DE HAAGSE HOGESCHOOL TussenProductenLijst 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. 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
1.0	1 mei 2015	Albert Haaksema	Opzet document en invulling ervan

Tabel 1: Verslielijst tussenproducten lijst.

2. Inleiding

Dit document dient als bijlage gebruikt te worden bij het afstudeerverslag Raspberry Pi Solar Challenge. In dit document staan alle tussenproducten die opgeleverd zijn bij de afstudeeropdracht Raspberry Pi Solar Challenge.

Inhoudsopgave

3. Plan van Aanpak

Risicoanalyse

In dit hoofdstuk worden de risico's behandelt, omdat risico's de loop van het project kunnen beïnvloeden. Het project zal hoe dan ook uitgevoerd worden, maar het is van belang om bewust te zijn van de risico's zodat daar rekening mee gehouden kan worden. De risico's zullen met behulp van een SWOT analyse vast gesteld worden. SWOT staat voor Strengths, Weaknesses, Opportunities en Threats ook wel Sterktes, Zwaktes, Kansen en Bedreigingen in het Nederlands.

Sterktes	**Zwaktes**
- Student wil graag leren - Veel vrijheid - Uitdagend	- Student is niet bekend erg bekend met elektronica - Door geen ervaring in de onderdelen zijn de tijdsschattingen erg misleidend
- Enorm veel ervaring en combineren van verschillende opleidingen - Veel collega's die kunnen helpen - Goede begeleiding	- Opdracht is complex voor de gegeven tijd
Kansen	**Bedreigingen**

De risico's in dit project zijn voornamelijk dat de opdracht complex is en dat de student meerdere onderdelen niet erg goed kent. Hierdoor is er veel tijd nodig om die dingen te beheersen en te kunnen toepassen in de opdracht. Deze risico's worden evenaart door de goede begeleiding, collega's en het enthousiasme van alle delen samen te koppelen tot een geheel. De sterktes en kansen in het project evenaren de zwaktes en bedreigingen.

Aanpak

Voor het project Raspberry Pi Solar Challenge zal gebruik gemaakt worden van Scrum. Scrum is een agile ontwikkelmethode. Scrum werkt met behulp van sprints om het project op te bouwen. Elke sprint is een vaste periode lang. In dit project zullen de sprints 2 weken lang zijn. Elke sprint bevat een deel of meerdere delen van het project waaraan gewerkt wordt. Hierdoor wordt er niet aan te veel dingen tegelijk gewerkt en zijn er meer tussentijdse opleveringen naar de klant. De klant kan hier eerder feedback opleveren en wijzigingen van de requirements kunnen beter door gevoerd worden. Sprint 0 voor dit project wordt de opstart sprint. Deze sprint heet de opstartsprint, omdat alle documentatie voor het opstarten van het project geregeld zal worden. Naast de documentatie zoals het plan van aanpak zal ook het onderzoek naar de raspberry pi en de servomotoren gedaan worden. Tevens zal ook een keuze gemaakt worden voor de hardware die gebruikt gaat worden bij de workshop. Dit is om de nodige voorkennis op te bouwen die nodig is voor het project. Aan het einde van elke sprint worden review momenten gehouden om te controleren hoever het werk is. Tevens wordt er ook besproken wat er in de opkomende sprint gaat gebeuren.

Daily stand up

Naast de sprint reviews zijn er ook dailly standups. De daily standup is een moment waarbij alle leden van het scrum team kort vertellen wat ze gedaan hebben en wat ze gaan doen. Ook is er een moment om verwachte problemen te behandelen. Dit project wordt zelfstandig uitgevoerd dus er is een kleine variatie op de daily standup. Er wordt dagelijks nagelopen wat er allemaal gedaan is en wat er allemaal nog gedaan moet worden. Tevens is er ook een moment om stil te staan bij de moeilijkheden en problemen die verwacht worden voor die dag. Zodoende kan er rekening gehouden worden met problemen en kan er eerder ingegrepen worden voordat het uit de hand loopt.

Sprint Backlog

Per sprint wordt bijgehouden welke onderdelen ook wel stories in die sprint gedaan gaan worden. Deze stories worden allemaal bijgehouden in de sprint backlog. De sprint backlog is erg vergelijkbaar met de product backlog, omdat de stories in principe van de product backlog overgezet worden naar de sprint backlog. Het verschil is dat de stories onder de sprint backlog voor die specifieke sprint zijn bedoelt en de stories in de product backlog een grote verzameling is van alle delen en taken. De sprint backlog is verdeeld in een To-do, In progress, Ready for Review, Reviewed, Done en Verified. In de To-do staan alle stories die open staan om gedaan te worden. Als een lid aan een taak wil beginnen, pakt hij of zij een story van de To-Do lijst en verzet hem naar In progress. In progress houdt in dat er gewerkt wordt aan de stories in die lijst. Deze worden overgezet naar review als ze klaar zijn voor feedback. Als een story klaar is om feedback te krijgen, kan een ander lid de story oppakken en controleren op kwaliteit. Na de review komt de story in Reviewed. Als alles compleet is kan de story afgesloten worden en komt in Done. Om te controleren dat het opgeleverde werk aan de story voldoet is er een verificatie stap. Stories uit de Done stap worden gecontroleerd en geverifieerd en overgezet als alles in orde is en de story hierdoor ook afsluit.

Product backlog

In de product backlog staan alle onderdelen en taken die uitgevoerd moeten worden voor het project. Deze punten worden bij scrum 'story's' genoemd. Per story is er een korte omschrijving van wat er gedaan moet worden. Er zit een prioriteit aan om te bepalen welke story's belangrijker zijn voor het project dan andere. Bijvoorbeeld een kritische story zal eerder gedaan worden, omdat het erg belangrijk is voor het project. Daarnaast bij een lage prioriteit is het minder belangrijk en zal dus later aan gewerkt worden. Tijdens een sprint is er meestal een combinatie van hoge en lage prioriteit story's om de tijd zo goed mogelijk op te vullen. Naast de prioriteit hangt er een tijdsduur aan. De tijdsduur geeft aan het aantal verwachte uren die er nodig zijn om de story af te ronden en klaar maken voor review.

Hieronder volgt een planning voor de eerste sprint en de productbacklog. De planning is niet volledig, omdat er per sprint punten bij komen of af kunnen vallen. Scrum houdt het project flexibel en zet niks vast in de planning, maar vult de planning per sprint op met story's uit de product backlog. Tijdens het project kan de product backlog groter of kleiner worden afhankelijk van de opdrachtgever. Als de opdrachtgever graag een functionaliteit erbij wil hebben of het net anders wil zien, dan kan de backlog daarop aangepast worden.

Sprint 0 (4 weken)
Plan van aanpak
Requirements analyse
Onderzoek naar Raspberry Pi en servomotoren

Sprint 1 (2 weken)
Hardware ontwerpen

Sprint 2 (2 weken)
Hardware opstelling maken
Software ontwerpen

Sprint 3 (2 weken)
Interface voor communicatie met de Raspberry Pi
Communicatie tussen Raspberry Pi en Motor aansturing maken

Sprint 4 (2 weken)
API voor deelnemers

Sprint 5 (2 weken)
Testopstelling maken

Sprint 6 (2 weken)
Scoresysteem

Sprint 7 (2 weken
Handleidingen

Sprint 8 (2 weken)
Sprint 9 (2 weken)
Sprint 10 (2 weken)

Plan van Aanpak	Tijd
Aanpak	
Planning	
Definition of Done	
Kwaliteitsgarantie	

Requirements analyse	Tijd
Requirements controleren	
Interviews bij ontbrekende requirements	
Requirements priotiseren	
Requirements mappen	
Requirements opdelen in verschillende stories met verwachtte resultaten	

Onderzoek Raspberry Pi	Tijd
Onderzoek hoe de Raspberry Pi werkt	
Hoe moet alles aangesloten worden op de Raspberry Pi	
OS installeren op de Raspbery Pi	
Connectie maken met Raspberry Pi via VNC	

Onderzoek Servomotoren
Onderzoek naar servomotoren
Pan Tilt opstelling zoeken en vergelijken
Bestellen servomotoren
Communicatie tussen servomotoren en Raspberry Pi verzorgen

Hardware opstelling	Tijd
Hardware opstelling ontwerpen	
Hardware opstelling maken	

Interfaces maken voor de workshop	Tijd
Programmeeromgeving en Raspberry Pi met elkaar laten communiceren	
API maken om de Raspberry aan te sturen	
Testen van de API	

Testopstelling maken om de algoritmes mee te kunnen testen	Tijd
Onderzoek hoe de testopstelling eruit moet zien	
Ontwerp maken van de testopstelling	
Bouwen van de testopstelling	
Testen van de testopstelling	

Scoresysteem	Tijd
Bepalen welke factoren in welke mate spelen voor het scoresysteem	
Ontwerp maken voor het scoresysteem	
Bouwen van het scoresysteem	
Testen van het scoresysteem	

Opzetten virtuele batterij	Tijd
Virtuele batterij maken	
Batterij kunnen opladen op basis van zonnepaneel	
Batterij energie laten verzien op basis van bewegingen en idle tijd	

Handleidingen schrijven	Tijd
Handleiding schrijven voor de API	
Handleiding schrijven voor het scoresysteem	
Handleiding schrijven voor de testopstelling	
Handleiding schrijven voor het aansluiten van de Raspberry Pi	
Handleiding schrijven voor connectie leggen met de Raspberry Pi	

Scriptie	Tijd
Sprint 0	
Sprint 1	
Sprint 2	
Sprint 3	
Sprint 4	
Sprint 5	
Sprint 6	
Sprint 7	
Sprint 8	
Sprint 9	
Inleiding	
Evaluatie	
Aanpak	
Opdracht	
Woordenlijst	
Bijlage	
Resultaat	

Versiebeheer Repo aanmaken	Tijd
Aanmaken van Repo	
Overleggen met collega voor zijn Repo	

Ondersteuning voor meerdere talen om gebruik te maken van de API	Tijd
Ondersteuning voor C#/.Net	
Ondersteuning voor Java	

Definition of done

In dit hoofdstuk wordt de definition of done vastgesteld. De definition of done is een definitie die beschrijft wanneer een deel of story als af mag worden beschouwt. Het speelt een rol in het project om te kunnen garanderen dat stories echt af zijn en dat onderdelen niet ontbreken. Daarnaast lopen sprints vaak op elkaar over en hierdoor is het dus van belang dat onderdelen echt af zijn voordat aan vervolg onderdelen kan begonnen worden.

Alle story's worden als af beschouwd als ze voldoen aan alle requirements en kwaliteitseisen. Elk onderdeel krijgt een review om na te gaan of er voldaan is aan de eisen en of alles compleet is. Mocht het compleet zijn en het voldoet aan de kwaliteitseisen en requirements dan is het af en kan het onderdeel als done beschouwt worden. Als het onderdeel niet voldoet aan het een of het ander dan moet het onderdeel verbetert worden en opnieuw een review krijgen.

Kwaliteitsgarantie

Bij kwaliteitsgarantie wordt beschreven hoe de kwaliteit van de producten gegarandeerd kan worden. Het is van belang om een goede kwaliteit op te leveren, omdat het eindproduct een workshop is waarbij algoritmes gemaakt moeten worden. De algoritmes sturen motoren aan via de Raspberry Pi. De motoren zitten gekoppeld aan zonnepanelen. Het doel van de algoritmes is om de panelen zo veel mogelijk naar de zon te laten staan. Deze opstelling moet zo efficiënt mogelijk stroom opwekken. Om deze reden is het belangrijk dat de opgeleverde producten van goede kwaliteit zijn, zodat de workshop betrouwbare resultaten oplevert. Om dit te kunnen garanderen wordt er grondig op requirements gecontroleerd. Naast de requirements wordt er gebruik gemaakt van bekende technieken om de kwaliteit per onderdeel te kunnen garanderen.

Voor de ontwerpen die gemaakt gaan worden zal gebruik gemaakt worden van Unified Modeling Language (UML). UML is een modelmatige taal om objectgeoriënteerde analyses en ontwerpen te kunnen maken. UML is vanuit school de standaard taal om ontwerpen en analyses mee te kunnen maken en is het meest bekend bij mij. Hierdoor zal ik deze techniek gebruiken bij het maken van de modellen. UML is een veel gebruikte taal om dit soort taken te kunnen doen en is beschouwd als een soort standaard over de wereld.

Voor het programmeren zal gebruik gemaakt worden van ANSI C. ANSI C is een standaard voor C programmeertalen. Hierbij kan gedacht worden aan type definities voor variabelen. ANSI C maakt het makkelijker voor de programmeur om tussen verschillende compilers te werken. Daarnaast zal er waar mogelijk en nodig de toepassing van design patterns plaats vinden. Design patterns houdt code meer uitbreidbaar en vaak ook overzichtelijker. Om extra kwaliteit aan te leveren zal de code duidelijk genoeg blijven, zodat men uit de code kan zien wat er gebeurt.

Om de functionaliteit van de onderdelen te garanderen zal er getest worden. Tests hebben ook een kwaliteitsgarantie, omdat een slechte kwaliteit inhoudt dat de resultaten minder of zelfs niet betrouwbaar zijn. Om de kwaliteit van de tests te kunnen garanderen zal gebruik gemaakt worden van TestGoal. TestGoal is een veel gebruikt boek op school die gebruikt wordt bij het testen. TestGoal beschrijft hoe je het testen goed kan aanpakken en levert duidelijke resultaten op.

4. Requirements

In dit hoofdstuk zullen alle functional en non-functional requirements terug te vinden zijn. De groepen zijn gescheiden van elkaar om een duidelijk overzicht te houden tussen het functionele en het niet-functionele. De requirements worden gebasseerd op User Stories. User Stories zijn opgebouwd als volgt: Als <Rol> wil ik <Eis> om <Resultaat> te behalen. Een groot deel van de User stories zal zich richten op het eerste deel: Als <Rol> wil ik <Eis>.

Naast de User story indeling zal ook MoSCoW toegepast worden. MoSCoW is afgekort voor Must have, Should have, Could have, Would like to have. Op basis van dit principe kunnen de requirements verdeeld worden in de hiervoor genoemende categorieën waarbij Must have de meest belangrijke punten zijn en Would like to have de minst belangrijke. De requirements zijn geprioriteerd in overleg met de product owner. Op deze manier is een lijst ontstaan van alle requirements op volgorde van prioriteit.

4.1 Functional Requirements

Alle functional requirements zijn hieronder gedocumenteerd. De requirements zijn op volgorde van prioriteit en elke requirement bevat een plek waar deze afgevangen wordt.

Nummer	Requirement	Beschrijving	Afgevangen	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.	Als twee servomotoren elkaar aansturen met een Pan en Tilt systeem ontstaat een soort koepel waarin het zonnepaneel zich in elke positie kan bevinden	Hardware ontwerp, Testopstelling ontwerp	Must have
F2	Als workshopgever wil ik dat de testopstelling de algoritmes consistent getest kunnen worden.	Als de testopstelling consistent test kunnen de algoritmes onderling vergeleken worden	Testopstelling ontwerp	Must have
F3	Als workshopgever wil ik dat de testopstelling de algoritmes controleerbaar test	Controleerbaar houd in dat er volledige controle is over wat er gebeurt en wanneer	Testopstelling ontwerp	Must have
F4	Als workshopgever wil ik dat de testopstelling de beweging van satelliet ten opzichte van zon kan simuleren	De satelliet hier is de hardware opstelling	Testopstelling ontwerp	Must have

Nummer	Requirement	Beschrijving	Afgevangen	Prioriteit
F5	Als deelnemer wil ik dat het algoritme niet hoeft te wachten op de rest van het systeem		Software ontwerp	Must have
F6	Als workshopgever wil ik dat de testopstelling niet hoeft te wachten op andere onderdelen van het systeem.		Software ontwerp	Must have
F7	Als workshopgever wil ik met VNC remote acces hebben tot de Raspberry Pi tijdens de workshop	VNC is een pakket geadviseerd door de Raspberry Pi foundation om remote de desktop van de Raspberry Pi te kunnen gebruiken	Bouwen hardware opstelling	Should have
F8	Als software developer wil ik dat alle objecten / klassen uitbreidbaar zijn om toekomstig nieuwe klassen toe te voegen.		Software ontwerp	Should have
F9	Als software developer wil ik dat de algoritmes blackbox met het systeem communiceert	Blackbox houdt in dat het algoritme alleen de functies weet en geen achterliggende informatie kent.	Software ontwerp	Should have
F10	Als workshopgever wil ik dat het scoresysteem consistent de scores kan bepalen van de algoritmes.	Het scoresysteem moet herhaadelijk dezelfde score opleveren voor dezelfde test.	Software ontwikkeling, Score bepaling	Should have
F11	Als workshopgever wil ik dat er een virtuele batterij gebruikt wordt om een werkelijke batterij te simuleren		Software ontwerp	Should have
F12	Als workshopgever wil ik dat de batterij energie genereert op basis van het zonnepaneel.	Hoe meer energie het zonnepaneel opwekt hoe meer de batterij oplaad.	Software ontwikkeling, Software ontwerp	Should have

Nummer	Requirement	Beschrijving	Afgevangen	Prioriteit
F13	Als workshopgever wil ik dat de virtuele batterij energie verliest op basis van de bewegingen van de servomotoren	De batterij verliest energie bij bewegingen van de motoren en dit moet gesimuleerd worden.	Software ontwikkeling, Software ontwerp	Should have
F14	Als workshopgever wil ik dat de deelnemers visuele feedback ontvangen over de stand van de batterij.	Feedback voor de deelnemers kan hulpzaam zijn bij het begrijpen wat de stand van zaken is tijdens het testen van de algoritmes	Software ontwerp, Keuze visuele feedback, Software ontwikkeling	Should have
F15	Als workshopgever wil ik dat de testopstelling minimaal 5V spanning kan generen uit het zonnepaneel.		Hardware ontwerp	Could have
F16	Als tester wil ik gemakkelijk meer tests kunnen toevoegen aan het te ontwikkelen systeem		Software ontwerp	Could have
F17	Als workshopgever wil ik dat het scoresysteem op basis van verzamelde energie als verbruikte energie de score berekent.		Score bepaling	Could have
F18	Als deelnemer wil ik de mogelijkheid hebben om de op te leveren interface te gebruiken uit andere talen zoals C# en Java	Dit zou dan alleen betrekken tot het deel dat waarmee het algoritme gaat communiceren	Software ontwerp, software ontwikkeling	Would like to have
F19	Als workshopgever wil ik dat het scoresysteem ook de complexiteit van het algoritme beordeelt.	Hoe complexer het algoritme hoe meer tijd nodig is voor berekeningen.	Score bepaling, Software ontwerp	Would like to have

4.2 Non-Functional Requirements

Alle non-functional requirements zijn hieronder gedocumenteerd. De requirements zijn op volgorde van prioriteit en elke requirement bevat een plek waar deze afgevangen wordt.

Nummer	Requirement	Beschrijving	Afgevangen	Prioriteit
NF1	Als workshopgever wil ik dat elk onderzoek op zijn minst drie resultaten met elkaar vergelijkt om een beeld te krijgen uit meerdere hoeken		Onderzoeksrapport	Must have
NF2	Als workshopgever wil ik dat de API voor de deelnemers gemakkelijk te gebruiken is.		Software ontwerp, Hardware ontwerp	Must have
NF3	Als software developer wil ik dat er een duidelijke code conventie gebruikt wordt		Software ontwikkeling	Must have
NF4	Als workshopgever wil ik dat deelnemers direct aan de slag moeten kunnen met algoritmes maken met behulp van de op te leveren producten		Hardware ontwerp, Software ontwerp, Software ontwikkeling	Must have
NF5	Als workshopgever wil ik dat er een duidelijke handleiding is die beschrijft hoe gebruik gemaakt kan worden van de op te leveren interface		Handleiding deelnemers, Handleiding API	Must have
NF6	Als workshopgever wil ik dat het onderzoek rekening houdt met een beperkt budget van maximaal 75 euro.	De hardware zal meerdere malen aangeschaft moeten worden voor de workshop	Onderzoeksrapport	Should have
NF7	Als workshopgever wil ik een duidelijk advies ontvangen over de resultaten van het onderzoek om de beste aankopen te kunnen doen	Uit gevolg van de onderzoeken zullen onder andere spullen gekocht worden (hardware onderdelen)	Onderzoeksrapport	Should have

Nummer	Requirement	Beschrijving	Afgevangen	Prioriteit
NF8	Als software developer wil ik dat de Raspberry Pi gebruik maakt van Raspbian	Raspbian is een veel gebruikt OS voor de Raspberry Pi en wordt ook geadviseerd door de Raspberry Pi foundation.	Hardware ontwerp / configuratie	Should have
NF9	Als workshopgever wil ik van te voren alle hardware klaar kunnen zetten met behulp van een checklist.		Handleiding hardware opstelling, Handleiding testopstelling	Should have
NF10	Als workshopgever wil ik dat de testopstelling niet groter is dan 50 bij 50 cm	De testopstelling moet makkelijk op te bergen zijn.	Testopstelling ontwerp	Should have
NF11	Als software developer wil ik dat er design patterns toegepast worden om de software uitbreidbaar te maken.	Op een later moment na de opdracht moeten meer objecten toegevoegd kunnen worden zonder bestaande objecten te wijzigen.	Software ontwerp	Should have
NF12	Als software developer wil ik dat elke klasse een duidelijke rol heeft		Software ontwerp	Should have
NF13	Als workshopgever wil ik de testopstelling kunnen herbouwen met behulp van een handleiding	Voor het geval dat hij herbouwt moet worden is een handleiding nodig	Handleiding testopstelling	Should have
NF14	Als software developer wil ik dat er voor de keuze van de ontwikkelingomgeving rekening gehouden wordt met de manier van communiceren met de Raspberry Pi	De communicatiemethode zal ook gebruikt worden tijdens de workshop	Uitzoeken ontwikkelomgeving	Could have
NF15	Als workshopgever wil ik dat de hardware opstelling onder de 5kg weegt.	Als de hardware te zwaar is, is het niet makkelijk mee te verplaatsen naar andere locaties toe	Hardware ontwerp	Could have
NF16	Als workshopgever wil ik dat de hardware opstelling niet groter is dan 40 bij 40 cm.	Als de hardware opstelling te lang / breed is, is deze moeilijker op te bergen	Hardware ontwerp	Could have

Nummer	Requirement	Beschrijving	Afgevangen	Prioriteit
NF17	Als software developer wil ik dat de library in C++ gemaakt wordt.		Ontwikkelomgeving, Software Ontwikkeling	Could have
NF18	Als software developer wil ik een handleiding die beschrijft hoe alle achterliggende logica werkt.	Deze handleiding is niet voor de deelnemers, maar om te begrijpen wat er gebeurt in de achtergrond.	Handleiding Logica	Could 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		Testopstelling handleiding, Hardware handleiding, Ontwikkelomgeving handleiding	Should have
NF20	Als workshopgever wil ik een duidelijke handleiding hebben hoe de hardware opstelling opgebouwd is.		Hardware ontwerp	Could have
NF21	Als software developer wil ik dat de multilanguage support onderdelen goed gedocumenteerd worden om dit eventueel uit te breiden naar andere talen toe.		Software ontwerp	Would like to have

5. Onderzoeksresultaten van servomotor onderzoek

In dit hoofdstuk wordt het resultaat van het onderzoek beschreven. Vervolgens worden de alternatieven gegeven op het resultaat. De alternatieven zijn oplossingen die ook prima kunnen functioneren, maar niet aan de eisen of de scope van het onderzoek hielden.

Pan Tilt	Formaat	Geadviseerde Servomotor	Kracht	Totale kosten
1.Sparkfun	Base: 30mm x 50mm Tilt platform: 30mm x 50mm Hoogte bij geen tilt: 60mm	SubMicro Servomotoren onder andere van sparkfun	4.8V: 16.6 oz-in / 1.195 kg-cm 6.0v: 20.8 oz-in / 1.49 kg-cm	Pan Tilt opstelling: $5,95 Servomotoren: $8,95 Totale set $5,95 + 2x $8,95 = $22,85. 20,88 Euro
2. Endurance-Rc	Base: 75.6mm x 51mm / 3.1 x 2.3 inches Tilt platform: 38mm x 70 mm / 1.5 x 2.878 inches	Hitec HS-311	4.8V: 42.0 oz-in / 3.02 kg-cm 6.0v: 48.6 oz-in / 3.50 kg-cm	$55 voor een de complete set en al in elkaar gezet. $30 dollar voor de pan tilt set en 2x $12-$13 dollar voor de servo. 50.26 Euro
3. Adafruit	Base: 37mm x 33mm x 3mm Hoogte bij geen tilt: 67mm Tilt platform: 38mm x 36mm	SG-90 of SG-92	4.8V 25 oz-in / 1.80 kg-cm	$18.95 voor complete in elkaar gezette set. +- 17,32 Euro
4. DFRobot	Totaal: 80 mm x 60 mm x 60 mm	DF15MG standaard servo	4.8V: 10 kg-cm 6.0V: 12 kg-cm 7.2V: 15 kg-cm	$39.00 voor de complete set. Deze is niet in elkaar gezet en moet dus zelf gedaan worden. 35,64 Euro
5. Dagu Pan Tilt Kit	?	?	4.5V: 1.2 kg-cm 6.0V: 1.4 kg-cm	12 Britse Pond voor de complete set. Deze moet nog in elkaar gezet worden. 16,75 Euro
6. Bizoner Pan Tilt Kit	?	Hitec HS-311	4.8V: 42.0 oz-in / 3.02 kg-com 6.0V: 48.6 oz-in / 3.50 kg-cm	$50 dollar voor de complete set. Onbekend of deze in elkaar gezet moet worden of niet, maar er is geen assembly guide beschikbaar. 45,69 Euro

Pan Tilt	Voordelen	Nadelen
1. Sparkfun (te zwak, in bepaalde hoeken gaan de servo's trillen vanwege het gewicht)	- Ongeveer 20 euro voor een prima werkende set. - Vrij klein formaat en is dus makkelijk te verplaatsen	- Servomotoren draaien maar ongeveer 160 graden - Moet in elkaar gezet worden - Tilt platform klein beetje aan de kleine kant, maar het moet te doen zijn.
2. Endurance-Rc	- Goede hoeveelheid kracht - Hoeft niet in elkaar gezet te worden - Genoeg ruimte voor zonnepaneel	- Vrij duur in vergelijking met de rest. - Moet geïmporteerd worden.
3. Adafruit	- Goedkoop voor complete set - Voor het formaat van de motoren hebben ze aardig wat kracht - Zeer mobiel door formaat en gewicht	- Vrij klein van formaat - Zonnepaneel aardig wat groter dan tilt platform - Tilt platform staat in nul-positie niet recht omhoog
4. DFRobot	- Genoeg kracht beschikbaar - Goede verhouding prijs-kracht	- 150 graden beschikbaar voor pan en tilt in plaats van 180 graden - Moet in elkaar gezet worden
5. Dagu Pan Tilt	- Licht - Klein - Goedkoop, maar niet veel informatie beschikbaar	- Formaat onbekend en servomotor onbekend - Weinig informatie beschikbaar over de kit - Twijfel aan de hoeveelheid kracht
6. Bizoner Pan Tilt	- Voldoende kracht beschikbaar - Nulpunt van de Tilt staat recht omhoog	- Formaat onbekend - Vrij duur

Een goedkope maar risico volle keuze is de optie van Adafruit. De SG-90 servomotoren zouden sterk genoeg moeten zijn, maar die zitten erg op de grens. De prijs is ideaal en het is van te voren al in elkaar gezet. Een wat vervelender nadeel is dat de opstelling op een schuin platform moet staan om ervoor te zorgen dat het tilt platform in de nul-positie recht omhoog staat.

Om zekerder te zijn van de keuze raad het onderzoek de opstelling van DFRobot aan. Deze kan helaas alleen 150 graden draaien, maar hij is meer dan sterk genoeg en de prijs is ongeveer het gemiddelde van alle opties. Het in elkaar zetten is alleen wat schroeven en bouten.

De meest ideale is die van Bizoner. De nul-positie kan recht omhoog gezet worden. Heeft voldoende kracht, niet zoveel als de DFrobot maar meer dan genoeg voor het project. Het is helaas onbekend of hij compleet aankomt of in elkaar gezet moet worden en hij is vrij duur. Tevens is het formaat niet bekend, maar de servomotoren zijn van standaard grote dus de gehele opstelling zal in ieder geval niet klein zijn. Hierdoor kan aangenomen worden dat er genoeg ruimte voor het zonnepaneel is.

Om alle zekerheid te hebben en om het compleet en al aangeleverd te krijgen kan de opstelling van endurance rc gekocht worden. Hij is vrij prijzig, maar is wel compleet. De servomotoren zijn sterk genoeg om het zonnepaneel aan te sturen. Daarnaast hoeft het systeem niet in elkaar gezet te worden en een leuke bonus dat de opstelling een mooie case heeft. Tevens is er meer dan genoeg ruimte om het zonnepaneel te monteren.

De pan en tilt opstelling van Adafruit is geselecteerd, omdat deze bijna een volledige bol kan maken. De opstelling is goedkoop en zou sterk genoeg moeten zijn om het zonnepaneel te kunnen dragen. Daarnaast is hij erg licht en mobiel waardoor deze makkelijk verplaatst kan worden. De nadelen zijn dat er kleine modificaties moeten gedaan worden om het zonnepaneel vast te maken aan het tilt platform. Daarnaast moet het gehele systeem op een helling gezet worden zodat de nul positie recht om hoog staan en niet scheef staat.

5.1 Onderzoeksresultaten aansluiten servomotoren op Raspberry Pi

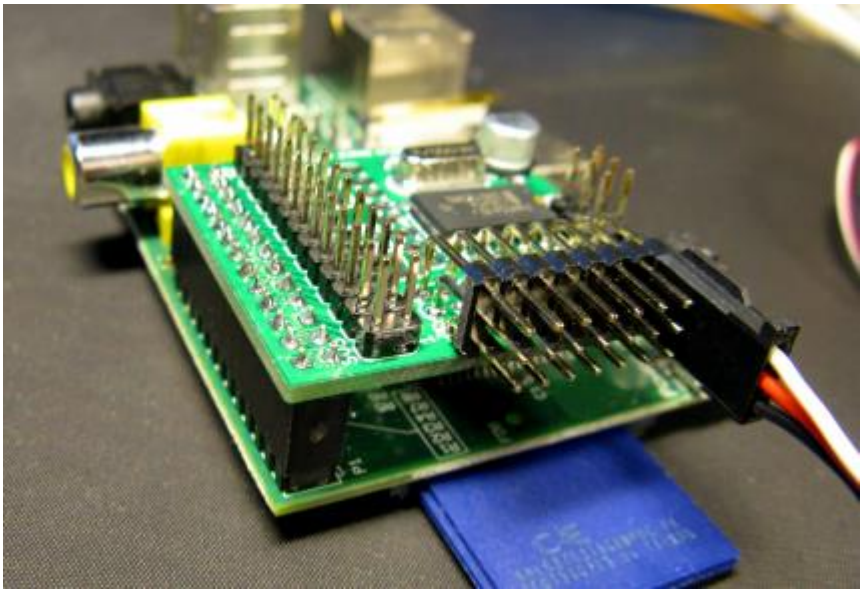
In dit hoofdstuk wordt het resultaat van het onderzoek beschreven. Vervolgens worden de alternatieven gegeven op het resultaat. De alternatieven zijn oplossingen die ook prima kunnen functioneren, maar niet aan de eisen of de scope van het onderzoek hielden.

De makkelijkste oplossing is om de servomotoren direct aan te sluiten aan de Raspberry Pi. Helaas is deze optie niet mogelijk, omdat de Raspberry Pi niet genoeg stroom om beide zichzelf en de servomotoren draaiende te houden. De uitzondering hierop zijn kleine microservomotoren, maar dan nog is een externe batterij of powersupply aangeraden. Zodoende is de makkelijste manier de servomotoren aan externe powersupply te zetten en het pwm signaal aan de Raspberry Pi.

De meest veilige oplossing is de toevoeging van een uitbreidingsbord. Er zijn meerdere borden gemaakt die als uitbreiding dienen voor de Raspberry Pi met verschillende functionaliteiten. Een van die functionaliteiten is het aansturen van servomotoren. Hieronder is een lijst van de verschillende borden die gevonden zijn en vergeleken worden.

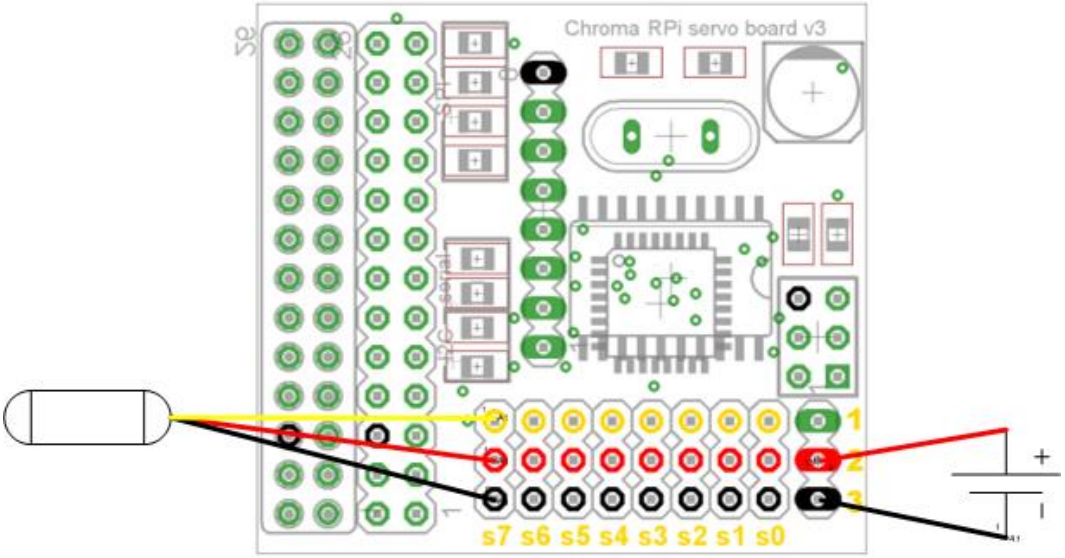
Uitbreiding	Kosten	Kenmerken	Voordelen	Nadelen
RasPiO Breakout Pro	14.16 Euro 9.95 Pond	- Mannelijke en Vrouwelijke aansluitingen	- Beveiliging voor Raspberry Pi - Geen stroom nodig	- Geen PWM ingebouwd
Chroma Servoboard	13.85 Euro 15 Dollar	- ASCII bestuurt het bord.	- 8 Servo aansluitingen - Seriële aansluiting met de Raspberry Pi	- De gegeven posities is van -1000 tot 1000 voor 0 en 180 graden.
Chroma Software Servoboard	18.46 Euro 20 Dollar	- Gebruikt Servoblaster kernel om al het werk te doen.	- 8 Servo aansluitingen - Connectie met DMA van de Raspberry Pi	- Afhankelijk van kernel
PiServoController	23.10 Euro 25 Dollar	- Compact	- 6 Servo aansluitingen - Signaal buffering - Power stabilization	- Vergeleken met de rest vrij duur. - Gebruikt stroom van de Raspberry Pi
Servo Pi	17 Euro 11.99 Pond	- Powerlink optie	- 16 Servo aansluitingen - Asynchrone controle over de servomotoren	- Kan gevaar leveren voor de Pi als de Powerlink functie gebruikt wordt. - *Functies zijn niet goed gedocumenteerd*

Het advies van dit document is om de Chroma servoboard te nemen, omdat deze serieel communiceert. Dit is van elke taal beschikbaar zonder rare libraries of dergelijke te moeten downloaden om specifieke functies aan te kunnen roepen. Daarnaast is het Servoboard vrij goedkoop en heeft meer dan genoeg aansluitingen om aan de eisen te voldoen. Bij externe powersupply hoeft de aarding tussen de Raspberry Pi en het bord niet gelijk gemaakt te worden of een connectie hebben. Het nadeel is dat er met -1000 tot 1000 gewerkt wordt en het graden aantal kan niet opgegeven worden. Hieronder de Chroma servoboard boven op de Raspberry Pi.



