Afstudeerverslag Mark Smid

Onderzoeken en integreren van een draadloos communicatieprotocol bij DWG

Periode: 8 feb 2021 - 4 jun 2021

Inleverdatum: 4 jun 2021

Bedrijf: DWG

DE HAAGSE HOGESCHOOL

Algemene gegevens

Auteur:
Naam: Mark Smid
Studentnummer:

Stagebedrijf:
Naam: DWG

Locatie: Admiraal Lucashof 5, 3115 HM Schiedam
Website: www.dwg.nl

Stagegegevens:
Studie: HBO-ICT Software Engineering
Studiejaar: 4
Stageperiode: 8 feb 2021 - 4 jun 2021
Inleverdatum verslag: 4 jun 2021 voor 12:00

Begeleiders:
Bedrijfsmentor: Niels IJpelaar
Stagebegeleider: Gerard Mijnarends

Referaat

Om mijn studie aan De Haagse Hogeschool af te ronden heb ik een Software Engineering afstudeerstage gelopen bij een bedrijf genaamd DWG. Er was onder een aantal klanten van DWG de vraag naar een functionaliteit die nog niet beschikbaar was in hun bestaande product ITS (Industrial Things System). Om aan die vraag te kunnen voldoen heeft DWG een stageopdracht geformuleerd die bestond uit het uitbreiden van ITS met een draadloos communicatieprotocol.

Voorwoord

Dit afstudeerverslag is geschreven in het kader van mijn opleiding HBO-ICT Software Engineering aan De Haagse Hogeschool. Het verslag beschrijft mijn afstudeerperiode bij DWG met aandacht op de aanpak, gebruikte methoden en opgeleverde producten.

Ik wil DWG bedanken voor de kans om bij hun af te studeren in deze voor iedereen gekke tijden. Door wat oponthoud aan de kant van school bij het regelen van een portfolio en begeleiders kon ik pas veel later beginnen dan gewild en DWG was zo gracieus om hierop te wachten. Dit waardeer ik natuurlijk enorm. Ook wil ik graag Niels IJpelaar, Mark Franken, Mark Verbaan, Merijn Koster, Marc van Dijk en Giovanni Scholten bedanken voor de hulp en begeleiding tijdens het uitvoeren van mijn opdracht. Deze mensen hebben het voor mij mogelijk gemaakt om mijn opdracht zo goed mogelijk uit te voeren door altijd voor mij klaar te staan als ik ergens hulp bij nodig had. Hier ben ik uiterst dankbaar voor. Tot slot wil ik Gerard Mijnarends en Wim Mooijekind bedanken voor de begeleiding en feedback vanuit school.

Inhoudsopgave

1. Inleiding

Dit is het afstudeerverslag van Mark Smid over de afstudeerstage bij DWG. Het doel van dit verslag is om duidelijk te maken hoe ik dit stagetraject heb doorlopen, wat voor werkzaamheden ik heb uitgevoerd, welke leerdoelen er zijn behaald en hoe ik de afstudeerstage heb ervaren. Ook zal ik met onderbouwingen aantonen dat ik de verrichte taken op HBO niveau uit heb gevoerd. De afstudeerstage heeft plaatsgevonden in het tweede semester van het laatste schooljaar (08-02-2021 tot 04-06-2021) en bestond uit 17 weken fulltime (40 uur per week) werken aan een afstudeeropdracht.

De opdracht bedroeg het onderzoeken en integreren van een draadloos communicatieprotocol in het door DWG ontwikkelde ITS systeem. Een onderzoek uitvoeren, software ontwerpen, programmeren en testen zijn een aantal van de concrete werkzaamheden die naar voren zijn gekomen tijdens de stage.

1.1 Leeswijzer

Hoofdstuk 2 beschrijft het bedrijf waarin ik mijn stage heb uitgevoerd en mijn plaats daarin. In hoofdstuk 3 worden het probleemdomein, de probleemstelling en de doelstelling omschreven. Ook wordt de ITS applicatie op een dieper niveau uitgebeeld om meer diepgang te geven aan het domein waarin de stage uitgevoerd is. Het plan van aanpak en bijbehorende informatie worden beschreven in hoofdstuk 4. In hoofdstuk 5 wordt er een begin gemaakt aan de beschrijving van de werkzaamheden door over het onderzoek te vertellen. De volgende fase, het proof of concept, wordt vervolgens beschreven in hoofdstuk 6. Hoofdstuk 7 geeft een inzicht in de integratie van het protocol in ITS. In hoofdstuk 8 wordt er verteld over de uitgevoerde tests en in hoofdstuk 9 wordt er gereflecteerd op de uitgevoerde stage door middel van een evaluatie van de aanpak en de opgeleverde producten. Tot slot worden in hoofdstuk 10 de gekozen beroepstaken toegelicht aan de hand van uitgevoerde werkzaamheden.

2. DWG

Om goed te kunnen begrijpen wat het doel van deze afstudeerstage is en in welk domein de opdracht plaatsvindt, is het belangrijk om een duidelijk beeld te schetsen van het bedrijf, de sector waarin dit bedrijf actief is en de omgeving waar de afstudeeropdracht zich in bevindt.

2.1 DWG

“Ahead of tomorrow’s challenges” is het motto wat groot op de voorpagina van DWG’s website staat. Het bedrijf noemt zichzelf een engineering specialist en houdt zich vanuit twee vestigingen (Schiedam en Amsterdam) vooral bezig met technologie, innovatie en automatisering. Dit doen zij binnen verschillende markten (productie, energie, terminals, industrie, infrastructuur en maritiem). Het doel is om de wereld een stukje efficiënter, veiliger en duurzamer te maken.

DWG is een jong maar snelgroeiend bedrijf. Ze zijn ontstaan in 2014 en hebben inmiddels meer dan 120 gemotiveerde werknemers in dienst. Het begon als een software engineering bedrijf maar breidde onder de naam DWG Automation al snel uit met een hardware-, engineering- en IT-afdeling. Deze uitbreiding zorgde ervoor dat er complete automatiseringsoplossingen geboden konden worden. Later werd ook nog de discipline proces & safety engineering toegevoegd.

Deze complete skill set is wat DWG bijzonder en uniek in de engineeringwereld maakt. Het is ook de reden waarom ze werken met grote partners als BAM, E.ON, Shell, BP en Siemens om er maar een paar te noemen.

Het deel van het bedrijf waar ik het meest te maken mee krijg is de in Schiedam gevestigde IT engineering afdeling. Het koppelen van softwaresystemen, productieorders automatisch laten verwerken of planning en boekhouding samen voegen zijn opdrachten die op deze afdeling uitgevoerd worden. Ook werken ze aan dashboards die big data uit processen overzichtelijk weergeeft. .NET C#, ASP.NET Core, MVC, SQL Server, Vue etc. zijn een aantal van de systemen en technieken waar gebruik van wordt gemaakt.

2.2 Mijn plaats binnen het bedrijf

Door de coronacrisis heb ik gedurende de hele stage vanuit huis gewerkt net als mijn collega’s. Hiervoor heb ik van DWG een laptop gekregen. Ik hoefde niet in te loggen op een remote-werkomgeving of iets in die richting. Ik kon gewoon lokaal in Visual Studio aan de slag met ontwikkelen. Wel moest ik een aantal keer gebruik maken van een VPN verbinding naar het kantoor om dingen te kunnen testen in de demo-omgeving van ITS. Alleen voor het ophalen van hardware en bij mijn eerste kennismaking met Mark Verbaan ben ik op het kantoor in Schiedam geweest. Ik maakte deel uit van het IT team. Dit is een team van ongeveer twaalf man dat bestaat uit ontwikkelaars en projectleiders. Daarbinnen vormde ik een eenmansteam voor de ontwikkelingen binnen ITS.

3. De opdracht

3.1 Probleemdomein en probleemstelling

Binnen de industriële sector wordt er veel gewerkt met sensoren. Dit is zodat fysieke processen via data gemonitord kunnen worden in software. Een voorbeeld hiervan is een sensor die aangeeft wanneer het vloeistofniveau in een silo een bepaald punt heeft bereikt. Dit is ook het enige wat die sensor aan kan geven. Hoeveel vloeistof er nu precies in de silo zit of wat de temperatuur hiervan is blijft dus onbekend. Dit is wel waardevolle informatie, maar door de kosten die bij het installeren van meer sensoren komen kijken is dit niet altijd mogelijk. Hetgeen wat die installatie zo duur maakt is de connectie naar de softwaresystemen. Dit gaat nu namelijk nog grotendeels via fysieke kabels.

Dat is waar DWG om de hoek komt kijken. Zij werken namelijk hard aan het ontwikkelen van het Industrial Things System (ITS). Dit is een systeem dat on-premise oplossingen biedt voor IoT. Het vormt de verbinding tussen het winnen van data in de industrie en het maken van juiste analyses om productieprocessen te kunnen verbeteren. Om dit mogelijk te maken hebben ze een systeem ontwikkeld dat draadloze sensoren kan beheren en data kan doorkoppelen naar bestaande datasystemen. In de huidige situatie is het zo dat ITS alleen het LoRa protocol ondersteunt voor het verzenden van data over grote afstanden. Die sensoren werken op batterijen en werken zo energiezuinig als mogelijk, zodat de batterijen jaren meegaan. Dit betekent ook dat de interval van zenden laag is (1x per uur) en dat er weinig data verstuurd kan worden. Dit is een probleem omdat ITS hierdoor voor een groot aantal toepassingen niet ingezet kan worden. Als een klant bijvoorbeeld trillingen wil meten van een ventilator om te bepalen wanneer de de lager vervangen moet worden, wil hij of zij misschien wel een aantal keer per seconden data ontvangen van hoe erg de ventilator trilt. Dit is niet mogelijk in de huidige situatie. Hier is echter wel vraag naar.

De **probleemstelling** is als volgt: “*Het gebruik van sensoren bij de klanten van DWG die werken met ITS is niet geoptimaliseerd door het ontbreken van een draadloos communicatieprotocol voor veel metingen op de korte afstand.*”

3.2 Doelstelling

Hoe meer sensoren, hoe meer informatie. Met meer informatie zijn processen beter te optimaliseren. DWG wil het daarom haalbaar maken om meer draadloze sensoren te plaatsen bij bedrijven in de industriële sector. Op deze manier kunnen ze meer waarde leveren aan hun klanten met de ITS software. Om dit te bereiken wil DWG zich gaan richten op het tegenovergestelde van LoRa: data ontsluiten op een hogere interval die niet (per sé) over een grote afstand verstuurd hoeft te worden. Het doel van de stageopdracht is dan ook als volgt:

“*Een uitbreiding op ITS waarbij we (DWG) over een (korte) afstand veel meetwaardes kunnen ontvangen en opslaan door middel van een draadloos communicatieprotocol.*”

Specifiek - Ik ga de software van ITS uitbreiden zodat het kan werken met een ander draadloos communicatieprotocol dan LoRa.

Meetbaar - Het doel is behaald wanneer data van draadloze sensoren ontvangen en opgeslagen kan worden op een interval van minimaal één keer per seconde. Dit moet kunnen over een afstand van minimaal vijf meter tussen de sensoren en de gateways. Korter dan vijf meter is niet wenselijk in verband met het beperken van het aantal mogelijke toepassingen.

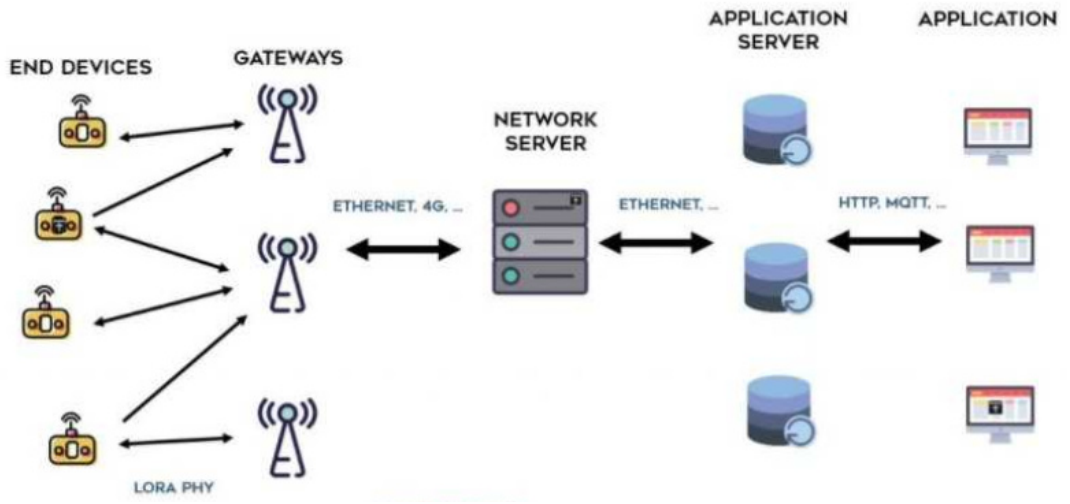
Acceptabel - Het is een waardevolle toevoeging aan DWG en ITS die ik met hulp en begeleiding vanuit DWG in mijn eentje kan ontwikkelen.

Realistisch - Er zijn genoeg protocollen beschikbaar.

Tijdsgebonden - Het doel dient behaald te worden binnen de stageperiode (8 feb - 4 juni)

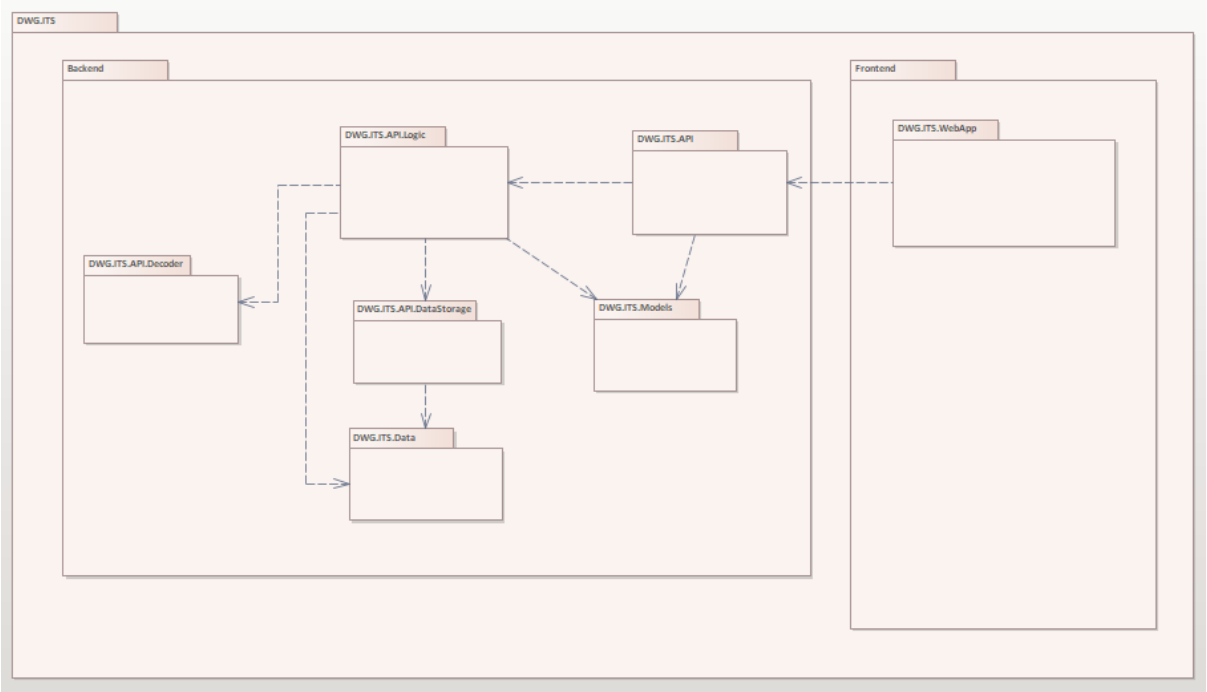
3.3 ITS

ITS is een webapplicatie die ontwikkeld is met een C#, ASP.NET Core back-end en een Vue, typescript front-end. Het wordt ingezet bij klanten om data van draadloze sensoren te ontvangen en vervolgens op te slaan. De huidige versie van ITS ondersteunt alleen het LoRa protocol. Voor de verbinding naar de sensoren wordt daarom gebruik gemaakt van LoRa Gateways. Deze verbinding is echter niet direct. Alle LoRa berichten gaan vanaf de gateway naar ChirpStack. Dit is een open source LoRa server[1]. ChirpStack stuurt deze berichten vervolgens via HTTP requests naar een API endpoint in ITS. Deze tussenstap is nodig, omdat hier een aantal handelingen binnen het LoRa protocol worden uitgevoerd. De berichten worden hier decrypt en het managen van devices en gateways gebeurt hier. Onderstaand diagram geeft een beeld van deze LoRa architectuur: De end services zijn de LoRa sensoren, de gateways zijn de LoRa gateways, de network server is ChirpStack en de application server en application zijn ITS.



Figuur 1: LoRa architectuur
Bron: https://forum.chirpstack.io/t/capturing-subscribing-data-to-application-server/1100

Als we dieper inzoomen op ITS zien we dat het opgedeeld is in verschillende packages. Hierdoor wordt er gezorgd voor een separation of concerns (single responsibility principle). De API bevat alle controllers, waaronder dus ook degene waar ChirpStack de berichten naar toe stuurt. De Logic package handelt, zoals de naam al aangeeft, alle logica af. Daarom staat deze ook in verbinding met alle andere packages. In de decoder package zitten alle decoders voor de sensoren. Deze zijn nodig om berichten van sensoren uit te pakken. De berichten komen namelijk binnen als een hexadecimale string met allerlei informatie achter elkaar. De decoders hebben als functie om deze strings om te zetten in leesbare data. In de Data package vinden we de models, migrations en seeding voor de database. Er wordt gebruik gemaakt van de code-first database aanpak. De Data storage package handelt de interacties met de database af. Deze interacties zijn bijvoorbeeld het ophalen van alle gateways of toevoegen van een device. De laatste package is de front-end. Dit alles wordt afgebeeld in onderstaande package diagram.



Figuur 2: Package diagram van ITS

Voor deze uitgangssituatie binnen ITS heb ik ook een klassendiagram gemaakt. Dit klassendiagram is beperkt tot de delen van ITS waarin ik mijn stage mee in aanraking ga komen. Dit klassendiagram is te vinden in (bijlage **A**).

4. Plan van Aanpak

4.1 Fases

Voorafgaand aan de afstudeerstage heb ik samen met mijn begeleider Niels een planning opgesteld. Dit is een globale planning waarmee de afstudeerstage opgesplitst wordt in drie fases. De eerste fase gaat bestaan uit het uitvoeren van een onderzoek. Het doel van dit onderzoek is om erachter te komen welk protocol het best geschikt is voor DWG en de mogelijke toepassingen bij hun klanten.

Zigbee, Bluetooth Low Energy en nog meer vergelijkbare protocollen zullen hierbij bekeken worden. Als alle informatie verzameld is zal ik hier een conclusie uit halen. Deze informatie en conclusie met onderbouwing zal verwerkt worden in een onderzoeksverslag. Dit onderzoeksverslag en mijn conclusie zal ik vervolgens presenteren aan mijn collega's. Samen zullen we deze resultaten dan beoordelen.

