SOGETI

DE HAAGSE HOGESCHOOL

# P0

# P0 communicatie SMA

**Opleiding:** Elektrotechniek, Haagse Hogeschool
**Shift:** 3, 08-02-2016 t/m 03-06-2016
**Afstudeerder:** Niels Hokke, 12033758
**Examplaar bestemd voor:** HBO kennisbank

# Voorwoord

Dit afstudeerverslag dient als afronding van mijn HBO elektrotechniek aan de Haagse Hogeschool te Delft. Via een werknemer van Sogeti op een Tech-beurs raakte ik bekend met Sogeti. Sogeti had een lopend project met slimme meters wat mij erg interessant leek. Na een kennismakingsgesprek heb ik de afstudeerplek aangenomen. De reistijd van twee uur was voor mij geen belemmering. Ik heb deze tijd goed kunnen gebruiken voor het maken van deze scriptie.

Ik ben geïnteresseerd in embedded systems en ben ook van plan, na het halen van mijn HBO diploma de embedded systems master te volgen aan de TU Delft. Deze opdracht met het programmeren op een embedded systeem past goed bij mijn interesse en toekomstige studie.

Graag wil ik Erik van Raalte, Marc Welleweerd en Urs Tathoff bedanken voor de hulp en feedback. Het praten over een probleem maakte het gemakkelijker om op te lossen. Ook wil ik graag Gerard van Teeffelen, Bart Kramer, Jos de Doelder, Johan Woudstra en Fidelis Theinert bedanken voor de feedback en begeleiding die zij hebben gegeven.

Niels Hokke

Amersfoort 2-6-2016

# Samenvatting

Sogeti is een ICT dienstverlener en wenst meer en gedetailleerdere kennis te vergaren over de werking van slimme meters. Hierdoor is personeel gemakkelijker in te zetten bij klanten en kan het bedrijf haar positie versterken door complementerende diensten aan te kunnen bieden en/of te kunnen ontwikkelen. Om deze reden heeft Sogeti in september 2015 een smart meter project opgezet. Het project bestaat uit het opzetten van een slimme meter test set-up. In dit project kunnen meerdere (afstudeer) stagiaires bekend worden met Sogeti. Dit document is het resultaat van een van de subopdrachten van het grotere slimme meter project. Het doel van deze afstudeeropdracht is het onderzoeken, ontwerpen en implementeren van de communicatie tussen de hardware van de test set-up en communicatie poort 0 (P0) van de slimme meters.

Tijdens het onderzoek naar P0 werd snel duidelijk dat P0 te veel functionaliteiten had om geheel geïmplementeerd te kunnen worden binnen een stageperiode. Daarom zijn alleen de voor het grotere project meest relevantie functies geïmplementeerd. Omdat de voorgestelde hardware crypto-core tijdens het programmeren van de P0 communicatie software nog niet beschikbaar was, is er gebruik gemaakt van een software crypto-core. Deze software crypto-core bleek snel genoeg voor de P0 communicatie. De software crypto-core is zo ontworpen dat deze later indien nodig gemakkelijk vervangen kan worden door de hardware crypto-core.

Omdat de P0 communicatie software *open-ended* ontworpen is kunnen de niet-geïmplementeerde functies alsnog gebruikt worden door middel van een sendRaw functie. Omdat de code object oriented opgezet is, kunnen onderdelen van de P0 communicatie software zoals, het serial driver blok, hergebruikt worden in toekomstige software van de slimme meter test set-up.

Het is grotendeels gelukt om de P0 communicatiesoftware te realiseren binnen de afstudeerperiode. Sogeti's slimme meter project is een stap dichter bij een volledige werkende test set-up.

# Summary

Sogeti is an ICT service provider who wants to build up knowledge and know-how about smart meters to provide beter services for their clients. That is why in September 2015 Sogeti started a smart meter project where multiple graduation interns could get to know Sogeti and generate knowledge about smart meters. The goal of the project is to design and build a smart meter test set-up. This document is the result of one of the graduation sub projects of the bigger smart meter project. The goal of this graduation project is to research, design and implement the communication between the measuring application of the test set-up and the P0 communication ports of the smart meters.

While researching P0 the conclusion was made that there where far too much functionalities to implement within a graduation period. This is why only the most relevant functions of P0 for the project where implemented. Because the suggested hardware encryption core wasn't available at the time the P0 communication software was written a software encryption core was used. This software crypto-core is fast enough for the P0 communication. The software crypto-core is designed in such a way that it can easily be swapped with the hardware crypto-core if necessary.

Because the communication software is designed in an open-ended way the not implemented functions of P0 can still be used through a sendraw function. The communication software is written in an object oriented way so some parts like the serial driver can be used in other parts of the measuring application software.

Almost all of the P0 communication software is devoloped within the graduation internship period. Sogeti's smart meter project is one step closer to a fully working smart meter test set-up.

# Verklarende woordenlijst

| | |
|---|---|
| 7E1 | 7 data bits, even parity bit, 1 stop bit |
| 8N1 | 8 data bits, no parity bit, 1 stop bit |
| AAD | Additional authenticated data |
| CT | Cypher text |
| DSMR | Dutch smart meter requirements |
| HLS | High level security |
| IV | Initialization vector |
| K | Encryptie key |
| LLS | Low level security |
| P | Plaintext |
| P0 | Communicatie poort 0 |
| P1 | Communicatie poort 1 |
| P2 | Communicatie poort 2 |
| P3 | Communicatie poort 3 |
| P4 | Communicatie poort 4 |
| SC | Security control fieldt |
| SMA | Smart meter aggregator |
| TAG | Authentication tag |

# Tabellenlijst

# Inhoudsopgave

# 1 Introductie

De huidige energie-, gas- en watermeters worden vervangen door slimme varianten. Naast het meten van het energie-, gas- en waterverbruik, stuurt een slimme meter automatisch de meterstanden naar de netbeheerder. Ook kunnen de slimme meters communiceren met huishoudelijke applicaties zoals slimme thermostaten. Dit creëert een slimme infrastructuur waardoor de energieleveranciers en de huishoudens een beter beeld van het verbruik krijgen.

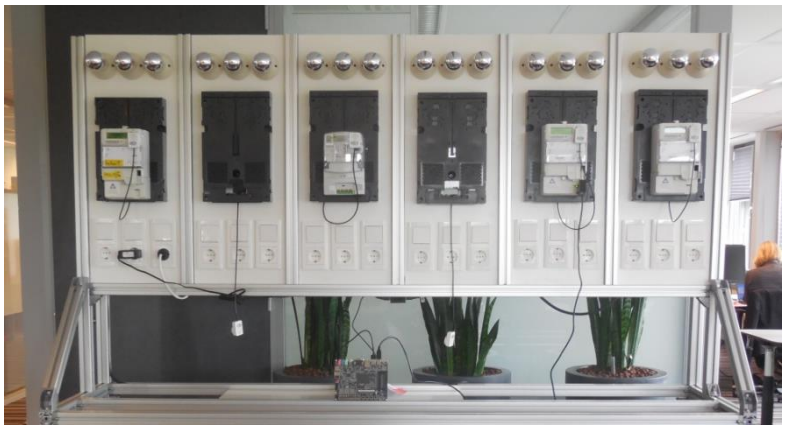
Sinds 2012 worden er slimme meters in Nederlandse huishoudens geplaatst. De Europese Commissie van Energie heeft als doel om 80% van alle huishoudens in Europa te voorzien van slimme energiemeters voor 2020 [1]. Dit creëert een grote markt in slimme meters en toebehoren.

Sogeti is een van de grootste IT dienstverleners van Nederland. Sogeti Heeft een missie: "De beste ICT dienstverlener van ons land zijn met de hoogste klant- en medewerker-tevredenheid en de beste financiële prestaties". Om deze missie te halen moet Sogeti bijblijven met de nieuwste technische ontwikkelingen. Een van deze nieuwe ontwikkelingen zijn de slimme meters.

## 1.1 Doorlopend project

Om meer en gedetailleerdere kennis over slimme meters te vergaren is Sogeti een doorlopend project gestart waar meerdere (afstudeer-) stagiaires aan werken [2]. Het doel van dit doorlopende project is het ontwikkelen van een test set-up voor slimme meters. In deze set-up kunnen meerdere slimme meters worden uitgelezen en aan diverse tests onderworpen worden. Dit zodat er binnen het bedrijf meer kennis wordt vergaart over deze meters. Daarnaast zal de set-up van slimme meters ook demonstreerbaar zijn aan klanten en mogelijke nieuwe werknemers.

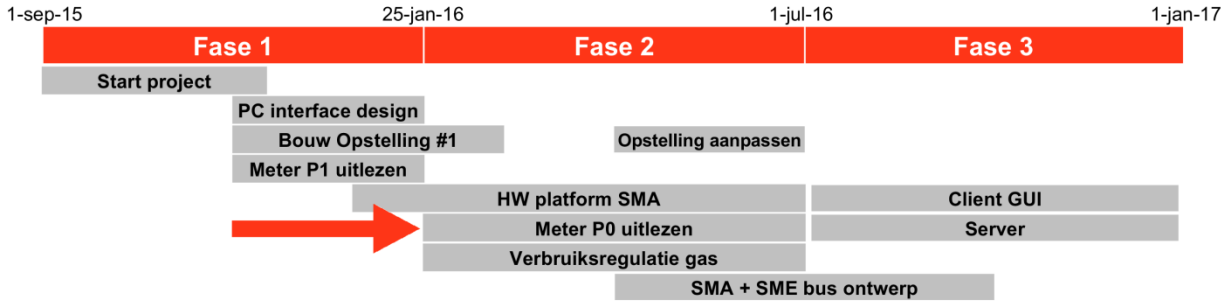
In september 2015 zijn twee afstudeerders begonnen met het doorlopend project. Zij hebben zich bezig gehouden met het verzamelen van data en het opzetten van de set-up. De set-up bestaat uit een mobiel frame waar meerdere slimme meters op gemonteerd zijn, zie Figuur 1.



**Figuur 1: Test setup**

In november 2015 is een derde afstudeerder bij het project gekomen. Deze afstudeerder heeft zich bezig gehouden met het selecteren van een hardware platform waarop de meetapplicatie gaat draaien. De meetapplicatie moet gaan communiceren met de slimme meters via de verschillende communicatiepoorten [3]. Ook moet de meetapplicatie variabele belastingen aan kunnen sturen om de meters bij verschillende situaties te testen.

In Figuur 2 is te zien hoe de opdracht van dit verslag in het grotere project past. De door de rode pijl aangegeven strook, is het onderwerp van deze scriptie.




**Figuur 2: Globale stroken planning**

## 1.2 Afbakening opdracht

Om de slimme meters te testen moet er met de meters gecommuniceerd worden. Dit kan over meerdere poorten. Een van die poorten is P0. Deze poort wordt gebruikt voor het programmeren en het uitgebreid uitlezen van de slimme meter.

Het doel van de opdracht is het realiseren van communicatie tussen de P0 poort van de slimme meters en de Smart Meter Aggregator (SMA) waarop de meetapplicatie draait. Deze communicatie zal in een *open ended* en *object-oriented* manier geschreven worden zodat er ook met meerdere en toekomstige slimme meters gecommuniceerd kan worden. De SMA is een hardware/software platform dat tussen de server en de slimme meters instaat. De SMA bevat een hardware crypto-engine waarmee geïncrypte berichten snel en zonder veel processing power gedecrypt worden.

Hoofdvraag:

- Hoe kan er tussen de SMA en de P0 poort van meerdere slimme meters gecommuniceerd worden in een *open ended* en *object-oriented* manier?

Subvragen:

- Hoe wordt er gecommuniceerd over P0?
- Wat is het verschil tussen de communicatie over P0 op verschillende meters?
- Hoe kan de communicatie met P0 gerealiseerd worden op de SMA in een *open ended* en *object-oriented* manier?
- Hoe kunnen de berichten gedecrypt worden via de crypto-engine van de SMA?
- Hoe kan een programma communiceren met meerdere P0 poorten?

# 2 Aanpak

Op basis van de hoofd- en subvragen zijn er tussenresultaten opgezet. Omdat het in het begin van de afstudeerperiode nog niet duidelijk was waaruit de SMA zou bestaan en wanneer deze beschikbaar zou zijn, is er voor gekozen om de P0 software eerst op een Windows machine te schrijven. Wanneer er meer informatie over de SMA beschikbaar kwam, zou de al geschreven software worden omgezet.

Tussenresultaten:

- Informatie over de werking van P0
  - Werking protocol
  - Verschillen tussen meters
  - Werking encryptie en decryptie
- Software voor op een PC waarmee met een P0 poort gecommuniceerd kan worden
  - verbinding opzetten
  - berichten opbouwen
  - verbinding afsluiten
- Software voor op de PC waarmee met meerdere P0 poorten gecommuniceerd kan worden
- Software die meerdere P0 poorten ondersteunt van meerdere verschillende slimme meters geschreven in een *openended* en *object-oriented* manier draaiend op de SMA
- Een eindverslag

De tussenresultaten zijn vervolgens in een planning gezet met de bij de tussenresultaten horende werkzaamheden.

Tijdens het onderzoek bleek de werking van P0 vele malen complexer dan voorheen gedacht. Omdat niet alle functionaliteiten van P0 nodig zijn, is er voor gekozen om niet alle mogelijkheden van P0 te implementeren en te focussen op de meest relevanten functies. Ook werd tijdens het begin van het onderzoek duidelijk dat een extra onderzoek naar de verschillen voor P0 op verschillende meters niet nodig was. Dit omdat P0 alleen op de slimme energiemeters aangesloten is en voor alle slimme energiemeters gelijk werkt. Om deze reden is de planning aangepast, zie Tabel 1. In Bijlage B: Stroken planning blz. 45 is de bijhorende stroken planning te vinden.

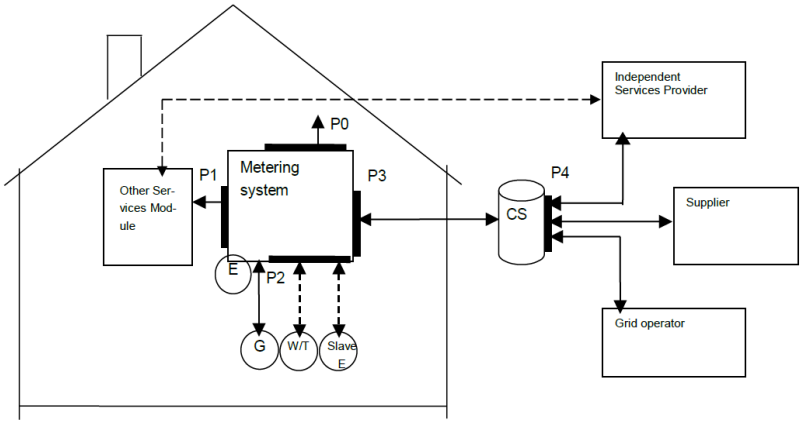
| Activiteiten | | van | | tot |
|---|---|---|---|---|
| **Totaal** | | **08-feb-2016** | | **10-jun-2016** |
| **PVA** | | **08-feb-2016** | | **24-feb-2016** |
| **Scriptie** | | **24-feb-2016** | | **3-jun-2016** |
| **P0** | | **25-feb-2016** | | **19-mei-2016** |
| Onderzoek Protocol P0 | | 17-feb-2016 | | 2-mei-2016 |
| *Onderzoek opzetten verbinding* | | *17-feb-2016* | | *10-apr-2016* |
| *Onderzoek bericht opbouw* | | *10-apr-2016* | | *28- apr -2016* |
| *Onderzoek sluiten verbinding* | | *28- apr -2016* | | *2-mei-2016* |
| Implementatie P0 | | 8-mrt-2016 | | 5-jun-2016 |
| *Opzetten verbinding PC* | | *8-mrt-2016* | | *5-apr-2016* |
| *Verzenden berichten PC* | | *15-apr-2016* | | *25- apr -2016* |
| *Sluiten verbinding PC* | | *28-apr-2016* | | *5-mei-2016* |
| *Meerdere verbindingen* | | *5-mei-2016* | | *25-mei-2016* |
| *Omzetten code SMA* | | *18-mei-2016* | | *5-jun-2016* |
| test en optimalisatie | | 13-mei-2016 | | 19-mei-2016 |

**Tabel 1: Aangepaste planning**

# 3 Slimme meter P0

De slimme energiemeter kan communiceren via meerdere interfaces bestaande uit 5 verschillende poorten: P0 tot P4. Figuur 3 laat de opbouw van de slimme meter infrastructuur zien. In de huishoudens is de slimme energiemeter het centrale punt waarmee alle andere slimme meters (Water en gas) communiceren. Het is ook hier waar alle verbruiksdata wordt opgeslagen en opgevraagd kan worden.

Communicatie poort 0 (voortaan P0) is een van de 5 interfaces en wordt gebruikt voor programmeren, onderhoud en uitlezen van de slimme energiemeter. Volgens de DSMR [4] is P0 niet aanwezig op de andere slimme meters. De communicatie over P0 verloopt via een seriële verbinding. Hardware matig gaat de verbinding via een optische kop, hierdoor is de slimme energiemeter galvanisch gescheiden van externe uitlees apparatuur.




**Figuur 3: Slimme meter interface overzicht (Bron: DSMR [4] )**

## 3.1 Protocol modes

In de project eisen [3] (Sup3), staat dat de test set-up alle Nederlandse slimme meters moet kunnen ondersteunen. Volgens de standaart [5] ondersteunt P0 5 verschillende protocol modes. Dit zijn protocol modes A, B, C, D en E. Volgens DSMR-M 4.3.80 en DSMR-M 4.3.80a [4] moeten alle Nederlandse slimme energiemeters een P0 poort hebben en volgens IEC 62056-21 [5] protocol mode E ondersteunen. Sommige Nederlandse slimme energiemeters ondersteunen naast mode E ook de andere protocol modes. Omdat alleen de aanwezigheid van mode E verzekert is, is er voor gekozen om te focussen op mode E en de andere protocol modes terzijde te laten[1]. Ook voldoet mode E aan alle andere projecteisen [3] die worden gesteld aan P0 zoals Tes1.1, Tes2.1, Tes3.4.1. Dit beantwoordt de subvraag: "Wat is het verschil tussen de communicatie over P0 op verschillende meters?".

[1] Door het *open-ended* ontwerp van de P0 software kunnen de andere protocol modes gebruikt worden via de sendRaw functie.

# 4 Werking mode E

Dit hoofdstuk bevat informatie over de werking van mode E en geeft antwoord op de subvraag: "Hoe wordt er gecommuniceerd over P0?". Dit hoofdstuk is onderverdeeld in Data architectuur slimme meter, Opzetten connectie, Gebruik Connectie, Afsluiten connectie, HDLC en Security. Het protocol dat gebruikt wordt om over P0 te communiceren bleek veel complexer dan verwacht. Hierdoor zal niet elk stuk voledig worden uitgelegd. Dit is voor de huidige stand van het project geen probleem omdat deze uitleg genoeg is om voor het project gebruikt te worden. De informatie in dit hoofdstuk is een samenvatting en verduidelijking van de volgende documenten: Green book [6], Blue Book [7], IEC 62056-21 [5] en DSMR [4].

## 4.1 Data architectuur slimme meter

De data in de slimme energiemeter is verdeeld in meerdere tabellen genaamd OBIS tabellen [7]. Elke tabel bevat data over een onderwerp zoals tijd, gas of water. Onderstaande Tabel 2 geeft een voorbeeld van een OBIS tabel over datum en tijd. Naast de huidige datum en tijd staat er ook wanneer de zomer/wintertijd ingaat. Via mode E kunnen de attributes van een OBIS tabel gezet en opgevraagd worden. Ook kunnen de OBIS tabel specifieke methods aangeroepen worden. Dit zijn functies die OBIS tabel specifiek zijn. De OBIS tijd tabel heeft bijvoorbeeld de method shift_time (method 6). Via deze method kan de tijd met een x aantal seconde verschoven worden.

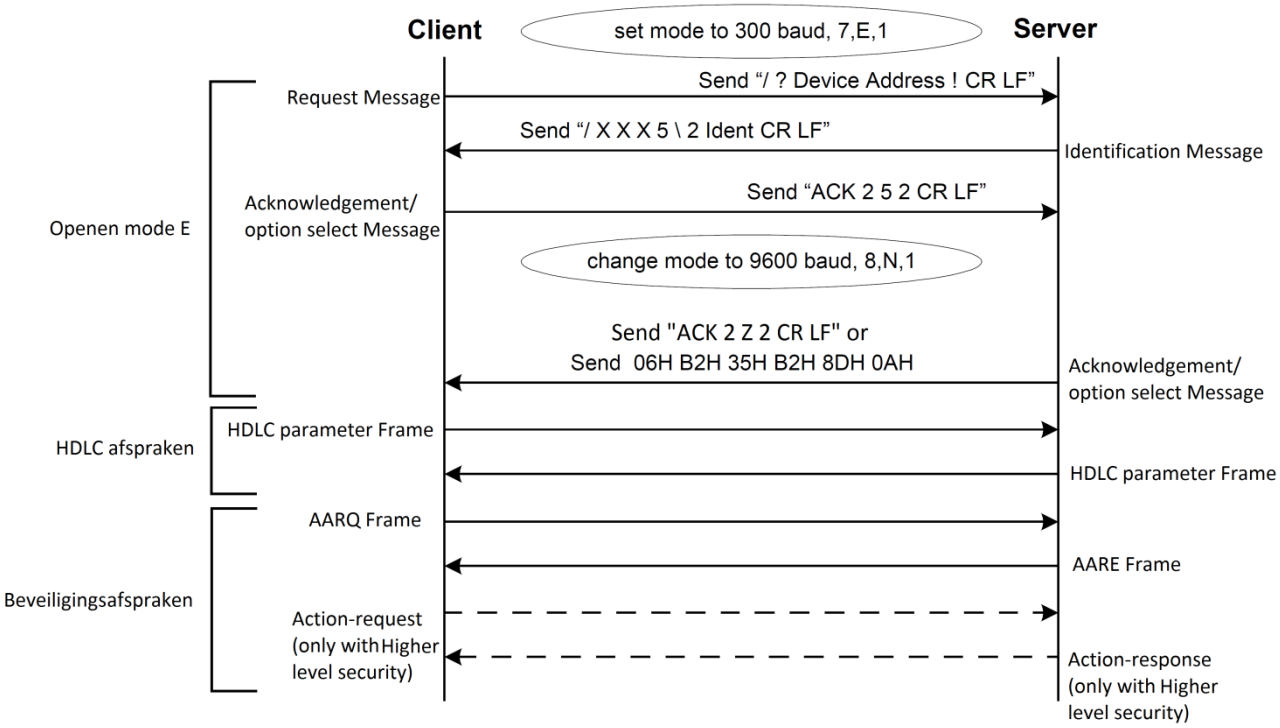
| **Clock** | | **0…n** | **Class_id = 8, version = 0** | | | |
|---|---|---|---|---|---|---|
| **Attributes** | | **Data type** | **Min.** | **Max.** | **Def.** | **Short name** |
| 1. logical_name | (static) | ocet-string | | | | x |
| 2. time | (dyn.) | ocet-string | | | | x + 0x08 |
| 3. time_zone | (static) | long | | ± 720 | | x + 0x10 |
| 4. status | (dyn.) | unsigned | | | | x + 0x18 |
| 5. daylight_savings_begin | (static) | ocet-string | | | | x + 0x20 |
| 6. daylight_savings_end | (static) | ocet-string | | | | x + 0x28 |
| 7. daylight_savings_deviation | (static) | integer | | ± 120 | | x + 0x30 |
| 8. daylight_savings_enabled | (static) | boolean | | | | x + 0x38 |
| 9. clock_base | (static) | enum | | | | x + 0x40 |
| **Specific methods** | | **m/o** | | | | |
| 1. adjust_to_quarter(data) | | o | | | | x + 0x60 |
| 2. adjust_to_measuring_period(data) | | o | | | | x + 0x68 |
| 3. adjust_to_minute(data) | | o | | | | x + 0x70 |
| 4. adjust_to_preset_time(data) | | o | | | | x + 0x78 |
| 5. preset_adjusting_time(data) | | o | | | | x + 0x80 |
| 6. shift_time(data) | | o | | | | x + 0x88 |

**Tabel 2: OBIS tabel clock (Bron: Blue Book [7])**

In 9.5 van het Green book [6] is een lijst van alle mode E commando's te vinden. Deze lijst is te groot om geheel te implementeren. Er is gekozen om te focussen op de Get, Set en Action commando's. Met gebruik van deze commando's kunnen de attributes van de OBIS tabellen opgehaald en gezet worden en kunnen de Specific methods worden aangeroepen. Hiermee wordt er voldaan aan alle projecteisen [3] van P0.

## 4.2 Opzetten connectie

Om in mode E te komen moeten er meerdere berichten heen en weer gestuurd worden. Figuur 4 geeft een overzicht van alle berichten die verstuurd worden bij het opzetten van mode E. Hierbij is de SMA de cliënt en de slimme energiemeter de server. Het opzetten van de connectie is onderverdeeld in Openen mode E, HDLC afspraken en Beveiligingsafspraken.




**Figuur 4: Opstart sequence (Bron: IEC 62056-21 [5])**

### 4.2.1 Openen mode E

De communicatie start met het openen van mode E. Deze communicatie start op 300 baud in 7E1 en gaat na een aantal berichten over naar 9600 baud in 8N1. Het openen van mode E begint met een Request message van de cliënt waarop de server een identification message stuurt. Vervolgens stelt de cliënt via een acknowledgement/option-select message een protocol mode en een nieuwe baudrate voor. Via een zelfde bericht laat de server weten of de voorgestelde data mogelijk is.

#### Request message

De communicatie wordt geïnitialiseerd door de cliënt via de Request Message, zie Tabel 3.

| | | | | | |
|---|---|---|---|---|---|
| / | ? | Device address | ! | CR | LF |

**Tabel 3: Request Message (Bron: IEC 62056-21 [5])**

Het Device address is volgens de standaard [5] optioneel. Echter niet alle slimme energiemeters beschikbaar bij Sogeti blijken dit te volgen, een aantal van de meters hebben: "P07210" als Device address en reageren niet bij een leeg Device address veld. Dit kan bij toekomstige slimme energiemeter ook het geval zijn.

## Identification Message

Als antwoord op het Request Message stuurt de server een Identification Message. In Tabel 4 is te zien hoe dit bericht is opgebouwd.

| / | X | X | X | Z | \ | W | Identification | CR | LF |
|---|---|---|---|---|---|---|---|---|---|

**Tabel 4: Identification Message (Bron: IEC 62056-21 [5])**

X X X is een 3 byte groot veld waar de Manufacturer's Identification tag in staat. Dit ID bestaat uit 3 letters, als de derde letter geen hoofdletter is, is de minimale reactie tijd van de server 20 ms i.p.v. 200 ms. Een lijst van alle Manufacturer's Identifications is te vinden op [8].

Z staat voor het baudrate identificatie karakter. Dit karakter geeft aan welke baudrate en protocol de server voorstelt om naar over te gaan. Tabel 5 geeft een overzicht van de mogelijkheden van het baudrate identificatie karakter.

| Mode A | Mode B | Mode C/E | Mode D | Baudrate |
|---|---|---|---|---|
| Alle andere karakters behalve "/" en "!" | | 0 | | 300 Bd |
| | A | 1 | | 600 Bd |
| | B | 2 | | 1200 Bd |
| | C | 3 | 3 | 2400 Bd |
| | D | 4 | | 4800 Bd |
| | E | 5 | | 9600 Bd |
| | F | 6 | | 19200 Bd |

**Tabel 5: Baudrate identification (Bron: IEC 62056-21 [5])**

Het Identification veld begint met "\W". W geeft aan of het protocol in binary mode (zie 4.5 HDLC blz. 20) zal overgaan. Dit wordt aan gegeven met een 2, alle andere karakters zijn gereserveerd voor toekomstige applicaties. De rest van het identification veld bevat een manufacturer specifiek id van maximaal 16 bytes lang.

## Acknowledgement/option select Message

Na het ontvangen van de Identification Message stuurt de cliënt een acknowledgement/option-select Message. Tabel 6 laat de opbouw van dit bericht zien.

| ACK | V | Z | Y | CR | LF |
|---|---|---|---|---|---|

**Tabel 6: Acknowledgement/option select Message (Bron: IEC 62056-21 [5])**

Als bevestiging stuurt de server hetzelfde bericht weer terug in de nieuwe baudrate. Dit kan gedaan worden in 7E1 of 8N1. Als dit in 7E1 verzonden wordt en in 8E1 gelezen, ontvangt de cliënt iets dat lijkt op Tabel 7. Deze onzekerheid kan opgelost worden door altijd het meest linker bit van elk ontvangen byte op 0 te zetten. Hierdoor is het antwoord altijd in het formaat van Tabel 6.

| ACK | B2 | 35 | B2 | 8D | LF |
|---|---|---|---|---|---|

**Tabel 7: Acknowledgement/option select Message 2 (Bron: IEC 62056-21 [5])**

V in het acknowledgement/option select message staat voor het protocol control character. Tabel 8 geeft een overzicht van de mogelijkheden van dit protocol control character.

| Protocol control character | Betekenis |
|---|---|
| 0 | Normal protocol procedure |
| 1 | Secondary protocol procedure |
| 2 | HDLC protocol procedure |
| 3-9 | Reserved |

**Tabel 8: Protocol control character (Bron: IEC 62056-21 [5])**

Z in het acknowledgement/option select message staat voor het baudrate identificatie karakter, zie Tabel 5.

Y in het acknowledgement/option select message staat voor het mode control character, Tabel 9 geeft een overzicht van de mogelijkheden voor dit control character

| Mode control character | Betekenis |
|---|---|
| 0 | Data readout |
| 1 | Programming mode |
| 2 | Binary mode (HDLC) |
| 3-5 and A-Z | Reserved |
| 6-9 | Manufacturer specific use |

**Tabel 9: Mode control character (Bron: IEC 62056-21 [5])**

### 4.2.2 HDLC afspraken

Vanaf dit punt is de communicatie in binary mode (HDLC). Dat wil zeggen dat alle berichten vanaf nu ingepakt moeten worden volgens het HDLC format, zie 4.5 HDLC blz. 20. De eerste HDLC frames die worden verzonden zijn de HDLC parameter frames. Via deze frames wordt er onderhandeld welke HDLC parameters de cliënt en server beiden kunnen ondersteunen [6].

