Procesverslag

Ontwikkeling van software om een trigger box aan te sturen

Tom Conijn

Procesverslag

Ontwikkeling van software om een trigger box aan te sturen

Studie: Technische Informatica, Delft
School: Haagse Hogeschool
Startdatum: 10 februari 2014
Einddatum: 6 juni 2014
Begeleidende docenten: Anthony van Geest & Tony Andrioli
Begeleiders Brightsight: Rob Bekkers & Remko Foekema
Stagiair: Tom Conijn, 10010017

Referaat

Tom Conijn, “Ontwikkeling van software om een trigger box aan te sturen”. Afstudeerverslag opleiding Technische Informatica, Haagse Hogeschool, 2014.

Dit verslag beschrijft de stappen die zijn uitgevoerd tijdens de afstudeeropdracht voor het bedrijf Brightsight te Delft. Tijdens deze opdracht is er gebruik gemaakt van Scrum om de werkzaamheden te plannen. Hierdoor zijn er drie sprints ontstaan waarin er gewerkt wordt aan de software om de trigger box aan te sturen. In de eerste twee sprints wordt de functionaliteit toegevoegd om de trigger box aan te sturen vanuit Java. In de derde sprint wordt gewerkt aan het verbeteren van de gebruikersinterface. Hierna zijn er twee weken over waaraan gewerkt wordt aan de afronding van dit project.

Descriptoren:

- Afstudeerverslag
- Afstudeeropdracht
- Java
- Trigger box
- Scrum
- JNA
- Softwareontwikkeling

Voorwoord

Dit verslag is geschreven naar aanleiding van mijn afstudeeropdracht die ik als student Technische Informatica aan de Haagse Hogeschool heb uitgevoerd. Ik heb deze opdracht uitgevoerd bij Brightsight van 10 februari 2014 tot 6 juni 2014. Met dit verslag wil ik aantonen dat ik werkzaamheden kan verrichten op HBO-niveau.

Het verslag is geschreven voor mensen die enige kennis hebben van programmeertalen en UML-diagrammen. Wanneer de lezer kort wil weten wat er per sprint(iteratie) is uitgevoerd is het mogelijk om de samenvatting per sprint te lezen.

Ik bedank graag iedereen die mij heeft geholpen met het uitvoeren van deze afstudeeropdracht, met name Rob Bekkers en Remko Foekema voor het begeleiden binnen Brightsight. Ook bedank ik de heren A. van Geest en A. Andrioli voor hun begeleiding vanuit de Haagse Hogeschool.

Schiedam, donderdag 5 juni 2014

Tom Conijn

Samenvatting

Tijdens het afstudeerproject is er gewerkt aan het maken van software om een trigger box vanuit het interne ontwikkelde softwaresysteem, de Matrix, aan te sturen. Deze trigger box kan een serie van waardes herkennen in het stroomverbruik van een chip en vervolgens een trigger genereren.

Om dit project te structureren is de ontwikkelmethode Scrum gekozen. De keuze voor Scrum heeft de structuur van dit verslag beïnvloed. Per sprint(iteratie) is een hoofdstuk geschreven met daarin uitleg van de belangrijke activiteiten.

Scrum is gekozen tijdens het uitvoeren van een oriëntatiefase. In deze fase is het project nader geanalyseerd en zijn de eisen van de opdrachtgever verduidelijkt. Vervolgens is in drie sprints de software voor het aansturen van de trigger box geschreven. De eerste twee sprints zijn gericht op het toevoegen van de basisfunctionaliteit en in de derde sprint is de gebruikersinterface ontwikkeld.

Na het uitvoeren van de drie sprints zijn er nog twee weken over, waarin een handleiding wordt geschreven over het gebruik van de trigger box en een presentatie voor de werknemers is voorbereid.

Het resultaat van dit afstudeerproject is, dat de trigger box vanuit de Matrix aangestuurd kan worden.

Inhoudsopgave

1 Inleiding

Dit verslag is geschreven naar aanleiding van een afstudeeropdracht die is uitgevoerd als afsluiting van de opleiding Technische Informatica aan de Haagse Hogeschool. De opdracht is uitgevoerd voor het bedrijf Brightsight te Delft.

Brightsight is een bedrijf gespecialiseerd in het uitvoeren van beveiligingsevaluaties op producten zoals smart card software, betalingsautomaten en geïntegreerde schakelingen. Voor het aansturen van testopstellingen wordt gebruik gemaakt van het intern ontwikkelde softwarepakket, de Matrix.

Op basis van metingen die uitgevoerd worden in de testopstelling kunnen er verschillende acties gestart worden als het gemeten signaal boven een vooraf ingestelde waarde komt. In de praktijk blijkt deze methode niet nauwkeurig genoeg om de acties op het goede moment te starten. Om dit probleem op te lossen is een meetinstrument aangeschaft, de icWaves trigger box, waarmee een actie gestart kan worden als het gemeten signaal overeenkomt met een serie van opgeslagen waardes.

Het doel van dit project is het ontwikkelen van een module voor de Matrix waarmee de icWaves trigger box aangestuurd kan worden. Hierbij kan een onderverdeling gemaakt worden in drie belangrijke functies, namelijk: het opnemen van een signaal, het selecteren van een serie opgeslagen waardes en het detecteren van de opgeslagen waardes in een volgende meting.

Dit verslag is verdeeld in drie onderdelen. Het eerste onderdeel bevat algemene informatie over dit project, zoals de werkomgeving, de opdracht en de aanpak. Vervolgens bevat het tweede onderdeel de ontwikkeling van de module, dit bestaat uit drie iteraties waarin de module is gemaakt. Het laatste onderdeel behandelt de afsluiting van dit project, met de conclusie en de evaluatie.

2 Werkomgeving

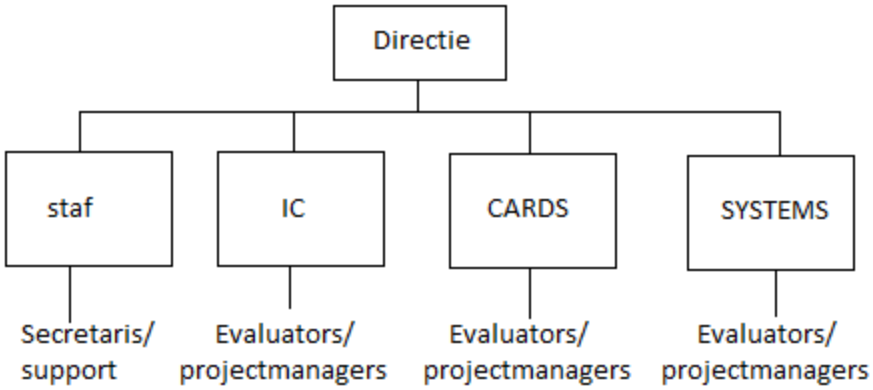
In dit hoofdstuk wordt de omgeving beschreven waarin de opdracht wordt uitgevoerd.

2.1 Bedrijfsomschrijving

Brightsight is een bedrijf gespecialiseerd in het uitvoeren van beveiligingsevaluaties op producten zoals smart card software, betalingsautomaten en geïntegreerde schakelingen. Brightsight is het grootste onafhankelijk evaluatie laboratorium in de wereld en heeft de naam Brightsight sinds 2006. Voordat het bedrijf de naam Brightsight heeft gekregen is het onderdeel geweest van TNO. Het bedrijf is gevestigd in het Delftechpark in Delft.

Brightsight staat in de top vijf van internationale bedrijven die beveiligingsevaluaties uitvoeren. Hierdoor komen opdrachten voor evaluaties uit verschillende delen van de wereld. Om ervoor te zorgen dat er goed met deze buitenlandse klanten gecommuniceerd kan worden is er bij Brightsight personeel uit verschillende landen aangenomen. Dit heeft als nadeel dat niet iedereen gemakkelijk Nederlands spreekt en daarom wordt er vooral in het Engels gecommuniceerd. Hierdoor moet documentatie in het Engels geschreven worden.

2.2 Plaats in de organisatie



Afbeelding 1 Organogram

Brightsight is als organisatie een plat bedrijf. Dit is ook te zien in Afbeelding 1. Brightsight bestaat uit een directie van twee personen. Vervolgens zitten er vier groepen onder de directie. De staf groep zorgt voor de ondersteuning en bestaat uit het secretariaat en de ICT afdeling. De andere drie groepen richten zich op het testen en evalueren van verschillende producten. In het organogram is geen aparte groep beschikbaar voor softwareontwikkeling. Dit project wordt uitgevoerd onder de Integrated Circuits(IC) groep, omdat de begeleider deel uitmaakt van deze groep. Doordat het een informele organisatie is en de begeleider op dezelfde kamer zit als de student is het gemakkelijk om vragen te stellen. Dit zorgt ervoor dat er afspraken gemaakt kunnen worden, zonder dat er interviews gehouden worden.

2.3 Opdrachtgever

Rob Bekkers is de contactpersoon namens de opdrachtgever voor dit project. Hij is senior Security Evaluator bij Brightsight.

Remko Foekema is de persoon binnen Brightsight die de technische kennis heeft over de werking van het softwarepakket dat gebruikt wordt. Dit betekent dat er tijdens het ontwikkelen van de module vooral overlegd wordt met Remko.

3 De opdracht

3.1 Probleemstelling

Brightsight gebruikt verschillende testopstellingen voor het uitvoeren van penetratietesten op producten die beveiligd zijn met cryptografie. Al deze testopstellingen worden aangestuurd met hetzelfde softwarepakket dat door Brightsight is ontwikkeld. Dit softwarepakket wordt de Matrix genoemd.

In een eenvoudige opstelling voor het testen van een secure microcontroller wordt gebruik gemaakt van drie verschillende apparaten, namelijk een oscilloscoop, de te testen microcontroller en een laser. Hierbij wordt het stroomverbruik van de microcontroller gemeten met de oscilloscoop. Op het moment dat de microcontroller een cryptografische berekening uitvoert heeft dit een verandering in het stroomverbruik tot gevolg.

Op het moment dat het stroomverbruik een vooraf bepaalde drempelwaarde overschrijdt, wordt een signaal naar de laser gestuurd om een fout te injecteren in de microcontroller. Deze overschrijding kan echter ook het gevolg zijn van verstoringen in het meetsignaal, waardoor de laser niet op het juiste moment wordt aangestuurd.

Om nauwkeuriger het moment te kunnen bepalen waarop de laser aangestuurd moet worden, is een apparaat aangeschaft waarmee een serie van opgeslagen meetwaarde herkend kan worden. Dit apparaat is de icWaves trigger box, die speciaal voor dit doel ontwikkeld is door de firma Riscure.

3.2 Doelstelling

Het doel van dit project is het ontwikkelen van een module waarmee de icWaves trigger box, vanuit het intern ontwikkelde softwarepakket de Matrix, aangestuurd kan worden.

Dit betekent dat het mogelijk moet zijn om de volgende acties uit te voeren vanuit de Matrix:

- Er moet een signaal opgenomen worden met de icWaves. Dit kan bijvoorbeeld het stroomverbruik zijn van een microcontroller die getest wordt.

- Er moet een patroon herkend worden, dat bestaat uit een aantal opeenvolgende meetpunten die geselecteerd worden uit het opgenomen signaal.

- Het opgeslagen signaal moet herkend worden, waarna een signaal gegenereerd wordt. Het signaal dat gegenereerd wordt kan bijvoorbeeld een laser laten weten dat een fout geïnjecteerd moet worden.

3.3 Resultaat

Aan het eind van de afstudeeropdracht kan de icWaves trigger box vanuit de Matrix aangestuurd worden. Hierdoor is het mogelijk om met een hogere nauwkeurigheid een fout te injecteren, waardoor er met meer zekerheid een advies gegeven kan worden over het beveiligingsniveau van de secure microcontroller.

4 Aanpak

In dit hoofdstuk wordt de keuze van de ontwikkelmethode besproken.

Bij de keuze van een goede ontwikkelmethode moet rekening gehouden worden met één randvoorwaarde van de opdrachtgever. De software moet gemaakt worden in onderdelen, waarbij elk onderdeel als afgerond product, dus getest en gedocumenteerd, opgeleverd kan worden.

Voor het kiezen van de ontwikkelmethode zijn de volgende iteratieve ontwikkelmethodes onderzocht: Rational Unified Process (RUP) [1], Scrum [2], Feature-Driven Development (FDD) [3] en Extreme Programming (XP) [4]. De methodes zijn in tabel 1 gezet en worden vergeleken met eisen die hieraan gesteld zijn. Deze eisen zijn opgesteld aan de hand van de opdracht en de randvoorwaarde van de opdrachtgever.

In de tabel worden onderstaande tekens gebruikt:
+ De methode voldoet aan de eis.
- De methode voldoet niet aan de eis.
~ De methode ondersteunt de eis niet optimaal.

Eigenschappen/eisen	Rup	Scrum	FDD	XP
Iteratieve aanpak: maakt gebruik van verschillende iteraties.	+	+	+	+
Werkend product per iteratie: levert een werkend product af per iteratie.	-	+	-	-
Testen per iteratie: voert testen uit per iteratie.	+	~	+	+
Voortgang bewaking: heeft genoeg momenten gepland waarop de voortgang gecontroleerd wordt.	~	+	~	+
Gedetailleerde planning: heeft een duidelijk overzicht van de taken die uitgevoerd moeten worden tijdens een iteratie.	+	+	+	-
Omgang bij verandering eisen: kan omgaan met veranderingen van eisen	~	+	+	+

tijdens het project.				
Individueel gebruik: is mogelijk om individueel uit te voeren.	~	-	+	-
Documentatie: Heeft genoeg momenten om documentatie te maken.	+	+	~	~

Tabel 1 Vergelijking ontwikkelmethodes

Wanneer er gekeken wordt naar de eisen die gesteld zijn aan de ontwikkelmethodes is te zien dat alle gekozen methodes gebruik maken van iteraties. Bij het vergelijken van de overige eisen komt Scrum het beste in de buurt van een bruikbare methode. Dat komt voornamelijk doordat deze voldoet aan de voorwaarde van de opdrachtgever, terwijl RUP, FDD en XP dit niet doen.

Bij gebruik van Scrum is er een probleem, omdat het project maar door één persoon wordt uitgevoerd. Dit betekent dat de dagelijkse gesprekken die voorgeschreven worden bij Scrum niet uitgevoerd kunnen worden. Dit wordt tijdens het project opgevangen door korte lijnen met zowel de opdrachtgever als de begeleider. De problemen die ontstaan bij de ontwikkeling of het proces kunnen gelijk besproken worden met de begeleider. Door wekelijks overleg is er controle op de voortgang van het project en worden problemen omtrent de planning snel gedetecteerd.

Bij Scrum wordt er gebruik gemaakt van sprints oftewel iteraties met een lengte tussen de één en vier weken. Aan het begin van een sprint worden de taken gekozen die tijdens deze sprint uitgevoerd worden. Hiervoor wordt gebruik gemaakt van een tabel met taken, waarbij per taak een aantal uren zijn gepland. Doordat aan het begin van een sprint taken worden gekozen is het mogelijk om, afhankelijk van de prioriteiten van de opdrachtgever, de volgorde van werken aan te passen.

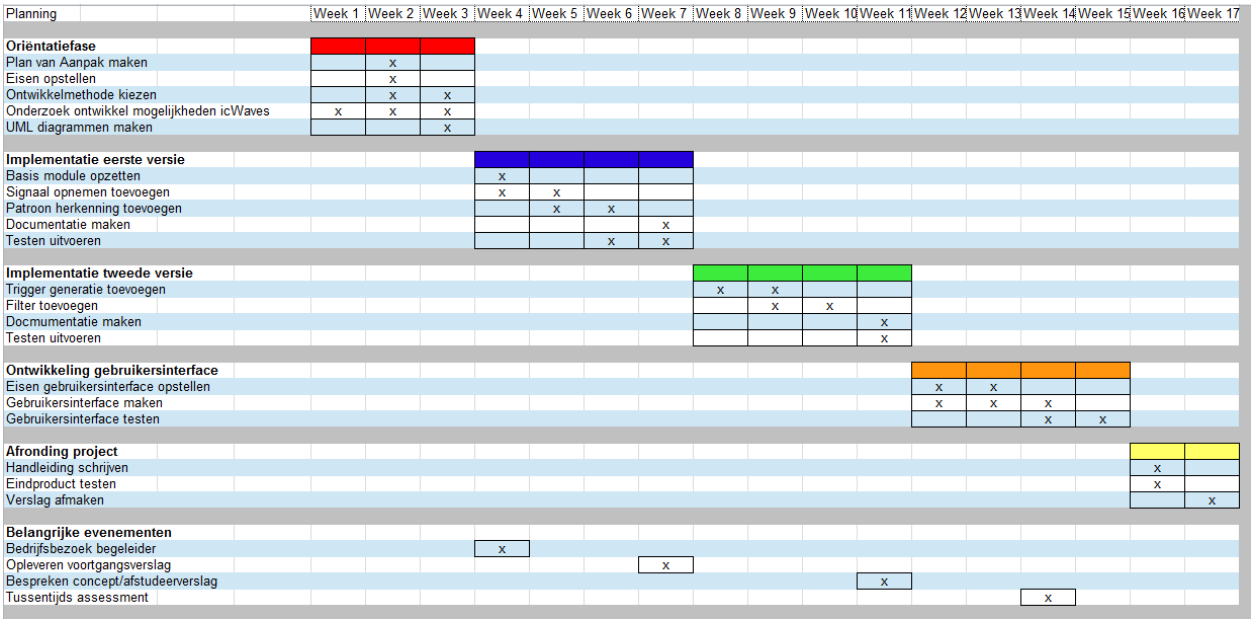
De keuze voor Scrum is gemaakt tijdens de oriëntatiefase die in het volgende hoofdstuk besproken wordt. In deze fase is een globale planning gemaakt waarin staat wat er per sprint uitgevoerd moet worden. Het verslag is vervolgens ingedeeld naar aanleiding van deze planning.

5 Oriëntatiefase

Dit hoofdstuk beschrijft de analyse die is uitgevoerd aan het begin van het project. Dit is tijdens het kiezen van een ontwikkelmethode en het schrijven van het plan van aanpak. Het plan van aanpak is te zien in Bijlage A. Als eerste wordt hieronder de planning besproken, hierna wordt de werking van icWaves uitgelegd en als derde zijn er eisen opgesteld waaraan de, nog te ontwikkelen, module moet voldoen. De ontwikkel pc wordt als vierde onderwerp besproken. Vervolgens is er uitgezocht hoe vanuit Java een C library aangesproken kan worden. Daarna wordt de werking van de Inspector software en een voorbeeldprogramma in C behandeld. Aan het eind van deze fase worden de UML-diagrammen uitgelegd, die zijn gemaakt om een beter beeld te krijgen van de werking van de Matrix.