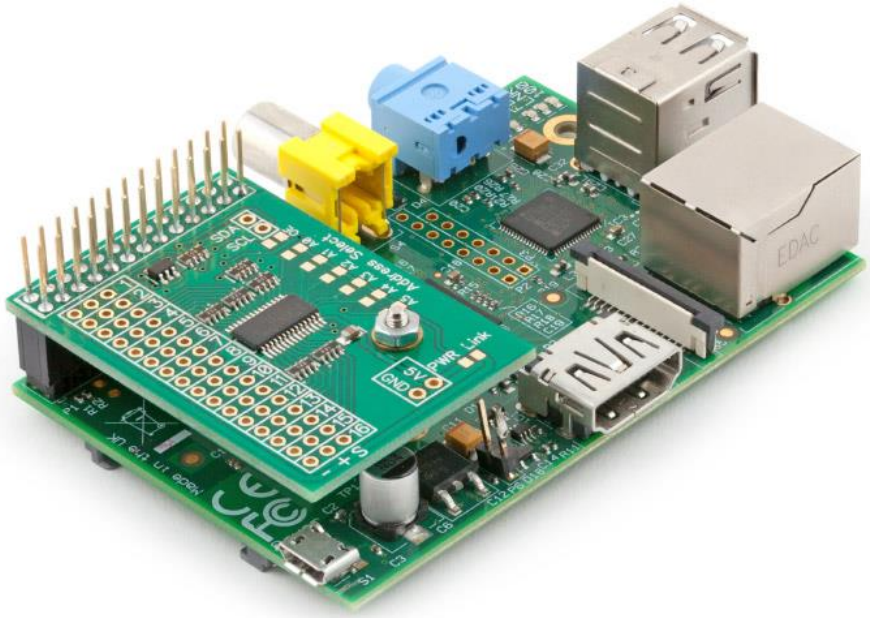
Figuur 1: Chroma servoboard aangesloten op de Raspberry Pi

Het chroma servoboard neemt ongeveer de helft van de ruimte van de Raspberry Pi inbeslag en hangt over de Raspberry Pi. Alle aansluitingen zijn prima beschikbaar. Hieronder volgt een diagram van hoe de aansluitingen zijn naar het servoboard toe. Aan de raspberry komt alleen het bord over de GPIO pinnen.



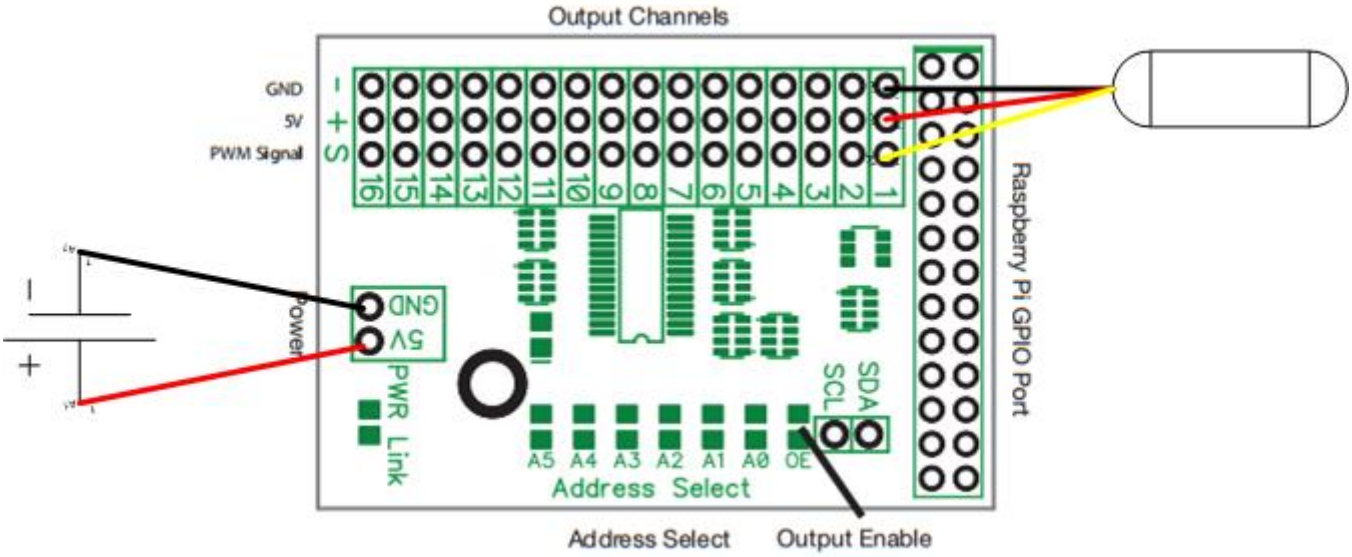
Figuur 2 : Aansluitingen Servomotor en Powersupply met Chroma Servoboard

De tweede optie zal de Servo Pi zijn, omdat deze de nodige bescherming geeft en een ruime hoeveelheid aansluitingen heeft voor servomotoren. Een nadeel is dat de libraries in python is geschreven en andere talen een wrapper nodig hebben om er mee te kunnen werken. Dit is vrij onhandig. De Servo Pi geeft de mogelijkheid om de Raspberry Pi stroom te geven via de Servo Pi. Hierdoor heeft deze zelf geen powersupply nodig, maar er is geen bescherming tegen hoge voltages als deze functie gebruikt word. Hieronder volgt de Servo Pi bovenop een Raspberry Pi.



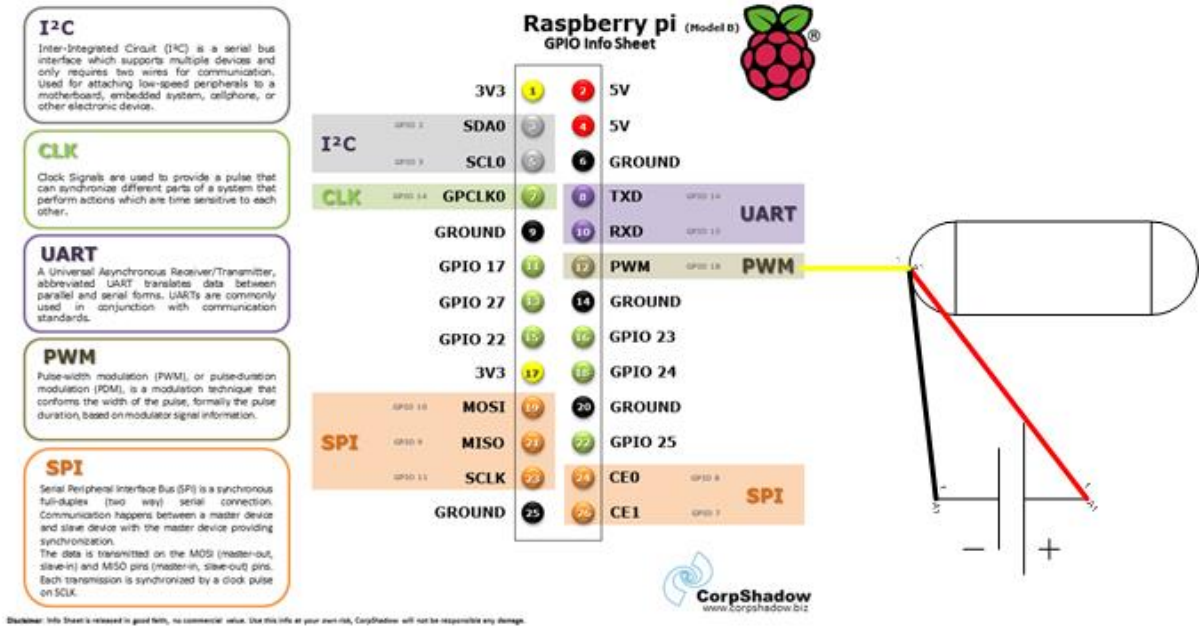
Figuur 3: Servo Pi aangesloten op de Raspberry Pi

Het meest opvallende verschil is dat de servomotor aansluitingen vrouwelijk zijn en niet mannelijk. Daarnaast is het qua grote en plek vergelijkbaar als de servoboard van Chroma. Hieronder volgt de aansluitingen van de servomotor en de powersupply naar de Servo Pi.



Figuur 4: Aansluitingen Servomotor en Powersupply met ServoPi.

Het is ook mogelijk om direct de servomotor aan de Raspberry Pi aan te sluiten met behulp van een externe power supply. Hier volgt abstract hoe het eruit kan zien. De Raspberry Pi bevat hier geen extra connecties naar andere apparatuur.

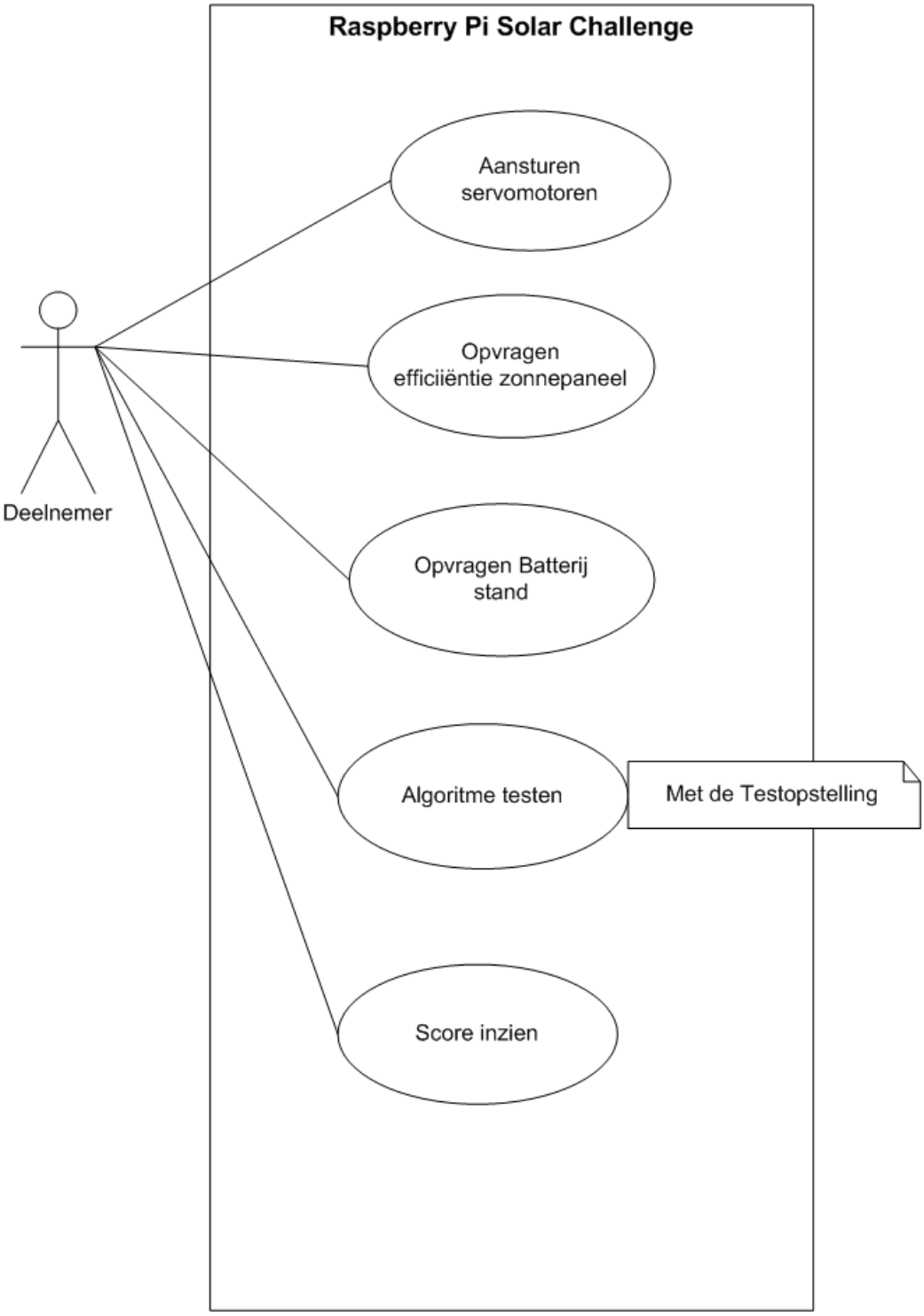


Figuur 5: Aansluitingen Servomotor direct met de Raspberry Pi. De Powersupply komt niet van de Raspberry Pi maar van een externe bron (bijvoorbeeld batterij).

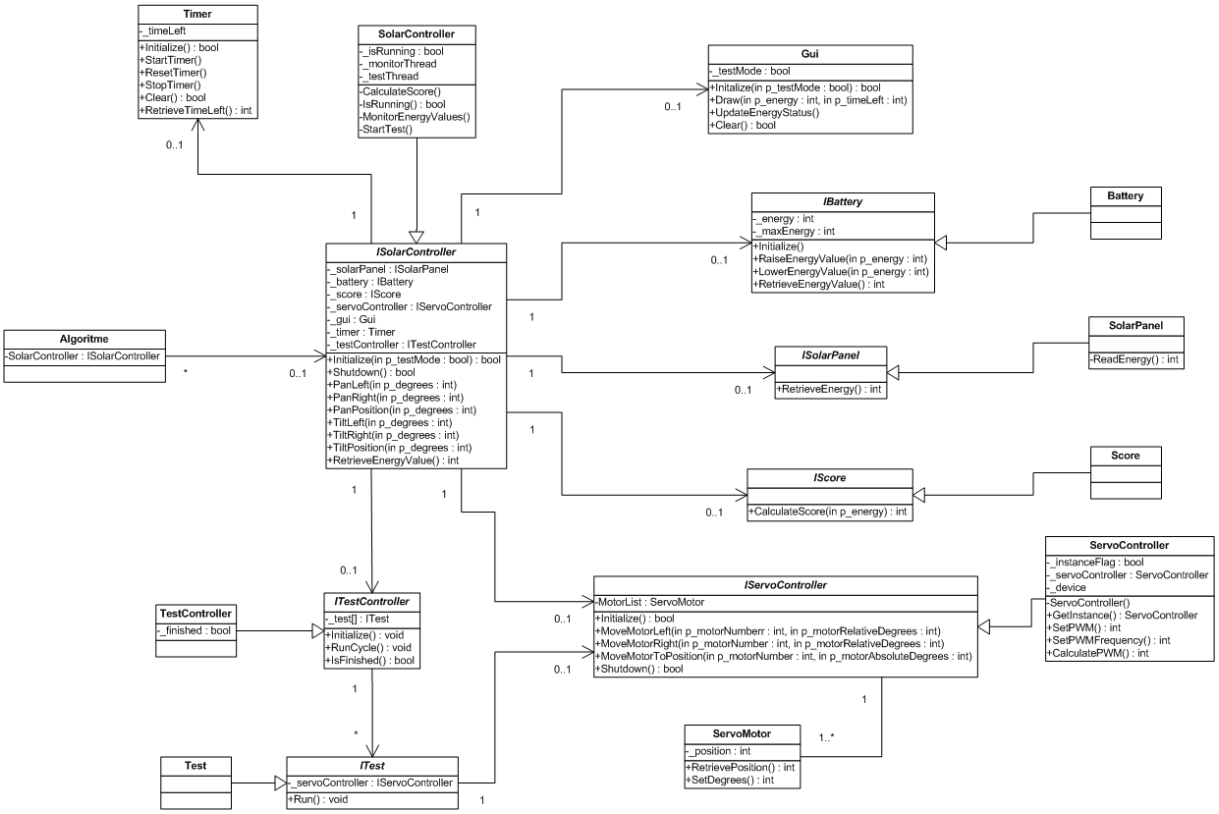
6. UML Diagrammen

Alle diagrammen hieronder zijn de laatste versies van de diagrammen. Een aantal diagrammen zijn meerdere keren gewijzigd gedurende het project zoals het klassendiagram en een paar sequence diagrammen. In de individuele sprints is te zien hoe het klassendiagram verandert is over de loop van het project.

