Camera-stabilisatie-systeem
Document met bijlagen

Document met bijlagen behorende bij het afstudeerverslag

Jeroen de Meij

Studentnummer: 11100877

Afstudeerverslag

De Haagse Hogeschool, opleiding Technische Informatica, afdeling Delft

Delft Dynamics B.V.

Bedrijfsmentor: Dhr. G. M. Voorsluijs

Examinatoren: Dhr. Tony Andrioli en mw. Cobie van der Hoek

Datum: 2 juni 2014

Bijlage A: Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2014-1.1 (start uiterlijk 10 februari 2014)
Startdatum uitvoering afstudeeropdracht: 10 februari 2013
Inleverdatum afstudeerdossier volgens jaarrooster: 6 juni 2014

Studentnummer: 11100877
Achternaam: dhr. De Meij
Voorletters: J.
Roepnaam: Jeroen
Adres: Beverlanderhof 9
Postcode: 2461 BV
Woonplaats: Ter Aar
Telefoonnummer: 0172-605821
Mobiel nummer: 06-27047341
Privé emailadres: jeroendemeij@gmail.com

Opleiding: Technische Informatica
Locatie: Delft
Variant: voltijd

Naam studieloopbaanbegeleider: Nu: Tony Andrioli – Bij aanvang stage: Pieter Burghout
Naam begeleidend examinator: Tony Andrioli
Naam tweede examinator: Cobie van der Hoek

Naam bedrijf: Delft Dynamics
Afdeling bedrijf: -
Bezoekadres bedrijf: Molengraaffsingel 12-14
Postcode bezoekadres: 2629 JD
Postbusnummer: -
Postcode postbusnummer: -
Plaats: Delft
Telefoon bedrijf: (+31) 15-7111009
Telefax bedrijf: -
Internetsite bedrijf: www.delftdynamics.nl/

Achternaam opdrachtgever: dhr. Voorsluijs
Voorletters opdrachtgever: G. M.
Titulatuur opdrachtgever: ir.
Functie opdrachtgever: -
Doorkiesnummer opdrachtgever: -
Email opdrachtgever: g.m.voorsluijs@delftdynamics.nl

Achternaam bedrijfsmentor: dhr. Voorsluijs
Voorletters bedrijfsmentor: G. M.
Titulatuur bedrijfsmentor: ir.
Functie bedrijfsmentor: -
Doorkiesnummer bedrijfsmentor: -
Email bedrijfsmentor: g.m.voorsluijs@delftdynamics.nl

Doorkiesnummer afstudeerder: -
Functie afstudeerder (deeltijd/duaal): -

Titel afstudeeropdracht:

Ontwikkelen van een camera-stabilisatie-systeem bij Delft Dynamics

Opdrachtomschrijving

- **Bedrijf**

  Delft Dynamics B.V. is in februari 2006 opgericht en heeft zich gespecialiseerd in het bouwen van robothelikopters: kleine onbemande helikopters die door het slim samenvoegen van computer- en sensorapparatuur gebruikt kunnen worden als stabiel en makkelijk te besturen sensorplatform. Een aantal robothelikopters is geleverd aan KLPD en Defensie voor operationele evaluatie.
  In 2013 brengt Delft Dynamics haar nieuwste product op de markt: de RH4 'Spyder'. Deze compacte en lichtgewicht multicopter kan verschillende soorten camera's dragen en is zeer makkelijk te bedienen. Op deze wijze is het mogelijk om op een eenvoudige manier vanuit de lucht foto- en videobeelden te maken van de omgeving.

Afbeelding 1: Delft Dynamics RH4 Spyder

- **Probleemstelling**

  De huidige avionica van de RH4 Spyder stabiliseert de multicopter en verzorgt ook de compensatie voor de bewegingen van de multicopter voor de camera. Door de ontwikkelingen in embedded processing en miniatuur sensoren wordt het nu rendabel om een extra (ARM)-processor met eigen sensoren toe te voegen in de camera zelf zodat een snellere camera-stabilisatie gerealiseerd wordt, zodat de camerabeelden beter worden. Hierna is het mogelijk om eventueel extra functionaliteit aan de ARM-processor toe te kennen. Dit is in de huidige situatie niet mogelijk, omdat de huidige avionica aan het maximum van zijn kunnen zit.

- **Doelstelling van de afstudeeropdracht**

  Op basis van beschikbare hardware (camera, gimbal met motortjes, ARM-processor, sensoren) zal een werkend camera-stabilisatie-systeem gemaakt worden. Aan het einde van de opdracht moet het huidige camera-stabilisatie-systeem zijn overgezet en geoptimaliseerd voor de ARM processor. Om dit te bereiken zullen de sensoren op hoge frequentie (1 kHz) uitgelezen worden en samen met de aansturing vanuit de avionica gebruikt worden om de motoren aan te sturen zodat de camera de gewenste stand aanneemt.

  Voor het uitvoeren van deze opdracht zal gebruik worden gemaakt van de GCC compiler voor ARM processoren en SPI, PWM en RS232 communicatie-protocollen. De werking van het stabilisatie-systeem zal zowel stand-alone als vliegend getest worden en beoordeeld op snelheid en nauwkeurigheid.

- **Resultaat**

  Het uiteindelijke resultaat is een geoptimaliseerde versie van het huidige camera-stabilisatie-systeem voor op de ARM processor.

Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

- 2 dagen – Opstellen definitie lijst met begrippen die tijdens de stage naar voren kunnen komen.
- 2 dagen – Plan van aanpak schrijven
- 3 dagen – Oriëntatie op het systeem zoals het nu draait.
- 3 dagen – Vaststellen van de requirements van het systeem zoals het op de ARM processor moet draaien.
- 3 dagen – Onderzoeken en installeren/configureren van de GCC compiler voor ARM processen
- 20 dagen – Onderzoeken van de geschikte optimalisatie van het systeem, een ontwerp voor de nieuwe situatie en het overzetten ervan.
- 5 dagen – Bevindingen documenteren voor bedrijf
- 7 dagen – Testen van het systeem
- 10 dagen – Implementeren van het systeem bij een helikopter en overdracht
- 15 dagen – Opbouwen afstudeer dossier

- **Op te leveren (tussen)producten**

  - Definitie lijst
  - Plan van aanpak
  - Oriëntatie document
  - Requirements analyse
  - Beschrijving installatie GCC compiler
  - Ontwerp van de code
  - Code voor het systeem
  - Aanvullende documentatie over gebruik/werking van de code
  - Testplan
  - Testsituaties
  - Testresultaten
  - Documentatie over de implementatie/overdracht

- **Te demonstreren competenties en wijze waarop**

  - A1 – Analyseren probleemdomein – niveau 3
    - Aan het begin van de stage moet worden bekeken waar het probleem ligt in de huidige situatie. Op deze manier ontstaat er een duidelijk beeld waar de rest van het project de oplossing voor moet gaan bieden.
  - A4 – Kiezen van een ontwikkelstrategie – niveau 2
    - Tijdens het uitvoeren van het project wordt gewerkt volgens een bepaalde ontwikkelstrategie. Bij aanvang van het project moet dus ook een afweging worden gemaakt welke methode er wordt gekozen.
  - C13 – Het betrekken van Real-time aspecten bij een ontwerp – niveau 3
    - Tijdens het uitvoeren van de opdracht moet er een ontwerp worden gemaakt waarin rekening wordt gehouden met het real-time aspect van het product.
  - D17 – Testen van softwaresystemen – niveau 3
    - Om uiteindelijk te controleren of het product naar wens functioneert moet het product uitvoerig getest worden. Via het verschillende testdocumenten wordt dit gewaarborgd.
  - H7 – Professioneel werken: Methodisch werken – niveau 3
    - De ontwikkel methode die aan het begin is vastgesteld wordt tijdens de gehele doorlooptijd van het project worden gevolgd.

Bijlage B: Plan van aanpak

Plan van aanpak

Project:
Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf:
Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

1. Achtergronden

Bedrijf

Delft Dynamics B.V. is in februari 2006 opgericht en heeft zich gespecialiseerd in het bouwen van robothelikopters: kleine onbemande helikopters die door het slim samenvoegen van computer- en sensorapparatuur gebruikt kunnen worden als stabiel en makkelijk te besturen sensorplatform. Een aantal robothelikopters is geleverd aan KLPD en Defensie voor operationele evaluatie.

In 2013 brengt Delft Dynamics haar nieuwste product op de markt: de RH4 'Spyder'. Deze compacte en lichtgewicht multicopter kan verschillende soorten camera's dragen en is zeer gemakkelijk te bedienen. Op deze wijze is het mogelijk om op een eenvoudige manier vanuit de lucht foto- en videobeelden te maken van de omgeving.

Afbeelding 1: Delft Dynamics RH4 'Spyder'

Probleemstelling

De huidige avionica van de RH4 Spyder stabiliseert de multicopter en verzorgt ook de compensatie voor de bewegingen van de multicopter voor de camera. Door de ontwikkelingen in embedded processing en miniatuur sensoren wordt het nu rendabel om een extra (ARM)-processor met eigen sensoren toe te voegen in de camera zelf, zodat een snellere camera-stabilisatie gerealiseerd wordt, zodat de camerabeelden beter worden. Hierna is het mogelijk om eventueel extra functionaliteit aan de ARM-processor toe te kennen, om bijvoorbeeld naast het stabiliseren ook het aansturen van de camera zelf te regelen. Dit is in de huidige situatie niet mogelijk, omdat de huidige avionica aan het maximum van zijn kunnen zit.

Betrokkenen

De opdrachtnemer van dit project is Jeroen de Meij. Hij voert dit project uit als afstudeer project voor de opleiding Technische Informatica, van de Haagse Hogeschool, afdeling Delft. Hierbij functioneert Gerwin Voorsluijs, werknemer bij Delft Dynamics, als begeleider bij het project. Hij heeft binnen het project ook de rol van opdrachtgever.

2. Projectopdracht

Doelstelling

Op basis van beschikbare hardware (camera, gimbal met motortjes, ARM-processor, sensoren) zal een werkend camera-stabilisatie-systeem gemaakt worden. Aan het einde van de opdracht moet het huidige camera-stabilisatie-systeem zijn overgezet en geoptimaliseerd voor de Teensy 3.0.

Om dit te bereiken zullen de sensoren op hoge frequentie (1 kHz) uitgelezen worden en samen met de aansturing vanuit de avionica gebruikt worden om de motoren aan te sturen zodat de camera de gewenste stand aanneemt.

Als extra ontwikkelings stap wordt de huidige sensor vervangen door een andere versie van hetzelfde type. De huidige sensor is de MPU6050. Deze kan alleen communiceren via I2C en stoort zo af en toe. De verwachting is dat dit aan de communicatie ligt. In het nieuwe systeem wordt de MPU6000 gebruikt. Dit is dezelfde sensor, maar dan met de mogelijkheid om via SPI te communiceren.

Voor het uitvoeren van deze opdracht zal gebruik worden gemaakt van de GCC compiler voor ARM processoren en SPI, PWM en RS232 communicatie-protocollen. De werking van het stabilisatie-systeem zal zowel stand-alone als vliegend getest worden en beoordeeld op snelheid en nauwkeurigheid.

De opdracht moet, inclusief alle bijbehorende documentatie, uiterlijk 6 juni 2014 voor 12:00 uur af zijn. Dit is de deadline waarneer het projectverslag van Jeroen bij zijn begeleider op school moet zijn ingeleverd.

Resultaat

Het uiteindelijke resultaat is een geoptimaliseerde versie van het huidige camera-stabilisatie-systeem voor op de Teensy 3.0.

3. Projectactiviteiten

Hieronder wordt opgesomd welke activiteiten er tijdens het project plaatsvinden:

Documentatie

- Maken begrippenlijst
- Maken plan van aanpak
- Maken requirements document
- Maken test plan
- Logboek met per dag de activiteiten aangegeven bijhouden

Onderzoek

- Onderzoeken van het huidige camera-stabilisatie-systeem en de problemen daarin
- Onderzoek naar de wensen en eisen van het nieuwe systeem
- Onderzoek naar de werking van de Teensy 3.0
- Onderzoek naar de configuratie van de gcc compiler

Ontwerp

- Globaal ontwerp van het camera-stabilisatie systeem
- Technisch ontwerp van het camera-stabilisatie-systeem

Ontwikkeling

- Ontwikkeling van de code voor het camera-stabilisatie-systeem
- Ontwikkelen van een testplan met bijbehorende testcases

Project afhandeling

- Oplevering documentatie
- Overdracht camera-stabilisatie-systeem

4. Projectgrenzen

Aan het einde van het project moet er een werkend camera-stabilisatie-systeem worden opgeleverd. Dit houdt in dat het systeem werkend onder de RH4 'Spyder' moet hangen en de camera moet stabiliseren. Het verwerken van die camerabeelden behoort niet binnen dit project. Net zo min het doorsturen van de cameraposities vanaf de grond naar de 'Spyder'.

De communicatie tussen het camera-stabilisatie-systeem en de 'Spyder' valt buiten de opdracht. Wel moet bedacht worden dat het in de toekomst gewenst is om informatie vanaf de 'Spyder' te ontvangen.

Wanneer het stabilisatie-systeem ruim voor tijd is overgezet op de Teensy 3.0 kan worden gekeken naar het toevoegen van extra functionaliteit aan het systeem, zoals het bedienen van de camera zelf.

Daarnaast moet al de informatie worden gedocumenteerd, zodat later terug is te vinden welke keuzes er zijn gemaakt tijdens de ontwikkeling van het systeem en op welke manier het systeem werkt.

Dit alles moet zijn afgerond voor 6 juni 2014, 12:00 uur.

5. Producten

Hieronder volgt een lijst met producten die tijdens dit project worden opgeleverd:

- Begrippenlijst
- Plan van aanpak
- Oriëntatie document
- Requirements analyse
- Beschrijving installatie gcc compiler
- Ontwerp van de code
- Code voor het systeem
- Aanvullende documentatie over gebruik/werking van de code
- Testplan
- Testsituaties
- Testresultaten
- Documentatie over de implementatie van het systeem in de werkomgeving
- Twee presentaties
  - Eerste presentatie over de werkzaamheden, aan het begin van het project.
  - Tweede presentatie over het resultaat, aan het eind van het project.
- Logboek met per dag de activiteiten aangegeven

6. Kwaliteit

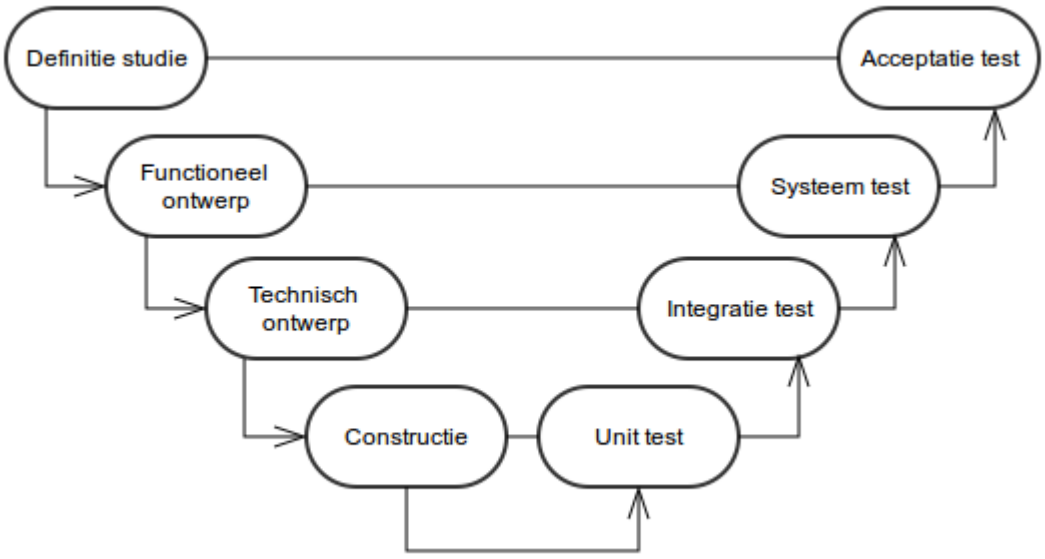
Het systeem moet aan het eind van het project op dezelfde manier functioneren als het huidige systeem. Dat houdt grofweg in dat het systeem stabiel moet kunnen filmen. Het is moeilijk om vooraf te omschrijven wanneer een beeld stabiel genoeg is. Daarom is vastgelegd dat het systeem met een snelheid van 1 kHz de sensoren moet uitlezen en deze informatie moet verwerken.

Kwaliteitsbewaking

Om de kwaliteit van het project te waarborgen is een aantal maatregelen genomen. Allereerst vindt er wekelijks op woensdag een overleg plaatst met de opdrachtgever/begeleider. Tijdens dit overleg wordt aan de hand van het logboek gekeken wat er die week is uitgevoerd. De resultaten van dit werk worden besproken en er wordt bekeken of dit nog overeenkomt met de wensen.

Bepaling ontwikkelmethode

Naast het wekelijkse overleg wordt er in dit project een bepaalde ontwikkelmethode gevolgd. Er is voor gekozen om te werken via de waterval methode en specifiek die van het V-model. Dit is een lineaire ontwikkelmethode. Hier is voor gekozen, omdat het project niet heel omvangrijk is, onderzoeken en ontwerpen neemt de meeste tijd in beslag binnen dit project, de gebruikers van het systeem zijn zeer technisch en het is duidelijk wat het uiteindelijke doel van het project is en dit doel zal niet tot nauwelijks veranderen. Het enige grote aandachtspunt is dat de opdrachtnemer nog niet bekend is met de te gebruiken hardware en zich daar in moet gaan verdiepen. In het V-model is hiervoor aan het begin de tijd voor, waardoor dit ook geen kritiek punt zal vormen.



Afbeelding 2: Het V-model

7. Projectorganisatie

De project organisatie bestaat hoofdzakelijk uit twee personen. De opdrachtgever, Gerwin Voorsluijs, die namens Delft Dynamics de contact persoon is voor de opdracht en de opdrachtnemer, Jeroen de Meij, die bij Delft Dynamics deze opdracht uitvoert als afstudeerproject. De overige werknemers bij Delft Dynamics zijn niet direct betrokken bij de ontwikkeling van dit project, maar zullen uiteindelijk wel betrokken worden bij de kwaliteitsbepaling van het systeem. Daarnaast is het mogelijk om de overige werknemers te benaderen voor vragen.

8. Planning

Bij het maken van de planning is rekening gehouden met de gekozen ontwikkelmethode. Het waterval model kent een aantal verschillende fasen:

- Definitiestudie/analyse
- Basisontwerp
- Technisch ontwerp/detailontwerp
- Bouw/implementatie
- Testen
- Integratie
- Beheer en onderhoud

In de onderstaande tabel wordt aangegeven in welke periode de verschillende fasen plaatsvinden, wat de globale werkzaamheden zijn en wat de mijlpalen zijn die aan het einde van de fase worden opgeleverd. Hierbij is de laatste fase erg lang, om op die manier ook wat uitlooptijd en tijd om aan verslaglegging te werken, in te plannen.

Definitiestudie/ analyse	Maandag 10 – dinsdag 18 februari 2014
	In deze fase wordt er een beter beeld gekregen van het project. Zo wordt er een plan van aanpak geschreven die als basis dient voor de rest van het project en wordt gekeken hoe het huidige systeem in elkaar zit en wordt er hierover een probleemanalyse uitgevoerd.
	Mijlpalen: Begrippenlijst, plan van aanpak, oriëntatie document
Basisontwerp	Woensdag 19 februari – zondag 9 maart 2014
	In deze fase wordt gekeken naar het toekomstige systeem. Er wordt gekeken wat het systeem moet kunnen en de informatie over de verschillende hardware onderdelen wordt opgezocht. Vervolgens wordt met deze informatie een globaal ontwerp van het systeem gemaakt.
	Mijlpalen: Presentatie, Requirements document, onderzoek naar gcc compiler
Technisch ontwerp/ detailontwerp	Maandag 10 – zondag 23 maart 2014
	In deze fase wordt dieper ingegaan op het globale ontwerp van de vorige fase. Er wordt bijvoorbeeld beschreven op welke manier de sensoren worden uitgelezen en op welke manier de motoren worden aangestuurd. Hierbij moet worden gedacht aan een beschrijving van communicatie protocollen die worden gebruikt en de manier waarop en op welke pinnen de verschillende componenten worden aangesloten. Ook wordt de laatste hand gelegd aan het testplan.
	Mijlpalen: Gesoldeerde Teeny 3.0, Technisch ontwerp, testplan met testcases
Bouw/ implementatie	Maandag 31 maart – zondag 27 april 2014
	In deze fase wordt het systeem ontwikkeld. Aan de hand van de eerste gemaakte ontwerpen wordt de code gemaakt voor dit systeem en worden tijdens het ontwikkelen unit tests uitgevoerd.
	Mijlpalen: code

Testen	Maandag 28 april – zondag 11 mei 2014
	In deze fase wordt het systeem uitvoerig getest aan de hand van het eerder opgestelde testplan. Hierbij worden de integratie tests en systeem tests uitgevoerd. Bij de integratietest wordt getest of de losse software onderdelen samen werken en ook of de koppeling tussen de 'Spyder' avionica en het camera-stabilisatie-systeem goed werkt. Daarnaast wordt in de systeem test gekeken of het systeem helemaal naar behoren functioneert. Tijdens het testen kan het voorkomen dat stukken van het systeem moeten worden aangepast.
	Mijlpalen: Testresultaten
Integratie	Maandag 12 mei – vrijdag 6 juni 2014
	In deze fase wordt het camera-stabilisatie-systeem aan de 'Spyder' bevestigd en wordt via een systeem en acceptatie test gekeken of het systeem ook in de lucht nog steeds goed functioneert en of de opdrachtgever tevreden is met het resultaat. Tijdens het testen kan het voorkomen dat stukken van het systeem moeten worden aangepast.
	Mijlpalen: Testresultaten, geïmplementeerd systeem, bijbehorende documentatie, presentatie
Beheer en onderhoud	Vanaf vrijdag 6 juni 2014
	Deze fase valt in principe buiten de scope van dit project. Maar tijdens dit project wordt documentatie geschreven over de werking van het systeem, waardoor het mogelijk wordt om het systeem in de toekomst te kunnen beheren en onderhouden.
	Mijlpalen: -

Tabel 1: Globale indeling projectfasen

Een stokenplanning is toegevoegd in hoofdstuk 11

9. Kosten en baten

Kosten

De grootste kostenpost in dit project is de vergoeding die de werknemer krijgt voor de uitvoering van dit project. Daarnaast zal de opdracht gever ook tijd kwijt zijn aan de begeleiding van dit project. De totale loon kosten van dit project zullen neer komen op ongeveer €2000,-.

Naast deze loonkosten wordt er niet heel veel geld uitgegeven aan dit project. Er is een Teensy 3.0 aangeschaft voor in totaal $32,-, wat neerkomt op ongeveer €23,50. De overige hardware componenten die tijdens dit project worden gebruikt zijn al aanwezig, waardoor deze componenten al bij een vorig project zijn gedeclareerd.

Baten

Wat de baten uiteindelijk zullen zijn, is op dit moment nog moeilijk in te schatten. Het project zorgen voor een verbetering van het huidige product. Het kan zijn dat door de verbeterde beeldkwaliteit van het systeem de huur/koopprijs van de 'Spyder' omhoog gaat en klanten eerder de 'Spyder' van Delft Dynamics kopen, in plaats van een gelijkwaardig product van een concurrerend bedrijf.

10. Risico's

Hieronder wordt een aantal risico's genoemd die een bedreiging kunnen vormen voor het slagen van het project. Per risico is aangegeven wat de maatregelen zijn om dit risico te verminderen.

Harde deadline

De deadline van het project is op 6 juni 2014 om 12:00 uur. Dit is het moment waarop het afstudeerverslag, behorend bij dit project, moet zijn ingeleverd bij de afstudeerbegeleider van de opdrachtnemer. Hierdoor is het niet mogelijk om het project iets te verlengen, mocht dat wenselijk zijn.

Om dit risico te verminderen is de planning ruim opgezet en is er uitlooptijd ingebouwd. Daarnaast vindt er iedere week een overleg plaatst met de begeleider/opdrachtgever, waarbij onder andere wordt gekeken of het project nog steeds haalbaar is binnen de vastgestelde tijd.