5.1 Planning

Nadat de keuze op Scrum als ontwikkelmethode is gevallen is onderstaande planning gemaakt waarin globaal beschreven staat wat er per sprint moet gebeuren. Deze sprints worden per hoofdstuk in dit verslag besproken. Voor de takenlijsten die zijn gemaakt per sprint kan Bijlage B bekeken worden.



Afbeelding 2 Globale planning

In de planning is te zien dat er drie weken gereserveerd zijn voor de oriëntatiefase. De werkzaamheden van deze fase worden besproken in dit hoofdstuk. Na deze drie weken zijn er drie sprints gepland, waarin gewerkt wordt aan de eerste versie van de module, de tweede versie van de module en de ontwikkeling van de gebruikersinterface. De laatste twee weken zijn geen onderdeel van een sprint, omdat deze bedoeld zijn om het project af te ronden en een handleiding te schrijven over het gebruik van de icWaves trigger box.

De planning die gemaakt is gaat uit van drie sprints van elk vier weken. Er is gekozen voor sprints van vier weken, omdat nog niet alle werkzaamheden bekend zijn en er genoeg tijd

uitgetrokken moet worden om zowel de implementatie als het testen uit te voeren. Wanneer er gebruik wordt gemaakt van sprints van vier weken is er meer tijd om eisen helder te krijgen en de implementatie uit te voeren. Bij dit project is het van belang dat er een langere tijd per sprint beschikbaar is. Vooral bij de implementatie is de kans groot dat er problemen optreden die meer tijd kosten dan vooraf verwacht. Bij een tweewekelijkse sprint kunnen problemen bij de implementatie het eindproduct in gevaar brengen, omdat er te weinig tijd over is om te testen of documentatie te schrijven.

In de planning is gekozen om in de eerste sprint te werken aan het opzetten van de basis voor de module, het opnemen van een signaal en het herkennen van een patroon. Het opslaan van een signaal wordt als eerste uitgewerkt, zodat de overige onderdelen dit opgeslagen signaal kunnen gebruiken. Bij de eerste sprint wordt extra tijd uitgetrokken voor het schrijven van documentatie en het kiezen van een teststrategie.

In de tweede sprint kan de overige functionaliteit worden toegevoegd. De belangrijkste onderdelen hiervan zijn het genereren van een triggersignaal en het instellen van de filter. Dit kan in dezelfde sprint toegevoegd worden, omdat al een basis is opgezet voor de module en het schrijven van schrijven van documentatie en testen kost minder tijd.

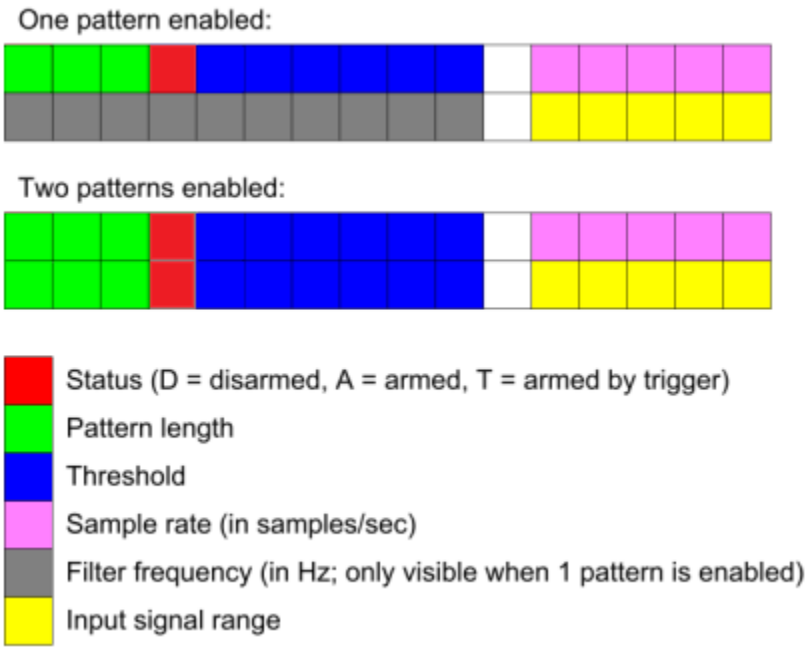
De derde sprint bestaat uit het maken van de gebruikersinterface. Hiervoor is gekozen, omdat het een belangrijk onderdeel is voor het gebruik van de module. Met de huidige planning is hiervoor alle benodigde functionaliteit toegevoegd en moet dit alleen nog beschikbaar worden voor de gebruiker.

5.2 icWaves trigger box

De icWaves is de trigger box [5], die gebruikt wordt tijdens dit project en uiteindelijk aangestuurd wordt vanuit het intern ontwikkelde softwarepakket, de Matrix. De icWaves wordt een trigger box genoemd, omdat het een apparaat is dat een signaal kan genereren op het moment dat een serie opgeslagen waardes is herkend in het ingangssignaal. De serie van opgeslagen waardes wordt ook wel het patroon genoemd dat herkend moet worden.

Afbeelding 3 icWaves trigger box

In Afbeelding 3 is het informatiescherm van de icWaves te zien, waarop de belangrijkste instellingen weergegeven worden. De instellingen veranderen afhankelijk van het aantal patronen dat herkend moet worden door de icWaves. Op het scherm kan een gebruiker direct aflezen wat de bemonsteringssnelheid, de lengte van het opgeslagen patroon en de status van het patronen herkennen is. Staat bij de status een A of T dan is de icWaves patronen aan het herkennen, maar staat er een D dan wordt niets herkend. De indeling van het scherm is te zien in Afbeelding 4



Afbeelding 4 icWaves informatiescherm

5.3 Eisen opstellen

Aan de hand van een gesprek met de opdrachtgever en de begeleider is aan het begin van het project duidelijk geworden dat de opdrachtgever zo optimaal mogelijk gebruik wil maken van de icWaves. Dit betekent dat het mogelijk moet zijn om alle functionaliteiten van de icWaves te gebruiken vanuit de module voor de Matrix.

Om een beter beeld te krijgen wat alle functionaliteiten zijn die de icWaves aanbied is de meegeleverde documentatie geanalyseerd. Hierbij zit een Excelbestand met een overzicht van functies, die ondersteund worden door de icWaves. Dit overzicht is te zien in Afbeelding 5. Door gebruik te maken van dit bestand en overleg met de opdrachtgever zijn de functionele eisen tot stand gekomen. De niet-functionele eisen zijn vervolgens opgesteld door overleg met de opdrachtgever en informatie dat gevonden is op het interne netwerk.

Related parameters	Target function block	Parameter Setter call	Parameter Getter call
Maximum sampling rate	**N/A**	N/A	icwaves_get_max_sampling_rate()
Time base	**Time Base**	icwaves_set_timebase()	icwaves_get_timebase()
AC/DC coupling	**AD/DC Coupling**	icwaves_set_acdc_switch()	icwaves_get_acdc_switch()
Signal input voltage range	**Input Range Select**	icwaves_set_range()	icwaves_get_range()
Maximum trace length	**N/A**	N/A	icwaves_get_max_trace_length
Acquiring trace	**Acquisition Control**	icwaves_start_acquisition()	icwaves_get_acq_data()
Maximum SAD tree size	**N/A**	N/A	icwaves_get_max_sad_size()
Number of pattern	**Pattern Matching**	icwaves_set_number_of_patterns()	icwaves_get_number_of_patterns()
Threshold	**Pattern Matching**	icwaves_set_threshold()	icwaves_get_threshold()
Hold off time	**Pattern Matching**	icwaves_set_holdoff()	icwaves_get_holdoff()
Pattern count	**Pattern Matching**	icwaves_set_pattern_count()	icwaves_get_pattern_count()
Pattern count timeout	**Pattern Matching**	icwaves_set_pattern_count_timeout()	icwaves_get_pattern_count_timeout()
Arming trigger	**Trigger Module**	icwaves_set_arm()	icwaves_get_arm()
Trigger counter	**Trigger Module**	N/A	icwaves_get_trigger_counter()
Trigger delay	**Trigger Module**	icwaves_set_trigger_delay()	icwaves_get_trigger_delay()
Input signal range	**Filter**	icwaves_set_filter_input_range()	icwaves_get_filter_input_range()
Filter center frequency	**Filter**	icwaves_set_filter_frequency()	icwaves_get_filter_frequency()

Afbeelding 5 Overzicht functies van de icWaves

Hieronder is een lijst te zien van de opgestelde functionele en niet-functionele eisen.

Functionele eisen:

1. De icWaves moet zichtbaar zijn in het instrumentenoverzicht van de Matrix.
2. De module moet alle functies uit Afbeelding 5 kunnen aanroepen op de icWaves.
3. De module moet een patroon kunnen opslaan op de icWaves welke herkend moet worden.
4. Het moet mogelijk zijn om de icWaves aan te sturen met behulp van de Script Engine binnen de Matrix.
5. De module moet de volgende oscilloscoopfuncties ondersteunen.
    a. De icWaves laten wachten op een triggersignaal om een meting te starten.
    b. De opgenomen data vanuit de icWaves als Waveform[1] aanbieden aan de Matrix.
6. De instellingen van de icWaves moeten aangepast kunnen worden met behulp van een eigenschappenscherm.
7. Een patroon moet geselecteerd kunnen worden met behulp van het plot-scherm in de Matrix.

Niet-functionele eisen:

1. De module moet werken binnen de Matrix.
2. De module moet ontwikkeld worden in de taal Java.
3. De module moet ontwikkeld worden in de Netbeans ontwikkelomgeving.
4. De module moet werken op Windows.
5. Code documentatie moet gemaakt worden met Javadoc.
6. Code commentaar moet in het Engels geschreven zijn.
7. Code standaarden van Java Code Conventions moeten gebruikt worden [6].
8. Tests moeten geschreven worden met behulp van JUnit.

[1] Een Waveform is een klasse binnen de Matrix waarin metingen opgeslagen kunnen worden.

5.4 Ontwikkel pc

Bij Brightsight wordt er gebruik gemaakt van twee verschillende pc's tijdens het werken. De eerste pc kan gebruikt worden om te internetten en e-mails te versturen. Op deze pc draait Kubuntu. De tweede pc is de ontwikkel pc. Deze pc is niet aangesloten op het internet, maar wel op het interne netwerk, zodat voorkomen wordt dat bedrijfsinformatie naar de buitenwereld wordt verspreid.

Op het interne netwerk staat de sourcecode van de Matrix en verschillende installatieprogramma's. Wanneer er software geïnstalleerd moet worden die niet beschikbaar is op het interne netwerk kan dit problemen opleveren, want bepaalde programma's hebben een internetverbinding nodig om nog extra informatie op te halen. Hiermee moet rekening gehouden worden tijdens de ontwikkeling van de module en het gebruik van eventuele programma's om de ontwikkeling te ondersteunen.

5.5 JNA en JNI

Om de icWaves aan te sturen tijdens dit project moet er vanuit Java gecommuniceerd worden met een C library. Om de communicatie op te zetten zijn er twee opties uitgezocht, namelijk Java Native Interface (JNI) [7] en Java Native Access (JNA) [8].

JNI is een raamwerk dat ervoor zorgt dat Java code native library functies kan aanroepen en vice versa. Native betekent in deze context dat de libraries geschreven zijn in een andere taal dan Java. Bij het gebruik van JNI moet er zowel Java code als C code geschreven worden. JNI zit in het Java platform en is gratis te gebruiken.

JNA is de tweede optie om vanuit een Java programma native library functies aan te roepen. Er moet hierbij wel opgemerkt worden dat JNA op dit moment nog geen C++ ondersteund. Wanneer er een C++ library gebruikt wordt moet worden overgestapt naar JNI. Bij gebruik van JNA is het onnodig om C-code te schrijven. JNA zit niet in het Java platform, maar valt onder de Apache Software Licence versie 2.0 [9] en daardoor kan het gratis gebruikt worden.

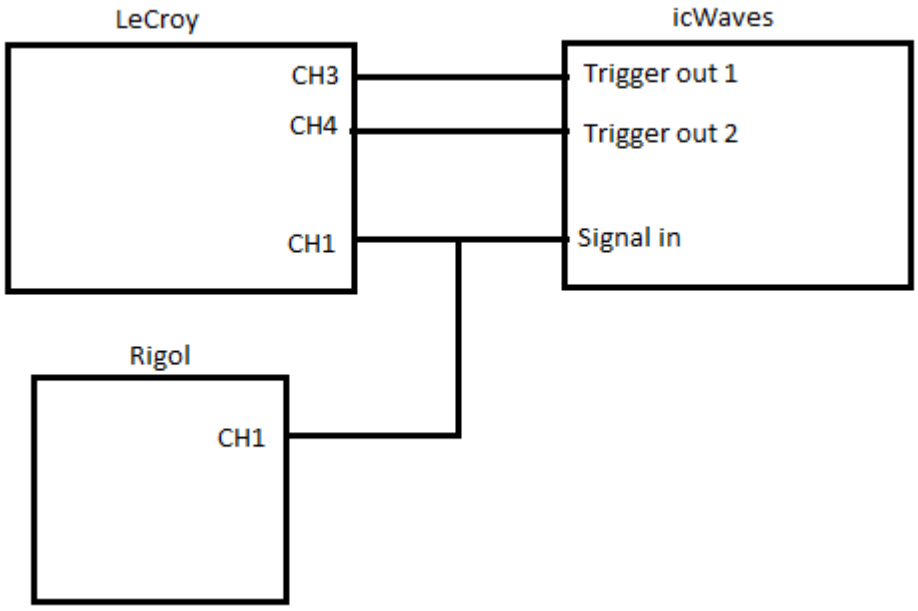
Bij het gebruik van JNA is het niet nodig om C-code te schrijven. Dit bespaart tijd tijdens de ontwikkeling van de module en de kans op fouten tijdens de implementatie is kleiner. De keuze is gevallen op JNA, omdat dit tijd bespaart tijdens de ontwikkeling van de module

5.6 Inspector software en een voorbeeldprogramma in C

Bij de icWaves zijn twee verschillende programma's meegeleverd. Dit is de Inspector software en een voorbeeldprogramma om de icWaves aan te sturen vanuit C. Om een signaal te genereren dat opgenomen moet worden door de icWaves is er gebruik gemaakt van een testopstelling. Deze testopstelling wordt als eerst uitgelegd, vervolgens wordt er gekeken naar het gebruik van de Inspector software en het voorbeeldprogramma. De Inspector software wordt als eerst onderzocht, omdat een gebruikersinterface gemakkelijker te gebruiken is dan code.

5.6.1 Testopstelling

Om de werking van de icWaves te achterhalen is een testopstelling gemaakt die bestaat uit drie apparaten. Het eerste apparaat is de icWaves waarvan de werking onderzocht wordt. Om een signaal te genereren is gebruik gemaakt van een Rigol functiegenerator. Het signaal dat de functiegenerator genereert wordt door een splitter aangeboden aan de icWaves en het laatste apparaat, de LeCroy oscilloscoop. De oscilloscoop kan gebruikt worden om het signaal van de functiegenerator weer te geven. De icWaves beschikt over twee uitgangspoorten, trigger out 1 en trigger out 2, waarover een signaal gestuurd kan worden bij het herkennen van een patroon. Deze poorten zijn aangesloten op de oscilloscoop, zodat het zichtbaar is als een patroon wordt herkend. Zie onderstaande afbeelding voor een schematische weergave van de testopstelling.



Afbeelding 6 Testopstelling oriëntatiefase

5.6.2 Inspector software

De Inspector software is een programma dat gemaakt is door het bedrijf Riscure, de producent van de icWaves. Dit programma wordt gebruikt om verschillende apparaten aan te sturen, waaronder de icWaves.

Bij het onderzoeken van dit programma is de vraag gesteld aan de opdrachtgever waarom er geen gebruik gemaakt wordt van de Inspector software. Er zijn hiervoor twee redenen gegeven. De eerste reden is dat er betaald moet worden voor het gebruik van het programma. De tweede reden is dat er binnen Brightsight gebruik wordt gemaakt van de Matrix en het niet gewenst is om verschillende programma's door elkaar te gebruiken.

Om de werking van de icWaves te onderzoeken met de Inspector software zijn er twee modules geselecteerd. Deze modules zijn gekozen uit de negen modules waarmee data opgenomen kan worden binnen de Inspector software en zijn geselecteerd vanwege de tekst scope of

oscilloscoop in de modulenaam. De overige modules hebben namen die verwijzen naar encryptie methodes, zoals AES of DES.

De eerste module is de oscilloscoopmodule. Hiermee is het mogelijk om verschillende apparaten uit een lijst te selecteren waarmee een signaal opgenomen kan worden, maar de icWaves staat hier niet tussen. Hierdoor valt de module af.

De tweede module is de ScopeAcquisition module. In deze module kan de icWaves geselecteerd worden. Om de module te laten werken moet er een doel worden opgegeven waar een signaal gegeneerd wordt. Hierbij kan gekozen worden uit verschillende trainingskaarten, waarop software staat die cryptografische berekeningen uitvoert. Bij de testopstelling is alleen een functiegenerator beschikbaar, hierdoor is deze module niet geschikt voor de huidige testopstelling.

5.6.3 Voorbeeld C programma

Met behulp van het voorbeeldprogramma wordt de werking van de icWaves onderzocht.

Het programma is gestart in de Netbeans ontwikkelomgeving [10] welke ook gebruikt wordt voor de te ontwikkelen module. Hierbij wordt als foutmelding gegeven dat een struct een incompleet type heeft. Deze fout kan niet gemakkelijk worden opgelost. In een handleiding van de icWaves wordt het gebruik van Visual Studio beschreven. Bij het starten van het voorbeeldprogramma in Visual Studio worden geen foutmeldingen weergegeven.

Het voorbeeldprogramma start met het openen van een verbinding met een verbonden icWaves. Nadat een verbinding geopend is worden een aantal instellingen veranderd, waaronder de bemonsteringssnelheid en het bereik van de AD-converter die het ingangssignaal omzet naar digitale waardes. Hierna wordt er een meting gestart en de opgenomen data opgevraagd. Het programma gebruikt vervolgens de opgenomen data nergens in het programma. Na het opnemen van een signaal wordt de icWaves klaargezet om een patroon te gaan herkennen. Er wordt geen patroon herkend doordat er geen patroon is opgeslagen. Als laatste wordt de verbinding weer verbroken met de icWaves. Bij elke stap in het programma wordt gecontroleerd op foutmeldingen en bij een fout stopt het programma.