6.1 Use Case Diagram

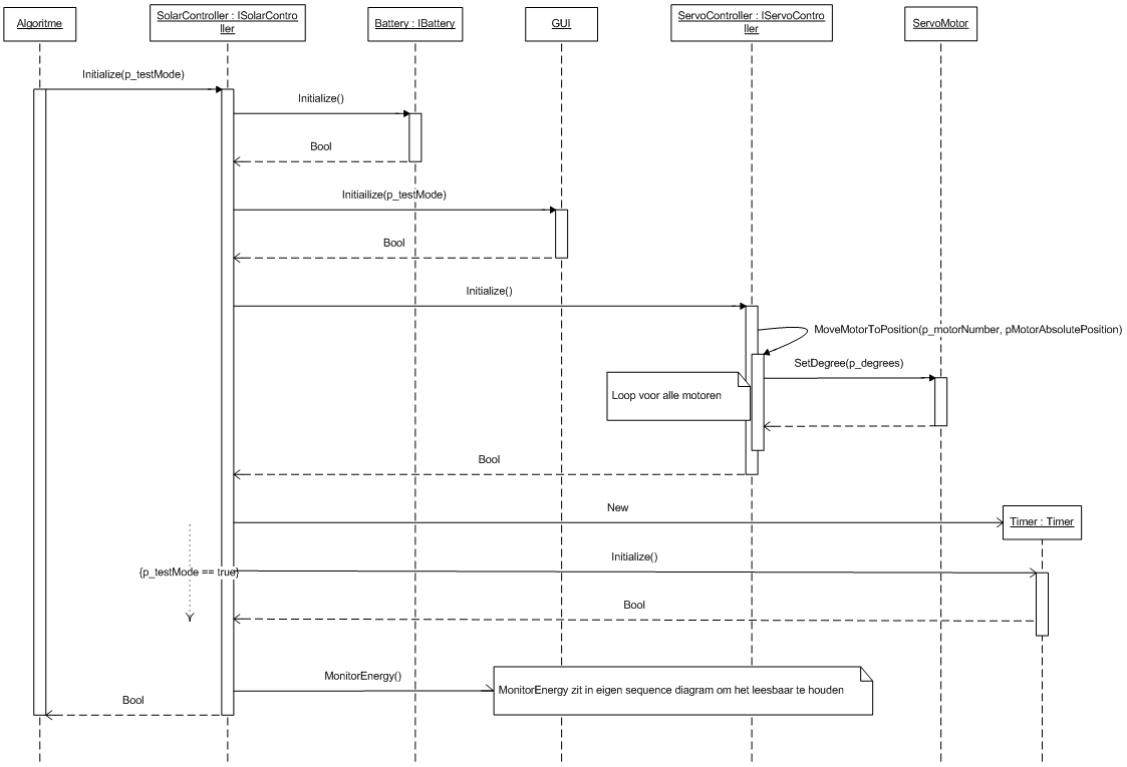


6.2 Klassen diagram

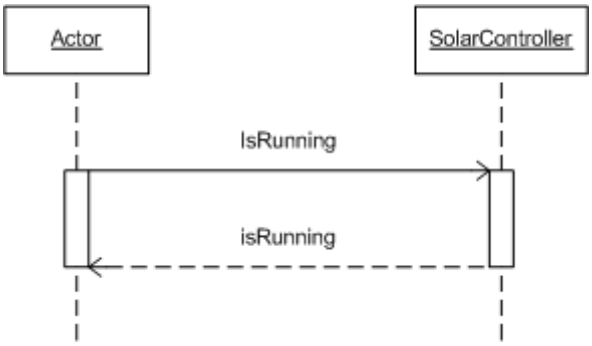


6.3 Sequence diagrammen

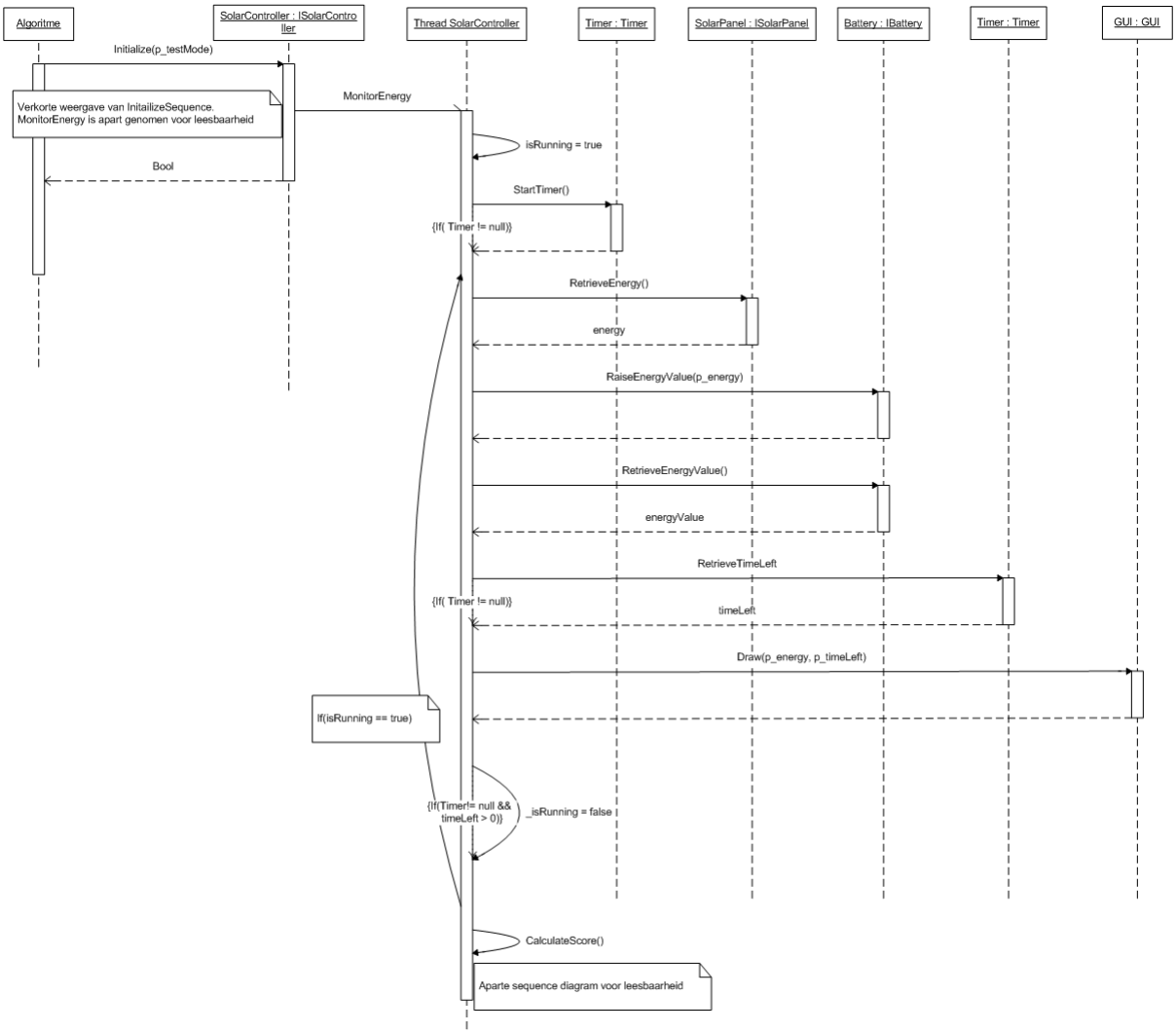
IsRunning - Sequence



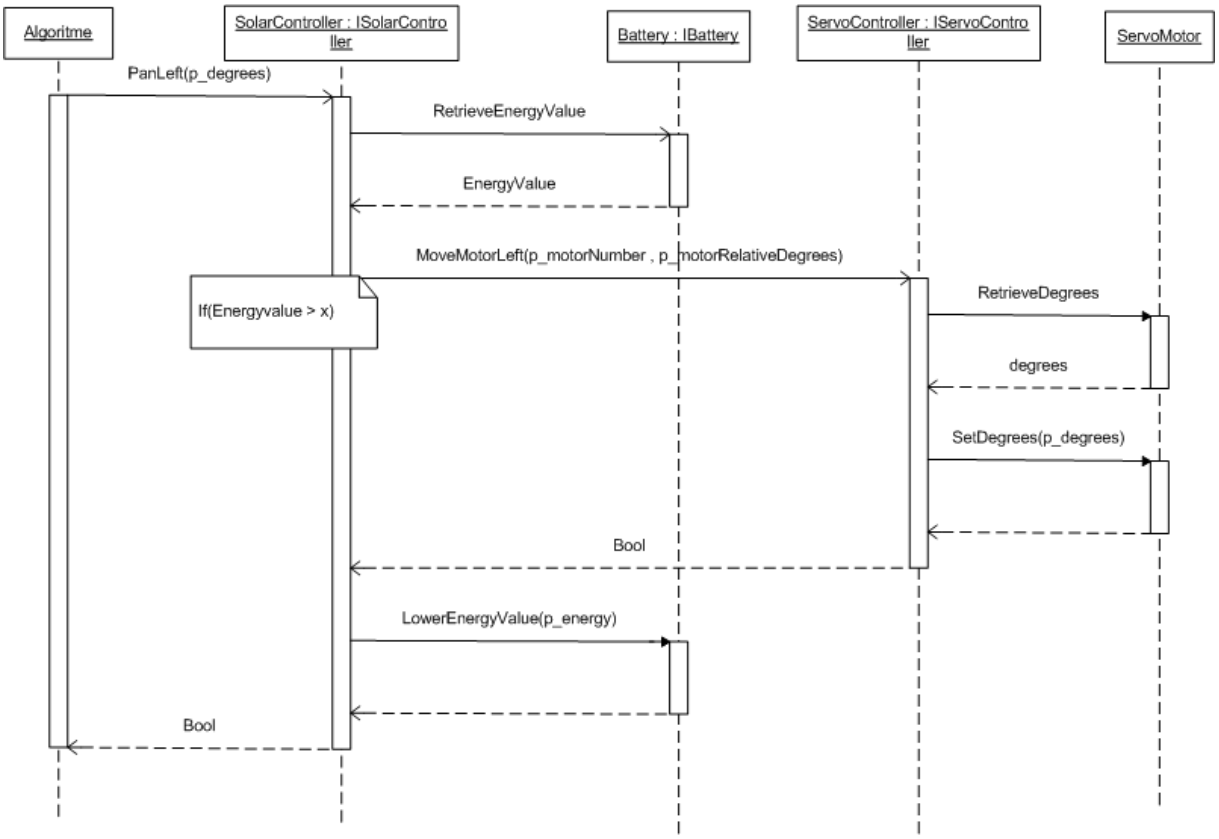
IsRunning - Sequence



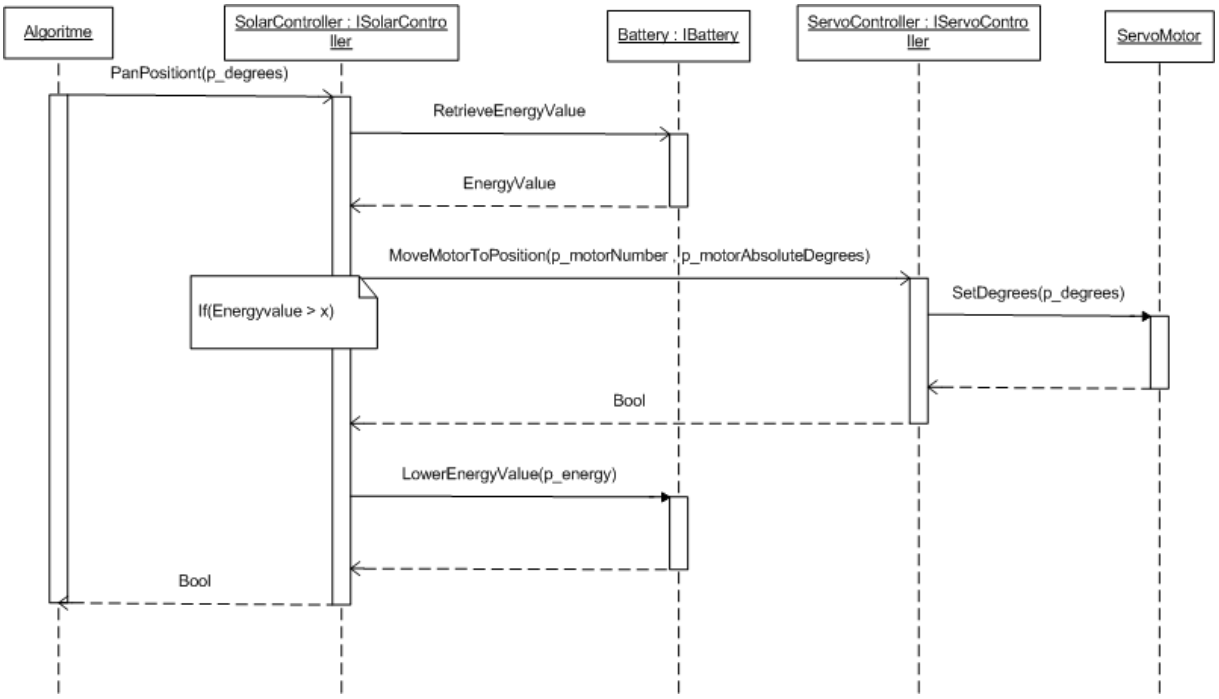
MonitorEnergy - Sequence



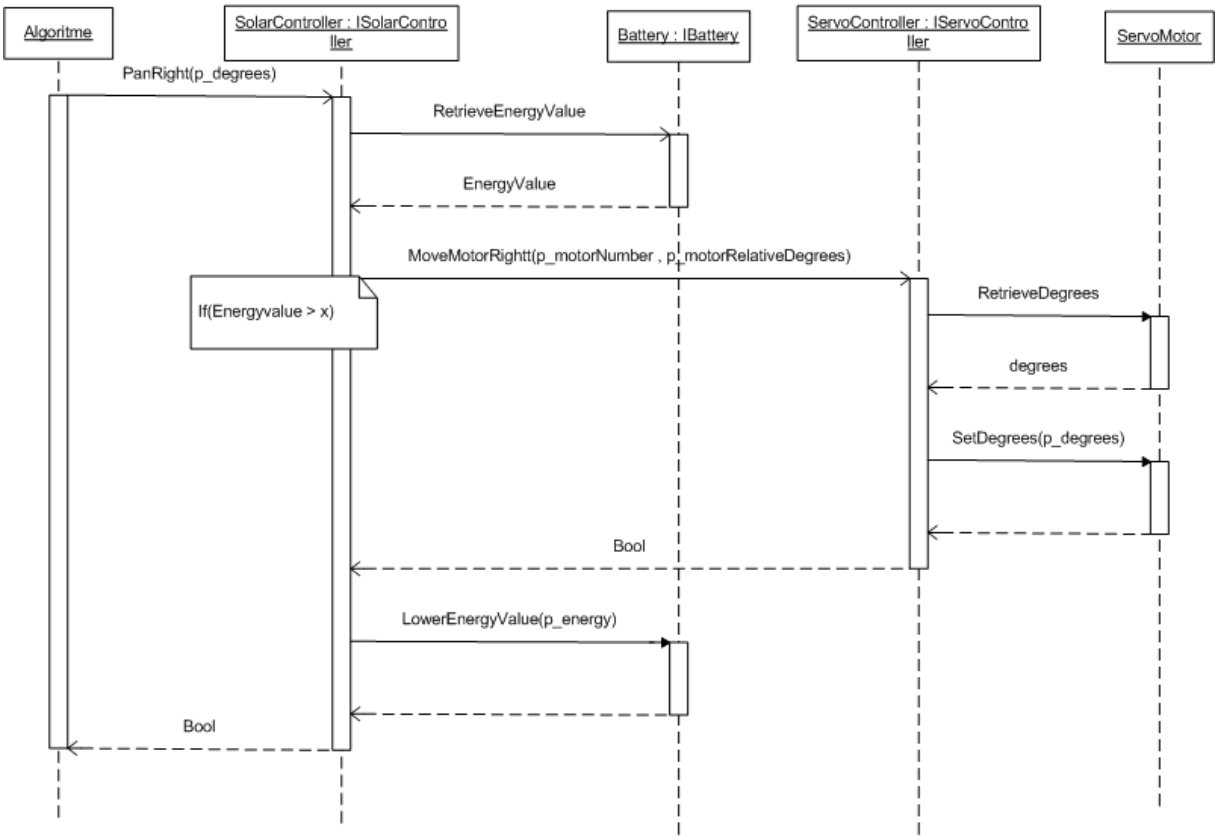
PanLeft - Sequence



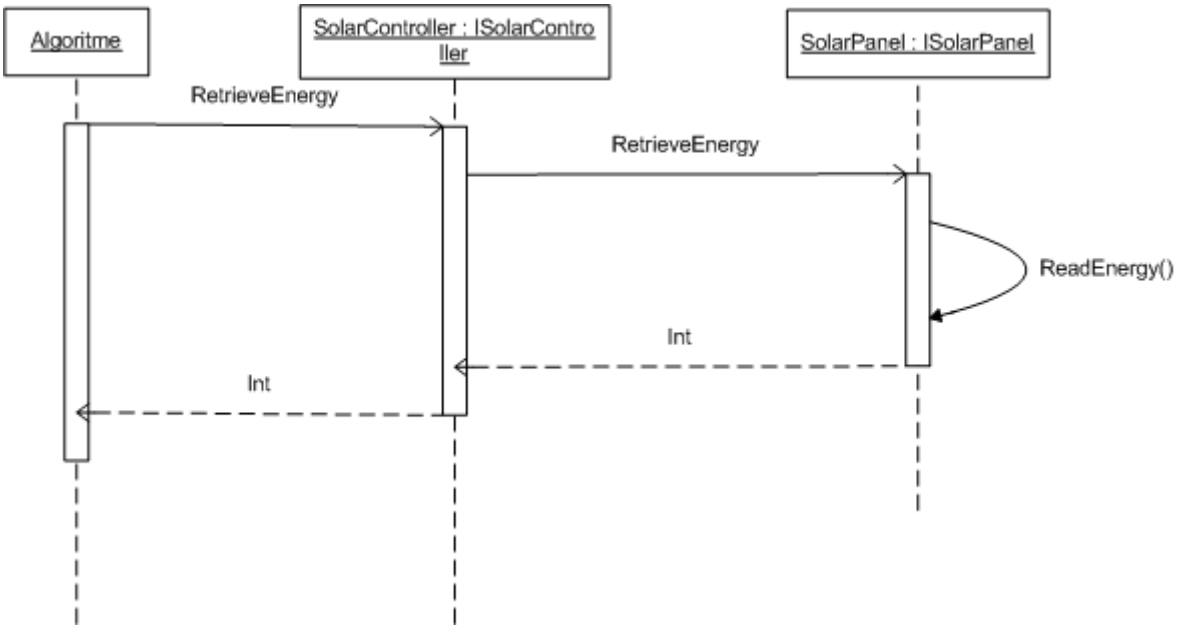
PanPosition - Sequence



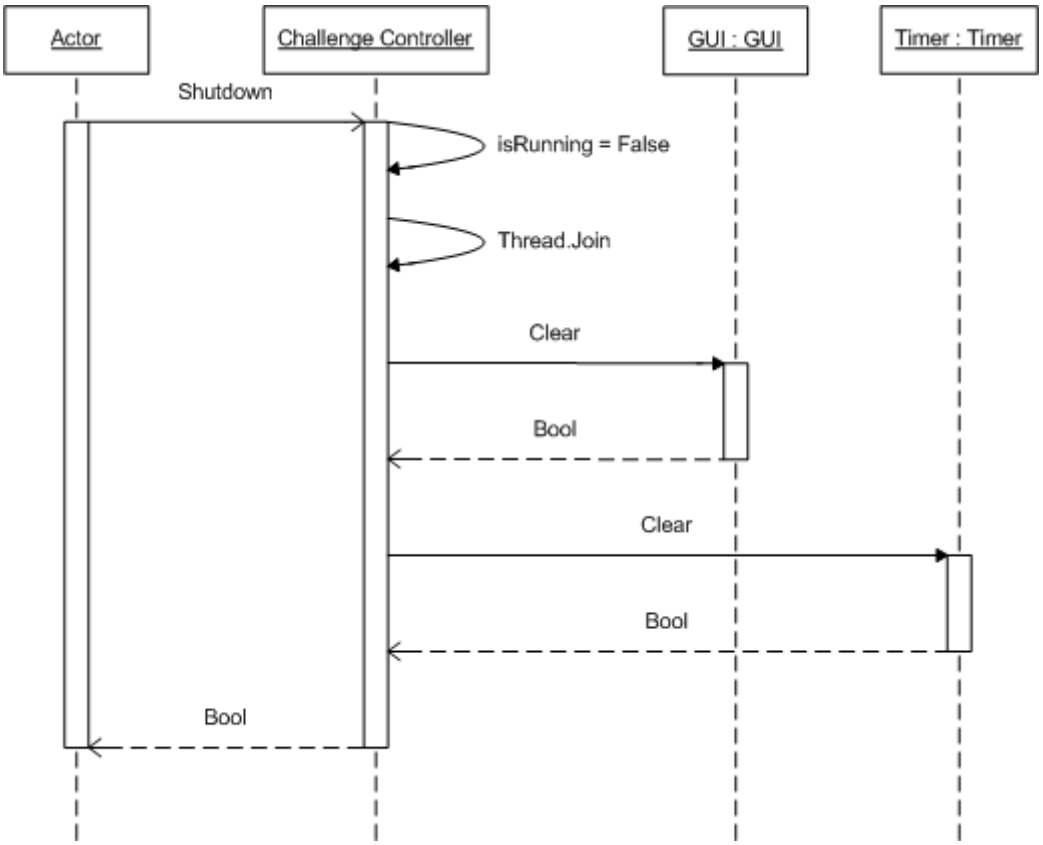
PanRight – Sequence



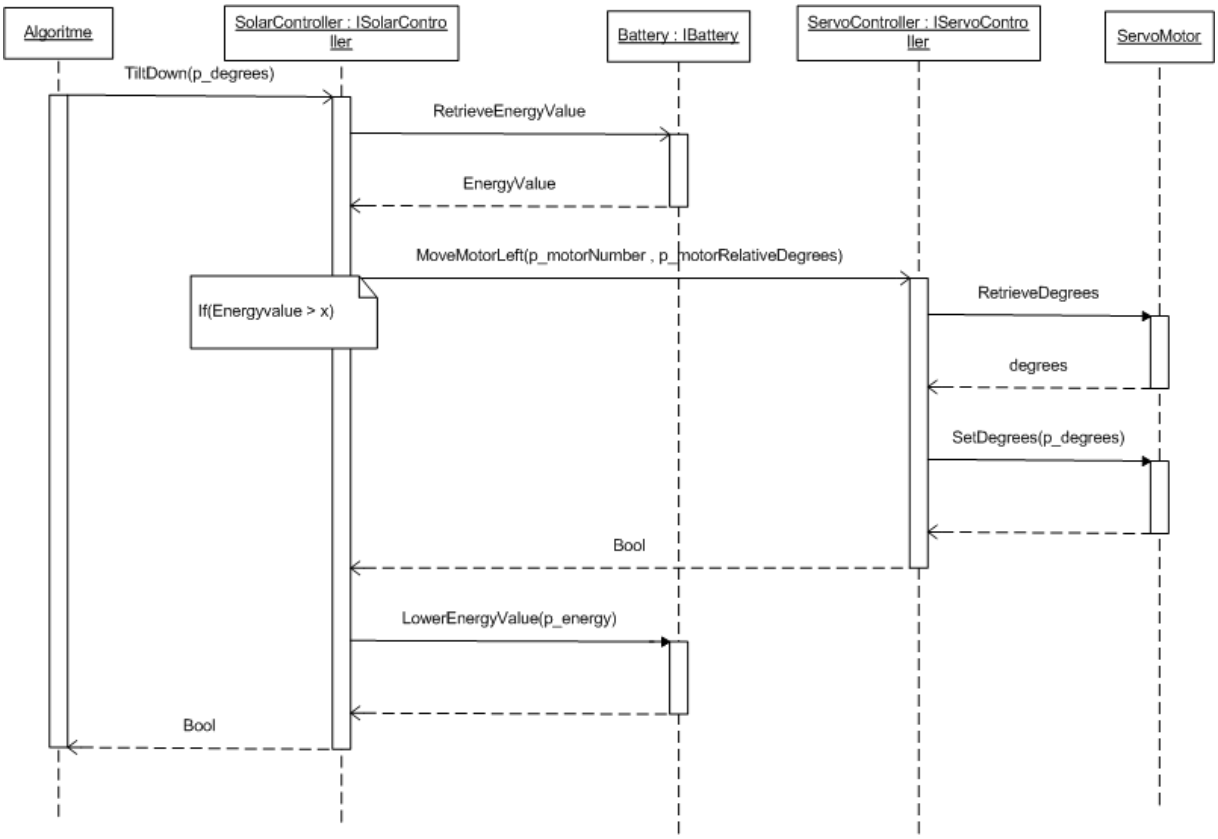
RetrieveEnergy – Sequence



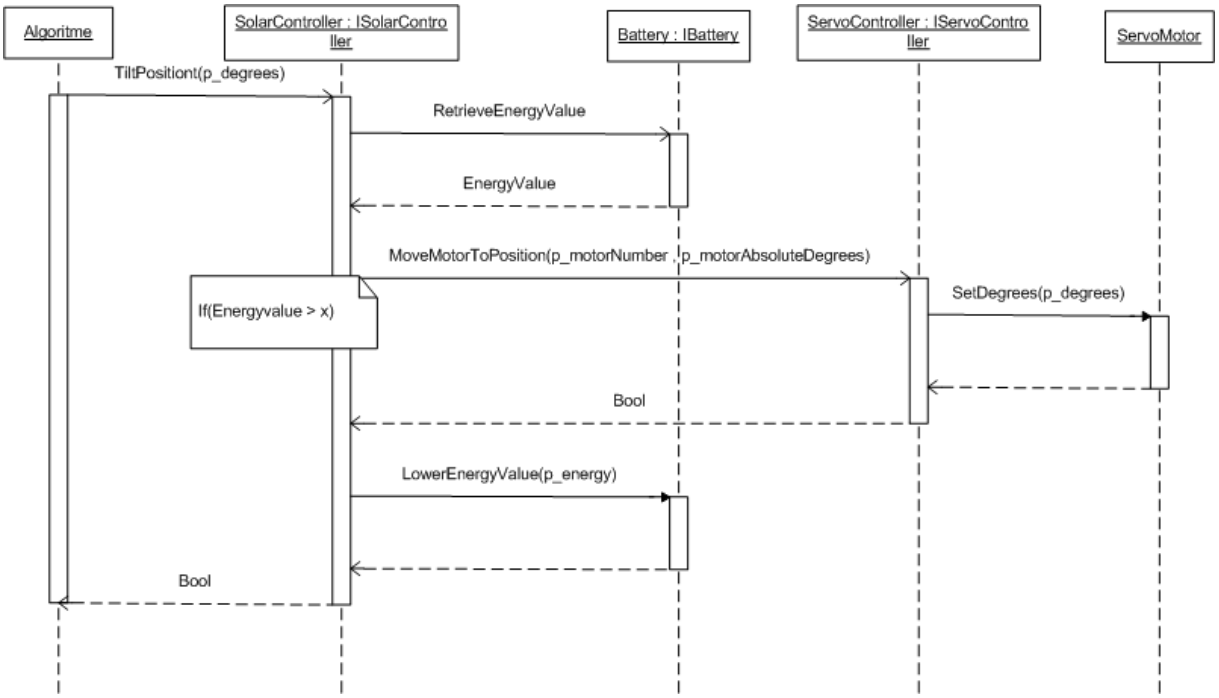
Shutdown – Sequence



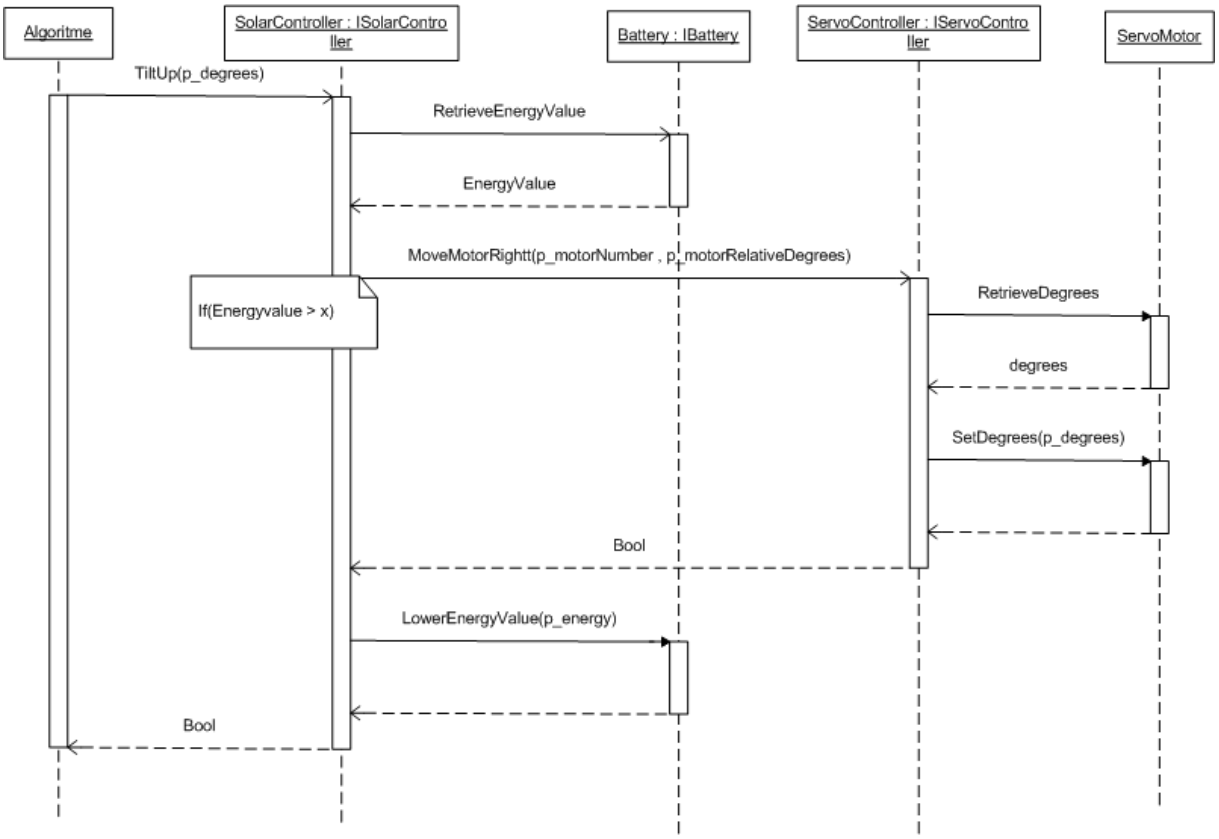
TiltDown – Sequence



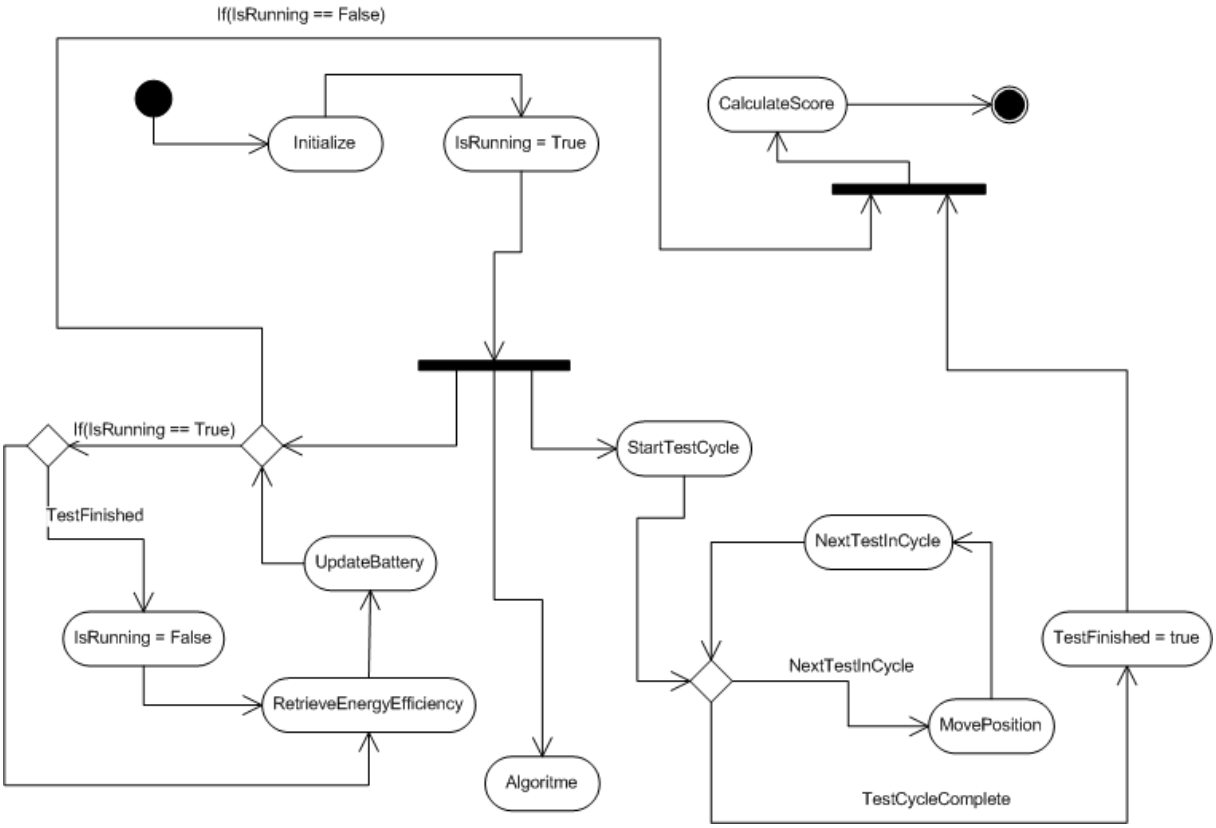
TiltPosition – Sequence



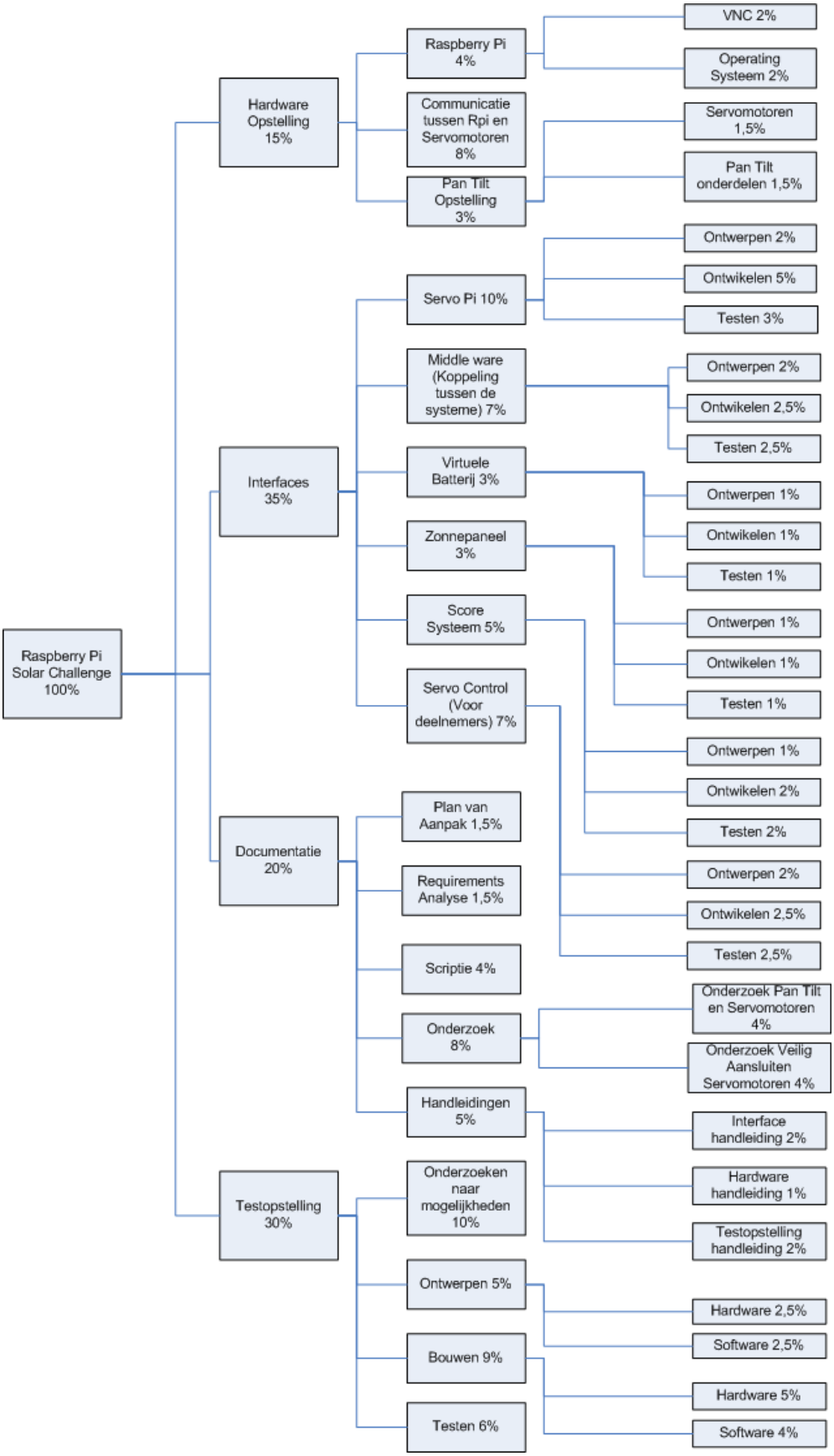
TiltUp – Sequence



6.4 Activity Diagram

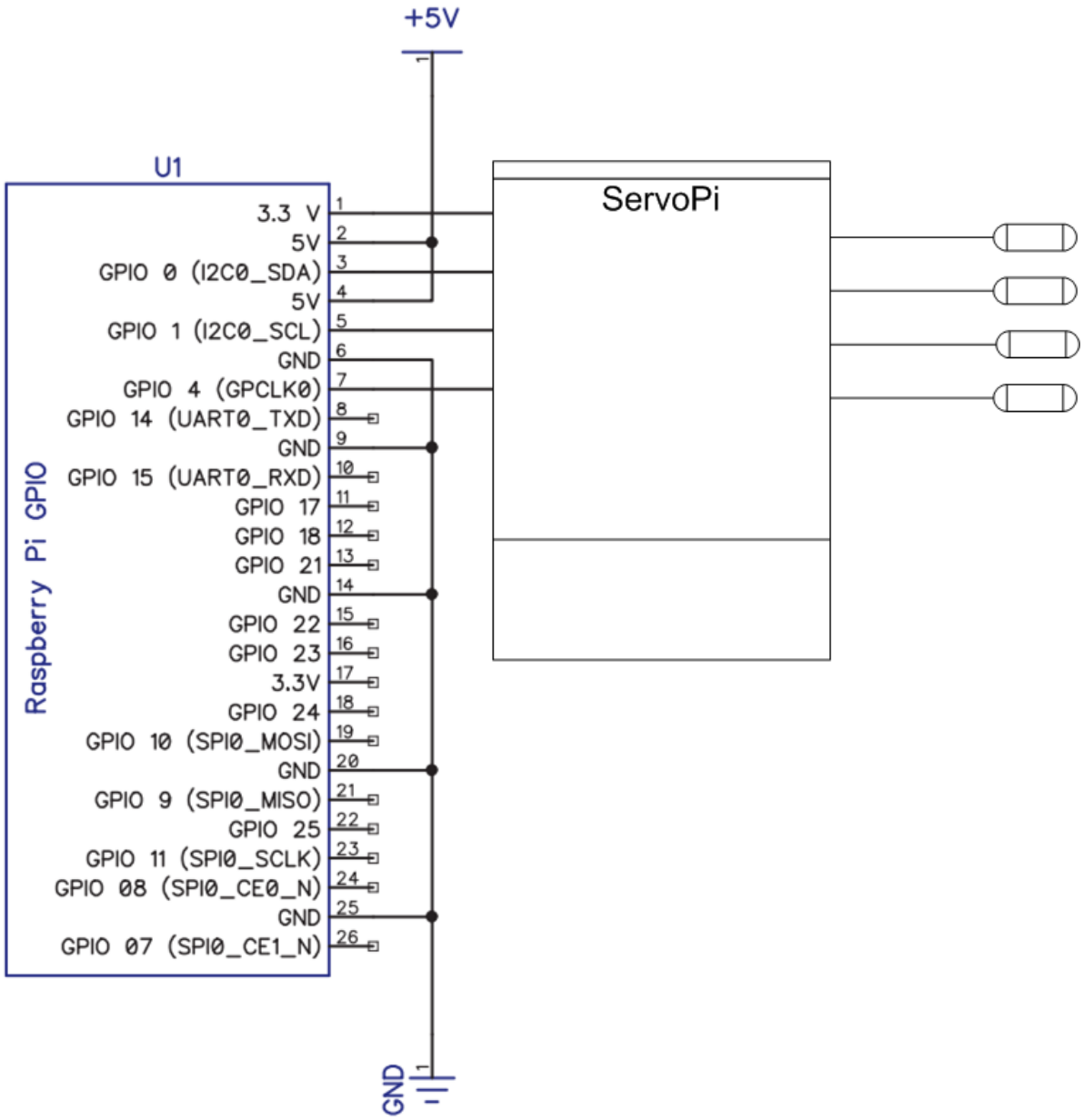


7. Work Breakdown Structure (WBS)



8. Aansluitschema en I2C uitleg

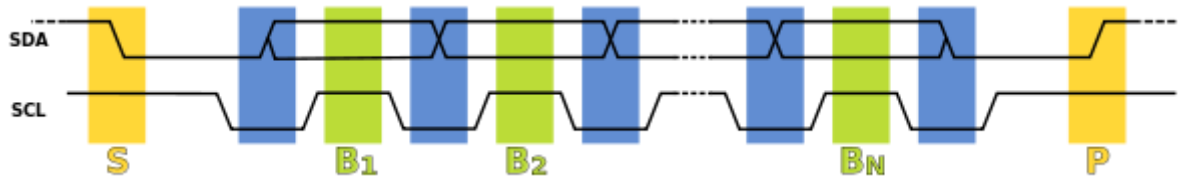
In onderstaande figuur 2 is in een abstracte weergave te zien hoe de GPIO pinnen van de Raspberry Pi gekoppeld zijn aan de Servo Pi. De Servo Pi is hier abstract gehouden, omdat hier gefocused wordt op de aansluitingen tussen beide systemen. Naast de Servo Pi zijn ook een set servomotoren. Deze zitten aangesloten op de Servo Pi, omdat deze de servomotoren zal aansturen.



Figuur 2: Abstracte weergave van de aansluitingen tussen Raspberry Pi GPIO pinnen en de Servo Pi.

De pinnen die de Servo Pi gebruiken zijn Pin 3 I2C0_SDA, Pin 5 I2C0_SCL en Pin 7 GPCLK0. Andere pinnen die aangesloten zijn aan de Servo Pi maar geen communicatie overheen verlopen zijn Pin 1 3,3 Volt, pinnen 2 en 4 voor 5 Volt en pinnen 6, 9, 14, 20 en 25 voor Ground ook wel Aarding. Dit document zal zich richten op Pin 3, 5 en 7, omdat hier alle communicatie over heen loopt.

Voorheen genoemd communiceert de Raspberry Pi met de Servo Pi via I2C. I2C is een synchrone, seriële bus met als doeleinde datacommunicatie tussen microprocessoren en andere integrated circuits op meestal dezelfde printplaat. In deze situatie zal het niet op dezelfde printplaat zijn, omdat het twee losse printplaten zijn die zullen communiceren via I2C op de GPIO pinnen van de Raspberry Pi. I2C gebruikt twee datalijnen om de communicatie te regelen. Dit zijn de data (SDA) lijn en de klok (SCL) lijn. In figuur 3 is een weergave van hoe de SDA en SCL lijnen samenwerken om een bericht over te krijgen.



Figuur 3: Voorbeeld van I2C communicatie

Om een bericht te versturen zal SDA van een hoog naar laag signaal gaan tijdens een hoog SCL signaal. Dit geeft een startbit aan en zal de communicatie gestart worden. Vervolgens zet SDA het eerste bit level terwijl SCL laag blijft gezien in de blauwe balken. De data is ontvangen als de SCL stijgt voor de eerste bit zoals te zien is in de groene balken. Dit proces herhaalt zich totdat alle data verstuurd en ontvangen is. Als de communicatie klaar is wordt een stopbit verstuurd. Dit is wanneer de SDA naar hoog gaat terwijl de SCL ook hoog is. De SDA verandert op elke SCL rising en falling edge om verkeerde detecties van het signaal te voorkomen. De SDA lijn voor onze situatie is pin 3 en de SCL lijn is pin 5.

Pin 7 met als functie GPCLK0 is een optionele pin voor de Servo Pi. De Servo Pi beidt de mogelijkheid aan om via externe methoden een PWM signaal te ontvangen in plaats van het PWM signaal van de Servo Pi te gebruiken. Als deze functie gebruikt wordt zal Pin 7 van de Raspberry Pi het signaal doorsturen naar de Servo Pi. Het is niet verplicht om Pin 7 te gebruiken, maar het kan een rol spelen in de besturing van de servomotoren als hij gebruikt wordt

De Servo Pi gebruikt niet alle pinnen, want uit de 26 zijn er 2 die gebruikt worden voor communicatie en 1 om een extern PWM signaal te generen voor de Servo Pi. Naast deze pinnen zijn er nog de voeding en aardings pinnen die aangesloten zitten aan de respectievelijke aansluitingen van de Servo Pi. De andere pinnen worden allemaal niet gebruikt, maar worden wel beschikbaar gesteld door de pin header op de Servo Pi. Met de pinheader zijn ze nog steeds beschikbaar om gebruikt te worden door de gebruiker van het product. De pinnen die niet door de Servo Pi gebruikt worden zijn pin 8, 10, 11, 12, 13, 15, 16, 17, 18, 19, 21, 22, 23, 24 en 26. Dit zijn voornamelijk pinnen die dienen als input, output of een speciale alternatieve functie zoals bijvoorbeeld de I2C pinnen.

9. Ontwikkelomgeving en Deployen naar Raspberry Pi

9.1 Software draaien op de Raspberry Pi

Om software te kunnen draaien op de Raspberry Pi zijn er meerdere programma's en manieren om dat voor elkaar te krijgen. De meest directe methode is op de Raspberry Pi programmeren en gelijk aanzetten, maar deze optie zal niet goed werken. Het is mogelijk om software als Eclipse of QT te draaien op de Raspberry Pi, maar het vergt enorm veel van de Raspberry Pi om projecten te builden en compilen. Voor de tweede revisie van de Raspberry Pi is het beter mogelijk om het op de directe methode uit te voeren. Voor ons project zal een alternatief gevonden moeten worden om onze software te krijgen op de Raspberry Pi.

Cross compiling

Cross compiling stelt een programmeur in staat om software te builden en compilen voor andere architecturen dan de architectuur waar hij zelf in zit. Bijvoorbeeld Windows naar Linux machine of vice versa. Dit is een prima methode voor de deelnemers om de software te schrijven en te builden naar de Raspberry Pi. Daar kan het programma zijn functie uitvoeren en afronden zoals normaal ook zo gebeuren. Er zijn meerdere software ontwikkeltools beschikbaar en van Visual studio, Eclipse en QT wordt uitgewerkt hoe cross compiling opgezet kan worden.

Visual studio: http://www.hassang.com/visual-studio-2012-template-and-the-raspberry-pi-toolchain-c

Visual studio kan cross compilen naar de Raspberry Pi. Hiervoor is een toolchain nodig. De toolchain maakt het mogelijk om een project te compilen voor andere architecturen en systemen. Daarnaast is een cross compiler nodig zoals VisualGBD of Cygwin. Cygwin is gratis en levert een grote hoeveelheid packages van Linux en maakt die beschikbaar voor Windows. Hierdoor is het mogelijk om met Windows te compileren voor Linux. VisualGBD is een pakket voor visual studio. Met het pakket kan bij het aanmaken van een nieuw project gekozen worden voor cross compiling. VisualGBD zal daarvoor een nieuw scherm openen met opties betreft de tooling en het platform waarvoor het gemaakt gaat worden. Uiteindelijk zal ook ingesteld worden hoe de bestanden overgezet gaan worden naar het andere platform. Dit soort pakketten maakt het mogelijk om cross te compileren met visual studio.

Eclipse:http://www.gurucoding.com/en/raspberry_pi_eclipse/raspberry_pi_cross_compilation_in_eclipse.php

Eclipse geeft dezelfde mogelijkheid van Windows naar Linux toe met een cross compiler. Hiervoor is ook een toolchain nodig zodat Eclipse het project kan builden voor de Raspberry Pi. Eclipse gebruikt een SSH connectie om een Remote Server Explorer op te zetten. Hiermee kan de gebruiker de terminal bereiken, de filestructuur en ook direct files overzetten van desktop naar Raspberry. De toolchain wordt verwacht gekocht te worden of zelf te maken met Cygwin en MinGW. Dit hele proces neemt vrij veel tijd in beslag en bij fouten mag het hele process opnieuw uitgevoerd worden.

QT

Voor QT is niet veel informatie te vinden om van windows naar de Raspberry te kunnen cross compilen. Het kan wel van linux naar de Raspberry Pi, maar voor windows is er niet veel te vinden.

9.2 Virtueel machine (Fedora / Debian)

Naast het via Windows te doen kan het ook via Linux. Op de meeste machines staat Windows en zal via een virtuele machine een Linux systeem gebruikt moeten worden. Compilen op Linux zal makkelijk werkend te krijgen zijn op de Raspberry Pi, omdat daar een iets aangepaste versie van Debian gebruikt wordt. Hierdoor zijn de libraries en linking veel hetzelfde. De toolchain is nog wel nodig om de binairies te maken, want een desktop linux heeft andere binaries als output dan een arm device.

Visual studio

Niet voor linux

QT: http://wiki.qt.io/RaspberryPi_Beginners_Guide

QT heeft ook de mogelijkheid om cross te compileren naar de Raspberry Pi toe. Helaas zijn hier meer stappen bij dan bij bijvoorbeeld Eclipse of Visual Studio. QT heeft veel libraries en extra's die naast de normale toolchain ook meegenomen moeten worden. Anders kunnen de QT functies niet toegepast worden en zal veel niet werken. Op de Raspberry Pi moeten de packages geïnstalleerd worden voor de QT libraries en daarnaast moet de toolchain ervan af weten. Daarnaast stelt het de gebruiker alleen in staat om te cross compilen, niet remote debuggen. Dit is nadelig, omdat gebruikers en in de context van de opdracht voor de deelnemers een slechte ervaring. De deelnemers zullen namelijk veel moeite krijgen om bugs en problemen uit hun algoritmes te krijgen en dit is juist wel gewenst.

Eclipse: https://www.youtube.com/watch?v=T9yFyWsyyGk

Eclipse heeft de mogelijkheid om lokaal software te developen en deze remote te deployen. Daarbij kan het zelfs debugged worden. Hiervoor worden functies van Eclipse gebruikt zoals de remote server explorer. Om het geheel werkend te krijgen moet de lokale machine klaar staan om voor een andere omgeving te compilen. Dat doet men met een cross compiler en een toolchain. De toolchain voor linux kan officieel gedownload worden van de Git repository van de Raspberry Foundation. Na de installatie van de toolchain kan Eclipse opgestart worden. Bij het aanmaken van een project kan aangegeven worden dat er Cross GCC nodig is voor cross compiling. Met verdere invulling van de gegevens zoals welke toolchain nodig is zal Eclipse alle informatie instellen betreft libraries etc. Na de instelling van deze mode is het mogelijk om C++ projecten te ontwikkelen in de desktop en op de Raspberry Pi te gebruiken. Om dit bestand te krijgen op de Raspberry Pi kan er via Eclipse een remote connectie gemaakt worden. Hiervoor wordt de plugin Remote server explorer gebruikt. Met deze plugin kan de gebruiker zijn projecten deployen op de Raspberry Pi, terminals openen op de Raspberry Pi of de File structuur browsen.

http://hertaville.com/2012/09/28/development-environment-raspberry-pi-cross-compiler/

Naast deployen kan er ook remote gedebugged worden. Dit is uitermate handig om te kunnen zien wat een programma runtime doet. Hiervoor wordt gebruik gemaakt van GDB. Deze kan handmatig gebruikt worden of automatisch door Eclipse. Automatisch door Eclipse laten doen werkt prima totdat het programma root rechten nodig heeft. Hiervoor kan het handmatig opgezet worden om alsnog root rechten te krijgen voor het programma. Dit is nodig om bijvoorbeeld bestanden te gebruiken die alleen door root gebruikers mogen gebruikt worden. In de context van de Raspberry Pi kan alleen de root user de GPIO pinnen gebruiken. Om handmatig gebruik te maken van remote debugging moet er op de Raspberry een gbdserver geïnstalleerd staan. Daarnaast zal er op de desktop een client geïnstalleerd moeten worden. Deze communiceren met elkaar om remote debuggen mogelijk te maken. Zo kan men de motoren zien bewegen tijdens het debuggen om te zien bij welke stap het misgaat. Deze functionaliteit is enorm handig om te hebben en maakt problemen oplossen een stuk makkelijker, want ze zijn te vinden via de debugger.

http://hertaville.com/2013/01/11/remote-debugging/

10. Handleiding opbouwen van Ontwikkelomgeving

10.1 Voorbereiding

Om de ontwikkelomgeving klaar te maken om te cross compilen voor de Raspberry Pi zijn een paar voorbereidingen nodig. De Raspberry Pi heeft een installatie van Raspbian draaien. Deze is verkrijgbaar via de RaspberryPiFoundation of Raspbian websites. Daarbij wordt ervanuit gegaan dat een Linux machine klaar staat voor gebruik. Dit kan zowel op een virtuele machine als direct op de pc. Dit stappenplan is voor Fedora gemaakt, maar zou in principe ook moeten werken voor Debian of Ubuntu versies van Linux.

