Cartracker met OBD ondersteuning

Extern bijlagenboek

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

Bijlage A	Afstudeerplan
Bijlage B	Visiondocument
Bijlage C	Analysedocument
Bijlage D	Architectuurdocument
Bijlage E	Ontwerpdocument iteratie 1
Bijlage F	Ontwerpdocument iteratie 2
Bijlage G	Testrapport iteratie 2
Bijlage H	Ontwerpdocument iteratie 3
Bijlage I	Testrapport iteratie 3
Bijlage J	Ontwerpdocument iteratie 4
Bijlage K	Testrapport iteratie 4

BIJLAGE A

Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2016-1.1 (start uiterlijk 8 februari 2016)
Startdatum uitvoering afstudeeropdracht: 8 februari 2016
Inleverdatum afstudeerdossier volgens jaarrooster: 3 juni 2016

Studentnummer: 12035769
Achternaam: dhr Eigenbrood (*) *weghalen niet van toepassing*
Voorletters: R I
Roepnaam: Robbert
Adres: Peterhof 126
Postcode: 2201TJ
Woonplaats: Noordwijk
Telefoonnummer: 0628998702
Mobiel nummer:
Privé emailadres: robberteigenbrood@gmail.com

Opleiding: Technische Informatica
Locatie: Den Haag (*) *weghalen niet van toepassing*
Variant: voltijd

Naam studieloopbaanbegeleider: Ellen van der Weijden
Naam begeleidend examinator: Jan Dirk Schagen
Naam tweede examinator: Hans de Vreught

Naam bedrijf: Decos
Afdeling bedrijf: Cartracker
Bezoekadres bedrijf: Huygensstraat 30
Postcode bezoekadres: 2201 DK
Postbusnummer: -
Postcode postbusnummer: -
Plaats: Noordwijk
Telefoon bedrijf: +31 883326701
Telefax bedrijf: +31 0713620833
Internetsite bedrijf: www.cartracker.nl (www.decos.nl)

Achternaam opdrachtgever: dhr van Hanegem (*) *weghalen niet van toepassing*
Voorletters opdrachtgever: A (Alain)
Titulatuur opdrachtgever:
Functie opdrachtgever: Software Architect
Doorkiesnummer opdrachtgever:
Email opdrachtgever: A.vanHanegem@decos.com

Achternaam bedrijfsmentor: dhr van Hanegem (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: A (Alain)
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Software Architect
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: A.vanHanegem@decos.com

NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Uitvoeren van onderzoek naar de implementatie van OBD(On-board diagnostics) aan de Cartracker van Decos

Opdrachtomschrijving

1. **Bedrijf**

Decos Cartracker is onderdeel van de Decos Technology Group. Decos maakt innovatieve software oplossingen op het gebied van Fleet Management en Enterprise Content Management en heeft een Outsourcing Center in Pune, India. Decos is opgericht in 1987 door CEO Paul Veger en heeft ruim 200 werknemers in Nederland en India.(bron: www.cartracker.nl)
De producten die bij Decos Cartracker ontwikkeld zijn zijn de Cartracker en de applicatie Flo. Beide producten zorgen voor minder CO2 uitstoot en helpen mee aan bewust rijden. De Cartracker functioneert nu als Track and Trace device, kilometerregistratie en kan gekoppeld worden aan de tankpas.

2. Probleemstelling

Wat op dit moment nog ontbreekt zijn gegevens over het functioneren van de auto zelf, zoals defecten en storingen. Door middel van het OBD systeem in een auto kunnen deze gegevens uitgelezen worden. Cartracker ondersteund het OBD systeem op dit moment nog niet. De OBD berichten kunnen niet direct door de Cartracker worden uitgelezen. Hiertussen komt de STN1170 om er voor te zorgen dat de Cartracker de berichten uit kan lezen.

Probleemstelling:
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.

3. **Doelstelling van de afstudeeropdracht**

De doelstelling is dat de huidige Cartracker ondersteuning biedt voor het OBD systeem in samenwerking met de STN1170.

4. **Resultaat**

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

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

Als ontwikkelmethode is er gekozen voor RUP(Rational Unified Process)

Werkzaamheden inclusief globale planning:
Een week bestaat uit 5 dagen waarvan één dag gereserveerd is voor het schrijven van de scriptie

Inception(1 weken)

- Verhelderen opdracht
    - Use-cases opstellen
    - Risico's analyseren
    - Globale fasen planning maken

Elaboration(5 weken)

- Huidige Cartracker hardware en software analyseren
- OBD domein analyseren
- STN1170 mogelijkheden analyseren
- Use cases uitwerken
- Functionele eisen opstellen
- Niet functionele eisen opstellen
- Risico's herzien
- Iteratie 1 plannen
- Architectuur ontwerpen

Construction (11 weken)

De construction fase zal worden opgedeeld in 4 fases. Binnen deze fases zullen alle functionaliteiten afgesproken in het visiondocument gerealiseerd worden. De globale planning voor de construction fase ziet er als volgt uit:

Iteratie 1: 2 weken

- OBD berichten inlezen STN1170

Iteratie 2: 3 weken

- Cartracker berichten ontvangen STN1170

Iteratie 3: 3 weken

- Cartracker OBD berichten interpreteren en verwerken

Iteratie 4: 3 weken

- OBD versturen naar Cartracker server

6. Op te leveren (tussen)producten(Milestones)

Inception milestones
- *Visiondocument*

Elaboration milestones
- *Architectuurdocument*

Construction milestones

Iteratie 1: - Implementatie use-case OBD berichten inlezen
- *Interface board van de STN1170 met bijbehorende software voor het uitlezen van OBD berichten*
- *Testrapport Implementatie OBD berichten inlezen inclusief ontwerp als bijlage*

Iteratie 2 - Implementatie use-case Cartracker berichten inlezen
- *Koppeling tussen de STN1170 en de Cartracker door middel van het interface board en software op de Cartracker*
- *Testrapport Cartracker berichten inlezen inclusief ontwerp als bijlage*

Iteratie 3 - Implementatie use-case Cartracker berichten interpreteren
- *Verwerking van OBD berichten op de Cartracker*
- *Testrapport Cartracker berichten interpreteren inclusief ontwerp als bijlage*

Iteratie 4 - Implementatie use-case OBD versturen naar server
- *Testrapport huidige en voorgaande iteraties inclusief ontwerpen.*

- *Ontwerpdocument*

7. Te demonstreren competenties en wijze waarop

G1 Praktische aspecten hanteren in projecten
- Risicoanalyse in het visiondocument

A1 Analyseren van het probleemdomein
- Probleembeschrijving in het architectuurdocument
- UML en Flowcharts in het architectuurdocument

C8 Ontwerpen van een technisch informatie systeem
- UML, PSD en flowchart ontwerpen te vinden in het ontwerpdocument

D17 Testen van software systemen
- Testrapport per Iteratie uit de Constructionfase

BIJLAGE B

Visiondocument

Visiondocument

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	: 1.0

Inhoudsopgave

1.INLEIDING

Dit document geeft de algemene visie tussen het Decos Cartracker team en afstudeerder Robbert Eigenbrood op het Cartracker OBD implementatie project. In dit document is de visie van beide partijen opgenomen. Deze visie zal dienen als basis voor verdere uitvoeringen binnen het project. De probleemstelling en de doelstelling zijn vastgesteld, alsmede een eerste versie van de use cases en verwachte risico's tijdens het project.

2. POSITIONERING

2.1.

Huidige bedrijfsproces

Decos Cartacker 3 is op dit moment operationeel in het veld. De Cartracker 3 maakt gebruik van ritregistratie en wordt er real-time inzicht gegeven over de weggebruiker. Deze gegevens worden op dit moment verzameld aan de hand van GPS. Het is mogelijk om via het OBD (On-Board Diagnostics) systeem informatie uit de auto op te vragen. Deze informatie zou gebruikt kunnen worden in combinatie met de Cartracker. Op deze manier kan de functionaliteit van de Cartracker vergroot worden. De reden voor het toevoegen van OBD aan de Cartracker is primair bedoelt voor het uitlezen van de tellerstand van het voertuig. Op dit moment moet dit ieder kwartaal met de hand ingevoerd worden per auto, omdat de huidige kilometer telling op basis van GPS niet 100% nauwkeurig is. Dit kan resulteren naar een foutieve kilometerstand. Via OBD kunnen er nauwkeuriger gegevens uit de auto gemeten worden.

Probleemstelling

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.

Doelstelling

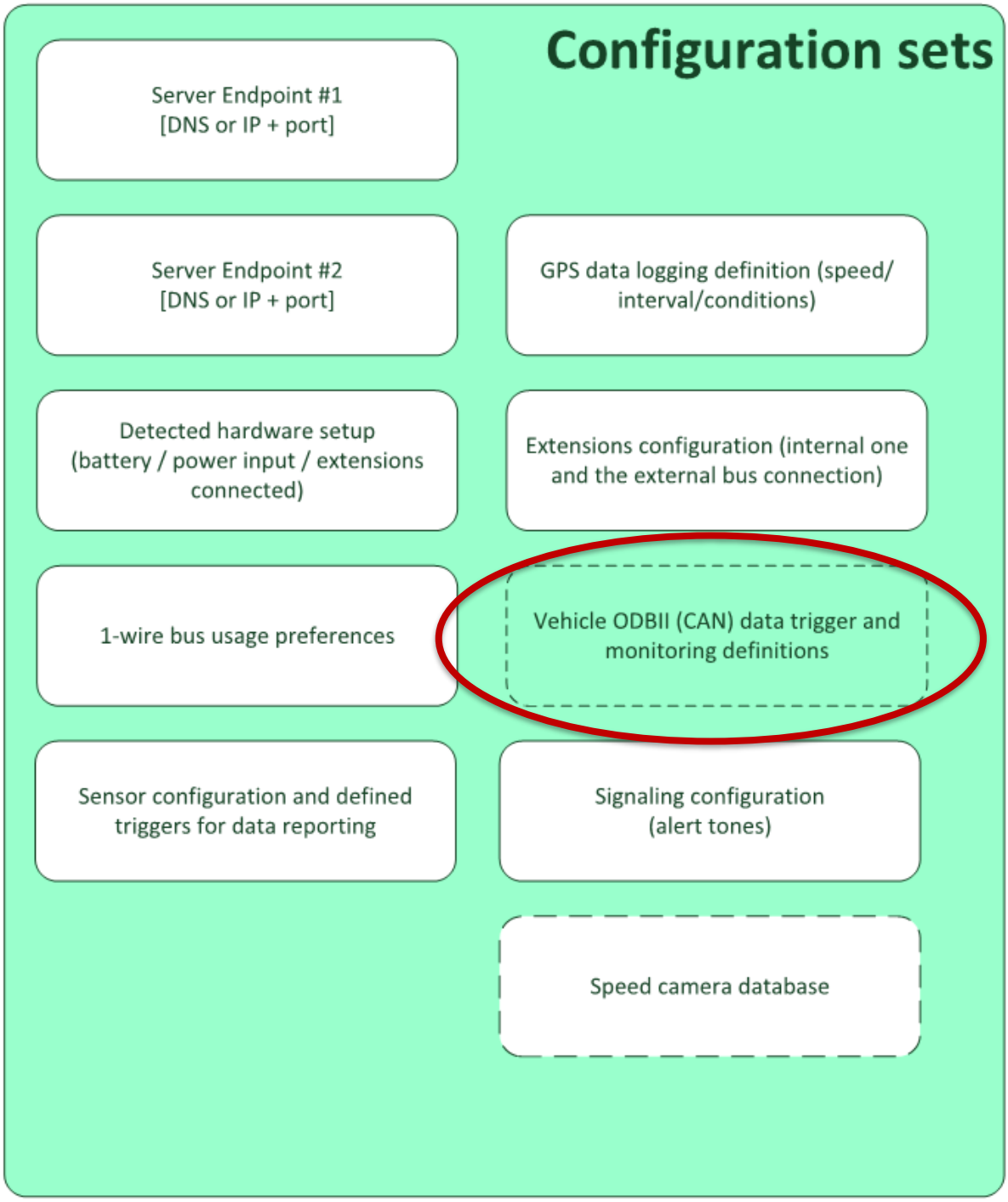
De doelstelling van het project is dat de Cartracker ondersteuning biedt voor het OBD systeem, in samenwerking met de STN1170.

Acceptatiecriteria

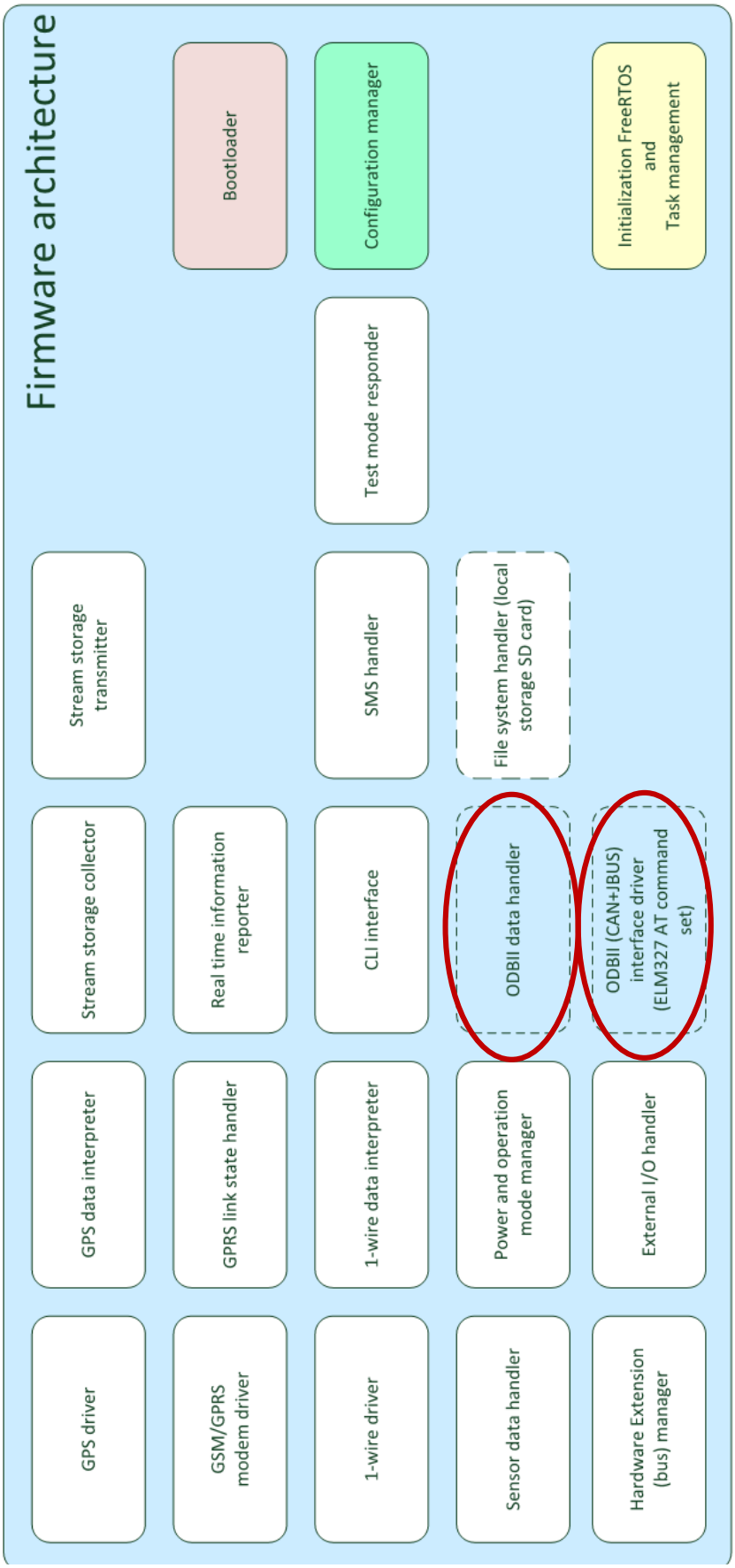
Het resultaat is een proof of concept van de Cartracker met ondersteuning van OBD berichten op het gebied van inlezen, verwerken en doorsturen hiervan, in samenwerking met de STN1170.

3. PRODUCTPERSPECTIEF

De OBD integratie zal een toevoeging zijn aan de huidige Cartracker Versie 3. Binnen de huidige Cartracker firmware zal er een OBD interface driver en OBD data handler toegevoegd worden. In Figuur 1 is de configuratie set van de Cartracker V3 te zien. In de set is de configuratie voor het OBD systeem al opgenomen maar nog niet ontworpen en geïmplementeerd. In Figuur 2 Cartracker V3 Firmware Blokdiagram is het firmware blokdiagram van de Cartracker V3 te zien. In het diagram is de OBD driver en handler al opgenomen in de architectuur, maar op dit moment zijn beide functies nog niet geïmplementeerd.



