicht spelen een secundaire rol, omdat gewenst is dat het makkelijk vervoerbaar is, maar dit is niet van kritisch belang. Consistentie is belangrijk, omdat het kritisch is dat de algoritmes op dezelfde wijze getest worden voor een eerlijke vergelijking tussen de algoritmes. De prijs is belangrijk , omdat het niet honderden euro's mag kosten.

Om de lichtbron te verplaatsen kan dat gedaan worden met bijvoorbeeld een pan systeem. Het pan systeem zal de lichtbron op een horizontale as draaien rondom de hardware opstelling. De oplossing is consistent, omdat de motoren door een microcontroller aangestuurd worden en hij is niet erg duur. Een nadeel is dat de lichtbron een vrij lange kabel krijgt anders wordt de testopstelling te klein en zal het systeem in de weg kunnen zitten van de hardware. Een voorbeeld hiervan is in Figuur 23.



Figuur 23: Testopstelling als het een Pan systeemgebruikt

Een ander voorbeeld is de draaischijf of platenspeler. De hardware opstelling kan op een draaiende plaat gezet worden. Een voorbeeld hiervan is in Figuur 24. De lichtbron zit boven de plaat naast het middelpunt waar de hardware opstelling om heen zal draaien. De lichtbron zit naast het middelpunt, omdat de zonnepaneel anders statisch kan zijn en niet de lichtbron hoeft te volgen. Deze oplossing is consistent, omdat platenspelers gemaakt zijn om een vast toerental te bewegen. Daarbij zijn tweede hands platenspelers vrij goedkoop aan te schaffen. Het nadeel aan deze oplossing is dat er sleepcontacten nodig zijn. Deze zijn duur, maar essentieel in het maken van een contact tussen draaiende en stilstaande objecten.



Figuur 24: Testopstelling als het een Pdraaischijf gebruikt

Naast de consistentie moeten de opstellingen de externe lichtbronnen kunnen blokkeren. Hiervoor zou een gehele ruimte verduisterd kunnen worden of men verpakt de testopstelling in duisternis. Om een hele kamer te verduisteren kan verduisteringsstof gekocht worden en opgehangen worden of op goedkopere wijze plasticplaten ophangen. Dit is vrij vervelend en onhandig, maar de beste oplossing als de testopstelling groot is, maar als deze klein is kan het op een andere manier gedaan worden. Bij kleine of vrij kleine testopstellingen kan er een grote kartonnen doos overheen gezet worden. Er blijven kleine kieren over, maar met een doek kan dat verholpen worden. De testopstelling zal op deze manier ook geen last hebben van externe bronnen en is er ook geen aparte ruimte nodig om te kunnen testen.

De lamp die nodig is om stroom op te wekken met het zonnepaneel hoeft niet sterk te zijn. De lampen uit een normaal huishouden zijn meer dan sterk genoeg. Bij een doos als verduistering kan deze aan de ondergrond gemonteerd worden en de kabel aan de achterkant eruit. Bij een aparte ruimte kunnen alle lampen verwijdert worden behalve één lamp die zal dienen als lichtbron.

Het onderzoek heeft nog geen eindresultaat, omdat het niet op tijd af gemaakt kon worden. Er kwamen aan het einde een paar punten tevoorschijn, zoals het berekenen van de kleinste stappen waarop gereageerd kan worden en die vergelijken onderling. Wel is bekend dat de kandidaten voor de testopstelling de platenspeler zijn of het zonnepaneel systeem monteren aan nog een pan en tilt set. Deze tweede set van servomotoren zal de beweging van zonnepaneel ten opzichte van de lichtbron simuleren.

13.2 Testen

In deze sprint zijn de unittests gemaakt en uitgevoerd. Unittesten is het testen van elke klassen of unit op zichzelf, dus losstaand van de rest van het systeem. Veel klassen zijn afhankelijk van elkaar dus er zijn stubs gemaakt om de functionaliteiten te imiteren. Om in Eclipse te kunnen Unittesten moet een plug-in geïnstalleerd worden en een testframe work geïnstalleerd worden. De plug-in kan standaard met 3 frameworks gebruikt worden. Het boost testframework, google testframework en QT testframework.

Het QT testframework is speciaal gemaakt om applicaties van QT te testen. Deze kan gebruikt worden voor andere applicaties, maar het is veel werk om alle references en dependencies van QT goed te zetten. Het Boostframework is een betere optie, omdat deze naast Boost projecten ook standaard C++ projecten kan testen. Het nadeel hieraan is dat het framework in een apart project moet draaien en de klassen apart include moeten worden. Het zal functioneren, maar met een grote hoeveelheid testcases wordt het erg onoverzichtelijk. Google test detecteert automatisch alle tests en draait ze of er kan handmatig geselecteerd worden welke gedraaid worden. Daarbij is de documentatie goed en makkelijk te begrijpen. Hierdoor is gekozen om het testframework van google te gebruiken.

In verband met tijdsdruk zijn niet alle functionaliteiten getest, omdat het testframework in Eclipse gebruikt wordt en het ingewikkeld is om deze los te draaien. Hierdoor zijn hardware specifieke functies zoals de GPIO niet getest met een Unittest. De klasse met de meeste impact is de ServoController, omdat die als enige de GPIO gebruikt. Wel kon getest worden was de hulpklasse ServoMotor voor ServoController. De ServoController heeft een lijst van servomotoren met daarin hun positie. Deze wordt gebruikt om relatieve posities te berekenen bij bewegingen. Naast de ServoController is SolarPanel nog niet getest, omdat deze vooralsnog een placeholder is zodat de batterij opgeladen kan worden uit wat later het zonnepaneel moet worden.

In Figuur 25 is een test voor Initialize te zien voor de Battery klasse.

```
TEST(Battery, Initialize) {
	Battery* bat = new Battery();
	bool test = bat->Initiliaze();

	EXPECT_TRUE(test);
	EXPECT_EQ(600, bat->RetrieveEnergyValue());
}
```

Figuur 25: Test voor Initialize met het GoogleTest framework

Deze Unittest test de Unit Battery en de functie Initiliaze. Initialize initialiseert de batterij met een beginstand. Deze is standaard 600. Initialize retouneerd een true of false op basis van of de actie geslaagd is. Deze wordt getest met EXPECT_TRUE(); Vervolgens om echt zeker te zijn wordt getest of de batterijstand echt 600 is. Dit wordt gedaan met EXPECT_EQ waarbij bat->RetrieveEnergyValue de huidige energiestand opvraagt. Als één van deze testen fout gaat komt er een melding op het scherm te staan dat de test niet geslaagd is en vervolgens kan daar naar gekeken worden.

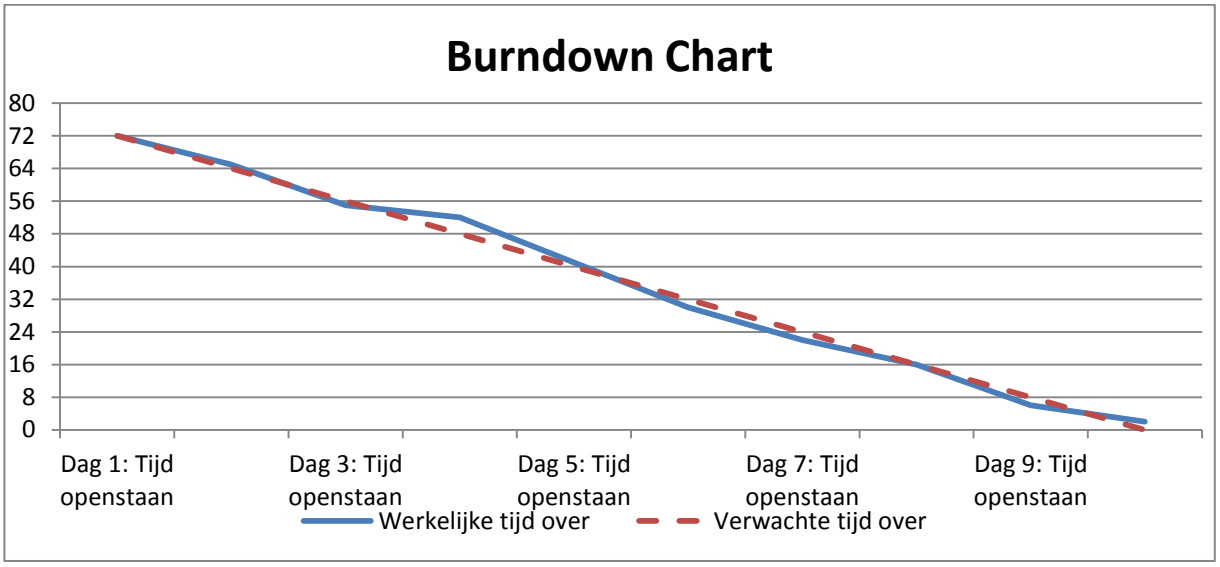
13.3 Documentatie

Voor de workshop is een API documentatie gemaakt. Deze zal voornamelijk door workshopgevers gebruikt worden om de laptops te configureren met de ontwikkelomgeving. Er is een API documentatie nodig, omdat er een paar complexe stappen zitten om programma's of software in het algemeen te ontwikkelen in C++ voor de Raspberry Pi.

Om code te ontwikkelen in C++ is in een vorig onderzoek gekozen om gebruik te maken van cross compileren. Hierbij wordt de code op een ander platform gemaakt dan de Raspberry Pi en later overgezet naar de Raspberry Pi toe. Om te zorgen dat de ontwikkelomgeving goed zal functioneren tijdens de workshop kan met behulp van de API documentatie alles geconfigureerd en klaar gezet worden.

13.4 Resultaat & Feedback

Het resultaat van deze sprint zijn de unitTests, de onderzoeken en de handleiding voor de API. Deze sprint is beetje heen en weer geschommeld, maar qua tijdsbesteding is er vrij goed geschat. In onderstaande Figuur 26 is de burndown chart te zien voor deze sprint.



Figuur 26: Burndown chart sprint 4

De API handleiding was bijna af. Er ontbraken plaatjes om het duidelijker te maken, dus dat moet nog gebeuren. Het onderzoek voor de ADC en het zonnepaneel was niet voldoende, omdat er geen goede vergelijkingen zijn gemaakt. De ene methode is beter dan de ander, maar er ontbreekt hoeveel beter is dan de ander. Het onderzoek naar de testopstellingen is vrijwel hetzelfde verhaal. Er ontbrak genoeg informatie om concreet te zeggen: Dit is de beste oplossing. Deze onderdelen heb ik onderschat en moeten verbeterd worden in de volgende sprint.

14. Sprint 5

Na de complicaties van sprint 4 zijn er deze sprint afspraken gemaakt dat de onderzoeken voor de analoog naar digitaal converter en de testopstelling af zijn. Deze sprint heeft veel onderzoek en een deel ontwerpen en bouwen gehad, maar dit is voornamelijk met betrekking tot de testopstelling. De software kant heeft weinig aandacht gekregen in deze sprint in verband met tijdsgebrek. Een paar requirements uit deze sprint staan in Tabel 9 hieronder vermeld.

Nummer	Requirement	Prioriteit
F2	Als workshopgever wil ik dat de testopstelling de algoritmes consistent getest kunnen worden.	Must have
F3	Als workshopgever wil ik dat de testopstelling de algoritmes controleerbaar test	Must have
F4	Als workshopgever wil ik dat de testopstelling de beweging van satelliet ten opzichte van zon kan simuleren	Must have
F6	Als workshopgever wil ik dat de testopstelling niet hoeft te wachten op andere onderdelen van het systeem.	Must have
NF10	Als workshopgever wil ik dat de testopstelling niet groter is dan 50 bij 50 cm	Should have
F15	Als workshopgever wil ik dat de testopstelling minimaal 5V spanning kan generen uit het zonnepaneel.	Could have

Tabel 9: Requirements voor sprint 5

14.1 Uitzoeken

De uitzoekdelen waren afgelopen sprint niet afgerond, omdat er het een en ander vergeten was zoals de resultaten met elkaar vergelijken met feiten. Deze sprint zijn voor beide onderzoeken specificaties gezocht en berekent om de resultaten met elkaar te kunnen vergelijken. Op deze manier is er naast de voordelen en nadelen ook een wiskundige ondersteuning die beschrijft welke methode de beste resultaten zal opleveren. Met dank hieraan kan gegarandeerd worden dat de uiteindelijke oplossing de beste oplossing is met harde feiten en niet alleen meningen.

14.1.1 Uitzoeken Zonnepaneel aansluiten op de Raspberry Pi

Voor het aansluiten op het zonnepaneel is in de vorige sprint achterhaald dat er een analoog naar digitaal converter nodig is. Daarbij waren twee concrete manieren gevonden die deze oplossing boden en dat was de ADC-DAC Pi en een circuit met behulp van een condensator. Naast de veronderstelling dat de condensator software afhankelijk is van de Raspberry Pi en een goedkopere methode is, is niet genoeg om te veronderstellen dat die methode beter of slechter is dan de ander.

Hierdoor zijn er berekeningen gedaan om te achterhalen wat de frequentie van de metingen zijn en wat de resolutie is bij deze metingen. De resolutie beschrijft hoe dicht het digitale signaal bij de werkelijke waarde is van het analoge signaal. De resolutie wordt vaak vergeleken met stappen, omdat de resolutie bepaalt hoeveel stappen er zijn om het aantal volt in op te slaan. De ADC-DAC Pi heeft een datasheet waarin de resolutie en de frequentie van de metingen staan. Deze staan niet bij de condensator oplossing, omdat deze oplossing een zelfgemaakt circuit is.