Onbekendheid bij hardware

Tijdens dit project wordt er gewerkt met een Teensy 3.0. De opdrachtnemer is nog onbekend met het type platform en de ARM processor die erop zit. Daarnaast moet tijdens het project het systeem in elkaar worden gesoldeerd. Dit heeft de opdrachtnemer ook nog nooit gedaan, waardoor dit ook binnen het project moet worden aangeleerd.

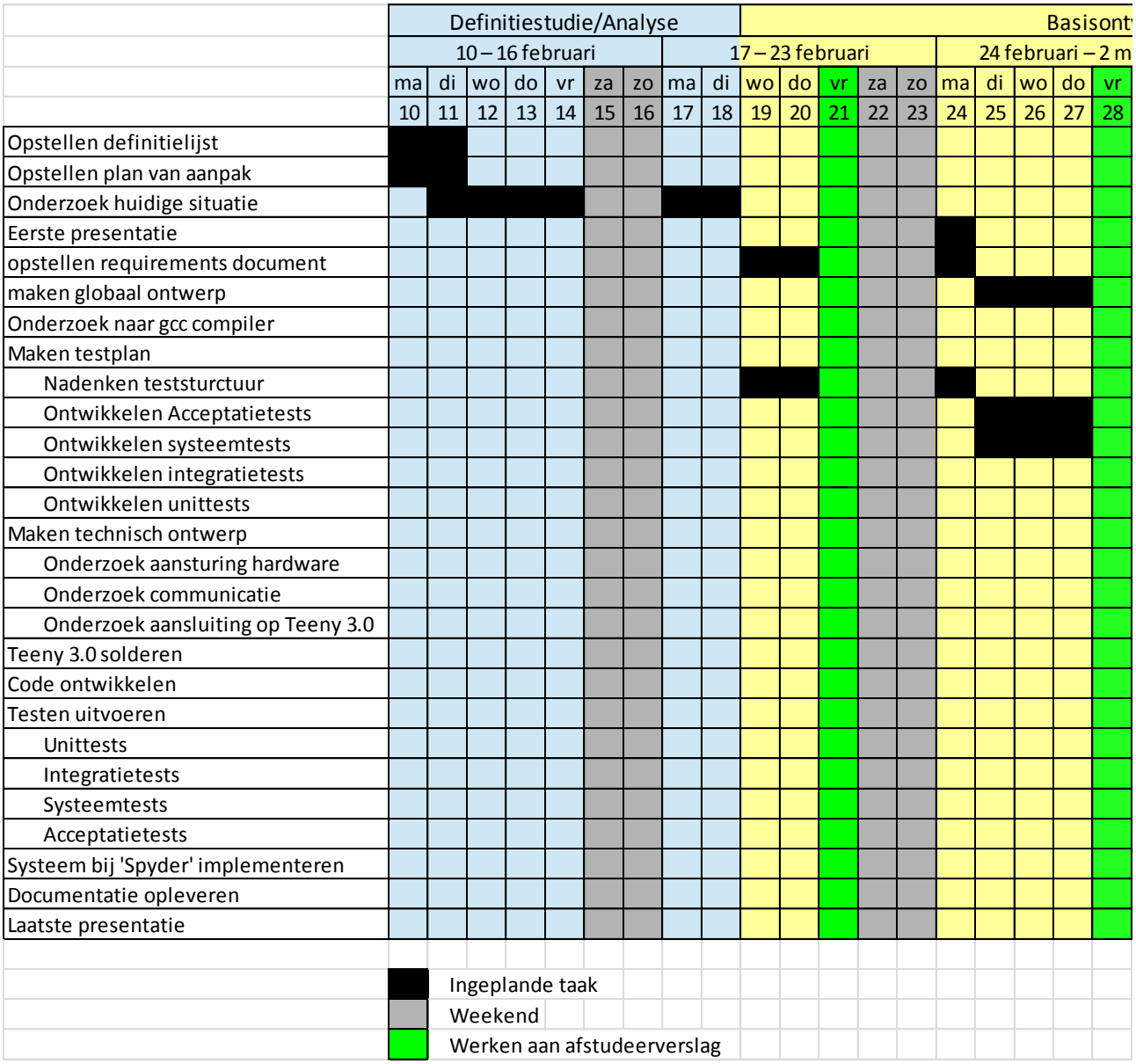
Om dit risico te verminderen is er extra tijd ingepland voor onderzoek en het aanleren van deze bekwaamheden.

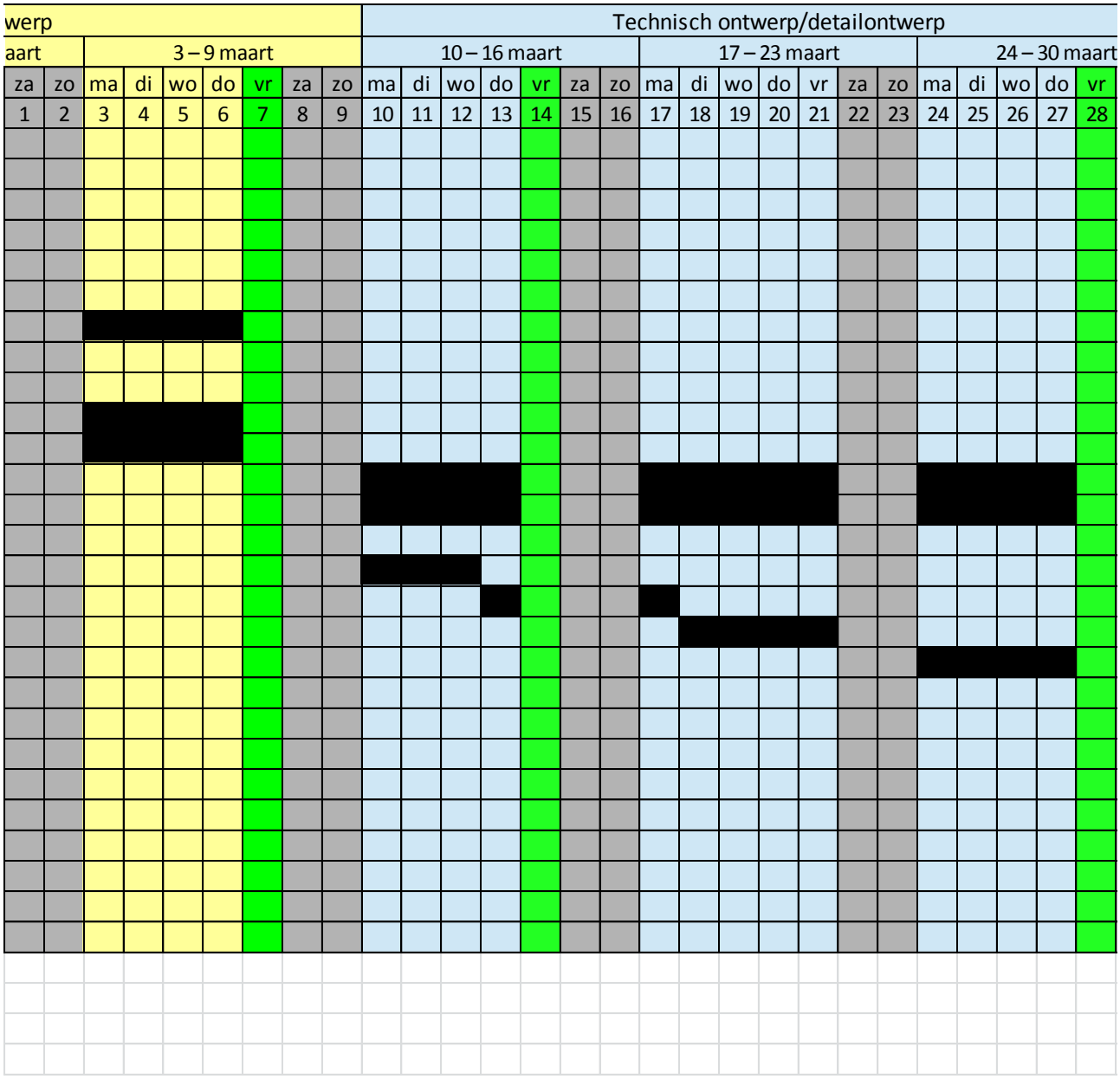
Onverwachte omstandigheden

Bij onverwachte omstandigheden moet worden gedacht aan niet geplande oorzaken waardoor het project niet kan worden voortgezet. Mogelijke oorzaken zijn bijvoorbeeld ziekte van een projectlid, brand, vaststaan in het verkeer et cetera.

Het is niet mogelijk het risico hiervan te verminderen. Wanneer dit probleem zich voordoet moet opnieuw naar de planning worden gekeken en moet deze eventueel worden aangepast. De planning is ruim opgezet, waardoor er een buffer is om deze onverwachte omstandigheden te kunnen opvangen.

11. Strokenplanning




werp
Technisch ontwerp/detailontwerp
aart
3 – 9 maart
10 – 16 maart
17 – 23 maart
24 – 30 maart
za zo ma di wo do vr za zo ma di wo do vr za zo ma di wo do vr za zo ma di wo do vr
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

		Bouw/implementatie																								
		31 maart – 6 april							7 – 13 april							14 – 20 april							21 – 27 april				
za	zo	ma	di	wo	do	vr	za	zo	ma	di	wo	do	vr	za	zo	ma	di	wo	do	vr	za	zo	ma	di	wo	do	vr
29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

		Testen																								
		28 april – 4 mei							5 – 11 mei							12 – 18 mei							19 – 25 mei				
za	zo	ma	di	wo	do	vr	za	zo	ma	di	wo	do	vr	za	zo	ma	di	wo	do	vr	za	zo	ma	di	wo	do	vr
26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Integratie																
		26 mei – 1 juni							2 – 8 juni							
za	zo	ma	di	wo	do	vr	za	zo	ma	di	wo	do	vr	za	zo
24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8

Bijlage C: Oriëntatie document

Oriëntatie document

Oriëntatie op het huidige camera-stabilisatie-systeem dat wordt gebruikt bij Delft Dynamics

Jeroen de Meij

Ter Aar, donderdag 8 mei 2014

Probleem analyse

Situatie beschrijving

Op dit moment bestaat het camera-stabilisatie-systeem uit drie verschillende hardware compontenten.

- Arduino Mini als controller
- MPU6050 sensor (3 assige gyroscoop en acceleratie sensor)
- Gymbal met twee motoren

Op het internet wordt een gratis framework aangeboden, wat is ontwikkeld voor de besturing van deze drie componenten.

Probleem

Op dit moment is de code van het framework niet uitbreidbaar. Het is een warboel van allemaal functies die te maken hebben met algemeen gebruik van het systeem. Hierdoor is niet meteen duidelijk wat de functie is van alle codes. Daarbij wordt ook nauwelijks gebruikt gemaakt van klasses, maar staan alle functies die behoren bij een aspect van het systeem los in een file gezet. Ook zijn er twee files waarin vrijwel alle defines en globale variabelen staan die in het systeem gebruikt worden, zonder dat duidelijk is bij welk onderdeel van het systeem ze thuis horen. De wens om in de toekomst meer functionaliteit aan dit systeem toe te voegen is daarom op dit moment niet mogelijk vanwege de warrige code. Daarnaast is ook het probleem dat de Arduino Mini aan het maximum van zijn kunnen zit en geen extra functionaliteit toebedeeld kan krijgen.

Naast de warrige code komt het af en toe voor dat het systeem vastloopt. Dit gebeurt op het moment dat het systeem meer moet stabiliseren dan fysiek nog mogelijk is. Op deze momenten tikt de camera tegen de onderkant van de helicopter aan en kan dus niet verder stabiliseren. Uit testen is gebleken dat het systeem vast loopt op het moment voordat er informatie vanaf de sensor wordt teruggegeven. De opdrachtgever denkt dat dit te maken heeft met de I2C communicatie, waarbij op dat moment de master (Arduino Mini) tevergeefs wacht op informatie van de slave (MPU6050).

De oplossing

Om in de toekomst meer functionaliteit aan het project te kunnen toevoegen, heeft de opdrachtgever ervoor gekozen om een heel ander bordje te gebruiken als controller, de Teeny 3.0. Daarnaast heeft de opdrachtgever er ook voor gekozen om een andere versie van de sensor te gebruiken. Dit wordt de MPU6000. Dit is vrijwel dezelfde versie als de 6050, maar met de extra mogelijkheid om via SPI te communiceren in plaats van alleen via I^2C. De hoop is dat het systeem met deze nieuwe communicatie mogelijkheid niet meer vastloopt.

Om het probleem van de warrige code op te lossen moet de huidige project code worden doorgekeken, om te ontdekken hoe deze in elkaar zit. Daarna kan op basis van die kennis een globale opzet van het systeem worden gemaakt. Daarbij moet worden gedacht aan het opstellen van de eisen van het systeem, het globaal kijken naar hoe de verschillende hardware onderdelen met elkaar gaan communiceren en op welke manier het project wordt ontwikkeld.

Als deze globale opzet bekend is kan dieper worden ingegaan op het technische ontwerp van het systeem. Zo kan worden bepaald hoe de hardware gaat worden aangestuurd, op welke pinnen de verschillende hardware componenten vervolgens moeten worden aangesloten en moet er een ontwerp komen van het framework waarbij er een aantal verschillende klassen wordt ontworpen om de structuur van het systeem netjes te maken. Daarna kan daadwerkelijk worden begonnen met de ontwikkeling van het framework. Hierbij moet worden gekeken welke code van het oude framework hergebruikt kan worden voor het nieuwe framework.Tegelijkertijd wordt er grondig aandacht besteed aan het testen van het systeem, om er zeker van te zijn dat na de herstructurering van het systeem de werking en kwaliteit van het systeem niet veranderd zijn.

De verschillende projectfiles

In de onderstaande tabel zijn de bestanden terug te vinden die in het framework gebruikt worden. Er is per file geprobeerd een beeld te krijgen wat de functie is van de file en hoe deze samenwerkt met de andere files in het framework.

BLcontroller.h	In deze file vindt de aansturing van de motoren plaats. Er is een initialisatie functie waar de juiste pinnen op output worden gezet, de PWM modus voor de motoren wordt gezet en de interrupts worden aangezet. Daarnaast zijn er drie verschillende functies om de motor aan te passen. Eén om de stand van de motor aan te passen, een andere om snel de motor te laten bewegen en een derde om de motor uit te zetten. Daarnaast is er een functie die de nieuwe stand van de motor kan berekenen. Deze functie maakt gebruik van een functie om snel een sinus array uit te rekenen. Vervolgens is er een TIMER1 overflow interrupt. Wanneer deze optreedt wordt gekeken of het al tijd is om de motoren up te daten. Wanneer dat het geval is wordt dit ook gedaan. Tot slot is er een functie om de motoren te testen. Hierbij draait eerst de ene motor helemaal heen en dan weer terug, waarna hetzelfde wordt gedaan met de tweede motor.
BruGi.ino	Dit bestand is het hoofdbestand van het project. Omdat het geschreven is volgens de Arduino architectuur bevat het een setup() functie, die wordt uitgevoerd als het systeem wordt aangezet, en een loop() functie, die herhaald wordt uitgevoerd. **Setup()** In de setup functie wordt eerst de seriële communicatie opgezet om debug berichten te kunnen tonen. Vervolgens wordt uit de EEPROM geprobeerd om een configuratie file uit te lezen. Wanneer dit niet lukt wordt de standaard configuratie gebruikt. Vervolgens worden de motoren in de juiste stand geplaatst. Wanneer dit is gebeurd worden de begin waarden van de PID-regelaar gezet voor de twee verschillende motoren. Als er gebruik wordt gemaakt van een MPU6050 wordt de I2C communicatie geconfigureerd. Vervolgens worden de timers en interrupts van de motoren aangezet. Als er gebruik wordt gemaakt van een MPU6050, wordt eerste de resolutie van de sensor gezet. Daarna wordt de IMU geïnitialiseerd, het adres van de mpu gezet en de verbinding getest. Ook wordt de sensor oriëntatie goed gezet en worden er nog een aantal configuraties aan de sensor aangepast. Vervolgens wordt de offset van de Gyroscoop bepaald en de low-pass filter van de gyroscoop gezet. Aan het einde worden de timers en interrupts van de motoren nogmaals aangezet. Tot slot worden de pinnen waarop de informatie van de radio control binnenkomt geïnitialiseerd en de bijbehorende interrupts gezet. **loop()** Deze functie wordt door de Arduino iedere keer opnieuw wordt aangeroepen en is als het ware een grote while(True) loop. Aan het begin van de functie wordt gekeken of de variabele motorUpdates True is.

Deze variabele wordt bij aanvang van het systeem in de file variables.h op False gezet, maar in de overflow interrupt van de file BLcontroller.h op True gezet. Wanneer de waarde True is wordt deze direct op False gezet. Vervolgens wordt gekeken of de MPU aan staat en wanneer dit zo is wordt de gyroscoop uitgelezen en vervolgens de Gyroscoop en de acceleratie sensor geüpdated. Daarna wordt de hoek van de sensor bepaald.

Vervolgens owrdt gekeken of de variabele do_setup op 0 staat. Wanneer dit zo is wordt de Seriele verbinding geïnitialiseerd.

Daarna worden van zowel de roll als de pitch motor de nieuwe PID waardes berekend en hiermee de positie van de motoren bepaald.

Hierna volgt een switch statement met als input de variabele count die aan het begin van het programma op 0 is gezet in de file variables.h. Deze variabele heeft 10 cases.

- Case 1:
  - De acceleratie van de roll as wordt uitgelezen.
- Case 2:
  - De acceleratie van de pitch as wordt uitgelezen.
- Case 3:
  - De acceleratie van de yaw as wordt uitgelezen.
- Case 4:
  - Hier wordt de acceleratie berekent.
- Case 5:
  - Doet niets
- Case 6:
  - In deze case komen weer twee switch statements voor. Beide switchs kijken naar de variabele gimState die de status van de gimbal bevat en aan het begin van het systeem op GIM_IDLE wordt gezet. Switch 1:
    - Case GIM_IDLE:
      - Hier wordt door middel van een if statement en een variabele die wordt opgehoogd, twee seconden gewacht voordat de gimState op GIM_UNLOCKED wordt gezet.
    - Case GIM_UNLOCKED:
      - Hier wordt door middel van een if statement en een variabele die wordt opgehoogd, een paar seconden gewacht voordat de gimState op GIM_LOCKED wordt gezet.
    - Case GIM_LOCKED:
      - Deze case doet niets

Switch 2:

- Case GIM_IDLE:
  - Zet enableMotorUpdates op false en setACCFastMode op true.
- Case GIM_UNLOCKED:

	• Zet enableMotorUpdates op true en setACCFastMode op true. ▪ Case GIM_LOCKED: • Zet enableMotorUpdates op true en setACCFastMode op false. • Case 7: ◦ Doet niets • Case 8: ◦ Doet niets • Case 9: ◦ Hier worden een aantal outputs gegenereerd mocht dat wenselijk zijn. • Case 10: ◦ Wanneer het aan staat wordt er gecontroleerd hoeveel geheugen er nog vrij is. Na deze switch statement wordt nog gekeken of er een RC timeout is in het geval radio control aan staat.
definitions.h	In deze file staan een heleboel define's. Hier worden vaste waarden gedefinieerd, maar ook worden er veel gebruikte functies naar een compactere naamgeving omgezet.
EEPROMAnything.h	In deze file is het mogelijk om via twee template functies alle mogelijke informatie op te slaan in of te lezen uit de EEPROM. Deze functies geven vervolgens aan hoeveel informatie er is gelezen of weggeschreven.
fastMathRoutines.h	In deze file staat een aantal functies voor het snel uitvoeren van grote wiskundige berekeningen. • constrain_int32 ◦ Deze functie wordt gebruikt in de berekeningen van de PID's van de gyroscopen. • Rajan_FastArcTan ◦ In deze functie wordt de Arc Tangens berekend. Deze functie wordt alleen gebruikt in de Rajan_FastArcTan2 functie • Rajan_FastArcTan2 ◦ Deze functie berekent de arc tangens van twee getallen en geeft een antwoord in radialen. Deze functie wordt alleen gebruikt in Rajan_FastArcTan2_deg1000 • Rajan_FastArcTan2_deg1000 ◦ Deze functie zet de uitkomst van de Rajan_FastArcTan2 om in graden. Deze functie wordt alleen gebruikt in de loop() van BruGi.ino om de richting van de vectoren te kunnen bepalen. • utilLP_float ◦ De werking van de functie is niet geheel duidelijk (**q += (i-*q)*coeff*), maar hij wordt alleen gebruikt in de functie evalRCChannelAbsolute uit de file RCDecode.h. En deze laatste functie wordt allen nog gebruikt in

	code die is weg gecomment. • stackCheck ◦ Deze functie kijkt of de stackbottem en de stacktop nog wel actueel zijn. Deze functie wordt alleen gebruikt in de loop van BruGi.ino op het moment dat debugging aan staat. • heapCheck ◦ Deze functie kijkt of de heapbottem en de heaptop nog wel actueel zijn. Deze functie wordt alleen gebruikt in de loop van BruGi.ino op het moment dat debugging aan staat. • StackHeapEval ◦ Deze functie zoekt uit hoeveel geheugen er nog vrij is. De functie wordt alleen aangeroepen in de loop van BruGi.ino op het moment dat STACKHEAPCHECK_ENABLE is gedefiniëerd. • CrcSlow ◦ Deze funcite voert een crc uit. De functie wordt alleen gebruikt in de file SerialCom.h bij het lezen en schrijven naar de EEPROM.
I2Cdev.cpp	Deze file bevat een aantal functies om gebruik te maken van I^2C communicatie. De file is goed gedocumenteerd, wat inhoudt dat met het programma doxygen een wiki te genereren is waarin precies staat uitgelegd wat de functie doet, wat de parameters betekenen en wat de functie mee teruggeeft. Er is verder niet veel aandacht besteed aan deze file, aangezien in het nieuwe systeem er hoogstwaarschijnlijk geen I^2C communicatie voorkomt.
I2Cdev.h	Dit is de headerfile behorende bij de file I2Cdev.cpp
IMU.ino	Deze file bevat een aantal functies om de IMU (inertial measurement unit) te besturen. Deze file maakt meerdere malen gebruik van de file MPU6050.cpp om precieze gegevens op te vragen aan specifieke sensor. In de funcite initSensorOrienetationDefault wordt een aantal variabelen gezet die aangeven op welke wijze de sensor is gemonteerd. Deze functie wordt gebruikt door de functie initSensorOrientation, waarbij na het aanroepen van die functie wordt gekeken of er een aantal assen moet worden verwisseld/omgedraaid. De functie setACCFastMode geeft de mate aan waarin de acceleratie sensor van invloed is op de stand van de motoren. De funcite initIMU zet een aantal variabelen die door de IMU wordt gebruikt. De functie rotateV roteert de meegegeven vector een klein beetje aan de hand van een meegegeven delta. De functie readGyros leest de waarde van de gyroscoop uit en past de waarde aan, aan de hand van de offset en plaatsing van de sensor. De funtie readACC berekent de acceleratie van de meegegeven as en past deze aan, aan de hand van de plaatsing van de sensor. De functie updateGyroAttitude zorgt ervoor dat de functie roteteV wordt aangeroepen met de juiste waardes. De functie updateACC berekent de totale versnelling van de sensor door middel van

	de versnelling van alle assen. Daarna wordt er gekeken of er sprake is van een grote of een kleine verplaatsing. De functie updateACCAttitude berekent de waarde waarmee de acceleratie van invloed is op de correctie van de motoren aan de hand van de gelezen versnelling en aanpassings constante. De funcitie getAttitudeAngles berekent via een offset en een Arc Tangens de hoek van de roll en de pitch as.
MPU6050.cpp	Deze file bevat een aantal functies om gebruik te maken van de MPU6050 sensor. De file is goed gedocumenteerd, wat inhoudt dat met het programma doxygen een wiki te genereren is waarin precies staat uitgelegd wat de functies doen, wat de parameters betekenen en wat de functies mee terug geven. Er is verder niet veel aandacht besteed aan deze file, aangezien in het nieuwe systeem een andere sensor wordt gebruikt.
MPU6050.h	Dit is de headerfile behorende bij de file MPU6050.cpp
orientationRoutines.h	Deze file bevat twee functies. De eerste functie zet de resolutie deler van de gyroscoop aan de hand van een define. De tweede functie berekent de offset van de gyroscoop. Dit gebeurt door de motoren uit te zetten en 1 seconde te wachten. Daarna is er een while loop die een vast aantal keer wordt doorlopen en telkens twee keer de stand van de gyrosensor opvraagt en met elkaar vergelijkt of er geen beweging is waargenomen. Daarna wordt het gemiddelde genomen van alle gemeten afwijking van het 0 punt en tot slot opgeslagen in een offset array die per as de offset bevat.
PinChangeInt.h	In deze file staat een aantal functies die het mogelijk maken om interrupts aan de arduino pinnen te hangen. Aangezien er met een heel ander bordje wordt gewerkt en de code alleen gebruikt wordt in de file Rcdecode, welke gaat over het verwerken van radio control signalen die niet gebruikt worden, wordt ervan uit gegaan dat deze code niet gebruikt gaat worden in het nieuwe systeem, waardoor er niet veel tijd aan is besteed.
Rcdecode.h	In deze file staat een aantal functies die het mogelijk maken om radio control signalen te gebruiken. Aangezien dit niet gebruikt gaat worden in het nieuwe systeem, is er niet veel tijd aan besteed.
SerialCom.h	In deze file staat een aantal functies die gebruikt worden om configuratie van bepaalde onderdelen aan te passen, gebruikers input te vragen en dingen weg te schrijven naar of te lezen uit de EEPROM. Aangezien veel van deze functionaliteit niet in het systeem geboden worden en anders zelf herschreven worden, is weinig tijd besteed aan het doornemen van deze file.
SerialCommand.cpp	Deze file bevat een aantal functies die gebruikt worden om User input te kunnen verwerken. Aangezien dit niet in eerste instantie van toepassing is op het nieuwe systeem is hier niet veel tijd aan besteed.
SerialCommand.h	Dit is de headerfile behorende bij de file SerialCommand.cpp
variables.h	In deze file wordt een aantal variabelen geïnitialiseerd en gedeclareerd die door het gehele systeem heen worden gebruikt.