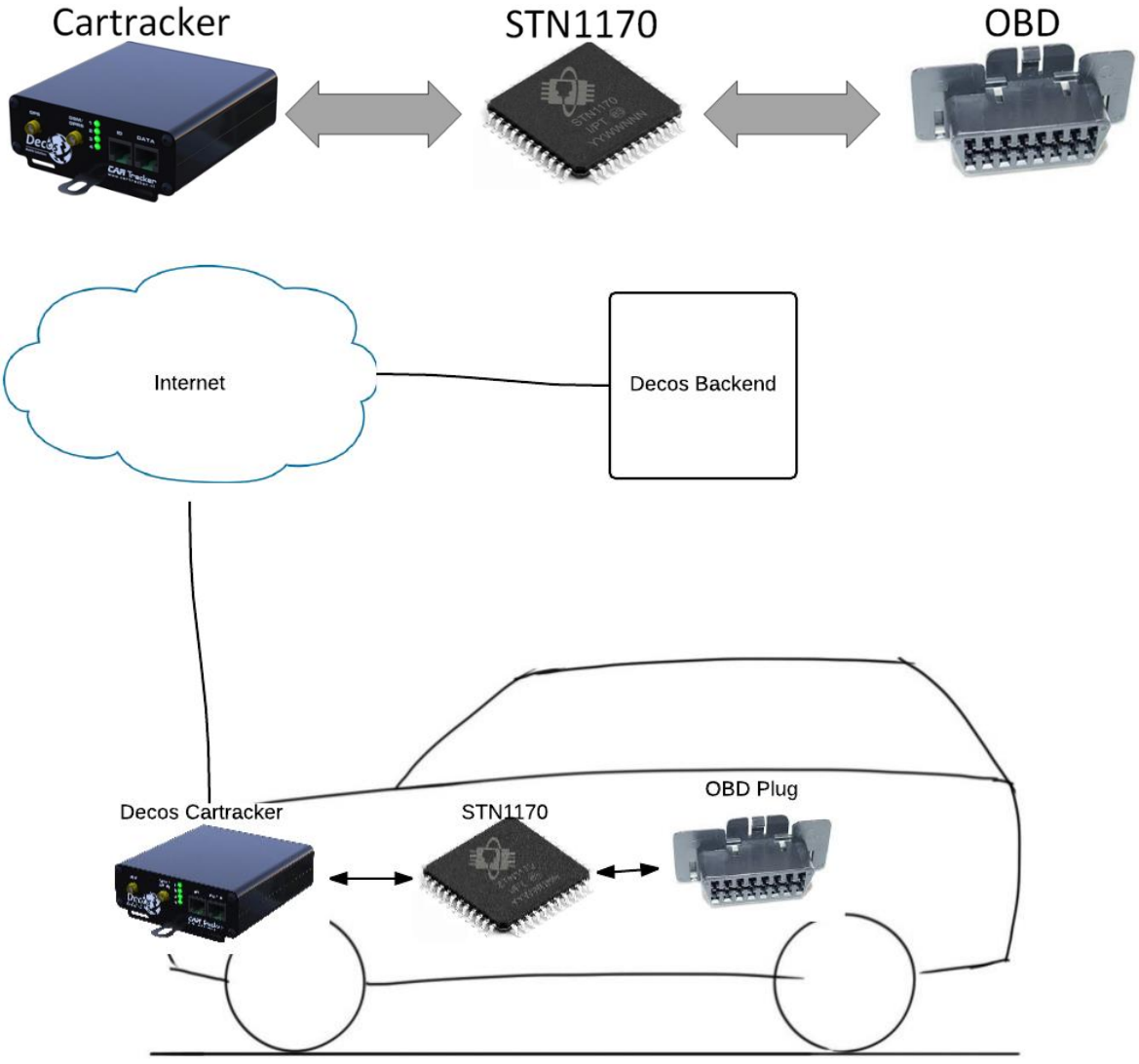
Figuur 1 Configuration settings Cartracker V3



Figuur 2 Cartracker V3 Firmware Blokdiagram

4. PRODUCTDEFINITIE

In dit hoofdstuk worden de producteigenschappen van de Cartracker met OBD functionaliteit beschreven. Als eerste wordt er een globaal overzicht gegeven van de Cartracker met OBD aansluiting. Vanuit het perspectief vanuit de Cartracker zal er een use case beschrijving gegeven worden. Alleen de nieuwe OBD functionaliteit zal beschreven worden in de use case beschrijving. De use cases zijn opgesteld aan de hand van de gegeven informatie uit het productperspectief. Iedere case wordt beschreven met een use case beschrijving.



Figuur 3 Globale tekening Cartracker met OBD ondersteuning

De Cartracker staat in verbinding met de OBD van de auto door middel van de STN1170. De informatie verkregen uit de OBD wordt via de STN1170 doorgestuurd naar de Cartracker die de informatie verder zal verwerken.

Behoeften opdrachtgever

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 use case diagram en bijbehorende use case scenario's zijn uitgewerkt en besproken met de opdrachtgever. De volgende behoeften zijn voortgekomen uit deze gesprekken:

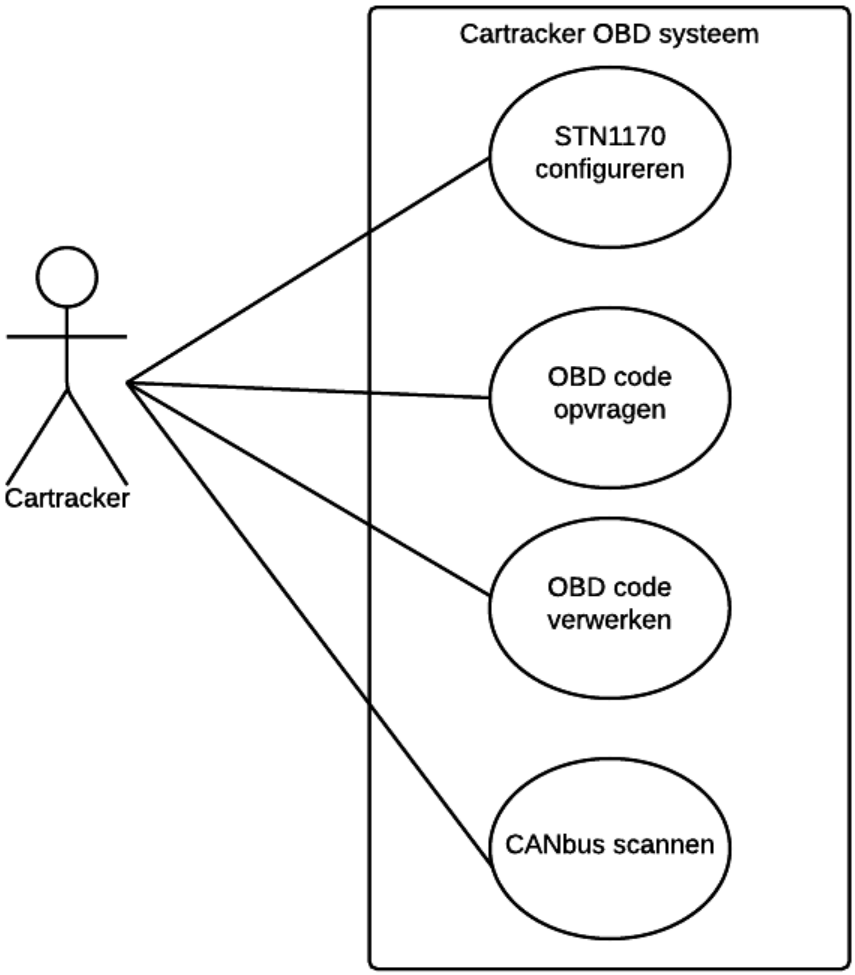
Functionele behoeften

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

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

5. USE CASE DIAGRAM



Figuur 4 Use case diagram

In het use case diagram zijn vier use cases opgenomen:

- STN1170 configureren. In deze use case wordt de STN1170 door de Cartracker geconfigureerd na het opstarten
- OBD code opvragen: De Cartracker vraagt een specifieke code OBD code op aan de STN1170
- OBD code verwerken: De Cartracker verwerkt het ontvangen OBD antwoord
- CAN bus scannen: De Cartracker scant de CAN bus naar relevante berichten

De use cases zijn uitgewerkt met een use case beschrijving. De volgende punten zijn per use case beschreven:

- Naam

- Actor
- Aannamen
- Beschrijving
- Uitzondering
- Resultaat

Use case beschrijvingen

Naam	**STN1170 configureren**
Actor	Cartracker
Aannamen	De Cartracker is met de juiste OBD configuratie geconfigureerd De STN1170 is nog niet geconfigureerd
Beschrijving	De STN1170 wordt geconfigureerd 1- Juiste commando's worden ingevoerd 2- De STN1170 geeft een melding terug aan de Cartracker
Uitzonderingen	
Resultaat	De STN1170 is correct geconfigureerd

Naam	**OBD code opvragen**
Actor	Cartracker
Aannamen	De STN1170 is correct geconfigureerd
Beschrijving	1- De Cartracker stuurt een request naar de STN1170 2- De STN1170 stuurt een request over de OBD bus naar de auto 3- De auto stuurt het antwoord naar de STN1170 4- De STN1170 controleert de geldigheid van het OBD bericht. Indien het bericht niet geldig is treedt uitzondering 1 op. 5- De STN1170 stuurt het OBD bericht door naar de Cartracker 6- De Cartracker ontvangt het opgevraagde bericht
Uitzonderingen	1- De STN1170 verstuurd het verkeerd ontvangen bericht niet en gaat terug naar stap 3
Resultaat	De opgevraagde code is ontvangen

Naam	**Bericht verwerken**
Actor	Cartracker
Aannamen	Er is een bericht opgevraagd en ontvangen
Beschrijving	1- De Cartracker interpreteert het ontvangen bericht 2- De Cartracker geeft het bericht weer aan de gebruiker
Uitzonderingen	
Resultaat	Het ontvangen bericht is geïnterpreteerd en verwerkt in de Cartracker

Naam	**CANbus scannen**
Actor	Cartracker
Aannamen	
Beschrijving	1- De Cartracker stuurt een filterspecificatie naar de STN1170 2- De STN1170 geeft een OK bericht terug aan de Cartracker 3- De Cartracker stuurt een Start Scan bericht naar de STN1170 4- De STN1170 stuurt het filterresultaat naar de Cartracker
Uitzonderingen	
Resultaat	Er is een gefilterd CAN bericht ontvangen op de Cartracker

6. RISICO ANALYSE

De risicoanalyse is opgezet om een inzicht te geven in de mogelijke risico's die binnen dit project kunnen voordoen.

Voor de risicoanalyse geldt het volgende:

Alle risico's zijn op het project toegesneden

Van alle risico's zijn de grootte van de kans en impact bekend

Van alle risico's zijn zowel kans verlagende als impact verlagende maatregelen opgesteld

Voor alle risico's is een plan B opgesteld.

Nummer	**1**
Omschrijving	Benodigde hardware is niet aanwezig
Risico	De benodigde hardware om de STN1170 te programmeren is niet aanwezig
Kans	Klein
Kansverlagende maatregel	In de analyse fase de beschreven benodigde hardware aanschaffen
Impact	Groot
Impactverlagende maatregel	Meerdere activiteiten per iteratie plannen, mocht het risico door gaan dan is niet één hele iteratie niet uit te voeren maar een gedeelte niet
Plan B	Een iteratie in de constructionfase omdraaien en de benodigede hardware aanschaffen zodat de iteratie waarbij de STN1170 nodig is later uitgevoerd kan worden.

Nummer	**2**
Omschrijving	De opdrachtgever is niet bereikbaar
Risico	Decos voert ook ontwikkeling in India uit. Hierdoor is de kans aanwezig dat de opdrachtgever in India werkzaam moet zijn voor een bepaalde periode. Hierdoor kan het voorkomen dat de opdrachtgever niet bereikbaar is.
Kans	Middel
Kansverlagende maatregel	Als communicatiemiddelen zijn email en Slack beschikbaar. In Decos India is de opdrachtgever ook geregeld online waardoor er via deze manieren ook naar Inda gecommuniceerd kan worden en terug. Ook moet er een vaste dag of tijd ingesteld worden om te communiceren.
Impact	Groot
Impactverlagende maatregel	Voordat de opdrachtgever weg gaat samen een planning opstellen en mogelijke problemen bespreken. Een vervangende opdrachtgever aanstellen die beschikbaar is in de tijd dat de huidige opdrachtgever niet bereikbaar is
Plan B	Een vervangende opdrachtgever aanspreken en indien de huidige opdrachtgever niet bereikbaar is de vervangende opdrachtgever aanspreken

Nummer	**3**
Omschrijving	Hardware raakt defect
Risico	Een Cartracker of STN1170 raakt defect
Kans	Klein
Kansverlagende maatregel	1- Voordat er met de hardware gewerkt wordt, eerst de handleiding lezen. 2- De hardware alleen in een ESD veilige omgeving gebruiken
Impact	Groot
Impactverlagende maatregel	Meerdere Cartrackers en STN1170 beschikbaar hebben voor testdoeleinden
Plan B	Defecte component repareren. Mocht dit niet lukken moet een reserve Cartracker of STN1170 gebruiken De volgende hardware is als spare aanwezig: 3x Cartracker3 + STN1170 expansieprint aanwezig bij Decos 1x STN1170 expansieprint bij de leverancier

BIJLAGE C

Analysedocument

Analyse Document

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	:1.0

Inhoudsopgave

1.INLEIDING

Dit document beschrijft de analyse ten behoeve van de Cartracker met OBD ondersteuning. De analyse is opgedeeld in vier verschillende elementen:

- OBD-II
- Cartracker 3v4
- STN1170

Deze drie onderdelen zijn de belangrijkste onderdelen voor het realiseren van de Cartracker met OBD ondersteuning. De analyse is uitgevoerd zodat er voldoende kennis aanwezig is voor het realiseren van een goede architectuur. De gebouwde architectuur naar aanleiding van de analyse is te vinden in het architectuur document.

2. OBD II

Sinds 2001 is On Board Diagnostics versie II (OBD-II) verplicht in alle Europese benzineauto's(Baek and Jang 69-75)(Baek & Jang, 2015). Via het OBD-II systeem is het mogelijk om informatie van de Electronic Control Unit (ECU) uit te lezen. De OBD-II standaard beschrijft welke uitleesmogelijkheden minstens aanwezig moeten zijn. Een fabrikant is vrij in het uitbreiden van deze standaard met extra berichtgeving en functionaliteit maar hoeft de uitbreidingen niet te standaardiseren.

De OBD-II standaard beschrijft de volgende onderdelen:

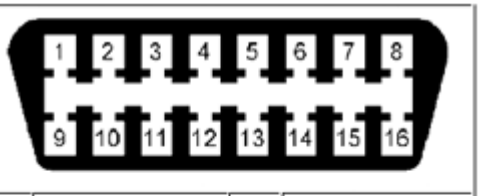
- Minimum op te vragen OBD berichten
- Toegestane bus protocollen
- OBD Connector

Het kan zijn dat niet alle onderdelen van de OBD standaard in een voertuig zijn geïmplementeerd. Het is niet noodzakelijk om alle onderdelen van de ISO 15031 te implementeren om een nuttige, gestandaardiseerd mechanisme voor uitstoot gerelateerde diagnose uit te voeren. Sommige onderdelen van de industrie kunnen onderdelen van de ISO15031 als irrelevant zien voor de benodigde standaardisatie. Voor voertuigen met alleen de basis diagnose hoeft er bijvoorbeeld geen veiligheid protocollen, gedefinieerd in de ISO 15031-7, geïmplementeerd te worden(ISO 15031-1 : 2010).

2.1. Connector

Een van de onderdelen van de OBD standaardisatie is de connector waarmee een diagnose analyse apparaat op de auto aangesloten kan worden.

De ISO 10531-3:2011 beschrijft de standaard pin-layout voor de OBD-II stekker als volgt:



PIN	DESCRIPTION	PIN	DESCRIPTION
1	Vendor Option	9	Vendor Option
2	J1850 Bus +	10	J1850 Bus –
3	Vendor Option	11	Vendor Option
4	Chassis Ground	12	Vendor Option
5	Signal Ground	13	Vendor Option
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 L-Line
8	Vendor Option	16	Battery Power

OBD-II Connector and Pinout

Figuur 5 OBD-II pin layout

Via de OBD stekker kunnen de volgende bus standaarden uitgelezen worden:

- SEA J1850 PWM(Pulse-width modulation)
- SEA J1850 VPW(Variable pulse width)
- ISO 9141-2
- ISO 14230 (Keyword Protocol 2000)
- ISO 15765 (CAN)
- SEA J1939-73 (CAN)

Een voertuig moet één van deze bus protocollen ondersteunen voor de OBD berichten. Welke bus hiervoor gebruikt wordt mag de fabrikant zelf bepalen. OBD test apparatuur moet alle standaarden ondersteunen. In de meeste gevallen wordt er door de fabrikant van dezelfde bus gebruik gemaakt in verschillende modellen.

2.2. OBD Data

OBD wordt door de ECU gecommuniceerd over één van de data bussen op de OBD connector. Om OBD informatie uit de auto te krijgen moet er via de OBD connector een verzoek bericht verstuurd worden. Na dit verzoek zal de ECU een antwoord terug sturen met de status.

Iedere personenauto met OBD-II biedt ondersteuning voor alle standaard parameter identification numbers (PID's) zoals beschreven in de ISO 15031-5:2011.

De OBD berichten zijn gesorteerd op verschillende modi, elke modus heeft een andere eigenschap. OBD-II heeft 10 verschillende modi waarin gecommuniceerd kan worden. Elke modus kan gebruikt worden voor een apart doeleinden. Zo biedt modus 01 alle huidige diagnostische data aan en kan in modus 09 statische voertuig informatie opgevraagd worden zoals het Voertuig Identificatie Nummer (vin nummer). Voor een compleet overzicht van alle modi met bijbehorende beschrijving zie Tabel 1 OBD-II modi.