#### HDLC parameter frame

De cliënt begint met het sturen van een HDLC SNRM frame (Tabel 35: Control types) met als informatie Tabel 10. De server vergelijkt de parameters van de cliënt met de eigen parameters en stuurt de kleinste van de twee terug naar de cliënt in een HDLC UA frame (Tabel 35: Control types). Hierdoor weten de client en de server welke HDLC parameters gebruikt kunnen worden.

| Bytes (hex) | | | betekenis |
|---|---|---|---|
| 81 80 12 | | | format identifier, group identifier, group length 18 |
| | 05 01 | | maximum information field length – transmit tag, length 1 |
| | | 80 | maximum information field length – transmit value, 128 |
| | 06 01 | | maximum information field length – receive tag, length 1 |
| | | 80 | maximum information field length – receive value, 128 |
| | 07 04 | | window size, transmit tag, length 4 |
| | | 00 00 00 01 | window size, transmit value, 1 |
| | 08 04 | | window size, receive tag, length 4 |
| | | 00 00 00 01 | window size, receive value, 1 |

**Tabel 10: Information HDLC parameter frame (Bron: Green book [6])**

### 4.2.3 Beveiligingsafspraken

Voordat er data van de slimme energiemeter opgevraagd kan worden moeten er beveiligingsafspraken worden gemaakt. Via deze beveiligingsafspraken wordt bepaald naar welk veiligheidsniveau de communicatie gaat en tot welke data de cliënt toegang heeft. Protocol mode E kent 3 verschillende niveaus:

- No security

Dit is het laagste veiligheidsniveau en kan gebruikt worden om basis data op te vragen. Voor dit niveau is geen autorisatie of encryptie nodig.

- Low Level Security (LLS)

Voor dit niveau moet de cliënt zich autoriseren. Op dit niveau wordt er wel gebruik gemaakt van encryptie.

- High Level Security (HLS)

Dit is het hoogste veiligheidsniveau. Van uit dit niveau kan de cliënt bij alle data. Voor dit niveau moeten de cliënt en de server zich allebei autoriseren. Ook hier wordt gebruik gemaakt van encryptie.

vanaf welk niveau data beschikbaar is wordt bepaald door de energieleveranciers die de slimme energiemeters leveren.

De Beveiligingsafspraken beginnen met een AARQ bericht vanaf de cliënt. Hierop antwoord de server met een AARE bericht. Als HLS geselecteerd is volgt er een Action-request bericht van de cliënt en een Action-response van de server, zie 4.3 Gebruik Connectie blz. 15. Via deze berichten wordt de autorisatie geregeld.

### AARQ

Het AARQ bericht bevat informatie over het veiligheidsniveau. Hierin staat naar welk veiligheidsniveau de cliënt wil gaan samen met de daarvoor benodigde parameters. Tabel 11 geeft een voorbeeld van een AARQ bericht[2].

| Bytes (hex) | | | | betekenis |
|---|---|---|---|---|
| 60 5d | | | | AARQ tag, length 93 |
| | a1 09 | | | Application context name tag, length 9 |
| | | 06 07 | | OBJECT IDENTIFIER, length 7 |
| | | | 60 85 74 05 08 01 03 | **Application context name** |
| | a6 0a | | | AP-title tag, length 10 |
| | | 04 08 | | OCTET STRING, length 8 |
| | | | 44 4e 56 30 31 32 33 34 | **AP-title** |
| | 8a 02 | | | ACSE-requirements tag, length 2 |
| | | 07 80 | | number of unused bits, **authentication functional unit** |
| | 8b 07 | | | Mechanism-name tag, length 7 |
| | | | 60 85 74 05 08 02 05 | **Mechanism-name** |
| | ac 12 | | | Authentication-value tag, length 18 |
| | | 80 10 | | CHAR STRING, length 16 |
| | | | 4d 2c 3a 53 5d 25 7b 66 41 58 34 36 69 3a 3a 42 | **Authentication-value** |
| | Be 23 | | | AA_AssociationInformation tag, length 35 |
| | | 04 21 | | OCTET STRING, Universal, length 32 |
| | | | 21 1f 30 1e 38 c7 ac f8 da 1a 1a 73 2b 36 fa d0 46 ef 0d 2c b2 11 0e 17 1f a4 b6 d6 05 98 c6 17 67 | **AA_AssociationInformation** (encrypted) |

**Tabel 11: AARQ voorbeeld (Bron: Green book [6])**

**Application context name**
Dit is een vaste byte string waar in het laatste byte het context_id is. Onderstaande Tabel 12: Context_id laat alle mogelijkheden en betekenissen van het Context_id zien.

| Context_id | Betekenis |
|---|---|
| 1 | Logical_Name_Referencing_No_Ciphering |
| 2 | Shortl_Name_Referencing_No_Ciphering |
| 3 | Logical_Name_Referencing_With_Ciphering |
| 4 | Shortl_Name_Referencing_ With _Ciphering |

**Tabel 12: Context_id (Bron: Green book [6])**

[2] Het AARQ bericht moet nog ingepakt worden volgens het HDLC formaat voor dat het verzonden kan worden, zie 4.5 HDLC blz. 22.

**AP-title**
Dit is een unieke constante byte string die voor elk apparaat anders is. Deze byte string wordt gebruikt in het IV in de encryptie, zie 4.6 Security blz. 22.

**Authentication functional unit**
Hier wordt aangegeven of er om autorisatie gevraagd word.

**Mechanism-name**
Dit is een vaste byte string waar in het laatste byte het mechanism_id is. De Onderstaande Tabel 13 laat alle mogelijkheden en betekenissen van het mechanism _id zien.

| Mechanism _id | Betekenis |
|---|---|
| 0 | COSEM_lowest_level_security_mechanism_name |
| 1 | COSEM_low_level_security_mechanism_name |
| 2 | COSEM_high_level_security_mechanism_name |
| 3 | COSEM_high_level_security_mechanism_name_using_MD5 |
| 4 | COSEM_high_level_security_mechanism_name_using_SHA-1 |
| 5 | COSEM_High_Level_Security_Mechanism_Name_Using_GMAC |

**Tabel 13: Mechanism _id (Bron: Green book [6])**

**Authentication-value**
Dit is een onwillekeurige byte string die elke keer anders is. Deze byte string wordt gebruikt als input voor het autorisatie mechanisme, (zie Action-request blz. 21).

**AA_AssociationInformation**
Dit veld bestaat uit een encrypted InitiateRequest, zie 4.6 Security blz. 22. Hierin staan de voorgestelde Context_parameters. Tabel 14 geeft een voorbeeld van een InitiateRequest.

| Initiate-Request tag | dedicated-key | response-allowed | proposed-quality-of-service | proposed-dlms-version-number | proposed-conformance | client-max-receive-pdu-size |
|---|---|---|---|---|---|---|
| 01 | 00 | 00 | 00 (not used) | 06 | 5F 1F 04 00 00 1F 1F | 08 00 |

**Tabel 14: InitiateRequest (Bron: Green book [6])**

## AARE

Als antwoord op het AARQ bericht stuurt de server een AARE bericht. Het AARE bericht lijkt heel erg op het AARQ bericht en heeft veel van dezelfde velden. Tabel 15 geeft een voorbeeld van een AARE bericht.

| Bytes | | | | Betekenis |
|---|---|---|---|---|
| 61 54 | | | | AARQ tag, length 93 |
| | a1 09 | | | Application context name tag, length 9 |
| | | 06 07 | | OBJECT IDENTIFIER, length 7 |
| | | | 60 85 74 05 08 01 03 | **Application context name** |
| | a2 03 | | | Association-result, length 3 |
| | | 02 01 | | INTEGER, length 1 |
| | | | 00 | **Association-result** |
| | a3 05 | | | Associate-source-diagnostic, length 5 |
| | | a1 03 | | ACSE-service-user tag, length 3 |
| | | | 02 01 0e | INTEGER, length 1, **Associate-source-diagnostic** |
| | a4 0a | | | AP-title, length 10 |
| | | 04 08 | | OCTET STRING, length 8 |
| | | | 4c 47 42 00 00 00 08 af | **AP-title** |
| | aa 0a | | | Authentication-value, length 10 |
| | | 80 08 | | CHARSTRING tag, length 8 |
| | | | 67 52 1c fc 14 47 2a 0b | **Authentication-value** |
| | Be 23 | | | AA_AssociationInformation tag, length 35 |
| | | 04 21 | | OCTET STRING, Universal, length 32 |
| | | | 28 1f 30 00 00 05 4d 71 13 bf 33 46 04 af 51 fb 2f ad a7 c5 c0 8b bf be 21 ef ec a9 cd f3 a7 68 2c | **AA_AssociationInformation** (encrypted) |

**Tabel 15: AARE voorbeeld (Bron: Green book [6])**

**Application context name**
Dit is net als in het AARQ een vaste byte string waar in het laatste byte het context_id is. Tabel 12 laat alle mogelijkheden en betekenissen van het Context_id zien. In het AARE bericht geeft de Application context name aan of de server akkoord gaat met het door de cliënt voorgestelde veiligheidsniveau.

**Association-result**
Dit veld is 1 byte lang en geeft aan of de server akkoord gaat met alle voorgestelde parameters van de cliënt. Tabel 16 geeft alle mogelijkheden en betekenissen van deze byte.

| byte | betekenis |
|---|---|
| 0 | Accepted |
| 1 | Rejected-permanent |
| 2 | Rejected-transient |

**Tabel 16: Association-result (Bron: Green book [6])**

**Associate-source-diagnostic**
Deze bytes geven bij een niet succesvolle AARE aan wat de fout is. Bij een succesvolle AARE kunnen ze aangeven wat nog nodig is. Denk hierbij aan autorisatie. Associate-source-diagnostic bestaat uit twee bytes waarin het eerste byte aangeeft of het een ACSE-service-user of een ACSE -service-provider source-diagnostic is. Zie Tabel 17 voor alle mogelijkheden en betekenissen.

| Byte | Betekenis | Byte | Betekenis |
|---|---|---|---|
| ACSE-service-user, 1 | | ACSE -service-provider, 2 | |
| 0 | Null | 1 | Null |
| 1 | No-reason-given | 2 | No-reason-given |
| 2 | Application-context-name-not-supported | 3 | no-common- ACSE -version |
| 11 | Authentication-mechanism-name-not-recognized | | |
| 12 | Authentication-mechanism-name-required | | |
| 13 | Authentication-failure | | |
| 14 | Authentication-required | | |

**Tabel 17: Associate-source-diagnostic (Bron: Green book [6])**

**AP-title**
Dit is net als in de AARQ een unieke constante byte string die voor elk apparaat anders is. Deze byte string wordt gebruikt in het IV in de encryptie, zie 4.6 Security blz. 22.

**Authentication-value**
Dit is net als in de AARQ een onwillekeurige byte string die elke keer anders is. Deze byte string wordt gebruikt als input voor het autorisatie mechanisme, zie Action-request blz. 14.

**AA_AssociationInformation**
Dit veld bestaat uit een encrypted InitiateResponse, Zie 4.6 Security blz. 22. Hierin staat het antwoord op de voorgestelde Context_parameters. Tabel 18 geeft een voorbeeld van een InitiateResponse.

| Initiate-Request tag | proposed-quality-of-service | proposed-conformance | server-max-receive-pdu-size | vaa-name |
|---|---|---|---|---|
| 08 | 00 (not used) | 5F 1F 04 00 00 1A 1F | 04 B0 | 00 07 |

**Tabel 18: InitiateResponse (Bron: Green book [6])**

### Action-request

Voor Low Level Security en High Level Security moet de cliënt zich autoriseren. In de AARE heeft de server een Authentication-value meegestuurd die als input dient voor het autorisatie mechanisme. De output van het autorisatie mechanisme wordt vervolgens via een Action-request terug naar OBIS-tabel 0x000F op de server gestuurd, zie 5.3.2 Get/Set/Action blz. 30.

Voor het autorisatie mechanisme wordt AES-GCM-128 gebruikt. Met de volgende input waardes:

| | |
|---|---|
| P | leeg |
| K | encryption key |
| AAD | security-Control-field (SC) + frame-counter + Authentication-value (AARE) |
| IV | system-title + frame-counter |

En de volgende output waardes:

| | |
|---|---|
| CT | (is leeg omdat P leeg is) |
| TAG | |

Vervolgens wordt SC + framecounter + TAG via het Action-request naar de server gestuurd. Zie 4.6 Security blz. 22 voor meer informatie over AES-GCM-128.

### Action-response

Voor High Level Security moet de server zich ook autoriseren. In de AARQ heeft de cliënt een Authentication-value meegestuurd die als input dient voor het autorisatie mechanisme. De output van het autorisatie mechanisme wordt vervolgens via een Action-response terug naar de cliënt gestuurd, zie 5.3.2 Get/Set/Action blz. 30.

Het antwoord van de server kan gecheckt worden via AES-GCM-128 door het te decrypten met als input waarden:

| | |
|---|---|
| TAG | TAG uit Action-response |
| CT | zelf encrypted string |
| K | encryption key |
| AAD | security-Control-byte + frame-counter + Authentication-value(AARQ) |
| IV | system-title + frame-counter |

Hierop krijgen we de volgende output waarden:

| | |
|---|---|
| P | decrypted string, autorisatie goed |
| Of | |
| P | null, autorisatie fout. |

Zie 4.6 Security blz. 22 voor meer informatie over AES-GCM-128.

## 4.3 Gebruik Connectie

In hoofdstuk 9.5 van het Green book [6] is een lijst van alle commando's te vinden. Zoals al eerder vermeld is deze lijst te groot om geheel te implementeren binnen een afstudeer periode. Er is gekozen om te focussen op de Get, Set en Action commando's en de encrypted versies hiervan. Met gebruik van deze commando's kunnen de attributes van de OBIS tabellen (zie Tabel 2) opgehaald en gezet worden en kunnen de specific methods worden aangeroepen.

Om aan te geven dat het informatie veld in een HDLC frame (zie HDLC blz. 20) een bericht is, worden bytes 0xE6, 0xE6 en 0x00 voor elk bericht gezet.

### Get-request

Get-request wordt gebruikt voor het opvragen van attributes uit OBIS tabellen, zie Tabel 2. Tabel 19 geeft een voorbeeld van een Get-request. Het Get-request begint met een Tag van 2 bytes lang dat aangeeft dat dit bericht een Get-request van het type normal is. Het volgende byte is het Invoke-id and Priority byte. Zoals in Tabel 20 te zien is, bestaat deze byte uit meerdere velden: Invoke-id, Reserved, Service class en Priority. Voor alle niet encrypted berichten is deze byte $81_H$ en voor alle encrypted berichten $C1_H$. Het OBIS, Class_id samen met de Logical Name geven aan voor welke OBIS tabel het Get-request bedoeld is, en het Attribute veld voor welke attribute binnen die OBIS tabel.

| Get-request Normal Tag | | Invoke-id and Priority | OBIS, Class_id | Logical Name | Attribute | |
|---|---|---|---|---|---|---|
| C0 | 01 | 81/C1 | 2 Bytes | 6 Bytes | 1 Byte | 00 |

**Tabel 19: Get-request voorbeeld (Bron: Green book [6])**

| Invoke-id | | | | Reserved | | Service class | Priority |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

**Tabel 20: Invoke-id and Priority (Bron: Green book [6])**

### Get-response

Als antwoord op een Get-request van de cliënt stuurt de slimme energiemeter een Get-response, zie Tabel 21. Een Get-response begint met een Get-response normal tag wat aangeeft dat dit een Get-response van het type normal is. Het volgende byte is net als bij een Get-request het Invoke-id and Priority byte, zie Tabel 20. Net als bij een Get-request is deze byte voor alle niet encrypted berichten $81_H$ en voor alle encrypted berichten $C1_H$. Daarna volgt het result byte. Deze byte geeft aan of het Get-request succesvol was of niet. Tabel 22 laat alle mogelijkheden en betekenissen van deze byte zien. Als het Get-request succesvol was volgt er een data-type byte. Deze byte geeft aan wat voor type data het Get-response bevat. Alle mogelijke datatypes en bijhorende data-type bytes zijn te vinden in Tabel 23. Na het data-type byte volgt het length veld dat aangeeft hoe lang het daarop volgend data veld is. Het data veld bevat de data uit de aangevraagde OBIS tabel.

| Get-response Normal Tag | | Invoke-id and Priority | Result | Data-type | length | data |
|---|---|---|---|---|---|---|
| C4 | 01 | 81/C1 | 1 Byte | 1 Byte | 1 Byte | N Bytes |

**Tabel 21: Get-response voorbeeld (Bron: Green book [6])**

| Result byte (dec) | betekenis | Result byte (dec) | betekenis |
|---|---|---|---|
| 0 | Success | 13 | Scope-of-access-violated |
| 1 | Hardware-fault | 14 | Data-block-unavailable |
| 2 | Temporary-failure | 15 | Long-get-aborted |
| 3 | Read-write-denied | 16 | No-long-get-in-progress |
| 4 | Object-undefined | 17 | Long-set-in-progress |
| 9 | Object-class-inconsistent | 18 | No-long-set-in-progress |
| 11 | Object-unavailable | 19 | Data-block-number-invalid |
| 12 | Type-unmatched | 250 | Other-reason |

**Tabel 22: Data-Access-Result (Bron: Green book [6])**

| Byte (dec) | Name | Data- type | Byte (dec) | Name | Data- type |
|---|---|---|---|---|---|
| 0 | Null-data | Null | 17 | Unsigned | Unsigned8 |
| 1 | Array | Multiple Data | 18 | Long-unsigned | Unsigned16 |
| 2 | Structure | Multiple Data | 19 | compact-array: description, content | Data, Byte-string |
| 3 | Boolean | Boolean | | | |
| 4 | Bit-string | Bit-string | 20 | Long64 | Integer64 |
| 5 | Double-long | Integer32 | 21 | Long64-unsigned | Unsigned64 |
| 6 | Double-long-unsigned | Unsinged32 | 22 | Enum | Unsigned8 |
| 7 | Floating-point | 4 Bytes | 23 | Float32 | 4 Bytes |
| 9 | Octet-string | Byte-string | 24 | Float64 | 8 Bytes |
| 10 | Visible-string | Visible-string | 25 | Date-time | 12 Bytes |
| 13 | BCD | Integer8 | 26 | Date | 5 Bytes |
| 15 | Integer | Integer8 | 27 | Time | 4 Bytes |
| 16 | Long | Integer16 | 255 | Don't-care | Null |

**Tabel 23: Data-type (Bron: Green book [6])**

### Set-request

Set-request wordt gebruikt voor het zetten of aanpassen van OBIS table attributes, zie Tabel 2. Het Set-request begint met een 2 byte lang Set-request normal tag. Deze twee bytes geven aan dat dit een Set-request bericht is van het type normal. Daarna volgt het invoke-id and priority byte, zie Tabel 20. Voor alle niet encrypted berichten is deze byte $81_H$ en voor alle encrypted berichten $C1_H$. De OBIS, Class_id; Logical Name en Attribute bytes geven aan welke attribute van welke OBIS tabel er gezet moet worden. Het Data-type en length byte geven aan van welk type de data is en hoe groot de data is die gezet moet worden. Zie Tabel 23 voor alle data types. Als laatste volgt het data veld waar zich de data bevind die in de OBIS tabel gezet moet worden. Tabel 24 geeft de opbouw van een Set-request.

| Set-request Normal Tag | | Invoke-id and Priority | OBIS, Class_id | Logical Name | Attribute | | Data-type | length | data |
|---|---|---|---|---|---|---|---|---|---|
| C1 | 01 | 81/C1 | 2 Bytes | 6 Bytes | 1 Byte | 00 | 1 Byte | 1 Byte | N Bytes |

**Tabel 24: Set-request (Bron: Green book [6])**

### Set-response

Als antwoord op een Set-request stuurt de slimme energiemeter een Set-response, zie Tabel 25. In dit bericht staat of de Set-request succesvol was of niet. Het bericht begint met een Set-response normal tag. Dit zijn 2 bytes die aangeven dat dit bericht een Set-response van het type normal is. Daarna volgt het invoke-id and priority byte, zie Tabel 20. Voor alle niet encrypted berichten is deze byte $81_H$ en voor alle encrypted berichten $C1_H$. Tenslotte volgt het result veld, dit is een 1 byte groot veld dat aangeeft of de voorgaande Set-reqeust succesvol was. Zie Tabel 22 voor alle result mogelijkheden.

| Set-response Normal Tag | | Invoke-id and Priority | Result |
|---|---|---|---|
| C5 | 01 | 81/C1 | 1 Byte |

**Tabel 25: Set-response (Bron: Green book [6])**

### Action-request

Het Action-request bericht wordt gebruikt om methods van OBIS tabellen uit te voeren, zie Tabel 2 voor meer informatie over methods. Tabel 26 geeft een overzicht van de opbouw van een Action-request. De eerste 2 bytes zijn de Action-request normal tag. Deze bytes geven aan dat het bericht een Action-request van het type normal is. Daarna volgt het invoke-id and priority byte, zie Tabel 20. Voor alle niet encrypted berichten is deze byte $81_H$ en voor alle encrypted berichten $C1_H$. De OBIS, Class_id; Logical Name en Method bytes geven aan welke Method van welke OBIS tabel er aangeroepen moet worden. Aan de meeste methods kan data worden mee gegeven als argument. Het data-type en de lengte worden aangegeven door de data-type en length bytes. Na deze bytes volgt de data. Zie Tabel 23 voor alle data types.

| Action-request Normal Tag | | Invoke-id and Priority | OBIS, Class_id | Logical Name | Method | | Data-type | length | data |
|---|---|---|---|---|---|---|---|---|---|
| C3 | 01 | 81/C1 | 2 Bytes | 6 Bytes | 1 Byte | 0 | 1 Byte | 1 Byte | N Bytes |

**Tabel 26: Action-request (Bron: Green book [6])**

### Action-response

Als antwoord op een Action-request stuurt de slimme energiemeter een Action-response. Dit bericht geeft aan of het voorgaande Action-request succesvol was. Ook kan dit bericht data bevatten als de hiervoor aangeroepen method data terug geeft. Tabel 27 laat de opbouw van een Action-response zien. De eerste twee bytes staan voor de Action-response normal tag en geven aan dat dit bericht een Action-response van het type normal is. Daarna volgt het invoke-id and priority byte, zie Tabel 20. Voor alle niet encrypted berichten is deze byte $81_H$ en voor alle encrypted berichten $C1_H$. Het volgende byte geeft aan of de Action-request succesvol was, Tabel 28 geeft alle mogelijkheden en betekenissen voor deze byte. De Data-type, length en data bytes zijn optioneel en zijn alleen aanwezig als de aangeroepen method data terug geeft. Data-type is 1 byte groot en geeft aan van welk type de data is, zie Tabel 23 voor alle data types. Length is ook 1 byte groot en geeft aan hoe lang het hier op volgende data veld is. Het Data veld bevat de data dat door de method terug gegeven wordt.

| Action-response Normal Tag | | Invoke-id and Priority | Result | Data-type | length | data |
|---|---|---|---|---|---|---|
| C7 | 01 | 81/C1 | 1 Byte | 1 Byte | 1 Byte | N Bytes |

**Tabel 27: Action-response (Bron: Green book [6])**

| Result byte (dec) | betekenis | Result byte (dec) | betekenis |
|---|---|---|---|
| 0 | Success | 12 | Type-unmatched |
| 1 | Hardware-fault | 13 | Scope-of-access-violated |
| 2 | Temporary-failure | 14 | Data-block-unavailable |
| 3 | Read-write-denied | 15 | Long-action-aborted |
| 4 | Object-undefined | 16 | No-long-action-in-progress |
| 9 | Object-class-inconsistent | 250 | Other-reason |
| 11 | Object-unavailable | | |

**Tabel 28: Action-Result (Bron: Green book [6])**

## 4.4 Afsluiten connectie

De verbinding met de slimme energiemeter kan worden afgesloten door 30 seconden geen bericht te verzenden of door de afsluitberichten te sturen. Via een RLRQ bericht vraagt de cliënt het einde van de connectie aan. De slimme energiemeter antwoord hierop met een RLRE bericht. Als laatst wordt ook de datalink laag gesloten door een HDLC DISC frame te sturen, zie Tabel 35.

### RLRQ

Het RLRQ bericht is het eerste bericht van de afsluitberichten. In dit bericht staat de reden van afsluiting en de AssociationInformation. Net als alle andere berichten wordt dit bericht verzonden in een HDLC frame, zie 4.5 HDLC blz. 20.

| Bytes | | | | Betekenis |
|---|---|---|---|---|
| 61 17 | | | | RLRQ tag, length 23 |
| | 81 01 | | | AA_ReleaseRequestReason tag, length 1 |
| | | 00 | | **AA_ReleaseRequestReason** |
| | Be 12 | | | AA_AssociationInformation tag, length 18 |
| | | 04 10 | | OCTET STRING, Universal, length 16 |
| | | | 21 1F 30 1E 38 C7 B1 E9 92 A3 7F 60 BE E7 6E 37 | **AA_AssociationInformation** (encrypted) |

**Tabel 29: RLRQ voorbeeld (Bron: Green book [6])**

**AA_ReleaseRequestReason**
In dit veld staat de reden van afsluiting. Tabel 30 laat alle mogelijkheden en betekenissen van dit veld zien.

| AA_ReleaseRequestReason | Betekenis |
|---|---|
| 0 | Normal |
| 1 | Urgent |
| 30 | User-defined |

**Tabel 30: AA_ReleaseRequestReason (Bron: Green book [6])**

**AA_AssociationInformation**
Dit veld bestaat uit een encrypted InitiateRequest, zie 4.6 Security blz. 22. Hierin staan de Context_parameters. Tabel 14 geeft een voorbeeld van een InitiateRequest.

### RLRE

In antwoord op het RLRQ bericht van de cliënt stuurt de slimme energiemeter een RLRE bericht. Dit bericht bevat geen extra data en geeft aan dat de verbinding afgesloten kan worden.

| Bytes | | | | Betekenis |
|---|---|---|---|---|
| 63 00 | | | | RLRE tag, length 0 |

**Tabel 31: RLRE voorbeeld (Bron: Green book [6])**

### DISC

Als laatste bericht wordt er een leeg HDLC DISC naar de slimme energiemeter gestuurd om ook de datalink laag te sluiten, Zie Tabel 35. Na het zenden van dit bericht is de verbinding afgesloten.

## 4.5 HDLC

In protocol mode E wordt gebruik gemaakt van HDLC als datalink laag. HDLC staat voor High-Level Data Link Control en wordt gebruikt voor het inpakken van de mode E berichten. Mode E gebruikt HDLC frame format type 3. Onderstaande Tabel 32 laat de opbouw van een HDLC frame format type 3 zien [6].

| Flag | Frame format | Dest. address | Src. address | Control | HCS | Information | FCS | Flag |
|---|---|---|---|---|---|---|---|---|
| 7E | 2 Byte | 1,2 of 4 Bytes | 1,2 of 4 Bytes | 1 Byte | 2 Byte | n Bytes | 2 Byte | 7E |

**Tabel 32: HDLC frame type 3 (Bron: Green book [6])**

### Flag

De Flag geeft het begin en het einde van het frame aan en heeft altijd de waarde 0x7E. Deze waarde kan ook in de rest van het bericht voorkomen. 0x7E kan dus niet gebruikt worden om het begin en einde van het bericht te vinden, maar kan wel gebruikt worden om de frame grootte te controlleren.

### Frame format

Frame format bestaat uit twee bytes waarin het frame type, segmentation bit en frame length staan. Onderstaande Tabel 33 geeft de opbouw van frame format weer.

| Format type | | | | Seg. | Frame length sub-field | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | S | L | L | L | L | L | L | L | L | L | L | L |

**Tabel 33: Frame format (Bron: Green book [6])**

**Format type**
Een vier bit lang gebied waarin het frame type staat, Bij de slimme energiemeters is dit altijd 0b1010 wat frame type 3 aangeeft.

**Segmentation bit (Seg.)**
Mode E berichten kunnen worden verdeeld over meerdere HDLC frames. Het laatste frame in een reeks wordt verstuurd met het segmentation bit 0. Ook als het bericht bestaat uit een enkel HDLC frame wordt het segmentation bit op 0 gezet.

**Frame length sub-field**
Een 11 bit lang veld dat de lengte van het gehele frame zonder de begin/eind flag aan geeft, zie Tabel 32.

### Destination (Dest.) Address & Source (Src.) Address

In deze velden staan de Destination en Source adressen. Afhankelijk van de richting van het frame (cliënt naar slimme energiemeter of slimme energiemeter naar cliënt) worden deze velden omgedraaid. Het adres van de cliënt is altijd 1 byte groot. Het adres van de slimme energiemeter kan 1,2 of 4 bytes groot zijn, zie Tabel 34.

Het adres van de slimme energiemeter kan worden opgebouwd uit een higher en een lower adres. Het higher adres wordt gebruikt om een Logical Device in de slimme meter te adresseren en het lower adres wordt gebruikt om een fysieke meter te selecteren.

| First byte | | Second byte | | Third byte | | Fourth byte | |
|---|---|---|---|---|---|---|---|
| Upper HDLC address | 1 | | | | | | |
| Upper HDLC address | 0 | Lower HDLC address | 1 | | | | |
| Upper HDLC address | 0 | Upper HDLC address | 0 | Lower HDLC address | 0 | Lower HDLC address | 1 |

**Tabel 34: Slimme energiemeter adres opbouw (Bron: Green book [6])**

### Control

Het control veld is 1 byte lang en geeft aan wat voor soort type het HDLC frame is. Tabel 35 laat alle mogelijke control bytes zien.

| Command | Response | MSB LSB | meaning |
|---|---|---|---|
| I | I | R R R P/F S S S 0 | Information |
| RR | RR | R R R P/F 0 0 0 1 | Receive Ready |
| RNR | RNR | R R R P/F 0 1 0 1 | Receive Not Ready |
| SNRM | | 1 0 0 P 0 0 1 1 | Set Normal Response Mode |
| DISC | | 0 1 0 P 0 0 1 1 | Disconnect |
| | UA | 0 1 1 F 0 0 1 1 | Unnumbered Acknowledgment |
| | DM | 0 0 0 F 1 1 1 1 | Disconnected Mode |
| | FRMR | 1 0 0 F 0 1 1 1 | Frame Reject |
| UI | UI | 0 0 0 P/F 0 0 1 1 | Unnumbered Information |

**Tabel 35: Control types (Bron: Green book [6])**

**R R R**
Een 3 bit teller die optelt, elke keer als er frames ontvangen worden.

**S S S**
Een 3 bit teller die optelt, elke keer als er frames verzonden worden

**P/F**
Poll/final bit, geeft aan dat de zender klaar is met zenden.

### HCS

HCS is een 2 byte header checksom, die wordt toegepast op alle bytes tussen de begin flag en de HCS. Als frames geen informatieveld hebben wordt de HCS weggelaten en vervangen door de FCS. HCS wordt op de zelfde manier berekend als FCS.

