Discman PDA

Udvikling af PDA software til kontrol af DISCOS overvågningssystemer

Bachelorprojekt i IT

Glen Lauridsen – s032261

Vejleder: Bjarne Poulsen

21. Maj 2007

Institut for Informatik og Matematisk Modellering.

Danmark Tekniske Universitet.

Indhold

1. FORORD 5

2. INTRODUKTION 6

2.1. PROJEKT SPECIFIKATION 6
2.1.1. Baggrund for projektet 6
2.1.2. Vision 7
2.1.3. Kravspecifikation 7
2.1.4. Risikoanalyse 9
2.1.5. Afgrænsninger 10
2.2. UDVIKLINGSPROCES 11
2.2.1. Unified Process 11
2.2.2. Iterative processer 13
2.3. TIDSPLAN 14
2.4. OVERSIGT 16

3. ANALYSE 17

3.1. TEKNOLOGI 17
3.1.1. DISCOS System 17
3.1.2. Måling og kontrol 18
3.2. PROBLEMSTILLING 20
3.2.1. System Arkitektur 20
3.2.2. Kommunikation 22
3.2.3. Sikkerhed 23
3.2.4. Protokol 24
3.2.5. Brugergrænseflade 26
3.2.6. Use Cases 32
3.3. DELKONKLUSION 35

4. DESIGN 36

4.1. SYSTEMARKITEKTUR 36
4.1.1. Tråde 37
4.2. KOMMUNIKATION 38
4.2.1. Bluetooth 38
4.3. SIKKERHED 40
4.3.1. Symmetrisk kryptering 40
4.3.2. Algoritmer Understøttet af .NET Framework'et 41
4.3.3. Nøgle administrering 42
4.4. PROTOKOL 43
4.4.1. Netværks Initialisering 43
4.4.2. Discman PDA 44
4.4.3. DISCOS kommunikations emulator 47
4.5. DEVICE 50
4.5.1. UnitFactory 51
4.5.2. Unit 51
4.5.3. Master 51
4.5.4. Opti 51

4.5.5. EEPROMController *51*
4.5.6. EEPROMMetaMap *51*
4.5.7. EEPROMMetaEntry *51*
4.6. APP 52
4.6.1. APPProxy *53*
4.6.2. LogicController *53*
4.6.3. DeviceController *53*
4.7. DATA 54
4.7.1. DataController *55*
4.7.2. XMLWriter *55*
4.7.3. DataContainer *55*
4.7.4. VoiceLog *55*
4.7.5. WaveIn *55*
4.7.6. WaveOut *55*
4.7.7. WaveNative *55*
4.8. GUI 56
4.8.1. Mediator *56*
4.8.2. Forms *57*
4.8.3. GUI Map *60*
4.9. USE CASE REALISERING 61
§ 0001 – Opret forbindelse *61*
§ 0004 – Søg efter DISCOS enheder *62*
4.10. DELKONKLUSION 63

5. IMPLEMENTERING 64

5.1. KOMMUNIKATION 64
5.1.1. Søgning efter Bluetooth enheder *64*
5.1.2. Oprettelse af forbindelse til en Bluetooth enhed *67*
5.2. SIKKERHED 68
5.2.1. Kryptering *68*
5.3. XML SERIALIZATION 68
5.3.1. Indlæs EEPROM Meta Map *68*
5.3.2. Gem målinger *70*
5.4. AUDIO 72
5.4.1. Optagning af voice log *73*
5.4.2. Afspilning af voice log *74*
5.5. DELKONKLUSION 75

6. TEST 75

6.1. KRYPTERINGSHASTIGHED: TDES VS AES 75
6.2. USE CASE TEST 76
6.2.1. Use case § 0001 – Opret forbindelse *76*
6.2.2. Use case § 0004 – Søg efter DISCOS enheder *77*
6.3. FUNKTIONSTEST 78
6.4. DELKONKLUSION 78

7. EVALUERING OG KONKLUSION 79

7.1. PROJEKTFORLØB 79

7.2. Udviklingsproces 79
7.3. Forbedringer 79
7.4. Fremtidsvision 80
7.5. Konklusion 80

8. REFERENCER 81

8.1. Bøger 81
8.2. Hjemmesider 81

9. BILAG .....

9.1. Use Cases .....
9.2. Use Case Realisering .....
9.3. Use Case Test .....
9.4. Protokol .....
9.5. EEPROM Memory map .....
9.6. CD Struktur .....

Figur liste

Figur 1 - Unified Process projektforløb. 11
Figur 2 - Illustration over det relative tidsforbrug for de enkelte udviklings faser. 12
Figur 3 - Diagram der illustrerer hvordan betydningen af de forskellige desipliner skifter gennem projektforløbet. 12
Figur 4 - Illustration af iterations cyklus. 13
Figur 5 - Unified Process projektforløb med iterations overblik. 13
Figur 6 - Illustration af Faraday effekt princippet 18
Figur 7 - System Arkitektur 20
Figur 8 - Specifikation af Bluetooth ramme format 24
Figur 9 - Splash vindue 26
Figur 10 - Connection Setup vindue 26
Figur 11 - Visual Connection vindue 27
Figur 12 - Settings vindue 27
Figur 13 - System Settings vindue 28
Figur 14 - Master Measurement vindue 31
Figur 15 - Opti Measurement vindue 31
Figur 16 - Use Case diagram 32
Figur 17 - System Sekvensdiagram: Opret forbindelse 33
Figur 18 - System Sekvensdiagram: Søg efter DISCOS enheder 34
Figur 19 - System arkitektur 36
Figur 20 – Klassediagram: Kommunikationslaget. 38
Figur 21 - Stream kryptering 40
Figur 22 - Blok kryptering 40
Figur 23 - Initialiseringssekvens 43
Figur 24 - Klassediagram: Protokollaget (Discman PDA) 44
Figur 25 - Parser tilstands diagram 45
Figur 26 - Klassediagram: Protokollaget (DISCOS kommunikations emulator) 47
Figur 27 - Parser tilstands diagram 48
Figur 28 - Klassediagram: Enhedslaget. 50

Figur 29 - Klassediagram: Applikationslaget. 52
Figur 30 - Klassediagram: Datalaget 54
Figur 31 - Klassediagram: GUIlaget 56
Figur 32 - Screenshot: Splash vinduet 57
Figur 33 - Screenshot: Connection Setup 57
Figur 34 - Screenshot: Audio Center 58
Figur 35 - Screenshot: Audio Center 58
Figur 36 - Screenshot: Settings 58
Figur 37 - Screenshot: System Settings 59
Figur 38 - Screenshot: Master Measurements 59
Figur 39 - Screenshot: Opti Measurements 59
Figur 40 - GUI Map 60
Figur 41 - Sekvensdiagram: Opret forbindelse 61
Figur 42 - Sekvensdiagram: Søg efter DISCOS enheder 62
Figur 43 - Flowchart over Bluetooth søgningen 66
Figur 44 - Adresserings format for Bluetooth socket adressering 67
Figur 45 - Flowchart over forbindelses proceduren. 67
Figur 46 – Screenshot: TDES vs AES 76

Tabel liste

Tabel 1 - Risiko vurdering. 9
Tabel 2 - Tidsplan 15
Tabel 3 - Milepæle 15
Tabel 4 - Use Case oversigt 32
Tabel 5 - Use Case §0001: Opret forbindelse 33
Tabel 6 - Use Case §0004: Søg efter DISCOS enheder 34
Tabel 7 - Sammenligning mellem Stream og Block kryptering 41

1. Forord

Denne rapport er udarbejdet af Glen Lauridsen i forbindelse med bachelorprojektet på Diplom IT retningen, ved Instituttet for Informatik og Matematisk Modellering på Danmarks Tekniske Universitet. Rapporten er udarbejdet ud fra de på instituttet tillærte færdigheder inden for IT.

Projektet er blevet gennemført i samarbejde med virksomhederne PowerSense A/S, der har stillet lokaler og udstyr til rådighed gennem projektperioden og Kasmatic Innovation A/S der har bidraget med feedback i forhold til de tekniske elementer i projektet. Kontakten mellem Kasmatic Innovation A/S og Glen Lauridsen blev oprettet i 4. semester af uddannelsen, da Glen Lauridsen blev ansat som studentermedhjælper. Kontakten mellem PowerSense A/S og Glen Lauridsen er opstået i forbindelse med tidligere projekter, der er blevet gennemført i sammenarbejde mellem Kasmatic Innovation A/S og PowerSense A/S.

Rapporten beskriver implementeringen af en prototype til en PDA applikation, der skal benyttes til kontrol af DISCOS overvågningssystemer. Applikationen baserer sig på den trådløse kommunikationsstandard Bluetooth, hvilket introducerer nogle sikkerheds kritiske elementer der skal varetages. Rapporten dækker udviklingen fra analyse fasen, frem til implementering og test af applikationen.

Udviklingen af et sådant system bunder i en kommende kommunikationsplatform til DISCOS systemet, der vil komme til at understøtte en trådløs kommunikationsstandard. Dette skal åbne op og illustrerer mulighederne indenfor den nye kommunikationsplatform samtidig med at systemet skal udgøre et brugervenligt redskab til vedligeholdelse af de enkelte DISCOS systemer.

2. Introduktion

Formålet med dette kapitel er at give en introduktion til projektet.

Desuden vil den udviklingsprocedure, der benyttes blive belyst, med henblik på at give en bedre forståelse for den valgte tilgang til projektet.

2.1. Projekt specifikation

De følgende afsnit vil være med til at opstille nogle rammer for, hvordan projektet skal udformes. Disse vil blive baseret på virksomhedens (PowerSense) intentioner med projektet samt, hvordan disse skal opnås. Dette vil endvidere være med til at belyse de enkelte krav til projektet og være med til at udpege, hvilke risici projektet medfører.

2.1.1. Baggrund for projektet

PowerSense er en virksomhed der udvikler og producerer høj kvalitets overvågnings- og kontrolsystemer til forsyningsindustrien. PowerSense DISCOS systemet er et integreret overvågnings- og kontrolsystem til strømfordelingsnetværker. Systemet giver mulighed for fjernkontrol og fjernmonitorering af eksisterende mellemspændings og lavspændings strømfordelingsnetværk.

Ved konfigurering og vedligeholdelse af DISCOS systemer benytter PowerSense sig af en software applikation kaldet Discman Laptop. Denne er specifikt udviklet til en bærbar PC til brug som et konfigureringsredskab til DISCOS systemet. Dette har gjort at funktionaliteten af applikationen er blevet prioriteret højere end selve brugervenligheden. Det kræver dermed en vis erfaring at betjene dette produkt, hvilket betyder at det kun er de specifikke teknikere, der administrerer DISCOS systemerne, der kan betjene programmet. Disse har som regel en ingeniørmæssig baggrund, der gør dem i stand til at sætte sig ind i programmets opbygning og de deri anvendte teknikker.

For at forenkle vedligeholdelsen af DISCOS systemerne er der derfor behov for en light version af dette produkt, der i stedet lanceres til den daglige medarbejder. Produktet skal fokusere på brugervenlighed, så det kan håndteres af medarbejdere, der ikke har den store indsigt i systemets anvendte teknikker. Produktet skal benyttes som det almene redskab til vedligeholdelsen af DISCOS systemet, og vil derfor have en væsentlig begrænsning i funktionaliteten i forhold til det eksisterende produkt. Dermed vil Discman Laptop udelukkende blive benyttet af administratorer ved opsætning og konfigurering af DISCOS systemet, mens det nye produkt vil blive benyttet af den daglige medarbejder, der skal foretage rutine undersøgelser af systemet.

PowerSense lancerer i en nær fremtid en ny kommunikationsplatform til DISCOS systemet, der kommer til at understøtte nogle forskellige kommunikationsstandarder, herunder en trådløs standard. Den kommende kommunikationsplatform vil introducere nogle nye muligheder for udvikling af teknologiske løsninger/produkter til DISCOS systemet, som vil gøre DISCOS systemet mere alsidig og være med til at øge funktionaliteten af systemet væsentlig.

Dette projekt er blevet sat i gang for at opbygge en prototype til et trådløst produkt, der tager udgangspunkt i udviklingen af en light version af det eksisterende Discman system, og som kombinerer dette med den kommende kommunikationsplatform. Projektet skal være med til at illustrere nogle af de muligheder der følger med den nye platform, og samtidigt være med til at belyse nogle af de

sikkerhedsmæssige behov der introduceres i forbindelse med indføringen af en trådløs kommunikations standard.

2.1.2. Vision

Den overordnede vision for projektet er at konstruere grundlaget for et nyt produkt, der vil være med til at fremme perspektivet for DISCOS systemet og dermed være med til at skabe en mere brugervenlig tilgang til systemet. På basis af projektet vil det dermed på længere sigt være muligt at lancere et nyt produkt i sammenhæng med den nye kommunikationsplatform.

Projektet skal være med til at afdække de sikkerhedsmæssige risici, der opstår i forbindelse med indføringen af en trådløs kommunikations standard, for dermed at skabe en handlingsplan til, hvordan disse kan blive elimineret.

Derudover er forhåbningerne at projektet udmunder i en prototype af Discman PDA applikationen, der fremstår som et intuitivt og brugervenligt redskab til vedligeholdelse af DISCOS systemet. Denne prototype vil desuden være med til at demonstrere mulighederne og fordelene ved, at indføre mobile enheder i brugen af DISCOS systemet.

2.1.3. Kravspecifikation

Der skal udvikles en PDA applikation til vedligeholdelse af DISCOS overvågningssystemer. Applikationen skal lanceres hovedsagelig til brug af teknikere og servicearbejdere, men dog også andre folk med tilknytning til DISCOS systemet. Produktet skal bruges i forbindelse med kontrol og vedligeholdelses procedure af transformerstationerne, samt som demonstrationsredskab i forbindelse med markedsføringen af DISCOS produkterne.

Der skal udvikles 2 versioner af applikationen:

- Service mode: Dette skal være et system der specifikt benyttes af servicearbejdere, og som derfor indeholder de mest anvendte informationer omkring DISCOS systemet, herunder de mest almene målinger. Det er endvidere muligt at indstille de mest almindelige ikke kritiske systemindstillinger.

- Administrator mode: Dette skal være en udvidet version af systemet i forhold til service mode versionen. Denne version giver mulighed for at se yderligere informationer omkring DISCOS systemet, og gør det muligt at justere nogle ekstra indstillinger. Denne version vil hovedsageligt være tilgængelig til virksomhedens interne teknikere.

Platform

Applikationen skal udvikles på Windows Mobile 5.0 platformen, der benytter sig af .NET Compact Framework'et. Dette Framework udgør en begrænset udgave af det fulde .NET Framework, designet specifik til mobile enheder. .NET Framework'et er et software komponent, der indeholder klasse biblioteker til håndtering af forskellige programmeringsbehov vedrørende Windows platformen. Komponentet benyttes desuden til at håndtere eksekveringen af applikationer skrevet specifikt til dette Framework. Implementeringen af applikationen vil foregå via Visual Studio og C#.

Brugerflade
Brugerfladen skal være enkel og overskuelig (brugervenlig), således folk uden direkte kendskab til applikationen kan håndtere og manøvrere gennem de forskellige elementer af applikationen. Desuden skal der fokuseres på at gøre brugerfladen så intuitiv som muligt, så de forskellige system funktionaliteter tilpasses til den ønskede platform. Dette vil f.eks. indebære muligheden for at indføre forskellige sprog indstillinger.

Kommunikation
Kommunikationen mellem DISCOS systemet og PDA applikationen vil i dette projekt foregå via Bluetooth, dog skal systemet konstrueres, så der vil være mulighed for at udskifte kommunikationsdelen med en anden trådløs standard.

Da der er store begrænsninger i den måde, hvorpå Bluetooth portene er specificeret i Windows Mobile 5.0, skal der fokuseres på at finde en alternativ måde at håndtere Bluetooth kommunikationen på.

Da der på nuværende tidspunkt ikke er blevet konstrueret noget DISCOS kommunikationsmodul skal der i stedet udvikles en PC løsning, der skal emulere DISCOS kommunikationsmodulet. Denne løsning vil bestå i at der udvikles en applikation, som skal køre på en PC med et installeret Bluetooth og PCAN interface. PC applikationen skal sørge for at konvertere beskeder mellem Bluetooth og CAN BUS forbindelsen, hvilket derved gør det muligt, at forbinde kommunikationen mellem PDA'en og DISCOS systemet. Dette skal så udmunde i en koncept idé der illustrerer, hvordan dette vil kunne implementeres i det kommende kommunikationsmodul.

Sikkerhed
Da der kommunikeres via en trådløs forbindelse skal der fokuseres på sikkerheden af denne forbindelse. Dette skal være med til at sikre at informationer sendt via den trådløse forbindelse, ikke bliver opsnappet af en tredje part, som derved kan skaffe sig adgang til DISCOS systemet.

Dette er et højtprioriteret område, da en tredje part ellers ville have mulighed for at ændre i konfigurationen af DISCOS systemet. For at tilgå denne sikkerhedsmæssige risiko skal de data der sendes over den trådløse forbindelse krypteres. Dette vil medføre, at der skal opstilles en nøgle håndteringsplan, som gør det muligt for virksomheden at vedligeholde sikkerheden af det samlede system.

Datahåndtering
Såfremt der skal foretages test målinger af DISCOS systemet, skal der være mulighed for at PDA applikationen automatisk kan foretage målingerne. Dette skal fungere ved at de enkelte værdier automatisk opdateres, når der er foretaget et vist antal målinger.

Det skal derudover være muligt at optage/gemme de målte værdier, over en bestemt periode, fra DISCOS systemet for senere at kunne analysere disse.

Voice Log
Såfremt at PDA'en har en indbygget mikrofon, skal det være muligt at optage en talebesked i forbindelse med vedligeholdelses rutiner, af de enkelte DISCOS systemer. Denne feature skal bruges til at indtale resultatet af test målingerne for den enkelte transformerstation, så det senere kan blive ajourført.

2.1.4. Risikoanalyse

Ud fra den specificerede kravspecifikation er der blevet udført en risikoanalyse, der skal fastslå hvilke områder af projektet, der er de kritiske i forbindelse med udviklingen. Derved vil de mest essentielle elementer blive designet og implementeret først.

Risikoanalysen baserer sig på følgende vurderingsskala:

- Høj risiko krav
  - Essentielle krav, som er yderst nødvendige for implementeringen. Normalt implementeres disse krav altid først.
- Mellem risiko krav
  - Krav som ikke nødvendigvis er essentielle, men som stadig kan vise sig at være en udfordring at implementere.
- Lav risiko krav
  - Ikke essentielle krav, som ikke er nødvendige for implementeringen, men som vil kunne forbedre udgaven. Implementeres ofte kun i senere versioner.

I den følgende liste er de vigtigste høj risiko og mellem risiko elementer blevet beskrevet.

ID	Risiko	Strategi
RISK1	Søgning efter Bluetooth enheder samt oprettelse og vedligeholdelse af Bluetooth forbindelse.	Der foretages en række forundersøgelser af muligheden for at tilgå Bluetooth Stack'en i Windows Mobile 5.0 platformen. Kommunikationslaget vil blive implementeret som det første lag, da dette er grundstenen for hele applikationen.
RISK2	Indføring af kryptering.	Der foretages en analyse af de forskellige mulige algoritmer der findes i Compact Framework'et, med henblik på at implementere en af disse. Da sikkerheden i applikationen er et højt prioriteret område vil der blive konstrueret en sikkerheds risikovurdering der skal fastslå, hvordan sikkerheden skal integreres i systemet.
RISK3	Oprettelse af protokol i såvel PDA som PC applikation.	Der foretages en analyse af den bedste mulighed for, at kombinere den eksisterende CAN protokol med en ny Bluetooth protokol. Hvad angår PC applikationen skal mulighederne for at koble en virtuel seriel port sammen med PCAN modulet undersøges.
RISK4	Oprettelse af talebesked vha. den indbyggede mikrofon.	Der foretages en række forundersøgelser af muligheden for at tilgå den indbyggede mikrofon i Windows Mobile 5.0 platformen. Da denne feature ikke er kritisk i forhold til det samlede system vil denne blive implementeret samme med de andre program funktionaliteter.

Tabel 1 - Risiko vurdering.

RISK 1-3 vil blive designet og implementeret først for at sikre at applikationens kernefunktionalitet kommer til at fungere efter hensigten. Dette er en nødvendighed i forhold til at kunne implementere de resterende programfunktionaliteter, da disse skal bygges ovenpå denne struktur.

RISK 4 er en "nice to have" feature, der vil blive implementeret såfremt at der ikke opstår for mange komplikationer under udviklingen af RISK 1-3.

2.1.5. Afgrænsninger

Den udviklede prototype vil ikke indeholde alle de ønskede funktionaliteter, og vil dermed ikke kunne lanceres som et funktionelt produkt. Det endelige resultat vil derimod nærmere udgøre en koncept prototype, der illustrerer mulighederne for det endelige produkt og dermed skabe grundlaget for produktet.

Afgrænsninger i projektet i forhold til det endelige produkt:

- Der vil blive udviklet en PC løsning til at emulere DISCOS kommunikationsenheden, da der ikke forefindes noget DISCOS modul, der på nuværende tidspunkt kan kommunikere via Bluetooth.
- Der vil ikke blive implementeret nogen form for nøgle håndtering i DISCOS systemet. Der vil i stedet blive benyttet en fast specificeret nøgle.
- Det vil ikke være muligt at skrive nye værdier til EEPROM'en, dog vil logikken til at gennemføre skrivningen blive implementeret. Dette vil sige at begrænsningen vil foreligge i brugerfladen. Årsagen for dette valg ligger i at det ville være nødvendigt er lave et omfattende undtagelses system, der validere samtlige bruger indtastninger i forhold til det specifikke felt.
- Brugerfladen vil kun have et engelsk sprog interface. Der vil dog blive taget højde for at applikationen skal understøtte flere sprog på længere sigt.

2.2. Udviklingsproces

Dette afsnit specificerer, hvilken type udviklings proces, der vil blive benyttet for at sikre udviklingen af et velstruktureret software produkt. Denne proces udgør kernen i dette projekt, og er med til at sikre såvel kvaliteten af produktet samt strukturen af hele udviklingsforløbet.

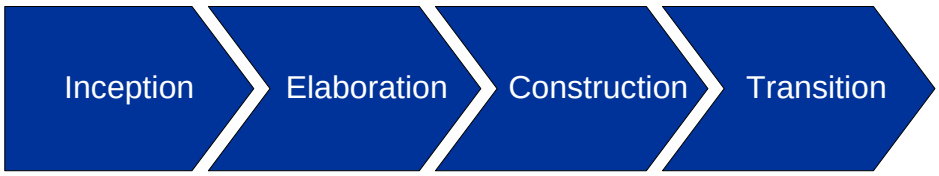
2.2.1. Unified Process

Der vil i dette projekt blive arbejdet med Unified Process (UP), hvilket er en software udviklings proces, der benytter sig af nogle specifikke redskaber til at opbygge velstrukturerede software applikationer. UP modellen er desuden med til at sikre at opstillede tidsplaner bliver overholdt, og at der udvikles i den rigtige retning i forhold til projekt specifikationen. Det elementære i UP modellen er dog ikke at gå 100 % efter den opstillede model, og dermed benytte sig af alle de introducerede redskaber, men derimod kun at benytte de elementer der kan bidrage til struktureringen af det specifikke projekt.

En af de grundlæggende idéer i UP er at de mest risikofyldte elementer i softwaren bliver varetaget allerede tidligt i forløbet. Denne proces kaldes for "Risk-Driven development" og er med til at sikre at der ikke opstår nogen uforudsete problemer i udviklingsforløbet.

Derudover er UP ensbetydende med iterativ udvikling, hvilket baserer sig på at opbygge software produkter gennem adskillige iterationer. Der forefindes en uddybende beskrivelse af iterativ udvikling i det efterfølgende afsnit.

UP introducerer 4 forskellige udviklingsfaser der tilsammen udgør følgende projektforløb:

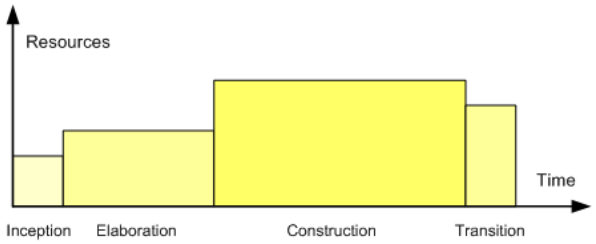


Figur 1 - Unified Process projektforløb.

- Inception
  - Etabler en business case for projektet.
  - Opret projekt rammer og afgrænsninger.
  - Opret en oversigt over Use Cases og krav der skal indgå i designet.
  - Opbyg en overordnet systemarkitektur.
  - Fortag en risikoanalyse af de enkelte systemelementer.
  - Etabler en foreløbig tids estimering af projektet.
- Elaboration
  - Design, implementer og valider en overordnet systemarkitektur.
  - Fortag en uddybende undersøgelse af de enkelte krav til systemet.
  - Design og implementer projektets høj risiko elementer.
  - Opbyg en projektplan for det resterende udviklingsforløb (Construction fasen).
- Construction
  - Beskriv de resterende krav til systemet.
  - Sikre at systemet imødekommer brugernes og virksomhedens behov.
  - Implementer og test de resterende dele af projektet.

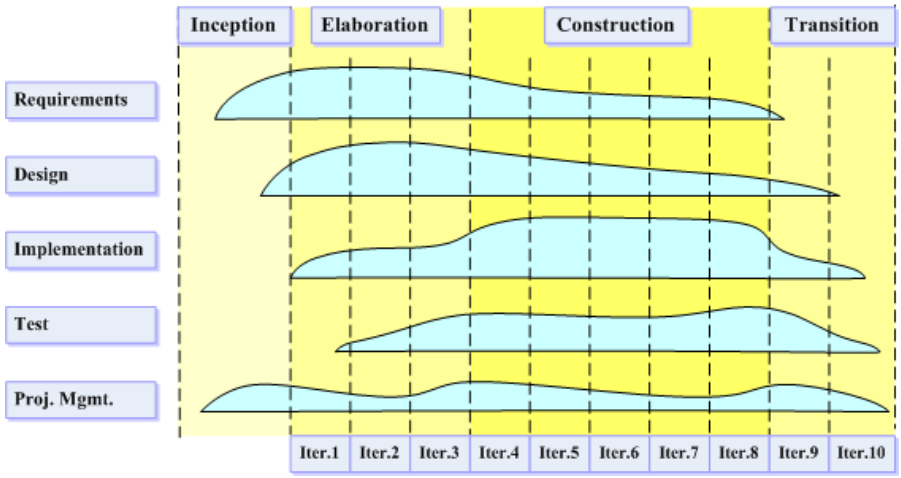
- Transition
  - Beta test systemet med henblik på at validere bruger/virksomheds forventninger.
  - Frigiv produktet.

Figuren nedenfor illustrerer sammenhængen mellem de 4 fasers tids og ressource behov.



Figur 2 - Illustration over det relative tidsforbrug for de enkelte udviklings faser.[1]

Gennem projektforløbet skifter de enkelte iterationer fokus fra i starten at fokusere på krav og design til implementering og test. Figuren nedenfor illustrerer denne proces.



Figur 3 - Diagram der illustrerer hvordan betydningen af de forskellige desipliner skifter gennem projektforløbet.[2]

[1] Reference: [3]
[2] Reference: [4]

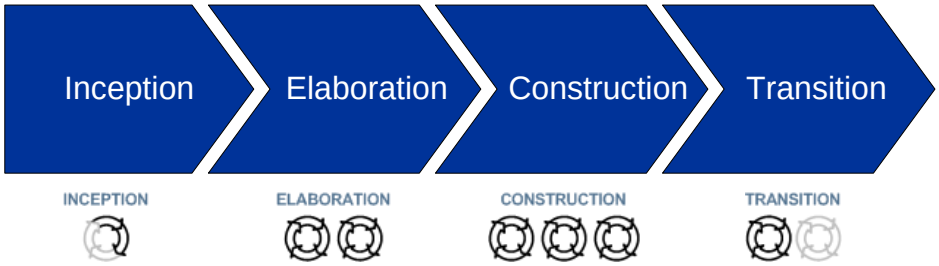
2.2.2. Iterative processer

Iterativ udvikling forgår ved at oprette et software produkt gennem adskillige mindre dele, iterationer. Hver iteration består som hovedregel af 4 overordnede opgaver, som er Planlægning, Analyse/Design, Implementering og Test og evaluering. Nedenfor ses en figur der illustrerer iterations cyklusen.

Figur 4 - Illustration af iterations cyklus.

Derved består det endelige program af adskillige mini releases, der alle har været igennem et omfattende udviklingsforløb. Dette sikre såvel kvaliteten af de enkelte dele som det endelige produkt og sikre at udviklingen forløb efter hensigten.

Følgende figur illustrerer sammenholdet mellem de enkelte faser i Unified Process projektforløbet og antallet af iterationer.



Figur 5 - Unified Process projektforløb med iterations overblik.

I inception fasen vil det kun være halvdelen af opgaverne i den iterative proces der er relevant, da der i denne fase som sagt fokuseres på at oprette specifikationen for selve projektet, og der derfor ikke vil indgå nogen form for implementering.

Elaboration fasen er den næststørste fase, der kan bestå af adskillige iterationer alt afhængig af de identificerede høj risiko områder.

Construction fasen er den største fase, der derved vil bestå af de fleste iterationer i forbindelse med implementering og test af de resterende softwarekomponenter.

Transition fasen kan variere i antallet af iterationer alt efter, hvor mange rettelser/releases software produktet skal igennem før det godkendes af kunden.

2.3. Tidsplan

Tidsplanen er opstillet i forhold til den overordnede system arkitektur, og baserer sig på at udmåle tidsforbruget for de enkelte moduler, der indgår i projektet. Dette gør det nemmere at estimere tidsforbruget for hele projektforløbet i stedet for at basere tidsplanen på de fire faser i Unified Process modellen.

Opgavenavn	Varighed	Start	Slut
- **Discman PDA**	**65,75d**	**Mon 19-02-07**	**Mon 21-05-07**
- **Forundersøgelse**	**1,25d**	**Mon 19-02-07**	**Tue 20-02-07**
- **Kommunikation**	**0,5d**	**Mon 19-02-07**	**Mon 19-02-07**
Bluetooth	4h	Mon 19-02-07	Mon 19-02-07
- **Indlejrede komponenter**	**0,5d**	**Mon 19-02-07**	**Mon 19-02-07**
Mikrofon	4h	Mon 19-02-07	Mon 19-02-07
PDA performance	2h	Tue 20-02-07	Tue 20-02-07
- **Iteration 1**	**8,63d**	**Tue 20-02-07**	**Fri 02-03-07**
Kravspecifikation	4h	Tue 20-02-07	Tue 20-02-07
Overordnet design	6h	Tue 20-02-07	Wed 21-02-07
- **Brugerflade**	**1d**	**Wed 21-02-07**	**Thu 22-02-07**
Markup	8h	Wed 21-02-07	Thu 22-02-07
- **Kommunikationslag**	**6,25d**	**Thu 22-02-07**	**Fri 02-03-07**
Specifikation	6h	Thu 22-02-07	Fri 23-02-07
Design	8h	Fri 23-02-07	Mon 26-02-07
Implementering	24h	Mon 26-02-07	Thu 01-03-07
Test	4h	Thu 01-03-07	Thu 01-03-07
Dokumentation	8h	Thu 01-03-07	Fri 02-03-07
- **Iteration 2**	**6d**	**Fri 02-03-07**	**Mon 12-03-07**
- **Sikkerhedslag**	**6d**	**Fri 02-03-07**	**Mon 12-03-07**
Analyse	10h	Fri 02-03-07	Tue 06-03-07
Design	10h	Tue 06-03-07	Wed 07-03-07
Implementering	12h	Wed 07-03-07	Thu 08-03-07
Test	4h	Thu 08-03-07	Fri 09-03-07
Dokumentation	12h	Fri 09-03-07	Mon 12-03-07
- **Iteration 3**	**5,5d**	**Mon 12-03-07**	**Tue 20-03-07**
- **Protokollag**	**4d**	**Mon 12-03-07**	**Fri 16-03-07**
Specifikation	6h	Mon 12-03-07	Tue 13-03-07
Design	6h	Tue 13-03-07	Wed 14-03-07
Implementering	10h	Wed 14-03-07	Thu 15-03-07
Test	4h	Thu 15-03-07	Fri 16-03-07
Dokumentation	6h	Fri 16-03-07	Fri 16-03-07
- **Midlertidig GUI**	**1,5d**	**Fri 16-03-07**	**Tue 20-03-07**
Design af simple GUI	4h	Fri 16-03-07	Mon 19-03-07
Implementering af GUI	8h	Mon 19-03-07	Tue 20-03-07
- **Iteration 4**	**7,75d**	**Tue 20-03-07**	**Fri 30-03-07**
- **Enhedslag**	**7,75d**	**Tue 20-03-07**	**Fri 30-03-07**
Specifikation	6h	Tue 20-03-07	Wed 21-03-07

Design	8h	Wed 21-03-07	Thu 22-03-07
Implementering	32h	Thu 22-03-07	Wed 28-03-07
Test	8h	Wed 28-03-07	Thu 29-03-07
Dokumentation	8h	Thu 29-03-07	Fri 30-03-07
- **Iteration 5**	**6,25d**	**Fri 30-03-07**	**Mon 09-04-07**
- **Applikationslag**	**6,25d**	**Fri 30-03-07**	**Mon 09-04-07**
Specifikation	6h	Fri 30-03-07	Fri 30-03-07
Design	8h	Fri 30-03-07	Mon 02-04-07
Implementering	24h	Mon 02-04-07	Thu 05-04-07
Test	6h	Thu 05-04-07	Fri 06-04-07
Dokumentation	6h	Fri 06-04-07	Mon 09-04-07
- **Iteration 6**	**4,5d**	**Mon 09-04-07**	**Fri 13-04-07**
- **Datalag**	**4,5d**	**Mon 09-04-07**	**Fri 13-04-07**
Specifikation	4h	Mon 09-04-07	Mon 09-04-07
Design	6h	Mon 09-04-07	Tue 10-04-07
Implementering	16h	Tue 10-04-07	Thu 12-04-07
Test	4h	Thu 12-04-07	Fri 13-04-07
Dokumentation	6h	Fri 13-04-07	Fri 13-04-07
- **Iteration 7**	**7d**	**Mon 23-04-07**	**Tue 01-05-07**
- **Brugerflade**	**7d**	**Mon 23-04-07**	**Tue 01-05-07**
Specifikation	6h	Mon 23-04-07	Mon 23-04-07
Design	8h	Mon 23-04-07	Tue 24-04-07
Implementering	28h	Tue 24-04-07	Mon 30-04-07
Test	6h	Mon 30-04-07	Mon 30-04-07
Dokumentation	8h	Tue 01-05-07	Tue 01-05-07
- **Iteration 8**	**2,75d**	**Wed 02-05-07**	**Fri 04-05-07**
Funktionstest	8h	Wed 02-05-07	Wed 02-05-07
Tilpasning efter test	8h	Thu 03-05-07	Thu 03-05-07
Dokumentation	6h	Fri 04-05-07	Fri 04-05-07
- **Iteration 9**	**11d**	**Fri 04-05-07**	**Mon 21-05-07**
Releasetest	4h	Fri 04-05-07	Mon 07-05-07
Resterende dokumentation	40h	Mon 07-05-07	Mon 14-05-07
Samling af dokumentation	8h	Mon 14-05-07	Tue 15-05-07
Korrektur og tilpasning	36h	Tue 15-05-07	Mon 21-05-07

Tabel 2 - Tidsplan

For at skabe en relativ sammenhæng mellem tidsplanen og Unified Process projektforløbet er følgende plan over milepæle opstillet.

Projektfase	Iteration	Estimeret deadline.
Inception	Forundersøgelse samt første del af iteration 1.	22-02-2007
Elaboration	Resten af iteration 1 + iteration 2 og iteration 3.	20-03-2007
Construction	Iteration 4 – 7.	01-05-2007
Transition	Iteration 8 – 9.	21-05-2007

Tabel 3 - Milepæle

2.4. Oversigt

Dette afsnit beskriver kort de enkelte kapitler i rapporten, hvilket vil være med til at belyse rapport strukturen.