Uit de berekeningen is achterhaald dat de frequentie van de condensator 1kHz is tegenover de 50kHz van de ADC-DAC Pi. De resolutie van de condensator is 4,3mV per stap en de resolutie van de ADC-DAC Pi is 0,73mV per stap. De frequentie van de ADC-DAC Pi is 50 maal zo hoog en de resolutie is bijna 6 maal zo hoog. De prijs is ongeveer 10 euro duurder. De condensator kan niet functioneren als het zonnepaneel onder 4,4V aan energie levert, omdat na de spanningsdeler nog maar 2,2 Volt overblijft. 2,2V is net genoeg voor de Raspberry Pi om een digitaal hoog signaal te kunnen lezen. Er is gekozen voor de ADC-DAC Pi, omdat deze veel hogere frequentie en resolutie heeft. De spanningswaarden zullen veel accurater en frequenter zijn. Daarbij zal de ADC-DAC Pi bij elk spanningsniveau een waarde meten in tegenstelling tot de condensator.

14.1.2 Uitzoeken Testopstelling

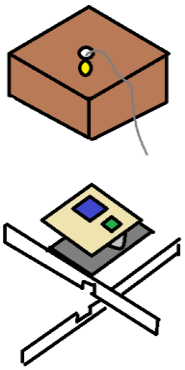
De testopstelling verder uitwerken heeft ook veel tijd ondervonden, omdat de feiten niet op gemakkelijke wijze te verkrijgen waren. Voor elke oplossing in het onderzoek, moest een componenten lijst gemaakt worden zodat er op gemakkelijke en duidelijke wijze oplossingen vergelijkt kunnen worden. Daarbij moesten de prijzen vermeld worden en hoe lang het duurt om het onderdeel te maken, zodat een beeld gemaakt kon worden over de prijs en de tijd die nodig is om het te maken.

Na de componenten lijst is een ontwerp gemaakt per opstelling zie onderstaande paragraaf voor voorbeelden daarvan. Deze ontwerpen zijn gebruikt om te bepalen hoe consistent en controleerbaar de opstellingen zijn. Bij de ontwerpen is ook bepaald of er limieten zijn aan de opstellingen. Deze resultaten zijn vergeleken en hieruit is een keuze gemaakt over welke testopstelling het beste is voor de testopstelling.

14.2 Ontwerpen

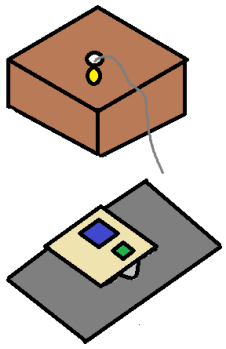
Na de resultaten van het onderzoek is een ontwerp gemaakt voor de testopstelling. Het ontwerp geeft weer hoe het totaal plaatje eruit zal gaan zien. Het ontwerp heeft twee iteraties gehad om een paar kinderziektes weg te werken. Hieronder staan de ontwerpen met als laatste het volledige ontwerp die gebouwd zal worden.

Hier rechts in Figuur 27 is de eerste testopstelling te zien. Deze bevatte planken die in een X vorm binnen in een doos zitten. De planken zorgen dat de opstelling op dezelfde plekken zouden staan. Helaas kon je niet de opstelling goed op zijn plek zetten als de doos op zijn plek stond en kon men niet meer verbeteringen aanpassen zonder het geheel te moeten bewegen. Hieruit is ontwerp 2 ontstaan en dat is het geheel monteren aan een plank waar de doos precies over heen past. Met behulp van een markering aan de rand kan gezien worden welke kant tegen welke rand moet zitten.

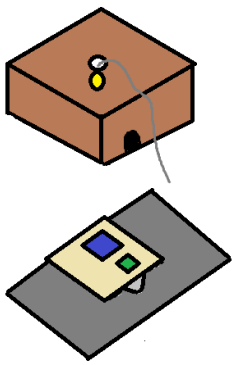


Figuur 27: Testopstelling Ontwerp Iteratie 1

Bij het tweede ontwerp (Figuur 28) zit alleen het probleem dat de kabels onder een rand door heen moeten gewurgd worden. Dit is niet goed voor de bedrading en om dit probleem te vermijden is een klein gat in de zijkant een oplossing. De kabels kunnen hierdoor en het kan makkelijk afgedekt worden met een doek om geen licht door te laten. Naast het gat kan een laptop en een stopcontact neergezet worden om alles aan te sturen. Als de opstelling dicht zit kan niemand meer zien wat er binnen gebeurt totdat de doos opgetild wordt. Het definitieve ontwerp is in Figuur 29 te zien.



Figuur 28: Testopstelling Ontwerp Iteratie 2



Figuur 29: Testopstelling Ontwerp Definitieve Versie

14.3 Bouwen

Deze paragraaf beschrijft het proces bij het bouwen van de testopstelling. De gehele testopstelling kon niet gemaakt worden, omdat onderdelen ontbraken, maar wel kon de doos en het platform klaar gemaakt worden. Tevens is het zonnepaneel klaar gemaakt voor gebruik met simpele bedrading.

Ten eerste is een doos gehaald. Deze is kant en klaar gekocht bij de Xenos voor 15 euro, omdat deze voldoende van formaat is, zeer stevig is en heel veel tijd bespaart. De gaten voor de lamp en de kabels zijn er uitgezaagd met wat hulp van mijn stiefvader, omdat ik de apparatuur niet heb en een trillende hand. Het zonnepaneel is met verschillende lichtbronnen getest om de juiste er uit te selecteren. Het belangrijkste was dat het later mogelijk was om de lamp met een relais te kunnen koppelen om gecontroleerd het licht aan en uit te kunnen doen. Om deze reden zijn naar lage voltage lampen gekeken zoals een fietslamp of 12v lampen uit de winkel. Uit alle lampen kwam op ongeveer 40 centimeter afstand rond de 6 volt uit en om het zo goedkoop mogelijk te houden is een tweede hands fietslamp gekocht. De lamp doet het vrij goed en is makkelijk aan te sluiten met het frame die er al bij zit geleverd. Hieronder is het resultaat van het bouwen te zien. Alle kabels komen uit dezelfde uitgang zodat het zeer makkelijk is alles aan te sluiten op stroom. Verder kan er een doek over de gaten gelegd worden om het licht zoveel mogelijk te blokkeren. Het resultaat van het bouwen is in Figuur 30 te zien.

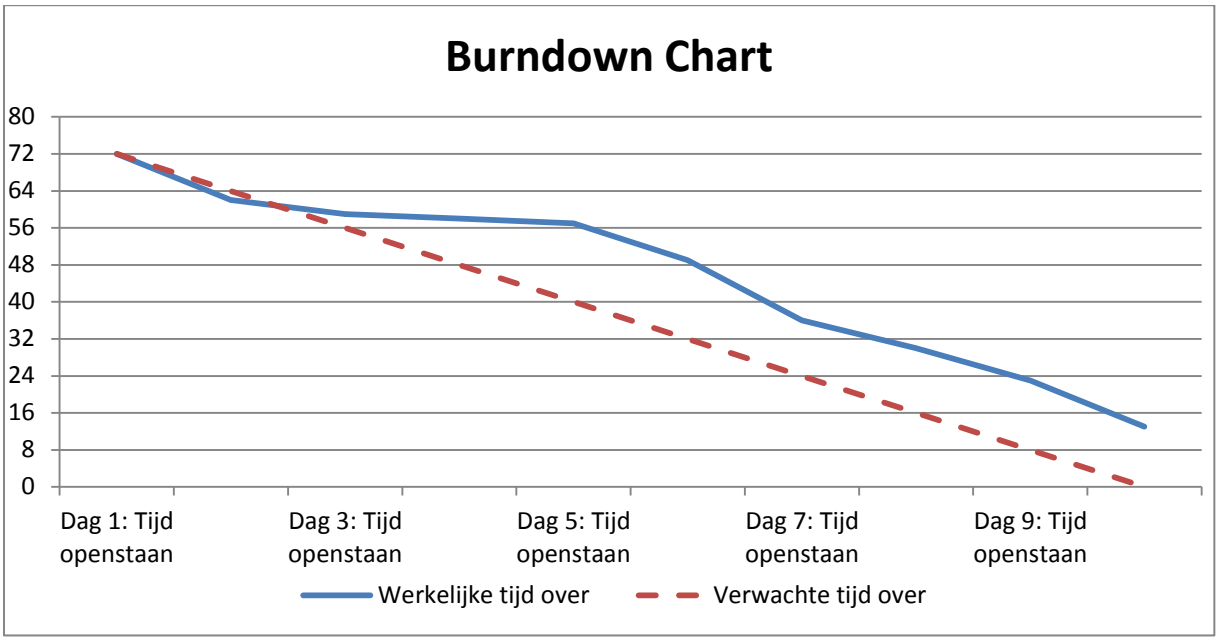
Figuur 30: Testopstelling in de huidige staat, onder de doos zit een plaat met daarop de Raspberry Pi. De kabels kunnen via het gaat naar buiten toe.

14.4 Testen

In deze sprint zijn er geen testen op de software uitgevoerd, maar wel een lichte test op de hardware opstelling. Het was zeer informeel, omdat het alleen functioneerde om te controleren of het zonnepaneel genoeg energie zou opwekken uit de lamp. Het enige dat gedaan werd is de lamp aanzetten en het zonnepaneel meten onder verschillende hoeken ten opzichte van de lichtbron.

14.5 Resultaat en Feedback

Het resultaat van deze sprint is beide onderzoeken volledig afgerond en op basis van de onderzoeken is er begonnen met bouwen aan de testopstelling. Hiervan zijn de structurele onderdelen aanwezig, maar de mechanica voor de aansturing nog niet. In de burndown chart hieronder in Figuur 31 is een weergave van het verloop van deze sprint.



Figuur 31: Burndownchart voor Sprint 5

Uit de burndown chart kan gehaald worden dat rond dag 3 een achterstand begon op te lopen. Deze is enorm opgebouwd over een periode van 2-3 dagen. Daarna is er veel werk ingehaald, omdat de onderzoeken bijna klaar waren en het bouwen van de testopstelling veel minder werk was dan ingeschat. De tijdsinschatting voor de stories was onderschat bij de onderzoeken en overschat bij de testopstelling.

15. Sprint 6

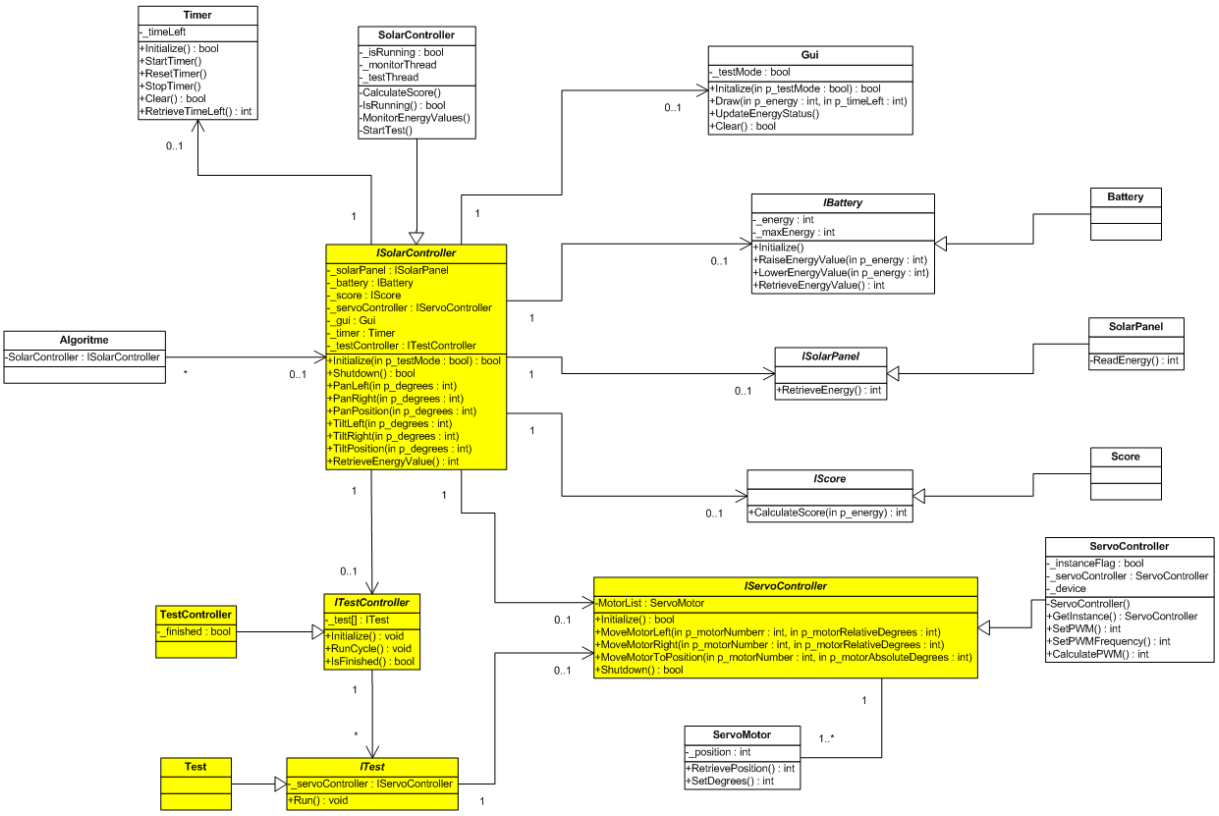
Deze sprint focust zich op het afmaken van de testopstelling, zowel hardware als software. De software zal voornamelijk de aansturing zijn van de testopstelling en het maken van de eerste test waaraan het algoritme zal moeten voldoen. Een deel van de requirements die in deze sprint afgevangen worden staan in Tabel 10 hieronder.