Aan de hand van het voorbeeldprogramma is het gelukt om een beter beeld te verkrijgen hoe een verbinding opgezet kan worden en welke stappen nodig zijn om een signaal op te nemen.

5.7 UML-diagrammen

Tijdens de oriëntatiefase zijn er verschillende UML-diagrammen [11] gemaakt om een beter beeld te krijgen van de werking van de Matrix en de wensen van de opdrachtgever. Alle gemaakte UML-diagrammen zijn te zien in Bijlage C.

Het eerste diagram dat is gemaakt is een use case diagram, waarin op een hoog niveau de interactie met de module wordt beschreven. In het use case diagram is één actor te zien, de gebruiker van de Matrix. De acties die de gebruiker wil uitvoeren zijn in overleg met de opdrachtgever opgesteld en beschrijven de vijf belangrijke functionaliteiten van de icWaves. Het

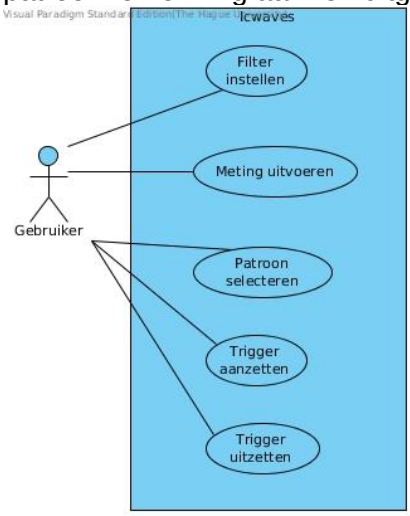
gemaakte use case diagram is te zien in Afbeelding 7 en de use cases in de afbeelding worden hieronder kort besproken.

De eerste actie is het instellen van de filter in de icWaves, die gebruikt kan worden om zwakke signalen in het ingangssignaal te vinden.

De tweede actie is het uitvoeren van een meting. Zonder deze actie is er geen signaal om een patroon uit te selecteren, waardoor de icWaves geen patronen kan herkennen.

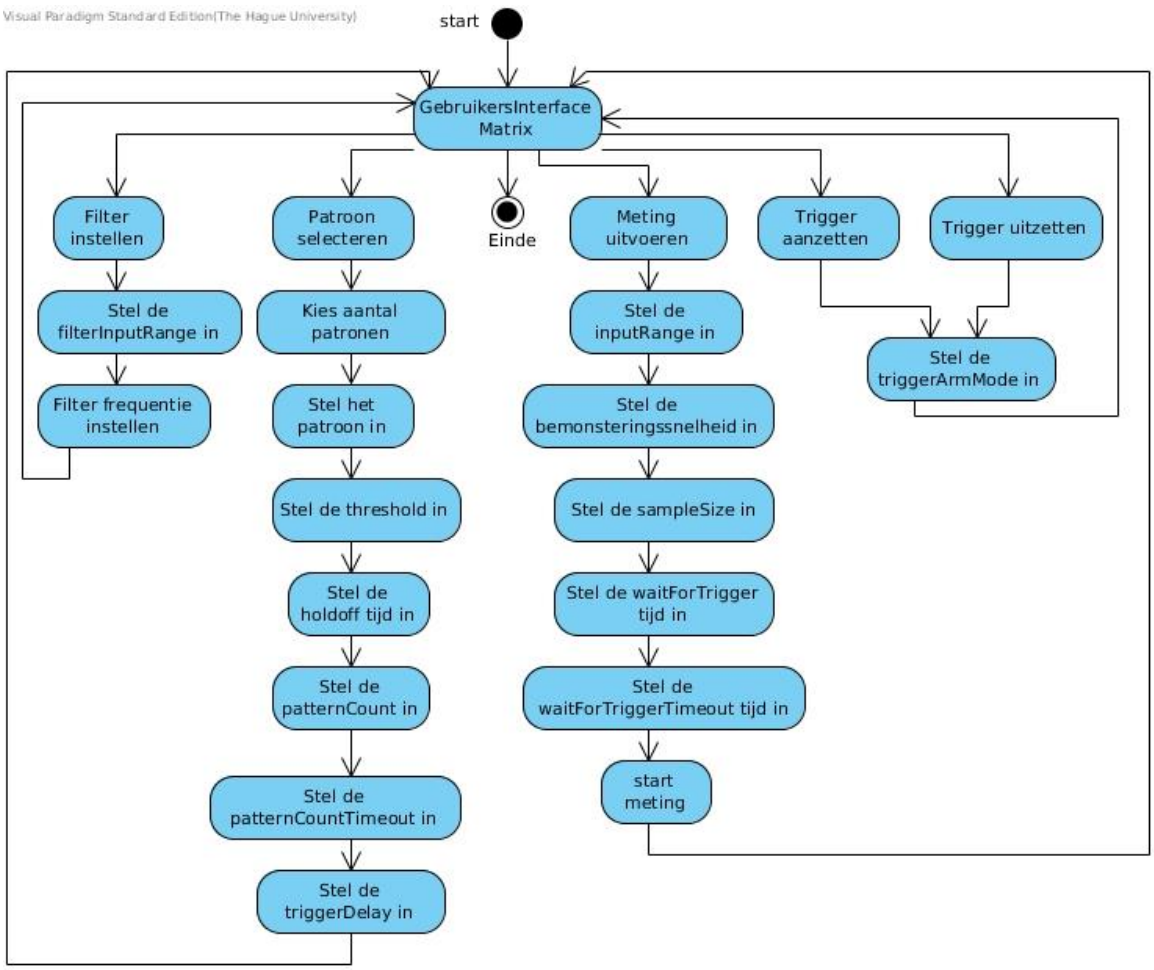
De derde actie is het selecteren van een patroon. Het selecteren van een patroon is belangrijk voor de gebruiker, zodat een goede selectie gemaakt kan worden waarop de icWaves moet reageren.

De laatste twee acties zijn voor het starten en stoppen van een trigger, dit betekent dat de patroonherkenning aan- en uitgezet kan worden.



Afbeelding 7 Use case diagram

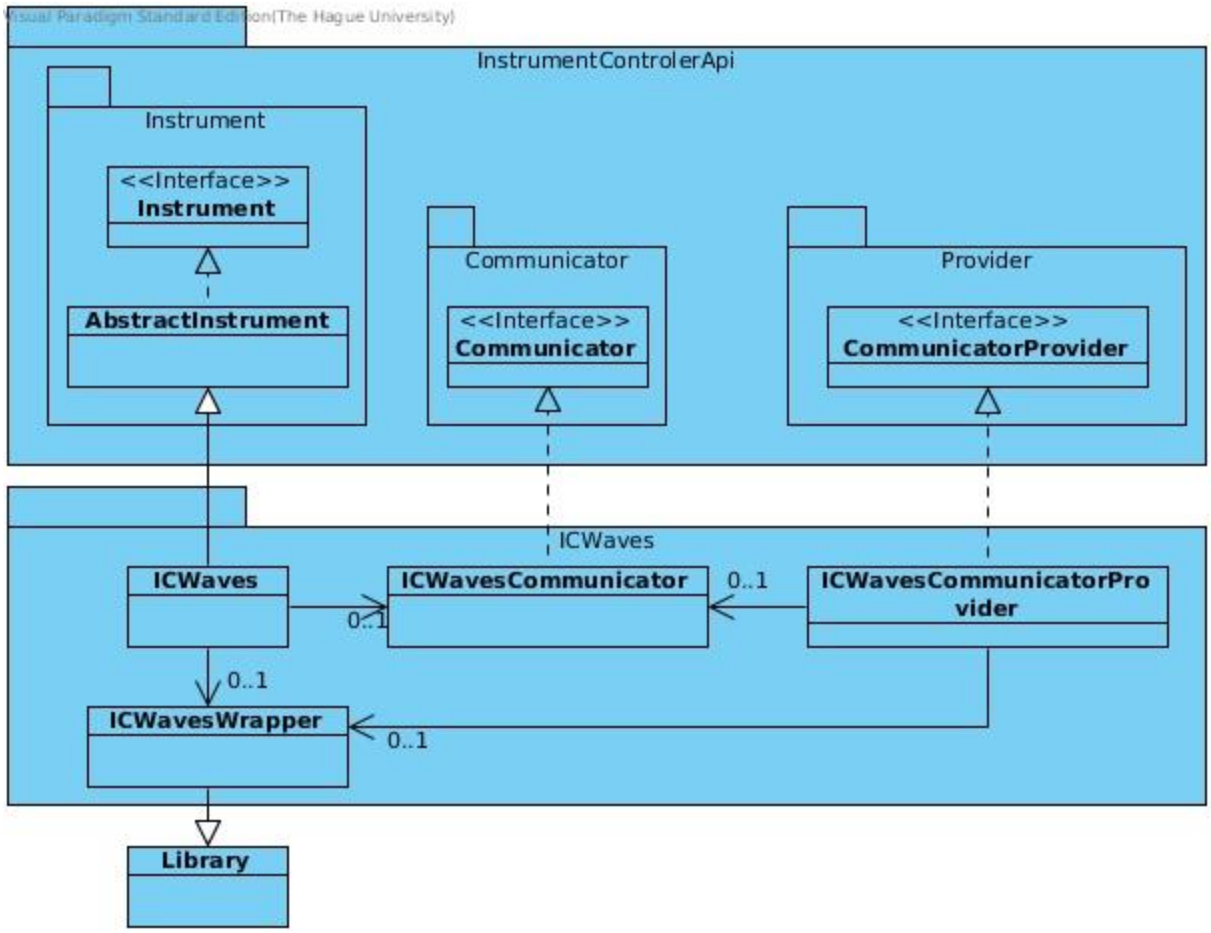
Behalve een use case diagram is er een activiteitendiagram en een klassendiagram gemaakt. Met het activiteitendiagram is vastgelegd welke stappen er genomen moet worden om de use cases uit te voeren, dit diagram is in Afbeelding 8 weergegeven.



Afbeelding 8 Activiteitendiagram oriëntatiefase

Het activiteitendiagram begint bij de gebruikersinterface van de Matrix. Vanuit deze activiteit is het mogelijk om alle vijf de use cases te starten. Op het moment dat een use case is gestart worden een aantal activiteiten uitgevoerd om de vereiste opties in te stellen en de actie uit te voeren. De instellingen zijn gevonden door naar het voorbeeldprogramma in C te kijken en de documentatie van de icWaves. In het activiteitendiagram wordt het instellen van de filter weergegeven, wanneer deze actie is uitgevoerd kan een nieuwe actie/use case gekozen worden in de activiteit Gebruikersinterface Matrix.

Als laatste diagram is er een klassendiagram gemaakt. Voor een beter overzicht zijn alle methodes en attributen uit het klassendiagram gehaald. Het gemaakte klassendiagram is te zien in Afbeelding 9 en Bijlage C geeft een overzicht van alle gemaakte UML-diagrammen tijdens dit project.



Afbeelding 9 Klassendiagram oriëntatiefase

In het klassendiagram zijn twee belangrijke packages te zien. De bovenste package bestaat uit onderdelen die al in de Matrix zitten en moeten gebruikt worden om een instrument aan te sturen. De onderste package bevat de klassen, die gemaakt moeten worden om de icWaves te gebruiken vanuit de Matrix. De onderste package is hieronder kort per klasse besproken.

De ICWaves klasse bevat de code voor het instrument en regelt de communicatie met de ICWavesWrapper.

De ICWavesCommunicator doet op dit moment weinig, omdat de communicatie opgezet wordt door de ICWaves klasse. Deze klasse wordt verwacht bij het aanmaken van een instrument en bij het zoeken van beschikbare instrumenten in de ICWavesCommunicatorProvider.

De ICWavesCommunicatorProvider maakt het instrument zichtbaar in het instrumentenoverzicht van de Matrix met behulp van de ICWavesCommunicator klasse.

De ICWavesWrapper is een interface die alle methodes van de library beschrijft.

5.8 Samenvatting

In dit hoofdstuk zijn de stappen besproken, die zijn genomen om meer inzicht te verkrijgen over dit project.

Als eerste is er een planning gemaakt, waarin de sprints staan die uitgevoerd moeten worden tijdens dit project. Vervolgens is de werking van de icWaves uitgelegd. De eisen waaraan de module moet voldoen zijn in overleg met de opdrachtgever opgesteld, de ontwikkel pc is kort besproken en er is gekozen voor JNA om te communiceren met de C library. De werking van twee programma's en de testopstelling is besproken en als laatste zijn de drie UML-diagrammen te zien die zijn gemaakt tijdens deze fase.

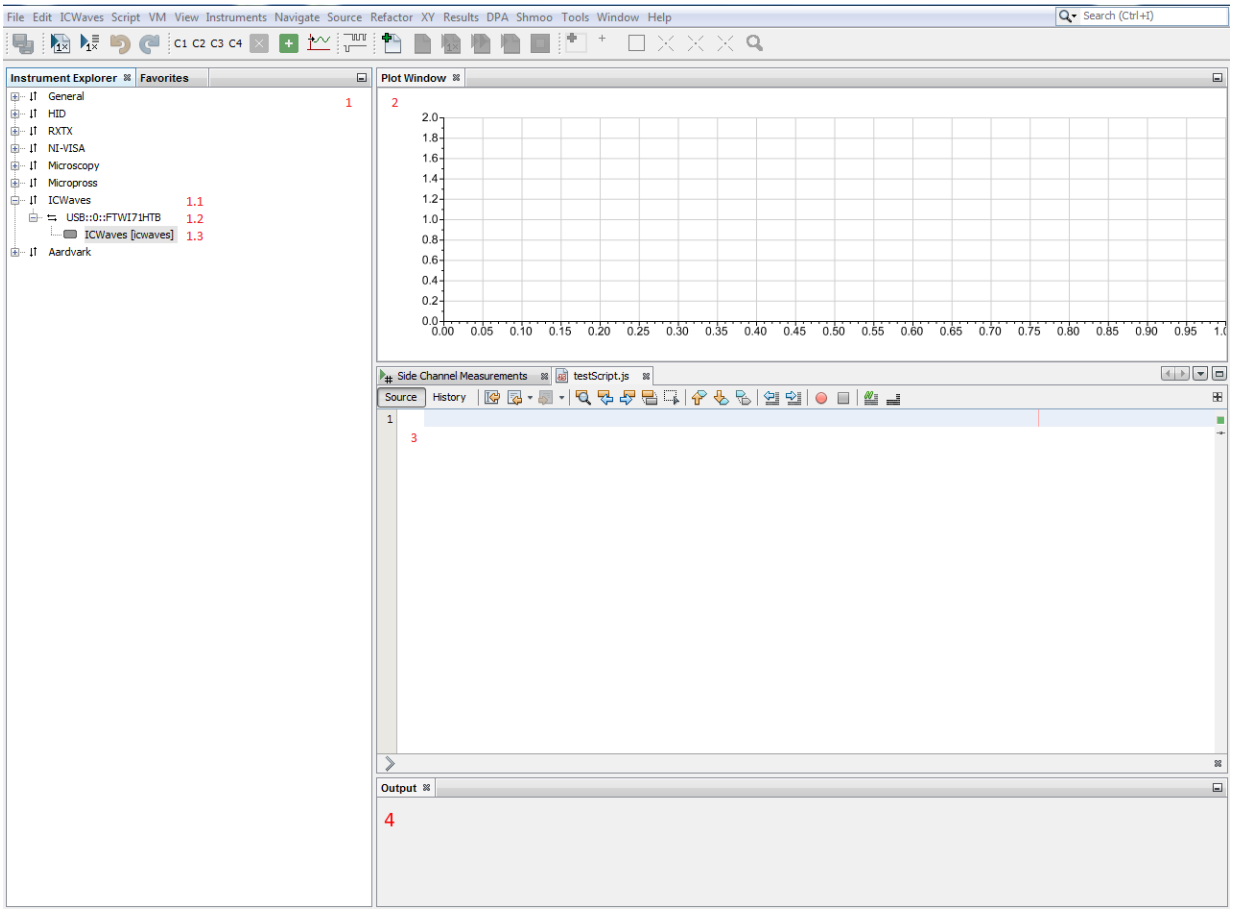
6 Implementatie eerste versie Matrix module

6.1 Introductie

In dit hoofdstuk wordt de implementatie besproken van de eerste versie van de Matrix module. In deze versie van de module worden verschillende acties geïmplementeerd. Dit houdt in dat de icWaves geopend en gesloten kan worden, er kunnen metingen gestart worden en de icWaves kan gebruikt worden als oscilloscoop. Als de icWaves als oscilloscoop gebruikt wordt is het mogelijk om te wachten op een triggersignaal en wordt het opgenomen signaal opgeslagen in de Waveform klasse. Veranderingen in de code worden bijgehouden met een versiebeheersysteem. Het gebruik van de Script Engine in de Matrix wordt uitgelegd. Aan het eind van de sprint wordt documentatie toegevoegd met behulp van Javadoc en getest met JUnit. In de planning staat het toevoegen van patroonherkenning, maar dit is in verband met tijdgebrek doorgeschoven naar de volgende sprint.

6.2 De Matrix

De Matrix bestaat uit verschillende onderdelen waarvan gebruik gemaakt kan worden tijdens dit project. Een aantal onderdelen is te zien in onderstaande afbeelding.



Afbeelding 10 De Matrix gebruikersinterface

Er zijn in Afbeelding 10 een aantal onderdelen genummerd die hieronder worden uitgelegd.

1. Het Instrument Explorer scherm, hierin zijn de verbonden apparaten te zien.
1.1 De instrument categorie waaronder de icWaves valt.
1.2 De poort waarmee de icWaves verbonden is.
1.3 Het daadwerkelijke instrument, met een label voor de Script Engine.
2. Het plot-scherm voor weergave van grafieken.
3. De Script Engine, waarin javascript code geschreven kan worden.
4. Het Output scherm

6.3 Versiebeheersysteem

Voor het bijhouden van veranderingen in de code van de module moet een versiebeheersysteem gekozen worden. [12] Bij dit project is er één grote limitatie. Er is namelijk geen internetverbinding op de ontwikkel pc, waardoor het niet mogelijk is om de code op een server te zetten. Hierdoor vallen gecentraliseerde versiebeheersystemen zoals CVS of SVN af en zijn er zijn twee versiebeheersystemen met elkaar vergeleken die zonder internetverbinding werken.

