CARTRACKER MET OBD ONDERSTEUNING

AFSTUDEERVERSLAG

Student : Robbert Eigenbrood
Studentnummer : 12035769
Onderwijsinstelling : Haagse Hogeschool
Opleiding : Technische Informatica
Afstudeerperiode : 3 februari 2016 t/m 3 juni 2016
Begeleider : Dhr. J.D. Schagen
Tweede examinator : Dhr. J.P.M. de Vreught

Bedrijf : Decos Cartracker
Bedrijfsmentor : Alain van Hanegem

Datum : 3 juni 2016
Versie : 1.0

Referaat

Titel

Robbert Eigenbrood, Cartracker met OBD ondersteuning, afstudeerverslag ter afronding van de bachelor Technische Informatica aan de Haagse Hogeschool, 2016.

Samenvatting

Met behulp van OBD is het mogelijk diagnostische data uit de auto te lezen. Hiermee kunnen uitstoot gerelateerde gegevens uit een auto gelezen worden. Het is verplicht om via de OBD stekker in de auto deze data uit te kunnen lezen. Via de OBD stekker is er vaak ook te filteren op niet gestandaardiseerde CAN berichten. Deze berichten bevatten informatie over vrijwel alle sensoren in een auto. In dit afstudeerverslag wordt een proof of concept gepresenteerd van de Cartracker, een track & trace black box, met OBD ondersteuning op het gebied van opvragen en verwerken van OBD en CAN bus berichten.

Descriptoren

OBD, On Board Diagnostics, STN1170 Cartracker, CAN

Voorwoord

Ter afsluiting van de studie Technische Informatica aan de Haagse Hogeschool heb ik een afstudeeropdracht uitgevoerd bij Decos Mobile Solutions.
Dit afstudeerverslag beschrijft het doorgelopen proces en de keuzes die zijn gemaakt gedurende de afstudeerperiode.

Tijdens mijn studie ben ik meer in aanraking gekomen met embedded systems. Deze mix van elektronica en 'low level' programmeren sprak mij ook enorm aan. Tijdens het oriënteren naar een afstudeerplek ben ik opzoek gegaan naar een afstudeeropdracht in deze richting. Deze opdracht heb ik dan ook mogen vinden bij Decos Mobile Solutions.

Ik wil graag mijn collega's bij Decos Mobile Solutions bedanken voor de gezellige en leerzame tijd. Tot slot wil ik Martine Bakker bedanken voor het aanzetten tot de keuze van de studie Technische Informatica.

Robbert Eigenbrood

Noordwijk, 3 juni 2016

INHOUDSOPGAVE

1 INLEIDING 1

2 ORGANISATIEBESCHRIJVING 3

3 DEFINITIEFASE 5
3.1 Probleemstelling 5
3.2 Doelstelling 6
3.3 Resultaat 6
3.4 Methode 6

4 INCEPTIONFASE 7
4.1 Productdefinitie 7
4.2 Use-Cases 9

5 ELABORATIONFASE 13
5.1 Analyse 13
5.2 Prototype 24
5.3 Architectuur 31

6 CONSTRUCTIONFASE 35
6.1 Iteratie 1 De eerste communicatie 35
6.2 Iteratie 2 Nieuw communicatie protocol 42
6.3 Iteratie 3 OBD berichten 47
6.4 Iteratie 4 CAN bus filtering 55

7 CONCLUSIE EN AANBEVELINGEN 57
7.1 Conclusie 57
7.2 Aanbevelingen 57

8 PROJECT EVALUATIE 61

9 AFKORTINGEN 63

BIBLIOGRAFIE 65

1 INLEIDING

Door middel van het On Board Diagnosics systeem, wat in iedere brandstof auto zit sinds 2001, is het mogelijk uitstoot gerelateerde informatie uit de auto te lezen(ISO 15031-1). Deze informatie kan erg interessant zijn voor bedrijven of particulieren die gericht zijn op zuinig rijden.

Decos Cartracker levert een track & trace oplossing voor het volgen van voertuigen. Voor het volgen van het voertuig wordt er een black box met gsm en gps in het voertuig gebouwd. Deze black box wordt de Cartracker genoemd. De Cartracker is een embedded system ontwikkeld bij Decos Mobile Solutions. De Cartracker houdt aan de hand van gps de locatie van de auto bij. Deze informatie is via een applicatie op de computer te bekijken en te gebruiken in bijvoorbeeld de administratie.

Aan de hand van het sleutelcontact krijgt de Cartracker een signaal van het voertuig of deze actief is of niet.
Op dit moment is het uitlezen van het sleutelcontact de enige informatie die direct uit het voertuig gelezen kan worden maar OBD zou hier een waardevolle toevoeging voor zijn. Het probleem is dat deze koppeling tussen de Cartracker en OBD nog niet bestaat.

Het onderzoek wat uitgevoerd is tijdens de afstudeerperiode is gericht om de Cartracker en OBD samen te laten werken.
De Cartracker hardware en software wordt op deze manier continu door ontwikkeld. Voor dit onderzoek is er gebruik gemaakt van de Cartracker 3 hardware versie 4 die begin vorig jaar op de markt is gekomen.

De Cartracker kan niet direct op de auto aangesloten worden. Om dit mogelijk te maken wordt de STN1170 gebruikt. Dit is een OBD naar UART-vertaler wat als vertaler tussen de Cartracker en de auto komt. De STN1170 kan OBD berichten opvragen en filteren uit de auto. Ook biedt de chip de mogelijkheid om niet gestandaardiseerde CAN berichten uit te lezen.

Figuur 1 Overzicht analysegebied

Om al deze informatie te achterhalen is er eerst een analyse uitgevoerd op een drietal gebieden: Cartracker, STN1170 en OBD. In Figuur 1 zijn de drie verschillende analyseonderwerpen te zien.
Na de analyse is er een prototype applicatie ontwikkeld om meer informatie te achterhalen van de OBD in een auto en vooral voor de niet gestandaardiseerde CAN berichten. Deze informatie is tijdens de bouw van het proof of concept gebruikt.

Aan het begin van de realisatie bleek er een fout in de hardware te zitten. Hierdoor is er gekozen om zelf een nieuwe interface te realiseren met de bestaande hardware die beschikbaar was. De resultaten uit het prototype zijn in de laatste iteratie gebruikt om CAN ondersteuning toe te voegen voor de Nissan Leaf.

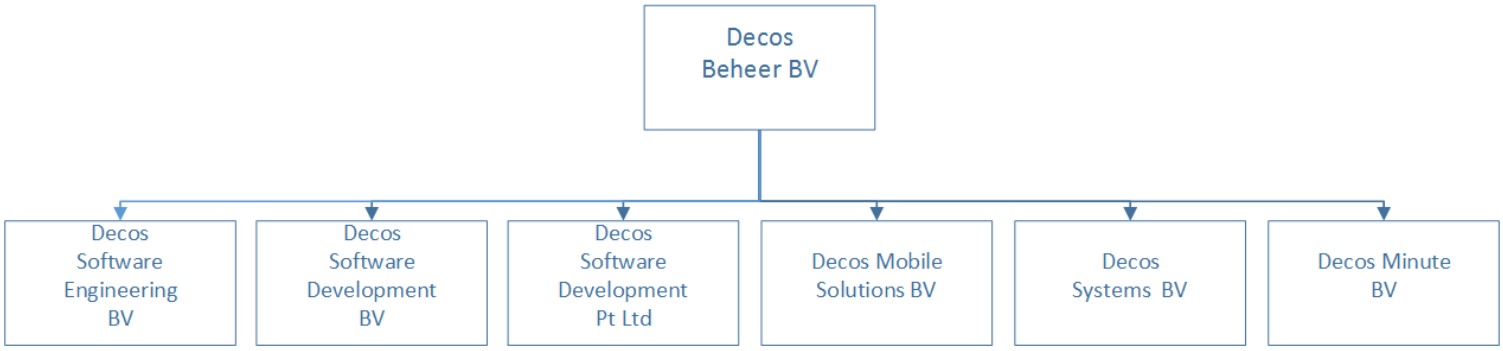
Het resultaat van het project is een proof of concept Cartracker 3 met OBD ondersteuning.

2 ORGANISATIEBESCHRIJVING

Decos is een innovatief en dynamisch bedrijf. Het bekendste statement van Decos is dat het gehele bedrijf papierloos is. In het bedrijf is buiten de post van andere geen papier op de werkvloer te vinden.

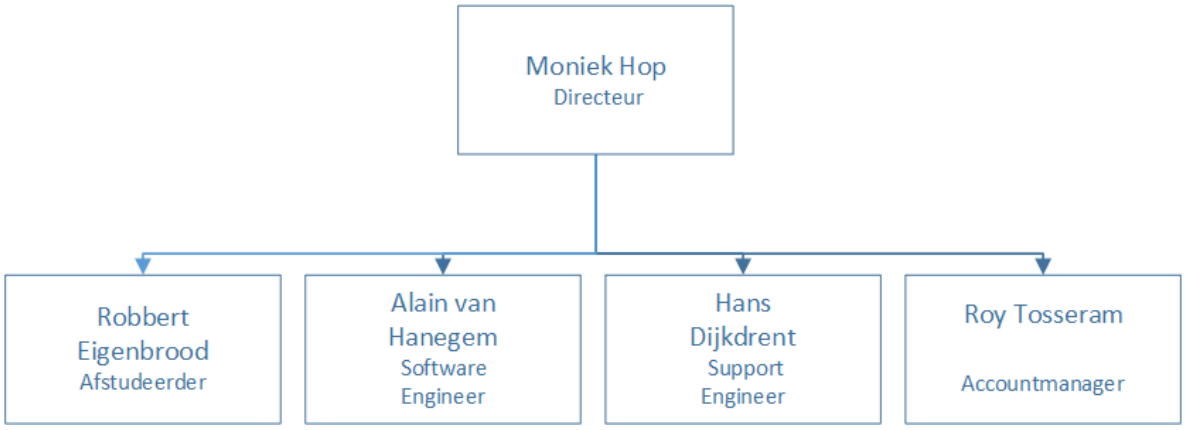
Bij Decos staat innovatie voorop, zo is het hoogstwaarschijnlijk dat je bij binnenkomst niet door een receptionist maar door de welkomstrobot wordt ontvangen. Het hoofdkantoor van Decos staat in Noordwijk maar grotendeels van het ontwikkelteam bevindt zich in India. Bij elkaar zijn er ruim 200 mensen werkzaam bij Decos.

Decos biedt als bedrijf verschillende oplossingen op het gebied van efficiëntie en duurzaamheid. De verschillende oplossing die door Decos aangeboden worden zijn: Digitaal werken, Samenwerking en Mobiliteit. Binnen deze oplossing vallen de verschillende producten die Decos aan hun klanten aan biedt. Zie het onderstaande organigram voor alle ondernemingen die binnen Decos Beheer BV vallen.



De afstudeeropdracht heeft plaats gevonden bij Decos Cartracker, wat onderdeel is van Decos Mobile Solutions. Binnen Decos Mobile Solutions worden twee producten ontwikkeld. Flo, een app die feedback geeft over het rijgedrag van een bestuurder tijdens een rit en de Cartracker.

Het team van Cartracker in Nederland bestaat uit vier personen. In de onderstaande organigram zijn de verschillende personen inclusief functie te zien. Als extra ben ik als afstudeerder ook in het organogram geplaatst om de plek van de afstudeerder aan te duiden.



3 DEFINITIEFASE

Voorafgaand aan de start van de afstudeerperiode is er door middel van een afstudeerplan een aantal onderdelen vastgesteld. In de eerste fase van dit project zijn deze onderdelen verder uitgewerkt en geanalyseerd. Zie bijlage A voor het volledige afstudeerplan

3.1 Probleemstelling

Vanuit de eerste gesprekken met de opdrachtgever is het probleem van de opdrachtgever vastgesteld. Het probleem is op de volgende manier gedefinieerd:

Het is onbekend hoe er door middel van de Cartracker in samenwerking met de STN1170 OBD berichten uit een personenauto gelezen en verwerkt kunnen worden door de Cartracker.

In de probleemstelling zijn een drietal problemen beschreven:

1 - Het is onbekend hoe de samenwerking tussen de Cartracker en de STN1170 werkt
2 - Het is onbekend hoe OBD berichten ingelezen kunnen worden
3 - Het is onbekend hoe de OBD berichten door de Cartracker verwerkt kunnen worden

Bij punt twee is het reeds bekend dat het uitlezen van een bericht door middel van de STN1170 gedaan kan worden. Welke handelingen hier voor nodig zijn, zijn onbekend.

In punt drie wordt het verwerken van berichten gezien als het decoderen en weergeven van de berichten aan de gebruiker.

Conclusie Probleemstelling

Alle problemen in de probleemstelling wijzen naar drie verschillende onderwerpen. De Cartracker, STN1170 en OBD waar op dit moment nog onvoldoende kennis van is.

Er is onvoldoende kennis aanwezig over de werking van de Cartracker
Er is onvoldoende kennis aanwezig over de werking van de STN1170
Er is onvoldoende kennis aanwezig over OBD

3.2 Doelstelling

De doelstelling is als volgt beschreven:

Het doel van de opdracht is inzicht geven op het verwerken van OBD berichten met de Cartracker, in samenwerking met de STN1170

Analyse

Met kijk op de conclusie van de probleemstelling is het benodigd kennis te vergaren over de benoemde onderwerpen. Om de doelstelling te behalen is het noodzakelijk deze kennis op te doen over de Cartracker, de STN1170 en OBD.

Software realiseren

Uit de probleemstelling is te halen dat de Cartracker nog geen communicatiemogelijkheden biedt met de STN1170. Ook is het nog niet mogelijk OBD berichten te verwerken met de Cartracker. Dit zal gerealiseerd moeten worden in het proof of concept.

Testen

Na iedere iteratie zal er een test uitgevoerd gaan worden waarmee de geïmplementeerde use case getest wordt. Aan het eind van het project moeten alle use cases getest zijn.

3.3 Resultaat

Het resultaat is een proof of concept van de Cartracker met ondersteuning op het gebied van inlezen, verwerken van OBD berichten met behulp van de STN1170. Onder het resultaat valt de documentatie en een proof of concept Cartracker met OBD ondersteuning.

3.4 Methode

Voorafgaand aan het project is er al besloten om RUP als ontwikkelmethode te gebruiken.

De keuze om RUP te gebruiken heeft te maken met de iteratieve manier van ontwikkelen. Hiermee kan de ontwikkeling in kleine onderdelen opgedeeld worden en kan er goed bijgestuurd worden waar nodig. De laatste fase, de transitie fase, zal niet worden uitgevoerd tijdens de afstudeerperiode. Omdat het resultaat een proof of concept van de Cartracker beschrijft, wordt deze niet in het huidige systeem geïmplementeerd, waar de transitie fase onder andere voor dient.

4 INCEPTIONFASE

De requirements uit de definitiefase zijn tijdens de inceptionfase opgenomen in het visiondocument. Het product van deze fase, het visiondocument, is te vinden in bijlage B. De volgende paragrafen vertellen de belangrijkste onderdelen van het visiondocument.