- Analyse – i dette kapitel vil de enkelte problemstillinger blive analyseret og beskrevet. Der tages udgangspunkt i kravspecifikationen fra dette kapitel.
- Design – i dette kapitel vil der i forhold til de specificerede problemstillinger blive oprettet nogle løsningsmodeller. Disse vil specificere, hvordan der skal tages hånd om de enkelte problemstillinger, med henblik på en detaljeret og struktureret løsning.
- Implementering – i dette kapitel vil den færdige implementerede løsning til nogle af kerneelementerne blive beskrevet. Der vil være tale om en detaljeret beskrivelse af, hvordan løsningen er blevet implementeret.
- Test – i dette kapitel vil de gennemførte test blive beskrevet.
- Evaluering og konklusion – i dette kapitel vil der blive udført evalueringer af projektforløbet og udviklingsprocessen, samt fremtidsvisionen for projektet. Kapitlet afsluttes med en overordnet konklusion af projektet fra den projektstuderende og fra virksomhedens synspunkt.

3. Analyse

Formålet med dette kapitel er at belyse de overordnede problemstillinger i projektet, for senere at kunne konstruere løsningsforslag til disse. Kapitlet tager udgangspunkt i det eksisterende DISCOS overvågningssystem, med henblik på de anvendte teknologier og system funktionaliteter.

3.1. Teknologi

3.1.1. DISCOS System

DISCOS systemet kommunikerer internt via et CAN netværk som består af en primær Master enhed, 0 til 2 sekundære Master enheder, 0 til 8 Opti enheder pr. master og en kommunikationsenhed. Disse moduler er identificeret ved et unikt nummer, der består af mobiltelefonnummer, stations id og felt id.

Master enheden udgør et overvågningsmodul for transformer tilledningen til lavspændings netværket. Dette indebærer overvågning af følgende elementer:

- Trefaset spændingsovervågning (lavspænding)
- Backup batteri.
- To analoge indgange
- To binære indgange
- To kontrol relæer.

Opti enheden udgør et overvågnings- og kontrolmodul til mellemspændings linie og transformer felter. Dette indebærer følgende funktionaliteter.

- Trefaset optisk måling af mellemspænding og lavspændings strømme.
- Strømafbryder kontrol.
- Strømafbryder feedback.

GSM kommunikationsmodulet tilbyder følgende funktionaliteter.

- To vejs kommunikation med kontrolcenter via SMS.
- Fjernparametrering af indstillinger.

Internt er DISCOS modulerne koblet sammen via en bus, der viderebringer analoge målinger, CAN bus kommunikation og strømforsyning.

Informationer fra DISCOS systemet er koblet sammen med et SCADA eller et DNM system, hvilket er et storindustrielt målings- og kontrolsystem, gennem kommunikationsmodulet.

3.1.2. Måling og kontrol

Ved normal operation beregner systemet spænding, strøm, aktiv og reaktiv effekt i alle felter.

Strøm

Mellemspændings og lavspændings strømme måles vha. optiske sensorer med ikke ledende fiber kabel. Disse er tilkoblet mellemspænding/lavspændings strømlederne med en sensor per fase. Udgangen fra de optiske strømsensorer bliver monitoreret i Opti modulet.

Sensorerne består af 95 % plastik og 5 % glas og er fastspænd direkte på en kabelflade og forbundet til måleenheden ved at benytte optiske fibre som viderefører det polariseret lys.

Faraday effekten[3]

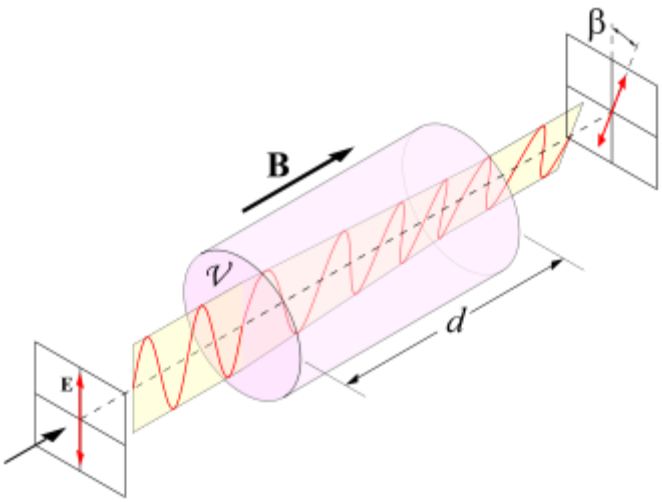
Til at beregne strømmene via de optiske sensorer benyttes Faraday effekt princippet, der beskriver:

Eftersom at en hver elektrisk strøm genererer et magnetfelt, kan strømmen måles ved at bestemme rotationsviklen af det polariserede plan. Relationen mellem rotationsvinklen af polariseringen og magnetfeltet i det diamagnetisk materiale er:

$$\beta = \sqrt{B}d$$

hvor

β er rotationsvinklen i radianer.
B er den magnetiske feltintensitet i udbredelsesretningen i teslas.
d er længden af afstanden i meter, hvor lyset og magnetfeltet interagere.
√ er verdet konstanten for materialet. Denne konstant varierer med bælgelængde og temperatur.



Figur 6 - Illustration af Faraday effekt princippet

[3] Der tages udgangspunkt i reference [15].

Spænding

Spændingsmålinger foretages kun på lavspændingssiden af transformeren af Master modulet. Transformerfaktor og vektorgruppe er indstillet ved opsætning og modulet kan derved beregne den tilsvarende spænding på mellemspændingssiden af transformeren.

Transformerens sekundære spænding er ført ind i systemet gennem master modulet og lavspændings repræsentationen af den sekundære spænding er distribueret mellem modulerne.

Aktiv og reaktiv effekt

Mellemspændingssidens aktive og reaktive effekt beregnes ud fra den trefaset strøm, der løber gennem mellemspændings linie felterne og de trefasede spændingsfaser i lavspændings transformer felterne.

Den aktive og reaktive effekt gennem transformeren beregnes i alle 3 faser på lavspændingssiden af transformeren.

Monitorerings funktioner

Systemet indeholder nogle forskellige indbyggede monitorerings funktioner, samt nogle udvidelses muligheder for tilslutning af eksterne sensorer, som f.eks. temperatur måler.

- Detektering af mellemspændings kortslutning.
- Detektering af mellemspændings jordfejl.
- Detektering af mellemspændings fasebrud.
- Overvågning af spændings grænseværdier på lavspændingssiden af transformeren.
- Strøm afbryder feedback.
- Ekstra input til monitorering.

Kontrol funktioner

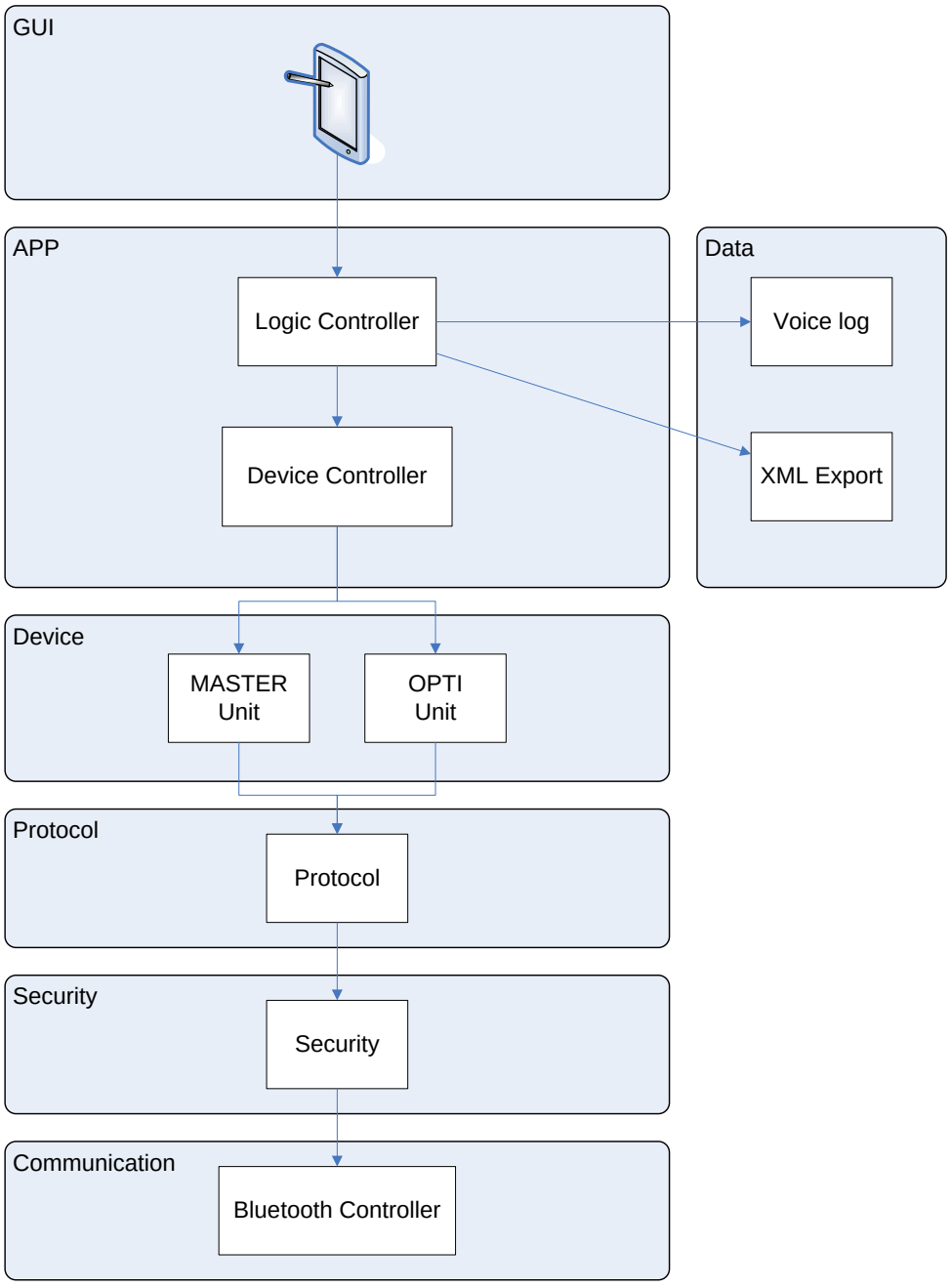
Systemet indeholder nogle basis kontrol funktioner, samt nogle ekstra udvidelses muligheder for at håndtere eksterne enheder.

- Strømafbryder kontrol.
- Ekstra output til kontrol.

3.2. Problemstilling

3.2.1. System Arkitektur

Der er blevet udarbejdet en foreløbig lagdelt system arkitektur, der skal danne grundlaget for implementeringen af applikationen. Denne struktur fokuserer på at uddelegere de overordnede system funktionaliteter til uafhængige lag moduler. Dette gør det endvidere også mere enkelt, at udskifte de enkelte lag i forbindelse med en udvidelse/ændring af applikationen. Nedenfor er der vist en illustration af system arkitekturen.



Figur 7 - System Arkitektur

Denne struktur vil gøre det enkelt at udskifte brugergrænseflade, at indføre en ny protokol og/eller at basere systemet på en anden kommunikations standard.

Kommunikationslag

Kommunikationslaget er ansvarlig for håndtering af Bluetooth enheder. Dette indebærer at søge efter enheder, samt at initialisere forbindelsen til disse. Ved succesfuld oprettelse af forbindelse til en enhed vil kommunikationslaget afgive den oprettede forbindelse til Protokol-laget.

Sikkerhedslag

Sikkerhedslaget er ansvarlig for at kryptere kommunikationen mellem applikationen og Bluetooth enheden. Såfremt forbindelsen til en enhed er blevet oprettet vil al udgående kommunikation blive krypteret, og alt indgående kommunikation blive dekrypteret.

Protokollag

Protokollaget er ansvarlig for at varetage al ind- og udgående trafik mellem applikationen og Bluetooth enheden. Efter at kommunikationslaget har oprettet forbindelse til en enhed, vil protokollaget overtage kontrollen med kommunikationen mellem enheden og applikationen.

Enhedslag

Enhedslaget er ansvarlig for håndtering af de modtagede DISCOS data, og vil udgøre en software repræsentation af DISCOS systemet.

Applikationslag

Applikationslaget er ansvarlig for styringen af det overordnede system. Dette lag sørger for, at sammenkoble GUI forespørgsler med de relevante systemfunktionaliteter.

GUIlag

GUIlaget udgør den overordnede brugergrænseflade struktur, og er ansvarlig for at give systemet en overskuelig og intuitiv facade.

3.2.2. Kommunikation

Det samlede system baserer sig på 2 forskellige kommunikations-standarder, hvilket er CAN bus og Bluetooth. Heraf benytter PDA applikationen sig af Bluetooth til at kommunikere med DISCOS systemet, hvorimod DISCOS systemet konverterer Bluetooth beskederne til CAN pakker, og sender disse ud på CAN bussen.

Da der ikke eksisterer noget kommunikationsmodul, på nuværende tidspunkt, der er i stand til at kommunikere Bluetooth, benyttes der i stedet en PC med tilhørende software til at emulere dette kommunikationsmodul. Modulet benytter en virtuel Bluetooth/seriel port til at kommunikere med PDA'en, samt et PCAN-USB modul til at kommunikere med CAN bussen.

CAN Bus

Kommunikationen med CAN bussen baserer sig som sagt på et PCAN modul, hvilket er et modul der konverterer USB til CAN. Ved hjælp af et medhørende bibliotek (DLL) til Windows er det muligt at udvikle applikationer, der kommunikere direkte med CAN bussen. For at simplificere CAN protokollen, med henblik på CAN pakkernes ramme struktur, varetager dette bibliotek hele opbygningen af CAN pakkerne. Dermed skal der kun tilføres en header og *payload* (data).

Bluetooth

Da mulighederne for at konfigurere PDA'ens Bluetooth enhed er yderst begrænset, specielt i Windows Mobile 5.0, vil der blive oprettet en virtuel port ved, at benytte *Windows Sockets* i stedet for at benytte en af de 2 virtuelle COM porte, der allerede er sat op i Windows Mobile. Dette skyldes først og fremmest at, hvis der skulle kommunikeres via de allerede oprettede porte ville det være nødvendigt at opsætte disse porte inden PDA applikationen startes, og det er derved ikke muligt at søge efter Bluetooth enheder fra applikationen.

Winsock

Winsock (Windows Sockets) er en specifikation, der definerer et netværks programmerings interface til Microsoft Windows, med henblik på hvordan netværks applikationer skal tilgå netværks tjenester såsom TCP/IP. Winsock baserer sig på BSD sockets[4], men med ekstra funktionalitet der gør at API'et overholder Windows programmerings modellen.

Microsoft Windows CE 5.0 benytter sig af Winsock 2.2, hvis funktionalitet er eksporteret fra Ws2.dll, der giver mulighed for at tilgå Bluetooth stack'en. Dette er den primære måde, hvorpå applikationer benytter sig af Bluetooth.

Profil

Bluetooth kommunikationen bruger Serial Port Profile (SPP), hvilket benytter RFCOMM[5] protokollen til at emulere RS232 serielle porte. Denne indfører en simpel trådløs erstatning til eksisterende RS-232 kommunikerende applikationer.

[4] Berkeley sockets er et API der tillader kommunikation mellem værtsmaskiner eller processor på en computer ved at benytte konceptet omkring en internet/network socket.
[5] Radio Frequency Communication (Serial Port Emulation).

3.2.3. Sikkerhed

Risikovurdering

Denne risikovurdering skal fastslå hvilke risici, der skal tages højde for i forhold til at producere et sikkert produkt, der kan lanceres til en bred mængde af kunder/virksomheder uden at der opstår sikkerhedsbrister. Endvidere vil det gennem denne risikovurdering være muligt at udpege, hvilke områder der skal fokuseres på, og dermed hvilke modforanstaltninger der skal til for at sikre systemet.

Hvilke risici kan true systemet?

Det ville være muligt for en tredjepart at opsnappe Bluetooth kommunikation mellem PDA'en og DISCOS kommunikationsenheden. Det ville kunne medføre at tredjeparten ville kunne sende beskeder til transformation, ved at imitere PDA'en, hvilket i værste fald ville kunne ændre på indstillingen af DISCOS systemet.

I det at produktet vil kunne blive lanceret til adskillige virksomheder ville det være muligt for konkurrenter at tilgå hinandens DISCOS systemer, med henblik på evt. at udføre spionage eller sabotage.

Hvilke konsekvenser ville disse risici kunne have?

Det ville være muligt at ændre på adskillelige af DISCOS systemet indstillinger. Dette vil primært dreje sig om indstillingerne af grænseværdierne for de forskellige alarmer, men det ville endvidere også være muligt at ændre på nogle af DISCOS systemet generelle indstillinger.

Ved ændring af alarm niveauer ville det kunne medføre, at der blev sendt en masse falske alarmer tilbage til centralen. Ved ændring af nogle af de generelle indstillinger vil det kunne påvirke den måde hvorpå DISCOS systemet reagere på forespørgsler eller andre hændelser, hvilket ville kunne få enheden til at fremstå som defekt. I begge tilfælde ville centralen skulle finde fejlen, hvorefter disse skulle rettes ved hjælp af fjern parametrering eller ved at sende en tekniker ud for at rette fejlen.

Der vil dog ikke kunne opstå nogen kritiske fejl i forbindelse med de nævnte risici, da applikationen ikke vil have tilgang til transformerens kritiske elementer.

Sandsynligheden for om de identificerede trusler vil finde sted?

Det er ikke særlig sandsynligt at de identificerede trusler vil finde sted, da det udelukkende vil have det formål at udøve sabotage. Der er umiddelbart ikke nogen informationer, der ville være interessant for andre end dem der styrer transformerstation. Dermed dog ikke sagt at situationen ikke ville kunne indtræffe.

Hvad ville en sikkerhedsbrist kunne påvirke?

Ville det vise sig at være muligt for konkurrenter at udøve sabotage af hinandens DISCOS systemer ville dette kunne medfører at virksomhedens produkter kommer til at fremstå som utroværdige og usikre.

Hvilke modforanstaltninger skal der iværksættes?

For at undgå at de identificerede risici kan opstå skal der indføres en form for kryptering af de data, der sendes via Bluetooth. Der vil desuden skulle opbygges et omfattende nøgle håndteringssystem, der skal sørge for at de enkelte virksomheder ikke kan få adgang til hinandens systemer.

Omkostninger ved at vedligeholde sikkerheden.

Såfremt krypteringen implementeres i systemet, vil der kun være omkostninger i forbindelse med håndteringen af nøgler i såvel DISCOS kommunikationsenheden, samt PDA'en.

3.2.4. Protokol

Da systemet benytter sig af to forskellige kommunikationsstandarder, CAN og Bluetooth, er det nødvendigt at specificere en protokol for hver af disse. Dette er nødvendigt, da der er nogle åbenlyse forskelle i, hvordan kommunikationen sendes via den enkelte standard.

CAN beskederne bliver sendt i hele pakker, der består af en header og et datafelt, hvorimod data sendt over Bluetooth, seriel forbindelse, bliver sendt som en *stream*[6] af bytes. Hvilket gør at det ikke er muligt med sikkerhed at kunne identificere, hvornår en besked starter og slutter.

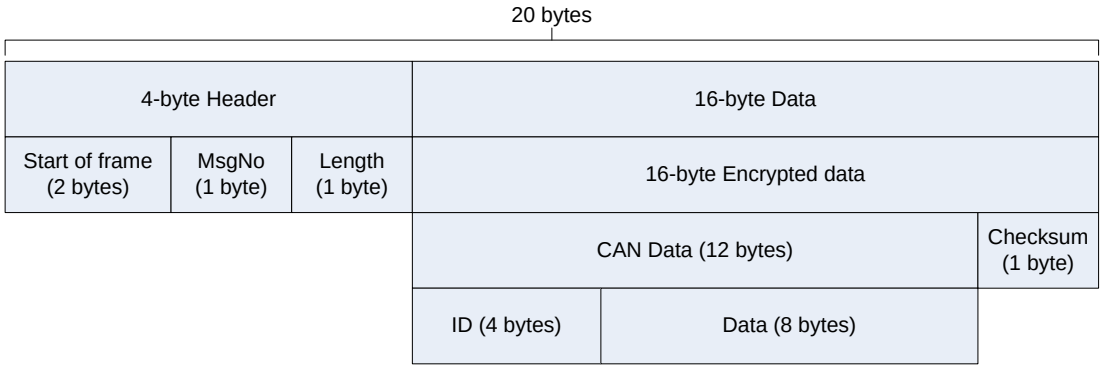
CAN (Controller Area Network)

CAN protokollen er på forhånd defineret i forhold til den måde der kommunikeres med DISCOS systemet. Denne protokol vil blive opretholdt i dette system, og vil danne kernen for, hvordan Bluetooth protokollen vil blive designet. Der bliver dog tale om en stærk reduceret CAN protokol, da dette system kun skal udnytte en del af DISCOS systemets funktionalitet (se 9.4.2).

Bluetooth

Bluetooth protokollen baseres på at opbygge en ramme til at indkapsle CAN protokollens identifikator felt og data felt (se 9.4.2). Dermed bliver al data pakket ind i en fast pakke struktur, som kan sendes over den serielle forbindelse, for dernæst at kunne blive konverteret til CAN pakker. Dette er nødvendigt for at kunne identificere de enkelte pakker, da de enkelte data bliver sendt som en datastrøm af bytes, og for at sikre integriteten af den sendte data.

Nedenfor stående figur viser, hvordan denne ramme er specificeret.



Figur 8 - Specifikation af Bluetooth ramme format

Da det maksimale *throughput*, ved at benytte rammen vist ovenfor, ligger på 8 Kbps (20*50*8), hvilket svare til at der sendes en CAN pakke hvert 20. ms, og Bluetooth har en kapacitet på omkring 721 Kbps er der ikke fokuseret på at optimere protokollen yderligere.

[6] En stream er en datastrøm der normalt er opdelt i individuelle bytes eller karakterer.

Header

Start of frame

Angiver starten af en Bluetooth pakke. Feltet består af 2 bytes og er sat til 0xFFFF for at mindske risikoen for redundant start sekvenser i data feltet.

MsgNo

Angiver pakkens beskednummer. Dette benyttes til at sikre at rækkefølgen på de modtagede pakker er korrekt.

Length

Angiver antallet af bytes med validt data i CAN Data'ens data felt.

Data

Dette felt består af CAN Data samt en checksum, der er blevet krypteret til et 16 bytes array. For yderligere information omkring krypteringen se 4.3.

CAN Data

Dette felt består af CAN pakkens ID felt og Data felt (se endvidere CAN (Controller Area Network) for yderligere informationer). Dette felt vil være krypteret for at beskytte indholdet.

Checksum

Dette felt indeholder checksummen for CAN Data feltet. Checksummen benyttes til at sikre integriteten af den sendte data. Der benyttes en simple checksum algoritme, der gemmer summen af de enkelte bytes.

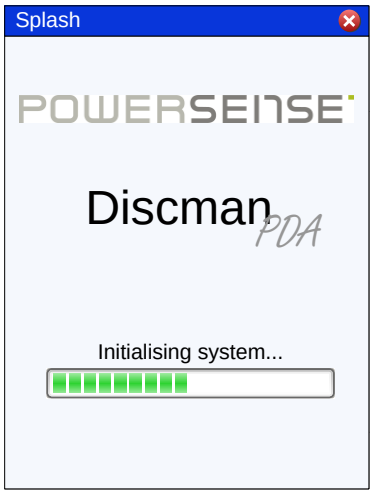
3.2.5. Brugergrænseflade

Der er i dialog med virksomheden blevet opstillet en oversigt over de forskellige vinduer, der skal indgå i den grafiske brugerflade. Specielt de enkelte indstillingsvinduer er meget detaljeret og omfattende, da disse danner grundlag for, hvilke muligheder brugeren skal have for at justere på de enkelte DISCOS enheder. Nedenfor ses en liste med de enkelte vinduer, samt en beskrivelse af disse.

Vinduer/moduler:

- *Splash*

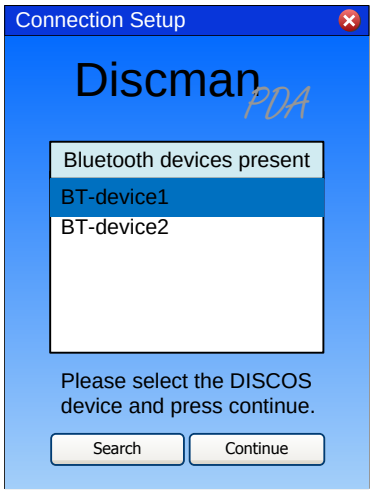
  Et introduktions vindue til visualisering af system initialiseringen.



Figur 9 - Splash vindue

- *Connection Setup*

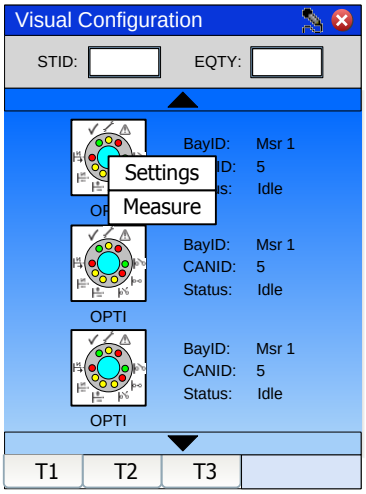
  Et vindue vedrørende oprettelse af Bluetooth forbindelse til DISCOS systemet.
  Alle fundne Bluetooth enheder vises i en liste, hvorefter brugeren vælger den ønskede enhed.



Figur 10 - Connection Setup vindue

- *Visual Settings*

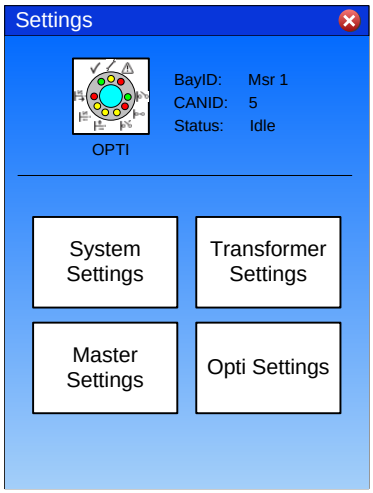
  Et vindue der visualiserer DISCOS systemets opsætning.
  Der vises en grafisk repræsentation af de enkelte DISCOS enheder monteret i transformerstationen, endvidere vises informationer omkring Bay id, CAN id og enheds status. Via de grafiske enheder er det muligt at se enhedens indstillinger og foretage test målinger.



Figur 11 - Visual Connection vindue

- *Settings*

  Et vindue der viser den valgte enhed og de mulige indstillingsgrupper.
  De mulige indstillingsgrupper er *System indstilling*, *Transformer indstilling*, *Master indstilling* og *Opti indstilling*.
  De indstillinger der er markeret med **fed** skrift skal kunne ændres fra service mode.
  De indstillinger der er markeret med **fed understreget** tekst skal kun kunne ændres fra administrator mode.

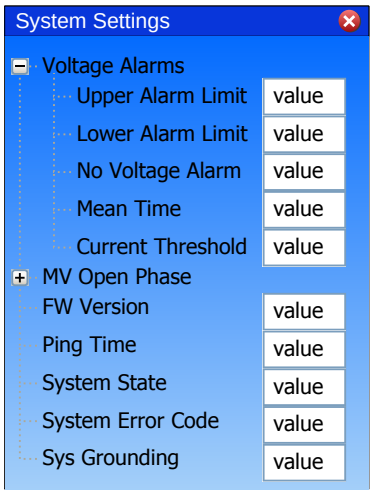


Figur 12 - Settings vindue

- *System Settings*

  Et vindue der viser de forskellige system-indstillinger:
  - Voltage Alarms – Spændings alarmer.

  - **Upper Alarm Limit** – Øvre alarm niveau.
  - **Lower Alarm Limit** – Nedre alarm niveau.
  - **No Voltage Alarm** – Grænseværdi svarende til ingen spænding.
  - **Mean Time** – Tidsinterval for måling af gennemsnits værdi i forhold til No Voltage Alarm niveau.
  - **Current Threshold** – Nedre grænseværdi for strøm.
- MV Open Phase – Fasebrud alarmer.
  - **Lower Alarm Limit** – Nedre alarm niveau.
  - **Current Threshold** – Nedre grænseværdi for strøm.
  - **Symmetry Factor** – Forholdet mellem minimum og maksimum strøm faser.
  - **Open Phase** – Tidsinterval for inspektion af fasebrud.
- FW Version – Enhedens firmware version.
- **PingTime** – Tidsinterval for afsending af system status beskeder.
- System State – System tilstand.
- System Error Code – Fejlkoder for systemet.
  - 0: Ingen fejl.
  - 1: EEPROM fejl.
  - 2: Processor Vd, Va og Ttemp er uden for grænseværdierne.
  - 3: Batteriet fejlede afladnings cyklusen eller har for lav spænding.
  - 4: Opti'ens lys værdier er lavere end service niveauet.
  - 5: Opti'ens lys værdier er lavere end alarm niveauet.
  - 6: MMI system genstart.
  - 7: Batteri spænding er normal.
  - 8: Batteri spænding er under grænseværdi.
  - 9: Ping tids fejl.
- Sys Grounding – Type af system jording.
  - Resonans spole.
  - Isoleret nulpunkt.
  - Modstand jordet nulpunkt.



Figur 13 - System Settings vindue

- *Transformer Settings*

  Et vindue der viser de forskellige transformer-indstillinger:
  - Vector Group Mounting – Vektor gruppe som følge af installation.
  - Vector Group – Vektor gruppe fra transformer specifikation.
  - Transformer Type – Prædefineret transformer type.
  - Rated Voltage – Nominel primær spænding.
  - Secondary Voltage – Nominel sekundær spænding.
  - Step Size – Transformer spændings trin i procent.
  - **Actual Tap Setting** – Det aktuelle udtag konfigureret på Transformeren.
  - Actual Primary Voltage – Den primære beregnede spænding.
  - **Transformer Ratio** – Transformer forholdet fra transformer type og udtag.

- *Master Settings*

  Et vindue der viser de forskellige Master-indstillinger:
  - Analogue Alarms – Analoge alarmer.
    - **AnAlarm1 Name** – Navnet på analog alarm 1.
    - **AnAlarm1 Threshold** – Grænseværdi for alarm 1.
    - **AnAlarm1 High/Low** – Alarm udløsnings niveau (over eller under grænseværdi) for alarm 1.
    - **AnAlarm1 Delay** – Alarm forsinkelse for alarm 1 i minutter.
    - **AnAlarm2 Name** – Navnet på analog alarm 2.
    - **AnAlarm2 Threshold** – Grænseværdi for alarm 2.
    - **AnAlarm2 High/Low** – Alarm udløsnings niveau (over eller under grænseværdi) for alarm 2.
    - **AnAlarm2 Delay** – Alarm forsinkelse for alarm 2 i minutter.
  - Binary Alarms – Binære alarmer.
    - **BiAlarm1 Name** – Navnet på binær alarm 1.
    - **BiAlarm1 High/Low** – Alarm udløsnings niveau (over eller under grænseværdi) for alarm 1.
    - **BiAlarm1 Delay** – Alarm forsinkelse for alarm 1 i minutter.
    - **BiAlarm2 Name** – Navnet på binær alarm 2.
    - **BiAlarm2 High/Low** – Alarm udløsnings niveau (over eller under grænseværdi) for alarm 2.
    - **BiAlarm2 Delay** – Alarm forsinkelse for alarm 2 i minutter.
  - Binary Output – Binære udgangs alarmer.
    - **BoAlarm1 Name** – Navnet på binær udgangs alarm 1.
    - **Bo1ActLvl** – Alarm aktiverings niveau for alarm 1.
      - Åben: relæ slået fra.
      - Lukket: relæ slået til.
    - **BoAlarm2 Name** – Navnet på binær udgangs alarm 2.
    - **Bo2ActLvl** – Alarm aktiverings niveau for alarm 2.
      - Åben: relæ slået fra.
      - Lukket: relæ slået til.

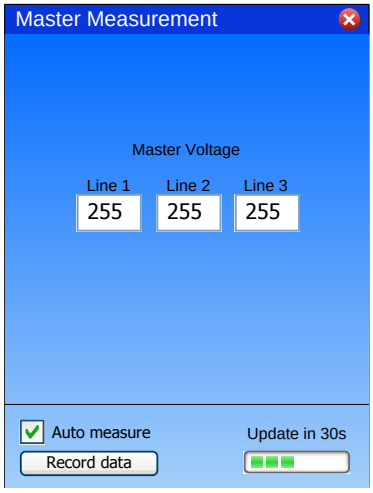
        - **Active Relay Time** – Aktiverings/deaktiverings tidsinterval for relæ.
    - Charge Control
        - **Battery Voltage** – Nominel batteri spænding.
        - **Battery Low** – Alarm niveau for batteri spænding.
        - **Battery Discharge** – Det maksimale tilladte spændingsfald under afladning.
        - **Battery Control** – Tidsinterval for tjek af batteri afladning.

- *Opti Settings*

  Et vindue der viser de forskellige Opti-indstillinger:
    - MV Switching – MV skift.
        - **Relay Active Time** – Tidsinterval for aktivt relæ.
        - **Relay Delay Time** – Tidsinterval for hvor lang tid relæet skal være uspecificeret før der afsendes en SMS (sek.).
    - Earth Fault – Jordlednings fejl.
        - **Short HARM Time** – Tidsinterval for detektering af harmonisk niveau ved jordfejl.
        - **Long HARM Time** – Tidsinterval for detektering af gennemsnitlig harmonisk niveau ved normal drift.
        - **HARM Trigger** – Nedre grænseværdi for forholdet mellem short og long harmonisk niveau ved jordfejl.
    - **Sys Grounding** – Type af system jording.
        - Resonans spole.
        - Isoleret nulpunkt.
        - Modstand jordet nulpunkt.
    - Short Circuit – Kortslutnings alarm.
        - **Short Circuit Alarm** – Kortslutnings alarm niveau.
        - **MTA** – Maksimal drejningsmoment vinkel.
    - Sensor Light
        - DC Value 1 – DC lysværdi for linie 1.
        - DC Value 2 – DC lysværdi for linie 2.
        - DC Value 3 – DC lysværdi for linie 3.
        - DC Service 1 – DC service alarm niveau for linie 1.
        - DC Service 2 – DC service alarm niveau for linie 2.
        - DC Service 3 – DC service alarm niveau for linie 3.
        - DC Alarm 1 – DC alarm niveau for linie 1.
        - DC Alarm 2 – DC alarm niveau for linie 2.
        - DC Alarm 3 – DC alarm niveau for linie 3.
    - Field Type – Transformer felt type.

- *Master Measurement*

  Et vindue der gør det muligt at foretage test målinger af BUS spænding.
  De aktuelle spændinger for de enkelte linier vises.

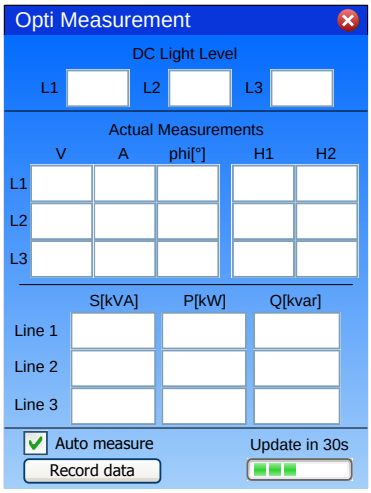


Figur 14 - Master Measurement vindue

- *Opti Measurement*

  Et vindue der gør det muligt at foretage test målinger af Opti'ens:

  - DC lys værdier – nice to have feature.
  - Spænding
  - Strøm
  - Faseforskel (phi)
  - Aktiv effekt (P)
  - Reaktiv effekt (Q)
  - Tilsyneladende effekt (S)
  - Harmoniske værdier – Kun i administrator mode.