Tabel 1 OBD-II modi

Mode	Beschrijving
01	Opvragen huidige diagnostische data
02	Opvragen frozen frame data (Bevroren op moment van storing optrede)
03	Opvragen uitstoot gerelateerde fout codes
04	Resetten van uitstoot gerelateerde informatie
05	Opvragen zuurstof sensor monitoring test resultaat
06	Opvragen on-board monitoring test resultaat voor specifiek systeem
07	Opvragen uitstoot gerelateerde fout codes gedetecteerd tijdens de huidige of laatste rit
08	Controle uitvoeren van on-board componenten en systemen
09	Opvragen voertuig informatie

0A	Opvragen van uitstoot gerelateerde fout codes met permanente status

Niet alle emissie gerelateerde informatie over de auto is relevant om in het Cartracker systeem te implementeren. Er een overzicht gemaakt van de meest nuttige OBD PIDs voor de Cartracker. Zie Tabel 2 voor het overzicht.

Tabel 2 Overzicht OBD codes

PID	Omschrijving	Return bytes	Formule
010d	Voertuig snelheid in Km/h Max 255	1	A
011F	Seconden sinds de motor gestart is Max 65,535	2	(A*256)+ B (AB = AB03 =) = 43779 sec
0121	Afstand afgelegd met de malfunction lamp aan in Km Max 65,535	2	(A*256) + B
0131	Afstand afgelegd sinds code reset in Km Max 65,535	2	(A*256)+B
0151	Brandstof type	1	Zie
015E	Engine fuel rate Max 3212,75 L/h	2	((A*256)+B)*0.05
0902	VIN	17-20	-

Tabel 3 Overzicht brandstof codes

Return value	Description
00	Not Available
01	Gasoline
02	Methanol
03	Ethanol
04	Diesel
05	LGPG
06	CNG
07	Propane
08	Electric

3. STN1170 ANALYSE

Om vanaf de Cartracker te communiceren met de OBD-II van het voertuig wordt als koppeling de STN1170 van Scantool gebruikt. Dit is een OBD naar UART vertaler die alle standaard OBD-II protocollen ondersteund. De STN1170 communiceert over UART en kan drie soorten commando's verwerken: AT commando's , ST commando's en OBD vragen.

De STN1170 is gebaseerd op de ELM327(ELM Electronics, 2014), een OBD naar RS232 vertaler met AT commando ondersteuning.

3.1. Communicatie met de STN1170

De STN1170 communiceert over UART met de volgende instellingen:

- 38400 baud (Dev board is standaard 9600)
- 8 data bits
- Geen pariteit bit
- Eén stop bit
- Geen handshake

Als de STN1170 verbonden en opgestart is wordt het volgende weergegeven in de console:

ELM327 v1.3a

>

Het '>' teken geeft aan dat de STN1170 input kan verwerken. Alle antwoorden van de STN1170 worden afgesloten met een carriage return. Hierna is het weer mogelijk om input te geven.

3.2. Commando's

De STN1170 ondersteund alle ELM327 AT commando's.

Als uitbreiding hierop heeft Scantool een aantal ST commando's toegevoegd.

De ST commando's geven de mogelijkheid om de PowerSave functie te gebruiken en extra filters toe te passen. Voor een overzicht van alle beschikbare commando's zie de STN11000 Family Reference and Programming Manual(OBD Solutions, 2009).

3.3. OBD gegevens opvragen

Een gewenste code kan opgevraagd worden door de mode gevolgd voor de PID naar de

Engine control unit(ECU) te sturen.

Om bijvoorbeeld het toerental van een auto op te vragen wordt de volgende code gestuurd:

010C

01 is mode 01

0C is de PID voor het toerental

Een antwoord van de ECU kan er als volgt uit zien:

41 0C 1A FA

41: opgevraagde mode + 40 (1 + 40)

0C: Herhaling van opgevraagde PID

1A FA: Toerental in HEX = 6906 in decimaal

6906 is niet het daadwerkelijke toerental. In de OBD-II PID tabel is de berekening te lezen. Voor het toerental is het het antwoord delen door vier. **6906 / 4 = 1726,5 rpm**

3.4. CAN-Bus

Buiten de standaard OBD berichten is het via de STN1170 ook mogelijk om direct CAN-bus berichten uit het CAN-bus systeem van de auto uit te lezen.

Controller Area Network is een serieel communicatie protocol, welke gedistribueerde real-time controle en multiplexing ondersteund voor weg voertuigen en andere controle applicaties(Nederlands Normalisatie-instituut).

CAN is in 1993 als standaard protocol uitgebracht in de ISO 11898(ISO 11898). Deze documentatie behandelde de CAN data link laag en de high-speed fysieke laag. De nieuwe ISO11898 is opgedeeld in 6 delen waarin verschillende onderdelen van het CAN protocol wordt behandeld.

Alle CAN-Bus berichten worden gebroadcast door het netwerk(Nederlands Normalisatie-instituut). Deze CAN-Bus berichten kunnen via filtering van de STN1170 doorgestuurd worden naar de Cartracker.

Voor de CAN-bus berichten zijn geen standaard PIDS over welke informatie welk PID moet dragen zoals bij OBD. Hierdoor moet het achterhalen van de PIDS op de CANBus gedaan worden door reverse engineering.

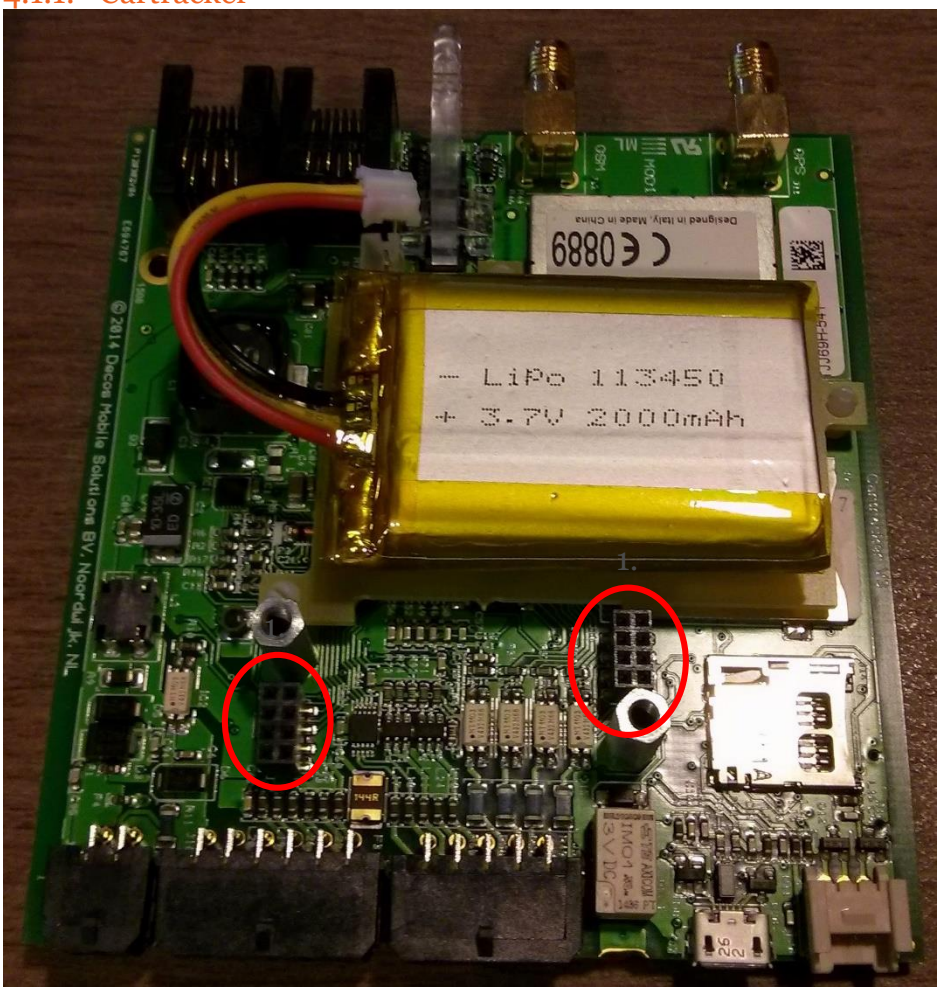
4. CARTRACKER ANALYSE

In dit hoofdstuk wordt de analyse van de Cartracker 3 Versie 4 beschreven. De analyse is in twee delen opgedeeld, hardware en software. In de hardware analyse wordt er gekeken naar de componenten die betrekking hebben op de implementatie van het OBD systeem. In de software analyse wordt er een globale analyse gegeven van het operating system. Verder wordt er een gedetailleerde beschrijving gegeven van de onewireTask. Deze taak zal gebruikt worden voor de uitbreiding van het OBD systeem.

4.1. Hardware

In deze paragraaf worden twee hardware onderdelen van de Cartracker met OBD ondersteuning uitgelicht. Als eerste wordt de Cartracker uitgelicht en hierna de STN1170 Extension print.

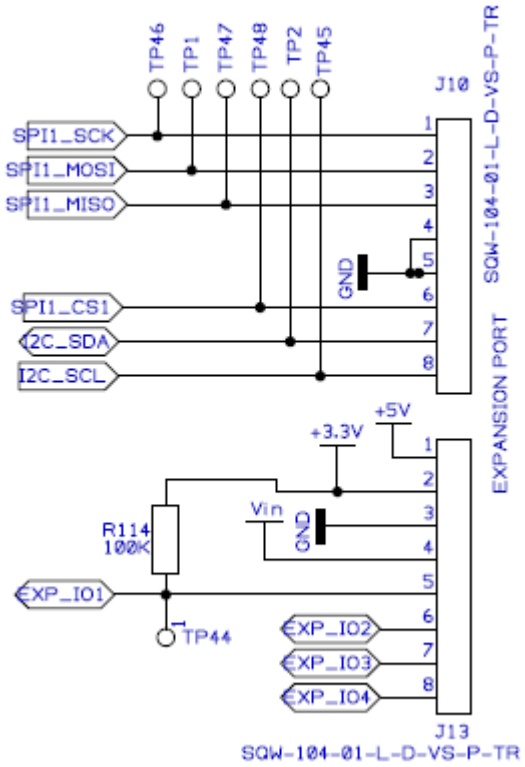
4.1.1. Cartracker



Figuur 6 Cartracker3 V4 PCB

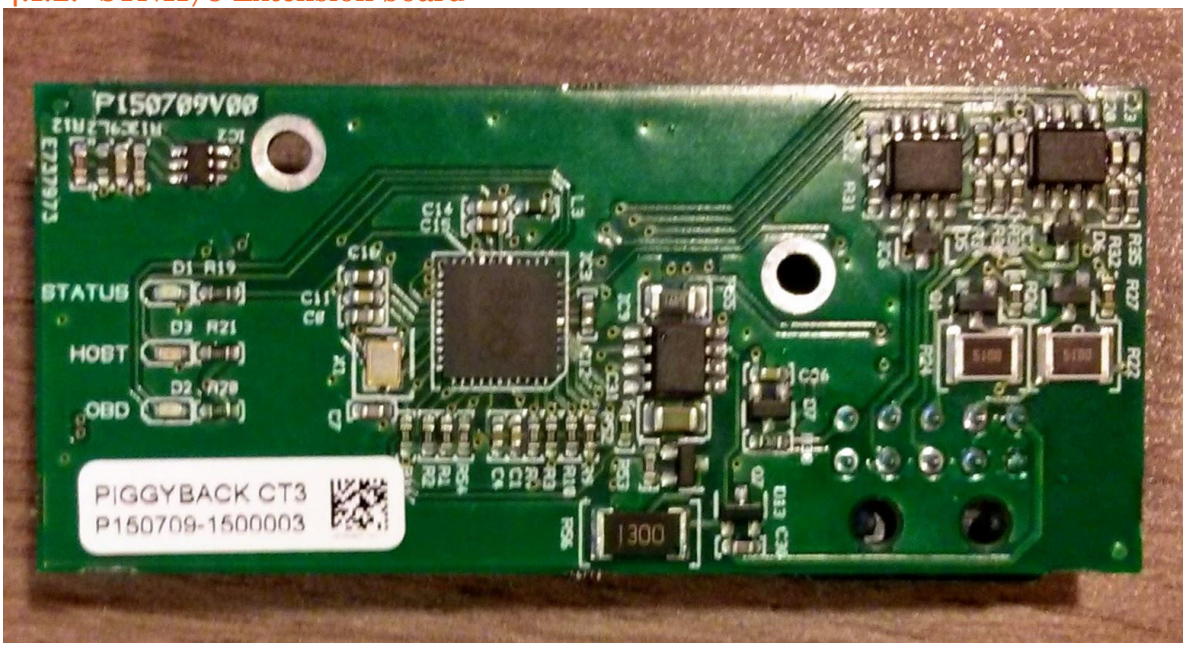
De Cartracker wordt aangestuurd door een Atmel AT32UC3A. Dit is een 32-bit AVR Microconroller Unit(MCU). Op de MCU draait FreeRTOS, een gratis embedded Real Time Operating System(RTOS). In paragraaf 4.2 wordt hier meer over uitgelegd. De PCB is voorzien van twee extension headers, nummers 1 in Figuur 6. Via de extension headers is het mogelijk de STN1170 extension board op de Cartracker aan te sluiten. Via de Serial Peripheral Interface (SPI) kan er gecommuniceerd worden.

In de schematische tekening van de Carttracker is te zien dat de SPI1 bus op de extentie headers is aangesloten. Zie Figuur 7 voor de schematische tekening.



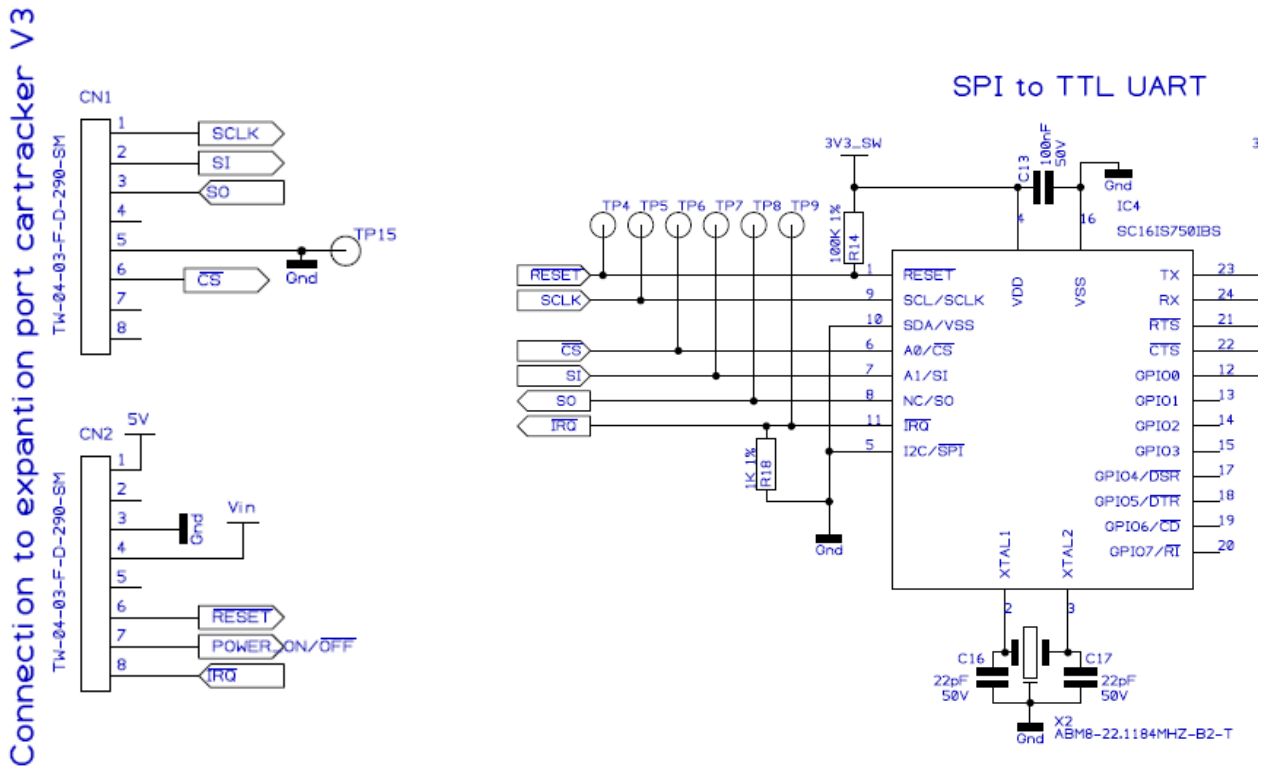
Figuur 7 Schema Cartracker 3v4 expansion ports

4.1.2. STN1170 Extension board



Figuur 8 STN1170 Extension board

De belangrijkste componenten van het STN1170 Extension board zijn de STN1170 en de SPI to UART. De SPI verbinding vanaf de Cartracker gaat eerst naar de SPI to UART, welke de SPI berichten om zet naar UART. De rede hiervoor is is dat de STN1170 alleen via UART kan communiceren met andere componenten. In Figuur 9 is te zien dat de header van de Cartracker aan de SPI van de SPI to UART chip is verbonden.



Figuur 9 Extension board headers

De STN1170 communiceert via de universal asynchronous receiver/transmitter(UART) en niet via SPI zoals de Cartracker. Om deze communicatie toch mogelijk te maken is er een SC16IS750, een SPI naar UART translater, op het extension board geplaatst. Deze maakt de communicatie tussen de Cartracker MCU en de STN1170 mogelijk.