Voor deze fase wordt twee weken uitgetrokken. Dit verschilt met de ene week die in mijn afstudeerplan stond aangegeven. De reden hiervoor is het aantal draadloze communicatie protocollen dat onderzocht moet worden. Een week zou te kort zijn om deze diep genoeg te kunnen onderzoeken. De informatie die ik hiervoor nodig heb zal van het internet verkregen moeten worden, omdat ik geen hardware tot mijn beschikking heb om bijvoorbeeld datasnelheid of batterijduur te testen. Uit dit onderzoek zal één protocol naar voren komen als meest geschikte. Met dit protocol zal vervolgens een proof of concept gemaakt worden. Aan het eind van deze fase zal ik dus een onderzoeksrapport opleveren en presenteren.

Het maken van een proof of concept zal de tweede fase gaan vormen. Als mijn collega's en ik op een lijn zitten over welk draadloze communicatieprotocol we willen gaan integreren in ITS, kan ik hier aan beginnen. Het doel van deze fase is het ontwikkelen van een proof of concept waarmee ik aan kan tonen dat het gekozen protocol ook daadwerkelijk geschikt is voor de toepassingen waar DWG het voor wil gebruiken. Een voorbeeld van zo'n toepassing is het meten van trillingen op leidingen waar kleppen open en dicht in gaan. Om dit aan te kunnen tonen zal er hardware besteld moeten worden waarmee ik een kleine werkende applicatie kan bouwen. Hoe dit eruit gaat zien kan pas ingevuld worden wanneer bekend is welk protocol uit het onderzoek als meest geschikt naar voren komt.

Ook deze fase zal langer duren dan aangegeven in mijn afstudeerplan. Ik had hier initieel een á twee weken voor ingepland, maar na het heroverwegen met mijn begeleider leek vijf weken ons een betere inschatting. Deze verandering in planning hebben we gemaakt omdat we ruimte willen overhouden voor eventuele tegenslagen en leermomenten. Aan het eind van deze fase zal er een Proof of Concept applicatie opgeleverd en gepresenteerd worden.

Als er een werkend proof of concept is en deze goedgekeurd is door mijn collega's zal de laatste en grootste fase ingaan. Hier zal het gekozen protocol daadwerkelijk geïntegreerd worden in de ITS software van DWG. Deze fase zal vooral bestaan uit het opstellen van requirements, ontwerpen van software, programmeren in ASP .NET Core, C# en Vue typescript en testen. Het doel van deze fase is het volledig in ITS integreren van het gekozen communicatieprotocol. Tijdens deze fase zal ik aan het eind van iedere SCRUM

sprint een aantal ontwikkelde functionaliteiten opleveren. Deze planning is verwerkt in onderstaande tabel:

Fase	Duurtijd	Op te leveren producten
Onderzoek	2 weken	- Onderzoeksrapport
Proof of Concept	5 weken	- PoC applicatie
ITS integratie	10 weken	- Ontwikkelde functionaliteit per SCRUM sprint - Testplan - Testrapport

Figuur 3: Plan van aanpak planning

4.2 Ontwikkelmethoden

Een proof of concept is een realisatie van een bepaalde methode of idee om de haalbaarheid ervan aan te tonen. Het is doorgaans een kleine applicatie en kan al dan niet volledig zijn[2]. Vandaar dat er geen formele ontwikkelmethode zoals bijvoorbeeld Scrum gekozen is om dit uit te werken. Dit is ook de gebruikelijke gang van zaken binnen DWG.

Het gewenste resultaat is een programma waarmee aangetoont kan worden dat integratie van het gekozen protocol in ITS haalbaar is. Samen met het onderzoek vormt het proof of concept een haalbaarheidsstudie. Bij de integratie in ITS zal volgens de Scrum methodiek gewerkt gaan worden. Hier is voor gekozen omdat DWG al hun software ontwikkelingen doen met deze methodiek. Ook ben ik vanuit school bekend met deze methodiek.

Bijna ieder bedrijf implementeert de Scrum methodiek op een aangepaste manier. DWG maakt gebruik van Azure DevOps. Hier staan het scrumboard, de sprints, de backlog, de burn-down, etc. Dit maakt het erg makkelijk om het scrumboard constant up-to-date te houden. Bij iedere taak die je af hebt, pas je de uren aan en zet je hem in de juiste kolom (To Do, Working, Unit Test, Test, Code Review of Done). Dit aantal kolommen is meer dan het traditionele scrumboard dat maar drie of vier kolommen heeft. DWG heeft hiervoor gekozen om meer informatie te hebben over in welke fase een taak zich bevindt. DevOps update dan ook gelijk de burn down grafiek.

Iedere dinsdag en donderdag om negen uur wordt er een stand-up meeting gehouden waarbij iedereen vertelt wat hij de voorgaande dag heeft gedaan, wat hij deze dag gaat doen en of hij daarbij hulp nodig heeft. Zo weet iedereen van elkaar waar ze mee bezig zijn en kunnen hulp momenten makkelijk ingepland worden. Deze meeting is niet projectspecifiek en is dus met iedereen van de afdeling en is geïntroduceerd door het thuiswerken.

Ook voor het ITS team worden dagelijk stand-up meetings gehouden. Omdat ik tijdens mijn stage in mijn eentje aan ITS zal werken zal ik deze meetings samen met mijn begeleider

Niels houden. Deze meetings zullen iedere dag om twaalf uur plaatsvinden. De sprints zullen twee weken duren. Deze duurtijd is gekozen omdat ik dan precies vijf sprints heb tot het einde van de stageperiode. Op iedere laatste dag van een sprint wordt er een meeting gehouden voor de retrospective, review en de planning voor de volgende sprint met de product owner.

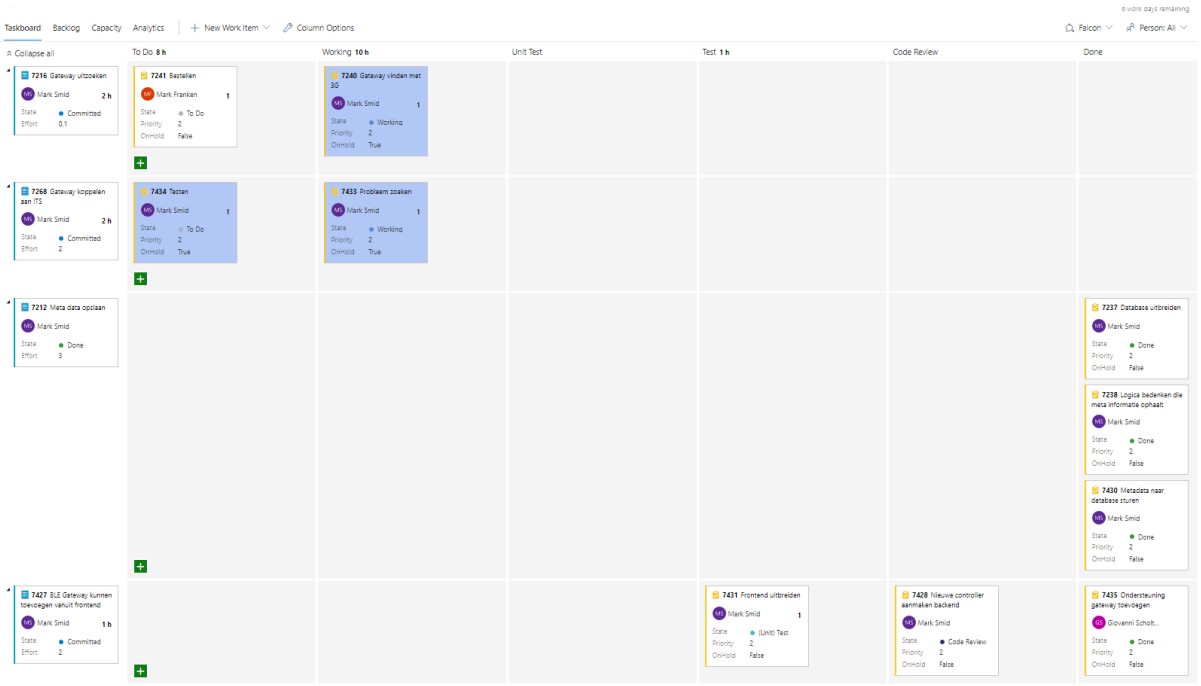
DWG probeert zich zo goed als mogelijk aan de regels van Scrum[3] te houden waar dit mogelijk is. Een van de regels waar dit niet mogelijk is, is de duurtijd van de stand-up meetings. Door het aantal mensen binnen het IT team duren de dinsdag en donderdag stand-up meetings vaak langer dan de door SCRUM als maximaal beschreven vijftien minuten.

4.3 Azure DevOps

Azure DevOps is een clouddienst van Microsoft voor versiebeheer, geautomatiseerde builds, testen en release management[4]. Het beschikt over een groot aantal functionaliteiten dat ervoor zorgt dat teams hele projecten kunnen managen en hun workflow en productiviteit kunnen verbeteren. Binnen DevOps wordt er door DWG (en dus ook door mij) het meest gebruik gemaakt van de Repos, Boards en Pipelines.

Repos is kort voor repositories. Dit deel van DevOps maakt het mogelijk om de code van projecten online te hosten. De git repository is de basis voor ieder software ontwikkelingsproject. Pull requests zijn hierin een uiterst waardevolle functie die veel gebruikt wordt door ontwikkelaars binnen DWG. Dit maakt het mogelijk om code reviews te doen voordat het in de master branch wordt gemerged. DevOps geeft hier een goed overzicht van de wijzigingen met pull request, de work items die erbij horen en wie de reviewer is.

De boards stellen je in staat om om je scrumboards in DevOps te managen. Hierop staan zogeheten work items. Dit kunnen bijvoorbeeld user stories, bugs, product backlog items of epics zijn. Bij deze boards horen ook de sprint boards. Deze boards geven op taakniveau weer welke teamleden bezig zijn met welke items. Deze taken staan verdeeld over een aantal kolommen. Deze kolommen (To Do, Working, Unit Test, Test, Code Review of Done) geven aan in welke fase een taak zit. De ontwikkelaars van DWG werken iedere dag aan de hand van deze boards. Dit is ook hoe ik te werk zal gaan.



Figuur 4: Screenshot van het sprint board in DevOps

De pipelines binnen DevOps komen aan bod wanneer een ontwikkelaar klaar is met zijn of haar code. Hier kan je de build en eventueel zelfs de release configureren in Yaml. DWG heeft het zo geconfigureerd dat er bij een pull request een automatische SonarQube controle over de gepushte code gaat met de volgende code:

```
- task: SonarQubePrepare@4
  displayName: "API - SonarQube prepare"
  inputs:
    SonarQube: 'BuildServer-SonarQube-ITS'
    scannerMode: 'MSBuild'
    projectKey: 'ITS'

- task: SonarQubeAnalyze@4
  displayName: "API - SonarQube analysis"

- task: SonarQubePublish@4
  displayName: "API - Publish Quality Gate Result"
  inputs:
    pollingTimeoutSec: '300'
```

Figuur 5: Code snippet van SonarQube configuratie

SonarQube fungeert als een geautomatiseerde code review voordat de code review door een andere ontwikkelaar wordt uitgevoerd. Hierdoor kan de reviewende ontwikkelaar zich focussen op de functionaliteit van de code. De SonarQube controle die DWG heeft ingesteld kijkt naar vier verschillende categorieën op het gebied van code kwaliteit[5]:

- Bugs
- Vulnerabilities
- Code Smells

- Security hotspots

Binnen deze vier categorieën zitten zogenaamde “rules”. Een voorbeeld van zo’n rule in de bug categorie is: “Empty collections should not be accessed or iterated”. Een bug is een rule die een error zal veroorzaken en de code zal breken. Deze punten moeten dus direct gefixt worden. Vulnerability slaan meer op de security van de code. Een voorbeeld van een rule uit deze categorie is: “Database queries should not be vulnerable to injection attacks”. Code Smells zijn iets minder gevaarlijk omdat ze op zichzelf geen kwetsbaarheden of errors veroorzaken. Wel kunnen code smells duiden op onderliggende problemen in de code. Ook is het vooral een probleem voor de onderhoudbaarheid van de code. Het maakt de code verwarrend en moeilijk om te begrijpen. Een voorbeeld van een rule uit deze categorie is: “Methods should not have identical implementations”. De laatste categorie waar SonarQube op checkt is Security Hotspots. Security Hotspots zijn security-gevoelige stukken code die de ontwikkelaar moet beoordelen. De ontwikkelaar bepaald dan vervolgens zelf of dit stukje code voor een kwetsbaarheid zorgt of niet. Het dient als een extra vangnet in situaties waarbij de vulnerability analysis een mogelijke kwetsbaarheid mist. De vulnerability rule detecteert namelijk alleen kwetsbaarheden bij een hoge zekerheid. De hotspot rule wijst code aan waar mogelijk een kwetsbaarheid zou kunnen liggen, ook al is daar geen vulnerability gedetecteerd. Een voorbeeld hiervan is het switchen tussen programmeertalen of het gebruik van third party libraries.

5. Onderzoek naar draadloze communicatieprotocollen

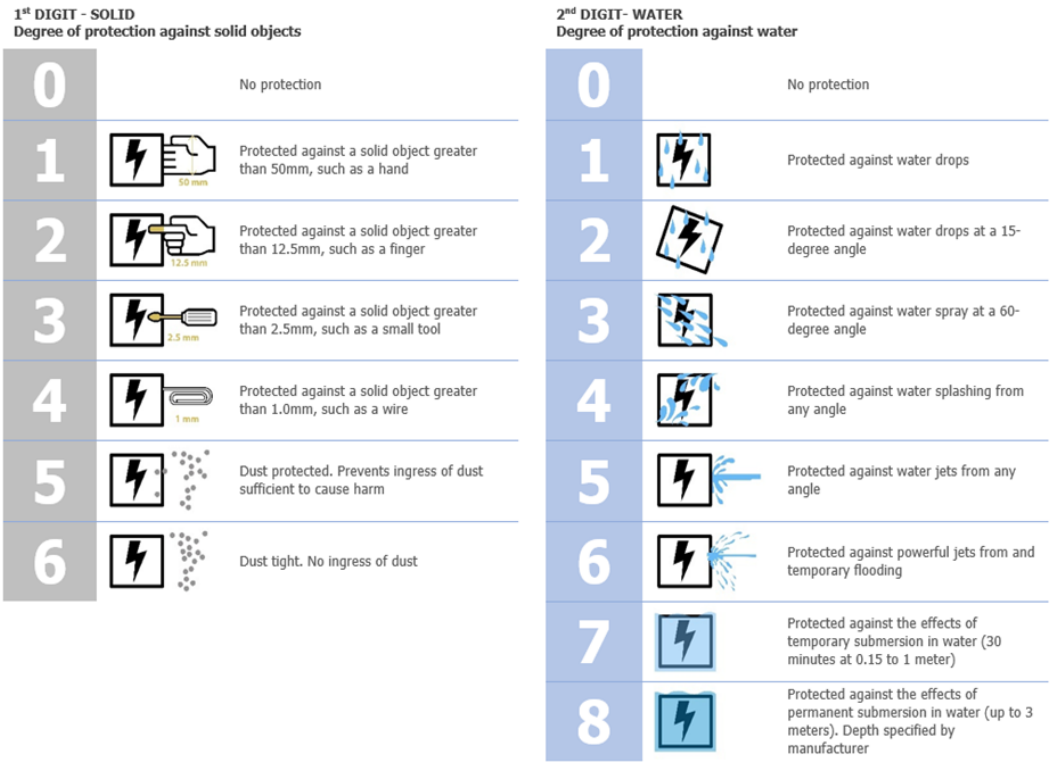
Op de eerste dag van de stage had ik meteen een gesprek met Mark Franken (product owner ITS) waarin hij duidelijk maakte wat zijn verwachtingen waren van het onderzoek. Hij wilde een onderzoek die de onderzoeksvraag *"Welk draadloos communicatieprotocol is het best geschikt om de tekortkomingen van LoRa in ITS op te vullen?"* zou beantwoorden. Hieruit moest vervolgens een onderbouwde conclusie gevormd worden. Dit alles moest op papier gezet worden in een beknopt onderzoeksverslag met een samenvatting, inleiding, een korte beschrijving per protocol, een tabel waarin de protocollen vergeleken worden, een conclusie en een lijst met bronnen. Ook gaf hij me nog een aantal tips over hoe het eraan toegaat in de industriële sector. Hij zei dat veel bedrijven in deze sector vaak huiverig zijn voor digitale systemen en ontwikkelingen. Ze zijn er vaak niet van overtuigd dat dit soort systemen robuust genoeg zijn. Om een goed onderzoek uit te kunnen voeren heb ik van te voren met mijn collega's een aantal requirements opgesteld waaraan een geschikt protocol zou moeten voldoen. Deze requirements waren als volgt, geprioriteerd volgens de MoSCoW methode:

Requirement	Prioriteit
De datasnelheid (bandbreedte) hoog genoeg is om minimaal één bericht per seconde te kunnen verzenden	Must-Have
Er geen limiet zit op het aantal berichten dat je mag verzenden	Must-Have
Er is hardware (sensoren en gateways) beschikbaar op industrieel niveau (IP66 of hoger)	Must-Have
Het bereik tussen sensoren en gateways minimaal 5 meter is	Must-Have
De batterijduur van de gebruikte apparaten zodanig is dat ze langer dan een jaar kunnen functioneren op enkele batterij (zend interval minimaal één keer per seconde)	Should-Have

Figuur 6: Requirements draadloze protocollen

We hebben besloten van te voren geen requirements vast te leggen op het gebied van security, omdat dit te afhankelijk is van de specifieke toepassing. Na dit gesprek ben ik meteen begonnen met onderzoek doen naar de verschillende draadloze communicatieprotocollen die interessant zouden kunnen zijn voor integratie in ITS. Mijn startpunt was een whitepaper van IoT Academy[6] waarin de meest gebruikte protocollen werden beschreven. Ik heb nog verder gezocht naar meer protocollen, maar heb besloten deze niet mee te nemen in het onderzoek omdat deze nog teveel in de kinderschoenen stonden. Vervolgens ben ik me per protocol verder in gaan lezen. Het onderzoek bestond grotendeels uit deskresearch / literatuuronderzoek waarbij ik informatie zocht op het internet.

Ook heb ik gebruik gemaakt van de kennis van mijn collega's. Door te spreken met mijn collega's kreeg ik een beter beeld van de uitgangssituatie (zoals beschreven in hoofdstuk 3) en konden zij mij handige tips geven over waar ik naar kon zoeken. Een concreet voorbeeld hiervan is dat zij mij konden vertellen naar wat voor specificaties en certificeringen ik moest kijken bij het zoeken van hardware. Deze hardware wordt namelijk vaak in de buitenlucht of andere omgevingen gebruikt waar de meeste elektronica niet goed tegen kan. Denk hierbij bijvoorbeeld aan een hele stoffige omgeving of een omgeving die dagelijks met een hogedrukspuit schoongemaakt wordt. Om te kunnen beoordelen hoe goed een apparaat tegen dit soort factoren kan is er een IP-rating. Dit is een rating die uit twee getallen bestaat die aangeven hoe beschermt het apparaat is tegen soliden en water.



Figuur 7: De IP standaard (International Protection Rating).

Bovenstaande diagram laat de verschillende IP niveaus zien. Op advies van mijn collega's heb ik dit diagram gebruikt bij het zoeken naar hardware voor de verschillende protocollen. Een IP-rating van IP66 was hierbij het minimum om een apparaat als geschikt te beschouwen.