Figur 15 - Opti Measurement vindue

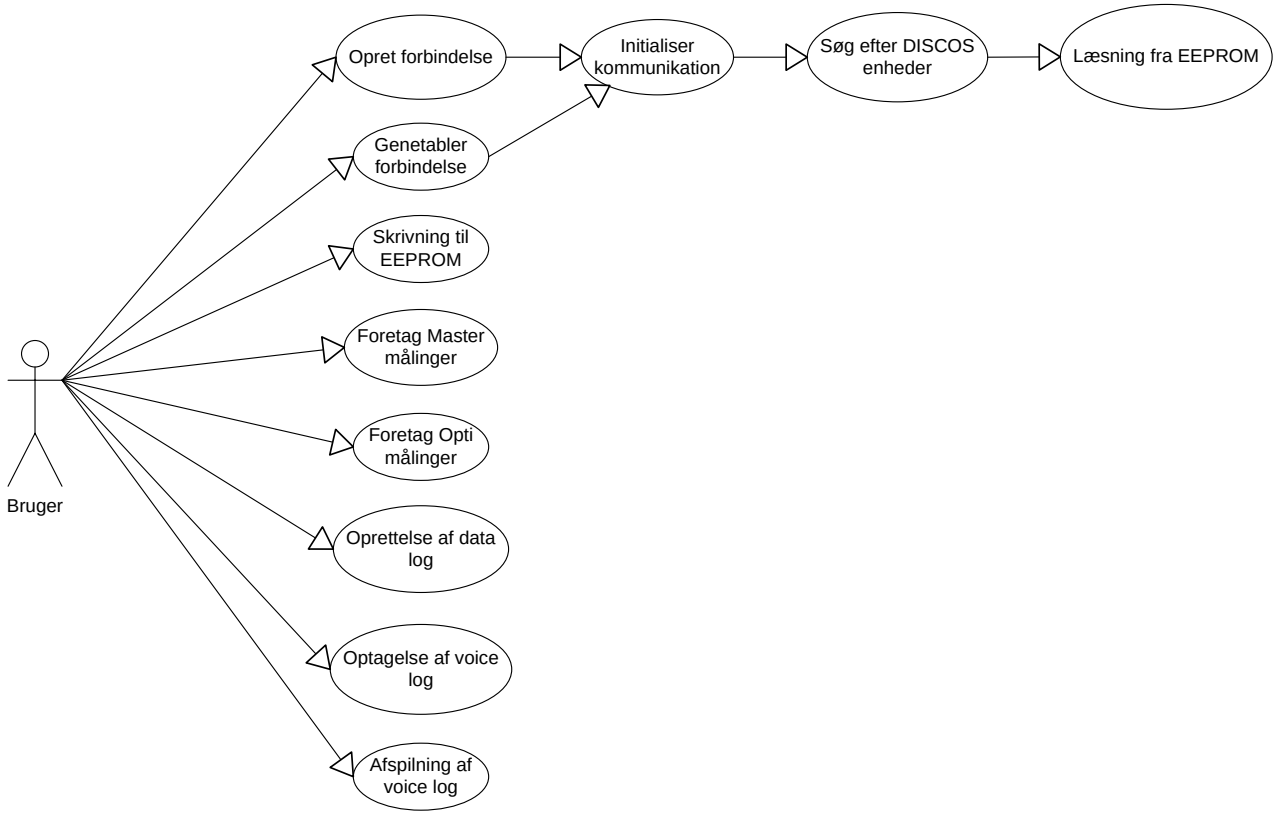
3.2.6. Use Cases

Ud fra de krav der er blevet opstillet i sammenarbejde med PowerSense, er følgende use cases blevet opstillet. Disse illustrerer det overordnede system og dets virkemåde. Ud af de opstillede 11 use cases vil 2 blive gennemgået i de følgende afsnit, mens de resterende kan findes i bilag 9.1.

Use Case	Id	Prioritet
Opret forbindelse	§ 0001	Høj
Initialiser kommunikation	§ 0002	Høj
Genetabler forbindelse	§ 0003	Mellem
Søg efter DISCOS enheder	§ 0004	Høj
Læsning fra EEPROM	§ 0005	Høj
Skrivning til EEPROM	§ 0006	Høj
Foretag Master målinger	§ 0007	Høj
Foretag Opti målinger	§ 0008	Høj
Oprettelse af data log	§ 0009	Lav
Optagelse af voice log	§ 0010	Lav
Afspilning af voice log	§ 0011	Lav

Tabel 4 - Use Case oversigt

Use Case diagram



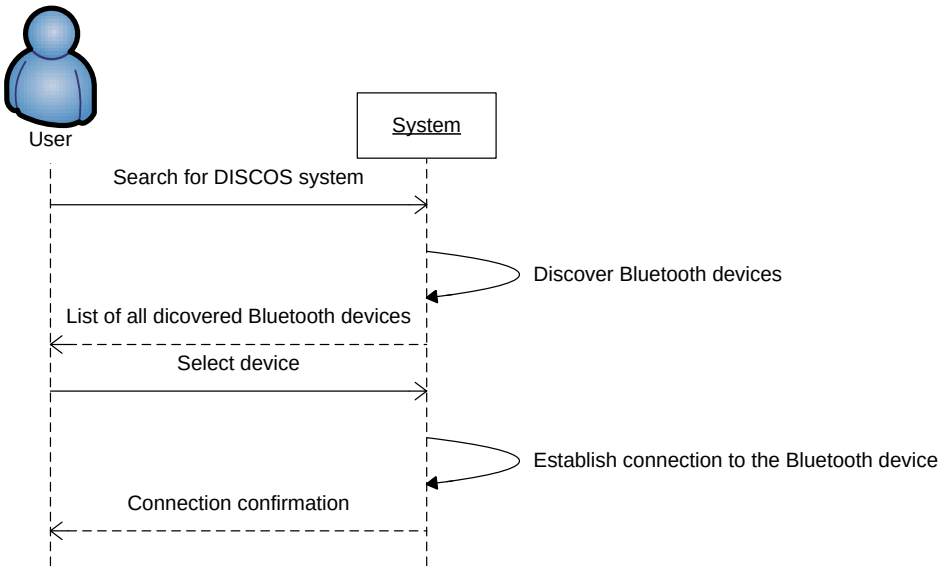
Figur 16 - Use Case diagram

§ 0001 – Opret forbindelse.

Denne use case vedrører oprettelse af forbindelse til DISCOS systemet via Bluetooth.

ID	§ 0001 – Opret forbindelse.
Formål	At oprette forbindelse til DISCOS systemet.
Primær aktører	Bruger.
Motiver og interesser	Indhentning af data/målinger fra DISCOS systemet.
Trigger	Opstart af applikationen.
Forudsætninger	Brugeren befinder sig indenfor 10 meter afstand fra et DISCOS system, med Bluetooth kommunikationsenhed.
Succesbetingelser	Succesfuld oprettelse af forbindelse til DISCOS systemet.
Fejlbetingelser	Der kan ikke oprettes forbindelse til DISCOS systemet.
Hovedforløb	1. Brugeren vælger at søge efter DISCOS systemet. 2. Der søges efter tilgængelige Bluetooth enheder. 3. En liste med enhedsnavnene på alle fundne Bluetooth enheder vises. 4. Brugeren vælger den ønskede enhed, der skal oprettes forbindelse til. 5. Der oprettes forbindelse til enheden. 6. Der vises en meddelelse omkring succesfuld oprettelse af forbindelse.
Udvidelser	
Alternativt forløb	4a. Den ønskede enhed forefindes ikke på listen. Brugeren kan foretage følgende valg: • Foretage en ny søgning (Gå til trin 1). • Afslutte programmet. 5a. Der kan ikke oprettes forbindelse til den valgte enhed. Brugeren informeres om situationen og kan nu foretage følgende valg: • Foretage en ny søgning (Gå til trin 1). • Vælge en anden enhed (Gå til trin 4). • Afslutte programmet.
Prioritet	Høj
Hyppighed	Benyttes hver gang applikationen startes.
Referencer	

Tabel 5 - Use Case §0001: Opret forbindelse



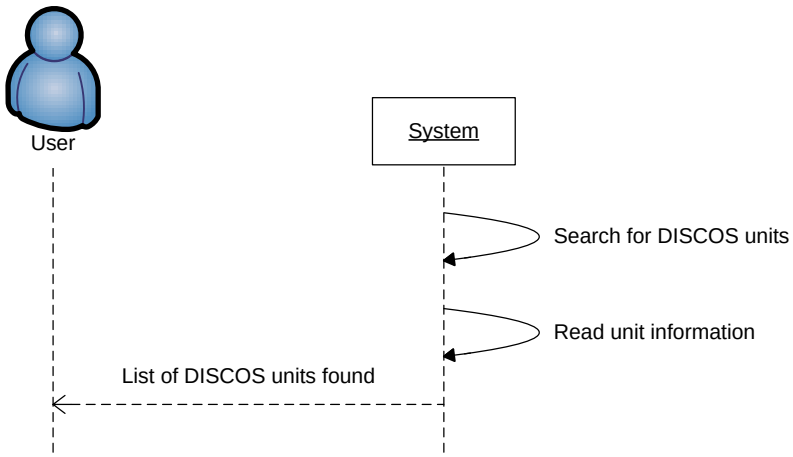
Figur 17 - System Sekvensdiagram: Opret forbindelse

§ 0004 – Søg efter DISCOS enheder

Denne use case vedrører søgning efter DISCOS enheder.

ID	§ 0004 – Søg efter DISCOS enheder.
Formål	At identificere de tilsluttede DISCOS enheder.
Primær aktører	Bruger.
Motiver og interesser	At kunne foretage test målinger af de eksisterende enheder, samt at give mulighed for at ændre på den enkelte enheds eksisterende konfiguration.
Trigger	Initialisering af kommunikationen er succesfuld, og der er dermed oprettet forbindelse til en DISCOS system.
Forudsætninger	Der er oprettet forbindelse til et DISCOS system, og kommunikationen er blevet initialiseret.
Succesbetingelser	Søgningen fuldføres fejlfrit.
Fejlbetingelser	Søgningen kan ikke gennemføres.
Hovedforløb	1. Systemet søger efter tilsluttede DISCOS enheder. 2. Indstillingerne for de fundne enheder læses fra EEPROM'en (Use Case §0005). 3. Der returneres en liste med de fundne enheder.
Udvidelser	1a. Systemet finder ikke nogen tilsluttede DISCOS enheder. Der fortsættes fra punkt 3.
Alternativt forløb	1a. Søgningen kan ikke fuldføres, da forbindelsen til DISCOS systemet er blevet afbrudt. Brugeren informeres om situationen og kan nu foretage følgende valg: • Forsøge at genetablere forbindelsen (Use Case §0003) • Foretage en ny søgning (Use case §0001).
Prioritet	Høj
Hyppighed	Benyttes hver gang der er blevet oprettet forbindelse til et DISCOS system.
Referencer	Use case: §0001, §0003, §0005

Tabel 6 - Use Case §0004: Søg efter DISCOS enheder



Figur 18 - System Sekvensdiagram: Søg efter DISCOS enheder

3.3. Delkonklusion

Gennem dette kapitel er de enkelte risikoområder blevet analyseret for, at fastlægge de enkelte problemstillinger i projektet. Derudover er der blevet opstillet use cases for hver af de foreliggende system scenarioer, hvilket vil danne grundlag for det videre design.

Efter at have arbejdet med de enkelte problemstillinger gennem dette kapitel, er der opnået en større forståelse for de enkelte behov/krav til systemet. Der er desuden blevet fastlagt en overordnet arkitektur for systemet, der lever op til de enkelte krav. Denne arkitektur ligger vægt på et modulært og struktureret design, der gør det enkelt at kunne udskifte enkelte dele af systemet, som f.eks. kommunikationsstandard.

Kommunikation
Der bliver fokuseret på at lave et kommunikations led mellem PDA'en og DISCOS systemet, da det eksisterende kommunikationsmodul ikke understøtter Bluetooth. Dette vil bestå af en PC med tilhørende software, der vil sørge for at sammenkoble Bluetooth kommunikationen med CAN kommunikationen. Endvidere vil der på PDA siden blive benyttet *Winsockets* til at styre Bluetooth kommunikationen, da dette giver den største frihed i forhold til tilgængelighed og kontrol.

Sikkerhed
Ud fra den opstillede risikovurdering er det blevet gjort klart, at det er nødvendigt at indføre kryptering af de enkelte beskeder for, at undgå uautoriseret adgang til de enkelte DISCOS systemer. Sikkerhedsniveauet anses dog som lav risiko, da det ikke er muligt at foretage høj risiko funktioner via PDA'en.

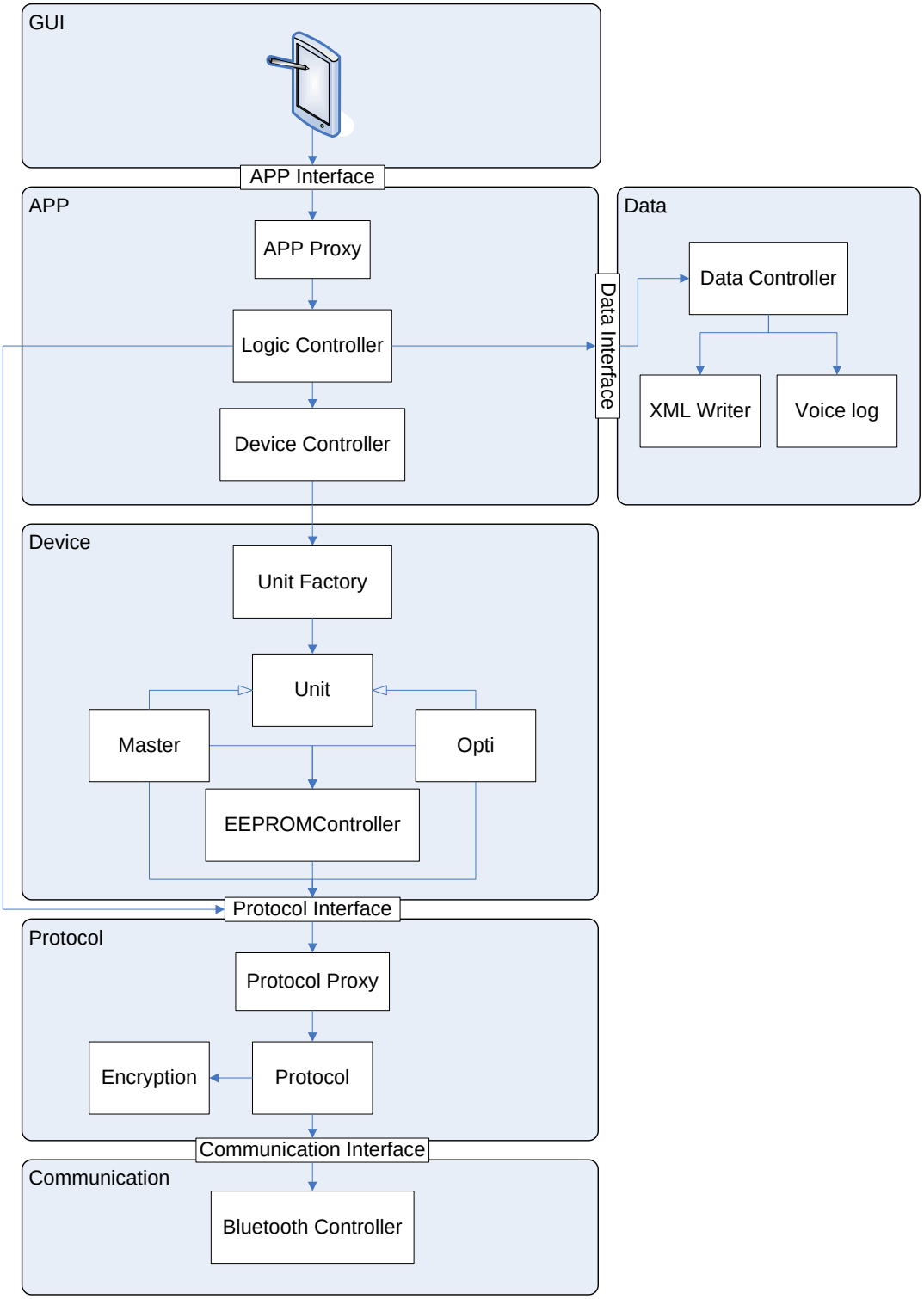
Protokol
Den eksisterende CAN protokol er blevet anvendt til at opbygge en Bluetooth ramme, der understøtter muligheden for kryptering af data, samt en enkel konvertering mellem Bluetooth og CAN. Desuden benytter denne ramme sig af besked nummerering, der skal sikre besked synkroniseringen, og checksum, der beskedens integritet.

Brugergrænseflade
Brugergrænsefladen har udgjort basis for hoveddelen af de opstillede problemstillinger. Denne er blevet opstillet i sammenarbejde med virksomheden, og specificerer hvilke funktionaliteter systemet skal understøtte. Det er blevet besluttet at systemet skal kunne vise konfigurationen af de enkelte enheder, med begrænset mulighed for ændring af specifikke felter. Derudover skal der kunne foretages basis kontrol målinger af de enkelte enheder.

4. Design

Formålet med dette kapitel er at opstille nogle løsningsmodeller i forhold til de specificerede problemstillinger. Disse skal være med til at belyse de enkelte elementer i systemet og danne grundlaget for den endelige implementering.

4.1. Systemarkitektur



Figur 19 - System arkitektur

Den endelige systemarkitektur består af 6 lag:

- GUI – Alle enheder/elementer der indgår i den grafiske brugerflade
- APP – Det centrale applikations lag. Håndterer forespørgsler for brugergrænsefladen ved at styre program logikken og dermed tilgå de relevante systemressourcer. Dette lag sammenkobler de enkelte lag.
- Device – Alle enheder/elementer der vedrører styringen af de enkelte DISCOS enheder.
- Protokol – Håndterer alt kommunikation til og fra applikationen. Til forskel for det første udkast af systemarkitekturen er security laget blevet sammenkoblet med protocol laget. Dermed sørger protocol klassen for at kryptere og dekryptere alt ind- og udgående kommunikation.
- Communication – Administrerer den trådløse forbindelsen mellem DISCOS systemet og applikationen.
- Data – Ansvarlig for at gemme data og optage voice log.

4.1.1. Tråde

Ved initialiseringen af applikationen oprettes der 3 tråde der udgør den elementære tråd-struktur for systemet:

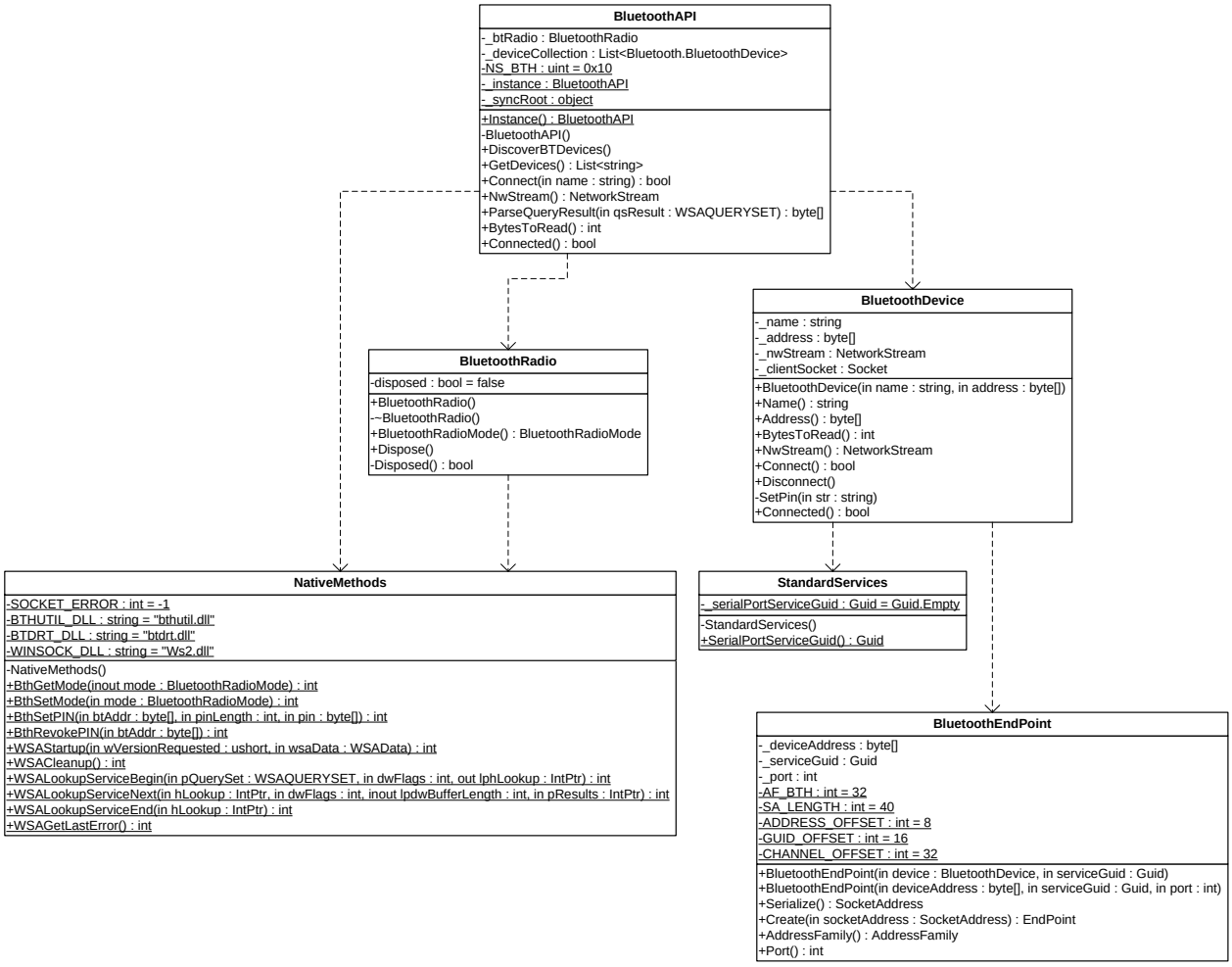
- GUI tråd – denne tråd er den overordnede applikations-tråd, og benyttes primært til at modtage/håndtere bruger input via den grafiske brugerflade.
- Parser tråd – denne tråd er ansvarlig for modtagelse og parsing af beskeder (se 4.4.2) for uddybende information om trådens funktion. Så snart en besked er blevet parset tilføjes denne til en besked kø.
- Besked tråd – denne tråd er ansvarlig for håndtering af beskeder fra besked køen. Tråden undersøger om der er nogle beskeder i besked køen, hvorefter den udsender et event til alle observers med besked informationerne og beskeden fjernes fra køen. Denne tråd kører derfor i mange tilfælde helt op til GUI laget, hvor den så påkalder GUI tråden for, at denne skal overtage data håndteringen. Derudover oprettes der for hver forespørgsel fra GUI laget, hvor der skal udføres en form for logisk operation, en ny tråd. Dette gøres for at sikre at GUI tråden kan blive ved med at håndtere bruger input selvom der udføres logiske operationer. Oprettelsen af disse tråde udføres i APPProxy klassen i forbindelse med implementeringen af et proxy pattern.

4.2. Kommunikation

Dette afsnit beskriver opbygningen af Communication namespace'et med henhold til de forskellige klasser der indgår i dette. Dette namespace vedrører kun PDA applikationen og ikke DISCOS kommunikations emulator delen. Dette skyldes at der til design af emulatorens kommunikationslag er blevet benyttet en standard seriel porte, der er blevet konfigureret via computerens Bluetooth driver, samt en fast specificeret PCAN driver. Derved indeholder dette namespace ikke noget nævneværdigt designmæssigt.

4.2.1. Bluetooth

Da der ikke er blevet implementeret et managed interface til at tilgå Bluetooth stack'en og *winsockets* i .NET Compact Framework'et, vil PDA applikationens kommunikationslag basere sig på at konstruere en managed "wrapper" til dette formål. Dette indebære at der fra managed kode vil blive kaldt nogle native metoder/funktioner, hvis data vil blive konverteret og pakket ind i managed kode. På den måde er det muligt at holde alt kodning i managed og dermed benytte sig af automatisk garbage collection.



Figur 20 – Klassediagram: Kommunikationslaget.

BluetoothAPI

Denne klasse udgør et API til håndteringen af Bluetooth funktionaliteter. Denne klasse giver mulighed for at benytte følgende Bluetooth funktionaliteter:

- Søgning efter Bluetooth enheder.
- Oprettelse af forbindelse til en Bluetooth enhed.
- Afbrydelse af forbindelse.

BluetoothRadio

Denne klasse håndterer styringen af PDA'ens Bluetooth radio. Det er muligt at skifte mellem følgende indstillinger:

- Tændt.
- Slukket.
- Tændt og synlig.

Endvidere sørger denne klasse også for initialiseringen og afslutningen af *winsocket* service's.

NativeMethods

Denne klasse håndterer al tilgang til de "native" OS indlejrede metoder der benyttes i forbindelse med oprettelsen af Bluetooth kommunikation. Dette er primært i forbindelse med oprettelsen af *winsockets*. Endvidere indgår alle de strukturer og enumerators der benyttes til at konvertere data fra unmanaged til managed også i denne klasse. Gennem denne klasse er det dermed muligt at få tilgang til følgende funktionaliteter:

- Håndtering af PIN kode til Bluetooth enhed.
- Håndtering af Bluetooth radio.
- Håndtering af *Winsockets*.

BluetoothDevice

Denne klasse benyttes til at repræsentere de enkelte fundne Bluetooth enheder. Fra denne klasse håndteres forbindelsen til den enkelte enhed. Gennem denne klasse er det muligt at:

- Hente konkrete informationer omkring enheden.
- Oprette/Afslutte forbindelse til enheden.
- Tilgå et NetworkStream objekt til at kommunikere over.

StandardServices

Denne klasse indeholder en GUID(Global Unique Identifier) for hver af de Bluetooth service's der skal benyttes. Disse benyttes i forbindelse med oprettelsen af et Bluetooth Endpoint. Da systemet kun vil kommunikere via Serial Port Profilen er det den eneste GUID der er i klassen.

BluetoothEndPoint

Denne klasse benyttes til at oprette et Bluetooth Endpoint, der skal bruges til oprettelsen af en socket, der kan kommunikere med en Bluetooth enhed.

4.3. Sikkerhed

For at sikre at en tredjepart ikke har mulighed for at opsnappe informationer sendt over den trådløse kommunikationskanal, vil systemet baseres på at indføre kryptering af de sendte data. Dermed vil det være muligt at hemmeligholde sendte informationer ved, at omdanne den oprindelige information til information, der er ulæselig for en tredjepart.

4.3.1. Symmetrisk kryptering

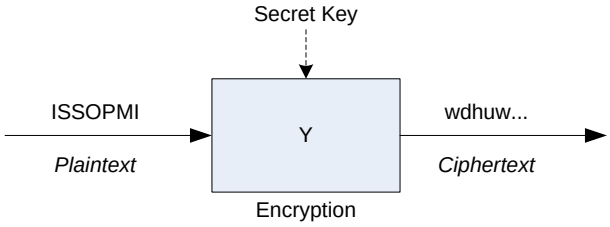
Systemet vil benytte sig af symmetrisk kryptering, der også er kendt som secret key kryptering, hvilket baserer sig på at benytte én nøgle til såvel kryptering og dekryptering. Det symmetriske system muliggør tovejs kommunikation mellem 2 parter ved at benytte en delt hemmelig nøgle. Denne benyttes af begge parter til såvel at sende krypterede informationer til hinanden, samt at dekryptere de modtagede informationer. Så længe at nøglen er hemmelig, sørger systemet dermed også for at sikre autenciteten af den enkelte part. Årsagen til at systemet benytter sig af symmetrisk kryptering frem for asymmetrisk skyldes, at der kun skal kunne udveksles informationer mellem nogle bestemte enheder, hvilket er sikret ved at det kun er den legitime afsenders besked, der kan dekrypteres rigtigt ved hjælp af den hemmelige nøgle.

Svagheden ved symmetrisk kryptering ligger i, at sikre uddelingen af de hemmelige nøgler til de bestemte enheder. Hvordan sikres det at en hemmelig nøgle ikke opfanges af en tredjepart? Dermed skal der ved alle krypterings algoritmer fokuseres på nøgle håndteringen.

Stream og Blok Chiffer

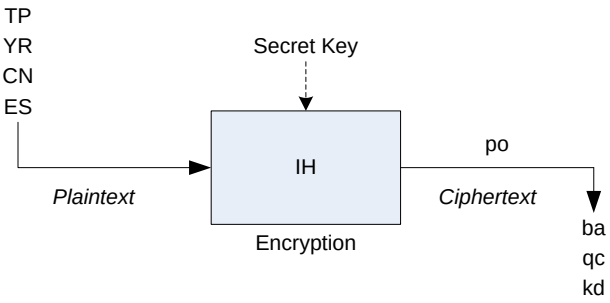
Symmetriske krypterings algoritmer kan opdeles i 2 grupper *stream* chiffer og *blok* chiffer.

Stream chiffer konverterer hvert enkelt klartekst-symbol til et chiffertekst-symbol.



Figur 21 - Stream kryptering

Blok chiffer krypterer en gruppe af klartekst-symboler som en blok og skaber chiffer blokke.



Figur 22 - Blok kryptering

	Stream Krypterings Algoritmer	Blok Krypterings Algoritmer
Fordele	• *Krypteringshastighed*. Da hvert symbol er krypteret hver for sig uden hensyntagen til den efterfølgende klartekst, afhænger krypteringshastigheden kun af selve algoritmen. • *Fejl udbredelse*. Da hvert symbol krypteres enkeltvis, vedkommer en fejl i krypteringsprocessen kun dette symbol.	• *Diffusion*. En chiffertekst blok kan indeholde adskillelige klartekst symboler. • *Modificering af datasekvens*. Da en blok af symboler er krypteret vil det ikke være muligt at indsætte et symbol uden at dekrypteringen fejler.
Ulemper	• *Diffusion*. Alle symboler er krypteret enkeltvis, derfor forefindes alle informationer i det enkelte chiffertekst-symbol. • *Modificering af datasekvens*. I tilfælde af at en tredjepart har knækket koden for et symbol, kan dette benyttes til at afdække de resterende dele af beskeden.	• *Krypteringshastighed*. Det er nødvendigt at vente på at en hel blok klartekst før krypteringen startes. • *Fejl udbredelse*. En fejl vil påvirke transmissionen af alle andre symboler i den samme blok.

Tabel 7 - Sammenligning mellem Stream og Blok kryptering[7]

Da blok krypterings algoritmen har nogle elementære fordele indenfor sikkerheden, vil systemet basere sig på at benytte denne form for symmetriske krypterings algoritmer.

4.3.2. Algoritmer Understøttet af .NET Framework'et

.NET platformen understøtter følgende symmetriske blok chiffer krypterings algoritmer[8]:

- DES – Data Encryption Standard, er en ældre algoritme der har vist svagheder i forhold til nutidens standarder. Dette skyldes hovedsageligt længden på nøglen og blok størrelsen. Derudover blev denne algoritme udviklet til at være effektiv i hardware, hvilket har gjort den forholdsvis langsom i software implementeringer.
  - Nøgle størrelse: 64 bits.
  - Blok størrelse: 64 bits.
- 3DES – Triple Data Encryption Standard, er en forstærket version af DES algoritmen. Denne fungerer ved at køre klarteksten igennem DES algoritmen 3 gange. Algoritmen benytter en større nøgle end DES, men samme blok størrelse og er derud en forholdsvis langsom algoritme, da denne skal køre DES algoritmen 3 gange.
  - Nøgle størrelse: 128, 192 bits (default 192).
  - Blok størrelse: 64 bits.
- RC2 – Rivest Cipher, anses for at være en forholdsvis god krypterings algoritme. Denne algoritme skulle være omkring 2 gange så hurtig som DES algoritmen.
  - Nøgle størrelse: 40 -128 bits (default 128).
  - Blok størrelse: 64 bits.
- AES – Advanced Encryption Standard (Rijndael), er en af de nyere algoritmer, der blev gjort til Federal Information Processing Standard (FIPS) af den amerikanske regering i 2001. Dermed er denne blevet anerkendt som en af de mest sikre algoritmer.

[7] Reference: [2] s. 61
[8] Reference: [6][7][8][9]

- Nøgle størrelse: 128, 192, 256 bits (default 256).
- Blok størrelse: 128, 192, 256 bits (default 128).

Da sikkerheden i systemet er prioriteret højt vil AES og 3DES algoritmen være mulige valg, dog med AES som den mest sikre, da denne har den længste nøgle. Valget mellem de 2 algoritmer vil afhænge af krypteringshastigheden for den enkelte algoritme (se 6.1).

4.3.3. Nøgle administrering

I forbindelse med indførelsen af en krypteringsalgoritme, er et af de vigtigste elementer for at opretholde sikkerheden, administreringen af krypteringsnøglerne. Holdes der ikke styr på de forskellige nøgler, vil sikkerheden i systemet blive svagere eftersom, at nøglerne kan havne i 3. parters hænder. De vil derefter have ubegrænset adgang til systemet, og dette vil sandsynligvis ikke blive opdaget, da der ikke forefindes nogen strukturering af de udsendte nøgler.

I det efterfølgende er der blevet specificeret et eksempel på et scenario til administrering af de forskellige krypteringsnøgler. Dette er blevet opstillet i overensstemmelse med risikoanalysen (se 3.2.3), og baserer sig på at finde en løsning, der tilbyder et godt kompromis mellem enkelthed og sikkerhed.

Løsningsmodellen bygger på at al administration af krypteringsnøgler foregår i hovedcentralen. Herfra styrer de generering af nye nøgler, samt at opdatere de enkelte stationers nøgler via fjernparametrering (via SMS). Modellen går ud på at man ved lanceringen af det nye produkt (PDA Discman) generer en specifik nøgle til alle virksomhedens DISCOS systemer. Denne vil udgøre virksomhedens basis nøgle og vil samtidig blive udsendt til de enkelte medarbejdere, der er ansvarlig for vedligeholdelsen af systemerne. Disse vil på den måde have adgang til alle virksomhedens stationer. Alle udsendte nøgler vil blive registreret sammen med den pågældendes PDA. Nøglerne vil bare kunne blive sendt ud via virksomhedens interne mails, da der ikke foreligger nogen høj risiko prioritering.

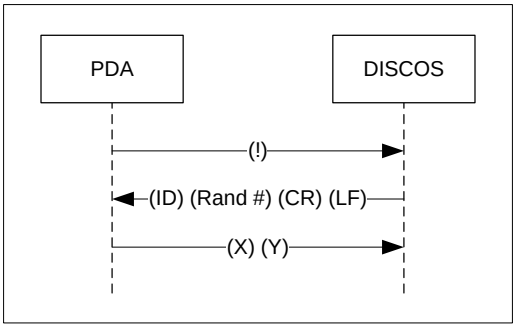
I tilfælde af at en elektriker eller en servicearbejder udefra skal have adgang til en specifik station, for at foretage nogle målinger og/eller ændringer af systemet, vil det være muligt at generere en ny nøgle, der sendes til den specifikke station, samt den person som skal have adgang. På den måde vil den udefra kommende arbejder kun have adgang til den specifikke station. Så snart kontrol besøget er afsluttet vil centralen sende basis nøglen tilbage til stationen. I tilfælde af at der pludselig begynder at forekomme nogle mistænkelige beskeder fra en eller flere DISCOS systemer, eller et antal af PDA'er er forsvundet vil centralen så generer en ny basis nøgle og udsende denne til alle stationer og medarbejdere. Dette kunne også indføres som fast rutine at nøglerne udskiftes hver måned.

4.4. Protokol

Dette afsnit beskriver sammenhængen for kommunikationen mellem PDA applikationen og DISCOS kommunikations emulatoren. Der tages udgangspunkt i initialiseringen af kommunikationen, hvorefter protocol namespace'et for PDA applikationen og for DISCOS kommunikations emulatoren vil blive beskrevet i nærmere detaljer.

4.4.1. Netværks Initialisering

For at kunne oprette forbindelse til DISCOS systemet skal der først gennemgås en initialiseringssekvens, der sikre at den fundne enhed reelt er en DISCOS kommunikationsenhed, og at den benyttede PDA har adgang til systemet. Nedenfor ses et sekvensdiagram der illustrerer denne opstart sekvens (alle værdier er på en størrelse af 8 bit).



Figur 23 - Initialiseringssekvens

Denne sekvens består i at der fra PDA'ens side først sendes et ASCII '!' (21h), hvilket åbner kommunikationen mellem PDA'en og DISCOS kommunikationsenheden.
DISCOS enheden returnere derefter en initialiseringsbesked der består af følgende:

- ID: CAN ID'et på kommunikationsenheden (03h).
- Rand #: En tilfældig 8 bit værdi.
- CR: Carriage Return (0Dh).
- LF: Linefeed (0Ah).

Disse 4 bytes er krypteret til en 16 bytes blok ved hjælp af kommunikationsenhedens hemmelige nøgle. Dette sikre at PDA'en har adgang til det specifikke DISCOS system, da beskeden skal dekrypteres med den samme hemmelige nøgle. For mere information omkring krypteringen se 4.3.

Herefter returnerer PDA'en en checksum der skal opfylde følgende kriterium:

- X + Y = Rand #.