Het eerste systeem is Git. [13] Het voordeel hiervan is dat er een lokale opslagplaats wordt bijgehouden die op elk moment gesynchroniseerd kan worden met een server. Het synchroniseren met een server is niet mogelijk, maar wijzigingen lokaal op te slaan volstaat ook. Door het bijhouden van veranderingen is het mogelijk om een oude versie van de code te laden. Git kan zowel vanuit een command-line als een gebruikersinterface aangestuurd worden.

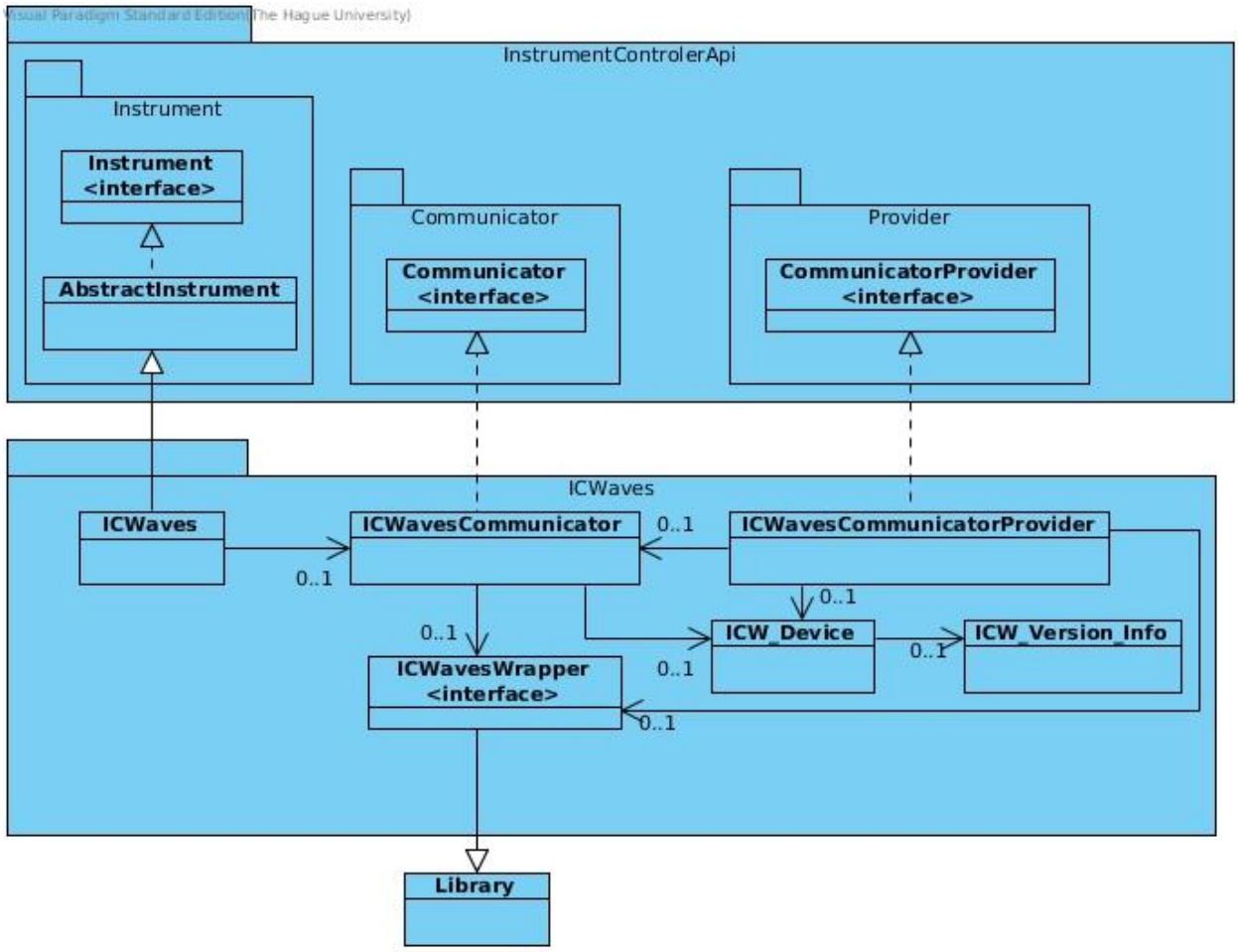
De tweede optie voor een versiebeheersysteem is Mercurial [14]. Dit systeem heeft ongeveer dezelfde voordelen als Git, maar de afstudeerder heeft geen ervaring met het gebruik van Mercurial.

Allebei de versiebeheersystemen hebben dezelfde mogelijkheden bij dit project, alleen heeft de afstudeerder ervaring met Git. Daarom valt de keuze op Git.

6.4 Klassen van de ICWaves module

De belangrijkste klassen van de eerste versie van de module worden besproken in deze paragraaf. Deze klassen zijn te zien in het diagram in Afbeelding 11. Wanneer er in dit verslag wordt verwezen naar de communicator, wordt de ICWavesCommunicator klasse bedoeld en wanneer er verwezen wordt naar de provider, wordt de ICWavesCommunicatorProvider klasse bedoeld.

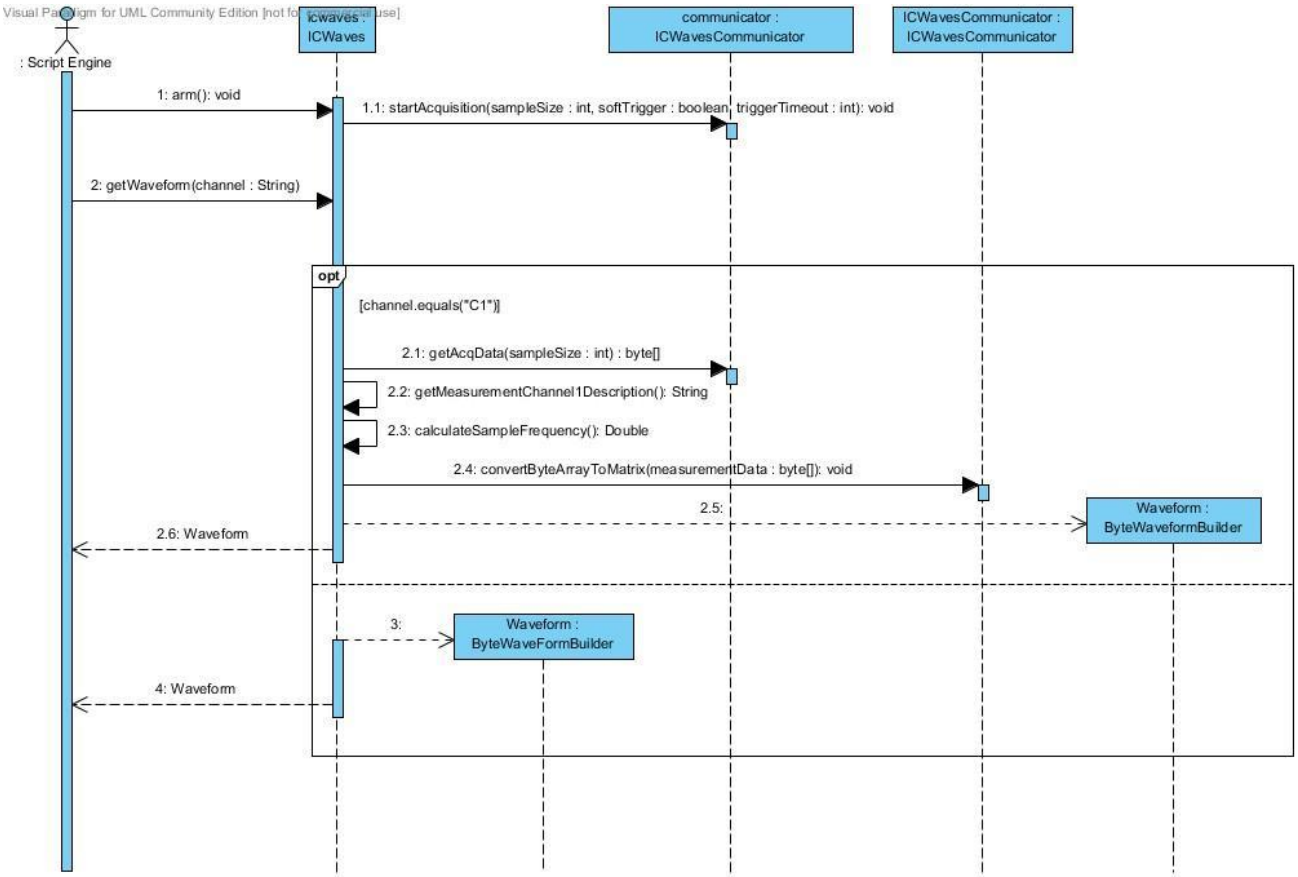
Het ontwerp is tijdens deze sprint aangepast, zodat de communicatie verloopt via de communicator en niet meer via de ICWaves klasse. Meer uitleg is te vinden bij Hoofdstuk 6.4.2 ICWavesCommunicator. In dit het ontwerp zijn ook de ICW_Device en de ICW_Version_info data structuren in eigen klassen gezet. De reden hiervoor is dat volgens de code standaarden een bestand maar één public klasse of interface mag bevatten. Als de klasse of interface private zijn is het wel toegestaan, maar dit is hier niet het geval.



Afbeelding 11 Klassendiagram versie 1

Behalve een klassendiagram is er ook een sequentiediagram gemaakt om de werking van de icWaves beter te begrijpen. Dit diagram laat de werking van de icWaves als oscilloscoop zien met behulp van een script.

Het diagram is gemaakt door de werking van de oscilloscoop in de Matrix te bekijken. Zie Afbeelding 12. De interactie met de library is weggelaten, omdat het de duidelijkheid niet ten goede komt. De interactie met de library vindt plaats op het moment dat er een methode wordt aangeroepen bij de communicator.



Afbeelding 12 Sequentiediagram versie 1

De werking van de belangrijke methodes in het sequentiediagram zijn hieronder besproken per klasse.

6.4.1 ICWavesCommunicatorProvider klasse

De provider heeft een aantal taken, waaronder het aantal verbonden icWaves apparaten weergeven. Om de taken te vervullen wordt in een static blok de library geladen en vervolgens wordt er een gesynchroniseerde versie van gemaakt. Deze versie zorgt ervoor dat maar één proces tegelijk de library kan aanspreken.

De eerste taak van de provider is de naam geven die gebruikt wordt om deze klasse in de Instrument Explorer weer te geven.

De tweede taak is een lijst met beschikbare icWaves apparaten aanmaken. Per verbonden icWaves wordt vervolgens informatie opgevraagd en een string opgebouwd. Deze string bestaat uit drie onderdelen.

Het eerste onderdeel is de naam USB, het tweede deel is het nummer van de poort waarop de icWaves verbonden is en als laatste het serienummer. Alle drie de onderdelen zijn gescheiden door twee keer een dubbelepunt. Dit ziet er als volgt uit voor een icWaves op poort nul: USB::0::serienummer.

Wanneer een communicator wordt opgevraagd wordt de aangemaakte string meegegeven en kan de communicator deze gebruiken om het apparaatnummer te bepalen. Er is gekozen om deze naamgeving te gebruiken in overleg met de opdrachtgever, zodat het voor een gebruiker mogelijk is om verbonden icWaves apparaten te onderscheiden.

6.4.2 ICWavesCommunicator klasse

De communicator verzorgt de communicatie tussen de ICWaves klasse en de ICWavesWrapper. In het eerste ontwerp was er een directe verbinding tussen de ICWaves klasse en de icWaves trigger box, waarbij het communicatieprotocol in de ICWaves klasse geïmplementeerd is. Het nadeel hiervan is dat voor bijvoorbeeld ondersteuning van seriële communicatie, de ICWaves klasse herschreven moet worden. Wanneer er gebruik wordt gemaakt van de communicator klasse is de communicatie gesplitst van de werking van het instrument. Dit betekent dat het communicatieprotocol aangepast kan worden, zonder dat dit invloed heeft op de werking van de ICWaves klasse.

In de eerste versie van de module is het mogelijk om een verbinding met de icWaves te openen en te sluiten. Om de verbinding met de icWaves te openen wordt het apparaatnummer uit de string gehaald die is meegegeven door de provider. Wanneer een verbinding geopend is kunnen verschillende methodes worden aangeroepen. Deze methodes gebruiken de ICW_Device klasse als parameter, zodat de library weet met welk verbonden apparaat gecommuniceerd moet worden. De communicator heeft in deze versie drie belangrijke methodes die nodig zijn om metingen uit te voeren.

De eerste methode is voor het starten van een meting. Deze methode heet startAcquisition en gebruikt drie parameters. De eerste parameter geeft aan hoeveel meetpunten opgenomen moeten worden. De tweede parameter geeft aan of er gebruik wordt gemaakt van een externe trigger om de meting te starten. De laatste parameter geeft de tijd aan dat er gewacht wordt op een externe trigger. Met deze methode is het mogelijk om de icWaves te laten wachten op een triggersignaal tot een maximale tijd van 10737 ms.

De tweede methode wordt gebruikt om de meetpunten op te slaan in een array. Deze methode heet getAcqData en maakt gebruik van één parameter, namelijk het aantal opgenomen meetpunten.

De derde methode wordt gebruikt om meetpunten die zijn opgenomen met de icWaves om te zetten naar een formaat dat gebruikt kan worden in het plot-scherm van de Matrix. De icWaves library is geschreven in C en gebruikt unsigned bytes om de meetpunten op te slaan, als dit vervolgens in Java wordt gelezen worden de waardes altijd als signed bytes geïnterpreteerd. Een unsigned byte kan waardes tussen de 0 en 255 bevatten, terwijl een signed byte tussen de -128 en 127 moet liggen. Om de verkregen meetpunten om te zetten naar de juiste waardes in Java is er gebruik gemaakt van de methode convertByteArrayToMatrix en de code hiervan is te zien in Codefragment 1.

```
static void convertByteArrayToMatrix(byte[] byteArray) {
        for (int i = 0; i < byteArray.length; i++) {
            byteArray[i] = (byte) (byteArray[i] ^ 0x80);
        }
}
```

Codefragment 1 ConvertByteArrayToMatrix

In Codefragment 1 wordt de data die als parameter is meegegeven omgezet. Dit wordt gedaan door op het sign bit een exclusive OR operatie uit te voeren en dan het resultaat te casten naar een byte. Dit heeft als resultaat dat een aangeboden signed byte wordt omgezet naar een unsigned byte en vice versa.

Elke methode die gebruik maakt van de library geeft een waarde terug, waarmee wordt aangegeven of er fouten zijn opgetreden. Deze waarde kan liggen tussen de 0 en 23, waarbij 0 aangeeft dat de methode zonder fouten is uitgevoerd. In alle andere gevallen wordt de teruggegeven waarde vertaald naar de bijbehorende foutmelding. Hierdoor is het mogelijk om de oorzaak van de fout sneller te achterhalen.

6.4.3 ICWaves klasse

De ICWaves klasse wordt gebruikt om de icWaves binnen de Matrix te representeren. In deze klasse zitten methodes die gebruikt kunnen worden door de eindgebruiker. Drie van deze methodes zijn direct betrokken bij het uitvoeren van een meting. Daarnaast zijn er verschillende methodes beschikbaar die parameters instellen of algemene informatie verschaffen aan de gebruiker.

De eerste methode, startMeasurement, is bedoeld voor het opnemen van een meting. De parameters die nodig zijn om een meting te starten worden in de methode opgehaald en meegegeven aan de communicator. Vervolgens wordt de opgenomen data opgevraagd en in een array gestopt. Na verwerking met de methode ConvertByteArrayToMatrix is de opgenomen data beschikbaar voor de gebruiker.

De tweede methode, arm, is bedoeld om de icWaves als oscilloscoop te laten werken. Met deze methode wordt de icWaves klaargezet om te reageren op een triggersignaal voor het starten van een meting.

De derde methode, getWaveform, wordt gebruikt om een Waveform te maken van de opgenomen data. De Waveform is een klasse waarin meetpunten opgeslagen worden en weergeven kunnen worden in het plot-scherm.

De methode getWaveform wordt zowel bij een oscilloscoop als een icWaves gebruikt voor het verkrijgen van een Waveform. Door middel van een string wordt aangegeven op welke poort van een oscilloscoop data opgenomen moet worden. De icWaves heeft maar één poort, dus er is met de opdrachtgever afgesproken dat altijd de naam “C1” meegegeven moet worden.

Wanneer C1 als naam is meegegeven wordt de opgeslagen data opgevraagd. De opgevraagde data wordt omgezet met de methode ConvertByteArrayToMatrix. Als laatste wordt de data in een Waveform gestopt die door de Matrix weergegeven kan worden in het plot-scherm. De bijbehorende code voor het verkrijgen van een Waveform is te zien in Codefragment 2

```
@Override
   public Waveform getWaveform(String channel) throws IOException {
      if(channel.equals("C1")) {
         measurementData = communicator.getAcqData(sampleSize);

         final String description = getChannel1Description();
         final double sampleFrequency = calculateSampleFrequency();

         ICWavesCommunicator.convertByteArrayToMatrix(measurementData);
         return new ByteWaveformBuilder(measurementData, description,
                        1.0/sampleFrequency);
      }
      return new ByteWaveformBuilder(0);
   }
```

Codefragment 2 GetWaveform

Het opnemen van data met de icWaves werkt net iets anders dan met een oscilloscoop. Dit kan problemen opleveren als de gebruikers dit niet weten. Wanneer bij een oscilloscoop de arm-methodes wordt aangeroepen blijft deze wachten tot een triggersignaal is ontvangen. Als er binnen tien seconde geen triggersignaal ontvangen wordt, krijgt de gebruiker een foutmelding te zien. Bij de icWaves wordt automatisch een meting gestart als binnen tien seconde geen triggersignaal is ontvangen.

Het opvragen van de data werkt ook net iets anders. Wanneer de data wordt opgevraagd van de oscilloscoop geeft deze de data die op dat moment op het scherm staat. De icWaves geeft de opgenomen data terug, maar op het moment dat deze methode nog een keer wordt aangeroepen wordt er een foutmelding gegeven. Dit heeft er mee te maken dat de library verwacht dat er een nieuwe meting wordt gedaan voordat de data opnieuw opgevraagd wordt.

Hiervoor is later in dit project een oplossing bedacht in overleg met de opdrachtgever. Wanneer de arm-methode wordt aangeroepen bij de icWaves wordt een vlag gezet en deze wordt gecontroleerd op het moment dat de data wordt opgevraagd. Afhankelijk van de vlag wordt nieuwe data opgevraagd of de vorige meting teruggegeven, zodat dit overeenkomt met de werking van een oscilloscoop.