Bijlage D: Requirements document

Requirements document

Project:
Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf:
Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

Introductie

Doel van het systeem

Het doel van het systeem is zorgen voor een gestabiliseerde camera op het moment dat deze onder een RH4 'Spyder' multicopter wordt gehangen. Hierbij wordt gebruikt gemaakt van een gymbal en een IMU. De roll en pitch bewegingen die de multicopter worden gemeten met de IMU, waarna er wordt berekend welke aanpassingen de gymbal moet maken om de camera stabiel te houden. Op deze manier is het mogelijk om met de multicopter een stabiel beeld op te nemen.

Wanneer dit doel bereikt is kunnen extra toevoegingen aan het systeem gedaan worden, waardoor, naast het stabiliseren van de camera, de camera bijvoorbeeld ook zelf te bedienen is. Hiermee moet rekening worden gehouden bij de ontwikkeling van het systeem, maar deze toevoegingen behoren nog niet bij de huidige eisen.

Definities

Het systeem wordt ontwikkelt voor Delft Dynamics. Gerwin Voorsluijs is namens Delft Dynamics de contactpersoon voor dit project.

Systeem overzicht

Het systeem moet op geheel automatische wijze in staat zijn om de bewegingen van de multicopter te meten en correcties uit te voeren om deze bewegingen te corrigeren. Hierbij hoeft de gebruiker zelf geen handelingen uit te voeren.

Referenties

De kennis die nodig was voor het opstellen van dit requirements document is in eerste instantie opgedaan door gesprekken met Gerwin Voorsluijs. Tijdens die gesprekken is duidelijk geworden wat de wensen van Delft Dynamics zijn aan het systeem en wat mogelijke uitbreidingen kunnen zijn voor in de toekomst.

Daarnaast is ook uitgebreid gekeken naar de code van het huidige systeem. Hierbij is gekeken naar de technieken die worden gebruikt om de stabilisatie mogelijk te maken en is geprobeerd, voor zover dat mogelijk was, de structuur van het framework te achterhalen.

Algemene eisen

Product perspectief

Systeem interfaces

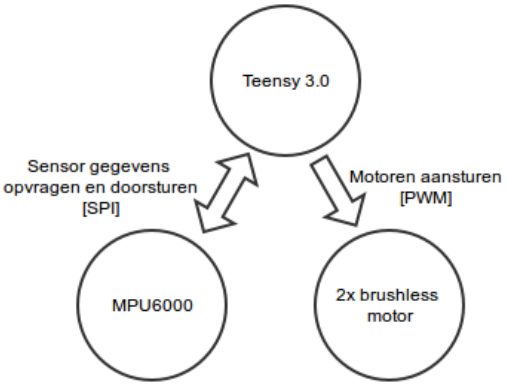
Het systeem krijgt een centrale unit die gegevens van de sensor binnen krijgt. Op deze unit worden de berekeningen uitgevoerd om vervolgens de juiste gegevens weer door te kunnen sturen naar twee motoren die de gymbal besturen, waaraan de camera bevestigd is.

User interfases

Voor het systeem wordt geen user interface ontwikkeld. De reden hiervoor is dat het systeem op geheel autonome wijzen moet kunnen functioneren. Het systeem gaat aan op het moment dat het van elektriciteit wordt voorzien. Dit gebeurt op het moment dat de multicopter waaraan het systeem is bevestigd, wordt aangezet.

Hardware interfaces

Als centrale unit wordt gebruik gemaakt van de Teensy 3.0. Dit is een devellopment bord die is uitgerust met een MK20DX128VLH5 processor. Daarnaast wordt gebruik gemaakt van een MPU6000 sensor. Deze sensor bevat een gyroscoop en acceleratie sensor. Als aansturing van de gymbal worden twee brushless motoren gebruikt.



Afbeelding 1: Communicatie tussen de hardware

Software interfaces

Het huidige systeem draait volledig in C++. Om de overzetting naar de Teensy 3.0 makkelijker te maken wordt deze taal aangehouden.

Communicatie interfaces

De communicatie tussen de Teensy 3.0 en de MPU6000 zal gebeuren via SPI. Het huidige systeem werkt op basis van I^2C, maar het lijkt erop dat deze communicatie ervoor zorgt dat het systeem soms vastloopt. De brushless motoren worden aangestuurd via PWM. De communicatie tussen de camera en de rest van het systeem wordt op dit moment nog niet ontwikkeld.

Operaties

De enige taak die het systeem op dit moment moet krijgen is het stabiliseren van de camera. Hiervoor moet het systeem zijn draaiing kunnen bepalen aan de hand van de MPU6000 en vervolgens de motoren op de juiste manier aansturen om de camera te stabiliseren.

User eigenschappen

De gebruikers van het systeem hoeven niets aan het systeem aan te passen. Wanneer het systeem aan wordt gezet, doordat de multicopter waaraan het camera-stabilisatie-systeem is bevestigd wordt aangezet, moet het systeem zelfstandig beginnen met calibreren. Het enige wat de gebruiker moet doen is het systeem op dat moment stil laten staan. Wanneer dit is gedaan moet het systeem vervolgens zelfstandig reageren op bewegingen die de multicopter maakt.

Specifieke eisen

Functionele eisen

- Het systeem moet aan de hand van de MPU6000 kunnen uitlezen wat de draaiing is van de roll en pitch as van de multicopter en met behulp van een gymbal en twee brushless motoren deze draaiing bij de camera tegen gaan. Op deze manier moet de camera stabiel kunnen filmen.
    - Om dit voor elkaar te krijgen moet de MPU6000 met een frequentie van 1kHz worden uitgelezen, waarna de stand van de motoren moet worden aangepast.

Prestatie eisen

Wanneer het systeem in gebruik is moet het de gehele tijd operationeel blijven, zonder dat hier menselijke input aan te pas hoeft te komen.

Software systeem vereisten

Reliability

Het enige moment waarop gegevens van buiten het systeem, het systeem binnenkomen, is het moment waarop de waarden van de sensor worden uitgelezen. Om ervoor te zorgen dat deze gegevens zo betrouwbaar mogelijk zijn, wordt het systeem gecalibreerd op het moment dat deze wordt aangezet. Op deze manier is het duidelijk wat de afwijkingen waren aan het begin van het systeem. Daarnaast wordt er in het systeem gebruik gemaakt van een PID-controller en een lowpass filter, om het systeem zijn eigen berekeningen en uitschietende meetwaarden te laten corrigeren.

Availability

Wanneer de RH4 'Spyder' wordt uitgerust met het camera systeem, mag het camera systeem niet falen. Hiervoor moet geprobeerd worden een availability van 100% te behalen. Hierbij wordt vanuit gegaan dat de gebruiker het systeem op de aangegeven manier gebruikt.

Security

Omdat het systeem niet in verbinding staat met de buitenwereld, hoeft op dit moment nog niet te worden gedacht aan beveiliging van het systeem tegen aanvallen van buitenaf. Wanneer er in de toekomst wel de mogelijkheid wordt geboden om gegevens van en naar het camera systeem te zenden, moet ook aan dit aspect worden gewerkt.

Het enige moment waarop gegevens van buiten het systeem, het systeem binnenkomen, is het moment waarop de waarden van de sensor worden uitgelezen. Hoe deze informatie is beveiligd is eerder uitgelegd onder het kopje Reliability.

Maintainability

De grootste reden waarom het wenselijk is de huidige situatie om te zetten naar een nieuwe versie, is om dat het systeem totaal niet onderhoudbaar is. De onderhoudbaarheid van de code is een van de belangrijkste niet-functionele eisen. Om dit voor elkaar te krijgen moet het systeem overzichtelijk worden ontworpen, wat inhoudt dat er gebruik gemaakt moet worden van klassen en bijbehorende UML diagrammen. Daarnaast moet alle code netjes zijn gedocumenteerd, zodat voor andere ontwikkelaars duidelijk is wat de code precies doet.

Op deze manier is het mogelijk om het huidige systeem aan te passen en te verbeteren, maar het is ook mogelijk om later extra functionaliteit toe te voegen aan het systeem.

Portability

Het huidige systeem draait op de Arduino Mini en moet worden overgezet en geoptimaliseerd voor de Teensy 3.0. Deze Teensy 3.0 zal vervolgens de basis worden van het gehele systeem, waarbij er geen rekening hoeft te worden gehouden met omzetting naar een eventueel nieuw platform in de toekomst.

Bijlage E: Testplan

Testplan

Project:
Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf:
Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

Versiebeheer

Versie	Datum	Geschreven door	Beschrijving
0.1	20 feb 2014	Jeroen de Meij	Eerste opzet van het document. Structuur is opgezet. Hoofdstuk 1 t/m 3.5 ingevuld.
1.0	24 feb 2014	Jeroen de Meij	Hoofdstuk 3.6 t/m 9 ingevuld. Hoofdstuk 10 – risico's verwijderd. (Worden al gedekt in plan van aanpak)
1.1	25 feb 2014	Gerwin Voorsluijs en Jeroen de Meij	Spellingfouten verbeterd en de manier van van het benoemen van testgevallen.

1. Testopdracht

1.1 Inleiding en korte projectbeschrijving

Dit project wordt uitgevoerd in opdracht van Delft Dynamics B.V. Dit bedrijf is in februari 2006 opgericht en heeft zich gespecialiseerd in het bouwen van robothelikopters: kleine onbemande helikopters die door het slim samenvoegen van computer- en sensorapparatuur gebruikt kunnen worden als stabiel en makkelijk te besturen sensorplatform. Een aantal robothelikopters is geleverd aan KLPD en Defensie voor operationele evaluatie.

In 2013 brengt Delft Dynamics haar nieuwste product op de markt: de RH4 'Spyder'. Deze compacte en lichtgewicht multicopter kan verschillende soorten camera's dragen en is zeer makkelijk te bedienen. Op deze wijze is het mogelijk om op een eenvoudige manier vanuit de lucht foto- en videobeelden te maken van de omgeving.

Onder deze multicopter hangt een camera-stabilisatie-systeem. Dit systeem maakt gebruik van een gratis open source framework. De code van het framework is alleen niet uitbreidbaar. Het is een warboel van allemaal functies die te maken hebben met algemeen gebruik van het systeem. Hierdoor is niet meteen duidelijk wat de functie is van alle code. Daarbij wordt ook nauwelijks gebruikt gemaakt van klassen, maar staan alle functies die behoren bij een aspect van het systeem los in een file gezet. Ook zijn er twee files waarin vrijwel alle 'defines' en globale variabelen staan die in het systeem gebruikt worden, zonder dat duidelijk is bij welk onderdeel van het systeem ze thuis horen. De wens om in de toekomst meer functionaliteit aan dit systeem toe te voegen is daarom op dit moment niet mogelijk vanwege de warrige code. Daarnaast is ook het probleem dat de Arduino Mini aan het maximum van zijn kunnen zit en geen extra functionaliteit toebedeeld kan krijgen.

Naast de warrige code komt het af en toe voor dat het systeem vastloopt. Dit gebeurt op het moment dat het systeem meer moet stabiliseren dan fysiek nog mogelijk is. Op deze momenten tikt de camera tegen de onderkant van de helikopter aan en kan dus niet verder stabiliseren. Uit testen is gebleken dat het systeem vast loopt het moment voordat er informatie vanaf de sensor wordt teruggegeven. De opdrachtgever denk dat dit te maken heeft met de I2C communicatie, waarbij op dat moment de master (Arduino Mini) tevergeefs wacht op informatie van de slave (MPU6050).

Doel van het project is om aan het einde de huidige software te hebben overgezet op de Teensy 3.0, waarbij de code gestructureerd is en het duidelijk is wat de verschillende onderdelen in de code doen.

1.2 Opdrachtgever, opdrachtnemer, stakeholders en acceptanten

Dhr. G. M. Voorsluijs is namens Delft Dynamics de opdrachtgever van het project. De opdrachtnemer is Jeroen de Meij, die deze opdracht uitvoert als afstudeer stage.

De overige stakeholders van het project zijn de werknemers van Delft Dynamics. Zij zullen mede bepalen of de kwaliteit van het systeem voldoende is.

1.3 Testopdracht en afbakening

Dit testplan beschrijft de manier waarop de optimalisatie van het camera-stabilisatie-systeem wordt getest.

Het systeem wordt globaal op twee punten getest:

- De stabiliteit van het camera beeld
- De onderhoudbaarheid van het systeem, wat concreet betekend dat duidelijk moet zijn wat de code doet en dat het mogelijk is om nieuwe code toe te voegen.

Hierbij moet worden gekeken of de stabiliteit van het camera beeld minimaal overeenkomt met het huidige systeem. Daarnaast moet worden gekeken of de code die hiervoor is ontwikkeld onderhoudbaar is, want inhoud dat de code goed gedocumenteerd moet zijn en duidelijk voor anderen ontwikkelaars wat de verschillende aspecten van de code doen. Globaal moet dus ook worden gekeken of de ontwikkelde code naar behoren werkt.

Er hoeft tijdens dit testen niet worden gelet op de kwaliteit van de multicopter zelf. Dit valt buiten de scope van het project. Het gaat bij het testen puur en alleen om het camera-stabilisatie-systeem zelf, waarbij niet gelet wordt op de systemen die hier gebruik van maken.

1.4 Uitgangspunten en randvoorwaarden

Om het systeem naar behoren te kunnen testen zijn er een aantal eisen waar aan voldaan moet worden:

- De hardware moet tijdig beschikbaar zijn. Met het huidige systeem worden zo nu en dan nog testvluchten gehouden. Het is van belang dat er een systeem vrij komt op het moment dat deze getest moet worden.
- De opdrachtgever moet voldoende tijd hebben om opgeleverde documentatie te bekijken en feedback op te geven. Op deze manier is duidelijk wat er van het systeem verwacht wordt en kan het systeem ook daadwerkelijk getest worden.

2. Testbasis

Als basis voor het opstellen van dit testplan wordt voornamelijk gebruik gemaakt van het requirements document. In dit requirements document staat precies omschreven wat de eisen zijn aan het systeem, staat aangegeven welke hardware er voor het systeem wordt gebruik en op welke manier deze componenten met elkaar moeten communiceren.

Daarnaast staat in dit document ook vermeld aan welke functionele en niet functionele eisen moet worden voldaan. Dit testplan is bedoelt om een methode uit te zetten om deze eisen te testen.

Naast dit requirements document hebben de gesprekken die zijn gehouden met dhr. Voorsluijs en overige werknemers van Delft Dynamics ook als basis gediend voor dit testplan.

Tot slot is gekeken naar het huidige systeem. Zo is gekeken hoe het systeem in de werkelijkheid wordt toegepast en is de code van het gebruikte framework grondig bestudeerd om een beeld te krijgen van de huidige kwaliteit van het systeem.

Deze testbasis bied voldoende voldoende informatie voor de verdere invulling van het test traject.

3. Teststrategie

3.1 Het testproces

Tijdens het testproces komen de twee eerder genoemde punten aan bod waar op gelet wordt tijdens het testen: de stabiliteit van het camerasysteem en de onderhoudbaarheid ervan. In de product risico matrix is vervolgens aangegeven welke testen er worden gebruikt om de verschillende aspecten van het systeem te kunnen testen.

3.2 Kwaliteitsaspecten

In onderstaande tabel staan de kwaliteitsaspecten aangegeven uit de ISO 9126 norm waar tijdens het testen op wordt gelet:

Kwaliteitsattributen volgens ISO 9126	Relatief belang
Functionaliteit	
Geschiktheid *Mate waarin de gewenste functies aanwezig aanwezig zijn en geschikt om gespecificeerde taken uit te voeren.*	10
Juistheid *Juistheid van de uitvoer van het systeem, overeenkomstig de invoer en de gespecificeerde bewerkingen.*	20
Betrouwbaarheid	
Volwassenheid *Mate waarin fouten en kinderziektes verholpen zijn en het systeem vrij blijft van storingen.*	10
Beschikbaarheid *Mate waarin het systeem op de gewenste tijden beschikbaar is voor de gebruiker.*	5
Tijdbeslag*Responstijd, transactiesnelheid, snelheid batchverwerking.*	10
Middelenbeslag *Hoeveelheid benodigde resources (voornamelijk geheugen).*	5
Overzetbaarheid	
Aanpasbaarheid *Gemak waarmee het systeem overgezet kan worden naar een ander hardware/software-platform of naar een nieuwe versie daarvan (zoals bij toevoeging van extra functionaliteit).*	20
Inpasbaarheid *Het gemakt waarmee het systeem een bestaand systeem kan vervangen. Mate waarin het systeem aansluit bij de bedrijfsprocessen en (handmatige) procedures.*	5

Onderhoudbaarheid	
Analyseerbaarheid *Gemak waarmee de oorzaak van fouten opgespoord kan worden waarmee te wijzigen onderdelen kunnen worden gevonden.*	5
Testbaarheid *Gemak waarmee de juiste werking getest en gevalideerd kan worden.*	5
Beheerbaarheid *Gemak waarmee het systeem in operationele staat gebracht en gehouden kan worden.*	5

Tabel 3.1: De kwaliteitsaspecten uit de ISO 9126 norm die aan bod komen in het systeem met het bijbehorende relatief belang.

3.3 Onderdelen

Aangezien het in dit geval om een klein systeem gaat, zijn er weinig onderdelen te benoemen. Om toch een indeling te kunnen maken in het systeem kan de volgende indeling worden gebruikt:

- Stabiliseren van de camera
- Genereren van (debug) output

3.4 PRIMA: Product Risico Matrix

De onderstaande Product Risico Matrix (PRIMA) geeft aan hoe belangrijk de kwaliteitsaspecten zijn bij de verschillende onderdelen van het systeem, aangegeven is paragraaf 3.3.

	Onderdelen ->	Stabiliseren	Output
Kwaliteitsaspecten	100	90	10
1. Geschiktheid	10	xx	x
2. Juistheid	20	xxx	x
3. Volwassenheid	10	xx	
4. Beschikbaarheid	5	x	
5. Tijdbeslag	10	xx	x
6. Middelenbeslag	5	x	
7. Aanpasbaarheid	20	xxx	
8. Inpasbaarheid	5	x	
9. Analyseerbaarheid	5	x	
10. Testbaarheid	5	x	
11. Beheerbaarheid	5	x	

Tabel 3.2: Product Risico Matrix (PRIMA)

3.5 Acceptatiecriteria

In de onderstaande lijsten zijn de acceptatiecriteria opgenomen die aan bod komen bij de testen. Hiermee wordt dus aangegeven welke aspecten wel met de testen gedekt worden en welke aspecten er niet gedekt worden.

De functionaliteit:

- SENSOR UITLEZEN – De sensor wordt 1000 keer per seconde correct uitgelezen.
- AANSTURING MOTOREN – De motoren zorgen ervoor dat de camera stabiel blijft.
- CONTINUÏTEIT – Het systeem mag niet vastlopen. Dit in tegenstelling tot het huidige systeem.
- GEBRUIKERS GEMAK – Het moet voor de gebruikers eenvoudig en duidelijk zijn hoe het systeem werkt.

Onderhoudbaarheid:

- STRUCTUUR FRAMEWORK – Aan de hand van de structuur van de code, denk bijvoorbeeld aan klassen, moet het duidelijk zijn hoe het framework in elkaar zit.
- DOCUMENTATIE – Aan de hand van de gemaakt documentatie en inline commentaar moet duidelijk zijn wat de specifieke onderdelen uit de code doen.
- UITBREIDBAARHEID – Het moet mogelijk zijn om het framework in de toekomst uit te breiden met extra functionaliteit.
-

Efficiëntie:

- UITVOERINGSDUUR CODE – Er moet tijd vrij zijn om in de toekomst uitbreidingen mogelijk te maken.
- GEBRUIK VAN RESOURCES – Niet het gehele geheugen en processor kracht mogen gebruikt worden om in de toekomst uitbreidingen mogelijk te maken.

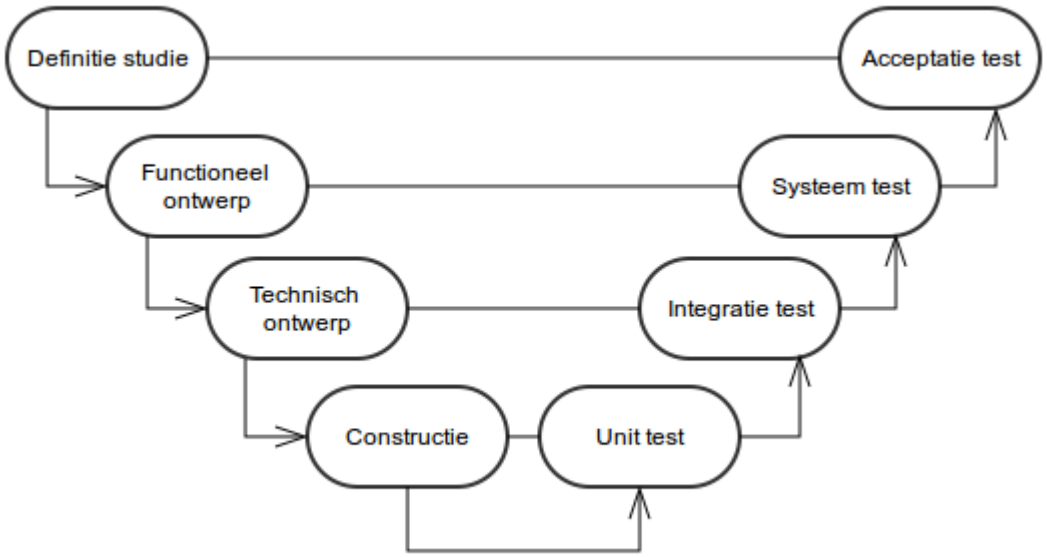
Niet testen

Onderstaande aspecten zullen niet aan bod komen tijdens het testen:

- De werking van de multicopter waaraan het systeem is bevestigd. Dit is een systeem wat los staat van het camera-systeem.
- De aansturing van de camera. Op dit moment wordt alleen de camera-stabilisatie getest. De aansturing van de camera is een eventuele uitbreiding.

3.6 Testsoorten

De gebruikte ontwikkelmethode voor de ontwikkeling van het camera-stabilisatie-systeem is het waterval model. Naar aanleiding van deze gekozen methode is besloten ook voor het testen gebruik te maken van een waterval gerelateerde test methode. Hierbij is de keuze gevallen op het V-model, te zien in afbeelding 3.1. Onder de afbeelding wordt deze methode verder toegelicht.