Nummer	Requirement	Prioriteit
F6	Als workshopgever wil ik dat de testopstelling niet hoeft te wachten op andere onderdelen van het systeem.	Must have
F8	Als software developer wil ik dat alle objecten / klassen uitbreidbaar zijn om toekomstig nieuwe klassen toe te voegen.	Should have
F12	Als workshopgever wil ik dat de batterij energie genereert op basis van het zonnepaneel.	Should have
NF12	Als software developer wil ik dat elke klasse een duidelijke rol heeft	Should have
F16	Als tester wil ik gemakkelijk meer tests kunnen toevoegen aan het te ontwikkelen systeem	Could have

Tabel 10: Requirements voor sprint 6

15.1 Ontwerpen

Er komen een paar nieuwe functionaliteiten bij zoals het testen van de algoritmes. Dit wordt gedaan door een aparte klasse de testcontroller. Die bepaalt welke tests gedraaid worden. De tests worden verzameld in een vector en elke test is een reeks van bewegingen die naar de servocontroller verstuurd worden. In Figuur 32 staat de nieuwe klassendiagram voor deze sprint. In de klassendiagram is in geel aangegeven welke klassen nieuwe relaties hebben gekregen en welke klassen nieuw toegevoegd zijn. De witte klassen hebben geen wijzigingen opgetreden en zijn gebleven zoals ze zijn.



Figuur 32: Sprint 6 klassen diagram. Geel zijn gewijzigde klassen en of hebben nieuwe relaties gekregen. Wit is niks aan verandert.

Net als bij de al bestaande klassen is er voor gekozen om de testcontroller en ook de tests onder interfaces te hangen. Hierdoor kunnen meerdere tests gemaakt worden en toegevoegd worden en eventueel ook een nieuwe testcontroller gemaakt worden. Dit is om het programma zo uitbreidbaar mogelijk te houden.

15.2 Bouwen

In deze sprint is er gebouwd aan de hardware voor de testopstelling en aan de software. De testopstelling is verder uitgebreid na vorige sprint. De servomotoren voor de testopstelling zijn toegevoegd en het geheel is gemonteerd aan de ondergrond van de testoptelling. Tevens is ook de Raspberry Pi hierop bevestigd zodat dit geheel af is. Het circuit voor het zonnepaneel met de ADC is af, maar deze is nog niet gemonteerd, omdat niet bekend is hoe dit te doen zonder het paneel te beschadigen of te veel gewicht toe te voegen.

In Figuur 33 is een snippet te vinden van de communicatie met de ADC. Deze communiceert via SPI dat vergelijkbaar werkt als I2C, maar een paar andere aspecten heeft. Bijvoorbeeld dat communicatie heen en terug tegelijkertijd kan plaatsvinden. Om de data te ontvangen moeten we een reeks van bytes sturen, met daarin de commando's maar ook een ruimte waarin de returnwaardes komen. Hiervoor is een buffer gemaakt van unsigned chars, want deze zijn precies 1 byte groot en SPI verstuurt in de meeste gevallen een byte aan grote. De commando's zijn verdeeld in het adres, het commando en een derde deel waarin het antwoord gezet zal worden. Deze wordt verstuurd met SPI en bij terugkomst kan deze uitgelezen worden waarbij in de snippet het antwoord in buffer[2] zou zitten. De communicatie wordt met de hulp van WiringPi voltooid en returned een errorcode die voldoet aan de standaard C errorcodes die gegeven kunnen worden.

```
buffer[0] = 1;
buffer[1] = (1 + 2) << 6;
buffer[2] = 0;
size = 3;

//WiringPiSPIDataRW(int channel, unsigned char* data, int length);
error = wiringPiSPIDataRW(0, buffer, size);
energy = ((buffer[1] & 0x0F) << 8) + (buffer[2]);

return energy;
```

Figuur 33: Snippet van SPI communicatie

In bovenstaande snippet wordt het SPI bericht voor bereid. Dit zijn een reeks van bytes die een in array staan om gezamenlijk verstuurd te worden. De buffer is gevuld met waardes uit de datasheet van het bord. Buffer[1] selecteer het adres met het commando (1 + kanaal) << 6. De draden zijn vast gesoldeerd dus er wordt een vaste waarde gebruikt. Buffer[0] moet een 1 zijn maar waarom kan niet gehaald worden uit de datasheet net als de shift van 6 voor buffer[1]. Buffer[2] is 0 omdat die een deel van de return waarde opneemt.

Het bord dat de waardes leest gebruikt 12 bits totaal en de eerste 4 bits daarvan worden terug gestuurd in buffer[1]. Buffer[2] bevat de overige bits. Beide zijn een byte groot en buffer[1] wordt vandaar eerst bitwise & met 0x0F om alle informatie behalve de 4 bits op 0 te zetten. Verder wordt er naar links geshift met 8, omdat het de eerste 4 bits van een reeks van 12 zijn. Als laatste wordt buffer[2] toegevoegd in zijn geheel, omdat deze de rest van de bits bevat van het gestuurde commando.

Zoals in de vorige paragraaf te zien is, zijn er een paar klassen bij gekomen in de software. De testcontroller zal los draaien van het geheel, omdat de test moet kunnen draaien en tegelijkertijd moet het algoritme kunnen draaien. Zoals daar ook is aangegeven, zijn er interfaces aangehangen zodat de testcontroller en de tests uitbreidbaar zijn met meerdere soorten testcontrollers of tests.

De servocontroller heeft een kleine verandering gekregen, omdat er maar 1 servocontroller voor het gehele systeem nodig is, is er een singleton van gemaakt. Een singleton is een design pattern waarbij een object, maar eenmalig aangemaakt mag worden en elke keer een nieuwe aangevraagd wordt, wordt het bestaande object terug gegeven. Er is een singleton van gemaakt, zodat er niet aparte servocontrollers komen voor de aansturing van de algoritme servomotoren en de test servomotoren, terwijl de aansturing precies hetzelfde is naast een nummer verschil. Daarbij hoeven er niet meer onnodige objecten bijgemaakt worden, zoals extra referenties naar de I2C en WiringPi library.

In onderstaande Figuur 34 is een snippet van de SolarController klasse. Hierin wordt getoond hoe de TestController zich uit en hoe deze gebruikt wordt. De TestController opzich zelf is niet zeer bijzonder want het is een klasse die alle tests uit een lijstje draait en vervolgens een flag omzet. Wat wel belangrijk is, is dat het testen van het algoritme geen onderbreking mag zijn voor het al bestaande systeem voor algoritmes en het monitoren van energie. Dit houdt in dat er wederom een thread wordt gebruik om het testen van algoritmes te onderscheiden van de andere onderdelen.

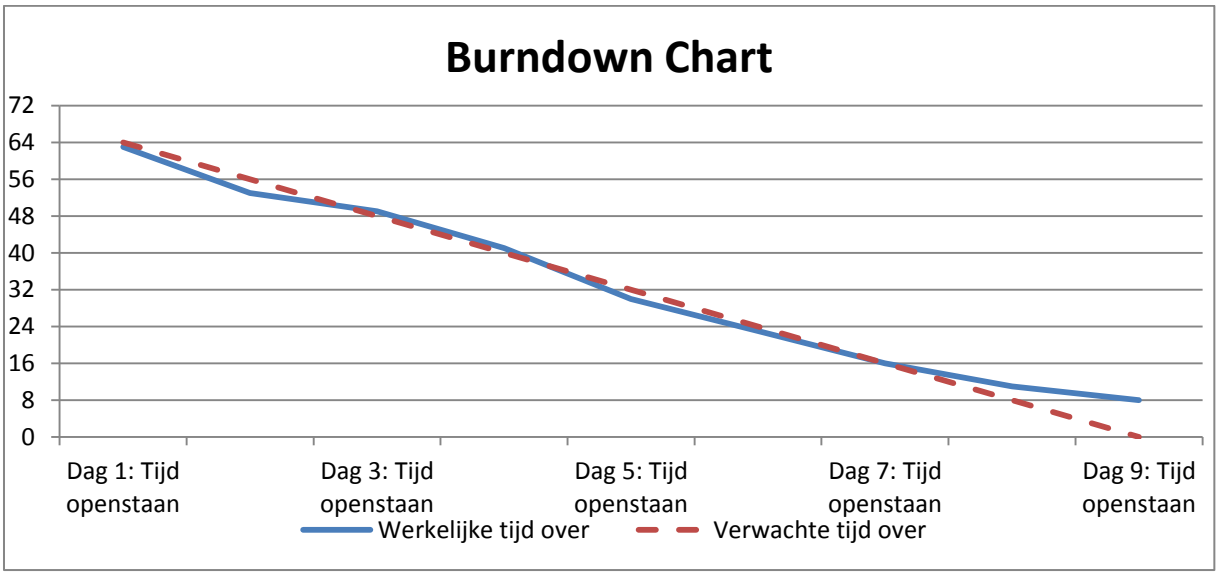
De testcontroller en de thread worden alleen aangemaakt als er daadwerkelijk getest moeten worden. Als voorlopige placeholder oplossing wordt dit nu met een parameter gedaan. Ook een functionaliteit is dat het systeem stop gezet zal worden zodra het beoordelen van het algoritme klaar is, omdat de testopstelling dan zijn functie voltooid heeft en het volgende algoritme klaar gezet mag worden. Aan het einde van de test zal ook automatisch de score berekent worden.

```
bool SolarController::Initialize(bool p_testMode) {
	//*****************************************
		Opstarten van alle klassen om het algoritme te kunnen draaien
	*****************************************//
	if(p_testMode == true){
		this->_iTestController = new TestController();
		this->_iTestController->Initialize();
		this->_testThread =thread(&SolarController::StartTest, this);
	}
	this->_monitorThread = thread(&SolarController::MonitorEnergy, this);
	return true;
}

void SolarController::MonitorEnergy() {
	float energyGain, energyLeft;
	int score = 0;
	Timer* idleTimer = new Timer();
	idleTimer->Initialize(1);
	idleTimer->StartTimer();
	this->_isRunning = true;

	while(this->_isRunning == true)
	{
		energyGain = this->_iSolarPanel->RetrieveEnergy();
		this->_iBattery->RaiseEnergyValue(energyGain);
		energyLeft = this->_iBattery->RetrieveEnergyValue();

		if(idleTimer->RetrieveTimeLeft() <= 0) {
			idleTimer->ResetTimer();
			this->_iBattery->LowerEnergyValue(5);
		}

		if(this->_iTestController != nullptr)
		{
			if(this->_iTestController->IsFinished())
			{
				this->_isRunning = false;
				score = this->_iScore->CalculateScore();
				//std::cout << "Test Finished score: " << score << std::endl;
			}
		}
	}
}
```

Figuur 34: Snippet van de werking van de TestController in de SolarController

15.3 Resultaat & Feedback

Deze sprint heeft vrij goede resultaten opgeleverd, maar niet alles is op tijd afgerond in deze sprint. In Figuur 35 is de burndown chart te zien voor deze sprint.



Figuur 35: Burndown chart Sprint 6

In sprint 6 is het prototype voor de testopstelling afgemaakt en is de functionaliteit voor het testsysteem klaar. Tevens is ook het aflezen van het zonnepaneel afgerond. Helaas is het zonnepaneel nog niet gemonteerd aan de testopstelling en heeft de testopstelling nog een paar kleine punten die gedaan moeten worden. Een voorbeeld hiervan is de binnenkant zwart maken, omdat dit licht absorbeert en niet reflecteert.

Andere stories die bijna af zijn, maar niet af zijn gekomen zijn het maken van de testcyclus en de API handleiding. Er zijn meerdere stories niet afgemaakt, omdat ondanks de tegenslag op dag 4 er veel tijd goed gemaakt konden worden op de latere dagen. Dit is tot een zekere mate gebeurt totdat nog een tegenslag kwam waardoor de testopstelling tijdelijk niet afgemaakt kon worden, omdat een servomotor kapot ging.

Uiteindelijk had op achteraf basis met overleg met de product owner een story weggezet moeten worden, zodat er meer tijd vrij gemaakt kon worden om de andere stories af te maken. Desniettemin zijn de resultaten vrij positief, want er is een werkende prototype, maar deze heeft alleen nog wat toevoegingen nodig en deze zullen plaatsvinden in de opvolgende sprint.