6.4.4 ICWavesWrapper klasse

De ICWavesWrapper klasse is een interface die alle methodes beschrijft die ondersteund worden door de library. Deze interface overerft van een JNA klasse en kan daardoor de methodes van de library aanroepen zonder problemen. De ICWavesWrapper wordt in deze versie gebruikt door de communicator om de icWaves aan te sturen en door de provider om het aantal verbonden apparaten te verkrijgen.

De meeste methodes hebben een ICW_Device klasse nodig. Deze klasse gebruikt de library om te bepalen met welk apparaat gecommuniceerd moet worden en bevat een aantal variabelen, zoals een serienummer en een beschrijving van de icWaves.

Met de hierboven genoemde klassen is de functionaliteit van de eerste versie toegevoegd. Vervolgens moet de documentatie gemaakt worden met Javadoc en de code getest met JUnit. Wanneer dit gedaan is wordt de code ingeleverd voor een review. Het commentaar van de review wordt verwerkt tijdens de ontwikkeling van de tweede versie.

6.5 Javadoc

Javadoc [15] is een generator die documentatie maakt in de vorm van HTML-bestanden op basis van commentaar in Java code. Er is gekozen voor het gebruik van Javadoc, omdat dit standaard werkt met de Netbeans omgeving en al gebruikt wordt binnen Brightsight. Een voordeel voor het gebruik van Javadoc is dat de werking van de methodes is beschreven als commentaar in de code. Het commentaar dat geschreven wordt voor dit project wordt gedaan in het Engels, omdat het hierdoor bruikbaar wordt voor alle medewerkers van Brightsight.

In het commentaar van Javadoc kan gebruik worden gemaakt van HTML-tags om de tekst beter leesbaar te maken. Wanneer er gebruik wordt gemaakt van HTML-tags in het commentaar, wordt het minder leesbaar in de code. Ten behoeve van de leesbaarheid van het commentaar in de code wordt er alleen gebruik gemaakt van <code> en
 HTML-tags.

Bij dit project worden alle methodes voorzien van Javadoc en hiervoor kunnen de volgende Javadoc tags gebruikt worden: @param, @return, @throws. Het gebruik van deze tags wordt eerst uitgelegd en daarna wordt er een voorbeeld gegeven.

- @param wordt gebruikt om de parameters van een methode te beschrijven. Hierbij is het gebruikelijk dat als eerste de naam van de parameter wordt opgeschreven en daarna een beschrijving.
- @return wordt gebruik om de return waarde van een methode te beschrijven.
- @throws wordt gebruikt om te beschrijven welke excepties een methode kan geven.

Hieronder is een voorbeeld te zien van een stuk Javadoc commentaar waarin er gebruik wordt gemaakt van de verschillende tags.

```
/**
 * Dit is een stuk voorbeeld Javadoc.
 *
 * <br> Hier wordt nog wat extra informatie gegeven over de methode.
 * @param parameter1 De parameter doet tekst tekst nog meer tekst.
 * @param parameter2 De parameter doet tekst tekst nog meer tekst.
 * @return parameter1 wordt teruggegeven.
 * @throws IOException wanneer een ongeldige parameter wordt meegegeven.
 */
```

Codefragment 3 Javadoc voorbeeld

6.6 Script Engine

Binnen de Matrix kan er gebruikt gemaakt worden van een Script Engine om, door middel van javascript, methodes van klassen aan te roepen. Bij het openen van een apparaat moet een label worden toegewezen door de gebruiker. Het label wordt vervolgens gebruikt om vanuit een script het instrumentobject te verkrijgen.

De Script Engine is handig voor het herhaaldelijk uitvoeren van vaste taken, zoals het uitvoeren van een meting met een oscilloscoop en het wegschrijven van deze meetgegevens naar een bestand of het laden van een groep van instellingen.

Tijdens de ontwikkeling van een module kan er gebruik gemaakt worden van de Script Engine om bijvoorbeeld functionaliteit te simuleren.

6.7 Analyse testniveaus

Bij het testen van software wordt er onderscheid gemaakt in vier niveaus, namelijk unittesten, integratietesten, systeemtesten en acceptatietesten. Voor dit project is de bruikbaarheid van elk van deze niveaus onderzocht. [16] Voor extra informatie over de uitgevoerde tests kan Bijlage D bekeken worden.

6.7.1 Unittest

Als eerste is onderzoek gedaan naar het gebruik van unittesten [17]. Een unittest is gericht op één methode, waarbij verschillende testgevallen doorlopen worden. Hierbij is het gewenst dat de methode die getest wordt onafhankelijk kan werken van andere methodes.
Wanneer dit niet mogelijk is kan er gebruik worden gemaakt van zogenaamde mockobjecten. Deze objecten simuleren het gedrag van de methodes waar de te testen methode van afhankelijk is.

Bij dit project zijn de meeste methodes binnen een klasse afhankelijk van methodes uit andere klassen, dit betekent dat voor het unittesten van deze methodes veel mockobjecten geschreven moeten worden. De voordelen van unittesten wegen in dit geval niet op tegen de nadelen van het maken van de benodigde mockobjecten. Daarom is besloten om binnen dit project geen unittests uit te voeren.

6.7.2 Integratietest

Als tweede is er onderzoek gedaan naar de mogelijkheden van integratietesten [18]. Een integratietest is gericht op het testen van de samenwerking tussen de verschillende onderdelen. Door de afhankelijkheid tussen de verschillende methodes is dit project geschikt voor het uitvoeren van integratietesten.

Tijdens de eerste integratietest wordt de samenwerking tussen de verschillende klassen getest, waarbij het gedrag van de icWaves gesimuleerd wordt. Pas in tweede instantie wordt er gebruikt gemaakt van de icWaves. Na afloop van de integratietest moet duidelijk zijn of de geïmplementeerde methodes in de verschillende klassen correct met elkaar samenwerken.

6.7.3 Systeemtest

Vervolgens is het gebruik van systeemtesten onderzocht [19]. Bij een systeemtest wordt er gekeken of het volledige systeem aan de vooraf gestelde eisen voldoet. Binnen dit project ligt de nadruk tijdens de systeemtest op het testen van de samenwerking tussen de Matrix, de ICWaves module en de icWaves trigger box. Daarnaast wordt de gebruikersinterface, de Script Engine en het gebruik van het plot-scherm getest. Uit de systeemtest moet blijken of alle functies correct werken en het systeem aangeboden kan worden voor de gebruikersacceptatietest.

6.7.4 Acceptatietest

Als laatste zijn de mogelijkheden van de acceptatietest onderzocht [20]. Hierbij ligt de nadruk op het testen van de bruikbaarheid van het systeem door een eindgebruiker. Uit de acceptatietest moet blijken of de werking van het systeem aansluit bij de werkwijze van de gebruiker. Zo moet de opbouw van de gebruikersinterface logisch en intuïtief zijn.

Tijdens het project is de gebruikersinterface in nauwe samenwerking met de eindgebruiker ontwikkeld. Verder is de module geïntegreerd in de bestaande software, waardoor in overeenstemming met de opdrachtgever besloten is om geen aparte acceptatietest uit te voeren.

6.8 Uitvoeren integratietests

Voor het testen van de eerste versie van de op te leveren module is gebruik van JUnit [21]. Dit is testframework voor Java waarmee testcode geschreven en uitgevoerd kan worden. De testcode is in een aparte package gestopt, zodat dit gescheiden is van de productiecode.

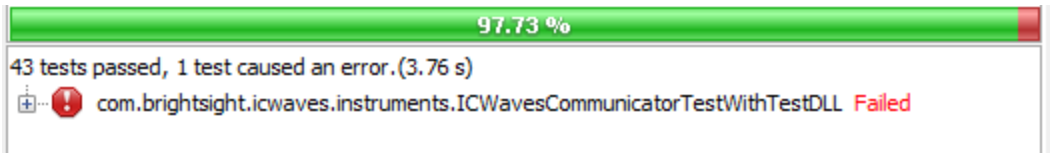
Bij het gebruik van JUnit is er een aantal tags dat gebruikt kan worden. De belangrijkste tags worden hieronder uitgelegd.

@Before kan gebruikt worden om een methode aan te geven die voor elke test uitgevoerd moet worden, zoals het vullen van ingangsvariabelen met de juiste startwaarde.

@After kan gebruikt worden om een methode aan te geven die na elke test uitgevoerd wordt. Deze methode kan gebruikt worden om gealloceerde bronnen na een test weer vrij te geven.

Er zijn twee tags die aangegeven dat de methode maar één keer wordt uitgevoerd door de test klasse, namelijk de @BeforeClass en @AfterClass. Deze methodes kunnen bijvoorbeeld gebruikt worden om een initiële verbinding te maken en deze na afloop van het testen weer te verbreken.

@Test wordt gebruikt om aan te geven dat de methode een test is. Het resultaat van de test wordt weergegeven in een overzichtsscherm, waarbij het percentage geslaagde tests wordt weergeven door middel van een groene balk. De niet geslaagde tests worden in het rood weergegeven. In het voorbeeld van Afbeelding 13 is te zien dat 43 van de in totaal 44 uitgevoerde tests geslaagd zijn.



Afbeelding 13 Overzicht test uitslag

De tests zijn gesplitst in twee groepen, een groep waarbij de icWaves gebruikt wordt en een tweede groep waarbij het gebruik van de icWaves gesimuleerd wordt. De tweede groep is bedoeld om beïnvloeding van de icWaves op de testresultaten te voorkomen. Door de gesimuleerde versie wordt altijd een vooraf bepaalde serie van meetwaardes teruggegeven. Voor het testen van het opslaan en ophalen van instellingen wordt er gebruik gemaakt van de icWaves.

Voor het testen van de communicator kan er zowel gebruik gemaakt worden van de icWaves als de simulatie. De communicator wordt op twee verschillende manieren getest. Bij de eerste manier wordt er gebruik gemaakt van de simulatie en worden alleen de lokale methodes van de communicator getest, zoals het weergeven van een foutmelding en het opvragen van meetwaarden. Bij de tweede manier wordt gebruik gemaakt van de icWaves waarbij het opslaan en doorgeven van instellingen wordt getest.

Vervolgens zijn er test geschreven voor de ICWaves klasse waarbij er gebruik wordt gemaakt van de communicator met zowel de simulatie als de icWaves. De simulatie wordt gebruikt voor het testen van de lokale methodes, zoals het instellen van het aantal op te nemen meetwaardes. De methodes die afhankelijk zijn van interactie met de icWaves, zoals het opslaan en ophalen van instellingen worden uitgevoerd met de icWaves.

6.9 Samenvatting

In dit hoofdstuk zijn de stappen besproken die zijn uitgevoerd om de eerste versie van de module te maken.

Als eerst is er uitgelegd wat de eerste versie van de module moet gaan doen, hierna is de werking van de klassen uitgelegd die gebruikt zijn in de module. Nadat de werking is uitgelegd wordt het gebruik van Javadoc besproken. Het gebruik van de Script Engine in de Matrix is uitgelegd en als laatste is het gebruik van JUnit besproken om de module te testen.

In het volgende hoofdstuk worden verbeteringen toegevoegd naar aanleiding van de code review en wordt functionaliteit toegevoegd voor de tweede versie van de module.

7 Implementatie tweede versie Matrix module

7.1 Introductie

In dit hoofdstuk wordt de implementatie besproken van de tweede versie van de Matrix module. Deze sprint start met het verwerken van de feedback van de code review. Vervolgens worden de belangrijkste wijzigingen per klasse besproken, zoals het selecteren en herkennen van een patroon en het toevoegen van filterinstellingen. Hierna wordt de nieuwe testopstelling besproken en wordt uitgelegd hoe een patroon geselecteerd kan worden met behulp van de gebruikersinterface. Als laatste is de testdekking besproken van dit project.

7.2 Code review

De code van de eerste versie van de module is gecontroleerd door Remko. Tijdens de review zijn geen grote problemen ontdekt, maar er zijn wel een aantal aanpassingen nodig. Zo zijn de namen van de ICW_Device en ICW_Version_Info klassen veranderd in DeviceStrucure en VersionInfoStructure om aan de code standaarden te voldoen. Enkele namen van methodes zijn veranderd om meer duidelijkheid te verschaffen aan de gebruiker en verschillende integer parameters zijn vervangen door enumeraties, zoals de bemonsteringsfrequentie.

In de eerste versie van de module waren de ondersteunende methodes public, terwijl deze niet aangesproken mogen worden door de gebruiker. De ondersteunde methodes worden private of package private gemaakt om er zeker van te zijn dat een gebruiker deze methodes niet kan aanroepen. Package private wordt gebruikt als andere klassen binnen de package nog wel deze methode nodig hebben.

Als laatste is er naar aanleiding van de review een methode weggehaald. Dit is de methode startMeasurement. De gebruikers van de Matrix zijn gewend om met de arm en getWaveform methodes data op te nemen en hebben daarom geen aparte methode nodig voor het uitvoeren van een losse meting.

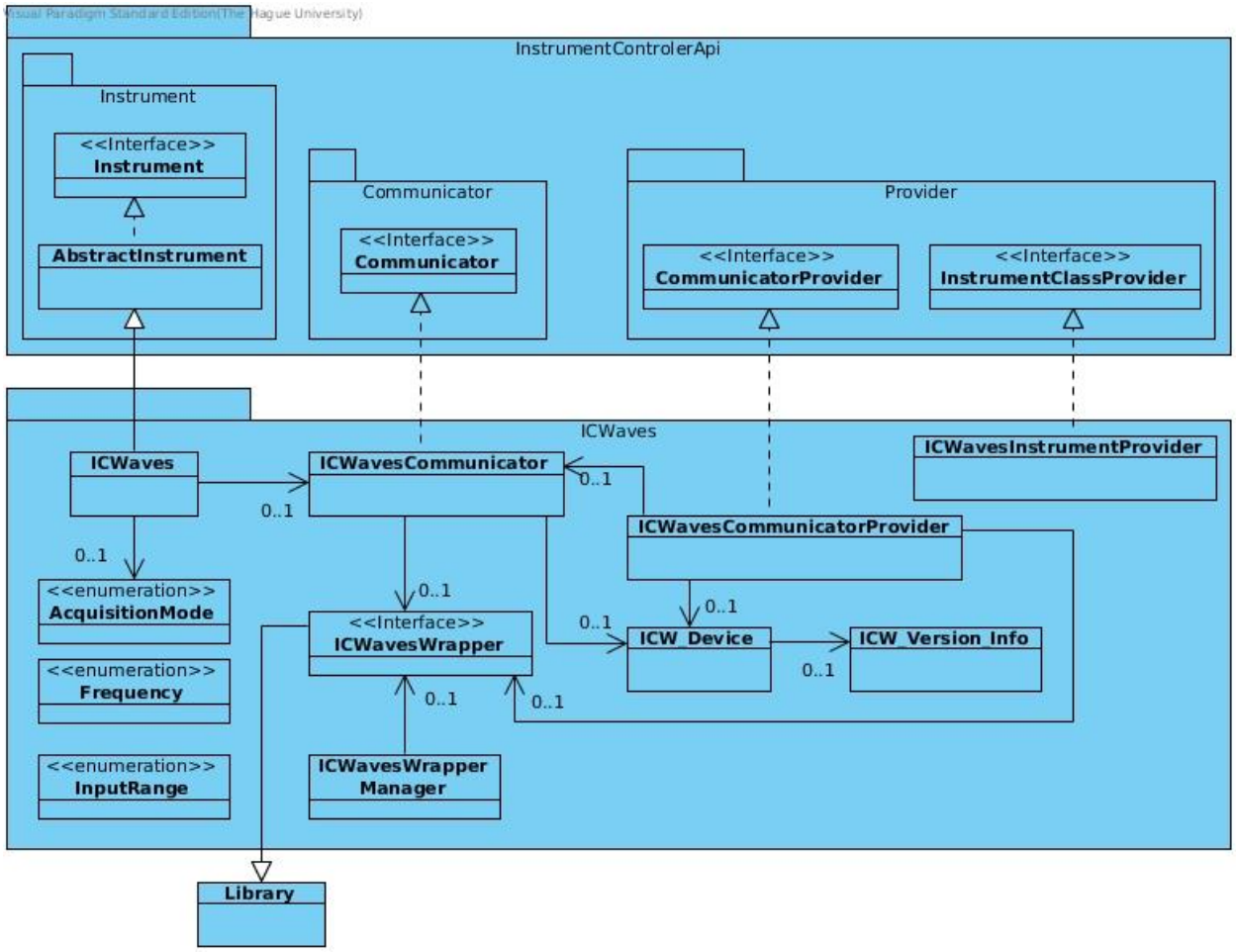
7.3 Klassen van de ICWaves module

Tijdens de ontwikkeling van de eerste versie zat de module in een project met implementaties van verschillende instrumenten. Hierin zitten vooral instrumenten, die maar een aantal acties kunnen uitvoeren en geen uitgebreide functionaliteit hebben. In deze sprint wordt de functionaliteit van de icWaves uitgebreid met een menu. Bij het opstarten van het project was het eenvoudiger om code aan een bestaand project toe te voegen en daarom is pas tijdens deze sprint de code verplaatst naar een apart project.

Doordat er een nieuw project is aangemaakt moeten alle afhankelijke projecten en libraries opnieuw toegevoegd worden. Hierna moet het versiebeheersysteem in de juiste folder gezet worden, waardoor de veranderingen in de code bijgehouden kunnen worden.

Als het project succesvol is overgezet wordt er gewerkt aan de implementatie van de tweede versie. Tijdens deze sprint is gestart met het bijwerken van het klassendiagram, hierbij zijn de

ICWavesWrapperManager en ICWavesInstrumentProvider toegevoegd. Deze nieuwe klassen zijn te zien in het klassendiagram in Afbeelding 14. Vervolgens wordt onder de afbeelding besproken wat de nieuwe klassen doen en wat de veranderingen aan de huidige klassen zijn. De DeviceStructure klasse en de VersionInfoStructure klasse worden niet uitgelegd, omdat hieraan geen aanpassingen zijn gemaakt.



Afbeelding 14 Klassendiagram versie 2

7.3.1 ICWavesInstrumentProvider

De ICWavesInstrumentProvider klasse registreert de ICWaves klasse als instrument. Wanneer dit niet wordt gedaan kan er geen verbinding gemaakt worden met de ICWaves, omdat deze niet als instrument gevonden wordt.