Afbeelding 3.1: Het V-model

Nadat het systeem is ontworpen en gebouwd wordt overgegaan naar de testfase. De uitzondering hierbij is dat tijdens de constructie fade, waarin de code daadwerkelijk wordt geprogrammeerd, er unittest plaatsvinden om te testen of de kleine stukjes code werken. Hierbij is te denken aan een enkele functie, waarbij bijvoorbeeld de waarden uit de sensor worden opgevraagd. De overige serie teststadia, integratie test, systeem test en acceptatie test worden hieronder verder toegelicht:

- Integratie test – In deze test wordt gekeken of de verschillende componenten samen goed werken. Hoe is de totale koppeling tussen de controller en de sensor en hoe communiceert deze weer met de motoren? Dat zijn testen die in deze fase naar voren komen.
- Systeem test – In deze test wordt gekeken of voldaan is aan de functionele eisen. In dit geval wordt gekeken of de sensor daadwerkelijk 1000 keer per seconden wordt uitgelezen en de berekeningen van de sensor waarden en de motor aansturing goed is. Hierdoor wordt gecontroleerd of de camera stabiel genoeg is om te filmen. Daarnaast wordt gekeken naar de totale duur van de uit te voeren functies. Daarnaast wordt gekeken hoeveel geheugen en processor kracht er wordt gebruikt. Bij deze testen is het camera systeem nog niet aangesloten op de multicopter.
- Acceptatie test – Deze test is onderverdeelt in een viertal sub tests. Deze testen worden uitgevoerd op het moment dat het systeem klaar is om aan de helikopter te bevestigen.
    - FAT (Functionele acceptatie test) – In deze test wordt het systeem aan de multicopter bevestigd en beoordelen de gebruikers (werknemers van Delft Dynamics) of het systeem voldoet aan de eisen.

- GAT (Gebruikers acceptatie test) – In deze test wordt gekeken of het systeem werkbaar is, dus makkelijk in gebruik te nemen is.
- PAT (Productie acceptatie test) – In deze test wordt gekeken of het systeem voor de beheerders makkelijk is te onderhouden.Continuïteit test – Hierbij wordt gekeken of het punt waar het vorige systeem vastliep is verholpen.

4. Testorganisatie

4.1 Rollen, taken en bemensing

Rol	Taken	Persoon
Opdrachtgever	• Goedkeuring testplan • Voortgangsbewaking • Uiteindelijke goedkeuring systeem	Gerwin Voorsluijs
Opdrachtnemer en test uitvoerder	• Opstellen testplan • Ontwikkelen testen • Testen uitvoeren/begeleiden • Documenteren van de resultatenRapporteren aan de opdrachtgever	Jeroen de Meij
Testgroep	• Deelnemen aan de verschillende acceptatietesten	Verschillende werknemers van Delft Dynamics

Tabel 4.1 De verschillende rollen die naar voren komen bij de testuitvoering, inclusief de bijbehorende taken en de persoon die deze rollen belkeed.

4.2 Communicatie

Iedere week is er op woensdag een overleg moment met de opdrachtgever. In deze momenten wordt de huidige status van het project besproken, inclusief de voortgang van de testprocedure. Tijdens deze besprekingen worden onder andere de documenten behandeld die in die week zijn gemaakt. Tussendoor wordt er feedback gegeven op de documenten die worden opgestuurd, waarbij wordt gecontroleerd of wat in de documenten staat correct is en voldoende.

4.3 Dechargeprocedure

De opdrachtgever is uiteindelijk de persoon die bepaald of het systeem naar wens functioneert en voldoende is getest.

De decharge-criteria zijn:

- Alle acceptatiecriteria zijn gedekt door daadwerkelijk uitgevoerde testen
- Eindrapport test beschikbaar
- De testen zijn naar behoren uitgevoerd
- De gevonden problemen zijn verholpen of afdoende gedocumenteerd

5. Testproducten

5.1 Projectdocumentatie

Deze documenten worden alleen gebruikt tijdens de uitvoering van de testprocedure:

- Testplan
- Test planning

5.2 Testware

Dit zijn de producten die worden opgeleverd tijdens of aan het einde van de testprocedure en bewaard blijven voor in de toekomst:

- Testscenario's met eventueel testgevallen
- Testscripts
- Bestand met testresultaten

5.3 Testdatabases en -bestanden

Voor de uitvoering van de testen zijn geen extra databases of bestanden nodig die aangeleverd moeten worden.

5.4 Tussentijdse rapportage

Tussentijdse rapportage gebeurt tijdens de wekelijkse bijeenkomst. Vooraf aan deze bijeenkomsten wordt documentatie opgeleverd die in de week voor de bijeenkomst is gemaakt of verbeterd. Tijdens de bijeenkomsten worden gekeken of de documenten correct zijn en of er voldoende inhoud in staat.

5.5 Eindrapportage

In de eindrapportage wordt samengevat welk proces is doorlopen tijdens het testen, waarbij wordt aangegeven welke documenten er zijn gemaakt, welke testen er zijn uitgevoerd en wat gedaan is met de verbeterpunten die tijdens deze tests naar voren kwamen en wat de uiteindelijke status is van het systeem.

6. Testinfrastructuur

6.1 Testomgeving

Voor het uitvoeren van de testen is geen speciale testomgeving beschikbaar. Het systeem hoeft niet te communiceren met andere systemen.

Wel kan het systeem via USB aan de computer worden gekoppeld om op deze manier debug messages te printen.

6.2 Test tools

Voor het uitvoeren van de testen worden geen speciale test tools gebruikt. Wel kan er gebruik gemaakt worden van speciale libraries die voor de Teensy zijn geschreven.

Zo kan door middel van de klassen elapsedMillis en elapsedMicros worden bepaald hoeveel tijd er wordt gedaan over een stuk code.

Daarnaast is er in in het huidige systeem al een stukje code gemaakt wat test hoeveel geheugen er nog vrij is.

6.3 Autorisaties

Voor het uitvoeren van de testen zijn geen extra autorisaties nodig. De test uitvoerder heeft root rechten op de ontwikkel pc, waar hij ook de testen op uitvoert.

7. Beheer

7.1 Opslaglocatie en conventies

Op de computer waar het systeem wordt ontwikkeld is een map met documentatie te vinden. In deze map wordt de map 'testware' aangemaakt. In deze map is alle documentatie te vinden die te maken heeft met het testen van het systeem.

De scenario's worden aangegeven met de eerste letter van de testsoort, gevolgd door de eerste letter van de sub testsoort gevolgd door een nummer. Op deze manier is ieder scenario uniek genummerd en terug te vinden. Vervolgens kan het in een scenario voorkomen dat er meerdere testgevallen voorkomen. Deze testgevallen worden ook genummerd. Hieronder volgt een tabel met afkortingen en daarbij aangegeven om welke testsoort het gaat.

AF	Acceptatie test – FAT
AG	Acceptatie test – GAT
AP	Acceptatie test – PAT
AC	Acceptatie test – Continuïteit test
S	Systeem test
I	Integratie test

Tabel 7.1: Testsoort afkortingen

Zo is de test met de code 'AP3-3': Acceptatie test, Productie acceptatie test, nummer 3, testgeval vijf.

Daarnaast worden de eventuele geautomatiseerde testscripts opgeslagen in de map testscripts bij de rest van de source-code van het systeem.

7.2 Bevindingenbeheer

De bevindingen worden in twee bestanden bijgehouden. In een spreadsheet wordt aangegeven wat de huidige status is van een test, waarbij de test zijn gesorteerd op testsoort en code. Hierbij wordt genoteerd: De testcode; Laatste datum van uitvoering; Laatste status.

De laatste status kan een van de volgende twee waarden hebben:

- Gelukt – Het resultaat komt overeen met het te verwachten resultaat.
- Mislukt – Het resultaat wijkt af van het te verwachten resultaat.
- Niet uitgevoerd – De test is nog nooit uitgevoerd.

Daarnaast wordt in een ander document per test bijgehouden op welke momenten deze is uitgevoerd. Hierbij wordt bij een mislukte test ook genoteerd wat het resultaat was van de test en wanneer bekent is al waarom de test mislukte. Op deze manier wordt er een overzicht gemaakt welke testen er zijn uitgevoerd.

7.2 Overdracht testware

De testware wordt, samen met het systeem, digitaal overgedragen aan de opdrachtgever, Gerwin Voorsluijs. Daarbij kan hij, in nader overleg, aangeven om een aantal documenten geprint te willen ontvangen.

8. Testplanning en begroting

Activiteit	Op te leveren product	Uren	Deadline
Voorbereiding en specificatie	Testplan	16	27 februari 2014
	Acceptatie test scenario's	16	6 maart 2014
	Systeem test scenario's	8	6 maart 2014
	Integratie test scenario's	8	27 maart 2014
Testuitvoering	Spreadsheet testresultaten; Uitgebreid document met testgegevens	24	8 mei 2014
Schrijven testrapport	Testrapport	16	8 mei 2014

Tabel 8.1: De verschillende activiteiten van het etsen, met bijbehorende producten, aantal geplande uren en deadline

9. Kwaliteitsbewaking testproces

Elk product wordt naar de opdrachtgever gestuurd waarna hij deze controleert en feedback op geeft. Daarnaast vindt iedere woensdag een bespreking plaats over de afgelopen week waarbij ter sprake komt wat er die week is uitgevoerd.

Bijlage F: Installatie handleiding gcc-compiler

Installatie handleiding gcc-compiler

Jeroen de Meij

Te-Aar, donderdag 8 mei 2014

Het installeren van de compiler

Het project is uitgevoerd op een 32 bits versie van Ubuntu. Deze handleiding is geschreven vanuit dit perspectief. Let hiervoor op, wanneer er op een ander OS wordt gewerkt.

Om een framework voor de Teensy 3.0 te kunnen ontwikkelen en deze vervolgens op de Teensy 3.0 te kunnen plaatsen, moet de gcc compiler worden geïnstalleerd. Op de webpagina http://www.pjrc.com/teensy/first_use.html staat aangegeven welke stappen er gevolgd moeten worden om software voor de Teensy 3.0 te kunnen ontwikkelen, compileren en op de Teensy te kunnen zetten.

STAP 1: Installeren van de Arduino software

Via de site http://www.arduino.cc/en/Main/Software#.UwXDhrOm3RE is het mogelijk de juiste versie van de Arduino IDE te downloaden (Linux: 32 bit). Op het moment van schrijven moet gekozen worden voor de versie 1.0.5. In een volgende stap zal namelijk Teensyduino worden geïnstalleerd en de huidige versie van Teensyduino (versie 1.18) is alleen compatible met versie 1.0.5 van de Arduino IDE.

Op dit moment heeft u een gecomprimeerde map gedownload. Pak deze map uit en onthoud goed waar u deze map heeft uitgepakt!

Opmerking:

De Teensy 3.0 is uitgerust met een ARM processor en kan gebruik maken van een aantal Arduino libaries. In tegenstelling tot de Teensy zijn Arduino's niet gebaseerd op een ARM processor, maar op een AVR processor. Hierdoor zijn niet alle Arduino libraries te gebruiken op de Teensy 3.0

STAP 2: Instellen van de Linux udev regels

Via de site http://www.pjrc.com/teensy/49-teensy.rules kan een bestand worden gedownload met Linux commando's. Voer dit bestand uit via het commando:

```
sudo cp 49-teensy.rules /etc/udev/rules.d/
```

De commando's in dit bestand geven niet-root gebruikers van een Linux OS het recht om ook de Teensy 3.0 te gebruiken.

STAP 3: Installeren van Teensyduino

Via de site http://www.pjrc.com/teensy/td_download.html kan de juiste versie van Teensyduino worden gedownload: Linux installer (32 bit). Wanneer dit bestand wordt uitgevoerd, wordt er een installer geopend. Op een gegeven moment vraagt de installer naar de map waarin de Arduino IDE is uitgepakt (stap 1). Navigeer naar deze map.

Gebruiken van de compiler

Na de installatie van de compiler is diep in de Arduino IDE map een makefile weggestopt. Navigeer naar:

[Map waar de Arduino IDE is uitgepakt]/arduino-1.0.5/hardware/teensy/cores/teensy3

Deze Makefile zorgt ervoor dat de bestanden in die map worden gecompileerd en dat de Teensy loader wordt opgestart, waardoor het project meteen op de Teensy kan worden gezet.

Aanmaken van een eigen project directory

Wanneer het wenselijk is om een nieuwe map aan te maken voor een project, moet de makefile naar deze map worden gecopiëerd. Zorg ervoor dat in deze makefile de verwijzingen naar de teensy bestanden (regel 19), de arduino libraries (regel 23) en de arm compiler tools (regel 26) zijn aangepast!

Bijlage G: Technisch ontwerp

Technisch ontwerp

Project:
Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf:
Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

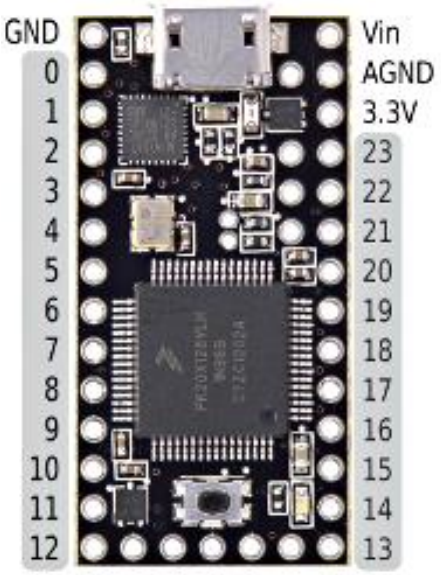
Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

Inleiding

In dit document wordt een technisch ontwerp van het nieuwe camera-stabilisatie-systeem beschreven. Zo wordt allereerst gekeken naar de pinnen waarop de verschillende hardware componenten zijn aangesloten. Vervolgens wordt het klassendiagram van het systeem getoond, zoals het in eerste instantie wordt overgezet. Daarna volgt een globale beschrijving van de werking van het systeem. Tot slot van het document is er een hoofdstuk opgenomen over de tijdgaranties van het ontwerp.

1. Pin aansluitingen

In onderstaande tabel staat per pin van de Teensy 3.0 aangegeven of er een extern apparaat op zit aangesloten:

GND – MPU	GND		Vin	
	0		AGND	
	1		3.3V	3.3V – MPU
	2		23	Pitch motor – C
	3		22	Pitch motor – B
	4		21	Pitch motor – A
Roll motor – A	5		20	
Roll motor – B	6		19	
	7		18	
	8		17	
Roll motor – C	9		16	
/CS – MPU	10		15	
SDI – MPU	11		14	
SDO – MPU	12		13	SCLK – MPU

Er is voor gekozen om de SPI communicatie via de daarvoor bedoelde pinnen te laten lopen, ook al wordt er geen gebruikt gemaakt van de standaard SPI library van de Teensy, maar van de library die ingebouwd zit in de MPU6000 library. Hiervoor is gekozen om op die manier de mogelijkheid te bieden deze library in de toekomst wel makkelijk te gebruiken.

Daarnaast is gekeken op welke pinnen de motoren worden aangesloten. Een motor heeft drie verschillende pwm signalen nodig voor de aansturing, dus in totaal zijn er zes poorten nodig die pwm signalen ondersteunen. In het basis ontwerp is al gekeken naar de mogelijkheden om de pwm signalen aan te passen op frequentie en resolutie. Bij dat onderzoek kwam naar voren dat de Teensy 2 timers gebruikt voor de pwm aansturing. Als op een pin de frequentie of resolutie wordt aangepast, geldt deze verandering voor alle pinnen die op dezelfde timer zijn aangesloten.

In onderstaande tabel is een overzicht te zien welke pinnen er op een timer zitten aangesloten.

Timer	pwm pinnen	Standaard frequentie
0	5, 6, 9, 10, 20, 21, 22, 23	488.28 Hz
1	3, 4	488.28 Hz

Er is geprobeerd om alle motoren op timer 0 aan te sluiten, omdat op deze manier timer 1 vrij blijft voor een eventuele andere pwm configuratie. Daarom is besloten de pinnen 5, 6 en 9 te gebruiken voor de roll motor en de pinnen 21, 22 en 23 voor de pitch motor.

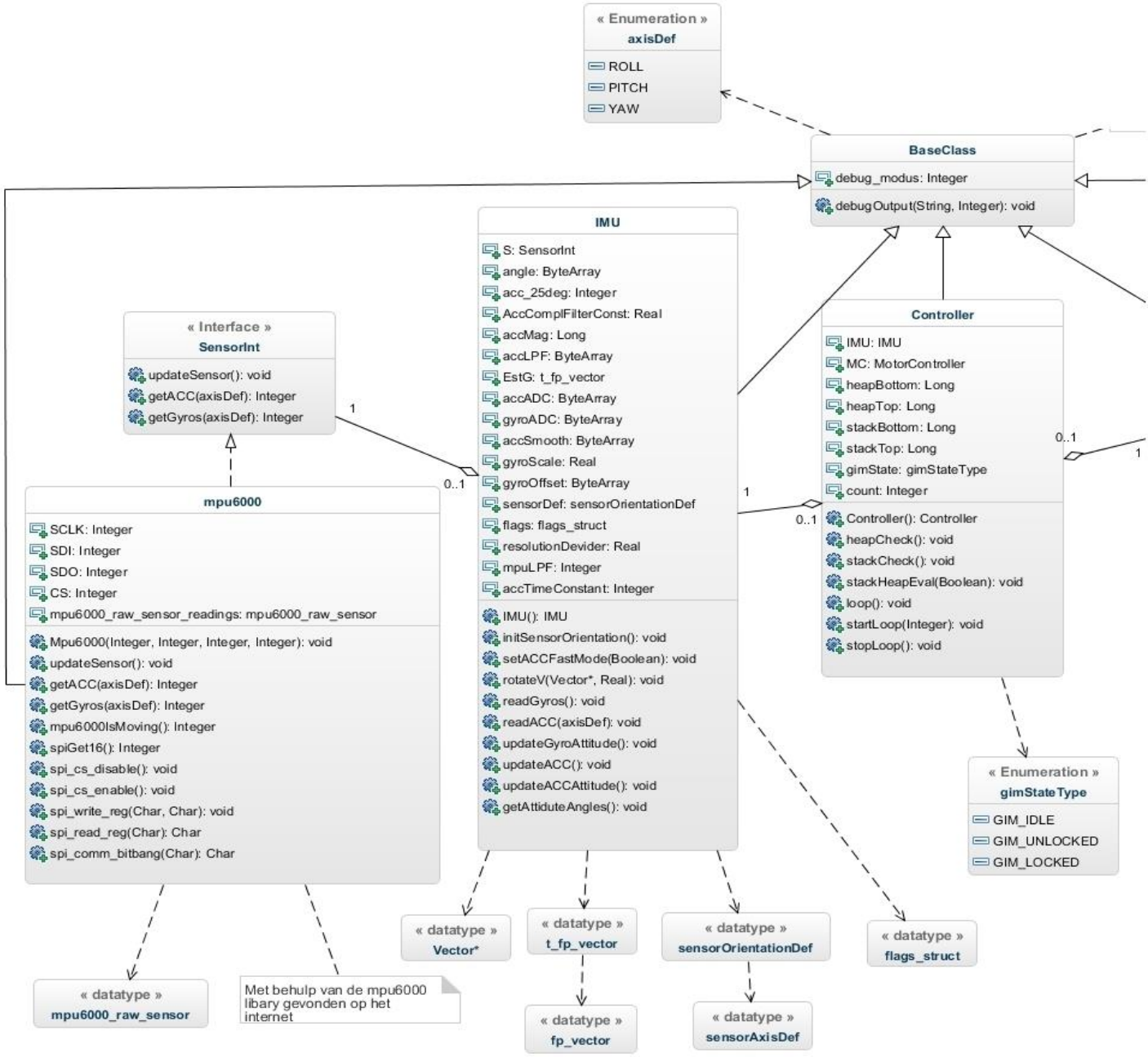
2. Gedetailleerd klassendiagram

Bij dit onderzoek is het huidige framework onderverdeeld in een aantal klassen. Dit is een vrijwel letterlijke overname van het huidige systeem, waarbij nog niet is gekeken naar het anders moduleren van bepaalde functies. Een voorbeeld hiervan is dat het gebruik van een enumeratie voor states letterlijk is overgenomen, zonder gebruik te maken van een state design pattern. Nadat het systeem op deze manier netjes is omgezet kan in een volgende stap worden bekeken op welke punten het kan worden verbeterd, dan wel door de code te optimaliseren, dan wel door de structuur anders te maken.

In dit document is het diagram opgedeeld in twee stukken, omdat het anders te groot werd om te kunnen tonen.

Sensor gedeelte

Hieronder volgt het diagram voor het sensor gedeelte. Op de volgende pagina wordt verdere toelichting gegeven.



Zoals al is beschreven in het basis ontwerp, is de Controller het regelcentrum van het systeem. De Controller zorgt ervoor dat alle hardware componenten worden geïnitialiseerd en bevat een loop functie die om de zoveel tijd (standaard om de milliseconden) wordt aangeroepen en de hardware componenten aanstuurt. Daarnaast bevat de Controller een aantal functies met debug mogelijkheden, zoals het checken of er nog voldoende geheugen over is. Tot slot beschikt de Controller over een enumeratie die een aantal States bevat, waar de Controller zich in kan bevinden. Deze States worden bij het initialiseren van het systeem gebruikt om onder andere de PIDcontroller de mogelijkheid te geven zichzelf te configureren en stabiliseren.

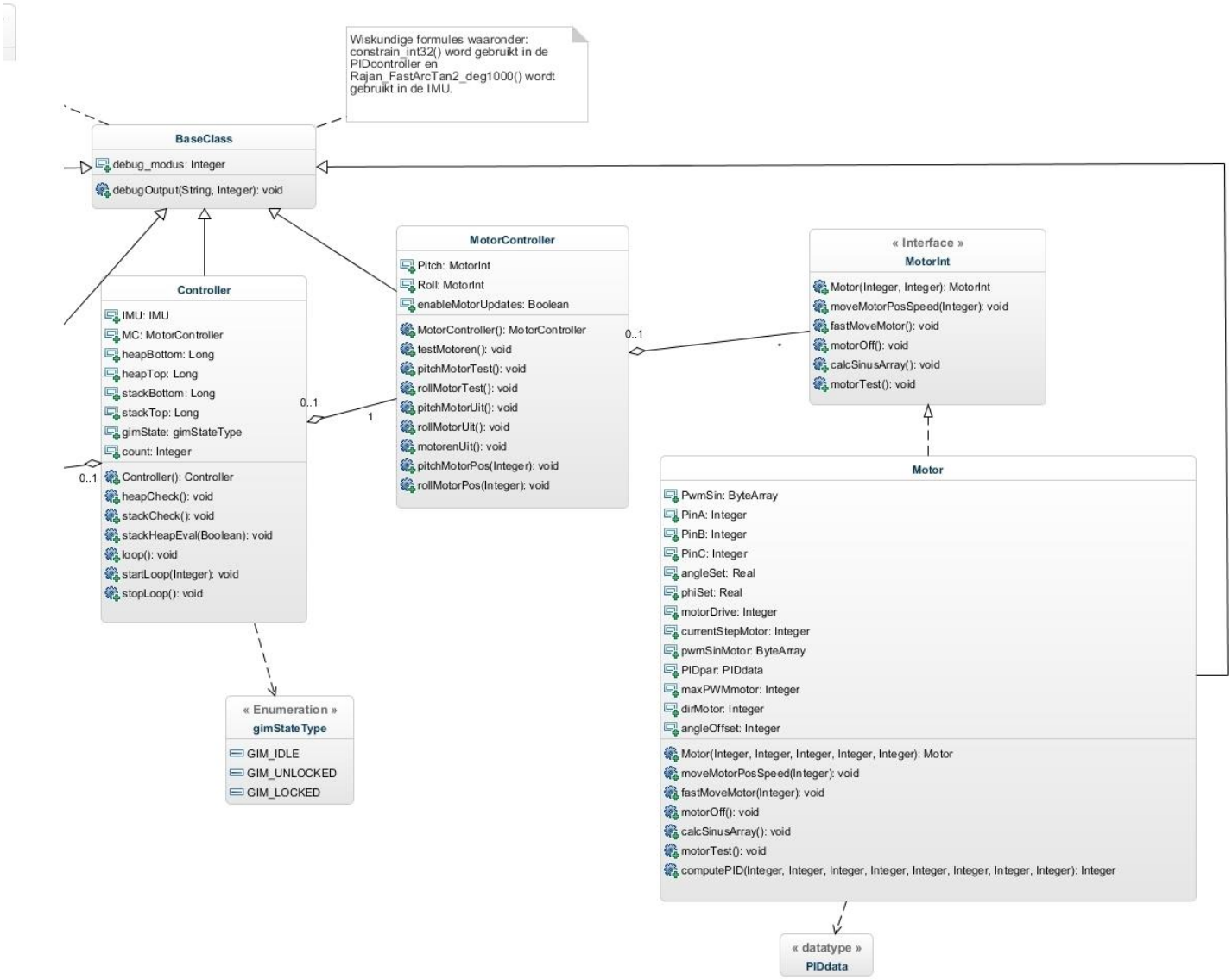
De Controller is afgeleid van de BaseClass. Deze klasse is de 'hoofdklasse' van het gehele systeem. Alle overige klassen zijn van deze BaseClass afgeleid. De BaseClass zorgt voor een aantal functies die alle overige klassen in het systeem moeten hebben, zoals de mogelijkheid om debug berichten te kunnen tonen. Daarnaast bevat de BaseClass een enumeratie met de drie verschillende draaiingen die de helikopter kan maken (roll, ptich en yaw). Deze enumeratie wordt in de rest van het systeem ook gebruikt.