4.2. Software

In deze paragraaf wordt de huidige software van de Cartracker geanalyseerd op globaal niveau. De werking van FreeRTOS wordt besproken en de verschillende taken die op de Cartracker draaien.

4.2.1. FreeRTOS

FreeRTOS is een Real Time Operating System (RTOS) wat klein genoeg is om op een microcontroller te draaien(Richard Barry). FreeRTOS maakt bij het toekennen van processortijd gebruik van tasks. Omdat in FreeRTOS alle functies worden uitgevoerd binnen een task wordt er op de werking van een task ingegaan in dit document. Niet alle mogelijkheden van FreeRTOS worden op dit detail niveau uitgelegd.

Een task in FreeRTOS werkt door middel van states en kan zich in vier verschillende states bevinden.

- Running

Als de taak wordt uitgevoerd bevind deze zich in de Running state

- Ready

In de Ready state kan een taak worden uitgevoerd, maar wordt dit op dit moment niet gedaan omdat er een andere taak wordt uitgevoerd.

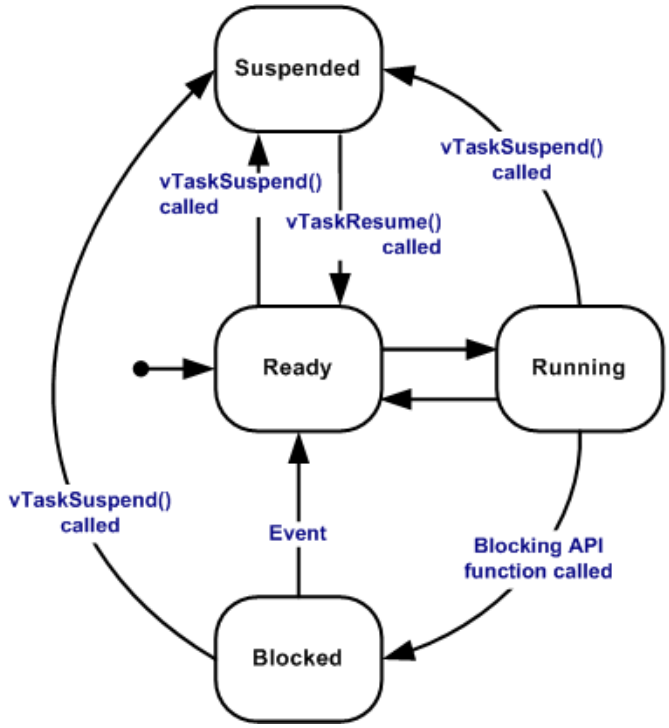
- Blocked

Een taak kan zich in de Blocked state bevinden als de taak moet wachten, of door middel van een vTaskDelay() of omdat de taak moet wachten op een event, zoals een queue of een semaphore.

- Suspended

Een taak kan in de Suspend state komen door het commando vTaskSuspend() en kan alleen terug naar de Ready state door middel van het vTaskResume() commando. Het verschil met de Blocked state is dat de task niet met behulp van een event naar de Ready state kan gaan maar alleen via het vTaskResume() commando.

Zie Figuur 10 voor het state diagram van een FreeRTOS task.



Figuur 10 FreeRTOS Task state diagram

Het aanmaken en activeren van een task gebeurt in twee verschillende stappen. Stap één is het aanmaken van de task, stap twee is het toevoegen van de task aan de scheduler.

Een task ziet er hetzelfde uit als een functie in c, zie Code snippet 1

```
void myTask( void *
pvParameters)

{

        for ( ;; )

        {

                // Task code

        }

}
```

Code snippet 1 FreeRTOS Task

Een task mag nooit returnen en wordt daarom altijd uitgevoerd met een oneindige loop. Om een task ook uitvoerbaar te maken moet deze aangemaakt en toegevoegd worden aan de task scheduler. Zie voor een voorbeeld met myTask Code snippet 2. De myTask uit Code snippet 1 wordt aan de task scheduler toegevoegd via de xTaskCreate functie. Hierbij worden de volgende parameters meegegeven:

- pvTaskCode: Pointer naar de task functie
- pcName: Naam van de functie, voornamelijk gebruikt voor debuggen
- uStackDepth: de grootte van de task stack waarin de variabelen van de task in kunnen
- pvParameters: pointer naar parameters die meegegeven kunnen worden bij het aanmaken van de task
- uxPriority: De prioriteit van de task

```
static unsigned char
ucParameterToPass;

Taskhandle_t xHandle =
NULL;



xTaskCreate(myTask,
"myTaskName",1000,&ucPar
ameterToPass,
tskIDLE_PRIORITY,&xHand
le);
```

Code snippet 2 FreeRTOS Task toevoegen

De task scheduler bepaald aan de prioriteit van een task wanneer deze processortijd krijgt. Een task met een hogere prioriteit heeft voorrang op task met een lagere prioriteit. De prioriteit wordt bij het aanmaken van een task meegegeven in xTaskCreate functie.

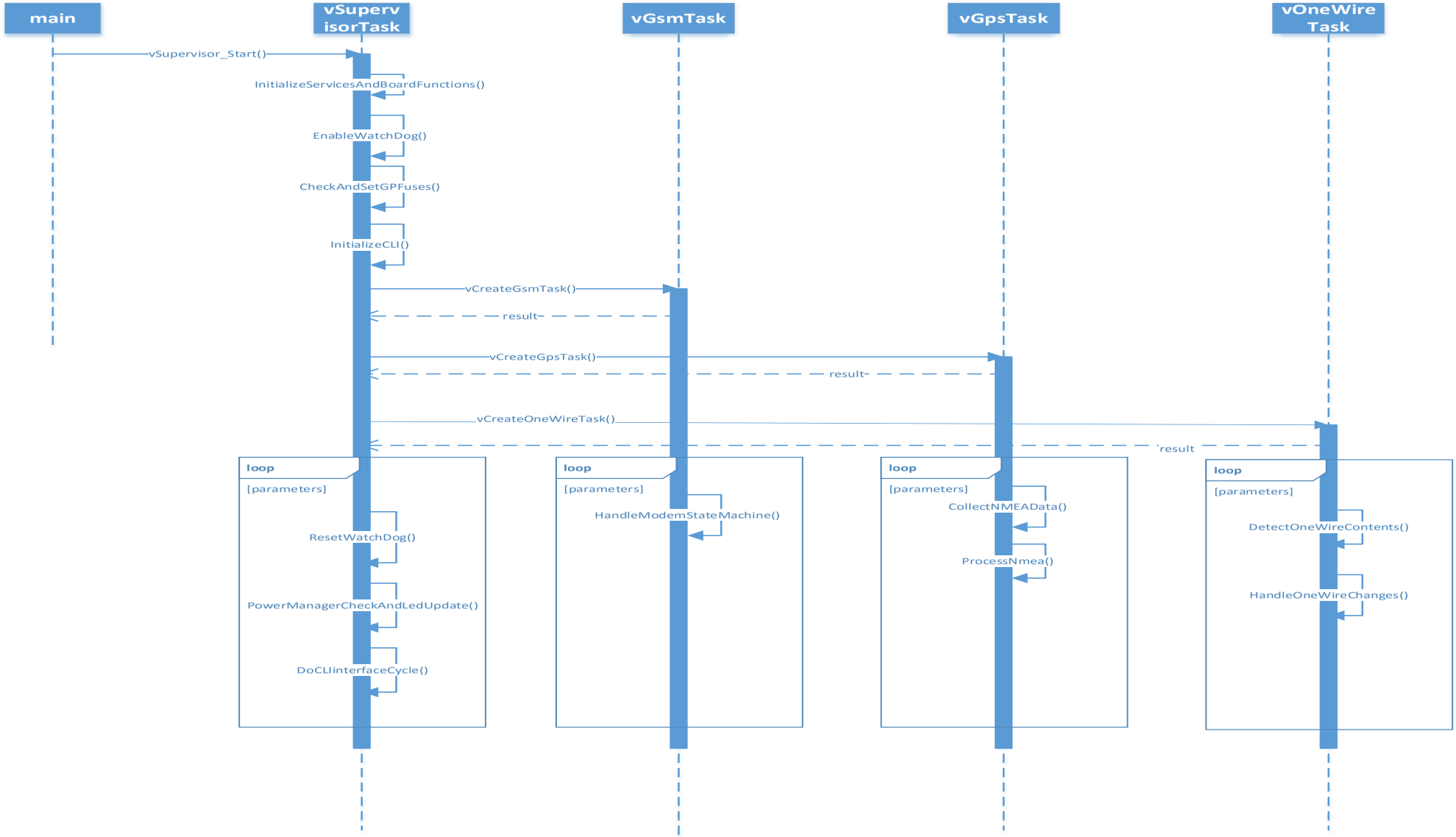
FreeRTOS maakt gebruik van multitasking, waarbij de scheduler tussen de taken wisselt.

4.2.2. FreeRTOS in de Cartracker

Binnen het FreeRTOS platform in de Cartracker wordt er gebruik gemaakt van een viertal taken. De volgende taken zijn aanwezig:

- SupervisorTask: Initialiseerd de verschillende hardware componenten en creëert de overige tasks
- GsmTask: Verzorgt de communicatie met het modem voor het versturen van berichten naar de Cartracker server
- GpsTask Handelt de communicatie met de GPS module af
- OneWireTask Zorgt voor alle communicatie op de OneWire draad, welke gebruikt wordt voor onder andere de iButtons

Op de volgende pagina is het sequentiediagram te vinden wat vanuit de Cartracker code is opgesteld. Hierin zijn de tasks te vinden met hun globale werking in de vorm van functies die uitgevoerd worden binnen de taak.


main
vSupervisorTask
vGsmTask
vGpsTask
vOneWireTask
vSupervisor_Start()
InitializeServicesAndBoardFunctions()
EnableWatchDog()
CheckAndSetGPFuses()
InitializeCLI()
vCreateGsmTask()
result
vCreateGpsTask()
result
vCreateOneWireTask()
result
loop
[parameters]
ResetWatchDog()
PowerManagerCheckAndLedUpdate()
DoCLIinterfaceCycle()
loop
[parameters]
HandleModemStateMachine()
loop
[parameters]
CollectNMEAData()
ProcessNmea()
loop
[parameters]
DetectOneWireContents()
HandleOneWireChanges()

5. PROTOTYPE

5.1. Experiment

De OBD applicatie zal via de STN1170 op CANbus en OBD berichten berichten kunnen filteren. Standaard worden alle berichten geblokkeerd door de STN1170. Voor het experiment zal er gekeken worden of het mogelijk is om OBD berichten te ontvangen en CAn bus berichten te ontvangen. Het CAN bus filter zal worden uitgezet op de STN1170 waardoor alle CANbus berichten zullen worden doorgelaten. Alle berichten worden gelogd en opgeslagen door de applicatie. Na het loggen worden de CANbus berichten gesorteerd op PID, code en aantal.

Via een experiment wordt er gekeken of het mogelijk is om via reverse engineering CANbus PID's te achterhalen voor de Nissan Leaf.

Op het Nissan Leaf forum zijn een aantal PIDS al achterhaald(Nissan leaf Forum). Met deze gegevens is er geprobeerd om een aantal van deze PIDS te verifiëren met eigen metingen.

Via de C# applicatie wordt de informatie uit de STN1170 in een tekst bestand gelogd welke door de C# applicatie hierna geanalyseerd kan worden.

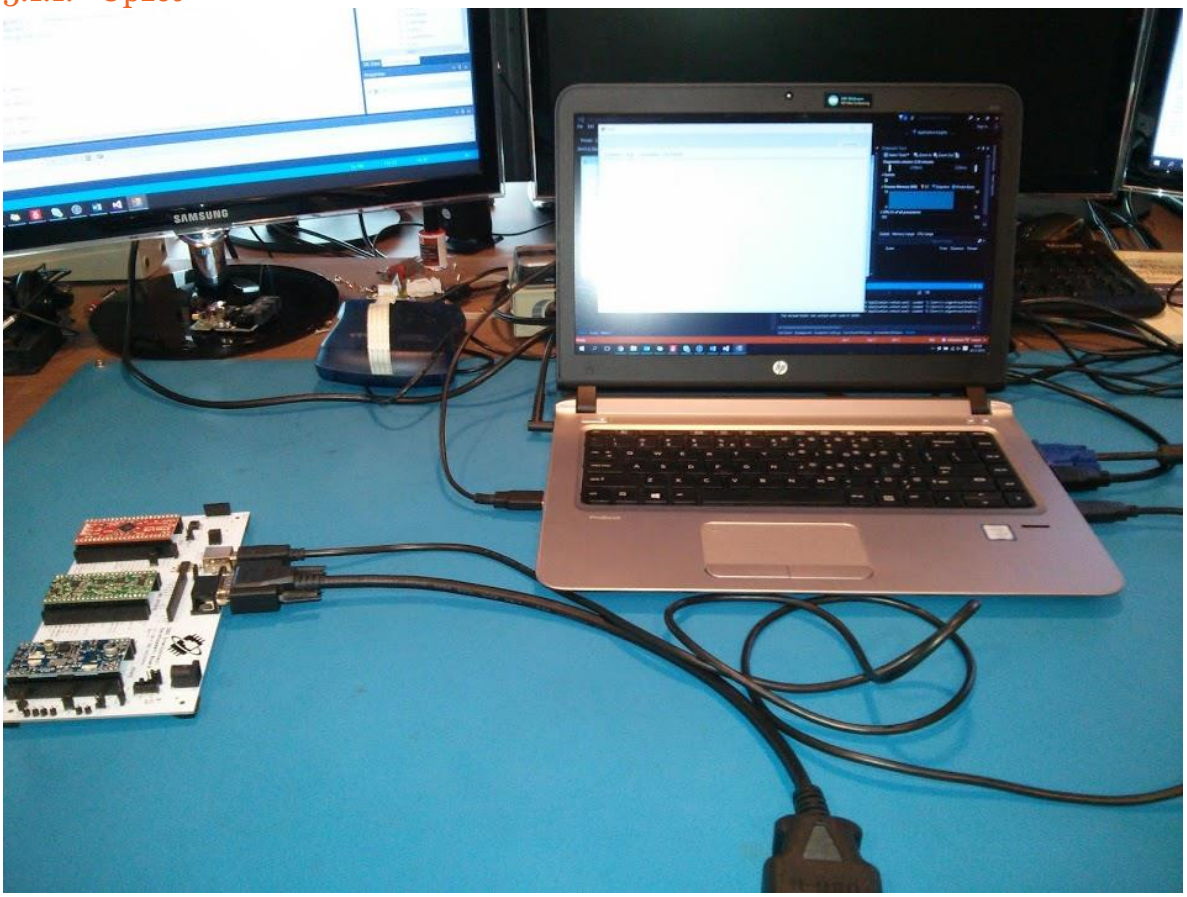
De log kan worden ingelezen en gesorteerd worden in de applicatie. Er wordt weergegeven hoe vaak een bepaald CAN bericht voorbij is gekomen en met welke data. Als de data hetzelfde is als bij een vorig bericht wordt deze niet apart weergegeven maar wordt bij het al bestaande PID opgeteld.

De volgende PIDS zijn gekozen om te verifiëren:

PID	Beschrijving
280	Bestuurders gordel
60D	Signalering lichten
5c5	Odometer

Het doel van het experiment was om via de STN1170 en een PC, OBD berichten en CAN bus berichten kunnen opvragen en filteren uit een auto met OBD ondersteuning.

5.1.1. Opzet



Het STN1170 Development board zit via de RS232-OBD-II stekker aan de auto gekoppeld. Via de USB kabel is het development board aan de laptop gekoppeld. Via de OBD applicatie kan er met de OBD van de auto gecommuniceerd worden.

De volgende commando's zijn gebruikt tijdens het experiment:

Commando	Uitleg
ATI	Print ELM327 versie ID string
ATH1	Zet header informatie aan
STSBR	Zet baud rate
STFAP	Zet pass filter

STFCP	Verwijder alle pass filters
STM	Start monitoring met filter instellingen
ATP	Zet bus protocol

5.1.2. Data analyse

Het verzamelen van CAN data is uitgevoerd aan de hand van het loggen van de resultaten uit de STN1170.

```
Log Entry : 16:09:43 dinsdag 15 maart 2016
------------------------------
>ati                                Timestamp ms:245640
ELM327 v1.3a                       Timestamp ms:245657
                                   Timestamp ms:245735
>ati                                Timestamp ms:245768
ELM327 v1.3a                       Timestamp ms:245784
                                   Timestamp ms:245797
>                                  Timestamp ms:245813
ELM327 v1.3a                       Timestamp ms:245843
                                   Timestamp ms:245859
>stfcp                             Timestamp ms:245878
OK                                 Timestamp
ms:245891
>stfap 280,FFF                     Timestamp ms:245938
OK                                 Timestamp
ms:245954
>stm                               Timestamp ms:246002
280 01 FF C0 00 00 00 FF C0        Timestamp
ms:246018
280 01 FF C0 00 00 00 FF C0        Timestamp
ms:246034
280 01 FF C0 00 00 00 FF C0        Timestamp
ms:246050
280 01 FF C0 00 00 00 FF C0        Timestamp
ms:246082
280 01 FF C0 00 00 00 FF C0        Timestamp
ms:246098
```

Figuur 11 STN filter instellen

280 01 FF C0 00 00 00 FF C0 Timestamp ms:26700

280 01 FF C0 00 00 00 FF C0 Timestamp ms:26716

280 01 FF C0 00 00 00 FF C0 Timestamp ms:26732

280 01 FF C0 00 00 00 FF C0 Timestamp ms:26748

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:26862

280 03 FF C0 00 00 00 FF C0 Timestamp ms:26877

...