16. Sprint 7

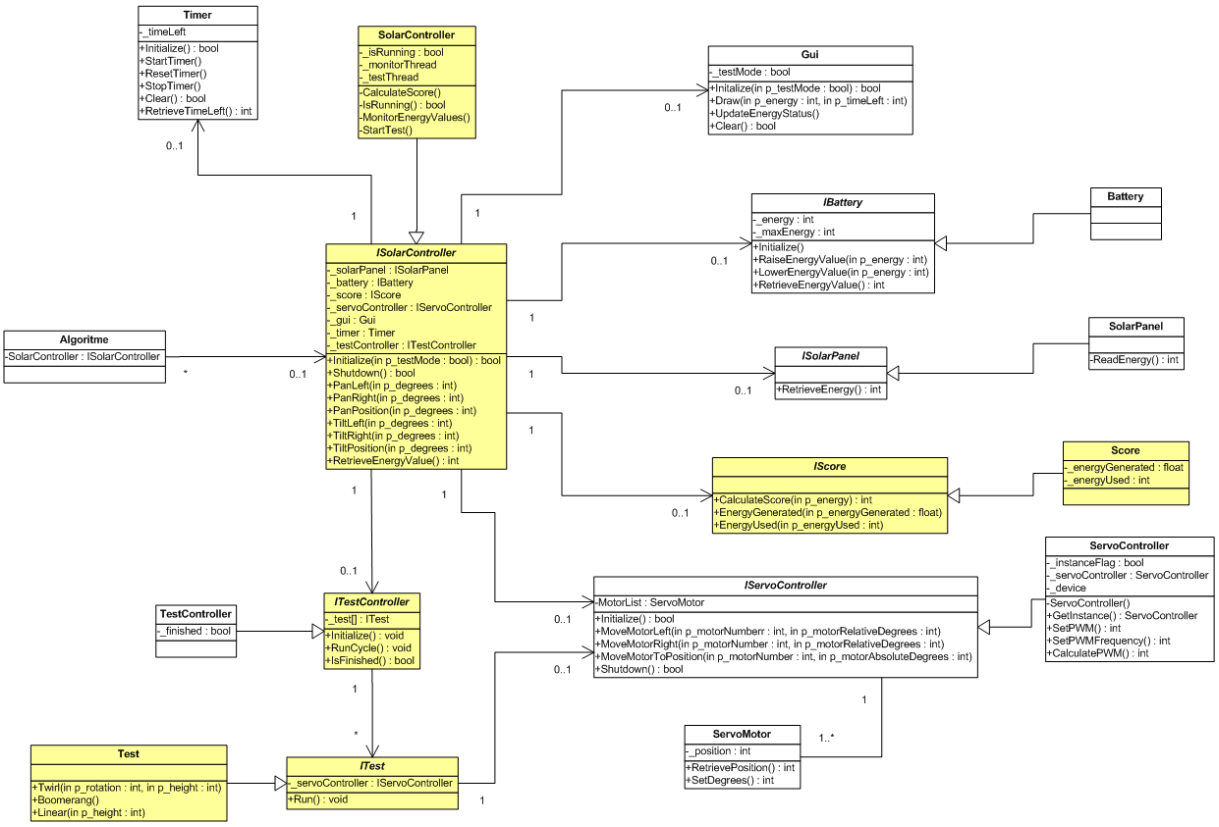
Deze sprint is gericht op het afronden van de testopstelling, het ontwerpen en bouwen van het scoresysteem en het afronden van de testcyclus. In deze sprint is ook ter sprake gekomen dat de grafische interface op een betere manier moet communiceren als dit mogelijk is en ook de score kan hieraan toegevoegd worden. Met overleg met de bedrijfsbegeleider is besloten deze story te beschouwen als een bonus, omdat er veel tijd is gereserveerd om de 80% beoordeling klaar te maken.

In Tabel 11 staat een deel van de requirements afgevangen in deze sprint.

Nummer	Requirement	Prioriteit
F2	Als workshopgever wil ik dat de testopstelling de algoritmes consistent getest kunnen worden.	Must have
F10	Als workshopgever wil ik dat het scoresysteem consistent de scores kan bepalen van de algoritmes.	Should have
NF10	Als workshopgever wil ik dat de testopstelling niet groter is dan 50 bij 50 cm	Should have
F17	Als workshopgever wil ik dat het scoresysteem op basis van verzamelde energie als verbruikte energie de score berekent.	Could have

Tabel 11: Requirements voor sprint 7

16.1 Ontwerpen



Figuur 36: Klassediagram voor sprint 7

In deze sprint zijn voornamelijk bestaande klassen gewijzigd (Figuur 36). De testcyclus moet gemaakt worden en om duidelijkheid te scheppen in de test is de test opgedeeld in een paar aparte functies. Verder is het scoresysteem gebouwd. Deze had al een bestaande structuur en deze is verder uitgewerkt tot het uiteindelijke resultaat.

16.2 Bouwen

Er zijn aan twee software onderdelen gewerkt en de testopstelling heeft ook aandacht gekregen, omdat deze, zoals besproken in de feedback van de vorige sprint, nog een laagje verf nodig had aan de binnenkant.

De testopstelling heeft een laag verf gekregen aan de binnenkant, maar ook aan de buitenkant. De buitenlaag was niet nodig, maar het gaf een extra tint en sfeer aan de opdracht. Er is gewerkt met spuitverf, omdat dit veel sneller werken is en een leuk mistig effect kan geven bij mixen van kleuren. Het eindresultaat is in Figuur 37 te zien.

Figuur 37: Testopstelling na het verven.

Naast het verven is er gewerkt aan de testcyclus en het scoresysteem. De testcyclus is niet heel bijzonder, omdat deze een opsomming is van bewegingen om als geheel een test te vormen. De bewegingen zijn ingedeeld in aparte functies om deze te kunnen hergebruiken. De kern beweging is een boog beweging die in twee vormen komt. Een daarvan is variabel, zodat die makkelijk kan aangepast kan worden. De tweede daarvan doet een soort boomerang effect, waarbij het zonnepaneel een volledige draai maakt en weer terug in het begin punt eindigt. Het is gemakkelijk uit te breiden, maar in de huidige staat kunnen algoritmes voldoende getest worden, omdat de veel voorkomende bewegingen aanwezig zijn. En dat is een vrij consistente boog beweging.

Het scoresysteem is iets complexer, omdat herhaaldelijk dezelfde score behaalt moet worden als de algoritme hetzelfde gedaan heeft. Tevens moet de score bepaalt worden op basis van hoeveel energie er binnen is gekomen en hoeveel er uit is gegaan. Hiervoor zijn de formules in Figuur 38 bedacht.

```
if(this->_energyGenerated > this->_energyUsed)
        {
                score += (this->_energyGenerated / this->_energyUsed) * 10;
                if(score > 100)
                {
                        score = 100;
                }
        }
        else if(this->_energyGenerated < this->_energyUsed)
        {
                score -= (this->_energyUsed / this->_energyGenerated) * 10;
                if(score < 1){
                        score = 1;
                }
        }
        else
        {
                score = 50;
        }
```

Figuur 38: Code snippet voor berekenen van de score

De score wordt berekent op basis van de energie opgevangen (energyGenerated) en de energie die gebruikt is (energyUsed). De score geeft een aanduiding over hoe goed of slecht de algortime is waarbij 50 de default score is en ook het gemiddelde. Hierbij is er evenveel energie opgelevert als verbruikt. Een score van 100 betekent dat er minimaal 5 keer zo veel energie opgeleverd is als verbruikt is. Een score van 1 betekent dat er minimaal 5 keer zo veel energie verbruikt is als opgelevert is. Er is gekozen om het op deze schaal te doen, zodat alle algoritmes die een volle batterij opleveren alsnog een onderscheid kunnen maken in efficiëntie. Tevens gaat het ook om de efficiëntie van de algoritmes en niet hoe vol de batterij eindigt aan het einde. Evenals is het ook vice versa, als meerdere algoritmes een lege batterij hebben is alsnog gewenst om te berekenen welke het minst efficiënt was.

16.3 Testen

In deze sprint is er gebruik gemaakt van error guessing en meer unittesting om de nieuwe onderdelen te testen. Hierbij is het scoresysteem voornamelijk door unittests getest en de testcyclus door error guessing.

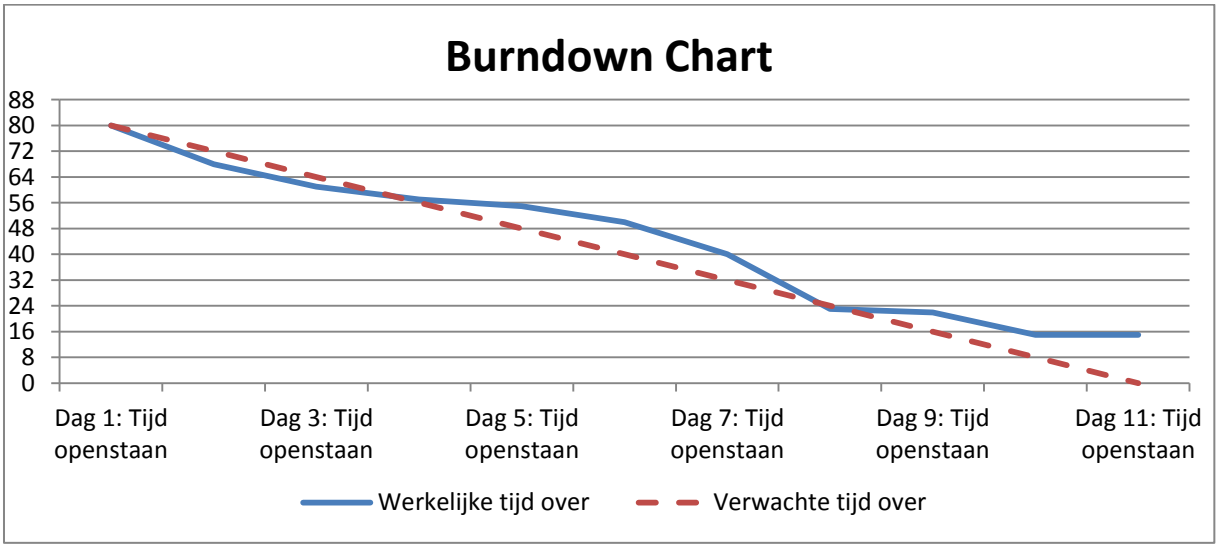
In Figuur 39 volgt een deel van de unit test voor score. In deze test wordt de consistentie van de berekening getest, want deze moet dezelfde cijfers opleveren bij dezelfde omstandigheden. Hiervoor zijn energiewaardes opgegeven en ook het verbruik opgegeven om vervolgens bij twee instanties van de score klasse te controleren wat de uitkomst zijn. De werkelijke uitkomst is handmatig berekent en is 80. De eerste score wordt vergeleken met het werkelijke resultaat en vervolgens met de andere score. Als deze alle drie gelijk zijn is de test goed en anders niet. Dit proces wordt herhaald voor meerdere waardes om te garanderen dat het geen toeval is.

```
calcScore1 = scoreTester1->CalculateScore();
calcScore2 = scoreTester2->CalculateScore();
expScore = 80;
EXPECT_EQ(calcScore1, expScore);
EXPECT_EQ(calcScore1, calcScore2);
```

Figuur 39: Deel unittest van de consistentie unit test voor Score

16.4 Resultaat & Feedback

Het resultaat van deze sprint is zowel positief als negatief. Het positieve is dat het scoresysteem functioneert en bijna de hele testopstelling af is. Het negatieve is dat de tiltmotor voor de testopstelling het gewicht niet meer aan kan. De precieze oorzaak is niet bekend, maar de motor functioneert niet meer naar behoren. De burndown chart voor deze sprint is in Figuur 40 te vinden.



Figuur 40: Burndown chart voor sprint 7

Naast de stories uit de sprint is er ook een hulpprogramma gemaakt om de Servomotoren los aan te kunnen sturen vanuit de commandline. Dit programma is een kleine extra functionaliteit om de motoren makkelijk en ook sneller aan te sturen. Voor de volgende sprint is het duidelijk dat de servomotor die niet functioneert naar behoren gerepareerd of vervangen moet worden. Verder is de grafische interface een ander belangrijk punt dat afgerond moet worden. Verder zijn de handleidingen voor de deelnemers en het project voor de deelnemers om de rest van de sprint op te vullen.

17. Sprint 8

Deze sprint had als doel de testopstelling afronden en de webinterface af te krijgen. Daarbij zijn ook de handleidingen voor de deelnemers en het debuggen toegevoegd aan de sprint. Deze handleiding zullen gebruikt worden gedurende de workshop om de deelnemers op weg te helpen bij het maken en debuggen van hun algoritme code. Dit allemaal om het proces voor hun makkelijker te maken zodat ze meer tijd aan het bouwen van het algoritme over houden.

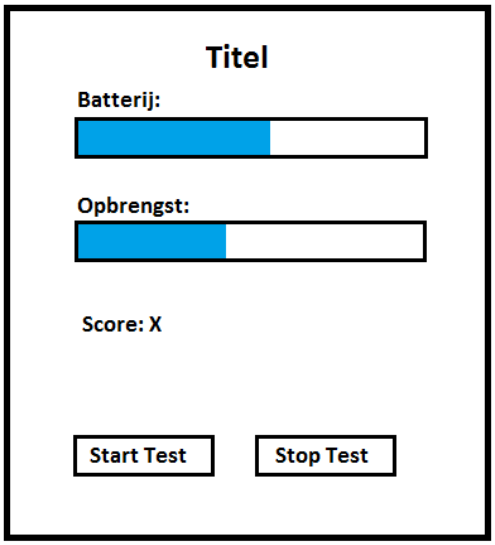
In Tabel 12 volgt een deel van de requirements die in deze sprint zijn afgevangen.