10.2 Opstellen Remote Acces

De programma's worden op de laptop ontwikkeld en later overgezet naar de Raspberry Pi. Hiervoor zal gebruik gemaakt worden van VNC en SSH om remote met de Raspberry Pi te communiceren. SSH staat op Raspbian automatisch aan, maar voor VNC moet een server zelf opgezet worden.

Op de Raspberry Pi kan TightVNC geinstalleerd worden met het commando:

```
sudo apt-get install tightvncserver
```

Als de installatie voltooid is kan de configuratie opgestart worden met het commando:

```
tightvncserver
```

Hierbij moet een password opgegeven worden en een optionele view-only password. Het view-only password zorgt ervoor dat de connectie niks mag wijzigen en alleen mag kijken. Het normale password is verplicht en zal later nodig zijn om de connectie te maken met de Raspberry Pi. Nu kan handmatig de server gestart worden met:

```
vncserver :0 -geometry 1920x1080 -depth 24
```

De :0 geeft het scherm aan die gebruikt moet worden. Geometry is de resolutie en de depth is voor multiple trueColor support.

Het opstarten van vncserver kan geautomatiseerd worden, zodat deze niet telkens handmatig opgestart hoeft te worden. Hiervoor moet een shell script gemaakt worden zoals de volgende:

```
#!/bin/sh
vncserver :0 -geometry 1920x1080 -depth 24 -dpi 96
```

Dit bestand moet opgeslagen worden onder een naam naar keuze, in dit voorbeeld zal 'vnc.sh' gebruikt worden. Vervolgens om het bestand uitvoerbaar te maken is er het volgende commando:

```
chmod +x vnc.sh
```

Nu kan het bestand op elk moment gebruikt worden met: ./vnc.sh

Om het volledig automatisch te maken, moet deze automatisch gedraaid worden bij het opstarten van de Raspberry Pi. Om dit uit te voeren zijn root rechten nodig in de terminal met behulp van:

```
sudo su
```

Navigeer naar de folder /etc/init.d/ met:

```
cd /etc/init.d/
```

Maak hier een nieuw bestand aan met het volgende script:

```
### BEGIN INIT INFO
# Provides: vncboot
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start VNC Server at boot time
# Description: Start VNC Server at boot time.
### END INIT INFO

#! /bin/sh
# /etc/init.d/vncboot

USER=pi
HOME=/home/pi

export USER HOME

case "$1" in
 start)
  echo "Starting VNC Server"
  #Insert your favoured settings for a VNC session
  su - pi -c "/usr/bin/vncserver :0 -geometry 1280x800 -depth 16 -
pixelformat rgb565"
  ;;

 stop)
  echo "Stopping VNC Server"
  /usr/bin/vncserver -kill :0
  ;;

 *)
  echo "Usage: /etc/init.d/vncboot {start|stop}"
  exit 1
  ;;
esac

exit 0
```

Sla dit bestand op met een naam naar keuze, dit voorbeeld gebruikt vncboot. Maak dit bestand uitvoerbaar met:

```
chmod 755 vncboot
```

Zet dependancy-based boot sequencing aan met:

```
update-rc.d /etc/init.d/vncboot defaults
```

Als dit commando succesvol is dan komt er het volgende te staan:

```
update-rc.d: using dependency based boot sequencing
```

Maar als je het volgende ziet:

```
update-rc.d: error: unable to read /etc/init.d//etc/init.d/vncboot
```

Kan het volgende commando gebruikt worden:

```
update-rc.d vncboot defaults
```

Als de Raspberry Pi opnieuw opgestart wordt zal VNCserver automatisch opstarten en ook netjes afsluiten bij het afsluiten van de Raspberry Pi

Om vanuit een computer bij de Raspberry Pi te komen hebben we een client nodig. Op Linux staat deze vaak standaard geïnstalleerd en bij Windows moet deze handmatig geïnstalleerd worden. Op Linux kan Remote Desktop Viewer gebruikt worden, deze is vaak te vinden in de internet sectie onder applicaties. Deze zou ook gevonden kunnen worden met de zoekbar in Fedora.

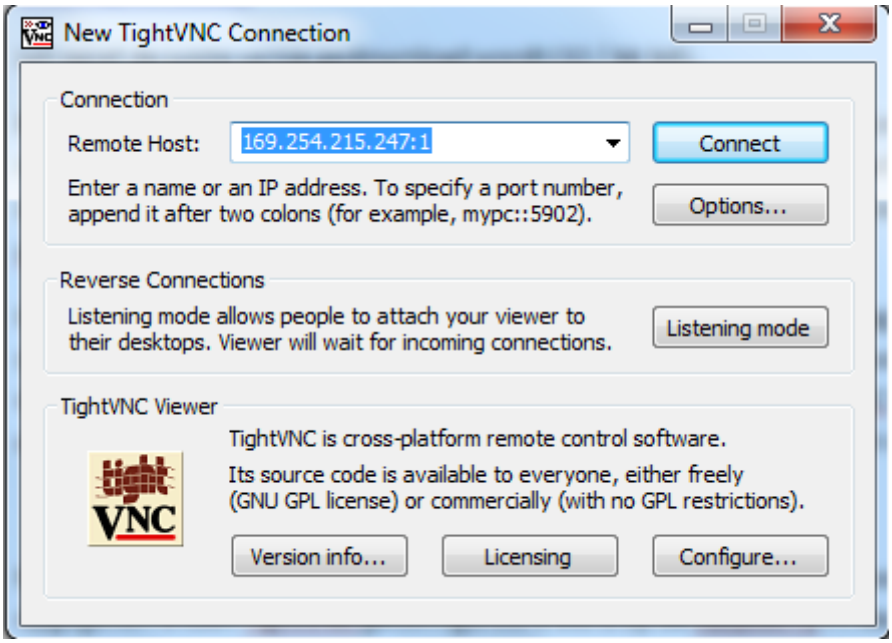
Zet het protocol op VNC. Voer het IP adres van de Raspberry Pi in gevolgd met een schermnummer (:0 of :1). Een voorbeeld is 192.168.0.6:1 Na het invullen van de gegevens kan op Connect gedrukt worden. Nu wordt gevraagd naar een wachtwoord, dit is het wachtwoord dat eerder is ingevoerd bij de configuratie van de VNCserver op de Raspberry Pi. Na de invulling hiervan zal de desktop van de Raspberry Pi te zien zijn. Als er afgesloten wil worden kan het gebruikelijke shutdown commando gebruikt worden om de Raspberry Pi af te sluiten. Om de VNC connectie te stoppen kan het scherm afgesloten worden en hoeft er verder niks meer gedaan te worden.

Onder Windows kan TightVNC geïnstalleerd worden. Hiervoor moet het gedownload worden van www.tightvnc.com

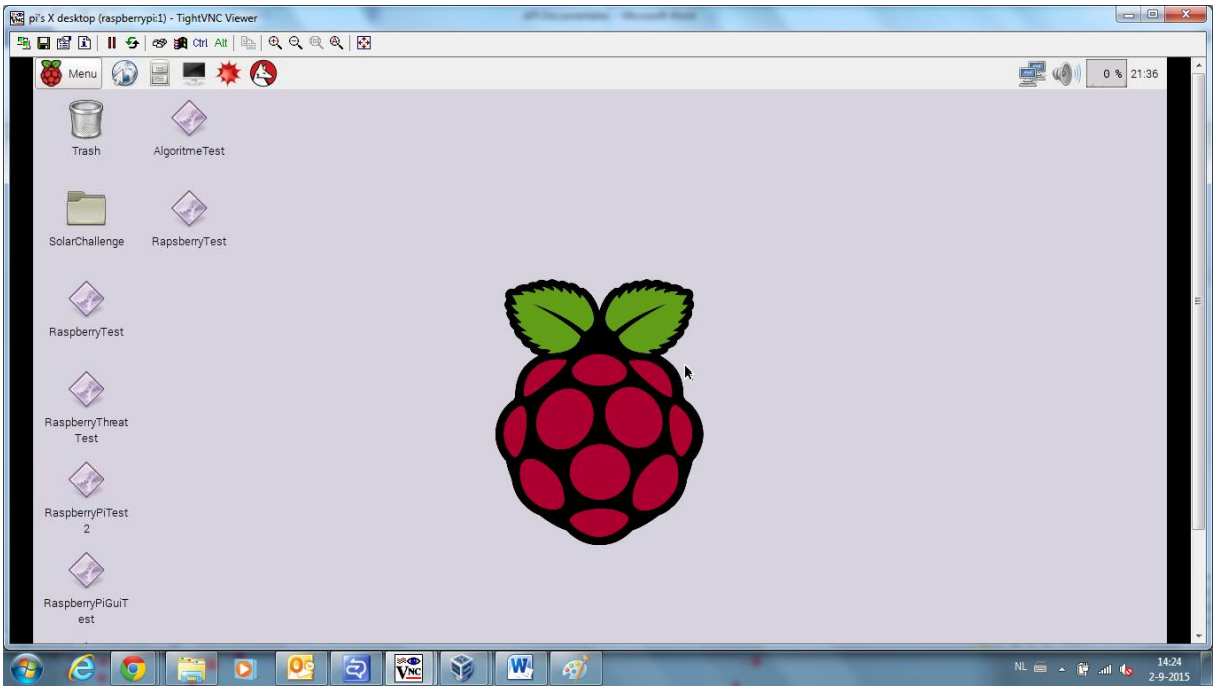
Let op of de juiste versie gedownload wordt (32 / 64 bit).

Bij het installeren worden drie opties aangeboden: Typical, Custom of Complete. Alleen de client is nodig op de Windows machine, dus de rest is overbodig. Om dit in te stellen moet er een Custom installatie gedaan worden. Selecteer vervolgens de TightVNC Server en kies Entire feature will be unavailable.

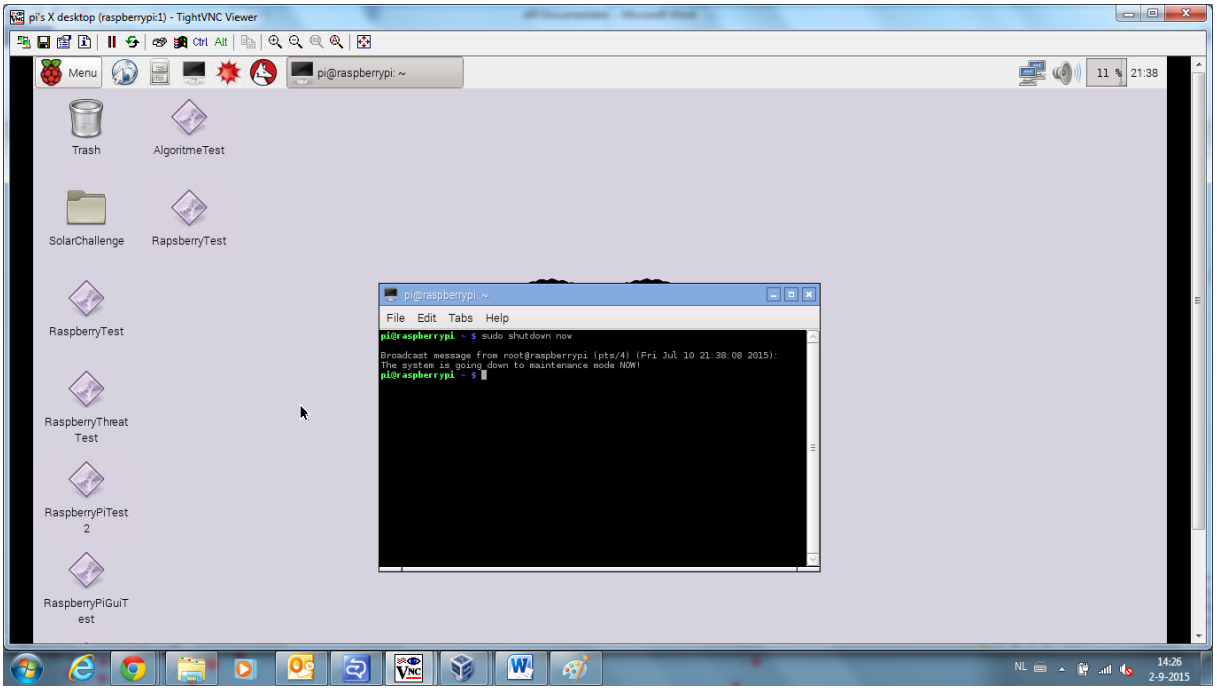
In het volgende scherm wordt naar de Firewall gevraagd, deze optie mag uit. De rest mag op de standaard staan en mag er geïnstalleerd worden. Als de installatie voltooid is kan het programma opgestart worden vanuit het start menu. Het programma heet TightVNC Viewer. Hier kan het IP adres ingevoerd worden gevolgd met het scherm nummer (:0 of :1). Een voorbeeld is 192.168.0.6:1.



Met Connect wordt de connectie open gemaakt. Er zal om het wachtwoord gevraagd worden die opgegeven is toen TightVNC geconfigureerd werd op de Raspberry Pi. Na invulling hiervan zal de Desktop van de Raspberry Pi verschijnen.



Om de verbinding te sluiten kan het scherm afgesloten worden. Om de Raspberry Pi af te sluiten kan sudo shutdown gebruikt worden. Het logout commando hoeft niet gebruikt te worden en zal de connectie niet afsluiten.



10.3 Installeren WiringPi

WiringPi wordt gebruikt om de GPIO aan te spreken. Als WiringPi niet geïnstalleerd is, zullen de controllers niet functioneren. Voor het ontwikkelen van algoritmes is dit niet nodig, omdat alles achter de schermen wordt gedaan. Het algoritme hoeft alleen aan te geven of de pan of tilt moet bewegen en de controller vertaald dat naar de juiste beweging.

WiringPi kan met GIT opgehaald worden. Hiervoor moet Git geïnstalleerd staan. Als dit nog niet gebeurt is kan dit met:

```
sudo apt-get install git-core
```

Mochten errors optreden kan Raspbian geüpdatet worden met:

```
sudo apt-get update

sudo apt-get upgrade
```

Daarna moet git succesvol geïnstalleerd kunnen worden.
Om WiringPi te krijgen is het volgende commando nodig:

```
git clone git://git.drogon.net/wiringPi
```

Mocht deze gebruikt zijn kan er naar de wiringPi folder genavigeerd worden en dan een pull gedaan worden met:

```
git pull origin
```

Om het te installeren is een script beschikbaar. Deze staat ook in de WiringPi folder. Het navigeren en het uitvoeren van de script is als volgt:

```
cd wiringPi

./build
```

Het script installeert alles automatisch en wordt klaar gemaakt voor gebruik. Onthoud dat dit op de Raspberry Pi staat en werkt om programma's op de Raspberry Pi te compileren, om te cross compileren zijn er extra stappen nodig waar later op terug gekomen wordt. Wederom wiringPi is alleen nodig om het programma te draaien, omdat het de communicatie regelt naar de GPIO toe.

Mocht dit niet lukken kan er op de volgende website een snapshot gehaald worden:
https://git.drogon.net/?p=wiringPi;a=summary

De bovenste snapshot is de meest recente. De naam zal vergelijkbaar zijn met WiringPi-98cbc20.tar.gz. Bij dit voorbeeld kan met de volgende commando's wiringPi geïnstalleerd worden:

```
tar xfz wiringPi-98bcb20.tar.gz

cd wiringPi-98bcb20

./build
```

10.4 Opstellen toolchain

De toolchain is een onderdeel dat het mogelijk maakt applicaties te maken voor de Raspberry Pi vanuit de desktop pc. Dit stappenplan gebruikt de officiële toolchain voor Raspbian. Om te beginnen instaleer de essentiële onderdelen met het commando:

```
Fedora: yum install make automake gcc gcc-c++ kernel-devel git
Debian: sudo apt-get install build-essential git
```

Nu zijn de essentiele onderdelen nodig voor het ontwikkelen en compilen voor C++ code. Tevens wordt ook Git geïnstalleerd, hiermee zal de toolchain gedownload worden.

Nu zal de toolchain gedownload worden. Het is aangeraden om deze in een makkelijk te vinden folder te zetten, zodat deze later makkelijk te vinden is. Voor dit voorbeeld maken we een directory in de home folder. Dit kan met mkdir of via de grafische interface:

```
mkdir rpi
```

Vervolgens gaan we naar deze directory:

```
cd rpi
```

Om dan de toolchain van github te halen:

```
git clone git://github.com/raspberrypi/tools.git
```

(Git is niet nodig om dit te kunnen doen, het kan ook direct gedownload worden in zip formaat van https://github.com/raspberrypi/tools)

Hierdoor zijn de officiële Raspbian cross compile toolchains opgehaald. Als alles goed gegaan is staat er nu een folder tools met daarin arm-bcm2708. In de folder arm-bcm2708 zijn de volgende folders te vinden:

1. Arm-bcm2708-linux-gnueabi
2. Arm-bcm2708hardfp-linux-gnueabi
3. Gcc-linaro-arm-linux-gnueabihf-raspbian
4. Gcc-linaro-arm-linux-gnueabihf-raspbian-x64

Dit kan gecontroleerd worden door naar de folder te cden:

```
cd ~/rpi/tools/arm-bcm2708
of
cd /home/fedora/rpi/tools/arm-bcm2708
```

De derde en de vierde toolchain zijn voor de Raspberry Pi. De derde is voor 32 bit Linux en de vierde voor 64 bit Linux. Als de versie niet bekend is kan dat achterhaald worden met het volgende commando:

```
uname -a
```

Als in de output i386/386 staat dan wordt er gebruik gemaakt van een 32 bit versie van Linux. Als er x86_64 / amd64 in de output staat dan is het een 64 bit versie van Linux.

10.5 Maken van een project en deployen naar de Raspberry Pi

Voor dit project is gebruik gemaakt van Eclipse. Eclipse is een gratis programma waarmee geprogrammeerd kan worden in verschillende talen. Eclipse is afhankelijk van java dus installeer voor alle zekerheid de laatste versie. De volgende commando's zouden voldoende moeten zijn.

```
Fedora: yum install java
Debian: sudo apt-get install openjdk-7-jre
```

Eclipse CDT kan vanaf internet gedownload worden. Het resultaat is een zip bestand. Deze kan unzipped met het tar commando of de interface. Eclipse hoeft niet te installeren. Het kan gebruikt worden door de Eclipse applicatie te starten in de folder waar Eclipse naar uitgepakt is. Eclipse kan gestart worden vanaf de terminal met:

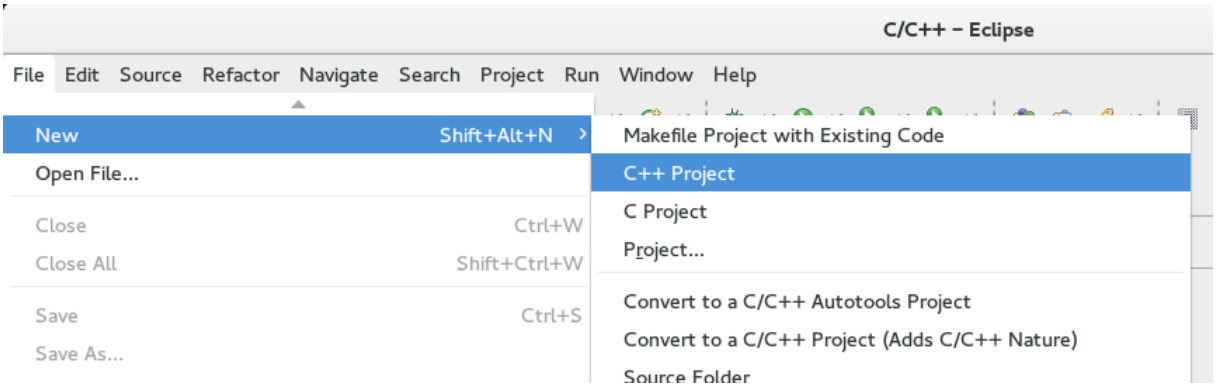
```
./eclipse &
```

Houd wel rekening dat je naar de folder moet gaan met CD.

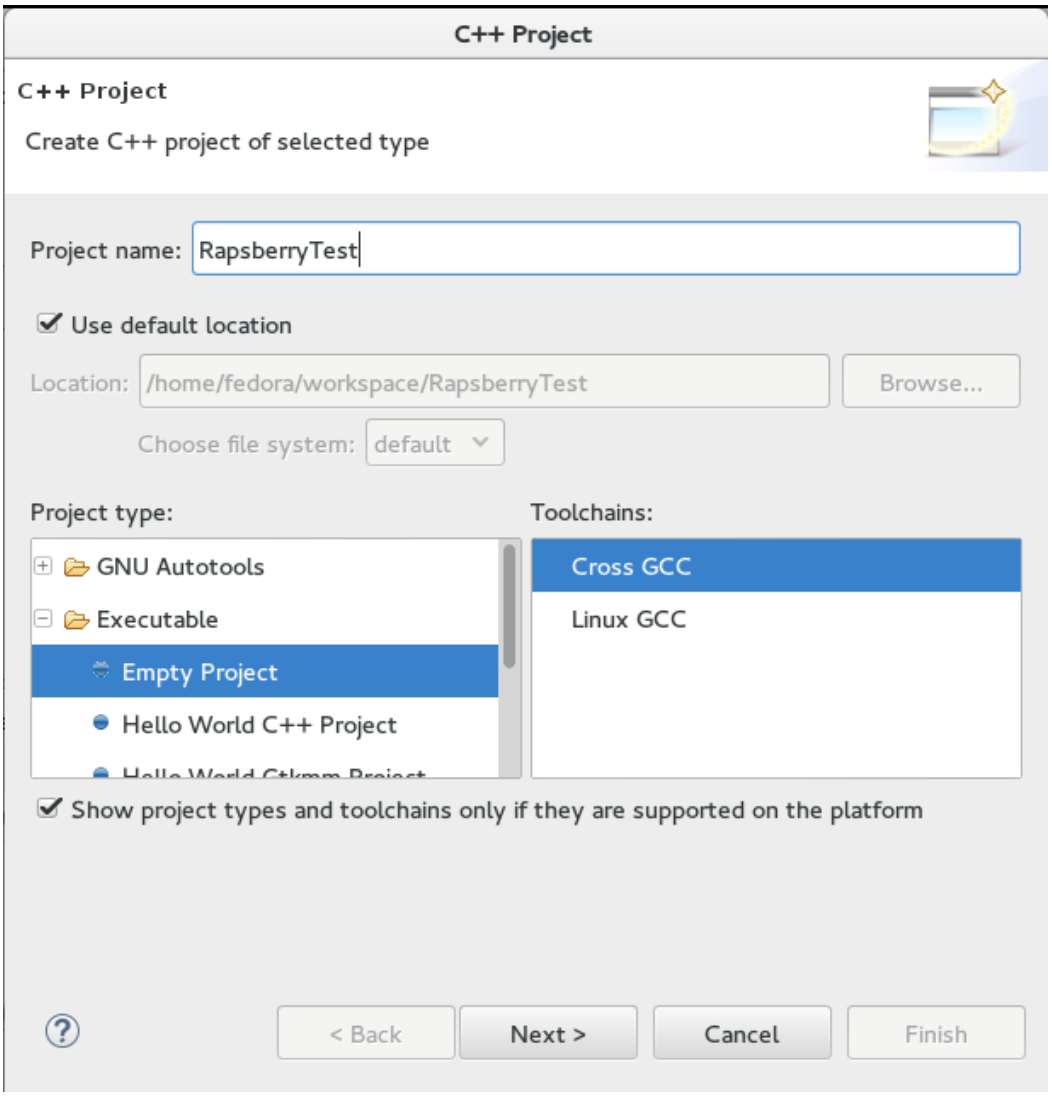
```
cd ~/eclipse
```

Het voorbeeld installeert eclipse in de home folder.

Zodra Eclipse opgestart is kan een nieuwe project aangemaakt worden. Het is aangeraden om het welkom scherm af te sluiten, omdat deze vaak in de weg zit. Om een nieuw project aan te maken klik je op File->New->C++ Project.

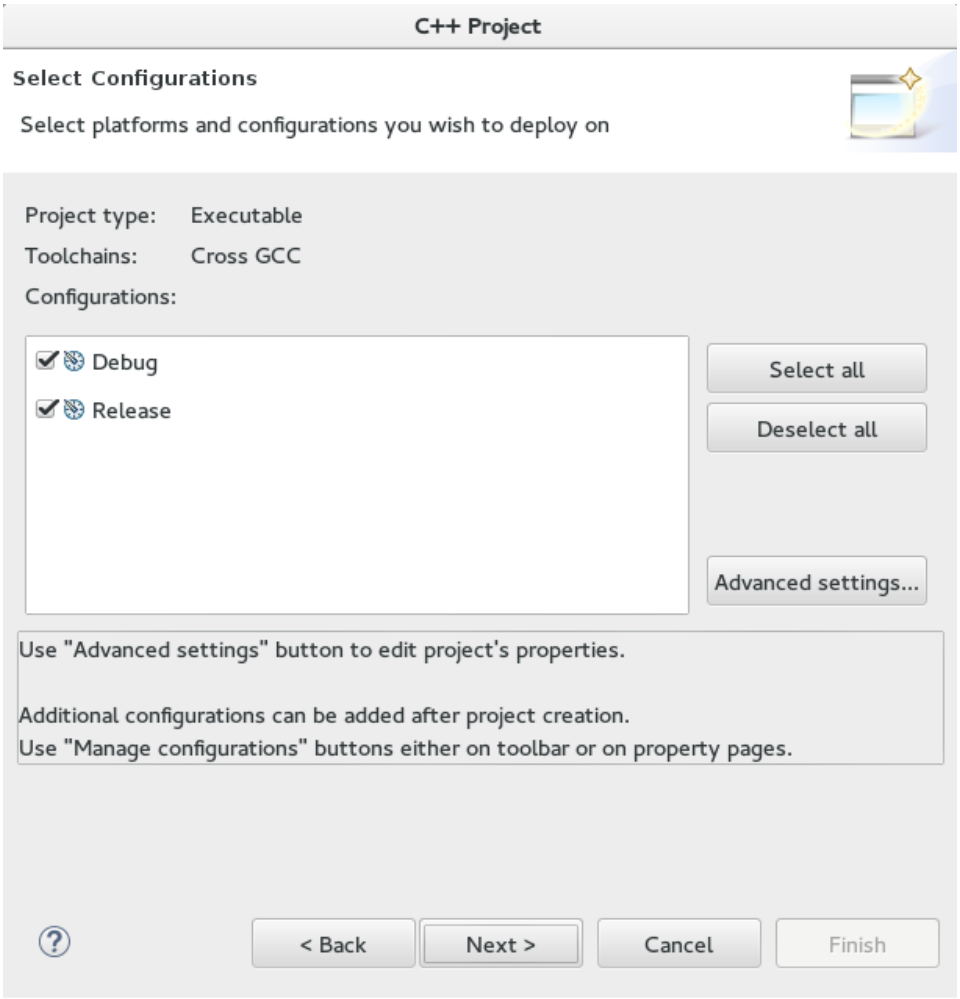


Vervolgens moet het project type geselecteerd worden. Dit voorbeeld neemt een Executable->Empty Project. Geef het project een naam en selecteer Cross GCC onder toolchains.



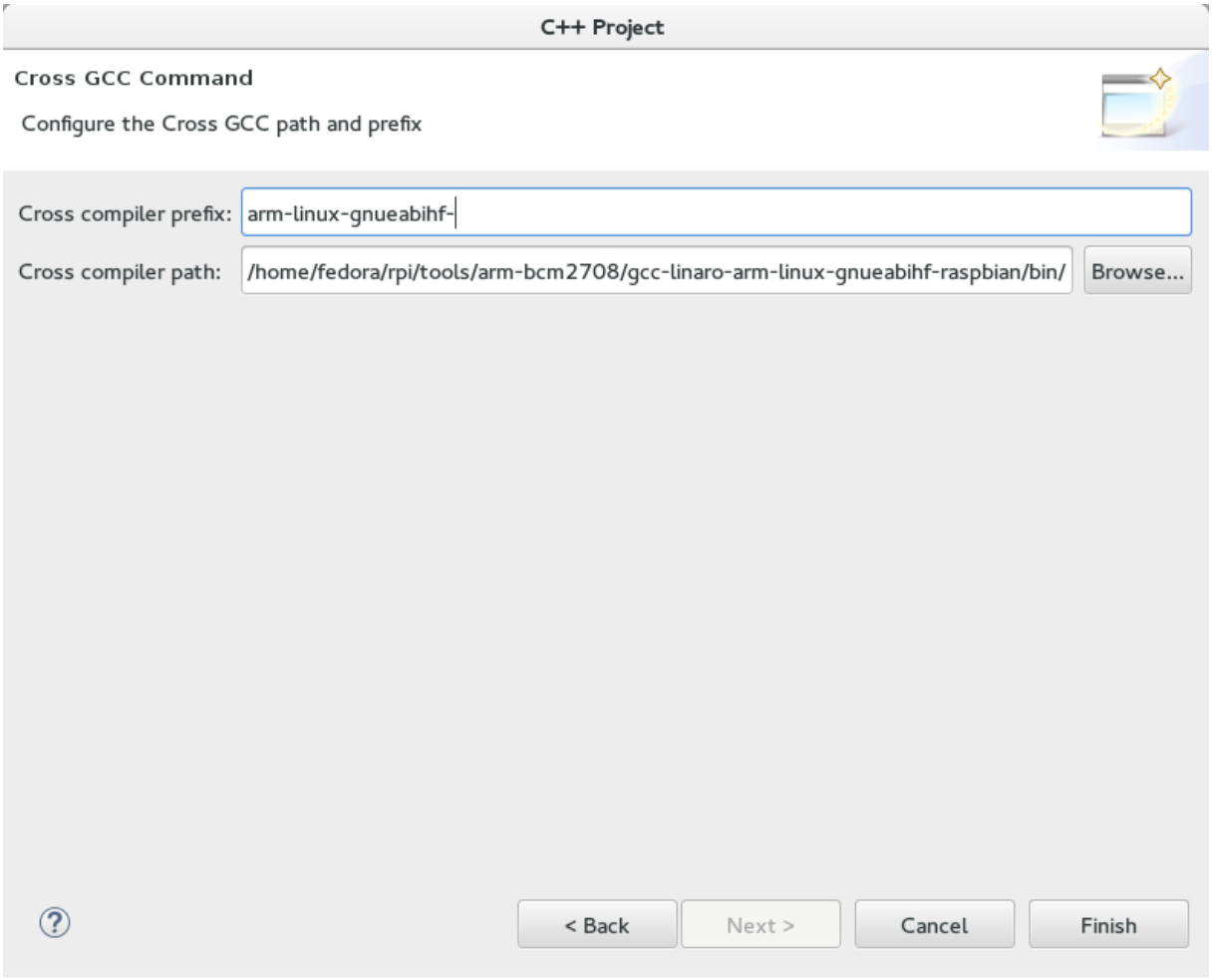
Klik op next om verder te gaan.

Bij het configurations scherm mag alles op default blijven staan.



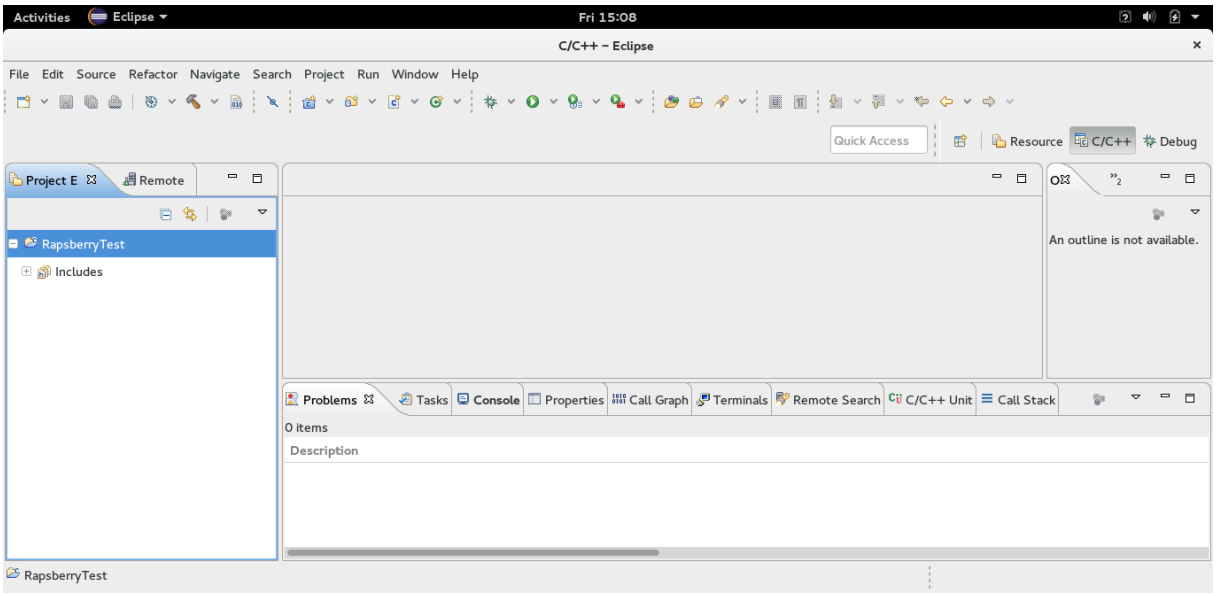
Nu komt het scherm waarbij Eclipse vraagt om een Cross Compiler prefix en een path. Deze velden geven aan welke commando of toolchain uitgevoerd moet worden bij het compilen en waar deze gevonden kan worden.

De prefix die nodig is is “arm-linux-gnueabihf-”. Vergeet niet de - aan het einde, want die telt zeker mee. De Cross Compiler path is te vinden op pad waar de toolchain geinstalleerd staat. /home/[user]/toolchain/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspian/bin/ is het pad voor het voorbeeld project.

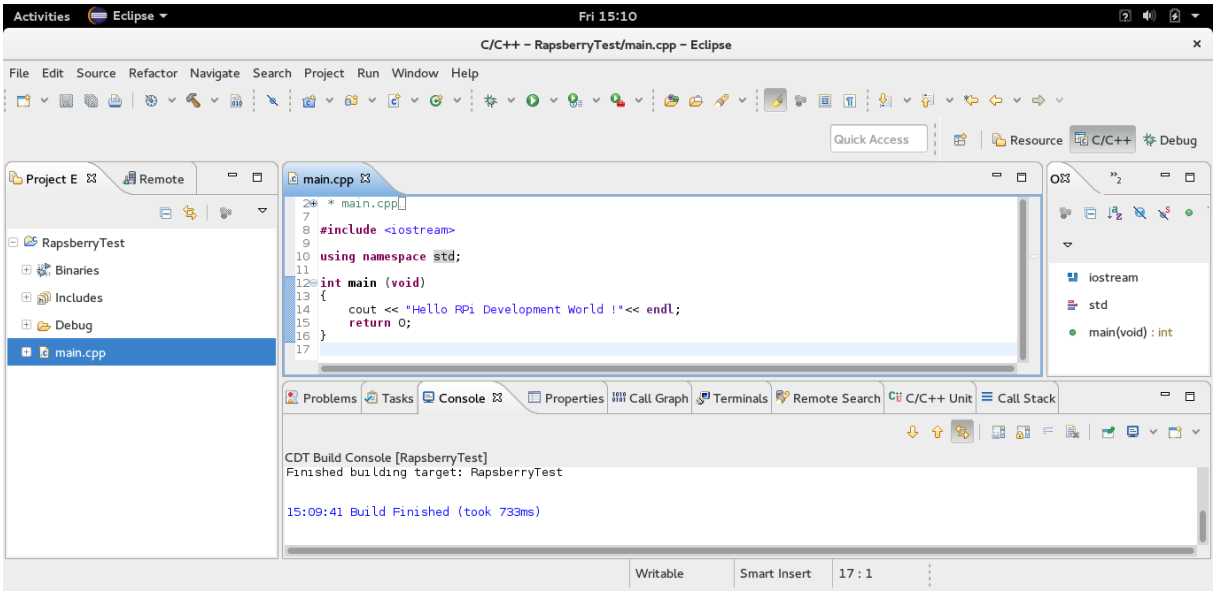


Klik op Finish zodra de toolchain opties opgegeven zijn. Nu zal Eclipse de locatie van de toolchain ophalen om alle libraries en headers te pakken. Deze worden gebruikt bij het compilen van de programma’s.