7.3.2 ICWavesWrapperManager

De ICWavesWrapperManager klasse zorgt ervoor dat maar één instantie bestaat van de gesynchroniseerde ICWavesWrapper klasse. Deze instantie kan door verschillende klassen worden opgevraagd, maar door het gebruik van de gesynchroniseerde klasse kan de library niet tegelijk aangesproken worden. In de vorige versie werd dit geregeld in de provider, maar het is netter om dit in een aparte klasse te doen.

7.3.3 ICWavesCommunicatorProvider

Er zijn een aantal veranderingen doorgevoerd aan de provider tijdens deze iteratie. De provider verkrijgt nu de instantie van de ICWavesWrapper door deze te vragen van de ICWavesWrapperManager.

In de vorige versie van de communicator werd een string ontvangen met een poortnummer en het serienummer van het verbonden apparaat. De communicator haalde vervolgens het nummer uit deze string. Nu vindt deze handeling plaats in de provider, omdat de provider weet hoe de string is opgebouwd.

7.3.4 ICWavesCommunicator

De communicator heeft methodes gekregen om een patroon met de daarbij behorende opties in te stellen. Daarnaast is het mogelijk om het bereik en de frequentie van de filter aan te passen. Deze nieuwe methodes worden gebruikt om de data van de ICWaves klasse door te geven aan de library. In de methodes waar gebruik wordt gemaakt van een enumeratie voert de communicator een extra stap uit, om deze om te zetten naar een integer die de library begrijpt.

7.3.5 ICWaves

De belangrijke methodes die zijn toegevoegd zijn het opslaan en ophalen van een patroon, de berekening van een drempelwaarde en het instellen van de filter.

Bij het opslaan en ophalen van een patroon wordt er gebruik gemaakt van een array. De array moet altijd omgezet worden met de methode convertUnsignedSignedByteArray. Deze methode zet unsigned waardes om naar signed waardes en vice versa, zodat er normale grafieken weergegeven worden in het plot-scherm. Doordat het patroon is omgezet voordat het in het plot-scherm wordt weergegeven is het nodig om het patroon weer terug te zetten wanneer een selectie wordt gebruikt uit het plot-scherm. Wanneer het geselecteerde patroon niet wordt omgezet zoekt de icWaves naar een heel ander patroon dan verwacht wordt, waardoor het juiste patroon nooit wordt herkend.

Bij het doorvoeren van deze aanpassingen zijn verschillende problemen aangetroffen, waarvan er twee besproken worden.

Als een patroon wordt opgeslagen op de icWaves met behulp van Java is het mogelijk om een array van nul elementen mee te geven. De icWaves geeft hierbij een foutmelding , maar gebruikt wel de lege array als patroon. Normaal wordt de oude waarde onthouden, maar in de library gaat het fout, omdat het niet mogelijk is om een lege array aan te maken. Dit probleem is opgelost door bij het opslaan van een patroon op de lengte van de array te controleren.

Een ander probleem doet zich voor bij het opslaan van instellingen voor de patroonherkenning, doordat Java unsigned integer en unsigned short niet ondersteund. De waardes kunnen niet in de code omgezet worden, omdat het mogelijk moet zijn om deze vanuit het menu in te vullen.

Bij het gebruik van het menu wordt een foutmelding gegeven op het moment dat er te grote waarde wordt ingevuld.

De oplossing is gebruik te maken van een groter datatype in Java. Bij het gebruik van deze datatypes moet gecontroleerd worden of de ingevulde waardes nog in het bereik zitten van de unsigned types, zodat er geen onverwachte situaties ontstaan.

De icWaves heeft één belangrijke methode voor het berekenen van een drempelwaarde, waarbij gebruik gemaakt wordt van een ingangssignaal en een opgeslagen patroon. Hiermee is het mogelijk om drempelwaardes te berekenen, waarmee aangegeven wordt in hoeverre een ingangssignaal op een opgeslagen patroon moet lijken. Om een beter overzicht te krijgen waarom de drempelwaarde van belang is wordt in Afbeelding 16 het proces weergegeven dat gebruikt wordt voor patroonherkenning en het genereren van een trigger.

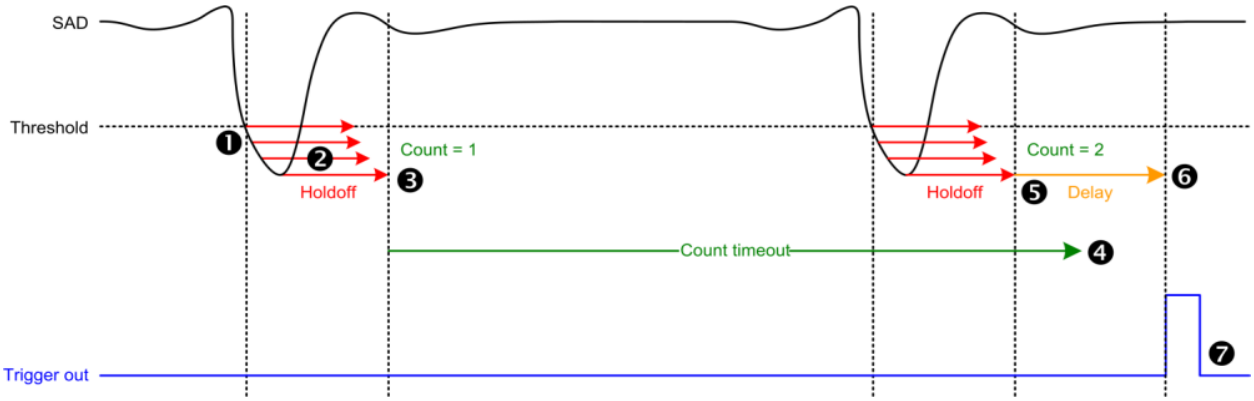
Om een opgeslagen patroon te herkennen wordt er gebruik gemaakt van de Sum of Absolute Differences(SAD). De SAD wordt berekend door van elke meetpunt het verschil te tellen met het opgeslagen patroon. In Afbeelding 15 is A het ingangssignaal, B het patroon en ABS diff. het verschil.

A	-26	-1	-52	9	115	-71	
B	60	96	54	-72	-82	118	-
Diff.	-86	-97	-106	81	197	-189	
ABS diff.	86 +	97 +	106 +	81 +	197 +	189	➔ SAD = 756

A	74	111	123	64	58	-35	
B	78	110	128	66	54	-39	-
Diff.	-4	1	-5	-2	4	4	
ABS diff.	4 +	1 +	5 +	2 +	4 +	4	➔ SAD = 20

Afbeelding 15 SAD-berekening

Om met deze SAD-waarde een patroon te herkennen kunnen een aantal instellingen opgegeven worden. In Afbeelding 16 zijn onderdelen genummerd en deze worden uitgelegd op de volgende bladzijde.



Afbeelding 16 Werking voor het genereren van een trigger

1: SAD-waarde is lager dan de drempelwaarde.
2: De holdoff tijd is de tijd die gewacht wordt op een beter SAD-waarde, als binnen de ingestelde holdoff tijd een lagere SAD-waarde is gemeten wordt de holdoff tijd gereset.
3: Wanneer de holdoff tijd is verlopen wordt de count met één opgehoogd. Wanneer de ingestelde waarde is bereikt wordt de volgende stap uitgevoerd..
4: Als de ingestelde waarde is bereikt binnen de count timeout wordt de volgende stap uitgevoerd, anders wordt de count gereset.
5: De count wordt opgehoogd en de ingestelde waarde is bereikt.
6: De ingestelde delay wordt gewacht.
7: Een triggersignaal wordt gegenereerd.

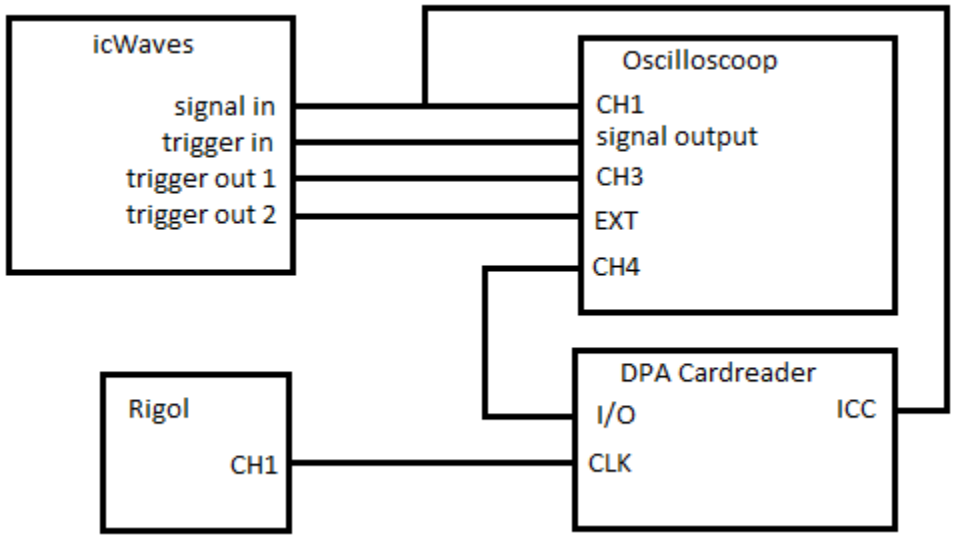
Voor het generen van een triggersignaal moet de icWaves in een trigger-arm-mode gezet worden. Er zijn drie verschillende modes beschikbaar voor de icWaves. De eerste mode is disarm en hiermee wordt aangegeven dat er geen triggersignalen gegenereerd moet worden. De tweede mode is arm always, dit betekent dat er een triggersignaal wordt gegenereerd wanneer er voldaan is aan de eisen van Afbeelding 16. De derde mode is arm by trigger en hiermee moet ook aan de eisen van Afbeelding 16 worden voldaan. Bij de laatste mode is er nog een extra eis, namelijk dat op de trigger in poort van de icWaves een logisch hoog signaal moet staan.

Als laatste zijn er aan deze klasse vier methodes toegevoegd voor het instellen van het bereik van de filter en het veranderen van de filterfrequentie.

7.4 Testopstelling en metingen opnemen

In de vorige sprint is er gebruik gemaakt van een testopstelling met een functiegenerator, een oscilloscoop en de icWaves. Dit kan gebruikt worden voor of het opnemen van data, maar werkt minder goed om een patroon te herkennen. Als er een sinus gegenereerd wordt zal steeds hetzelfde signaal herkend worden, terwijl het de bedoeling is dat juist een afwijkend/opvallend patroon wordt herkend. Om een patroon te herkennen is er een nieuwe testopstelling nodig om een signaal te genereren.

De nieuwe opstelling maakt gebruik van de apparaten uit de vorige opstelling. Voor de nieuwe opstelling is er een Differential Power Analysis(DPA) kaartlezer toegevoegd. Deze kaartlezer meet het stroomverbruik van een smart card als een cryptografische berekening wordt uitgevoerd. Een schematische weergave van de opstelling is te zien in Afbeelding 17.



Afbeelding 17 Testopstelling versie 2

Op de DPA kaartlezer zit een ICC uitgang om het stroomverbruik te meten. De CLK ingang verwacht een kloksignaal. De I/O poort geeft de ingangssignalen en uitgangssignalen weer van de kaartlezer. De functiegenerator Rigol heeft een uitgang die het kloksignaal genereerd voor de kaartlezer. De oscilloscoop heeft een EXT ingang om een meting op te slaan zodra er een triggersignaal herkend wordt.

Hieronder staan de vier stappen die nodig zijn voor de patroonherkenning en de generatie van het triggersignaal.

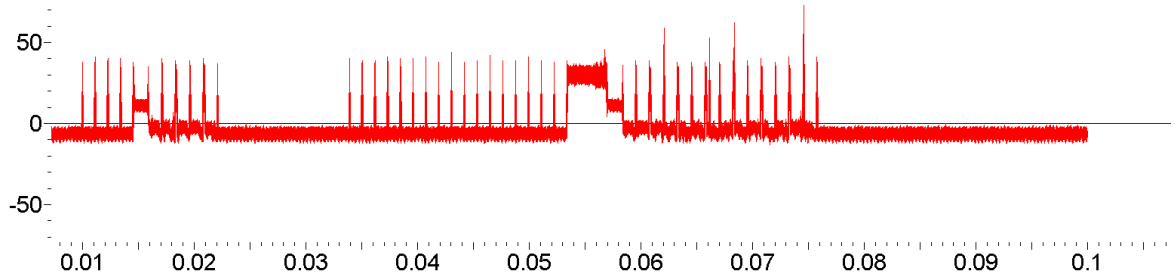
De eerste stap is de oscilloscoop instellen zodat deze een trigger genereerd voordat een interessante eigenschap wordt gezien in het stroomverbruik. Hierdoor neemt de icWaves alleen data op vlak voordat er een interessant patroon komt, waardoor met een hogere snelheid data opgenomen kan worden.

De volgende stap is het weergeven van de opgenomen data in de Matrix en een patroon opslaan op de icWaves. Het selecteren van een patroon kan zowel met de Script Engine als met de gebruikersinterface. De werking van het patroon instellen met de gebruikersinterface wordt uitgelegd in hoofdstuk 7.5. De code in de Script Engine is bijna hetzelfde als de code van de menu actie, waarvan de code te zien is in hoofdstuk 7.5.

De derde stap is het kiezen van een goede drempelwaarde. De SAD-berekening maakt gebruik van een meting in het plot-scherm waarin het opgeslagen patroon op de icWaves herkend moet worden.

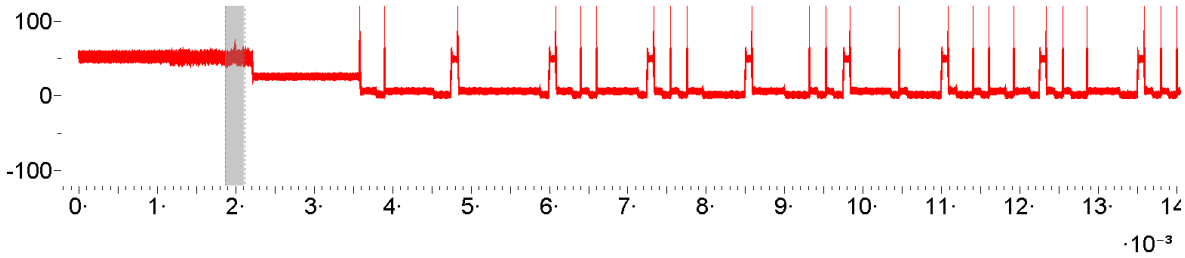
De laatste stap is het instellen van een trigger mode, zodat de icWaves het ingangssignaal gaat vergelijken en een triggersignaal stuurt wanneer het patroon herkend is. Dit triggersignaal kan vervolgens gebruikt worden om een actie uit te voeren.

In Afbeelding 18 wordt het stroomverbruik weergegeven van een DES berekening op de smart card in de DPA kaartlezer.



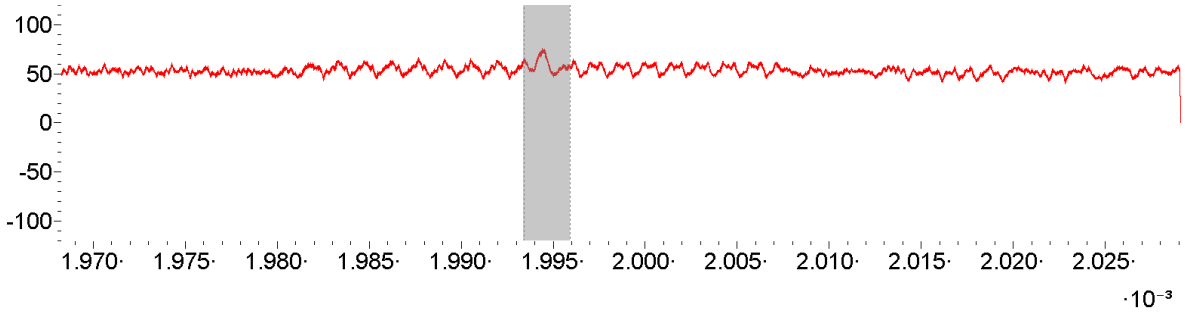
Afbeelding 18 Volledig stroomverbruik DES berekening

Vervolgens is in Afbeelding 19 een deel geselecteerd van het patroon waar een piek zichtbaar is die herkend moet worden door de icWaves.



Afbeelding 19 Selectie interessant deel stroomverbruik

In Afbeelding 20 is de uiteindelijke selectie te zien. Als de icWaves nu klaar wordt gezet om een trigger te genereren en de drempelwaarde is ingesteld wordt het patroon herkend en een triggersignaal verstuurd.

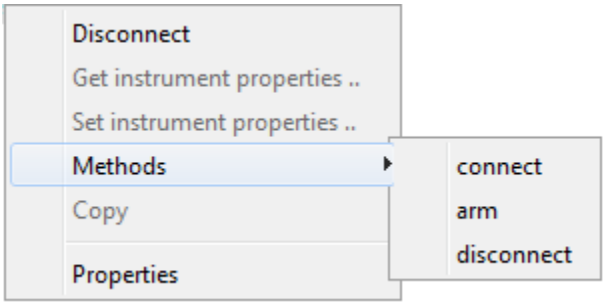


Afbeelding 20 Werkelijke selectie van het patroon

7.5 Gebruikersinteractie met de module

In de Matrix kunnen acties op verschillende manieren worden uitgevoerd met de gebruikersinterface. Hieronder zijn twee van deze manieren beschreven.

De eerste manier werkt automatisch en alleen met methodes, die geen parameters nodig hebben en niets teruggeven. Deze methodes kunnen gestart worden door op een instrument te klikken. Hierdoor wordt een menu weergegeven zoals te zien is in Afbeelding 21. Op het moment dat er geklikt wordt op methods worden alle beschikbare methodes weergegeven. In dit geval zijn er drie methodes beschikbaar die uitgevoerd kunnen worden, dit zijn connect, arm en disconnect.



Afbeelding 21 Eigenschappenmenu icWaves

De tweede manier is het toevoegen van één of meer knoppen aan de menubalk van de Matrix. Hierdoor kan een gebruiker gemakkelijker acties uitvoeren als bij het gebruik van de Script Engine. De actie, die wordt toegevoegd, moet een geselecteerd patroon uit het plot-scherm opslaan op de icWaves. Om ervoor te zorgen dat veranderingen aan de Plot klasse minder invloed hebben op de werking van de acties is gebruik gemaakt van een interface die de volgende methodes heeft:

- Het aantal grafieken opvragen die zichtbaar zijn in het plot-scherm.
- Een grafiek opvragen met behulp van een indexnummer.
- Het aantal selecties opvragen.
- Een selectie opvragen met behulp van een indexnummer.
- Een lijst van selecties opvragen.