Nummer	Requirement	Prioriteit
NF5	Als workshopgever wil ik dat er een duidelijke handleiding is die beschrijft hoe gebruik gemaakt kan worden van de op te leveren interface	Must have
F14	Als workshopgever wil ik dat de deelnemers visuele feedback ontvangen over de stand van de batterij.	Should have
NF19	Als workshopgever wil ik dat er een duidelijke handleiding is die beschrijft hoe de ontwikkelomgeving geconfigureerd moet worden om te ontwikkelen voor de Raspberry Pi	Should have

Tabel 12: Deel requirements voor sprint 8

17.1 Ontwerpen

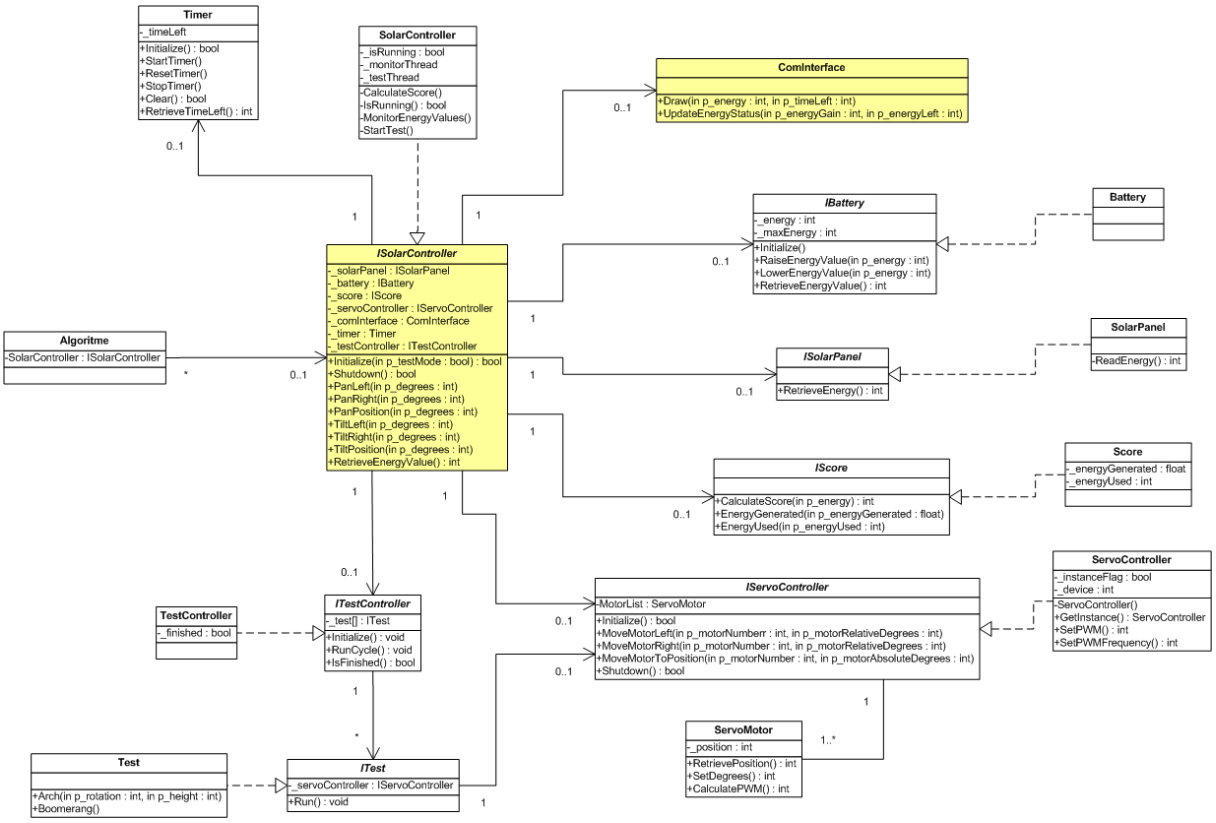
In deze sprint is een ontwerp gemaakt voor de webinterface. De webinterface ontwerpen was een vrij korte stap en het belangrijkste doel ervan was dat het feedback kan leveren aan de deelnemers. Het ontwerp hiervan is in Figuur 41 te vinden.



Figuur 41: Ontwerp voor de WebInterface

Het idee is een simpele en vrij kleine interface, waarbij er 2 balken zijn die representeren hoeveel energie er nog in de batterij zit en hoeveel opbrengst het zonnepaneel heeft. Verder zal het mogelijk zijn om de score te zien. Als laatste is het ook gewenst om via knoppen het programma te kunnen starten en stoppen.

Verder zijn een paar diagrammen gewijzigd zoals het klassendiagram. Deze is in Figuur 42 weer gegeven. Hierbij is de web interface in het midden van alle aandacht geweest, omdat dat het laatste grote onderdeel van het systeem is.



Figuur 42: Klassediagram voor sprint 8. Geel geeft aan de klassen waaraan gewerkt is / gewijzigd zijn.

17.2 Bouwen

Deze sprint ging voornamelijk om de webinterface. Deze is gemaakt in html , php en jquery. Hier is niet heel veel bijzonders aan de hand op een punt na. De communicatie tussen de webinterface en de C++ applicatie is met sockets gedaan. Sockets bieden een laag aan om te communiceren via UDP en TCP. Dit wordt voornamelijk gedaan tussen applicaties op twee verschillende locaties, maar dit kan ook gebruikt worden om op dezelfde computer te communiceren tussen applicaties. Er is gekozen voor sockets, omdat dit een van de weinige manieren is om direct te kunnen communiceren tussen een C++ applicatie en een webinterface op de Raspberry Pi. Er zijn enkele frameworks die deze functionaliteiten ook bieden, maar deze hadden te veel tijd gekost om te implementeren en te kunnen cross compileren.

Hier volgt een snippet van de socket code aan de C++ kant in Figuur 43. De PHP kant van de socket code staat na het C++ deel in Figuur 44.

```
socklen_t addrlen;
struct sockaddr_in address;
string chararray;
ostringstream convert;
if ((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
	{
		return;
	}

address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons(4000);

	if(connect(create_socket, (struct sockaddr *) &address, sizeof(address)) == 0)
	{
		convert << p_energyLeft << "," << p_energyGain;
		chararray = convert.str();

		n = write(create_socket, chararray.c_str() , chararray.length());
	}
close(create_socket);
```

Figuur 43: Snippet socketcode C++ kant

In Figuur 43 is een snippet van de socketcode te zien uit de C++ kant. Een socket is een methode om te communiceren via andere sockets via bijvoorbeeld TCP/IP of UDP. Voor een socket is informatie nodig betreft het IP adres, poort nummer en wat voor type socket het is. De family geeft aan wat voor type het is. Bij het maken van een socket met socket() worden de family, type socket en eventueel het protocol meegegeven. Een AF_INET zal met ip adressen werken en een streaming socket is de standaard gebruikte en deze gebruikt TCP. Het derde argument voor het protocol mag TCP of UDP zijn, maar bij een 0 wordt het standaard protocol gebruikt bij de betreffende type socket.

Vervolgens wordt met behulp van een struct de gegevens van de connectie meegegeven. Meer specifiek zijn dit de gegevens waarmee de connectie gemaakt gaat worden. In de bovenstaande code wordt elk IP adres geaccepteerd in de AF_INET Family en een poort van 4000. Htons(4000) vertaald het poortnummer naar TCP/IP netwerk byte volgorde. Dit is vereist, omdat de socket anders de connectie niet goed kan vastleggen.

Als laatste wordt de data aan elkaar gezet in een string met behulp van een stringstream. De gegevens worden komma seperated achter elkaar gezet om het later gemakkelijk uit elkaar te kunnen halen. Deze data wordt verstuurd met de write() functie. Deze verwacht een kanaal om de data over te sturen zoals een socket, de data die verstuurd moet worden en hoe lang deze dat is.

Voordat de functie afgelopen is wordt de connectie afgesloten.

```php
<?php
	$service_port = 4000;
	$address = “localhost”;
	$data;
	$new_socket;
	$socket = socket_create(AF_INET, SOCK_STREAM, 0);
	If(!$socket) {
		Exit;
	}
	Socket_set_option($socket, SOL_SOCKET, SO_REUSEADDR, 1);

	If(socket_bind($socket, $address, $service_port))
	{
		Socket_listen($socket, 3);
		$new_socket = socket_accept($socket);

		$data = socket_read($new_socket, 8);
		socket_close($new_socket);
	}
	Socket_close($socket);
	$array = explode(	, $data);
	Echo json_encode($array);
?>
```

Figuur 44: Snippet Socket code PHP kant

De socket communicatie aan de PHP kant heeft een vergelijkbaar begin als de C++ kant zoals te zien is in Figuur 44. Eerst wordt de socket gemaakt. Vervolgens wordt een optie ingesteld op de socket dat het adres en port eventueel hergebruikt mogen worden mocht er iets fout gaan en de socket opnieuw gemaakt moet worden.

De socket wordt gebonden aan een adres en dit is het adres waar de c++ client naar toe communiceert. De socket kan pas luisteren naar inkomende verbindingen als deze gebonden is.

Zodra de socket in php gebonden is en de socket luistert naar verbinden kan de communicatie baan geopend worden. Zodra accept aangeroepen is zal de eerst volgende verbinding geacepteerd worden en deze wordt ook opgeslagen. De nieuwe socket kan gebruikt worden voor lezen en schrijven en de oude socket kan gebruikt worden voor het aanmaken van meer verbindingen.

De socket leest data uit met read(). Read vereist de socket als parameter en hoe lang het bericht maximaal mag zijn voordat het afgesneden wordt. Berichten die korter zijn dan de aangegeven waarde komen prima door. De tweede parameter dient eerder als limiet op de grote van het bericht.

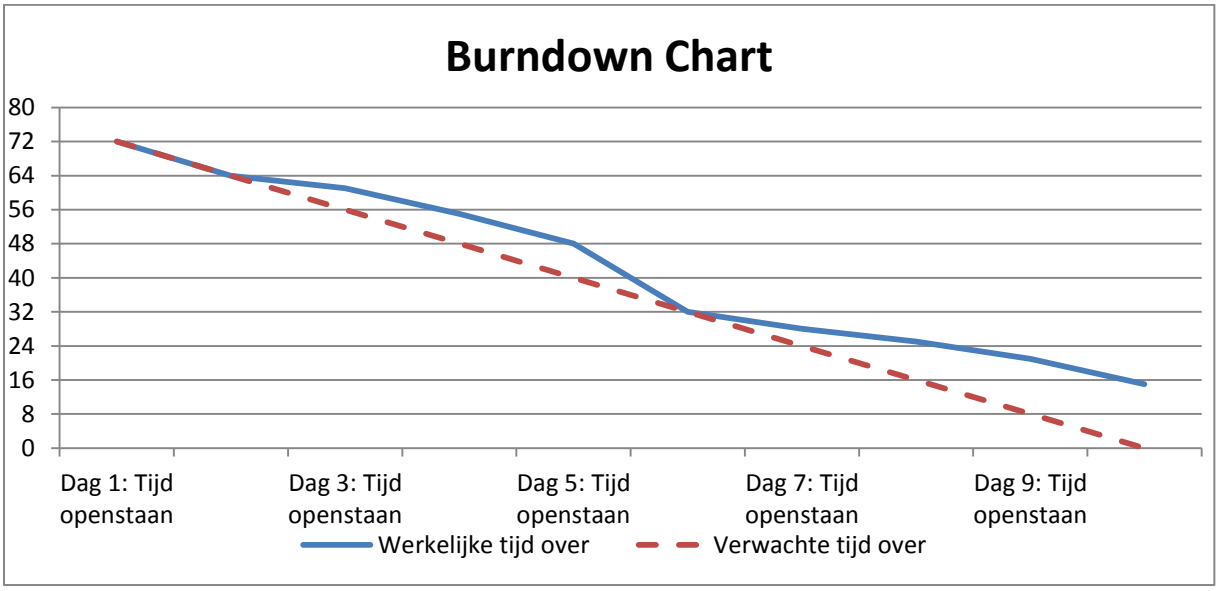
Vervolgens worden de sockets gesloten en kan de data uitgelezen worden. Het bericht is een char * die vertaald wordt naar een string in php, maar deze string bevat informatie voor meerdere onderdelen dus dit moet nog gescheiden worden. Met behulp van explode in php kan een array gebouwd worden uit een string met behulp van een scheidingsteken. Hiervoor is de ‘,’ gebruikt. Om te garanderen dat de data als array goed terug gestuurd wordt naar de Ajax call wordt gebruik gemaakt van JSON. In de Ajax call wordt een Jquery functie aangeroepen die de betreffende onderdelen op de interface update.

17.3 Testen

Er is het een en ander getest rondom de webinterface . Het grootste deel van het testen is met error guessing gedaan, omdat er voor de PHP code in een tekst editor is gewerkt en niet met tooling als notepad++. Aan de C++ kant zijn ook error guesses gedaan met behulp van een apart project. Na een paar revisies is de communicatie werkend gekregen en was het een kwestie van tijd voordat de communicatie werkte zoals gewenst.

17.4 Resultaat & Feedback

Het resultaat van deze sprint is een volledig functioneren testopstelling met een grafische interface om feedback te ontvangen. De interface kan alleen nog niet bestuurd worden door user input. Naast de webinterface is er gewerkt aan de api handleiding, de deelnemerhandleiding en aan het verwerken van de feedback van de 80% beoordeling. Helaas was ik een vrij ruim deel van de sprint afwezig in verband met ziekte. Desondanks is hieronder in Figuur 45 de burndown chart te zien.