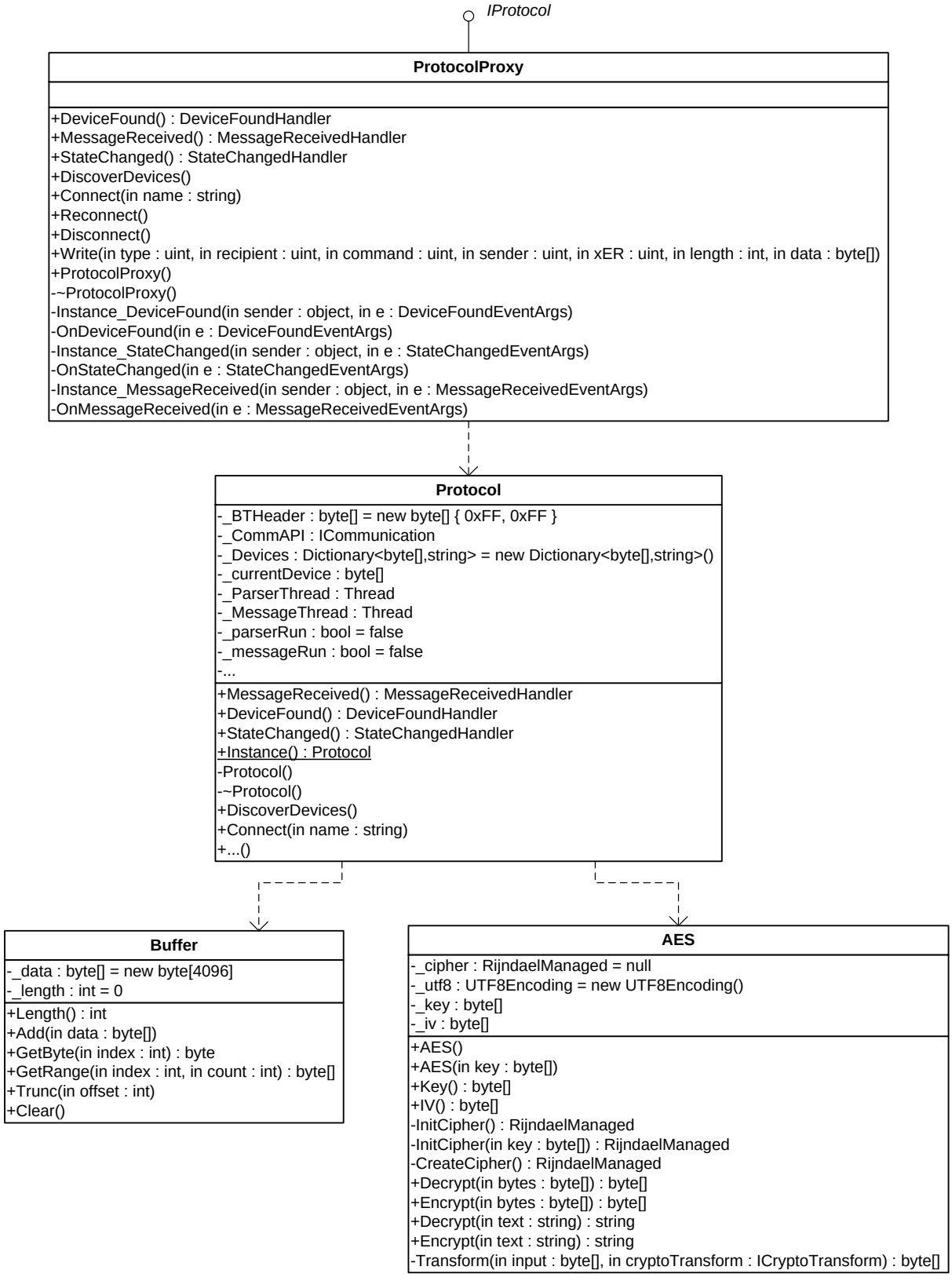
Selv med kendskab til initialiseringssekvensen, vil der uden den hemmelige nøgle kun være omkring 0,39 % chance for at returnere den rigtige checksum. Såfremt den rigtige checksum returneres vil al kommunikation mellem PDA'en og DISCOS enheden derefter være krypteret.

Derudover skal initialiserings timingen overholdes, hvilket medfører at PDA'en skal returnere checksummen inden for 1 sekund. Dette skal sørge for at DISCOS systemets kommunikationsenhed går tilbage til idle tilstand, hvis initialiseringssekvensen blev startet tilfældigt.

Efter at initialiserings proceduren er blevet gennemført succesfuldt vil det nu være muligt at kommunikere efter den specificerede protokol.

4.4.2. Discman PDA

Dette afsnit specificerer opbygningen af Protocol namespace'et i forhold til PDA applikationen.



Figur 24 - Klassediagram: Protokollaget (Discman PDA)

ProtocolProxy

Denne klasse benyttes som interface klasse, i forbindelse med et Proxy pattern, til Protocol namespace'et. Dette gør at namespace'et kommer til at være mere modulært, og gør det derved mere enkelt at kunne udskifte dette lag. Det har været nødvendigt at oprette en specifik interface klasse, da Protocol klassen er blevet implementeret med et singleton pattern.

Buffer

Buffer klassen udgør en datacontainer, hvori alt modtaget data gemmes til det kan blive behandlet.

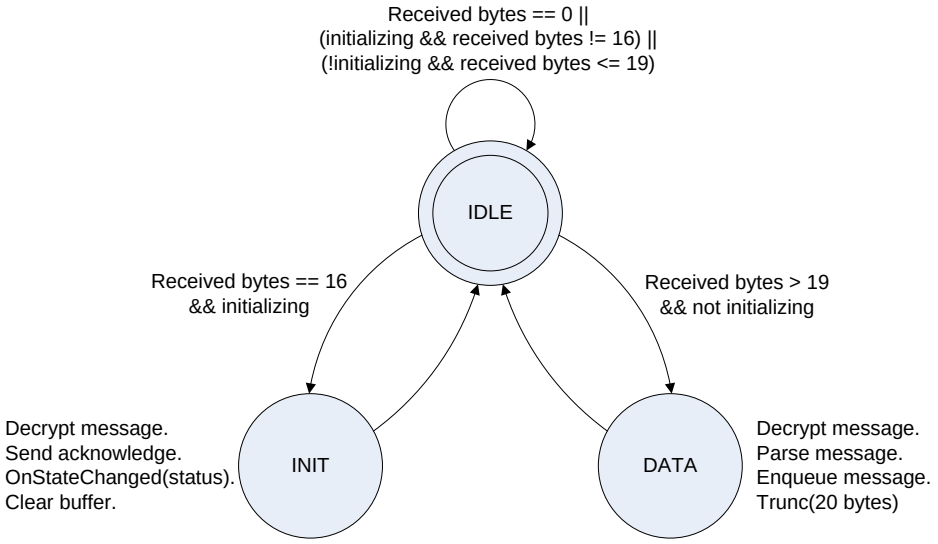
AES

Denne klasse udgør implementeringen af krypteringsalgoritmen i forhold til *Advanced Encryption Standard.* Denne klasse er ansvarlig for at kryptere alle udgående beskeder, samt at dekryptere alle modtagede beskeder.

Protocol

Denne klasse repræsenterer selve kommunikations protokollen. Klassen er ansvarlig for at håndtere alle ind- og udgående beskeder til/fra DISCOS systemet i forhold til den opstillede protokol (se 9.4.2). Al kommunikation styres ud fra en NetworkStream, der modtages fra kommunikationslaget ved initialiseringen af systemet. Klassen holder desuden styr på besked synkroniseringen ved, at sammenligne de modtagede besked numrene med klassens indbyggede tæller. Denne tæller bruges til såvel ind- og udgående beskeder og nulstilles såfremt forbindelsen går tabt.

For at parse de modtagede data køres der en parser tråd i baggrunden, der forløber efter følgende tilstandsdiagram:



Figur 25 - Parser tilstands diagram

IDLE

I denne tilstand undersøges der først om der er modtaget nogle data bytes. Er dette tilfældet undersøges der om systemet er i en initialiseringsfase, og at der er modtaget præcis 16 bytes svarende til en initialiseringspakke (INIT) eller om systemet er blevet initialiseret, og at der er modtaget mere end 19 bytes svarende til en datapakke (DATA). Er ingen af tilfældene opfyldt undersøges der om der er blevet modtaget nogle nye bytes og tilstanden køres forfra.

INIT

I denne tilstand undersøges det først om den modtagede pakke kan krypteres i henhold til den eksisterende nøgle. Kan dette ikke lade sig gørem er det en indikation af, at systemet er i besiddelse af en forkert krypteringsnøgle. Dette medfører, at der ikke er adgang til det valgte DISCOS system og tilstandsmaskinen afsluttes. Kan den modtagede pakke derimod krypteres, undersøges der om den første byte er lig med 0x03, svarende til en DISCOS kommunikationsenhed, og at byte 3 og 4 udgør 0x0D0A, svarende til carriage return. Er disse kriterier opfyldt genereres en acknowledge besked (se 4.4.1), der sendes retur til DISCOS systemet. Til sidst slettes alle elementer i bufferen.
Tilstandsmaskinen returneres til IDLE tilstanden.

DATA

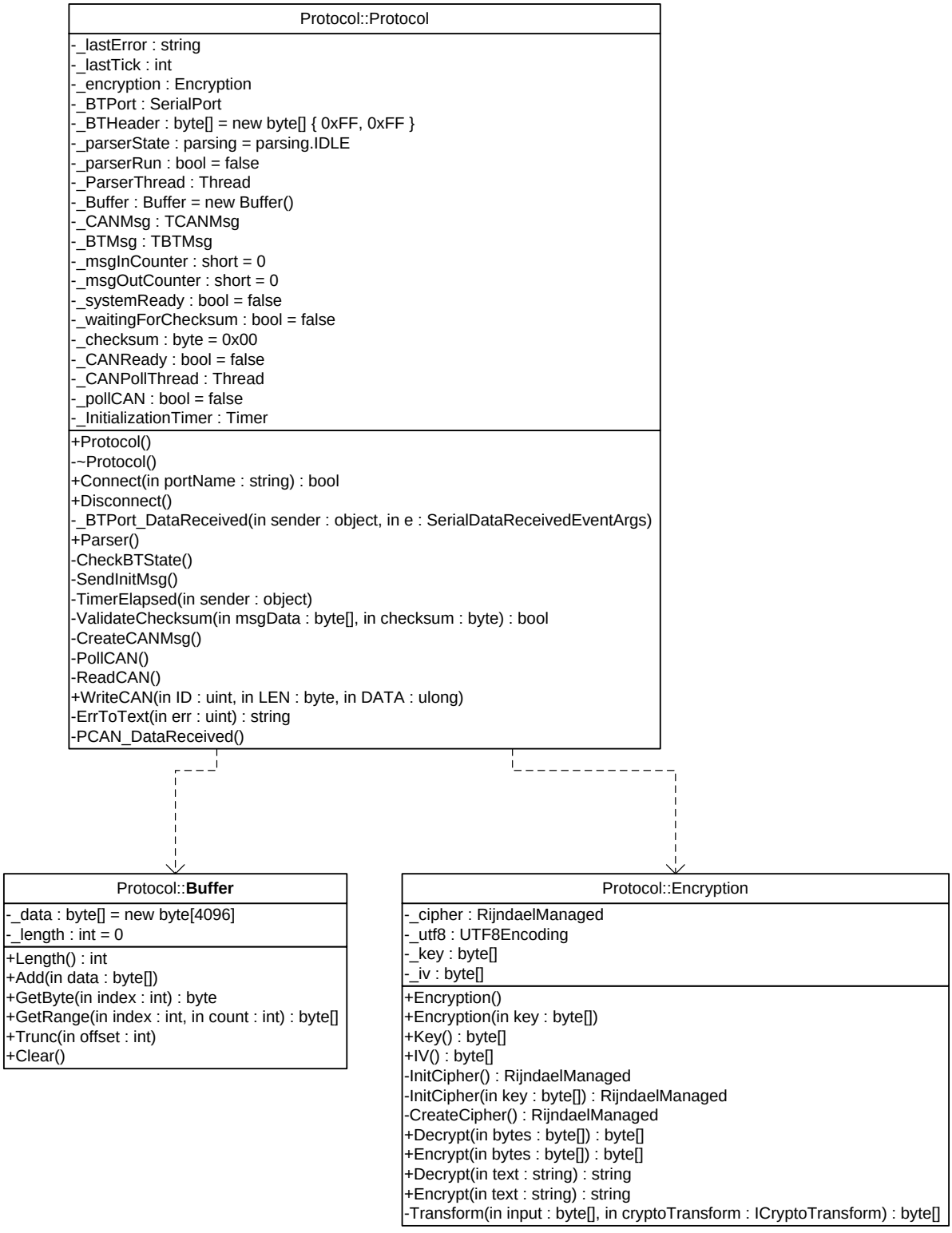
I denne tilstand undersøges der først om de 2 første bytes er lig med 0xFFFF. Er dette tilfældet undersøges der om den modtagede pakke kan dekrypteres i henhold til den eksisterende nøgle. Kan dette ikke lade sig gøre, er det en indikation af, at systemet er i besiddelse af en forkert krypteringsnøgle. Dette medfører, at der ikke er adgang til det valgte DISCOS system og tilstandsmaskinen afsluttes. Kan den modtagede pakke derimod krypteres, tjekkes pakkens checksum. Svare pakkens checksum overens med de modtagede data parses beskeden, hvorefter den tilføjes til besked køen. Efterfølgende slettes bufferens 20 forreste bytes. Er headeren forkert eller passer checksummen ikke overens med beskeden fjernes den første byte. Tilstandsmaskinen returneres til IDLE tilstanden.

I bilag 9.4.1 er der et detaljeret flowchart over Parser tilstandsmaskinen.

Ud over en parser tråd består Protocol namespace'et også af en besked tråd, der sørger for at udtage beskeder fra køen og sende disse videre til de observer objekter, der har registreret sig til Protocol klassens MessageReceived event.

4.4.3. DISCOS kommunikations emulator

Dette afsnit specificerer opbygningen af Protocol namespace'et i forhold til DISCOS kommunikations emulatoren på PC'en.



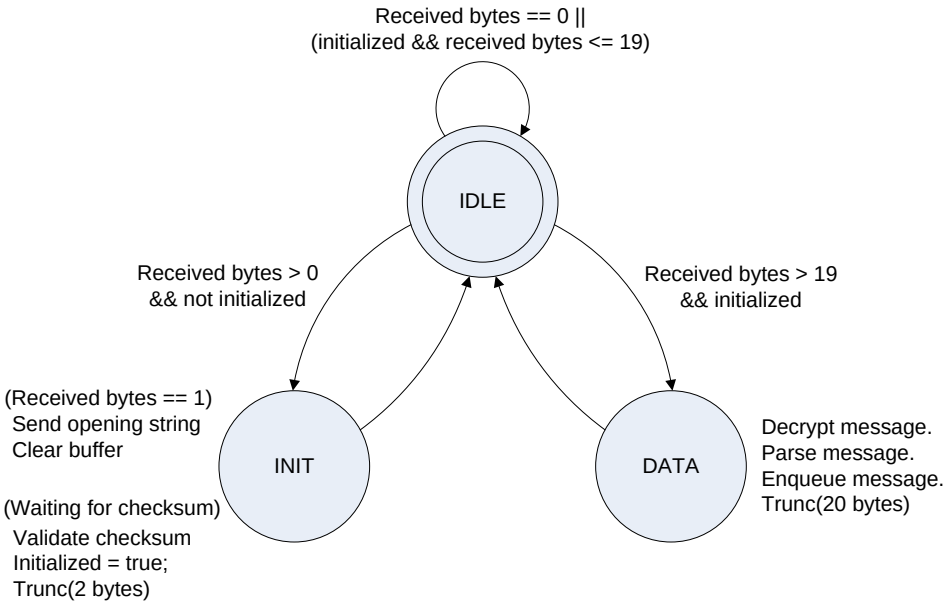
Figur 26 - Klassediagram: Protokollaget (DISCOS kommunikations emulator)

Buffer
Buffer klassen udgør en datacontainer, hvori alt modtaget data gemmes til det kan blive behandlet.

Encryption
Denne klasse udgør implementeringen af krypteringsalgoritmen i forhold til *Advanced Encryption Standard.* Denne klasse er ansvarlig for at kryptere alle udgående beskeder, samt at dekryptere alle modtagede beskeder.

Protocol
Denne klasse repræsenterer selve kommunikations protokollen. Klasse er ansvarlig for at håndterer alle ind- og udgående beskeder til/fra DISCOS systemet i forhold til den opstillede protokol (se 9.4.2). Al kommunikation styres ud fra en NetworkStream der modtages fra kommunikationslaget ved initialiseringen af systemet. Klassen holder desuden styr på besked synkroniseringen ved, at sammenligne de modtagede besked numrene med klassens indbyggede tæller. Denne tæller bruges til såvel ind- og udgående beskeder og nulstilles såfremt forbindelsen går tabt.

For at parse de modtagede data, køres der en parser tråd i baggrunden, der forløber efter følgende tilstandsdiagram:



Figur 27 - Parser tilstands diagram

IDLE
I denne tilstand undersøges der først om der er modtaget nogle data bytes. Er dette tilfældet undersøges der om kommunikationen er blevet initialiseret, og at der er modtaget mere end 19 bytes svarende til en datapakke (DATA) eller om kommunikationen stadig ikke er initialiseret (INIT). Er ingen af tilfældene opfyldt køres tilstanden forfra. Er der ikke blevet modtaget nogen bytes og kommunikationen er blevet initialiseret, undersøges der om der stadig er forbindelse til Bluetooth enheden, inden tilstanden køres forfra.

INIT

I denne tilstand undersøges det først om der er modtaget præcis en byte, eller om initialiseringen af kommunikation er blevet startet.

Er der blevet modtaget præcis en byte, undersøges der om denne er lig 0x21. Er dette tilfældet startes initialiseringen, og der sendes en åbning kommunikations streng til PDA'en. Til sidst slettes alle elementer i bufferen.

Er initialiseringen derimod blevet startet, hvilket indikerer at der ventes på en checksum, valideres den modtagede checksum. Er denne valid er kommunikationen blevet initialiseret og de 2 bytes checksum fjernes fra bufferen. Er checksummen derimod ikke valid slettes alle elementer fra bufferen.

Tilstandsmaskinen returneres til IDLE tilstanden.

DATA

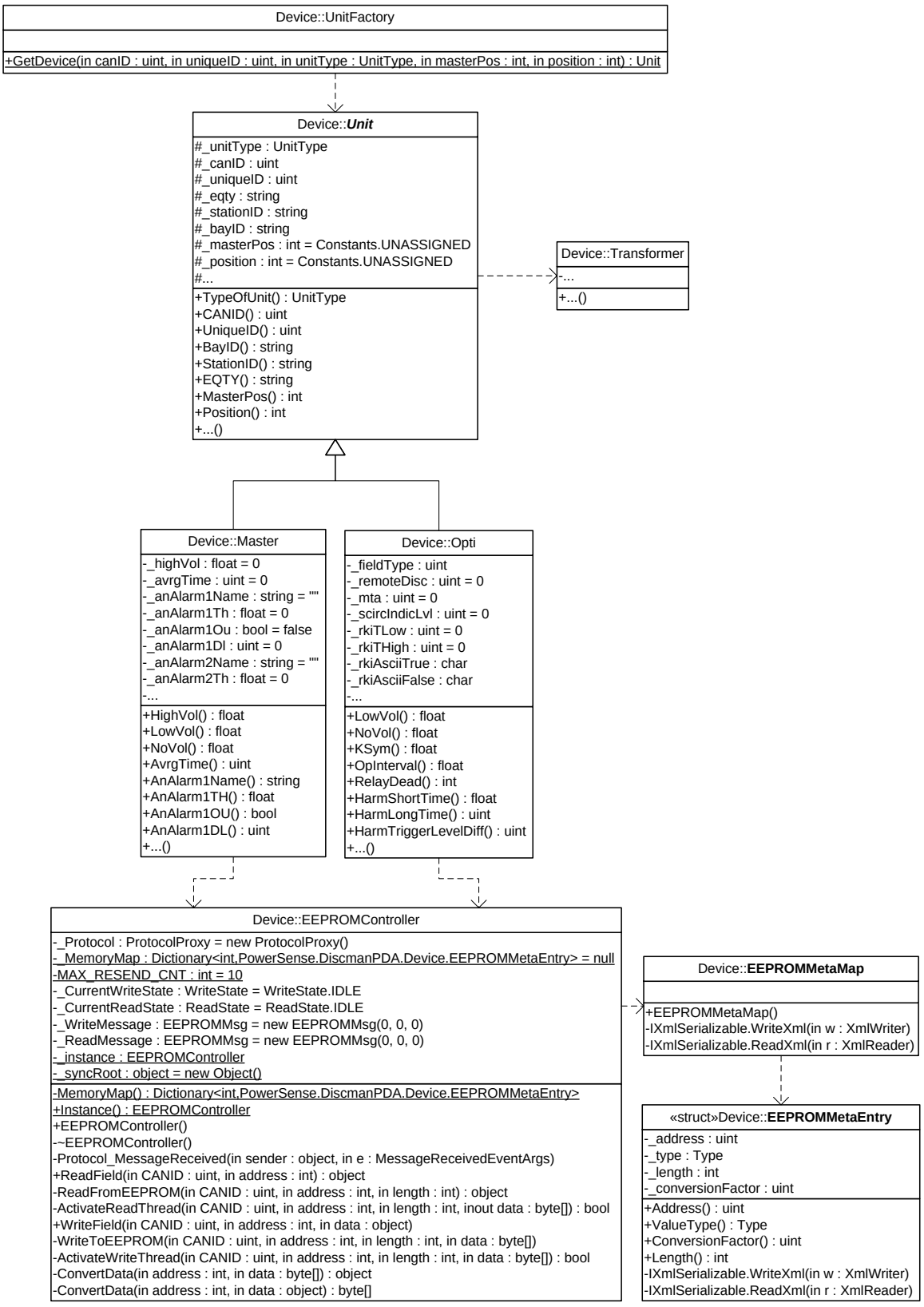
I denne tilstand undersøges der først om de 2 første bytes er lig med 0xFFFF. Er dette tilfældet undersøges der om den modtagede pakke kan dekrypteres, i henhold til den eksisterende nøgle. Kan dette ikke lade sig gøre, er det en indikation af, at systemet er i besiddelse af en forkert krypteringsnøgle. Dette medfører, at der ikke er adgang til det valgte DISCOS system og tilstandsmaskinen afsluttes. Kan den modtagede pakke derimod krypteres, tjekkes pakkens checksum. Svare pakkens checksum overens med de modtagede data parses beskeden, hvorefter den tilføjes til besked køen. Efterfølgende slettes bufferens 20 forreste bytes. Er headeren forkert eller passer checksummen ikke overens med beskeden fjernes den første byte. Tilstandsmaskinen returneres til IDLE tilstanden.

I bilag 9.4.1 er der et detaljeret flowchart over Parser tilstandsmaskinen.

Ud over en parser tråd består Protocol namespace'et også af en besked tråd, der sørger for at udtage beskeder fra køen og sende disse videre til DISCOS enhederne via CAN bussen.

4.5. Device

Dette afsnit beskriver opbygningen af Device namespace'et.



Figur 28 - Klassediagram: Enhedslaget.

4.5.1. UnitFactory
Denne klasse implementerer et Factory pattern og er ansvarlig for oprettelsen af objekter til forskellige enhedsklasser. Factory klassen identificerer den rigtige klasse ud fra enhedstype parameteren.

4.5.2. Unit
Dette er en basis klasse for alle enhedsklasser, der indeholder alle de fælles indstillinger og metoder for de enkelte enheder. Denne klasse arves af alle de andre enhedsklasser.

4.5.3. Master
Denne klasse udgør en software repræsentation af en Master enhed. Denne er ansvarlig for håndtering af alt DISCOS kommunikation mellem den reelle Master enhed og PDA applikationen. Dette indebærer at hente og opdatere de relevante enheds indstillinger, samt at håndtere målings procedure. Klassen arver Unit klassen for, at få adgang til enhedens basis informationer og benytter sig af EEPROMController klassen til at tilgå enhedens EEPROM.

4.5.4. Opti
Denne klasse udgør en software repræsentation af en Opti enhed. Denne er ansvarlig for håndtering af alt DISCOS kommunikation mellem den reelle Opti enhed og PDA applikationen. Dette indebærer at hente og opdatere de relevante enheds indstillinger, samt at håndtere målings procedure. Klassen arver Unit klassen for, at få adgang til enhedens basis informationer og benytter sig af EEPROMController klassen til at tilgå enhedens EEPROM.

4.5.5. EEPROMController
Denne klasse udgør en handler klasse til håndtering af beskeder vedrørende læsning fra og skrivning til den enkelte enheds EEPROM. Denne implementerer et Singleton pattern, der gør at der kun sendes EEPROM beskeder ud på CAN bussen til en enhed ad gangen. For at sikre at data sendt til og fra enhedens EEPROM bliver håndteret korrekt, i såvel software som i firmware, benytter klassen sig af EEPROMMetaMap klassen.

4.5.6. EEPROMMetaMap
Denne klasse implementerer et meta map over de forskellige EEPROM adresser. Klassen udgør et indekseret dictionary over EEPROMMetaEntry objekter, der specificerer håndteringen af data svarende til de enkelte adresser i EEPROM'en. Dette meta map indeholder informationer omkring data type, størrelse af værdi og konverteringsfaktoren mellem softwaren og firmwaren (se 9.5). Under bilag forefindes der nogle flowcharts der illustrerer, hvordan denne konvertering forgår (se Meta Map Konvertering 9.5.1).

Indlæsningen af dette meta map foregår ved initialiseringen af klassen, og udføres ved at læse de enkelte elementer fra en XML fil, ved hjælp af deserialization. Dette gør det enkelt at foretage ændringer i forbindelse med, at der ændres i EEPROM arkitekturen, og sikre at der ikke skal hardcodes en mængde statisk data i applikationen.

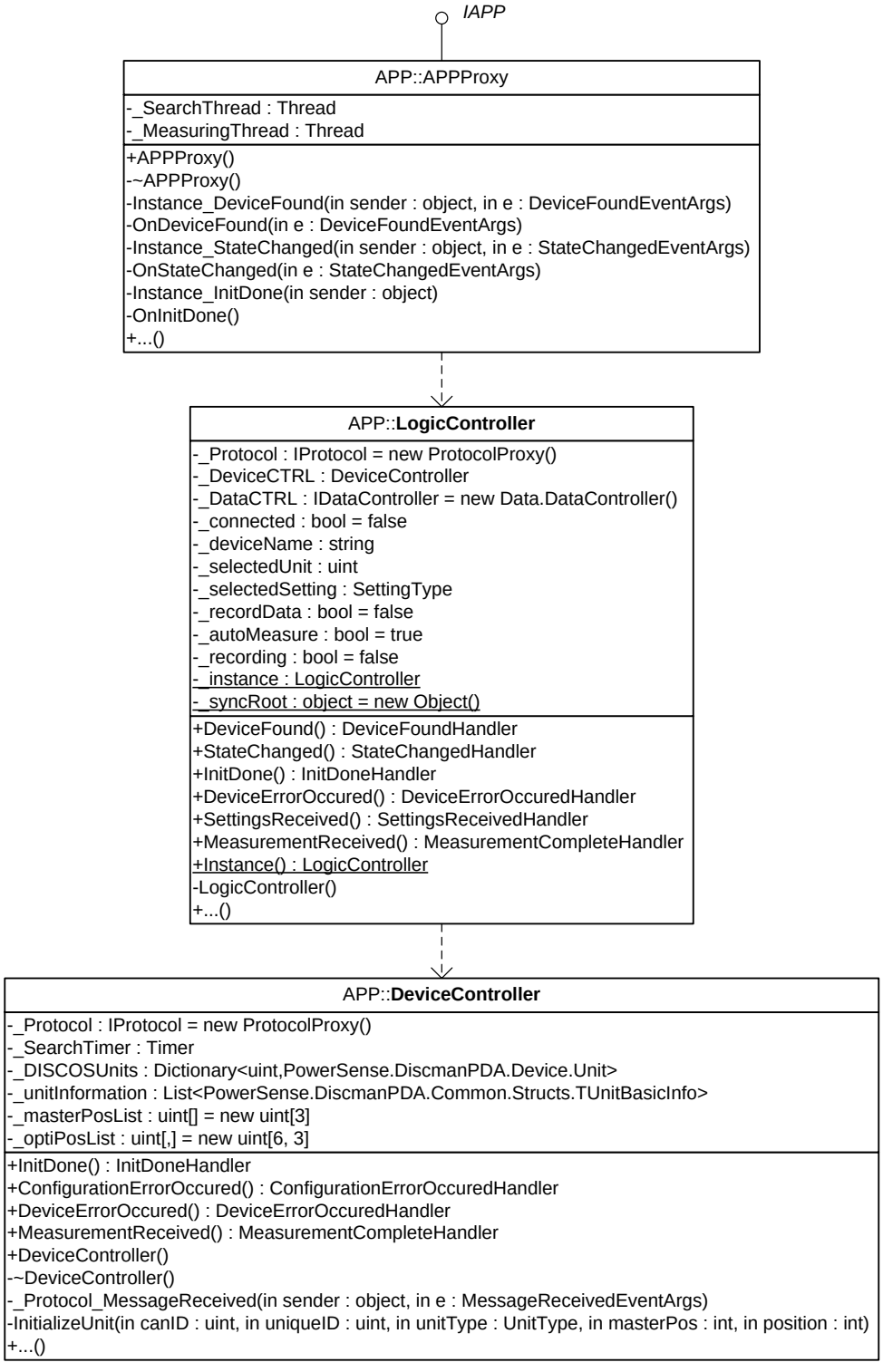
4.5.7. EEPROMMetaEntry
Denne struktur indeholder data omkring de enkelte elementer i EEPROMMetaMap. Denne består af:

- Hukommelses adresse.
- Data type (int, float, string osv.).
- Konverteringsfaktor (mellem software og hardware).
- Antal data bytes.

Denne struktur implementer desuden XMLSerializable interfacet, der specificerer hvordan de enkelte data skal læses/skrives til en XML fil.

4.6. APP

Dette afsnit beskriver opbygningen af APP namespace'et med henhold til de forskellige klasser, der indgår i dette.



Figur 29 - Klassediagram: Applikationslaget.

4.6.1. APPProxy

Denne klasse benyttes som interface klasse, i forbindelse med et Proxy pattern, til APP namespace'et. Dette gør at namespace'et kommer til at være mere modulært, og gør det derved mere enkelt at kunne udskifte dette lag. Det har været nødvendigt at oprette en specifik interface klasse, da LogicController klassen er blevet implementeret med et singleton pattern. Derudover benyttes denne klasse til at adskille GUI tråden fra logiklaget, hvilket sikre at den grafiske brugerflade ikke låser, mens der udføres logiske operationer.

Tråd håndteringen foregår ved, at der ved hver GUI forespørgsel oprettes en ny tråd, der udfører den ønskede handling. Den nye tråd kører indtil at den logiske operation er afsluttet, hvorefter den afsluttes. En af de underliggende tråde sørger derefter for at påkalde GUI tråden, så den kan opdatere displayet.

4.6.2. LogicController

Denne klasse udgør den centrale del af applikationen, da den er ansvarlig for hele program-flow'et. Dette gøres ved, at klassen implementerer et Facade pattern der sammenkobler de enkelte GUI forespørgsler med de relevante systemressourcer. Derudover informerer klassen GUI laget om relevante hændelser, der medfører opdateringer af den grafiske brugerflade.

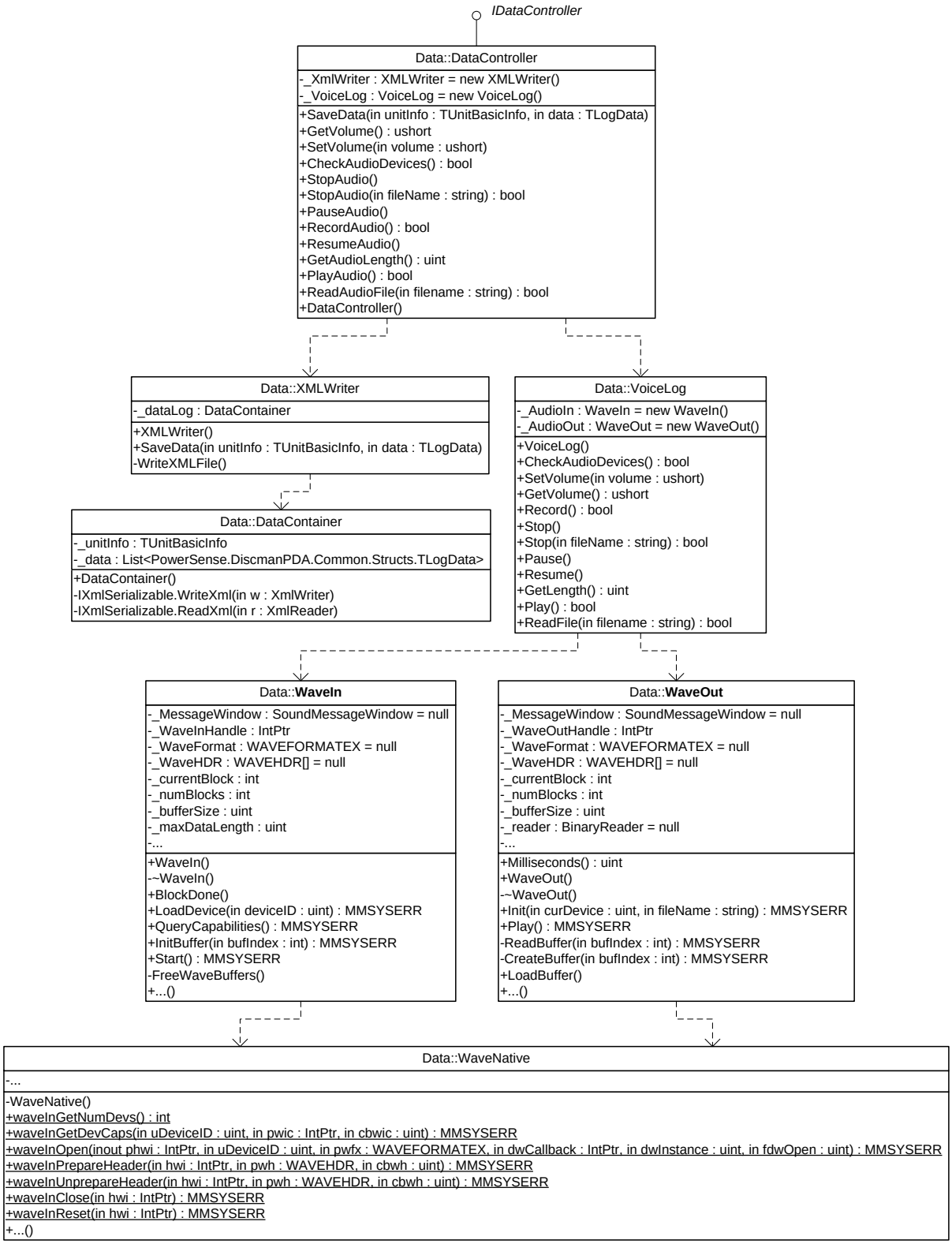
4.6.3. DeviceController

Denne klasse er ansvarlig for håndteringen af alle de enheder der er tilsluttet et DISCOS system. Dette indebærer:

- At søge og registrere de enkelte enheder ved opstart.
- At administrere forespørgsler til de forskellige enheder.
- At administrere modtagede informationer fra de forskellige enheder.

4.7. Data

Dette afsnit beskriver opbygningen af DATA namespace'et med henhold til de forskellige klasser der indgår i dette.



Figur 30 - Klassediagram: Datalaget

4.7.1. DataController
Denne klasse implementerer et Facade pattern og er ansvarlig for at sammenkoble forespørgsler fra Applikationslaget med de relevante systemressourcer i datalaget.

4.7.2. XMLWriter
Denne klasse er ansvarlig for at skrive alle modtage data til en XML fil. Dette gøres ved først at oprette et DataContainer objekt, hvori alt data bliver gemt. Dernæst bliver denne serialized, hvilket skriver informationerne omkring DataContainer objektet til en XML fil.

4.7.3. DataContainer
Denne klasse udgør en data beholder for enhedsinformation, samt målte data.

Klassen implementerer desuden XMLSerializable interfacet, der specificerer hvordan de enkelte data skal læses/skrives til en XML fil.

4.7.4. VoiceLog
Denne klasse udgør et overordnet API til at tilgå audio input/output funktionalitet. Denne klasse giver mulighed for at benytte følgende audio funktionaliteter:

- Optagelse af lyd, via mikrofonen, til en wav fil.
- Afspilning af wav lydfil (play, pause og stop).

4.7.5. WaveIn
Denne klasse udgør et API til audio input enheden. Denne består af alle metoder og strukturer der benyttes til håndteringen af enheden. Klassen giver adgang til følgende funktionaliteter:

- Søgning efter audio input enheder.
- Undersøgning af understøttelse til lyd format.
- Optagning lyd.
- Gemme optaget lyd data til wav fil.