Met de hierboven genoemde methodes is het mogelijk om altijd vanuit de eerste grafiek een selectie te verkrijgen en deze als patroon in te stellen. Er is hiervoor gekozen, zodat het voor de gebruiker duidelijk is en er geen problemen ontstaan wanneer er met de muis op een andere grafiek is geklikt.

De code van de actie om een patroon in te stellen is te zien in onderstaand codefragment.

```
@Override
   public void actionPerformed(ActionEvent ev) {
      //get waveform selection list.
      List<WaveformSelection> selectionlist = context.getWaveformSelectionList();
```

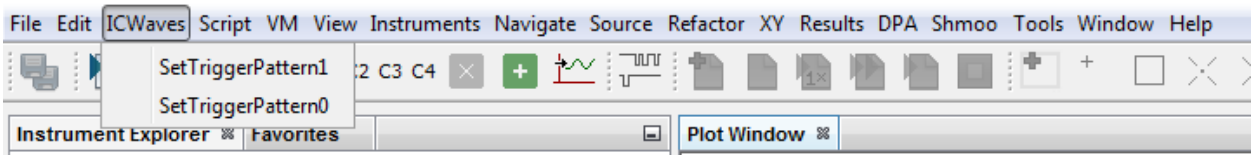
```
        //get the user selection from the list
        WaveformSelection selection = selectionlist.get(0);
        double begin = selection.begin();
        double end = selection.end();

        //get the whole waveform
        Waveform pattern = context.getWaveform(0);
        int minX = pattern.timeToIndexClosest(begin);
        int maxX = pattern.timeToIndexClosest(end);
        int length = maxX - minX;

        //copy the data from the selection to a new array that is used to set the pattern.
        byte[] patternArray = new byte[length];
        byte[] totalWaveform = (byte[]) pattern.getValues();
        System.arraycopy(totalWaveform,minX,patternArray,0,length);

        //get the instrument from the InstrumentManager with the label icwaves.
        final ICWaves icwaves = InstrumentManager.getInstrument(ICWAVES,ICWaves.class);
        if (icwaves == null) {
            String message = "Could not find instrument with label " + ICWAVES;
            throw new NullPointerException(message);
        }
        try {
              icwaves.setPattern(0, patternArray);
        } catch (IOException ex) {
              Exceptions.printStackTrace(ex);
        }
    }
```

Codefragment 4 Menuactie om een patroon op te slaan

In de bovenstaande code wordt als eerste een lijst van selecties opgehaald. Van deze lijst wordt het eerste element opgevraagd en het begin- en eindtijdstip opgeslagen. Hierna wordt de positie van de selectie in de totale meting berekend door middel van de verkregen tijd om te zetten naar twee indexwaarden. Met de indexwaarden wordt een nieuwe array aangemaakt en hierin worden de meetwaardes van de selectie opgeslagen. Het is niet mogelijk om gelijk de waardes van de selectie op te vragen, omdat de selectie losgekoppeld is van de daadwerkelijke metingen.

De selectie moet vervolgens opgeslagen worden op de icWaves en hiervoor wordt het icWaves instrument opgevraagd met behulp van de InstrumentManager klasse. Deze klasse maakt gebruik van een standaardlabel. Dit houdt in dat de icWaves altijd hetzelfde label moet krijgen, omdat anders het instrument niet gevonden wordt. De laatste stap is het opslaan van het patroon op de icWaves. Het selecteren van het juiste label wordt in de volgende sprint verbeterd, zodat het niet nodig is om een standaardlabel te gebruiken.

Als de klasse is gemaakt met de bovenstaande code ziet de menubalk van de Matrix er uit als Afbeelding 22



Afbeelding 22 Menubalk Matrix

De menubalk heeft een menuknop ICWaves. Wanneer er op deze knop geklikt wordt kunnen twee acties geselecteerd worden, dit is het opslaan van patroon nul of patroon een. Voor deze sprint is de werking van deze acties voldoende.

De huidige werking van de gebruikersinterface is niet ideaal, maar het is al gebruiksvriendelijker dan een patroon in stellen met behulp van de Script Engine. Het selecteren van een patroon wordt in de volgende sprint verbeterd, zodat de selecties automatisch wordt aangepast aan de maximale selectie grootte.

7.6 Javadoc en testen

Tijdens deze sprint is er wederom gebruik gemaakt van Javadoc om documentatie te generen. Dit kan binnen kortere tijd worden gemaakt, omdat de werking van Javadoc al is uitgezocht. Er zijn tijdens deze sprint ook nieuwe integratietests geschreven, maar als eerste is er gezocht naar een programma die de codedekking kan weergeven van de tests.

Bij het gebruik van een dergelijk programma moet er rekening mee worden gehouden, dat als alle code is uitgevoerd niet alle combinaties van situaties zijn getest. Het gebruik van een programma zorgt ervoor dat eenvoudig gecontroleerd kan worden of alle code getest is. Dit geeft de programmeur een beter overzicht van de testcode. Hiervoor zijn er twee verschillende plug-ins bekeken die beschikbaar zijn voor de Netbeans ontwikkelomgeving.

De eerste plug-in is een code coverage plug-in voor Netbeans. [22] Deze plug-in is ondersteund voor Netbeans 7.3, terwijl er gebruik wordt gemaakt van versie 8.0. Een voordeel van de plug-in is dat deze geïntegreerd is met JUnit. De plug-in geeft aan of een regel code volledig of gedeeltelijk is uitgevoerd. Dit wordt aangegeven door middel van een groene of een gele kleur. De codedekking van de tests wordt in een rapport weergeven.

De tweede plug-in is tikione jacocoverage [23]. Deze plug-in is wel beschikbaar voor versie 8.0. De plug-in kan ongeveer hetzelfde als de eerste plug-in, alleen geeft deze plug-in ook de regels aan die niet getest zijn. Volgens de beschrijving op github worden modules ondersteund, terwijl dit niet is aangegeven op de Netbeans website.

Na het installeren van de tweede plug-in blijkt dat modules niet ondersteund worden. Hierdoor is het niet mogelijk om de codedekking te meten met de tweede plug-in. De eerste plug-in blijkt na het installeren wel te werken met netbeans versie 8.0 en is daarom gekozen.

In Afbeelding 23 zijn de bestanden te zien waarvan de codedekking is berekend met de plug-in.

com.brightsight.icwaves.actions.Bundle	0% (0 / 4)
com.brightsight.icwaves.actions.SetTriggerPattern0	0% (0 / 33)
com.brightsight.icwaves.actions.SetTriggerPattern1	0% (0 / 28)
com.brightsight.icwaves.instruments.DeviceStructure	100% (4 / 4)
com.brightsight.icwaves.instruments.ICWaves	85% (166 / 195)
com.brightsight.icwaves.instruments.ICWavesCommunicator	92% (155 / 168)
com.brightsight.icwaves.instruments.ICWavesCommunicatorProvider	83% (15 / 18)
com.brightsight.icwaves.instruments.ICWavesInstrumentProvider	0% (0 / 4)
com.brightsight.icwaves.instruments.VersionInfoStructure	100% (1 / 1)
com.brightsight.icwaves.instruments.WrapperManager	83% (5 / 6)

Afbeelding 23 Overzicht testdekking

Wat gelijk opvalt, is dat er in een aantal bestanden geen code is uitgevoerd. De Bundle bevat alleen de naam van het project en heeft niets te maken met de werking van de testen. Vervolgens zijn er twee acties die niet getest zijn, omdat hiervoor gebruikersinteractie en verschillende klassen nodig zijn die niet goed te simuleren zijn. Voor het testen van de acties is het zinvoller om een scenario te schrijven en deze uit te voeren tijdens de systeemtest. De laatste klasse die niet getest wordt is de InstrumentProvider, omdat deze ervoor zorgt dat de ICWaves klasse als instrument herkend wordt. Dit heeft te maken met de werking van de Matrix en heeft geen invloed op de werking van de andere tests.

Er zijn twee klassen die helemaal uitgevoerd zijn tijdens het testen. Dit zijn de klassen die nodig zijn om aan te geven welk apparaat gebruikt wordt.

Als laatste zijn er een aantal klassen te zien die een codedekking hebben van ongeveer 85%. De ICWaves klasse heeft geen volledige dekking, omdat de klasse een aantal methodes bevat die niet goed getest kunnen worden. Dit is bijvoorbeeld het berekenen van de drempelwaarde.

De ICWavesCommunicator is niet helemaal getest, omdat de klasse er een lege methode heeft. De tweede reden dat er geen volledige dekking is, komt doordat niet alle mogelijke foutmeldingen zijn getest. De redenering hierbij is dat als een aantal van de foutmeldingen werken de rest ook werkt.

De ICWavesCommunicatorProvider is niet volledig getest, omdat deze methodes bevat die alleen worden aangeroepen door de Matrix en niet gemakkelijk te testen zijn. De methodes die door de communicator worden gebruikt zijn wel getest.

7.7 Samenvatting

Tijdens deze sprint is er gewerkt aan het toevoegen van functionaliteit om een patroon te selecteren en vervolgens op te slaan. Het opgeslagen patroon kan herkend worden in het ingangssignaal en vervolgens wordt een triggersignaal verstuurd. Om een patroon op te slaan op de icWaves zijn er acties aan de gebruikersinterface toegevoegd. De gebruikersinterface mist nog een aantal opties en wordt daarom verbeterd in een volgende sprint. Vervolgens zijn er tests geschreven om de nieuwe functionaliteit te testen. Voor een beter overzicht van de uitgevoerde tests wordt er gebruik gemaakt van een plug-in.

8 Gebruikersinterface

8.1 Introductie

In dit hoofdstuk worden de keuzes, die tijdens het ontwikkelen en verbeteren van de gebruikersinterface, in vier delen behandeld.

Het eerste onderdeel is het selecteren van de juiste icWaves waarop de patronen opgeslagen worden. Het tweede onderdeel is het selecteren en het opslaan van dit patroon op de icWaves. Als derde onderdeel wordt het berekenen van een drempelwaarde met behulp van de Sum of Absolute Differences besproken. Het laatste onderdeel behandeld het eigenschappenscherm dat gebruikt wordt om de instellingen van de icWaves weer te geven en aan te passen.

8.2 Ontwerpen gebruikersinterface

Voor het ontwerpen van de gebruikersinterface is als eerste bedacht welke acties een gebruiker moet uitvoeren. Deze acties kunnen globaal in twee groepen worden verdeeld.

Als eerste is er een groep van methodes die parameter waardes aanpast. Deze methodes zorgen bijvoorbeeld voor het instellen van de bemonsteringssnelheid en het aantal te vergelijken patronen.

De tweede groep bestaat uit methodes die de daadwerkelijke acties uitvoeren en gebruik maken van de parameters die door de eerste groep van methodes zijn ingesteld. Dit zijn de methodes voor het uitvoeren van een meting, het opslaan van een patroon en het berekenen van de SAD-waarde.

Het uitvoeren van een meting valt vervolgens af als onderdeel waarvoor een gebruikersinterface wordt gemaakt. Hiervoor zijn al knoppen in de Matrix beschikbaar, omdat de icWaves zich kan gedragen als een oscilloscoop.

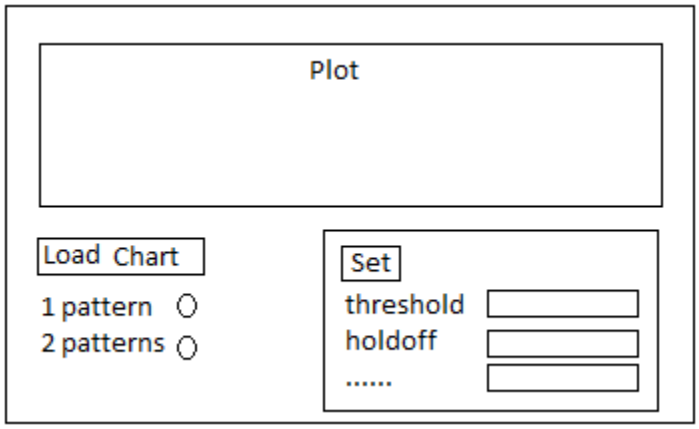
De volgende onderdelen blijven over:

- De groep van methodes die parameter waardes aanpast.
- Het opslaan van een patroon.
- Het berekenen van een SAD-waarde.

Voor de groep van methodes, die parameters waardes aanpast, kan er gebruik gemaakt worden van een eigenschappenscherm dat al binnen de Matrix beschikbaar is. Voor zowel het opslaan van een patroon als het berekenen van een SAD-waarde moet een gebruikersinterface worden ontworpen. Als laatste is een ontwerp nodig voor de icWaves selectie, zodat het mogelijk is om meer dan één icWaves te gebruiken met de menuacties.

Nadat het duidelijk is welke onderdelen een gebruikersinterface nodig hebben zijn er een aantal ontwerpen gemaakt van schermen. Hierbij is er als eerst gekeken naar het patroonselectiescherm in de Inspector software.

Hierin is het mogelijk om de patrooninstellingen aan te passen en één of twee patronen op de icWaves te zetten. Bij onderstaande schets is nog het plot-scherm toegevoegd waarin het patroon aangegeven kan worden en een radioknop die aangeeft of instellingen zichtbaar moeten zijn voor één of twee patronen.



Afbeelding 24 Eerste ontwerp gebruikersinterface

Na het maken van een aantal ontwerpen is er een overleg geweest met de begeleiders. Hierin zijn de ontwerpen besproken met als resultaat dat het scherm te groot is. Dit komt voornamelijk doordat het plot-scherm vaak al los geopend is in de Matrix. Bij dit ontwerp nemen de instellingen van een patroon onnodige ruimte in, omdat deze ook veranderd kunnen worden in het eigenschappenscherm.

Uit het overleg met de begeleiders is duidelijk geworden dat de interface onderdelen niet te groot moeten zijn en indien mogelijk moeten de onderdelen zich dynamisch opbouwen. Na het gesprek is besloten om voor de icWaves selectie, de patroonselectie en de SAD-berekening te beginnen met een eenvoudige menuknop..

8.3 ICWaves selectie

Bij het ontwikkelen van de gebruikersinterface wordt een eenvoudige ontwerp gekozen. Deze manier is gekozen, zodat er altijd een werkende versie kan worden opgeleverd. Als dit ontwerp geïmplementeerd is worden de mogelijkheden uitgebreid om aan de uiteindelijke wens van de opdrachtgever te voldoen.

Een eenvoudige manier om een label te selecteren met behulp van een menuknop is een pop-up scherm weer te geven. In dit scherm kan vervolgens het label van de icWaves worden ingevuld.

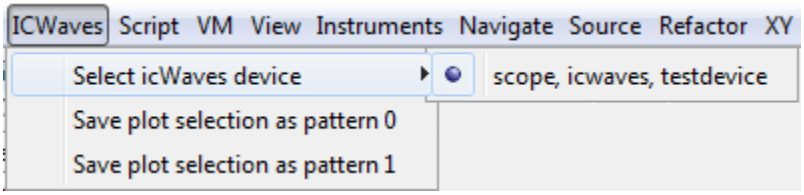
Het gebruik van een pop-up scherm is gemakkelijk te implementeren, maar niet erg gebruikersvriendelijk. In overleg met de opdrachtgever is de volgende stap het toevoegen van het dynamische opbouwen van een lijst van de verbonden icWaves apparaten. Wanneer er één apparaat beschikbaar is moet deze automatisch gekozen worden.

8.3.1 Dynamische opbouwen apparaatlijst

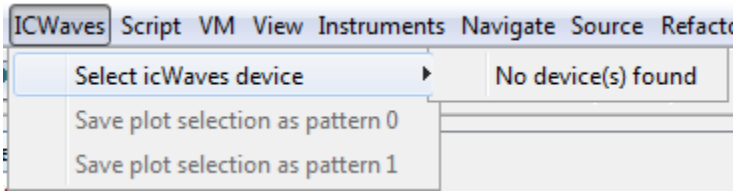
In het menu start de actie om dynamisch de lijst met beschikbare icWaves instrumenten op te bouwen.

De actie start met het verwijderen van de bestaande apparaatlijst, zodat niet gecontroleerd moet worden op labels die niet meer bestaan. Als het menu leeg is kunnen alle labels worden opgevraagd in een aparte methode, waarbij alle instrumentlabels gezocht worden die horen bij de ICWaves klasse. Het is in deze situatie mogelijk dat er meer labels worden teruggegeven dan verbonden apparaten. Om ervoor te zorgen dat alleen uniek verbonden apparaten overblijven worden deze in een set gestopt. Doordat in een set alleen maar unieke objecten bestaan blijven de unieke apparaten over. Van deze apparaten wordt het label opgevraagd en in een lijst gestopt die terug wordt gegeven.

Voor elke label in de teruggegeven lijst wordt een knop aangemaakt met het label als naam. Door gebruik te maken van radioknoppen is het mogelijk om bij selectie een cirkel voor de knop te zetten. In het geval dat er een lege lijst wordt teruggeven wordt een standaardknop geladen met een tekst die aangeeft dat er geen apparaten gevonden zijn. In afbeelding 25 en 26 is een lijst te zien met elementen en een lijst zonder elementen.



Afbeelding 25 Menu met een geselecteerd apparaat



Afbeelding 26 Menu wanneer geen apparaten gevonden zijn

8.3.2 Automatische selectie icWaves

Automatisch een verbonden icWaves selecteren heeft als voordeel dat er niet op het menu geklikt moet worden, wanneer er maar één icWaves verbonden is. Zijn er meer icWaves verbonden, dan wordt er niets geselecteerd en een foutmelding weergegeven. Er is hiervoor gekozen, omdat het programma niet weet welk apparaat gebruikt moet worden.

8.4 Patroon selectie

Het selecteren van een patroon is tijdens deze sprint ook verbeterd. Zoals is uitgelegd in Hoofdstuk 7.5 is het mogelijk om van het bestaande plot-scherm een selectie op te vragen en deze in te stellen als patroon. Met de huidige werking wordt er geen rekening gehouden met het apparaat dat geselecteerd is en de maximale selectie lengte.

Om hier rekening mee te houden wordt de geselecteerde icWaves opgevraagd van de klasse die de icWaves selectie regelt. Zie hoofdstuk 8.3.2 Automatische selectie icWaves.