5.1 Samenvatting

Ik heb al deze informatie vervolgens verwerkt in een onderzoeksrapport (bijlage **B**). Hierin heb ik voor ieder protocol kort uitgelegd hoe het werkt, waar het doorgaans voor gebruikt wordt, of er genoeg (industrieel geschikte) hardware beschikbaar voor is en wat de voor- en nadelen zijn. Het werd al vrij snel duidelijk dat een aantal van de onderzochte protocollen ongeschikt waren. Zigbee en Z-Wave leken bijvoorbeeld aanvankelijk een interessante optie door de datasnelheid en het ontbreken van een limiet op het aantal te versturen berichten.

Na op zoek te gaan naar industriële hardware hiervoor bleek echter het tegendeel. Bijna alle beschikbare hardware voor Zigbee en Z-Wave zijn gericht op home automation. Dit zijn lichte, plastic apparaten die het nog geen week zouden volhouden in een industriële omgeving.

Ook RFID en daarmee NFC vielen al snel af. Deze protocollen kunnen alleen op een peer-to-peer basis gebruikt worden. Hierdoor is het aantal toepassingen waarbij RFID gebruikt kan worden erg beperkt. Hierbij komt ook nog eens dat het bereik van RFID over het algemeen erg kort (10 cm - 12 m), wat het aantal mogelijke toepassingen nog kleiner maakt. NFC is met een bereik van 4 cm nog specifieker qua toepassingen (OV chipkaart in de tram of bus bijvoorbeeld).

Wi-Fi HaLow heeft op papier hele interessante cijfers. Een datasnelheid van 150 kbit/s - 80 Mbit/s bij een bereik van ongeveer een kilometer is erg indrukwekkend voor een protocol dat weinig energie moet verbruiken. Het probleem bij Wi-Fi HaLow zit het hem in de frequentie waar het mee werkt. Doordat er voor elke regio een andere frequentie wordt gebruikt is er hier (in Nederland) bijna geen hardware beschikbaar voor industriële doeleinden. Het is een technologie die vooral in de Verenigde Staten gebruikt wordt.

Het rapport wordt afgesloten met een conclusie waarin ik uitleg waarom Bluetooth Low Energy het best geschikte protocol is om de tekortkomingen van de uitgangssituatie te verhelpen. Ik ben tot deze conclusie gekomen doordat dit protocol als enige aan alle requirements voldeed. Het heeft een hoge datasnelheid ten opzichte van veel andere draadloze communicatieprotocollen (2 Mbit/s) en er zit geen limiet op het aantal berichten dat verstuurd kan worden. Ook is het te gebruiken voor een groot aantal toepassingen door de flexibiliteit in mogelijke netwerk topologieën (broadcast, one-to-one, ster, mesh). Beschikbaarheid van hardware is ook erg belangrijk en dat zit bij Bluetooth Low Energy goed. Er zijn robuuste sensoren en gateways te vinden die specifiek ontwikkeld zijn voor gebruik in industriële omgevingen (IP66 en hoger). Onderstaande tabel geeft een overzicht over welke protocollen voldoen aan welke van de gestelde requirements. De specificaties per protocol (bereik, datasnelheid, etc.) zijn te vinden in het onderzoeksrapport (bijlage **B**).

	Limiet aantal berichten	Datasnelheid	Bereik	Batterijduur	Beschikbare Hardware
Bluetooth LE	X	X	X	X	X
Wi-Fi HaLow	X	X	X	X	
Zigbee	X	X	X	X	
Z-Wave	X		X	X	
RFID	X			X	
NFC	X			X	
SigFox	X		X	X	

Figuur 8: Tabel uit het onderzoeksrapport

Toen werd het tijd om mijn bevindingen te presenteren aan mijn collega's. Met een Powerpoint presentatie ter ondersteuning heb ik ieder protocol kort beschreven (bijlage **C**). Vervolgens heb ik uitgelegd waarom ik Bluetooth Low Energy het best geschikte protocol vond. Na het stellen van een aantal kritische vragen waren mijn collega's het eens met mijn conclusie en advies en is er besloten hardware te bestellen voor het proof of concept. Deze hardware betrof een temperatuursensor (INGICS iBS03T) en een gateway (INGICS iGS02E). Wel was er besloten om ook nog wat dieper in te gaan op de security van Bluetooth Low Energy. Dit kwam omdat in mijn rapport de security van dit protocol werd aangeduid als matig bij een oudere versie, maar goed bij een latere versie.

5.2 Bluetooth Low Energy security

Niels IJpelaar en ik zijn de volgende dag in gesprek gegaan met een security expert van DWG, William Finch. William had van te voren kort onderzoek gedaan naar de security van Bluetooth Low Energy en kon mij en Niels daarom goed helpen. Hij bevestigde een aantal kwetsbaarheden die ik ook in mijn onderzoek tegen was gekomen. Deze kwetsbaarheden zitten in het pairing proces tussen apparaten. Hierbij moeten er keys uitgewisseld worden om de pairing tot stand te brengen. Het hacken van Bluetooth Low Energy apparaten die al gepaired zijn is (bijna) onmogelijk.

Er zit een groot verschil op het gebied van security tussen Bluetooth Low Energy 4.2 en eerdere versies. Het protocol kan gebruikt worden op verschillende manieren. Beacons zijn apparaten waarmee niet gepaired hoeft te worden. Deze zijn constant berichten aan het broadcasten die opgevangen kunnen worden met een gateway. Dit zou een security risico kunnen vormen als de gebroadcaste data gevoelig is. Er zijn echter ook Bluetooth Low Energy apparaten waarbij pairing benodigd is (zoals je je Bluetooth koptelefoon aan je telefoon paired). Dit soort apparaten worden doorgaans gebruikt voor het opzetten van mesh netwerken. Het bluetooth Low Energy protocol voor versie 4.2 heeft maar één manier van pairen genaamd: "legacy pairing"[7]. Dit is een methode zonder enige vorm van beveiliging en is daardoor gevoelig voor "passive eavesdropping" en man-in-the-middle aanvallen.

Versies 4.2 en nieuwer hebben dit probleem niet. Deze versies van Bluetooth Low Energy hebben secure connections. Dit is een veiligere manier van pairing omdat het gebruik maakt van het Elliptic Curve Diffie Hellman algoritme voor het genereren van keys. Dit is een sterk algoritme dat gebruikt wordt door software als WhatsApp, Facebook Messenger en Skype. Binnen secure connections heb je dan weer de keuze uit vier mogelijke pairing methodes:

- Just works pairing
- Out of band pairing
- Passkey pairing
- Numeric comparison pairing.

Het verschil tussen deze methodes zit hem in de manier waarop de keys gegenereerd worden. Just Works is de standaard pairing methode voor het overgrote deel van Bluetooth Low Energy netwerken. De temporary key word hierbij op 0 gezet waarna de devices de short term key hiervan genereren. Deze manier van verbinden biedt daarom totaal geen beveiliging. Deze manier van verbinding bestond al binnen legacy pairing, maar met de

introductie van secure connections is het mogelijk om deze pairing methode te beschermen tegen eavesdropping aanvallen. Om de verbindingen ook te beveiligen tegen man-in-the-middle aanvallen is dit echter niet genoeg.

De out of band pairing methode werkt middels een ander draadloos protocol. Hierbij worden de keys tussen de apparaten niet via Bluetooth Low Energy uitgewisseld, maar via een ander, intrinsiek veiliger protocol.

De passkey pairing methode maakt gebruik van input van de gebruiker. Een van de apparaten genereert een zes-cijferige code. Deze code vult de gebruiker vervolgens in op het andere apparaat. Over het algemeen gebeurd dit door middel van een touchscreen op het apparaat of via een mobiele applicatie waarmee de sensor geconfigureerd kan worden. Dit maakt een man-in-the-middle aanval natuurlijk compleet onmogelijk.

Als laatst hebben we de numeric comparison pairing methode. Deze methode maakt ook gebruik van input van de gebruiker. Hierbij genereren beide apparaten een zes-cijferige code, die de gebruiker dan handmatig moet vergelijken.

Persoonlijk vind ik de laatste twee methoden het best. Als je ervoor kiest om het Bluetooth Low Energy protocol te gebruiken vind ik het onlogisch om een compleet ander protocol in te zetten voor de pairing van de Bluetooth Low Energy apparaten. Dit voegt een hoop complexiteit toe wat ook weer meer punten van falen kan introduceren. Natuurlijk zijn ook de laatste twee methoden suboptimaal. In een situatie waar de sensoren bijvoorbeeld op een locatie hangen waar ze lastig te bereiken zijn door een gebruiker zal het een hoop moeite kosten om de pairing tot stand te brengen. Gelukkig hoeft dit in principe maar één keer te gebeuren. Wat de best geschikte pairing methode is zal dus verschillen per toepassing.

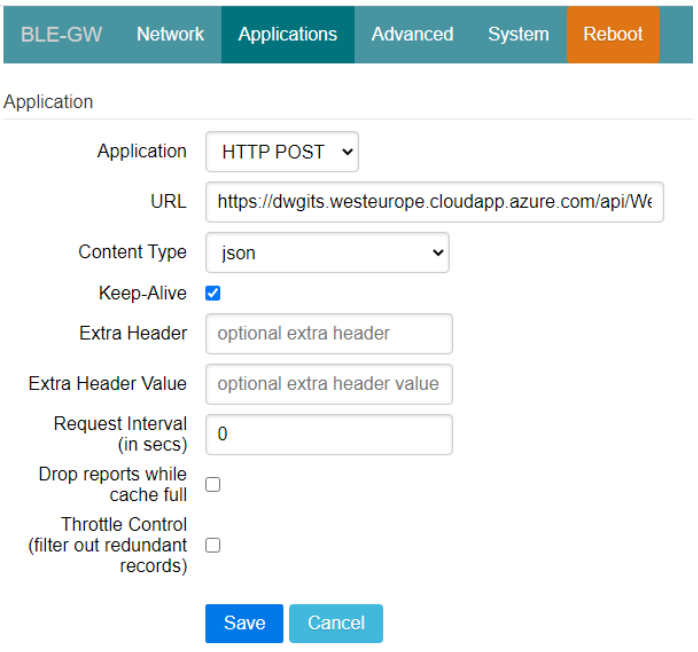
6. Proof of Concept

Nadat de bevindingen die uit het onderzoek waren gekomen goedgekeurd waren, kon ik beginnen aan het ontwikkelen van het Proof of Concept. Het doel van dit Proof of Concept was het aantonen dat een integratie van Bluetooth Low Energy in ITS haalbaar zou zijn. Hiervoor moesten als eerst een stel requirements opgesteld worden. Deze requirements waren als volgt, geprioriteerd volgens de MoSCoW methode:

Requirement	Prioriteit
Het PoC moet aantonen dat het mogelijk is om Bluetooth Low Energy berichten te ontvangen	Must-Have
Het PoC moet Bluetooth Low Energy berichten kunnen uitpakken (decoden)	Must-Have
Het PoC moet in C# geschreven worden zodat het overeenkomt met ITS	Must-Have
Het Poc moet ontvangen data kunnen weergeven	Should-Have
De hardware voor het Proof of Concept moet minder dan 200 euro kosten	Could-Have

Figuur 9: Requirements proof of concept

Bij het bestellen van de hardware voor het proof of concept is er geen rekening gehouden met een minimale IP-rating zodat de kosten onder de 200 euro konden blijven. Om simpel te beginnen hadden we een Bluetooth Low Energy gateway en een Bluetooth Low Energy temperatuursensor besteld. Beiden apparaten waren plug & play waardoor er niet op geprogrammeerd hoefde te worden. Dit is gebruikelijk bij dit soort apparaten. Wel is er voor de gateway een webpagina waarop bijvoorbeeld het protocol tussen de gateway en de API en zend intervallen geconfigureerd kunnen worden. Instellingen zoals de zend interval van de sensor kunnen geconfigureerd worden door middel van een applicatie op je telefoon via Bluetooth.

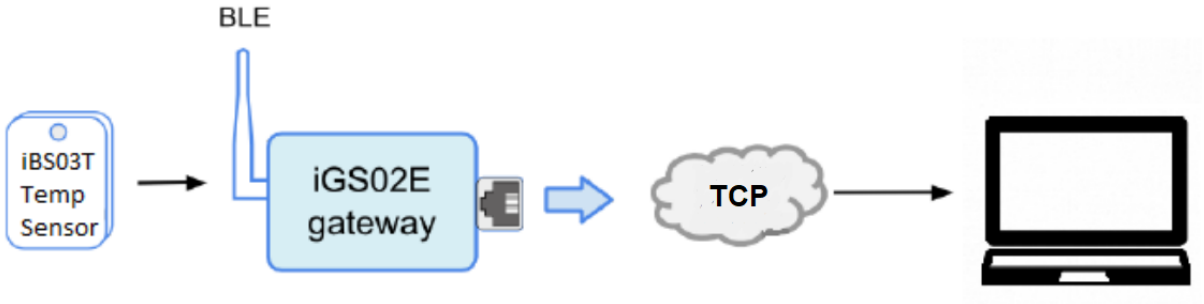


Figuur 10: Screenshot van de configuratiepagina van de voor het PoC bestelde gateway.

De bestelde sensor was een zogenaamde “beacon”. Dit houd in dat de sensor constant zijn data aan het broadcasten is. Deze data kan dan worden opgevangen door gateways die binnen het bereik zijn. Deze keuze is gemaakt om een aantal redenen. De voornaamste hiervan is dat de prioriteit van mijn opdracht lag bij het ontwikkelen van de functionaliteit binnen ITS. Deze integratie heeft vooral te maken met wat er gebeurd na de gateway. Hoe de gateways en sensoren met elkaar communiceren heeft dus geen invloed op de integratie. De keuze is daarom gevallen op de meest simpele manier van verbinden tussen sensoren en gateways. Daarbij komt ook dat het overgrote deel van de beschikbare sensoren van het beacon type zijn. Deze zijn over het algemeen ook goedkoper dan sensoren die gepaired moeten worden aan de gateway.

6.1 TCP proof of concept

Toen deze spullen binnen waren kon er begonnen worden aan het maken van het proof of concept. De bestelde gateway beschikte over meerdere mogelijkheden voor het verzenden van data naar een server. Dit was mogelijk via TCP, HTTP en MQTT. Het idee was om te beginnen met de meest simpele versie van een werkend proof of concept. Dit was een simpele C# console application waarin de data van de sensor in realtime in een console wordt weergegeven. Bij deze toepassing vervult de gateway de rol van de TCP server en de applicatie de rol van TCP client. De applicatie ontvangt de berichten van de gateway via TCP. Het onderstaande diagram beeld deze constructie uit:



Figuur 11: Diagram die de constructie van het PoC weergeeft (laptop representeert de PoC applicatie).

De ruwe data die via de socket binnenkomt wordt vervolgens uitgepakt. Hierna wordt de uitgepakte data weergegeven in de console. De werking van deze applicatie is te zien in een video bijlage (bijlage **D**).

Om te kunnen achterhalen hoe de ruwe data uitgepakt moet worden is er voor iedere sensor een schema dat wordt meegeleverd in de user guide vanuit de leverancier. In dit schema wordt uitgebeeld welke bytes voor wat staan. De ruwe data string wordt omgezet in een byte array. Vervolgens worden de bytes gesplitst en omgezet naar strings of doubles. Deze decoder verschilt per sensor en is dus niet generiek te maken. Het uitpakken van de berichten van de temperatuursensor ziet eruit als volgt:

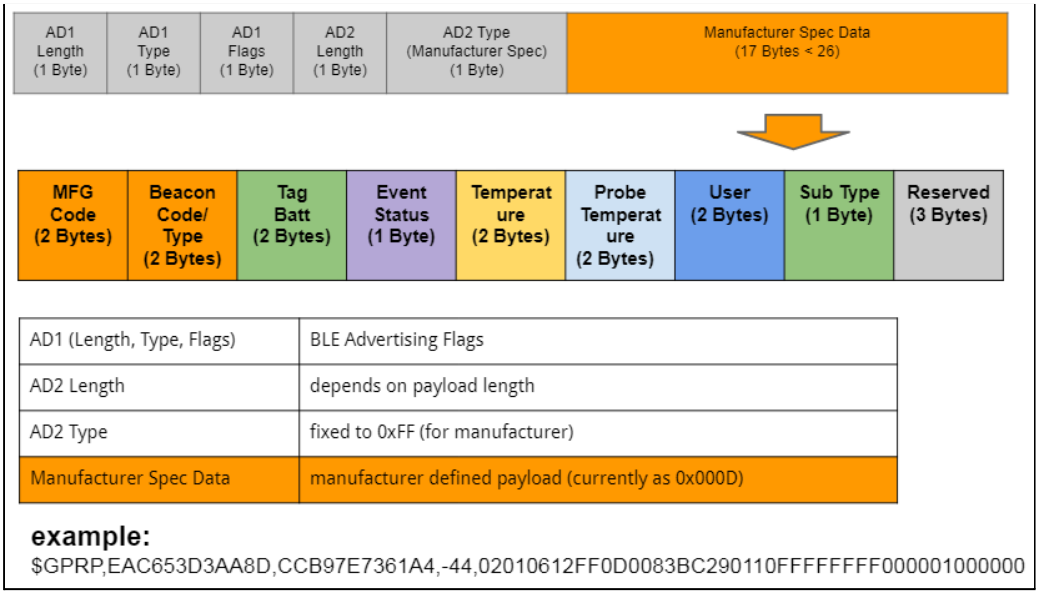
```
1 reference
public String packetDecoder(string rawData)
{
    byte[] rawDataBytes = ConvertHexStringToByteArray(rawData);

    string mfgCode = BitConverter.ToString(rawDataBytes.Skip(5).Take(2).Reverse().ToArray(), 0).Replace("-", string.Empty);
    string beaconCode = BitConverter.ToString(rawDataBytes.Skip(7).Take(2).Reverse().ToArray(), 0).Replace("-", string.Empty);
    double batteryVolt = BitConverter.ToInt16(rawDataBytes.Skip(9).Take(2).ToArray(), 0) / 100.0;
    string eventStatus = BitConverter.ToString(rawDataBytes.Skip(11).Take(1).Reverse().ToArray(), 0);
    double temp = BitConverter.ToInt16(rawDataBytes.Skip(12).Take(2).ToArray(), 0) / 100.0;
    double probeTemp = BitConverter.ToInt16(rawDataBytes.Skip(14).Take(2).ToArray(), 0) / 100.0;
    string user = BitConverter.ToString(rawDataBytes.Skip(16).Take(2).Reverse().ToArray(), 0).Replace("-", string.Empty);
    string subType = BitConverter.ToString(rawDataBytes.Skip(18).Take(1).Reverse().ToArray(), 0);

    return "mfgCode: " + mfgCode +
        " beaconCode: " + beaconCode +
        " batteryVolt: " + batteryVolt +
        " eventStatus: " + eventStatus +
        " temp: " + temp +
        " probeTemp: " + probeTemp +
        " user: " + user +
        " subType: " + subType;
}
```

Figuur 12: Stuk code van de decoder van de temperatuur sensor.