4.1 Productdefinitie

Om een beeld te geven op welke gebieden er tijdens de elaborationfase een analyse uitgevoerd zal worden, is er tijden de inceptionfase gekeken naar welke hoofdonderdelen het project bevat.
Naar aanleiding van de probleemstelling zijn de volgende drie onderdelen uitgelicht:

- Cartracker
- STN1170
- OBD

Aan de hand van de probleemstelling is de volgende globale tekening opgesteld:

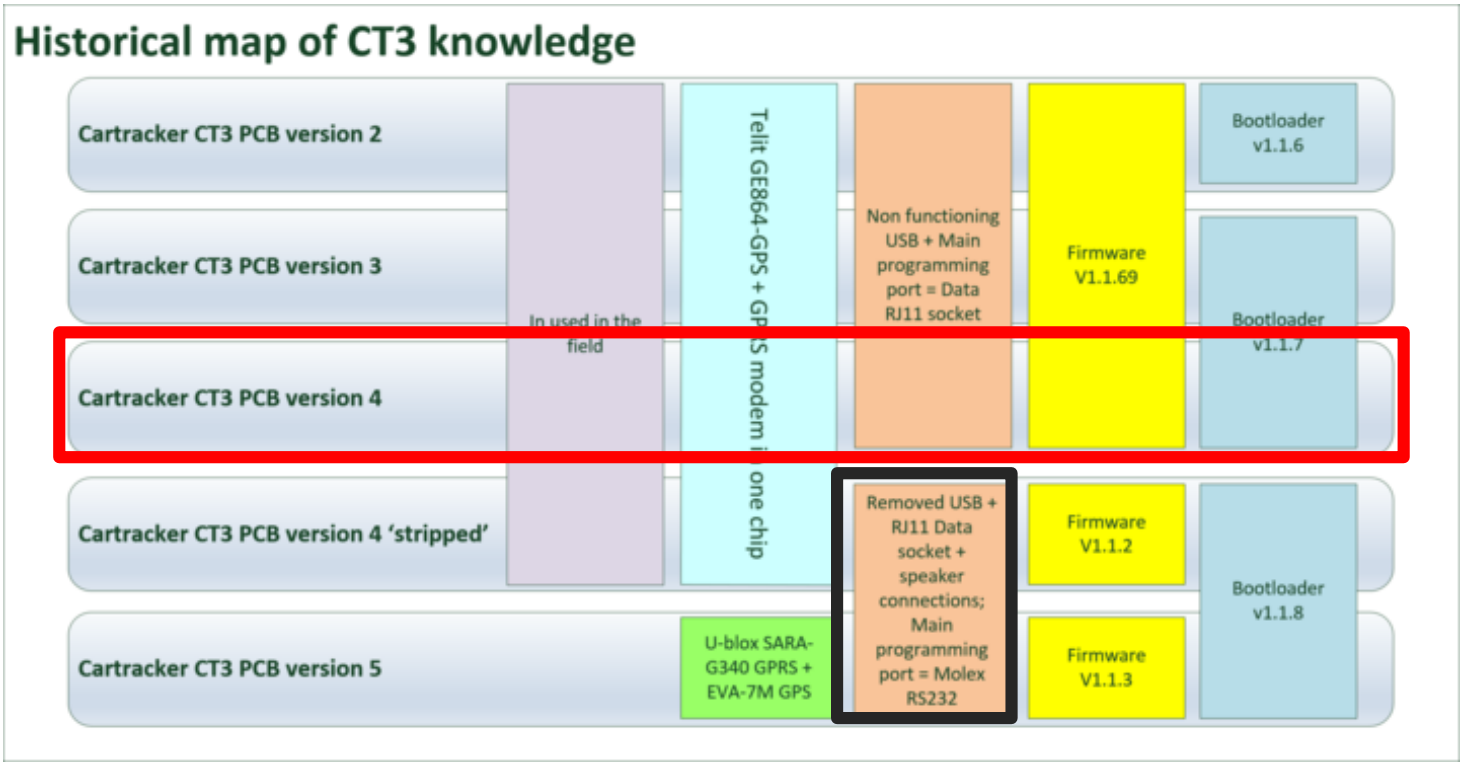
Figuur 2 Globale tekening Cartracker met OBD ondersteuning

In de tekening zijn de drie onderdelen, de Cartracker, STN1170 en OBD te zien. Er is communicatie tussen de Cartracker en de STN1170 en tussen de STN1170 en het OBD systeem in de auto.

Zoals in de inleiding te lezen is heeft de opdrachtgever specifiek aangegeven dat het project met de Cartracker 3 versie 4 uitgevoerd zal gaan worden. Om te weten waarom deze versie gebruikt gaat worden voor de OBD oplossing en of dit een onderdeel van de analyse moet zijn in de elaborationfase is hier tijdens de product definitie een vraag richting de opdrachtgever over gesteld.

Er is vanuit de opdrachtgever een historische map van de Cartracker 3 gegeven. In deze map zijn de huidige in het veld aanwezige Cartrackers te zien. Er is te zien dat de Cartracker 3 versie 4 niet de meest recente Cartracker is. De keuze om toch de Cartracker 3 versie 4 te gebruiken heeft er mee te maken dat er al een reeds ontwikkeld opzetboard is voor de Cartracker 3 versie 4 met de STN1170. Ontworpen voor het uitlezen van OBD berichten.

In Figuur 3 in het rode kader is de Cartracker 3 versie 4 te zien. In het zwarte kader is te lezen dat er een aantal onderdelen, onder andere de headers voor de opzetprint, is verwijderd. Hierdoor is het niet mogelijk om het opzetboard op de nieuwere Cartracker versies te gebruiken. Met deze uitleg is er besloten verder geen analyse uit te voeren naar de verschillende type Cartrackers maar alleen te focussen op de Cartracker 3 versie 4.



Figuur 3 Cartracker 3 historical map

4.2 Use-Cases

Tijdens de eerste gesprekken met de opdrachtgever is er gesproken over de behoeften van de opdrachtgever op gebied van hardware en software. De behoeften zijn vertaald naar een use case diagram en bijbehorende use case scenario's zijn uitgewerkt en besproken met de opdrachtgever. De volgende behoeften zijn voortgekomen uit deze gesprekken:

Tabel 1 Functionele behoeften

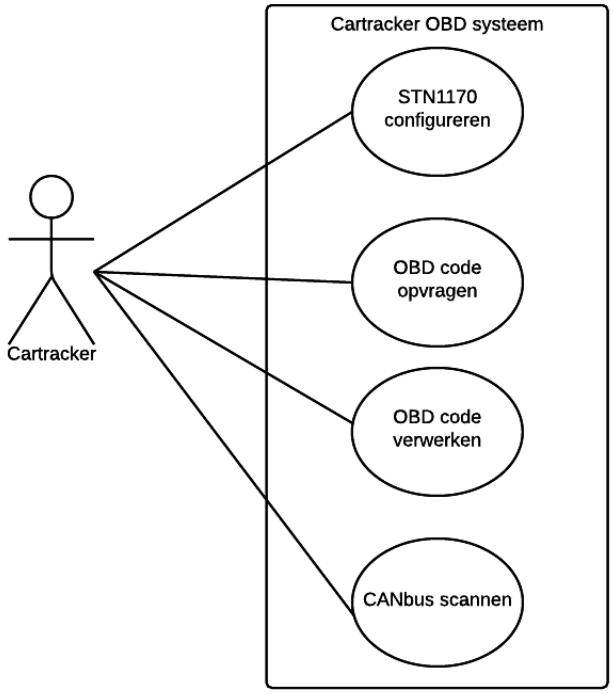
Nr.	Beschrijving
F1	De Cartracker moet de STN1170 kunnen configureren
F2	De Cartracker moet via het opzetboard OBD berichten kunnen opvragen
F3	De Cartracker moet verschillende OBD berichten kunnen opvragen
F4	De Cartracker moet OBD berichten kunnen verwerken
F5	De Cartracker moet verschillende CAN berichten kunnen opvragen
F6	De Cartracker moet verschillende CAN berichten kunnen verwerken

Tabel 2 Niet-functionele behoeften

Nr.	Beschrijving
Nf1	De OBD functionaliteit wordt geschreven voor de Cartracker 3 versie 4
Nf2	Atmel Studio 7.0 wordt gebruikt voor het ontwikkelen van de code voor de Cartracker
Nf3	Iedere 5 seconden moet een OBD bericht uitgelezen kunnen worden
Nf4	De CAN bus functionaliteit zal alleen gelden voor de Nissan Leaf
Nf5	De Cartracker moet op iedere auto met OBD ondersteuning aangesloten kunnen worden
Nf6	De Cartracker moet na het aansluiten op een spanningsbron direct de OBD verwerking starten.
Nf7	De Cartracker moet via het opzetboard met de auto verbonden worden

De opdrachtgever heeft aangegeven OBD berichten uit een auto te willen lezen. Hiervoor zal de Cartracker gebruik worden in combinatie met de STN1170. Er zal code geschreven gaan worden voor de Cartracker om de STN1170 aan te sturen en de ingekomen OBD berichten te verwerken.

Omdat het gehele systeem geen input vereist van een gebruiker, buiten het voorzien van een voedingsspanning was het de vraag wat de actor van het systeem ging worden. In overleg met de opdrachtgever is de Cartracker als actor gekozen.
De Cartracker wordt gezien als actor op de rest van het systeem. Uit de opgeschreven behoeften is een use case diagram gemaakt. Het use case diagram is te zien in Figuur 4.



Figuur 4 Inception fase Use Case

De use case toont de vier verschillende use cases met betrekking op de OBD functionaliteit. In de vier use cases worden alle behoeften van de opdrachtgever uitgevoerd.
Hieronder is een van de use case uitwerkingen te zien zoals te vinden in het visiondocument.

Tabel 3 use-case uitwerking STN configureren

Naam	**STN1170 configureren**
Actor	Cartracker
Aannamen	De STN1170 is nog niet geconfigureerd
Beschrijving	De STN1170 wordt geconfigureerd 1- Juiste commando's worden vanaf de Cartracker naar de STN1170 gestuurd 2- De STN1170 geeft een bevestiging terug aan de Cartracker
Uitzonderingen	
Resultaat	De STN1170 is correct geconfigureerd

5 ELABORATIONFASE

Nadat het visiondocument was opgesteld in de inceptionfase is de nieuwe fase, de elaborationfase, gestart. In deze fase is de analyse uitgevoerd en is het architectuurdocument opgesteld. Zie bijlage C voor het analysedocument en bijlage D voor het architectuurdocument.

Voordat het eindproduct van de elaborationfase, het architectuurdocument, geschreven kon worden moest er eerst kennis opgedaan worden over de Cartracker, STN1170 en OBD zoals geconcludeerd uit de probleemstelling. Deze analyse is in drie verschillende fases uitgevoerd. De hardware analyse, de STN1170 analyse en OBD analyse.

De tekening opgesteld in de productdefinitie (Figuur 2) is gebruikt om het onderzoeksgebied tijdens de analyse weer te geven.

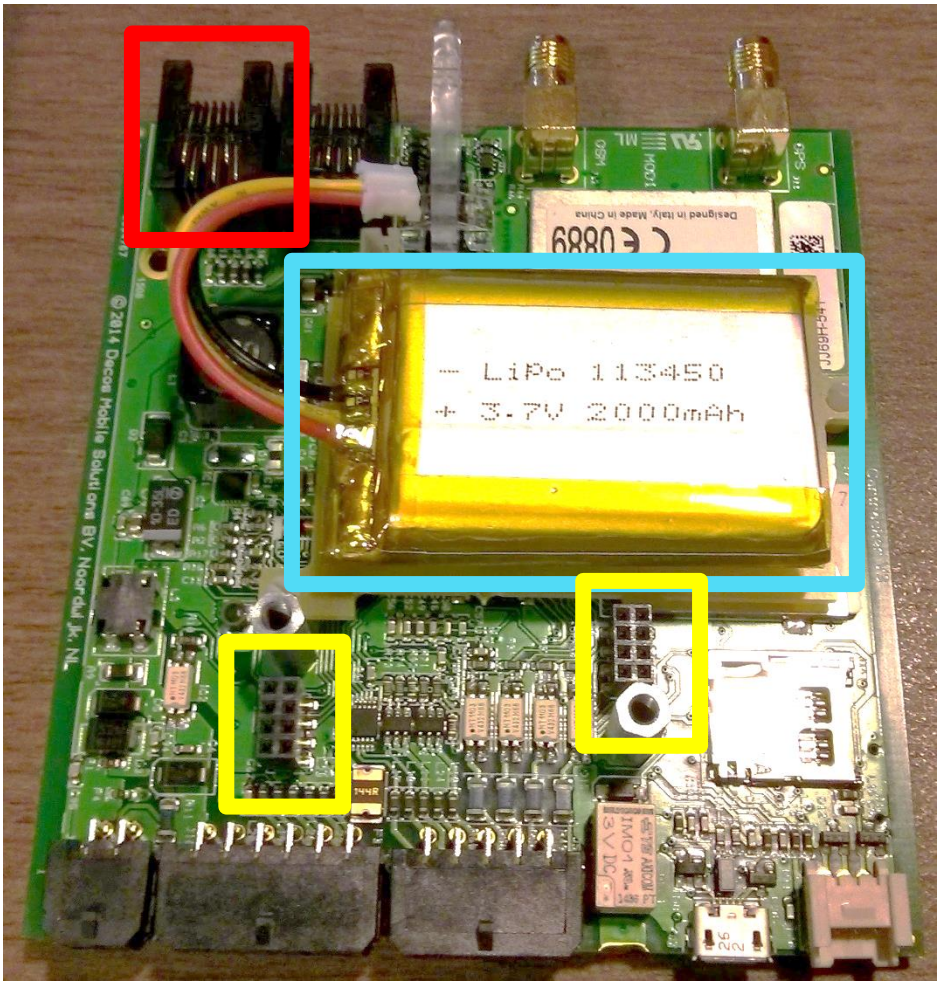
Om de analyse van het OBD protocol uit te voeren en informatie te verzamelen is er vanuit de opdrachtgever een voorstel geweest voor het opstellen van een prototype applicatie. Met deze applicatie zijn tijdens de elaborationfase gegevens uit de auto verzameld. Deze gegevens dienen onder andere als referenties tijdens de constructionfase.

5.1 Analyse

Tijdens het opstellen van de probleemstelling bleek al gauw dat de benodigde kennis op het gebied van de Cartracker, de STN1170 en OBD nog niet aanwezig is. Om voldoende informatie te verzamelen en het project op bachelor niveau uit te voeren is op ieder van deze drie gebieden een analyse uitgevoerd.

Cartracker Analyse

De eerste analyse is gericht op de Cartracker hardware. De hardware analyse bestond uit twee onderdelen. Het eerste deel is de Cartracker 3 en de tweede het STN1170 opzetboard. Beide delen maken onderdeel uit van het proof of concept. Deze analyse is uitgevoerd zodat er in de constructionfase bekend was welke mogelijkheden en beperkingen er met de hardware waren.



Figuur 5 Cartracker 3 Hardware

De Cartracker wordt aangestuurd door een 32-bit Atmel microcontroller. Op de microcontroller draait FreeRTOS, een RTOS speciaal ontwikkeld voor embedded systems(Barry).

De opdrachtgever had aangegeven het proof of concept in een los project te willen hebben. FreeRTOS hoefde hier niet voor gebruikt te worden. Wel is de werking van de huidige firmware geanalyseerd om te weten wat er nodig is als de projecten wel bij elkaar gevoegd gaan worden.