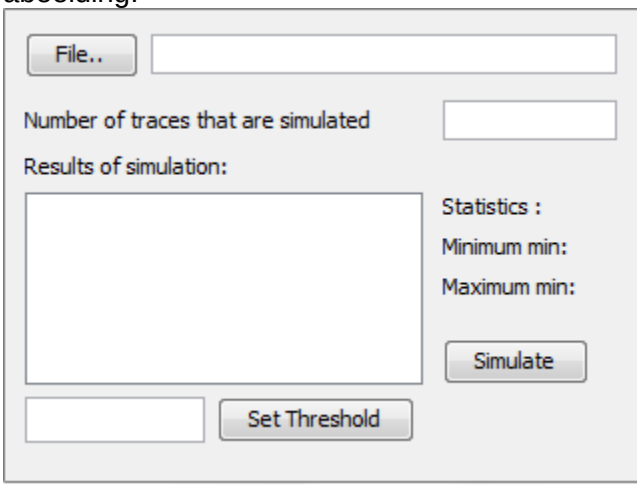
Worden er meer punten geselecteerd dan opgeslagen kunnen worden als patroon zijn er twee acties die uitgevoerd kunnen worden. Met de huidige werking wordt er een foutmelding gegeven en moet de gebruiker ervoor zorgen dat een kleiner aantal punten wordt geselecteerd.
De tweede optie is het veranderen van de selectie, zodat deze gebruikt kan worden als een patroon. Het verkleinen van de selectie is tijdens deze sprint toegevoegd. In overleg met de opdrachtgever is gekozen om altijd het beginpunt van de selectie te houden en het eindpunt zo te verschuiven dat de maximale patroonlengte wordt geselecteerd. Een melding wordt gegeven als de selectie is veranderd. De gebruiker kan vervolgens de selectie verplaatsen in het plot-scherm zonder dat de lengte veranderd.

8.5 SAD-berekening

Het eerste ontwerp moet voldoen aan de volgende punten:

- De minimale berekende SAD-waarde moet weergegeven worden, dit geeft inzicht in de laagste waarde die als drempelwaarde ingesteld kan worden.
- Meer berekeningen uitvoeren
- Alle berekende minimale waardes weergeven
- Verschillende ingangssignalen gebruiken

Het eerste ontwerp dat gemaakt is naar aanleiding van deze punten is de zien in onderstaande abeelding.



Afbeelding 27 SAD-berekeningsscherm

In dit ontwerp wordt een bestand opgegeven waaruit het ingangssignaal gehaald wordt. Er is een veld beschikbaar waarin het maximum aantal ingangssignalen uit het bestand aangegeven wordt. In het witte vlak wordt per berekening een minimale en een maximale waarde weergegeven. Van alle metingen die zijn uitgevoerd wordt de minimale minimumwaarde en de maximale minimumwaarde weergegeven.

Dit ontwerp is besproken met de opdrachtgever en tijdens dit gesprek zijn een aantal grote nadelen naar voren gekomen. Het is nu niet mogelijk om te zien hoe vaak de berekening bij de minimale waarde in de buurt komt. Het kan gebeuren dat het patroon dat gezocht wordt overal in het ingangssignaal aanwezig is, zonder dat dit gedetecteerd wordt.

Om de problemen op te lossen is het gebruik van het plot-scherm nader bekeken. Als alle data van de berekening in het plot-scherm wordt weergegeven is het gemakkelijk te zien waar de minimale waardes zich bevinden. Een voorbeeld van een berekend SAD-signaal is te zien in afbeelding 28.

Afbeelding 28 SAD berekening in het plot-scherm

In afbeelding 28 is in het rood het ingangssignaal te zien en in het groen het resultaat van de SAD-berekening. De positie van de grijze balk is de selectie die gebruikt is als patroon. In deze Bij de selectie worden de laagste SAD-waardes bereikt. In de Matrix is het mogelijk om de waarde van het laagste punt af te lezen.

Het berekenen is mogelijk voor één ingangssignaal, maar er is tijdens dit project geen tijd om de SAD-berekening goed te verwerken in de gebruikersinterface. In overleg met de opdrachtgever

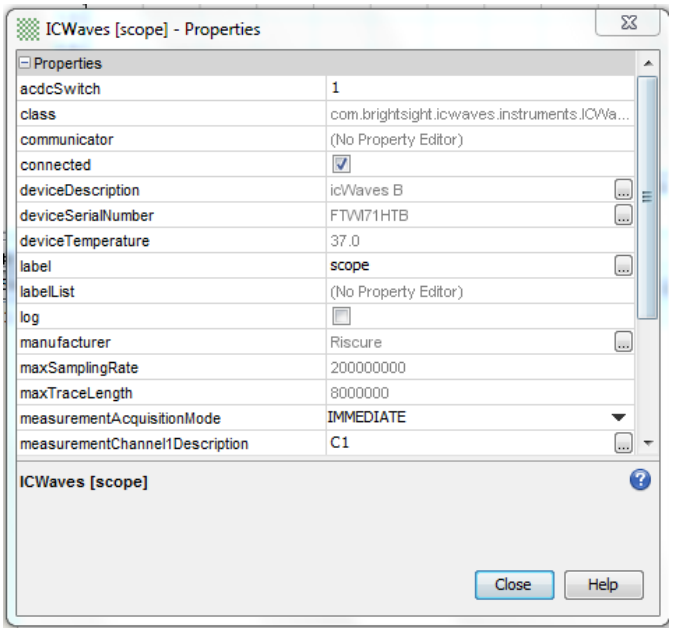
is ervoor gekozen om deze berekening niet in de gebruikersinterface te stoppen en alleen beschikbaar te maken met de Script Engine. Hiervoor wordt wel een voorbeeldscript gemaakt.

8.6 Eigenschappenscherm

De instellingen van de icWaves kunnen veranderd worden in het automatisch aangemaakte eigenschappenscherm. Het scherm zorgt ervoor dat een gebruiker op een simpele manier een overzicht kan krijgen van de verschillende instellingen van de icWaves. Zo kan het scherm bijvoorbeeld gebruikt worden om een label van de icWaves in te stellen of patrooninstellingen aan te passen.

Het eigenschappenscherm wordt zichtbaar gemaakt met behulp van JavaBeans [24]. De programmeur maakt gebruik van get en set methodes om instellingen zichtbaar te maken in het scherm. Voor de get en set methodes gelden een aantal eisen. De set methodes moeten één parameter meekrijgen en mogen niets teruggeven. De get methodes mogen alleen iets teruggeven, maar kunnen geen parameters gebruiken. Een andere eis is dat de methodes public zijn. Wanneer JavaBeans gebruikt wordt met standaard datatypes, zoals integers of strings kan dit weergegeven worden zonder problemen. Het is zelfs mogelijk om een enumeratie als parameter te gebruiken.

Door de limitatie op de parameters is in overleg met de opdrachtgever gekozen om een aantal extra get en set methodes toe te voegen voor de instellingen van een patroon. Dit zijn methodes die een vast getal gebruiken om aan te geven of de instellingen van patroon nul of patroon één worden aangepast. Het voordeel hiervan is dat de opties ingesteld kunnen worden met behulp van het eigenschappenscherm. Zie Afbeelding 29.



Afbeelding 29 Eigenschappenscherm icWaves

Er zijn een aantal instellingen grijs gekleurd in Afbeelding 29, wat betekent dat alleen get methodes beschikbaar zijn. Een selectievakje wordt geladen bij gebruik van een boolean. Het label maakt gebruik van een string en als er op het veld geklikt wordt kan deze ingevuld worden.

Bij het instellen van de measurementAcquisitionMode wordt een lijst zichtbaar met alle beschikbare waardes uit de enumeratie.

8.7 Samenvatting

Tijdens deze sprint is er gewerkt aan het toevoegen en verbeteren van de gebruikersinterface. Hiervoor is de gebruikersinterface verdeeld in vier verschillende onderdelen.

Het eerste onderdeel zorgt voor de selectie van de icWaves waarop het patroon opgeslagen moet worden. Dit gebeurt door dynamisch een lijst te laden met de labels van een verbonden apparaat. Wanneer er één icWaves is verbonden, wordt deze automatisch geselecteerd.

Het tweede onderdeel is het selecteren van een patroon. Voor het opslaan van deze patronen zijn twee knoppen toegevoegd. Wanneer een geselecteerd patroon te groot is voor de icWaves wordt de selectie verkleind en opgeslagen.

Het derde onderdeel is het berekenen van de SAD. Hiervoor zijn verschillende ontwerpen gemaakt, maar dit menu onderdeel is in overleg met de opdrachtgever niet in de uiteindelijke versie van de module toegevoegd.

Het laatste onderdeel is het eigenschappenscherm. Dit scherm kan gebruikt worden om de instellingen van de icWaves aan te passen met de gebruikersinterface. In dit scherm is gemakkelijk te zien wat de huidige instellingen zijn en welke waardes aanpasbaar zijn.

9 Conclusie

Het doel, dat aan het begin van dit project is opgesteld, is een module ontwikkelen waarmee de icWaves trigger box aangestuurd kan worden vanuit de Matrix.

Dit doel is verdeeld in de onderstaande onderdelen:

- Er moet een signaal opgenomen worden met de icWaves. Dit kan bijvoorbeeld het stroomverbruik zijn van een microcontroller die getest wordt.

- Er moet een patroon herkend worden, dat bestaat uit een aantal opeenvolgende meetpunten die geselecteerd worden uit het opgenomen signaal.

- Het opgeslagen signaal moet herkend worden, waarna een signaal gegenereerd wordt. Het signaal dat gegenereerd wordt kan bijvoorbeeld een laser laten weten dat een fout geïnjecteerd moet worden

Doordat er een signaal opgenomen kan worden met de bestaande menu onderdelen van de oscilloscoop of vanuit de Script Engine is het eerste onderdeel van het doel gehaald.

Het tweede onderdeel is verwezenlijkt, doordat het mogelijk is om in het plot-scherm een selectie te maken en deze vervolgens op te slaan met de knoppen in het ICWaves menu.

Het laatste onderdeel van het doel is behaald, doordat het mogelijk is om vanuit het eigenschappenscherm of de Script Engine de trigger mode en de patrooninstellingen aan te passen.

10 Evaluatie

In dit hoofdstuk wordt kort uitgelegd wat er geleerd is tijdens dit project en vervolgens wordt het proces, het resultaat en de beroepstaken geëvalueerd.

10.1 Kennis opgedaan

Tijdens dit project is kennis opgedaan over verschillende onderdelen en hiervan worden een aantal belangrijke punten genoemd. Met dit project is kennis opgedaan van Java, omdat de afstudeerder nog geen ervaring had. Vervolgens is meer inzicht verkregen over het testproces en het schrijven van documentatie met Javadoc. Als laatste is geleerd hoe apparaten, zoals een oscilloscoop of een functiegenerator gebruikt worden.

10.2 Proces

Tijdens de oriëntatiefase van drie weken is Scrum gekozen als ontwikkelmethode met sprints van vier weken. De laatste twee weken van dit project zijn geen onderdeel van Scrum en worden gebruikt om het project af te ronden.

Het doel van de oriëntatiefase was om de probleemstelling duidelijk te krijgen en een plan van aanpak te maken. Daarnaast is deze fase gebruikt om Java kennis op te doen. Na drie weken was genoeg kennis opgedaan om te starten met de ontwikkeling van de module.

Vervolgens zijn er drie sprints uitgevoerd van vier weken. Vier weken was een goede keuze, omdat er nu genoeg tijd was om eisen duidelijk te krijgen en onverwachte problemen bij de implementatie op te lossen. Bij dit project was het niet mogelijk geweest om sprints van twee weken te kiezen, omdat er dan niet genoeg tijd was geweest om de software te testen en documentatie te schrijven. In de eerste sprint waren te veel taken gepland en daarom was het ontwikkelen van de patroonherkenning naar de tweede sprint verplaatst.

Aan het eind van dit project waren er twee weken ingepland om het project af te ronden. Twee weken is genoeg om het project af te ronden, maar het was beter geweest om voor deze tijd al een keer de software op een andere pc te installeren. Nu waren hier een aantal problemen mee die pas ontdekt werden tijdens het voorbereiden van de presentatie op een laptop.

Tijdens het project zijn verschillende vragen gesteld aan zowel de opdrachtgever als de begeleider. Deze vragen waren vooral informeel en hiervan zijn geen uitwerkingen gemaakt. In dit bedrijf maak dat niet uit, omdat er korte lijnen zijn en genoeg momenten om te overleggen. Bij een vervolgproject is het een goed idee om ook formeel vragen te stellen en de antwoorden op te schrijven, zodat de afspraken op papier staan.

10.3 Resultaat

Aan het eind van het project is de code van de module opgeleverd met de bijbehorende tests. Dit product voldoet aan het doel dat aan het begin van het project is opgesteld en de icWaves kan aangestuurd worden vanuit de Matrix. Het enigste onderdeel wat nog niet naar behoren werkt is het weergeven van de berekende SAD-waardes in het plot-scherm. Dit werkt nu vanuit de Script Engine, met het eerste signaal uit het plot-scherm en het opgeslagen patroon. Het berekenen moet nog uitgebreid worden, zodat het werkt met een bestand en een menu in de Matrix.

10.4 Beroepstaken

A1 Analyseren van het probleemdomein
Om het probleem van de opdrachtgever te begrijpen zijn er verschillende informele gesprekken gehouden en is de bestaande software en de programmeertaal Java onderzocht. Dit is allemaal gebeurd tijdens het uitvoeren van de oriëntatiefase.

A4 Kiezen van een ontwikkelstrategie en ontwikkelmethodiek
Op hetzelfde moment als de oriëntatiefase is een keuze gemaakt voor een ontwikkelmethode. Hierbij is de randvoorwaarde van de opdrachtgever gebruikt om de doorslaggevende beslissing te maken voor de methode. Dit is te lezen in Hoofdstuk 4.

A5 Opstellen van systeemeisen (requirements)
De eisen zijn opgesteld aan de hand van een gesprek, de documentatie van de trigger box en informatie op het interne netwerk. Dit is te lezen in Hoofdstuk 5.3

C11 Ontwerpen van een mens-machine interface
Tijdens dit project zijn er verschillende ontwerpen gemaakt voor schermen in de Matrix. Het bleek later niet nodig te zijn om nieuwe schermen te maken en daarom is er gekozen om gebruik te maken van bestaande menu onderdelen en menuknoppen. Dit is te lezen in Hoofdstuk 8.

D17 Testen van software systemen
Er is tijdens dit project op drie momenten software opgeleverd, dit is na het uitvoeren van de eerste en tweede sprint en aan het eind van het project. In de eerste sprint zijn de verschillende testniveaus bekeken en is de keuze gemaakt om integratietests en een systeemtest uit te voeren tijdens dit project. De integratietests zijn gemaakt in de eerste twee sprints en de systeemtest is uitgevoerd in de afrondingsfase van dit project. Meer informatie over deze systeemtest is te vinden in Bijlage D.

11 Bronvermelding

De volgende afbeeldingen zijn geheel of gedeeltelijk overgenomen uit de documentatie die bij de Inspector software zit van Riscure: Afbeelding 3,4,15,16.

[1] Rational Software, „Rational Unified Process Best Practices for Software," [Online]. Available: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf. [Geopend 19 januari 2013].

[2] K. Schwaber en J. Sutherland, „The Scrum Guide," [Online]. Available: http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf. [Geopend 19 Januari 2013].

[3] Onbekend, „Feature-Driver-Development-Practices," 14 januari 2002. [Online]. Available: http://www.pearsonhighered.com/assets/hip/us/hip_us_pearsonhighered/samplechapter/0130676152.pdf. [Geopend 20 februari 2014].

[4] D. Wells, „The Rules of Extreme Programming," [Online]. Available: http://www.extremeprogramming.org/rules.html. [Geopend 19 januari 2013].

[5] Riscure, „ICWaves," [Online]. Available: https://www.riscure.com/benzine/documents/ICWaves.pdf. [Geopend 5 mei 2014].

[6] Oracle, „Java Code Conventions," [Online]. Available: http://www.oracle.com/technetwork/java/codeconventions-150003.pdf. [Geopend 5 maart 2014].

[7] Android Developers, „JNI Tips," [Online]. Available: http://developer.android.com/training/articles/perf-jni.html. [Geopend 5 mei 2014].

[8] T. Wall, „JNA API Documentation," [Online]. Available: http://twall.github.io/jna/4.1.0/overview-summary.html#overview_description. [Geopend 5 mei 2014].

[9] The Apache Software Foundation, „Apache license and Distribution FAQ," [Online]. Available: http://www.apache.org/foundation/license-faq.html#License. [Geopend 5 mei 2014].

[10] Netbeans, „Netbeans IDE - Overview," [Online]. Available: https://netbeans.org/features/index.html. [Geopend 5 mei 2014].

[12] G. Lionetti, „ What is Version Control: Centralized vs. DVCS," [Online]. Available: http://blogs.atlassian.com/2012/02/version-control-centralized-dvcs/. [Geopend 19 januari 2013].

[13] Git, „git about distributed," [Online]. Available: http://www.git-scm.com/about/distributed. [Geopend 19 januari 2013].

[14] Mercurial SCM, „Mercurial SCM," [Online]. Available: http://mercurial.selenic.com/about. [Geopend 5 mei 2014].

[15] Oracle, „How to Write Doc Comments for Javadoc Tool," [Online]. Available: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#tag. [Geopend 5 mei 2014].

[16] Software Testing Fundamentals, „Software Testing Levels," [Online]. Available:

http://softwaretestingfundamentals.com/software-testing-levels/. [Geopend 17 april 2014].

[18] Wikipedia, „Integration testing,” 1 mei 2014. [Online]. Available: http://en.wikipedia.org/wiki/Integration_testing. [Geopend 17 april 2014].

[21] Onbekend, „Github JUnit,” [Online]. Available: https://github.com/junit-team/junit/wiki. [Geopend 5 mei 2014].

[22] Onbekend, „Unit Test Code Coverage Plugin,” [Online]. Available: http://plugins.netbeans.org/plugin/38945/unit-tests-code-coverage-plugin-updated-for-netbeans-7-0. [Geopend 5 mei 2014].

[23] J. Lermitage, „tikione-jacocoverage,” [Online]. Available: https://github.com/jonathanlermitage/tikione-jacocoverage. [Geopend 5 mei 2014].

[24] Wikipedia, „Javabeans,” 13 mei 2014. [Online]. Available: http://en.wikipedia.org/wiki/JavaBeans. [Geopend 14 maart 2014].

12 Figuren

13 Bijlage

Bij dit verslag zijn de volgende bijlage ingevoegd:

Bijlage A	Plan van Aanpak
Bijlage B	Scrum takenlijsten
Bijlage C	UML-diagrammen
Bijlage D	Testplan