### FCS

FCS is een 2 byte frame checksom, die wordt toegepast op alle bytes tussen de begin flag en de FCS. In Hoofdstuk 8.5 van het Green Book [6] is voorbeeld code te vinden over hoe de FCS berekend wordt.

### Information

Het informatie veld bevat het bericht, zie 4.3 Gebruik Connectie blz. 15.

## 4.6 Security

De meeste data op de slimme energiemeter is beveiligd en kan niet zomaar opgevraagd worden. Bij het opzetten van de connectie wordt geselecteerd naar welk beveiligingsniveau de connectie gaat, zie Tabel 13. Bij LLS en HLS worden de berichten ge-encrypt. Volgens DSMR-M 4.4.22 [4] moeten alle Nederlandse slimme energiemeters AES-128 gebruiken voor de encryptie. Ook moeten volgens DSMR-M 4.4.8a [4], alle Nederlandse slimme energiemeters mechanism_id(3,4 en 5) ondersteunen, zie Tabel 13. Omdat mechanism_id(5) GMAC ook via AES-GCM-128 werkt en Sogeti alleen de keys van dit autorisatie mechanisme heeft is er besloten om alleen op dit autorisatie mechanisme te focussen.

### AES-GCM-128

AES-GCM-128 heeft 4 inputs en 2 outputs. Tabel 36 laat de inputs zien en Tabel 37 de outputs van het AES-GCM-128 mechanisme zien.

| Input | Betekenis |
|---|---|
| K | Encryption key |
| P | Plaintext |
| AAD | additional authenticated data |
| IV | initialization vector |

**Tabel 36: AES-GCM-128 inputs (Bron: Green book [6])**

| Output | Betekenis |
|---|---|
| CT | Cyphertext |
| TAG | Authentication tag |

**Tabel 37: AES-GCM-128 outputs (Bron: Green book [6])**

K en AAD zijn geheime byte strings (keys) die de normale gebruiker van de slimme energiemeter niet weet. Sogeti heeft van een aantal slimme energiemeters deze byte strings gekregen en kan daardoor een LLS en HLS verbinding opzetten.

P is het bericht dat geencrypt moet worden. Dit kan elk soort bericht zijn waaronder de berichten beschreven in 4.3 Gebruik Connectie blz. 15.

Voor het IV wordt de System-title + frame-counter gebruikt. De system-title is een constante unieke byte string van 8 bytes lang. De frame-counter is een 4 byte grote teller die na elk bericht optelt. Door de System-title en frame-counter is het IV voor elk bericht van elke slimme energiemeter uniek. De system-title wordt in het AARQ en AARE meegestuurd, zie 4.2.3 Beveiligingsafspraken blz. 9. De frame-counter zit in elk encryptie frame, zie Tabel 38.

### Encryptie frame

Bijna alle berichten van mode E waaronder de berichten in 4.3 Gebruik Connectie kunnen ge-encrypt verzonden worden. Al deze encrypted berichten hebben de opbouw zoals in Tabel 38 te zien is. De encrypted berichten moeten nog wel in een HDLC frame worden gestopt voordat ze verzonden kunnen worden, zie 4.5 HDLC blz. 20.

| Message tag, length | SC | Frame-counter | Cyphertext | Authentication tag |
|---|---|---|---|---|
| 2 bytes | 1 Byte | 4 bytes | N Bytes | 12 Bytes |

**Tabel 38: Encryption frame (Bron: Green book [6])**

**Message tag, length**
Het eerste byte van dit twee byte grote veld geeft aan wat voor type bericht dit frame bevat, zie Tabel 39. In Tabel 39 zijn alleen de tags te zien die in dit project gebruikt worden. In 9.5 van het Green book [6] is een volledige lijst van alle tags te vinden. Het tweede byte van dit veld geeft aan hoe lang de rest van het encryption frame is.

| Message tag | Encrypted Message Type |
|---|---|
| C8 | Get-request |
| CC | Get-response |
| C9 | Set-request |
| CD | Set-response |
| CB | Action-request |
| CF | Action-response |
| 21 | Initiate-request |
| 28 | Initiate-response |

**Tabel 39: Encrypted Message Type Tags (Bron: Green book [6])**

**SC**
SC staat voor Security Control field. In deze byte wordt aangegeven of het Encryption frame encrypted data, authenticated data of een combinatie van de twee bevat. Tabel 40 geeft een overzicht van deze byte. Alleen bit 4 en 5 worden gebruikt.

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3…0 |
|---|---|---|---|---|
| Reserved, set to 0 | Key_set | Encrypted data | authenticated data | Security_Suite_Id |

**Tabel 40: Security Control field (Bron: Green book [6])**

**Frame-counter**
Dit is een 4 byte grote teller die na elk bericht optelt en is onderdeel van het IV.

**Cyphertext**
Dit is de encrypted data die uit het AES-GCM-128 mechanisme komt.

**Authentication tag**
Dit is de Authentication tag die uit het AES-GCM-128 mechanisme komt.

# 5 P0 Software

De P0 communicatie software moet gaan draaien op de SMA. SMA staat voor Smart Meter Aggregator en is de kern van de meet applicatie. Het hardware platform waar de SMA op draait is geselecteerd door mede afstudeerder Marc Welleweerd. Uit zijn onderzoek [9] kwam de DE1-SoC van Altera als beste naar voren. De DE1-SoC bestaat uit een dual-core Cortex-A9 in combinatie met een FPGA. Op de Cortex-A9 draait een Linux distributie waarop de P0 communicatie software moet gaan draaien.

De hoofdvraag van deze scriptie is hoe de SMA met meerdere P0 poorten kan communiceren. Omdat de hardware van de SMA pas aan het einde van de afstudeerperiode beschikbaar was, is er voor gekozen om de code eerst op een Windows machine werkend te krijgen en daarna om te zetten naar het gekozen hardware platform voor de SMA.

Voor het aanpassen en compilen van de software op de SMA is er gebruik gemaakt van Netbeans [10]. Netbeans is een IDE waarmee er remote developed kan worden. Dit betekent dat er op de Sogeti bedrijfslaptop code kan worden geschreven en dat deze code automatisch op de SMA compiled en uitgevoerd word.

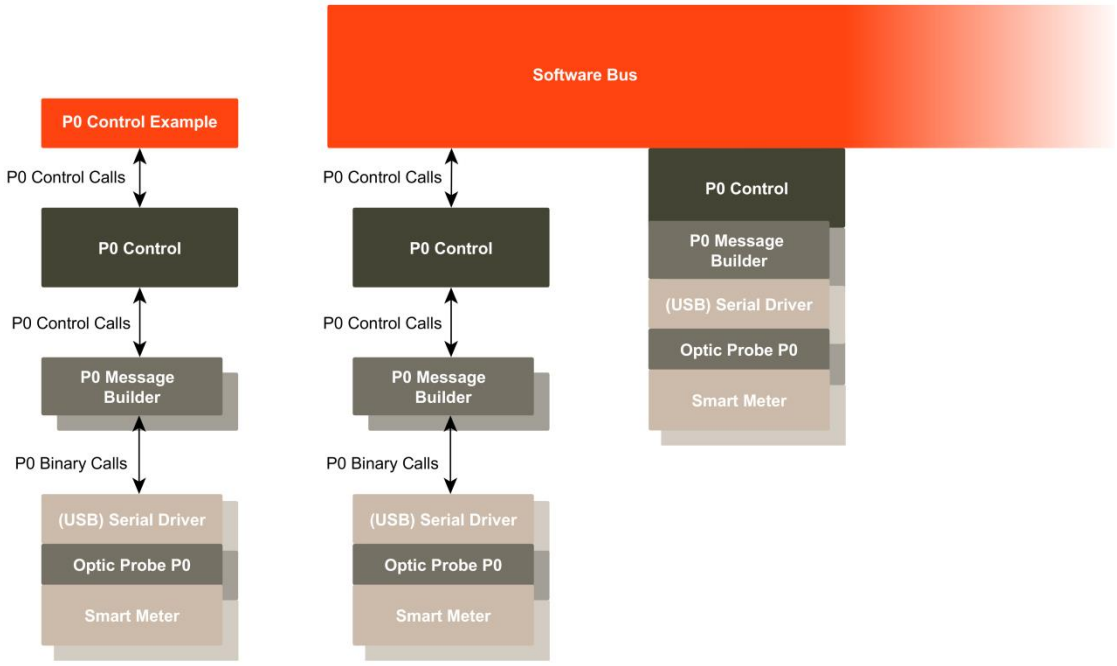
De code die in dit verslag besproken wordt is maar een klein deel van wat er uiteindelijk op de SMA komt te draaien. Omdat de code die met P0 communiceert ook met de rest van de SMA moet communiceren is er samen met de andere leden van het slimme meter project een SMA software structuur opgezet, zie Bijlage A: SMA software architectuur. Deze architectuur geeft een overzicht van welke software er op de SMA draait en welk deel er met welk deel communiceert.

## 5.1 Globaal overzicht software P0

Figuur 5 laat een deel van de totale architectuur zien met betrekking op de P0 communicatie software. Zoals te zien is, is de P0 communicatie stack opgebouwd uit 4 lagen: P0 Control, P0 Message Builder, Serial Driver en Optic Probe.

Via de Service Bus krijgt het P0 Control blok P0 Control Calls van een hoger liggend stuk software. Het P0 Control blok kan op basis van deze Control Calls nieuwe P0 Stack Instanties aanmaken, verwijderen of aansturen. Als er bijvoorbeeld een Get–request voor P0 instantie 1 binnen komt kijkt het P0 control blok of deze instantie aanwezig is en geeft het daarna door. Het Get-request komt dan aan in het P0 Message Builder blok van instantie 1. Het P0 Message Builder blok zet P0 Control Calls om in P0 Binary Calls. P0 Binary Calls zijn de bytes die naar de slimme energiemeter verzonden worden. Eenmaal opgebouwd worden de P0 Binary Calls verzonden via de Serial Driver en de optische probe naar de slimme meter. Deze opbouw van objecten geeft antwoord op de subvraag: “Hoe kan een programma communiceren met meerdere P0 poorten?”. En geeft gedeeltelijk antwoord op de subvraag: “Hoe kan de communicatie met P0 gerealiseerd worden op de SMA in een open ended en object-oriented manier?”. De volledige P0 code is te vinden in Bijlage C tot I.

De Software bus is nog niet geschreven en zal gemaakt worden door een andere stagiair. Om een voorbeeld te geen hoe de software bus gebruik kan maken van de P0 software is er een P0 Control Example blok gemaakt. Dit blok dient als voorbeeld over hoe het P0 Control blok gebruikt kan worden. Dit blok kan later in het project vervangen worden door de software van de software bus.




**Figuur 5: P0 communicatie stack overzicht**

## Errors

Omdat bijna elke functie van de P0 communicatie software af te leiden is van een aanroep over de software bus waar weer een antwoord op terug moet, is de error handeling erg simpel. Bij een error in een laag liggend blok wordt via een throw catch mechanisme, de error naar de software laag er boven gebracht. De hogere laag kijkt of de error kritisch is (bijna altijd), voegt data toe (zoals van welke meter de error komt) en brengt het weer via een throw catch mechanisme naar een hogere software laag. Eenmaal bij de hoogste laag aan gekomen, gaat de error de software bus op, als ontwoord op de oorspronkelijk aangeroepen functie.

## Logging

Door gebruik te maken van defines kan de output op verschillende niveaus geregeld worden. Ook kan als later besloten wordt dat alle errors naar een bestand moeten worden geschreven, dit simpel gedaan worden door de defines aan te passen. De rest van de code hoeft dan niet aan gepast te worden. Zie Code voorbeeld 1.

```
#define debug    3
#define info     2
#define warning  1
#define error    0

#define log_level debug
#define LOG(Lev) if (log_level >= Lev) std::cout
#define D(x) if (log_level == debug) x

LOG(warning) << "warning: Somethings wrong" << endl;
LOG(debug) << "First byte: " << firstByte << endl;

D(functionOnlyUsedInDebug());
```

**Code voorbeeld 1: Logging**

## 5.2 Gebruik

De P0 communicatie software is ontworpen zodat het op twee manieren gebruikt kan worden: Standaard gebruik en de Low Level gebruik.

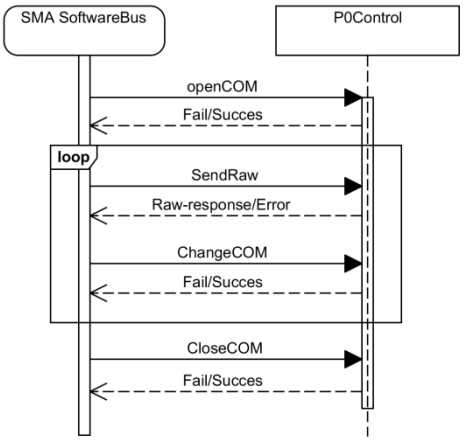
### Standaard gebruik

Bij het standaard gebruik wordt er gebruik gemaakt van de mode E functies: OpenModeE, Get, Set, Action, SendRaw, Stream, CloseStream en CloseModeE, zie 4 Werking mode E blz. 5.

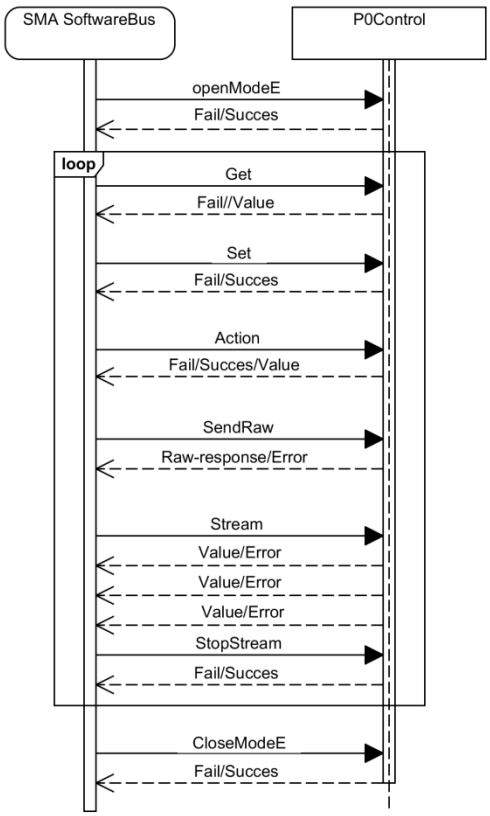
Via OpenModeE wordt de verbinding opgezet. Daarna kan er via Get, Set, Action, SendRaw en Stream met de slimme energiemeter gecommuniceerd worden. De verbinding wordt uiteindelijk weer afgesloten via de CloseModeE functie. Zie Figuur 7 voor een standaard gebruik voorbeeld.

### Low Level gebruik

Bij het low level gebruik wordt er gebruik gemaakt van OpenCOM, SendRaw, ChangeCOM en CloseCOM. Via deze functies is er tot op het byte niveau controle over wat en hoe er naar de slimme meter wordt gestuurd. De communicatie beperkt zich hierdoor niet tot alleen mode E, zie 3.1 Protocol modes blz. 4. Het low level gebruik geeft antwoord op de subvraag: "Hoe kan de communicatie met P0 gerealiseerd worden op de SMA in een open ended en object-oriented manier?". Omdat de communicatie met de slimme energiemeter via op manier open ended is.



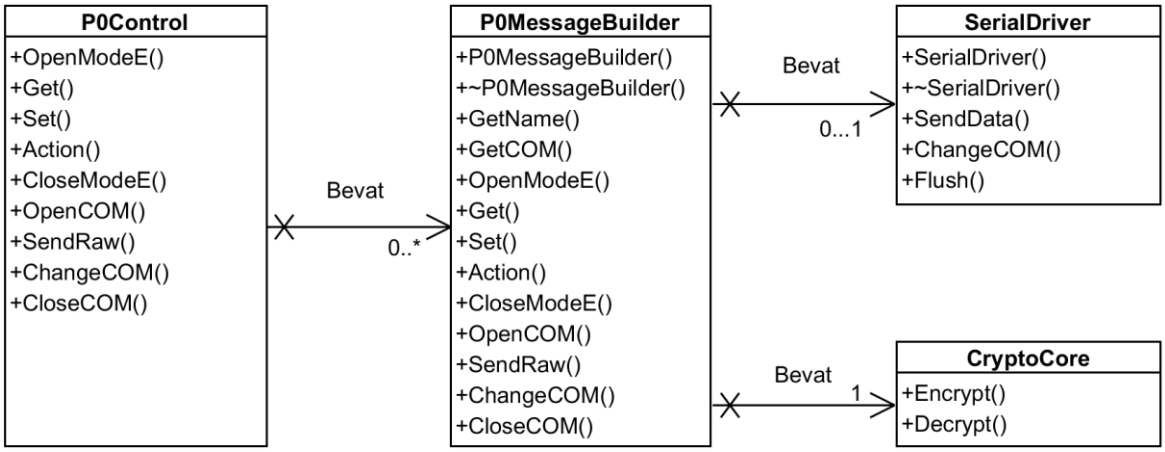

**Figuur 6: Low level gebruik voorbeeld**




**Figuur 7: Standaard gebruik voorbeeld**

## 5.3 Uitleg functies

In dit stuk worden de public functies van het P0 control object en de onderliggende public functies van de andere objecten uitgelegd. Dit is nodig zodat er documentatie is over hoe elk object gebruikt kan worden en mogelijk later in het doorlopende project ergens anders ingezet kan worden. Figuur 8 laat de samenhang van de verschillende objecten zien.



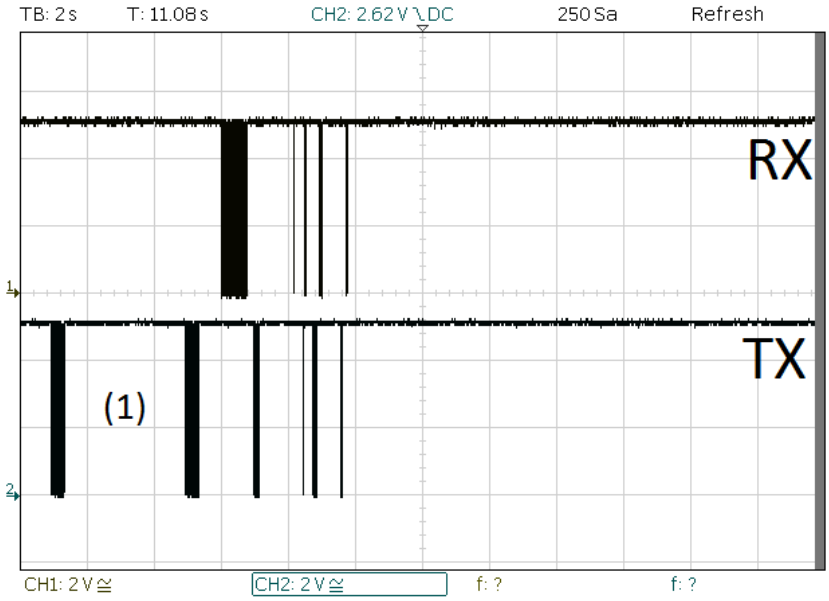

**Figuur 8: P0 Class Diagram**

### 5.3.1 OpenModeE

OpenModeE is de grootste functie binnen de P0 communicatie software. OpenModeE opent een nieuwe verbinding in mode E. Dit wordt gedaan door eerst een nieuwe P0 instantie aan te maken en daarna meerdere berichten naar de slimme energiemeter te sturen. Figuur 11 geeft een overzicht van de werking van OpenModeE.

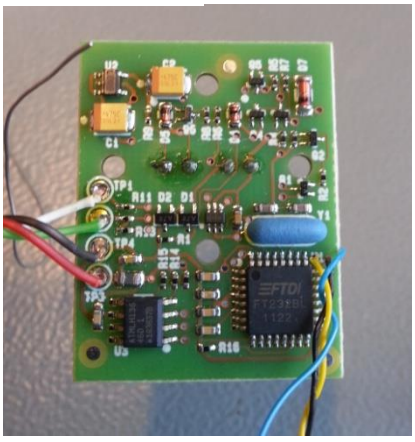
Via de software bus op de SMA komt er een OpenModeE aanvraag voor het P0 control blok. Het P0 control blok checkt de meegegeven argumenten en kijkt of de aangegeven COM poort niet al in gebruik is. Als de argumenten zijn gecontroleerd, maakt het P0 control blok een nieuw P0MessageBuilder object aan. Eenmaal aangemaakt roept het P0 control blok de OpenModeE functie van het P0MessageBuilder object aan. Deze begint op basis van de meegegeven argumenten meerdere berichten op te bouwen, te verzenden en de antwoorden te ontleden, zie 4.2 Opzetten connectie op blz. 5 voor meer over deze berichten. Na het verzenden van alle berichten laat het P0MessageBuilder object weten of het opzetten van de verbinding gelukt is of niet. Na het ontvangen van een antwoord van het P0MessageBuilder object slaat het P0Control Block de status van de verbinding op en stuurt het antwoord naar de software bus.

Bij een succesvolle verbinding start het P0MessageBuilder object ook een anti time-out loop. Dit is een loop die om de 4 seconde een bericht naar de slimme energiemeter verstuurt om de verbinding hoog te houden. Deze loop draait op een aparte thread zodat de andere functies van het P0MessageBuilder object door kunnen gaan. Via een mutex wordt er gezorgd dat de anti time-out loop niet tegelijkertijd met een van de andere functies data naar de slimme meter stuurt.

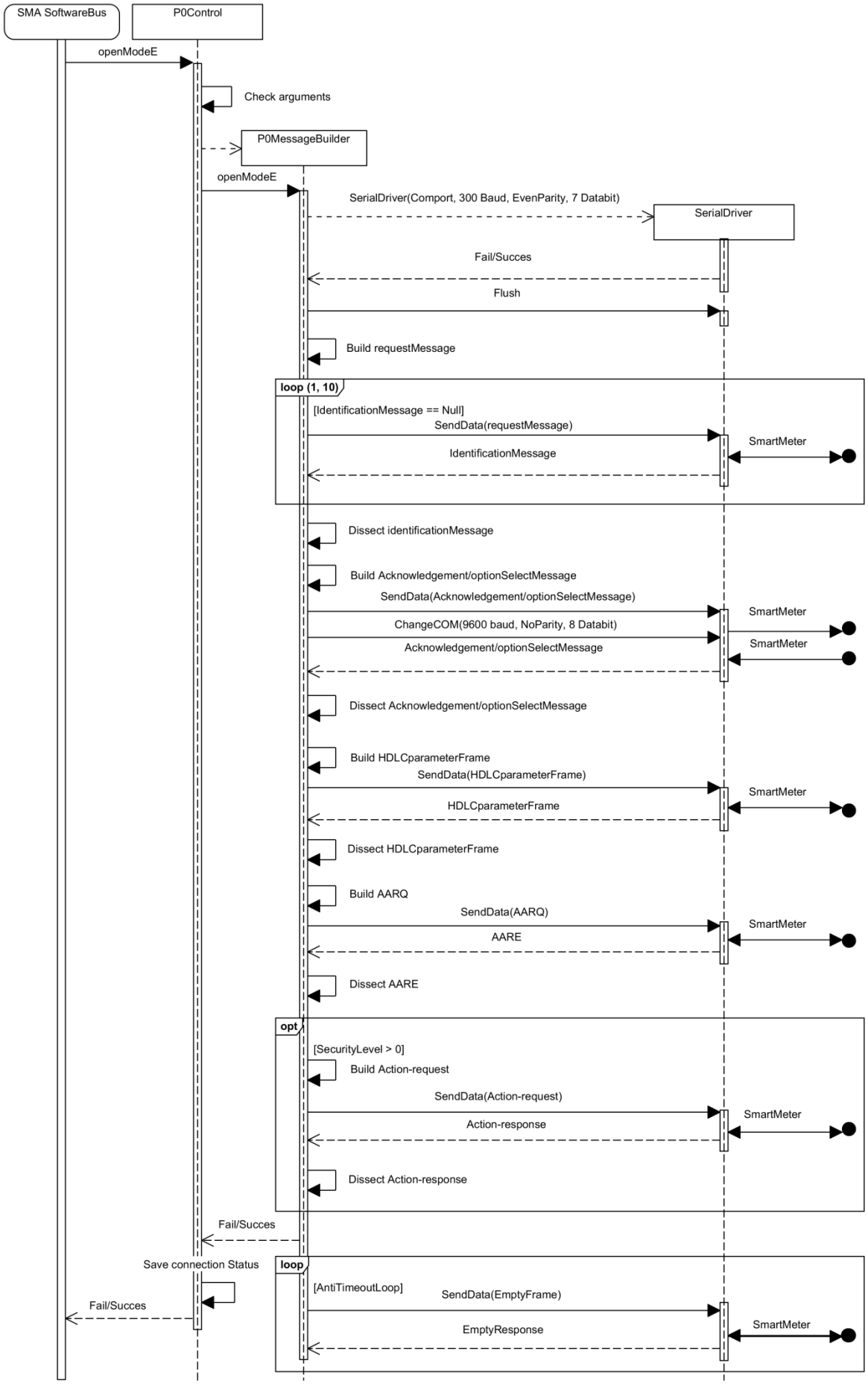



**Figuur 9: Open Mode E op de oscilloscoop**

Om beter te kunnen debuggen is een van optische probes opengemaakt en zijn er draden aan de tx, rx en gnd verbonden, zie Figuur 10. Hierdoor kan er via een oscilloscoop gezien worden of er data via de optische probe gaat. Dit kan goed gebruikt worden bij het debuggen van de Serial Driver. Figuur 9 laat het data verkeer bij de Open Mode E functie zien. De TX lijn is de data van SMA richting slimme energiemeter en de RX van de slimme energiemeter richting SMA. Wat opvalt, is dat er een groot gat valt waarin niks gebeurt, aangegeven met (1). Dit gat wordt veroorzaakt door het feit dat de slimme energiemeter niet altijd reageert op het de eerste requestMessage. Als de software voor 2 seconde niets ontvangt stuurt hij het bericht nog een keer.



**Figuur 10: binnenkant probe**



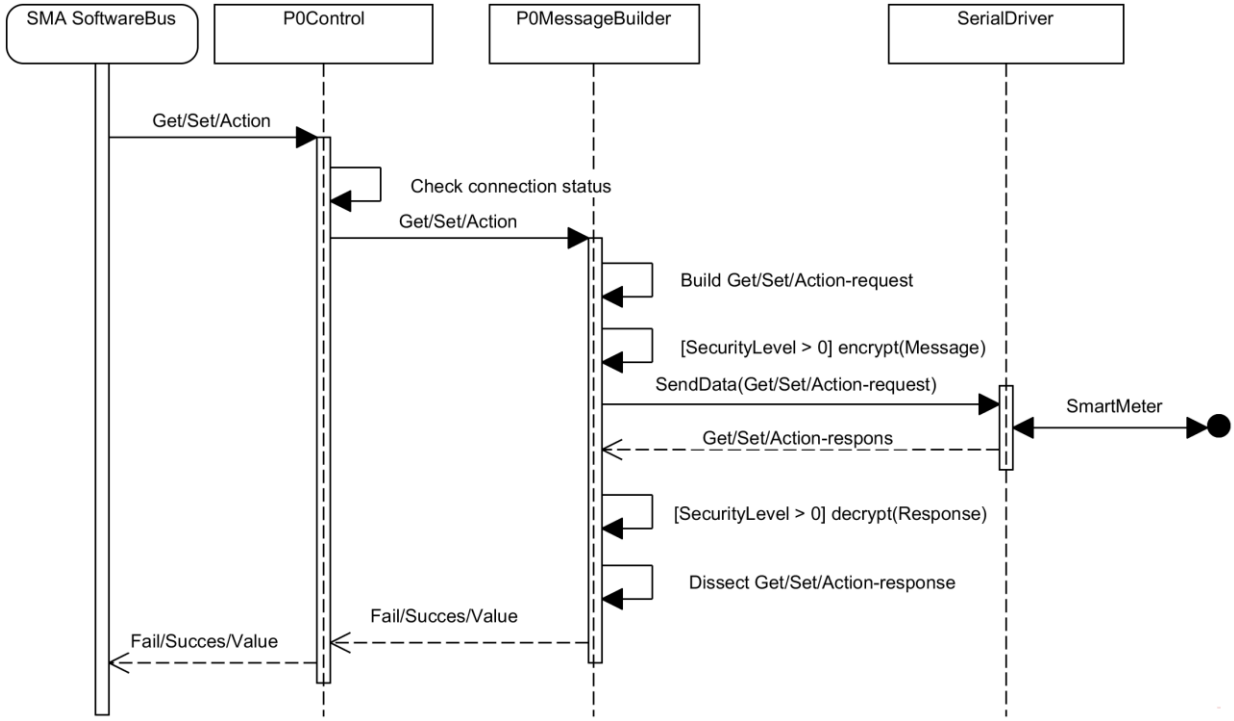

**Figuur 11: OpenModeE sequence diagram**

### 5.3.2 Get/Set/Action

Als de verbinding met de slimme energiemeter succesvol is opgezet kan er via de functies Get, Set en Action gecommuniceerd worden met de slimme energiemeter. De sequence diagrammen van deze drie functies zijn vrijwel gelijk en kunnen weergegeven worden in een diagram, zie Figuur 12.

Via de software bus komt er een Get, Set of Action aanroep met de benodigde argumenten. Het P0Control Blok kijkt of de bestemde connectie in een status is waarin de functies Get, Set of Action gebruikt kunnen worden. Als dit zo is wordt het commando doorgegeven aan het bij de connectie horende P0MessageBuilder object. Dit object bouwt het bericht met behulp van de bijhorende argumenten op, encrypt het indien nodig en verzendt het richting de slimme energiemeter. Het antwoord van de slimme energiemeter wordt gedecrypt, uitgepakt en terug gezonden naar het P0 control blok. Het P0 control blok stuurt vervolgens het antwoord terug de software bus op.