4.7.6. WaveOut
Denne klasse udgør et API til audio output enheden. Denne består af alle metoder og strukturer der benyttes til håndteringen af enheden. Klassen giver adgang til følgende funktionaliteter:

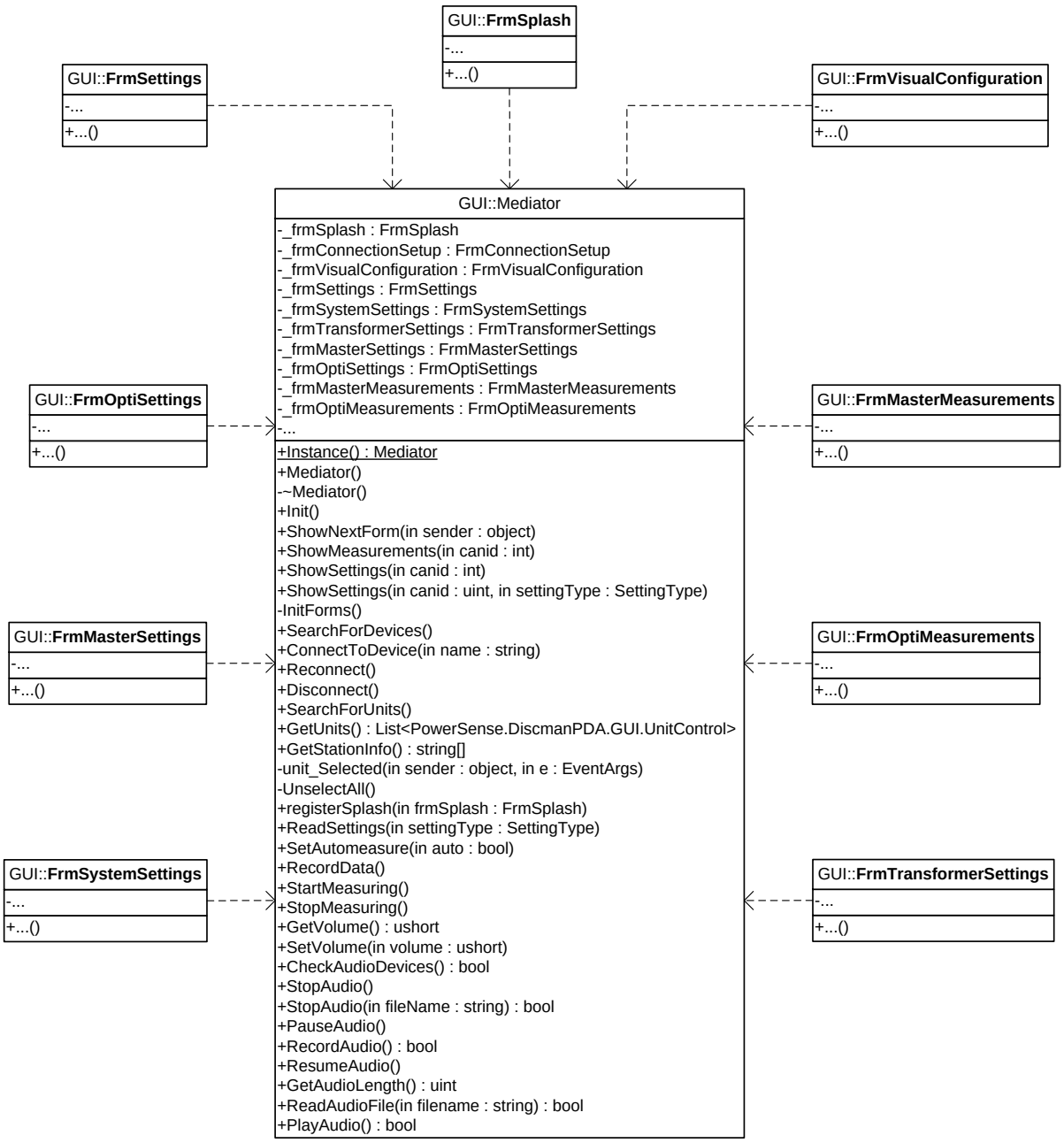
- Søgning efter audio output enheder.
- Afspilning af wav fil (herunder pause og stop muligheder).

4.7.7. WaveNative
Denne klasse håndterer al tilgang til de ”native” OS indlejrede metoder der benyttes i forbindelse med brugen af audio I/O enheder. Dette er primært i forbindelse med optagning og afspilning af wav filer. Endvidere indgår alle de strukturer og enumerators der benyttes til, at konvertere data fra unmanaged til managed også i denne klasse. Gennem denne klasse er det dermed muligt at få tilgang til følgende funktionaliteter:

- Håndtering audio input enheder (mikrofon), i forbindelse med optagning af lyd.
- Håndtering audio output enheder, i forbindelse med afspilning af wav filer.

4.8. GUI

Dette afsnit beskriver opbygningen af GUI namespace'et med henhold til de forskellige klasser, der indgår i dette.



Figur 31 - Klassediagram: GUIlaget

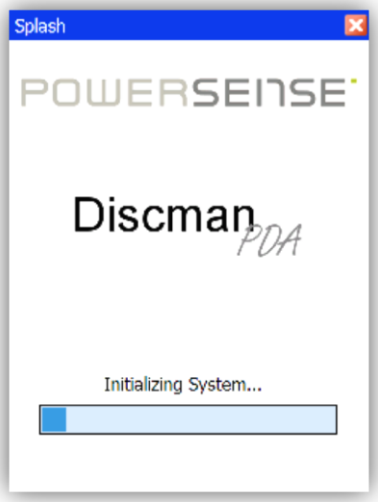
4.8.1. Mediator

Denne klasse er designet efter Mediator pattern'et, og udgør knudepunktet mellem de enkelte forms og applikationslaget. Ved opstart registrerer FrmSplash formen sig i Mediatoren, hvorefter at Mediator klassen overtager styringen af brugergrænsefladen. Dette gøres ved at klassen opretter instanser af alle de resterende forme for at til og frakoble events til de forme i forhold til, at de får og mister fokus. Derudover indeholder klassen også de data, der er fælles for de individuelle forms.

4.8.2. Forms

Dette afsnit beskriver kort de enkelte vinduers/form's funktion.

FrmSplash



Figur 32 - Screenshot: Splash vinduet

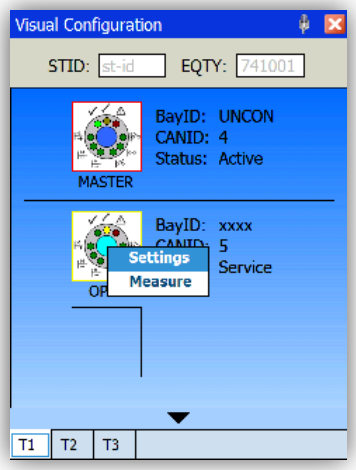
Splash formen er applikationens opstartsskærm og dermed indgangspunktet til systemet. Formen starter initialiseringen af systemet ved at påbegynde søgningen efter Bluetooth enheder.

FrmConnectionSetup

Figur 33 - Screenshot: Connection Setup

Connection Setup formen benyttes til at søge efter Bluetooth enheder for dernæst at oprette forbindelse til en disse.
Søge tiden varierer efter, hvor mange Bluetooth enheder der er indenfor rækkevidde. Såfremt den ønskede enhed er blevet vist på listen, kan der oprettes forbindelse til denne også selvom, at søgningen ikke er afsluttet.

FrmVisualConfiguration

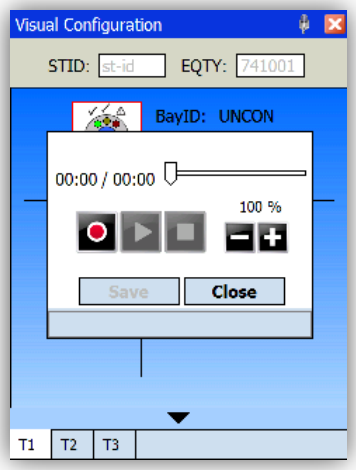


Figur 34 - Screenshot: Audio Center

Visual Configuration formen illustrerer DISCOS systemets konfiguration med henblik på tilsluttede Master og Opti enheder.

Formen søger efter DISCOS enheder, og der oprettes nogle grafiske UserControls, der repræsenterer de fundne enheder. Disse pladseres dernæst svarende til enhedernes positions indstillinger.

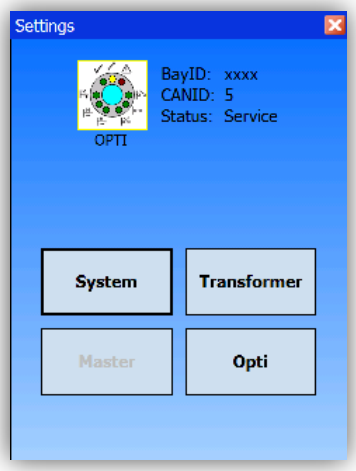
Såfremt der er fundet mindst en enhed er det muligt via en menu, at tilgå den enkelte enheds indstillinger eller foretage test målinger.



Figur 35 - Screenshot: Audio Center

Det er endvidere muligt via formen at tilgå audio centeret via mikrofon ikonet i top baren. Dette åbner et nyt panel, hvor der er mulighed for at optage en voice log, der senere kan blive ajourført.
Efter at der er blevet oprettet en voice log er det muligt at afspille denne før den gemmes til PDA'en.

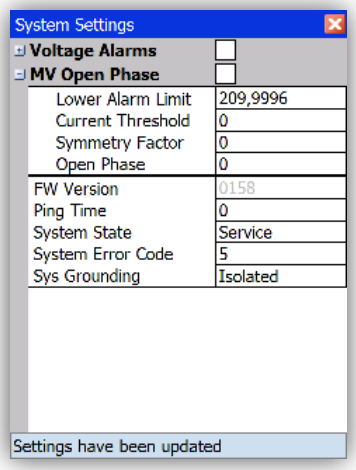
FrmSettings



Figur 36 - Screenshot: Settings

Settings formen benyttes til at vise de forskellige indstillingsgrupper svarende til den valgte enhed. Herfra kan man tilgå system, transformer og enhedens specifikke indstillinger.

FrmSystemSettings, FrmTransformerSettings, FrmMasterSettings, FrmOptiSettings

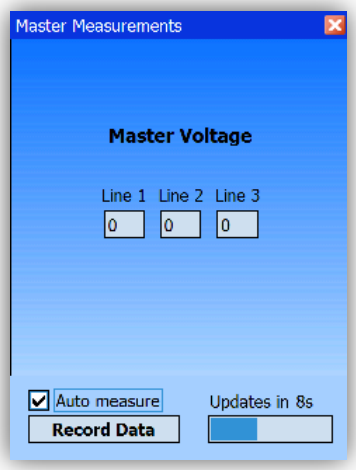


Figur 37 - Screenshot: System Settings

De forskellige Settings forme henter de relevante information fra enhedens EEPROM, og viser disse i den specifikke form.

Herfra vil det desuden være muligt at ændre på enhedens konfiguration.

FrmMasterMeasurements

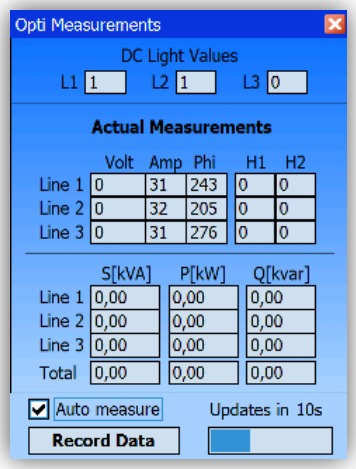


Figur 38 - Screenshot: Master Measurements

Master Measurement formen benyttes til at udføre test målinger af en Master enhed. Målingerne kan startes manuelt eller automatisk, hvilket vil gøre at systemet vil blive ved med at køre målinger igennem for dernæst at opdatere displayet.
Det er desuden muligt at optage alle målte data, i en målings cyklus, til en XML fil.

Der udføres kun målinger på Master'ens bus spænding.

FrmOptiMeasurements



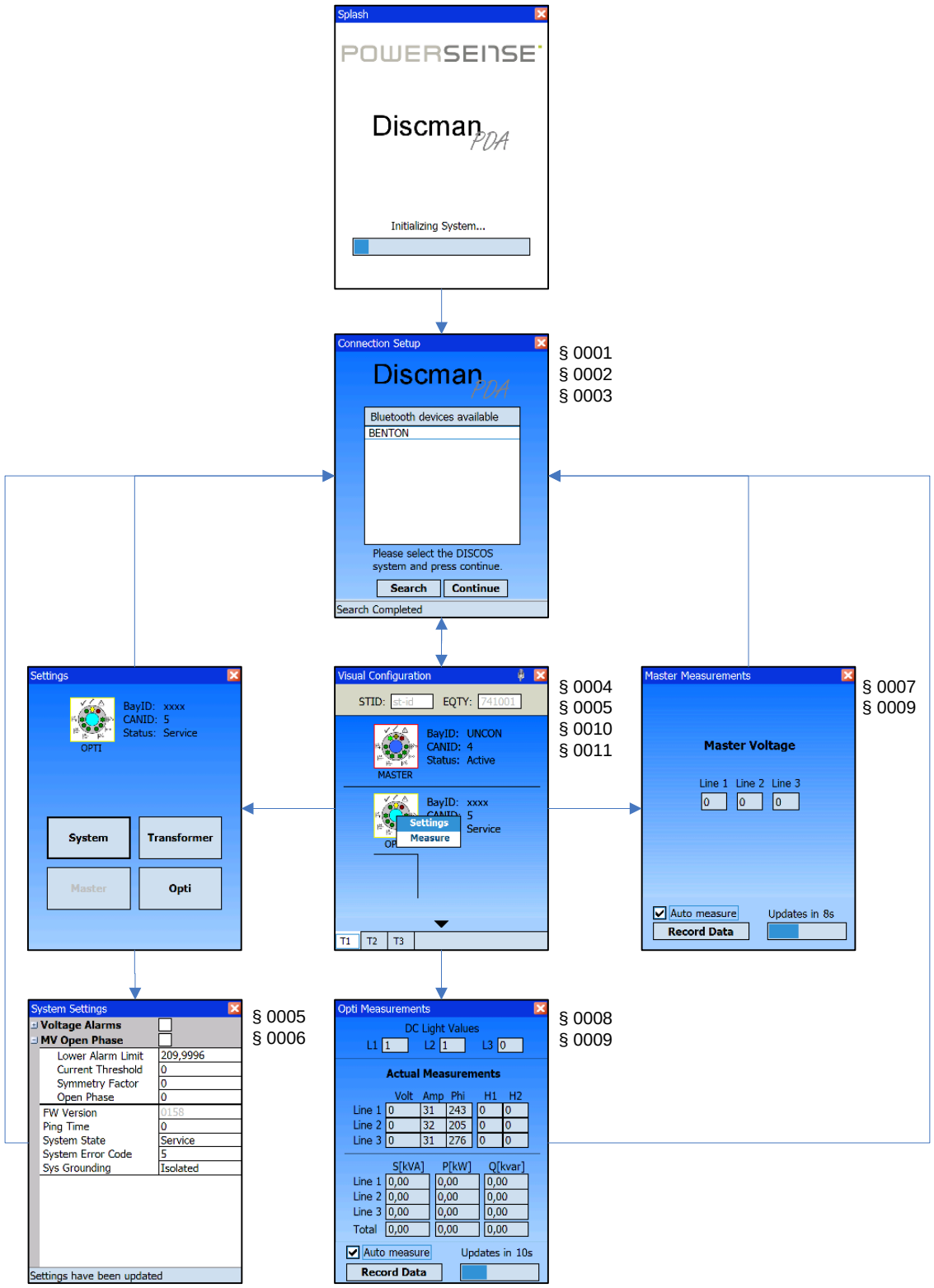
Figur 39 - Screenshot: Opti Measurements

Opti Measurement formen benyttes til at udføre test målinger af en Opti enhed. Målingerne kan startes manuelt eller automatisk, hvilket vil gøre at systemet vil blive ved med at køre målinger igennem for dernæst at opdatere displayet.
Det er desuden muligt at optage alle målte data, i en målings cyklus, til en XML fil.

Der udføres målinger på Opti'ens lys værdier, spænding, strøm, effekt, faseforskelle og harmoniske værdier.

4.8.3. GUI Map

Der er blevet designet følgende GUI map, der illustrerer sammenhængen mellem de enkelte forms. Desuden er use case id'erne blevet tilføjet til de enkelte forms i GUI map'et, for at indikere hvor de forskellige use cases benyttes.

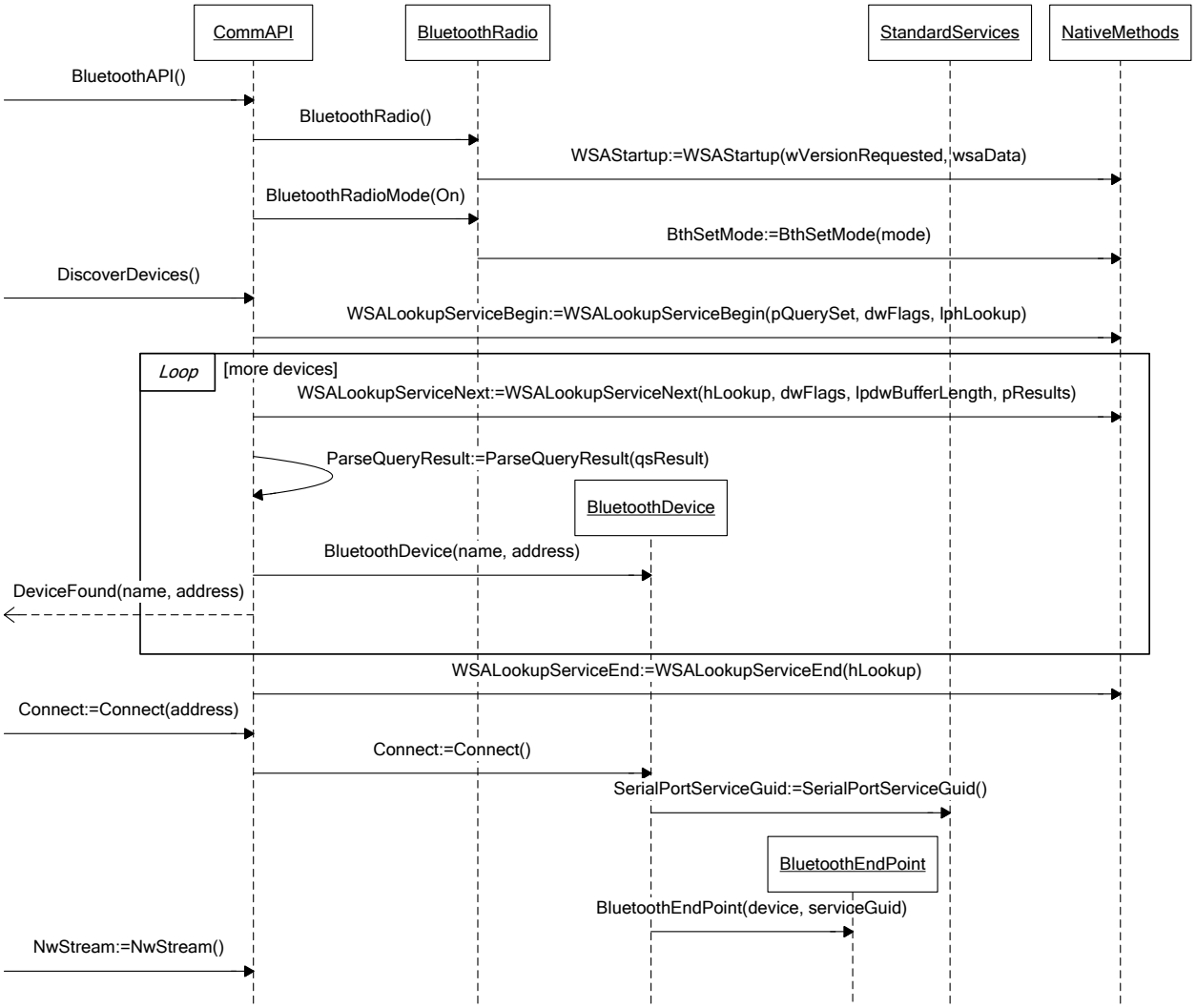


Figur 40 - GUI Map

4.9. Use Case Realisering

Dette afsnit indeholder sekvensdiagrammerne til de opstillede use cases. Heraf er 2 beskrevet i de efterfølgende underafsnit, mens de resterende er placeret under bilag 9.2.

§ 0001 – Opret forbindelse



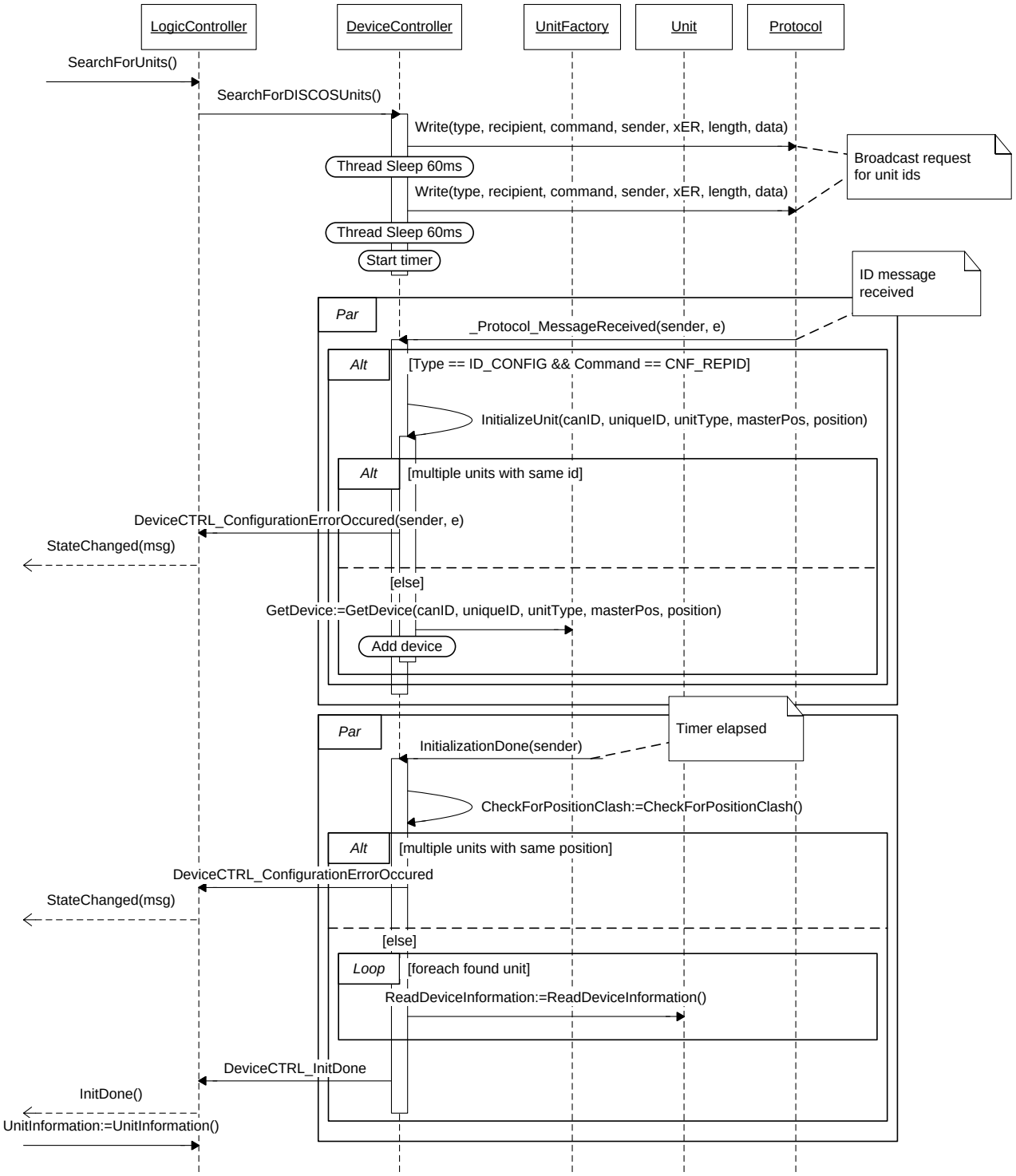
Figur 41 - Sekvensdiagram: Opret forbindelse

Dette sekvensdiagram illustrerer, hvordan der oprettes forbindelse til en Bluetooth enhed.

1. PDA'ens Bluetooth radio aktiveres.
2. Winsocket initialiseres.
3. Søger efter og identificerer Bluetooth enheder indenfor rækkevidde.
4. Brugeren vælger, hvilken enhed der skal oprettes forbindelse til.
5. Der oprettes forbindelse til den valgte enhed ud fra enhedens Bluetooth address.

For yderligere information omkring søgning efter Bluetooth enheder og oprettelse af forbindelse til disse, se 5.1.

§ 0004 – Søg efter DISCOS enheder



Figur 42 - Sekvensdiagram: Søg efter DISCOS enheder

Dette sekvensdiagram illustrerer søgningen efter DISCOS enheder.

1. Der udsendes broadcast beskeder til DISCOS systemet, der anmoder de enkelte enheder om at sende deres id.
2. Der oprettes en software repræsentation for hver af de fundne enheder.
3. Systemet sikre sig at der ikke er flere enheder med samme id og/eller position.

4.10. Delkonklusion

Gennem dette kapitel er der blevet udarbejdet nogle forskellige løsningsmodeller til de i analyse afsnittet belyste problemstillinger. Disse løsningsmodeller ligger vægt på at designe et struktureret system, der fokuserer på at uddelegere de enkelte systemfunktionaliteter til enkeltstående moduler, for på den måde at skabe et overordnet system design, der er overskueligt og enkelt at arbejde med. Endvidere er der blevet designet en endelig version af system arkitekturen, hvilket benyttes som udgangspunkt til de enkelte løsninger.

Designfasen har fokuseret på at realisere de opstillede use cases ved at konstruere sekvensdiagrammer og ved at opstille klassediagrammer for de enkelte lag i systemet. Disse klassediagrammer specificerer funktionen af de enkelte klasser og lag i systemet.

Sikkerhed

De forskellige krypterings algoritmer der understøttes af .NET framework'et er blevet gennemgået, med henblik på at finde den mest passende løsning til sikkerheds behovet i systemet. Der er blevet fokuseret på at finde en algoritme, der har den største sikkerhed, men som også kan opfylde de tidsmæssige krav til systemet. Det blev valgt at fokusere på at benytte en symmetrisk blok ciffer algoritme, og valget faldt på TDES og AES algoritmen. For at vælge den endelige algoritme skal der foretages nogle test af krypteringshastigheden for hver af de udvalgte algoritmer (se 6.1).

Protokol

I forbindelse med designet af protokollaget, blev der indført en ekstra sikkerhedsfunktion, der skal sikre at der kommunikeres med et reelt DISCOS system. Denne løsning baserer sig på at udvikle en initialiseringssekvens, der åbner for kommunikationen mellem DISCOS systemet og PDA'en såfremt, at denne sekvens bliver overholdt. Dette er endvidere også med til at sikre at den rigtige krypteringsnøgle er placeret på PDA'en.

Efter at have designet de forskellige løsningsmodeller og fastlagt den endelige systemarkitektur, kan det ses, hvordan den endelige applikation vil fungere. Det er lykkedes at designe nogle velstrukturerede og gennemtænkte løsninger, der vil forenkle den videre udvikling af applikationen.

5. Implementering

Formålet med dette kapitel er at fremhæve nogle af de mere teknisk krævende løsninger, med henblik på at beskrive implementeringen af disse. Dokumentationen af det resterende system er blevet dokumenteret som indlejret XML dokumentation i kildekoden. Dette gør at kildekoden og dokumentationen følges ad, hvilket er en stor fordel ved den videre udvikling. Da dokumentationen eksisterer som XML gør det muligt at lave udtræk af forskellige kendte typer dokumenter, som f.eks. Visual Studio Integration, Web Help, PDF, Windows Help, Help, HTML Help ved hjælpe af værktøjer som Doc-O-Matic.

På den vedlagte CD forefindes et udtræk som PDF dokument (DiscmanPDA_source.pdf).

5.1. Kommunikation[9]

Dette afsnit omhandler implementering af kommunikationslaget primære funktionaliteter.

5.1.1. Søgning efter Bluetooth enheder[10]

For at udføre søgningen efter Bluetooth enheder og modtage enhedernes navne og adresser:

1. Initialiseringen af winsocket's foretages ved at kalde **`WSAStartup`** funktionen. Denne kaldes med versionen på winsocket implementationen (i dette tilfælde version 2.2) og **`WSADATA`** strukturen, der kan benyttes til at indhente informationer omkring winsocket implementeringen.
2. Der oprettes et "*query set*" af typen **`WSAQUERYSET`** der benyttes til at specificere parametrene for søgningen. Her sættes **`dwNameSpace`** til NS_BTH (0x10), hvilket begrænser søgningen til Bluetooth enheder. Følgende kode afsnit viser opsætningen af **`WSAQUERYSET`** strukturen.

```
NativeMethods.WSAQUERYSET querySet = new NativeMethods.WSAQUERYSET();
querySet.dwSize = (uint)Marshal.SizeOf(typeof(NativeMethods.WSAQUERYSET));
querySet.dwNameSpace = NS_BTH;
```

3. For at aktivere forespørgslen kaldes funktionen **`WSALookupServiceBegin`**. Forinden specificeres kontrolflag parameteren til LUP_CONTAINERS (0x0002), hvilket specificerer at der skal udføres en enhedssøgning. Derudover oprettes en pointer til en handle der benyttes i forbindelse med de forskellige winsocket kald. Følgende kode afsnit viser **`WSALookupServiceBegin`** kaldet.

```
Int32 dwControlFlags = (int)NativeMethods.WSALookup.LUP_CONTAINERS;
IntPtr hLookup;
int result = NativeMethods.WSALookupServiceBegin(querySet, dwControlFlags,
out hLookup);
```

4. For at gennemse de fundne enheder benyttes funktionen **`WSALookupServiceNext`** der returnerer en pointer til en buffer der indeholder et "*result set*" i en **`WSAQUERYSET`** struktur. Forinden specificeres kontrolflag parameteren til LUP_RETURN_NAME | LUP_RETURN_ADDR, hvilket specificerer at enhedens navn og adresse skal returneres. Derudover allokeres der plads i hukommelsen til bufferen. Følgende kode afsnit viser **`WSALookupServiceNext`** kaldet.

```
Int32 dwBuffer = 0x10000;
IntPtr pBuffer = Marshal.AllocHGlobal(dwBuffer);
dwControlFlags = (int)(NativeMethods.WSALookup.LUP_RETURN_NAME |
NativeMethods.WSALookup.LUP_RETURN_ADDR);
result = NativeMethods.WSALookupServiceNext(hLookup, dwControlFlags, ref
dwBuffer, pBuffer);
```

[9] Dette afsnit tager udgangs punkt i reference [10].
[10] Dette underafsnit tager udgangspunkt i reference [11].

5. Den modtagede pointer konverteres til en **WSAQUERYSET** struktur.

```
NativeMethods.WSAQUERYSET queryResult = new NativeMethods.WSAQUERYSET();
Marshal.PtrToStructure(pBuffer, queryResult);
```

6. For at parse **WSAQUERYSET** strukturen, med henblik på at finde frem til enhedens adresse, kaldes funktionen **ParseQueryResult**. Denne metode udfører følgende:
   a. **WSAQUERYSET** strukturens **lpcsaBuffer** pointer konverteres til en **CSADDR_INFO** struktur, der indeholder informationer omkring winsocket adresser.
   b. **CSADDR_INFO** strukturens **RemoteAddr.lpSockaddr** pointer konverteres til en **SOCKADDR_BTH** struktur, der definerer Bluetooth socket adressen.
   c. Bluetooth addressen hentes ud af **SOCKADDR_BTH** strukturen og placeres i et byte array, der returneres.

   Følgende kode afsnit viser implementeringen af **ParseQueryResult** funktionen.

```
private byte[] ParseQueryResult(NativeMethods.WSAQUERYSET qsResult)
{
     byte[] addr;
     NativeMethods.CSADDR_INFO csAddr =
(NativeMethods.CSADDR_INFO)Marshal.PtrToStructure(qsResult.lpcsaBuffer,
typeof(NativeMethods.CSADDR_INFO));
     NativeMethods.SOCKADDR_BTH btAddr =
(NativeMethods.SOCKADDR_BTH)Marshal.PtrToStructure(csAddr.RemoteAddr.lpSoc
kaddr, typeof(NativeMethods.SOCKADDR_BTH));
     addr = BitConverter.GetBytes(btAddr.btAddr);
     return addr;
}
```

7. Der oprettes et nyt **BluetoothDevice** objekt svarende til de modtagede data.

```
new BluetoothDevice(queryResult.szServiceInstanceName,
ParseQueryResult(queryResult))
```

   Trin 4-7 gentages såfremt der er flere fundne enheder.

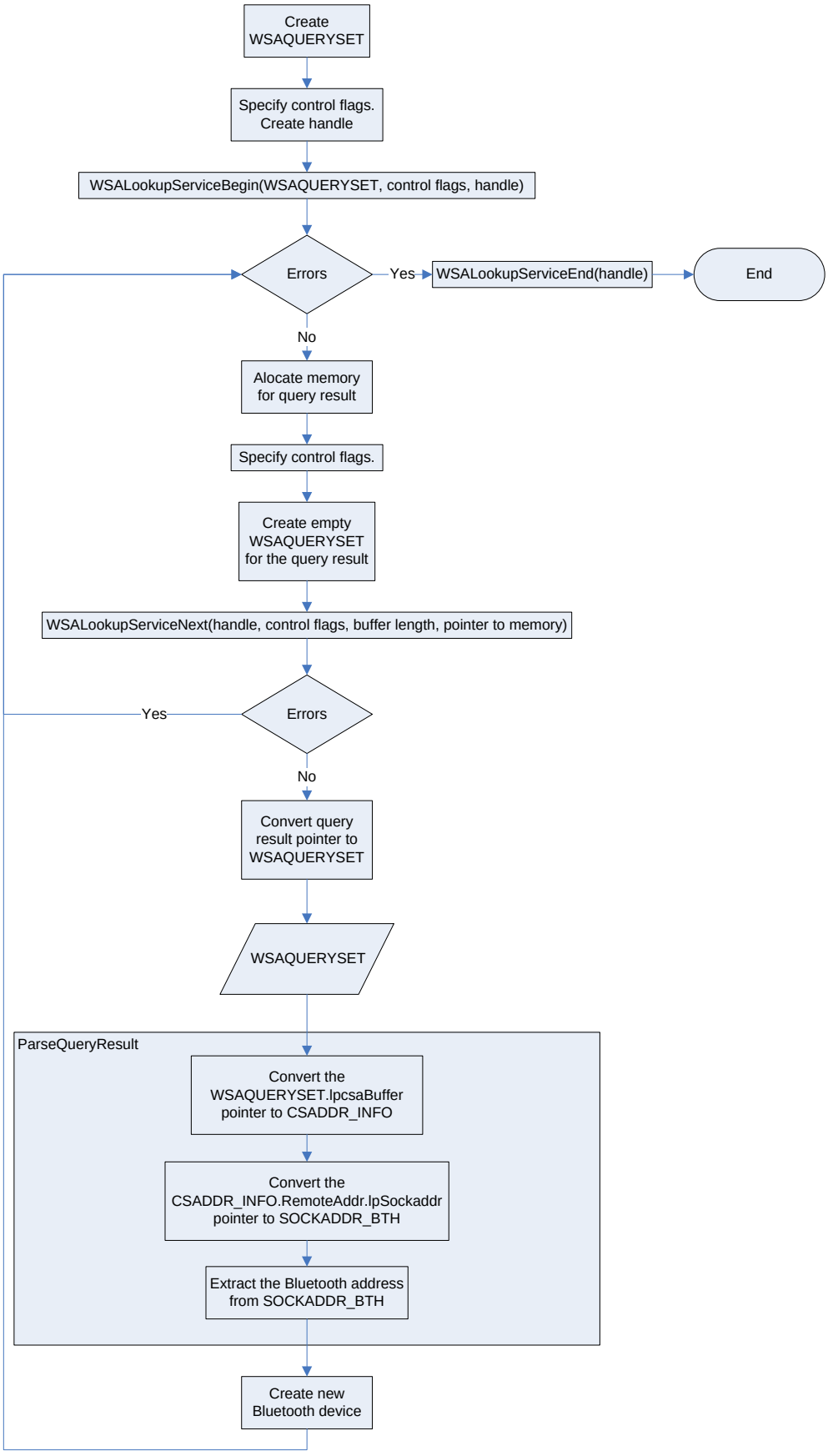
8. For at afslutte søgningen kaldes funktionen **WSALookupServiceEnd**. Denne metode frigør den handle der blev oprettet i trin 3.

```
result = NativeMethods.WSALookupServiceEnd(hLookup);
```

9. For at afslutte brugen af winsock services kaldes **WSACleanup** funktionen, der frigiver Ws2.dll filen.

```
NativeMethods.WSACleanup();
```

På den efterfølgende side forefindes et flowchart diagram over den netop specificerede procedure.