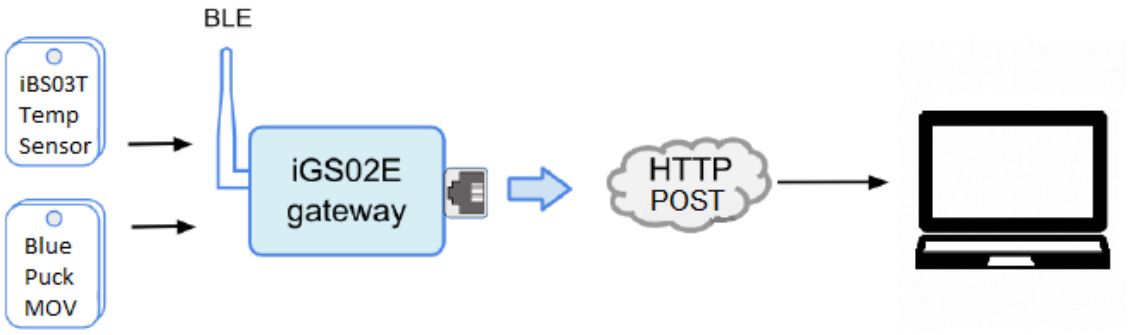
Dit stukje code heb gemaakt aan de hand van het schema uit de user guide van de sensor. Het schema laat zien welke data in welk stuk van de string staat. Ook wordt er onderaan een voorbeeld gegeven van een bericht dat uit de sensor zou kunnen komen. Dit schema ziet eruit als volgt:



Figuur 13: Schema uit het “Sensor Beacon Payload Format” bestand van de iBS03T temperatuursensor.

6.2 HTTP proof of concept

De volgende stap was het ontvangen van berichten van de gateway met HTTP. Dit was een betere representatie van hoe de integratie in ITS eruit zou gaan zien. Ook wilden we hier nog een sensor aan toevoegen. Hiervoor hebben we een bewegingssensor besteld. Nu had ik dus twee sensoren die data stuurden naar de gateway, die op zijn beurt HTTP POST requests naar de door mij gemaakte API stuurt met daarin de berichten van de sensoren. Ook bij deze applicatie wordt de ruwe data uitgepakt. Nu wordt deze data echter weergegeven op een simpele web UI (cshtml pagina) in plaats van de console. Het onderstaande diagram beeld deze constructie uit.



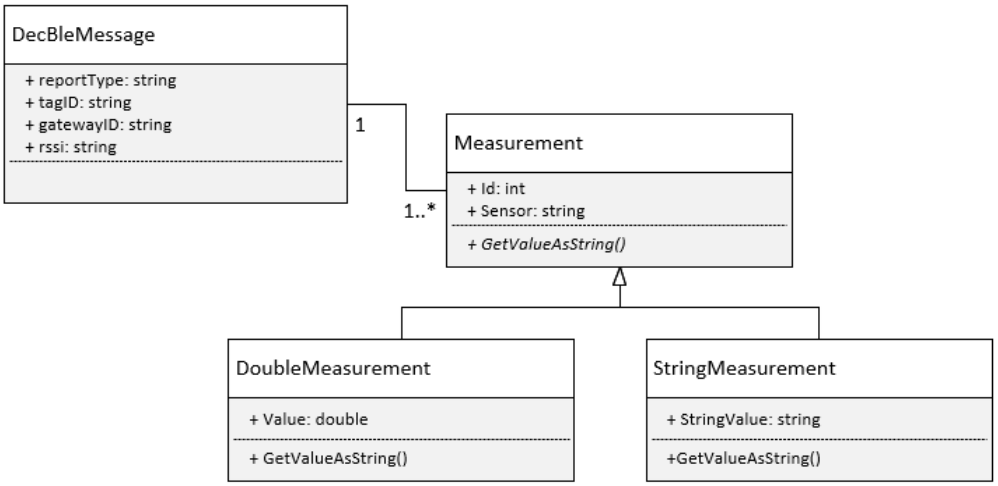
Figuur 14: Diagram die de constructie van het Proof of Concept weergeeft (laptop representeert de PoC applicatie).

De berichten tussen de sensoren en de gateway gaan via Bluetooth Low Energy. De sensoren zijn constant aan het broadcasten op een door mij geconfigureerde zend interval. De gateway (client) stuurt deze berichten vervolgens via HTTPS POST requests door naar de API (server). De API verwerkt deze berichten, pakt de berichten uit en stuurt ze naar de front-end.

Om te kunnen zien hoe de HTTP POST requests van de gateway eruit zagen heb ik gebruik gemaakt van WireShark. Met een filter kon ik makkelijk een pakket van de gateway naar de API selecteren en uitpluizen. Hierdoor kwam ik erachter dat de gateway een lijst met ontvangen BLE berichten in de body van de POST request stopt. Dit was natuurlijk belangrijke informatie bij het programmeren van de controller in de API. Bij deze applicatie was het zoals eerder vermeld ook de bedoeling om de data van meerdere sensoren te kunnen verwerken. Buiten de decoders die specifiek voor ieder Bluetooth Low Energy apparaat gemaakt moeten worden, moest het proof of concept dus generiek werken. Om dit te realiseren zit de applicatie in elkaar als volgt.

Een Bluetooth Low Energy sensor apparaat kan meerdere sensoren hebben. Een bewegingssensor heeft bijvoorbeeld een sensor voor de X-as, de Y-as en de Z-as. Omdat dit per apparaat verschilt is er een constructie gemaakt die gebruik maakt van measurements. Uit een Bluetooth Low Energy bericht van het hiervoor genoemde apparaat komen dus drie metingen. Dit kan of een string zijn of een double. De reden om deze constructie te gebruiken bij het proof of concept, was dat het binnen ITS met LoRa berichten ook op deze manier werd afgehandeld. Wanneer er bericht binnenkomt in de controller van de API, wordt gekeken van welke sensor deze is. De data wordt vervolgens naar de juiste decoder

gestuurd. Deze geeft een lijst van measurements terug die aan een DecBleMessage (decoded Bluetooth Low Energy message) worden gekoppeld.



Figuur 15: UML klassendiagram die de constructie van measurements uitbeeldt

Deze DecBleMessage word vervolgens via een ChatHub naar de front-end gestuurd. Om de data van de sensoren vervolgens ook daadwerkelijk in real-time in de front-end te kunnen laten zien heb ik gebruik gemaakt van SignalR. Dit is een library waarmee je in real-time data naar je front-end kan pushen. Het bevat componenten aan server- en client zijde. De data van de trillingssensor wilde ik laten zien in een Chart.js library grafiek en de data van de temperatuursensor in een tabel. De volgende code pusht de nieuwe data naar de grafiek en tabel:

```
connection.on("UpdateSensorMonitor", function (decBleMessage) {
    switch (decBleMessage.tagID) {
        case "0C61CFC14A84":
            document.getElementById("t1ReportType").innerHTML = decBleMessage.reportType;
            document.getElementById("t1TagID").innerHTML = decBleMessage.tagID;
            document.getElementById("t1GatewayID").innerHTML = decBleMessage.gatewayID;
            document.getElementById("t1Rssi").innerHTML = decBleMessage.rssi;
            document.getElementById("batteryVolt").innerHTML = decBleMessage.rawPacketContent.find(item => item.id == 1).value;
            document.getElementById("eventStatus").innerHTML = decBleMessage.rawPacketContent.find(item => item.id == 2).stringValue;
            document.getElementById("temp").innerHTML = decBleMessage.rawPacketContent.find(item => item.id == 3).value;
            break;
        case "E7DBE2F723F3":
            xValues.push(decBleMessage.rawPacketContent.find(item => item.id == 1).value);
            yValues.push(decBleMessage.rawPacketContent.find(item => item.id == 2).value);
            zValues.push(decBleMessage.rawPacketContent.find(item => item.id == 3).value);
            myChart.update();
            break;
    }
})
```

Figuur 16: Stuk code van het gebruik van SignalR in de front-end

We hebben besloten om niet ook nog de MQTT functionaliteit van de gateway te testen. De belangrijkste reden hiervoor is het feit dat HTTP beter past bij ITS en hoe de LoRa berichten daarin al worden afgehandeld. Ook waren we tijdens het zoeken naar Bluetooth Low Energy gateways voor het Proof of Concept erachter gekomen dat HTTP het meest ondersteunde protocol is. Om er zeker van te zijn dat ITS zoveel mogelijk gateways zou kunnen ondersteunen is daarom de keuze op HTTP gevallen. Een video van de werking van dit proof of concept is te vinden in de bijlagen (bijlage **E**).

6.3 Acceptatietest proof of concept

Voor het proof of concept stonden in principe vijf weken ingepland. Ik was echter sneller klaar dan verwacht, waardoor ik de demo (acceptatietest) een week naar voren heb geschoven. Om er zeker van te zijn dat de van te voren gestelde requirements behaald waren is er een acceptatietest uitgevoerd met collega Mark Franken. Tijdens de demo van het HTTP Proof of Concept heb ik laten zien dat ik de data van de twee sensoren ontving, deze uitpakte en vervolgens liet zien in een simpele front-end pagina. Ook heb ik Mark laten zien dat het programma in C# geschreven was. De totale kosten van de twee sensoren en de gateway waren uitgekomen op 178,46 euro. Na deze demo waren we het er beiden over eens dat alle requirements behaald waren en dat verder testen niet nodig was. Dit betekende dat we konden gaan beginnen aan de integratie in ITS.

7. Integratie in ITS

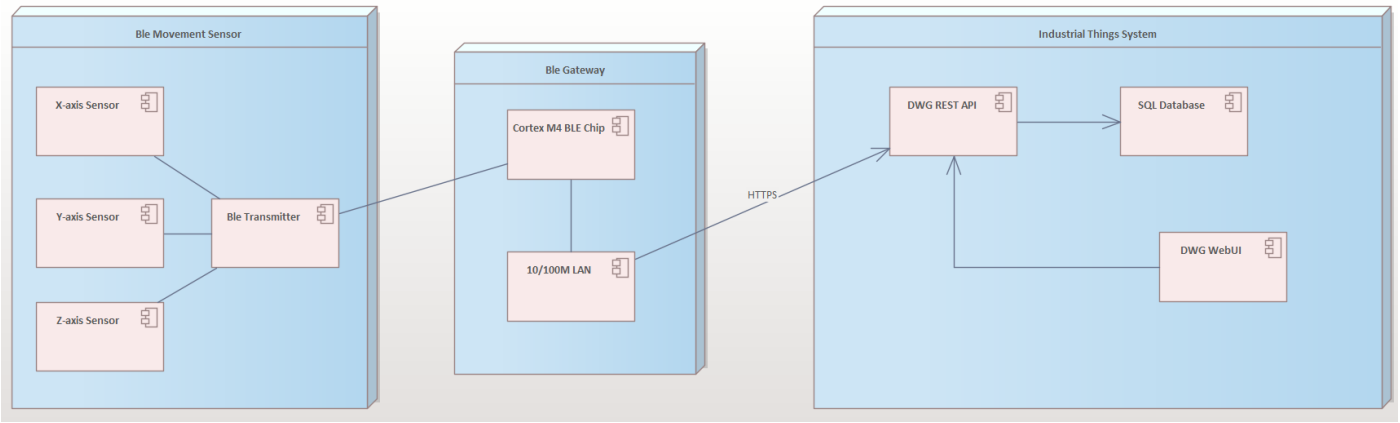
Nadat ik het Proof of Concept succesvol gepresenteerd had, was het tijd om te beginnen aan de derde fase: het integreren van Bluetooth Low Energy in ITS. De eerste stap van deze fase was het uitvoeren van een Scrum sprint planning. Samen met Niels IJpelaar, Mark Franken en Merijn Koster (product owner ITS) heb ik een aantal user stories gemaakt en op grootte beoordeeld door middel van planning poker. Dit is deel van de Scrum methodiek. Merijn heeft de prioriteit van de user stories bepaald. Het thema van de ITS sprints was Marvel. Elke sprint kreeg dus de naam van een Marvel karakter. Toen dit gedaan was kon ik beginnen aan de eerste sprint: “Deadpool”.

Er is gekozen om de hardware van de Proof of Concepts te blijven gebruiken voor de integratie binnen ITS. Dit betekent dus dat ik niet bezig ben geweest met pairing methodes. Deze keuze is gemaakt omdat de prioriteit van mijn opdracht lag bij het uitbreiden van ITS om het nieuwe protocol te kunnen ondersteunen. Hier heeft de verbinding tussen de sensoren en gateways geen invloed op.

7.2 Architectuur

Op hoog niveau is dit hoe de architectuur van de integratie van Bluetooth Low Energy in ITS eruit ziet. Het begint allemaal bij de sensoren. Een sensor bestaat eigenlijk uit meerdere sensoren. Als we als voorbeeld een trillings/bewegingssensor pakken, bevat dit sensor apparaat drie sensoren. Een X-as accelerometer, een Y-as accelerometer en een Z-as accelerometer. Het sensor apparaat bevat ook een Bluetooth Low Energy transmitter die de data van deze drie sensoren broadcast.

De Bluetooth Low Energy gateway heeft twee componenten in zich. Een cortex M4 BLE Chip waarop de berichten van sensoren worden ontvangen. Vervolgens stuurt de gateway deze berichten via een 10/100 ethernet netwerkadapter over HTTPS naar de REST API van ITS. De API stuurt deze berichten naar een code-first SQL database. De structuur binnen ITS bestaat al voor LoRa, maar nog niet voor andere draadloze communicatieprotocollen. Om Bluetooth Low Energy te integreren zal ik dus alle drie de componenten binnen ITS uit moeten breiden.



Figuur 17: Deployment diagram ITS met componenten.

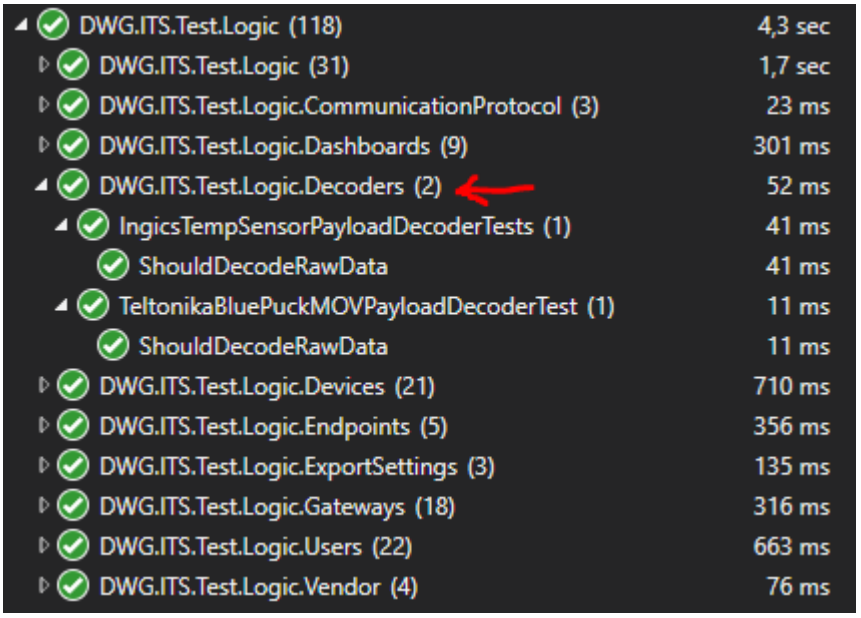
Het belangrijkste doel van de stageopdracht is het kunnen ontvangen en opslaan in de database van Bluetooth Low Energy berichten. Dat proces zal in werking gaan als volgt. De gateway stuurt een array met strings (messages van sensoren) naar de *WebHookController* in de API. Dit is de naam van de controller waar de endpoints staan waarin de data van sensoren wordt ontvangen. Deze stuurt op zijn beurt het array met messages door naar de *WebHookLogic* klasse in de Logic laag. Hier wordt een foreach loop gestart. Het volgende proces wordt dus voor iedere message in de array herhaalt. De message wordt naar de ProcessBleMessageAsync methode gestuurd. Vervolgens wordt het device waar de message vandaan komt en de bijbehorende sensors opgehaald. Het opgehaalde device en de message worden naar de bijbehorende decoder gestuurd. Deze geeft een lijst van measurements terug. De laatste stap is het toevoegen van de nieuwe measurements in de database. Dit wordt afgehandeld in de *BulkInsertAsync* methode. Dit gehele proces heb ik uitgebeeld in een sequentiediagram dat te vinden is in bijlage **F**.

7.3 Deadpool (sprint 1)

De eerste sprint bestond vooral uit het uitbreiden en aanpassen van de API van ITS. Hiervoor waren de volgende user stories ingeplant:

- Als developer wil ik een instantie van ITS op mijn laptop hebben draaien zodat ik kan developen.
- Als ITS gebruiker wil ik ondersteunde devices kunnen toevoegen op basis van protocol, leverancier, type en hiervoor een ID in kunnen vullen. Hierbij kan ik ook de versie van het protocol kunnen aangeven zodat ik deze (het device) kan gebruiken.
- Als ITS gebruiker wil ik voor iedere sensor een decoder hebben zodat ik data hiervan uit kan lezen.

Natuurlijk was het installeren van ITS de eerste stap. Samen met Niels heb ik dit dan ook op de eerste dag van de sprint meteen gedaan zodat ik kon beginnen met ontwikkelen. De tweede user story die ik heb opgepakt was het maken van de decoders omdat ik hier bij het ontwikkelen van het proof of concept al een basis voor had gelegd. Hiervoor moest ik dus programmeren in .NET Core C#. Ik moest de decoders die ik voor het proof of concept gemaakt had aanpassen en integreren in ITS. Ik kon een goed voorbeeld nemen aan de LoRa decoders die al in ITS zaten. Als laatst heb ik Unit Tests geschreven voor de decoders. Hier is meer over te lezen in hoofdstuk 8. De resultaten hiervoor zijn te zien in onderstaande screenshot:

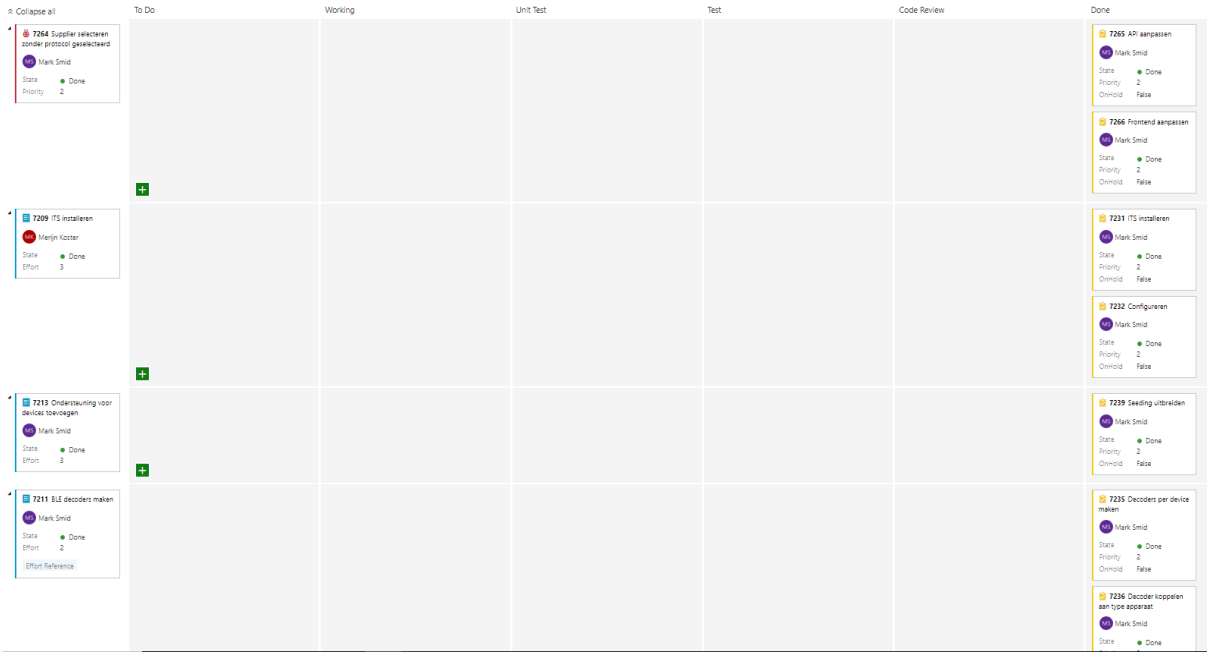


Figuur 18: Resultaten Unit tests ITS (Pijl wijst naar de Unit tests van de decoders).