De Cartracker is voorzien van allerlei extensie onderdelen. In Figuur 5 is de Cartracker hardware te zien met diverse onderdelen. Degene die betrekking hebben op het project zijn de volgende:

- Accu (blauw)
  De accu wordt opgeladen zolang de Cartracker is aangesloten op een externe voeding, in de meeste gevallen zal dit de geschakelde voeding van de auto zijn. De accu kan tijdens het project gebruikt worden om de Cartracker te voeden zonder een externe voeding aan te hoeven sluiten. Dit draagt bij aan de mobiliteit tijdens het testen
- Extension headers (geel)
  Hierop kan nog een extra module op de Cartracker worden aangesloten. In dit geval wordt hier het STN1170 opzetboard op aangesloten
- RJ11 connector (rood)
  Communicatie via UART voor het programmeren van nieuwe firmware

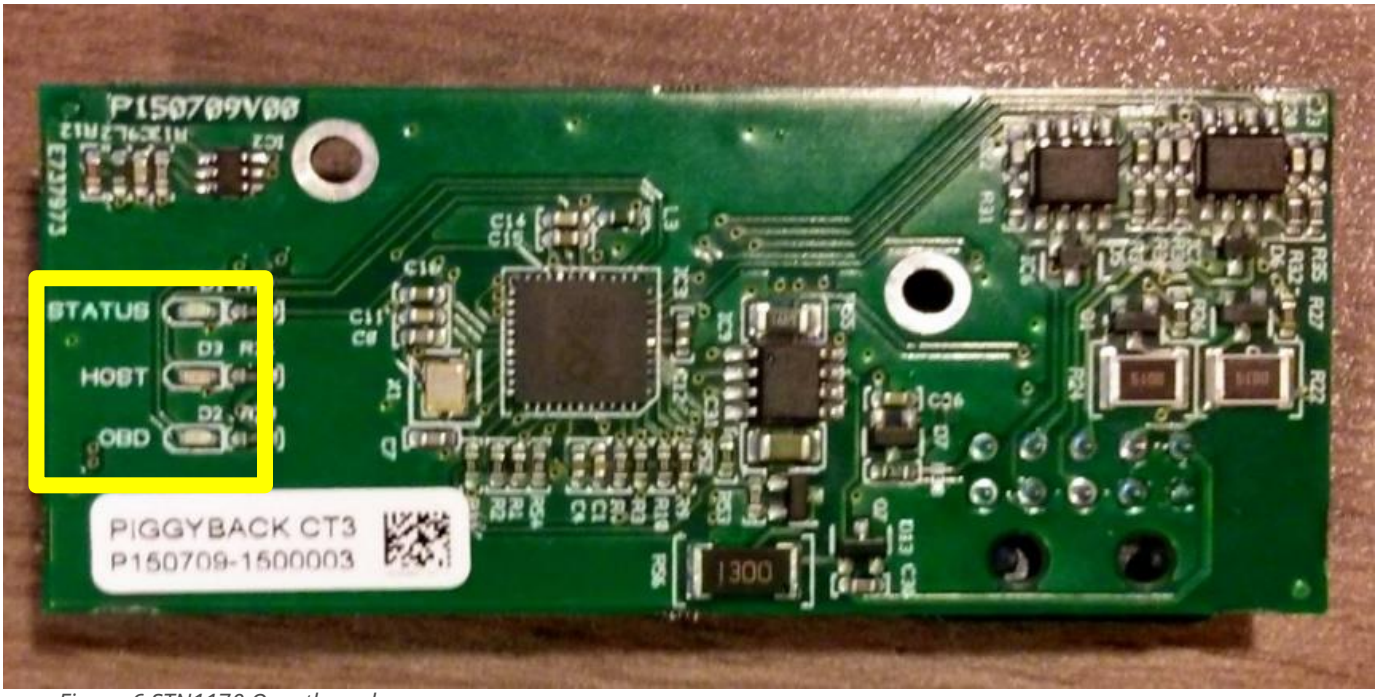
Via de extension headers is het mogelijk de STN1170 opzetboard op de Cartracker aan te sluiten en aan te sturen.

Opzetboard analyse

Voor de OBD ondersteuning is een opzetboard ontwikkeld die op de Cartracker aangesloten kan worden. Het opzetboard was al ontwikkeld voordat de afstudeeropdracht was begonnen.

Via het opzetboard kunnen OBD berichten uitgelezen worden. In Figuur 6 is het opzetboard te zien. In het gele kader zijn een drietal LEDs te zien. Deze LEDs geven visuele feedback aan de gebruiker over de werking van het board de STATUS LED brandt als de STN1170 voorzien wordt van spanning. De HOST LED knippert als er communicatie is van of naar de HOST (in dit geval de Cartracker). De OBD LED knippert als er communicatie is met het OBD systeem van de auto.

De STN1170 communiceert via UART. De Cartracker headers zitten verbonden aan de SPI pinnen van de microcontroller. Om de vertaalslag tussen SPI en UART te kunnen maken is er een SPI naar UART vertaler op het board aanwezig.



Figuur 6 STN1170 Opzetboard

STN1170 analyse

In de volgende paragraaf worden de STN1170 analyseresultaten beschreven.
Om uiteindelijk de OBD berichten uit de auto te kunnen lezen is het van belang om te weten hoe de communicatie brug tussen de Cartracker en het OBD systeem, in het geval van dit project de STN1170, werkt.

De STN1170 dient als communicatie chip tussen de Cartracker en de OBD in de auto. Deze chip filtert berichten van de CAN bus uit de auto en stuurt dit door over de UART.

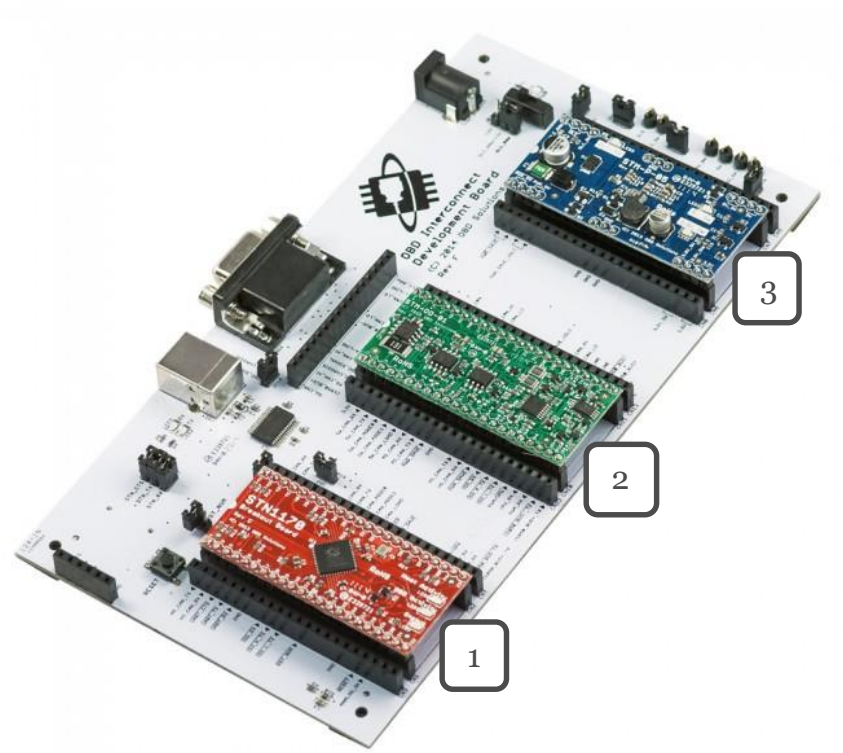
De STN1170 is gebaseerd op de ELM327 welke de de facto standaard is voor het uitlezen van OBD-II. De STN1170 kan geconfigureerd worden door middel van het versturen van AT of ST commando's. Hiermee kan bijvoorbeeld de baudrate of een filter instelling aangepast worden.
De STN1170 maakt gebruik van dezelfde AT commando set als de ELM327(OBD Solutions 23).
Scantool (de ontwikkelaar van de STN1170) heeft de chip uitgebreid met een aantal eigen ST commando's. Met de ST commando's kan onder andere de PowerSave functie aangestuurd worden, welke standaard niet op de ELM327 zit. Ook kan monitoring op specifieke CAN berichten toegepast worden. De laatste functie wordt gebruikt voor de use case CAN bus scannen.

Development Board

Om testen uit te kunnen voeren met de STN1170 zonder de Cartracker inclusief het opzetboard te hoeven gebruiken is er gebruik gemaakt van het STN1170 development board. Het development board is voorzien van een USB port om naar de pc te verbinden en een db9 port om via een db9 naar OBD stekker met de auto te verbinden. Op het development board zitten drie modules.

1. OBD interpreter module (STN1170)
2. OBD transreceiver module
3. Power module

Alle modules zijn voorzien van headers zodat er op alle pinnen gemeten kan worden mocht dit nodig zijn.
Op het ontwikkelde opzetboard voor de Cartracker zit de functionaliteit als van deze drie bordjes. Ook op het development board zitten de drie indicatie LEDs.



Figuur 7 STN1170 Development Board

OBD Analyse

Het laatste onderdeel van de analyse is het OBD protocol.
Een volledige analyse van het OBD protocol zou te groot zijn en te veel tijd kosten voor de afstudeeropdracht. Om een gerichte analyse uit te voeren is een analyse vraag opgesteld:

Welke mogelijkheden biedt het OBD protocol bij het uitlezen van data uit een auto?

Om deze vraag te beantwoorden zijn de volgende deelvragen opgesteld:

- *Op welke manier kan er met het OBD protocol gecommuniceerd worden?*
- *Hoe kan OBD data worden opgevraagd?*
- *Welke informatie is er via OBD uit de auto te lezen?*

Onderstaande paragraaf geeft de antwoorden weer van de verschillende deelvragen.

Op welke manier kan er met het OBD protocol gecommuniceerd worden?

De ISO 10531-1 verplicht alle Europese benzineauto's tot de implementatie van OBD-II vanaf 2001(ISO 15031-1). OBD werkt via de CAN bus in de auto. Via deze bus is het mogelijk om met de ECU te communiceren. De ECU verwerkt de data van de verschillende sensoren en stuurt hiermee de actuatoren in de auto aan.

Om de OBD informatie uit de auto te lezen zit er in iedere auto een OBD stekker. De OBD stekker is een verplicht onderdeel in de auto. Via deze stekker moet het mogelijk zijn om via het OBD protocol met de auto te communiceren. In **Error! Reference source not found.** is de OBD stekker in een auto te zien.

De OBD standaard beschrijft het gebruik van verschillende bus protocollen(ISO 15031-1).
Een voertuig moet een van deze bus protocollen ondersteunen voor de OBD berichten. Welke bus hiervoor gebruikt wordt mag de fabrikant zelf bepalen(NEN-ISO 15031-4). OBD test en uitlees apparatuur moet alle standaarden ondersteunen. De STN1170 ondersteund dan ook alle standaard protocollen.
Via de OBD stekker kan een extern analyse apparaat OBD gegevens opvragen uit de auto(Wang et al. 242-245).

Om erachter te komen hoe een extern analyse apparaat gegevens kan opvragen uit de auto is de volgende deelvraag onderzocht:

Hoe kan OBD data opgevraagd worden?

Om de OBD data met de Cartracker te kunnen interpreteren is het van belang om te weten hoe de OBD data eruitziet.

Alle OBD informatie is gekoppeld aan een OBD PID, dit is een vaste waarde voor het opvragen van een specifieke meetwaarde uit het voertuig.

Iedere personenauto met OBD-II ondersteuning moet ondersteuning bieden voor alle relevante PID's(NEN-ISO 15031-5). Een elektrische auto met OBD ondersteuning hoeft geen ondersteuning te bieden voor het uitlezen van het brandstofverbruik in liter per uur omdat dit niet relevant is voor een elektrische auto.

Om OBD-informatie uit de auto te krijgen moet er een verzoek over de ondersteunde bus gestuurd worden naar de ECU, de ECU verwerkt deze vraag en stuurt het antwoord terug de bus op waarna de reactie door de analyseapparatuur opgevangen kan worden.

Zoals hierboven beschreven is werkt het OBD met een vraag en antwoord. De STN1170 vraagt een PID op en de ECU van de auto geeft hierop een antwoord. De exacte werking hiervan is beter onder de loep genomen omdat de Cartracker deze functionaliteit straks moet uitvoeren.

Request

Een OBD request bestaat uit twee bytes, de eerste byte geeft aan welke modus het bericht is en de tweede byte geeft het PID aan. Er zijn tien verschillende modi waarin gecommuniceerd kan worden. Elke bericht modus wordt gebruikt voor berichten met specifieke doeleinden. Zo biedt modus 01 realtime diagnostische data zoals voertuig snelheid en kan er in modus 09 statische voertuig informatie opgevraagd worden zoals het VIN. Alle verschillende modi en de toepassingen hiervan zijn in het analysedocument opgenomen.

Answer

Om te weten hoe het antwoord van de ECU gedecodeerd en gecontroleerd kan worden is de opbouw van een answer bericht onderzocht.
Een answer bericht bestaat uit drie of meer bytes. De eerste byte is gereserveerd voor het antwoordnummer, dit bestaat uit de opgevraagde mode plus 40. De tweede byte geeft het opgevraagde PID terug. De bytes die hierop volgen zijn specifiek per PID, het antwoord kan uit een of meerdere bytes bestaan.

Om de snelheid van de auto in kilometer per uur op te vragen moet de volgende hexadecimale code naar de ECU gestuurd worden:

```
Request: 01.0D
```
01: mode 01
0D: PID voor het opvragen van de snelheid in kilometer per uur

Als het voertuig 55 km/h rijdt ziet het antwoord er als volgt uit:
```
Answer: 41.0D.37
```
41: opgevraagde mode + 40 (01 + 40)
0D: herhaling van het opgevraagde PID
37: snelheid in hexadecimaal, 55 in decimaal

Welke informatie is er via OBD uit de auto te lezen?

Om te weten welke informatie er uit de auto gehaald moet worden, moet er eerst bekend zijn welke informatie er allemaal beschikbaar is. Er is gezocht naar een overzicht van alle OBD PIDs. Het volledige OBD PID overzicht was vrijwel direct gevonden.

Dit overzicht is beschikbaar op: https://en.wikipedia.org/wiki/OBD-II_PIDs

Uit het overzicht van alle beschikbare PIDs is in samenspraak met de opdrachtgever een lijst opgesteld met de belangrijkste PIDs voor de Cartracker. Deze PIDs moeten via de Cartracker uit de auto gelezen kunnen worden. In Tabel 4 zijn de geselecteerde PIDS inclusief omschrijving te zien.

Tabel 4 OBD codes selectie

PID	Omschrijving	Return bytes
010d	Voertuig snelheid in Km/h Max 255	1
010F	Seconden sinds de motor gestart is Max 65,535	2
0121	Afstand afgelegd met de malfunction lamp aan in Km Max 65,535	2
0132	Afstand afgelegd sinds code reset in Km Max 65,535	2
0151	Brandstof type	1
015E	Engine fuel rate Max 3212,75 L/h	2
0902	VIN	17-20

CAN berichten

Naast de standaard OBD berichten is het ook mogelijk om op CAN berichten te filteren vanaf de OBD stekker. Alle CAN berichten worden namelijk altijd gebroadcast over de CAN bus(ISO 11898-1). De opbouw van de berichten is gestandaardiseerd maar de betekenis hiervan mag iedere fabrikant zelf bepalen. Over de berichtbetekenis van alle auto's is dan ook geen vrije documentatie beschikbaar.

Een CAN bericht is opgebouwd uit een message ID, gevolgd door een aantal datavelden. Het dataveld bestaat bij een CAN bericht uit 1 tot 8 bytes. Via de CAN berichten kan er vaak specifieke informatie over de auto uitgelezen worden. Ook is het mogelijk om via het CAN netwerk actuatoren aan te sturen, zoals de elektrische ramen(Wang et al. 242-245).

Tijdens de analyse naar de CAN berichten gaf de opdrachtgever aan meer informatie over de beschikbare CAN berichten van de Nissan Leaf en de betekenis van de verschillende berichten.