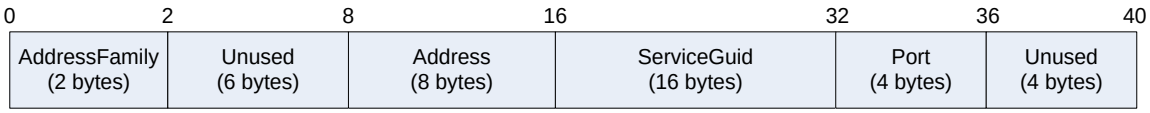


Figur 43 - Flowchart over Bluetooth søgningen.

5.1.2. Oprettelse af forbindelse til en Bluetooth enhed[11]

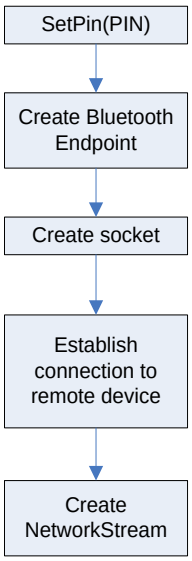
For at oprette forbindelse til en Bluetooth enhed:

1. Først gemmes Bluetooth enhedens PIN kode, ved at kalde **`SetPin`** metoden. Denne funktion konverterer en streng indeholdende PIN koden til et byte array og kalder dernæst **`BthSetPIN`** metoden der gemmer PIN koden i systemet.
2. Der oprettes et Bluetooth **`Endpoint`** objekt til fjern enheden. Dette objekt benyttes til at identificere netværksadressen på såvel den lokale samt fjern enheden, med henblik på adresse, service og port.
   Bluetooth baserer sig på **`AddressFamily`** 32, hvilket er et adresserings format, der specificerer, hvordan socket adressen skal opbygges. Da denne standard ikke er integreret fuldt ud i managed C# er det nødvendigt at oprette et specifikt Bluetooth **`Endpoint`**, der implementerer dette adresserings format. Det har været muligt at finde frem til adresseringsformatet ved at analysere den modtagede socket struktur. Ud fra netop denne struktur kan det ses at systemet arbejder i blokke af 8 bytes, hvilket er gengivet af illustrationen nedenfor.



Figur 44 - Adresserings format for Bluetooth socket adressering[12]

3. Der oprettes en socket med følgende parametre:
   - AddressFamily - 32: Svarende til Bluetooth adressering.
   - SocketType - Stream: Tovejs byte stream.
   - ProtocolType – Ggp: Port til port protokol.
4. Der etableres forbindelse til fjern-enheden ved at benytte det oprettede Endpoint objekt.
5. Der oprettes et NetworkStream objekt, der vil overtage kontrollen med den oprettede socket. Dette objekt styrer dermed alt kommunikation mellem enhederne.
6. Såfremt kommunikationen skal afbrydes lukkes NetworkStream'en.



Figur 45 - Flowchart over forbindelses proceduren.

[11] Dette under afsnit tager udgangspunkt i reference [12].
[12] Det har ikke været muligt at finde yderligere dokumentation omkring dette adresseringsformat.

5.2. Sikkerhed

Dette afsnit omhandler implementeringen af AES og TDES krypterings algoritmer.

5.2.1. Kryptering

Der tages udgangspunkt i AES algoritmen, hvilket fremgår af nedenfor stående implementerings eksempel.

For at kryptere/dekryptere et givent bytearray:

1. Der oprettes en instans af den ønskede algoritme klasse.

```
RijndaelManaged cipher = new RijndaelManaged();
```

2. Krypteringsnøgle og initialiseringsvektor initialiseres.
3. For at kryptere et byte array oprettes der et krypterings objekt ud fra algoritme instansen.

```
_cipher.CreateEncryptor(this._key, this._iv);
```

   For at dekryptere et byte array oprettes der et dekrypterings objekt ud fra algoritme instansen.

```
_cipher.CreateDecryptor(this._key, this._iv);
```

4. Der oprettes 2 stream's:
   - En hukommelses stream, der skal indeholde de bytes der skal behandles.
   - En krypterings stream, der skal foretage krypteringen/dekrypteringen.
     Denne oprettes med hukommelses stream'en, (de)krypterings objektet samt stream tilstanden som parametre.

```
MemoryStream memory = new MemoryStream();
CryptoStream stream = new CryptoStream(memory, cryptoTransform,
CryptoStreamMode.Write);
```

5. Bytene krypteres/dekrypteres i forhold til krypterings stream'en.

```
stream.Write(input, 0, input.Length);
stream.FlushFinalBlock();
```

6. De behandlede bytes læses fra hukommelses stream'en og konverteres tilbage i et bytearray.

```
memory.Position = 0;
result = new byte[memory.Length];
memory.Read(result, 0, result.Length);
```

7. De enkelte stream's lukkes.

5.3. XML Serialization

Dette afsnit omhandler læsning og skrivning af data objekter fra/til en XML fil.

5.3.1. Indlæs EEPROM Meta Map[13]

For at indlæse EEPROM Meta Map'et oprettes der en klasse til at indeholde det samlede map, samt en struktur til de enkelte indlæg:

- EEPROMMetaMap – denne klasse udgør et indekseret dictionary indeholdende de enkelte indlæg.
- EEPROMMetaEntry – denne struktur indeholder alle data fra de enkelte indlæg.

Strukturen og klassen arver `System.Xml.Serialization.IXmlSerializable` interfacet, hvilket gør at man kan specificere, hvordan data fra en XML fil indføres i det specifikke objekt. Dette gøres via `IXmlSerializable.ReadXml` metoden.

[13] I bilagene forefindes der en tabel der specificerer det aktuelle EEPROM meta map (9.5).

EEPROMMetaEntry

Denne struktur består af følgende felter:

- Adresse – adresse i hukommelsen.
- Type – data typen af det data der er placeret i hukommelsen.
- Længde – antal bytes der er allokeret i hukommelsen.
- Konverteringsfaktor – konverteringsfaktor mellem hardware og software repræsentation.

Indlæsningen af disse data sker via en XmlReader, der benyttes til at manøvrerer rundt i XML filen. Nedenfor ses koden for indlæsningen af de enkelte felter.

```
void IXmlSerializable.ReadXml(System.Xml.XmlReader r)
{
    r.MoveToAttribute("Address", String.Empty);
    this._address = (uint)r.ReadContentAs(typeof(uint), null);
    r.MoveToAttribute("ValueType", String.Empty);
    string typeName = r.ReadContentAsString();
    if (typeName.Equals("PowerSense.DiscmanPDA.Common.Enumerators.VectorGroupType"))
        this._type = typeof(Common.Enumerators.VectorGroupType);
    else this._type = Type.GetType(typeName);
    r.MoveToAttribute("ConversionFactor", String.Empty);
    this._conversionFactor = (uint)r.ReadContentAs(typeof(uint), null);
    r.MoveToAttribute("Length", String.Empty);
    this._length = (int)r.ReadContentAs(typeof(int), null);
    r.Skip();
}
```

EEPROMMetaMap

Denne klasse sørger for at samle de enkelte meta map indlæg i et dictionary der indekseres via hukommelsesadressen. Denne klasse håndterer derved den overordnede XML struktur i forhold til de enkelte *nodes*. De enkelte nodes *deserializes* til EEPROMMetaEntry objekter og indføres i dictionary'et. Nedenfor ses koden for indlæsningen af de enkelte noder.

```
void IXmlSerializable.ReadXml(System.Xml.XmlReader r)
{
    System.Xml.Serialization.XmlSerializer keySer = new
System.Xml.Serialization.XmlSerializer(typeof(int));
    System.Xml.Serialization.XmlSerializer valueSer = new
System.Xml.Serialization.XmlSerializer(typeof(EEPROMMetaEntry));
    r.Read();

    while (r.NodeType != System.Xml.XmlNodeType.EndElement)
    {
        r.ReadStartElement("item");
        r.ReadStartElement("key");
        int key = (int)keySer.Deserialize(r);
        r.ReadEndElement();
        r.ReadStartElement("value");
        EEPROMMetaEntry value = (EEPROMMetaEntry)valueSer.Deserialize(r);
        this.Add(key, value);
        r.ReadEndElement();
        r.ReadEndElement();
        r.MoveToContent();
    }
}
```

5.3.2. Gem målinger

Til at gemme de målte data til en XML file oprettes der 2 strukturer, en til at indeholde informationer omkring DISCOS enheden, hvor målingerne blev udført og en til at indeholde de enkelte målte data. Derudover oprettes der en klasse til at samle informationerne omkring DISCOS enheden og alle målte data:

- TUnitBasicInfo – denne struktur indeholder basis informationer om en DISCOS enhed.
- TLogData – denne struktur indeholder de enkelte log indlæg.
- DataContainer – denne klasse indeholder alle målte data.

Strukturerne og klassen arver `System.Xml.Serialization.IXmlSerializable` interfacet, hvilket gør at man kan specificere, hvordan data fra et objekt skrives til en XML fil. Dette gøres via `IXmlSerializable.WriteXml` metoden.

TUnitBasicInfo

Denne struktur indeholder følgende felter der skal medtages i XML filen:

- StationID – Transformerstationens id.
- EQTY – Transformerens equipment type.
- UnitType – Enhedstypen.
- FieldType – Opti enhedens felttype.
- BayID – Enhedens Bay id.
- CANID – Enhedens CAN id.

Skrivning af disse felter til XML filen sker via en XmlWriter, der håndterer skrivningen af data til XML filer. Nedenfor ses koden for skrivningen af de enkelte attributter.

```
void System.Xml.Serialization.IXmlSerializable.WriteXml(System.Xml.XmlWriter w)
{
    string unitType = "";
    string fieldType = "";

    w.WriteAttributeString("StationID", _stationID);
    w.WriteAttributeString("EQTY", _eqty);

    if (_unitType == PowerSense.DiscmanPDA.Common.Enumerators.UnitType.MASTER)
        unitType = "Master";
    else if (_unitType == PowerSense.DiscmanPDA.Common.Enumerators.UnitType.OPTI)
    {
        unitType = "Opti";
        if (_fieldType == PowerSense.DiscmanPDA.Common.Enumerators.FieldType.LINE)
            fieldType = "Line";
        else if (_fieldType == PowerSense.DiscmanPDA.Common.Enumerators.FieldType.TRAFO)
            fieldType = "Trafo";
    }
    w.WriteAttributeString("UnitType", unitType);

    if (_unitType == PowerSense.DiscmanPDA.Common.Enumerators.UnitType.OPTI)
         w.WriteAttributeString("FieldType", fieldType);

    w.WriteAttributeString("BayID", _bayID.ToString());
    w.WriteAttributeString("CANID", _canID.ToString());
}
```

TLogData

Denne struktur indeholder de enkelte målte data værdier.

Nedenfor ses koden for skrivningen af de enkelte attributter.

```
void System.Xml.Serialization.IXmlSerializable.WriteXml(System.Xml.XmlWriter w)
{
    w.WriteAttributeString("TimeStamp", _TimeStamp.ToString());
    w.WriteAttributeString("Recipient", _recipient.ToString());
    w.WriteAttributeString("Command", _command.ToString());
    w.WriteAttributeString("Sender", _sender.ToString());
    w.WriteAttributeString("xER", _xER.ToString());
    w.WriteAttributeString("Length", _length.ToString());

    string tmp = "0x";
    for (int i = 0; i < _length; i++)
    {
        tmp += _data[i].ToString("x2");
    }
    w.WriteAttributeString("Data", tmp);
    w.WriteAttributeString("Line", _line.ToString());
    …
}
```

DataContainer

Denne klasse sørger for at samle de enkelte data log indlæg og enhedsinformationerne i en overordnet XML struktur. Dette gøres ved at *serialize* TUnitBasicInfo strukturen som hovedelement efterfulgt af TDataLog elementer.

```
void System.Xml.Serialization.IXmlSerializable.WriteXml(System.Xml.XmlWriter w)
{
    System.Xml.Serialization.XmlSerializer unitSer = new
System.Xml.Serialization.XmlSerializer(typeof(TUnitBasicInfo));
    System.Xml.Serialization.XmlSerializer dataSer = new
System.Xml.Serialization.XmlSerializer(typeof(TLogData));

    w.WriteString("\n\t");
    w.WriteStartElement("UnitInfo");
    w.WriteString("\n\t\t");

    // remove xsi and xsd namespace prefixes from the output.
    System.Xml.Serialization.XmlSerializerNamespaces xmlNamespaces = new
System.Xml.Serialization.XmlSerializerNamespaces();
    xmlNamespaces.Add(String.Empty, String.Empty);
    unitSer.Serialize(w, _unitInfo, xmlNamespaces);
    w.WriteString("\n\t");
    w.WriteEndElement();

    w.WriteStartElement("LogEntries");
    w.WriteString("\n\t\t");

    for (int i = 0; i < _data.Count; i++)
    {
        w.WriteStartElement("Data");
        w.WriteString("\n\t\t\t");
        dataSer.Serialize(w, _data[i], xmlNamespaces);
        w.WriteString("\n\t\t");
        w.WriteEndElement();
    }
    w.WriteString("\n\t");
    w.WriteEndElement();
    w.WriteString("\n");
}
```

5.4. Audio[14]

Dette afsnit omhandler implementeringen af audio I/O funktionaliteter og tager udgangspunkt i Waveform Audio API'et featurene:

- Søgning og initialisering af audio I/O enheder.
- Allokering af audio data blokke.
- Optagning og afspilning af audio filer.
- Brugen af windows beskeder til håndtering af audio optagning/afspilning.

De funktioner der vedrører optagning af audio filer er listet nedenfor:

- `waveInGetNumDevs()`: Fastslår antallet af audio drivere tilgængelig til input.
- `waveInOpen()`: Opretter en instans af den specificerede audio input enhed.
- `waveInPrepareHeader()`: Klargøre en `WAVEHDR` og data blok.
- `waveInUnprepareHeader()`: Frigiver en tidligere klargjort `WAVEHDR` og data blok.
- `waveInClose()`: Lukker den specificerede instans af audio enheden.
- `waveInReset()`: Stopper optagelsen og tømmer køen.
- `waveInStart()`: Starter optagelsen til buffer køen.
- `waveInStop()`: Stopper optagelsen.
- `waveInAddBuffer()`: Tilføjer en klargjort buffer til optagnings-køen.
- `waveInGetDevCaps()`: Henter en specificeret audio enheds kompetencer.

De funktioner der vedrører afspilningen af audio filer er listet nedenfor:

- `waveOutGetNumDevs()`:Fastslår antallet af audio drivere tilgængelig til output.
- `waveOutOpen()`:Opretter en instans af den specificerede audio output enhed.
- `waveOutGetVolume()`: Returnere output enhedens volume.
- `waveOutSetVolume()`: Sætter output enhedens volume.
- `waveOutPrepareHeader()`:Klargøre en `WAVEHDR` og data blok.
- `waveOutUnprepareHeader ()`:Frigiver en tidligere klargjort `WAVEHDR` og data blok.
- `waveOutWrite()`: Starter afspilningen af buffer køen.
- `waveOutClose()`:Lukker den specificerede instans af audio enheden.
- `waveOutReset()`:Stopper afspilningen og tømmer køen.
- `waveOutPause()`: Pauser afspilningen.
- `waveOutRestart()`: Fortsætter en afspilning der er sat på pause.
- `waveOutGetDevCaps()`:Henter en specificeret audio enheds kompetencer.

Derudover benyttes 2 funktioner til at allokere og frigive hukommelse:

- `LocalAlloc()`:Allokerer et specifikt antal bytes i hukommelsen.
- `LocalFree()`:Frigiver hukommelsen til det specificerede objekt.

[14] Dette afsnit tager udgangspunkt i reference [13].

5.4.1. Optagning af voice log

For at optage en lydfil via den indbyggede mikrofon:

1. Der søges efter audio input enheder ved at kalde **waveInGetNumDevs** funktionen. Denne returnerer en integer, der indikerer antallet af fundne enheder.
2. Det undersøges om audio enheden understøtter det ønskede format. Dette gøres ved at initialisere hukommelse til at indeholde **WAVEINCAPS** strukturen, der beskriver enhedens kompetencer, oprette en pointer til denne for derefter at kalde **waveInGetDevCaps** funktionen med pointeren som parameter.

```
WaveNative.WAVEINCAPS caps = new WaveNative.WAVEINCAPS();
IntPtr capsPtr = Marshal.AllocHGlobal(size);
Marshal.StructureToPtr(caps, capsPtr, false);
result = WaveNative.waveInGetDevCaps(0, capsPtr, (uint)size);
```

   Derefter konverteres pointeren tilbage til strukturen og det undersøges om resultat fra strukturens **dwFormats** felt, der indikerer hvilke formater der er understøttet, understøtter 11.025 kHz – mono - 8-bit formatet.

```
caps = (WaveNative.WAVEINCAPS)Marshal.PtrToStructure(capsPtr,
typeof(WaveNative.WAVEINCAPS));
if ((caps.dwFormats & WaveNative.WAVE_FORMAT_1M08) == 0)
    return MMSYSERR.NOTSUPPORTED;
else return MMSYSERR.NOERROR;
```

3. Audio input enheden initialiseres ved først at oprette en instans af **WAVEFORMATEX** klassen, der indeholder informationer omkring wave formatet, for dernæst at kalde **waveInOpen** funktionen. Denne funktion forsøger at åbne enheden med det valgte wave format.

```
result = WaveNative.waveInOpen(ref _WaveInHandle, deviceID, _WaveFormat,
_MessageWindow.Hwnd, 0, WaveNative.CALLBACK_WINDOW);
```

   Derefter oprettes en liste af buffere, svarende til den længste optagelse (er sat til 4 min.), ved at oprette et array af **WAVEHDR** objekter (maks længde / blokstørrelse). **WAVEHDR** klassen benyttes til at tildele informationer omkring og til at tilgå audio data blokke.
   Herefter initialiseres de to første buffere ved at allokere plads, forberede buffernes header og sende bufferne til audio input enheden. Allokeringen foregår ved at kalde **LocalAlloc** funktionen, klargøringen af headeren foregår ved at kalde **waveInPrepareHeader** funktionen, mens afsending af bufferne foregår ved at kalde **waveInAddBuffer** funktionen.

```
_WaveHDR[bufIndex].Init(writeLength, false);
WaveNative.waveInPrepareHeader(_WaveInHandle, _WaveHDR[bufIndex],
(uint)Marshal.SizeOf(_WaveHDR[bufIndex]));
return WaveNative.waveInAddBuffer(_WaveInHandle, _WaveHDR[bufIndex],
(uint)Marshal.SizeOf(_WaveHDR[bufIndex]));
```

4. Optagelsen startes ved at kalde **waveInStart** funktionen.

```
MMSYSERR result = WaveNative.waveInStart(_WaveInHandle);
```

5. Såfremt at en audio blok er færdig med at optage, modtages der en windows besked om dette. Windows beskederne modtages ved at implementere en klasse der arver **MessageWindow** klassen, og som overstyrer den oprindelige **WndProc** metode. Modtages der en besked med ID nummer MM_WIN_DATA (0x3C0) betyder dette, at en audio blok er færdig med at blive optaget.
   Såfremt at der er flere buffere, som endnu ikke er blevet optaget initialiseres den næste buffer, på samme måde som de 2 første.

6. Når optagningen er afsluttet stoppes audio input enhedens optagning ved at kalde **waveInReset** funktionen.
7. De optagede audio blokke gemmes ved at skrive de enkelte data fra buffer listen til en wav fil. For yderligere information omkring wav PCM fil formatet se reference [14].
8. De brugte ressourcer frigives ved at kalde **waveInUnprepareHeader** funktionen og ved at frigive de enkelte buffere via **LocalFree** funktionen. Derefter lukkes audio input enheden ved at kalde **waveInClose** funktionen.

5.4.2. Afspilning af voice log

For at afspille en lydfil:

1. Der søges efter audio output enheder ved at kalde **waveOutGetNumDevs** funktionen. Denne returnerer en integer der indikerer antallet af fundne enheder.
2. Audio output enheden initialiseres ved først at indlæse audio filen, ved at benytte en **BinaryReader**, og oprette en instans af **WAVEFORMATEX** klassen, der indeholder informationer omkring wave formatet, fra filens data (for yderligere information omkring wav PCM fil formatet se reference [14]), for dernæst at kalde **waveOutOpen** funktionen.
   Denne funktion forsøger at åbne enheden med det i filen specificerede wave format.

```
_WaveFormat.Read(_reader);
MMSYSERR result = WaveNative.waveOutOpen(ref _WaveOutHandle, curDevice,
_WaveFormat, _MessageWindow.Hwnd, 0, WaveNative.CALLBACK_WINDOW);
```

   Derefter oprettes to buffere, ved at oprette et array af **WAVEHDR** objekter. **WAVEHDR** klassen benyttes til at tildele informationer omkring og til at tilgå audio data blokke.
   Herefter initialiseres den første buffer ved at indlæses data fra filen og allokere plads ved at kalde **LocalAlloc** funktionen, hvorefter at bufferens header klargøres ved at kalde **waveOutPrepareHeader** funktionen.

```
lpData = LocalAlloc(LMEM_FIXED, bufferLength);
WaveNative.waveOutPrepareHeader(_WaveOutHandle, _WaveHDR[bufIndex],
(uint)Marshal.SizeOf(_WaveHDR[bufIndex]));
```

3. Afspilningen startes ved at kalde **waveOutWrite** funktionen med den første buffer. Derefter initialiseres den anden buffer, hvorefter den skrives til audio output enhedens buffer kø, ved at kalde **waveOutWrite** funktionen.

```
MMSYSERR result = WaveNative.waveOutWrite(_WaveOutHandle, _WaveHDR[0],
(uint)Marshal.SizeOf(_WaveHDR[0]));
```

4. Såfremt at en audio blok er blevet afspillet modtages der en windows besked om dette. Windows beskederne modtages ved at implementere en klasse der arver **MessageWindow** klassen og som overstyrer den oprindelige **WndProc** metode. Modtages der en besked med ID nummer **MM_WON_DONE** (0x3BD) betyder dette at en audio blok er færdig med at blive afspillet.
   Såfremt at der er flere audio blokke som endnu ikke er blevet afspillet, initialiseres den buffer der ikke er placeret i køen.
5. Når afspilningen er afsluttet stoppes audio output enhedens afspilning ved at kalde **waveOutReset** funktionen.
6. De brugte ressourcer frigives ved at kalde **waveOutUnprepareHeader** funktionen og ved at frigive de enkelte buffere via **LocalFree** funktionen. Derefter lukkes audio output enheden ved at kalde **waveOutClose** funktionen.

5.5. Delkonklusion

Gennem dette kapitel er implementeringen af nogle af de mere krævende elementer i systemet blevet beskrevet. Dette drejer sig om implementeringen af:

- Kommunikation – dette vedrører søgning efter Bluetooth enheder og oprettelse af forbindelse til disse. Der fokuseres på brugen af *winsockets* og konverteringen mellem managed of unmanaged kode.
- Sikkerhed – dette vedrører kryptering og dekryptering af data. Der fokuseres på at gennemgå implementeringen af AES algoritmen i forhold til de indbyggede algoritme klasser i .NET Compact Framework'et.
- XML – dette vedrører skrivning og læsning af data til/fra en XML fil. Der fokuseres på serialization og deserialization af de forskellige typer objekter der benyttes til dette.
- Audio – dette vedrører optagning og afspilning af lyd via PDA'en. Der fokuseres på at fremhæve forbindelsen til audio I/O enheden.

Ved at have gennemgået implementeringen af de enkelte elementer opnås en bedre forståelse for den teknologi, der er blevet benyttet i udviklingen af applikationen.

6. Test

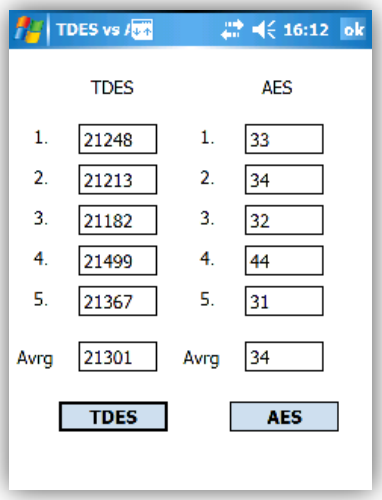
Formålet med dette kapitel er at få testet de enkelte dele af applikationen, for dermed at sikre systemet fungerer efter hensigten. Eftersom at der på nuværende tidspunkt ikke er blevet indført unit testing i Compact Framework'et, er der valgt at se bort fra disse test. Formålet med at køre unit tests ligger i, at der oprettes nogle test specifikationer til de enkelte dele af systemet, som derved er med til at sikre at systemet stadig opfylder de opstillede krav efter, at der er blevet foretaget ændringer til systemet. Da unit testing ikke er blevet indført i framework'et vil resultatet af disse test i stedet skulle ses ud fra det grafiske user interface. Der er i stedet blevet taget udgangspunkt i de opstillede use cases, hvor der er blevet udført use case tests af samtlige use cases. Resultatet af disse test ses via den eksisterende brugerflade og sikre dermed at funktionaliteten af de enkelte use cases fungerer efter hensigten.

6.1. Krypteringshastighed: TDES vs AES

For at afgøre hvilken krypterings algoritme, der vil være det optimale for systemet, foretages der en krypteringshastigheds test. Denne vil sammenligne krypteringshastighederne mellem TDES og AES krypterings standarderne. For at sikre at begge algoritmer bliver benyttet ens er de blevet implementeret efter den samme program struktur (se 5.2), hermed sikres det at den enkelte algoritme har samme betingelser for at gennemføre testen.

Testen implementeres direkte på PDA'en, for at give det mest sigende resultat i forhold til den endelige applikation, og køres via et grafisk vindue, der består af en række tekstfelter og nogle knapper til at aktivere den enkelte algoritme.

Testen foregår ved at et 13 bytes data array krypteres 200 gange for hver krypterings algoritme. Dette forløb gennemføres 5 gange og krypterings forløbets varighed for hver gennemløb registreres i antal tick's (ms). Til sidst udregnes et gennemsnit over de 5 gennemløb.



Figur 46 – Screenshot: TDES vs AES

Ovenfor ses et screenshot af det endelige resultat af testen. Testen viste at TDES algoritmen var ca. 626 gange langsommere en AES algoritmen og det kan derved konkluderes at AES er det rigtige valg for en krypterings algoritme til netop dette system. Denne store forskel må ligge i at DES algoritmen blev udviklet til at køre i hardware, hvilket har medført at algoritmen er væsentlig langsommere i software. Da TDES derudover køre gennem DES algoritmen 3 gange vil dette yderligere sænke krypteringshastigheden.

6.2. Use Case Test

6.2.1. Use case § 0001 – Opret forbindelse

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	05-03-2007	1.0	1.0	GL
Titel:	Use case §0001 – Opret forbindelse.			
Formål:	At teste funktionaliteten af use case §0001.			
Beskrivelse:	Testen vil blive udført ved opstart af applikationen.			
Forudsætninger:	At der er en Bluetooth enhed inden for rækkevidde med mulighed for at aktivere/deaktivere COM port service.			
Test cases:	**Hovedforløb:** 1. Bluetooth enheden startes og COM port service aktiveres. 2. Der søges efter Bluetooth enheder. 3. En liste med tilgængelige enheder vises. 4. Den ønskede enhed vælges. 5. Der oprettes forbindelse til den valgte enhed og en bekræftelse vises. **Alternativforløb:** 4a)Den ønskede enhed forefindes ikke på listen. 1. Bluetooth enheden stoppes. 2. Der søges efter Bluetooth enheder. 3. En liste med tilgængelige enheder vises. 4. Den ønskede enhed findes ikke. a. Mulighed for ny søgning. b. Mulighed for at afslutte programmet.			

	5a)Der kan ikke oprettes forbindelse til den valgte enhed. 1. Bluetooth enheden startes og COM port service deaktiveres. 2. Der søges efter Bluetooth enheder. 3. En liste med tilgængelige enheder vises. 4. Den ønskede enhed vælges. 5. Der kan ikke oprettes forbindelse til den valgte enhed. a. Fejlbesked vises. b. Mulighed for ny søgning. c. Mulighed for at oprette forbindelse til en ny enhed. d. Mulighed for at afslutte programmet.
Tilladte afvigelser/fejl:	Ingen.

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	05-03-2007	1.0	1.0	GL
Titel:	Use case §0001 – Opret forbindelse.			
Resultater:	**Hovedforløb:** Bluetooth enhed blev vist og der blev etableret forbindelse til denne. **Alternativforløb:** 4a) Det var såvel muligt at foretage en ny søgning, samt at afslutte programmet. 5a) Der blev vist en fejlbesked og det var dernæst muligt at foretage en ny søgning samt at afslutte programmet.			
Konklusion:	Testen forløb succesfuldt.			

6.2.2. Use case § 0004 – Søg efter DISCOS enheder

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	06-04-2007	8.0	1.0	GL
Titel:	Use case § 0004 – Søg efter DISCOS enheder.			
Formål:	At teste funktionaliteten af use case §0004.			
Beskrivelse:	Der foretages en søgning efter DISCOS enheder.			
Forudsætninger:	At der er oprettet forbindelse til et DISCOS system, at kommunikationen er initialiseret.			
Test cases:	**Hovedforløb:** 1. Tilslut en Master og en Opti enhed til DISCOS systemet. 2. Systemet søger efter tilsluttede DISCOS enheder. 3. Indstillingerne for de fundne enheder læses fra EEPROM'en (Use Case §0005). 4. Der returneres en liste med de fundne enheder. **Alternativforløb:** 2a) Søgningen kan ikke fuldføres, da forbindelsen til DISCOS systemet er blevet afbrudt. 1. Tilslut en Master og en Opti enhed til DISCOS systemet. 2. Systemet søger efter tilsluttede DISCOS enheder. 3. Søgningen kan ikke gennemføres. a. Fejlbesked vises.			

	b. Mulighed for at genetablere forbindelsen. c. Mulighed for at gå tilbage til enhedssøgningen.
Tilladte afvigelser/fejl:	Ingen.

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	06-04-2007	8.0	1.0	GL
Titel:	Use case § 0004 – Søg efter DISCOS enheder.			
Resultater:	**Hovedforløb:** Søgningen blev gennemført og der blev vist en Master og en Opti enhed. **Alternativforløb:** 2a) Søgningen kunne ikke gennemføres, der blev vist en fejlbesked. Det var dernæst muligt at forsøge at genetablere forbindelsen, samt at gå tilbage til enhedssøgningen.			
Konklusion:	Testen forløb succesfuldt.			

6.3. Funktionstest

Funktionstesten skal være med til at belyse at målingerne og de modtagede data fra DISCOS systemet behandles korrekt.

Da systemet ikke er blevet færdig udviklet er der kun blevet udført en meget begrænset funktionstest. Denne bestod i at sætte en kendt strøm og spænding på DISCOS systemet for at se at applikationen kunne vise de korrekte værdier. Resultatet af testen var at de viste målinger stemmede overens med den tilsluttede strøm og spænding.

Derudover blev de enkelte indstillinger verificeret ved at sammenligne resultatet med det eksisterende Discman system.

6.4. Delkonklusion

Gennemføringen af krypteringshastigheds testen gav en god indikation af nødvendigheden for at gennemføre sådanne test, inden der vælges en specifik krypterings algoritme. At der var så stor forskel i krypteringshastigheden mellem de enkelte algoritmer var meget overraskende, men gjorde derimod valget mellem dem meget enkelt.

Use case testene forløb uden de store problemer, hvilket også afspejles af de endelige resultater. Alle testene forløb som forventet og systemet har derved vist sig at fungere efter hensigten.

Funktionstesten viste at systemet håndterede de basale målinger og indstillinger korrekt. Efter at systemet er færdig udviklet vil der skulle udføres en detaljeret funktionstest af systemet, hvilket dog vil kunne tage adskillige uger at gennemføre.

7. Evaluering og konklusion

Formålet med dette kapitel er at evaluere det gennemgåede projektforløb, med henblik på den benyttede udviklingsproces, samt at beskrive udbyttet for såvel den projektstuderende som for virksomheden.

7.1. Projektforløb

Visionen for projektet var at skabe grundlaget for udviklingen af et nyt produkt i DISCOS serien, der kunne være med til at illustrere nogle af de nye muligheder, der introduceres ved lanceringen af en ny kommunikationsplatform til DISCOS systemet. Endvidere skulle projektet afdække nogle af de problemstillinger udviklingen af et trådløst kommunikerende redskab medfører. Hovedpunkterne indenfor visionen af projektet omhandlede:

- Udviklingen af en prototype der fremstår som et intuitivt og brugervenligt produkt.
- At afdække de sikkerhedsrisici der introduceres i forbindelse med indføring af en trådløs kommunikationsstandard – med henblik på at opstille et løsningsforslag.

Der blev i samtale med PowerSense specificeret et grundlag for hvilke system funktionaliteter, der skulle indgå i det nye produkt, hvoraf der blev taget udgangspunkt i den grafiske brugerflade. Dette er blevet benyttet gennem hele opbygningen af den almene applikations struktur til såvel at sikre at de ønskede funktionaliteter blev understøttet, samt at opbygge et system med den nødvendige feedback til brugeren.

Der blev først og fremmest fokuseret på at skabe en grundstruktur, der kunne imødegå de sikkerhedsmæssige problemstillinger i projektet. Dette indebar udviklingen af kommunikationslaget og protokollaget, som tilsammen udgør "*backbone*" af projektet. Dermed blev der fra start taget hånd omkring de største risikoområder i projektet, hvilket gjorde tilgangen til det videre projektforløb mere enkelt.
Der er endvidere blevet fokuseret på at opbygge hvert lag, så det understøttede alle tænkelige scenarioer, for på den måde at forenkle arbejdet med de interagerende lag.

Resultatet af den valgte fremgangsmåde gjorde at der hurtigt blev oprettet en fast arkitektur, der kombinerede funktionalitet og brugervenlighed, og som medførte et struktureret projektforløb.

7.2. Udviklingsproces

Der blev gennem hele udviklingsprocessen benyttet nogle forskellige remedier fra Unified Process (UP), såsom use cases, sekvensdiagrammer og klassediagrammer, der blev benyttet til at analysere og designe det endelige system. Disse var i sammenhæng med grundprincipperne indenfor UP med til at skabe et struktureret og velfungerende udviklingsforløb. Den bedste indikation af at netop UP var et godt valg i forbindelse med dette projekt ligger i, at der kun forekom små ændringer i forhold til den opstillede tidsplan. Specielt ved at udpege de forskellige risikoområder i projektet fra start gjorde, at der ikke dukkede nogen uventede problemer op, der ville kunne have kompromiseret udviklingsforløbet.

7.3. Forbedringer

Følgende områder skal færdigudvikles før, at det er muligt at tage applikationen i brug:

- Mulighed for at ændre på de enkelte enheders konfiguration – dette indebærer implementering af et undtagelsessystem, der validerer alle bruger inputs.

- Oprettelse af service og administrator brugerflade – dette indebærer, at der oprettes 2 forskellige kompilerings opsætninger, der kan identificeres af systemet. Derved vil det være muligt at adskille de 2 udgaver og dermed ændre brugerfladen svarende til den specifikke konfiguration.
- Dynamisk krypteringsnøgle – dette indebærer, at det er muligt at placere en nøgle fil i applikationsmappen, som så indlæses i systemet under opstart.
- Opsætning af kommunikationsmodul – dette indebærer, at implementere den specificerede DISCOS kommunikation emulators protokol i det kommende kommunikationsmodul. Endvidere skal der implementeres en mulighed for at ændre krypteringsnøglen i modulet – dette kunne gøres ved at udvide den eksisterende SMS protokol.
- Funktionstest af systemet – dette indebærer at udfører en omfattende funktionstest af alle elementer i systemet. Dette vil skulle gennemføres ved at teste op mod de forskellige typer af transformerstationer og dets DISCOS systemer.

Det anslås at disse forbedringer vil tage i omegnen af 2-3 måneder at få implementeret og testet, hvis der er 1 mand på opgaven.

7.4. Fremtidsvision

Det næste trin i udviklingen af Discman PDA vil være at introducere de potentielle kunder for produktet. Dette skal vise, om der er et marked for systemet og vise hvilke specifikke behov, der stilles til systemet, specielt i forhold til den grafiske brugerflade.