Figuur 45: Burndown chart sprint 8

18. Sprint 9

Deze sprint is de laatste sprint van het project. Vanzelf spreken zullen geen grote punten meer langs komen en zal de focus meer liggen op het afronden van wat er nu staat. Dit is de testopstelling met visuele feedback via een webinterface. Verder liggen er handleidingen klaar die beschrijven hoe alles werkt, gebruikt kan worden en hoe het gewijzigd kan worden.

In Tabel 13 volgt een paar van de requirements die in de laatste sprint afgevangen zijn.

Nummer	Requirement	Prioriteit
NF4	Als workshopgever wil ik dat deelnemers direct aan de slag moeten kunnen met algoritmes maken met behulp van de op te leveren producten	Must have
F10	Als workshopgever wil ik dat het scoresysteem consistent de scores kan bepalen van de algoritmes.	Should have
F17	Als workshopgever wil ik dat het scoresysteem op basis van verzamelde energie als verbruikte energie de score berekent.	Could have
NF20	Als workshopgever wil ik een duidelijke handleiding hebben hoe de hardware opstelling opgebouwd is.	Could have

Tabel 13: Requirements voor Sprint 9

18.1 Project afronding

In deze sprint is weinig gedaan aan het project zelf naast de laatste wijzigingen om het plaatje rond te krijgen. Hier onder vallen kleine updates aan het ontwerp, maar ook het scoresysteem aanpassen zodat realistische waardes eruit komen. Een voorbeeld score van 70 ontvangen ook al heeft het algoritme maar 1 beweging gemaakt is verassend hoog en niet zozeer de bedoeling. In deze situatie van uitgaande dat het zonnepaneel veel zon mist zou de score rond de 30 moeten zijn.

Daarnaast had de interface een kleine feature nodig, want deze gaf nog geen scores weer. Gewenst is om ook knoppen te hebben op de interface, maar in verband met tijd is dit achterwege gelaten.

18.2 Hardware componentenlijst

Om de workshop mogelijk te maken worden hardware opstelling gemaakt waarop de deelnemers hun algoritmes kunnen maken en testen. Hiervoor is al een ontwerp gemaakt, maar er is nog geen duidelijke componenten lijst en hoe deze in elkaar gezet moeten worden. Hiervoor wordt een korte handleiding geschreven zodat de hardware opstelling makkelijk na te maken moet zijn. Let op dat dit de hardware opstelling betreft en niet de testopstelling.

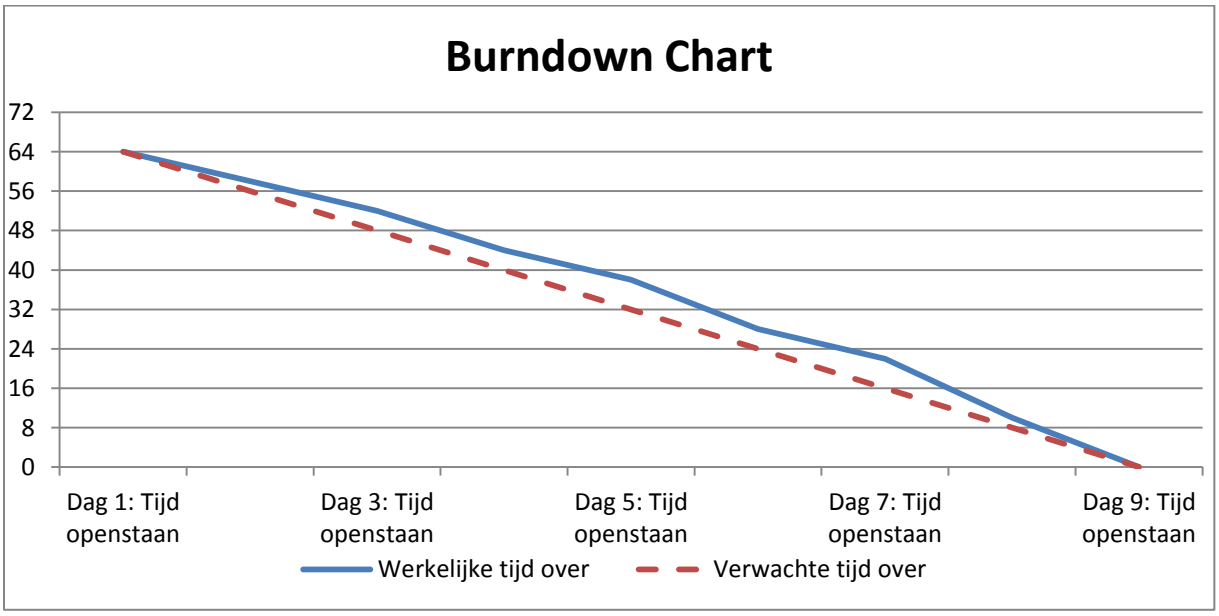
18.3 Final touches testopstelling en scoresysteem

Het testsysteem had kleine bugs van de vorige sprint mee gekregen. Met name is ook gewenst om het duidelijker te maken welke motoren op welke aansluitingen thuishoren, zodat deze makkelijk losgekoppeld kunnen worden en weer aangesloten kunnen worden. Verder is het scoresysteem niet helemaal compleet, omdat de scores die berekend worden wel consistent zijn, maar niet erg realistisch. Eerst kwamen scores bij niks doen uit op bijna 60 tot 70 en dit moet eerder richting de 30 zijn. Hierom is de formule iets gewijzigd, zodat de begin score begin bij 25 en een score van 0 betekent dat de batterij leeggelopen is. Een score van 100 betekent dat de batterij niet alleen volgeladen is, maar dat het algoritme ruim 5 maal zo veel energie oplevert als verliest.

18.4 Resultaat & Feedback

Het resultaat van deze sprint is tevens ook het einde van het gehele project. Voor deze sprint zijn de laatste features toegevoegd zoals de scores weergeven op de user interface. Verder zijn de laatste documenten en onderdelen klaar gemaakt zodat CGI verder kan gaan met het opgeleverde werk. Hier speelt de hardware opstelling een grote rol in, want nu is er alleen een testopstelling. De hardware opstelling is het deel dat de deelnemers ontvangen om hun algoritmes mee te ontwikkelen en te testen. De testopstelling zal de algoritmes beoordelen en niet dienen als opstelling om met iedereen algoritmes te maken.

Hieronder in Figuur 46 is de burndown chart te zien voor deze sprint.



Figuur 46: Burndown chart voor sprint 9

Als laatste sprint is de sprint niet zeer slecht gegaan. Er waren een aantal puntjes om af te ronden betreft het project, maar het grootste deel zat in het scriptie om af te maken voor inleveren. Enkele bijzonderheden was met name hoeveelheid werk er nog in het scriptie zat en de kleine foutjes die ik vond in de code die verbeterd zijn. Er is uiteraard een sprintdemo geweest en een eindgesprek met de begeleiders die over het geheel tevreden zijn ondanks dat het niet volledig af is.

19. Eindresultaat

19.1 Tussenproducten

Plan van aanpak:
Het plan van aanpak beschrijft de aanpak van het project, de planning ervan, de risico's en de kwaliteitsgarantie. Deze is een paar keer verbeterd gedurende het project omdat een paar onderdelen ontbraken of niet volledig waren behandeld.

Requirementsrapport:
Het requirements rapport bevat alle requirements die gevonden en geformuleerd zijn. De requirements zijn geprioriteerd met behulp MoSCoW. Vervolgens zijn deze allemaal in een mapping gekomen naar de verschillende onderdelen van het systeem.

Onderzoeksrapport:
Het onderzoeksrapport bevat de onderzoeken met hun resultaten. Hieronder vallen de onderzoeken naar de servomotoren, hoe deze aan te sluiten, hoe de testopstelling eruit moet zien en welke ADC de beste resultaten oplevert. Een paar van deze onderzoeken zijn veel dieper uitgewerkt dan andere, maar voor elk zijn dezelfde eisen opgesteld waaraan het onderzoek moet voldoen voordat een conclusie getrokken mocht worden.

Ontwerp diagrammen:
De ontwerp diagrammen zijn alle UML diagrammen die gemaakt zijn voor het project. Deze zijn gemaakt met Visio 2003 van microsoft office. Deze bevatten alle use cases met beschrijvingen, klassendiagram , sequence diagrammen en activity diagrammen.

Grafische Interface voor de Raspberry Pi:
De grafische interface is een webbrowser met Jquery, Php en Html op een apache server. De grafische interface is passief en ontvangt alleen data om weer te geven. De C++ library verstuurt informatie via XML naar de webbrowser toe en de browser ververst elke zoveel tellen om de batterijstand ''real time'' weer te geven.

API ten behoeve van de workshop deelnemers:
De API is het deel van library waar tegen de deelnemers spreken. De API is hierdoor een belangrijk deel, omdat het de entree is voor de deelnemers om alle onderdelen te besturen via de SolarController. Deze is zo duidelijk en helder mogelijk gehouden zodat er meer tijd besteed kan worden aan het maken van een goed algoritme dan ontcijferen van een API.

Testopstelling voor de algoritmes:
De testopstelling is de opstelling waarop alle algoritmes getest worden. De testopstelling is gemaakt om consistent en controleerbaar te testen en zal herhalend dezelfde test kunnen uitvoeren per algoritme. De testopstelling is een kleine doos met daarin een lichtbron en een stel servomotoren die vergelijkbaar zijn met de hardware opstelling. De lichtbron kan ten opzichte van het zonnepaneel verandert worden van locatie en het algoritme moet zich hierop aanpassen. Hoe goed deze dat kan wordt beoordeeld met het scoresysteem.

Scoresysteem:
Het scoresysteem beoordeelt de algoritmes op efficiëntie. De algoritmes worden getest met de testopstelling en aan het einde van de test wordt een cijfer bepaald. Het cijfer is gebaseerd op hoeveel energie er verloren gaat aan bewegingen en aan hoeveel energie er gewonnen wordt uit het zonnepaneel. Gezamenlijk kan hieruit een cijfer berekent worden over hoe efficiënt het algoritme was.

Handleiding opbouwen van ontwikelomgeving:
Voor de workshopgevers wordt een handleiding geschreven waarmee ze de ontwikkelomgeving na kunnen maken om te compileren en te deployen voor de Raspberry Pi. De ontwikkelomgeving staat op een virtuele machine, dus die kan gekopieerd worden, maar in het geval dat het nagemaakt moet worden is de handleiding essentieel.

Handleiding Deelnemers:
Voor de workshop wordt een handleiding geschreven die de deelnemers kunnen gebruiken om ze op weg te helpen. De handleiding bevat een lijst van hoe alle functies, kleine tutorial over Eclipse en een tutorial in het deployen en debuggen van hun algoritmes op de Raspberry Pi.

Afstudeerdossier:
Het afstudeerdossier zal gebruikt worden bij de eindbeoordeling. Het afstudeerdossier bevat een verzameling van alle opgeleverde documenten, dit verslag, alle feedback momenten en het rapport van de begeleider.

19.2 Eindproduct

Het eindproduct is bijna een complete workshop onder de naam Raspberry Pi Solar Challenge. De hele testopstelling functioneert naar behoren, de webinterface geeft feedback die regelmatig geüpdatet wordt. Naast de testopstelling zijn er documenten gemaakt die beschrijven hoe alles werkt en opgebouwd kan worden. Verder is er een omgeving gemaakt voor de deelnemers waarin de algoritmes gebouwd kunnen worden en debuggen.

Wat er niet is, zijn de hardware opstellingen en de mogelijkheid om via de interface het systeem aan te sturen met user input. Daarnaast zal de zon gedurende de test altijd branden en in de werkelijkheid kan de zon verloren raken en moet het algoritme deze weer kunnen vinden.

19.3 Proces

Het proces achter de opdracht is vrij goed gegaan. Al is er wel veel begeleiding nodig geweest aan het begin, omdat het wat langzaam op gang kwam. Dit komt omdat ik niet gewent was aan de manier van werken op bedrijfsniveau en omdat er een aantal onderdelen waren die voor mij heel nieuw waren. Hieronder vallen onder andere ook scrum die ik na een sprint of twee echt onder de knie begon te krijgen. Mijn begeleider meneer Bosvelt heeft mij veel geholpen om goed op weg te gaan en voortgang te boeken. Elke sprintdemo heeft hij mij feedback gegeven over de slechte en ook de goede dingen en gedurende sprints heeft hij ook af en toe gekeken hoe alles verliep als hij van te voren wist dat ik vast zou lopen. Daarnaast ben ik zelf ook langs gegaan bij problemen, omdat ik bij sommige berekeningen vast liep door een gebrek aan informatie of iets niet begrijpen.

Het proces is niet helemaal perfect verlopen, want het geheel is niet helemaal af, maar voor deze situatie vond ik het de beste keuze, omdat ik vrij veel onderdelen niet helemaal onder de knie had of nog geen ervaring mee had. Dankzij het proces heb ik deze vrij snel kunnen oppakken en kunnen toepassen waardoor een complete testopstelling staat met een compleet software plaatje. Met een ander proces had het denk ik moeizamer gegaan zijn, omdat het aantal ingeplande feedback momenten een stuk minder zou geweest zijn.

19.4 Hoe kan CGI er mee verder

Om de workshop compleet te maken is op zijn minst een paar hardware opstellingen nodig. Deze hardware opstellingen kunnen gemaakt worden met behulp van de beschrijving die ik opgeleverd heb in de laatste sprint.