Als alles goed gegaan is, komt er een leeg project te staan met verder geen source files.

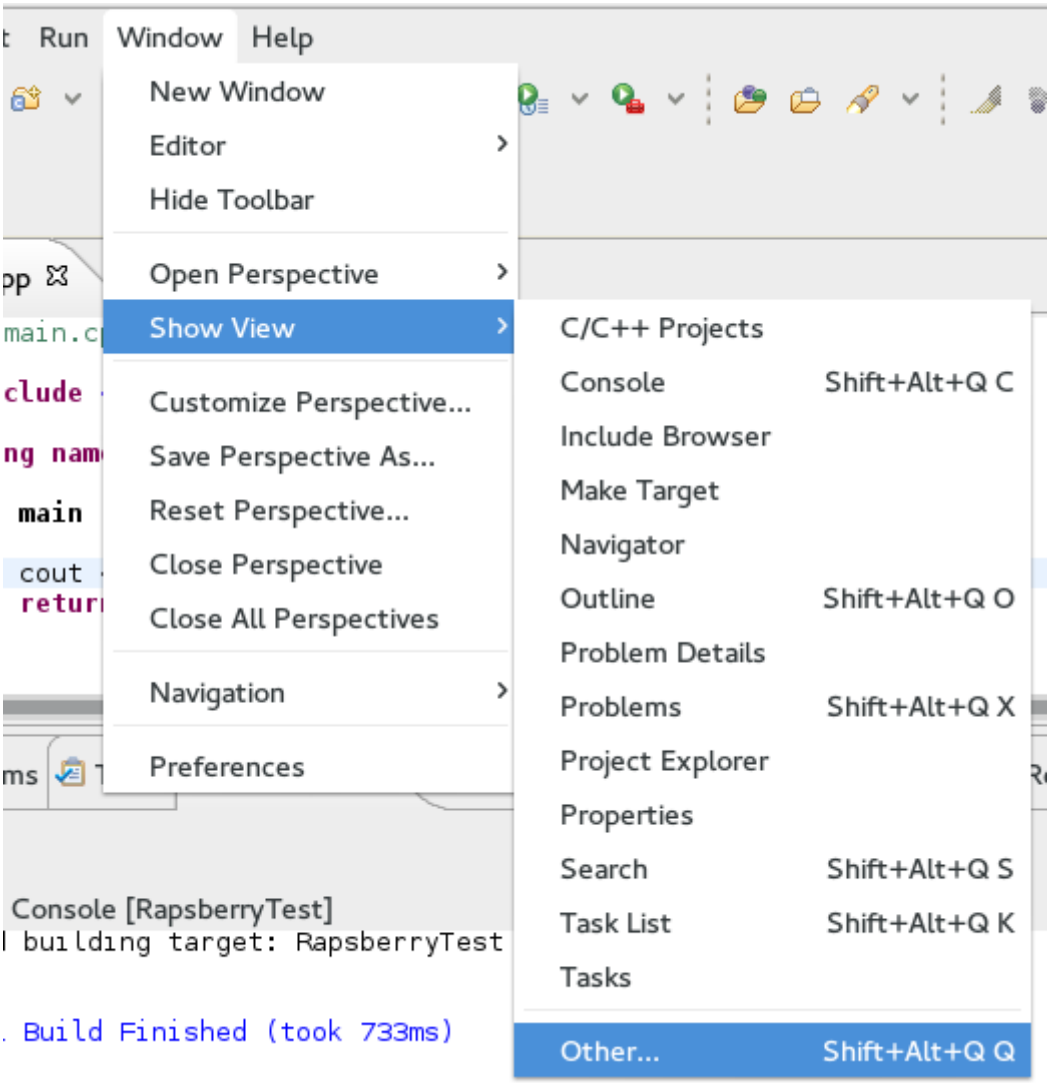


Laten we hier een bestand aan toevoegen om te zien of alles werkt. Dit kan door File->New->Source File, of door rechtermuisknop op het project te klikken en dan New->Source File. Geef een leuke naam en voeg de standaard hello world code toe. Probeer deze te builden en als alles gelukt is zal het slagen. (NOTE: de eerste keer kan even duren, omdat eclipse alle references en bestanden probeer te vinden)

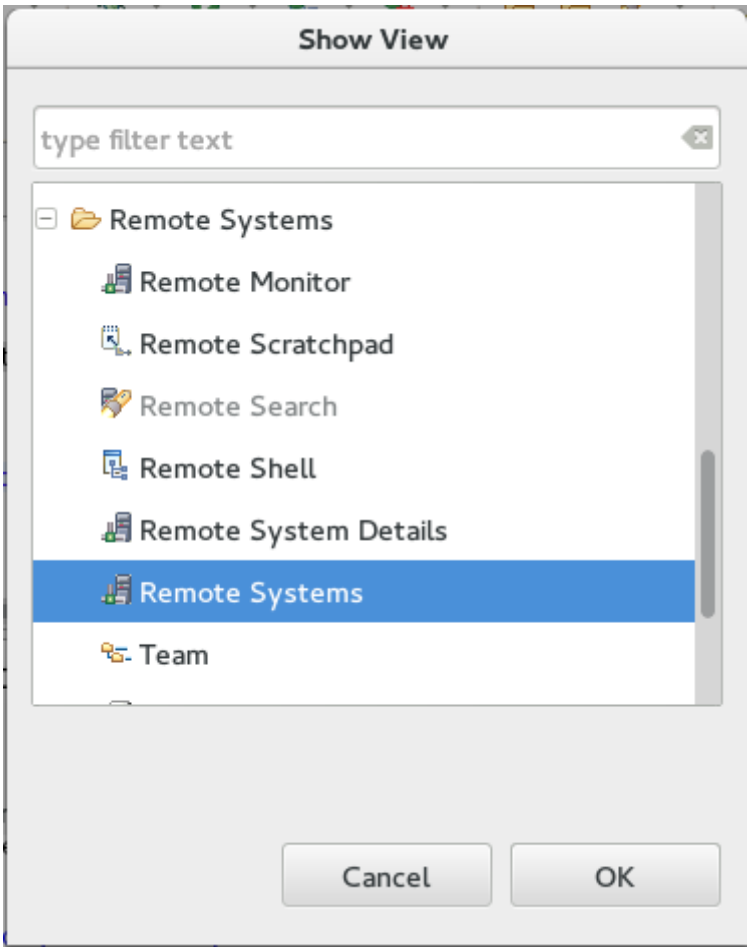


Nu er een executable is die werkt voor de Raspberry Pi moet het bestand nog wel op de Raspberry Pi gezet worden. Dit kan via meerdere methoden, maar de snelste om te gebruiken is via een plugin voor Eclipse. Eclipse heeft een plugin met de naam Remote Systems Explorer. Deze kan al geïnstalleerd zijn, maar als dat nog niet gebeurt is kan met Help->Install New Software… gezocht worden naar Remote Systems Explorer. Eventueel kan ook de Market place geraadpleegd worden om het te installeren.

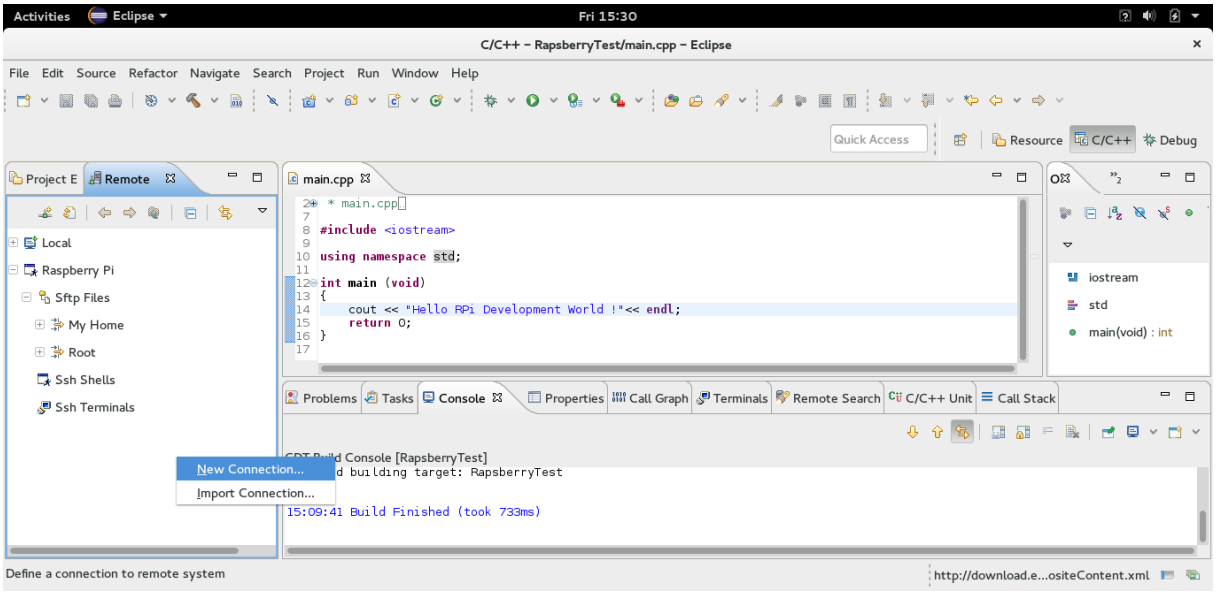
Zodra deze geïnstalleerd is kan met Window->Show view->Others… een scherm geopend worden.



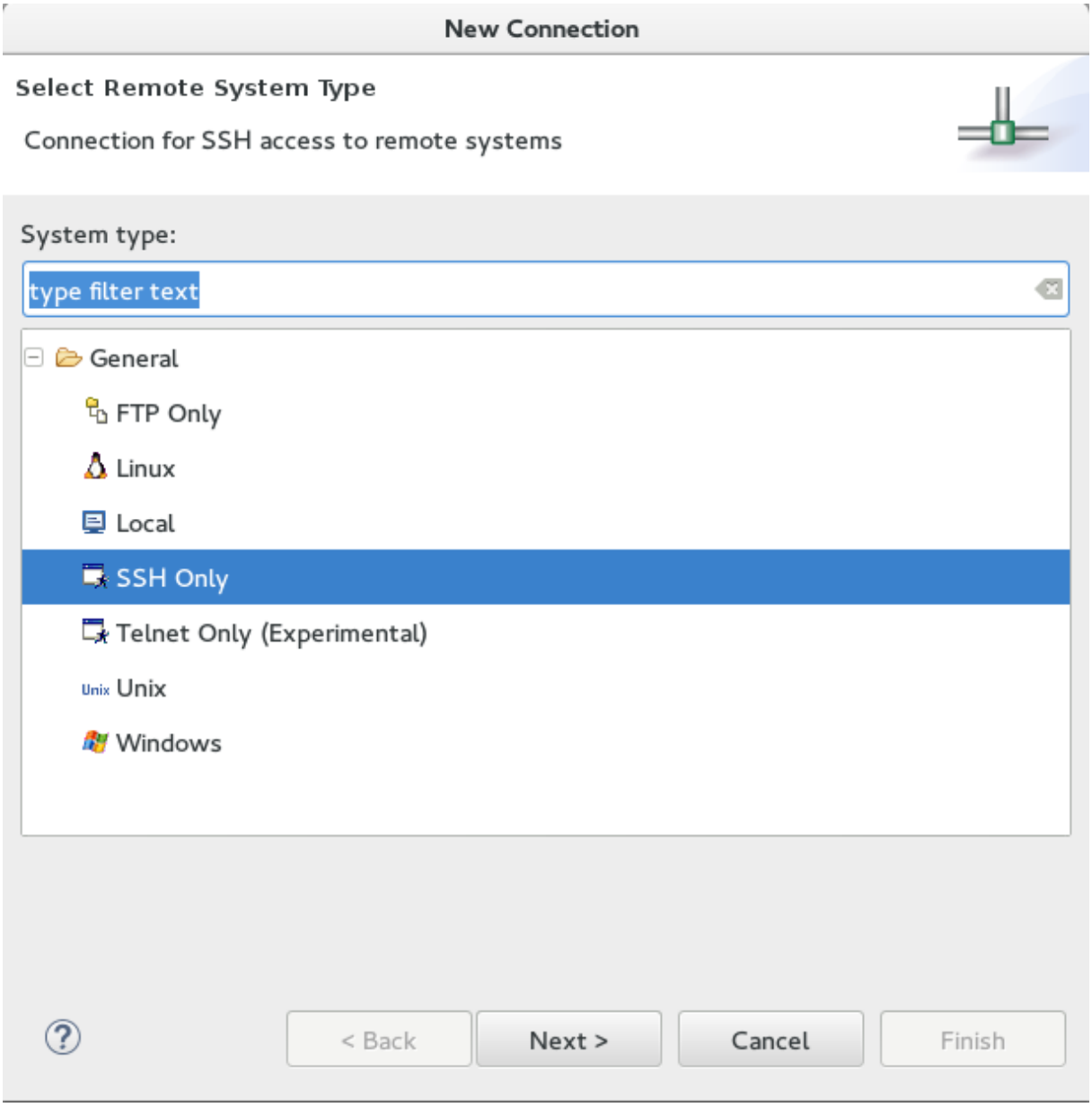
In dit scherm kan het Remote Systems scherm geopend worden.



Standaard komt deze aan de bodem te staan, deze kan verplaatst worden naar de zijkanten om het eventueel leesbaarder te maken. Open het remote systems tab en rechtermuisklik in het scherm en selecteer New Connection.

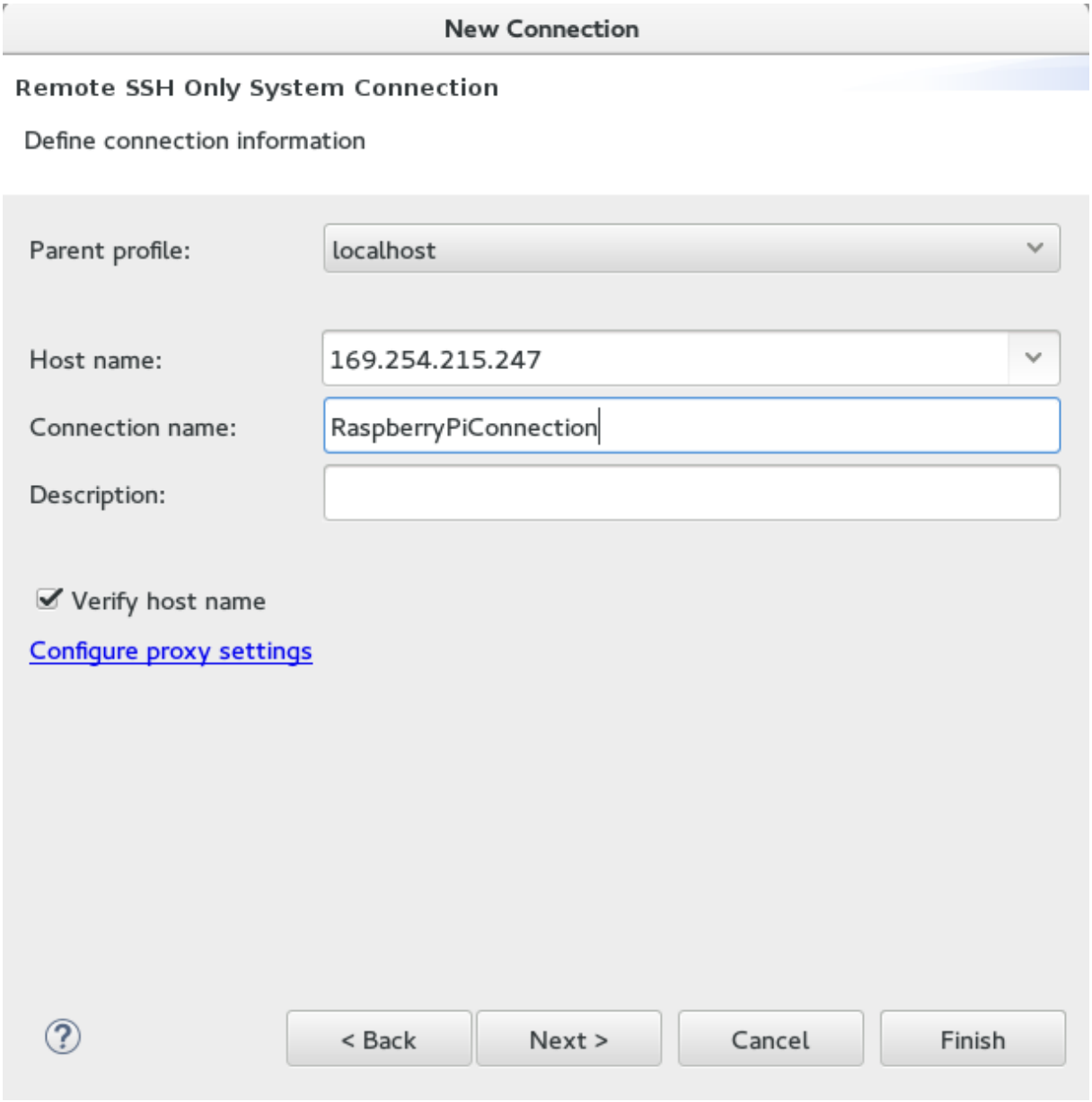


Dit opent een nieuw scherm om een Remote Connectie te maken.



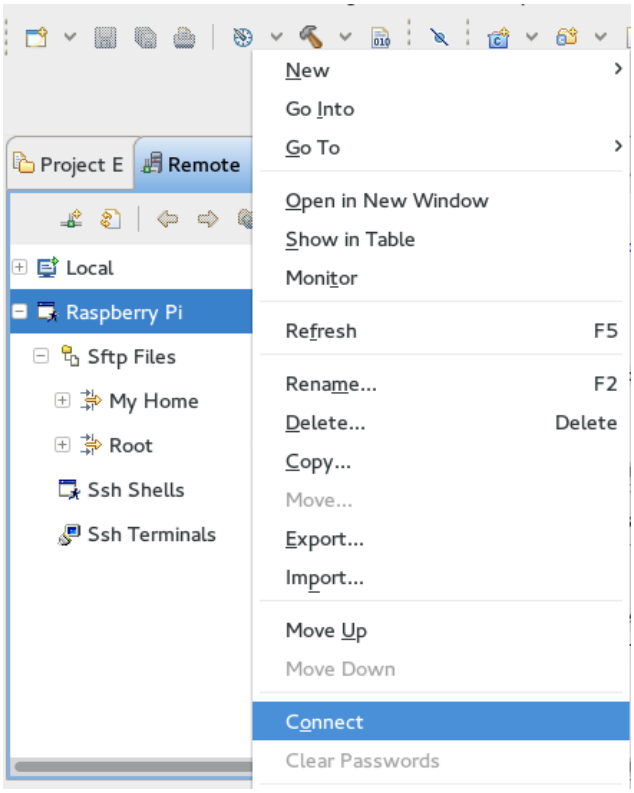
Voor de Raspberry Pi gaan we een SSH Only connectie gebruiken, omdat we alleen bestanden over hoeven te zetten en hooguit de terminal willen raadplegen.

In het volgende scherm moet een IP adres gegeven worden en kan een naam toegevoegd worden om de connectie herkenbaar te houden.

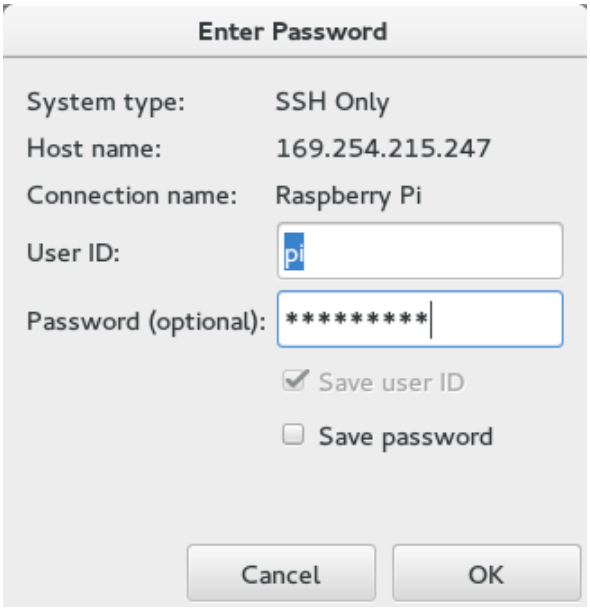


Er kan met next meerdere opties en configuraties doorgelopen worden, maar alle defaults moeten voldoende zijn en betekent dat er nu afgesloten kan worden met Finish.

Nu zou er een tweede connectie moeten zijn in het Remote Systems scherm. Als er met de rechtermuisknop op geklikt wordt kan de connectie opgestart worden met Connect.

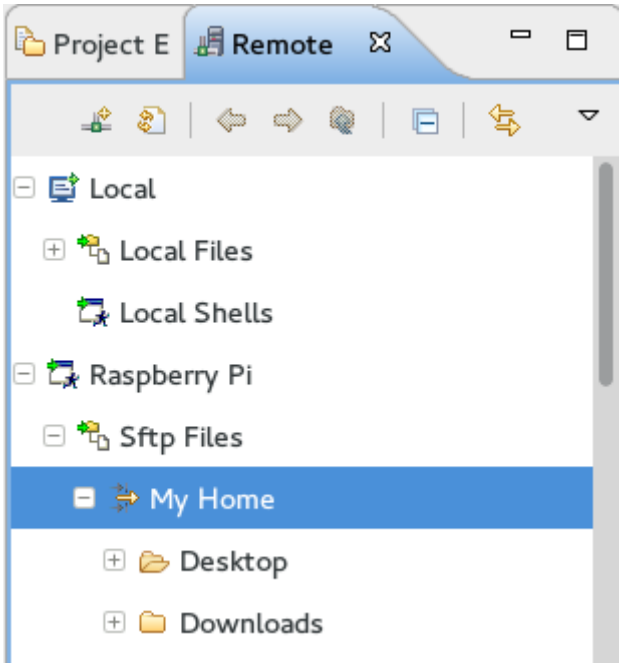


Eclipse zal dan met SSH een connectie proberen te maken en zal vragen om credentials om in te kunnen loggen op de Raspberry Pi. Dit zijn de inlogcredentials van de Raspberry Pi en niet van Eclipse of de host computer.

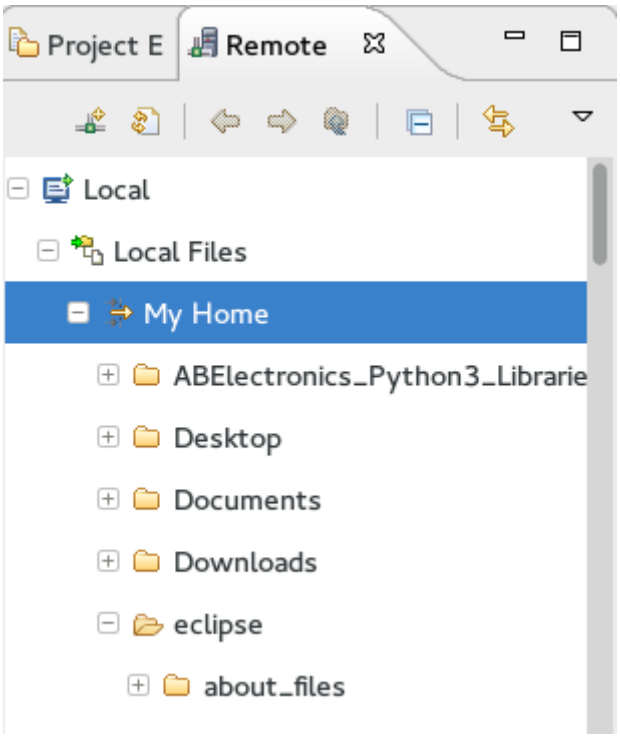


Het is mogelijk dat een waarschuwing komt om zeker te zijn dat de verbinding vertrouwt wordt. Deze kan bevestigd worden, want we proberen met een eigen connectie verbinding te krijgen.

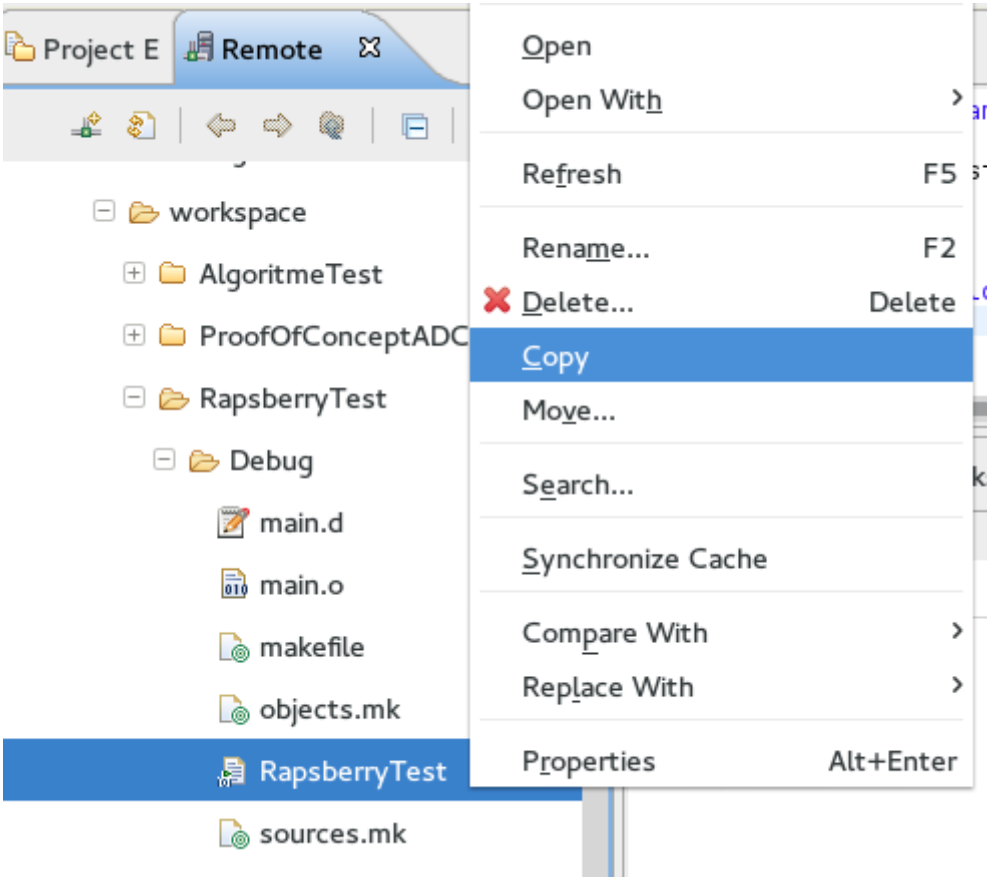
Op dit punt zou de connectie klaar voor gebruik moeten zijn. Dit kan bevestigd worden door de folder structuur te navigeren van de Raspberry Pi.



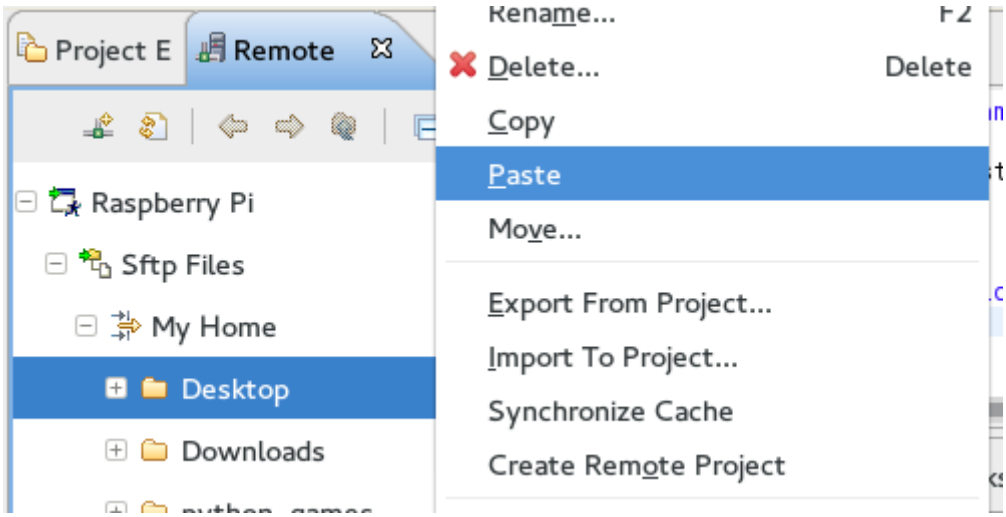
Tevens kan ook de folderstructuur op de lokale machine genavigeerd worden in hetzelfde scherm.



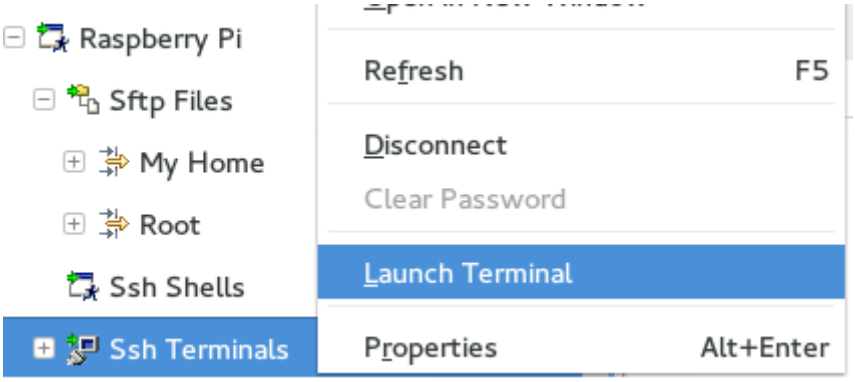
Zoek naar de binary file van het project die net gemaakt is.



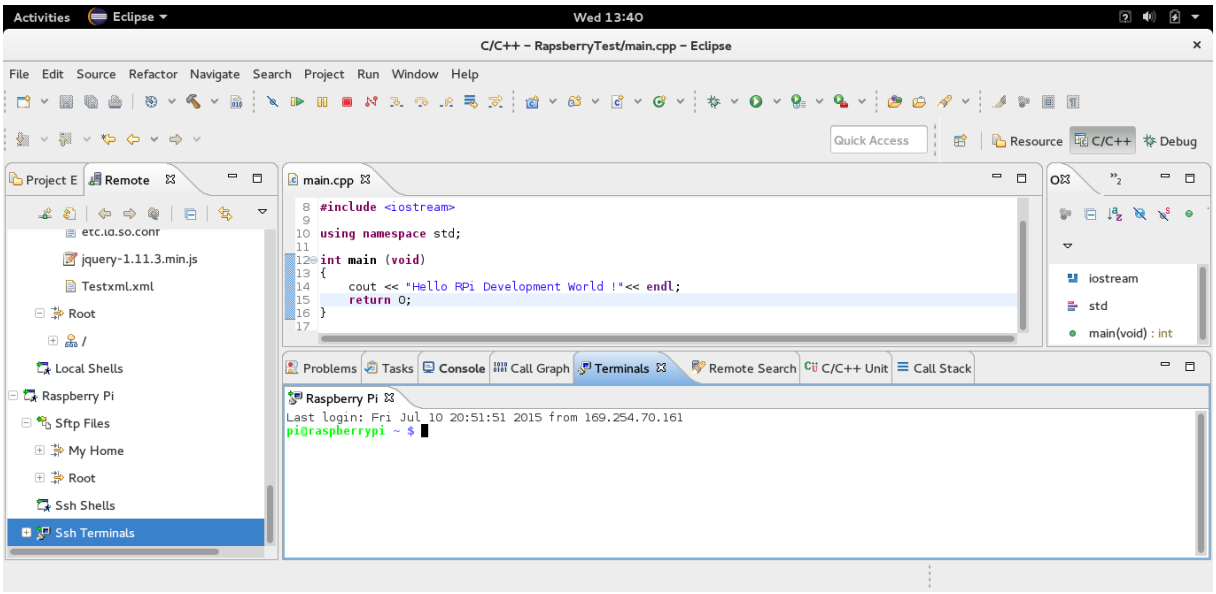
Kopieer deze in Remote Systems van de lokale machine naar de Raspberry Pi. Eclipse kan op deze manier bestanden via SSH oversturen naar de remote machine.



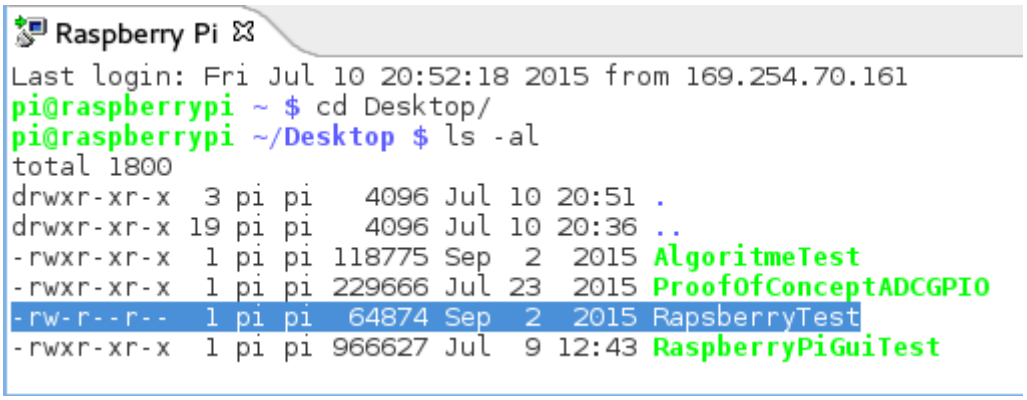
Om het programma te draaien kunnen we de terminal gebruiken. Dit kan ook via Remote Systems. Onder de Raspberry Pi node staat een SSH terminals deel. Met de rechtermuisklik op dit deel kan een nieuwe terminal geopend worden.



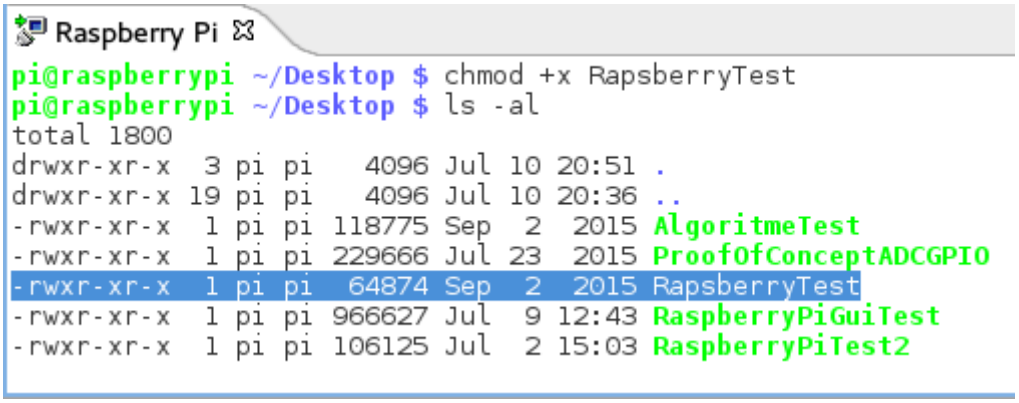
De terminal opent standaard in de home folder van de gebruiker (default: pi).