De Controller bevat altijd één instantie van een IMU. Deze IMU zorgt voor de communicatie met de daadwerkelijke sensor. Zo bevat de IMU een functie om de sensor te initialiseren en gegevens van de sensor op te vragen. Daarnaast bevat de IMU een aantal functies om met de gegevens van de sensor de draaiing van het systeem te bepalen. De IMU maakt gebruik van een aantal structs om de gegevens te ordenen. Deze structs zijn niet als dusdanig weer te geven in het diagram, door beperkingen in de modulatie software, maar ze zijn weergegeven als *<< datatype >>*.

De IMU heeft altijd één verwijzing naar een daadwerkelijke sensor. Deze verwijzing loopt via een sensor interface, om op deze manier de mogelijkheid te bieden om in de toekomst de sensor makkelijk met een andere sensor te verwisselen of extra sensor toe te voegen. Deze klasse zorgt voor de daadwerkelijke communicatie met de sensor. In dit geval is alleen de klasse gemaakt voor de MPU6000, aangezien dit de sensor is die in het systeem gebruikt gaat worden. Deze klasse biedt de mogelijk om via SPI in een keer alle gegevens uit te lezen uit de sensor en deze in het object op te slaan. Dit is sneller dan wanneer de sensor iedere keer opnieuw een variabele uit de sensor moet lezen. De MPU6000 leest zelf met een snelheid van 1 kHz de waarden uit en plaatst ze in zijn eigen register, waarna de mpu6000 klasse alleen nog maar de juiste registers hoeft uit te lezen. Voor deze klasse is gebruik gemaakt van een library die op het internet is gevonden en is omgebouwd tot gebruik voor dit systeem.
(http://code.google.com/p/gluonpilot/source/browse/trunk/Firmware/lib/mpu6000/?r=146 laatst geraadpleegd: 5 maart 2014)

Motor gedeelte

Hieronder volgt het diagram voor het motor gedeelte. Op de volgende pagina volgt verdere toelichting.



Naast dat de Controller een instantie van de IMU bevat, bevat de Controller ook een instantie van een MotorController. Deze MotorController bevat op zijn beurt weer altijd twee instanties van Motoren: een roll motor en een pitch motor. Via de MotorController is het mogelijk om de motoren aan te maken en te configureren. Daarnaast kunnen de twee motoren worden bediend, zoals het uitzetten van de motoren of de motoren naar een bepaalde positie bewegen. Tot slot kunnen de motoren worden getest.

Zoals al is aangegeven, bevat de MotorController altijd twee Motor instanties. Deze verwijzing loopt, vanwege dezelfde reden als bij de sensor, via een interface. Op deze manier kunnen ook andere type motoren makkelijk aan het systeem worden toegevoegd. De Motor klasse bevat de mogelijkheid om de motoren in een bepaalde stand te zetten. Hierbij wordt gebruik gemaakt van een PIDcontroller. Deze PIDcontroller maakt gebruik van een struct met variabelen die in het diagram als *<< datatype >>* PIDdata staat aangegeven. Daarnaast geeft de motor klasse ook de mogelijkheid om de motor uit te zetten of te testen.

3. Uitleg globale werking nieuwe systeem

Globaal is het nieuwe systeem op te delen in twee verschillende momenten. Het eerste is het moment waarop het systeem opstart en zichzelf gereed maakt voor gebruik. Het tweede moment is wanneer het systeem actief is en door middel van een timer de sensor uitleest en de motoren aanstuurt. Hieronder staat per moment uitgelegd wat het systeem voor handelingen gaat uitvoeren.

Opstarten

Bij het opstarten van het systeem wordt eerste de verbinding aangelegd met de IMU en de sensor. Van deze objecten worden alle gegevens goed gezet, zoals het bepalen van de offset, low pass filter et cetera. Wanneer dit is gedaan wordt hetzelfde gedaan met de motoren, waarbij onder andere de juiste frequentie en resolutie worden gezet.

Timer

Wanneer de timer afgaat wordt de loop() functie van de Controller aangeroepen.

Bij het initialiseren van het systeem bevindt het susteem zich in de IDLE state. In deze staat zorgt de loop() functie ervoor de de PID controller wordt geconfigureerd.

Daarna komst het systeem in de UNLOCKED state en worden de motoren goed gezet.

Tot slot komt het systeem in de LCOKED state en houdt het zich bezig met het stabiliseren van de camera. Op dat moment bevat de loop() functie eigenlijk twee stappen. Als eerste stap worden via de IMU alle gegevens uit de sensor uitgelezen, waarna de IMU deze gegevens omzet naar hoekberekeningen en dergelijke. In de tweede stap worden deze hoekberekeningen doorgestuurd naar de motoren die daardoor in de juiste stand komen te staan.

4. Real-time aspect

Aangezien het bij het ontwikkelen van dit systeem gaat om een real-time systeem, is er ook gekeken naar bepaalde tijdgaranties die het systeem moet nakomen en de vormgeving van het systeem, lettend op deze real-time aspecten.

Op dit moment kent het systeem maar één hoofdtaak: het stabiliseren van een camera. Hierdoor is het dus niet nodig om een geheel embedded operating system op de Teensy te installeren. De processen die door de Teensy worden uitgevoerd volgen elkaar in chronologische volgorde op en herhalen zichzelf daarna weer. Hierdoor komt het niet voor dat verschillende processen gescheduled hoeven te worden en kan het systeem zonder de hulp van een real-time operating system worden ontwikkeld.

Wel moet er gelet worden op de duur van bepaalde processen. Het moet namelijk wel mogelijk blijven om met een snelheid van 1 kH de sensor uit te kunnen lezen en de gegevens te bewerken voor de motor aansturing. Dit houdt in dat deze loop in totaal maar 1 milliseconde mag duren. (1 seconde / 1000 keer uitlezen = 1 milliseconde) Om dit voor elkaar te krijgen moet gekeken worden naar een aantal bottlenecks.

De eerste bottleneck is het verkrijgen van de informatie uit de sensor. Om deze tijd zo klein mogelijk te maken wordt alle informatie uit de sensor in een keer uitgelezen en in het sensor object opgeslagen. Dit is sneller dan ieder gegeven een voor een uitlezen. Wanneer er namelijk een gegeven uit de MPU moet worden uitgelezen moet eerst het register adres naar de MPU worden geschreven. Wanneer dit is gebeurd moet een byte met de waarde 0 naar de MPU worden geschreven. Op dat moment zal de MPU de waarde uit dat register terugsturen. Wanneer vervolgens nog een 0 wordt weggeschreven, stuurt de MPU automatisch de waarde in het volgende register mee terug. Aangezien alle registers die uitgelezen moeten worden na elkaar liggen, is het sneller om alle registers in een keer uit te lezen.

Een tweede bottleneck is het uitvoeren van complexe of veel voorkomende berekeningen. Hier een daar is in het framework geprobeerd om deze berekeningen al van te voren uit te voeren. Zo wordt bij het aanmaken van het framework een arry gemaakt, die wordt gevuld met arcsinus berekeningen. Op deze manier hoeft er tijdens het uitvoeren van het programma niet telkens de arcsinus te worden uit gerekend van een bepaalde waarde, maar kan de juiste waarde uit een array worden gehaald. Daarnaast wordt in het standaard framewrok gebruik gemaakt van inline functies. Deze korte functies, die vaak veel worden aangeroepen, worden tijdens het compileren letterlijk op de positie geplakt waar ze gebruikt moeten worden. Dit bespaart weer de tijd die anders verloren zou gaan om naar die plek in het geheugen te springen. Al met al moet tijdens het omzetten worden gekeken in hoe verre deze strategieën toe te passen zijn, om ervoor te zorgen dat er zo min mogelijk tijd verloren gaat.

Bijlage H: Ontwerp variabelen aanpassen

Ontwerp van de uitbreiding: variabelen aanpassen

Project: Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf: Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

1. Requirements

Dit hoofdstuk vormt een aanvulling op het requirements document van het camera-stabilisatie-systeem. In dit hoofdstuk worden de specifieke eisen opgesteld voor dit subonderdeel van het systeem. De algemene systeem eisen die in het grote requirements document staan gelden ook voor dit subsysteem.

Doel van het systeem

Met dit subsysteem moet het mogelijk zijn om een aantal variabelen in het stabilisatie systeem aan te passen. De variabelen die aan te passen zijn, bepalen voor een groot gedeelte de werking van het programma. Op deze manier wordt het eenvoudiger om het framework te kunnen fine-tunen.

Wanneer deze variabelen zijn aangepast moeten ze op de Teensy worden opgeslagen. Op deze manier blijven de veranderingen behouden op het moment dat de Teensy uit wordt gezet.

Interface

Voor het aanpassen van de variabelen wordt een tweede programma geschreven. Iedere keer als dit programma op de Teensy wordt gezet schrijft dit programma de variabelen weg in de EEPROM. Er is voor gekozen om geen gebruikers interface in het stabilisatie systeem in te bouwen, omdat dit meer werk op levert en niet perse noodzakelijk is, aangezien de gebruikers zelf beschikken over voldoende technische kennis.

Functionele eisen

- Op het moment dat het programma wordt geüpload om variabelen aan te passen, moeten deze variabelen worden opgeslagen in de EEPROM van de Teensy.
- Wanneer de gebruiker het stabilisatie programma upload, moeten de variabelen uit de EEPROM worden gelezen.

Niet functionele eisen

- Het systeem moet samen werken met het huidige camera-stabilisatie-systeem.
- Het moet duidelijk zijn hoe de variabelen aan te passen zijn.

Naast deze eisen zijn de algemene systeemeisen die in het hoofd requirements document zijn opgesteld ook op dit systeem van toepassing.

2. Basis/technisch ontwerp

In dit hoofdstuk wordt er een ontwerp gemaakt voor het sub systeem. Hierbij wordt gekeken naar de specificaties van de Teensy, de mogelijke libraries die gebruikt kunnen worden, de samenwerking met de rest van het programma en hoe het klassen diagram eruit komt te zien.

Specificaties

Voor dit systeem is het belangrijk dat er voldoende EEPROM is om de variabelen in op te slaan. In de specificaties van de Teensy 3.0 staat aangegeven dat de Teensy beschikt over 2kb aan EEPROM. Dit is vier keer zoveel als de huidige Arduino, die beschikt over 512 bytes aan EEPORM.

Mogelijke libraries

Voor dit systeem kan gebruik gemaakt worden van de volgende library:

- De EEPROM library (http://pjrc.com/teensy/td_libs_EEPROM.html)

Deze library wordt door het subsysteem gebruikt om de variabelen op te slaan in de EEPROM en door het stabilisatie systeem om deze variabelen te kunnen uitlezen. Daarnaast bevatte het oude systeem al een mogelijkheid om op een dergelijke manier een aantal variabelen te kunnen aanpassen. De code hiervoor zal worden bekeken en waar mogelijk overgenomen.

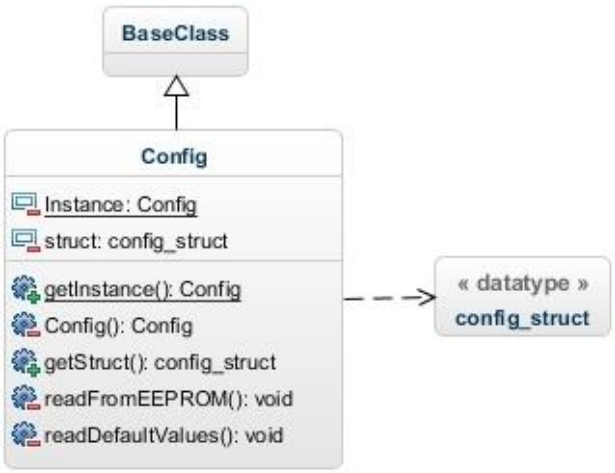
Samenwerking met het grote systeem

Op het moment dat het systeem wordt aangezet, wordt er gekeken naar het versie nummer van de variabelen die in de EEPROM zijn opgeslagen. Wanneer het versie nummer die in de code is opgeslagen verschilt met het versienummer uit de EEPROM, worden de standaard waarden gebruikt zoals ze zijn opgeslagen in de code. Op deze manier is het mogelijk om in de toekomst variabelen aan het systeem toe te voegen of te verwijderen, zonder dat het programma hier op vast kan lopen.

In het grote systeem zal de Config klasse worden uitgebreid om de mogelijkheid te bieden variabelen uit de EEPROM uit te lezen. Het klassen diagram hiervan is te vinden in de volgende paragraaf. Daarnaast wordt een tweede programma geschreven om de variabelen in de EEPROM te kunnen wegschrijven.

Klassendiagram

In onderstaande afbeelding is het klasse diagram te zien voor de Config klasse. Dit is de klasse die zorg gaat dragen voor de aan te passen variabelen. Onder de afbeelding wordt deze uitgelegd.



De Config klasse wordt een Singleton, waardoor er in het gehele framework maar een van kan bestaan. Alle communicatie via Config object moet verlopen via de static klasse getInstance(). Deze functie maakt een Config object aan wanneer deze nog niet bestaat en geeft de pointer

De klasse maakt gebruik van een struct waarin alle aanpasbare variabelen in zijn opgenomen. De functie getStruct geeft de pointer naar deze struct mee, zodat andere objecten in het framework gebruik kunnen maken van deze variabelen.

Wanneer het Config object wordt aangemaakt, worden via de functie readFromEEPROM, de waarden die in de struct staan uit de EEPROM gelezen . In deze functie wordt gebruik gemaakt van de EEPROM library. Wanneer het versie nummer van de struct uit de code niet overeenkomst met die uit de EEPROM, wordt de functie readDefaultValues aangeroepen. In deze variabelen worden de standaard waarden voor de variabelen uit de struct toegekend.

3. Testen

Dit hoofdstuk vormt een aanvulling op het testplan. De punten die in het testplan naar voren komen gelden ook voor deze uitbreiding. In dit hoofdstuk worden daarom ook alleen de aspecten behandeld waarin het testen van het subsysteem verschilt met het testen van het stabilisatie systeem of hier een aanvulling op geeft.

Testopdracht

Als extra aanvulling op de testopdracht die al is beschreven in het testplan, moet ook deze uitbreiding worden getest. Hierbij wordt getest of het systeem naar behoren werkt en wordt gekeken of het voor de beheerders duidelijk is hoe het systeem werkt en moet worden onderhouden.

Alle benodigdheden voor het testen van het systeem zijn al aanwezig.

Teststrategie

Hieronder is de uitgebreide versie van de PRoduct RIsico Matrix te zien. In deze uitbreiding is een extra tabel toegevoegd voor het subsysteem.

PRIMA	Onderdelen ->	Stabiliseren	Variabelen	Output
Kwaliteitsaspecten	100	65	15	15
1. Geschiktheid	10	xx	x	x
2. Juistheid	20	xxx	xx	x
3. Volwassenheid	10	xx	x	
4. Beschikbaarheid	5	x	x	
5. Tijdbeslag	10	xx	x	x
6. Middelenbeslag	5	x	x	
7. Aanpasbaarheid	20	xxx	xx	
8. Inpasbaarheid	5	x	x	
9. Analyseerbaarheid	5	x	x	
10. Testbaarheid	5	x	x	
11. Beheerbaarheid	5	x	x	

Als extra acceptatie criteria is erbij gekomen:

- VARIABELEN AANPASSEN – De gebruikers moeten in staat zijn om de variabelen

van het stabilisatie systeem aan te passen.Aan de hand van deze extra test eisen, zijn er een aantal testen ontwikkeld die zijn opgenomen in de testscenario's.

Bijlage I: Ontwerp pitch hoek aansturen

Ontwerp van de uitbreiding: Hoeken aanpassen

Project: Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf: Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

1. Requirements

Dit hoofdstuk vormt een aanvulling op het requirements document van het camera-stabilisatie-systeem. In dit hoofdstuk worden de specifieke eisen opgesteld voor deze systeem uitbreiding. De algemene systeem eisen die in het grote requirements document staan gelden ook voor dit subsysteem.

Doel van het systeem

Met deze uitbreiding moet de gebruiker de mogelijkheid krijgen om de hoek van de pitch motor aan te passen. Dit komt er op neer dat de gebruiker kan aangeven of de camera in een hoek van 45° omhoog moet kijken, tot een hoek van -135° omlaag moet kijken.

Interface

De gegevens voor deze mogelijkheid worden vanuit de avionica van de spyder, via een seriele verbinding, doorgestuurd naar de Teensy. Deze gegevens bestaan uit 12 bits, wat inhoud dat er waarden van 0 tot en met 4095 ($12^2 - 1$) kunnen worden doorgestuurd. Deze waarden worden doorgestuurd in twee bytes, met daarin de 6 LSB met informatie. De 6 MSB bits van de 12, worden aangeduid door de voorste twee bits van de byte 1 te maken. Op deze manier kunnen de 6 MSB en 6 LSB worden onderscheiden.

Deze waarden komen overeen met een bereik van 180° (45 + 135), wat betekend dat de resolutie van deze gegevens 0.04°/bit is. Wanneer er een 0 wordt doorgestuurd, moet de camera op +45° worden gestabiliseerd en wanneer er 4095 wordt doorgestuurd, moet de camera op -135° worden gestabiliseerd.

Functionele eisen

- De gegeven moet met een twee signaals- seriële verbinding (RX en TX) worden doorgestuurd met een baudrate van 115200.
- De waarden, die variëren van 0 tot en met 4095, moeten worden omgezet is een hoek die loopt tussen de +45° en –135° waarop pitch as stabiliseert.

Niet functionele eisen

- De bautrate van de seriele verbinding moet 115200 zijn.

Naast deze eisen zijn de algemene systeemeisen die in het hoofd requirements document zijn opgesteld ook op dit systeem van toepassing.

2. Basis/technisch ontwerp

In dit hoofdstuk wordt er een ontwerp gemaakt voor de uitbreiding. Hierbij wordt gekeken naar de specificaties van de Teensy, de mogelijke libraries die gebruikt kunnen worden, de samenwerking met de rest van het programma en hoe het klassen diagram eruit komt te zien.

Specificaties

Voor dit systeem is het belangrijk dat de Teensy beschikt over een vrije seriële poort. Op dit moment zijn alle drie de seriële verbindingen nog vrij. Deze poorten ondersteunen allemaal een baudrate van 115200.

Mogelijke libraries

Voor uitbreiding kan gebruik gemaakt worden van de volgende library:

- De UART library (https://www.pjrc.com/teensy/td_uart.html)

Deze library wordt in de uitbreiding gebruikt om met de seriele verbinding te werken. Via deze library kan per seriële poort van de Teensy, de baudrate worden gezet, de poort worden uitgelezen, gegevens via de poort worden verzonden, et cetera.

Samenwerking met het grote systeem

Allereerst wordt in de constructor van de Controller de baudrate van de seriële gezet. Daarnaast wordt in de loop functie van de controller iedere keer gekeken of er gegevens zijn binnengekomen op de seriële verbinding. Wanneer alle twee de bytes zijn ontvangen wordt berekend met welke hoek dit overeenkomt. Deze waarde wordt vervolgens doorgestuurd naar de IMU, die zorg draagt dat deze waarde op de juiste manier wordt verwerkt.

Klassendiagram

Voor deze uitbreiding hoeft het klassendiagram van het huidige systeem niet heel erg te worden uitgebreid. De IMU krijgt een extra variabele, die de meegegeven hoek van de pitch motor bevat die de gebruiker heeft doorgegeven. Daarnaast krijgt de IMU ook een functie die deze variabelen kan setten.

3. Testen

Dit hoofdstuk vormt een aanvulling op het testplan. De punten die in het testplan naar voren komen gelden ook voor deze uitbreiding. In dit hoofdstuk worden daarom ook alleen de aspecten behandeld waarin het testen van het subsysteem verschilt met het testen van het stabilisatie systeem of hier een aanvulling op geeft.

Testopdracht

Als extra aanvulling op de testopdracht die al is beschreven in het testplan, moet ook deze uitbreiding worden getest. Hierbij wordt getest of het systeem naar behoren werkt en wordt gekeken of het voor de beheerders duidelijk is hoe het systeem werkt en moet worden onderhouden.

Alle benodigdheden voor het testen van het systeem zijn al aanwezig.

Teststrategie

Hieronder is de uitgebreide versie van de PRoduct RIsico Matrix te zien. In deze uitbreiding is een extra tabel toegevoegd voor het subsysteem.

PRIMA	Onderdelen ->	Stabiliseren	Variabelen	Pitch hoek	Output
Kwaliteitsaspecten	100	65	15	15	5
1. Geschiktheid	10	xx	x	x	x
2. Juistheid	20	xxx	xx	xx	x
3. Volwassenheid	10	xx	x	x	
4. Beschikbaarheid	5	x	x	x	
5. Tijdbeslag	10	xx	x	xx	x
6. Middelenbeslag	5	x	x	x	
7. Aanpasbaarheid	20	xxx	xx	x	
8. Inpasbaarheid	5	x	x	x	
9. Analyseerbaarheid	5	x	x	x	
10. Testbaarheid	5	x	x	x	
11. Beheerbaarheid	5	x	x	x	

Als extra acceptatie criteria is erbij gekomen:

- PITCH HOEK AANPASSEN – De gegevens die via de seriële port binnenkomen, moeten worden vertaald naar de hoek van de pitchmotor.

Aan de hand van deze extra test eisen, zijn er een aantal testen ontwikkeld die zijn opgenomen in de testscenario's.

Bijlage J: Ontwerp basis camera aansturing

Ontwerp van de uitbreiding:
Camera aansturing

Project:
Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf:
Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

1. Requirements

Dit hoofdstuk vormt een aanvulling op het requirements document van het camera-stabilisatie-systeem. In dit hoofdstuk worden de specifieke eisen opgesteld voor deze systeem uitbreiding. De algemene systeem eisen die in het grote requirements document staan gelden ook voor dit subsysteem.