Het hardware platform waar de meet applicatie op gaat draaien heeft een crypto-core om snel data te kunnen encrypten/decrypten. Tijdens het schrijven van de P0 software bleek dat het encrypten/decrypten via software snel genoeg was. Samen met mede afstudeerder Marc Welleweerd die aan de hardware crypto-core werkt, is afgesproken om de hardware crypto-core dezelfde argumenten te geven als de software crypto-core. Hierdoor kan later de software crypto-core vervangen worden door de hardware crypto-core zonder dat de rest van de P0 code aangepast hoeft te worden. Dit beantwoordt de subvraag: “Hoe kunnen de berichten gedecrypt worden via de crypto-engine van de SMA?”. Voor de software crypto-core wordt gebruik gemaakt van de openSSL library [11].




**Figuur 12: Get/Set/Action sequence diagram**

De functies Get, Set en Action hebben invloed op een attribuut of methode uit een OBIS tabel, zie Tabel 2 blz. 7. De attribuut/methode wordt geselecteerd door een interface class, logical name en attribuut/methode nummer mee te geven aan de Get, Set en Action functies. Omdat dit veel data is om elke keer op te zoeken, is het ook mogelijk om een tag mee te geven in plaats van de drie argumenten. Via een XML bestand worden de benodigde argumenten dan automatisch in gevuld, zie Code voorbeeld 2. Deze functie is handig voor attributen/methodes die vaak gebruikt worden zoals: datum/tijd, water verbruik, energie verbruik en gas verbruik.

```
<?xml version="1.0"?>
<tags>
   <tag name="OBIS_CODE_CLOCK" type="get/set">
      <interface_class>0x08</interface_class>
      <logical_name>0x0000010000FF</logical_name>
      <attribute>2</attribute>
      <datatype>DATA_TYPE_OCTET_STRING</datatype>
      <description>tijd en datum</description>
   </tag>
   <tag name="OBIS_CODE_WATER" type="get/set/action">
      <interface_class>0x16</interface_class>
      <logical_name>0x0000010200FE</logical_name>
      <attribute>4</attribute>
      <datatype>DATA_TYPE_OCTET_STRING</datatype>
      <description>water verbruik</description>
</tags>
```

**Code voorbeeld 2: XML OBIS tags**

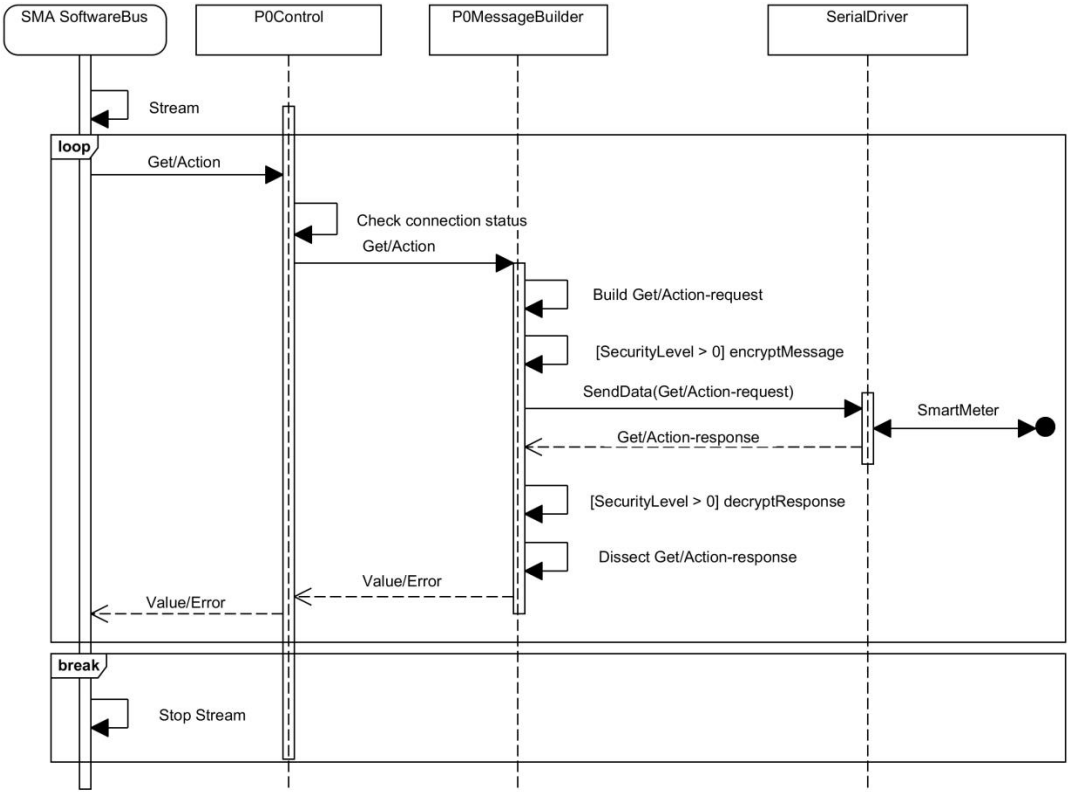
Het XML bestand bestaat uit een lijst van tags. Elke tag heeft als attribuut een naam en een type. De naam is de string die gebruikt kan worden als argument in de Get, Set en Action functies. Het type geeft aan in welke van Get, Set en Action functies de tag gebruikt kan worden. Naast de interface class, logical name en attribuut/methode nummer bevatten sommige tags ook een dataype veld. Dit veld wordt gebruikt bij Set en Action om aan te geven van welk type de mee gegeven data is. Als laatste is er een description veld, hier staat een korte beschrijving van de tag. In de P0 software wordt er gebruik gemaakt van de pugixml library [12] om de XML te parsen.

### 5.3.3 Stream/StopStream

De P0 communicatie software heeft ook een stream en stopstream functie. Via deze functies kan een Get of een Action in een loop uitgevoerd worden zonder dat de functies continue aan geroepen hoeven te worden via de software bus, zie Figuur 13. Deze functie kan gebruikt worden bij langdurende datalog tests waar meerdere keren de zelfde aanroep gedaan moet worden.

Via de software bus komt er een Stream aanroep met de benodigde argumenten. Omdat er bij elk bericht naar de slimme energiemeter een antwoord terug de bus op gaat, wordt de stream loop bij de software bus code gestart en niet in het P0Control blok. De stream loop draait op een aparte thread waardoor de andere functies van de software bus ongerstoord blijven. Via een mutex wordt er voor gezorgd dat er niet tegelijkertijd gebruikt wordt gemaakt van het P0Control blok.

Bij elke Get of Action request kijkt het P0Control Block of de bestemde connectie in een status is waarin de functies Get en Action gebruikt kunnen worden. Als dit zo is doet het P0Control blok een Get/Action aanroep op het bij de aangegeven connectie horende P0MessageBuilder object. De antwoorden van het P0MessageBuilder object worden vervolgens weer terug op de bus gezet. De loop met aanroepen blijft door gaan totdat er een stopstream commando komt die de stream thread afsluit.

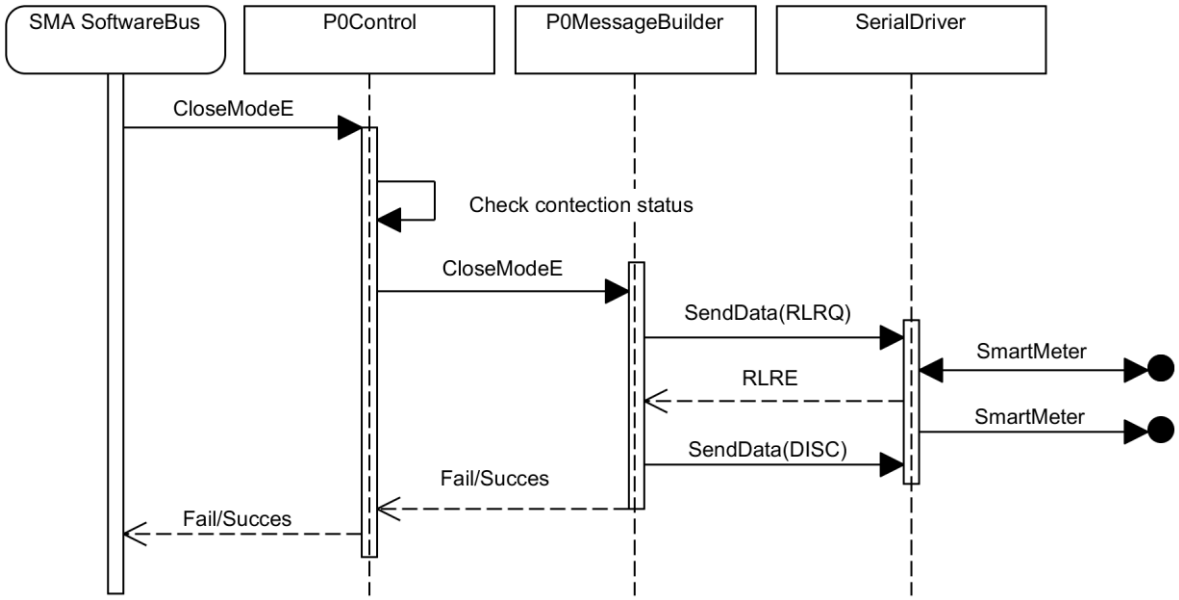



**Figuur 13: Stream & StopStream sequence diagram**

### 5.3.4 CloseModeE

CloseModeE is een functie die een protocol mode E verbinding afbreekt. Deze functie kan niet gebruikt worden als er via de OpenCOM functie een andere protocol mode geopend is. Zie Figuur 14 voor het sequentie diagram van deze functie.

Via de Software bus op de SMA komt er een CloseModeE aanroep. Het P0 control blok checkt of de bestemde connectie in een status is waarin de CloseModeE functie gebruikt kan worden. Als dit zo is wordt er CloseModeE aangeroepen op het bij de bestemde connectie horende P0MessageBuilder object. Dit object zendt vervolgens via de SerialDriver een RLRQ bericht naar de slimme energiemeter. De slimme energiemeter antwoordt hier op met een RLRE bericht. Als laatst stuurt het P0MessageBuilder Object een leeg HDLC frame van het type DISC. Een Fail of Succes wordt terug de bus opgestuurd om aan te geven of het sluiten van de verbinding gelukt is. Zie 4.4 Afsluiten connectie blz. 19 voor meer over de gebruikte berichten bij het afsluiten van een mode E connectie.

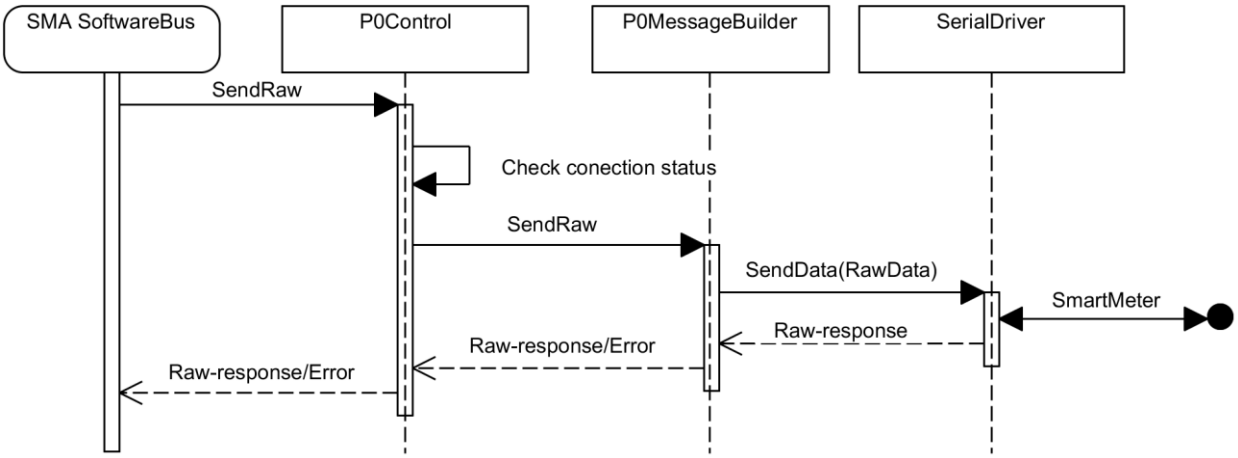



**Figuur 14: CloseModeE sequency diagram**

### 5.3.5 SendRaw

Naast mode E en Get/Set/Action zijn er nog andere protocolmodes en functies die niet in de P0 communicatie software geïmplementeerd zijn. Om deze protocolmodes en functies toch nog beschikbaar te stellen en de software open-ended te houden is er de SendRaw functie. Via deze functie kan er direct, byte voor byte, met de slimme energiemeter gecommuniceerd worden, zie Figuur 15.

Via de Software bus op de SMA komt er een SendRaw aanroep. Het P0 control blok kijkt of de bestemde connectie in een status is waarin de functie SendRaw gebruikt kan worden. Als dit zo is wordt er SendRaw aangeroepen op het P0MessageBuilder object van de bestemde connectie. Het P0MessageBuilder object zendt vervolgens de meegegeven bytes via de SerialDriver naar de slimme energiemeter. Het antwoord van de slimme energiemeter wordt zonder enige bewerkingen weer terug op de software bus gezet.

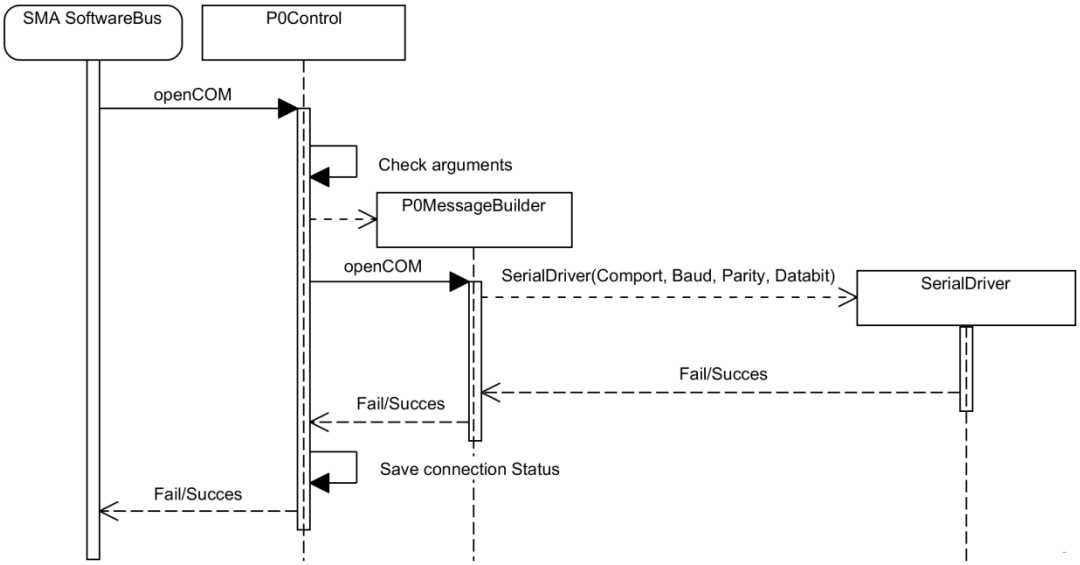



**Figuur 15: SendRaw sequence diagram**

### 5.3.6 OpenCOM

De OpenCOM functie kan gebruikt worden wanneer er een nieuwe verbinding moet worden opgezet die niet gebruik maakt van protocolmode E. De OpenCOM functie zet de COM poort naar een slimme energiemeter open maar verzendt verder niets, zie Figuur 16. Via SendRaw kan hierna een andere protocolmode worden geselecteerd, zie 5.3.5 SendRaw.

Vanaf de Software bus op de SMA komt er een OpenCOM aanroep. Het P0 control blok checkt de argumenten en kijkt of de COM poort niet al wordt gebruikt. Als dit niet zo is wordt er een nieuwe P0MessageBuilder object aangemaakt en wordt daar OpenCOM op aangeroepen. Het P0MessageBuilder object maakt vervolgens een nieuw SerialDriver object aan met de meegegeven COM poort settings. Als het aanmaken van deze twee objecten succesvol is slaat het P0 control blok de status van de connectie op. Het resultaat van het openen van de COM poort wordt terug op de software bus gezet.

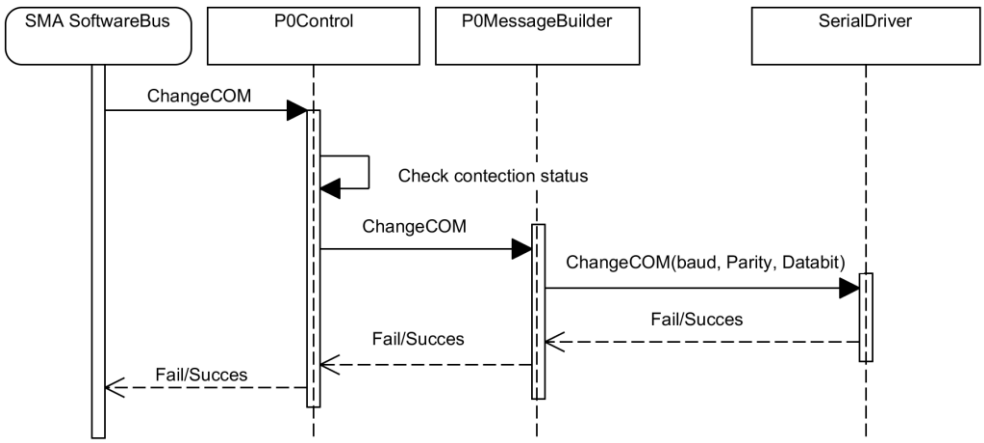



**Figuur 16: OpenCOM sequence diagram**

### 5.3.7 ChangeCOM

Sommige protocolmodes, waaronder mode E, hebben een COM poort settings verandering halverwege de communicatie. Om deze protocolmodes te ondersteunen is er de ChangeCOM functie, die halverwege de communicatie de COM poort settings kan aanpassen, zie Figuur 17.

Via de Softwarebus op de SMA komt er een ChangeCOM aanroep. Het P0 control blok checkt of de bestemde connectie in een status is waarin de COM poort settings veranderd kunnen worden. Als dit zo is wordt er op het bij de bestemde connectie horende P0MessageBuilder object de functie ChangeCOM aan geroepen. Het P0MessageBuilder object roept vervolgens ChangeCOM van zijn SerialDriver aan waardoor de COM settings veranderen. Het resultaat van het veranderen van de COM poort wordt terug op de software bus gezet.



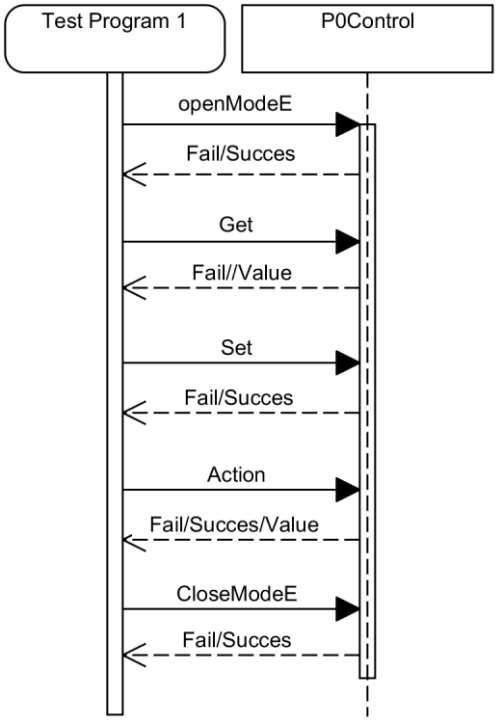

**Figuur 17: ChangeCOM sequence diagram**

# 6 Test en validatie

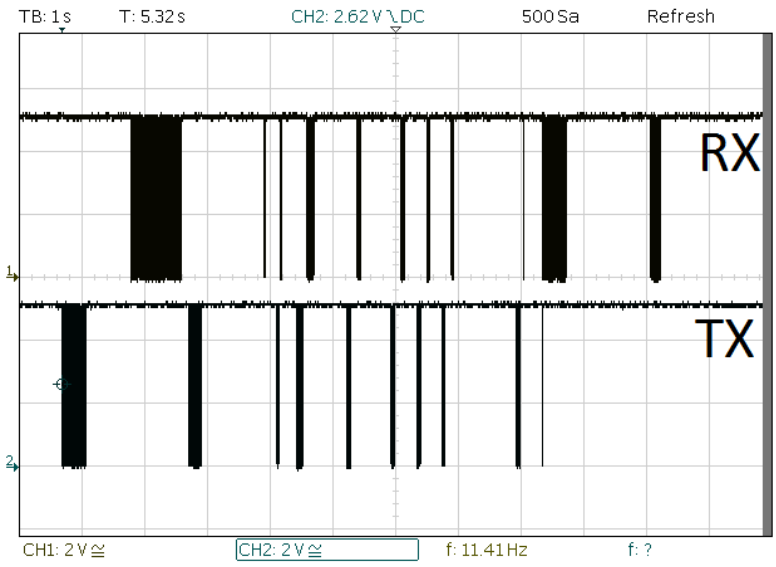
Dit hoofdstuk bestaat uit een aantal tests die laten zien dat de P0 communicatie software doet waar voor het ontworpen is. Daarnaast dienen de tests ook als voorbeeld voor hoe de software gebruikt kan worden.

## 6.1 Basis, test 1

In deze test worden de functies: OpenModeE, Get, Set, Action en CloseModeE getest. Figuur 19 geeft een overzicht van het basis test 1 programma. Figuur 18 laat het data verkeer over de optische probe tijdens de test zien. Hierin is de TX lijn de data richting de slimme energiemeter is en RX de data vanaf de slimme energiemeter. Code voorbeeld 3 geeft het resultaat van de test. Hier is aan de terug gekomen antwoorden te zien dat de test succesvol is uitgevoerd en alle functies die hier getest worden werken.




**Figuur 19: Test 1 sequense diagram**



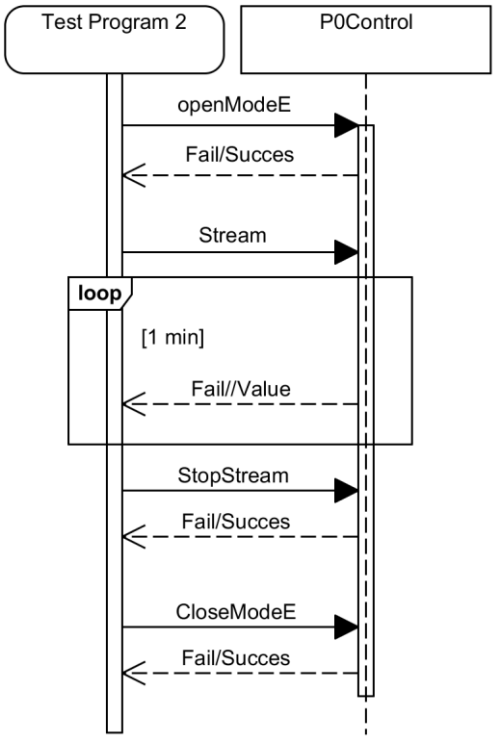

**Figuur 18: Test 1 oscilloscoop**

```
starting program

opening meter 1 on /dev/ttyUSB0
Succes opening meter 1 on /dev/ttyUSB0

Getting date Time from meter 1
Succes getting from meter 1, answer:
00000000  c4 01 c1 00 09 0c 07 e0 01 19 01 0b 32 13 00 ff
00000010  c4 00

Setting  meter 1
Succes Setting for meter 1, answer:
00000000  c5 01 c1 00

Action for meter 1
Succes Action for meter 1, answer:
00000000  c7 01 c1 00 00

Closing Meter1
Succes closing for meter 1

end program
```

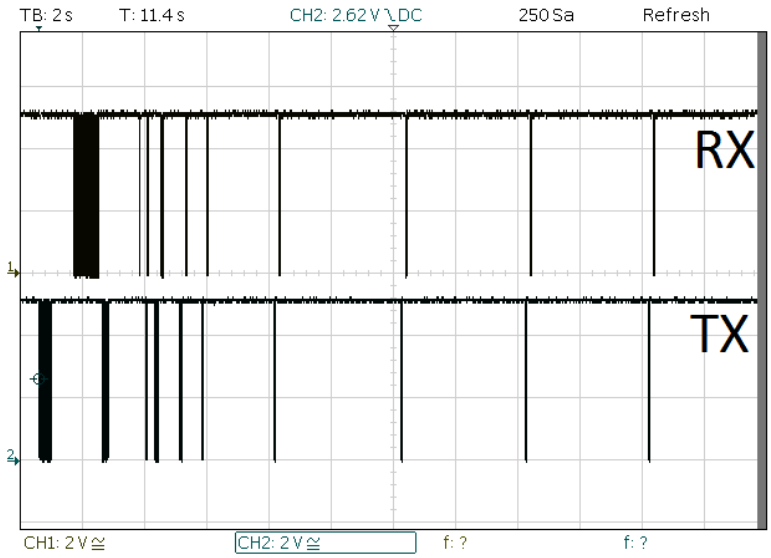
**Code voorbeeld 3: Test 1 output**

## 6.2 Stream, test 2

In deze test wordt het streamen getest. In de test wordt een Get stream gestart. Na 1 minuut wordt de stream gestopt via de StopStream functie, zie Figuur 20. Figuur 21 laat het begin van de test op de oscilloscoop zien. Hierin zijn de openModeE berichten te zien met daarna het om de 4 seconde herhalende Get request. Code voorbeeld 4 geeft de output van de test weer. Na het openen van mode E en starten van de Get stream wordt 60 keer het Get resultaat geprint. Dit laat zien dat de stream functie werkt.




**Figuur 20: Test 2 sequense diagram**



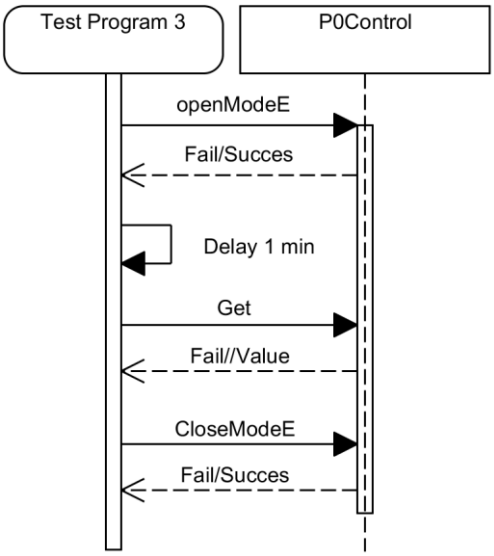

**Figuur 21: Test 2 oscilloscoop**

```
starting program
opening  meter 1 on /dev/ttyUSB0
Succes opening  meter 1 on /dev/ttyUSB0

starting get stream on meter 1
start sleeping 1 min

Stream get result:
00000000  c4 01 c1 00 09 0c 07 e0 01 19 01 0c 30 31 00 ff
00000010  c4 00

(59 times)
Stream get result:
00000000  c4 01 c1 00 09 0c 07 e0 01 19 01 0c 30 31 00 ff
00000010  c4 00

end sleeping 1 min
Stopping stream
Stream stopped

Closing Meter1
end program
```

**Code voorbeeld 4: Test 2 output**

## 6.3 Anti timout, test 3

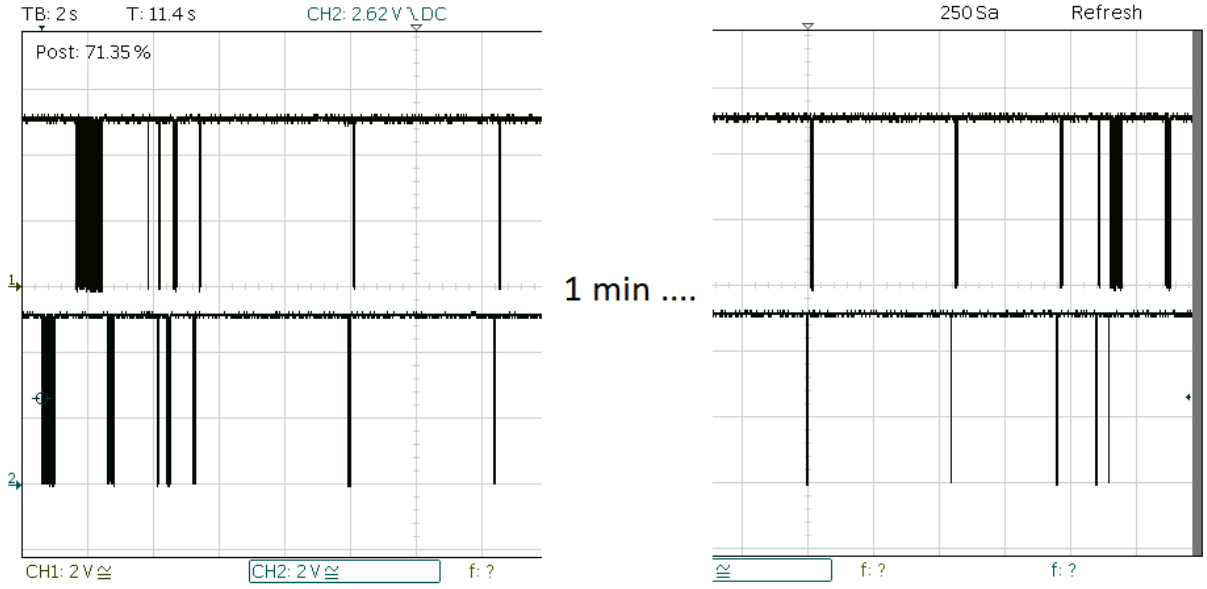
In deze test wordt de anti-timeout loop getest. De anti-timeout loop wordt getest door de verbinding te openen, 1 minuut te wachten en een get-request te sturen, zie Figuur 22. Alleen als de verbinding open is gehouden door de anti-timeout loop geeft het get-request de opgevraagde data terug. Figuur 23 laat het begin en einde van de test zien. Ook zijn de anti timeout berichten te zien die om de 4 seconde de verbinding open houden. Code voorbeeld 5 geeft de output van de test en laat zien dat de verbinding open wordt gehouden en dat de software in staat is na 1 minuut een get-request te doen.