Såfremt at der er et marked for produktet på nuværende tidspunkt, vil der være mulighed for at lancere produktet i sammenhæng med det kommende DISCOS kommunikationsmodul.

7.5. Konklusion

Fra den projektstuderendes synspunkt er det endelig resultat af projektet blevet meget tilfredsstillende, da applikationen fremstår som et funktionelt produkt nærmere end en prototype. Dette ses blandt andet i sammenhængen mellem en simpel og brugervenlig brugerflade, og system funktionaliteten. Projektet har desuden givet et bedre indtryk af mulighederne indenfor udviklingen af mobile applikationer og heri de forskellige sikkerhedsmæssige aspekter, der introduceres i forbindelse med indføringen af en trådløs kommunikationsstandard. Endvidere har det været en meget indsigtsgivende oplevelse at arbejde intenst og fokuseret med et projekt af denne størrelse, da det giver et godt perspektiv af de muligheder og problemer, der er i et sådant projekt.

Følgende udmelding er fra PowerSense's salgschef:
Fra et salgs og marketings synspunkt har Glen fanget de input han fik i projektets begyndelse, og fået dem ind i det endelige resultat uden at gå på kompromis med de tekniske krav, der var stillet. Dette betyder at udover, at projektet har afført et produkt, der teknisk lever op til de krav, der blev stillet fra projektets begyndelse, så er der i dag også et produkt, der kan afprøves direkte i marken af en "final user" med gennemtænkt og afstemt brugergrænseflade. Det er en vigtig faktor for PowerSense at potentielle kunder tidligt i udviklings forløbet kan teste på vores produkter, dette gør at udviklingstiden mindskes, og det er en vigtig faktor for et udviklingshus som os. Alt i alt blev Glens resultat; flot og gennemtænkt.

Stud. Ing. Glen Lauridsen
Kongens Lyngby, maj 2007

8. Referencer

8.1. Bøger

[1] "Applying UML and Patterns, 3rd edition", Craig Larman, Prentice Hall.

[2] "Security in Computing, 3rd edition", Charles P. Pfleeger and Shari Lawrence Pfleeger, Prentice Hall.

8.2. Hjemmesider

[3] Rational Unified Process
http://en.wikipedia.org/wiki/Rational_Unified_Process

[4] Unified Process
http://en.wikipedia.org/wiki/Unified_Process

[5] Controller Area Network
http://en.wikipedia.org/wiki/Controller_Area_Network

[6] Data Encryption Standard
http://en.wikipedia.org/wiki/Data_Encryption_Standard

[7] Triple DES
http://en.wikipedia.org/wiki/TDES

[8] RC2
http://en.wikipedia.org/wiki/RC2

[9] Advanced Encryption Standard
http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[10] Bluetooth Application Development
http://msdn2.microsoft.com/en-us/library/ms880960.aspx

[11] Discovering Bluetooth Devices Using Winsock
http://msdn2.microsoft.com/en-us/library/ms881713.aspx

[12] Creating a Connection to a Remote Device Using Winsock
http://msdn2.microsoft.com/en-us/library/ms881660.aspx

[13] Using the Waveform Audio Interface
http://msdn2.microsoft.com/en-us/library/ms903669.aspx

[14] Wave PCM soundfile format
http://ccrma.stanford.edu/CCRMA/Courses/422/projects/WaveFormat/

[15] Faraday Effect
http://en.wikipedia.org/wiki/Faraday_effect

9. Bilag

9.1. USE CASES 2

§ 0002 – Initialiser kommunikation 2
§ 0003 – Genetabler forbindelse 3
§ 0005 – Læsning fra EEPROM 4
§ 0006 – Skrivning til EEPROM 5
§ 0007 – Foretag Master målinger 6
§ 0008 – Foretag Opti målinger 7
§ 0009 – Oprettelse af data log 8
§ 0010 – Optagelse af voice log 9
§ 0011 – Afspilning af voice log 10
9.2. USE CASE REALISERING 11
§ 0002 – Initialiser kommunikation 11
§ 0003 – Genetabler forbindelse 12
§ 0005 – Læsning fra EEPROM 13
§ 0006 – Skrivning til EEPROM 14
§ 0007 – Foretag Master målinger 15
§ 0008 – Foretag Opti målinger 17
§ 0009 – Oprettelse af data log 20
§ 0010 – Optagelse af voice log 21
§ 0011 – Afspilning af voice log 22
9.3. USE CASE TEST 23
9.3.1. Use case § 0002 – Initialiser kommunikation 23
9.3.2. Use case § 0003 – Genetabler forbindelse 24
9.3.3. Use case § 0005 – Læsning fra EEPROM 25
9.3.4. Use case § 0006 – Skrivning til EEPROM 26
9.3.5. Use case § 0007 – Foretag Master målinger 27
9.3.6. Use case § 0008 – Foretag Opti målinger 28
9.3.7. Use case § 0009 – Oprettelse af data log 29
9.3.8. Use case § 0010 – Optagelse af voice log 30
9.3.9. Use case § 0011 – Afspilning af voice log 31
9.4. PROTOKOL 32
9.4.1. Flowchart 32
9.4.2. DISCOS Beskeder 34
9.5. EEPROM MEMORY MAP 39
9.5.1. Meta Map Konvertering 42
9.6. CD STRUKTUR 44

Figur liste

Figur 47 - System Sekvensdiagram: Initialiser kommunikation 2
Figur 48 - System Sekvensdiagram: Genetabler forbindelse 3
Figur 49 - System Sekvensdiagram: Læsning fra EEPROM 4
Figur 50 - System Sekvensdiagram: Skrivning til EEPROM 5
Figur 51 - System Sekvensdiagram: Foretag Master målinger 6
Figur 52 - System Sekvensdiagram: Foretag Opti målinger 7
Figur 53 - System Sekvensdiagram: Oprettelse af data log 8
Figur 54 - System Sekvensdiagram: Optagelse af voice log 9
Figur 55 - System Sekvensdiagram: Afspilning af voice log 10
Figur 56 - Sekvensdiagram: Initialiser kommunikation i Discman PDA. 11
Figur 57 - Sekvensdiagram: Initialiser kommunikation i DISCOS kommunikations emulator. 12
Figur 58 - Sekvensdiagram: Genetabler forbindelse 12
Figur 59 - Sekvensdiagram: Læsning fra EEPROM 13
Figur 60 - Sekvensdiagram: Skrivning til EEPROM 14
Figur 61 - Sekvensdiagram: Foretag Master målinger (del 1 af 2) 15
Figur 62 - Sekvensdiagram: Foretag Master målinger (del 2 af 2) 16
Figur 63 - Sekvensdiagram: Foretag Opti målinger (del 1 af 3) 17
Figur 64 - Sekvensdiagram: Foretag Opti målinger (del 2 af 3) 18
Figur 65 - Sekvensdiagram: Foretag Opti målinger (del 3 af 3) 19
Figur 66 - Sekvensdiagram: Oprettelse af data log 20
Figur 67 - Sekvensdiagram: Optagelse af voice log 21
Figur 68 - Sekvensdiagram: Afspilning af voice log 22
Figur 69 - Flowchart: Parser tilstandsmaskine (Discman PDA) 32
Figur 70 - Flowchart: Parser tilstandsmaskine (DISCOS kommunikations emulator) 33
Figur 71 - Specifikation af CAN pakkernes identifikator felt 35
Figur 72 - Flowchart: Konvertering af data fra software objekt til firmware data 42
Figur 73 - Flowchart: Konvertering af data fra firmware data til software objekt. 43

Tabel liste

Tabel 8 - Use Case §0002: Initialiser kommunikation 2
Tabel 9 - Use Case §0003: Genetabler forbindelse 3
Tabel 10 - Use Case §0005: Læsning fra EEPROM 4
Tabel 11 - Use Case §0006: Skrivning til EEPROM 5
Tabel 12 - Use Case §0007: Foretag Master målinger 6
Tabel 13 - Use Case §0008: Foretag Opti målinger 7
Tabel 14 - Use Case §0009: Oprettelse af data log 8
Tabel 15 - Use Case §0010: Optagelse af voice log 9
Tabel 16 - Use Case §0011: Afspilning af voice log 10
Tabel 17 - CAN extended frame format 34
Tabel 18 - EEPROM Meta Map 41

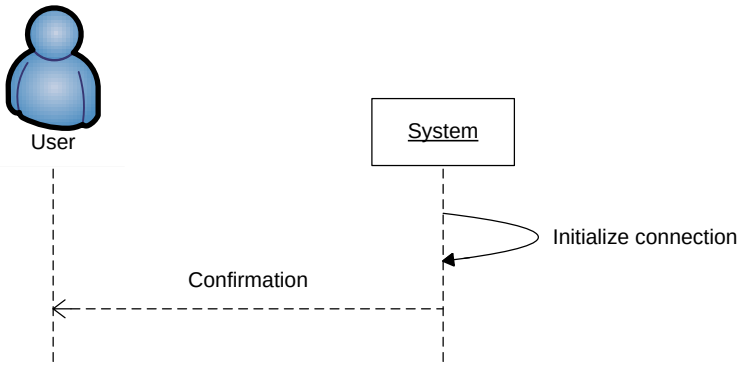
9.1. Use Cases

§ 0002 – Initialiser kommunikation

Denne use case vedrører initialiseringen af kommunikationen til DISCOS systemet.

ID	§ 0002 – Initialiser kommunikation.
Formål	At validere/initialisere forbindelsen til DISCOS systemet
Primær aktører	Bruger.
Motiver og interesser	At sikre at der er oprettet forbindelse til et reelt DISCOS system og dermed initialisere forbindelsen.
Trigger	Ved succesfuld oprettelse af forbindelse til en Bluetooth enhed.
Forudsætninger	Der er oprettet forbindelse til et potentielt DISCOS system.
Succesbetingelser	Initialiseringen fuldføres fejlfrit.
Fejlbetingelser	Initialiseringen fejler.
Hovedforløb	1. Forbindelsen valideres/initialiseres. 2. Der vises en bekræftelse på at forbindelsen er klar.
Udvidelser	
Alternativt forløb	2a. Initialiseringen fejler – forbindelsen til enheden afbrydes. Brugeren informeres om situationen og kan nu foretage følgende valg: • Foretage en ny søgning (Use case §0001). • Forsøge at oprette forbindelse til en ny enhed (Use case §0001 punkt 4). • Afslutte programmet.
Prioritet	Høj
Hyppighed	Benyttes hver gang der oprettes forbindelse til et potentielt DISCOS system.
Referencer	Use case: §0001

Tabel 8 - Use Case §0002: Initialiser kommunikation



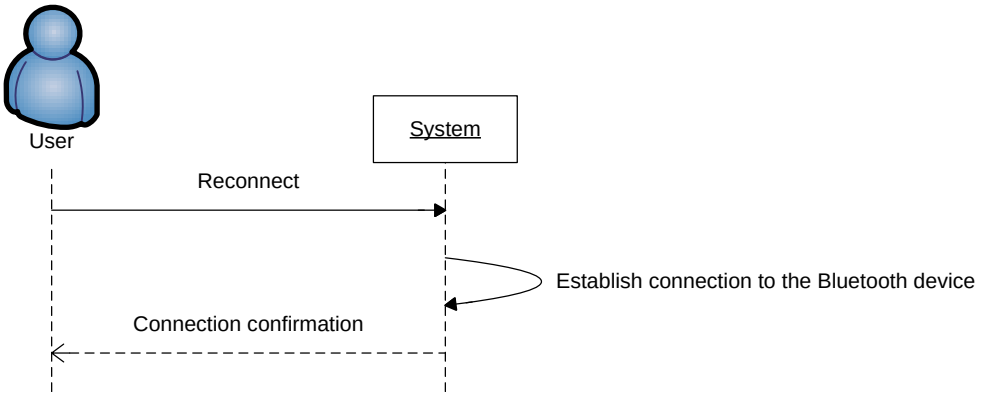
Figur 47 - System Sekvensdiagram: Initialiser kommunikation

§ 0003 – Genetabler forbindelse

Denne use case vedrører genetablering af forbindelsen til DISCOS systemet.

ID	§ 0003 – Genetabler forbindelse.
Formål	At genetablere den tabte forbindelse til DISCOS systemet
Primær aktører	Bruger.
Motiver og interesser	At gøre det muligt at genoptage vedligeholdelses proceduren ved tabt forbindelse.
Trigger	Forbindelsen til DISCOS systemet mistes og brugeren ønsker at genetablere forbindelsen.
Forudsætninger	Der har været oprettet forbindelse til DISCOS systemet, men forbindelsen er blevet afbrudt.
Succesbetingelser	Forbindelsen til DISCOS systemet genoprettes.
Fejlbetingelser	Der kan ikke genoprettes forbindelse til DISCOS systemet.
Hovedforløb	1. Brugeren vælger at genetablere forbindelsen. 2. Der etableres forbindelse til enheden. 3. Der vises en meddelelse omkring succesfuld oprettelse af forbindelse.
Udvidelser	
Alternativt forløb	2a. Der kan ikke oprettes forbindelse til den valgte enhed. Brugeren informeres om situationen og kan nu foretage følgende valg: • Forsøge at genetablere forbindelsen (punkt 1). • Foretage en ny søgning (Use case §0001).
Prioritet	Mellem
Hyppighed	Benyttes hver gang kommunikation til DISCOS systemet afbrydes.
Referencer	Use case: §0001

Tabel 9 - Use Case §0003: Genetabler forbindelse



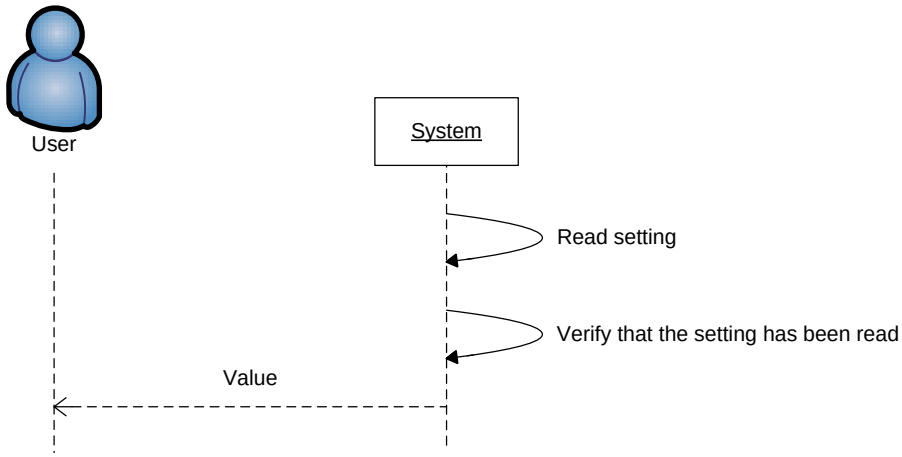
Figur 48 - System Sekvensdiagram: Genetabler forbindelse

§ 0005 – Læsning fra EEPROM

Denne use case vedrører læsning af enhedens EEPROM.

ID	§ 0005 – Læsning fra EEPROM.
Formål	At indlæse DISCOS enhedernes indstillinger.
Primær aktører	Bruger.
Motiver og interesser	At kunne se hvordan den enkelte DISCOS enhed er konfigureret.
Trigger	Søgningen efter enheder returnerer en eller flere enheder.
Forudsætninger	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret. Derudover er der fundet mindst en DISCOS enhed.
Succesbetingelser	Indstillingen indlæses fra EEPROM'en.
Fejlbetingelser	Indstillingen kan ikke indlæses.
Hovedforløb	1. Systemet anmoder om indstillingen fra EEPROM'en. 2. Systemet verificerer at værdien er blevet læst.
Udvidelser	
Alternativt forløb	1a. Der kan ikke læses fra EEPROM'en, da forbindelsen til DISCOS systemet er blevet afbrudt. Brugeren informeres om situationen og kan nu foretage følgende valg: • Forsøge at genetablere forbindelsen (Use Case §0003) • Foretage en ny søgning (Use case §0001). 2a. Værdien er ikke blevet læst fra EEPROM'en. Brugeren informeres om fejlen.
Prioritet	Høj
Hyppighed	Benyttes hver gang systemet initialiseres, for at indhente alle enhedsindstillinger.
Referencer	Use case: §0001, §0003

Tabel 10 - Use Case §0005: Læsning fra EEPROM



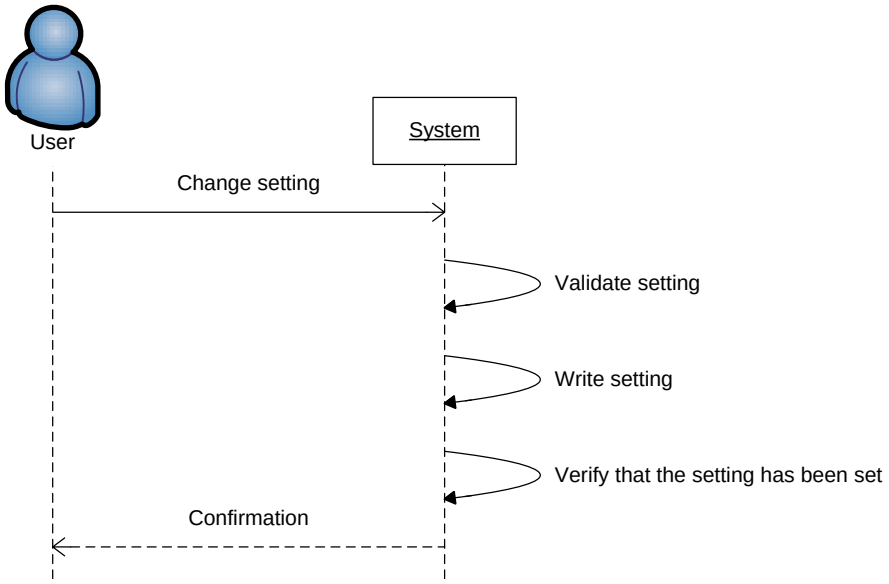
Figur 49 - System Sekvensdiagram: Læsning fra EEPROM

§ 0006 – Skrivning til EEPROM

Denne use case vedrører skrivning til enhedens EEPROM.

ID	§ 0006 – Skrivning til EEPROM.
Formål	At ændre på en af DISCOS enhedernes indstillinger.
Primær aktører	Bruger.
Motiver og interesser	At kunne ændre DISCOS enhedens opsætning.
Trigger	Brugeren ønsker at foretage en ændring i opsætningen af en DISCOS enhed.
Forudsætninger	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret.
Succesbetingelser	Indstillingen gemmes i EEPROM'en.
Fejlbetingelser	Indstillingen bliver ikke gemt.
Hovedforløb	1. Brugeren ændre en given indstilling. 2. Systemet validerer den nye indstilling. 3. Værdien skrives til EEPROM'en. 4. Systemet verificerer at værdien er blevet skrevet i EEPROM'en.
Udvidelser	3a. Indstillingen læses fra EEPROM'en (Use Case §0005).
Alternativt forløb	2a. Den nye indstilling kan ikke valideres.. Brugeren informeres om fejlen. 3a. Der kan ikke skrives til EEPROM'en, da forbindelsen til DISCOS systemet er blevet afbrudt. Brugeren informeres om situationen og kan nu foretage følgende valg: • Forsøge at genetablere forbindelsen (Use Case §0003) • Foretage en ny søgning (Use case §0001). 4a. Værdien er ikke blevet skrevet til EEPROM'en. Brugeren informeres om fejlen.
Prioritet	Høj
Hyppighed	Benyttes hver gang der skal foretages ændringer til DISCOS systemet.
Referencer	Use case: §0001, §0003, §0005

Tabel 11 - Use Case §0006: Skrivning til EEPROM



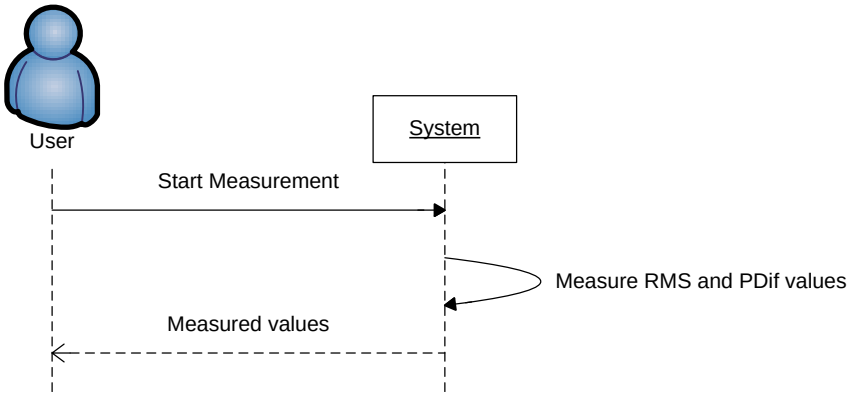
Figur 50 - System Sekvensdiagram: Skrivning til EEPROM

§ 0007 – Foretag Master målinger

Denne use case vedrører udføring af Master målinger.

ID	§ 0007 – Foretag Master målinger.
Formål	At foretage målinger af Master'ens RMS værdier.
Primær aktører	Bruger.
Motiver og interesser	At kunne modtage nogle data der beskriver, hvordan transformerens drift tilstand er.
Trigger	Der ønskes foretaget nogle målinger af hvordan transformeren opererer.
Forudsætninger	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret. Derudover er der fundet mindst en Master enhed.
Succesbetingelser	Alle målinger gennemføres.
Fejlbetingelser	En eller flere målinger kan ikke gennemføres.
Hovedforløb	1. Brugeren vælger at foretage målinger af Master enheden. 2. Systemet foretager målinger af RMS værdier. 3. De målte værdier vises for brugeren.
Udvidelser	
Alternativt forløb	2a. Der modtages ikke nogen målinger, da forbindelsen til DISCOS systemet er blevet afbrudt. Brugeren informeres om situationen og kan nu foretage følgende valg: • Forsøge at genetablere forbindelsen (Use Case §0003) • Foretage en ny søgning (Use case §0001). 2b. Der modtages ikke nogen målinger, da der er en fejl i CAN kommunikationen. Brugeren informeres om fejlen.
Prioritet	Høj
Hyppighed	Benyttes hver gang der skal foretages test målinger af DISCOS systemet.
Referencer	Use case: §0001, §0003

Tabel 12 - Use Case §0007: Foretag Master målinger



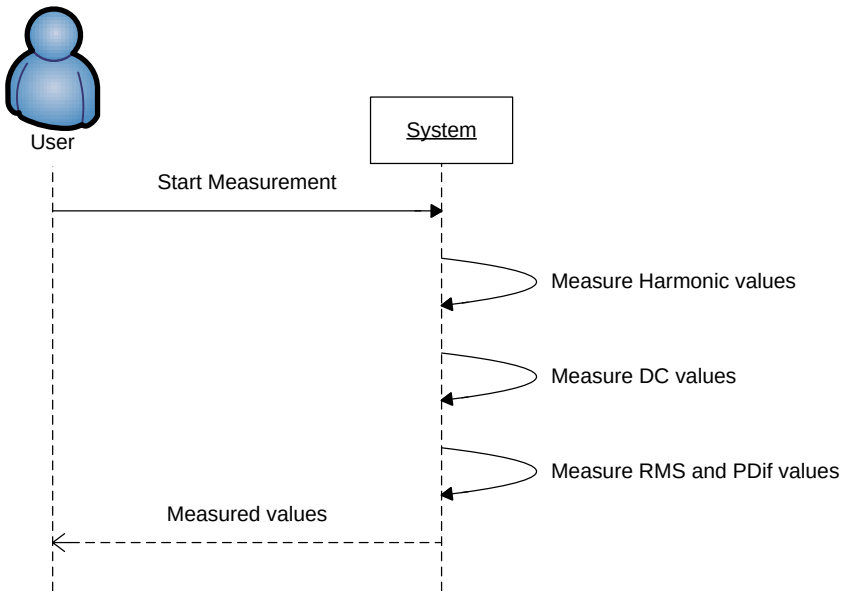
Figur 51 - System Sekvensdiagram: Foretag Master målinger

§ 0008 – Foretag Opti målinger

Denne use case vedrører udføring af Opti målinger.

ID	§ 0008 – Foretag Opti målinger.
Formål	At foretage målinger af Opti'ens Harmoniske værdier, DC værdier RMS værdier og faseforskelle.
Primær aktører	Bruger.
Motiver og interesser	At kunne modtage nogle data der beskriver, hvordan transformerens drift tilstand er.
Trigger	Der ønskes foretaget nogle målinger af hvordan transformeren opererer.
Forudsætninger	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret. Derudover er der fundet mindst en Opti enhed.
Succesbetingelser	Alle målinger gennemføres.
Fejlbetingelser	En eller flere målinger kan ikke gennemføres.
Hovedforløb	1. Brugeren vælger at foretage målinger af Opti enheden. 2. Systemet foretager målinger af de harmoniske værdier. 3. Systemet foretager målinger af DC værdierne. 4. Systemet foretager målinger af RMS værdier og faseforskelle. 5. De målte værdier vises for brugeren.
Udvidelser	
Alternativt forløb	2,3,4a. Der modtages ikke nogen målinger, da forbindelsen til DISCOS systemet er blevet afbrudt. Brugeren informeres om situationen og kan nu foretage følgende valg: • Forsøge at genetablere forbindelsen (Use Case §0003) • Foretage en ny søgning (Use case §0001). 2,3,4b. Der modtages ikke nogen målinger, da der er en fejl i CAN kommunikationen. Brugeren informeres om fejlen.
Prioritet	Høj
Hyppighed	Benyttes hver gang der skal foretages test målinger af DISCOS systemet.
Referencer	Use case: §0001, §0003

Tabel 13 - Use Case §0008: Foretag Opti målinger



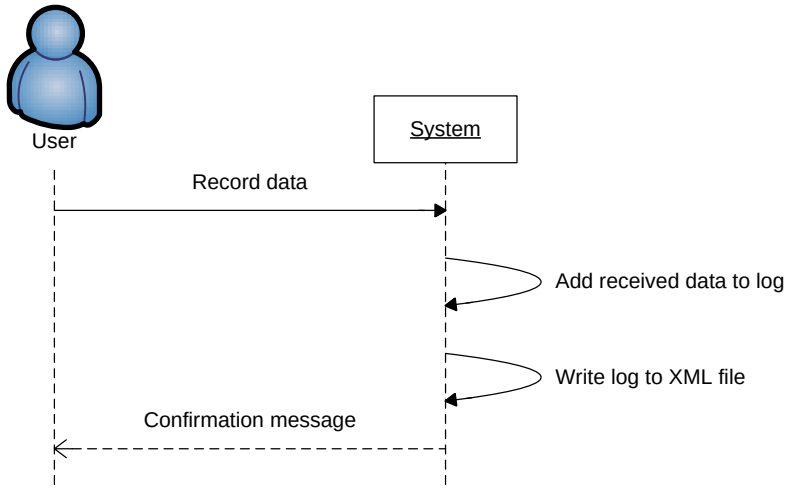
Figur 52 - System Sekvensdiagram: Foretag Opti målinger

§ 0009 – Oprettelse af data log

Denne use case vedrører oprettelsen af en log med data fra de modtagende målinger.

ID	§ 0009 – Oprettelse af data log.
Formål	At gemme de målte data i en XML fil.
Primær aktører	Bruger.
Motiver og interesser	At efterfølgende have mulighed for at analysere de målte data.
Trigger	Brugeren ønsker at oprette en log med de målte data.
Forudsætninger	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret. Derudover er der blevet foretaget målinger af enten en Master eller Opti enhed.
Succesbetingelser	De målte data gemmes til en XML fil.
Fejlbetingelser	De målte data gemmes ikke.
Hovedforløb	1. Brugeren vælger at gemme de målte data. 2. Systemet gemmer de enkelte modtagende målinger data i en log. 3. Log'en skrives til en XML fil. 4. Brugeren informeres om at de målte data er blevet gemt.
Udvidelser	
Alternativt forløb	
Prioritet	Lav
Hyppighed	Benyttes kun når der modtages nogle mistænkelige værdier fra Master eller Opti enheden.
Referencer	

Tabel 14 - Use Case §0009: Oprettelse af data log



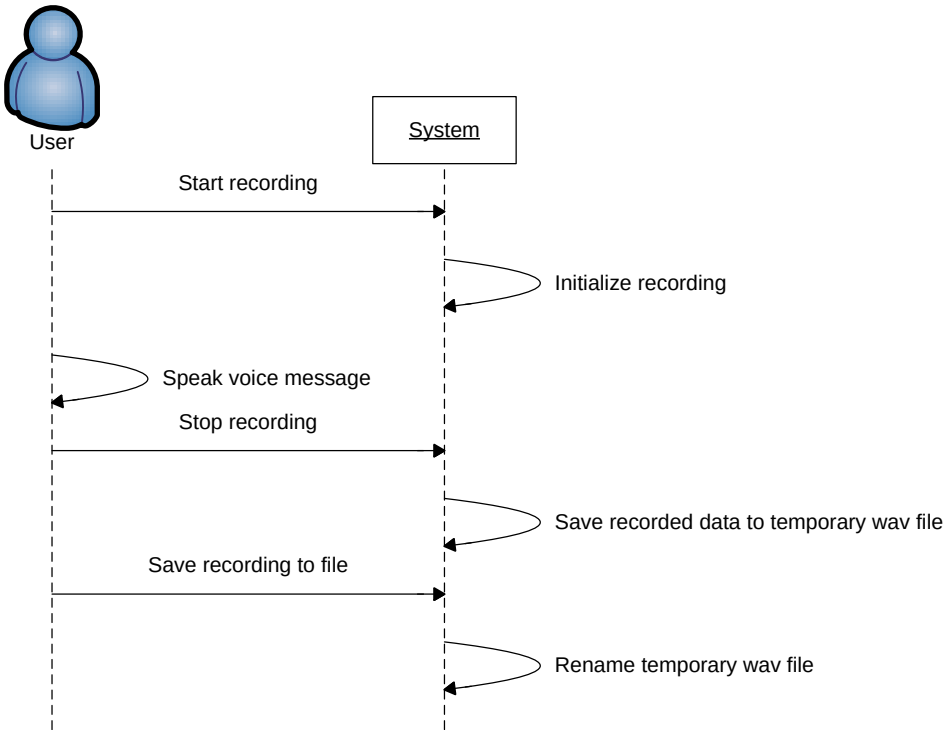
Figur 53 - System Sekvensdiagram: Oprettelse af data log

§ 0010 – Optagelse af voice log

Denne use case vedrører optagelsen af en voice log.

ID	§ 0010 – Optagelse af voice log.
Formål	At kunne gemme en indtalt besked.
Primær aktører	Bruger.
Motiver og interesser	At kunne indtale en besked på PDA'en som senere kan blive ajourført. Dette vil specielt være gavnligt ved kontrol besøg af adskillige transformerstationer på samme dag.
Trigger	Brugeren ønsker at indtale en besked.
Forudsætninger	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret.
Succesbetingelser	Der oprettes en wav fil med den indtalte besked.
Fejlbetingelser	Der oprettes ikke nogen fil.
Hovedforløb	1. Brugeren vælger at påbegynde optagelsen. 2. Systemet initialiserer og starter optagelsen. 3. Brugeren indtaler en besked. 4. Brugeren stopper optagelsen. 5. Systemets stopper optagelsen og gemmer de optagede data til en midlertidig wav fil. 6. Brugeren indtaster et filnavn og vælger at gemme optagelsen. 7. Systemet omdøber den midlertidige fil det ønskede filnavn.
Udvidelser	5a. Brugeren ønsker at høre den indtalte besked (§0011).
Alternativt forløb	2a. Der forefindes ikke nogen audio input enhed i PDA'en. Brugeren informeres om hændelsen og Use Case'en afsluttes.
Prioritet	Lav
Hyppighed	Benyttes hver gang der forekommer nogle afvigelser i kontrol besøget, der skal ajourføres.
Referencer	Use case: §0011

Tabel 15 - Use Case §0010: Optagelse af voice log



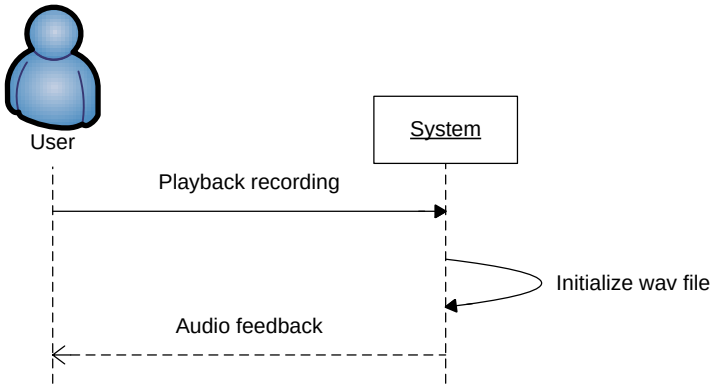
Figur 54 - System Sekvensdiagram: Optagelse af voice log

§ 0011 – Afspilning af voice log

Denne use case vedrører afspilningen af en optaget voice log.