Doel van het systeem

Door deze uitbreiding moet de gebruiker met de Camera kunnen communiceren. De camera die hiervoor gebruikt wordt bevat twee video mogelijkheden. Een gewone video stand, maar ook een infrarood stand. De keuzen die de gebruiker moet krijgen, zijn:

- Kiezen tussen videosignaal 1 of 2
- De infraroodcamera kalibreren

Interface

De informatie wordt doorgestuurd vanuit de avionica van de spyder. Hoe deze informatie wordt verzonden en op welke manier de gebruiker de keuze kan maken valt buiten de scope van deze uitbreiding. Er hoeft alleen te worden gekeken naar de ontvangst van de signalen en de verwerking ervan. Wel mag een voorstel worden gegeven voor de structuur van de informatie bytes. Hierbij moet rekening worden gehouden met de communicatie tussen de avionica en de Teensy die gaat over het aanpassen van de pitch hoek. Deze twee informatie stromen moeten over dezelfde seriële verbinding lopen. De structuur van de informatie byte is nu als volgt:

- De eerste 2 bits geven het type informatie aan
  - 00 – LS6B voor het aansturen van de pitch as
  - 11 – MS6B voor het aansturen van de pitch as
  - 10 – Aansturen van de camera zelf
  - 01 – Deze mogelijk is nog vrij voor uitbreidingen
- De laatste 6 bits geven de waarde door van de pitch as of geven aan wat de gebruiker wil aanpassen aan de camera

De structuur van de informatie voor het aansturen van de pitch as is al in een ander document beschreven. De structuur van de 6 bits die de gebruikerskeuze doorgeeft om de camera aan te passen is als volgt:

- 000000 – Switch naar video 1
- 000001 – Switch naar video 2
- 000010 – Switch naar video 3
- 000011 – Kalibreer de infrarood camera
- 000100 tot en met 111111 – Deze 60 mogelijkheden zijn nog vrij voor uitbreidingen

Functionele eisen

- De gebruiker moet kunnen switchen tussen video stroom 1 en 2
- De gebruiker moet de infrarood camera kunnen kalibreren. Hierbij moeten de volgende stappen worden uitgevoerd:
    - Allereerst moet de camera naar de infrarood optie worden gezet.
    - Vervolgens moet de camera recht tegen de onderkant van de helikopter aan kijken.
    - Dan moet het kalibratie signaal naar de camera worden verzonden.
    - Tot slot moet de camera weer terug naar zijn oorspronkelijke staat.

Niet functionele eisen

- Het systeem moet samen werken met het huidige camera-stabilisatie-systeem.

Naast deze eisen zijn de algemene systeemeisen die in het hoofd requirements document zijn opgesteld ook op dit systeem van toepassing.

2. Basis/technisch ontwerp

In dit hoofdstuk wordt er een ontwerp gemaakt voor de uitbreiding. Hierbij wordt gekeken naar de specificaties van de Teensy, de mogelijke libraries die gebruikt kunnen worden, de samenwerking met de rest van het programma en hoe het klassendiagram eruit komt te zien.

Specificaties

Voor deze uitbreiding moeten er twee communicatie stromen naar de camera worden opgezet. Dit zijn een pwm signaal en een seriële communicatie. Het pwm signaal wordt gebruikt om tussen de verschillende videostanden te kunnen switchen en de seriele communicatie wordt gebruikt om de camera te vertellen dat hij moet kalibreren. Om deze reden zijn er een pwm poort en een seriële poort nodig op de Teensy. Voor de seriële verbinding worden pinnen 7 en 8 gebruikt, respectievelijk RX3 en TX3. Hier is voor gekozen, omdat deze pinnen geen extra functionaliteit bieden. De pinnen 9 en 10 kunnen namelijk ook nog dienen als pwm signaal. Voor het pwm signaal wordt pin 3 gebruikt, omdat deze gebruikt maakt van een andere timer dan de pwm signalen voor de motoren.

Mogelijke libraries

Voor dit systeem kan gebruik gemaakt worden van de volgende twee libraries:

- De UART library (http://www.pjrc.com/teensy/td_uart.html)
- De PWM library (http://www.pjrc.com/teensy/td_pulse.html)

De UART library wordt gebruikt om serieel met de camera te kunnen communiceren en de pwm library om een pwm signaal naar de camera te kunnen versturen.

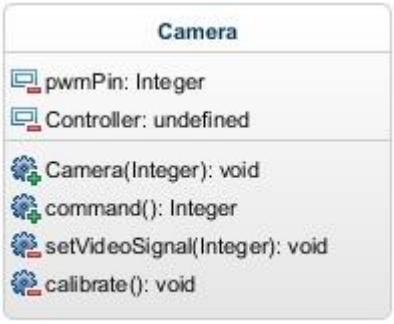
Samenwerking met het grote systeem

Op dit moment bevat de controller een functie die de seriële poort uitleest. Deze functie zal worden uitgebreid om ook de gegevens van de camera te kunnen verwerken.

Daarnaast wordt er een klasse gemaakt om de camera te bedienen. Deze klasse wordt aangestuurd door de controller functie die de seriële informatie uitleest.

Klassendiagram

In onderstaande afbeelding is het klassendiagram te zien voor de Camera klasse. Dit is de klasse die zorg gaat dragen voor de communicatie met de camera. Onder de afbeelding wordt deze klasse verder toegelicht.



Het diagram van de camera klasse is niet heel uitgebreid. Het bevat twee variabelen. De ene variabele houd bij op welke pwm pin de camera zit aangesloten en de tweede variabele bevat een pointer naar het Controller object waar de camera deel van uit maakt.

Wanneer de Controller een commando door krijgt wat voor de camera is bedoeld, wordt deze aan de camera doorgegeven door middel van de functie command(). Deze functie zorgt er vervolgens voor dat de juiste private functies van de Camera klasse worden uitgevoerd.

3. Testen

Dit hoofdstuk vormt een aanvulling op het testplan. De punten die in het testplan naar voren komen gelden ook voor dit subsysteem. In dit hoofdstuk worden daarom ook alleen de aspecten behandeld waarin het testen van het subsysteem verschilt met het testen van het stabilisatie systeem of hier een aanvulling op geeft.

Testopdracht

Als extra aanvulling op de testopdracht, die al is beschreven in het testplan, moet ook het subsysteem worden getest. Hierbij wordt getest of het systeem naar behoren werkt, maar ook of het voor de gebruikers duidelijk is hoe het systeem moet worden gebruikt. Tot slot wordt ook gekeken of het voor de beheerders duidelijk is hoe het systeem werkt en moet worden onderhouden.

Alle benodigdheden voor het testen van het systeem zijn al aanwezig.

Teststrategie

Hieronder is de uitgebreide versie van de PRoduct RIsico Matrix te zien. In deze uitbreiding is een extra tabel toegevoegd voor het subsysteem.

PRIMA	Onderdelen ->	Stabiliseren	Variabelen	Pitch hoek	Camera	Output
Kwaliteitsaspecten	100	50	15	15	15	5
1. Geschiktheid	10	xx	x	x	x	x
2. Juistheid	20	xxx	xx	xx	xx	x
3. Volwassenheid	10	xx	x	x	x	
4. Beschikbaarheid	5	x	x	x	x	
5. Tijdbeslag	10	xx	x	xx	x	x
6. Middelenbeslag	5	x	x	x	x	
7. Aanpasbaarheid	20	xxx	xx	x	xx	
8. Inpasbaarheid	5	x	x	x	x	
9. Analyseerbaarheid	5	x	x	x	x	
10. Testbaarheid	5	x	x	x	x	
11. Beheerbaarheid	5	x	x	x	x	

Als extra acceptatie criteria zijn erbij gekomen:

- VIDEO MODUS KUNNEN BEDIENEN – De gebruikers moeten in staat zijn om tussen de twee verschillende film modussen van de camera te kunnen switchen.
- INFRAROOD KUNNEN KALIBREREN – De gebruiker moet is staat zijn om de infrarood camera te kunnen kalibreren.

Aan de hand van deze extra test eisen, zijn er een aantal testen ontwikkeld die zijn opgenomen in de testscenario's.

Bijlage K: Ontwerp logging

Ontwerp van de uitbreiding:
Logging

Project:
Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf:
Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

1. Requirements

Dit hoofdstuk vormt een aanvulling op het requirements document van het camera-stabilisatie-systeem. In dit hoofdstuk worden de specifieke eisen opgesteld voor deze systeem uitbreiding. De algemene systeem eisen die in het grote requirements document staan, gelden ook voor dit subsysteem.

Doel van het systeem

Door deze uitbreiding moet er een aantal gegevens gelogd kunnen worden op een Micro SD kaart. Om dit mogelijk te maken wordt er gebruik gemaakt van de Catalex MicroSD Card Adapter. De reden om te willen loggen is om te kijken hoe goed het systeem stabiliseert. Deze gegevens kunnen vervolgens worden gebruikt om het stabilisatie systeem te kunnen finetunen.

Interface

De MicroSD Card Adapter zit via SPI verbonden aan de Teensy. De Teensy kan op deze manier via SPI gegevens naar de Teensy zenden, waardoor het mogelijk is om gegevens op te slaan op de MicroSD kaart.

Functionele eisen

- Het moet mogelijk zijn om real-time via SPI een comma seperated file op de MicroSD kaart te zetten.
- Deze file moet op de computer uit te lezen zijn.
- Wanneer er al een file op de MicroSD kaart staat, moet er tweede logfile worden aangemaakt. Wanneer er al een tweede is een derde enzovoorts.

Niet functionele eisen

- Het systeem moet samen werken met het huidige camera-stabilisatie-systeem.
- De structuur van de file moet er als volgt uitzien:
    - [Milliseconden vanaf de start], [Uitleg over een bepaalde waarde], [de waarde]
    - [Milliseconden vanaf de start], [Uitleg over een bepaalde waarde], [de waarde]
    - ….

Naast deze eisen zijn de algemene systeemeisen die in het hoofd requirements document zijn opgesteld ook op dit systeem van toepassing.

2. Basis/technisch ontwerp

In dit hoofdstuk wordt er een ontwerp gemaakt voor de uitbreiding. Hierbij wordt gekeken naar de specificaties van de Teensy, de mogelijke libraries die gebruikt kunnen worden, de samenwerking met de rest van het programma en hoe het klassendiagram eruit komt te zien.

Specificaties

Voor deze uitbreiding moeten naast de sensor ook de MicroSD Card Adapter via SPI aan de Teensy worden aangesloten. Voor de MISO, MOSI en SCK worden uiteraard dezelfde pinnen gebruikt. Voor de SC wordt pin 14 gebruikt. Deze pin heeft namelijk geen extra functionaliteit, maar ligt wel in de buurt van de SC van de sensor (pin 15) en de SCK pin (pin 13)

Mogelijke libraries

Voor dit systeem kan gebruik gemaakt worden van de volgende library:

- De SdFat library (https://code.google.com/p/sdfatlib/downloads/detail?name=sdfatlib20131225.zip&can=2&q=)

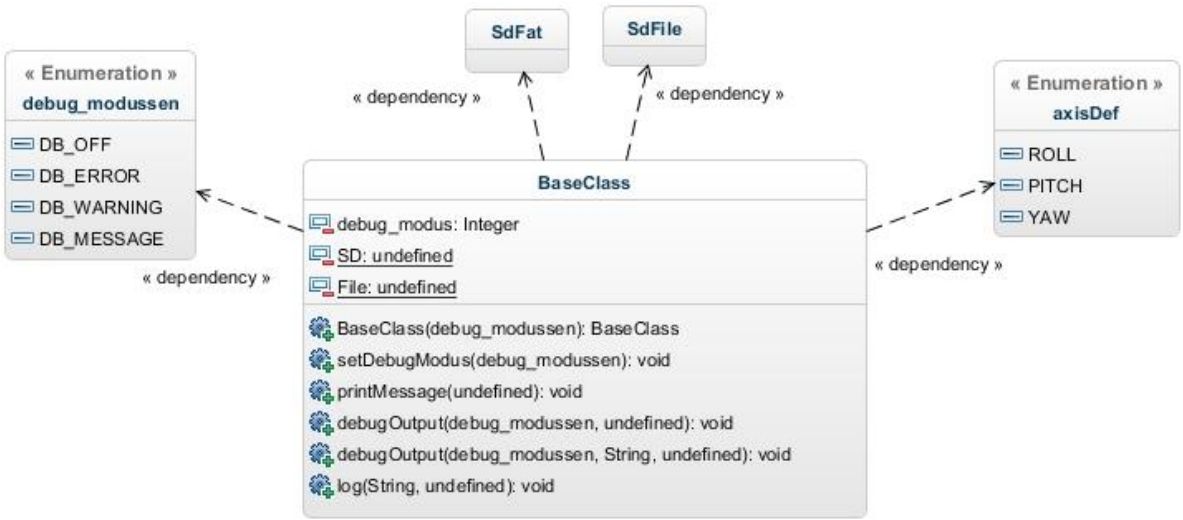
Deze library bevat alle functionaliteit die nodig is om files op de Sdkaart te kunnen aanmaken, verwijderen en wijzigen.

Samenwerking met het grote systeem

Voor deze uitbreiding zal de BaseClass verder worden uitgebreid met een functie om gegevens naar de MicroSD kaart te kunnen wegschrijven. Op deze manier kunnen alle objecten gegevens loggen.

Klassendiagram

In onderstaande afbeelding is het klassendiagram te zien voor de BaseClass uitbreiding. Onder de afbeelding wordt deze klasse verder toegelicht.



Aan de bestaande functies in de BaseClass is niets veranderd, alleen zijn er een functie en twee attributen aan de klasse toegevoegd. De statische attributen SD en File zijn pointers naar klassen uit de SdFat library. Deze attributen bevatten respectievelijk de pointer naar de SD kaart en naar het geheugen op de SD kaart. Deze attributen zijn statisch gemaakt, zodat alle objecten dezelfde SD kaart aanspreken.

3. Testen

Dit hoofdstuk vormt een aanvulling op het testplan. De punten die in het testplan naar voren komen gelden ook voor dit subsysteem. In dit hoofdstuk worden daarom ook alleen de aspecten behandeld, waarin het testen van het subsysteem verschilt met het testen van het stabilisatie systeem of hier een aanvulling op geeft.

Testopdracht

Als extra aanvulling op de testopdracht, die al is beschreven in het testplan, moet ook het subsysteem worden getest. Hierbij wordt getest of het systeem naar behoren werkt. Tot slot wordt ook gekeken of het voor de beheerders duidelijk is hoe het systeem werkt en moet worden onderhouden.

Alle benodigdheden voor het testen van het systeem zijn al aanwezig.

Teststrategie

Hieronder is de uitgebreide versie van de PRoduct RIsico Matrix te zien. In deze uitbreiding is een extra tabel toegevoegd voor het subsysteem.

PRIMA	Onderdelen ->	Stabiliseren	Variabelen	Pitch hoek	Camera	Logging	Output
Kwaliteitsaspecten	100	45	12	12	12	12	7
1. Geschiktheid	10	xx	x	x	x	x	x
2. Juistheid	20	xxx	xx	xx	xx	xx	x
3. Volwassenheid	10	xx	x	x	x	x	
4. Beschikbaarheid	5	x	x	x	x	x	
5. Tijdbeslag	10	xx	x	xx	x	xx	x
6. Middelenbeslag	5	x	x	x	x	x	
7. Aanpasbaarheid	20	xxx	xx	x	xx	x	
8. Inpasbaarheid	5	x	x	x	x	x	
9. Analyseerbaarheid	5	x	x	x	x	x	
10. Testbaarheid	5	x	x	x	x	x	
11. Beheerbaarheid	5	x	x	x	x	x	

Als extra acceptatie criteria zijn erbij gekomen:

- GEGEVENS IN EEN LOGFILE KUNNEN OPSLAAN – Het moet mogelijk zijn om gegevens in een logfile op de SD kaart op te slaan.
- LOGFILES JUIST BEHEREN – De logfiles moeten juist worden beheerd. Wanneer er bijvoorbeeld al een logfile is, moet er een tweede worden aangemaakt.

Aan de hand van deze extra test eisen, zijn er een aantal testen ontwikkeld die zijn opgenomen in de testscenario's.

Bijlage L: Testscenario's

Testscenario's

Project:
Ontwikkelen van een camera-stabilisatie-systeem

Bedrijf:
Delft Dynamics B.V.

Plaats, datum: Ter-Aar, 8 mei 2014

Opgesteld door:

Jeroen de Meij | 06 – 27 04 73 41 | jeroendemeij@gmail.com

Versiebeheer

Versie	Datum	Geschreven door	Beschrijving
0.1	24 feb 2014	Jeroen de Meij	Eerste opzet van het document. Structuur is opgezet. Inleiding, uitleg bij de testsoorten en scenario's AF1 en AG1 geschreven.
0.2	25 feb 2014	Jeroen de Meij	Testgeval nummering aangepast, typefouten verbeterd en scenario's AP1, AP2, AC1, AC2, S1, S2, S3, S4 en S5 geschreven.
0.3	26 feb 2014	Jeroen de Meij	De kwaliteitsaspecten opgenomen in de testbasis van de geschreven testen.
0.4	6 mrt 2014	Jeroen de Meij	Scenario's I1, I2 en I3 geschreven.
0.5	28 mrt 2014	Jeroen de Meij	Scenario's AG1-2, S6 en I4 geschreven.
0.6	1 apr 2014	Jeroen de Meij	Scenario's AG1-3 en I 5 geschreven.
0.7	7 apr 2014	Jeroen de Meij	Scenario's AG1-4, AG1-5 en I6 geschreven.
0.8	14 apr 2014	Jeroen de Meij	Scenario's AG1-6 en I7 geschreven.
0.9	29 apr 2014	Jeroen de Meij	Scenario's I1, I2, I3 en I4 aangepast.
0.10	5 mei 2014	Jeroen de Meij	Scenario S7 geschreven.
1.0	27 mei 2014	Jeroen de Meij	Functies om te testen toegevoegd aan Scenario AP1 en AP2-2 en diagrammen om te testen toegevoegd aan Scenario AP2-1

Inleiding

In dit document staan alle testscenario's die zijn ontwikkeld voor het project: “Ontwikkelen van een camera-stabilisatie-systeem”. Deze scenario's staan gesorteerd per testfase. De volgende drie fasen komen daarbij aan bod:

- Acceptatie tests
- Systeem tests
- Integratie tests

Deze drie fasen hebben ieder een hoofdstuk binnen dit document. Hierbij staat eerst per testfase aangegeven wat het globale doel is van deze testfase en vervolgens staan per fasen de testscenario's weergegeven.

Acceptatie tests

In de acceptatie tests wordt het product getest in de daadwerkelijke gebruiksomgeving, wanneer deze helemaal af is.

Functionele acceptatie tests

In de functionele acceptatie test wordt in de daadwerkelijke gebruiksomgeving de functionaliteit van het product nogmaals getest.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **AF1**

Omschrijving **Gebruikers testen de beeldkwaliteit**

Versie	Datum	Auteur	Opmerkingen
0.1	24 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis

---

TESTOBJECT EN ACHTERGROND

Het uiteindelijke doel van het systeem is om de camera te stabiliseren waardoor er een stabiel camera beeld ontstaat. Hierbij is het belangrijk om te weten wat de gebruikers van het resultaat vinden.

TESTMISSIE

Test of de gebruikers tevreden zijn met het camera beeld.

TESTBASIS

- Deze test is gebaseerd op de functionele eis, dat het systeem stabiel moet kunnen filmen, vastgesteld in het requirements document.
- Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid en inpasbaarheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

Het systeem is met succes vastgekoppeld aan de multicopter. Daarnaast moet de camera aanstaan en te besturen zijn via de Android applicatie.

TESTDOELEN MET TESTAANWIJZINGEN

- HET CAMERA BEELD MOET STABIEL ZIJN

De gebruikers bepalen of ze tevreden zijn met het eindresultaat.

NIET TESTEN

Het gebruiksgemak van het systeem.

TESTGEVALLEN

Geval 1:

- Zorg ervoor dat de camera helemaal uitgezoomd is en laat de camera filmen.
- Laat de multicopter opstijgen.
- Beweeg de multicopter minimaal 1 keer in alle richtingen (naar voren, naar achteren, naar link naar rechts).
- Laat de multicopter een keer om zijn as draaien.
- Laat de multicopter weer landen.

Verwacht resultaat: De gebruikers zijn tevreden over de stabiliteit van het gemaakte beeldmateriaal

Geval 2:

- Zorg ervoor dat de camera helemaal ingezoomd is en laat de camera filmen.
- Laat de multicopter opstijgen.
- Beweeg de multicopter minimaal 1 keer in alle richtingen (naar voren, naar achteren, naar link naar rechts).
- Laat de multicopter een keer om zijn as draaien.
- Laat de multicopter weer landen.

Verwacht resultaat: De gebruikers zijn tevreden over de stabiliteit van het gemaakte beeldmateriaal

Geval 3:

- Zorg ervoor dat de camera helemaal uitgezoomd is.
- Laat de multicopter opstijgen.
- Beweeg de multicopter minimaal 1 keer in alle richtingen (naar voren, naar achteren, naar link naar rechts) waarbij telkens een aantal foto's worden genomen: Op het moment dat de multicopter zijn beweging inzet en wanneer de multicopter stabiel vliegt.
- Laat de multicopter een keer om zijn as draaien waarbij een aantal foto's worden genomen.
- Laat de multicopter weer landen.

Verwacht resultaat: De gebruikers zijn tevreden over de stabiliteit van de gemaakte foto's

Gebruikers acceptatie tests

In de gebruikers acceptatie test wordt in de daadwerkelijke gebruiksomgeving getest of gebruikers goed met het systeem kunnen omgaan.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **AG1**

Omschrijving **Gebruikers testen het gebruiksgemak**

Versie	Datum	Auteur	Opmerkingen
0.1	24 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis
0.3	28 mrt 2014	Jeroen de Meij	Testgeval 2 toegevoegd
0.4	1 apr 2014	Jeroen de Meij	Testgeval 3 toegevoegd
0.5	7 apr 2014	Jeroen de Meij	Testgeval 4 en 5 toegevoegd

TESTOBJECT EN ACHTERGROND

Het systeem moet voor gebruikers makkelijk te gebruiken zijn, waarbij het belangrijk is dat gebruikers weten op welk moment het systeem klaar is voor gebruik.

TESTMISSIE

Test of de gebruikers snappen hoe het systeem moet worden gebruikt.

TESTBASIS

1. Deze test is gebaseerd op het feit dat er geen userinterface beschikbaar is, maar de gebruikers toch moeten weten hoe het systeem te bedienen is.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid en inpasbaarheid van het stabiliseren. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

Er is een multicopter beschikbaar die in staat is om te vliegen, waar het camera-systeem aan vastgekoppeld kan worden.

TESTDOELEN MET TESTAANWIJZINGEN

- DE GEBRUIKERS MOETEN HET SYSTEEM KUNNEN BEDIENEN

Het moet voor de gebruikers duidelijk zijn hoe het systeem wordt aangezet en wanneer het systeem gereed is voor gebruik.

NIET TESTEN

Er hoeft niet te worden gekeken naar de kwaliteit van het systeem en of het stabiliseren wel werkt.

Geval 1:

- Vraag van te voren aan de gebruikers of ze aan willen geven wanneer zij denken dat het systeem klaar is voor gebruik.
- Geef de gebruikers de multicopter en het camera-systeem.
- Laat de gebruikers het systeem bevestigen en aanzetten.

Verwacht resultaat: Het camera-systeem is gereed voor gebruik en de gebruikers geven op het juiste moment aan wanneer dit het geval is.

Geval 2:

- Vertel aan de gebruikers welke variabelen er aangepast moeten worden.
- Laat de gebruikers de veranderingen doorvoeren.

Verwacht resultaat: De gebruikers hebben de variabelen in het systeem veranderd met het ene programma en het camera-stabilisatie-systeem werkt nu met de aangepast variabelen.

Geval 3:

- Vertel aan de gebruikers dat de camera eerst op + 45 graden moet worden gezet, daarna op –135 graden en tot slot op 0 graden.
- Laat de gebruikers de veranderingen doorvoeren.

Verwacht resultaat: De gebruikers hebben ervoor gezorgd dat de pitch hoek van de camera veranderde, overeenkomstig met de eerder genoemde hoeken.

Geval 4:

- Vertel aan de gebruikers dat de camera eerst op infrarood modus moet worden gezet en wanneer dat is gebeurt weer terug op de normale videomodus.
- Laat de gebruikers de veranderingen doorvoeren.

Verwacht resultaat: De gebruikers hebben ervoor gezorgd dat de camera tussen de twee videostromen is geswitcht.

Geval 5:

- Vertel aan de gebruikers dat de infrarood modus van de camera moet worden gekalibreerd en vraag of ze willen aangeven wanneer ze denken dat het kalibreren klaar is.
- Laat de gebruikers de veranderingen doorvoeren.

Verwacht resultaat: De gebruikers hebben ervoor gezorgd dat de infrarood modus van de camera is aangepast.

Geval 6:

- Vertel aan de gebruikers dat er informatie in een logfile is opgeslagen en laat ze deze logfile op de computer uitlezen.

Verwacht resultaat: De gebruikers hebben de logfile op de computer terug kunnen vinden en uit kunnen lezen.

Productie acceptatie tests

In de productie acceptatie test testen of de beheerders het product goed in beheer kunnen nemen.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **AP1**

Omschrijving **Beheerders bekijken de source-code**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis
0.3	27 mei 2014	Jeroen de Meij	In het scenario zijn bij het testgeval een aantal functies genoteerd waarvan belangrijk is dat de beheerders ze kunnen uitleggen.