**Figuur 22: Test 3 sequense diagram**

```
starting program

opening  meter 1 on /dev/ttyUSB0
Succes openening  meter 1 on /dev/ttyUSB0

start sleeping 1 min
end sleeping 1 min

Getting date Time from meter 1
Succes getting from meter 1, answer:
00000000  c4 01 c1 00 09 0c 07 e0 01 19 01 0d 3a 28 00 ff
00000010  c4 01

Closing Meter1

end program
```

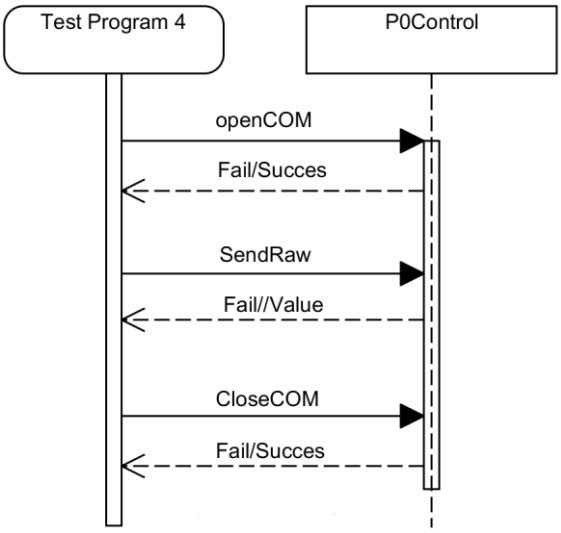
**Code voorbeeld 5: Test 3 output**



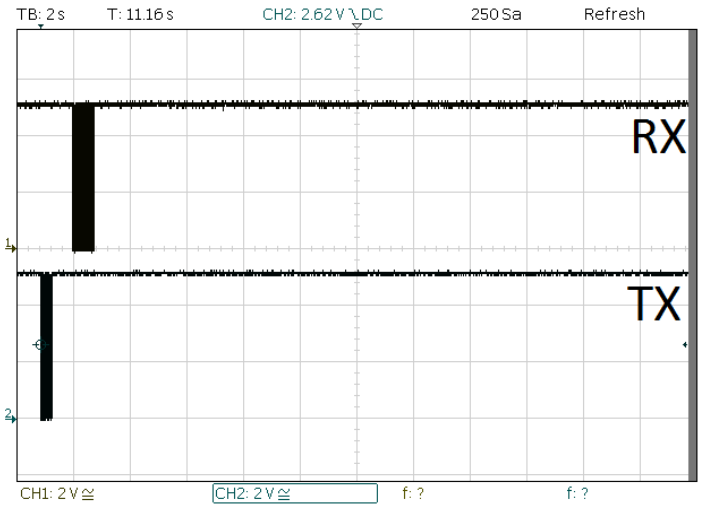

**Figuur 23: Test 3 oscilloscoop**

## 6.4 Send Raw, test 4

Deze simpele test, test de functies die gebruikt worden bij de Low Level manier, zie 5.2 Gebruik blz. 26. Eerst wordt de COM poort geopend via openCOM daarna worden er via sendRaw bytes over de COM poort verstuurd het antwoord wordt ontvangen en de COM poort wordt gesloten, zie Figuur 25. Figuur 24 laat het data verkeer over de optische probe zien. In Code voorbeeld 6 is het resultaat van de test te zien. De data wordt succesvol naar de slimme energiemeter verzonden en de het antwoord wordt succesvol ingelezen.




**Figuur 25: Test 4 sequense diagram**




**Figuur 24: Test 4 oscilloscoop**

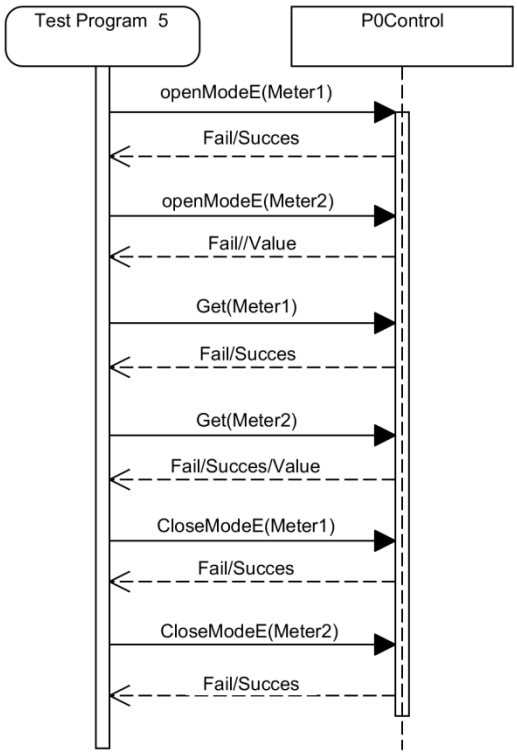
```
starting program

opening  meter 1 on /dev/ttyUSB0
Succes opening  meter 1 on /dev/ttyUSB0

Sending raw for Meter1
Succes Sending raw for meter 1,  raw answer:
00000000  2f 58 4d 58 35 5c 32 58 4d 58 5f 4e 34 30 5f 47  /XMX5\2XMX_N40_G
00000010  70 72 73 56 33 30 0d 0a                          prsV30..

Closing meter1
Succes Closing meter1

end program
```

**Code voorbeeld 6: Test 4 output**

## 6.5 Meerdere meters, test 5

In deze test wordt getest of er met meerdere meters tegelijkertijd gecommuniceerd kan worden. Dit wordt gedaan door direct na het openen van meter 1, meter 2 te openen. Daarna wordt op beide meters een get-request gedaan. Na de get-requests worden beide meters gesloten. Zie Figuur 26 voor een overzicht van de test.

Zoals de output in Code voorbeeld 7 laat zien, is deze test nog niet succesvol afgerond. Bij het openen van een tweede meter, (terwijl er al een andere meter geopend is) gaat het fout. Het praten met meter 2 na het sluiten van meter 1 werkt wel. Samen met de errors in het main console van de SMA (niet die output van de test), zie Code voorbeeld 8, lijkt het er op dat er een fout in de ftdi_sio driver zit. Na het inleveren van de scriptie zal er naar gekeken worden of deze geïnstalleerde driver geüpdate kan worden. Er kan wel omstebeurt met meerdere meters gepraat worden door eerte meter 1 te sluiten voor meter 2 te openen.




**Figuur 26: Test 5 sequense diagram**

```
starting program

openening  meter 1 on /dev/ttyUSB0
Succes openening  meter 1 on /dev/ttyUSB0

openening  meter 2 on /dev/ttyUSB1
ERROR: Incorrect response for Meter2

Geting date Time from meter 1
Succes geting  meter 1, awnser:
00000000  c4 01 c1 00 09 0c 07 e0 01 1a 02 0c 04 06 00 ff
00000010  c4 01

Geting date Time from meter 2
Error: Name Meter2 does not exist

Closing Meter1
Succes Closing meter1

Closing Meter2
Error: Name Meter2 does not exist

end program
```

**Code voorbeeld 7: Test 5 output**

```
ftdi_sio ttyUSB1: failed to get modem status: -110
ftdi_sio ttyUSB1: ftdi_set_termios urb failed to set baudrate
```

**Code voorbeeld 8: Test 5 errors**

# 7 Conclusie en aanbevelingen

Na veel onderzoek is duidelijk geworden hoe er over P0 gecommuniceerd kan worden. De gebruikte communicatie protocollen bleken complexer dan oorspronkelijk verwacht, waardoor niet alle mogelijkheden van P0 geïmplementeerd zijn in de software. Omdat de software open-ended geschreven is kunnen de achtergelaten functies van P0 toch nog gebruikt worden door een hoger liggend stuk software. Door de software op te delen in een P0 Control object, meerdere P0 MessageBuilder objecten en meerdere Serial Driver objecten kan er goed met meerdere P0 poorten gecommuniceerd worden.

Omdat alleen de slimme energiemeter een aangesloten P0 poort heeft hoeft er geen rekening gehouden te worden met de andere type slimme meters. Tijdens het schrijven van de P0 software bleek dat de software crypto-core meer dan snel genoeg was voor het encrypten en decrypten van de P0 berichten. Om de mogelijkheid van een hardware crypto-core open te houden is de software crypto-core met dezelfde argumenten geschreven als de hardware crypto-core. Hierdoor kan de hardware crypto-core indien later nodig alsnog worden gebruikt zonder dat er P0 software hoeft te worden aangepast.

De P0 communicatie software kan nog verbeterd worden. Het is nog niet mogelijk om tegelijkertijd met meerdere meters te praten. Hier zal na het inleveren van de scriptie nog aandacht aan worden besteed.

Het doorlopende slimme meter project van Sogeti is een behoorlijk stuk verder gekomen nu er gecommuniceerd kan worden via de P0 poort. De volgende stap in het doorlopende project zou het realiseren van de software bus samen met een SMA manager kunnen zijn. Zodat de lossen delen software op de SMA aan elkaar kunnen worden verbonden.

# Bibliografie


[1] J. R. Centre, „Smart Meter Deployment in the European Union," Joint Research Centre, 11 02 2016. [Online]. Available: http://ses.jrc.ec.europa.eu/smart-metering-deployment-european-union. [Geopend 16 02 2016].

[2] R. Keijsers en M. Rombouts, „Project Plan," Sogeti, Amersfoort, 2015.

[3] R. Keijsers en M. Rombouts, Functional Requirements Document, Amersfoort: Sogeti, 2015.

[4] *Dutch Smart Meter Requirements, V4.2.2, Final Main,* 2014.

[5] *Electricity metering - Direct local data exchange ,IEC 62056-21,* 2002.

[6] DLMS User Association, Green Book - 7th edition, DLMS/COSEM Architecture and Protocols, 2009.

[7] DLMS User Association, Blue Book - 11th Edition, COSEM interface classes and OBIS identification system, 2013.

[8] „FLAG MANUFACTURERS ID," [Online]. Available: http://dlms.com/organization/flagmanufacturesids.

[9] M. Welleweerd, Hardware onderzoeks document, Amersfoort: Sogeti, 2016.

[10] „Netbeans," [Online]. Available: https://netbeans.org.

[11] „openSSL," [Online]. Available: https://www.openssl.org/.

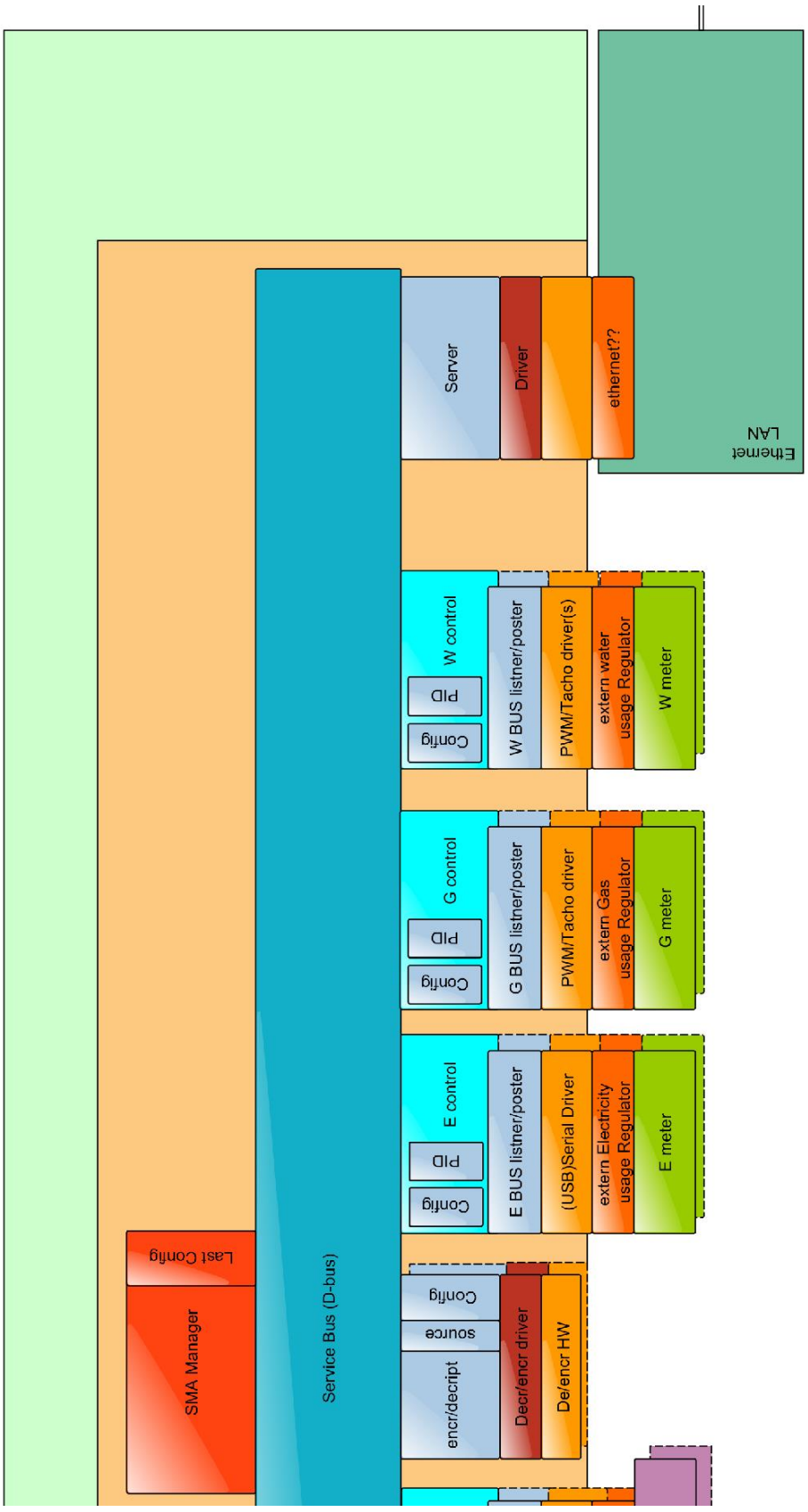
[12] „pugixml," [Online]. Available: http://pugixml.org/.

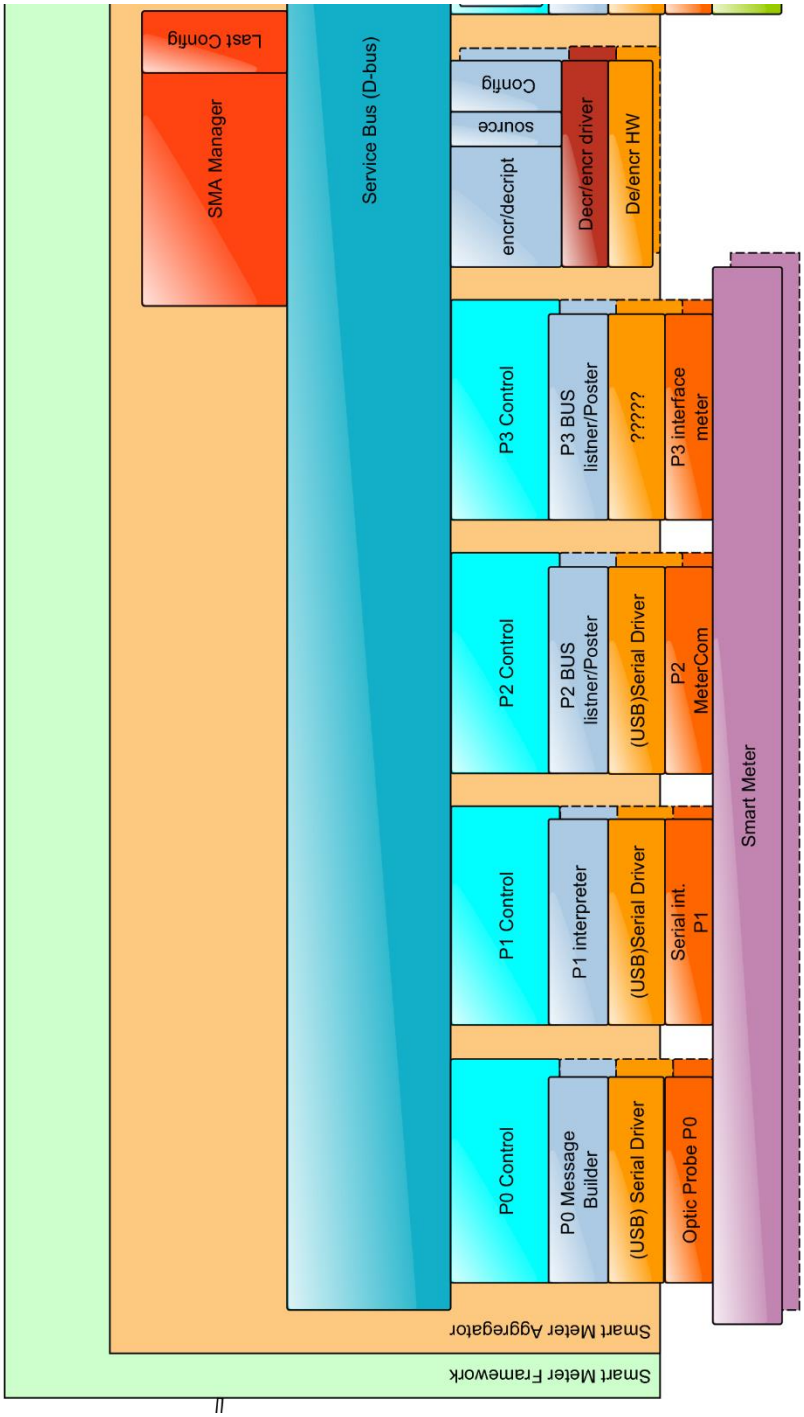
[13] R. Keijsers en M. Rombouts, Research Document, Test set-up Smart Energy Meter, Amersfoort: Sogeti, 2015.

[14] R. Keijsers en M. Rombouts, Project Plan, Test set-up Smart Energy Meter, Amersfoort: Sogeti, 2015.

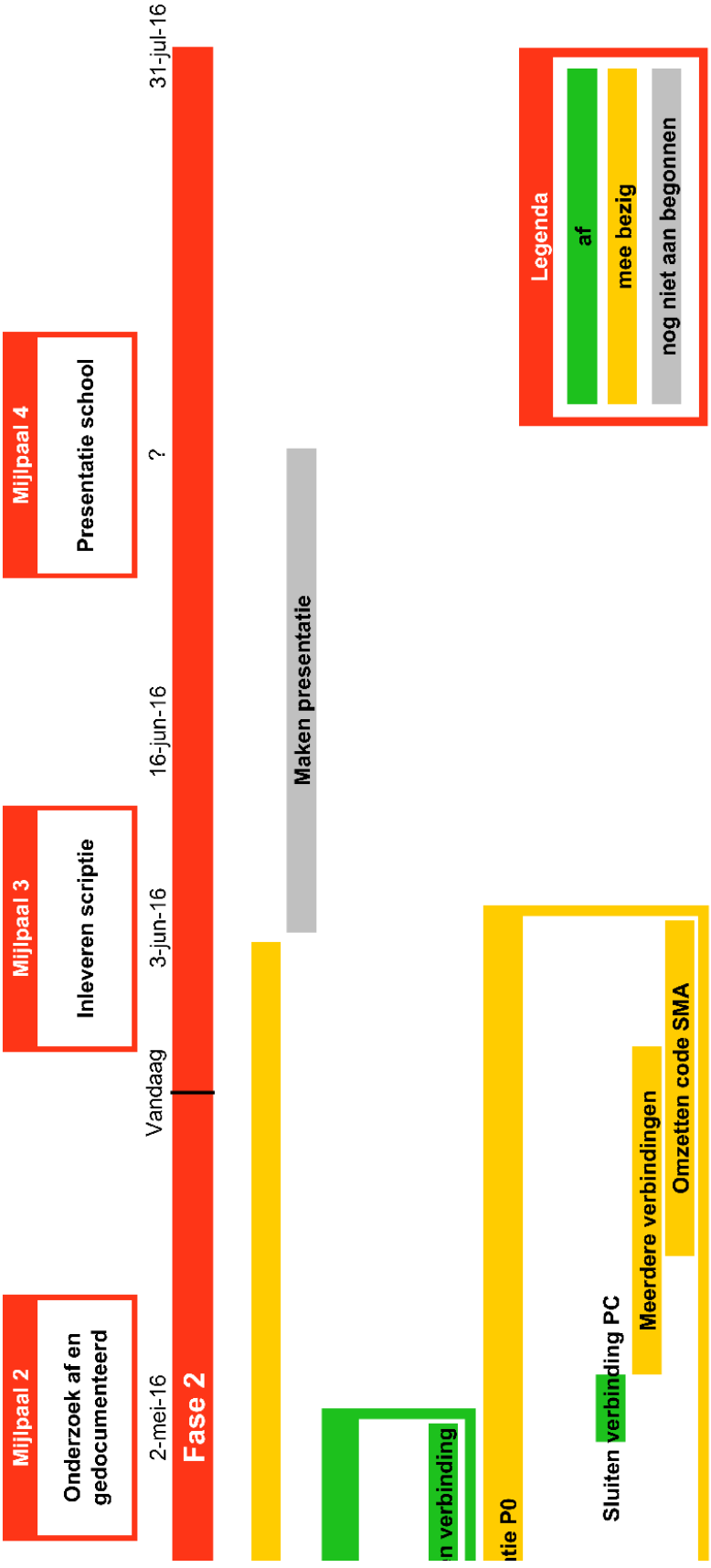
[15] M. Rombouts, Afstudeerverslag, Amersfoort: Sogeti, 2016.

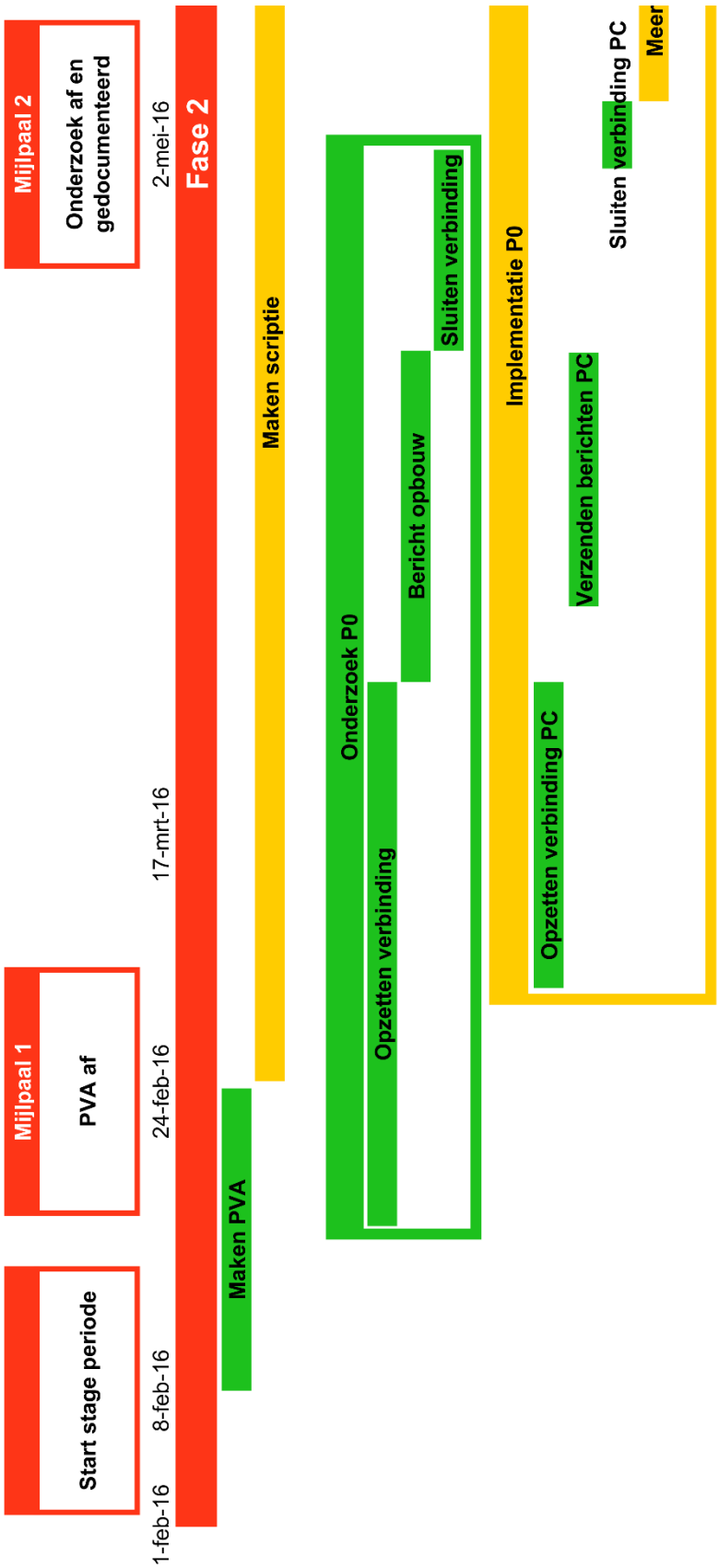
# Bijlage A: SMA software architectuur




Smart Meter Framework
Smart Meter Aggregator
Last Config
SMA Manager
Service Bus (D-bus)
Config
source
encr/decript
Decr/encr driver
De/encr HW
P3 Control
P3 BUS listner/Poster
????
P3 interface meter
P2 Control
P2 BUS listner/Poster
(USB)Serial Driver
P2 MeterCom
P1 Control
P1 interpreter
(USB)Serial Driver
Serial int. P1
P0 Control
P0 Message Builder
(USB) Serial Driver
Optic Probe P0
Smart Meter

# Bijlage B: Stroken planning




Start stage periode
Mijlpaal 1
PVA af
Mijlpaal 2
Onderzoek af en gedocumenteerd
1-feb-16
8-feb-16
24-feb-16
17-mrt-16
2-mei-16
Fase 2
Maken PVA
Maken scriptie
Onderzoek P0
Opzetten verbinding
Bericht opbouw
Sluiten verbinding
Implementatie P0
Opzetten verbinding PC
Verzenden berichten PC
Sluiten verbinding PC
Meer

# Bijlage C: P0ControlExample.cpp

```
//============================================================================================================
// Name        : P0ControlExample.cpp
// Author      : Niels Hokke
// Company     : Sogeti
// Description : test for openmodeE, get, set, action, closemodeE
//============================================================================================================

#include <iostream>
#include <stdio.h>
#include <thread>
#include <mutex>
#include "P0Control.h"

using namespace std;

#define CR 13
#define LF 10

u8 K[] = {
    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0E,
    0x0f,
};

u8 AAD[] = //AK
{
    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0E,
    0x0f,
};

P0Control *p0control;

/* displays byte arrays */
static char ascii1(char s) {
    if (s < 0x20) return '.';
    if (s > 0x7E) return '.';
    return s;
}

/* displays byte arrays */
void hexdump1(void *d, int len) {
    if (len < 1) return;
    u8 *data;
    int i, off;
    data = (u8*) d;
    for (off = 0; off < len; off += 16) {
        printf("%08x  ", off);
        for (i = 0; i < 16; i++)
            if ((i + off) >= len) printf("   ");
            else printf("%02x ", data[off + i]);

        printf(" ");
        for (i = 0; i < 16; i++)
            if ((i + off) >= len) printf(" ");
            else printf("%c", ascii1(data[off + i]));
        printf("\n");
    }
    cout << endl;
}

int main() {
    cout << "starting program 1" << endl << endl;

    u8 response[255];

    //P0Control object
    p0control = new P0Control();

    //Opening mode E on /dev/ttyUSB0 for Meter1
    try {
        cout << "openening  meter 1 on /dev/ttyUSB0" << endl;
        p0control->OpenModeE("/dev/ttyUSB0", "P07210", K, AAD, "Meter1");
        cout << "Succes openening  meter 1 on /dev/ttyUSB0" << endl << endl;
    } catch (std::string &msg) {
```

```
        cout << msg << endl << endl;
    }

    //Getting data from Meter1
    try {
        cout << "Geting date Time from meter 1" << endl;
        int l = p0control->Get(0x08, 0x0000010000FF, 2, response, "Meter1");
        cout << "Succes geting  meter 1, awnser:" << endl;
        hexdump1(response, l);
    } catch (std::string &msg) {
        cout << msg << endl << endl;
    }

    //Setting data for Meter1
    try {
        cout << "Seting  meter 1" << endl;
        //2016-01-25 11:46:57 (Data_OctetString)
        u8 dataIn[] = {0x07, 0xE0, 0x01, 0x19, 0x01, 0x0B, 0x2E, 0x39, 0x00, 0xFF, 0xC4, 0x00};
        int r = p0control->Set(0x08, 0x0000010000FF, 2, DATA_TYPE_OCTET_STRING, dataIn,
        sizeof(dataIn), response, "Meter1");
        cout << "Succes Seting for meter 1, awnser:" << endl;
        hexdump1(response, r);
    } catch (std::string &msg) {
        cout << msg << endl << endl;
    }

   //Action for Meter1
   try {
        cout << "Action for  meter 1" << endl;
        u8 dataIn[] = {0x00};
        int l = p0control->Action(0x08, 0x0000010000FF, 3, DATA_TYPE_OCTET_STRING, dataIn,
        sizeof(dataIn), response, "Meter1");
        cout << "Succes Action for meter 1, awnser:" << endl << endl;
        hexdump1(response, l);
    } catch (std::string &msg) {
        cout << msg << endl << endl;
    }

    //Closing Meter1
    try {
        cout << "Closing Meter1" << endl;
        p0control->CloseModeE("Meter1");
        cout << endl;
    } catch (std::string &msg) {
        cout << msg << endl << endl;
    }

    cout << "end program" << endl;
    return 0 ;
```