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

Figuur 12 CAN bus filter data verandering

Tabel 4 CAN bus Nissan Leaf Resultaten

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 Turnsignal light off 0x26 Left turn signal light 0x46 Right turn signal light 0x66 gevarenlicht						
180			Gaspedaal positie ???					
176		Vehicle speed???						Counter 0x00 - 0x0F
175				AA: Neutral BB: Drive 99 Reverse	Counter 0x00 - 0x0F			
355		Speed		Speed				
5C5		ODOMETER						

5.1.3. Conclusie

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 is aangetoond dat PID 5C5 byte 1 t/m 4 correspondeert met de odometer. Dit is met de kijk op de Cartracker implementatie de belangrijkste PID. 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 de zelfde informatie uit de Nissan Leaf te achterhalen.

Works Cited

1. Baek, Sung-hyun, and Jong-Wook Jang. "Implementation of Integrated OBD-II Connector with External Network." *Information Systems* 50 (2015): 69-75. Print.

2. ELM Electronics. "ELM327 OBD to RS232 Interpreter." 2014. Web. 23-2-2016 <http://www.elmelectronics.com/DSheets/ELM327DS.pdf>.

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. Nederlands Normalisatie-instituut. "Road Vehicles - Controller Area Network (CAN) - Part 1: Data Link Layer and Physical Signalling (ISO 11898-1:2015,IDT)." (2015)Print.

BIJLAGE D

Architectuurdocument

Architectuurdocument

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	: 1.0

Inhoudsopgave

1.INLEIDING

Dit document beschrijft de architectuur gemaakt voor het project Cartracker met OBD ondersteuning.

Als eerste worden de eisen en wensen van de opdrachtgever.

Hierna volgt de uitwerking van de eisen in de vorm van use-case beschrijvingen en sequentiediagrammen.

De eisen en het ontwerp van het prototype zijn te vinden in het laatste hoofdstuk.

2. ARCHITECTUUR EISEN

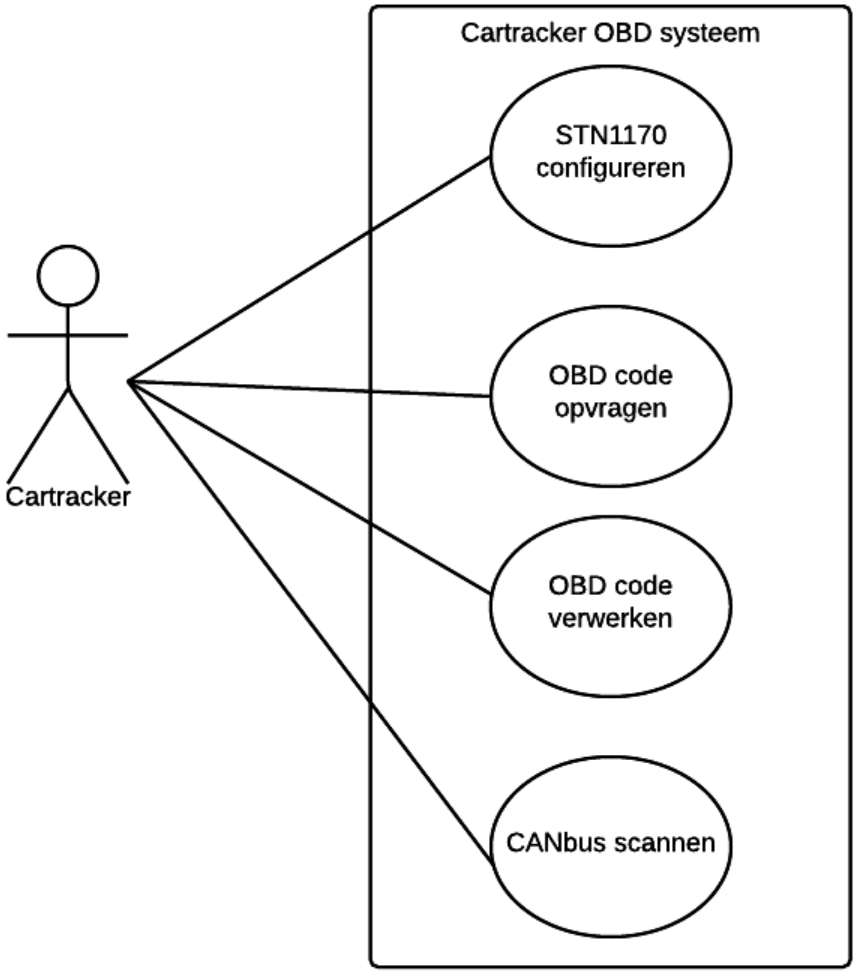
Dit hoofdstuk beschrijft de eisen die betrekking hebben op het OBD-implementatie in de Cartracker. Het hoofdstuk is opgedeeld in de functionele en niet-functionele eisen.

Niet functionele eisen

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 OBD functionaliteit moet voor alle auto's met OBD ondersteuning moeten werken

2.1. Functionele eisen

De functionele eisen worden gerepresenteerd door middel van use cases. Deze use cases zijn opgesteld in het Visiondocument. In deze paragraaf worden de use cases in meer detail uitgewerkt



Figuur 13 Use case diagram

2.2. Use case beschrijvingen

Naam	**STN1170 configureren**
Actor	Cartracker
Aannamen	De Cartracker is met de juiste OBD configuratie geconfigureerd De STN1170 is nog niet geconfigureerd
Beschrijving	De STN1170 wordt geconfigureerd 3- Juiste commando's worden ingevoerd 4- Cartracker krijgt melding dat de STN correct is geconfigureerd
Uitzonderingen	
Resultaat	De STN1170 is correct geconfigureerd

Naam	**OBD bericht opvragen**
Actor	Cartracker
Aannamen	De STN1170 is correct geconfigureerd
Beschrijving	7- De Cartracker stuurt een request naar de STN1170 8- De STN1170 stuurt een request over de OBD bus naar de auto 9- De auto stuurt het antwoord naar de STN1170 10- De STN1170 controleert de geldigheid van het OBD bericht. Indien het bericht niet geldig is treedt uitzondering 1 op. 11- De STN1170 stuurt het OBD bericht door naar de Cartracker 12- De Cartracker ontvangt het opgevraagde bericht

Uitzonderingen	2- De STN1170 verstuurt het verkeerd ontvangen bericht niet en gaat terug naar stap 3
Resultaat	De opgevraagde code is ontvangen

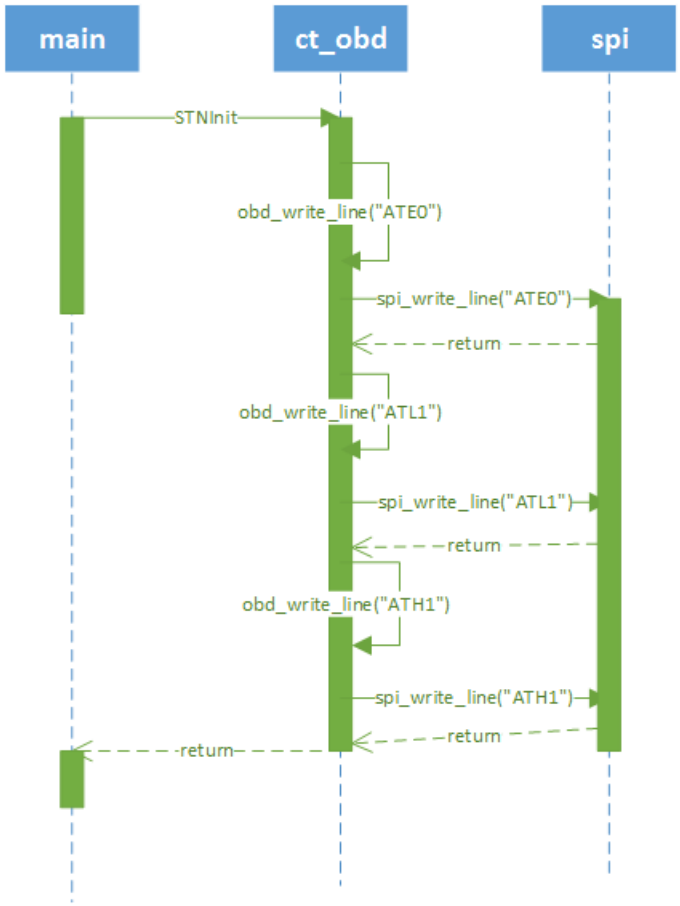
Naam	**Bericht verwerken**
Actor	Cartracker
Aannamen	Er is een bericht opgevraagd en ontvangen
Beschrijving	3- De Cartracker interpreteert het ontvangen bericht 4- De informatie in het bericht wordt verwerkt
Uitzonderingen	
Resultaat	Het ontvangen bericht is geïnterpreteerd en verwerkt in de Cartracker

Naam	**CANbus scannen**
Actor	Cartracker
Aannamen	
Beschrijving	5- De Cartracker stuurt een filterspecificatie naar de STN1170 6- De gefilterde berichten worden vanaf de STN1170 doorgestuurd naar de Cartracker
Uitzonderingen	
Resultaat	Er is een gefilterd CAN bericht ontvangen op de Cartracker

3. USE CASE UITWERKINGEN

Aan de hand van de use case beschrijvingen is het ontwerp verder verfijnd aan de hand van flowcharts. Deze flowcharts dienen als basis voor de uiteindelijke code die geschreven gaat worden tijdens de Construction fase. Iedere use case is uitgewerkt in een apart messagediagram

3.1. STN1170 Configuratie

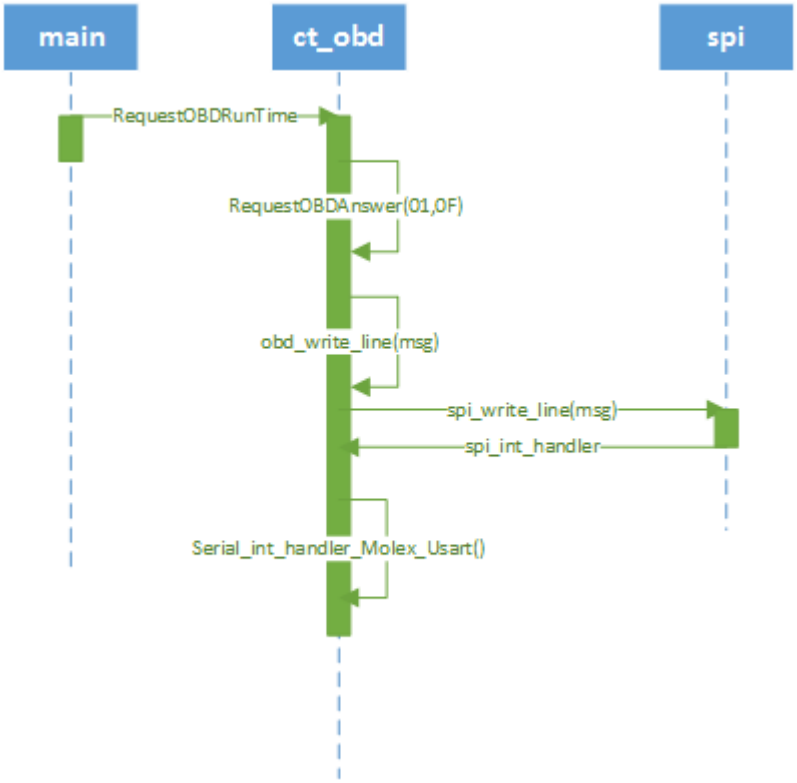


Figuur 14 Flowchart STN1170 configuratie

De Cartracker stuurt configuratie commando's over de SPI bus naar de STN1170. De STN1170 antwoord over de SPI bus terug naar de Cartracker. Na alle configuratie commando's gestuurd te hebben is de STN1170 geconfigureerd voor verder gebruik.

3.2. OBD bericht opvragen

Standaard worden niet alle OBD statussen over het netwerk gebroadcast. De codes moeten opgevraagd worden aan de Electronic Control Unit(ECU) waarna deze een antwoord terugstuurt.



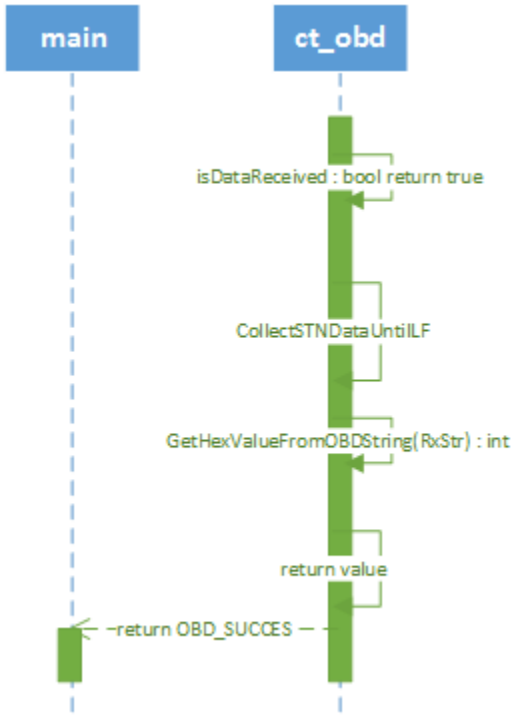
Figuur 15 Messagediagram OBD bericht opvragen

De Cartracker vraagt een OBD bericht op aan de auto.

Het op te vragen OBD bericht wordt op de SPI verzonden waarna er een antwoord in de spi_interrupt_handler binnen komt. Hierna kan het opgevraagde bericht verwerkt worden.

3.3. Verwerk bericht

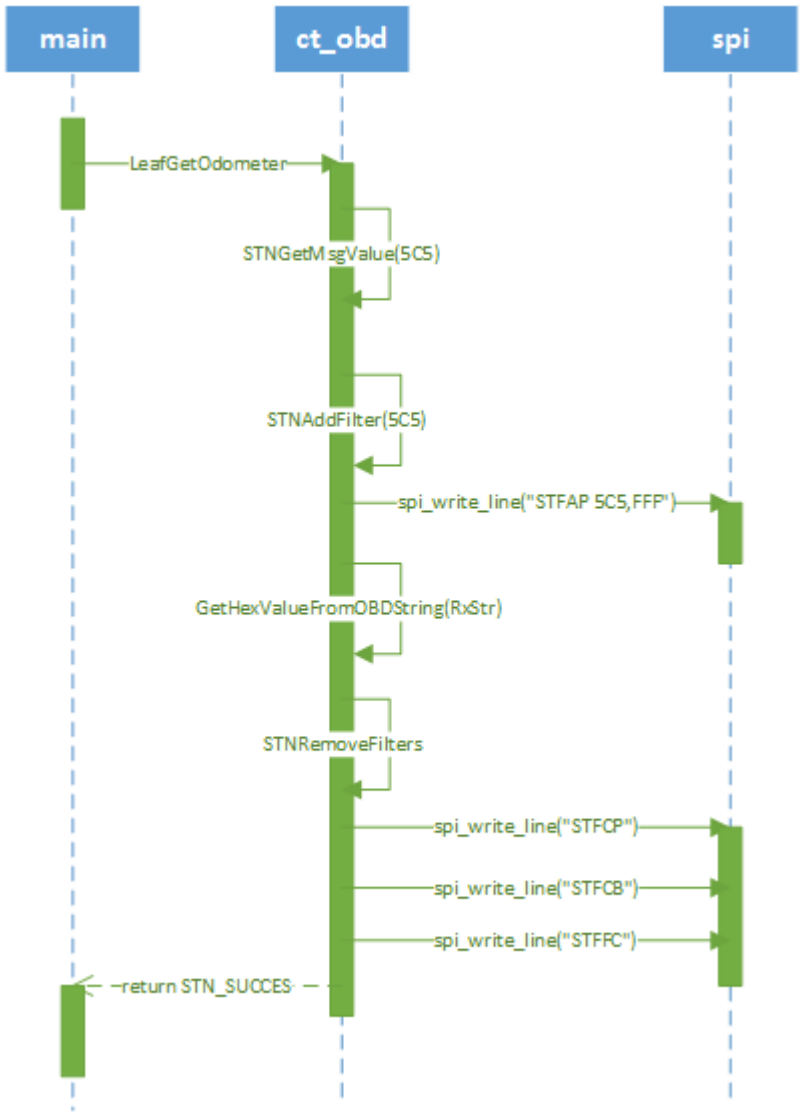
In de verwerk bericht use case wordt het bericht wat ontvangen is verwerkt. In eerste instantie wordt het ontvangen bericht gedecrypt en de nuttige informatie uit het bericht gehaald en vertaald.