Ik heb tijdens het ontwikkelen natuurlijk ook gebruik gemaakt van GIT. Ook hiervoor gebruikt DWG Azure DevOps wat het erg makkelijk maakt om direct uit Visual Studio te kunnen committen en pushen. Voor iedere user story waar ik aan ging werken maakte ik een nieuwe branch. Als de code af was committe ik de veranderingen en pushte ik deze. Vervolgens maakte ik een pull request aan voor de veranderingen. Een collega pakte de request op en controleerde de veranderingen. Ook ging er een controle van SonarQube overheen die bijvoorbeeld code smells automatisch herkent. Als alles klopte werden de veranderingen aan ITS toegevoegd en werd de branch verwijderd.

De laatste user story die ontwikkeld moest worden was het toevoegen van ondersteunde devices. Hiervoor moest ik logica toevoegen in de back-end. Dit was functionaliteit om nieuwe device types te verwerken en op te slaan in de database, zodat ITS deze kan herkennen en deze in gebruik genomen kunnen worden. Ook moest ik wat aanpassingen maken in de front-end van ITS. Omdat ik geen ervaring met Vue had moest ik een collega erbij halen om mij op weg te helpen.

Gedurende de sprint was er ook nog een bug naar voren gekomen waarbij de volgorde van het invullen van velden bij het aanmaken van een device niet klopte. Ik kwam hier achter tijdens het ontwikkelen van het toevoegen van devices en heb deze bug toen in DevOps toegevoegd aan de huidige sprint omdat daar voldoende tijd voor was. Aan het einde van de sprint zag het sprintboard er als volgt uit. Alle taken stonden op “Done”:



Figuur 19: Screenshot van het sprintboard einde sprint

Op de laatste dag van de sprint hebben Merijn, Niels en ik de sprint retrospective, review en planning voor de volgende sprint gedaan. Hierin leverde ik uitgewerkte user stories op. Het begon met een demo van de dingen die ik had gemaakt tijdens de sprint. Vervolgens hebben we het gehad over wat goed ging en wat beter kon. Een voorbeeld van een punt die als een positief werd beschouwd was dat als ik ergens mee vast liep en er geen collega beschikbaar was om te helpen, ik niet ging zitten wachten. Ik pakte dan een andere taak op of ging verder met het schrijven van dit afstudeerverslag. Een van de mindere punten was dat Niels en ik de dagelijkse stand-up meeting om twaalf uur nog wel eens vergaten. Merijn gaf als mogelijke oplossing dat we deze meetings in onze agenda moesten zetten. Nog een punt wat beter kon is het eerder klaarzetten van de demo voor de sprint review. Het ging in dit geval specifiek om het demo-en van de koppeling tussen de gateway en de demo omgeving van ITS die in de cloud staat. Lokaal werkte het goed en was het getest, waardoor ik er vertrouwen in had dat het in de cloud omgeving ook zou werken. Dit was echter niet het geval. Toen ik hier achter kwam was het al laat op de dag voor de sprint review. Er was

hierdoor geen tijd meer om dit op te lossen waardoor ik het alleen lokaal kon demo-en. De demo op de cloud omgeving moest dus op een later moment gegeven worden.

Als laatste hebben we de planning voor de volgende sprint (Falcon) gedaan. Deze retrospective, review en planning sessie moest een aantal dagen worden uitgesteld doordat Merijn niet beschikbaar was op de originele datum. Om de extra tijd op te vullen heb ik een extra product backlog item opgepakt.

7.4 Falcon (sprint 2)

De user stories die voor de tweede sprint waren ingeplant waren als volgt:

- Als gebruiker van ITS wil ik opgeslagen metadata kunnen bereiken zodat ik kan debuggen en verder onderzoek kan doen naar issues.
- Als projectuitvoerder wil ik dat er een gateway type beschikbaar is die data kan doorsturen via mobiel netwerk naar de cloud zodat ik zonder aparte netwerkverbinding data naar de cloud omgeving kan krijgen.
- Als gebruiker van ITS wil ik een Bluetooth Low Energy gateway in de front-end kunnen toevoegen zodat ik nieuwe gateways kan aanmaken.
- Als gebruiker van ITS wil ik een Bluetooth Low Energy device in de front-end kunnen toevoegen zodat ik nieuwe devices kan aanmaken.
- Als ITS ontwikkelaar wil ik de gateway koppelen aan de ITS demo-omgeving zodat ik het ontvangen van waardes kan testen en demo-en.

Het extra item wat ik aan het eind van de eerste sprint had opgepakt was het opslaan van metadata van alle binnenkomen LoRa en Bluetooth Low Energy metingen. Dit was aan het eind van de sprint nog niet af waardoor het doorgeschoven werd naar de tweede sprint. Bij het opslaan van metadata kan je denken aan een ‘receivedAt’ time stamp of de RSSI (Received Signal Strength Indicator) van het bericht waar de meting uit komt. Met deze data kan een gebruiker van ITS problemen in de plaatsing van de de hardware (sensoren en gateways) lokaliseren en verhelpen. Om deze functionaliteit te realiseren moest ik de logica aanpassen om extra data te verwerken. Omdat de productowner deze functionaliteit ook voor LoRa berichten wilde hebben, heb ik dezelfde aanpassingen ook toegepast op de LoRa logica. Ook moest ik hiervoor de database uitbreiden. Ik heb een tabel toegevoegd waarin de metadata wordt opgeslagen met een uniek ID. Dit ID is vervolgens opgenomen in de metingen tabel. Elke meting heeft dus een ‘MeasurementMetaDataId’ die verwijst naar een record in de metadata tabel. Dit heb ik gedaan door nieuwe migrations toe te voegen en de seeding aan te passen.

Id	IsDeleted	GatewayID	ReceivedAt	Rssi	Discriminator	DataRate	Adr	Snr	Channel
119	0	CC4B73906E64	2021-05-26 15:58:43.0601106	-80	BleMeasurementMetaData	NULL	NULL	NULL	NULL

Figuur 20: Database record uit de metadata tabel (de velden die ‘NULL’ zijn alleen beschikbaar bij LoRa berichten).

Nog een item dat door was geschoven naar de tweede sprint was het koppelen van mijn gateway aan de demo (cloud)omgeving van ITS. Lokaal werkte het naar behoren, maar het lukte niet om dit voor elkaar te krijgen naar de cloud. De berichten van de gateway kwamen niet aan op de demo server. De eerste gedachte was dat het misschien aan mijn thuisnetwerk zou liggen. Om dit te testen heb ik de rol van de gateway nagebootsts vanuit

Postman. Deze berichten kwamen echter wel goed aan bij de demo server. Het volgende idee was om het te proberen op een andere, in principe identieke server in de cloud. Ook dit werkte gewoon naar toebehoren en de berichten kwamen aan op de server. Het laatste wat we getest hebben is het versturen van berichten van de gateway naar de server via HTTP in plaats van HTTPS. Deze berichten kwamen gewoon goed aan op de demo server. Het werkte dus alleen niet met HTTPS tussen de specifieke server en gateway. Nadat Vincent, Niels en ik alles geprobeerd hadden wat we konden bedenken, hebben we besloten dat het gewoon niet ging lukken met de gateway die we op dat moment hadden en dat we er geen uren meer in zouden stoppen.

Ik heb vervolgens Merijn uitgenodigd voor de stand-up meeting van de volgende dag om input te krijgen over wat ik hier mee aan moest en ook wat ik met deze taak op het scrumboard moest doen. Merijn vertelde mij dat ik de taak on-hold moest zetten tot er een nieuwe gateway beschikbaar was om mee te testen. In de tussentijd kon ik gewoon verder aan de volgende taak.

Die volgende taak was het kunnen toevoegen van een Bluetooth Low Energy gateway via de front-end van ITS. Dit is belangrijk, omdat hierdoor het gatewayId in de database komt te staan. Dit id wordt gebruikt om te bepalen via welke gateway berichten in ITS aankomen. Ook wordt dit Id opgeslagen in de metadata tabel. Dit betekende dat er aanpassingen gemaakt moesten worden in de API en in de front-end. Een techniek die ik in de API heb gebruikt om deze functionaliteit te maken is een generic function. Dit is een functie waarbij je het type pas meegeeft bij het aanroepen daarvan. Bij het gebruik van zo'n functie wil je niet zomaar alle typen kunnen meegeven. Om dit af te vangen en alleen typen mee te geven die onder ITSGateway vallen (dus LoRaGateway of BleGateway), heb ik gebruik gemaakt van een generic type constraint. Door het gebruik van deze techniek heb ik van twee functies een functie kunnen maken. Deze techniek heb ik op meerdere plekken in de code gebruikt. Bij het aanroepen van de generic function moet je het type meegeven tussen de methodenaam en de parameters. Voorbeeld: *GetGatewayById<BleGateway>(3)*

```
0 references
public LoRaGateway GetLoRaGatewayById(int id)
{
    return FindSingle<LoRaGateway>(g => g.Id == id && g.IsDeleted == false);
}

0 references
public BleGateway GetBleGatewayById(int id)
{
    return FindSingle<BleGateway>(g => g.Id == id && g.IsDeleted == false);
}
```

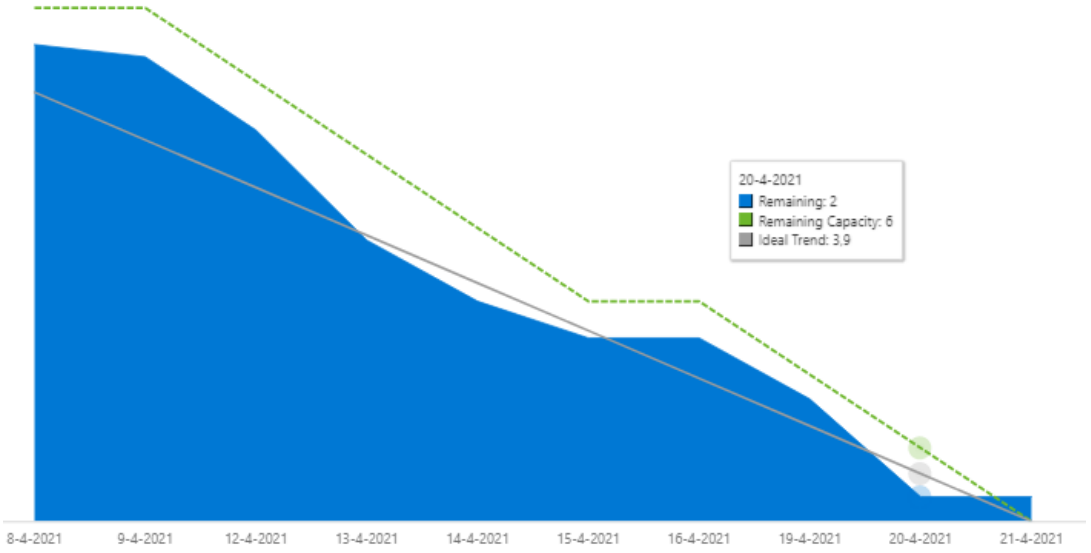
```
3 references
public T GetGatewayById<T>(int id) where T : ITSGateway
{
    return FindSingle<T>(g => g.Id == id && g.IsDeleted == false);
}
```

Figuur 21: Het ophalen van gateways uit de database by Id zonder generic function vs. met generic function

Vervolgens heb ik de taak opgepakt om Bluetooth Low Energy devices (sensor apparaten) toe te kunnen voegen via de front-end van ITS. Dit is belangrijk, omdat hierdoor het deviceId in de database komt te staan. Dit id wordt gebruikt om te bepalen van welk device berichten in ITS aankomen. Metingen zijn in de database daarom ook gekoppeld aan het device

waarvan ze afkomstig zijn. Deze taak was erg vergelijkbaar met het kunnen toevoegen van gateways. Ook was deze taak het laatste puzzelstukje om de de doelstelling te behalen: "*Een uitbreiding op ITS waarbij we (DWG) over een (korte) afstand veel meetwaardes kunnen ontvangen en opslaan door middel van een draadloos communicatieprotocol.".* Nu konden de gateways en devices toegevoegd worden via de front-end en konden de berichten die van de toegevoegde apparaten kwamen ontvangen en opgeslagen worden.

De laatste user story was het vinden van een Bluetooth Low Energy gateway met de mogelijkheid om berichten via mobiel netwerk door te kunnen sturen. Het idee was om een nieuwe gateway te bestellen om op de cloudomgeving te kunnen testen en die daarna ingezet zou kunnen worden bij een eventuele klant. Dit wilde Merijn graag, omdat klanten over het algemeen geen ethernet of Wi-Fi beschikbaar hebben voor de gateways. Daarom maakt DWG in deze gevallen gebruik van gateways die de data via mobiel netwerk kunnen versturen naar ITS. Hiervoor heb ik een aantal van deze gateways geselecteerd en voorgelegd aan de collega die hier de doorslaggevende beslissing voor moest maken. Hiermee waren alle user stories (buiten de gestaakte koppeling met de demo omgeving) af. Het niet af kunnen maken van de ene user story is te zien in de burn-down grafiek. Je ziet dat er nog "remaining work" over is:



Figuur 22: Screenshot burn-down grafiek

Om de sprint af te sluiten is er weer een meeting gehouden met Merijn en Niels waarin de sprint review, retrospective en planning voor de volgende sprint werden gedaan. Hierin leverde ik uitgewerkte user stories op. De minpunten van de vorige sprint kwam dit keer niet voor. Ik ben eerder begonnen met testen op de demo omgeving, waardoor ik eventuele problemen hiermee eerder kon ontdekken. Waar de dagelijkse stand-ups in de vorige sprint nog wel eens vergeten werden, was dat deze sprint heel anders. Gedurende de twee weken van de sprint is er geen stand-up meeting gemist. Het enige punt voor verbetering was dat ik Merijn beter up-to-date kon houden tijdens de sprint over het verloop van de ontwikkelingen.

7.5 Groot (sprint 3)

De derde sprint was iets anders ingedeeld dan de vorigen. Ik miste een werkdag doordat koningsdag (27 april) binnen deze sprint viel. Ook was er met het tussentijds assessment in het achterhoofd meer tijd ingepland voor het afstudeerverslag. Toch hebben we een goed aantal user stories in kunnen plannen:

- Als gebruiker van ITS wil ik een Bluetooth Low Energy device in de front-end kunnen bewerken en verwijderen zodat ik de ITS omgeving up-to-date kan houden.
- Als gebruiker van ITS wil ik een Bluetooth Low Energy gateway in de front-end kunnen bewerken en verwijderen zodat ik de ITS omgeving up-to-date kan houden.
- Als gebruiker van ITS wil ik dat de Bluetooth Low Energy trillingssensor (Blue Puck MOV) werkt en data verstuurd op een interval van 10x per seconden zodat ik deze kan gebruiken.
- Als ITS product owner wil ik ontwerpen van een filter algoritme hebben zodat ik kan bepalen of dat de richting is die we op willen gaan.

Als eerst heb ik mij kunnen bezighouden met de functionaliteit om Bluetooth Low Energy devices en gateways vanuit de front-end van ITS aan te kunnen passen en te kunnen verwijderen. Het ontwikkelen van deze user stories bestond vooral uit het toevoegen van functionaliteit in de back-end. Deze functionaliteit was het aanpassen en verwijderen van devices en gateways in de gateway in de database. Devices worden echter niet compleet verwijderd uit de database. Measurements in de database zijn gekoppeld aan een device. Deze data moest bewaard blijven. Om die reden wordt er bij het verwijderen van devices het ‘isDeleted’ veld op true gezet. Zo kan het device in de database blijven bestaan terwijl het voor de eindgebruiker verwijderd lijkt te zijn.

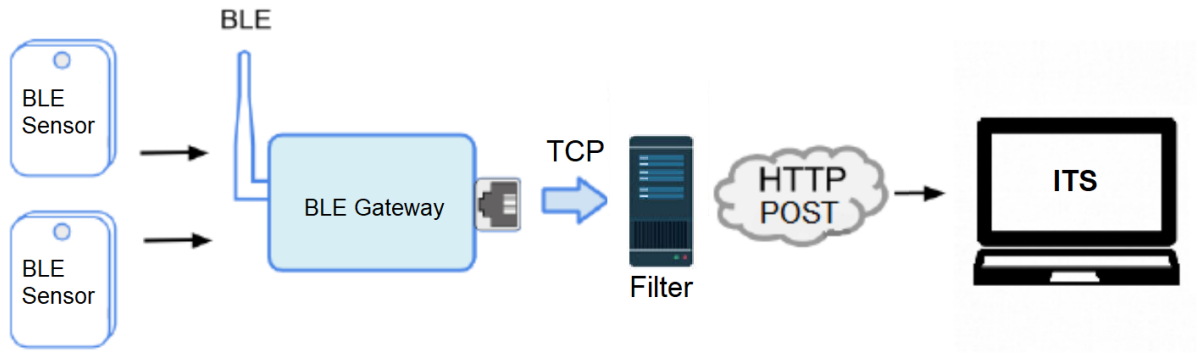
De trillingssensor user story was eenvoudiger dan verwacht. Deze was defect, wat de reden was waarom de user story aangemaakt was. Ik kon de zend interval niet sneller instellen dan 1x per seconde terwijl de sensor tot 10x per seconde zou moeten kunnen verzenden volgens de meegeleverde documentatie. Ik heb de sensor teruggestuurd naar de leverancier en kreeg deze een aantal dagen later werkend terug. Ook heb ik in deze sprint relatief veel tijd besteed aan het schrijven van dit verslag. Dit komt omdat ik het verslag zo ver mogelijk af wilde hebben voor het tussentijds assessment, zodat ik op zoveel mogelijk feedback zou kunnen krijgen.