De CAN berichten moesten opgeslagen en gefilterd kunnen worden. Om dit te realiseren is er besloten een prototype applicatie te maken. Het prototype kan gebruikt worden om de betekenis achter de CAN berichten te achterhalen via reverse engineering. De volgende stap in de analyse is dan ook om het prototype te realiseren en CAN informatie achterhalen.

5.2 Prototype

Om de gevonden informatie over CAN en de STN1170 te verifiëren is er besloten om een prototype applicatie te schrijven waarin de verschillende onderdelen van communicatieprotocollen getest en gelogd kunnen worden.

Eisen prototype

Om een duidelijk beeld te krijgen welke functionaliteiten het prototype moest hebben zijn de eisen voor de applicatie vastgelegd.
De eisen zijn als user story's opgeschreven. Bij iedere user story is een prioriteit gekoppeld volgende de MoSCoW methode.
De volgende eisen zijn opgesteld voor het prototype:

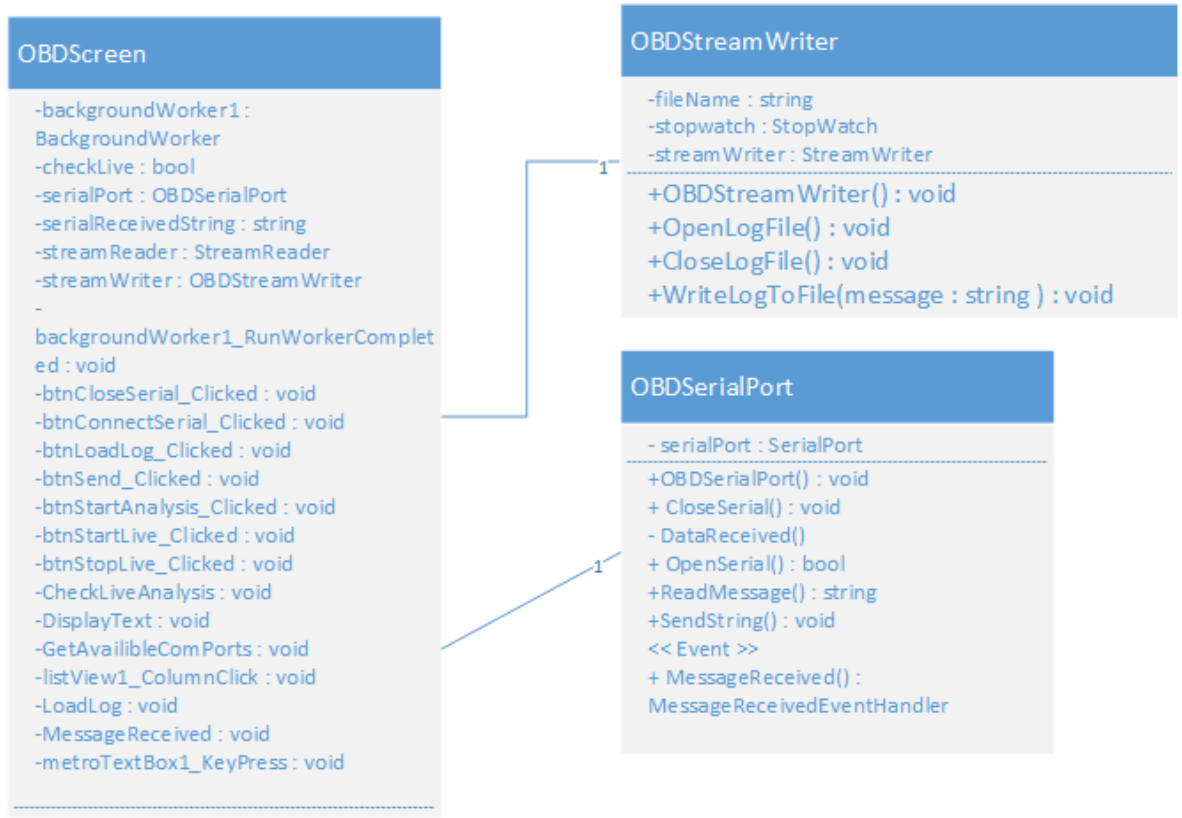
Tabel 5 Prototype eisen

Nr.	Omschrijving	Prioriteit
1	Als gebruiker wil ik de COM port waar de STN1170 op aangesloten zit kunnen selecteren	Must
2	Als gebruiker wil ik verschillende baudrates kunnen selecteren	Must
3	Als gebruiker wil ik een commando naar de STN1170 kunnen sturen	Must
4	Als gebruiker wil ik de antwoorden van de STN1170 zien	Must
5	Als gebruiker wil ik de communicatie tussen de gebruiker en de STN1170 automatisch loggen	Must
6	Als gebruiker wil ik een STN1170 log inladen in het programma	Must
7	Als gebruiker wil ik een ingeladen log kunnen sorteren	Must
8	Als gebruiker wil ik live op CAN berichten kunnen filteren	Could

User story 8 is de enige user story welke geen must prioriteit heeft gekregen. De optie voor het live filteren van CAN berichten kan het achterhalen van CAN bericht betekenissen een stuk sneller maken. Het is geen doel van de afstudeeropdracht om alle CAN berichten te achterhalen. Om alle berichten te achterhalen kan een lange tijd in gaan zitten. Omdat niet het gehele project op CAN is gebaseerd heeft deze user story een Could prioriteit gekregen.

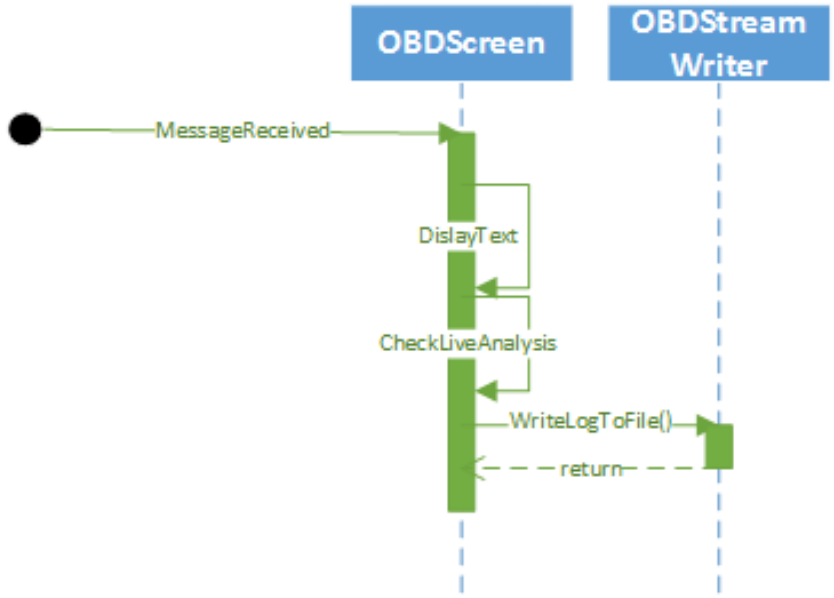
Ontwerp

Nadat de eisen bekend waren is het ontwerp van de applicatie gemaakt aan de hand van de opgestelde eisen. De applicatie bestaat uit een drietal klassen. De OBDScreen klasse is verantwoordelijk voor de interactie met de gebruiker. De diverse knoppen afhandelingen worden hier uitgevoerd. Ook de berichten van de STN1170 worden in deze Form weergegeven. Deze berichten komen uit de OBDSerialPort klasse. Deze klasse heeft een Event handler welke wordt getriggerd op inkomende berichten op de COM poort. Dit zijn de berichten van de STN1170. Om de berichten op te kunnen slaan is de klasse OBDStreamWriter. Deze klasse zorgt voor alle afhandeling met de logfiles die gecreëerd worden tijdens het gebruiken van de applicatie. In Figuur 8 is het volledige klassendiagram te zien.



Figuur 8 Klassendiagram prototype applicatie

Zodra er een bericht op de COM port binnen komt wordt dit via een event uitgelezen. In Figuur 9 is het sequentie diagram te zien die bij deze actie hoort. De MessageReceived functie wordt aangeroepen waarna het bericht weergegeven en opgeslagen wordt door de OBDStreamWriter klasse.



Figuur 9 Sequentie diagram message received

Experiment uitvoer

Met de prototype applicatie is een experiment uitgevoerd met verschillende auto's. Het doel van het experiment was het achterhalen en verifiëren van CAN message ID's en betekenissen. Deze betekenissen kunnen later in de use case CAN bus scannen gebruikt worden om zo gericht naar informatie te scannen.
De wens de opdrachtgever was om alleen op CAN berichten te scannen bij de Nissan Leaf. Om de betrouwbaarheid van de applicatie en de STN1170 te testen is er besloten om het prototype bij verschillende voertuigen te testen. Alleen voor de Nissan Leaf was het van belang om specifieke CAN berichten te achterhalen.

Het experiment is uitgevoerd met de volgende voertuigen:

- Nissan Leaf
- Toyota Auris
- BMW i3

Tijdens het experiment is er gecontroleerd of er OBD aanwezig was in de auto en of hier informatie uitgehaald kon worden via het prototype.

Resultaten

Bij de Toyota Auris werd het OBD protocol herkend door de STN1170. Bij de Nissan Leaf werd wel een CAN protocol herkend maar geen OBD ondersteuning.
Beide auto's maken gebruik van dezelfde type CAN bus. Bij de BMW i3 werd er geen ondersteund OBD protocol gevonden en geen CAN protocol herkend.
Het was ook niet vreemd dat de BMW geen ondersteuning biedt voor OBD.
Alleen auto's met brandstof uitstoot moeten voldoen aan de OBD standaarden (ISO 15031-1).

De Nissan Leaf biedt ook ondersteuning voor een niet gestandaardiseerd CAN protocol. De EV CAN, welke op pin 12 en 13 van de OBD stekker zitten(Nissan leaf Forum). Met de OBD naar RS232 kabel welke standaard bij het development board geleverd wordt is het niet mogelijk om op andere dan de standaard pinnen te scannen. Met een gemodificeerde kabel zou dit wel mogelijk zijn maar was tijdens het uitvoeren van de analyse niet beschikbaar. Via de EV CAN is het onder andere mogelijk om de State of Charge (SoC) van de Nissan Leaf uit te lezen. Dit kan in de toekomst waardevolle informatie bieden voor de gebruikers van de Cartracker.

Aan de hand van het filter wat op de STN1170 wordt ingesteld worden bepaalde CAN message ID's doorgegeven naar de applicatie. Bij het experiment is er gefilterd op een specifiek CAN message ID om het overzicht te bewaren.

Met het filter aan zijn de handelingen die aan de specifieke ID's zijn gekoppeld uitgevoerd om de gevonden informatie te bevestigen. In Tabel 6 is de verandering te zien van CAN message ID 280 byte 0 van waarde 01 naar waarde 03(Rode kader). Deze verandering treedt op als de bestuurders gordel wordt vastgeklikt. Bij het uitvoeren van deze handeling is er gelet op veranderingen in de data bytes. Hier is de live log voor gebruikt.

Bij het filteren van meerdere PIDs waarbij veel verandering optreedt, bijvoorbeeld een counter, werd de GUI traag en niet responsive genoeg om voor live analyse te gebruiken. Dit probleem trad op omdat de volledige listview bij ieder CAN bericht waar een verandering in zit, wat om de 16 milliseconde kan zijn, ververst wordt. In Tabel 6 in het groene kader is de tijd van de berichten te zien.

280 01 FF C0 00 00 00 FF C0	Timestamp ms:26783
280 **01** FF C0 00 00 00 FF C0	Timestamp ms:26799
280 **03** FF C0 00 00 00 FF C0	Timestamp ms:26815
280 03 FF C0 00 00 00 FF C0	Timestamp ms:26831
...	
280 03 FF C0 00 00 00 FF C0	Timestamp ms:28013
280 03 FF C0 00 00 00 FF C0	Timestamp ms:28029
280 **03** FF C0 00 00 00 FF C0	Timestamp ms:28060
280 **01** FF C0 00 00 00 FF C0	Timestamp ms:28076
280 01 FF C0 00 00 00 FF C0	Timestamp ms:28092
280 01 FF C0 00 00 00 FF C0	Timestamp ms:28109

Tabel 6 CAN bus filter data verandering

Omdat de focus in het project op de Nissan Leaf ligt is er besloten om de CAN messages alleen voor de Nissan Leaf te achterhalen. Om het experiment in complexiteit te beperken zijn er een viertal PIDS geverifieerd met het prototype. Op het Nissan Leaf Forum was enige informatie te vinden over de CAN messages van de Nissan Leaf(Nissan leaf Forum).

De onderstaande tabel geeft een overzicht van de betekenis van elke binaire encodering van de onderzochte PID-waarden.

PID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
280	0x01 gordel vast 0x03 gordel los							
60D		0x06 Turn signal light off 0x26 Left turn signal light 0x46 Right turn signal light 0x66 gevarenlicht						
175				AA: Neutral BB: Drive 99 Reverse				
5C5		Odometer						

Tabel 7 Nissan Leaf CAN Pid overzicht

Deze vier PIDS zijn gekozen omdat deze bytes gecontroleerd te triggeren zijn door de gebruiker. Het voertuig hoeft niet te rijden voor de test wat de test eenvoudiger maakt dan bijvoorbeeld de snelheid achterhalen via de CAN bus.

De belangrijkste PID, 5C5, welke voor de odometer gebruikt wordt is ook achterhaald. Voor de implementatie in het Cartracker systeem is dit een van de belangrijkste PIDS.

Conclusie prototype

Via de STN1170 is het mogelijk OBD en CAN bus berichten uit de Nissan Leaf te lezen. De BMW i3 biedt daarentegen geen ondersteuning voor OBD en gebruikt tevens niet de standaard CAN bus pinnen. De verzamelde gegevens zijn alleen van toepassing op de Nissan Leaf.
Met de data analyse zijn diverse CAN berichten achterhaald voor verder gebruik in het project. Ook is het mogelijk door middel van het toepassen van filters andere PIDS uit het CAN bus systeem te filteren. Het is niet bekend wat alle PIDS betekenen, hier kan reverse engineering op toegepast worden om de PID en bytes betekenis te achterhalen.
De bevindingen en gebruikte commando's kunnen in het latere ontwerp toegepast worden om zodoende dezelfde informatie uit de Nissan Leaf te achterhalen.

De live log van de applicatie blijkt niet volledig te voldoen. Als er teveel CAN berichten veranderen op het scherm wordt de responsetijd van de applicatie te hoog en is hier niet meer goed mee te werken.