Het voorbeeld is gezet in de desktop dus daar moet heen genavigeerd worden om het programa vervolgens te draaien. Met ls –al kan bevestigd worden dat de binary op de Raspberry Pi staat, maar deze heeft weinig rechten ontvangen bij overdracht.



Om deze executable te maken op de Raspberry Pi kan met chmod +x [File name] het bestand executable gemaakt worden.



Als laatste kan met ./[File name] het bestand uitgevoerd worden. Als alles goed gegaan is kan er nu in de terminal window “Hello Raspberry World” gezien worden.

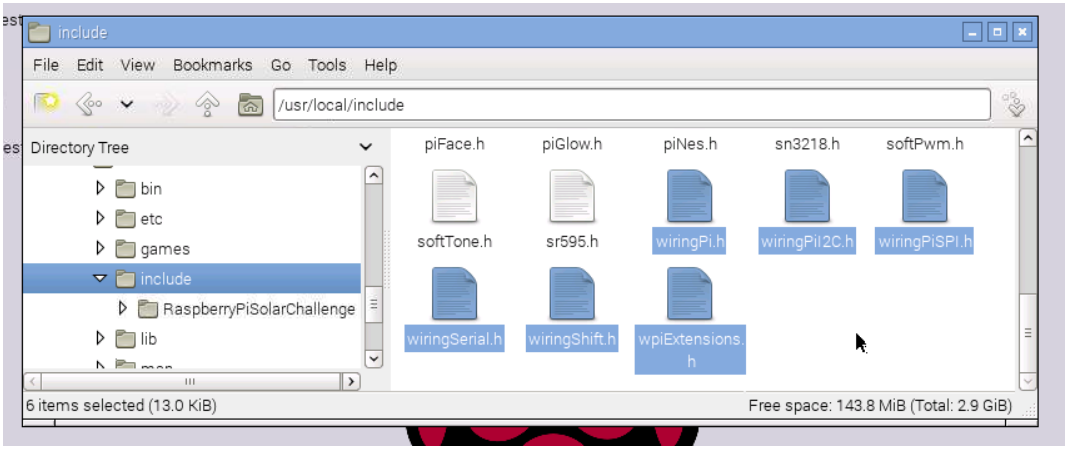
```
pi@raspberrypi ~/Desktop $ ./RapsberryTest
Hello RPi Development World !
pi@raspberrypi ~/Desktop $
```

Note: Dit project is vrij standaard en basic. Bij gebruik van WiringPi moet een library gelinked worden. Belangrijk hierbij is dat dit NIET nodig is voor het maken van algortimes en alleen nodig is al seen nieuw project op vergelijkbare manier opgezet wil worden. In principe hoeft er niks gedaan te worden om het bestaande project te veranderen omdat daar de link al gelegd is.

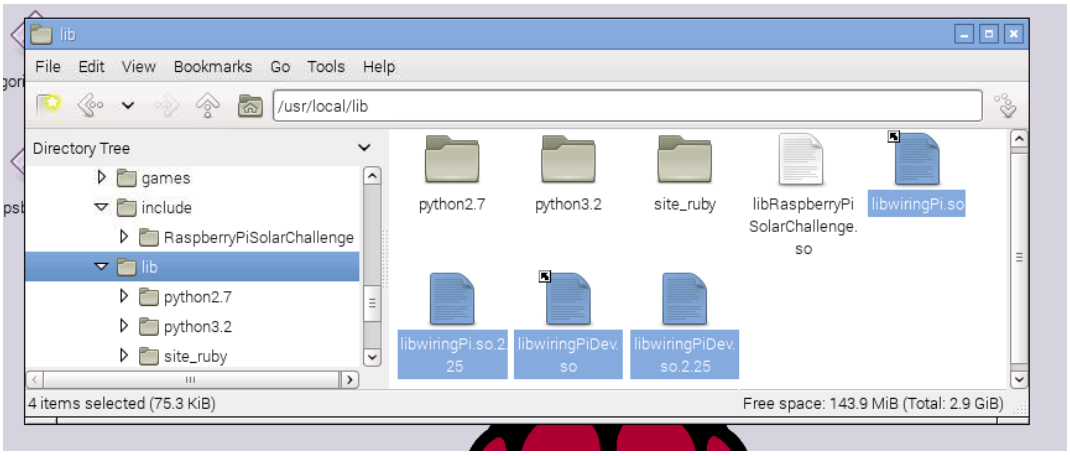
Allereerst moet WiringPi geinstalleerd staan op de Raspberry Pi. Dit is behandelt in het oorspronkelijke hoofdstuk voor WiringPi. De libraries en headers moeten vervolgens gekopieerd worden vanaf de Raspberry Pi naar dezelfde folders op de Fedora machine (of andere OS).

Headers en Libs Raspberry Pi:

Header files:

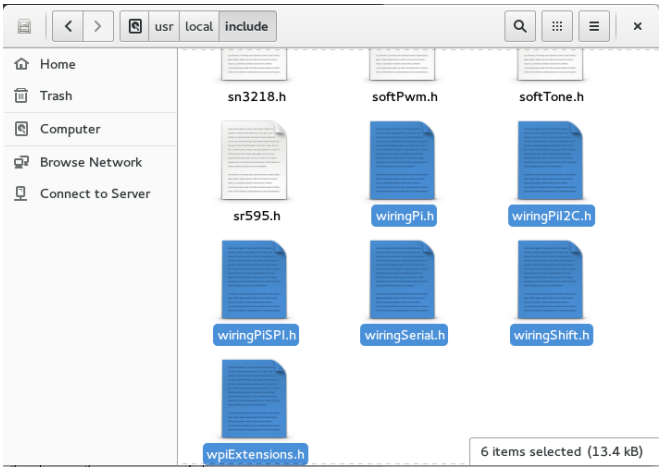


Libaries:

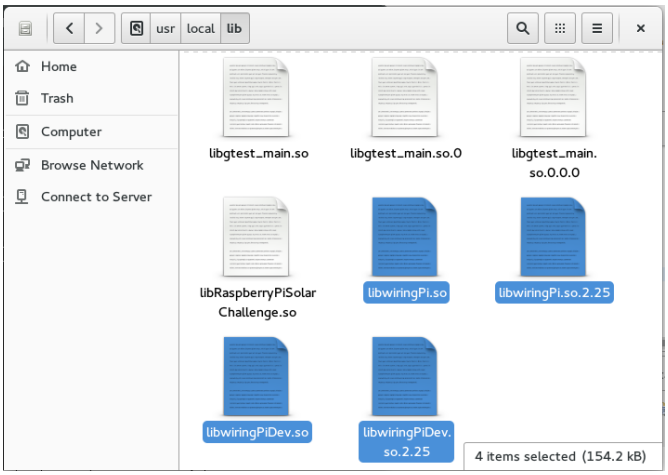


Deze moeten naar dezelfde folder op de ontwikkelomgeving. Hieronder het voorbeeld van de Fedora machine.

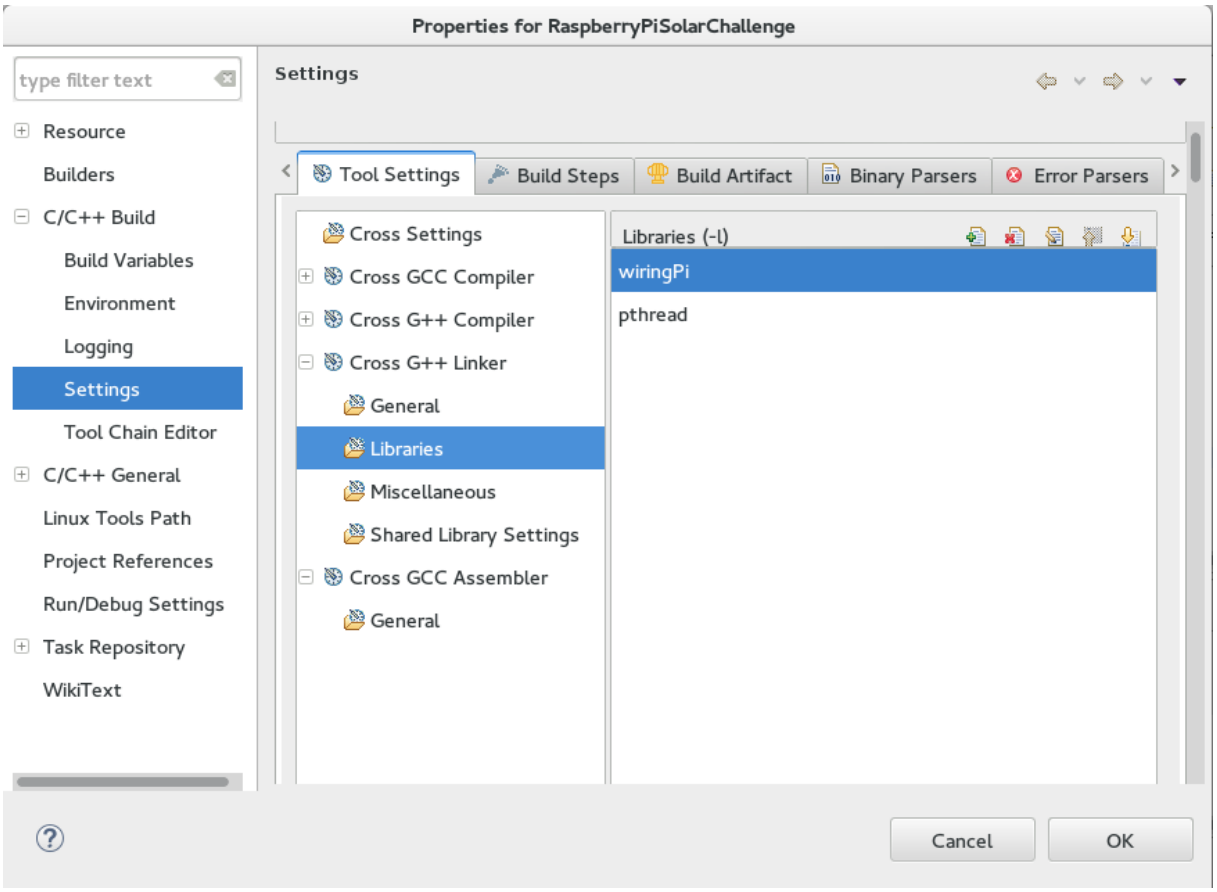
Header files:



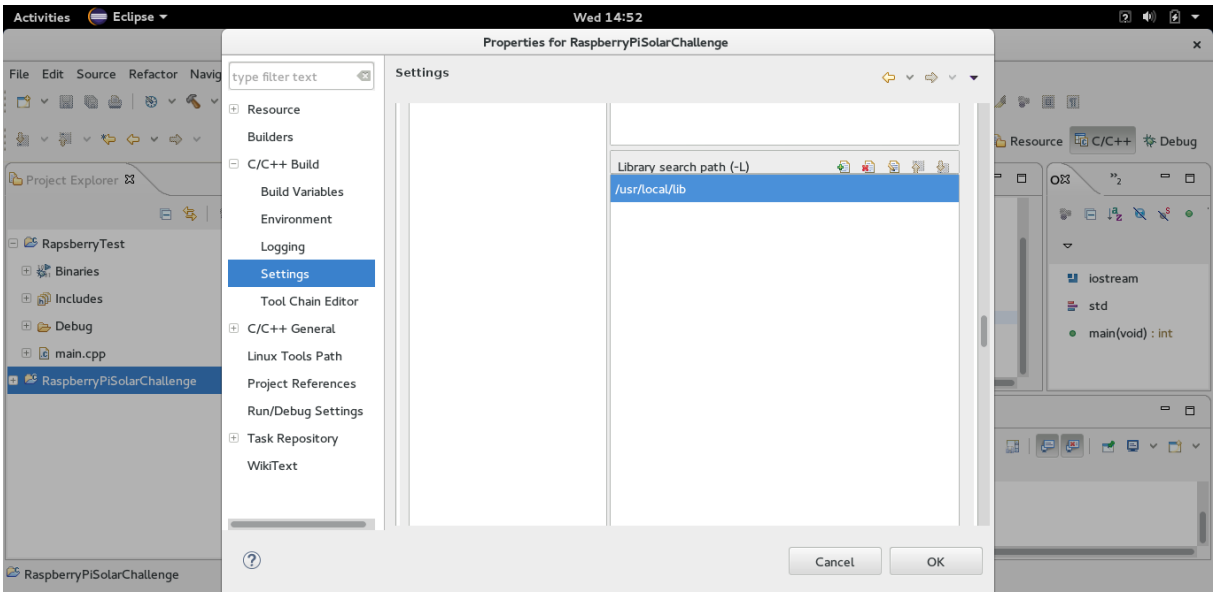
Libaries:



Verder moet in Eclipse alleen de library gelinked worden. Dit gebeurt in de properties van een project. Hier moet ‘wiringPi’ toegevoegd worden en het is hoofdletter gevoelig. Er hoeft geen extensie of volledige namen gegeven worden, dat kan Eclipse zelf uitzoeken.



Het is goed mogelijk dat Eclipse de WiringPi library niet vind. In deze situatie moet een search path op gegeven worden. Dit kan in het zelfde scherm, maar een stukje naar beneden. De libraries zitten in /usr/local/lib/



Als de libraries gevonden worden maar errors opleveren is er een compatibility fout. Het is aannemelijk dat de verkeerde libraries zijn gekopieerd naar de ontwikkelomgeving.

10.6 Debuggen van projeten

Debuggen is voor veel programmeurs essentieel. Eclipse biedt de mogelijkheid om automatisch en handmatig de debugger op te stellen. Automatisch wordt alles door Eclipse geregeld en hoeft de gebruiker weinig input te doen, maar als de software root rechten nodig heeft dan heeft Eclipse dat ook nodig. De andere optie is handmatig en deze is geadviseerd omdat daar meer controle over is. Het is aangeraden om eerst de cross compiler werkend te krijgen voordat de debugger geconfigureerd wordt voor remote debuggen. Om het gemakkelijk te houden passen we het voorbeeld project aan met een loop voor demonstratie van de debugger.

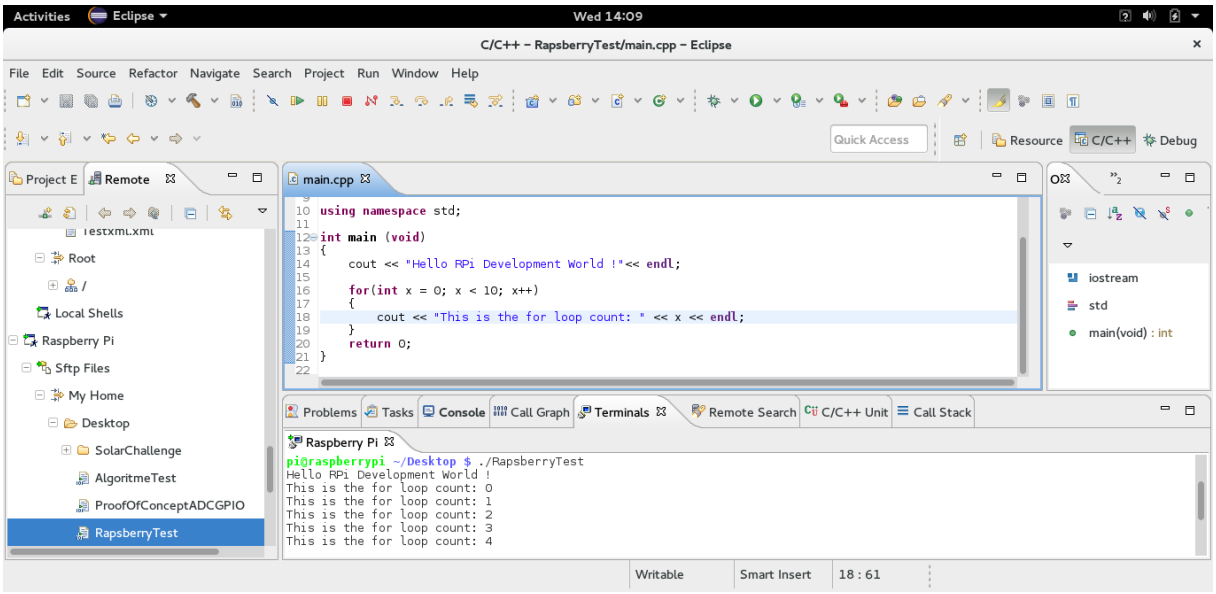
Eclipse gebruikt GDB server om remote te debuggen. Zoals hiervoor genoemd kan dit automatisch en handmatig. Dit stappenplan richt zich op het handmatige debuggen.

Om zeker te zijn dat de debugger werkt kan met het toolchain path gecontroleerd worden of GDB server klaar is voor gebruik.
/home/[user]/toolchain/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-gdb –v

Als dat commando succesvol draait is GDB klaar voor gebruik. Als deze niet klaar is voor gebruik kan het zijn dat er een update nodig is of een bestand moet geinstalleerd worden. Yum update / apt-get update of yum install / apt-get install.

Om handmatig te kunnen debuggen moet het programma deployed zijn op de raspberry pi. Tevens moet deze executable zijn. Dit is uitgelegd in het maken en deployen van projecten in Eclipse voor de Raspberry Pi. Als dit mogelijk is en begrepen wordt kan het debuggen opgesteld worden. Er wordt dus aangenomen dat de lezer de voorgaande onderdelen begrijpt.

Het eerdere project is licht aangepast om debuggen makkelijker te demonstreren. Vanzelfsprekend wordt deze overgezet naar de Raspberry Pi en werkend gekregen volgens de behandelde methode.



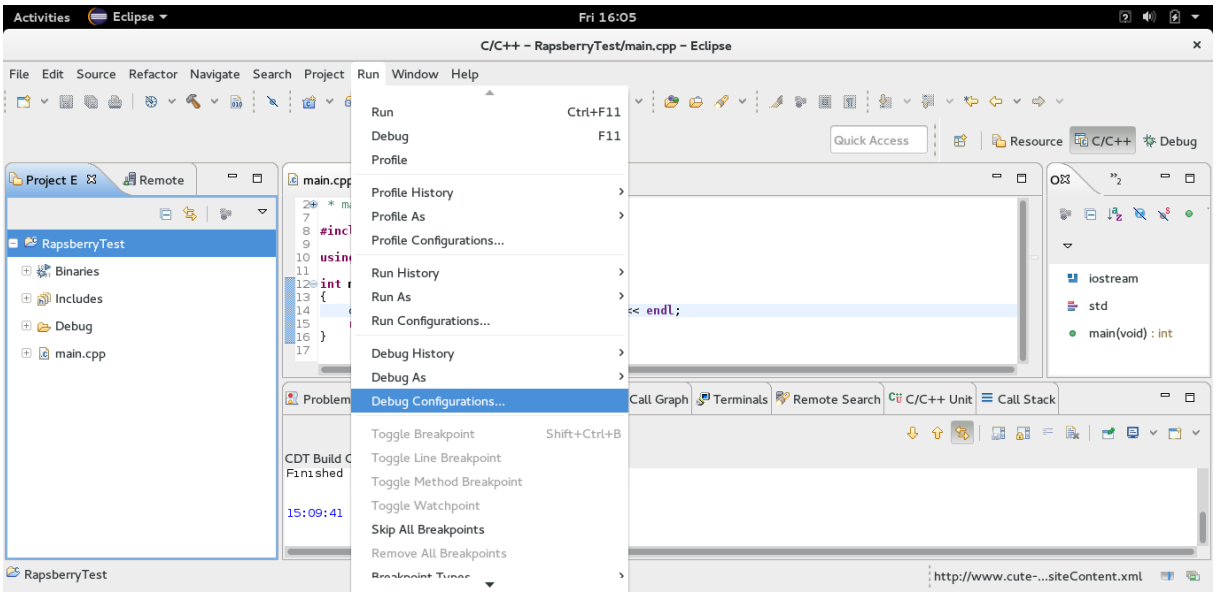
Als laatste moet GDB server opgestart worden met ‘’sudo gdbserver ipadres:port HelloRPi’’ Dit commando moet uitgevoerd worden op de Raspberry Pi, omdat hier het programma zal draaien. Dit kan ook vanaf de terminal gebeuren.

```
pi@raspberrypi ~/Desktop $ sudo gdbserver 169.254.215.247:12345 RapsberryTest
Process RapsberryTest created; pid = 2769
Listening on port 12345
```

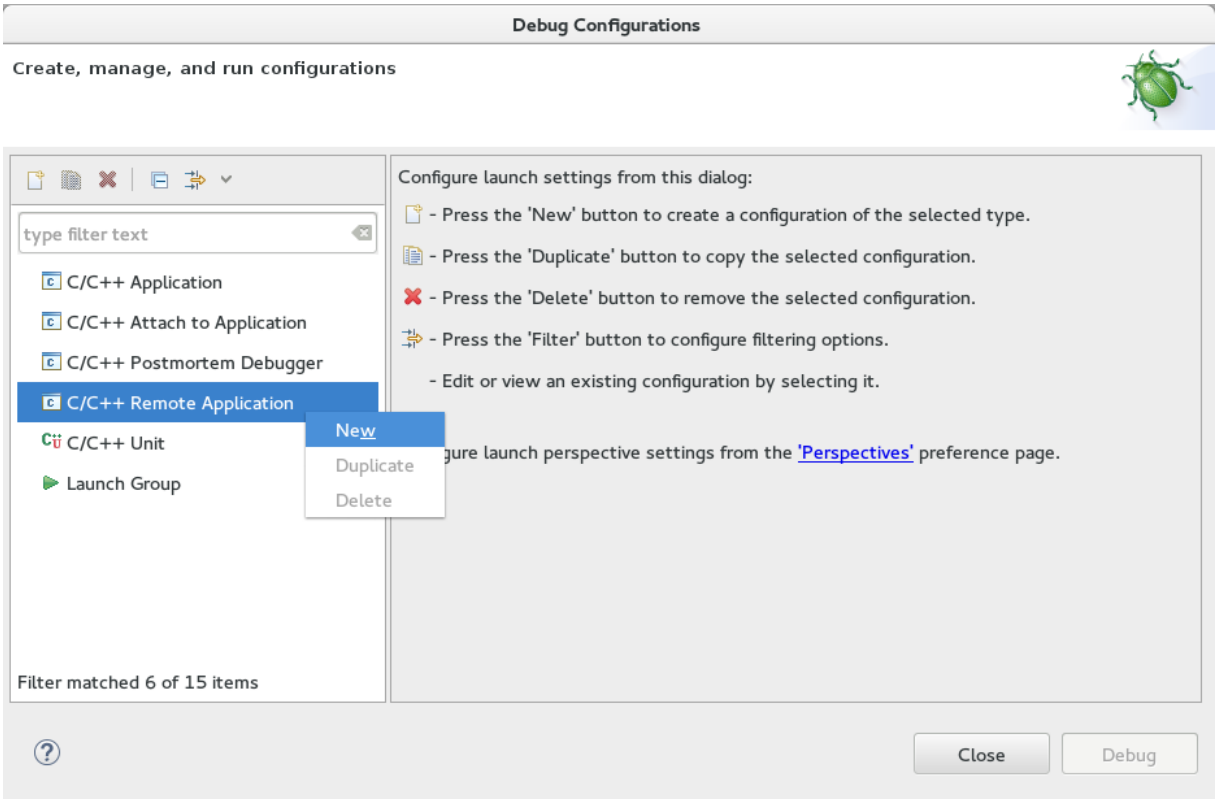
Voor een lokaal netwerk met maar 1 aansluiting kan hoeft het ip-adres niet te kloppen, want hier wordt het Ip adres van de Raspberry Pi gebruikt en zal alsnog functioneren. Dit is een lichtelijke magie van GDB en Eclipse, maar hij vangt hem goed af zolang alles goed ingesteld is aan de Eclipse kant.

Het GDB commando bestaat uit 3 delen. Sudo is optioneel maar wel aangeraden als de GPIO gebruikt wordt in het programma. GDB server luistert op een Ipadres en een poortnummer. Het Ipadres is het ipadres waar de host computer staat met Eclipse. Het poortnummer kan elk getal zijn zolang het geen poortnummers zijn die door andere applicaties wordt gestart zoals 80(http) en 22(ssh). Het laatste deel is de bestandsnaam.

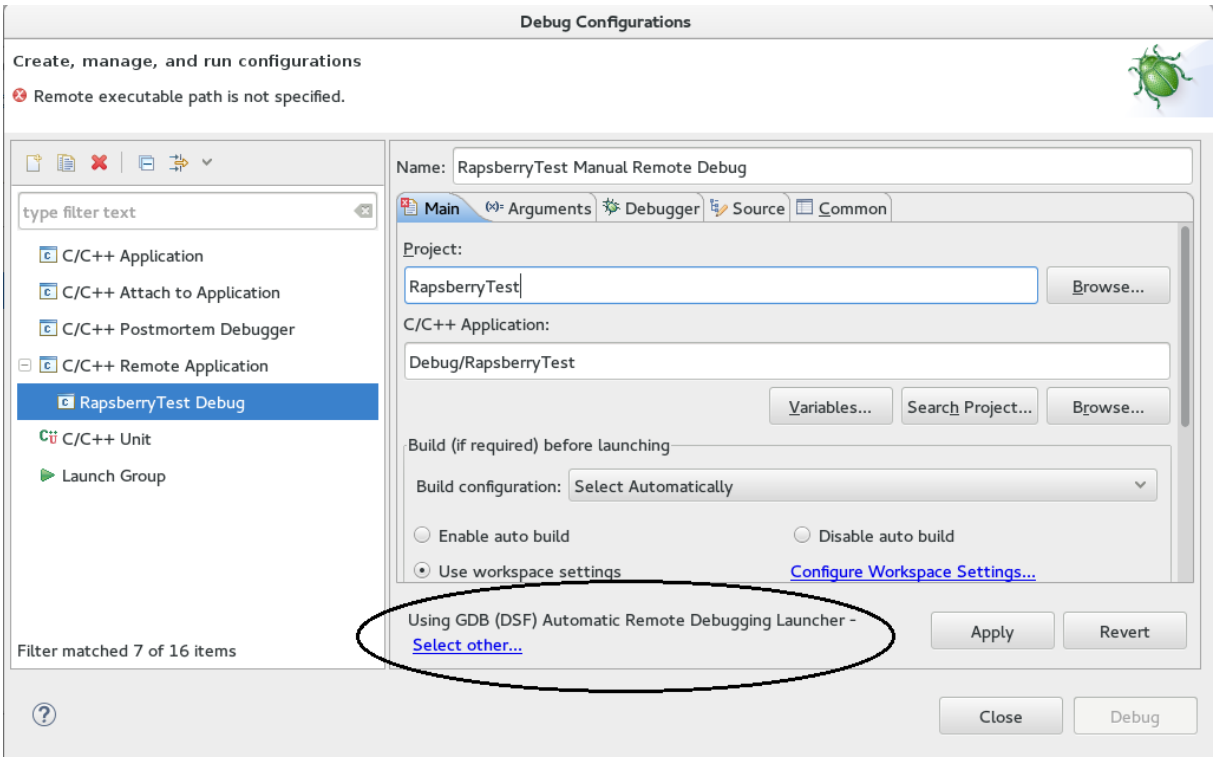
De Raspberry Pi staat klaar om te debuggen, maar nu moet de Eclipse nog geconfigureerd worden. Om een debug configuratie te maken moet het scherm Debug Configurations geopend worden. Deze kan gevonden worden onder Run->Debug Configurations.



Om een nieuwe remote configuratie te maken dubbel klikken we op C/C++ Remote Application.

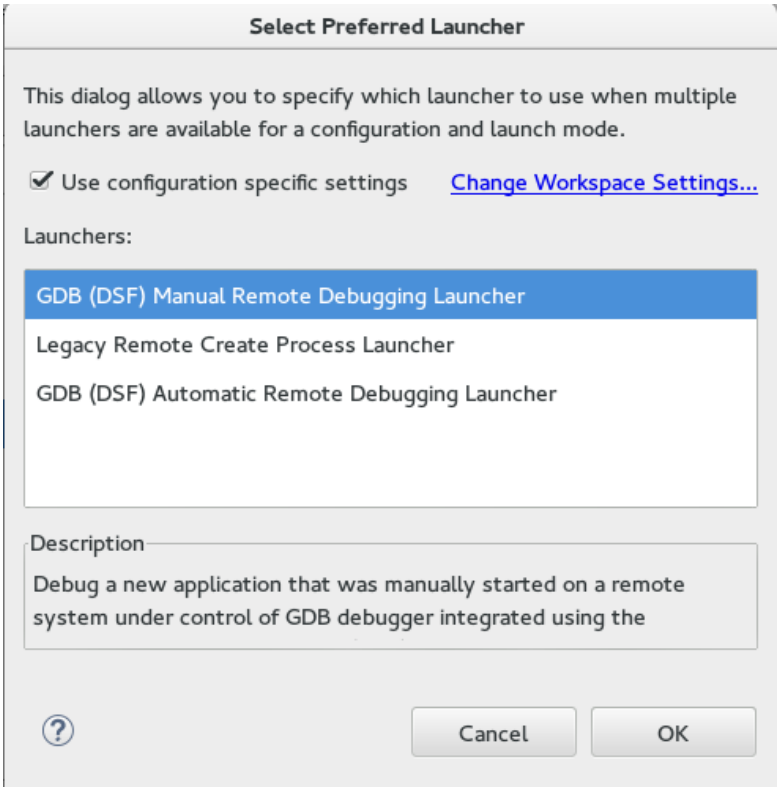


Geef de configuratie een herkenbare naam om hem later terug te vinden.

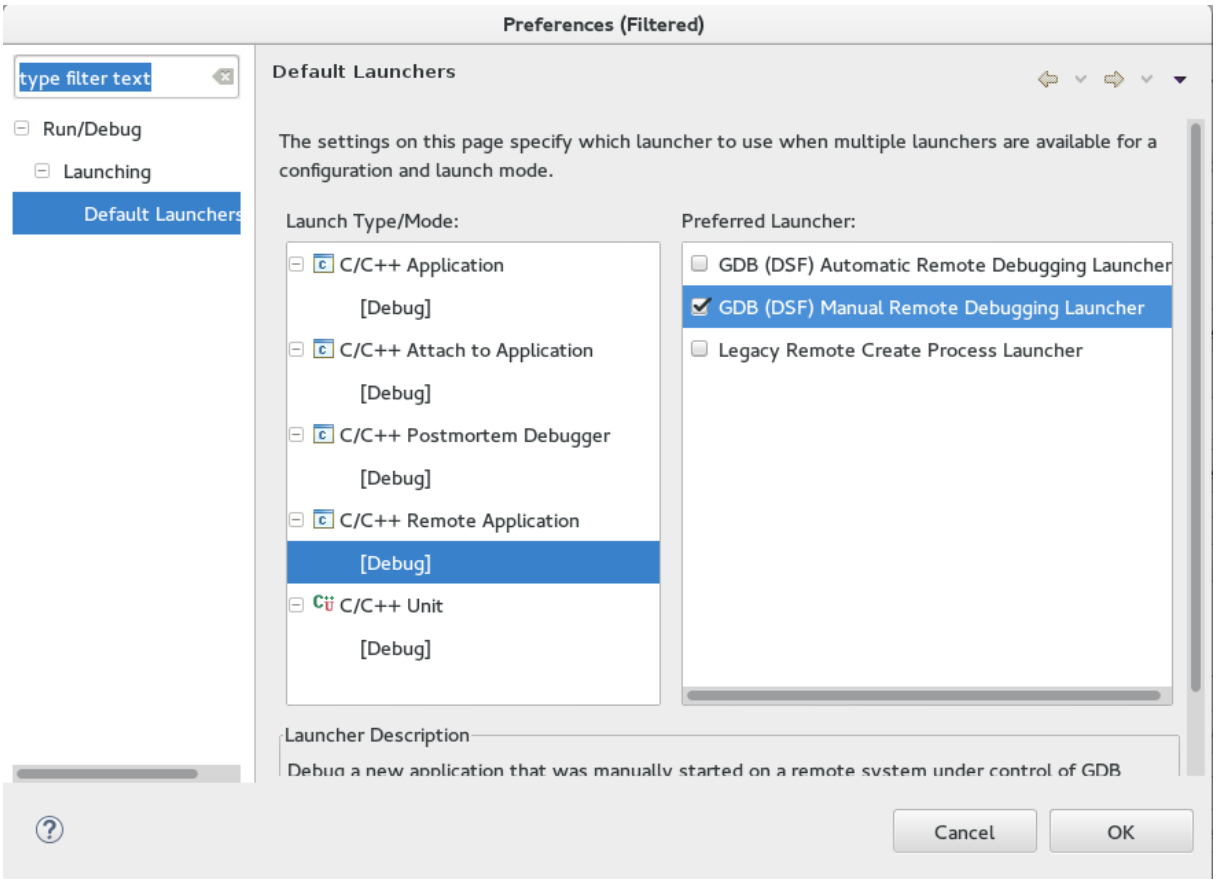


Controleer of aan de linke onderkant “Using GDB (DSF) Manual Remote Debugging Launcher” staat.

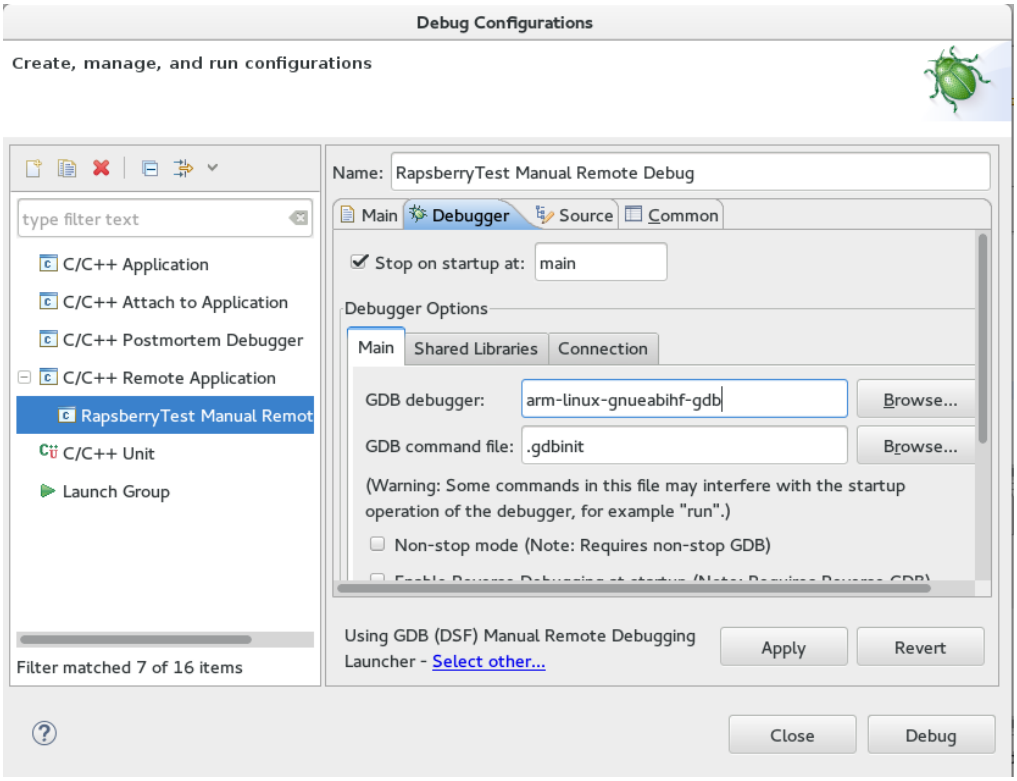
Als dat niet zo is, kan deze verandert worden door op ‘’Select Other..’’ te klikken. In dit scherm kan aangegeven worden welke debugging launcher gebruikt moet worden.