7.5.1 Het data probleem

Een van de mogelijke toepassingen voor Bluetooth Low Energy in ITS waar ook al daadwerkelijk vraag naar was door een klant, was het meten van trillingen op grote leidingen waarin kleppen open en dicht gaan. De klant wil de trillingen meten die gebeuren wanneer de klep in zo’n leiding open of dicht gaat. Dit moet voor ongeveer twintig leidingen gebeuren op een hoge zend interval van de sensoren (tien keer per seconde). Na hier met collega’s over te spreken kwamen we erachter dat dit voor veel te veel data zou zorgen in de database. Zoals eerder aangegeven heeft een trillingssensor drie sensoren (X-as, Y-as en Z-as). Dit zou dus zorgen voor 3 x 20 x 10 = 600 metingen per seconde. Dit komt uit op 51.840.000 metingen per dag. Ook wilden we de API van ITS niet overspoelen. Hier moest dus een oplossing voor gevonden worden. Er moest een stukje edge computing gebeuren

om data van de trillingssensoren te filteren voordat het opgeslagen werd in de database. Edge computing is een manier om lokaal dataverkeer van apparaten te verwerken[8]. Het wordt vaak ingezet om bandbreedte te besparen. In het geval van ITS zal dit stukje Edge computing gebeuren bij de gateway, voordat de data naar ITS verstuurd wordt. Alleen de metingen die gebeuren tijdens het openen of dicht doen van een klep zijn interessant voor het bedrijf. De metingen die gebeuren terwijl de kleppen niet bewegen kunnen dus weggefilterd worden door middel van edge computing. Het edge computing programma verwerkt de data lokaal waardoor de data stroming naar de centrale opslagplaats (ITS) wordt verminderd.

De oplossing voor dit probleem bestaat uit meerdere delen. De software die de data filtert en de hardware waarop dit filter draait. Deze hardware moet tussen de Bluetooth Low Energy gateways en ITS komen te staan. Het filter verkrijgt berichten van de gateway via TCP zoals dat ook gedaan werd bij het eerste proof of concept. Vervolgens worden de gefilterde berichten naar ITS verstuurd via HTTP POST requests. Het volgende diagram geeft dat weer:



Figuur 23: Diagram dat plaatsing van het filter weergeeft

Ik heb samen met mijn collega's besloten dat het uitzoeken van de hardware hiervoor buiten de scope van mijn afstudeerstage zou vallen. Wel kon ik de code die het filteren afhandeld schrijven. Hiervoor moest ik een soort peak detection algoritme gaan ontwikkelen dat het filteren van data kon afhandelen voor meerdere sensoren tegelijk. Dit was een té grote klus om in de tweede helft van deze sprint te kunnen verwezenlijken. Daarom heb ik samen met de product owner bepaald dat voor deze sprint alleen het ontwerp van het algoritme gemaakt zou moeten worden. De daadwerkelijke implementatie van deze ontwerpen zou in de volgende sprint gaan gebeuren.

De requirements voor de filter applicatie waren als volgt, geprioriteerd volgens de MoSCoW methode:

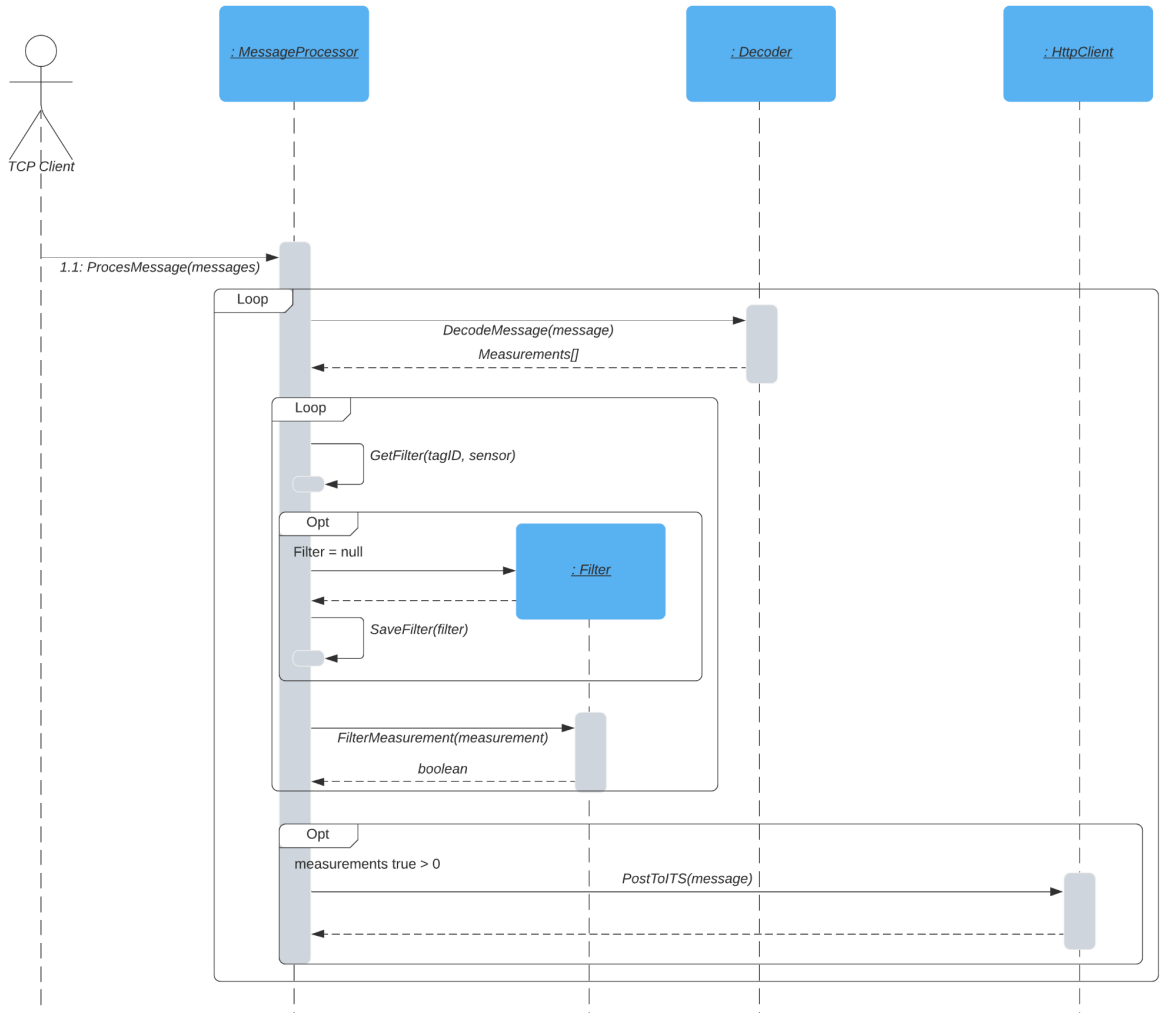
Requirement	Prioriteit
De applicatie moet in c# geschreven worden	Must-Have
Het algoritme moet configureerbaar zijn zodat deze geoptimaliseerd kan worden voor de specifieke toepassing	Must-Have
Het algoritme moet kunnen werken met meerdere sensoren	Must-Have
De applicatie moet een console applicatie zijn	Must-Have

Figuur 24: Requirements filter applicatie

Als eerst ben ik op zoek gegaan naar bestaande peak detection algoritmen. Ik ging er van uit dat we niet de eerste zouden zijn met dit probleem. Het zou dan natuurlijk zonde zijn om het wiel zelf opnieuw uit te gaan vinden. Tijdens deze zoektocht ben ik het algoritme van Jean-Paul van Brakel[9] tegengekomen. Dit is een veelgebruikt algoritme dat 30+ keer academisch geciteerd is. Het werkt door constant een gemiddelde te berekenen aan de hand van de ontvangen datapunten en hier een threshold omheen te zetten. De datapunten die buiten deze threshold vallen zijn de “interessante” datapunten. Dit kwam natuurlijk goed overeen met het probleem van ITS. Dit algoritme zou de basis worden voor mijn oplossing.

Het algoritme van van Brakel was een belangrijk deel van de oplossing, maar het was niet plug-and-play. Het zou aangepast en uitgewerkt moeten worden voor de specifieke situatie van ITS. Als er data van 20 sensoren door elkaar heen in het algoritme worden gestopt, kan daar natuurlijk nooit een accuraat gemiddelde uit worden gehaald. Voor iedere sensor zou dus een eigen dataset moeten komen waar het algoritme op zou draaien. Ook moesten de datapunten die buiten de threshold zouden vallen doorgestuurd worden naar ITS. Deze datapunten zouden dus teruggekoppeld moeten worden aan het bericht waar deze uitkwamen.

Ik heb een document gemaakt waarin ik de hierboven genoemde werking heb ontworpen door middel van UML-diagrammen en schriftelijke uitleg. Het ontwerp van het algoritme was net voor het einde van de sprint af waardoor ik het kon presenteren aan Merijn tijdens de gebruikelijke sprint review. Onderstaand diagram is deel van het complete ontwerp. Het totale ontwerp document is te vinden in bijlage **G**



Figuur 25: UML Sequentiediagram ter ontwerp van het filter algoritme.

Met het afmaken van dit ontwerp kwam de sprint tot zijn eind. Net als alle anderen werd ook deze sprint afgesloten met een meeting met Merijn en Niels waarin de sprint review, retrospective en planning voor de volgende sprint werden gedaan. Hierin leverde ik uitgewerkte user stories op. Het verbeterpunt van de vorige sprint was dat ik Merijn beter up-to-date kon houden tijdens de sprint. Dit had ik ook gedaan dus dat was positief.

7.6 Hawkeye (sprint 4)

Voor de vierde sprint waren de volgende user stories ingepland:

- Als gebruiker van ITS wil ik een filter applicatie zodat waardes die continu hetzelfde zijn niet worden doorgestuurd naar de ITS API, zodat de API en database niet overspoeld worden.
- Als gebruiker van ITS wil ik verwijderde devices met dezelfde naam kunnen toevoegen zodat ik de data historie kan koppelen aan het nieuwe device.

Ook was er een bug toegevoegd aan de sprint. Na het bewerken van een device werd de lijst van suppliers niet geleegd wat voor problemen zorgde bij het toevoegen van het volgende device.

Uit de review van de vorige sprint was naar voren gekomen dat het ontwerp op een aantal punten verbeterd kon worden. Dit waren gelukkig kleine punten die ik meteen in het sequentiediagram heb verwerkt. Het ontwikkelen van het algoritme had daarna de hoogste prioriteit dus dat was de volgende stap. Aan de hand van het ontwerp ben ik begonnen met programmeren. Gedurende de sprint werd echter duidelijk dat het programmeren van het algoritme meer tijd in beslag ging nemen dan was gepland. Daarom heb ik met de product owner besloten om de andere user story en de bug uit de sprint te verwijderen.

Tijdens het ontwikkelen van de filter applicatie heb ik een aantal technieken voor het eerst toegepast. Het eerste voorbeeld hiervan is het singleton design pattern. Om de gefilterde berichten naar ITS te sturen heb ik een HTTP client gemaakt. De reden hiervoor is dat het volgens de Microsoft documentatie de bedoeling is dat er in de hele applicatie maar één client is. De code voor dit design pattern is als volgt:

```
public static class ItsHttpClient
{
    private static HttpClient client;
    private static readonly object lockObject = new object();
    1 reference
    public static HttpClient Instance // singleton.
    {
        get
        {
            if (client == null)
            {
                lock (lockObject)
                {
                    if (client == null)
                    {
                        var httpClientHandler = new HttpClientHandler();
                        client = new HttpClient(httpClientHandler);
                    }
                }
            }
            return client;
        }
    }
}
```

Figuur 26: Stuk code dat het singleton design pattern weergeeft.

Wanneer er in de applicatie een instance van deze klasse wordt gemaakt controleert deze code of er al een instance bestaat. Als dit het geval is, wordt deze instance teruggegeven. Als dit niet het geval is wordt een nieuwe (eerste) instance gemaakt en wordt deze

teruggeven. Op deze manier wordt er voor gezorgd dat er te allen tijde maximaal één instance van de klasse bestaat. Als je naar de code kijkt vraag je je misschien af waarom er tweemaal wordt gechecked op (client == null). Dit is om de code thread safe te maken. Het zou kunnen gebeuren dat er meerdere threads op hetzelfde moment de “get” aanroepen. Deze komen in het geval dat er nog geen instance is allemaal door de eerste if. Daarom is de volgende line code een lock. Op het moment dat de eerste van de threads op deze line komt worden de anderen hier tegengehouden. Deze lock gaat pas weer open wanneer de eerste thread de code binnen de lock heeft doorlopen (het aanmaken van een nieuwe client). Als dit gebeurd is gaan de andere threads pas de tweede if in, waar ze erachter komen dat er al een client bestaat. Zo wordt er voorkomen dat er meerdere clients aangemaakt worden op het moment dat in meerdere threads op hetzelfde moment de “get” wordt aangeroepen.

Ook heb ik voor het eerst configuratie door middel van een appsettings.json gemaakt in een console app. Hierdoor kan de gebruiker van de filter applicatie het algoritme (lag, threshold en influence), de gegevens van de gateway waar de applicatie bij draait en het ITS endpoint waar de applicatie gefilterde berichten naar toe moet sturen instellen. Dit wordt gedaan met een builder. Er wordt onderscheid gemaakt tussen productie settings en development settings door middel van een speciale if else. Dit heet een “conditional compilation”. Hiermee kan je bepaalde stukken code alleen laten compileren als er voldaan wordt aan een bepaalde conditie. In mijn situatie is die conditie of er gedebugged word of niet. Als dat het geval is word het appsettings.Development.json bestand gebruikt voor de appsettings. Als dit niet het geval is wordt het appsettings.json bestand hiervoor gebruikt. Onderstaande code geeft dit weer:

```
1 reference
static IConfigurationRoot BuildConfig(IConfigurationBuilder builder)
{
    return builder.SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
#if !DEBUG
        .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
#else
        .AddJsonFile("appsettings.Development.json", optional: true)
#endif
        .AddEnvironmentVariables()
        .Build();
}
```

Figuur 27: Stuk code dat de configuratie builder weergeeft

Aan het eind van de sprint was de filter applicatie zo goed als af, maar er was geen tijd meer om deze te kunnen testen. Dit zorgde voor een hele korte sprint review. Tijdens de retrospective kwam naar voren dat de grootte van de filter algoritme user story niet goed was ingeschat. Dit zou voor de volgende sprint beter moeten. Buiten dit ene punt ging de sprint erg goed en Merijn was erg enthousiast over het algoritme. Tot slot hebben we de volgende sprint gepland.

7.7 Iron Man (sprint 5)

De vijfde en daarmee ook laatste sprint was anders dan de anderen. Ik heb samen met mijn begeleider afgesproken om de laatste week van de stage te gebruiken voor het afmaken van het afstudeerverslag. Merijn kwam daarom met het idee om de laatste sprint één week te maken in plaats van twee. Dit is niet gebruikelijk binnen SCRUM, maar dat leek ons toch de beste optie voor mijn situatie. Hierbij kwam ook nog eens dat Pinksteren in deze ene week viel wat ervoor zorgde dat ik maar vier dagen had voor de laatste sprint. Met dit in ons achterhoofd hebben we maar een enkele user story ingepland:

- Als gebruiker van ITS wil ik een filter applicatie zodat waardes die continu hetzelfde zijn niet worden doorgestuurd naar de ITS api zodat de API en database niet overspoeld worden.

Dit was een voortzetting van de ontwikkelingen van de vorige sprint, omdat deze user story nog niet af was. Voor deze user story moest ik een aantal producten opleveren:

- De afgemaakte filter applicatie (algoritme)
- Testplan (het deel voor het algoritme)
- Testrapport (het deel voor het algoritme)
- Een handleiding met richtlijnen voor de configuratie van het algoritme

De filter applicatie zelf was na de vorige sprint zo goed als af. Toen ik hier de laatste loodjes voor gelegd had kon ik me gaan richten op het uitbreiden van het testplan (bijlage **H**) met tests voor de filter applicatie en het schrijven van deze tests. Door de beperkte tijd heb ik maar drie unit tests kunnen maken. Deze worden verder toegelicht in hoofdstuk 8 en het testrapport (bijlage **I**). In dit testrapport zijn de resultaten van de uitgevoerde tests opgenomen.

Zoals eerder toegelicht kan de gebruiker van de filter applicatie een aantal instellingen configureren. Om dit zo gebruikersvriendelijk mogelijk te maken wilde Merijn graag een handleiding met richtlijnen voor deze instellingen. Deze handleiding is te vinden in bijlage **J**. Met ondersteuning van de beschrijving van het originele algoritme van Jean-Paul van Brakel heb ik in de handleiding kort beschreven hoe het algoritme werkt en welke instellingen door de gebruiker geconfigureerd kunnen worden. Vervolgens geef ik per instelling een korte beschrijving van hoe deze het best geconfigureerd kan worden. Dit betekent niet dat dit gelijk voor een perfecte configuratie zorgt, maar het zijn vuistregels die een leidraad bieden.

Bij de sprint review heb ik alle van te voren afgesproken op te leveren producten opgeleverd en hier een demo van gegeven. Ik heb hier ook aangegeven dat meer testen benodigd was voor de filter applicatie. Merijn was het hier mee eens, maar was ook erg tevreden met het opgeleverde werk. Al het werk dat voor de sprint was gepland was ook af. Het verbeterpunt die in de vorige retrospective naar boven was gekomen was dus verbeterd. De schatting klopte dit keer precies. Omdat dit de laatste sprint van de stage was vond er geen sprintplanning meer plaats.

8. Testen

Om de kwaliteit van mijn code te garanderen moest er getest worden. Hiervoor heb ik een test plan uitgewerkt waarin wordt beschreven wat er getest moest worden en hoe dit gedaan moest worden. Dit document is te vinden in bijlage **H**. Ook heb ik een testrapport uitgewerkt waarin de resultaten van de uitgevoerde tests te vinden zijn. Dit document is te vinden in bijlage **I**. In dit hoofdstuk zullen een aantal van deze uitgewerkte tests en de resultaten hiervan toegelicht worden.