Verder is de library erg uitbreidbaar en kunnen zelf gemaakte onderdelen toegevoegd worden voor nog meer type solar challenges met andere besturingen, zonnepanelen, etc. Een leuke uitbreiding is de toevoeging van tests waarbij de zon tijdelijk verdwijnt en weer verschijnt, zodat de algoritmes ook actief moeten zoeken naast alleen volgen.

Wat een toevoeging is waar ik zelf niet aan gekomen ben is user input voor de user interface, zodat de workshop gemakkelijker uit te voeren is voor de workshopgevers. In de huidige opstelling moet alles met een terminal gebeuren en dit kan soms wat onhandig zijn als ook de user interface open moet staan.

Uiteindelijk is het doel om workshops uit te voeren met de opstelling die opgeleverd is en het is mogelijk om al een soort workshop mee te geven, maar dit is vrij gelimiteerd omdat de hardware opstelling nog niet beschikbaar zijn. Als deze gemaakt zijn, is het meer dan mogelijk om workshops te geven omtrent het maken van een algoritme voor een zonnepaneel.

20. Evaluatie

20.1 Aanpak

De aanpak met Scrum is naar mijn mening goed verlopen, omdat er veel geleerd is en goede opleveringen gedaan zijn. Scrum op school en Scrum in het bedrijfsleven waren wel anders naar mijn ervaringen, omdat studenten onderdelen meer door de vingers zien. Het was wel apart om Scrum dat vooral om communicatie draait zelfstandig te doen. De eerste paar sprints waren best moeilijk om het project op te starten, omdat het zo uitgebreid was. De omgevingen klaar zetten, het leren begrijpen van de Raspberry Pi, Scrum onder de knie krijgen, Requirements en Onderzoeken.

Het project was een moeilijke opstart met Scrum, maar ik vind het alsnog de beste keuze, omdat ik veel kon communiceren met mijn begeleider die vrij actief stuurde gedurende het project. Mijn begeleider stuurde aan door bepaalde onderdelen wat te versoepelen of andere stories voorrang gaf over andere. Tevens heeft mijn begeleider her en der even met mij om de tafel gezeten om uitleg te geven over de hardware componenten zodat ik hierover goede beslissingen kon maken. Mijn begeleider heeft daarnaast ook wat tips gegeven over welke richting ik moet zoeken om de juiste informatie te verkrijgen. Hij heeft mij betreft de opdracht niet veel voorgekauwd en eerder gezorgd dat ik nadenk over het een en ander. Er is een uitzondering bij een paar hardware onderdelen, maar dit komt omdat ik er te lang aan bleef hangen en het project weinig vooruit gang kreeg.

Al met al ben ik tevreden over de aanpak, hoewel ik soms veel tegen onderzoeken aan liep, omdat deze vrij uitgebreid en ingewikkeld waren. Dit komt omdat het over onderdelen ging die niet in de studie Informatica zijn behandelt zoals elektrische schakelingen. Scrum hierbij was de beste aanpak, omdat ik veel feedback kon krijgen op een regelmatige basis en dit was een kritisch deel bij een uitgebreide opdracht waarbij een groot deel buiten de comfort zone gewerkt wordt.

20.2 Opgeleverde producten

- Plan van Aanpak
  Het plan van aanpak is een vrij rechttoe rechtaan document. Het enige waar ik hierbij tegen aan liep is dat ik vergeten was de risico's te documenteren. Deze zijn later toegevoegd, maar dat was een sprint of 2 later. Tegen die tijd was er al tegen het een en ander aangelopen waar van te voren beter rekening mee gehouden kon worden, zoals het moeten cross compileren van C++ code naar de Raspberry Pi.

- Requirementsrapport
  Het requirementsrapport is een verzameling van alle requirements. Ik ben tevreden met de hoeveelheid requirements en de mapping van alle requirements. Het is erg makkelijk om te vinden per onderdeel welke requirements hier een rol in spelen. Waar ik veel moeite mee had was het goed formuleren van de requirements en de prioritering goed krijgen. Er zijn enorm veel Should have's en de lagere prioriteiten komen niet veel voor. De reden hiervoor is dat de opdracht vrij uitgebreid is en veel onderdelen nodig heeft om echt goed te functioneren.

- Onderzoeksrapport
  Naar mijn mening iets te veel onderzoek. Bijna alles moest ik onderzoeken of leren, omdat ik het niet wist met uitzondering van de onderdelen geleerd op school zoals het ontwerpen, ontwikkelen en testen. Het rapport zelf is erg uitgebreid door de hoeveelheid onderzoeken en is wat moeilijk leesbaar. Ik ben over het algemeen tevreden, maar soms wat twijfelend over de argumentatie voor de resultaten.

- Ontwerpdiagrammen
  Het ontwerpen van het systeem ben ik trots op, omdat er een ontwerp staat die zeer uitbreidbaar is dankzij het gebruik van Design Patterns. Daarbij vond ik nog leuker om zelfs een tweede design pattern te gebruiken en die zelfs te kunnen combineren met de bridge pattern, want de servocontroller is een Singleton, want de SolarController kent alleen een IServoController en weet verder weinig af van wat er achter speelt. Het zou nog netter kunnen door Factories te gebruiken bij de klassen, maar in verband met tijd is dat achterwege gelaten.

  Wat ik wel beter had willen zien waren de sequence diagrammen, want sommige zijn erg onduidelijk of erg groot door de complexiteit die er achter zit. Dit neemt de gedachte aan dat er functionaliteiten geschoven of verplaatst konden worden, maar ik vind niet dat dat de situatie veel beter zou gemaakt hebben.

- Grafische User Interface voor de Raspberry Pi
  In het begin een slechte en haastig besluit genomen door het GTK++ framework proberen te gebruiken. Al met al veel tijd verspilt aan het proberen te cross compileren voor de Raspberry Pi. Dit is wel gelukt om later uit te vinden dat de nodige functionaliteiten het niet deden. Het is niet bekend of dat door de code kwam, door de cross compiler of door het framework zelf, maar er is overgestapt naar het gebruiken van een webbrowser met Jquery, Php en Html. De website ziet er veel beter uit en is letterlijk een losstaand deel van het gehele c++ systeem waar ik erg blij mee ben. De communicatie tussen beide het c++ systeem en de webinterface is met sockets gedaan. De oplossing is vrij nieuw voor mij maar ik vind het zelf erg mooi om werkend te zien. Het is misschien niet de beste oplossing betreft het regelen van de communicatie, maar in verband met tijd is het goed gelukt.

- API ten behoeve van de workshop deelnemers
  De deelnemers moeten gebruik maken van een API om alle motoren aan te sturen. Er is gedocumenteerd hoe de API werkt en gebruikt moet worden en wat er niet mee gedaan kan worden. De API is prima gelukt en alle functies doen wat er verwacht wordt. Voor de deelnemers is het zo simpel mogelijk gehouden zoals initialisatie, beweegPanmotor, etc.

- Testopstelling voor de algoritmes
  Het is niet het mooiste van de wereld, maar functioneren doet het zeker. Daarom ben ik er alsnog trots op, want als Informatica student met vrij weinig ervaring met microcontrollers, servomotoren en elektronica is er toch een fysieke testopstelling gebouwd. Desondanks het werkt had ik toch liever wat mooiers er uit willen maken, maar hier raak ik aan mijn limiet, omdat ik niet weet hoe ik aan de eisen had kunnen voldoen op andere manieren.

- Scoresysteem
  Het scoresysteem is erg functioneel en beoordeelt de algoritmes vrij consistent. De scores kunnen oplopen tot en met 100 en wordt afhankelijk van de batterijstand berekent, dus ook al is de batterij vol betekent niet gelijk een 100. Het algoritme krijgt een hogere score op basis van hoeveel energie opgeleverd is en hoeveel verbruikt, dus hoe efficiënter energie wordt opgeleverd hoe hoger de score wordt. Ik ben vrij blij met de resultaten betreft het scoresysteem.

- Handleidingen
  De API handleiding alhoewel nog een work in progress vind ik er vrij gestructureerd uitzien. Ik ben daar tevreden over. Maar ik vind de API handleiding erg uitgebreid en dat komt deels door de complexiteit van de cross compiler, maar ook deels omdat er een paar extra onderdelen bij moesten komen die uitlegden hoe de virtuele machine opgesteld moet worden en hoe VNC en SSH werken.

- Afstudeerdossier
  Het afstudeerdossier vind ik erg leuk, maar ook erg uitgebreid. Ik heb veel verschillende documenten en onderzoeken gedaan waarin berekeingen en resultaten staan. Het verslag zelf (dit document) vind ik erg groot en soms wat te groot. Het hoofddeel is ongeveer 60 pagina’s groot en de rest zijn inleidende, afsluitende of aanvullende delen, zoals de inhoudsopgave, bijlages enzovoorts.

20.3 Competenties

Deze paragraaf beschrijft hoe de afgesproken competenties uit het afstudeerplan zijn uitgevoerd. Bij elke competentie wordt beschreven waarvoor de competentie staat en hoe deze gerealiseerd is.

Analyse door definitie van requirements

Deze competentie beschrijft het analyseren met behulp van de definitie van de requirements in een project. Deze competentie is gerealiseerd door een rapport op te stellen met alle requirements betreft het project. Voor de start van het project waren een selectie van requirements al bekend, voordat deze zijn opgenomen in het rapport zijn deze geverifieerd met de begeleider. Naast de bestaande requirements zijn er ook interviews gehouden met de product owner om meerdere requirements te achterhalen.

Er is gebruik gemaakt van MoSCoW om de requirements te priotiseren. Als laatste zijn alle requirements in een mapping gezet naar de onderdelen waarin deze afgevangen moeten worden. Er is gebruik gemaakt van SMART om de requirements te definieren, maar deze zijn later vertaald naar een meer user story formaat. Dit haalde het principe van SMART weg en is uiteindelijk dus geen SMART meer toegepast.

Ik vind dat ik voldoe aan deze competentie, omdat ik een bestaande lijst van requirements heb geanalyseerd en bijgevuld met interviews. Deze zijn met behulp van MoSCoW geprioriteerd. De requirements zijn geformuleerd op basis van het principe: Als (rol) wil ik (Actie) of zeer vergelijkbaar hiermee.

Ontwerpen systeemdeel

Deze competentie beschrijft het ontwerpen van een systeem of een onderdeel ervan en waaraan deze moeten voldoen. Deze competentie is gerealiseerd door gebruik te maken van UML ontwerpen. Voor het project zijn een Use case diagram, klassendiagram, sequence diagram en een activity diagram gemaakt. Deze diagrammen beschrijven elk hun deel over het systeem en hoe het zich gaat gedragen.

Er is bij het ontwerpen van het systeem twee design patterns gebruikt namelijk de Bridge pattern en de Singleton pattern. Deze patronen zijn gebruikt om bepaald gedrag te forceren en om het programma zo uitbreidbaar mogelijk te houden. Bijna alle onderdelen zitten achter een interface waardoor alles zeer uitbreidbaar is. De Singleton pattern is gebruikt om te forceren dat er maar 1 ServoController bestaat. Deze zou kunnen gebruikt worden in meerdere klassen, maar de ServoController werkt iets anders, omdat deze via een library direct de hardware aanspreekt. Om verwarringen te verkomen aan de hardware kant is er maar 1 servocontroller die alle servomotoren aansturen. Deze situatie kan herbouwd worden voor het zonnepaneel, maar deze wordt door maar 1 klassen gebruikt.

Ik vind dat ik voldoe aan deze competentie, omdat ik een uitgebreid ontwerp heb gemaakt met gebruik van de UML standaard. Hierbij zijn usecases gemaakt met use case beschrijvingen, klassendiagram gemaakt, sequence diagrammen gemaakt en een activity diagram gemaakt. Bij het ontwerpen is rekening gehouden met uitbreidbaarheid en er is gebruik gemaakt van de Bridge Pattern om dit te realiseren. Hiernaast heeft elke klasse een duidelijke unieke rol gekregen waarbij uit de naam vrij snel duidelijk moet zijn waar deze voor dient. Er is ook gebruik gemaakt van het

Singleton pattern om te forceren dat er maar 1 servocontroller te allen tijde bestaat om verwarringen in de hardware te vermijden.

Bouwen Applicatie
Deze competentie beschrijft het ontwikkelen en bouwen van een applicatie. Deze competentie is gerealiseerd door het opstellen en configureren van de ontwikkelomgeving met cross compiling en remote acces om te kunnen builden, compileren en te deployen naar de Raspberry Pi.

Gedurende het project is gehouden aan de hongaarse conventie voor programmeren, waarbij alle functies in klassen Uppercamel case zijn, member variabelen beginnen met een underscore (_), parameter variabelen beginnen met p, etc. Om de code leesbaar en duidelijk te houden is het voorzien van commentaar in .cpp files die beschrijven wat elke functie doet.

In de Solar Challenge Library wordt met threads gewerkt die gezamenlijk het geheel vormen. Elke thread dient een eigen functie uit en onderling weten ze weinig van elkaar af. De mainthread houdt rekening met de andere threads om bij te houden wat de situatie is en stuurt deze ook aan. Met behulp van flags in de klassen kan bijgehouden worden wanneer het tijd is dat het programma afgesloten moet worden, zodat de threads niet op elkaar hoeven te wachten. Dit is het gewenste gedrag, omdat de aparte onderdelen in principe altijd door moeten lopen met uitzondering van de testthread, want die heeft een bekend eindpunt.