TESTOBJECT EN ACHTERGROND

Voor de mensen die het systeem in beheer gaan nemen moet het duidelijk zijn wat de source-code doet, zodat zij later extra functionaliteit kunnen toevoegen.

TESTMISSIE

Test of de beheerders aan de hand van het inline commentaar snappen wat de code doet.

TESTBASIS

- Als basis voor deze test dient het kopje maintainability in het requirements document. Deze eis stelt dat het voor de beheerders mogelijk moet zijn om het systeem te onderhouden.
- Daarnaast is deze test gebaseerd op de kwaliteitseisen beheerbaarheid en aanpasbaarheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De source-code is beschikbaar voor de beheerders.

TESTDOELEN MET TESTAANWIJZINGEN

- DE BEHEERDERS MOETEN DE SOURCE CODE BEGRIJPEN

Het moet voor de beheerders duidelijk zijn wat de source-code doet

NIET TESTEN

Tijdens deze test hoeven de beheerders niet te kijken of de source-code werkt. Het kan natuurlijk zijn dat ze opmerkingen hebben in het geval dat ze de code begrijpen en fouten tegenkomen. Noteer deze opmerkingen wel, maar ze zijn niet het doel van de test.

TESTGEVALLEN

- Overhandig de beheerders de code en vraag ze naar de werking van een aantal functies en dergelijken.
  - De functies loop en readAvionicaInput van de Controller
  - De functies writeserial en clocktick van de Camera
  - De functies getInstance en readFromEEPROM van de Config
  - De functie setAngle van de Motor

Verwacht resultaat: De beheerders kunnen zelf uitleggen wat de verschillende stukken code doen.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **AP2**

Omschrijving **Beheerders bekijken de documentatie**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis
0.3	27 mei 2014	Jeroen de Meij	In het scenario zijn bij GEVAL I een aantal diagrammen en bij GEVAL II een aantal functies genoteerd waarvan belangrijk is dat de beheerders ze kunnen uitleggen.

TESTOBJECT EN ACHTERGROND

Voor de mensen die het systeem in beheer gaan nemen moet aan de hand van de documentatie duidelijk zijn hoe het systeem werkt, zodat zij later extra functionaliteit kunnen toevoegen.

TESTMISSIE

Test of de beheerders aan de hand van de documentatie snappen hoe het systeem werkt.

TESTBASIS

- Als basis voor deze test dient het kopje maintainability in het requirements document. Deze eis stelt dat het voor de beheerders mogelijk moet zijn om het systeem te onderhouden.
- Daarnaast is deze test gebaseerd op de kwaliteitseis beheerbaarheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De documentatie is beschikbaar voor de beheerders.

TESTDOELEN MET TESTAANWIJZINGEN

- DE BEHEERDERS MOETEN DE UML-DIAGRAMMEN BEGRIJPEN

Aan de hand van een aantal uml-diagrammen, moet de globale werking en samenhang van het systeem duidelijk worden.

- DE BEHEERDERS MOETEN DE DOCUMENTATIE BEGRIJPEN

Aan de hand van de documentatie moet het voor de beheerders duidelijk zijn wat de verschillende functies van het systeem doen en hoe ze moeten worden gebruikt.

NIET TESTEN

Tijdens deze test hoeven de beheerders niet te kijken of de werking van het systeem en de uml en overige notatie correct is. Het kan natuurlijk zijn dat ze opmerkingen hebben in het geval dat ze de documentatie begrijpen en fouten tegenkomen. Noteer deze opmerkingen wel, maar ze zijn niet het doel van de test.

TESTGEVALLEN

GEVAL 1:

- Overhandig de beheerders de volgende sequentie diagrammen en laat ze aan de hand van deze diagrammen het systeem uitleggen:
  - Opstarten
  - Loop
  - Camera kalibratie

Verwacht resultaat: De beheerders kunnen aan de hand van de uml-diagrammen de samenhang en globale werking van het systeem uitleggen.

GEVAL 2:

- Overhandig de beheerders de documentatie en vraag ze naar de werking en manier van gebruiken van een aantal functies.
  - De functie writeSerial van de Camera
  - De functie debugOut() van de BaseClass

Verwacht resultaat: De beheerders kunnen aan de hand van de documentatie uitleggen hoe de verschillende functies moeten worden gebruikt.

Continuïteit tests

In de continuïteit test wordt in de daadwerkelijke gebruiksomgeving getest of het product niet vastloopt.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **AC1**

Omschrijving **Het systeem mag niet vastlopen**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis

TESTOBJECT EN ACHTERGROND

In de vorige versie kwam het voor dat het systeem vast liep op het moment dat de camera tegen de onderkant van de multicopter aankwam en zodoende niet extra kon stabiliseren. In deze test wordt gekeken of het systeem in die situatie nog steeds vastloopt.

TESTMISSIE

Test of het moment waarop het oude systeem vastliep is verholpen.

TESTBASIS

1. Als basis voor deze test dient het kopje availability in het requirements document. Deze eis stelt dat het systeem, altijd moet werken als het in gebruik is.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen beschikbaarheid en volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

Het systeem is met succes vastgekoppeld aan de multicopter.

TESTDOELEN MET TESTAANWIJZINGEN

- HET SYSTEEM MAG NIET VASTLOPEN

De situatie waarbij de vorige keer het systeem vastliep wordt nagebootst.

NIET TESTEN

Het gebruiksgemak van het systeem

TESTGEVALLEN

- Pak de multicopter op.
- Draai de multicopter, zodanig dat de camera tegen de onderkant van de multicopter aan tikt en daarna iets verder.
- Draai de multicopter weer recht.
- Draai de multicopter de andere kant op, zodanig dat de camera tegen de onderkant van de multicopter aan tikt en daarna iets verder.
- Draai de multicopter weer recht.
- Heerhaal deze stappen 50 keer.

Verwacht resultaat: Op het moment dat de multicopter weer recht wordt gedraaid moet de camera weer gewoon worden gestabiliseerd.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **AC2**

Omschrijving **Het systeem moet blijven werken bij normaal gebruik**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis

TESTOBJECT EN ACHTERGROND

Bij deze test wordt gekeken of het gehele systeem langere tijd gebruikt kan worden. Deze test wordt uitgevoerd tijdens een daadwerkelijke vlucht.

TESTMISSIE

Test of het systeem voor een langere tijd blijft werken.

TESTBASIS

1. Als basis voor deze test dient het kopje availability in het requirements document. Deze eis stelt dat het systeem, altijd moet werken als het in gebruik is.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen beschikbaarheid en volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

Het systeem is met succes vastgekoppeld aan de multicopter. Daarnaast moet de camera aanstaan en te besturen zijn via de Android applicatie.

TESTDOELEN MET TESTAANWIJZINGEN

- HET SYSTEEM MOET BLIJVEN WERKEN

Het systeem wordt voor langere tijd gebruikt in een reële situatie.

NIET TESTEN

Het gebruiksgemak van het systeem

TESTGEVALLEN

- Zorg ervoor dat de camera helemaal uitgezoomed is.
- Laat de multicopter opstijgen.
- Beweeg de multicopter minimaal 1 keer in alle richtingen (naar voren, naar achteren, naar link naar rechts).
- Laat de multicopter een keer om zijn as draaien.
- Heerhaal deze handelingen een aantal keren, hoe meer hoe beter en noteer dit.

Verwacht resultaat: Het systeem blijft de gehele tijd de camera stabiliseren.

Systeem tests

In de systeem tests wordt, nog niet in de daadwerkelijke gebruiksomgeving, gekeken of het systeem alle functionaliteit heeft, zoals die van te voren is opgesteld.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **S1**

Omschrijving **Debugging moet mogelijk zijn**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis
0.3	1 mei 2014	Jeroen de Meij	Het testgeval uitgebreid

TESTOBJECT EN ACHTERGROND

Om het systeem goed te kunnen testen moet het mogelijk zijn om debug informatie naar de pc te zenden en daar te tonen.

TESTMISSIE

Testen of de debug berichten op de pc getoond kunnen worden

TESTBASIS

1. Als basis voor deze test is de eis om het systeem te kunnen onderhouden. Hiervoor moet het dus mogelijk zijn om bepaalde informatie tijdens het gebruik te kunnen tonen. Hiervoor is de debug mogelijkheid ontworpen.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen analyseerbaarheid en testbaarheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet via een USB-kabel aan de pc zijn verbonden.

TESTDOELEN MET TESTAANWIJZINGEN

- DE DEBUGGING MOET WERKEN

Via de USB verbinding worden er een aantal debug berichten doorgestuurd die vervolgens op het scherm worden toont.

NIET TESTEN

Er hoeft niet getest te worden of het systeem de camera stabiliseert.

TESTGEVALLEN

- Zet in de code de debug modus op DB_MESSAGE
- Laat het programma normaal draaien.

Verwacht resultaat: Op het scherm verschijnen er een aantal debug berichten. Hierbij wordt gemeld welke objecten er zijn aangemaakt, of de EEPROM wel of niet is uitgelezen en wat de offset is van de gyroscoop.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **S2**

Omschrijving **De controller moet sensor waarden kunnen opvragen**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis

TESTOBJECT EN ACHTERGROND

In deze test wordt gekeken of de koppeling tussen de controller en de sensor naar behoren werkt, waarbij de sensor 1000 keer per seconde gegevens moet uitlezen.

TESTMISSIE

Testen of via de controller de sensor uitgelezen kan worden.

TESTBASIS

1. Als basis voor deze test is de eis om de camera te stabiliseren door onder andere gegevens uit de sensor uit te lezen. Hierbij is als eis gesteld dat de sensor 1000 keer per seconde moet worden uitgelezen en de gegevens worden verwerkt.
2. Daarnaast is deze test gebaseerd op de kwaliteitseis juistheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet via een USB-kabel aan de pc zijn verbonden. En de sensor moet zijn aangesloten aan de Teensy.

TESTDOELEN MET TESTAANWIJZINGEN

- DE SENSOR MOET VOLDOENDE WORDEN UITGELEZEN

In het systeem moet worden bijgehouden hoe vaak de sensor per seconde wordt uitgelezen. Dit wordt vervolgens op het scherm getoond.

NIET TESTEN

Er hoeft niet te worden getest of de gegevens van de sensor juist worden verwerkt om de motoren aan te sturen.

TESTGEVALLEN

- Start de timer die de sensor moet uitlezen.
- Houd een teller bij met het aantal keer dat de sensor wordt uitgelezen.
- Toon na een minuut de inhoud van de teller op het scherm.

Verwacht resultaat: Op het scherm wordt getoond dat de sensor in totaal 1000 x 60 = 60.000 keer is aangeroepen.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **S3**

Omschrijving **De camera moet worden gestabiliseerd**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis

TESTOBJECT EN ACHTERGROND

Tijdens deze test wordt de hoofdtaak van het systeem getest: De mogelijkheid om de camera te stabiliseren.

TESTMISSIE

Testen of de camera bij beweging van het systeem stabiel blijft filmen.

TESTBASIS

1. Deze test is gebaseerd op de functionele eis, dat het systeem stabiel moet kunnen filmen, vastgesteld in het requirements document.
2. Daarnaast is deze test gebaseerd op de kwaliteitseis juistheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet via een USB-kabel aan de pc zijn verbonden. En zowel de sensor als de twee motoren moeten op de Teensy zijn aangesloten.

TESTDOELEN MET TESTAANWIJZINGEN

- HET SYSTEEM MOET STABIEL KUNNEN FILMEN

Tijdens deze test wordt er gekeken of de camera wordt gestabiliseerd. Hiervoor wordt het systeem volgens een patroon heen en weer bewogen.

NIET TESTEN

Er hoeft hierbij niet te worden getest of het systeem makkelijk te gebruiken is.

TESTGEVALLEN

- Zet de camera aan, zorg ervoor dat deze helemaal is ingezoomd en laat de camera filmen.
- Pak het systeem op
- Kantel het systeem een keer in alle richtingen met de camera als het middelpunt waar omheen wordt gedraaid. (vooruit, achteruit, naar links, naar rechts)
- Zet het systeem weer neer en zet de camera uit.

Verwacht resultaat: de camera heeft een stabiel beeld opgenomen

ADMINISTRATIEVE GEGEVENS

Scenario-ID **S4**

Omschrijving **Het systeem moet voor langere tijd blijven werken**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis

TESTOBJECT EN ACHTERGROND

In deze test wordt gekeken of het hele systeem voor langere tijd operationeel kan blijven. Op deze manier wordt getest of het systeem ook in staat is om bij langere vluchten te worden ingezet, zonder dat het systeem ermee stopt.

TESTMISSIE

Test of het systeem langere tijd operationeel kan blijven.

TESTBASIS

1. Als basis voor deze geldt de functionele eis dat het systeem stabiel moet kunnen filmen, vastgesteld in het requirements document.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen beschikbaarheid en volowassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. De sensor en de twee motoren moeten op de Teensy zijn aangesloten.

TESTDOELEN MET TESTAANWIJZINGEN

- HET SYSTEEM MAG NIET VASTLOPEN

De situatie waarin het oude systeem vastliep, wordt nagebootst. Hierbij mag het nieuwe systeem niet meer vastlopen.

- HET SYSTEEM MOET LANGERE TIJD STABIEL KUNNEN FILMEN

Tijdens deze test wordt er gekeken of het systeem langere tijd kan worden gebruikt. Hiervoor wordt het systeem volgens een patroon heen en weer bewogen, over een langere tijd.

NIET TESTEN

Er hoeft hierbij niet te worden getest of het systeem makkelijk te gebruiken is.

GEVAL 1:

- Pak het systeem op
- Kantel het systeem naar links, waarbij de camera aan het einde zachtjes de andere kant op wordt geduwd.
- Kantel het systeem naar rechts, waarbij de camera aan het einde zachtjes de andere kant op wordt geduwd.
- Heerhaal dit 50 keer.

Verwacht resultaat: Ondanks dat het systeem wordt tegengewerkt, moet de camera hierna weer worden gestabiliseerd.

GEVAL 2:

- Zet de camera aan, zorg ervoor dat deze helemaal is uitgezoomd en laat de camera filmen.
- Pak het systeem op
- Kantel het systeem een minuut lang een aantal keer in alle richtingen, met de camera als het middelpunt waar omheen wordt gedraaid. (vooruit, achteruit, naar links, naar rechts)
- Zet het systeem weer neer en wacht 10 minuten.
- Pak het systeem op
- Kantel het systeem een minuut lang een aantal keer in alle richtingen, met de camera als het middelpunt waar omheen wordt gedraaid. (vooruit, achteruit, naar links, naar rechts)
- Zet het systeem neer en de camera uit.

Verwacht resultaat: de camera heeft een stabiel beeld opgenomen

ADMINISTRATIEVE GEGEVENS

Scenario-ID **S5**

Omschrijving **Het systeem moet ruimte bieden voor uitbreiding**

Versie	Datum	Auteur	Opmerkingen
0.1	25 feb 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	26 feb 2014	Jeroen de Meij	Kwaliteitseisen toegevoegd aan de testbasis

TESTOBJECT EN ACHTERGROND

Bij deze test wordt van een aantal functies bekeken hoe lang ze duren. Op deze manier kan worden bepaald of er nog ruimte vrij is om extra functionaliteit toe te voegen.

TESTMISSIE

Testen of het systeem niet 100% van zijn capaciteit gebruikt

TESTBASIS

1. Als testbasis dient de eis dat het systeem uitbreidbaar moet zijn om later extra functionaliteit te kunnen toevoegen. Deze eis is opgenomen onder het kopje maintainability van het requirements document.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen aanpasbaarheid, tijdbeslag en middelenbeslag van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. De sensor en de twee motoren moeten op de Teensy zijn aangesloten.

TESTDOELEN MET TESTAANWIJZINGEN

- HET SYSTEEM MOET DE MOGELIJKHEID TOT UITBREIDEN BIEDEN

Van een aantal functies wordt bepaald hoe lang het duurt om deze functies uit te voeren. Hiervoor wordt de klasse elapsedMicros gebruikt.

NIET TESTEN

Het gebruiksgemak en de werking van het systeem hoeven niet te worden getest. Uiteraard kan het wel voorkomen dat tijdens het testen blijkt dat het systeem niet wekt. Vermeld dit dan bij de opmerkingen.

TESTGEVALLEN

- Test van minimaal de volgende functies hoe lang het duurt om ze uit te voeren, door voor dat de functie wordt aangeroepen op te slaan hoeveel micro seconden er voorbij zijn sinds het starten van het systeem, de functie uit te voeren, en vervolgens het huidig aantal micro seconden die er voorbij zijn sinds het starten van het systeem af te trekken van de eerste opgeslagen waarden:
  - Uitlezen van de sensor waarden
  - Berekenen en aansturen van de stand van de motoren
  - Genereren van debug berichten
  - De totale loop functie

Verwacht resultaat: Door een combinatie van alle gegevens te maken moet blijken dat alles bij elkaar minder dan 1000 micro seconden duurt.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **S6**

Omschrijving **Variabelen moeten aanpasbaar zijn**

Versie	Datum	Auteur	Opmerkingen
0.1	28 mrt 2014	Jeroen de Meij	Eerste versie van het scenario

TESTOBJECT EN ACHTERGROND

Het moet mogelijk zijn om de variabelen van het systeem aan te passen.

TESTMISSIE

Test of het aanpassen van de variabelen werkt.

TESTBASIS

1. Deze test is gebaseerd op de functionele eisen, dat het mogelijk moet zijn om variabelen in de EEPROM te schrijven en dat het stabilisatie systeem deze variabelen moet kunnen uitlezen, vastgesteld in het ontwerp document van het subsysteem, variabelen aanpassen.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid en volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het ontwerp document van het subsysteem, variabelen aanpassen.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. De sensor en de twee motoren moeten op de Teensy zijn aangesloten.

TESTDOELEN MET TESTAANWIJZINGEN

- VARIABELEN MOETEN AANPASBAAR ZIJN

Het moet mogelijk zijn om via een een los programma variabelen in de EEPROM weg te schrijven, waarna het stabilisatie systeem deze variabelen uitleest.

NIET TESTEN

Het gebruiksgemak van het systeem.

TESTGEVALLEN

- Pas minimaal de bovenste en onderste variabele uit de struct aan.
- Laat het programma om de variabelen aan te passen op de Teensy.
- Laat vervolgens het stabilisatie programma op de Teensy.

Verwacht resultaat: Het stabilisatie systeem moet de doorgevoerde veranderingen hebben overgenomen.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **S7**

Omschrijving **Het systeem moet gebruikers input kunnen verwerken**

Versie	Datum	Auteur	Opmerkingen
0.1	28 mrt 2014	Jeroen de Meij	Eerste versie van het scenario

TESTOBJECT EN ACHTERGROND

Het moet mogelijk zijn om via een seriële verbinding de pitch hoek van de camera aan te passen, te switchen tussen de verschillende video modi en om de camera te calibreren.

TESTMISSIE

Test of de seriële communicatie werkt.

TESTBASIS

1. Deze test is gebaseerd op de functionele eisen, dat de gebruiker in staat moet zijn om de pitch hoek van de camera aan te passen, vastgesteld in het ontwerp document van het subsysteem, pitch hoek aanpassen en dat de gebruiker in staat moet zijn om tussen videostromen te switchen en de infrarood camera moet kunnen kalibreren, vastgesteld in het ontwerp document van het subsysteem, camera aansturing.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid en volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het ontwerp document van het subsysteem, variabelen aanpassen.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. Op de Arduino moet het juiste programma voor deze test draaien. Daarnaast moeten de TX en RX porten van de Arduino en Teensy juist met elkaar verbonden zijn, evenals de TX en RX porten van de infrarood camera. Tot slot moet de video stroom op de pc zichtbaar worden gemaakt.

TESTDOELEN MET TESTAANWIJZINGEN

- DE PITCHHOEK VAN DE CAMERA MOET AAN TE STUREN ZIJN

Het moet mogelijk zijn om via een seriële code, van de Arduino naar de Teensy te sturen, de pitchhoek van de camera in te stellen.

- ER MOET TUSSEN VIDEO STROMEN KUNNEN WORDEN GESWITCHED

Het moet mogelijk zijn om via een seriële code, van de Arduino naar de Teensy te sturen, tussen de verschillende videostromen van de camera te kunnen switchen.

- DE INFRAROOD CAMERA MOET KUNNEN WORDEN GEKALIBREERD

Het moet mogelijk zijn om via een seriële code, van de Arduino naar de Teensy te sturen, de infrarood camera te kunnen calibreren.

NIET TESTEN

Het gebruiksgemak van het systeem.

Geval 1:

- Zorg ervoor dat het juiste programma op de Arduino staat.
- Zet eerst de Teensy aan en vervolgens de Arduino.
- Laat de Arduino een aantal seriële commando's sturen die de pitch hoek van de camera moeten aanpassen.

Verwacht resultaat: Het stabilisatie systeem moet de pitch hoek aannemen die door de Arduino wordt gestuurd.

Geval 2:

- Zorg ervoor dat het juiste programma op de Arduino staat.
- Zet eerst de Teensy aan en vervolgens de Arduino.
- Laat de Arduino om en om de seriële commando's sturen om te schakelen tussen de verschillende videostromen.

Verwacht resultaat: Wanneer het commando is gestuurd om een andere videostroom te activeren, moet deze video stroom zichtbaar worden op het computer scherm.

Geval 3:

- Zorg ervoor dat het juiste programma op de Arduino staat.
- Zet eerst de Teensy aan en vervolgens de Arduino.
- Laat de Arduino het commando sturen om de infrarood camera te kalibreren.

Verwacht resultaat: Wanneer het commando is verstuurd moet het systeem stoppen met het stabiliseren van de camera en overschakelen naar de infrarood modus. Vervolgens moet de camera recht omhoog naar de onderkant van de helikopter gaan kijken. Wanneer dat het geval is, moet het commando DO_FFC naar de camera worden gestuurd om de infrarood camera te kalibreren. Wanner dat is gedaan moet het systeem weer verder gaan met stabiliseren.

Integratie tests

In de integratie test wordt gekeken of de verschillende onderdelen binnen het systeem goed werken in samenhang met elkaar.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **I1**

Omschrijving **Debug mogelijkheden testen**

Versie	Datum	Auteur	Opmerkingen
0.1	6 mrt 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	29 apr 2014	Jeroen de Meij	De functie debugOutput veranderd in debugOut en testgeval 5 verwijderd

---

TESTOBJECT EN ACHTERGROND

In deze test wordt bekeken of objecten debug berichten kunnen versturen. Hierbij worden verschillende standen van de debugger getest.

TESTMISSIE

Testen of de debug functionaliteit naar behoren werkt.

TESTBASIS

1. Als testbasis dient de eis dat het systeem onderhoudbaar moet zijn, waarbij het duidelijk moet zijn wat het systeem doet. Deze eis is opgenomen onder het kopje maintainability van het requirements document.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid, volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. In de constructor van de Motor moeten de volgende statements worden op genomen:

- debugOut(DB_MESSAGE, “Debug messages”);
- debugOut(DB_WARNING, “Debug waring”);
- debugOut(DB_ERROR, “Debug error”);

TESTDOELEN MET TESTAANWIJZINGEN

- HET SYSTEEM MOET ONDERHOUDBAAR ZIJN

Om ervoor te zorgen dat het systeem onderhoudbaar is, moet worden getest of het mogelijk is debug output te generen.

NIET TESTEN

Er hoeft niet te worden getest of de aansturing van de motoren werkt. Het gaat bij deze test alleen om het testen van de juistheid van de debug output.