Figuur 16 Messagediagram OBD bericht verwerken

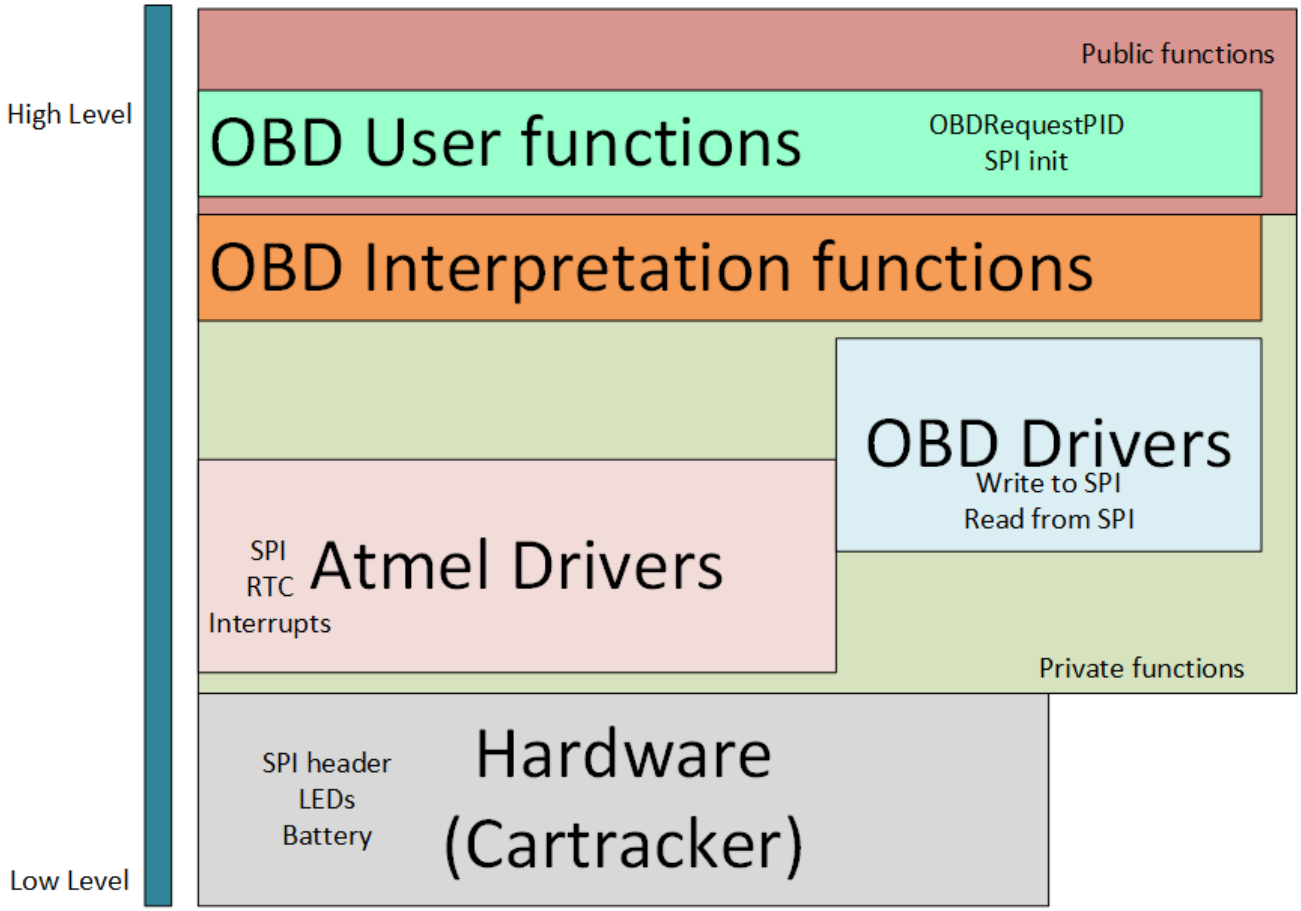
3.4. CAN bus scannen

De CANbus berichten zijn voor ieder model auto verschillend. Hierdoor is het belangrijk om specifieke filters toe te passen op CANbus berichten. De eerste stap van het scannen naar CANbus berichten is het toepassen van filters. De filters worden op PID gefilterd. Als alle filters zijn ingesteld wordt er via het STM commando gestart met monitoren. Als er een bericht door het filter komt wordt deze naar de Cartracker gestuurd. Vanaf hier kan de Cartracker de nuttige informatie uit het bericht halen en verwerken.



4. ABSTRACTIE LAGEN

Om een beeld te geven welke verschillende lagen zich in de code gaan bevinden is er een software abstractie tekening gemaakt. In **Error! Reference source not found.** zijn de verschillende lagen van de OBD implementatie te zien.



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 al standaard in de software aanwezig. 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. De private functions worden alle afhandelingen en vertalingen van de OBD berichten uitgevoerd.

5. PROTOTYPE

Om de eerste stappen met de STN1170 te maken is er een prototype applicatie geschreven waarmee OBD en CAN bus berichten via de COM port op een Windows machine getoond kunnen worden. De resultaten uit de applicatie worden gebruikt om de data die op de Cartracker wordt weergegeven te verifiëren.

Het doel van het prototype is om te controleren of een auto de OBD standaard ondersteund en of er nog meer CAN berichten uit de auto gefilterd kunnen worden.

Het prototype is op twee elektrische voertuigen getest. De Nissan Leaf en de BMW i3. Beide auto's zijn voorzien van een OBD stekker ook hoeven ze deze volgens de regels niet te hebben omdat een OBD stekker alleen voor auto's die uitstoot veroorzaken verplicht is.

5.1. Ontwerp

Om de onderzochten gegevens over de OBD communicatie en de STN1170 is er besloten om een prototype applicatie te maken waarin de verschillende onderdelen van communicatie getest en gelogd kunnen worden. De applicatie is in C# geschreven met Visual Studio 2015. Aan de hand van het klasse diagram zullen de mogelijkheden van de applicatie uitgelegd worden. Met deze applicatie zijn een aantal testen uitgevoerd met het STN1170 Development board

Eisen

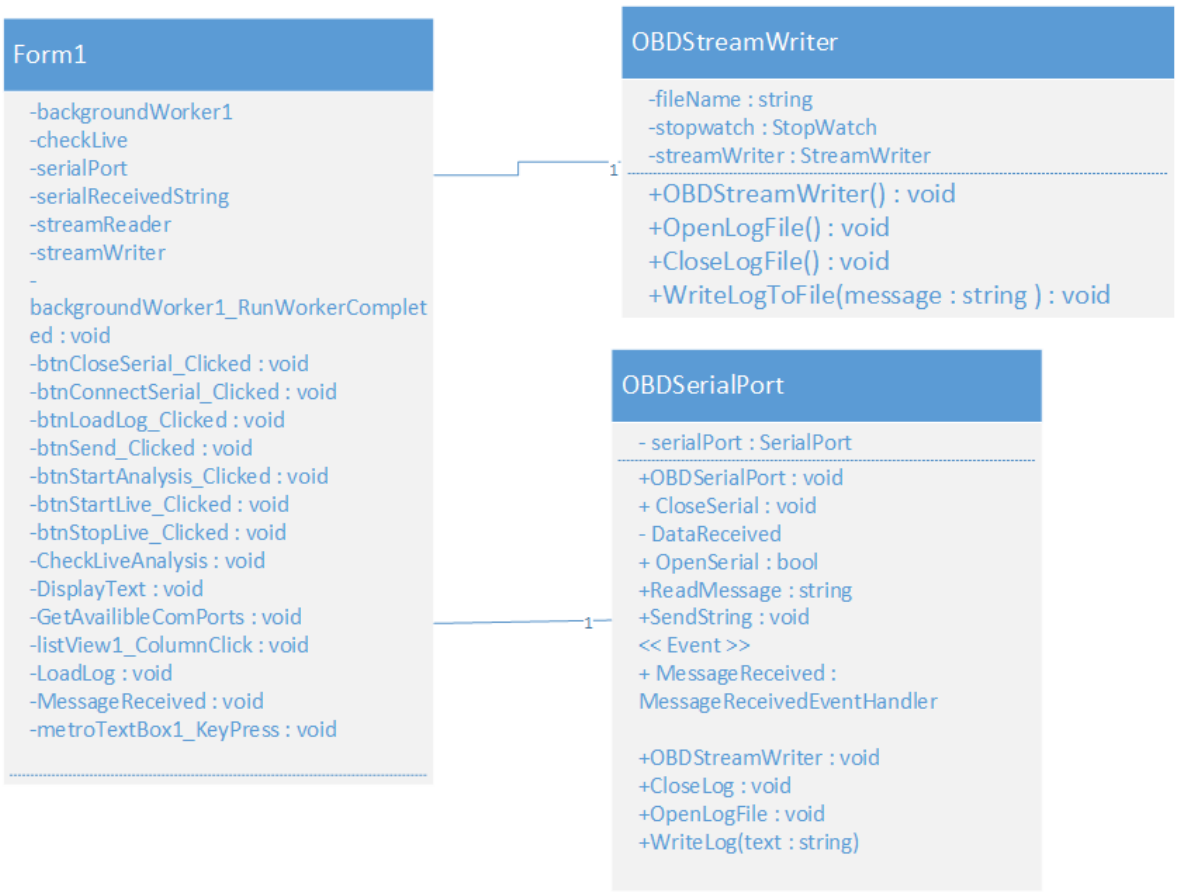
De volgende eisen zijn gesteld aan het ontwikkelen van het prototype:

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

5.2. Ontwerp

De OBD applicatie bestaat uit twee klasse. De Form klasse waar alle UI in afgehandeld wordt en de OBDSerialPort klasse waarin de seriële verbinding met het STN Development board.



Figuur 17 Klassendiagram prototype

5.3. Visualisatie

De OBD applicatie bestaat uit een viertal tabs.

- Connection
  De COMport en de baudrate moeten hier ingesteld worden
- Serial
  In deze tab kan er door middel van de seriële verbinding met de STN1170 gecommuniceerd worden. De verbinding kan hier opgezet en afgebroken worden. Via het tekstinvoer veld kunnen er commando's naar de STN1170 verstuud worden. De reacties van de STN1170 zijn terug te zien in de richtextbox. De informatie die hier op wordt weergegeven wordt automatisch in een log bestand opgeslagen.

- Log analysis
  De logbestanden van de STN1170 communicatie kunnen op CAN bus berichten geanalyseerd worden. De berichten kunnen in volgorde gezet worden van PID waarbij het aantal dezelfde berichten en de bijbehorende data weergegeven wordt.

- Live analysis
  De laatste tab kan gebruikt worden tijdens het loggen. In deze tab is real-time te zien welke CAN bus berichten er door de STN1170 doorgegeven worden. Hierbij is ook te zien hoe vaak een bericht het dit PID voorbij is gekomen en wat de laatste data van dit PID is.

BIJLAGE E

Ontwerpdocument iteratie 1

Ontwerpdocument
Iteratie 1

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	: 1.0

Inhoudsopgave

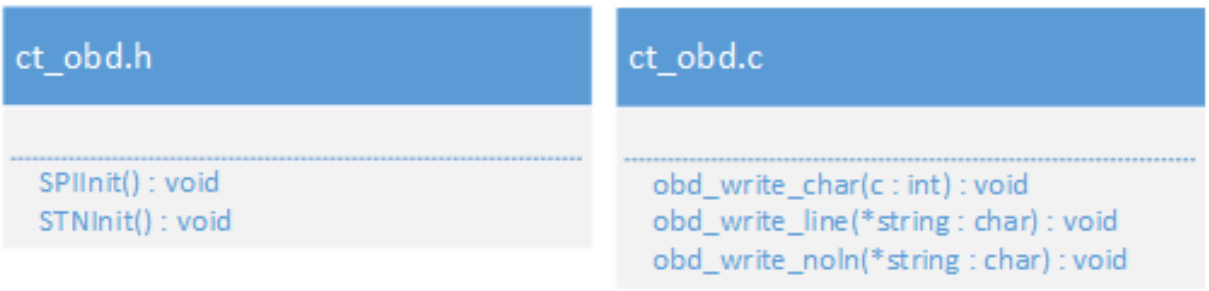
1 INLEIDING

Dit is het ontwerpdocument voor iteratie 1 van het project Cartracker met OBD ondersteuning. In dit document is de eerste use-case,STN1170 configureren, uitgewerkt in detail en omzet naar code. Er is geen testplan of vervolg iteratie geweest op deze iteratie vanwege het feit dat de hardware waarmee gewerkt zou gaan worden niet de juiste werking had. Hierdoor is er gekozen om een nieuwe iteratie te beginnen.

2 ONTWERP

Het eerste ontwerp van deze iteratie is gefocust op de vertaling van SPI naar UART. De SPI bus moet op de Cartracker geinitialiseerd worden. De microcontroller van de Cartracker is voorzien van twee SPI bussen waarop meerdere slave devices per bus op aangesloten kunnen worden. Het is dus van belang om in het ontwerp de juiste bus te selecteren.

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 als input of als uitput gebruikt kan worden. Tijdens deze initialisatie moeten ook de SPI pinnen gedefineerd worden. Na de initialisatie van de pinnen kan de SPI softwarematig geinitialiseerd worden. Deze softwarematige initialisatie gebeurt in de functie spi_init_module. Na de initialisatie kan de SPI bus gebruikt worden om



Figuur 18 klassendiagram STN1170 configureren functies

berichten over te versturen en ontvangen.

Functie naam	**SPIInit()**
Gedrag	Initialiseren van de SPI bus
Parameters	Geen
Return waarde	Geen

Functie naam	**STNInit()**
Gedrag	Initialiseren van de STN1170
Parameters	Geen
Return waarde	Geen

Functie naam	**Obd_write_char()**
Gedrag	Stuurt een karakter over de SPI bus met line feed
Parameters	c : te versturen karakter
Return waarde	Geen

Functie naam	**Obd_write_line()**
Gedrag	Stuurt een string over de SPI bus met line feed
Parameters	String : te versturen string
Return waarde	Geen

Functie naam	**Obd_write_noln()**
Gedrag	Stuurt een string over de SPI bus zonder line feed
Parameters	String : te versturen string
Return waarde	Geen

Tijdens het uitvoeren van de test bleek het opzetboard verkeerd ontworpen te zijn. Er is besloten om de huidige iteratie af te sluiten en een nieuwe iteratie met een nieuw ontwerp te beginnen.

BIJLAGE F

Ontwerpdocument iteratie 2

Ontwerpdocument
Iteratie 2

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	: 1.0

Inhoudsopgave

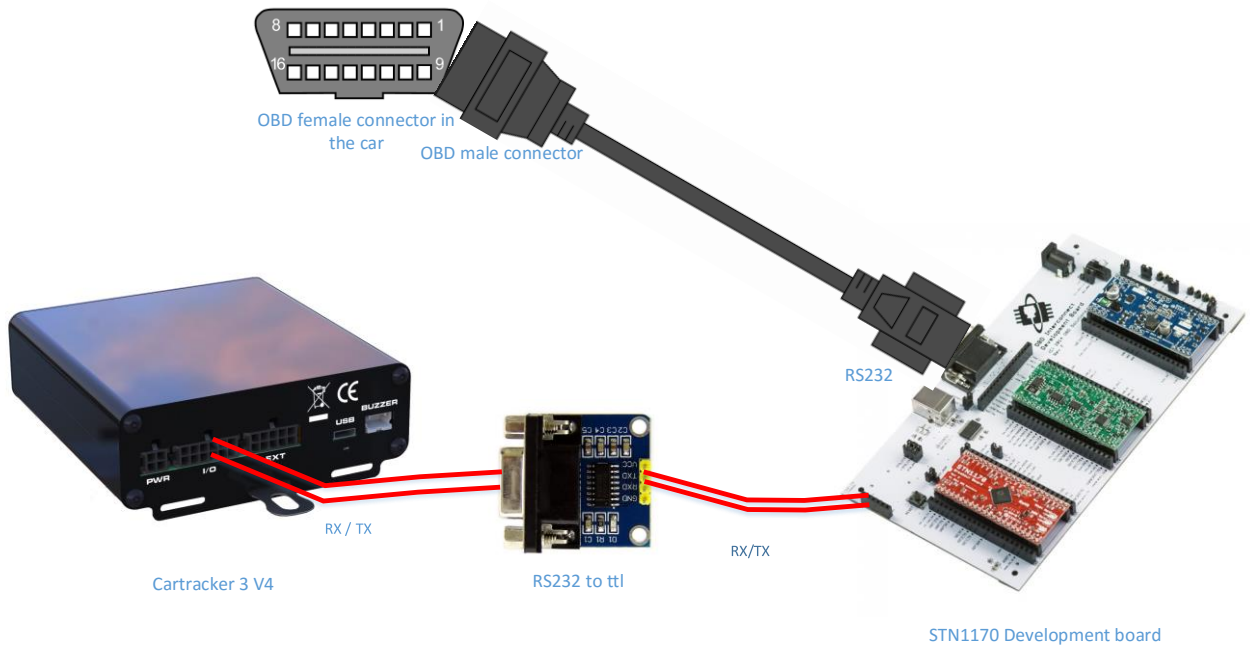
1.INLEIDING

In de tweede iteratie van het project is er een nieuw ontwerp gemaakt voor het project. Dit komt voort uit de problemen waar tegenaan is gelopen in de eerste iteratie. In plaats van de SPI wordt in het nieuwe ontwerp de USART van de Cartracker gebruikt. De STN1170 opzetprint is vervangen voor het STN1170 development board. Het doel van de iteratie is om de USART en het STN1170 development board te initialiseren.