Mocht deze er niet tussen staan kan deze gevonden worden in ‘Change workspace settings..’

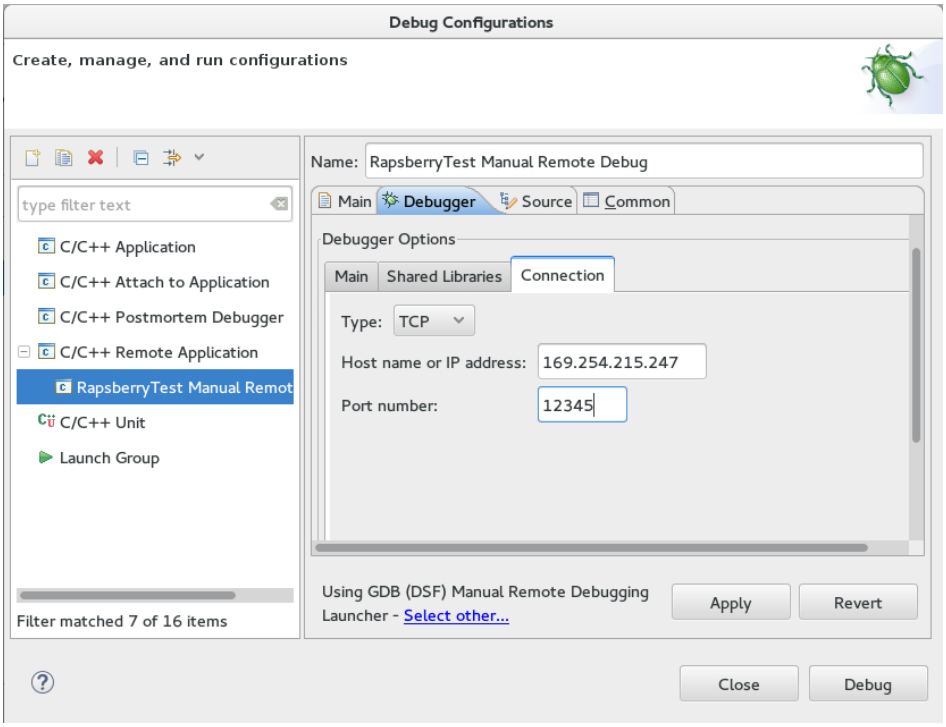


Als deze goed staat kan het tabblad “Debugger” geopend worden.

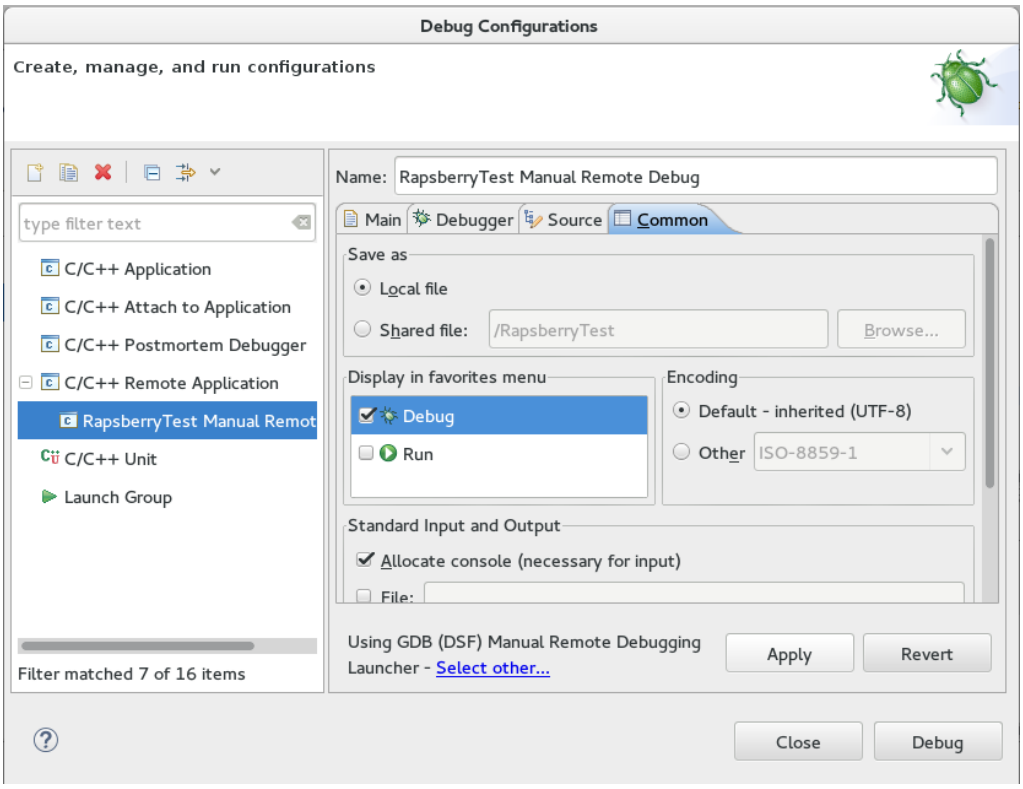


In “Debugger” zijn een paar subtabs. Aller eerst in “Main” moet de GDB debugger verandert worden naar: ‘’arm-linux-gnueabihf-gdb”.

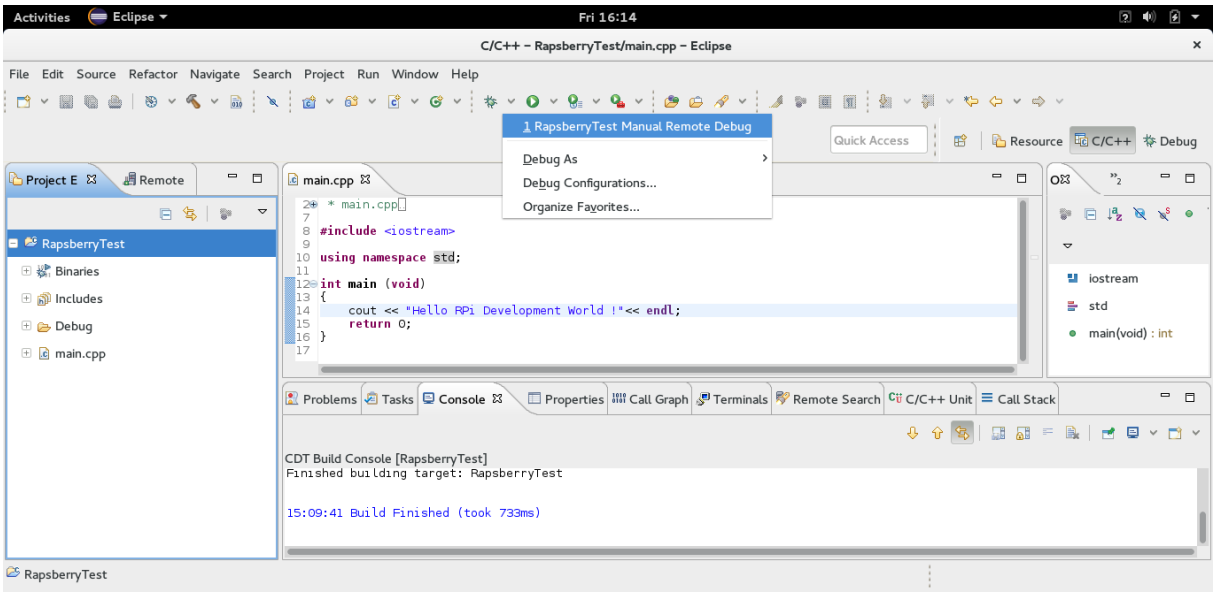
Vervolgens kan de “Connection” geopend worden. Specificeer hier het IP adress en de poortnaam van de Raspberry Pi en de poort waarop geluisterd wordt. De poort is hiervoor aangegeven op de Raspberry Pi en het IP adres is het ip adres van de Raspberry Pi.



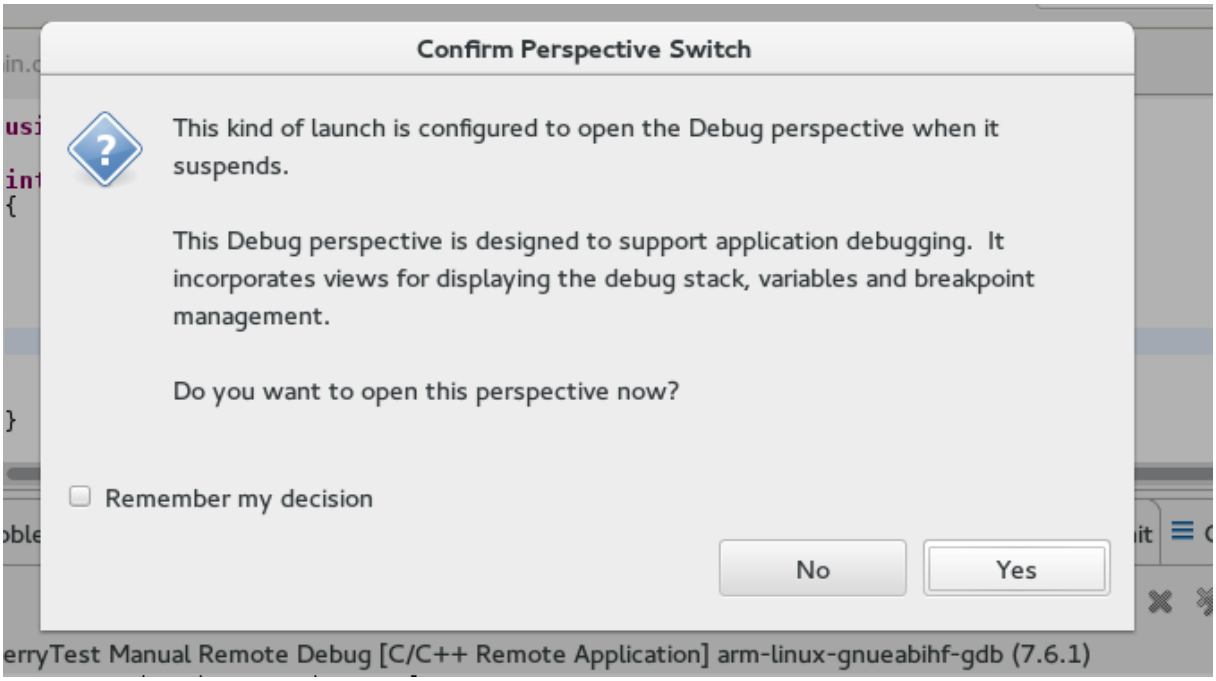
Nu kan in het laatste tab “Common” deze configuratie toegevoegd worden in de favoriete lijst.



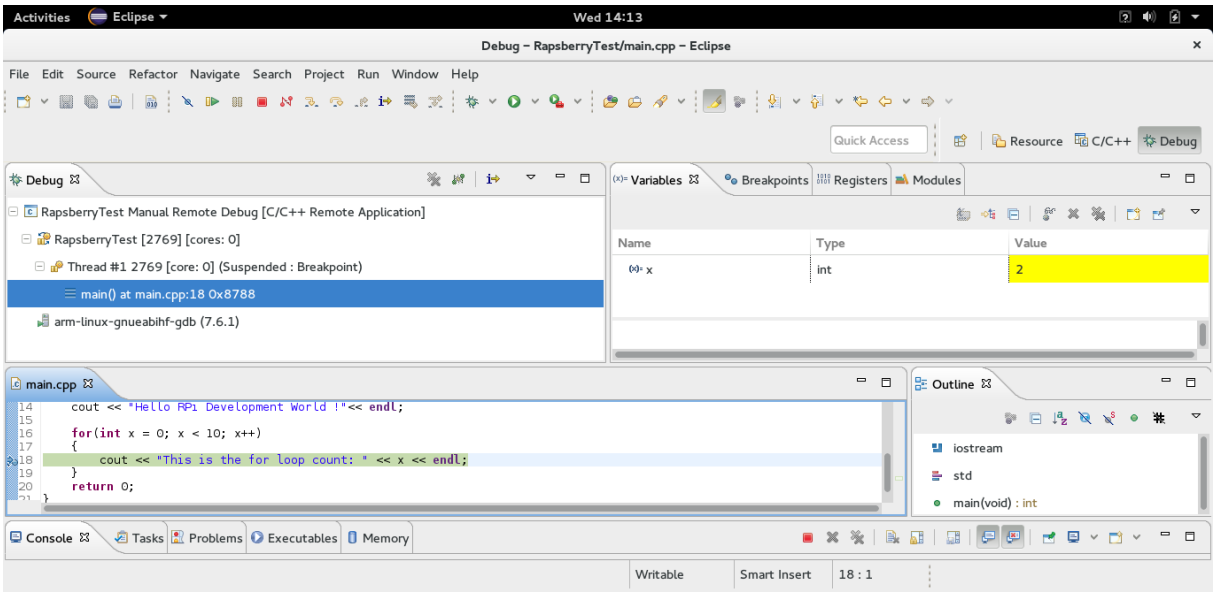
Als hij in de favoriete lijst staat kan hij via de toolbar geactiveerd worden en hoeft er niet naar de Debug Configurations scherm genavigeerd te worden om de configuratie te starten. Op dit punt kan nu de debugger gestart worden via de knop Debug of via de toolbar.



Als alles goed ingesteld is komt er een melding dat het scherm van perspectief zal veranderen en zal het programma gaan debuggen. Hier kan op ja gedrukt worden om naar de debug interface te gaan.



De debugger interface ziet er uit zoals in onderstaande afbeelding. Aan de rechter kant van het hoofdscherm zijn variabelen te vinden die in de scope van het breakpoint bekend zijn. In dit voorbeeld is alleen variabele x bekend en die is aan het optellen bij elke herhaling van de loop. Aan de linkerkant zijn alle threads te zien en waar deze gestopt zijn bij het vinden van een breakpoint. Onder aan het scherm is de code te zien om eventueel te kijken wat er rondom het breakpoint zit.



Om door alle breakpoints heen te lopen of tijdens het debuggen het programma te kunnen stoppen zijn een paar knoppen beschikbaar aan de bovenkant van het scherm.

1. Skip all breakpoints
   Hiermee worden alle toekomstige breakpoints overgeslagen.

2. Resume
   Het programma zal verder draaien alsof het niet meer in debug mode zit

3. Suspend
   Het programma zal pauzeren op de huidige positie

4. Terminate
   Het programma wordt getermineerd en de GDB server en client krijgen een kill signaal.

5. Disconnect
   De GDB server en client zullen disconnecten van elkaar.

6. Step Into
   Met Step Into zal er binnen in de functie verder gegaan worden of een volgende instructie als er niet dieper gegaan kan worden.

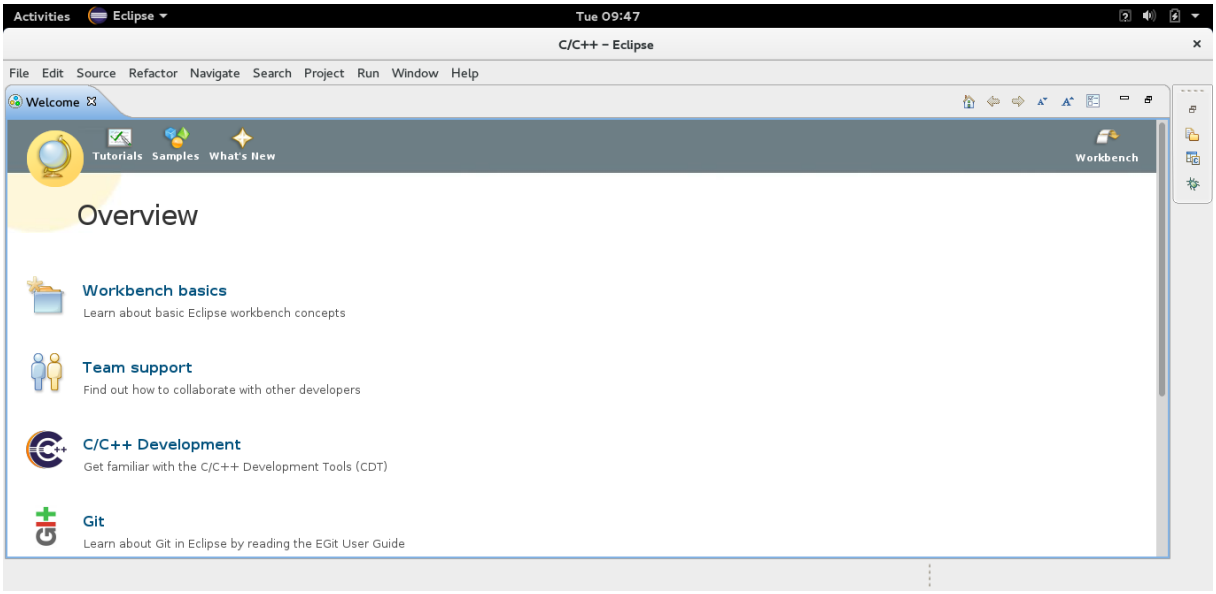
7. Step Over
   Step Over zal over functies heen stappen of door gaan naar de volgende instructie.

8. Step Return
   Met Step Return kan er weer uit functies gestapt worden naar de oproepende partij.

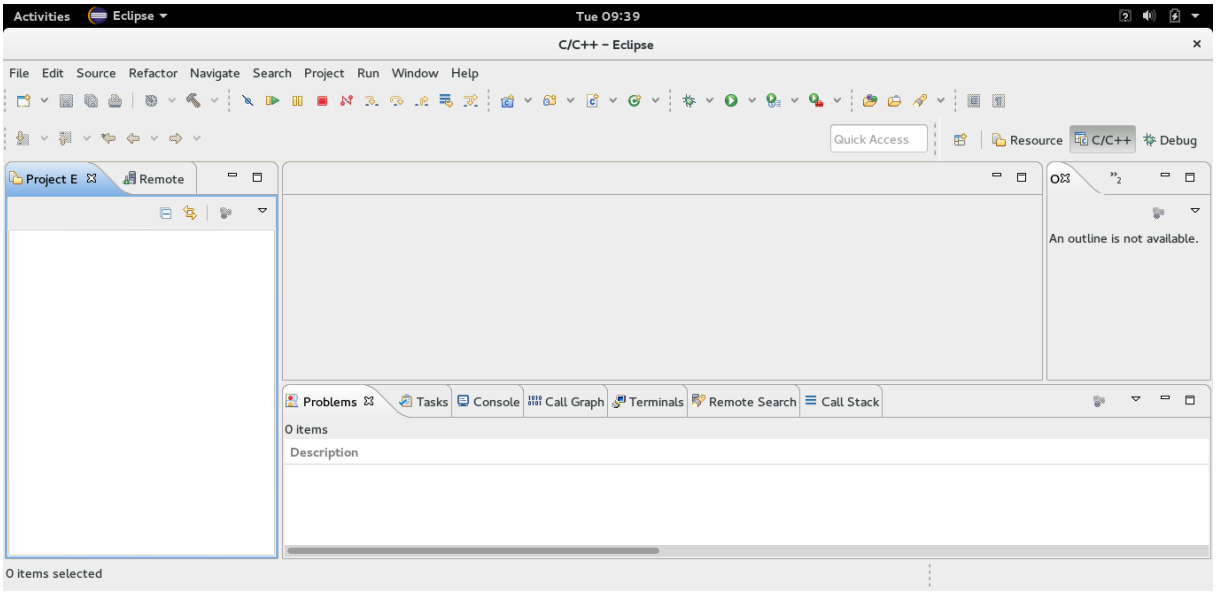
11. Handleiding Deelnemers

11.1 Ontwikkelomgeving Eclipse

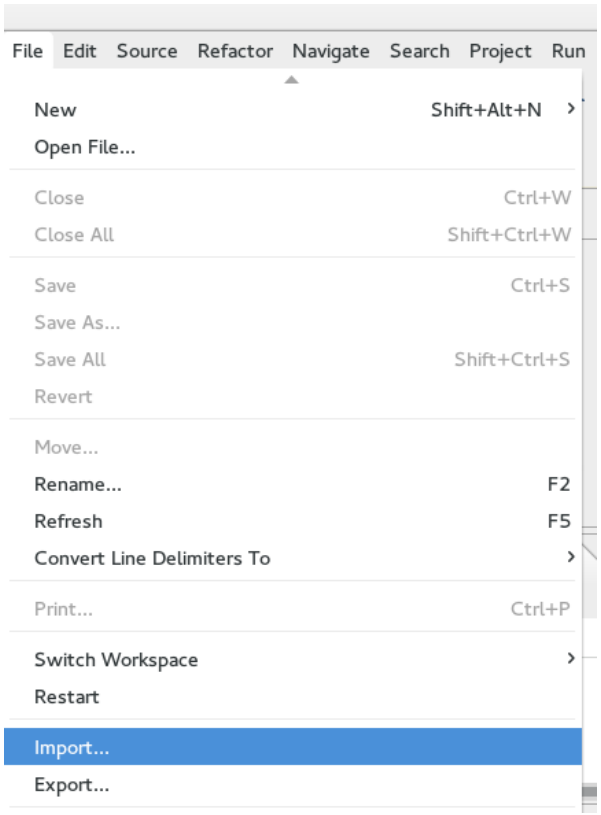
Als Eclipse opgestart wordt kan een welkomstscherm (Figuur 1) verschijnen of de ontwikkelomgeving staat al open (Figuur 2). Het welkomstscherm kan gesloten worden door op X te drukken in het tabblad aan de linkerkant.



Figuur 1: Welkomstscherm in Eclpse



Figuur 2: Lege ontwikkelomgeving in Eclipse

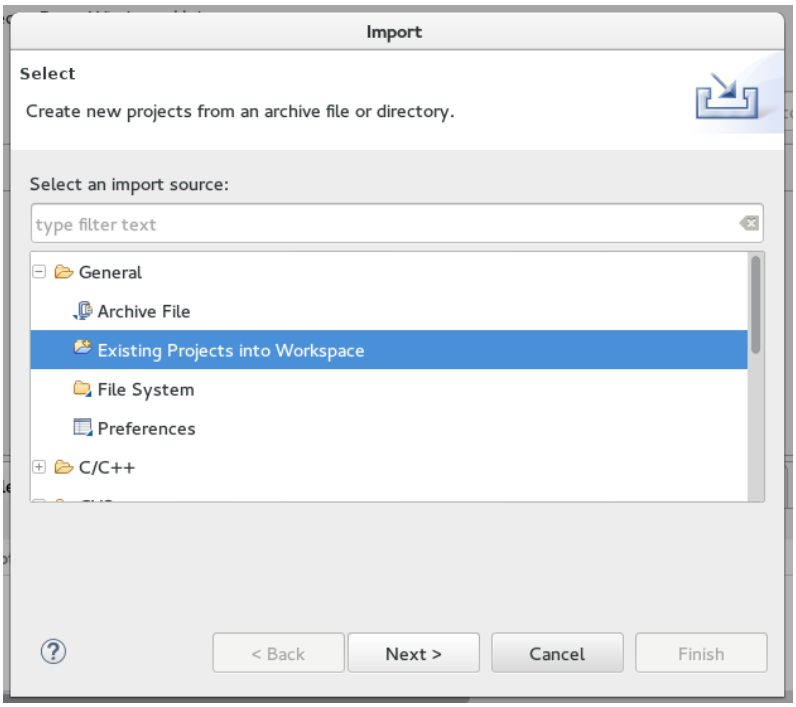


Het is mogelijk dat het project voor de workshop al open staat, mocht dit zo zijn dan mag door gegaan worden naar het volgende hoofdstuk betreft compileren, deployen en welke functies beschikbaar zijn om het geheel aan te sturen.

Als er nog geen project openstaat en de project explorer aan de linkerkant leeg staat moet het projec t geïmporteerd worden. Dit kan via File en Import… (Figuur 3)

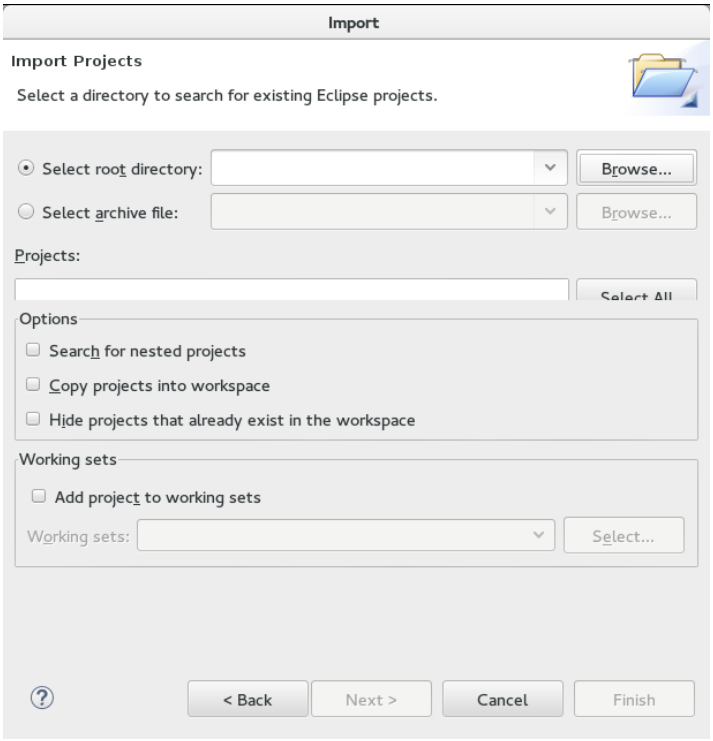
Figuur 3: Importeer optie in Eclipse

In het importeer scherm is de optie Existing Projects into Workspace voldoende, want het project bestaat al (Figuur 4)



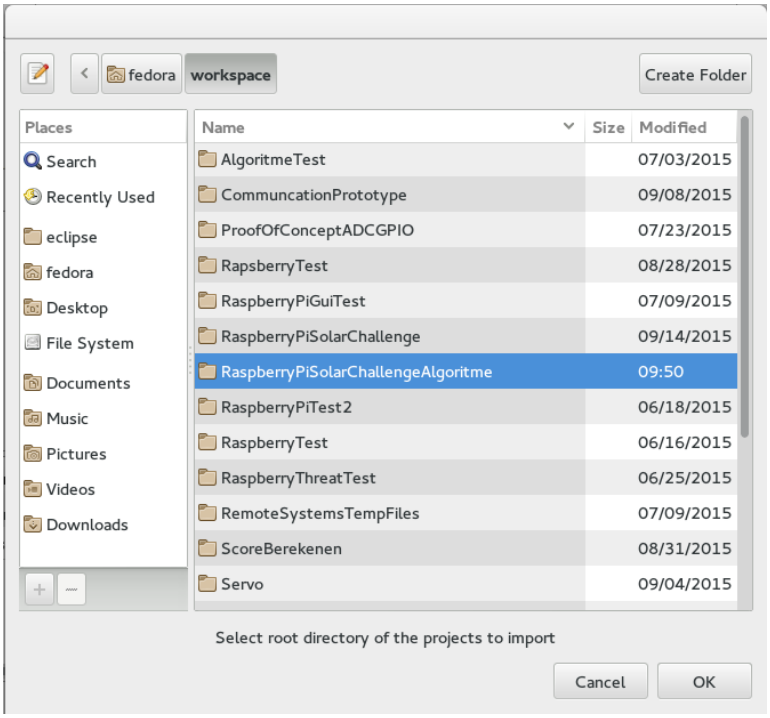
Figuur 4: Importeer selectie van bron

Vervolgens moet een project geselecteerd worden. Dit kan gedaan worden met de Browse functie die boven aan staat (Figuur 5)



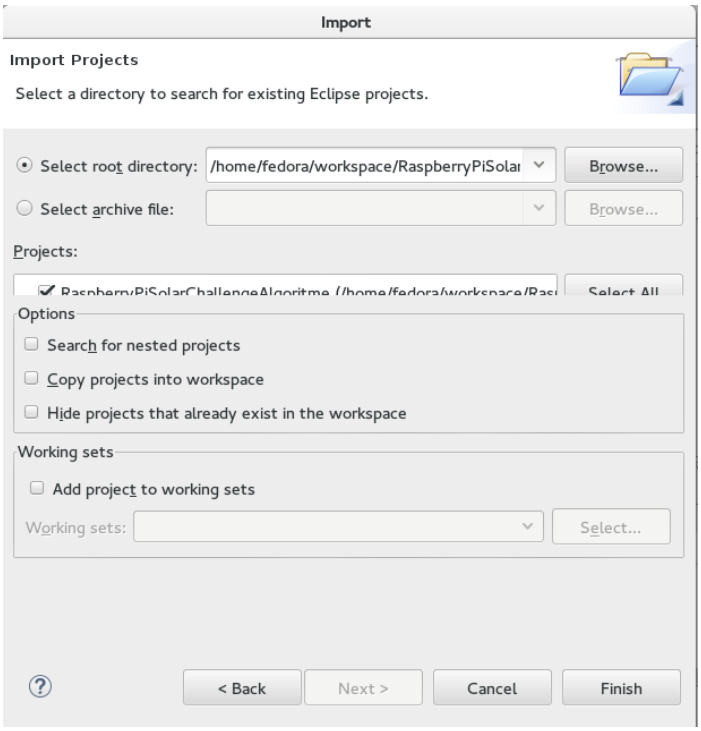
Figuur 5: Importeer scherm voor bestaande Eclipse projecten.

Vervolgens kan het project geselecteerd worden. Het project dat klaar gemaakt is voor de workshop is de RaspberryPiSolarChallengeAlgoritme project. Na het selecteren kan op ok gedrukt worden. (Figuur 6)



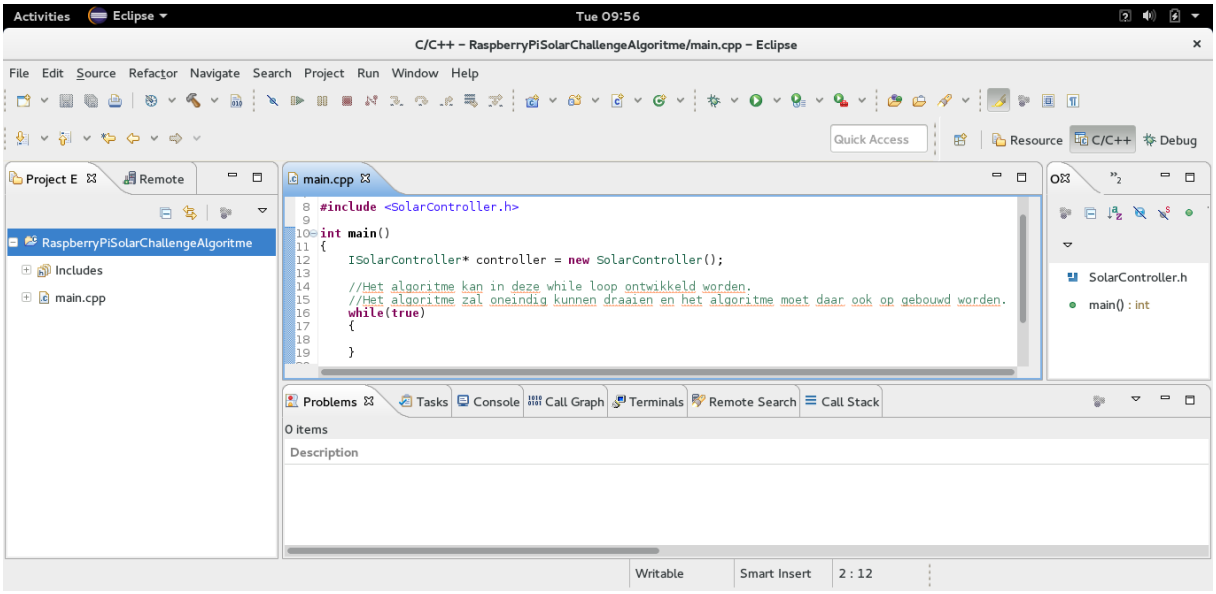
Figuur 6: Selecten van het bestaande project

Als het geselecteerd is kan op Finish gedrukt worden (Figuur 7)



Figuur 7: Project geselecteerd en importeren kan afgerond worden

Na het importeren voltooid is zal een project in de project explorer verschijnen. Deze kan geopend worden door op het + icoon te klikken dat er naast zit. Vervolgens kan de main.cpp gezien worden en deze kan geopend worden. Het hoofdbestand opend zich in het midden van het Eclipse scherm en hier zal ook ontwikkeld worden. Alle bestanden betreft het project staan aan de linkerkant van het Eclipse scherm. Aan de rechterkant staan includes en functies met betreffende return waarden, deze dienen ook als navigatie als er opgeklikt wordt. Als laatste staat de compiler informatie, console, call stack en zelfs een terminal aan de onderkant van het scherm verdeeld in tabs. Figuur 8 toont dit.



Figuur 8: Ontwikkelomgeving klaar voor gebruik

11.2 Bouwen van het algoritme

Het project is klaar gemaakt zodat gelijk begonnen kan worden aan het maken van het algoritme. Er is één verwijzing achter gelaten om gebruik te maken van voor gedefinieerde functies die de hardware kunnen aansturen. Tevens staat er een while loop om het algoritme in te maken, deze bevat een infinite loop omdat per definitie verwacht wordt dat het algoritme oneindig kan draaien.

Hieronder staan de functies die mogen gebruikt worden om het algoritme op te bouwen. Per functie staat een korte beschrijving over wat ze doen en eventuele waardes die meegegeven kunnen worden. De functies zijn gerelateerd met het opstarten, afsluiten of bewegen van servomotoren. De motoren hebben een zonnepaneel aan het uiteinde en hiervan kunnen energiewaardes opgevraagd worden.

11.2.1 Zonnepaneel:

Float RetrieveEnergy()

RetrieveEnergy retourneerd de huidige energiewinst van het zonnepaneel. Dit is een getal tussen 0 en 3 waarbij 3 het hoogste is en 0 het laagste energiewinst.

11.2.2 Motoren Bewegen:

Void PanLeft(int degrees)

Beweeg de horizontale motor naar Links met degrees aantal graden. Degrees is hierbij een getal van elke grote, maar onthoudt dat de motor maar van 0 tot 180 graden kan draaien. De motor kan niet verder dan 0 graden bij het naar links draaien.

Void PanRight(int degrees)

Beweeg de horizontale motor naar Rechts met degrees aantal graden. Degrees is hierbij een getal van elke grote, maar onthoudt dat de motor maar van 0 tot 180 graden kan draaien. De motor kan niet verder dan 180 graden bij het naar rechts draaien.

Void PanToPosition(int degrees)

Beweeg de horizontale motor naar Positie degrees. Degrees kan hierbij een getal zijn van elke grote, maar de motor zal alleen bewegen naar een positie tussen 0 en 180 graden. Getallen kleiner dan 0 worden vertaald naar de 0 graden positie en getallen groter dan 180 worden vertaald naar 180.