Naast de Solar Challenge is de GUI gemaakt in JQuery, Php en Html waarmee gecommuniceerd wordt met Sockets. De GUI is een compleet los systeem dat communiceert met de C++ applicatie. De C++ applicatie stuurt actief updates door naar de GUI om realtime feedback te kunnen leveren aan de deelnemers.

Een kleine opsomming van waarom ik vind dat ik voldoe aan deze competentie: Ik heb zelf de volledige ontwikkelomgeving ingericht en geconfigureerd om te kunnen bouwen, compileren en deployen naar de Raspberry Pi. Hiervoor is ook uitzoekwerk gedaan om de beste methode hiervoor te vinden. Naast het opstellen van de omgeving heb ik naar mijn mening een vrij uitgebreid systeem gemaakt die onderling communiceert over de stand van zaken om te bepalen wanneer het programma ten einde mag komen of niet. Om alles duidelijk te houden heb ik mij vast gehouden aan de vastgestelde conventie en heb ik commentaar geleverd bij elke functie om uit te leggen waarvoor het dient en hoe het werkt. Naast alle C++ code, heb ik ook een GUI gemaakt met JQuery, Php en Html waarmee gecommuniceerd wordt in de Solar Controller Library via Sockets.

Uitvoeren en rapporteren over het testproces
Deze competentie beschrijft het testproces, hoe deze uitgevoerd en de rapportage ervan. In dit project zijn een paar testen uitgevoerd, waar naast formele testen ook een informele test is gebruikt. Testen is gedaan met behulp van een testplan waarin staat welke testen uitgevoerd gaan worden en hoe deze uitgevoerd gaan worden. Er is getest met behulp van Unittesten. Naast deze vorm is er ook gebruik gemaakt van Error guessing. De Unittests zijn gemaakt in Eclipse met een plugin en er is gebruik gemaakt van het GTest framework. Deze moesten zelf geconfigureerd worden. Voor bijna alle klassen zijn Unittests gemaakt met uitzondering van de klassen die hardware aanstuurden, omdat ik die tests niet kon draaien buiten de Eclipse omgeving op de ontwikkelcomputer.

Ik vind dat ik voldoe aan deze competentie, omdat ik verschillende vormen van testen heb uitgevoerd op dit project om te controleren of aan de eisen voldaan is. Verder is er gebruik gemaakt van een testplan om op een rij te zetten welke tests plaatsvinden en hoe deze uitgevoerd gaan worden. Ik had meer tests willen uitvoeren, maar door tijdsdruk en stress ook wel angst om het niet te halen is hier niet aan toe gekomen. Als ik de tijd zou hebben gehad zou ik op zijn minst een Intergratietest toevoegen om het systeem als geheel te kunnen testen. Daarbij een Acceptatietest had ook goed geweest om te testen of het test voldoet aan de gewenste eisen.

20.4 Eindevaluatie

Voor dit project vond ik het erg goed gaan, omdat er een erg leuk eindresultaat opgeleverd is namelijk de Raspberry Pi Solar Challenge! Het project was vrij moeilijk, omdat er een aantal onderdelen onbekend waren of buiten de comfort zone lagen zoals elektronica. Het project is niet compleet gegaan als gepland. Het was de bedoeling om verder te zijn in het project tot het punt dat er ook een paar hardware opstellingen klaar zouden zijn. Met het geheel zouden workshops georganiseerd kunnen worden voor interne werknemers of derde partijen om ze uit te dagen een goed algoritme te bedenken voor de zonnepaneel aansturing op een satelliet. Desondanks ben ik tevreden met het resultaat, omdat er een systeem staat waarop algoritmes gemaakt en getest kunnen worden. Er kunnen ook algoritmes op gemaakt kunnen worden, maar het is niet genoeg om een workshop te kunnen geven. Dit komt omdat iedere deelnemer zou moeten wachten op elkaar om een algoritme te kunnen maken / debuggen en dat wordt snel onhandig in grotere groepen.

Dit project ging niet zonder moeilijkheden. Het onderzoek voor de testopstelling vond ik erg ingewikkeld, omdat deze consistent moet zijn en dat bewijzen is moeilijk. Verder vond ik de elektronische circuits ontwerpen en maken moeilijk om te doen, omdat er veel rekening gehouden moest worden met het limiet op de spanning voor de Raspberry Pi. De Raspberry Pi kan niet meer dan 3,3 Volt aan, omdat het anders schade kan oplopen.

Ik heb veel geleerd van dit project, zoals dat ik nog veel te leren heb over het bedrijfsleven en hoe er in te werken, omdat de sfeer en het tempo erg anders zijn dan op school. Ik vond het een leuk project en ik vond het erg goed gaan. Ik weet van mijzelf dat ik halverwege Augustus een burn-out gevoel begon te krijgen en ik had het erg moeilijk om hier door heen te bijten.

Punten die beter gekund hadden, waren het onderzoek en de elektronica delen. Binnen de scope van de studie vind ik dat het opstellen van de requirements beter had gekund en het testen ging goed, maar kon beter. Ontwerpen had ik vanaf het begin beter kunnen doen, maar vrij snel had ik door dat ik design patterns kon gebruiken en heb er ook een paar gebruikt zoals de bridge pattern en veel later ook nog de singleton pattern.

21. Geraadpleegde literatuur

In dit hoofdstuk staan alle bronnen die gebruikt zijn bij de afstudeeropdracht. Alleen de relevante bronnen worden gedocumenteerd waarvan informatie vandaan is gehaald en gebruikt.

21.1 Literatuurlijst

Adafruit pan & tilt kit. (sd). Opgehaald van www.Adafruit.com.

Adafruit tutorials. (sd). Opgehaald van www.Adafruit.com.

ADC-DAC Pi. (sd). Opgehaald van www.ABElectronics.co.uk.

Boost test. (sd). Opgehaald van www.Boost.org.

C++ references. (sd). Opgehaald van www.Cplusplus.com.

GTest. (sd). Opgehaald van www.Code.google.com.

Pan & Tilt van Endurance-rc. (sd). Opgehaald van www.Endurance-rc.com.

Pan & Tilt van Sparkfun. (sd). Opgehaald van www.Sparkfun.com.

Pulse Width Modulation. (sd). Opgehaald van www.Wikipedia.org.

Pulse Width Modulation Tutorial. (sd). Opgehaald van www.Learn.spearkfun.com.

Qt test. (sd). Opgehaald van www.Doc.qt.io.

Raspberry Pi Documentation. (sd). Opgehaald van www.Raspberrypi.org.

Raspberry Pi Forums. (sd). Opgehaald van www.Raspberrypi.org.

RC Charging Circuit. (sd). Opgehaald van www.Electronics-tutorials.ws.

Rc-servos. (sd). Opgehaald van www.rchelicopterfun.com.

Reading analog values from digital pins of Raspberry Pi. (sd). Opgehaald van www.Rpiblog.com.

Servo Pi. (sd). Opgehaald van www.ABElectronics.co.uk.

Servomotor database. (sd). Opgehaald van www.Servodatabase.com.

Setting up a Cross Compiler for Raspberry Pi with Eclipse. (sd). Opgehaald van www.Hertaville.com.

Setting up remote debuggin in Eclipse. (sd). Opgehaald van www.Hertaville.com.

Solar Panel 6V @ 1W. (sd). Opgehaald van www.Parallex.com.

Spanningsdeler. (sd). Opgehaald van www.Wetenschap.infonu.nl.

Using a Cross-compiler for Raspberry Pi in Visual Studio. (sd). Opgehaald van www.Visualgdb.com.

Wiring Pi. (sd). Opgehaald van www.Wiringpi.com.

22. Woordenlijst

In dit hoofdstuk staat een lijst van alle woorden en termen die verwarrend, onduidelijk of als vaktaal worden gezien. De woorden en termen worden uitgelegd voor de lezers die de woorden niet kent of begrijpt.

Woord	Betekenis
Application Programming Interface (API)	Een API is een verzameling van definities die beschrijven hoe een programma kan communiceren met een ander programma of onderdeel. Dit gebeurt meestal in de vorm van bibliotheken zoals DLL of SO bestanden. Een API vormt een scheiding tussen abstractielagen, zodat sommige taken uitbesteed kunnen worden aan het andere programma. Bijvoorbeeld word die niet hoeft te weten hoe een printer bestuurd moet worden en alleen weet dat de printer een document moet hebben om te printen.
ARM Architectuur	ARM staat voor Acorn RISC Machine, waarbij RISC staat voor Reduced Instruction Set Computer. ARM is een veel gebruikte architectuur voor processoren. ARM is een instructieset voor de processor en is erg snel en krachtig.
Bridge Pattern	De bridge pattern is een design pattern. De bridge pattern is om abstractie en implementatie los van elkaar te koppelen zodat deze onafhankelijk van elkaar kunnen variëren.
Cross Compiling	Cross compiling of cross compiler is een compiler die een executabel kan maken voor een platform die anders is dan waar die op gemaakt wordt. Bijvoorbeeld een compiler die op windows 8 draait die een applicatie maakt voor Linux is een cross compiler.
Dynamic Link Library (DLL)	Een DLL is een bibliotheek met functies die door meerdere programma's gebruikt kunnen worden. Het doel van een DLL is dat schijfruimte bespaard wordt omdat de library maar één keer op de schijf hoeft te staan. In tegenstelling tot een statische library waarbij elk programma de library ingebouwd moet hebben.
Eclipse	Eclipse is een opensource framework voor software-ontwikkelomgevingen. Eclipse is het meest bekend om te dienen als omgeving voor Java ontwikkelen, maar Eclipse kan met plug-ins ook gebruikt worden voor bijvoorbeeld C-talen als C++.
Fedora	Fedora is een operating systeem die draait op Linux. Fedora werkt gezamenlijk met RedHat om een goede kwaliteit Operating Systeem op te leveren.
General Pins for Input and Output (GPIO)	De GPIO is de fysieke interface tussen de Raspberry Pi en de buitenwereld. De GPIO pinnen kunnen gebruikt worden om interactie te hebben met van alles in de buiten wereld, zoals sensoren, schermen, servomotoren, led lampen, speakers, etc.
GTK	GTK of GIMP Toolkit is een GUI toolkit die gebruikt wordt voor onder andere GIMP en GNOME. Met GTK kunnen grafische interfaces gemaakt worden op elk platform.
Intergrated Circuit (IC)	Een IC is een samenstel van verschillende elektronische componenten op een stuk halfgeleidermateriaal. Grotere IC's worden vaak chip of microchip genoemd en vormen samen het brein achter elektronische apparaten zoals een wasmachine of koelkast.

Woord	Betekenis
JQuery	JQuery is een JavaScript-framework voor dynamische en interactieve websites. Het is onder andere voor het bewerken van DOM en CSS en AJAX.
PCB	PCB staat voor Printed Circuit Board. In het Nederlands wordt de term printplaat gebruikt. Een printplaat is een drager van elektronische componenten waarbij de bedrading op de printplaat aangebracht zijn.
PWM	PWM of Pulse-Width Modulation is een manier om een elektrisch signaal in de vorm van een blokgolf te sturen. Het signaal gebruikt een vaste frequentie en de golven kunnen variëren. De golf bepaald wat voor signaal gestuurd wordt. Bij aansturen van een led zou die gedimd kunnen worden bijvoorbeeld door het PWM signaal laag te zetten. Zet hem hoog en brand de led weer helder.
Pan & Tilt	Pan en Tilt is een term bekend in de camera wereld. Pannen staat voor zijwaarste bewegen van links naar rechts of vice versa. Tilten is op en neer bewegen.
Raspberry Pi	De Raspberry Pi is een mini computer gemaakt voor educatieve doeleinden. De raspberry pi is speciaal gemaakt om op een goedkope en leuke manier programmeren te leren aan mensen van alle leeftijden.
Remote Acces	Remote Acces is het benaderen van een computer systeem op afstand. Een computer systeem kan bijvoorbeeld een computer, laptop of microcontroller zijn.
Secure Shell (SSH)	SSH is een protocol uit de toepassingslaag van het TCP/IP protocol. SSH maakt het mogelijk om op een beveiligde manier in te loggen op een andere computer en op een afstand commando's en andere taken uit te voeren. Dit kan worden gedaan door middel van een shell ook well command line.
Servo Pi	De Servo Pi is een expansion board voor de Raspberry Pi. Met de Servo Pi kunnen meerdere servomotoren aangestuurd worden (of andere PWM draaiende componenten).
Shared Object (So)	Een shared object is erg vergelijkbaar met de DLL van Windows. Een shared object is een library onder linux die net als een DLL maar eenmalig op de schijf hoeft te staan. Een shared object bevat functionaliteiten die door meerdere programma's gebruikt kunnen worden.
Tool Chain	Een toolchain is een set van programmeer tools. Met deze tools kunnen complexe taken gedaan worden of producten gemaakt worden die niet thuis horen op de machine waar de toolchain op gebruikt wordt.
User Story	Een user story is een korte beschrijving van wat een gebruiker wil.
Virtual Network Computing (VNC)	VNC is een manier om het bureaublad van een computer te delen. Hiermee kan de computer op afstand bediend worden.
XWidget	Xwidget is vergelijkbaar met GTK, omdat het een manier is om grafische interfaces te maken voor desktop en mobiele telefonen.