2. GLOBAAL ONTWERP

Om een goed beeld te krijgen van de nieuwe hardware welke 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 19 Oud ontwerp Cartracker met OBD ondersteuning



Figuur 20 Nieuw ontwerp Cartracker met OBD ondersteuning

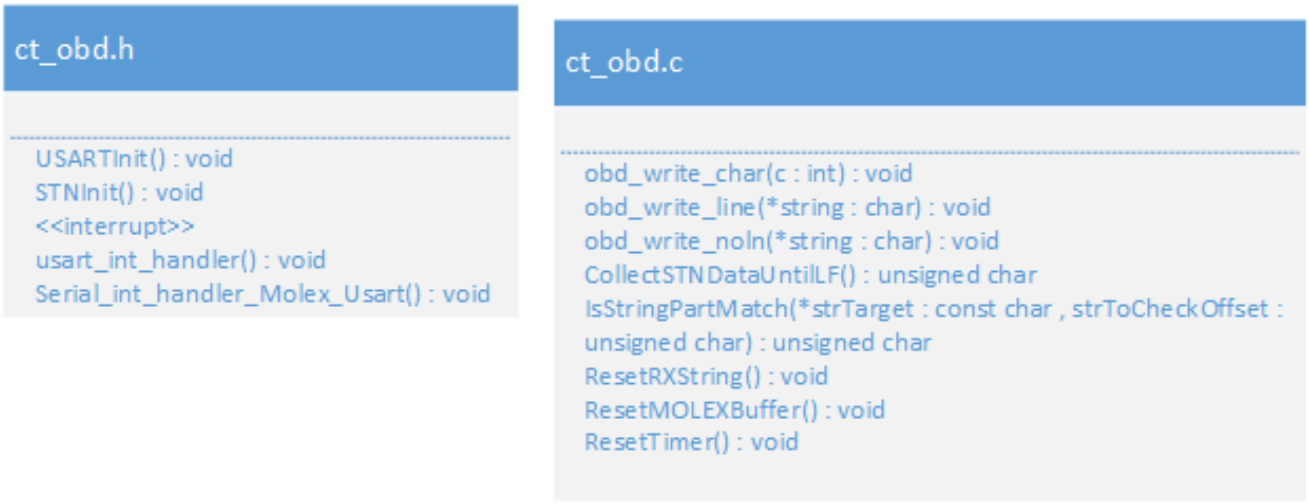
Het nieuwe ontwerp gebruikt in plaats van de STN1170 opzetprint de RS232 port op de Cartracker zelf. Via een RS232 naar TTL converter wordt het signaal omgezet

naar een 5V vertaald. Dit signaal wordt naar de RX en TX van het STN1170 development board gestuurd, welke in zijn beurt in verbinding staat met de auto via een RS232 naar OBD kabel. De vertaalslag tussen 5V uart en RS232 wordt op de translate en op het development board gedaan door de MAX3224. Via de RJ11 connector aan de achterkant van de Cartracker wordt er een seriele verbinding naar de pc gelegd. Deze connectie wordt gebruikt voor het debuggen en uitlezen van de STN1170.

3. ONTWERP

In deze iteratie is de volgende use-case uitgewerkt: STN1170 configureren. De initialisatie van het systeem wordt uitgevoerd en er wordt verbinding gelegd met de STN1170.

In het klassendiagram zijn de verschillende functies te zien welke geïntroduceerd zijn in deze iteratie.



Figuur 21 Klassendiagram STN1170 configureren

De verschillende functies worden hier onder met een omschrijving toegelicht.

Ct_obd.h functies

Naam	**USARTinit()**
Omschrijving	Initialisatie van de USART.

Naam	**STNInit()**
Omschrijving	Initialisatie van de STN1170.

Naam	**Usart_int_handler() : void**
Omschrijving	USART interrupt handler. Wordt aangeroepen bij een ingekomen karakter op de USART bus

Naam	Serial_int_handler_Molex_Usart() : void
Omschrijving	Handelt de ingekomen karakter in de Molex USART bus af. Het karakter wordt in een buffer opgeslagen

Ct_obd.c functies

Functie naam	Obd_write_char()
Gedrag	Stuurt een karakter over de USART bus met line feed
Parameters	c : te versturen karakter
Return waarde	Geen

Functie naam	Obd_write_line()
Gedrag	Stuurt een string over de USART bus met line feed
Parameters	String : te versturen string
Return waarde	Geen

Functie naam	Obd_write_noln()
Gedrag	Stuurt een string over de USART bus zonder line feed
Parameters	String : te versturen string
Return waarde	Geen

Functie naam	**CollectSTNDataUntilLF**
Gedrag	Leest data uit de MOLEX buffer tot het LineFeed character
Parameters	
Return waarde	1 – LF character gelezen 0 – Geen LF character gelezen

Functie naam	**IsStringPartMatch**
Gedrag	Vergelijkt string met deel van de MOLEX buffer
Parameters	strTarget : te vergelijken string strToCheckOffset : Offset van de string tov de buffer
Return waarde	1 – Match 0 – Geen match

Functie naam	**IsRXDataMatch**
Gedrag	Controleert of de strTarget binnen gekomen is in de RXMolexBuffer binnen de opgegeven timeout
Parameters	strTarget : De te vergelijken string timeoutDecaSeconds : Timeout
Return waarde	0 – Geen data 1 – Match 2 – Timeout bereikt 3 – Geen match

Functie naam	**ResetRXString**
Gedrag	Leeg de RXString en reset de counter
Parameters	Geen
Return waarde	Geen

Functie naam	**ResetMOLEXBuffer**
Gedrag	Leeg de MolexBuffer en reset de counter
Parameters	Geen

Return waarde	Geen

Functie naam	**ResetTimer**
Gedrag	Reset de interne timeout timer
Parameters	Geen
Return waarde	Geen

BIJLAGE G

Testrapport iteratie 2

Testrapport
Iteratie 2

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	: 1.0

Inhoudsopgave

4. INLEIDING

Dit testrapport is voortgekomen uit Iteratie 2 van het Cartracker met OBD ondersteuning project. In dit testrapport is de acceptatietest van Iteratie 2 te vinden. De uitgevoerde use-case wordt toegelicht en de resultaten zijn te vinden in het laatste hoofdstuk.

5. USE CASE

De use case welke getest gaat worden is : STN1170 configureren.

Use case beschrijvingen:

Naam	**STN1170 configureren**
Actor	Cartracker
Aannamen	De Cartracker is met de juiste OBD configuratie geconfigureerd De STN1170 is nog niet geconfigureerd
Beschrijving	De STN1170 wordt geconfigureerd 1- Juiste commando's worden ingevoerd 2- Cartracker krijgt melding dat de STN correct is geconfigureerd
Uitzonderingen	
Resultaat	De STN1170 is correct geconfigureerd

In het uitvoeren van de use-case is ook de initialisatie van de microcontroller meegenomen. De juiste configuratie van de STN1170 houdt in dat de USART correct is geconfigureerd en er een response check wordt uitgevoerd.

De te testen eisen voor deze iteratie zijn:

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

6. TEST EN RESULTATEN

In dit hoofdstuk zijn de testresultaten verwerkt in een overzicht. Bij iedere test case is er een verwachte uitvoer gemaakt op basis van de resultaten van het prototype.

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

BIJLAGE H

Ontwerpdocument iteratie 3

Ontwerpdocument
Iteratie 3

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	: 1.0

Inhoudsopgave

1.INLEIDING

In de derde iteratie is er gekeken naar de OBD informatie verwerking. Deze verwerking heeft betrekking op twee use-cases. Beide use-cases zijn tijdens deze iteratie geïmplementeerd. Op deze manier is de opvraag en verwerking van de OBD berichten mogelijk, wat een milestone is in het project Cartracker met OBD ondersteuning.

2. USE CASE

Iteratie 3 van het ontwikkelproces is gewijd aan twee use-cases: OBD code opvragen en OBD code verwerken. Vanwege het feit dat beide use-cases altijd direct op elkaar volgen is er besloten om beide use-cases op te nemen en hier één use-case van te maken.

De use-case beschrijving uit het Visiondocument ziet er als volgt uit:

Naam	**OBD code opvragen**
Actor	Cartracker
Aannamen	De STN1170 is correct geconfigureerd
Beschrijving	1- De Cartracker stuurt een request naar de STN1170 2- De STN1170 stuurt een request over de OBD bus naar de auto 3- De auto stuurt het antwoord naar de STN1170 4- De STN1170 controleert de geldigheid van het OBD bericht. Indien het bericht niet geldig is treedt uitzondering 1 op. 5- De STN1170 stuurt het OBD bericht door naar de Cartracker 6- De Cartracker ontvangt het opgevraagde bericht
Uitzonderingen	1- De STN1170 verstuurd het verkeerd ontvangen bericht niet en gaat terug naar stap 3
Resultaat	De opgevraagde code is ontvangen

Tijdens het implementeren van deze use-case waren er meer details bekend over de uitvoering van deze use-case. De use-case is daarom aangevuld met de laatste details. Dit resulteerde in een herziende versie van de use-case:

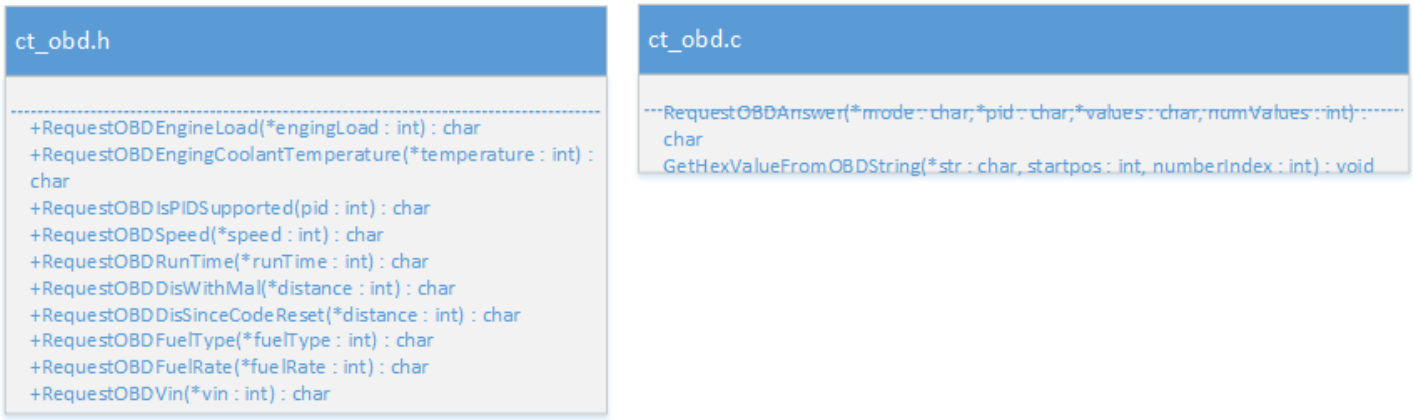
Naam	**OBD bericht opvragen**
Actor	Cartracker
Aannamen	De STN1170 is correct geconfigureerd
Beschrijving	1. De Cartracker stuurt een OBD request naar de STN1170 2. De STN1170 stuurt de request over de OBD bus naar de auto 3. De auto stuurt het antwoord naar de STN1170 4. De STN1170 stuurt het verkregen antwoord door naar de Cartracker 5. De Cartracker controleert de geldigheid van het bericht. Als het bericht niet voldoet aan het verwachte patroon treed uitzondering 1 op. 6. De Cartracker verwerkt de opgevraagde data op de juiste* wijze. 7. De Cartracker geeft het antwoord terug. De use-case wordt afgesloten en de Cartracker geeft een “OK” melding terug.
Uitzonderingen	1. De OBD reactie voldoet niet aan het verwachte patroon. Het bericht wordt niet verder geïnterpreteerd en er wordt uit de use-case gestapt. De Cartracker geeft een error melding terug.
Resultaat	Het opgevraagde OBD bericht is verwerkt en het resultaat is terug gestuurd.

*De juiste wijze duidt op de omrekening van de teruggekregen getallen. Per antwoord is het bekend hoe dit omgerekend moet worden.

3. ONTWERP

Het ontwerp is aan de hand van het architectuurdocument gemaakt. Deze iteratie zijn er een aantal nieuwe functies bijgekomen ten opzichte van het vorige ontwerp.

De nieuwe functies zijn te zien in Figuur 22. De functies worden in de volgende paragraaf toegelicht.



Figuur 22 Klassendiagram iteratie 3

Ct_obd.h functies

Functie naam	**RequestOBDSpeed**
Gedrag	Vraagt de huidige voertuigsnelheid op uit de auto
Parameters	*speed : Snelheid in km/h
Return waarde	OBD_SUCCES: OBD bericht correct ontvangen OBD_ERROR : OBD bericht niet correct ontvangen

Functie naam	**RequestOBDRunTime**
Gedrag	Vraagt de tijd in seconden sinds de auto is gestart
Parameters	*runtime : draaitijd in seconden
Return waarde	OBD_SUCCES: OBD bericht correct ontvangen OBD_ERROR : OBD bericht niet correct ontvangen

Functie naam	**RequestOBDDisWithMal**
Gedrag	Vraagt de gereden afstand met MAL lamp aan
Parameters	*distance : Afstand in km
Return waarde	OBD_SUCCES: OBD bericht correct ontvangen OBD_ERROR : OBD bericht niet correct ontvangen

Functie naam	**RequestOBDDisSinceCodeReset**
Gedrag	Vraagt de gereden afstand sinds code reset
Parameters	*distance : Afstand in km
Return waarde	OBD_SUCCES: OBD bericht correct ontvangen OBD_ERROR : OBD bericht niet correct ontvangen

Functie naam	**RequestOBDFuelType**
Gedrag	Vraagt het brandstoftype van de auto
Parameters	*fuelType : Brandstoftype
Return waarde	OBD_SUCCES: OBD bericht correct ontvangen OBD_ERROR : OBD bericht niet correct ontvangen

Functie naam	**RequestOBDFuelRate**
Gedrag	Vraagt het verbruik
Parameters	*fuelRate : Brandstofverbruik in l/h
Return waarde	OBD_SUCCES: OBD bericht correct ontvangen OBD_ERROR : OBD bericht niet correct ontvangen

Functie naam	**RequestOBDVin**
Gedrag	Vraagt het VIN nummer op van het voertuig
Parameters	*vin : Het VIN nummer van het voertuig
Return waarde	OBD_SUCCES: OBD bericht correct ontvangen OBD_ERROR : OBD bericht niet correct ontvangen

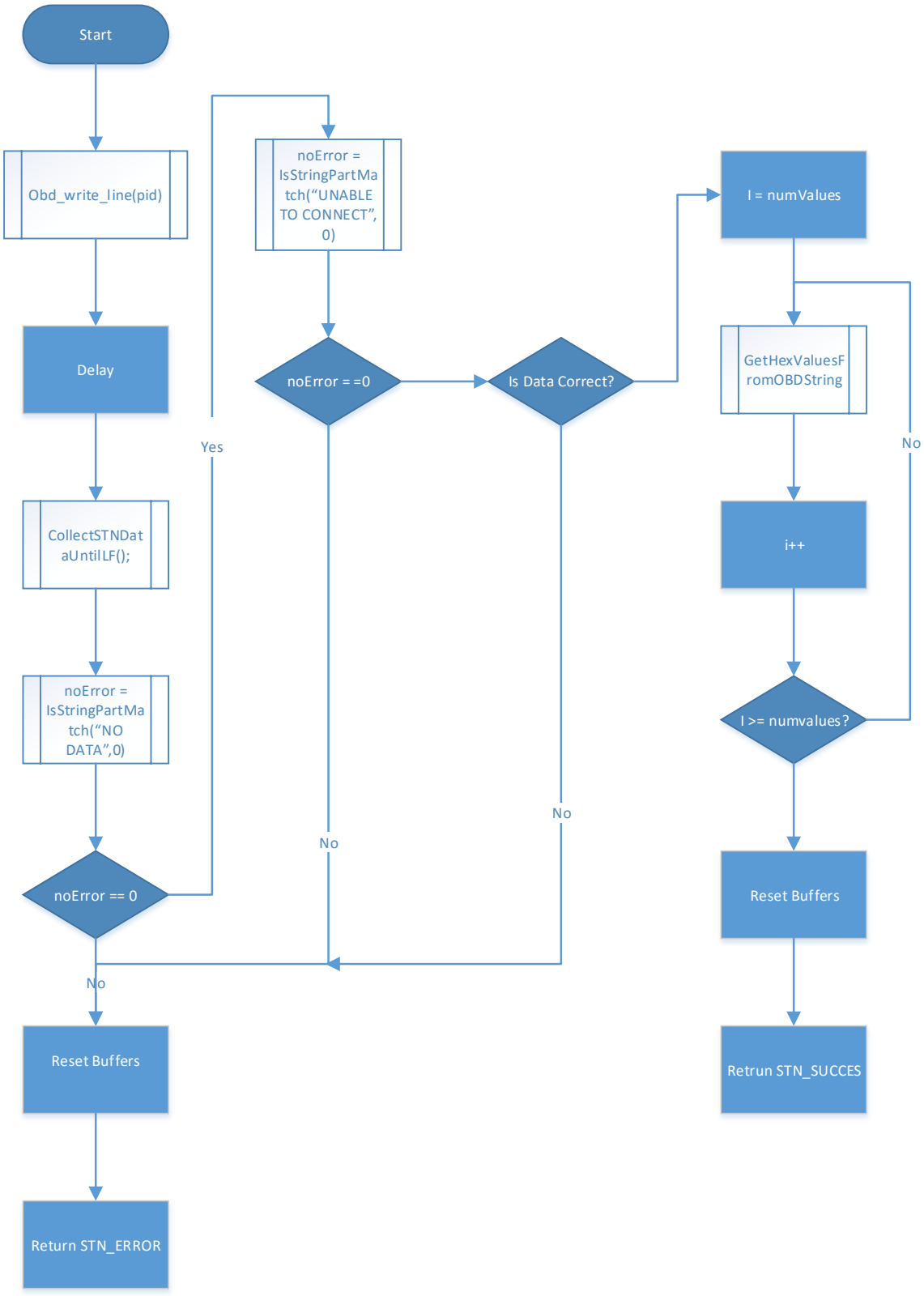
Ct_obd.c functies

Functie naam	**RequestOBDAnswer**
Gedrag	Vraagt een OBD code op en stuurt het antwoord terug
Parameters	*mode : OBD request mode *pid : OBD request Pid *values : Waardes uit het OBD antwoord numValues : Aantal verwachte antwoord velden
Return waarde	OBD_SUCCES: OBD bericht correct verwerkt OBD_ERROR : OBD bericht niet correct verwerkt