# Bijlage D: P0Control.h

```
//============================================================================================================
// Name        : P0Contol.h
// Author      : Niels Hokke
// Company     : Sogeti
// Description : Header File of P0Control.cpp
//============================================================================================================

#ifndef P0CONTROL_H
#define P0CONTROL_H_

#include <string>
#include <unistd.h>
#include <iostream>
#include <vector>
#include "P0MessageBuilder.h"

typedef unsigned long long int u64;
typedef unsigned int u32;
typedef unsigned short u16;
typedef unsigned char u8;

class P0Control {
private:

    /* List of P0 MessageBuilder object pointers */
    std::vector<P0MessageBuilder*> p0messagebuilderList;

    /* Finds meter from p0messagebuilderList basted on name */
    int findMeter(std::string meterName);

    /* Finds meter from p0messagebuilderList basted on COM port */
    int findCOM(std::string commPortName);

    /* Destroys meter based on name */
    void destroyMeter(std::string meterName);

public:

    /* makes a new P0 instance, opens COM port and start Mode E */
    void OpenModeE(std::string commPortName, std::string DeviceAddress, u8 K[], u8 AAD[],
                   std::string meterName);

    /* Gets data in mode E of a specific meter */
    int Get(u16 interface_class, u64 logical_name, u8 attribute, u8 dataOut[],
            std::string meterName);

    /* Gets data in mode E of a specific meter */
    int Get(std::string tag, u8 dataOut[], std::string meterName);

    /* Sets data in mode E of a specific meter */
    int Set(u16 interface_class, u64 logical_name, u8 attribute, u8 datatype, u8 dataIn[],
            int dataInSize, u8 dataOut[], std::string meterName);

    /* Sets data in mode E of a specific meter */
    int Set(std::string tag, u8 dataIn[], int dataInSize, u8 dataOut[], std::string meterName);

    /* Does an Action on data in mode E for a specific meter */
    int Action(u16 interface_class, u64 logical_name, u8 methode, u8 datatype, u8 dataIn[],
               int dataInSize, u8 dataOut[], std::string meterName);

    /* Does an Action on data in mode E for a specific meter */
    int Action(std::string tag, u8 dataIn[], int dataInSize, u8 dataOut[],
               std::string meterName);

    /* Closes the COM port for a specific meter and deletes P0 instance */
    void CloseModeE(std::string meterName);

    /* Opens the COM port for a specific meter */
    void OpenCOM(std::string commPortName, int bitRate = 300, int prarity = 1, int byte = 7,
                 std::string meterName = "");
```

```
    /* Sends raw byte's and returns raw answer for a specific meter */
    int SendRaw(u8 *bufferIn, int bufferInLenght, u8 *bufferOut, int readDelay, int timeout,
                std::string find, std::string meterName);

    /* Closes COM port of a specific meter and deletes P0 instance */
    void CloseCOM(std::string meterName);

};

#endif /* P0CONTROL_H_ */
```

# Bijlage E: P0Control.cpp

```
//============================================================================================================
// Name        : P0Contol.cpp
// Author      : Niels Hokke
// Company     : Sogeti
// Description : Controls and manages different P0 instances
//============================================================================================================

#include <unistd.h>
#include <stdio.h>
#include "P0MessageBuilder.h"
#include "P0Control.h"
#include "pugixml.hpp"
#include <stdlib.h>


using namespace std;

/*****************************************************************
 * Private Functions
 *****************************************************************/

void findTAG(string TAG, string type, u16 &interface_class, u64 &logical_name, u8 &attribute,
             u8 &datatype) {
    pugi::xml_document doc;

    pugi::xml_parse_result result = doc.load_file("/home/sogeti/Niels/OBIS_TAGS.xml");//OBIS_TAGS

    //cout << "Load result: " << result.description() << endl;
    pugi::xml_node tags = doc.child("tags");

    for (pugi::xml_node tag = tags.child("tag"); tag; tag = tag.next_sibling("tag")) {
        string tagName = tag.attribute("name").as_string();
        string tagType = tag.attribute("type").as_string();

        if (!tagName.compare(TAG)) {
            if (tagType.find(type) != string::npos) {
                string input = tag.child_value("logical_name");

                char input1[7] = {0, 0, 0, 0, 0, 0, 0};
                char input2[7] = {0, 0, 0, 0, 0, 0, 0};

                for (int i = 0; i < 6; i++) {
                    input1[i] = input[2 + i];
                    input2[i] = input[8 + i];
                }

                u64 output1 = strtoul(input1, NULL, 16);
                u64 output2 = strtoul(input2, NULL, 16);

                logical_name = output2 + (output1 << 24);
                interface_class = strtoul(tag.child_value("interface_class"), NULL, 16);
                attribute = strtoul(tag.child_value("attribute"), NULL, 10);

                string datatypeS = tag.child_value("datatype");
                if (datatypeS.compare("DATA_TYPE_NULL_DATA") == 0) {
                    datatype = DATA_TYPE_NULL_DATA;
                } else if (datatypeS.compare("DATA_TYPE_ARRAY") == 0) {
                    datatype = DATA_TYPE_ARRAY;
                } else if (datatypeS.compare("DATA_TYPE_STRUCTURE") == 0) {
                    datatype = DATA_TYPE_STRUCTURE;
                } else if (datatypeS.compare("DATA_TYPE_BOOLEAN") == 0) {
                    datatype = DATA_TYPE_BOOLEAN;
                } else if (datatypeS.compare("DATA_TYPE_BIT_STRING") == 0) {
                    datatype = DATA_TYPE_BIT_STRING;
                } else if (datatypeS.compare("DATA_TYPE_DOUBLE_LONG") == 0) {
                    datatype = DATA_TYPE_DOUBLE_LONG;
                } else if (datatypeS.compare("DATA_TYPE_DOUBLE_LONG_UNSIGNED") == 0) {
                    datatype = DATA_TYPE_DOUBLE_LONG_UNSIGNED;
                } else if (datatypeS.compare("DATA_TYPE_FLOATING_POINT") == 0) {
                    datatype = DATA_TYPE_FLOATING_POINT;
                } else if (datatypeS.compare("DATA_TYPE_OCTET_STRING") == 0) {
                    datatype = DATA_TYPE_OCTET_STRING;
                } else if (datatypeS.compare("DATA_TYPE_VISIBLE_STRING") == 0) {
```

```
                    datatype = DATA_TYPE_VISIBLE_STRING;
                } else if (datatypeS.compare("DATA_TYPE_BCD") == 0) {
                    datatype = DATA_TYPE_BCD;
                } else if (datatypeS.compare("DATA_TYPE_INTEGER") == 0) {
                    datatype = DATA_TYPE_INTEGER;
                } else if (datatypeS.compare("DATA_TYPE_LONG") == 0) {
                    datatype = DATA_TYPE_LONG;
                } else if (datatypeS.compare("DATA_TYPE_UNSIGNED") == 0) {
                    datatype = DATA_TYPE_UNSIGNED;
                } else if (datatypeS.compare("DATA_TYPE_LONG_UNSIGNED") == 0) {
                    datatype = DATA_TYPE_LONG_UNSIGNED;
                } else if (datatypeS.compare("DATA_TYPE_COMPACT_ARRAY") == 0) {
                    datatype = DATA_TYPE_COMPACT_ARRAY;
                } else if (datatypeS.compare("DATA_TYPE_LONG_64") == 0) {
                    datatype = DATA_TYPE_LONG_64;
                } else if (datatypeS.compare("DATA_TYPE_LONG_64_UNSIGNED") == 0) {
                    datatype = DATA_TYPE_LONG_64_UNSIGNED;
                } else if (datatypeS.compare("DATA_TYPE_ENUM") == 0) {
                    datatype = DATA_TYPE_ENUM;
                } else if (datatypeS.compare("DATA_TYPE_FLOAT_32") == 0) {
                    datatype = DATA_TYPE_FLOAT_32;
                } else if (datatypeS.compare("DATA_TYPE_FLOAT_64") == 0) {
                    datatype = DATA_TYPE_FLOAT_64;
                } else if (datatypeS.compare("DATA_TYPE_DATE_TIME") == 0) {
                    datatype = DATA_TYPE_DATE_TIME;
                } else if (datatypeS.compare("DATA_TYPE_DATE") == 0) {
                    datatype = DATA_TYPE_DATE;
                } else if (datatypeS.compare("DATA_TYPE_TIME") == 0) {
                    datatype = DATA_TYPE_TIME;
                } else if (datatypeS.compare("DATA_TYPE_DONT_CARE") == 0) {
                    datatype = DATA_TYPE_DONT_CARE;
                } else {
                    throw ("ERRPR: Error in OBIS_TAGS.xml datatype: " + datatypeS +
                           " not recognized");
                }
                return;
            } else {
                throw ("ERROR: OBIS_TAG: " + TAG + " not found for " + type);
            }
        }
    }
    throw ("ERROR: OBIS_TAG: " + TAG + " not found");
}

/*****************************************************************
 * Private Member Functions
 *****************************************************************/

/* Finds meter from p0messagebuilderList basted on name */
int P0Control::findMeter(string meterName) {
    if (p0messagebuilderList.size() > 0) {
        vector<P0MessageBuilder*>::iterator it;
        int index = 0;
        for (it = p0messagebuilderList.begin(); it != p0messagebuilderList.end(); it++) {
            if ((*it)->GetName() == meterName) {
                return index;
            }
            index++;
        }
    }
    return -1;
}

/* Finds meter from p0messagebuilderList basted on COM port */
int P0Control::findCOM(string commPortName) {
    if (p0messagebuilderList.size() > 0) {
        vector<P0MessageBuilder*>::iterator it;
        int index = 0;
        for (it = p0messagebuilderList.begin(); it != p0messagebuilderList.end(); it++) {
            if ((*it)->GetComm() == commPortName) {
                return index;
            }
            index++;
        }
    }
    return -1;
}
```

```
/* Destroys meter based on name */
void P0Control::destroyMeter(string meterName) {
    int index = findMeter(meterName);
    if (index != -1) {
        p0messagebuilderList.at(index)->~P0MessageBuilder();
        p0messagebuilderList.at(index) == NULL;
        p0messagebuilderList.erase(p0messagebuilderList.begin()+index);
    }
}

/*****************************************************************
 * Public Member Functions
 *****************************************************************/

void P0Control::OpenModeE(std::string commPortName, std::string DeviceAddress, u8 K[], u8 AAD[],
                          std::string meterName) {

    if (findMeter(meterName) != -1) {

        throw ("Error: Name " + meterName + " is already in use. Choose a different name");
    }

    if (findCOM(commPortName) != -1) {
        throw ("Error: COMport: " + commPortName + " is already in use. Choose a different one");
    }

    try {
        P0MessageBuilder *p0messagebuilder = new P0MessageBuilder(meterName, commPortName);

        if (p0messagebuilder->OpenModeE(commPortName, DeviceAddress, K, AAD)) {
            p0messagebuilderList.push_back(p0messagebuilder);
            cout << "p0messagebuilder: " << p0messagebuilder << endl;
            cout << "Success opening " + meterName + " on " + commPortName << endl;
        } else {
            throw ("ERROR: couldn't open modeE");
        }

    } catch (string msg) {
        throw ( msg + " for " + meterName);
    }

}

int P0Control::Get(u16 interface_class, u64 logical_name, u8 attribute, u8 dataOut[],
                   std::string meterName) {

    int index = findMeter(meterName);
    if (index == -1) {
        throw ("Error: Name " + meterName + " does not exist");
    }

    try {
        P0MessageBuilder* p0messagebuilder = p0messagebuilderList.at(index);
        return p0messagebuilder->Get(interface_class, logical_name, attribute, dataOut);
    } catch (string msg) {
        throw (msg + " for " + meterName);
    }
}

int P0Control::Get(string tag, u8 dataOut[], std::string meterName) {
    int index = findMeter(meterName);
    if (index == -1) {
        throw("Error: Name " + meterName+ " does not exist");
    }

    u16 interface_class;
    u64 logical_name;
    u8 attribute;
    u8 datatype;
    findTAG(tag, "get", interface_class, logical_name, attribute, datatype);

    try {
        P0MessageBuilder* p0messagebuilder = p0messagebuilderList.at(index);
        return p0messagebuilder->Get(interface_class, logical_name, attribute, dataOut);
    } catch (string msg) {
        throw (msg + " for " + meterName);
    }
    return 0;
```

```
}

int P0Control::Set(u16 interface_class, u64 logical_name, u8 attribute, u8 datatype, u8 dataIn[],
                   int dataInSize, u8 dataOut[], std::string meterName) {
    int index = findMeter(meterName);
    if (index == -1) {
        throw("Error: Name " + meterName+ " does not exist");
    }
    try {
        P0MessageBuilder* p0messagebuilder = p0messagebuilderList.at(index);
        return p0messagebuilder->Set(interface_class, logical_name, attribute, datatype, dataIn,
                                     dataInSize, dataOut);
    } catch (string msg) {
        throw (msg + " for " + meterName);
    }
    return 0;
}

int P0Control::Set(string tag, u8 dataIn[], int dataInSize, u8 dataOut[], std::string meterName)
{
    int index = findMeter(meterName);
    if (index == -1) {
        throw("Error: Name " + meterName+ " does not exist");
    }

    u16 interface_class;
    u64 logical_name;
    u8 attribute;
    u8 datatype;
    findTAG(tag, "set", interface_class, logical_name, attribute, datatype);

    try {
        P0MessageBuilder* p0messagebuilder = p0messagebuilderList.at(index);
        return p0messagebuilder->Set(interface_class, logical_name, attribute, datatype, dataIn,
                                     dataInSize, dataOut);
    } catch (string msg) {
        throw (msg + " for " + meterName);
    }
    return 0;
}

int P0Control::Action(u16 interface_class, u64 logical_name, u8 methode, u8 datatype,
                      u8 dataIn[], int dataInSize, u8 dataOut[], std::string meterName) {
    int index = findMeter(meterName);
    if (index == -1) {
        throw("Error: Name " + meterName+ " does not exist");
    }
    try {
        P0MessageBuilder* p0messagebuilder = p0messagebuilderList.at(index);
        return p0messagebuilder->Action(interface_class, logical_name, methode, datatype, dataIn,
                                        dataInSize, dataOut);
    } catch (string msg) {
        throw (msg + " for " + meterName);
    }
    return 0;
}

int P0Control::Action(string tag, u8 dataIn[], int dataInSize, u8 dataOut[],
                      std::string meterName) {
    int index = findMeter(meterName);
    if (index == -1) {
        throw("Error: Name " + meterName+ " does not exist");
    }
    P0MessageBuilder* p0messagebuilder = p0messagebuilderList[index];

    u16 interface_class;
    u64 logical_name;
    u8 methode;
    u8 datatype;
    findTAG(tag, "action", interface_class, logical_name, methode, datatype);

    try {
        P0MessageBuilder* p0messagebuilder = p0messagebuilderList.at(index);
        return p0messagebuilder->Action(interface_class, logical_name, methode, datatype, dataIn,
                                        dataInSize, dataOut);
    } catch (string msg) {
        throw (msg + " for " + meterName);
    }
```

```
    return 0;
}

void P0Control::CloseModeE(string meterName) {
    int index = findMeter(meterName);
    if (index == -1) {
        throw("Error: Name " + meterName+ " does not exist");
    }
    P0MessageBuilder* p0messagebuilder = p0messagebuilderList[index];
    p0messagebuilder->CloseModeE();
    destroyMeter(meterName);
    return;
}

void P0Control::OpenCOM(string commPortName, int bitRate, int prarity, int byte,
                        std::string meterName) {
    try {
        if (findMeter(meterName) != -1) {
            throw ("Error: Name " + meterName + " is already in use. Choose a different name");
        }
        if (findCOM(commPortName) != -1) {
            throw ("Error: COMport: " + commPortName +
                   " is already in use. Choose a different one");
        }

        P0MessageBuilder *p0messagebuilder = new P0MessageBuilder(meterName, commPortName);

        if (p0messagebuilder->OpenCOM(commPortName, bitRate, prarity, byte)) {
            p0messagebuilderList.push_back(p0messagebuilder);
        } else {
            throw ("ERROR: couldn't open " + commPortName);
        }

    } catch (string &msg) {
        throw(msg + " for " + meterName);
    }
}

int P0Control::SendRaw(u8 *bufferIn, int bufferInLenght, u8 *bufferOut, int readDelay,
                       int timeout, string find, std::string meterName) {
    int index = findMeter(meterName);
    if (index == -1) {
        throw("Error: Name " + meterName+ " does not exist");
    }
    P0MessageBuilder* p0messagebuilder = p0messagebuilderList[index];

    try {
        P0MessageBuilder* p0messagebuilder = p0messagebuilderList.at(index);
        return p0messagebuilder->SendRaw(bufferIn, bufferInLenght, bufferOut, readDelay,
                                         timeout, find);
    } catch (string msg) {
        throw (msg + " for " + meterName);
    }
    return 0;
}

void P0Control::CloseCOM(string meterName){
    int index = findMeter(meterName);
    if (index == -1) {
        throw("Error: Name " + meterName+ " does not exist");
    }
    destroyMeter(meterName);
}
```

# Bijlage F: P0MessageBuilder.h

```
//============================================================================================================
// Name        : P0MessageBuilder.h
// Author      : Niels Hokke
// Company     : Sogeti
// Description : Header File of P0MessageBuilder.cpp
//============================================================================================================

#ifndef P0MESSAGEBUILDER_H
#define P0MESSAGEBUILDER_H_

#include <string>
#include <unistd.h>
#include <iostream>
#include <time.h>
#include <thread>
#include <mutex>
#include "SerialDriver.h"

typedef unsigned long long int u64;
typedef unsigned int u32;
typedef unsigned short u16;
typedef unsigned char u8;

//HDLC frame types
#define FRAME_TYPE_I          1
#define FRAME_TYPE_RR         2
#define FRAME_TYPE_RNR        3
#define FRAME TYPE SNRM       4
#define FRAME TYPE DISC       5
#define FRAME TYPE UA         6
#define FRAME_TYPE_DM         7
#define FRAME_TYPE_FRMR       8
#define FRAME_TYPE_UI         9

//Mode E data types
#define DATA_TYPE_NULL_DATA                 0
#define DATA_TYPE_ARRAY                     1
#define DATA_TYPE_STRUCTURE                 2
#define DATA_TYPE_BOOLEAN                   3
#define DATA TYPE BIT STRING                4
#define DATA_TYPE_DOUBLE_LONG               5
#define DATA_TYPE_DOUBLE_LONG_UNSIGNED      6
#define DATA_TYPE_FLOATING_POINT            7
#define DATA_TYPE_OCTET_STRING              9
#define DATA TYPE VISIBLE STRING            10
#define DATA TYPE BCD                       13
#define DATA_TYPE_INTEGER                   15
#define DATA_TYPE_LONG                      16
#define DATA TYPE UNSIGNED                  17
#define DATA TYPE LONG UNSIGNED             18
#define DATA TYPE COMPACT ARRAY             19
#define DATA TYPE LONG 64                   20
#define DATA_TYPE_LONG_64_UNSIGNED          21
#define DATA_TYPE_ENUM                      22
#define DATA TYPE FLOAT 32                  33
#define DATA TYPE FLOAT 64                  24
#define DATA TYPE DATE TIME                 25
#define DATA TYPE DATE                      26
#define DATA_TYPE_TIME                      27
#define DATA_TYPE_DONT_CARE                 255

class P0MessageBuilder
{
private:

    /* Pointer of serialDriver object */
    SerialDriver *serialdriver;

    /* Name of meter instance */
    std::string Name;

    /* COM port of meter instance */
    std::string ComPortName;
```

```
    /* state variable */
    int status;

    /* Makes an unreadable ascii char readable */
    bool asciiSpecial(unsigned int symbol, std::string &response, bool enter);

    /* Makes unreadable ascii chars readable */
    std::string readable(u8 input[], int size);

    /* Time stamp of last send */
    time_t lastSendTime;

    /* Keeps the connection high when nothing is send */
    void antiTimeoutLoop();

    /* Used for the antiTimeoutLoop */
    std::thread antiLoopThread;

    /* Used to lock serial driver, the antiTimeoutLoop */
    std::mutex mtx1;


public:

    /* Object constructor, reserves a name and COM port */
    P0MessageBuilder(std::string meterName, std::string commPortName);

    /* Object destructor, destroys COM port object */
    ~P0MessageBuilder();

    /* Returns the name */
    std::string GetName();

    /* Returns the COM port */
    std::string GetComm();

    /* Opens mode E */
    bool OpenModeE(std::string commPortName, std::string DeviceAddress, u8 K[], u8 AAD[]);

    /* Gets data in mode E */
    int Get(u16 interface_class, u64 logical_name, u8 attribute, u8 dataOut[],
            bool ignoreConnection = false);

    /* Sets data in mode E */
    int Set(u16 interface_class, u64 logical_name, u8 attribute, u8 datatype, u8 dataIn[],
            int dataInSize, u8 dataOut[]);

    /* Does an Action on data in mode E */
    int Action(u16 interface_class, u64 logical_name, u8 methode, u8 datatype, u8 dataIn[],
               int dataInSize, u8 dataOut[]);

    /* Closes mode E */
    void CloseModeE();

    /* Opens the COM port */
    bool OpenCOM(std::string commPortName, int bitRate = 300, int prarity = 1, int byte = 7);

    /* Sends raw byte's and returns raw answer */
    int SendRaw(u8 bufferIn[], int bufferInLenght, u8 bufferOut[], int readDelay, int timeout,
                std::string find);

    /* Closes COM port */
    void CloseCOM();

};


#endif /* P0MESSAGEBUILDER_H_ */
```