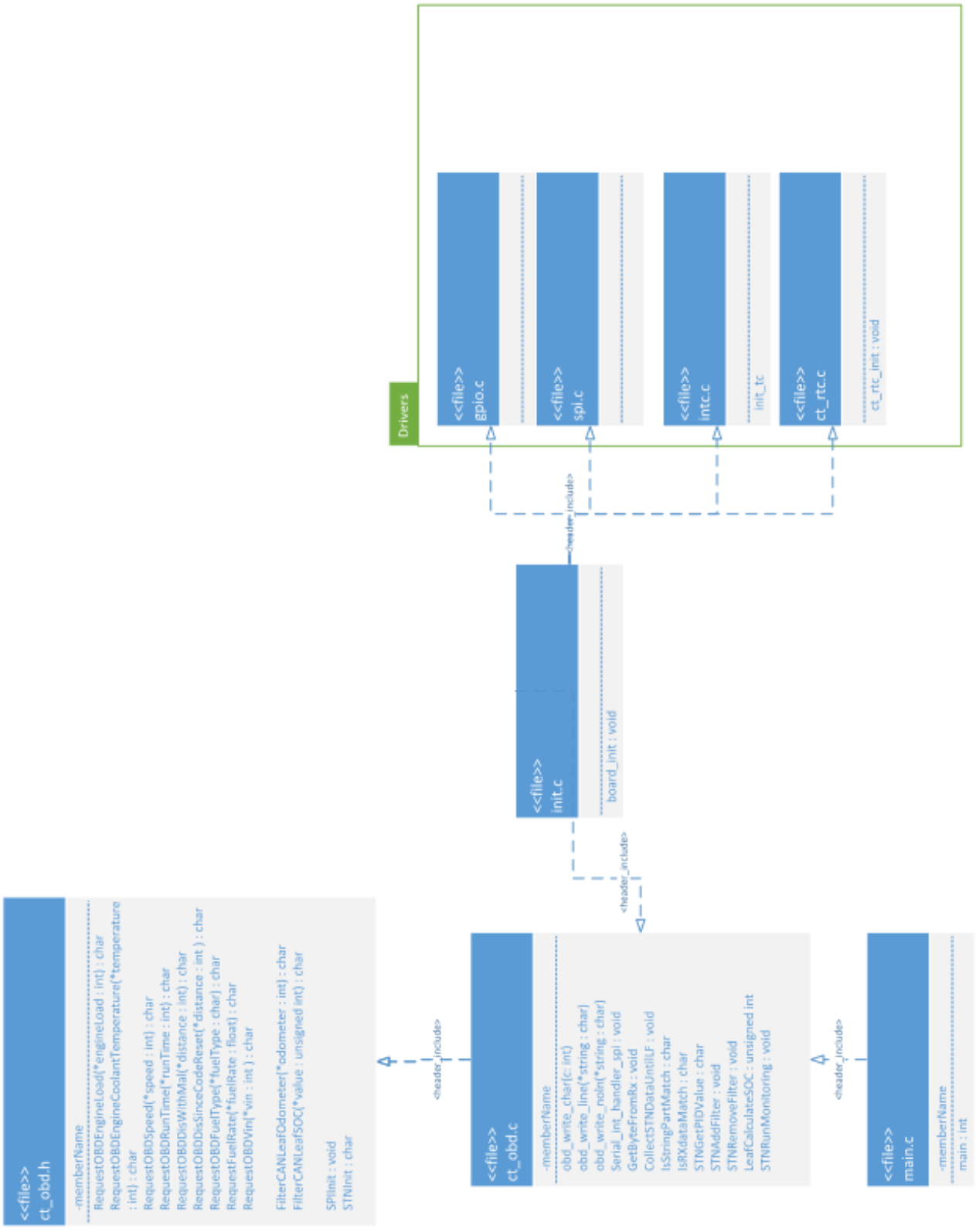
De opdrachtgever gaf voor het uitvoeren van het experiment aan dat hij geïnteresseerd was in de odometer en de SoC van de Nissan Leaf. Met het experiment is het gelukt om de odometer wel te achterhalen. De SoC, welke op de EV-CAN zit is niet geverifieerd.

5.3 Architectuur

Naast de analyse is tijdens de Elaboratie fase ook de architectuur voor het project vastgesteld om het meer details te geven en ter voorbereiding op de Construction fase. Dit is gedaan aan de hand van het architectuurdocument waarin de use cases uit het visiondocument verder zijn uitgewerkt. Het uitwerken van UML diagrammen met c code was nog niet eerder gedaan door de afstudeerder. Omdat c geen object georiënteerde taal is kunnen niet alle basisprincipes van UML direct overgenomen worden. Om een eigen draai aan de interpretatie van de UML modellen te geven is er gebruik gemaakt van UML for the C programming language(Douglass).

Use-Case uitwerkingen

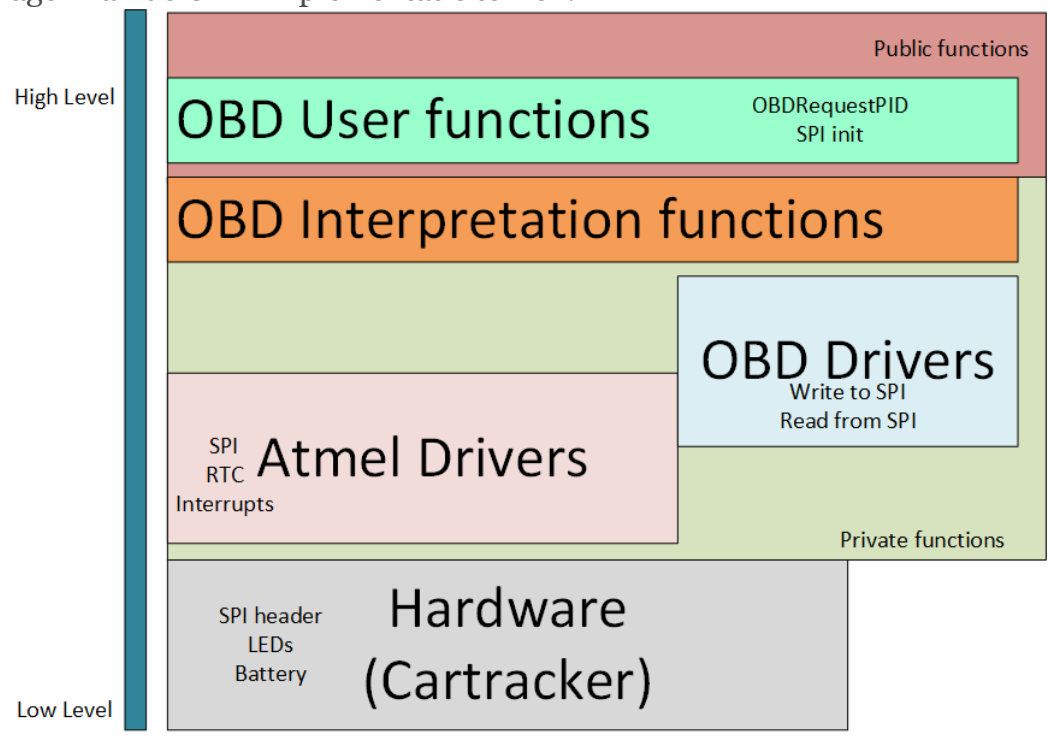
In plaats van een klassendiagram is er een file diagram gecreëerd. Dit diagram geeft alle files in het systeem en de samenhang tussen de verschillende files weer. In Figuur 10 is het file diagram voor de Cartracker te zien. De standaard Atmel drivers zijn te ondergebracht in de driver package.



Figuur 10 file diagram Cartracker software

Abstractie lagen

Om een beeld te geven welke verschillende lagen zich in de code gaan bevinden is er een software abstractie tekening gemaakt. In Figuur 11 zijn de verschillende lagen van de OBD implementatie te zien.



Figuur 11 Software abstractie lagen

De meest low level laag is de hardware van de Cartracker. Eén stap daarboven bevinden zich de Atmel en OBD drivers. De Atmel drivers zijn de standaard low level drivers aangeboden door Atmel zelf. Deze drivers worden aangesproken door de OBD drivers welke nog wel geschreven moeten worden. Om de code zo gebruiksvriendelijk mogelijk te maken hoeft de gebruiker straks alleen te de User functions aan te roepen. In de private functions worden alle afhandelingen en vertalingen van de OBD berichten uitgevoerd.

Risico’s

Tijdens de Elaboratie fase was een van de risico’s al uitgekomen.
De opdrachtgever was tijdens de afstudeerperiode een week op zakenbezoek in India. Hierdoor was er geen fysiek contact meer mogelijk in deze week.
De impact van het uitgekomen risico was achteraf klein. Dit komt vanwege de kans verlagende maatregelen. Voor vertrek is er een besprekingsmoment geweest waarbij de planning voor komende week en de communicatiemanieren waren besproken. Tijdens de zakenreis is er van Slack gebruik gemaakt om te communiceren. Hierdoor is het risico niet geheel uitgekomen en was het niet nodig om plan B, een vervangende opdrachtgever aanwijzen, toe te passen.

6 CONSTRUCTIONFASE

Na de architectuur te hebben gecreëerd is er aan de constructionfase begonnen. In iedere fase wordt er een use-case verder uitgewerkt en ontwikkeld.

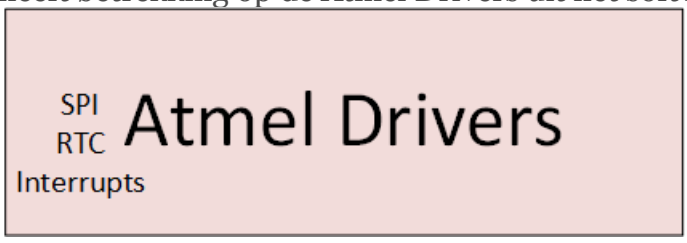
6.1 Iteratie 1 De eerste communicatie

De eerste iteratie van de constructionfase is gereserveerd voor de SPI verbinding van de Cartracker naar het opzetboard. Met het ontwerp kan de eerste use case worden gerealiseerd: STN1170 Configureren. Na de realisatie van deze verbinding kan er met de STN1170 gecommuniceerd worden. Om de eerste communicatie tussen de Cartracker en het opzetboard zo snel mogelijk op te zetten is er gekozen om niet de huidige Cartracker firmware direct uit te breiden maar een losstaand project te maken. Zo is het duidelijk welke drivers er voor het project nodig zijn en wordt de ontwikkeling niet tegengehouden door invloeden van de huidige firmware.

Ontwerp

Atmel Drivers

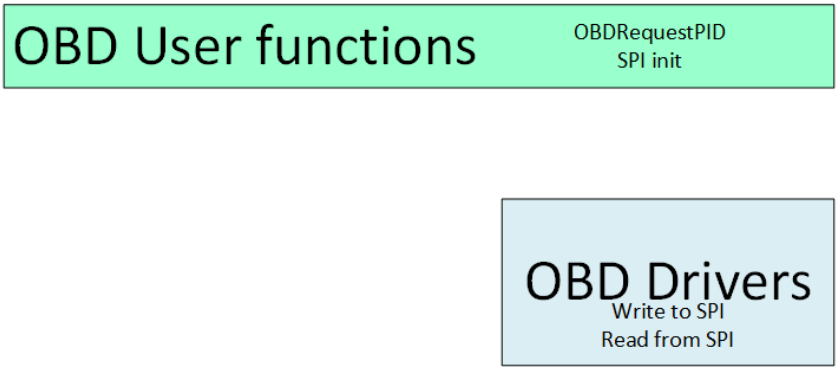
Het ontwerp van deze iteratie is gefocust op de SPI communicatie. Dit ontwerp heeft betrekking op de Atmel Drivers uit het software abstractie model.



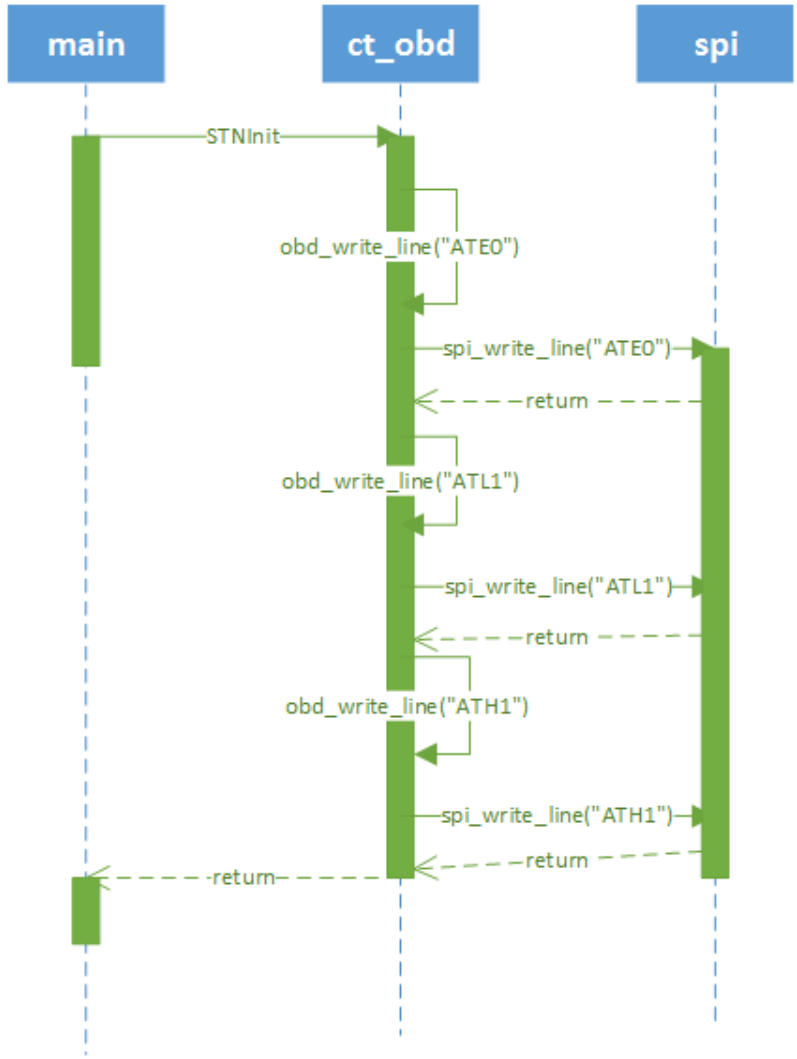
Figuur 12 Software abstractie model Atmel Driver laag

Voordat alle I/O pinnen van de microcontroller gebruikt kunnen worden moet er eest een initialisatie van alle te gebruiken pinnen uitgevoerd worden.
De initialisatie moet door de programmeur zelf geschreven worden omdat een pin vaak verschillende functies kan uitvoeren. Een pin kan bijvoorbeeld dienen als input of als output. Tijdens deze initialisatie moeten ook de SPI pinnen gedefinieerd worden. Na de initialisatie van de pinnen kan de SPI softwarematig geinitialiseerd worden.

OBD Drivers



Nadat de Atmel drivers zijn geinitialiseerd was de volgende stap een hoger level functie te hebben om de drivers met de juiste informatie te voorzien. Hiervoor zijn functies in twee lagen gemaakt. In de OBD User functions laag en de OBD Drivers laag moet deze functionaliteit toegevoegd worden. Vanuit de OBD User Function laag kan de SPI communicatie met bepaalde instellingen geladen worden. Tijdens het project is er gebruik gemaakt van de RS232 communicatie standaard.



Figuur 13 STN1170 configuratie message diagram

Na de initialisatie van de SPI bus kan de STN1170 geconfigureerd worden met een aantal commando's. In het architectuurdocument is er een message diagram opgesteld met de uit te voeren commando's. In Figuur 13 is het message diagram te zien met de AT commando's staan welke gestuurd gaan worden. De functie STNInit, aangeroepen vanuit de Main, valt onder de User functions laag, de self messages in de ct_obd vallen onder de OBD Drivers. Alle functies inclusief omschrijving zijn te vinden in het ontwerpdocument van de eerste iteratie (Bijlage E).