Void TiltUp(int degrees)

Beweeg de verticale motor naar Boven met degrees aantal graden. degrees is hierbij een getal van elke grote, maar onthoudt dat de motor maar van 0 tot 180 graden kan draaien. De motor kan niet verder dan 0 graden bij het naar omhoog draaien.

Void TiltDown(int degrees)

Beweeg de verticale motor naar Beneden met x aantal graden. x is hierbij een getal van elke grote, maar onthoudt dat de motor maar van 0 tot 180 graden kan draaien. De motor kan niet verder dan 180 graden bij het naar omlaag draaien.

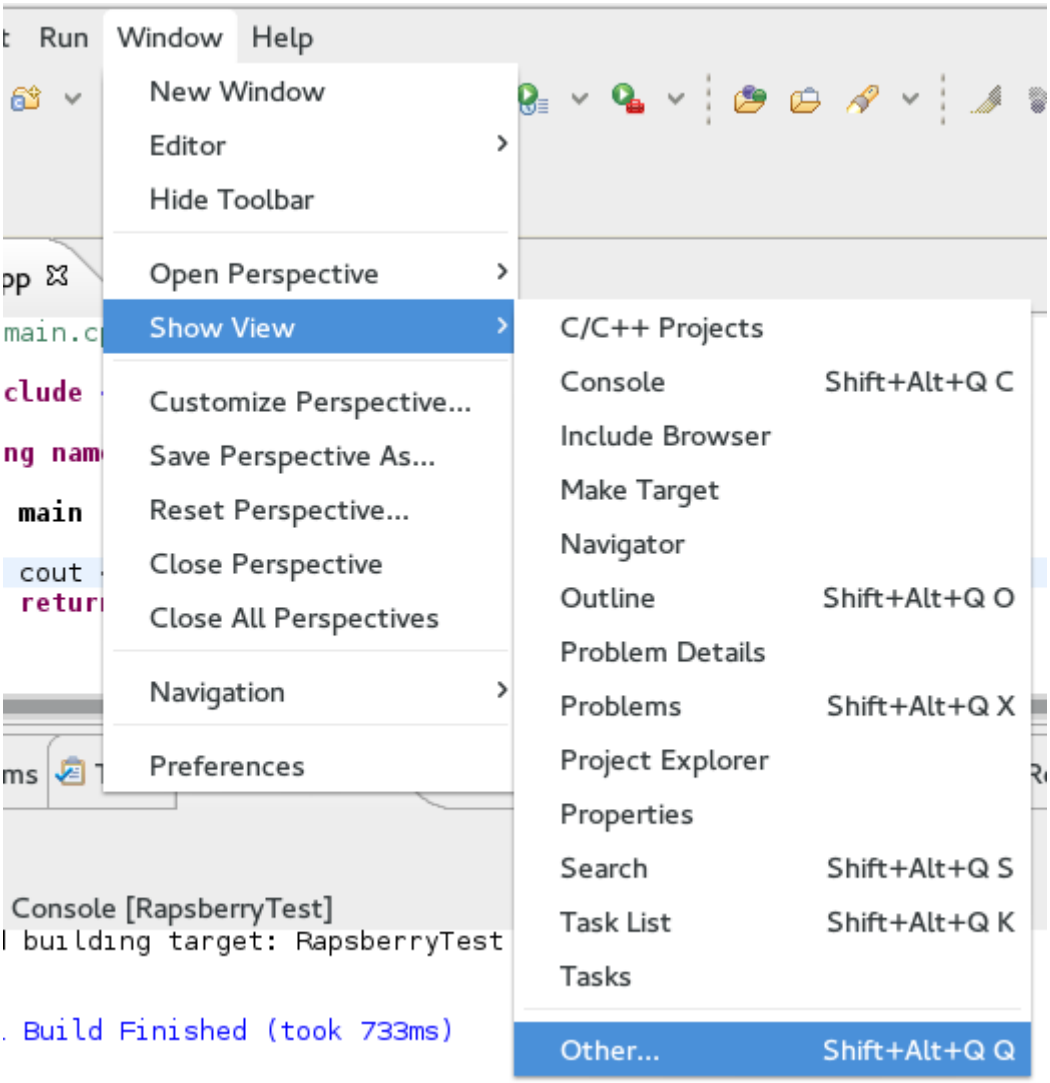
Void TiltToPosition(int degrees)

Beweeg de Verticale motor naar Positie X. X kan hierbij een getal zijn van elke grote, maar de motor zal alleen bewegen naar een positie tussen 0 en 180 graden. Getallen kleiner dan 0 worden vertaald naar de 0 graden positie en getallen groter dan 180 worden vertaald naar 180.

11.3 Deployen

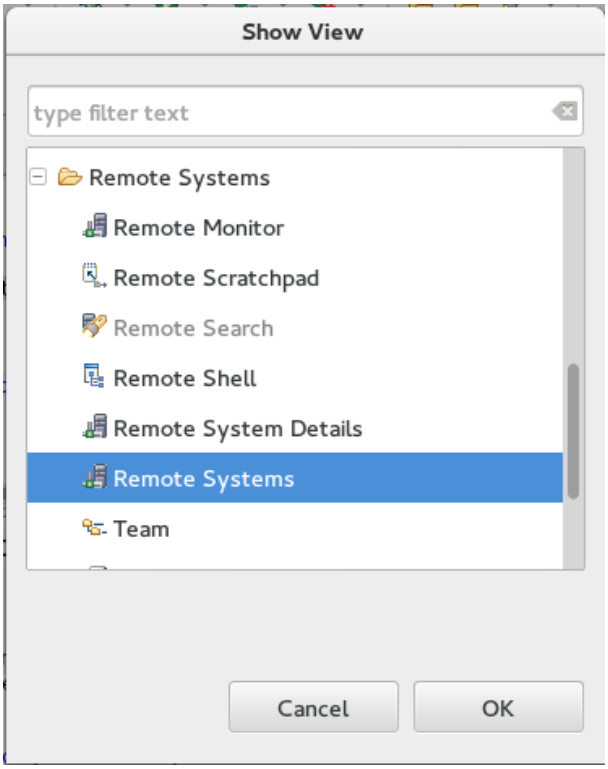
De ontwikkelomgeving staat niet op de Raspberry Pi en de code wordt dus remote gebouwd voor de Raspberry Pi. Deze moet overgezet worden voordat het zijn taak pas kan uitvoeren want deze code werkt NIET op de computer of virtuele machine ook al is het hetzelfde operating systeem. Dat komt omdat het project dat opgesteld is compileert voor een Linux arm systeem waarop de Raspberry Pi draait. Om het algoritme en de code op de Raspberry Pi te krijgen om alles aan te sturen moet deze overgezet worden. Dit kan via remote acces en nog makkelijker via Eclipse zelf.

Hiervoor wordt remote systems gebruikt binnen in Eclipse. Dit tab kan al openstaan aan de linker of onderkant en als dit zo is mag door gegaan worden naar het openen van de connectie. Het Remote Systems tab kan gevonden worden in de lijst van views en die kan gevonden worden in Window -> Show View -> Other… (Figuur 9)



Figuur 9: Openen van de lijst van alle views in Eclipse

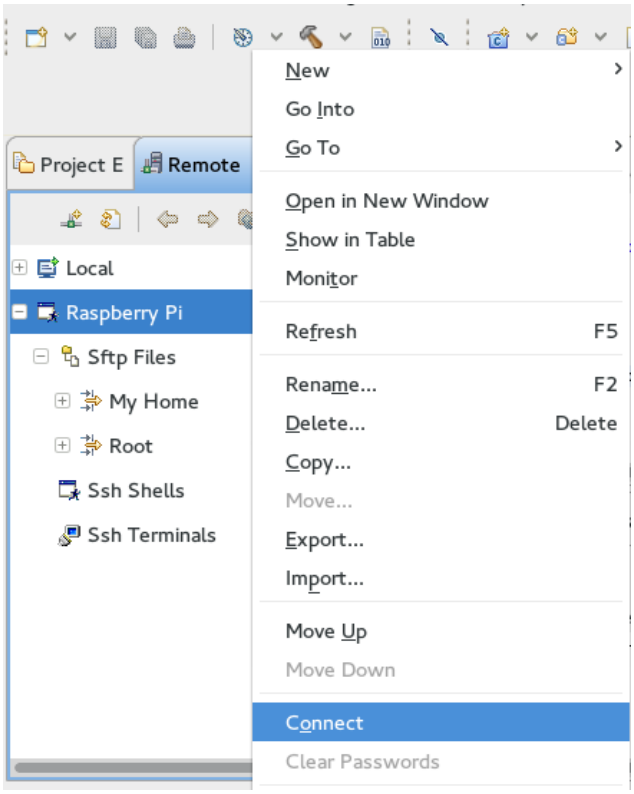
In de lijst van views zit een folder met Remote System met daarin Remote Systems. Selecteer deze en druk op ‘Ok’ (Figuur 10)



Figuur 10: Remote Systems in de lijst van views.

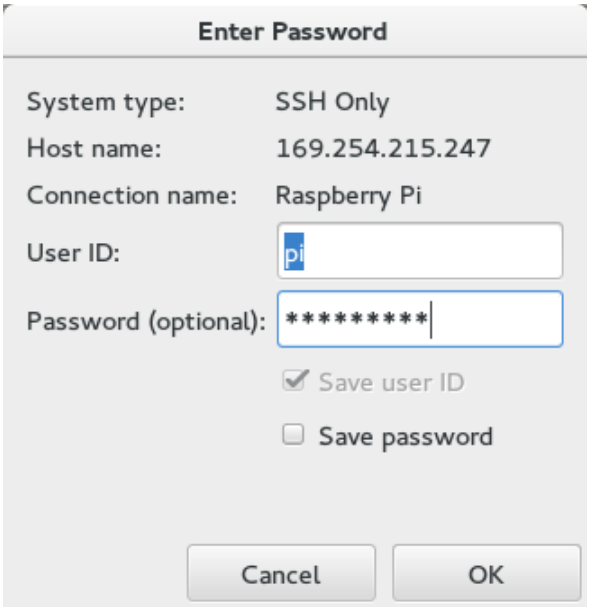
11.3.1 Connectie maken met de Raspberry Pi

Met het Remote Systems scherm open is te zien dat er twee connecties staan. De Local connectie is de lokale omgeving en Raspberry Pi is vanzelfsprekend de Raspberry Pi. Om de connectie te maken kan met de rechtermuisknop en Connect de connectie gemaakt worden (Figuur 11)



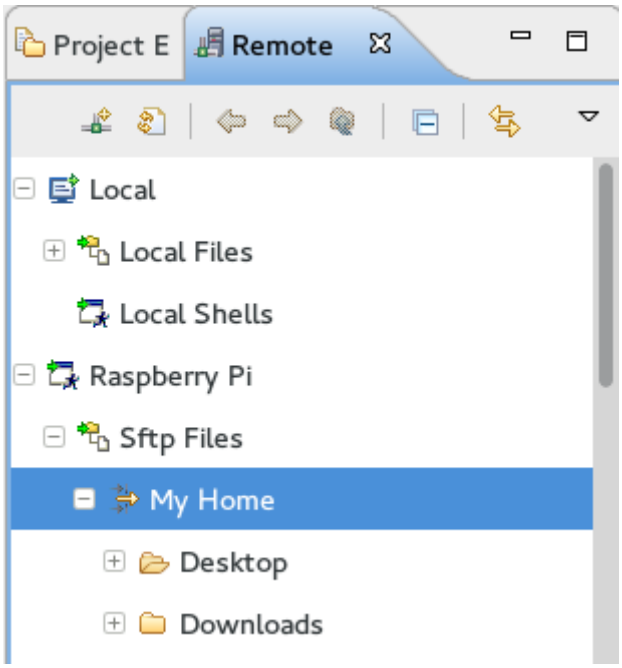
Figuur 11: Connect met de Raspberry Pi

Het is mogelijk dat er om een userId en wachtwoord gevraagd worden, standaard is dit ‘pi’ en ‘raspberry’ (Figuur 12).



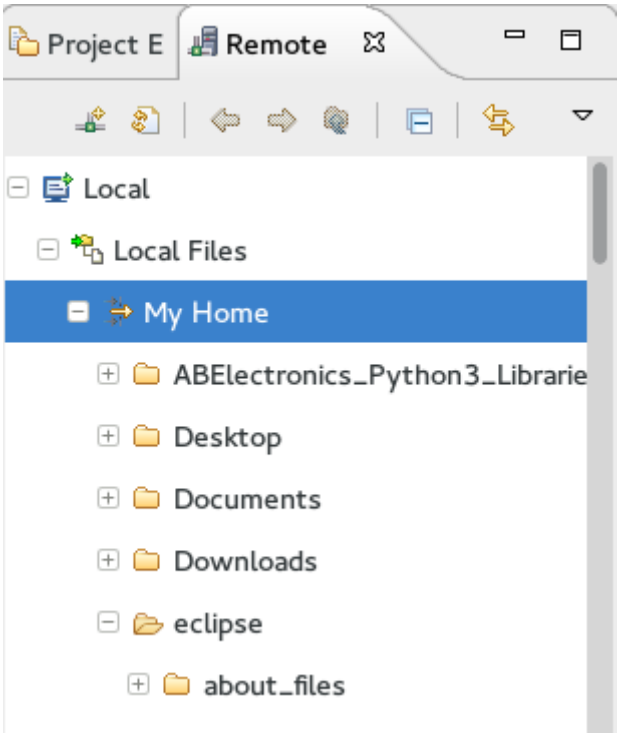
Figuur 12: Inlogscherm voor Remote Acces

In het remote systems scherm kan genavigeerd worden naar de folders op de connecties de openstaan. Om het project te deployen heeft het een plek nodig en als voorbeeld wordt hier de Desktop voor gebruikt (Figuur 13).



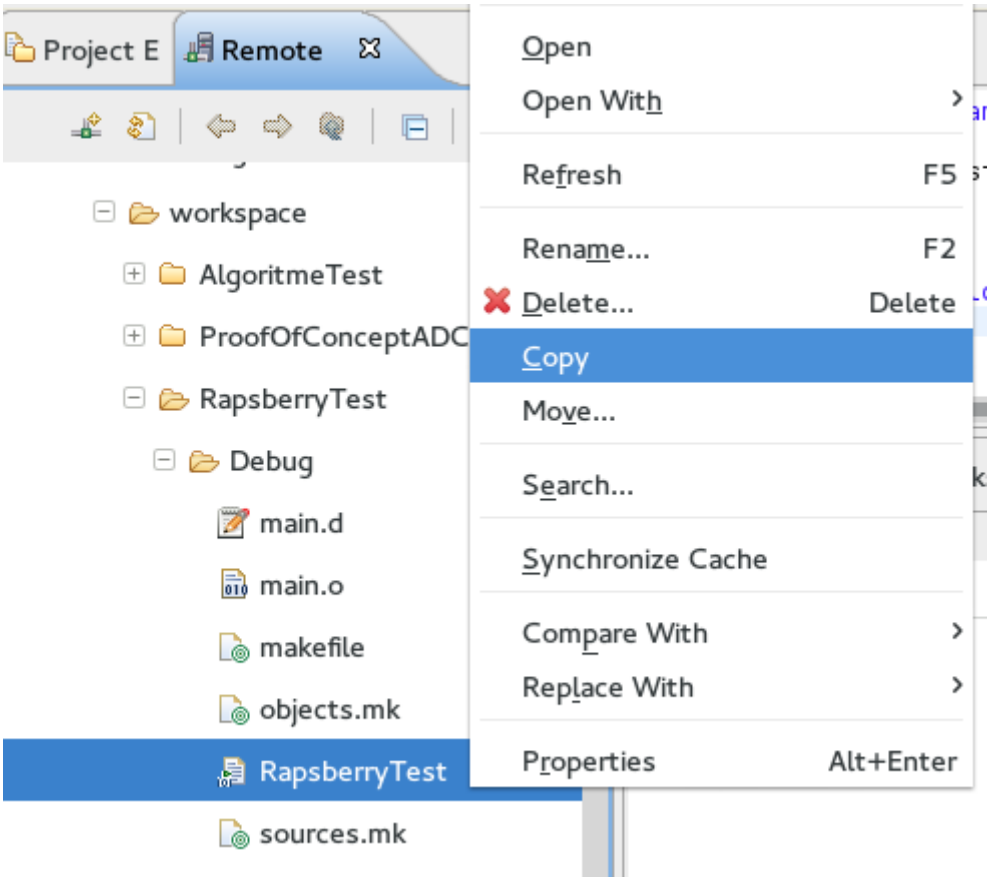
Figuur 13: Home folder Raspberry Pi

Het project moet gevonden worden op de lokale machine. Deze staat zeer waarschijnlijk in de workspace folder van Eclipse. Deze staat hier in de home folder (Figuur 14).



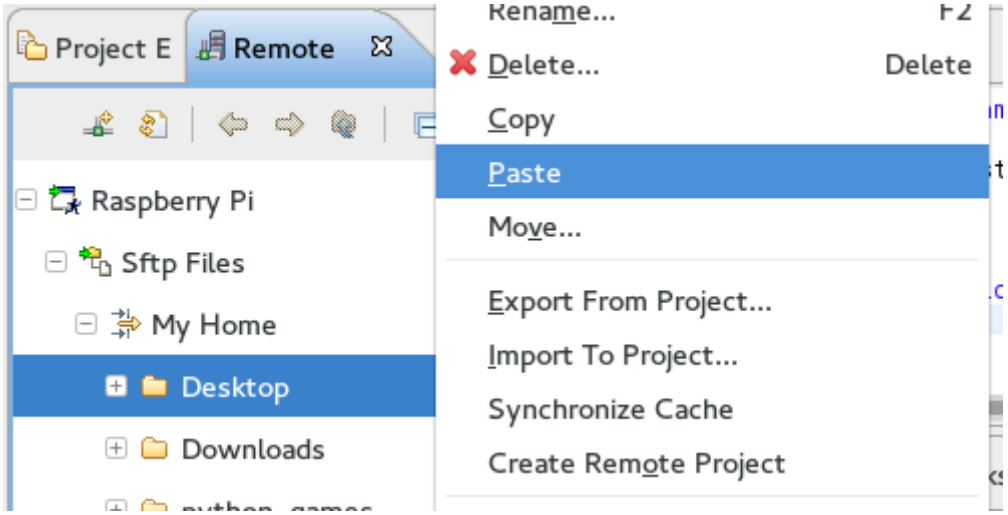
Figuur 14: Home folder Lokale machine

In de workspace moet de project folder gevonden en in de Debug folder staat een bestand met dezelfde naam als het project en een ontbrekende extensie (Figuur 15).



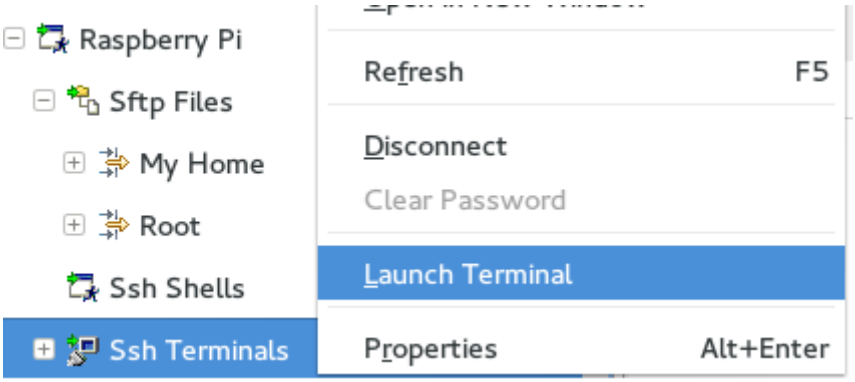
Figuur 15: Kopiëren van de executabel van het project.

Deze kan gekopieerd worden naar de locatie op de Raspberry Pi. Hier is dat de Desktop. (Figuur 16)



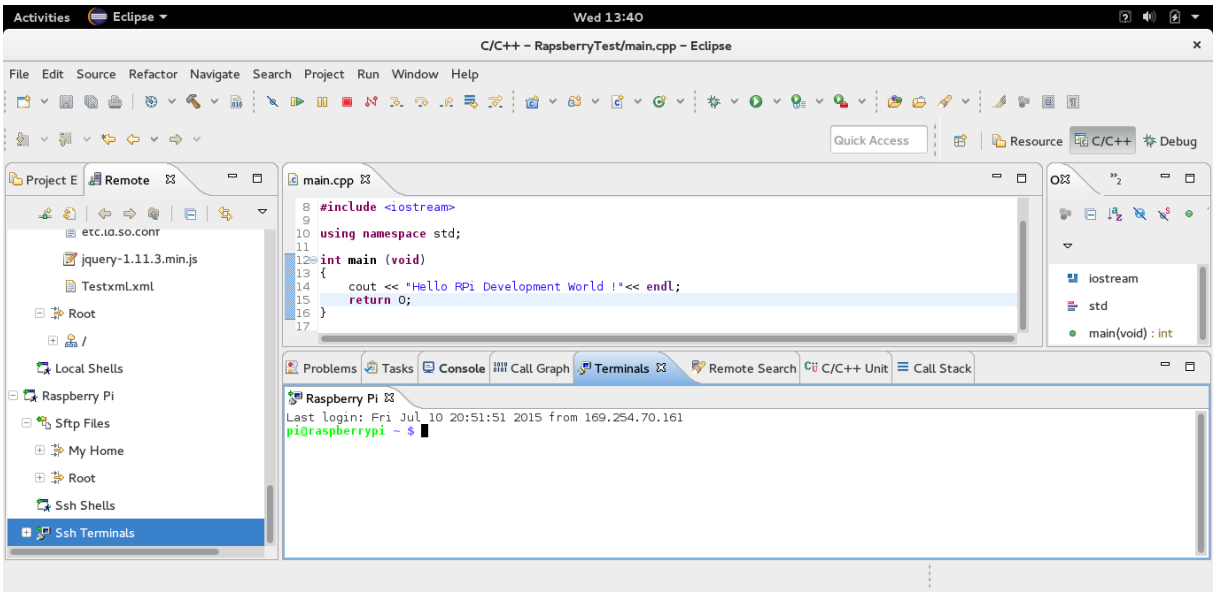
Figuur 16: Plakken naar de doelfolder op de Raspberry Pi

Het bestand is over gezet. Nu hoeft alleen een terminal opengemaakt te worden om het bestand te executeren. Hiervoor wordt een remote terminal gebruikt eveneens in remote systems. Rechtermuisklik op SSH terminals en launch terminal (Figuur 17).



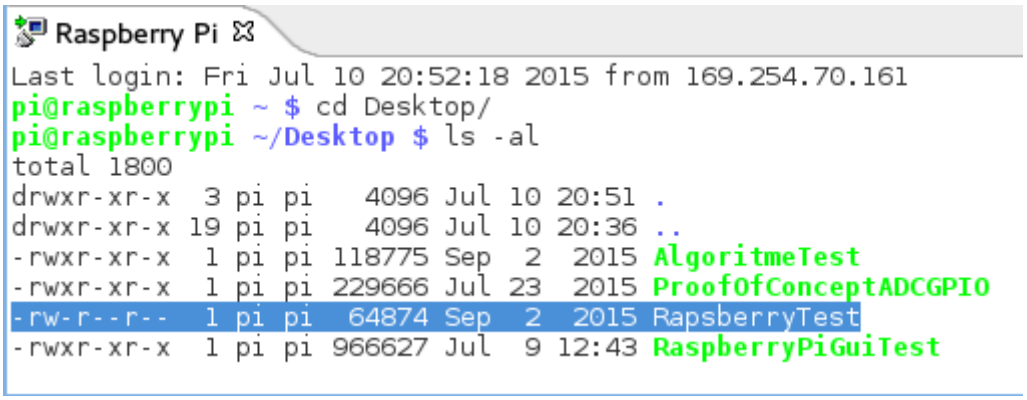
Figuur 17: Openenn van een remote terminal window in Eclipse.

De terminal opent in een tabblad en deze staat standaard aan de onderkant van het Eclipse scherm (Figuur 18).



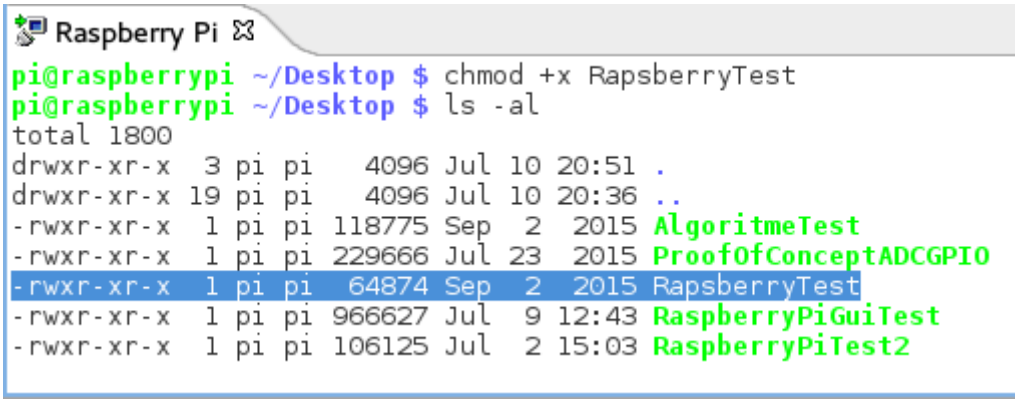
Figuur 18: Terminal Tab open aan de onderkant

Na het overzetten van het bestand worden standaard rechten verleend aan het bestand van andere gebruikers. Dit zijn voornamelijk leesrechten en root heeft ook schrijf rechten, maar geen van deze zorgt ervoor dat het bestand uitgevoerd kan worden(Figuur 19).



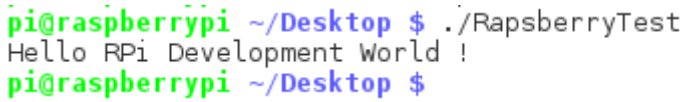
Figuur 19: Rechten bestand na het overzetten van de executable

Om het bestand executable te maken kan chmod +x gebruikt worden op het bestand om executable rechten toe te wijzen(Figuur 20).



Figuur 20: Rechten bestand na chmod

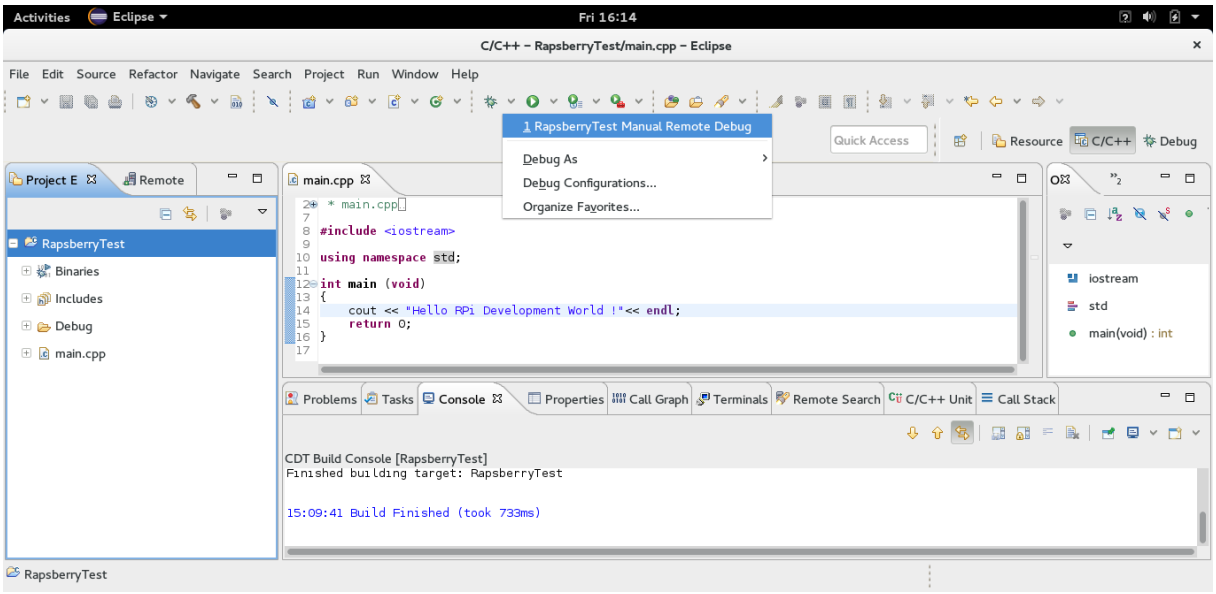
Daarna kan het bestand uitgevoerd worden zoals elk andere executable ./File (Figuur 21)



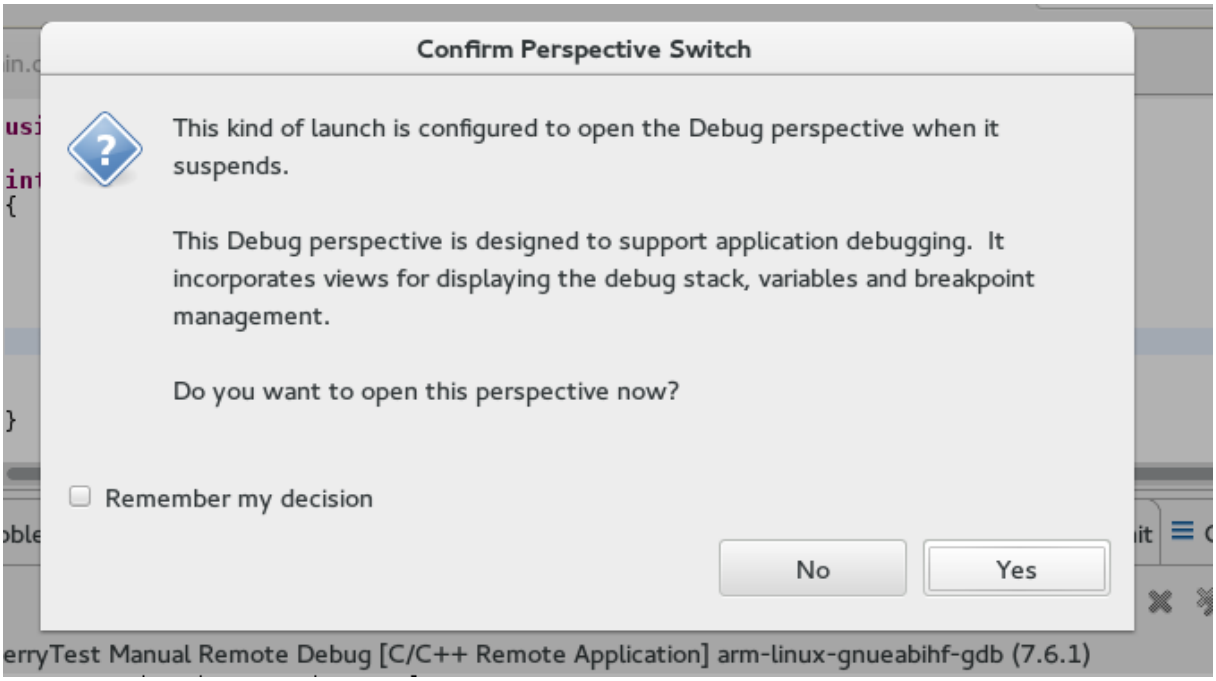
Figuur 21: Uitvoering van het executable bestand.

11.4 Debuggen

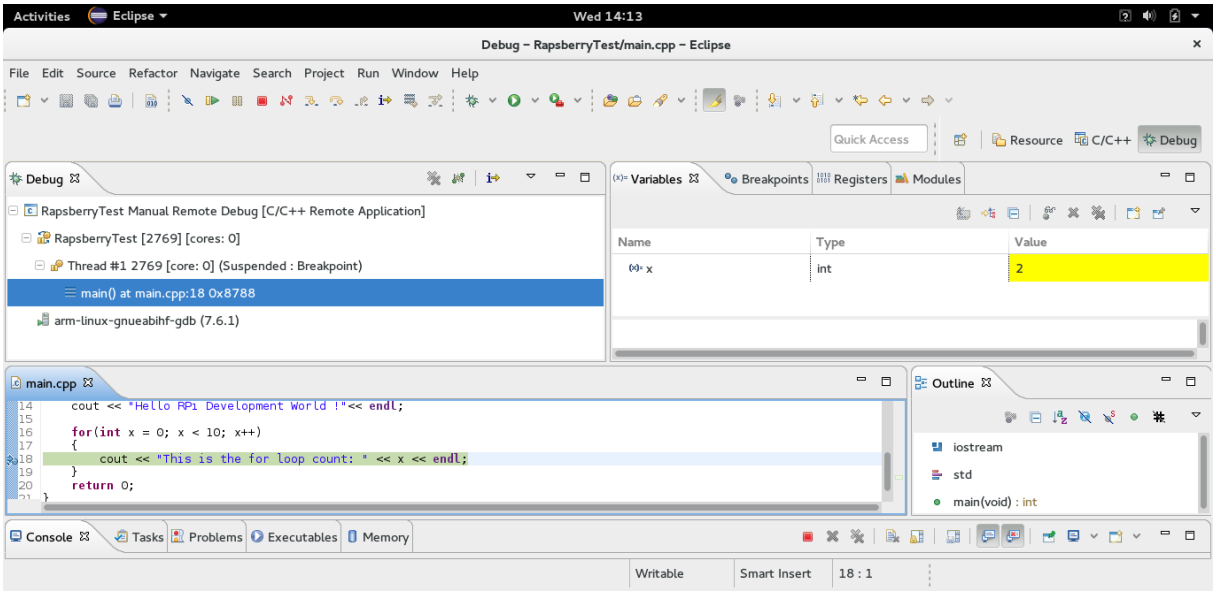
Om te debuggen in Eclipse maken we gebruik van een voor ingestelde debug configuratie. Deze is in het favoriete menu te zien onder debuggen (De insect knop). In het voorbeeld heet deze Raspberry Test Manual Remote Debug.



Mocht onderstaande popup tevoorschijn komen, mag het vinkje aangezet worden en op ja gedrukt worden. Hierna komt het debug scherm in Eclipse open te staan.



De debugger interface ziet er uit zoals in onderstaande afbeelding. Aan de rechter kant van het hoofdscherm zijn variabelen te vinden die in de scope van het breakpoint bekend zijn. In dit voorbeeld is alleen variabele x bekend en die is aan het optellen bij elke herhaling van de loop. Aan de linkerkant zijn alle threads te zien en waar deze gestopt zijn bij het vinden van een breakpoint. Onder aan het scherm is de code te zien om eventueel te kijken wat er rondom het breakpoint zit.



Om door alle breakpoints heen te lopen of tijdens het debuggen het programma te kunnen stoppen zijn een paar knoppen beschikbaar aan de bovenkant van het scherm.

1. Skip all breakpoints
   Hiermee worden alle toekomstige breakpoints overgeslagen.
2. Resume
   Het programma zal verder draaien alsof het niet meer in debug mode zit
3. Suspend
   Het programma zal pauzeren op de huidige positie
4. Terminate
   Het programma wordt getermineerd en de GDB server en client krijgen een kill signaal.
5. Disconnect
   De GDB server en client zullen disconnecten van elkaar.
6. Step Into
   Met Step Into zal er binnen in de functie verder gegaan worden of een volgende instructie als er niet dieper gegaan kan worden.
7. Step Over
   Step Over zal over functies heen stappen of door gaan naar de volgende instructie.
8. Step Return
   Met Step Return kan er weer uit functies gestapt worden naar de oproepende partij.