# Bijlage G: P0MessageBuilder.cpp

```
//============================================================================================================
// Name        : P0MessageBuilder.cpp
// Author      : Niels Hokke
// Company     : Sogeti
// Description : Builds binary Messages in mode E to send to Smart Meter
//============================================================================================================

#define debug    3
#define info     2
#define warning  1
#define error    0

#define log level debug
#define LOG(Lev) if (log_level >= Lev) std::cout
#define D(x) if (log_level == debug) x

#include <string>
#include <unistd.h>
#include <iostream>
#include <stdio.h>
#include <thread>
#include <chrono>
#include "P0MessageBuilder.h"
#include "cryptoBlock.h"


//most used data types
typedef unsigned long long int u64;
typedef unsigned int u32;
typedef unsigned short u16;
typedef unsigned char u8;

//Most used ascii chars
#define CR 13
#define LF 10
#define ACK 6

//Initial and good FCS value
#define PPPINITFCS16 0xffff
#define PPPGOODFCS16 0x0f47

//Global HDLC params
u8 GLOBAL_upperServerAddress = 0x01;
u8 GLOBAL_lowerServerAddress = 0x11;
u8 GLOBAL_ClientAdress = 0x01;
u8 GLOBAL_RRR = 0;
u8 GLOBAL_SSS = 0;

//Global encryption/autorisation params
u8 GLOBAL_K[16];
u8 GLOBAL_AAD[17];
u8 GLOBAL_System_Titel[8] = {'N', 'H', 'H', '1', '2', '2', '9', '5'};
u8 GLOBAL_System_Titel_SM[8];
u32 GLOBAL_frameCounter = 0x0F;

//TODO
bool GLOBAL_Encryption = false;

//TODO
bool GLOBAL_ANTI_TIMEOUT = false;

using namespace std;

/*********************************************************************************************************
 * Private Functions
 *********************************************************************************************************/

/* displays byte arrays */
static char ascii(char s) {
    if (s < 0x20) return '.';
    if (s > 0x7E) return '.';
    return s;
}
```

```
/* displays byte arrays */
void hexdump(void *d, int len) {
    bool debugL = false;
    D(debugL = true);
    if (debugL){
        return;
    }
    if (len < 1) return;
    u8 *data;
    int i, off;
    data = (u8*) d;
    for (off = 0; off < len; off += 16) {
        printf("%08x  ", off);
        for (i = 0; i < 16; i++)
            if ((i + off) >= len) printf("   ");
            else printf("%02x ", data[off + i]);

        printf(" ");
        for (i = 0; i < 16; i++)
            if ((i + off) >= len) printf(" ");
            else printf("%c", ascii(data[off + i]));
        printf("\n");
    }
}

/* used to calculates checksum */
u16 pppfcs16(u16 fcs, unsigned char *cp, int len) {
    static u16 fcstab[256] = {
        0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,
        0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,
        0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,
        0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,
        0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,
        0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xfae7, 0xc87c, 0xd9f5,
        0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,
        0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,
        0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,
        0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,
        0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,
        0xdecd, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
        0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
        0xef4e, 0xfec7, 0xcc5c, 0xddd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,
        0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,
        0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,
        0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,
        0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,
        0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,
        0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
        0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
        0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
        0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
        0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
        0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
        0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
        0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
        0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
        0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
        0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
        0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
        0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
    };
    while (len--)
        fcs = (fcs >> 8) ^ fcstab[(fcs ^ *cp++) & 0xff];
    return ~(fcs);
}

/* calculates checksum */
u16 UpdateCheckSum(void *Data, u32 Length) {
    if (Length < 5)
        return -1;
    u8 *ptr = (u8*) Data;
    u16 chk = pppfcs16(PPPINITFCS16, ptr, Length);
    return chk;
}
```

```
/* Builds HDLC frame around data */
int HDLCbuilder(u16 UpperDestinationAddress, u16 LowerDestinationAddress, u16 SourceAddress,
                int FrameType, u8 RRR, u8 SSS, u8 PF, u8 information[], int informationLenght,
                u8 output[]) {
    int pointer = 0;

    int formatType = 10;
    int segmentationBit = 0;
    int frameLenghtSubField = 0; //yet unknown
    int HFClength = 0;

    //flag
    output[0] = 0x7e;

    //destination / source adress
    if (UpperDestinationAddress > 265 || LowerDestinationAddress > 265) {
        output[3] = (UpperDestinationAddress >> 6) & 0xfe;
        output[4] = (UpperDestinationAddress << 1) & 0xfe;
        output[5] = (LowerDestinationAddress >> 6) & 0xfe;
        output[6] = (LowerDestinationAddress << 1) | 1;
        pointer = 7;
    } else if (LowerDestinationAddress > 0) {
        output[3] = (UpperDestinationAddress << 1) & 0xfe;
        output[4] = (LowerDestinationAddress << 1) | 1;
        pointer = 5;
    } else {
        output[3] = (UpperDestinationAddress << 1) | 1;
        pointer = 4;
    }
    output[pointer] = (SourceAddress << 1) | 1;
    pointer++;
    HFClength = pointer;

    //control byte
    u8 bitOne;
    if (FrameType == FRAME_TYPE_I) {
        bitOne = 0;
    } else if (FrameType == FRAME_TYPE_RR) {
        bitOne = 1;
        SSS = 0;
    } else if (FrameType == FRAME_TYPE_RNR) {
        bitOne = 1;
        SSS = 2;
    } else if (FrameType == FRAME_TYPE_SNRM) {
        bitOne = 1;
        SSS = 1;
        RRR = 4;
    } else if (FrameType == FRAME_TYPE_DISC) {
        bitOne = 1;
        SSS = 1;
        RRR = 2;
    } else if (FrameType == FRAME_TYPE_UA) {
        bitOne = 1;
        SSS = 1;
        RRR = 3;
    } else if (FrameType == FRAME_TYPE_DM) {
        bitOne = 1;
        SSS = 7;
        RRR = 0;
    } else if (FrameType == FRAME_TYPE_FRMR) {
        bitOne = 1;
        SSS = 3;
        RRR = 4;
    } else if (FrameType == FRAME_TYPE_UI) {
        bitOne = 1;
        SSS = 1;
        RRR = 0;
    } else {
        throw (string("Error: Frametype not supported."));
    }
    output[pointer] = ((RRR << 5) & 0xe0) | ((PF << 4) & 0x10) | ((SSS << 1) & 0x0e) |
                      (bitOne & 1);
    pointer += 3;
```

```
    //frame format / HCS / information / FCS / flag
    if (informationLenght == 0) {
        frameLenghtSubField = pointer;
        u16 frameFormat = ((formatType & 0x0f) << 12) | ((segmentationBit & 0x01) << 11) |
                          (frameLenghtSubField & 0x7FF);
        output[1] = frameFormat >> 8;
        output[2] = frameFormat & 0xff;
        u16 HCS = UpdateCheckSum(output + 1, HFClength);
        output[HFClength + 1] = HCS & 0xFF;
        output[HFClength + 2] = HCS >> 8;
        output[pointer] = 0x7e;
    } else {
        int i;
        for (i = 0; i < informationLenght; i++) {
            output[pointer + i] = information[i];
        }
        pointer += i;
        frameLenghtSubField = pointer + 1;
        u16 frameFormat = ((formatType & 0x0f) << 12) | ((segmentationBit & 0x01) << 11) |
                          (frameLenghtSubField & 0x7FF);
        output[1] = frameFormat >> 8;
        output[2] = frameFormat & 0xff;

        u16 HCS = UpdateCheckSum(output + 1, HFClength);
        output[HFClength + 1] = HCS & 0xFF;
        output[HFClength + 2] = HCS >> 8;

        u16 FCS = UpdateCheckSum(output + 1, frameLenghtSubField - 2);
        output[pointer] = FCS & 0xFF;
        output[pointer + 1] = FCS >> 8;
        pointer += 2;
        output[pointer] = 0x7e;
    }
    return (int) frameLenghtSubField + 2;
}

/* Builds SNMR frame */
int SNMRframeBuilder(u8 frame[], int &framelenght, u8 upperServerAddress,
                     u8 lowerServerAddress, u8 ClientAdress, u8 MaxfieldTx,
                     u8 MaxFieldRx, unsigned int MaxWindowTx, unsigned int MaxWindowRx) {
    //building SNMR information field
    u8 information[21];
    information[0] = 0x81;
    information[1] = 0x80;
    information[2] = 0x12;
    information[3] = 0x05;
    information[4] = 0x01;
    information[5] = MaxfieldTx;
    information[6] = 0x06;
    information[7] = 0x01;
    information[8] = MaxFieldRx;
    information[9] = 0x07;
    information[10] = 0x04;
    information[11] = (MaxWindowTx >> 24) & 0xFF;
    information[12] = (MaxWindowTx >> 16) & 0xFF;
    information[13] = (MaxWindowTx >> 8) & 0xFF;
    information[14] = (MaxWindowTx) & 0xFF;
    information[15] = 0x08;
    information[16] = 0x04;
    information[17] = (MaxWindowRx >> 24) & 0xFF;
    information[18] = (MaxWindowRx >> 16) & 0xFF;
    information[19] = (MaxWindowRx >> 8) & 0xFF;
    information[20] = (MaxWindowRx) & 0xFF;

    //building HDLC frame around SNMR information field
    framelenght = HDLCbuilder(upperServerAddress, lowerServerAddress, ClientAdress,
                              FRAME_TYPE_SNRM, 0, 0, 0, information, sizeof (information),
                              frame);
    return framelenght;
}

/* Dissects a HDLC frame and returns the information field */
int HDLContleed(unsigned char input[], int inputlenght, u8 information[]) {
    //min lenght HDLC frame 8
    if (inputlenght < 8) return -1;
```

```
    int pointer = 0;


    //finding first flag
    while (input[pointer] != 0x7e) {
        pointer++;
        if (pointer == inputlenght) {
            LOG(warning) << "warning: No beginning flag found" << endl;
            return -1;
        }
    }
    int start = pointer;
    D(printf("%02x \tBeginning Flag \n", input[pointer]));

    //frame format
    D(printf("%02x %02x \tFrame format\n", input[pointer + 1], input[pointer + 2]));
    u16 frameFormat = (input[pointer + 1] << 8) + input[pointer + 2];
    int type = frameFormat >> 12;
    int Segmentation = (frameFormat >> 11) & 1;
    int frameLenght = frameFormat & 0x7FF;
    if (type == 10) {
        D(printf("\t\tFrame type: 3\n"));
    } else {
        D(printf("\t\tFrame type: %0d\n", type));
    }
    D(printf("\t\tSegmentation bit: %0d\n", Segmentation));
    D(printf("\t\tFrameLenght: %0d\n", frameLenght));
    if (frameLenght > 500) {
        throw (string("Error: frameLenght > 500"));//warning?
    }
    pointer += 3;

    //Destination address
    int teller = 0;
    while ((input[pointer + teller] & 0x01) != 1) {
        teller++;
    }
    teller++;
    if (teller == 1) {
        D(printf("%02x \tDestination address: %0d\n", input[pointer], (input[pointer] >> 1)));
    } else if (teller == 2) {
        D(printf("%02x \tDestination address (server) upper HDLC address: %0d\n", input[pointer],
                (input[pointer] >> 1)));
        D(printf("%02x \tDestination address (server) lower HDLC address: %0d\n",
                input[pointer + 1], (input[pointer + 1] >> 1)));
    } else if (teller == 4) {
        D(printf("%02x %02x \tDestination address (server) upper HDLC address: %0d\n",
                input[pointer], input[pointer + 1], ((input[pointer] << 7) +
                (input[pointer + 1] >> 1))));
        D(printf("%02x %02x \tDestination address (server) lower HDLC address: %0d\n",
                input[pointer + 2], input[pointer + 3], ((input[pointer + 2] << 7) +
                (input[pointer + 3] >> 1))));
    } else {
        LOG(warning) << "warning: Destination addres not found " << teller << endl;
        return -1;
    }
    pointer += teller;

    //Source  address
    teller = 0;
    while ((input[pointer + teller] & 0x01) != 1) {
        teller++;
    }
    teller++;
    if (teller == 1) {
        D(printf("%02x \tSource  address: %0d\n", input[pointer], (input[pointer] >> 1)));
    } else if (teller == 2) {
        D(printf("%02x \tSource address (server) upper HDLC address: %0d\n", input[pointer],
                (input[pointer] >> 1)));
        D(printf("%02x \tSource address (server) lower HDLC address: %0d\n", input[pointer + 1],
                (input[pointer + 1] >> 1)));
    } else if (teller == 4) {
        D(printf("%02x %02x \tSource address (server) upper HDLC address: %0d\n", input[pointer],
                input[pointer + 1], ((input[pointer] << 7) + (input[pointer + 1] >> 1))));
        D(printf("%02x %02x \tSource address (server) lower HDLC address: %0d\n",
                input[pointer + 2], input[pointer + 3], ((input[pointer + 2] << 7) +
                (input[pointer + 3] >> 1))));
    } else {
```

```
        LOG(warning) << "warning: Source addres not found " << teller << endl;
        return -1;
    }
    pointer += teller;

    //Control byte
    D(printf("%02x \tControl Byte\n", input[pointer]));
    if ((input[pointer] & 1) == 0) {
        D(printf("\t\tFrame type I frame\n"));
        D(printf("\t\tRRR: %d\n", input[pointer] >> 5));
        D(printf("\t\tSSS: %d\n", (input[pointer] >> 1) & 0x07));
        D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
    } else {
        if (((input[pointer] >> 1) & 0x07) == 0) {
            D(printf("\t\tFrame type RR frame\n"));
            D(printf("\t\tRRR: %d\n", input[pointer] >> 5));
            D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
        } else if (((input[pointer] >> 1) & 0x07) == 2) {
            D(printf("\t\tFrame type RNR frame\n"));
            D(printf("\t\tRRR: %d\n", input[pointer] >> 5));
            D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
        } else if (((input[pointer] >> 1) & 0x07) == 1) {
            if ((input[pointer] >> 5) == 4) {
                D(printf("\t\tFrame type SNRM frame\n"));
                D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
            } else if ((input[pointer] >> 5) == 2) {
                D(printf("\t\tFrame type DISC frame\n"));
                D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
            } else if ((input[pointer] >> 5) == 3) {
                D(printf("\t\tFrame type UA frame\n"));
                D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
            } else if ((input[pointer] >> 5) == 0) {
                D(printf("\t\tFrame type UI frame\n"));
                D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
            } else {
                LOG(warning) << "warning: incorrect ontrol byte" << endl;
            }
        } else if (((input[pointer] >> 1) & 0x07) == 7) {
            D(printf("\t\tFrame type DM frame\n"));
            D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
        } else if (((input[pointer] >> 1) & 0x07) == 3) {
            D(printf("\t\tFrame type FRMR frame\n"));
            D(printf("\t\tP/F: %d\n", (input[pointer] >> 4) & 0x01));
        } else {
            LOG(warning) << "warning: incorrect ontrol byte" << endl;
        }
    }
    pointer++;

    //HCS
    D(printf("%02x %02x \tHCS Bytes\n", input[pointer], input[pointer + 1]));
    u16 HCS = UpdateCheckSum(input + start + 1, pointer - start + 1);
    if (HCS == PPPGOODFCS16) {
        D(printf("\t\tHCS correct\n"));
    } else {
        D(printf("\t\tHCS incorrect\n"));
    }
    pointer += 2;

    //Information field and FCS
    int informationLenght = 0;
    if (frameLenght + 1 != pointer) {
        for (int i = 0; pointer < frameLenght - 1; pointer++) {
            if (i == 1) {
                D(printf("%02x \tInformation\n", input[pointer]));
            } else if (i % 2 || pointer == frameLenght - 2) {
                D(printf("%02x\n", input[pointer]));
            } else {
                D(printf("%02x ", input[pointer]));
            }
            information[i] = input[pointer];
            informationLenght = ++i;
        }

        D(printf("%02x %02x \tFCS Bytes\n", input[pointer], input[pointer + 1]));
        u16 FCS = UpdateCheckSum(input + start + 1, pointer - start + 1);
        if (FCS == PPPGOODFCS16) {
            D(printf("\t\tFCS correct\n"));
```

```
        } else {
            D(printf("\t\tFCS incorrect\n"));
        }
        pointer += 2;
    } else {
        D(printf("\tNo information field\n"));
    }

    //End flag
    if (input[pointer] == 0x7e) {
        D(printf("%02x \tEnd Flag \n", input[pointer]));
    } else {
        D(printf("No end flag found\n"));
    }
    return informationLenght;
}

/* Builds encryption string */
int encryptedStringBuilder(u8 information[], int information_size, u8 ouputstring[]) {
    u8 TAG[12];
    u8 IV[12];
    u8 ciphertext[100];
    int r = -1;

    //Software crypto-core
    cryptoBlock cryptoBlock;

    //Building IV
    for (int i = 0; i < 8; i++) {
        IV[i] = GLOBAL_System_Titel[i];
    }
    IV[8] = (GLOBAL_frameCounter >> 24) & 0xff;
    IV[9] = (GLOBAL_frameCounter >> 16) & 0xff;
    IV[10] = (GLOBAL_frameCounter >> 8) & 0xff;
    IV[11] = GLOBAL_frameCounter & 0xff;

    //encrypting data
    r = cryptoBlock.encrypt(information, information_size, GLOBAL_AAD, sizeof (GLOBAL_AAD),
                            GLOBAL_K, IV, ciphertext, TAG);

    //Building encryption frame
    //SC
    ouputstring[0] = 0x30;
    //frame counter
    ouputstring[1] = (GLOBAL_frameCounter >> 24) & 0xff;
    ouputstring[2] = (GLOBAL_frameCounter >> 16) & 0xff;
    ouputstring[3] = (GLOBAL_frameCounter >> 8) & 0xff;
    ouputstring[4] = GLOBAL_frameCounter & 0xff;

    GLOBAL_frameCounter++;

    //Cypher text
    for (int i = 0; i < r; i++) {
        ouputstring[1 + 4 + i] = ciphertext[i];
    }
    //Authentication tag
    for (int i = 0; i < 12; i++) {
        ouputstring[1 + 4 + r + i] = TAG[i];
    }

    return (1 + 4 + r + 12);
}

/* Builds AARQ frame */
int AARQBuilder(u8 frame[]) {
    //Building AARQ information field
    u8 information[128];
    information[0] = 0xe6; //LLC Bytes
    information[1] = 0xe6;
    information[2] = 0x00;
    information[3] = 0x60; //AARQ TAG
    information[4] = 93; //Lenght 93
    information[5] = 0xa1; //Application context name tag
    information[6] = 0x09; //Lenght 9
    information[7] = 0x06; //AA_ApplicationContextName tag
    information[8] = 0x07; //Lenght 7
    information[9] = 0x60; //value
    information[10] = 0x85;
```

```
    information[11] = 0x74;
    information[12] = 0x05;
    information[13] = 0x08;
    information[14] = 0x01;
    information[15] = 0x03;
    information[16] = 0x06; //Application context name tag
    information[17] = 0x0a; //Lenght 10
    information[18] = 0x04; //AA_APTitle_Calling tag (system title)
    information[19] = 0x08; //Lenght
    information[20] = GLOBAL_System_Titel[0]; //value
    information[21] = GLOBAL_System_Titel[1];
    information[22] = GLOBAL_System_Titel[2];
    information[23] = GLOBAL_System_Titel[3];
    information[24] = GLOBAL_System_Titel[4];
    information[25] = GLOBAL_System_Titel[5];
    information[26] = GLOBAL_System_Titel[6];
    information[27] = GLOBAL_System_Titel[7];
    information[28] = 0x8a; //Application context name tag
    information[29] = 0x02; //Lenght 2
    information[30] = 0x07; //AA_ACSERequirements_Sender
    information[31] = 0x80; //value (True)
    information[32] = 0x8b; //AA_MechanismName tag
    information[33] = 0x07; //Lenght 7
    information[34] = 0x60; //value
    information[35] = 0x85;
    information[36] = 0x74;
    information[37] = 0x05;
    information[38] = 0x08;
    information[39] = 0x02;
    information[40] = 0x05;
    information[41] = 0xac; //Application context name tag
    information[42] = 0x12; //Lenght
    information[43] = 0x80; //AA_AuthenticationValue_Calling
    information[44] = 0x10; //Lenght 16
    information[45] = 0x4d; //value
    information[46] = 0x2c;
    information[47] = 0x3a;
    information[48] = 0x53;
    information[49] = 0x5d;
    information[50] = 0x25;
    information[51] = 0x7b;
    information[52] = 0x66;
    information[53] = 0x41;
    information[54] = 0x58;
    information[55] = 0x34;
    information[56] = 0x36;
    information[57] = 0x69;
    information[58] = 0x3a;
    information[59] = 0x3a;
    information[60] = 0x42;
    information[61] = 0xbe; //AA_AssociationInformation-TAG
    information[62] = 0x23; //length 35
    information[63] = 0x04; //Data type OCTET STRING, universal
    information[64] = 0x21; //Lenght 33
    information[65] = 0x21; //glo-initiateRequest, length 31
    information[66] = 0x1f; //length 31

    //Association information
    u8 message[] = {0x01, 0x00, 0x00, 0x00, 0x06, 0x5f, 0x1f, 0x04, 0x00, 0x00, 0x1f, 0x1f, 0x08,
                    0x00};
    u8 encryptedMessage[100];
    int r = encryptedStringBuilder(message, sizeof (message), encryptedMessage);
    for (int i = 0; i < r; i++) {
        information[67 + i] = encryptedMessage[i];
    }

    //Building HDLC frame around AARQ information
    int framelenght = HDLCbuilder(0x01, 0x11, 0x01, FRAME_TYPE_I, 0, 0, 1, information, 67 + r,
                                  frame);
    return framelenght;
}

/* Dissects a AARE frame */
int AAREontleed(u8 informatie[], int lenght, u8 AuthenticationString[]) {
    int autothorisationTagLenght = -1;
    int pointer = 0;

    //finding LLC bytes
```

```
    while (!(informatie[pointer] == 0xe6 && informatie[pointer + 1] == 0xe7 &&
             informatie[pointer + 2] == 0x00)) {
        if (pointer > lenght) {
            throw ("ERROR: error, LLC bytes not found");
        }
        pointer++;
    }
    D(printf("%02x %02x %02x\tLLC Bytes\n", informatie[pointer], informatie[pointer + 1],
             informatie[pointer + 2]));
    pointer += 3;

    //AARE tag
    if (informatie[pointer] != 0x61) {
        throw ("ERROR: error, not a AARE message");
    }
    D(printf("%02x %02x\tAARE-tag, Lenght %d\n", informatie[pointer], informatie[pointer + 1],
             informatie[pointer + 1]));
    pointer += 2;

    //AARE fields
    while (pointer < lenght) {
        //Application context name
        if (informatie[pointer] == 0xa1) {
            D(printf("%02x %02x\tApplication-context-name tag, lenght %d\n", informatie[pointer],
                     informatie[pointer + 1], informatie[pointer + 1]));
            pointer += 2;
            D(printf("\t%02x %02x\tSomething tag, lenght %d\n", informatie[pointer],
                     informatie[pointer + 1], informatie[pointer + 1]));
            int lengte = informatie[pointer + 1];
            pointer += 2;
            D(printf("\t\t"));
            for (int i = 0; i < lengte; i++) {
                D(printf("%02x ", informatie[pointer + i]));
            }
            D(printf("Application-context-name\n"));
            pointer += lengte;

        //Association result
        } else if (informatie[pointer] == 0xa2) {
            D(printf("%02x %02x\tAssociation-result tag, lenght %d\n", informatie[pointer],
                      informatie[pointer + 1], informatie[pointer + 1]));
            pointer += 2;
            D(printf("\t%02x %02x\tSomething tag, lenght %d\n", informatie[pointer],
                     informatie[pointer + 1], informatie[pointer + 1]));
            int lengte = informatie[pointer + 1];
            pointer += 2;
            D(printf("\t\t"));
            for (int i = 0; i < lengte; i++) {
                D(printf("%02x ", informatie[pointer + i]));
            }
            D(printf("Association-result\n"));
            pointer += lengte;

        //Associate source diagnostic
        } else if (informatie[pointer] == 0xa3) {
            D(printf("%02x %02x\tAssociate-source-diagnostic tag, lenght %d\n",
                     informatie[pointer], informatie[pointer + 1], informatie[pointer + 1]));
            pointer += 2;
            D(printf("\t%02x %02x\tSomething tag, lenght %d\n", informatie[pointer],
                     informatie[pointer + 1], informatie[pointer + 1]));
            int lengte = informatie[pointer + 1];
            pointer += 2;
            D(printf("\t\t"));
            for (int i = 0; i < lengte; i++) {
                D(printf("%02x ", informatie[pointer + i]));
            }
            D(printf("Associate-source-diagnostic\n"));
            pointer += lengte;

        //AP title
        } else if (informatie[pointer] == 0xa4) {
            D(printf("%02x %02x\tAP-title tag, lenght %d\n", informatie[pointer],
                     informatie[pointer + 1], informatie[pointer + 1]));
            pointer += 2;
            D(printf("\t%02x %02x\tSomething tag, lenght %d\n", informatie[pointer],
                     informatie[pointer + 1], informatie[pointer + 1]));
            int lengte = informatie[pointer + 1];
            pointer += 2;
```

```
            D(printf("\t\t"));
            for (int i = 0; i < lengte; i++) {
                D(printf("%02x ", informatie[pointer + i]));
                GLOBAL_System_Titel_SM[i] = informatie[pointer + i];
            }
            D(printf("AP-title\n"));
            pointer += lengte;

        //Authentication value
        } else if (informatie[pointer] == 0xaa) {
            D(printf("%02x %02x\tAuthentication-value tag, lenght %d\n", informatie[pointer],
                    informatie[pointer + 1], informatie[pointer + 1]));
            pointer += 2;
            D(printf("\t%02x %02x\tSomething tag, lenght %d\n", informatie[pointer],
                    informatie[pointer + 1], informatie[pointer + 1]));
            int lengte = informatie[pointer + 1];
            pointer += 2;
            D(printf("\t\t"));
            for (int i = 0; i < lengte; i++) {
                D(printf("%02x ", informatie[pointer + i]));
                AuthenticationString[i] = informatie[pointer + i];
            }
            autothorisationTagLenght = lengte;
            D(printf("Authentication-value\n"));
            pointer += lengte;

        //Association information
        } else if (informatie[pointer] == 0xbe) {
            D(printf("%02x %02x\tAssociation-information tag, lenght %d\n", informatie[pointer],
                    informatie[pointer + 1], informatie[pointer + 1]));
            pointer += 2;
            D(printf("\t%02x %02x\tSomething tag, lenght %d\n", informatie[pointer],
                    informatie[pointer + 1], informatie[pointer + 1]));
            int lengte = informatie[pointer + 1];
            pointer += 2;
            for (int i = 0; i < lengte; i++) {
                if (i % 2 != 0) {
                    D(printf("%02x\n", informatie[pointer + i]));
                } else {
                    D(printf("\t\t%02x ", informatie[pointer + i]));
                }
            }
            D(printf("\n"));
            pointer += lengte;
        } else {
            D(printf("%02x unknown\n", informatie[pointer]));
            pointer++;
        }
    }
    return autothorisationTagLenght;
}

/* decrypts an encrypted string */
int stringdecrypter(u8 encryptedInfo[], int SL, u8 plaintext[]) {
    //min lenght 1
    if (SL < 1) {
        return -1;
    }

    int r = 0;
    try {
        //finding start encrytion frame
        int start = 0;
        while (encryptedInfo[start] != 0x30 && encryptedInfo[start] != 0x10) {
            start++;
            if (start >= SL) {
                throw ("Error: Encryption tag not found, wrong encryption key?");
            }
        }

        //getting length
        int lenght = encryptedInfo[start - 1];
        start++;

        //frame counter
        u8 framecounter[4];
        for (int i = 0; i < 4; i++) {
            framecounter[i] = encryptedInfo[start + i];
```

```
        }

        //Cypher text
        u8 CT[128];
        for (int i = 0; i < (lenght - 13 - 4); i++) {
            CT[i] = encryptedInfo[start + 4 + i];
        }

        //Building IV
        u8 IV[12];
        for (int i = 0; i < 8; i++) {
            IV[i] = GLOBAL_System_Titel_SM[i];
        }
        for (int i = 0; i < 4; i++) {
            IV[8 + i] = framecounter[i];
        }

        //TAG
        u8 TAG[12];
        for (int i = 0; i < 12; i++) {
            TAG[i] = encryptedInfo[start + lenght - 13 + i];
        }

        //decrypting
        cryptoBlock cryptoBlock;
        r = cryptoBlock.decrypt(CT, lenght - 13 - 4, GLOBAL_AAD, sizeof (GLOBAL_AAD),
                                GLOBAL_K, IV, plaintext, TAG);

        //printing encryption params
//        LOG(debug) << endl << "CT" << endl;
//        hexdump(CT, lenght-13-4);
//
//        LOG(debug) << endl << "AAD" << endl;
//        hexdump(GLOBAL_AAD, sizeof(GLOBAL_AAD));
//
//        LOG(debug) << endl << "K" << endl;
//        hexdump(GLOBAL_K, 16);
//
//        LOG(debug) << endl << "TAG" << endl;
//        hexdump(TAG, 12);
//
//        LOG(debug) << endl << "IV" << endl;
//        hexdump(IV, 12);
//        cout << endl << "plaintext" << endl;
//        hexdump(plaintext, r);

    } catch (const char *msg) {
        LOG(error) << msg << endl;
        return -1;
    }
    return r;
}

/*****************************************************************************************************
 * Private Member Functions
 *****************************************************************************************************/

/* keeps the connection high when nothing is send */
void P0MessageBuilder::antiTimeoutLoop() {
    LOG(debug) << "anti timeout started" << endl;
    mtx1.lock();
    while (GLOBAL_ANTI_TIMEOUT) {
        double seconds_since_last_send = difftime(time(0), lastSendTime);
        mtx1.unlock();
        LOG(debug) << "anti timeout checking last send" << endl;

        if (seconds_since_last_send > 4) {
            u8 response[128];

            LOG(debug) << "anti timeout get" << endl;
            int r = 0;
            try{
                r = Get(0x08, 0x0000010000FF, 2, response, true);
                if (r < 1){
                    r = Get(0x08, 0x0000010000FF, 1, response, true);
                }
            } catch (string &msg) {
                //no response
```

```
            }
            mtx1.lock();
            if (r > 0) {
                status = 1;
            } else {
                LOG(debug) << "anti timeout no response!" << endl;
                status = 0;
            }
            mtx1.unlock();
        }

        this_thread::sleep_for(chrono::milliseconds(2*1000));

        mtx1.lock();

    }
    mtx1.unlock();
}

/* Makes an unreadable ascii char readable */
bool P0MessageBuilder::asciiSpecial(unsigned int symbol, string &response, bool enter) {
    switch (symbol) {
        case 0: response += " "; break;
        case 1: response += "<SOH>"; break;
        case 2: response += "<STX>"; break;
        case 3: response += "<ETX>"; break;
        case 4: response += "<EOT>"; break;
        case 5: response += "<ENQ>"; break;
        case 6: response += "<ACK>"; break;
        case 7: response += "<BEL>"; break;
        case 8: response += "<BS>"; break;
        case 9: response += "<TAB>"; break;
        case 10:
            if (enter) {
                response += "<LF>\n";
            } else {
                response += "<LF>";
            }
            break;
        case 11: response += "<VT>"; break;
        case 12: response += "<FF>"; break;
        case 13: response += "<CR>"; break;
        case 14: response += "<SO>"; break;
        case 15: response += "<SI>"; break;
        case 16: response += "<DLE>"; break;
        case 17: response += "<DC1>"; break;
        case 18: response += "<DC2>"; break;
        case 19: response += "<DC3>"; break;
        case 20: response += "<DC4>"; break;
        case 21: response += "<NAK>"; break;
        case 22: response += "<SYN>"; break;
        case 23: response += "<ETB>"; break;
        case 24: response += "<CAN>"; break;
        case 25: response += "<EM>"; break;
        case 26: response += "<SUB>"; break;
        case 27: response += "<ESC>"; break;
        case 28: response += "<FS>"; break;
        case 29: response += "<GS>"; break;
        case 30: response += "<RS>"; break;
        case 31: response += "<US>"; break;
        default: return false;
    }
    return true;
}

/* Makes unreadable ascii chars readable */
string P0MessageBuilder::readable(u8 input[], int size) {
    string output = "";
    for (int i = 0; i < size; i++) {
        if (!asciiSpecial(input[i], output, false)) {
            output += input[i];
        }
    }
    return output;
}
```

```
/*************************************************************
 * Public Member Functions
 *************************************************************/

/* Object constructor, reserves a name and COM port */
P0MessageBuilder::P0MessageBuilder(string meternaam, string commPortName) {
    Name = meternaam;
    ComPortName = commPortName;
    lastSendTime = time(0);
    status = 0;
    serialdriver = NULL;
    //thread pointer = NULL
}

/* Object destructor, closes COM port if still open */
P0MessageBuilder::~P0MessageBuilder(){

    //if(antiLoopThread != NULL){
        mtx1.lock();
        GLOBAL_ANTI_TIMEOUT = false;
        mtx1.unlock();

        antiLoopThread.join();
    //}

    delete &serialdriver;
    serialdriver = NULL;
}

/* Opens mode E */
bool P0MessageBuilder::OpenModeE(string commPortName, string DeviceAddress, u8 K[], u8 AAD[]) {
    if (status == 1) {
        throw ("ERROR: already connected");
    }
    //Opening commPort
    LOG(debug) << "Opening " << commPortName << endl;
    serialdriver = new SerialDriver(commPortName, 300, EVEN, 7);
    ComPortName = commPortName;
    serialdriver->flush();
    LOG(debug) << commPortName << " opened\n" << endl;

    /****************************** Sending request message *******************************/

    //Building request message, IEC 62056-21 6.3.1
    u8 message[50] = {0};
    message[0] = 47;
    message[1] = '?';
    for (unsigned int i = 0; i < DeviceAddress.length(); i++) {
        message[i + 2] = DeviceAddress[i];
    }
    message[DeviceAddress.length() + 2] = '!';
    message[DeviceAddress.length() + 3] = CR;
    message[DeviceAddress.length() + 4] = LF;

    //Sending request message
    u8 response[50] = {0};
    LOG(debug) << "sending1: " << readable(message, DeviceAddress.length() + 5) << endl;


    int bytesRead = serialdriver->SendData(message, DeviceAddress.length() + 5, response, 100,
                                           2000, "\r\n");



    /*************************** Reading Identification Message ****************************/

    //reading response, IEC 62056-21 6.3.2
    string readableResponse = readable(response, bytesRead);
    LOG(debug) << "received1: " << readableResponse << endl << endl;
    int teller = 1;
    while (bytesRead == 0) {
        bytesRead = serialdriver->SendData(message, DeviceAddress.length() + 5, response, 100,
                                           2000, "\r\n");
        readableResponse = readable(response, bytesRead);
        teller++;
```

```
        if (teller > 10) {
            throw (string("ERROR: 10 times no response"));
        }
    }
    if (readableResponse.find("<CR><LF>") == string::npos) {
        throw (string("ERROR: Incorrect response"));
    }

    //Dissecting response
    string ManufactIDShort = readableResponse.substr(readableResponse.find("/") + 1, 3);
    LOG(debug) << "ManufactIDShort: " << ManufactIDShort << endl;
    char bautrateChar = readableResponse[readableResponse.find("/") + 4];
    int baudrate = -1;
    string mode = "C or E";
    switch (bautrateChar) {
        case '0': baudrate = 300; break;
        case 'A': mode = 'B';
        case '1': baudrate = 600; break;
        case 'B': mode = 'B';
        case '2': baudrate = 1200; break;
        case 'C': mode = 'B';
        case '3': baudrate = 2400; break;
        case 'D': mode = 'B';
        case '4': baudrate = 4800; break;
        case 'E': mode = 'B';
        case '5': baudrate = 9600; break;
        case 'F': mode = 'B';
        case '6': baudrate = 19200; break;
        default: mode = 'A';
    }
    LOG(debug) << "Baudrate: " << baudrate << endl;
    LOG(debug) << "Mode: " << mode << endl;
    if (readableResponse[readableResponse.find("/") + 5] == '\\') {
        char w = readableResponse[readableResponse.find("\\") + 1];
        if (w == '2') {
            LOG(debug) << "Binary mode (HDLC)" << endl;
        } else {
            throw (string("ERROR: No binary mode (HDLC)"));
        }
    }
    string ManufactID = readableResponse.substr(readableResponse.find("/") + 5,
                        readableResponse.find("<CR><LF>") - readableResponse.find("/") - 5);
    LOG(debug) << "ManufactID: " << ManufactID << endl << endl;


    /********************* Sending Acknowledgement/option select Message *********************/

    //Send Acknowledgement/option select message, IEC 62056-21 6.3.3
    u8 message2[6] = {ACK, '2', bautrateChar, '2', CR, LF};

    u8 response2[6] = {0};
    LOG(debug) << "sending2: " << readable(message2, 6) << endl;
    bytesRead = serialdriver->SendData(message2, 6, response2, 300, 0, "");

    //change COMport settings
    serialdriver->ChangeCOM(baudrate, NONE, 8);

    /********************* Reading Acknowledgement/option select Message *********************/

    //Recieve Acknowledgement/option select message, IEC 62056-21 6.3.3
    u8 message3[1] = {0};
    u8 response3[50] = {0};

    bytesRead = serialdriver->SendData(message3, 0, response3, 0, 2000, "\n");
    for (int i = 0; i < bytesRead; i++) {
        response3[i] = response3[i] & 0x7F;
    }
    string readableResponse3 = readable(response3, bytesRead);
    LOG(debug) << "received2: " << readableResponse3 << endl << endl;
    if (readableResponse3.find("<ACK>2") != string::npos && readableResponse3.find("2<CR><LF>")
        != string::npos) {
        LOG(debug) << "Binary mode (HDLC) opened" << endl;
    } else {
        throw (string("ERROR: Could not open binary mode (HDLC)"));
    }
```

```
    /***************************** Sending HDLC parameter frame ******************************/

    //Sending HDLC parameter frame, greenbook 8.4.5
    u8 message4[50];
    int message4Lenght;

    SNMRframeBuilder(message4, message4Lenght, 0x01, 0x11, 0x01, 0xF0, 0xF0, 0x01, 0x01);
    u8 response4[50] = {0};

    LOG(debug) << "sending3: " << endl;
    hexdump(message4, message4Lenght);
    u8 informatie1[100];

    HDLContleed(message4, message4Lenght, informatie1);
    LOG(debug) << endl;

    bytesRead = serialdriver->SendData(message4, message4Lenght, response4, 0, 500, "~");

    /***************************** Reading HDLC parameter frame ******************************/

    //Reading HDLC parameter frame, greenbook 8.4.5
    LOG(debug) << "received3: " << endl;
    hexdump(response4, bytesRead);

    HDLContleed(response4, bytesRead, informatie1);
    LOG(debug) << endl;
    LOG(debug) << endl;

    /********************************* Sending AARQ frame ***********************************/

    //check if keys are set
    if(K != NULL){
        //setting global keys
        for (int i = 0; i < 16; i++) {
            GLOBAL_K[i] = K[i];
            GLOBAL_AAD[i + 1] = AAD[i];
        }
        GLOBAL_AAD[0] = 0x30;

        //Building AARQ frame
        u8 AARQframe[255];
        u8 AAREframe[255];
        int framelenght;
        framelenght = AARQBuilder(AARQframe);

        LOG(debug) << "Sending AARQ-frame" << endl;
        hexdump(AARQframe, framelenght);
        LOG(debug) << endl;

        u8 informatie[100];
        HDLContleed(AARQframe, framelenght, informatie);

        bytesRead = serialdriver->SendData(AARQframe, framelenght, AAREframe, 500, 500, "~");

        /********************************* Reading AARE frame ***********************************/

        lastSendTime = time(0);
        LOG(debug) << endl << "Received AARE-frame " << bytesRead << endl;
        hexdump(AAREframe, bytesRead);
        LOG(debug) << endl;
        int AARElenght = HDLContleed(AAREframe, bytesRead, informatie);

        GLOBAL_RRR++;
        GLOBAL_SSS++;

        //Recieve AARE
        u8 AuthenticationString[64];
        int l = AAREontleed(informatie, AARElenght, AuthenticationString);

        /******************************** Sending Action request ********************************/

        //ActionRequest
        u8 data[17];
        data[0] = 0x10;
        data[1] = (GLOBAL_frameCounter >> 24) & 0xff;
```

```
        data[2] = (GLOBAL_frameCounter >> 16) & 0xff;
        data[3] = (GLOBAL_frameCounter >> 8) & 0xff;
        data[4] = GLOBAL_frameCounter & 0xff;

        cryptoBlock cryptoBlock;

        //authorisation values
        u8 TAG[12];
        u8 specialAAD[128]; // SC || AK || StoC
        specialAAD[0] = 0x10;
        for (int i = 1; i < 17; i++) {
            specialAAD[i] = GLOBAL_AAD[i];
        }
        for (int i = 0; i < l; i++) {
            specialAAD[i + 17] = AuthenticationString[i];
        }

        u8 IV[12];
        for (int i = 0; i < 8; i++) {
            IV[i] = GLOBAL_System_Titel[i];
        }
        IV[8] = (GLOBAL_frameCounter >> 24) & 0xff;
        IV[9] = (GLOBAL_frameCounter >> 16) & 0xff;
        IV[10] = (GLOBAL_frameCounter >> 8) & 0xff;
        IV[11] = GLOBAL_frameCounter & 0xff;

        u8 nullArray[5];

        cryptoBlock.encrypt(nullArray, 0, specialAAD, 17 + l, GLOBAL_K, IV, nullArray, TAG);

        for (int i = 0; i < 12; i++) {
            data[5 + i] = TAG[i];
        }

        u8 Info[128];
        Info[3] = 0xFF;
        status = 1;
        int SL = this->Action(0x0F, 0x0000280000FF, 0x01, DATA_TYPE_OCTET_STRING, data,
                              sizeof (data), Info);

        /****************************** Reading Action response ******************************/

        status = 0;
        if (SL > 0) {
            LOG(debug) << endl << "decrypted Action-response:" << endl;
            hexdump(Info, SL);
            LOG(debug) << endl;
        } else {
            LOG(debug) << endl << "no Action-response:" << endl;
        }

        if (Info[3] == 0x00) {
            LOG(info) << "mode E succesvol opgezet!" << endl;
            status = 1;
            GLOBAL_ANTI_TIMEOUT = true;
            antiLoopThread =  thread(&P0MessageBuilder::antiTimeoutLoop, this);
            return true;
        } else {
            throw (string("Error: no succes"));
        }
    }

}

/* Gets data in mode E */
int P0MessageBuilder::Get(u16 interface_class, u64 logical_name, u8 attribute, u8 dataOut[],
                          bool ignoreConnection) {
    lock_guard<mutex> lg(mtx1);
    LOG(debug) << "kom ik hier? 1" << endl;

    //checking connection
    if (status == 0 && !ignoreConnection) {
        LOG(debug) << "hier in?" << endl;
        throw (string("ERROR: not connected"));
    }

    LOG(debug) << "kom ik hier? 2" << endl;
```

```
    //Building get information field
    u8 information_size = 16;
    u8 message[50];
    message[0] = 0xE6; //LLC bytes
    message[1] = 0xE6;
    message[2] = 0x00;
    message[3] = 0xC0; // GET.request.normal
    message[4] = 0x01;
    message[5] = 0x81; // invoke-id and priority
    message[6] = interface_class >> 8; // interface class
    message[7] = interface_class & 0xff;
    message[8] = (logical_name >> 48) & 0xff; // logical name, OBIS code
    message[9] = (logical_name >> 32) & 0xff;
    message[10] = (logical_name >> 24) & 0xff;
    message[11] = (logical_name >> 16) & 0xff;
    message[12] = (logical_name >> 8) & 0xff;
    message[13] = logical_name & 0xff;
    message[14] = attribute;
    message[15] = 0x00;

    LOG(debug) << "kom ik hier? 3" << endl;
    //secure connection
    if (status = 1) {
        LOG(debug) << "kom ik hier? 4" << endl;
        //Build encryption frame
        u8 increptedString[100];
        int lenght = 0;
        message[5] = 0xc1; // change invoke-id and priority
        lenght = encryptedStringBuilder(message + 3, 13, increptedString);
        message[3] = 0xc8;
        message[4] = lenght; //lengte correct?
        for (int i = 0; i < lenght; i++) {
            message[5 + i] = increptedString[i];
        }
        information_size = 5 + lenght;
    }

    //Building HDLC frame around get information
    u8 frame[50];
    int framelenght = HDLCbuilder(GLOBAL_upperServerAddress, GLOBAL_lowerServerAddress,
                                  GLOBAL_ClientAdress, FRAME_TYPE_I, GLOBAL_RRR, GLOBAL_SSS, 1,
                                  message, information_size, frame);

    LOG(debug) << "sending Get-request" << endl;
    hexdump(frame, framelenght);
    LOG(debug) << endl;

    u8 information[128];
    HDLContleed(frame, framelenght, information);
    LOG(debug) << endl;

    //sending get
    u8 response[128];
    int responseSize = serialdriver->SendData(frame, framelenght, response, 200, 2000, "");
    lastSendTime = time(0);
    GLOBAL_RRR++;
    GLOBAL_SSS++;

    LOG(debug) << "reading Get-response" << endl;
    hexdump(response, responseSize);
    LOG(debug) << endl;

    //dissecting data from HDLC frame
    int informationSize = HDLContleed(response, responseSize, information);
    if (informationSize == 0) {
        throw (string("Error: Empty response. no permissions?"));
    }

    if(status == 1){
        informationSize = stringdecrypter(information, informationSize, dataOut);
        LOG(debug) << "decrypted Get-response information" << endl;
        hexdump(dataOut, informationSize);
        LOG(debug) << endl;
    }else{
        LOG(debug) << "Get-response information" << endl;
```

```
        hexdump(response, informationSize);
        LOG(debug) << endl;
    }
    return informationSize;
}

/* Sets data in mode E */
int P0MessageBuilder::Set(u16 interface_class, u64 logical_name, u8 attribute, u8 datatype,
                          u8 dataIn[], int dataInSize, u8 dataOut[]) {
    lock_guard<mutex> lg(mtx1);

    //checking connection
    if (status == 0) {
        throw (string("ERROR: not connected"));
    }
    //checking data size
    if(dataInSize > 100){
        throw (string("ERROR: Set data to large"));
    }

    //Building set information
    u8 message[128];
    u8 message_size = 18 + dataInSize;
    message[0] = 0xE6; //LLC bytes
    message[1] = 0xE6;
    message[2] = 0x00;
    message[3] = 0xC1; // GET.request.normal
    message[4] = 0x01;
    message[5] = 0x81; // invoke-id and priority
    message[6] = interface_class >> 8; // interface class
    message[7] = interface_class & 0xff;
    message[8] = (logical_name >> 48) & 0xff; // logical name, OBIS code
    message[9] = (logical_name >> 32) & 0xff;
    message[10] = (logical_name >> 24) & 0xff;
    message[11] = (logical_name >> 16) & 0xff;
    message[12] = (logical_name >> 8) & 0xff;
    message[13] = logical_name & 0xff;
    message[14] = attribute;
    message[15] = 0x00; //SelectiveAccessDescriptor
    message[16] = 0x09; //dataIn type byte string
    message[17] = dataInSize;
    for (int i = 0; i < dataInSize; i++) {
        message[18 + i] = dataIn[i];
    }

    //if secure connection encrypt
    if (status == 1) {
        u8 increptedString[100];
        int lenght = 0;
        message[5] = 0xc1; // change invoke-id and priority

        LOG(debug) << " Set-request information" << endl;
        hexdump(message + 3, message_size - 3);
        LOG(debug) << endl;

        lenght = encryptedStringBuilder(message + 3, message_size - 3, increptedString);
        message[3] = 0xc9;
        message[4] = lenght;
        for (int i = 0; i < lenght; i++) {
            message[5 + i] = increptedString[i];
        }
        message_size = 5 + lenght;
    }

    //Building HDLC frame around get information
    u8 frame[50];
    int framelenght = HDLCbuilder(GLOBAL_upperServerAddress, GLOBAL_lowerServerAddress,
                                  GLOBAL_ClientAdress, FRAME_TYPE_I, GLOBAL_RRR, GLOBAL_SSS, 1,
                                  message, message_size, frame);

    LOG(debug) << "sending Set-request" << endl;
    hexdump(frame, framelenght);
    LOG(debug) << endl;

    u8 information[128];
    HDLContleed(frame, framelenght, information);
    LOG(debug) << endl;
```

```
    //sending get request
    u8 response[128];
    int responseSize = serialdriver->SendData(frame, framelenght, response, 200, 2000, "");
    lastSendTime = time(0);
    GLOBAL_RRR++;
    GLOBAL_SSS++;

    LOG(debug) << "reading Set-response" << endl;
    hexdump(response, responseSize);
    LOG(debug) << endl;

    //dissecting data from HDLC frame
    int informationSize = HDLContleed(response, responseSize, dataOut);
    if (informationSize == 0) {
        throw (string("Error: Empty response. no permissions?"));
    }

    if(status == 1){
        int informationSize = stringdecrypter(dataOut, informationSize, dataOut);
        LOG(debug) << "decrypted Set-response information" << endl;
        hexdump(dataOut, informationSize);
        LOG(debug) << endl;
    }else{
        LOG(debug) << " Set-response information" << endl;
        hexdump(dataOut, informationSize);
        LOG(debug) << endl;
    }

    if (dataOut[4] != 0) {
        throw (string("Error: no succes"));
    }
    return informationSize;
}

/* Does an Action on data in mode E */
int P0MessageBuilder::Action(u16 interface_class, u64 logical_name, u8 method, u8 datatype,
                             u8 dataIn[], int dataInSize, u8 dataOut[]) {
    lock_guard<mutex> lg(mtx1);

    //checking connection
    if (status == 0) {
        throw (string("ERROR: not connected"));
    }
    //checking data size
    if(dataInSize > 100){
        throw (string("ERROR: Set data to large"));
    }

    //Building action information field
    u8 message[128];
    u8 messageSize = 18 + dataInSize;
    message[0] = 0xE6; //LLC bytes
    message[1] = 0xE6;
    message[2] = 0x00;
    message[3] = 0xC3; // ACTION.request.normal
    message[4] = 0x01;
    message[5] = 0x81; // invoke-id and priority
    message[6] = interface_class >> 8; // interface class
    message[7] = interface_class & 0xff;
    message[8] = (logical_name >> 48) & 0xff; // logical name, OBIS code
    message[9] = (logical_name >> 32) & 0xff;
    message[10] = (logical_name >> 24) & 0xff;
    message[11] = (logical_name >> 16) & 0xff;
    message[12] = (logical_name >> 8) & 0xff;
    message[13] = logical_name & 0xff;
    message[14] = method;
    message[15] = 0x00; //? why the 1?
    message[16] = datatype; //value type byte string
    message[17] = dataInSize;
    for (int i = 0; i < dataInSize; i++) {
        message[18 + i] = dataIn[i];
    }

    //if secure connection encrypt
    if (status == 1) {

        u8 increptedString[100];
        int lenght = 0;
```

```
        message[5] = 0xc1; // change invoke-id and priority

        LOG(debug) << "Action-request information" << endl;
        //hexdump(message + 3, messageSize - 3);
        LOG(debug) << endl;
        lenght = encryptedStringBuilder(message + 3, messageSize - 3, increptedString);
        message[3] = 0xcb;
        message[4] = lenght;
        for (int i = 0; i < lenght; i++) {
            message[5 + i] = increptedString[i];
        }
        messageSize = 5 + lenght;
    }

    //Building HDLC frame around action information
    u8 frame[255];
    u8 responseFrame[255];
    int responseSize = 0;
    int framelenght = HDLCbuilder(GLOBAL_upperServerAddress, GLOBAL_lowerServerAddress,
                                  GLOBAL_ClientAdress, FRAME_TYPE_I, GLOBAL_RRR, GLOBAL_SSS, 1,
                                  message, messageSize, frame);

    LOG(debug) << "Sending Action-request" << endl;
    hexdump(frame, framelenght);
    LOG(debug) << endl;

    u8 information[128];
    HDLContleed(frame, framelenght, information);
    LOG(debug) << endl;

    //Sending action request
    responseSize = serialdriver->SendData(frame, framelenght, responseFrame, 500, 2000, "");
    lastSendTime = time(0);
    GLOBAL_RRR++;
    GLOBAL_SSS++;

    LOG(debug) << "Reading Action-response" << endl;
    hexdump(responseFrame, responseSize);
    LOG(debug) << endl;

    int informationSize = HDLContleed(responseFrame, responseSize, dataOut);

    if(status == 1){
        informationSize = stringdecrypter(dataOut, informationSize, dataOut);
        LOG(debug) << "decrypted Action-response information" << endl;
        hexdump(dataOut, informationSize);
        LOG(debug) << endl;
    }else{
        LOG(debug) << " Action-response information" << endl;
        hexdump(information, informationSize);
        LOG(debug) << endl;
    }

    if (dataOut[3] != 0) {
        throw (string("Error: no succes"));
    }
    return informationSize;
}

/* Sends raw byte's and returns raw answer */
int P0MessageBuilder::SendRaw(u8 bufferIn[], int bufferInLenght, u8 bufferOut[], int readDelay,
                              int timeout, string find) {
    if(serialdriver == NULL){
        throw(string("Error: COM port not opend yet"));
    }
    int bufferOutLenght = 0;
    static int teller = 1;

    LOG(debug) << "sendingRaw" << teller << ": " << endl;
    hexdump(bufferIn, bufferInLenght);
    LOG(debug) << endl;

    //sending/reading data
    bufferOutLenght = serialdriver->SendData(bufferIn, bufferInLenght, bufferOut, readDelay,
                                             timeout, find);

    LOG(debug) << "receivedRaw" << teller << ": " << endl;
    hexdump(bufferOut, bufferOutLenght);
```

```
    LOG(debug) << endl;

    teller++;
    return bufferOutLenght;
}

/* Opens the COM port */
bool P0MessageBuilder::OpenCOM(string commPortName, int bitRate, int prarity, int byte) {
    lock_guard<mutex> lg(mtx1);

    //checking connection
    if (status == 0) {
        throw (string("ERROR: already connected"));
    }

    //Opening commPort
    LOG(debug) << "Opening " << commPortName << endl;
    serialdriver = new SerialDriver(commPortName, bitRate, prarity, byte);
    ComPortName = commPortName;
    serialdriver->flush();
    status = 2;
    return true;
}

/* Closes mode E */
void P0MessageBuilder::CloseModeE() {


    //if(antiLoopThread != NULL){
        mtx1.lock();
        GLOBAL_ANTI_TIMEOUT = false;
        mtx1.unlock();

        antiLoopThread.join();
    //}

    //check status
    if (status == 0) {
        throw (string("ERROR: not connected"));
    }

    //if secure close secure connection
    if(status == 1){
        //Building RLRQ information
        u8 RLRQ[50];
        int RLRQSize = 14;
        RLRQ[0] = 0xe6; //LLC Bytes
        RLRQ[1] = 0xe6;
        RLRQ[2] = 0x00;
        RLRQ[3] = 0x62;
        RLRQ[4] = 0x28;
        RLRQ[5] = 0x80;
        RLRQ[6] = 0x01;
        RLRQ[7] = 0x00;
        RLRQ[8] = 0xbe;
        RLRQ[9] = 0x23;
        RLRQ[10] = 0x04;
        RLRQ[11] = 0x21;
        RLRQ[12] = 0x21;
        RLRQ[13] = 0x1f;

        u8 AA_AInfo[14] ={
            0x01, 0x00, 0x00, 0x00, 0x06, 0x5F, 0x1F, 0x04, 0x00, 0x00, 0x1F, 0x1F, 0x08, 0x00,
        };

        u8 encryptedString[50];
        int r = encryptedStringBuilder(AA_AInfo, 14, encryptedString);

        for (int i = 0; i < r; i++) {
            RLRQ[12 + i] = encryptedString[i];
        }
        RLRQSize += r;

        LOG(debug) << "RLRQ" << endl;
        hexdump(RLRQ, RLRQSize);
        LOG(debug) << endl;
```

```
        //Building HDLC frame around RLRQ information
        u8 frame[128];
        int framelenght = HDLCbuilder(GLOBAL_upperServerAddress, GLOBAL_lowerServerAddress,
                                      GLOBAL_ClientAdress, FRAME_TYPE_I, GLOBAL_RRR, GLOBAL_SSS,
                                      1, RLRQ, RLRQSize, frame);

        u8 informatie[50];
        LOG(debug) << "sending RLRQ" << endl;
        HDLContleed(frame, framelenght, informatie);
        LOG(debug) << endl;

        //sending RLRQ frame and reading RLRE frame
        u8 response[128];
        int responseSize = serialdriver->SendData(frame, framelenght, response, 200, 2000, "");

        GLOBAL_RRR++;
        GLOBAL_SSS++;

        LOG(debug) << "reading RLRE" << endl;
        hexdump(response, responseSize);
        LOG(debug) << endl;
    }

    //Bulding empty disc frame
    u8 frame[128];
    int framelenght = HDLCbuilder(GLOBAL_upperServerAddress, GLOBAL_lowerServerAddress,
                                  GLOBAL_ClientAdress, FRAME_TYPE_DISC, GLOBAL_RRR, GLOBAL_SSS,
                                  1, frame, 0, frame);

    //Sending empty disc frame
    u8 response[128];
    int responseSize = serialdriver->SendData(frame, framelenght, response, 200, 1000, "");
    status = 0;
    LOG(debug) << "last response?" << endl;
    hexdump(response, responseSize);
    LOG(debug) << endl;

    delete &serialdriver;
    serialdriver = NULL;
}

/* Closes COM port */
void P0MessageBuilder::CloseCOM(){
    //stop anti timeout loop?
    if(serialdriver == NULL){
        throw(string("Error: COM port already closed"));
    }
    delete &serialdriver;
    serialdriver = NULL;
}

/* Returns the name */
string P0MessageBuilder::GetName() {
    return Name;
}

/* Returns the COM port */
string P0MessageBuilder::GetComm() {
    return ComPortName;
}
```