8.1 Unit tests

Gedurende de stage heb ik een aantal unit tests geschreven. Hierbij heb ik gebruik gemaakt van het xUnit framework omdat dit de standaard is binnen DWG. Het eerste voorbeeld hiervan zijn de tests die ik geschreven heb voor de decoders in ITS. Deze tests controleren of de decoders het ruwe bericht correct uitpakken. Hiervoor worden test sensoren, een test device en de bijbehorende decoder aangemaakt. Om dit te testen wordt er een ruw bericht meegegeven waarvan de correcte meetwaardes bekend zijn. Door middel van Asserts wordt er vervolgens gecontroleerd of de decoder ook daadwerkelijk deze waardes teruggeeft. Onderstaande code is de unit test voor de DecodeRawData() methode binnen de decoder van de iBS03T temperatuursensor:

```
[Fact]
0 references
public async Task ShouldDecodeRawData()
{

    sensors.Add(await CreateAndGetTestSensorAsync(66, "Battery"));
    sensors.Add(await CreateAndGetTestSensorAsync(67, "EventStatus"));
    sensors.Add(await CreateAndGetTestSensorAsync(68, "Temperature"));
    this.decoder = new IngicsTempSensorPayloadDecoder(sensors);

    IEnumerable<Measurement> measurements = new List<Measurement>();
    BleDevice testDevice = await CreateAndGetTestBleDeviceAsync(1, DefaultVendorDevices.IngicsTempSensor.Id);
    measurements = decoder.Decode(testDevice, "02010612FF0D0083BC4A0100A10AFFFF000015000000");

    Assert.NotNull(measurements.ElementAt(0) as DoubleMeasurement);
    Assert.Equal(3.3, ((DoubleMeasurement)measurements.ElementAt(0)).Value);

    Assert.NotNull(measurements.ElementAt(1) as StringMeasurement);
    Assert.Equal("00", ((StringMeasurement)measurements.ElementAt(1)).StringValue);

    Assert.NotNull(measurements.ElementAt(2) as DoubleMeasurement);
    Assert.Equal(27.21, ((DoubleMeasurement)measurements.ElementAt(2)).Value);
}

3 references | 0/1 passing
private async Task<Sensor> CreateAndGetTestSensorAsync(int id, string sensorName)
{
    SensorGenerator sensorGenerator = new SensorGenerator(fixture);
    return await sensorGenerator.CreateAndGetTestSensorGeneratorAsync(id, sensorName);
}

1 reference | 0/1 passing
private async Task<BleDevice> CreateAndGetTestBleDeviceAsync(int id, int vendorDeviceID)
{
    DeviceGenerator deviceGenerator = new DeviceGenerator(fixture);
    deviceGenerator.VendorDeviceID = vendorDeviceID;
    return await deviceGenerator.CreateAndGetTestBleDeviceAsync(id, "EAC653D3AA8D");
}
```

Figuur 28: Stuk code van de unit test voor de iBS03T temperatuursensor decoder

Ook voor het edge filtering algoritme heb ik een aantal unit tests geschreven zoals beschreven in het testplan. Het doel was om als eerst zo veel mogelijk code coverage te krijgen voor het algoritme. Met de volgende test voor de FilterMeasurement() methode binnen de Filter klasse is een code coverage van 100% behaald:

```
[Fact]
0 references
public void ShouldFilterMeasurement() // Test typical situation
{
    Filter filter = new Filter(15, 4.0, 0);

    Measurement[] measurements = FillDataset();
    List<Measurement> filteredMeasurements = new List<Measurement>();
    Measurement[] expectedResult = FillExpectedResult(measurements);

    foreach (Measurement measurement in measurements)
    {
        if (filter.FilterMeasurement(measurement))
        {
            filteredMeasurements.Add(measurement);
        }
    }

    Assert.Equal(10, filteredMeasurements.Count);

    Measurement[] actualResult = filteredMeasurements.ToArray();

    Assert.Equal(expectedResult, actualResult);
}
```

Figuur 29: Stuk code van de unit test voor het filter algoritme

Bij deze test wordt er een dataset in het algoritme gestopt waarvan van te voren bekend is welke waardes buiten de threshold zouden moeten vallen voor de gegeven instellingen (lag = 15, Threshold = 4.0 en Influence = 0). Deze dataset is te vinden in bijlage **K**. Vervolgens wordt er een lijst met resultaten vergeleken met een lijst met de verwachte resultaten. Als deze lijsten overeenkomen slaagt de test. Om te controleren hoeveel code coverage behaald was met deze test heb ik gebruik gemaakt van de volgende packages:

- coverlet.msbuild
- Microsoft.CodeCoverage
- Microsoft.NET.Test.Sdk
- xunit
- xunit.runner.visualstudio
- XunitXml.TestLogger
- ReportGenerator (by Daniel Palme)

Dit is zoals voorgeschreven in de documentatie van DWG. Hierin wordt verwezen naar de volgende bron: [10] Uit dit proces was het volgende report gekomen:

– EdgeFiltering	58	211	269	564	21.5%		8	34	23.5%	
Device	0	4	4	39	0%		0	0		
EdgeFiltering.Filter	55	0	55	93	100%		8	8	100%	
EdgeFiltering.ItsHttpClient	0	19	19	34	0%		0	4	0%	
EdgeFiltering.Measurement	3	0	3	10	100%		0	0		
EdgeFiltering.MessageProcessor	0	94	94	154	0%		0	20	0%	
EdgeFiltering.Program	0	18	18	31	0%		0	0		
EdgeFiltering.SendItsHttpClient	0	28	28	50	0%		0	0		
EdgeFiltering.TcpClient	0	43	43	75	0%		0	2	0%	
Root	0	1	1	39	0%		0	0		
Sensor	0	4	4	39	0%		0	0		

Figuur 30: Code Coverage report van filter algoritme unit test

De rest van de uitgewerkte unit tests zijn zoals eerder vermeld te vinden in het testrapport (bijlage **I**).

8.2 Use case test

Een van de belangrijkste functionaliteiten binnen ITS is het toe kunnen voegen van Bluetooth Low Energy devices. Om de kwaliteit van mijn ontwikkelingen op dit gebied te verifiëren heb ik hiervoor een use case test gemaakt. Voor het bewerken en verwijderen van devices zijn ook use case tests gemaakt. Ook zijn er use case test gemaakt voor deze handelingen (bewerken en verwijderen) van gateways. Al deze uitgewerkte use case tests zijn te vinden in het testplan (bijlage **H**).

Use-case 1: Toevoegen Ble device vanuit front-end ITS.

Scenario	Toevoegen Ble device vanuit de front-end van ITS door een gebruiker	
Description:	Deze use case laat zien hoe een gebruiker van ITS een Bluetooth Low Energy device toe kan voegen vanuit de user interface (front-end)	
Triggering event	Er wordt een nieuw device in gebruik genomen bij een klant.	
Actor(s)	ITS gebruiker	
Stakeholder(s)	Gebruiker, bedrijf gebruiker	
Pre-condition	Gebruiker moet vanuit bedrijf toestemming hebben om devices toe te voegen in ITS. Gebruiker is ingelogd in ITS. Gebruiker moet tagId (MAC-adres) van device weten. Supplier van device moet ondersteund worden door ITS. Device type van device moet ondersteund worden door ITS.	
Post-condition	Device is opgeslagen in database en wordt weergegeven in ITS front-end	
Flow of events	**Actor**	**System**
	1. Schaft nieuw device aan. 2. Klikt op ‘devices’ in menu ITS. 4. Klikt op + knop op het devices scherm in ITS. 6. Vult de benodigde velden in: - ‘*Name’* met de naam van het device. - ‘*Description’* met een korte beschrijving van het device. - ‘*Protocol’* met ‘Bluetooth Low Energy’.	3. Opent Devices scherm. 5. Opent scherm met in te vullen velden 8. Slaat nieuwe device op in database. 9. Toont nieuwe device in devices scherm.

	- '*Supplier'* met de supplier van het device. - '*Device Type'* met het juiste type van het device - '*Protocol version'* met 'NA'. - '*TagId'* met het tagId (MAC-adres) van het device. - '*Device Uplink Interval'* met een getal dat aangeeft hoe vaak het apparaat data zendt in seconden. 7. Klikt op 'Save' knop.	
Results	Gebruiker heeft succesvol een nieuw Bluetooth Low Energy device toegevoegd aan ITS.	
Exception	1. TagId is 'invalid'. Het ingevulde tagId komt niet overeen met het correcte format: __-__-__-__-__-__ 2. TagId is a 'duplicate key'. Het ingevulde tagId bestaat al in de database. 3. 'Field is required'. Niet alle benodigde velden zijn ingevuld. 'Save' knop doet niets.	

Figuur 31: Use case test toevoegen BLE device

8.3 Integratietest en systeemtest

Helaas liep de stage tegen zijn einde voordat ik de kans kreeg om de in het testplan beschreven integratietest en systeemtest uit te voeren.

9. Evaluatie

9.1 De aanpak

Als ik terugkijk op de aanpak van deze stage ben ik erg tevreden. Aan het begin van de stage die ik heb uitgevoerd in mijn derde studiejaar heb ik de fout gemaakt om te veel tijd te spenderen aan het uitvoeren van een onderzoek. Hierdoor bleef er te weinig tijd over om het product zover te kunnen ontwikkelen als dat ik had gewild. Met deze leerervaring in mijn achterhoofd heb ik tijdens het plannen van de drie fases (onderzoek, PoC, integratie) het uitvoeren van het onderzoek kort gehouden, namelijk twee weken. Tijdens de stage heb ik mij ook aan deze planning kunnen houden, waardoor ik veel tijd had om producten te ontwikkelen. Ook de uitvoering van het onderzoek ben ik grotendeels tevreden over. Ik ben van mening dat Bluetooth Low Energy (het protocol dat als “winnaar” uit het onderzoek is gekomen) ook daadwerkelijk het best geschikte protocol was voor integratie in ITS.

Dat ik tevreden ben betekent niet dat het perfect is gegaan. Door het aantal protocollen dat in het onderzoek aan bod kwam, was het namelijk niet mogelijk om binnen twee weken heel diep in te gaan op ieder protocol. Het onderzoeksrapport reflecteert dit. Mijn collega’s en de product owner waren hier juist blij mee, omdat dat precies was wat ze van mij gevraagd hadden. Ik kreeg hier vanuit school echter meerdere malen negatieve feedback op. Volgens de feedback miste het onderzoek diepgang. Als ik het onderzoek opnieuw zou uitvoeren zou ik het daarom verlengen met een week. Dan zou ik het onderzoek wat meer diepgang kunnen geven. Ik betwijfel echter of dit ook toegevoegde waarde had opgeleverd voor de volgende fases.

Voor het ontwikkelen van het proof of concept was van te voren geen specifieke ontwikkelmethode gekozen. Uiteindelijk is dit op een iteratieve manier gebeurd. Ik was begonnen met de meest simpele versie, en heb daarna een uitgebreidere versie met front-end gebouwd. Dit was echter geen “formele” methode waarbij de regels van een specifieke methodiek gevolgd zijn. Ik had het idee dat dit gebruikelijk was bij het maken van proof of concepts. Net als bij mijn vorige stage was het de bedoeling dat het proof of concept zo snel mogelijk het doel zou behalen (bewijzen dat een integratie in ITS mogelijk was). Het maken van ontwerpen en uitgebreid testen zijn daarom stappen die worden overgeslagen, omdat deze in het geval van een (klein) proof of concept geen toegevoegde waarde bieden. Dit was ook in de lijn van verwachting vanuit DWG. Deze manier van werken zorgde ervoor dat ik snel naar de volgende, en meeste belangrijke, fase van de stage kon; de integratie. Tijdens de feedbackmomenten werd echter duidelijk dat de verwachting vanuit school anders was. Ik kreeg de feedback dat voor het proof of concept ook ontwerpen gemaakt en tests uitgevoerd moesten worden. Ik heb de beslissing genomen om deze werkzaamheden niet meer uit te voeren omdat ik inmiddels al diep in de integratie in ITS zat. Als ik de stage opnieuw zou doen zou ik dit wel anders aanpakken.

Voor het ontwikkelen van de integratie in ITS is de SCRUM methodiek gevolgd. Dit is naar mijn mening erg goed gegaan. Ik was hier al mee bekend vanuit school en mijn vorige stage waardoor het makkelijk was om op te pakken. Ik ben blij dat ik hier nog meer ervaring mee op heb kunnen doen omdat ik het idee heb dat SCRUM de standaard is binnen software development tegenwoordig.

9.2 De producten

9.2.1 Onderzoek draadloze communicatieprotocollen

Het uiteindelijke product wat uit het onderzoek naar draadloze communicatieprotocollen is gekomen is het onderzoeksrapport. Het doel van dit rapport was het informeren en adviseren over de mogelijk te integreren protocollen. Ik ben zelf erg tevreden over dit rapport, mede door de feedback van mijn collega's. Zij vonden het erg duidelijk en waren blij dat ik het beknopt had gehouden. Ik ben van mening dat het de relevante informatie duidelijk weergeeft en mijn keuze goed onderbouwd.

9.2.2 Het proof of concept

Het doel van het proof of concept was om uit te zoeken hoe Bluetooth Low Energy hardware werkt, hoe het te gebruiken is en ook vooral om te bewijzen dat het inderdaad geschikt was om te integreren in ITS. Naar mijn mening is dit ook goed gelukt. Zoals in de video bijlage (bijlage **E**) te zien is wordt de data van meerdere sensoren tegelijk in real time weergegeven in een simpele front-end. Hierdoor kon ik laten zien dat die belangrijke functionaliteit van het frequent metingen kunnen verwerken mogelijk was met Bluetooth Low Energy. Aan het eind van de proof of concept fase heb ik een acceptatietest uitgevoerd met Mark Franken. Dit was geen hele formele test maar simpelweg een demo en een overleg met Mark. Dit vonden wij voldoende voor het proof of concept. Er is hier daarom ook geen rapport over opgesteld.

De proof of concept fase vond ik erg leuk. Dit komt omdat het vooral bestond uit nieuwe dingen uitzoeken en leren. Ook heb ik een beter begrip gekregen over hoe Bluetooth Low Energy berichten in elkaar zitten en hoe je die data moet verwerken.

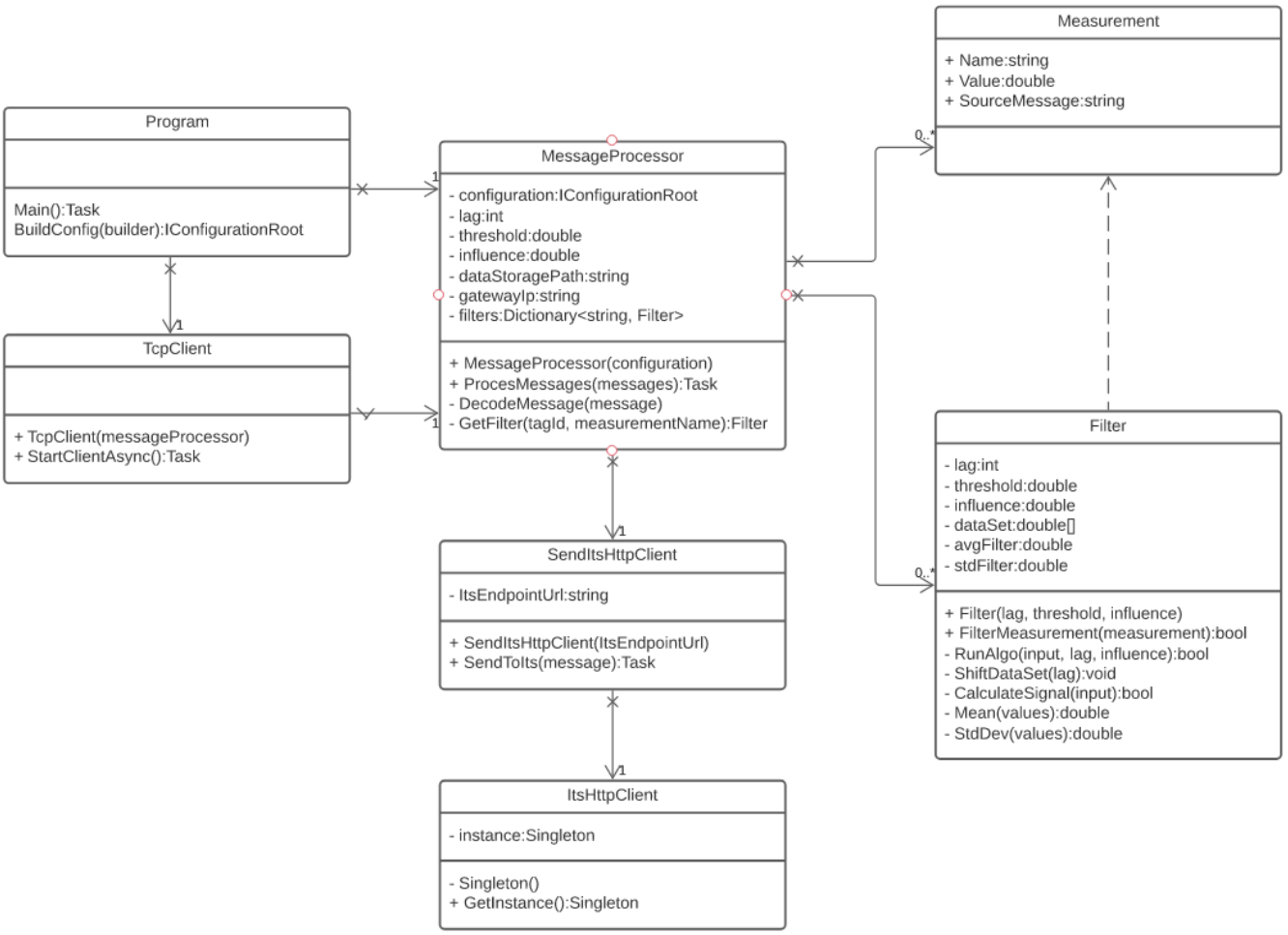
9.2.3 Bluetooth Low Energy in ITS

Het doel van de integratie van een draadloos communicatieprotocol was het kunnen ontvangen en opslaan van berichten van sensoren. Dit doel was na de tweede sprint al behaald. Ik heb daarom tijdens de overige sprints deze integratie kunnen uitbreiden met bijvoorbeeld het edge filtering algoritme. DWG en zijn daarom erg tevreden met het eindresultaat van de integratie. Aan het begin van het verslag heb ik een klassendiagram van ITS gepresenteerd zoals het was voor mijn stage (beperkt tot de delen waar ik aan gewerkt heb). Ik heb ook een klassendiagram van ITS gemaakt van de situatie na mijn stage, met mijn uitbreidingen. Deze is te vinden in (Bijlage **L)**

9.2.4 Filtering algoritme applicatie

De filtering algoritme applicatie is een product dat als uitbreiding op de originele opdracht is gemaakt om de integratie in ITS voor meer toepassingen inzetbaar te maken. Dit product was qua programmeren het moeilijkste deel van de stage. Ik ben dan ook erg trots en tevreden met het product wat ik uiteindelijk heb opgeleverd. Ik heb een aantal technieken voor het eerst kunnen gebruiken (bijv. singleton en Big O,) en heb ook een aantal nieuwe technieken geleerd (bijv. Dictionary, generics, conditioneel compileren en het maken van appsettings). Dit betekent niet dat het product perfect is. Doordat de stage tegen het einde aanliep heb ik niet de tijd gehad om de applicatie zo grondig te testen als ik graag gewild

had zoals toegelicht in hoofdstuk 8. Onderstaand klassendiagram geeft weer hoe de filter applicatie in elkaar zit:



Figuur 32: Klassendiagram van de filter applicatie.

De applicatie is een C# console application. Wanneer deze op wordt gestart begint de applicatie dus bij de “Program” klasse. Deze bouwt de configuratie zoals eerder toegelicht en bezit een instance van de “TcpClient” klasse en de “MessageProcessor” klasse. De TcpClient is waar de berichten van de gateway binnen komen. De gateway is in dit geval dus de TCP server. Als de StartClientAsync() methode aangeroepen wordt begint de TcpClient met het opvragen van de data. De “messageProcessor” klasse is waar een groot deel van de logica plaatsvindt. Vanuit hier worden de berichten naar filters of de HTTP client gestuurd. De “ItsHttpClient” is zoals eerder toegelicht een singleton. Deze wordt in de “SendItsHttpClient” gebruikt om berichten naar ITS te sturen. De “Filter” klasse is mijn implementatie van het eerder toegelichte J.P.G. van Brakel algoritme. Een sequentiediagram van de werking van deze applicatie is te vinden in bijlage **G**.

In de “messageProcessor” klasse wordt zoals eerder uitgelegd een filter opgeslagen voor iedere sensor uit ieder sensor apparaat (bijvoorbeeld X, Y en Z as). Dit wordt gedaan door middel van een Dictionary. Een Dictionary is een C# collection met een key, value structuur. In dit geval is de key een string. Deze string is de samenvoeging van het “tagId” van de

sensor waar de meting vandaan komt en de naam van de meting: “measurementName” (bijvoorbeeld “E7DBE2F723F3AccX”). De value is het Filter object.