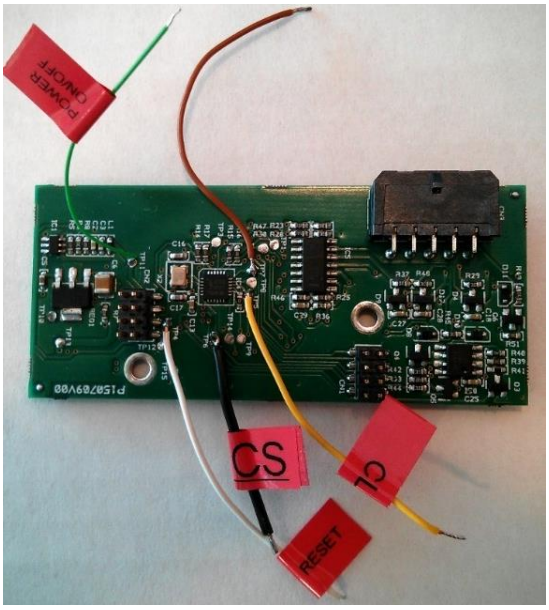
Realisatie

Om zo min mogelijk invloeden van buitenaf te hebben is er gekozen om de SPI communicatie in een nieuw project te bouwen in plaats van direct aan de bestaande Cartracker firmware toe te voegen. Omdat het OBD project op dezelfde hardware draait als de huidige Cartracker 3 versie 4 firmware zijn alle pin namen in beide projecten hetzelfde gehouden. Hierdoor kan het OBD project later makkelijker aan de huidige firmware worden toegevoegd.

Om de standaard drivers van de microcontroller aan het project toe te voegen is het Atmel Software Framework gebruikt. Via het Atmel Software Framework is het mogelijk om de benodigde drivers in het project in te laden(Atmel).

Nadat de drivers in het project toegevoegd waren moesten deze nog wel handmatig geinitialiseerd worden. Met de initialisatie worden bepaalde functies aan de pinnen gekoppeld. Na de initialisatie is het mogelijk om berichten te verzenden en ontvangen met het SPI protocol.
Om te verifiëren dat de SPI bus correct werkte werden alle signalen van de SPI met de oscilloscoop gemeten.

Na de SPI werkend bleek te zijn op de Cartracker werd het opzetboard aangesloten. Het verwachte antwoord van de STN1170 verscheen echter niet op de MISO bus. Het probleem was niet uit de software te herleiden. Om de signalen op het opzetboard te meten is er weer gebruik gemaakt van de oscilloscoop.
Door de leverancier van het opzetboard is er een aantal testpunten aangebracht zodat hier op gemeten kan worden. Alle meetpunten zijn op de onderkant bevestigd waardoor er alsnog niet op gemeten kon worden terwijl het bordje op de Cartracker zat. Er zijn een vijftal draden aan de testpunten gesoldeerd zodat hier alsnog tijdens de uitvoer gemeten kon worden. In Figuur 14 is het opzetboard met de gesoldeerde testpunten te zien. Aan de gesoldeerde draden kon de oscilloscoop aangesloten worden.



Figuur 14 STN1170 opzetboard met gesoldeerde testpunten

Tijdens het testen is de het opzetboard éénmalig verkeerd bevestigd op de Cartracker. Dit had tot gevolg dat de Cartracker en het opzetboard geen leven meer vertoonden. Dit risico was al opgenomen in het visiondocument. Hierdoor ging plan B van het risico in werking. Plan B was in eerste instantie de schade repareren en mocht dit niet lukken, reserve hardware gebruiken.
De leverancier van de hardware zat dichtbij en hier waren goede contacten mee. Dit bood de mogelijkheid om zelf de Cartracker en het opzetboard te repareren bij de leverancier zelf.

Reparatie

De reparatie vond bij de leverancier plaats omdat hier meer tools en onderdelen beschikbaar waren dan bij Decos. De gebruikte tools tijdens de reparatie zijn een oscilloscoop, desoldeerapparaat en een warmtecamera. Het desoldeerapparaat en de warmtecamera zijn niet aanwezig bij Decos.

Om erachter te komen welke componenten er defect waren is er gemeten naar kortsluitingen, wat meestal op een defect duidt. Hierbij bleek de STN1170 een kortsluiting te bevatten. Na vervanging leek het opzetboard ook weer correct te functioneren. Ook de Cartracker zelf bleek een kortsluiting te bevatten. Hierbij is een voeding ic vervangen.
Na reparatie brandde het status led van het opzetboard weer. De communicatie bleek nog steeds niet aanwezig te zijn en de HOST LED bleef dan ook uit. Door de kortsluiting bleek de MOSI van de Cartracker niet goed meer te functioneren. Omdat na de reparatie de Cartracker nog steeds niet goed functioneerde is er gekozen om ook stap twee van het risico in te laten gaan. Er is een reserve Cartracker gebruikt.

Ontwerpfout

Ook met de reserve Cartracker was er nog steeds geen succesvolle communicatie tussen de Cartracker en de opzetboard. Na een aantal metingen op het opzetboard te hebben verricht is er vastgesteld dat de pin waarop de MOSI binnen moest komen op het opzetboard aan de ground was verbonden.

Na contact te hebben opgenomen met de hardware leverancier bleek er een fout in het ontwerp van het board te zitten waardoor er een aantal pinnen van de SPI-vertaler verkeerd waren aangesloten. Deze fout leidde ertoe dat het opzetboard niet bruikbaar is in het project.

Op dit punt is er in overleg met de stagebegeleider besloten de huidige iteratie direct te stoppen en een nieuw ontwerp te maken met de beschikbare reeds aanwezige communicatiepoorten op de Cartracker.

Evaluatie iteratie 1

In het eerste ontwerp van de code is ervan uit gegaan dat de SPI gebruikt zou gaan worden. De werking van de SPI op de Cartracker is aan de hand van de oscilloscoop getest. Bij deze meting bleek de SPI correct te functioneren. Toch zodra het opzetboard op de Cartracker werd gezet verdween het MOSI signaal en was er geen communicatie met het opzetboard. Door dit probleem kon het project niet verder. De SPI interface is van belang om te communiceren met het opzetboard. Tijdens het testen zijn er een aantal defecten opgetreden welke gerepareerd moesten worden. De reparatie van het opzetboard is goed verlopen. Na reparatie bleek de Cartracker zelf meer schade te hebben opgelopen.

Na betere inspectie van het opzetboard bleek deze verkeerd te zijn ontworpen. Met dit probleem kon het gehele opzetboard niet meer gebruikt worden. Er is snel een nieuwe oplossing gekozen en de eerste iteratie kwam helaas door hardware problemen eerder ten einde.

Volgende iteratie

Nu het opzetboard niet meer gebruikt kan worden moet er naar een andere manier van communicatie worden gekeken. De eerste stap zal zijn welke communicatie port er gebruikt zal gaan worden. Hierna wordt er een nieuw ontwerp met de nieuwe communicatievorm gemaakt en getest.

6.2 Iteratie 2 Nieuw communicatie protocol

Vanwege de hardware problemen in de eerste iteratie is nogmaals gericht op de communicatie tussen de Cartracker en de STN1170.

Van SPI naar USART

De eerste stap in de nieuwe iteratie was het zoeken naar een nieuw communicatie bus. In eerste instantie is er gekeken welke poorten er beschikbaar zijn op de Cartracker. In overleg met de begeleider is er besloten om gebruik te maken van de 12 polige Molex connector. In plaats van de SPI communicatie welke in eerste instantie gebruikt zou worden zou er nu via USART bus gecommuniceerd worden.

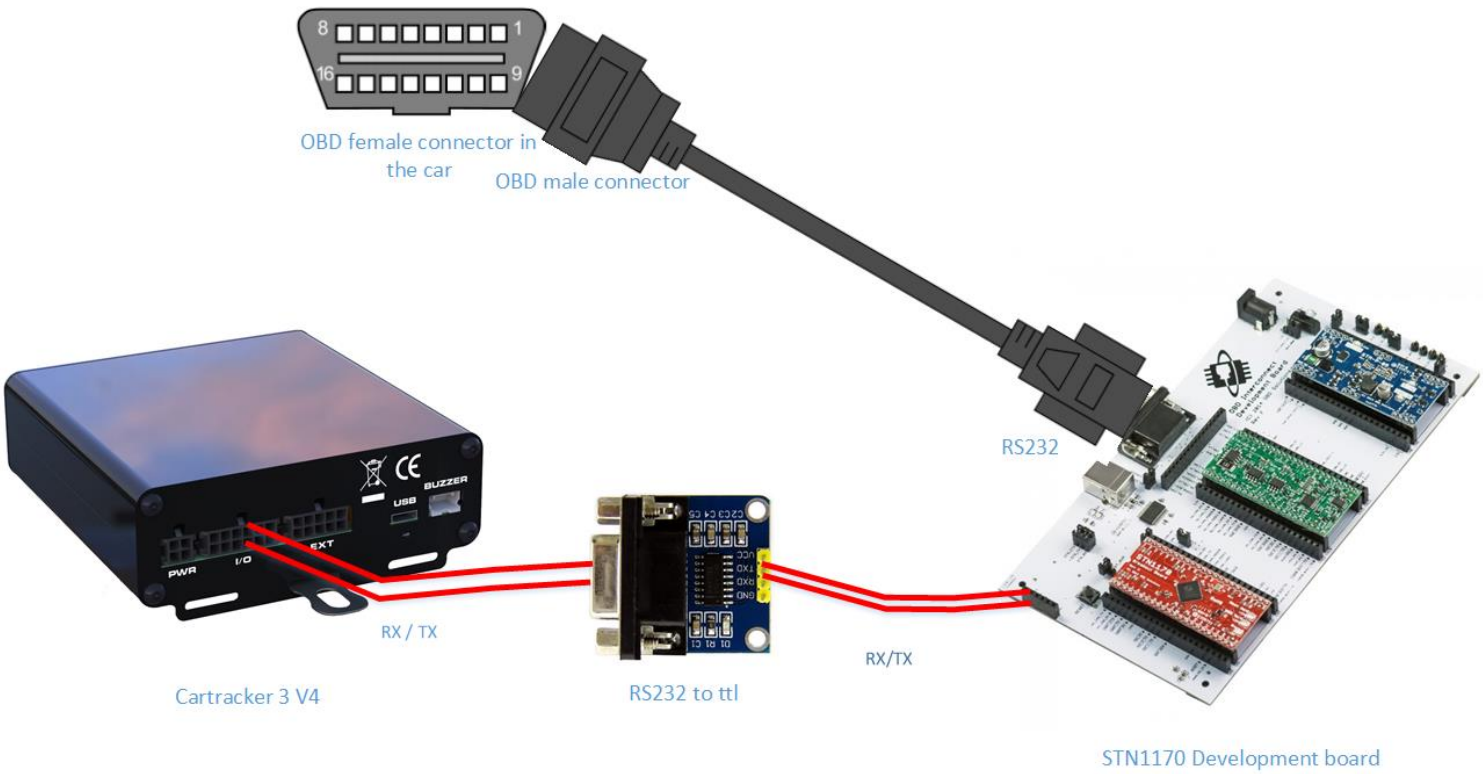
Nieuw ontwerp

Om een goed beeld te krijgen van de nieuwe hardware die in nieuwe ontwerp gebruikt gaat worden is er een globaal overzicht van alle hardware onderdelen gemaakt en hoe deze onderling aan elkaar zijn verbonden.

Figuur 15 Oude componenten tekening

In de oude componenten tekening is de verbinding tussen de Cartracker en de OBD van de auto geregeld door het opzetboard. Nu het opzetboard komt te vervallen moet hier een nieuwe vertaler voor in de plaats komen. De communicatie tussen de Cartracker en de OBD van de auto moet nog wel via de STN1170 gebeuren. Vanwege de korte resterende tijd is er gekeken om een oplossing te realiseren met de beschikbare onderdelen.

Om geen nieuwe aankopen van ontwikkel boarden te hoeven doen is er gebruik gemaakt van het STN1170 Development Board. Hier was tijdens de elaborationfase al van bewezen dat deze communicatie werkte.



Figuur 16 Nieuwe componenten tekening

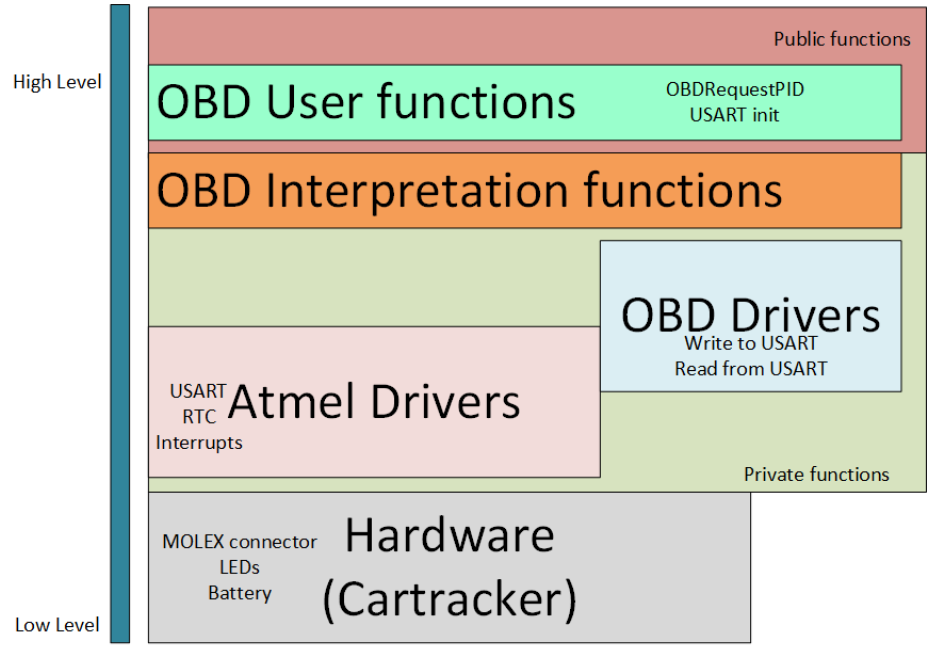
In de nieuwe componenten tekening, zie Figuur 16, wordt de molex connector gebruikt op de Cartracker. De molex poort geeft een 12V signaal uit, om met het STN1170 development board te communiceren is er een 12V naar 5V vertaler nodig. In de tekening zit deze tussen de Cartracker en het STN1170 development board.

Nadat er nieuwe hardware gekozen is voor het project is er naar het ontwerp gekeken van de communicatie tussen de Cartracker en de STN1170.

Ontwerp

Nadat het opzetboard weg was gevallen uit het ontwerp zijn er andere poorten gebruikt om de STN1170 aan te sturen. In plaats van de SPI bus wordt er gebruik gemaakt van een van de USART porten op de Cartracker. Dit heeft onder andere tot gevolgen dat in plaats van de SPI bus de USART bus geinitialiseerd moet worden in de software.

Uit de abstractie laag tekening zijn de SPI functies vervangen voor USART functies. In Figuur 17 is de nieuwe tekening te zien. In de Atmel Drivers en de OBD Drivers zijn de SPI drivers vervangen voor de USART drivers.



Figuur 17 Software abstractie lagen USART aanpassingen

Net zoals in iteratie 1 was het doel de eerste communicatie tussen de Cartracker en de STN1170 realiseren. Buiten de initialisatie van de andere communicatiepoort hoefde er aan het ontwerp voor de code weinig te veranderen. De aansturing van de STN1170 kon nu direct via USART gaan, er was geen SPI naar USART vertaler meer nodig.

Test

Gezien de vorige iteratie vroegtijdig is afgebroken is er in iteratie 2 voor het eerst getest met behulp van een testrapport (Bijlage G). De use case STN1170 Configureren is hierbij uitgevoerd en getest. Ook de initialisatie van de microcontroller en de benodigde USART bussen zijn hierin meegenomen.

De geteste eisen voor deze iteratie waren:

F1	De Cartracker moet de STN1170 kunnen configureren
Nf6	De Cartracker moet na het aansluiten op een spanningsbron direct de OBD verwerking starten.