# Bijlage H: SerialDriver.h

```
//============================================================================================================
// Name        : SerialDriver.h
// Author      : Niels Hokke
// Company     : Sogeti
// Description : Header File of SerialDriver.cpp.
//============================================================================================================

#ifndef   SERIALDRIVER H
#define __SERIALDRIVER_H__

//COM port parity settings
#define EVEN 1
#define ODD  2
#define NONE 3

typedef unsigned char u8;

class SerialDriver {
private:

    /* Writes to COM port */
    int writeCOM(const void *buffer, int buffLen);

    /* Reads from COM port */
    int readCOM(u8 *buffer, int buffLen, bool nullTerminate = true);

    /* Closes COM port */
    void closeCOM();

    /* COM port file handle */
    int USB;

public:

    /* Object constructor, opens COM port */
    SerialDriver(std::string COMPortName, int bitRate = 300, int prarity = 1, int byte = 7);

    /* Object destructor. Closes connection */
    ~SerialDriver();

    /* Changes COM port settings */
    void ChangeCOM(int bitRate = 9600, int prarity = 1, int byte = 8);

    /* Sends Data and returns the answer */
    int SendData(const void *bufferIn, int bufferInSize, void *bufferOut, int readDelay = 200,
                 int timeout = 1000, std::string endChars = "");

    /* Flushes COM port */
    void flush();
};

#endif
```

# Bijlage I: SerialDriver.cpp

```
//============================================================================================================
// Name        : SerialDriver.cpp
// Author      : Niels Hokke
// Company     : Sogeti
// Description : Serial Driver object which handels opening, setting, sending, receiving and
//               closing of a COM port
//============================================================================================================


#include <iostream>
#include "SerialDriver.h"
#include <stdio.h>      // standard input / output functions
#include <stdlib.h>
#include <string.h>     // string function definitions
#include <unistd.h>     // UNIX standard function definitions
#include <fcntl.h>      // File control definitions
#include <errno.h>      // Error number definitions
#include <termios.h>    // POSIX terminal control definitions
#include <sstream>

#define FLUSH_BUFFSIZE 10

using namespace std;

/*****************************************************************************************
* Private Functions                                                                      *
*****************************************************************************************/

/* Writes to COM port */
int SerialDriver::writeCOM(const void *buffer, int buffLen)
{
    //Writing data to COM port
    int bytesRead = write( USB, buffer, buffLen);

    if ( bytesRead == -1 ) {
        //Throwing error if -1 bytes written
        stringstream tempss;
        tempss << "Error" << errno << " writeCOM: " << strerror(errno) << endl;
        throw(tempss.str());
    }

    return bytesRead;
}

/* Reads from COM port */
int SerialDriver::readCOM(u8 *buffer, int buffLen, bool nullTerminate)
{
    if(nullTerminate) --buffLen;

    //Reading data from COM port
    int bytesRead = read( USB, buffer, buffLen );
    //cout << "bytesRead: " <<  bytesRead << endl;
    if ( bytesRead == -1 ) {
        //Throwing error if -1 bytes read
        stringstream tempss;
        tempss << "Error" << errno << " readCOM: " << strerror(errno) << endl;
        throw(tempss.str());
    }

    if(bytesRead < 1) return bytesRead;

    //adding null termination to end string
    if(nullTerminate) buffer[bytesRead] = '\0';

    return bytesRead;
}

/* Closes COM port */
void SerialDriver::closeCOM()
{
    close(USB);
}
```

```
/*****************************************************************************************
* Public Functions                                                                       *
*****************************************************************************************/

/* Object constructor, opens COM port */
SerialDriver::SerialDriver(string COMPortName, int bitRate, int prarity, int byte)
{
    //See https://www.cmrr.umn.edu/~strupp/serial.html for more about COM port settings

    //Opening COM port
    USB = open(COMPortName.c_str(), O_RDWR | O_NOCTTY | O_NDELAY);

    if (USB < 0) {
        //Throwing error
        stringstream tempss;
        tempss << "Error while opening " << COMPortName << " errno = " << errno << endl;
        throw(tempss.str());
    }

    fcntl(USB, F_SETFL, FNDELAY); // Open the COM port in nonblocking mode

    struct termios options;

    //Connecting options struct to COM port
    if ( tcgetattr ( USB, &options ) != 0 ) {
        stringstream tempss;
        tempss << "Error " << errno << " from tcgetattr: " << strerror(errno) << endl;
        throw(tempss.str());
    }

    //Clearing old settings
    bzero(&options, sizeof(options));

    //Selecting baudrate
    speed_t baudrate;
    switch(bitRate){
        case 50 :       baudrate = B50;     break;
        case 75 :       baudrate = B75;     break;
        case 110 :      baudrate = B110;    break;
        case 134 :      baudrate = B134;    break;
        case 150 :      baudrate = B150;    break;
        case 200 :      baudrate = B200;    break;
        case 300 :      baudrate = B300;    break;
        case 600 :      baudrate = B600;    break;
        case 1200 :     baudrate = B1200;   break;
        case 1800 :     baudrate = B1800;   break;
        case 2400 :     baudrate = B2400;   break;
        case 4800 :     baudrate = B4800;   break;
        case 9600 :     baudrate = B9600;   break;
        case 19200 :    baudrate = B19200;  break;
        case 38400 :    baudrate = B38400;  break;
        case 57600 :    baudrate = B57600;  break;
        case 115200 :   baudrate = B115200; break;
        default :  throw(string("ERROR: baudrate not supported"));
    }

    //Setting baudrate
    cfsetospeed (&options, baudrate);
    cfsetispeed (&options, baudrate);

    //Selecting and setting parity
    switch(prarity){
        case EVEN  :
            options.c_cflag |= PARENB; //Enable parity bit
            options.c_cflag &= ~PARODD; //Use odd parity instead of even
            break;
        case ODD :
            options.c_cflag |= PARENB; //Enable parity bit
            options.c_cflag |= PARODD; //Use odd parity instead of even
            break;
        case NONE  :
            options.c_cflag &= ~PARENB; //Disable parity bit
            break;
        default : //Optional
           throw(string("ERROR: Serial, parity not supported"));
```

```
    }


    options.c_cflag |= ( CLOCAL | CREAD);//Enable reading and disable CSV check
    options.c_cflag     &=  ~CSTOPB;     //1 stop bit
    options.c_cflag     &=  ~CSIZE;      //No bit mask for data bits

    //Selecting and setting number of data bits
    switch(byte){
        case 5  : options.c_cflag |= CS5; break;
        case 6  : options.c_cflag |= CS6; break;
        case 7  : options.c_cflag |= CS7; break;
        case 8  : options.c_cflag |= CS8; break;
        default : throw(string("ERROR: Serial byte size not supported"));
    }

    options.c_iflag |= ( IGNPAR | IGNBRK );
    options.c_cc[VTIME]=0;      // Timer unused
    options.c_cc[VMIN]=0;       // At least 0 characters on read

    // Flush Port, then apply settings
    tcflush( USB, TCIFLUSH );
    if ( tcsetattr ( USB, TCSANOW, &options ) != 0) {
        stringstream tempss;
        tempss << "Error " << errno << " from tcsetattr" << endl;
        throw(tempss.str());
    }
}

/* Object destructor. Closes connection */
SerialDriver::~SerialDriver()
{
    close(USB);
}

void SerialDriver::ChangeCOM(int bitRate, int prarity, int byte)
{
    struct termios options;

    //Connecting options struct to COM port
    if ( tcgetattr ( USB, &options ) != 0 ) {
        stringstream tempss;
        tempss << "Error " << errno << " from tcgetattr: " << strerror(errno) << endl;
        throw(tempss.str());
    }

    //Clearing old settings
    bzero(&options, sizeof(options));

    //Selecting baudrate
    speed_t baudrate;
    switch(bitRate){
        case 50 :       baudrate = B50;     break;
        case 75 :       baudrate = B75;     break;
        case 110 :      baudrate = B110;    break;
        case 134 :      baudrate = B134;    break;
        case 150 :      baudrate = B150;    break;
        case 200 :      baudrate = B200;    break;
        case 300 :      baudrate = B300;    break;
        case 600 :      baudrate = B600;    break;
        case 1200 :     baudrate = B1200;   break;
        case 1800 :     baudrate = B1800;   break;
        case 2400 :     baudrate = B2400;   break;
        case 4800 :     baudrate = B4800;   break;
        case 9600 :     baudrate = B9600;   break;
        case 19200 :    baudrate = B19200;  break;
        case 38400 :    baudrate = B38400;  break;
        case 57600 :    baudrate = B57600;  break;
        case 115200 :   baudrate = B115200; break;
        default :  throw(string("ERROR: baudrate not supported"));
    }

    //Setting baudrate
    cfsetospeed (&options, baudrate);
    cfsetispeed (&options, baudrate);

    //Selecting and setting parity
    switch(prarity){
```

```
        case EVEN   :
            options.c_cflag |= PARENB; //Enable parity bit
            options.c_cflag &= ~PARODD; //Use odd parity instead of even
            break;
        case ODD :
            options.c_cflag |= PARENB; //Enable parity bit
            options.c_cflag |= PARODD; //Use odd parity instead of even
            break;
        case NONE   :
            options.c_cflag &= ~PARENB; //Disable parity bit
            break;
        default : //Optional
           throw(string("ERROR: Serial, parity not supported"));
    }

    options.c_cflag |= ( CLOCAL | CREAD);//Enable reading and disable CSV check
    options.c_cflag     &=  ~CSTOPB;     //1 stop bit
    options.c_cflag     &=  ~CSIZE;      //No bit mask for data bits

    //Selecting and setting number of data bits
    switch(byte){
        case 5  : options.c_cflag |= CS5; break;
        case 6  : options.c_cflag |= CS6; break;
        case 7  : options.c_cflag |= CS7; break;
        case 8  : options.c_cflag |= CS8; break;
        default : throw(string("ERROR: Serial byte size not supported"));
    }

    options.c_iflag |= ( IGNPAR | IGNBRK );
    options.c_cc[VTIME]=0;      // Timer unused
    options.c_cc[VMIN]=0;       // At least 0 characters on read

    // Flush Port, then apply settings
    tcflush( USB, TCIFLUSH );
    if ( tcsetattr ( USB, TCSANOW, &options ) != 0) {
        stringstream tempss;
        tempss << "Error " << errno << " from tcsetattr" << endl;
        throw(tempss.str());
    }
}

/* Sends Data and returns the answer */
int SerialDriver::SendData(const void *bufferIn, int bufferInSize, void *bufferOut,
                           int readDelay, int timeout, string endChars){

    //Writing data to COM port
    int bytesWriten = writeCOM(bufferIn, bufferInSize);

    if(bytesWriten != bufferInSize){
        throw(string("ERROR: Writing to the serial port timed out"));
        return 0;
    }

    //delay between sending and reading
    usleep(readDelay*1000);

    int timeoutCounter = 0;
    int bufferOutCounter = 0;
    int timeoutDelay = 100000;
    int bufferOutSize = 0;
    string message = "";

    //Converting ms to us
    timeout *= 1000;

    //try reading data until endChars are found or time-out
    while(timeoutCounter < timeout && (message.find(endChars,7) == std::string::npos ||
          endChars == "")){
        //reading data from COM port
        u8 buffer[50];
        int charsRead = readCOM(buffer, 50);

        //Building Message to find endChars
        message += (char*)buffer;
        bufferOutSize += charsRead;

        //Adding read bytes to output buffer
        for(int i=0; i < charsRead; i++){
```

```
            ((char*) bufferOut)[bufferOutCounter + i] = buffer[i];
        }
        bufferOutCounter += charsRead;

        //Counting for timeout when no bytes read
        if(charsRead < 1){
                timeoutCounter += timeoutDelay;
        }else{
                timeoutCounter = 0;
        }

        //delay between each read
        usleep(timeoutDelay);
    }
    return bufferOutSize;
}

/* Flushes COM port */
void SerialDriver::flush()
{
    u8 buffer[FLUSH_BUFFSIZE];

    //Read data from COM port
    int numBytes = readCOM(buffer, FLUSH_BUFFSIZE, false);

    //Keep reading until serial buffer is empty
    while(numBytes > 0)
    {
        numBytes = readCOM(buffer, FLUSH_BUFFSIZE, false);
        usleep(1000);
    }
}
```