In de “Filter” klasse bestaat de “ShiftDataSet(lag)” methode. Binnen deze methode wordt het updaten van de dataset waarop het algoritme het gemiddelde en de standaarddeviatie berekend afgehandeld. Dit gebeurt iedere keer dat er een nieuwe meting in het Filter komt. Deze dataset is een array met een vaste lengte. De oudste waarde in het array wordt eruit gehaald, waarna een nieuwe waarde aan het eind van het array moet worden toegevoegd. Je zou in de eerste instantie kunnen denken dat een Queue hier een betere oplossing voor zou zijn, omdat een Queue dat als werkingsprincipe heeft. Dit was echter geen optie omdat de lengte van de dataset van tevoren bepaald moet worden door middel van de ingestelde Lag. Ook zou er dan door de hele queue ge-iterate moeten worden bij een ander stuk code in het algoritme, omdat een Queue een soort linked list is en niet met indexes werkt. Als oplossing heb ik daarom voor de “Array.Copy(sourceArray, sourceIndex, destinationArray, destinationIndex, length)” methode op de dataset gekozen met als parameters (dataSet, 1, dataSet, 0, dataSet.Length - 1).

“Array.Copy()” was niet de enige optie hiervoor. Nadat ik echter de tijdscomplexiteit (Big-O) van deze methoden vergeleken had, bleek “Array.Copy()” de beste optie met een big O van O(n) waar n de lengte van het array is[11]. Deze Big-O notatie is een manier om te bepalen hoelang een bepaalde handeling duurt. Met deze notatie kun je bijvoorbeeld meerdere methoden vergelijken om de efficiëntste te achterhalen, zoals ik in bovengenoemde situatie gedaan heb.

9.3 Eigen ervaring

Ik heb het tijdens mijn stage erg naar mijn zin gehad bij DWG. Ik vond het een leuke en interessante opdracht en de collega's waren erg fijn. DWG was ook zeer tevreden met het resultaat van de stage wat mij erg blij maakte.

Doordat er geen stages in cyber security te vinden waren in verband met de coronacrisis, ben ik een software engineering afstudeerstage gaan doen zonder het laatste semester software engineering gevolgd te hebben. Hierdoor was ik niet zeker of ik voldoende kennis zou hebben om de opdracht zo goed mogelijk uit te kunnen voeren. Dit is echter geen probleem gebleken.

Ik heb erg veel nieuwe kennis opgedaan tijdens de stage. Mijn begeleider Niels heeft me een aantal nieuwe technieken kunnen leren en ik heb zelf ook nieuwe technieken ontdekt tijdens het op het internet zoeken naar oplossingen voor problemen. Ik kan met zekerheid zeggen dat ik een verder gevorderde software ontwikkelaar ben dan voor de stage.

Tijdens de stage heb ik gemerkt dat mijn idee van een afstudeerstage niet helemaal overeenkwam met die van school. Ik was onder de impressie dat ik mij zou moeten aanpassen aan de werkwijzen van het stagebedrijf. Tijdens de feedbackmomenten kwam ik er echter achter dat school meer methoden wil zien zoals ze op de opleiding worden aangeleerd. Een voorbeeld hiervan is het maken van UML-diagrammen voor ontwerpen. Dit wordt eigenlijk niet gedaan binnen DWG, maar werd wel verwacht vanuit school. Ik vind het jammer dat ik daar niet eerder achter was gekomen.

9.4 Aanbevelingen/doorontwikkelingen DWG

In het testplan waren een integratietest en een systeemtest beschreven voor de integratie van Bluetooth Low Energy in ITS en de filter applicatie. Ik ben hier door gebrek aan tijd echter niet meer aan toegekomen. Mijn aanbeveling aan DWG is om dat alsnog te doen voordat het product door klanten in gebruik genomen wordt. Ook vind ik dat de filter applicatie nog uitgebreider getest moet worden. Er zou een code coverage van ongeveer 80% (de standaard binnen DWG) behaald moeten worden binnen de complete filter applicatie. Door de beperkte tijd heb ik alleen de kans gekregen om het algoritme zelf (de Filter klasse unit test) met een code coverage van 100% te testen. Dit zorgde maar voor een totale code coverage van 23.5% procent zoals te zien is in hoofdstuk 8.1. Deze aanbevelingen heb ik aan DWG gedaan tijdens mijn eindpresentatie voor mijn collega's.

10. Beroepstaken

10.1 A1: Analyseren probleemdomein & opstellen probleemstelling

Het analyseren van een probleemdomein en het opstellen van een probleemstelling daarover is een beroepstaak die voorafgaand en tijdens aan de stage naar voren is gekomen. Voor het afstudeerplan moest ik een probleemstelling opstellen. De informatie die ik nodig had om dit te doen heb ik verkregen uit mail- en telefonisch contact met DWG voorafgaand aan het begin van de stage. Ook heb ik hiervoor de website van DWG geraadpleegd.

Tijdens de stage is deze beroepstaak naar voren gekomen bij het ontwerpen van de filter applicatie (algoritme). Daarvoor moest ik eerst het probleemdomein eigen maken om te weten wat er ontworpen moest worden. Toen het probleemdomein duidelijk was, heb ik hier een probleemstelling voor opgesteld. De beschrijving van dit probleemdomein en de probleemstelling zijn te vinden in het ontwerpdocument van de filter applicatie (bijlage **G**)

10.2 A3: Informatie beoordelen, waarderen & prioriteiten

Het beoordelen, waarderen en prioriteren van informatie is vooral aan bod gekomen tijdens het uitvoeren van het onderzoek naar draadloze communicatieprotocollen (hoofdstuk 4.1). Het enige draadloze communicatie protocol waar ik voorafgaand aan dit onderzoek mee in aanraking was gekomen was LoRa. Alle te onderzoeken protocollen en de technologieën daaromheen waren dus nieuw.

Er was een gigantische hoeveelheid informatie te vinden over alle verschillende protocollen dus het was erg belangrijk om alleen de relevante informatie hieruit te halen en te verwerken in het onderzoeksverslag.

De eerste stap was het samenstellen van een lijst van protocollen die ik in het onderzoek wilde opnemen. Dit had ik gedaan aan de hand van een whitepaper over de meest gebruikte draadloze communicatieprotocollen. Hierdoor kon ik voorkomen dat ik veel tijd zou investeren in het onderzoeken van een protocol dat nog in een te vroeg ontwikkelstadium verkeert om daadwerkelijk te kunnen gebruiken bij klanten van DWG. Vervolgens ben ik per protocol op zoek gegaan naar zo veel mogelijk informatie.

Ik beoordeelde informatie op basis van bijvoorbeeld geloofwaardigheid. Als de ontwikkelaar van een protocol allerlei grandioze beweringen deed over bijvoorbeeld het bereik of de batterijduur van apparaten zette ik hier mijn twijfels bij. Als een onpartijdige bron met cijfers kwam uit een door henzelf in de praktijk uitgevoerde test of onderzoek hechtte ik daar meer waarde aan. Ook was het belangrijk om te letten op hoe oud de informatie was. Doordat de meeste draadloze communicatieprotocollen nog in de kinderschoenen staan veranderen ze veel bij iedere nieuwe versie. Informatie of statistieken die gaan om verouderde versies van een protocol zijn daardoor niet of minder relevant dan recentere informatie.

Om het overzicht te behouden en de grote hoeveelheid data te kunnen waarderen en prioriteren heb ik per protocol alle gevonden informatie op een kladblok geschreven. Vervolgens ben ik punten gaan wegstrepen die ik niet relevant genoeg vond voor het onderzoek. Op deze manier hield ik alleen relevante informatie over. Met deze informatie ben ik vervolgens het onderzoeksrapport gaan schrijven.

10.3 C1: Ontwerpen software

Om een duidelijk beeld te krijgen van de architectuur van de door mij ontwikkelde software en de bestaande software waar ik mee in aanraking ben gekomen, heb ik een aantal UML-diagrammen opgesteld. Het eerste voorbeeld hiervan is te zien in hoofdstuk 3.3. Daar heb ik een package diagram en een klassendiagram gemaakt om uit te beelden hoe ITS in elkaar steekt (figuur 2 & bijlage **A**). In deze situatie heb ik dus bestaande software in een diagram verwerkt om een duidelijk beeld te krijgen van de verschillende delen hierin. Aan de hand van dit diagram kon ik dus bepalen waar ik bepaalde stukken code/functionaliteit moest ontwikkelen.

Ook heb ik UML-diagrammen gemaakt om software te ontwerpen. In hoofdstuk 6.2 heb ik een klassendiagram geproduceerd die uitbeeldt hoe de constructie van measurements in elkaar zit (figuur 15). Deze constructie heb ik vervolgens gebruikt bij het realiseren van het HTTP Proof of Concept en ook bij de integratie van Bluetooth Low Energy in ITS. Deze manier van doen was nodig om het Proof of Concept zo generiek als mogelijk te maken.

Om af te beelden hoe de Bluetooth Low Energy sensoren, gateways en de ITS applicatie met elkaar in verbinding zouden gaan staan heb ik in hoofdstuk 7.2 een deployment diagram met componenten gemaakt (figuur 17). Dit diagram is vervolgens gebruikt bij de integratie in ITS. In datzelfde hoofdstuk heb ik een sequentiediagram gebruikt om af te beelden hoe het proces van het ontvangen en opslaan van Bluetooth Low Energy berichten in ITS in werking zou gaan (Bijlage **F**). Dit diagram is net als het deployment diagram gebruikt bij de integratie binnen ITS.

Voorafgaand aan het ontwikkelen van het filter algoritme heb ik een ontwerp document gemaakt. Hierin heb ik gebruik gemaakt van tekst, een activity diagram en een sequentiediagram om uit te beelden hoe de applicatie er uit zou moeten gaan zien.

Het maken van deze diagrammen heeft ervoor gezorgd dat ik een duidelijk beeld kon krijgen van hoe de software eruit zou moeten gaan zien, voordat ik deze ging schrijven. Tijdens het schrijven van de software dienden ze als houvast aan hoe de functionaliteit was uitgedacht. Deze manier van werken zorgt ervoor dat je goed nadenkt wat er gedaan moet worden voor je begint met code schrijven.

10.4 D1: Realiseren van software

Deze beroepstaak is gedurende de tweede en derde fasen aan bod gekomen. Het eerste voorbeeld hiervan is het maken van de proof of concept projecten. Dit waren kleine applicaties die ik moest schrijven. In hoofdstuk 6 geef ik een beschrijving van de hieraan

gerelateerde werkzaamheden. In hoofdstuk 6.1 laat ik met figuur 12 een stuk code zien uit het eerste (TCP) Proof of Concept. Nog zo'n stuk code is te vinden in hoofdstuk 6.2 (figuur 16). Hier word een stuk code toegelicht uit het tweede (HTTP) Proof of Concept.

Het grootste deel software realiseren heeft natuurlijk plaatsgevonden tijdens de integratie van Bluetooth Low Energy in ITS. Aan de hand van de eerder toegelichte ontwerpen heb ik ITS uitgebreid om Bluetooth Low Energy berichten te kunnen ontvangen en in een database op te kunnen slaan. Hiervoor heb ik controllers geschreven, logica geschreven en de database uitgebreid. Ook heb ik aanpassingen gemaakt in de DataStorage package. In hoofdstuk 7.4 licht ik hiervan een klein stukje code toe (figuur 21).

Ook heb ik een complex algoritme geschreven waarmee interessante data wordt gefilterd uit een stroom van een grote hoeveelheid data. Dit wordt in hoofdstuk 7.5.1 en hoofdstuk 9.2.4 toegelicht. Dit bestond uit het schrijven van code in C# aan de hand van eerder gemaakt ontwerpen. Het realiseren van software is waar verreweg de meeste tijd aan op is gegaan tijdens de stage.

10.5 Gc: Kritisch, onderzoekend en methodisch werken

De beroepstaak Kritisch, onderzoekend en methodisch werken is gedurende de gehele stage aan bod gekomen. Ik heb gebruik gemaakt van verschillende methoden tijdens de stage. Als eerst heb ik een descriptief/beschrijvend onderzoek uitgevoerd. Tijdens het maken van de Proof of Concepts heb ik gebruik gemaakt van een iteratieve ontwikkelmethode. Ik heb gebruik gemaakt van Scrum (agile) tijdens het integreren van Bluetooth Low Energy in ITS. Ik heb deze methodiek zo goed mogelijk volgens de regels toegepast zoals ik meerdere malen beschreven heb in hoofdstuk 7. Over de totale afstudeerperiode zou je kunnen zeggen dat ik volgens de watervalmethode heb gewerkt. Ik kon pas door naar de volgende fase wanneer de huidige compleet was afgerond (onderzoek => Poc => integratie in ITS).

Bij iedere stap of beslissing is er kritisch nagedacht over wat de beste oplossing zou kunnen zijn. Hierbij heb ik veel gebruik gemaakt van de kennis en expertise van mijn collega's. In dit soort situaties vonden vaak meetings plaats met collega's die zich bezighielden met de onderwerpen in kwestie. Met het advies wat ik uit deze meetings kreeg en informatie die ik van het internet verkreeg kon ik goed geïnformeerde beslissingen maken. Een voorbeeld hiervan is het niet maken van een MQTT Proof of Concept. Er was bepaald dat het maken van een Proof of Concept hiervan niet genoeg toegevoegde waarde zou bieden voor de tijd dat het zou kosten.

Een ander goed voorbeeld van deze beroepstaak wordt toegelicht in hoofdstuk 9.2.4. Hier leg ik uit dat ik een aantal opties had om een probleem op te lossen (Array shiften). Ik heb toen door middel van de Big-O notatie bepaald welke van de opties het efficiëntst zou zijn.

10.6 Gf: Leren leren: voorbereiden op volgenden studiefase en beroep

Doordat er geen stages in cyber security te vinden waren in verband met de coronacrisis, ben ik een software engineering afstudeerstage gaan doen zonder het laatste semester software engineering gevolgd te hebben. Dit betekende dat ik een hoop te leren had tijdens de stage. Gelukkig waren mijn collega's altijd bereid om dingen aan mij uit te leggen als ik ergens vragen over had. Ook heb ik veel gebruik gemaakt van het internet om informatie op te zoeken. Een concreet voorbeeld hiervan is de generic function die ik heb beschreven in hoofdstuk 7.3. Ik had dit eerst geschreven als twee functies, maar toen Niels mijn pull request hiervoor controleerde zei hij dat dit op een betere manier kon. Hij heeft me uitgelegd hoe generic functions werken waarna ik het op het internet heb opgezocht om de techniek eigen te maken. Ik heb deze techniek vervolgens op meerdere plekken in de code gebruikt.

Een van de grootste leerpunten van deze stage was simpelweg het functioneren binnen een IT ontwikkelteam. Ondanks de beperkingen die de pandemie met zich meebracht is dit naar mijn idee goed gelukt. Het communiceren met collega's en product owners zijn zaken die in mijn vorige stage minder naar voren zijn gekomen omdat ik daar geen deel uitmaakte van een team. Dit is een zeer nuttige les voor mijn toekomstige carrière als startende IT professional en ben daarom dus erg blij dat dit aspect meer naar voren gekomen is tijdens deze stage.

Het ontwikkelen van het filter algoritme is nog een goed voorbeeld van leren leren. Van te voren had ik geen idee hoe ik zo iets zou moeten maken. Ook wist ik niet hoe ik zo'n situatie aan moest pakken. Door mij aan het proces van requirements opstellen, ontwerpen, ontwikkelen en testen heb gehouden heb ik het algoritme succesvol kunnen ontwikkelen. Dit is een waardevolle les voor als ik later in een vergelijkbare situatie zit.

Bronnen

[1] ChirpStack open-source LoRaWAN® Network Server. (z.d.). ChirpStack. https://www.chirpstack.io/?__hstc=217413636.daf60cefb2c102192a5e94bba4794f7f.1619779782522.1619779782522.1619779782522.1&__hssc=217413636.1.1619779782523&__hsfp=673583646&hsCtaTracking=69c3740c-13ff-414c-9d59-2890afe035e2%7Cfbf459a9-49d9-4502-b972-1d47a60f8f59

[2] Rodela, J. (2021, 3 januari). Your Complete Guide to Proof of Concept. The Blueprint. https://www.fool.com/the-blueprint/proof-of-concept/

[3] Schwaber, K., & Sutherland, J. (2017, november). The Scrum Guide. scrumguides.org. https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[4] Azure DevOps Server. (z.d.). Microsoft Azure. https://azure.microsoft.com/nl-nl/services/devops/server/

[5] SonarQube Documentation | SonarQube Docs. (z.d.). SonarQube. https://docs.sonarqube.org/latest/

[6] Geurts, A. (z.d.) IoT connectiviteit: 10 veelgebruikte draadloze communicatie protocollen. IoT Academy. https://www.iotacademy.nl/whitepaper

[7] How Secure Is the BLE Communication Standard? (2020, 30 oktober). Lemberg Solutions. https://lembergsolutions.com/blog/how-secure-ble-communication-standard#:%7E:text=Bluetooth%20Low%20Energy%20is%20a,with%20an%20appropriate%20pairing%20method.

[8] Hamilton, E. (2018, 27 december). What is Edge Computing: The Network Edge Explained. Cloudwards. https://www.cloudwards.net/what-is-edge-computing/

[9] Brakel, J.P.G. van (2014). "Robust peak detection algorithm using z-scores". Stack Overflow. Available at: https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data/22640362#22640362 (version: 2020-11-08).

[10] Gunnar Peipman. (z.d.). Code coverage reports for ASP.NET Core. Gunnar Peipman - Programming Blog. Geraadpleegd op 1 juni 2021, van https://gunnarpeipman.com/aspnet-core-code-coverage/

[11] Array.Copy Method (System). (z.d.). Microsoft Docs. Geraadpleegd op 4 juni 2021, van https://docs.microsoft.com/en-us/dotnet/api/system.array.copy?view=net-5.0

Bijlagen

A. Klassendiagram ITS uitgangssituatie

Te vinden in apart document: *Klassendiagram ITS uitgangssituatie*

B. Onderzoeksrapport draadloze communicatie protocollen

Te vinden in apart document: *Onderzoeksrapport Draadloze Communicatie Protocollen*

C. Presentatie onderzoek draadloze communicatie protocollen

Te vinden in apart document: *Presentatie onderzoek draadloze communicatie protocollen*

D. BLE TCP Client schermopname

Te vinden in apart document: *BLE TCP Client Screen Record*

E. BLE PoC API schermopname

Te vinden in apart document: *BLE PoC API Screen Record*

F. Sequentiediagram BLE messages ontvangen en opslaan in database

Te vinden in apart document: *Sequentiediagram Ble messages ontvangen en opslaan in database*

G. Filter applicatie ontwerpdocument

Te vinden in apart document: *Filtering Algorithm ontwerpdocument*

H. Test plan integratie Bluetooth Low Energy in ITS

Te vinden in apart document: *Test plan integratie Bluetooth Low Energy in ITS*

I. Testrapport integratie Bluetooth Low Energy in ITS

Te vinden in apart document: *Testrapport integratie Bluetooth Low Energy in ITS*

J. Handleiding filter algoritme

Te vinden in apart document: *Handleiding filter algoritme*

K. Dataset voor filter unit test

Te vinden in apart document: *Dataset voor filter unit test*

L. Klassendiagram ITS resultaat

Te vinden in apart document: *Filtering Algorithm ontwerpdocument*