Bij iedere test case is er een verwachte uitvoer gemaakt op basis van de resultaten van het prototype. In Tabel 8 Use-Case STN1170 configureren test resultatenzijn de testresultaten te zien zoals te vinden in het eerste testrapport (bijlage).

Tabel 8 Use-Case STN1170 configureren test resultaten

Te testen eis	Invoer	Verwachte uitvoer	Daadwerkelijke uitvoer	Opmerking	Akkoord
Nf6	De Cartracker wordt op een spanningsbron aangesloten	LED1 gaat om de seconde knipperen	LED1 gaat om de seconde knipperen	Kon niet via prototype getest worden.	Ja
F1	De Cartracker stuurt het commando “ATH1” naar de STN1170	De Cartracker krijgt het antwoord “OK” binnen	De Cartracker krijgt het antwoord “OK” binnen		Ja
F1	De Cartracker stuurt het commando “ATE0” naar de STN1170	De Cartracker krijgt het antwoord “OK” binnen	De Cartracker krijgt het antwoord “OK” binnen		Ja
F1	De Cartracker stuurt het commando “ATL1” naar de STN1170	De Cartracker krijgt het antwoord “OK” binnen	De Cartracker krijgt het antwoord “OK” binnen		Ja

Evaluatie iteratie 2

Na de problemen van de eerste iteratie is er direct naar een alternatieve communicatiemethode gekeken. Uit het overleg met de opdrachtgever bleken de USART bussen het beste alternatief. Hiermee is een nieuw ontwerp gemaakt. Ook het STN1170 development board is in dit ontwerp meegenomen omdat het opzetboard, waar de STN1170 op zit helemaal niet meer bruikbaar was. De USART communicatie is vlot opgezet en het verwachte resultaat is in alle testgevallen bereikt. Hiermee is aangetoond dat de communicatie tussen de Cartracker en de STN1170 correct werkt.

Volgende iteratie

Nu de communicatie werkt kan er ingegaan worden op de inhoud van de OBD berichten.

Met de gemaakte functies in deze iteratie kunnen er berichten naar de STN1170 gestuurd worden en kan het antwoord worden gelezen. De volgende stap wordt het interpreteren van de antwoorden. Hiermee wordt de volgende use-case OBD code opvragen uitgevoerd.

6.3 Iteratie 3 OBD berichten

In de derde iteratie is er gekeken naar de OBD informatie verwerking. Het doel van deze iteratie was om twee use cases te implementeren. OBD berichten opvragen en OBD berichten verwerken.

In iteratie 2 is een correcte werking van de hardware aangetoond. In de derde iteratie lag de focus op de software voor de OBD. De berichten worden opgevraagd en het antwoord wordt verwerkt naar een leesbare waarde voor de gebruiker.

Het is voor de Cartracker niet noodzakelijk om alle mogelijke OBD berichten uit te kunnen lezen. De eerdere verzamelde belangrijkste OBD PIDs, te vinden in Tabel 4, zijn geïmplementeerd in deze iteratie.

Gezien vanuit de software abstractie tekening werd ervan af onderaf steeds verder naar boven gewerkt. De driver lagen zijn nu werkend, zo konden de OBD User function laag en de OBD Interpretation functions laag ingericht worden.

Interpretatie laag

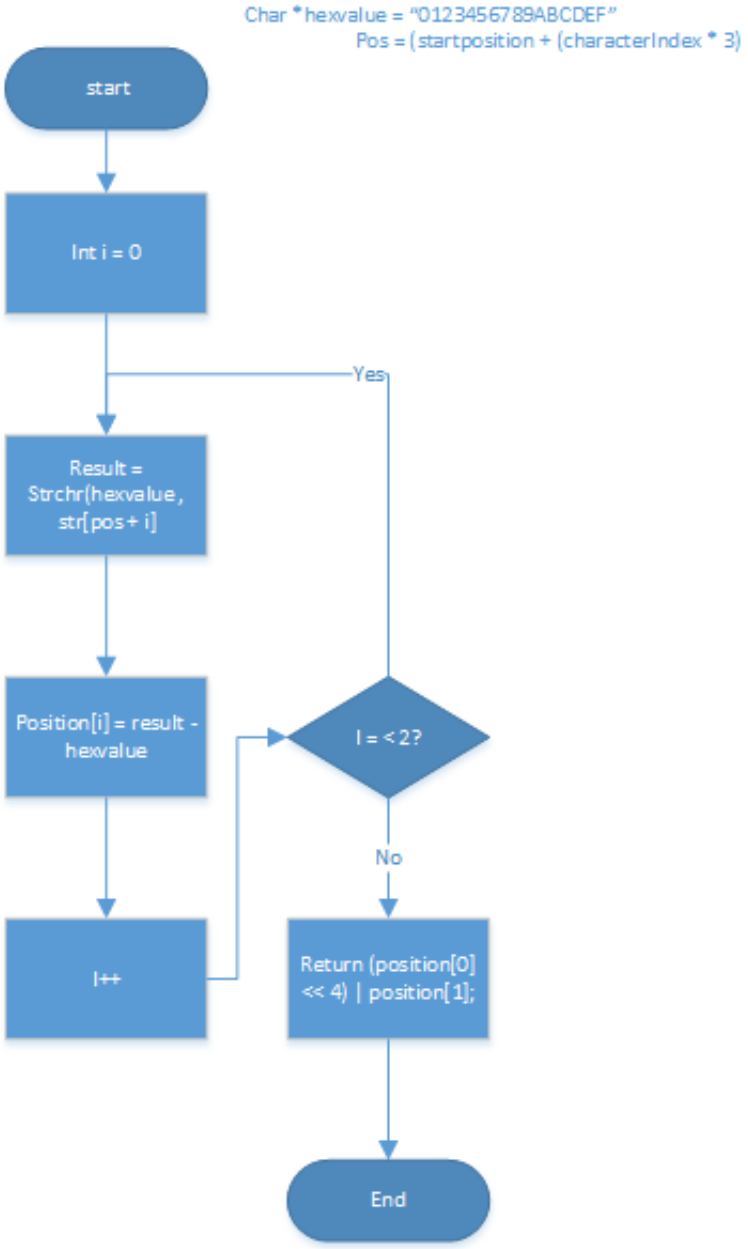
Om te beginnen is het opvragen van een bericht richting de driver laag uitgewerkt. De implementatie hiervan was snel uitgevoerd en konden vanaf nu OBD berichten opgevraagd worden.
In de huidige Cartracker 3 versie 4 firmware wordt ook gebruik gemaakt van UART communicatie.
Om het wiel niet opnieuw uit te hoeven vinden en de OBD ondersteuning compatible te houden met de huidige software is er gekozen om UART verwerking uit de huidige firmware te implementeren in het OBD project. Alleen het inlezen van de berichten was te gebruiken uit de huidige firmware. Voor het verwerken van de ingekomen berichten moest een eigen implementatie geschreven worden. Dit was de volgende stap in deze iteratie.

De OBD berichten komen als ASCII karakters binnen. Hier kan niet direct mee gerekend worden. Om het mogelijk te maken om berekeningen uit te voeren met de ingekomen waardes moesten deze vertaald worden naar decimale getallen. Er zijn twee manieren van verwerken bekeken.

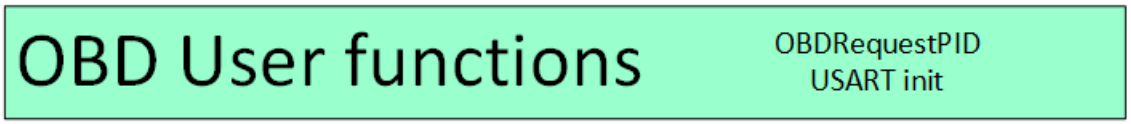
Bij de eerste oplossing wordt het karakter omgezet naar de decimale waarde uit de ASCII tabel(ASCII). Als het karakter tussen de 0 en 9 zit wordt er 48 van de decimale waarde afgehaald. Zo blijft het decimale getal over. Voor karakters van A t/m F wordt er 55 afgehaald.

Bij de tweede oplossing wordt het karakter vergeleken met een string waarin alle hexadecimale waardes op volgorde zitten. Het ingelezen karakter wordt vergeleken met de string. De positie van het karakter in de string is de waarde van het karakter.

Voor de implementatie is oplossing twee gekozen. Kan er een functie voor zowel de getallen als de letters gebruikt worden. Bij de eerste oplossing moet er een vergelijking gemaakt worden of het om een getal of om een letter gaat. In Figuur 18 is de flowchart van de functie GetHexValueFromOBDString te zien. Deze functie wordt gebruikt voor de vertaling van de ASCII codes naar integers.



Figuur 18 Flowchart GetHexValueFromOBDString functie



User functions laag

Om het gebruik van de OBD functionaliteit zo makkelijk mogelijk te maken worden alle OBD verzoek berichten ondergebracht in de publieke OBD User functions laag. Op deze manier is het niet nodig om per OBD bericht een verwerking te schrijven, dit wordt namelijk door de OBD interpretation functions gedaan.
Alle gekozen PID's uit Tabel 4 zijn geïmplementeerd in de OBD User function laag en zijn via de STN1170 uit de auto op te vragen. Voor alle functies zie Bijlage H.

Simulator

Tijdens de derde iteratie was het tussentijds testen van een stuk code een proces met veel handelingen. De code kon alleen in een auto getest worden.

Tijdens het testen werd het idee geopperd om een OBD simulator te zoeken zodat het testen productiever gedaan kon worden. Hiermee kon tijd bespaard worden tijdens het ontwikkelen.

Na overleg zijn de volgende eisen opgesteld:

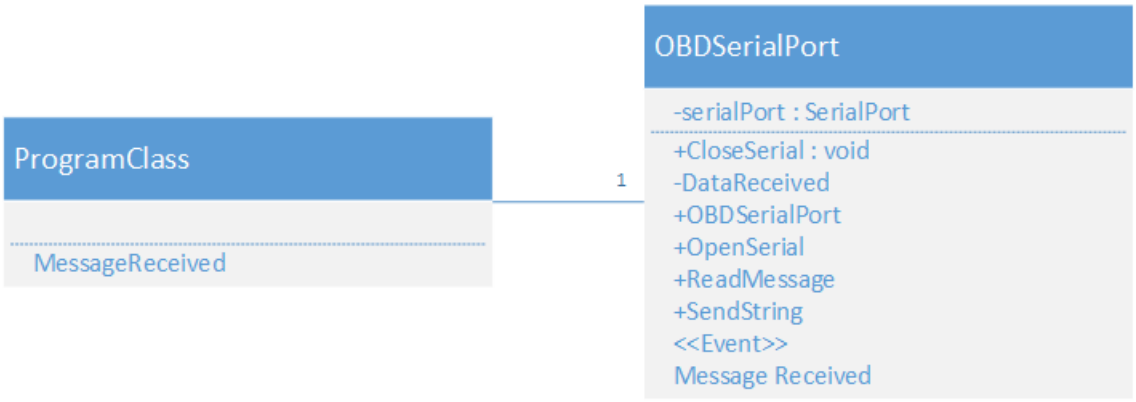
Nr.	Beschrijving
1	De simulator moet op een Windows machine kunnen draaien
2	De simulator moet verschillende reacties kunnen geven op een OBD request
3	De simulator moet meerdere OBD berichten ondersteunen
4	De simulator moet niet gestandaardiseerde CAN berichten kunnen versturen
5	De antwoorden op de OBD berichten moeten door de gebruiker gewijzigd kunnen worden
6	De simulator moet gratis te gebruiken zijn

Tijdens het zoekproces is er één simulator gevonden die aan de eisen voldeed: OBDSim(Gordon). Helaas bleek de simulator niet meer werkend onder Windows 10.

Om toch een simulator te kunnen gebruiken is er vanwege het feit dat er geen simulator beschikbaar is zelf een simulator gebouwd.

Simulator bouwen

Bij het ontwerpen van de simulator is de seriële verbinding van het prototype gebruikt. De SerialPort klasse uit het prototype is overgenomen. In plaats van de data alleen te loggen moest de data nu geïnterpreteerd worden en kon er een antwoord terug gestuurd worden over de COM port.



Figuur 19 Simulator klasse diagram

Test

De software is in eerste instantie getest met de simulator. Hierbij kon een eigen uitvoer bedacht worden waarna de verwachte uitvoer van de Cartracker vastgesteld kon worden. Voor het uitvoeren van de test werd er eerst via de pc en een terminal de verschillende PIDs gecontroleerd of dit overeenkwam met de uitvoer van de simulator.

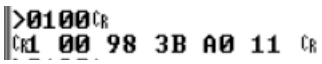
Tijdens het testen bleken de bedachte PIDs uit Tabel 4 niet in alle auto's ondersteund te worden. De test kon dus niet met één auto voltooid worden. Tijdens het testen was er beschikking over twee auto's:

- Peugeot 307
- Renault Megane

Overige PIDs welke op beide auto's niet ondersteund worden, zijn via de OBD simulator getest. Alle testresultaten zijn te vinden in bijlage I.

PID ondersteuning

Via een OBD request is het mogelijk om de ondersteunde PIDs van de auto op te vragen. Met bericht 0100 wordt er een 4 bytes aan data teruggegeven. In Figuur 20 zijn de return waardes van de Renault Megane te zien.



Figuur 20 OBD-data Renault Megane

Om een beeld te geven hoe deze gegevens verwerkt kunnen worden is het antwoord van de Renault verwerkt in de volgende paragraaf.

Ondersteunde PIDs achterhalen

De vier hexadecimale bytes worden vertaald naar binaire waardes. Iedere bit staat voor één pid welke wel of niet ondersteund wordt. In Tabel 9 is de decodering van pid 0100 van de Renault Megane te zien. De groene PIDs (binair 1) worden wel ondersteund en de rode PIDs (binair 0)

Tabel 9 OBD ondersteunde PIDs decodering

HEX	**9**				**8**				**3**				**B**			
BIN	1	0	0	1	1	0	0	0	0	0	1	1	1	0	1	1
PID	1	2	3	4	5	6	7	8	9	0a	0b	0c	0d	0e	0f	10
HEX	**A**				**0**				**1**				**1**			
BIN	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
PID	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20

Aan het begin van de iteratie bij het ontwerp is deze functionaliteit niet meegenomen. Tijdens de test bleek deze functionaliteit wel gewenst voor verder gebruik. Tijdens de test is deze functionaliteit alsnog toegevoegd. Met deze functie kan er opgevraagd worden of een bepaald PID ondersteund wordt door de auto of niet.

Evaluatie iteratie 3

De verdeling van functies in de verschillende abstractie lagen gaf een overzichtelijk resultaat in de verschillende functies. Met de huidige code is het simpel om nieuwe OBD PIDs te ondersteunen en toe te voegen aan de huidige code. Het aantal ondersteunde PIDs in voertuigen bleek minder te zijn dan verwacht. Door tijdens het testen gebruik te maken van verschillende auto's konden de verschillende OBD berichten toch geverifieerd worden. Om dit probleem voor te zijn in verdere ontwikkelingen is er tijdens de test nog een nieuwe functie toegevoegd aan het ontwerp. De OBD ondersteuning werkt op de Cartracker als bedacht en is toepasbaar op verschillende voertuigen. De simulator is een goede toevoeging geweest voor het hele project. Hiermee is de nodige tijd bespaard en kan ook voor verdere ontwikkelingen gebruikt worden.

Volgende Iteratie

Nu de OBD ondersteuning is gerealiseerd wordt de volgende iteratie gewijd aan de CAN bus berichten. De Cartracker zal uiteindelijk ondersteuning voor OBD en voor niet gestandaardiseerde CAN berichten moeten bieden. Via de prototype applicatie is er aangetoond dat er met de STN1170 ook niet OBD CAN berichten uitgelezen kunnen worden. Hierbij is het van belang dat de Cartracker filters kan toepassen op de STN1170 en de CAN berichten kan verwerken.