TESTGEVALLEN

GEVAL 1:

- Maak een Motor object aan en zet de Debug op DB_MESSAGES, door middel van het volgende statement: Motor M = Motor(DB_MESSAGES, 1, 1, 1, 1, 1, 1);

Verwacht resultaat: Op het scherm moeten de volgende drie berichten worden getoond:

- Debug messages
- Debug waring
- Debug error

GEVAL 2:

- Maak een Motor object aan en zet de Debug op DB_WARNING, door middel van het volgende statement: Motor M = Motor(DB_WARNING, 1, 1, 1, 1, 1, 1);

Verwacht resultaat: Op het scherm moeten de volgende twee berichten worden getoond:

- Debug waring
- Debug error

GEVAL 3:

- Maak een Motor object aan en zet de Debug op DB_ERROR, door middel van het volgende statement: Motor M = Motor(DB_ERROR, 1, 1, 1, 1, 1, 1);

Verwacht resultaat: Op het scherm moet het volgende bericht worden getoond:

- Debug error

GEVAL 4:

- Maak een Motor object aan en zet de Debug op DB_OFF, door middel van het volgende statement: Motor M = Motor(DB_OFF, 1, 1, 1, 1, 1, 1);

Verwacht resultaat: In tegenstelling tot de andere testgevallen mogen er geen debug berichten worden getoond.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **I2**

Omschrijving **De communicatie met sensor wordt getest**

Versie	Datum	Auteur	Opmerkingen
0.1	6 mrt 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	29 apr 2014	Jeroen de Meij	Testgeval 1 ingevuld en 2 verwijderd

TESTOBJECT EN ACHTERGROND

In deze test wordt bekeken of de communicatie met de sensor werkt. Hierbij wordt de sensor geconfigureerd en worden de gegevens uit de sensor uitgelezen en door de IMU bewerkt.

TESTMISSIE

Testen of de communicatie met de sensor werkt.

TESTBASIS

1. Als testbasis dient de eis dat het systeem de MPU6000 moet kunnen uitlezen en daarmee de draaiing kunnen bepalen van de roll en pitch as van de multicopter. Deze eis is opgenomen onder het kopje functionele eisen van het requirements document.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid, volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. De sensor en de twee motoren moeten op de Teensy zijn aangesloten.

TESTDOELEN MET TESTAANWIJZINGEN

- DE SENSOR WORDT JUIST UITGELEZEN

Als eerste stap moet het duidelijk zijn dat de sensor zelf goed wordt uitgelezen. Wanneer dit niet het geval is worden de berekeningen uitgevoerd met foutieve waarden.

- DE JUISTE BEREKENINGEN WORDEN OP DE SENSOR UITGEVOERD

Nadat zeker is dat de sensor juiste informatie binnenkrijgt, moet worden gekeken of vervolgens de berekeningen die op de waarden worden uitgevoerd wel juist zijn.

NIET TESTEN

Tijdens deze test hoeft niet te worden gekeken of de snelheid waarmee de sensor wordt uitgelezen wel correct is. Het gaat in dit geval alleen om de juistheid van de output.

TESTGEVALLEN

- Maak een Controller aan, maar zorg ervoor dat de controller alleen een Imu aanmaakt en geen camera of motorcontroller. Laat vervolgens de loop() functie met de juiste tempo lopen. Laat eens per seconden de hoeken uit de IMU op het scherm printen.

Verwacht resultaat: Op het scherm worden lopen de hoeken naar de waarde die de sensor op dat moment heeft.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **I3**

Omschrijving **De communicatie met motoren wordt getest**

Versie	Datum	Auteur	Opmerkingen
0.1	6 mrt 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	29 apr 2014	Jeroen de Meij	Testgevallen 1 en 2 ingevuld en 3 toegevoegd

TESTOBJECT EN ACHTERGROND

In deze test wordt bekeken of de communicatie met de motoren werkt. Hierbij worden de motoren geconfigureerd en worden de motoren aangestuurd. Eerst door middel van vooraf ingestelde vaste waarde. Vervolgens door middel van uitgelezen waarden uit de sensoren.

TESTMISSIE

Testen of de communicatie met de motoren werkt.

TESTBASIS

1. Als testbasis dient de eis dat het systeem de bewegingen van de helikopter moet kunnen tegengaan door een gymbal met twee brushless motoren aan te sturen. Deze eis is opgenomen onder het kopje functionele eisen van het requirements document.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid, volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het testplan.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. De sensor en de twee motoren moeten op de Teensy zijn aangesloten.

TESTDOELEN MET TESTAANWIJZINGEN

- DE MOTOREN KUNNEN WORDEN AANGESTUURD

Als eerste stap moet het duidelijk zijn dat de motoren zelf kunnen worden aangestuurd.

- DE MOTOREN KUNNEN IN DE GEWENSTE POSITIE WORDEN GEZET

Nadat zeker is dat de motoren kunnen worden aangestuurd, moet worden gekeken of ze in de juiste stand zijn te plaatsten met onder andere de output van de sensor berekeningen.

NIET TESTEN

Tijdens deze test hoeft niet te worden gekeken of de snelheid waarmee de motoren worden aangestuurd wel correct is. Het gaat in dit geval alleen om de juistheid van de output.

TESTGEVALLEN

GEVAL 1:

- Maak een controller aan en zorg ervoor dat de motortest wordt uitgevoerd.

Verwacht resultaat: Als eerste moet de roll motor een stukje heen en weer bewegen en vervolgens de pitch motor.

GEVAL 2:

- Maak een controller aan en zorg ervoor dat de motoren reageren op de sensor input.

Verwacht resultaat: De camera moet gestabiliseerd worden

GEVAL 3:

- Maak een controller aan en zorg ervoor dat de motoren reageren op de sensor input.
- Zet na 10 seconden de motoren uit

Verwacht resultaat: De camera moet gestabiliseerd worden en na 10 seconden moeten de motoren stilvallen.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **I4**

Omschrijving **De communicatie met de EEPROM wordt getest**

Versie	Datum	Auteur	Opmerkingen
0.1	28 mrt 2014	Jeroen de Meij	Eerste versie van het scenario
0.2	28 april 2014	Jeroen de Meij	Uitgangssituatie&randvoorwaarden uitgebreid en testgeval aangepast

TESTOBJECT EN ACHTERGROND

Het moet mogelijk zijn om gegevens weg te schrijven in de EEPROM en deze vervolgens weer uit te lezen.

TESTMISSIE

Test of het wegschrijven en uitlezen van de EEPROM werkt.

TESTBASIS

1. Deze test is gebaseerd op de functionele eisen, dat het mogelijk moet zijn om variabelen in de EEPROM te schrijven en dat het stabilisatie systeem deze variabelen moet kunnen uitlezen, vastgesteld in het ontwerp document van het subsysteem, variabelen aanpassen.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid en volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het ontwerp document van het subsysteem, variabelen aanpassen.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. De sensor en de twee motoren moeten op de Teensy zijn aangesloten. De volgende code moet aan de controller worden toegevoegd:

- debugOut(DB_MESSAGE, “Debug messages”);
- debugOut(DB_WARNING, “Debug waring”);
- debugOut(DB_ERROR, “Debug error”);

TESTDOELEN MET TESTAANWIJZINGEN

- DE EEPROM MOET AANSPREEKBAAR ZIJN

Het moet mogelijk zijn om gegevens in de eeprom weg te schrijven en deze vervolgens weer uit te lezen.

NIET TESTEN

-

- Schijf met het extra programma weg naar de EEPROM dat de debugmodus op DB_WARNING moet staan.
- Start het de stabilisatie systeem.
- Schijf vervolgens met het extra programma weg naar de EEPROM dat de debugmodus op DB_MESSAGE moet staan.
- Start het de stabilisatie systeem.

Verwacht resultaat: De eerste keer moeten allen de warning en error debug berichten worden getoond, maar de 2e keer moeten ze alle drie worden getoond

ADMINISTRATIEVE GEGEVENS

Scenario-ID **I5**

Omschrijving **De seriële communicatie voor de pitch hoek wordt getest**

Versie	Datum	Auteur	Opmerkingen
0.1	1 apr 2014	Jeroen de Meij	Eerste versie van het scenario

TESTOBJECT EN ACHTERGROND

Het moet mogelijk zijn om verschillende waarden door te sturen naar de Teensy, die ze vervolgens omzet in de juiste hoek.

TESTMISSIE

Test of het aanpassen van de pitch hoek werkt.

TESTBASIS

1. Deze test is gebaseerd op de functionele eisen, dat het mogelijk moet zijn om via de seriële communicatie waarden door te zetten, die worden omgezet in de hoek van de pitch motor, vastgesteld in het ontwerp document van de uitbreiding, pitch hoek aanpassen.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid en volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het ontwerp document van het subsysteem, variabelen aanpassen.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. Op de arduino moet het juiste programma voor deze test draaien. Daarnaast moeten de tx en rx porten van de Arduino en Teensy juist met elkaar verbonden zijn. TX Teensy – RX Arduino en RX Teensy – TX Arduino.

TESTDOELEN MET TESTAANWIJZINGEN

- DE SERIELE VERBINDING MOET WERKEN

Het moet mogelijk zijn om gegevens via de seriële port naar de Teensy te sturen.

- DE GEGEVENS MOETEN JUIST WORDEN VERWERKT

De gegevens die naar de Teensy worden verzonden moeten naar de juiste hoek worden omgezet, wat resulteert in de pitch hoek van de motor die wordt aangepast.

NIET TESTEN

Er hoeft niet te worden getest of het systeem makkelijk te bedienen is en snel genoeg werkt.

TESTGEVALLEN

- Zet eerst de Teensy aan en wacht totdat deze volledig is opgestart.
- Zet vervolgens de Arduino aan die een aantal waarden tussen de 0 en 4095 doorstuurt met een paar seconden rust ertussen.

Verwacht resultaat: De pitch motor neemt de juiste stand aan.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **I6**

Omschrijving **De seriële communicatie voor de camera wordt getest**

Versie	Datum	Auteur	Opmerkingen
0.1	7 apr 2014	Jeroen de Meij	Eerste versie van het scenario

TESTOBJECT EN ACHTERGROND

Het moet mogelijk zijn om verschillende waarden door te sturen naar de Teensy om de film modus van de camera te kunnen bepalen en de infrarood modus te kunnen kalibreren

TESTMISSIE

Test of het de film modus is in te stellen en de infrarood kalibratie werkt.

TESTBASIS

1. Deze test is gebaseerd op de functionele eisen, die stellen dat de gebruiker moet kunnen switchen tussen video stroom 1 en 2 en de infrarood camera moet kunnen kalibreren, vastgesteld in het ontwerp document van de uitbreiding, camera aansturing.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid en volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het ontwerp document van het subsysteem, variabelen aanpassen.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. Op de Arduino moet het juiste programma voor deze test draaien. Daarnaast moeten de TX en RX porten van de Arduino en Teensy juist met elkaar verbonden zijn. TX Teensy – RX Arduino en RX Teensy – TX Arduino. Tot slot moet de video stroom op de pc zichtbaar worden gemaakt.

TESTDOELEN MET TESTAANWIJZINGEN

- DE VIDEO MODUS MOET WORDEN BEDIEND

Het moet mogelijk zijn om via de seriële port gegevens naar de Teensy te sturen die ervoor zorgen dat tussen de twee verschillende video modussen kan worden geswitcht.

- DE INFRAROOD CAMERA MOET WORDEN GEKALIBREERD

Het moet mogelijk zijn om via de seriële port gegevens naar de Teensy te sturen die ervoor zorgen dat de infrarood kan worden geswitcht.

NIET TESTEN

Er hoeft niet te worden getest of het systeem makkelijk te bedienen is en snel genoeg werkt.

TESTGEVALLEN

Geval 1:

- Zet eerst de camera aan en daarna de Teensy en wacht totdat deze volledig is opgestart. Zorg ervoor dat de camera op de normale video modus staat.
- Zet vervolgens de Arduino aan die eerst het signaal stuurt om de camera naar de infrarood modus te zetten. Daarna moet 10 seconden worden gewacht voordat de Arduino het signaal stuurt om weer naar de normale videomodus over te gaan.

Verwacht resultaat: De camera moet eerst van de normale videomodus naar de infrarood modus overgaan en na 10 seconden weer terug naar de normale videomodus gaan.

Geval 2:

- Zet eerst de camera aan en daarna de Teensy en wacht totdat deze volledig is opgestart. Zorg ervoor dat de camera op de normale video modus staat.
- Zet vervolgens de Arduino aan die het signaal stuurt om de infrarood functie te kalibreren.

Verwacht resultaat: De camera moet eerst van de normale videomodus naar de infrarood modus overgaan, vervolgens omhoog draaien, zodat deze naar de onderkant van de helikopter zou kijken, vervolgens moet de camera hierop vast blijven staan en het signaal krijgen om de infrarood te kalibreren. Wanneer de camera gekalibreerd is moet het systeem weer terug gaan naar de oorspronkelijke stand.

ADMINISTRATIEVE GEGEVENS

Scenario-ID **I7**

Omschrijving **De logging mogelijkheid wordt getest**

Versie	Datum	Auteur	Opmerkingen
0.1	14 apr 2014	Jeroen de Meij	Eerste versie van het scenario

TESTOBJECT EN ACHTERGROND

Het moet mogelijk zijn om bepaalde informatie weg te schrijven in een logfile op een MicroSD kaart.

TESTMISSIE

Test of gegevens in de logfile kunnen worden geschreven en of het beheren van de logfile op de SD kaart goed werkt.

TESTBASIS

1. Deze test is gebaseerd op de functionele eisen, dat het mogelijk moet zijn om real-time via SPI een comma seperated file op de MicroSD kaart te zetten, de file op de computer uit te lezen en wanneer er al een file op de MicroSD kaart staat, moet er tweede logfile worden aangemaakt, wanneer er al een tweede is een derde, enzovoorts.
2. Daarnaast is deze test gebaseerd op de kwaliteitseisen geschiktheid, juistheid en volwassenheid van het systeem. Te vinden in de PRIMA, opgenomen in het ontwerp document van het subsysteem, variabelen aanpassen.

UITGANGSSITUATIE EN RANDVOORWAARDEN

De Teensy moet met de USB kabel aan de pc zijn verbonden. De MicroSD Card Apdapter moet via SPI op de Teensy zijn aangesloten met de SC op pin 14.

TESTDOELEN MET TESTAANWIJZINGEN

- DE GEGEVENS MOETEN WORDEN GELOGD

Het moet mogelijk zijn om gegevens via SPI naar de SD kaart te schrijven en daar in een file op te slaan.

- DE FILE BEHEER MOET GOED WERKEN

De gegevens die naar de Teensy worden verzonden moeten naar de juiste hoek worden omgezet, wat resulteert in de pitch hoek van de motor die wordt aangepast.

NIET TESTEN

Er hoeft niet te worden getest of het systeem makkelijk te bedienen is en snel genoeg werkt.

TESTGEVALLEN

Geval 1:

- Zet de Teensy aan en schrijf een bepaalde regel naar een logfile.
- Koppel de SD kaart aan de computer.

Verwacht resultaat: Er staat een logfile op de SD kaart.

Geval 2:

- Zet de Teensy aan en schrijf een bepaalde regel naar een logfile.
- Zet de Teensy uit en weer aan en schrijf een bepaalde regel naar een logfile.
- Koppel de SD kaart aan de computer.

Verwacht resultaat: Er staat twee logfiles op de SD kaart.

Geval 3:

- Zet de Teensy aan en schrijf een bepaalde regel naar een logfile.
- Koppel de SD kaart aan de computer en verwijder de logfile.
- Zet de Teensy uit en weer aan en schrijf een bepaalde regel naar een logfile.
- Koppel de SD kaart aan de computer.

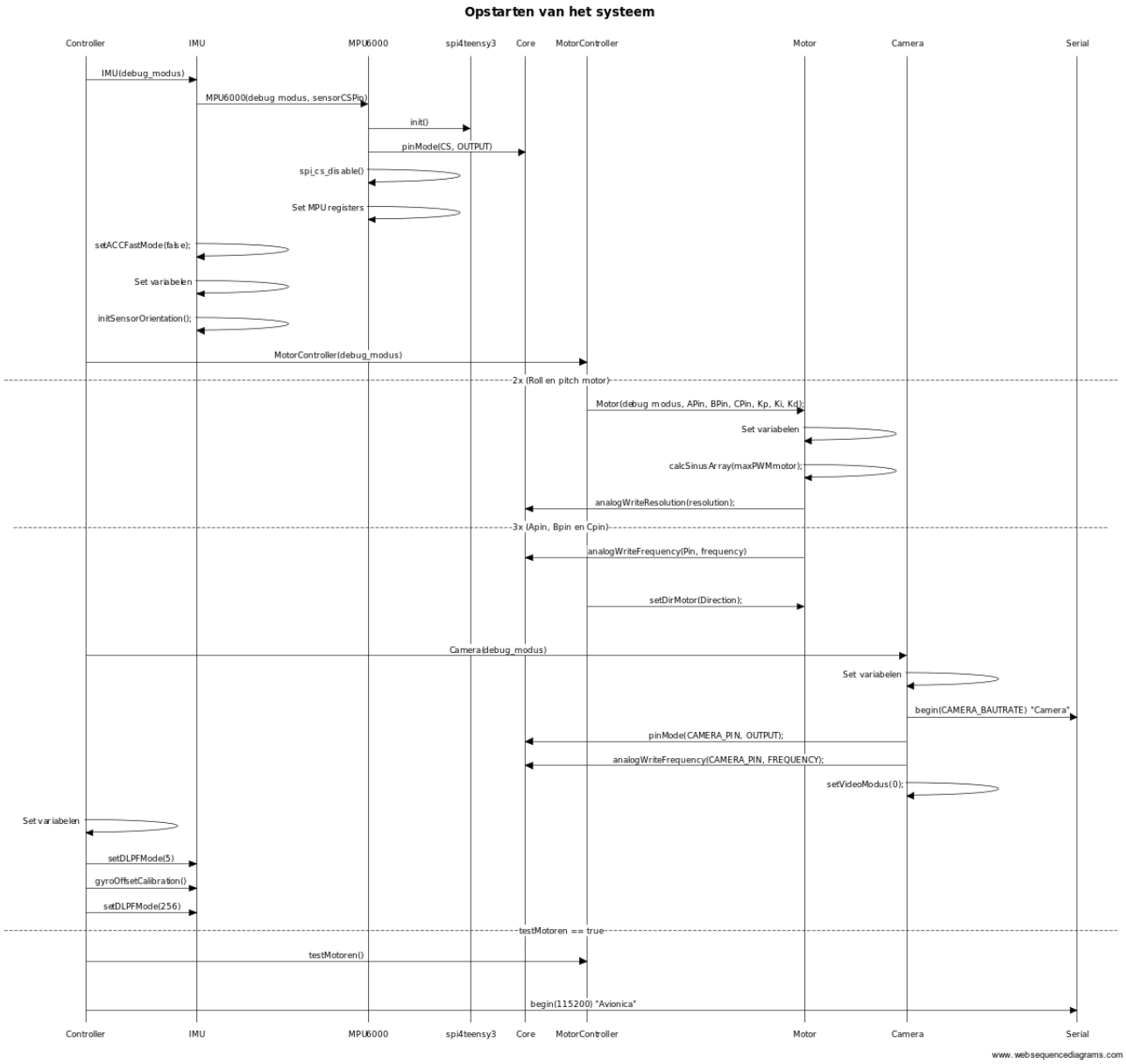
Verwacht resultaat: Er staat één logfile op de SD kaart.

Bijlage M: Testresultaten

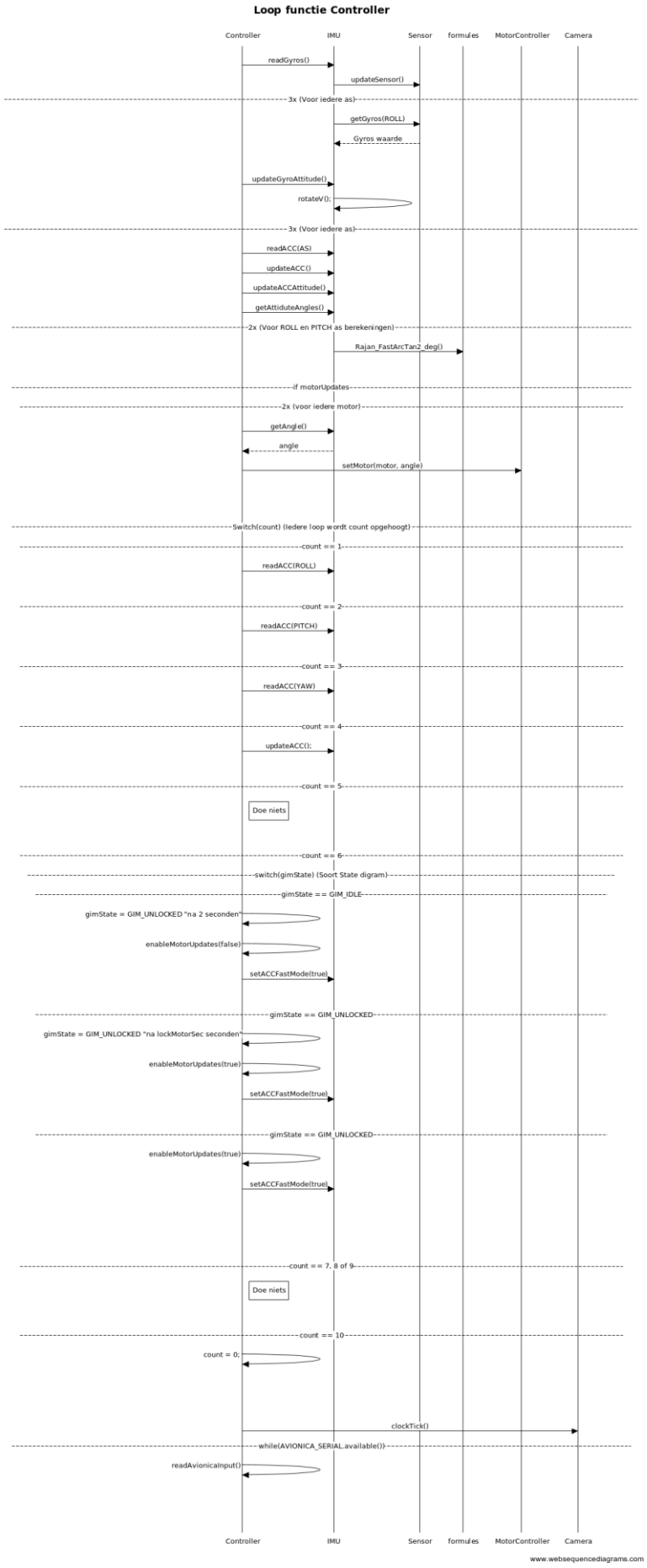
Scenario-ID	Laatst uitgevoerde datum	Huidige status
AF1-1	28-5-2014	geslaagd
AF1-2	XX-XX-XXXX	NIET UITGEVOERD
AF1-3	XX-XX-XXXX	NIET UITGEVOERD
AG1-1	28-5-2014	geslaagd
AG1-2	XX-XX-XXXX	NIET UITGEVOERD
AG1-3	28-5-2014	geslaagd
AG1-4	XX-XX-XXXX	NIET UITGEVOERD
AG1-5	XX-XX-XXXX	NIET UITGEVOERD
AG1-6	XX-XX-XXXX	NIET UITGEVOERD
AP1	XX-XX-XXXX	NIET UITGEVOERD
AP2-1	XX-XX-XXXX	NIET UITGEVOERD
AP2-2	XX-XX-XXXX	NIET UITGEVOERD
AC1	XX-XX-XXXX	NIET UITGEVOERD
AC2	XX-XX-XXXX	NIET UITGEVOERD
S1	14-05-14	geslaagd
S2	14-05-14	geslaagd
S3	22-05-14	redelijk
S4-1	14-05-14	geslaagd
S4-2	14-05-14	geslaagd
S5	22-05-14	geslaagd
S6	22-05-14	geslaagd
S7-1	22-05-14	geslaagd
S7-2	22-05-14	geslaagd
S7-3	22-05-14	geslaagd
I1-1	29-04-14	geslaagd
I1-2	29-04-14	geslaagd
I1-3	29-04-14	geslaagd
I1-4	29-04-14	geslaagd
I2	29-04-14	geslaagd
I3-1	29-04-14	geslaagd
I3-2	29-04-14	geslaagd
I3-3	29-04-14	geslaagd
I4	29-04-14	geslaagd
I5	29-04-14	geslaagd
I6-1	22-05-14	geslaagd
I6-2	05-05-14	geslaagd
I7-1	XX-XX-XXXX	NIET UITGEVOERD
I7-2	XX-XX-XXXX	NIET UITGEVOERD
I7-3	XX-XX-XXXX	NIET UITGEVOERD

Bijlage N: Sequentie diagrammen

Opstarten van het systeem



Loop functie uit de Controller



Camera kalibreren