Functie naam	**GetHexValueFromOBDString**
Gedrag	Filtert de benodigde waarde uit de binnengekomen OBD string
Parameters	*str : binnengekomen OBD string startpos : startpositie van het benodigde antwoord numberIndex: Hoeveelste byte van het antwoord
Return waarde	Int : Waarde uit de binnengekomen OBD string in een integer

RequestOBDanswer

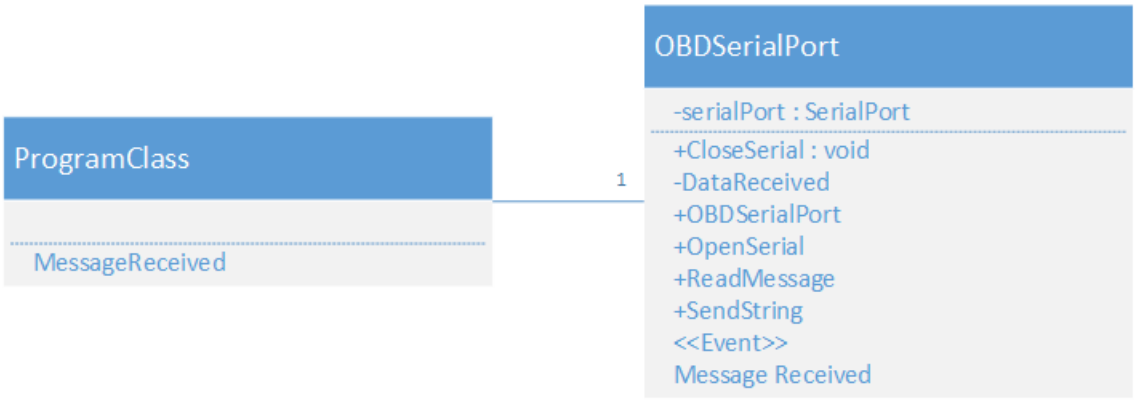
Het binnengekomen PID wordt via de USART naar de STN1170 gestuurd. Dit gebeurt met de functie Obd_write_line. Na het versturen wordt er gewacht zodat de STN1170 het bericht kan verwerken. Als het bericht verwerkt is wordt de data uit de STN1170 gelezen. Er wordt een error check op de verschillende binnengekomen velden uitgevoerd. Als de data niet correct is worden de buffers leeggemaakt en er wordt een STN_ERROR terug gestuurd. Als de data correct is wordt de waarde van hexadecimaal omgezet in decimaal. Deze waarde wordt opgeslagen en de buffers worden geleegd en er wordt een STN_SUCCES terug gestuurd.



Figuur 23 Flowchart Request OBD Answer

4. SIMULATOR

Om het ontwikkel en test proces in handelingen te verminderen is er besloten om een OBD simulator te maken. Dit is een consoleapplicatie welke een OBD verzoek binnen krijgt en een vooraf gedefinieerd antwoord terug stuurt. Hierdoor hoeft het testen van de software niet direct in een auto te gebeuren maar kan hier de simulator voor worden gebruikt.



Figuur 24 klassendiagram simulator

De simulator bestaat uit twee klassen. De ProgramClass handelt het ingekomen bericht uit de OBDSerialPort af en stuurt een bericht terug.

Voor de simulator is de eerdere SerialPort code van de prototype applicatie gebruikt.

BIJLAGE I

Testrapport iteratie 3

Testrapport
Iteratie 3

Auteur	:Robbert Eigenbrood
Bedrijf	:Decos Cartracker
Opdrachtgever	:Alain van Hanegem
Datum	: 3 juni 2016
Versie	:1.0

Inhoud

1.INLEIDING

In de derde iteratie van de Cartracker met OBD ondersteuning is de OBD afhandeling ontworpen en getest. Hierbij is de use-case OBD bericht opvragen en OBD bericht verwerken uitgevoerd.

2. USE CASE

Tijdens iteratie zijn er twee use cases geïmplementeerd. Beide use cases worden getest.

Naam	**OBD code opvragen**
Actor	Cartracker
Aannamen	De STN1170 is correct geconfigureerd
Beschrijving	1- De Cartracker stuurt een request naar de STN1170 2- De STN1170 stuurt een request over de OBD bus naar de auto 3- De auto stuurt het antwoord naar de STN1170 4- De STN1170 controleert de geldigheid van het OBD bericht. Indien het bericht niet geldig is treedt uitzondering 1 op. 5- De STN1170 stuurt het OBD bericht door naar de Cartracker 6- De Cartracker ontvangt het opgevraagde bericht
Uitzonderingen	1- De STN1170 verstuurd het verkeerd ontvangen bericht niet en gaat terug naar stap 3
Resultaat	De opgevraagde code is ontvangen

Naam	**Bericht verwerken**
Actor	Cartracker
Aannamen	Er is een bericht opgevraagd en ontvangen
Beschrijving	1- De Cartracker interpreteert het ontvangen bericht 2- De Cartracker geeft het bericht weer aan de gebruiker
Uitzonderingen	
Resultaat	Het ontvangen bericht is geïnterpreteerd en verwerkt in de Cartracker

De volgende eisen zijn getest:

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

3. TEST EN RESULTATEN

In dit hoofdstuk zijn de testresultaten verwerkt in een overzicht. Bij iedere test case is er een verwachte uitvoer gemaakt op basis van resultaten behaald tijdens een OBD simulatie.

De testen zijn op verschillende voertuigen uitgevoerd. Niet iedere auto ondersteund alle OBD PIDs, hierdoor kan niet iedere auto in het Decos wagenpark gebruikt worden voor alle testen. De auto's welke ter beschikking waren gesteld zijn:

- Peugeot 307
- Renault Megane

Overige PIDS welke op beide auto's niet ondersteund worden, zijn via de OBD simulator getest.

Te Testen eis	**Omschrijving**	**Verwachte uitvoer**	**Daadwerkelijke uitvoer**	**Opmerkingen**	**Akkoord**
F2 F3 F4	De Cartracker vraagt de snelheid van de auto op	De functie OBDRequestSpeed Returned de snelheid in km/h	De functie OBDRequestSpeed Returned de snelheid in km/h	Getest in Renault Megane	Ja
F2 F3 F4	De Cartracker vraagt de engine runtime op	De functie OBDRequestRuntime returned de engine runtime in seconden	De functie OBDRequestRuntime returned de engine runtime in seconden	Getest in Peugeot 307	Ja
F2 F3 F4	De Cartracker vraagt de afstand met de malfunction lamp op	De functie OBDRequestDisWithMal returned de afstand afgelegd met het malfunction licht aan in km	De functie OBDRequestDisWithMal returned de afstand afgelegd met het malfunction licht aan in km	Getest in Renault Megane	Ja
F2 F3 F4	De Cartracker vraagt de afstand afgelegd sinds een code reset op	De functie OBDRequestDisSinceCodeReset returned de afstand sinds de laatste code reset in km	De functie OBDRequestDisSinceCodeReset returned de afstand sinds de laatste code reset in km	Getest in Renault Megane	Ja
F2 F3 F4	De Cartracker vraagt het brandstoftype van de auto op	De functie OBDRequestFuelType returned het gebruikte brandstoftype	De functie OBDRequestFuelType returned het gebruikte brandstoftype	Getest in Simulator	Ja

F2 F3 F4	De Cartracker vraagt het brandstofverbruik van de auto op	De functie OBDRequesFuelrate returned het brandstofverbruik in L/h	De functie OBDRequesFuelrate returned het brandstofverbruik in L/h	Getest in Simulator	Ja
F2 F3 F4	De Cartracker vraagt het vin nummer op	De functie OBDGetVin returned het VIN nummer	De functie OBDGetVin returned het VIN nummer	Getest in Renault Megane	Ja

BIJLAGE J

Ontwerpdocument iteratie 4

Ontwerpdocument
Iteratie 4

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	: 1.0

Inhoudsopgave

4. INLEIDING

In iteratie 4 is er gekeken naar het uitlezen van CAN berichten via de OBD-II stekker. Uit de analyse is gehaald hoe deze berichten eruitzien en welke informatie sommige van deze berichten bevatten.

5. USE-CASE

In iteratie 4 is er naar de laatste use-case van het use-case diagram gekeken. CAN-bus scannen.

Naam	**CAN-bus scannen**
Actor	Cartracker
Aannamen	
Beschrijving	1- De Cartracker stuurt een filterspecificatie naar de STN1170 2- De Cartracker krijgt een “OK” bericht terug van de STN1170 3- De Cartracker stuurt het commando “STM” 4- De STN1170 start met monitoren, alleen berichten die voldoen aan het filter worden doorgelaten 5- De STN1170 stuurt de berichten door naar de Cartracker 6- De Cartracker stuurt een Carriage Return naar de STN1170 7- De STN1170 stopt met monitoren 8- De Cartracker interpreteert de ingekomen berichten op de juiste* wijze.
Uitzonderingen	
Resultaat	Er is een gefilterd CAN bericht ontvangen op de Cartracker

De juiste wijze houdt in dat de waardes relevant voor de opgevraagde informatie uit het bericht worden gefilterd en vertaald worden naar een decimale waarde.

Uit de bevindingen in de Elaboratie fase is gebleken dat het via de STN1170 en de OBD-II stekker mogelijk is CAN-bus berichten uit een auto te lezen. De STN1170 biedt de mogelijkheid om op niet gestandaardiseerde CAN bus berichten te filteren.

6. OBD KABEL

In de Nissan Leaf zijn een aantal CAN bussen aanwezig op de OBD stekker. De twee CAN bussen waar informatie op gelezen worden zijn de standaard CAN. Deze zit op pin 6 en 14 van de OBD-II stekker. De niet standaard pinnen die gebruikt worden voor de andere CAN bus, ook wel de EV-CAN genoemd omdat hier alle status berichten van het elektrische gedeelte van het voertuig over verzonden worden, zitten op pin 12 en 13.

Om van de verschillende CAN bussen gebruik te kunnen maken is er een OBD verloop kabel gemaakt welke pin 12 en 13 vanaf de OBD-II stekker in de Leaf omzet naar pin 6 en 14 op de STN1170. De EV-CAN

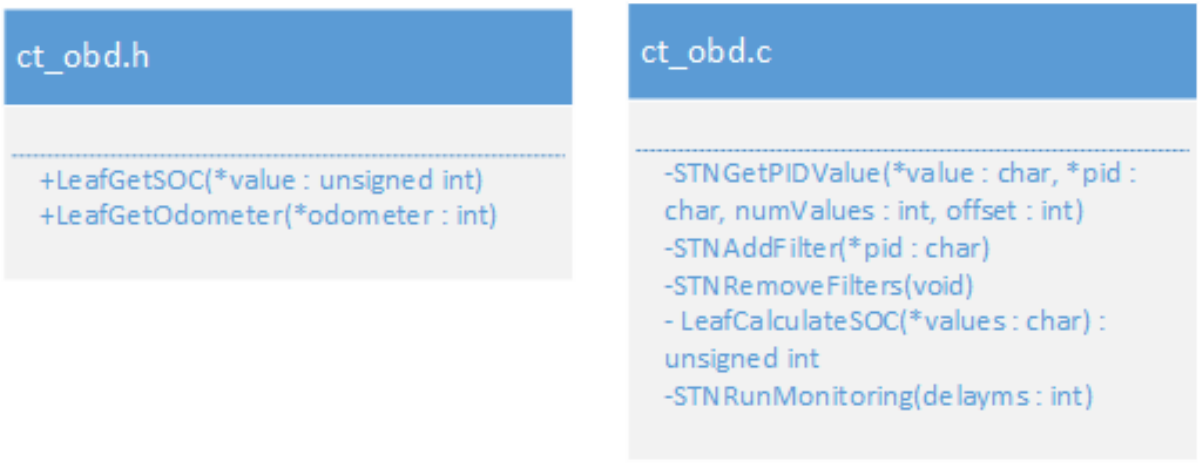
Cable Colours:

Male	Female	Type	Colour	
5	5	Signal ground	Black	
4	4	Chassis ground	Black-white	
16	16	12V	Red	
12	14	EV CAN-L	Orange	
13	6	EV CAN-H	Green	

Met deze kabel kan er op de EV-CAN informatie verzameld worden. Het is dan niet mogelijk om op een andere CAN bus data te lezen.

7. ONTWERP

In iteratie 4 zijn een aantal functies toegevoegd aan de huidige file. Er zijn twee public functies toegevoegd en een viertal private functies.



Figuur 25 Klassendiagram iteratie 4 functies

Ct_obd.h functies

De volgende functies zijn aan de ct_obd.h file toegevoegd.

Functie naam	**LeafGetSOC**
Gedrag	Vraagt de SOC op aan de STN1170
Parameters	*soc: State of Charge waarde van de auto accu's
Return waarde	

Functie naam	**LeafGetOdometer**
Gedrag	Vraagt de Odometer waarde op aan de STN1170
Parameters	*odometer : Odometer waarde in kilometers
Return waarde	

Ct_obd.c functies

De volgende functies zijn aan de ct_obd.c file toegevoegd:

Functie naam	**STNGetPIDValue**
Gedrag	Vraagt een specifiek CAN bericht op aan de STN1170
Parameters	[out]*values : Hier worden de waardes van het opgevraagde PID ingezet [in]*pid : Het op te vragen PID [in] numValues : De hoeveelheid bytes er uit het bericht gehaald moeten worden [in] offset : De offset in bytes waar de benodigde bytes in het bericht beginnen
Return waarde	STN_SUCCES : Het bericht is correct verwerkt STN_ERROR : Er is een fout opgetreden tijdens het verwerken of ophalen

Functie naam	**STNAddFilter**
Gedrag	Voeg een filter toe aan de STN1170
Parameters	*pid : Het toe te voegen filter
Return waarde	Geen

Functie naam	**STNRemoveFilters**
Gedrag	Verwijder alle filters uit de STN1170
Parameters	Geen
Return waarde	Geen

Functie naam	**LeafCalculateSOC**
Gedrag	Bereken de actuele SOC waarde en geef deze terug
Parameters	*values : De hexadecimale waarde uit het opgevraagde bericht
Return waarde	Unsigned int: Berekende soc waarde

Functie naam	**STNRunMonitoring**
Gedrag	Monitor met de toegepaste filters op de STN1170 voor de delaytijd
Parameters	Delayms: Tijd hoe lang er gemonitord moet worden
Return waarde	Geen

BIJLAGE K

Testrapport iteratie 4

Testrapport
Iteratie 4

Auteur	: Robbert Eigenbrood
Bedrijf	: Decos Cartracker
Opdrachtgever	: Alain van Hanegem
Datum	: 3 juni 2016
Versie	: 1.0

Inhoudsopgave

1.INLEIDING

De laatste iteratie van het proces is de CAN functionaliteit geïmplementeerd. Dit heeft betrekking op de laatste use case CAN bus scannen. Voor de test is een extra CAN verloop kabel gemaakt om op verschillende CAN bussen te kunnen scannen.

2. USE CASE

In de laatste iteratie van het proces is de laatste use case CAN bus scannen uitgevoerd

Naam	**CAN bus scannen**
Actor	Cartracker
Aannamen	
Beschrijving	1- De Cartracker stuurt een filterspecificatie naar de STN1170 2- De STN1170 geeft een OK bericht terug aan de Cartracker 3- De Cartracker stuurt een Start Scan bericht naar de STN1170 4- De STN1170 stuurt het filterresultaat naar de Cartracker
Uitzonderingen	
Resultaat	Er is een gefilterd CAN bericht ontvangen op de Cartracker

De te testen eisen zijn:

F5	De Cartracker moet verschillende CAN berichten kunnen opvragen
F6	De Cartracker moet verschillende CAN berichten kunnen verwerken

3. TEST EN RESULTATEN

In dit hoofdstuk zijn de testresultaten verwerkt in een overzicht. Bij iedere test case is er een verwachte uitvoer gemaakt op basis van resultaten behaald tijdens een OBD simulatie.

De testen hebben alleen betrekking gehad op de CAN bussen van de Nissan Leaf.

Getestte eis	Omschrijving	Verwachte uitvoer	Daadwerkelijke uitvoer	Opmerkingen	Akkoord
F5 F6	De Cartracker vraagt de Odometer op	De functie LeafGetOdometer returned de odometer van de auto	De functie LeafGetOdometer returned de odometer van de auto	De Odometer is op de standaard CAN bus getest	Ja
F5 F6	De Cartracker vraagt de SoC op	De functie LeafGetSOC returned de SoC van de auto	De functie LeafGetSOC returned de SoC van de auto	De SoC is op de EV-CAN met verloopkabel getest	Ja