6.4 Iteratie 4 CAN bus filtering

De laatste iteratie stond in het teken van het uitlezen van CAN berichten via de OBD stekker uit de Nissan Leaf. Uit de analyse in de elaborationfase is achterhaald welke type berichten er over de CAN bus gestuurd worden van de Nissan Leaf en hoe de opbouw van de berichten eruitziet.

Om aan te tonen dat de Cartracker buiten OBD berichten ook CAN berichten kan inlezen en verwerken zijn er een tweetal CAN berichten gekozen om te implementeren. Tijdens de analyse is ontdekt dat er meerdere CAN bussen op de OBD stekker van de Nissan Leaf te vinden zijn. Om aan te tonen dat er op verschillende CAN bussen gelezen kan worden was de doelstelling om op twee verschillende bussen twee verschillende berichten uit te lezen.

Het odometer CAN bericht is gekozen voor de standaard CAN bus. Op de EV-CAN is er voor de SoC gekozen.

Standaard CAN

Een niet gestandaardiseerd CAN bericht is niet op dezelfde manier op te vragen uit de STN1170 dan een OBD bericht. Om een specifiek CAN bericht te willen weten kan er een filter toegepast worden.
Om niet op alle CAN berichten te moeten filteren in de Cartracker wordt er gebruik gemaakt van de STN1170 CAN filter. De verschillende filter opties worden beschreven in de STN1170 Family Reference and Programming Manual(OBD Solutions 23).

EV-CAN

De berichten op de EV-CAN zijn qua opbouw hetzelfde als de CAN berichten op een andere bus. Toch was het niet direct mogelijk om op de EV-CAN berichten uit te lezen.

De kabel geleverd bij het STN1170 development board kan alleen uitlezen op de standaard OBD pinnen. Om op de EV-CAN van de Nissan Leaf CAN berichten uit te lezen is er een gemodificeerde kabel gemaakt. Hiermee worden de pinnen van de EV-CAN verwisseld met de standaard pinnen. Op deze manier hoeft er op het development board geen aanpassingen plaats te vinden om toch op een niet standaard bus te scannen. Voor alle toegevoegde functies met betrekking op het CAN filter zie Bijlage J.

Test

Net als bij iteratie 3 is de test in eerste instantie via de simulator getest. De simulator geeft ook bij het instellen van een filter een OK commando terug zoals de STN1170 zou doen. In Figuur 21 is de simulator output te zien. Eerst wordt het filter ingesteld met het commando: STFAP 55B, FFF. Hiermee wordt er op CAN header 55B gefilterd. Hierna wordt er met het commando STM gefilterd met de ingevoerde filters. In het geval van de test bestaat het filter uit een CAN bericht.

```
Received: STFAP 55B,FFF
Return:OK
Received: STM
Return:55B B2 B2 53 E9 00 0C 00 00
```

Figuur 21 simulator STN filtering

Na het testen met de simulator zijn er twee testen uitgevoerd in de Nissan Leaf. De eerste test is uitgevoerd op de standaard zonder gebruik te maken van de EV-CAN kabel. De tweede test werd uitgevoerd. In beide gevallen kwamen de resultaten over een met de resultaten van de simulator. Voor alle test resultaten voor de CAN bus implementatie zie bijlage K.

CAN database

Na de implementatie van de CAN bericht verwerking zijn de opties bekeken voor het uitbreiden van de CAN berichten database. Tijdens de analyse in de elaborationfase zijn er enkele CAN bericht betekenissen achterhaald via internet. Veel meer beschikbare informatie is er tijdens de analyse niet gevonden.
De twee opties voor het verkrijgen van een CAN database zijn zelf CAN berichten achterhalen met reverse engineering of een reeds bestaande CAN database inkopen bij een andere partij.

Omdat de verschillende type auto's die gebruik maken van de Cartracker zeer divers is viel de reverse engineering optie af. Dit neemt ten eerste zeer veel tijd in beslag en er is geen beschikking over grotendeels van de auto's. Daarom is er naar het inkopen van een bestaande CAN database gekeken. Na verschillende partijen te hebben gebeld bleek deze informatie minder goed beschikbaar dan in eerste instantie gedacht. Eén partij biedt een redelijk complete CAN database aan, in combinatie met hun eigen CAN uitlees product. Dit is niet in combinatie met de STN1170 te gebruiken maar zou mogelijk in combinatie met de Cartracker allen samen kunnen werken.

7 CONCLUSIE EN AANBEVELINGEN

In dit hoofdstuk wordt de conclusie van het gehele project beschreven, verder worden er aanbevelingen gedaan voor verder gebruik van de bevindingen uit dit project.

7.1 Conclusie

Voor de aanvang van het project lag de probleemstelling al vast:

> *Het is onbekend hoe er door middel van de STN1170 in samenwerking met de Cartracker OBD berichten uit een personenauto gelezen en verwerkt kunnen worden door de Cartracker.*

De benodigde hardware voor het project leek al reeds aanwezig. De opdracht zou uitgevoerd gaan worden met een al ontworpen opzetboard.

Kijkend naar de probleemstelling en de behaalde resultaten kan er gezegd worden dat alle problemen in de probleemstelling aangepakt en beantwoord zijn. Tijdens het project zijn er toevoegingen geweest op het project, zoals de simulator en de verschillende CAN bus uitleesmogelijkheden, die het project en voor later gebruik ten goede hebben gedaan. Tijdens de afstudeerperiode is er aangetoond dat er door middel van externe hardware OBD berichten uit de auto gelezen kunnen worden en verwerkt kunnen worden door de Cartracker. De OBD ondersteuning is nog niet aan de huidige software toegevoegd.

7.2 Aanbevelingen

PID ondersteuning

Het is op dit moment mogelijk om de OBD informatie op te vragen van geselecteerd aantal PIDs. De lijst met beschikbare PIDs in een auto is vele male groter. Het opvragen van een OBD bericht is zo ontworpen dat er makkelijk nieuwe PIDs toegevoegd kunnen worden. Omdat het mogelijk is dat verschillende klant van Cartracker verschillende informatie uit de auto willen weten kan er nagedacht worden om alle beschikbare PIDs te implementeren.

PID configuratie
Zoals tijdens het testen in iteratie 3 bekend is geworden ondersteunen niet alle auto's alle OBD PIDs. Via diverse websites is er te achterhalen welke PIDs er in de diverse voertuigen ondersteund wordt. Er is een functie geïntroduceerd om een specifieke PID op te vragen. In de praktijk is dit niet erg effectief om iedere PID te moeten af gaan. Om ervoor te zorgen dat er alleen PIDs opgevraagd worden die daadwerkelijk door een auto ondersteund worden kan het interessant zijn deze informatie vanaf een website op te vragen en deze informatie door te sturen naar de Cartracker.

Triggers
Met het proof of concept worden alle uitgelezen OBD en CAN berichten aan de gebruiker getoond. Als de OBD functionaliteit in de huidige firmware geïmplementeerd gaat worden zal deze informatie via een simkaart over het internet verstuurd worden. Om het dataverkeer van de OBD berichten naar de server te beperken kan er gekeken worden naar diverse controles op de berichten. Het is niet nodig om iedere seconde de odometer van de auto te weten. Deze controles zitten nog niet in het proof of concept.

Fout codes
Met OBD is het mogelijk diverse foutcodes uit een auto te lezen. Er is tijdens de bouw van het proof of concept helaas geen tijd geweest om deze functionaliteit nog toe te voegen. Deze informatie kan voor een wagenpark beheerder zeer waardevol zijn en lijkt mij daarom lijkt mij dit ook zeker een feature om in de toekomst te gebruiken in combinatie met een melding via de mail met de foutcodes hierin van een voertuig. Op deze manier kan de wagenpark beheerder hier direct op inspelen.

CAN database
Tijdens het project is er maar van een aantal CAN berichten de betekenis achterhaald. Deze betekenissen kunnen veel meerwaarde bieden bij het uitlezen van voertuigen zoals de odometer of dat de bestuurder zijn gordel wel om heeft tijdens het rijden.
Om ondersteuning te bieden met het uitlezen voor dit soort CAN berichten is het noodzakelijk om een database met diverse CAN messages en betekenissen ter beschikking te hebben. Op dit moment is deze database niet aanwezig. Het is waarschijnlijk niet rendabel genoeg om dit zelf uit te zoeken via reverse engineering.

Hardware
Tijdens de eerste iteratie bleek het ontwikkelde opzetboard niet goed te functioneren. Dit resulteerde in het gebruik van andere hardware. De uiteindelijke gebruikte hardware voor het project kan niet in de huidige Cartracker behuizing gemaakt worden. Om dit te kunnen realiseren moet er een nieuw ontwerp voor een opzetboard gemaakt worden. Met kijk op de toekomst en de nieuwe Cartracker 3 versie 5 is het gebruik van het huidige opzetboard sowieso geen optie.

Simulator

De simulator is een waardevolle toevoeging geweest tijdens het testen van de OBD functionaliteiten op de Cartracker. Om het testen van diverse berichten makkelijker te maken kan de simulator met verschillende opties uitgebreid worden. Zoals een UI waarbij de waardes van de berichten gezien en ingesteld kunnen worden. Ook zou er een echte CAN simulator ingebouwd kunnen worden die diverse CAN berichten naar de STN1170 stuurt. Zo kunnen de filter opties getest worden.

8 PROJECT EVALUATIE

In dit hoofdstuk wordt de projectevaluatie beschreven.

Op de doelstelling:

Inzicht geven op het verwerken van OBD berichten met de Cartracker, in samenwerking met de STN1170

Is een antwoord gegeven in de vorm van een proof of concept.

Het implementeren van de OBD ondersteuning in de huidige Cartracker 3 versie 4 software is helaas niet aan bod gekomen. Door het project niet in een nieuw project te maken maar direct aan de firmware toe te voegen had dit uiteraard wel aangetoond.

Defenitiefase

De keuze van een iteratieve ontwikkelmethode heeft het project ten goede gedaan. Als er gebruik was gemaakt van een watervalmethode zou het project vastgelopen bij het realiseren van het proof of concept. Om RUP te gebruiken had voor het project nog niet vast hoeven te staan maar heeft het project niet benadeeld.

Inceptionfase

Tijdens de inceptionfase zijn de behoeften van de opdrachtgever achterhaald. Er is alleen zeer gericht op de Cartracker hardware en software ingegaan. Op dit punt had er op een bredere manier naar het project gekeken moeten worden. Zo had het prototype en de simulator al in de eisen meegenomen moeten worden tijdens deze fase. Gelukkig bood RUP hier wel de ruimte voor om deze onderdelen te implementeren.

Elaborationfase

De analyse heeft veel opheldering en genoeg details geleverd voor de rest van het project. Uit de analyse is ook de wens voor het prototype ontstaan. Dit had met het doel CAN message betekenissen achterhalen een extra product kunnen zijn in plaats van een prototype. Als hier meer tijd aan was besteed had dit na het project meer informatie kunnen achterhalen in de toekomst. Op dit moment is het concept nog niet ver genoeg uitgewerkt naar eigen mening om goed in het veld te gebruiken.

Constructionfase

In de eerste iteratie uit de constructionfase kwam een fout met een grote impact aan het licht. In het defecte opzetboard is achteraf tijd gaan zitten wat hier niet aan besteed had hoeven worden. Na het verdwijnen van de signalen op het opzetboard had er wellicht gerichter gezocht moeten worden naar het probleem. Op alle meet punten is gemeten, terwijl het probleem zich alleen richtte op de MOSI pin. Tijdens het op en afhalen van het board is er met te weinig oplettendheid gehandeld. Hierdoor is het opzetboard en een Cartracker defect geraakt, dit had simpel voorkomen kunnen worden door een visuele check uit te voeren.

De simulator bood tijdens het project zeker meerwaarde. Achteraf hadden de eventuele voordelen van een simulator al tijdens de inceptionfase aan het licht moeten komen. Dan hadden er specifiekere eisen en een betere analyse uitgevoerd kunnen worden. Ook had de simulator als een resultaat meegenomen kunnen worden voor de Cartracker met OBD ondersteuning. Gelukkig bood de iteratie voldoende ruimte voor een korte analyse en realisatie van de simulator.

Er had wat meer tijd besteed moeten worden aan het ontwerpen van de OBD PID opvragen. Tijdens het ontwerpen van de OBD PIDs opvragen had er beseft moeten worden dat niet iedere auto alle PIDs ondersteund. Tijdens het testen met verschillende auto's kwam dit pas naar voren. Er is geen analyse gedaan welke PIDs er ondersteund worden bij de test auto's ook al is deze informatie wel op internet beschikbaar

9 AFKORTINGEN

CAN	Controller Area Network Seriële bus gebruikt in voertuigen
ECU	Electronic Control Unit Microcontroller in de auto voor het aansturen van actuatoren
EV CAN	Electric Vehicle CAN CAN bus met alleen elektrische voertuig specifieke informatie
MCU	Micro controller unit
MISO	Master In Slave Out
MOSI	Master Out Slave In
OBD	On Board Diagnostics
PID	Parameter Identification number Uniek nummer gegeven aan een bepaalde parameter
RTOS	Real Time Operating System
SoC	State of Charge Percentage van opladen
SPI	Serial Pheripheral Interface Synchrone seriële data link
uC	Microcontroller
USART	Universal Synchronous/Anynchronous Receiver/Transmitter Chip voor het vertalen van data over bijvoorbeeld RS232
VIN	Voertuig Identificatie Nummer Uniek identificatie nummer van een voertuig

BIBLIOGRAFIE

1. ISO 15031-1. *Road Vehicles — Communication between Vehicle and External Equipment for Emissions-Related Diagnostics — Part 1: General Information.*, 2001. Print.

2. Barry, Richard. *Using the FreeRTOS Real Time Kernel: A Practical Guide*. Real Time Engineers, 2010. Print.

3. OBD Solutions. "STN 11000 Family Reference and Programming Manual." 2009. Web. 23-2-2016 <https://www.scantool.net/scantool/downloads/98/stn1100-frpm.pdf>.

4. NEN-ISO 15031-4. "Road Vehicles - Communication between Vehicle and External Equipment for Emissions-Related Diagnostics - Part 4: External Test Equipment." (2005) Print.

5. Wang, Dai Qiang, et al. "Intelligent Control System Based on CAN-Bus for Car Doors and Windows". *Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference on*. IEEE, 2009. 242-245. Print.

6. NEN-ISO 15031-5. "Road Vehicles — Communication between Vehicle and External Equipment for Emissions-Related Diagnostics — Part 5: Emissions-Related Diagnostic Services." (2015) Print.

7. ISO 11898-1. "11898-1–Road vehicles–Controller Area Network (CAN)–Part 1: Data Link Layer and Physical Signalling." *International Organization for Standardization* (2003) Print.

8. Nissan leaf Forum. "LEAF CAN bus decoding. (Open Discussion)." 2015. Web. 24-2-2016 <http://www.mynissanleaf.com/viewtopic.php?t=4131>.

9. Douglass, Bruce Power. "UML for the C Programming Language." (2009) Web. 27-2-2016.

10. Atmel. "Atmel Software Framework." 2016. Web. 8-3-2016 <http://www.atmel.com/tools/avrsoftwareframework.aspx>.

11. ASCII. "ASCII table." 2010. Web. <http://www.asciitable.com/>.

12. Gordon, Ryan C. "OBDSIM." 2012. Web. 19-4-2016 <https://icculus.org/obdgpslogger/obdsim.html>.