ID	§ 0010 – Afspilning af voice log.
Formål	At kunne afspille en indtalt besked.
Primær aktører	Bruger.
Motiver og interesser	At kunne høre kvaliteten af den indtalte besked.
Trigger	Brugeren ønsker at høre den indtalte besked.
Forudsætninger	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret. Derudover er der blevet indtalt en besked via mikrofonen.
Succesbetingelser	Den indtalte besked afspilles.
Fejlbetingelser	Der indtalte besked kan ikke afspilles.
Hovedforløb	1. Brugeren vælger at afspille optagelsen. 2. Systemet initialiserer wav filen. 3. Systemet afspiller optagelsen.
Udvidelser	3a. Brugeren ønsker at pause afspilningen. Systemet stopper afspilningen midlertidig. • Brugeren kan nu genoprette eller stoppe afspilningen. 3b. Bruger ønsker at stoppe afspilningen. Systemet stopper afspilningen og Use Case'en afsluttes.
Alternativt forløb	2a. Der forefindes ikke nogen audio output enhed i PDA'en. Brugeren informeres om hændelsen og Use Case'en afsluttes.
Prioritet	Lav
Hyppighed	Benyttes hver gang en indtalt besked ønskes afspillet.
Referencer	

Tabel 16 - Use Case §0011: Afspilning af voice log

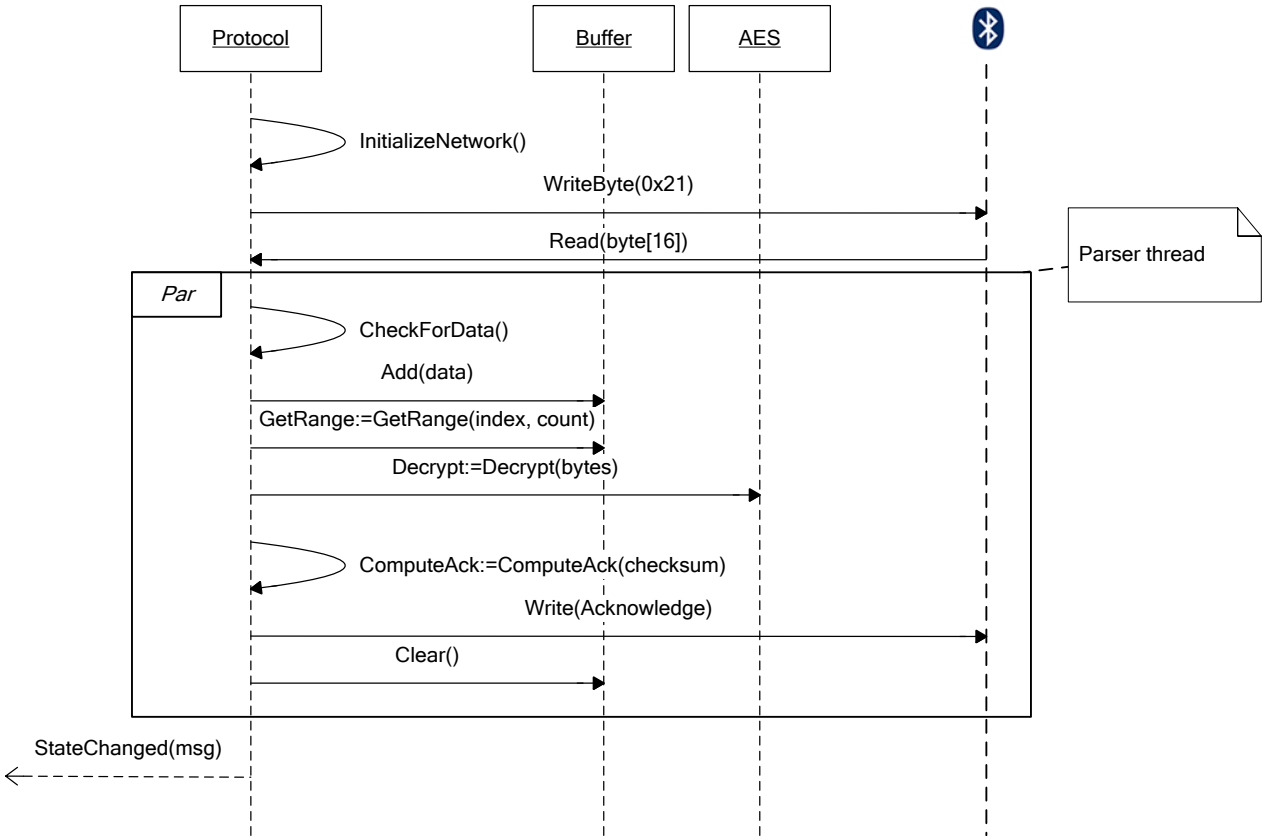


Figur 55 - System Sekvensdiagram: Afspilning af voice log

9.2. Use Case Realisering

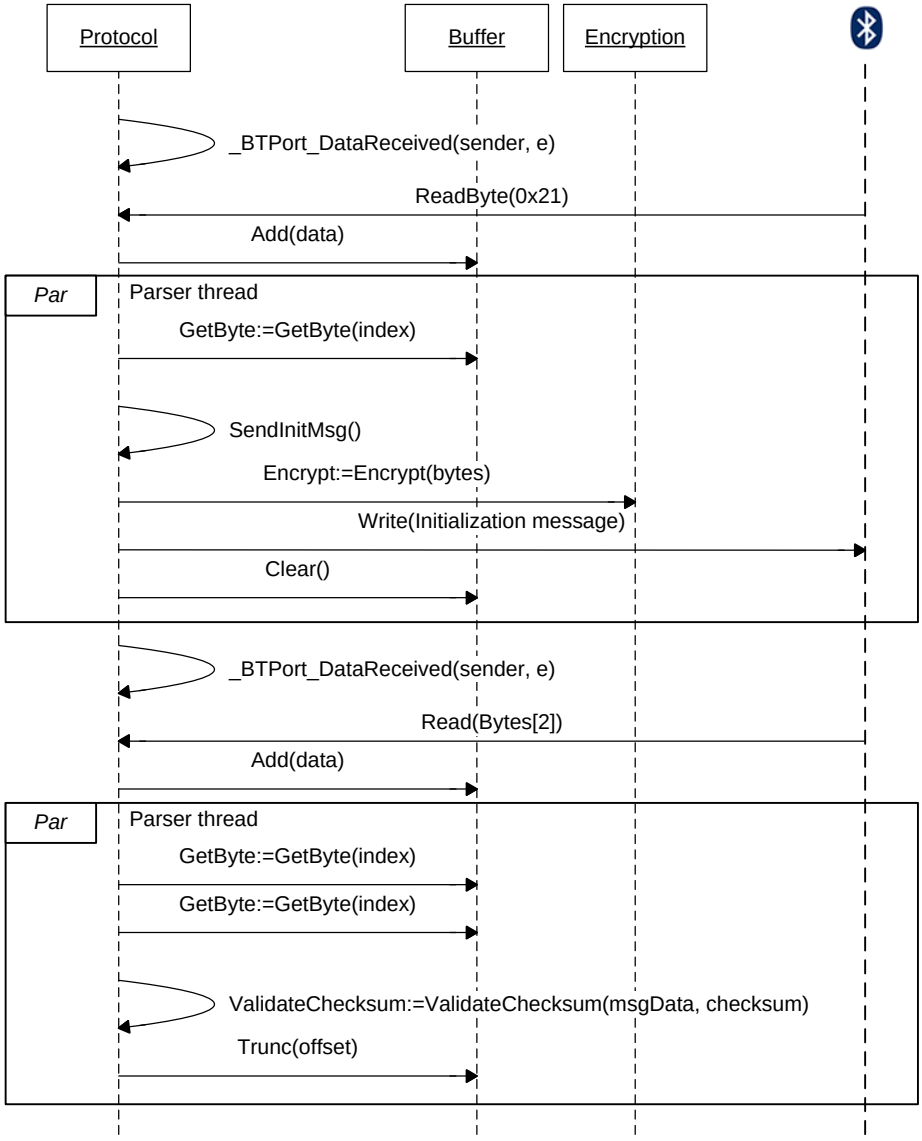
§ 0002 – Initialiser kommunikation

Discman PDA



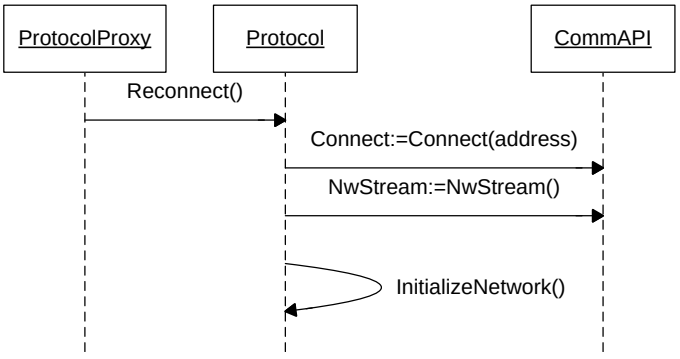
Figur 56 - Sekvensdiagram: Initialiser kommunikation i Discman PDA.

DISCOS kommunikations emulator



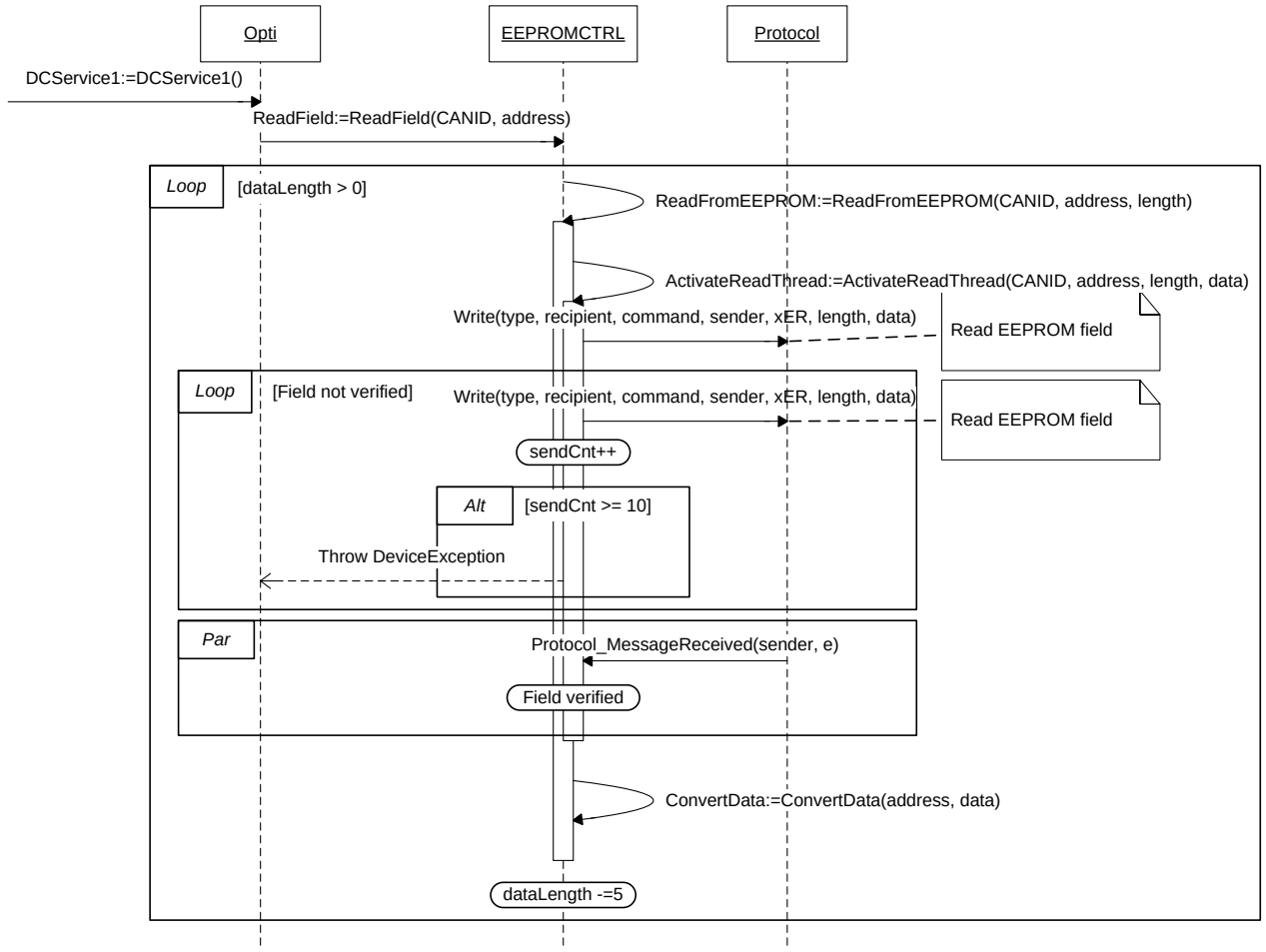
Figur 57 - Sekvensdiagram: Initialiser kommunikation i DISCOS kommunikations emulator.

§ 0003 – Genetabler forbindelse



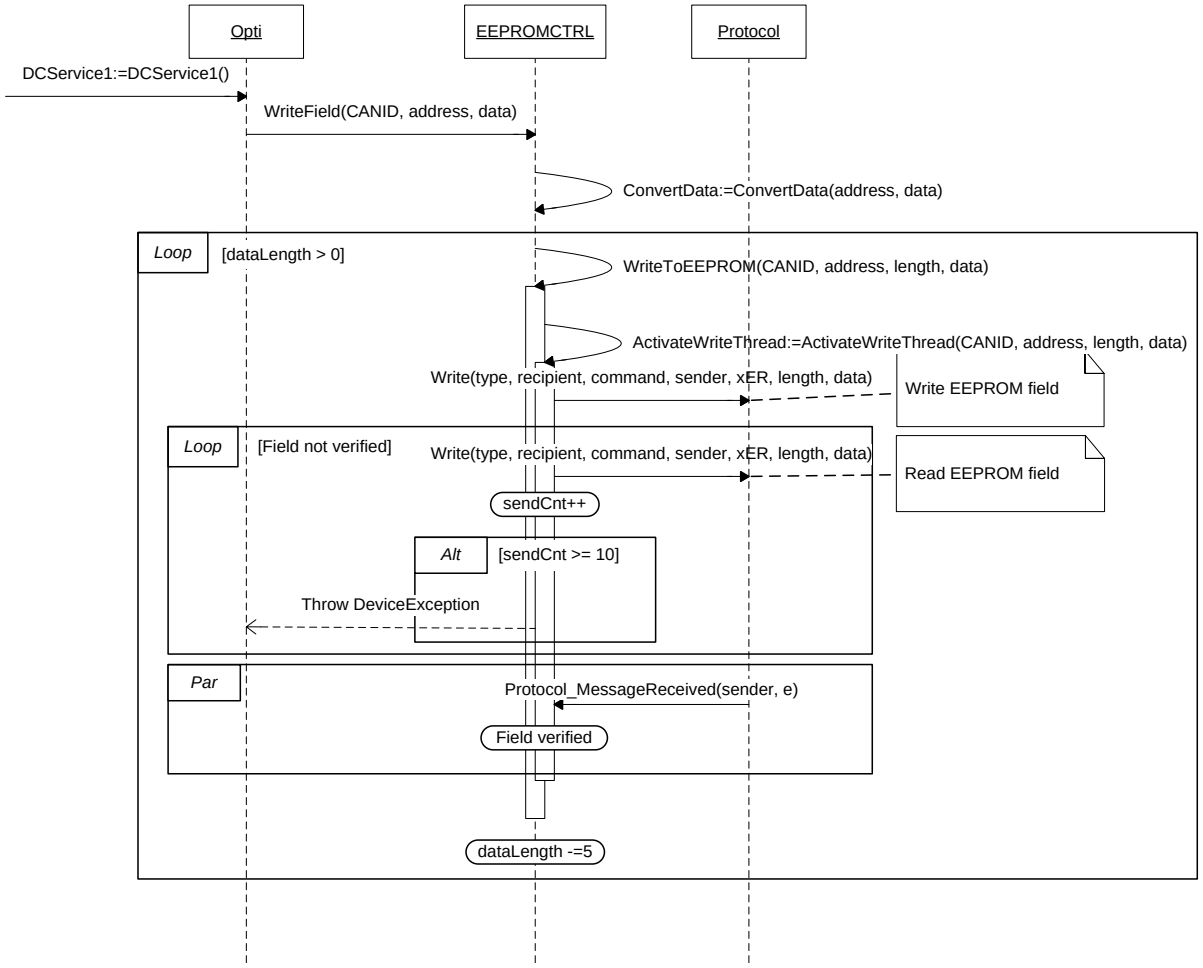
Figur 58 - Sekvensdiagram: Genetabler forbindelse

§ 0005 – Læsning fra EEPROM



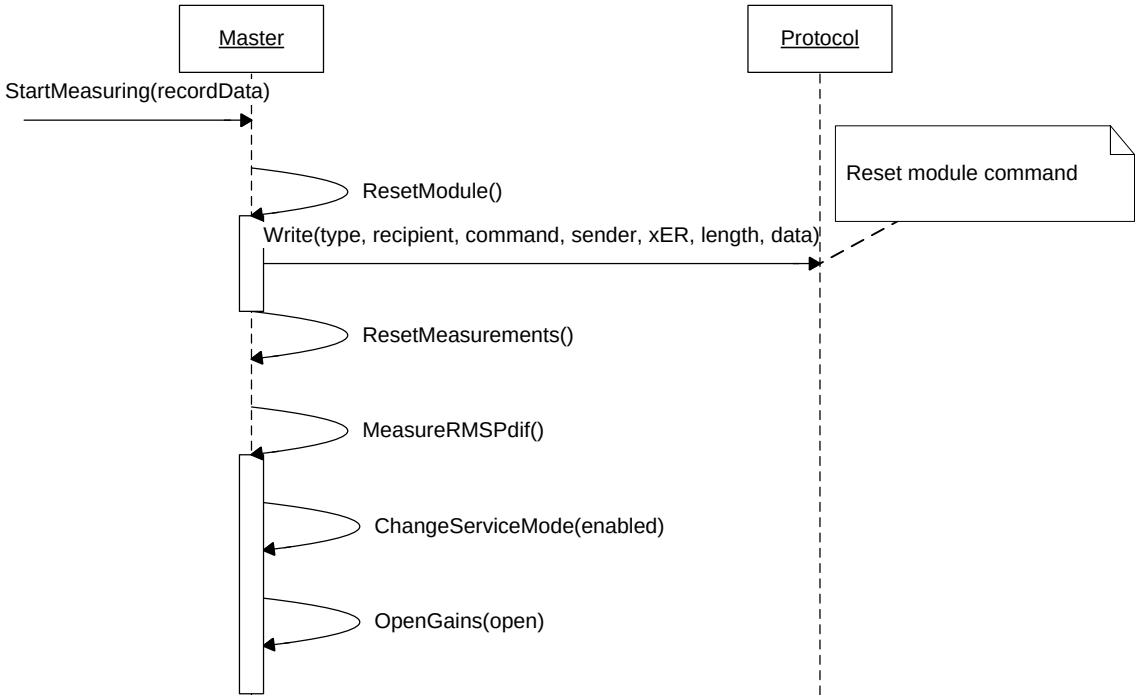
Figur 59 - Sekvensdiagram: Læsning fra EEPROM

§ 0006 – Skrivning til EEPROM

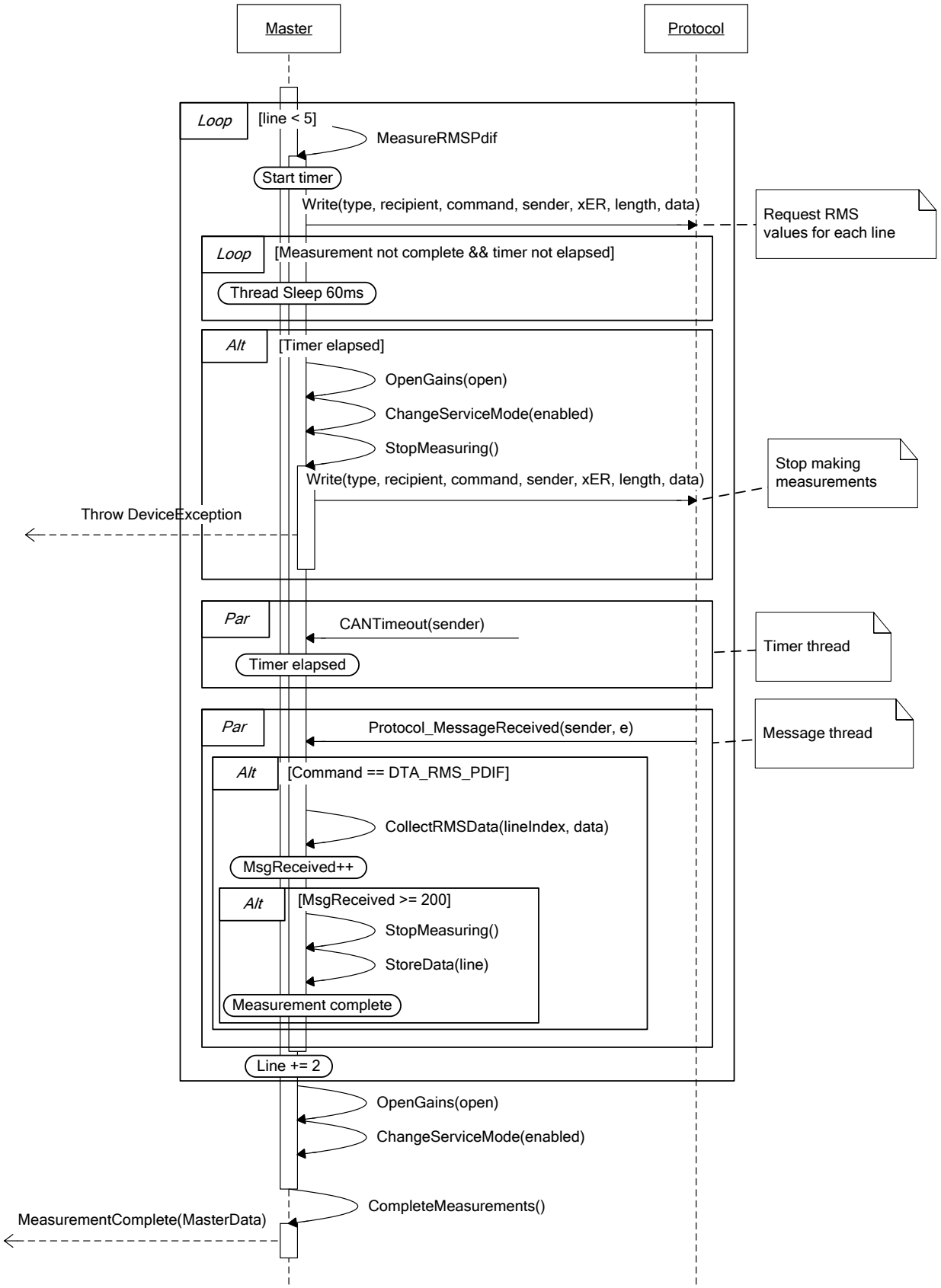


Figur 60 - Sekvensdiagram: Skrivning til EEPROM

§ 0007 – Foretag Master målinger

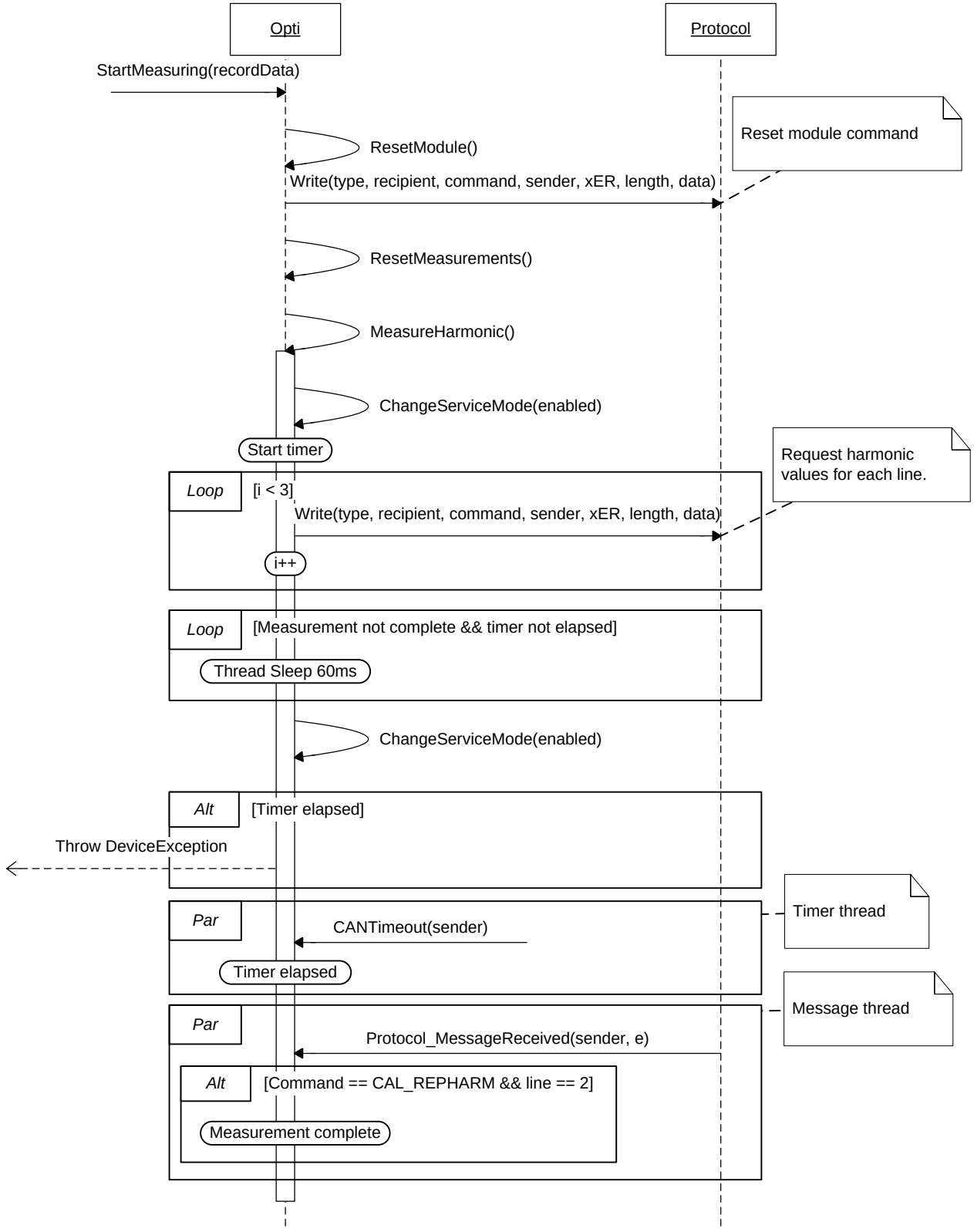


Figur 61 - Sekvensdiagram: Foretag Master målinger (del 1 af 2)

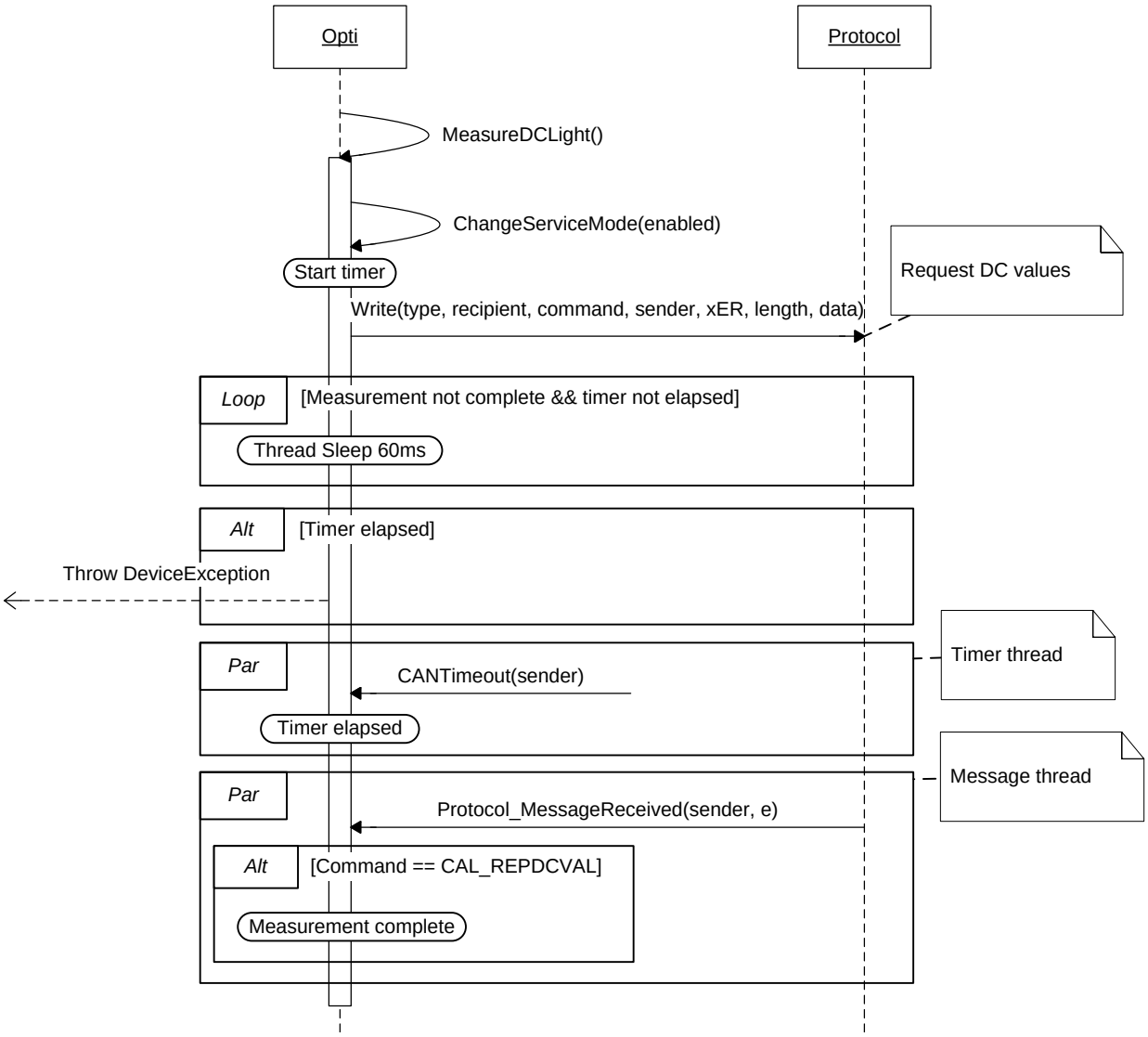


Figur 62 - Sekvensdiagram: Foretag Master målinger (del 2 af 2)

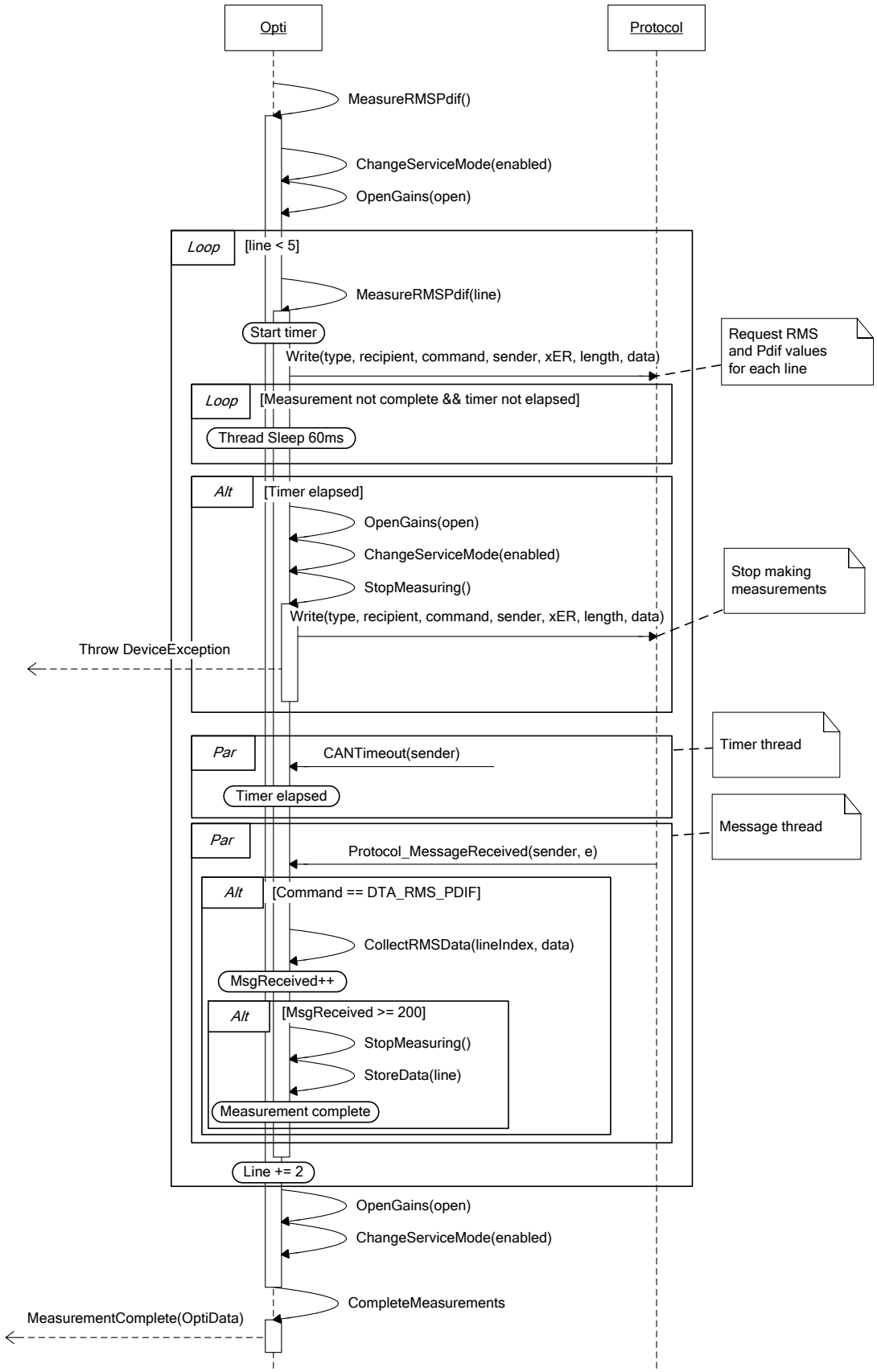
§ 0008 – Foretag Opti målinger



Figur 63 - Sekvensdiagram: Foretag Opti målinger (del 1 af 3)

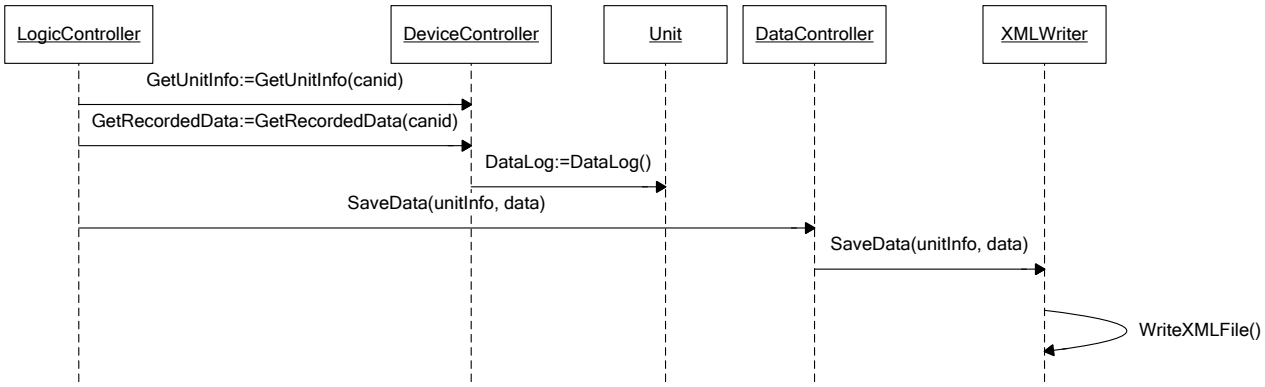


Figur 64 - Sekvensdiagram: Foretag Opti målinger (del 2 af 3)



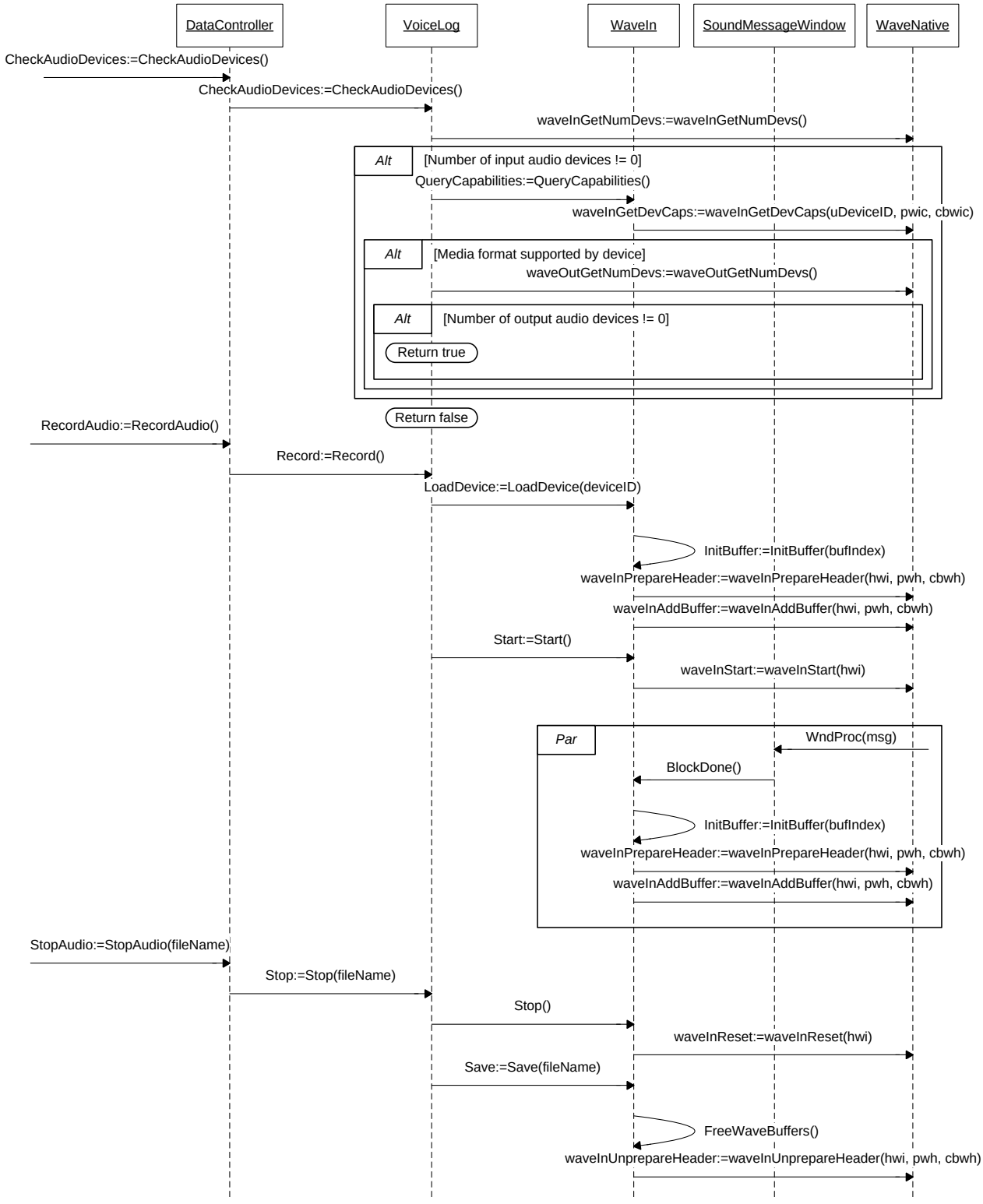
Figur 65 - Sekvensdiagram: Foretag Opti målinger (del 3 af 3)

§ 0009 – Oprettelse af data log



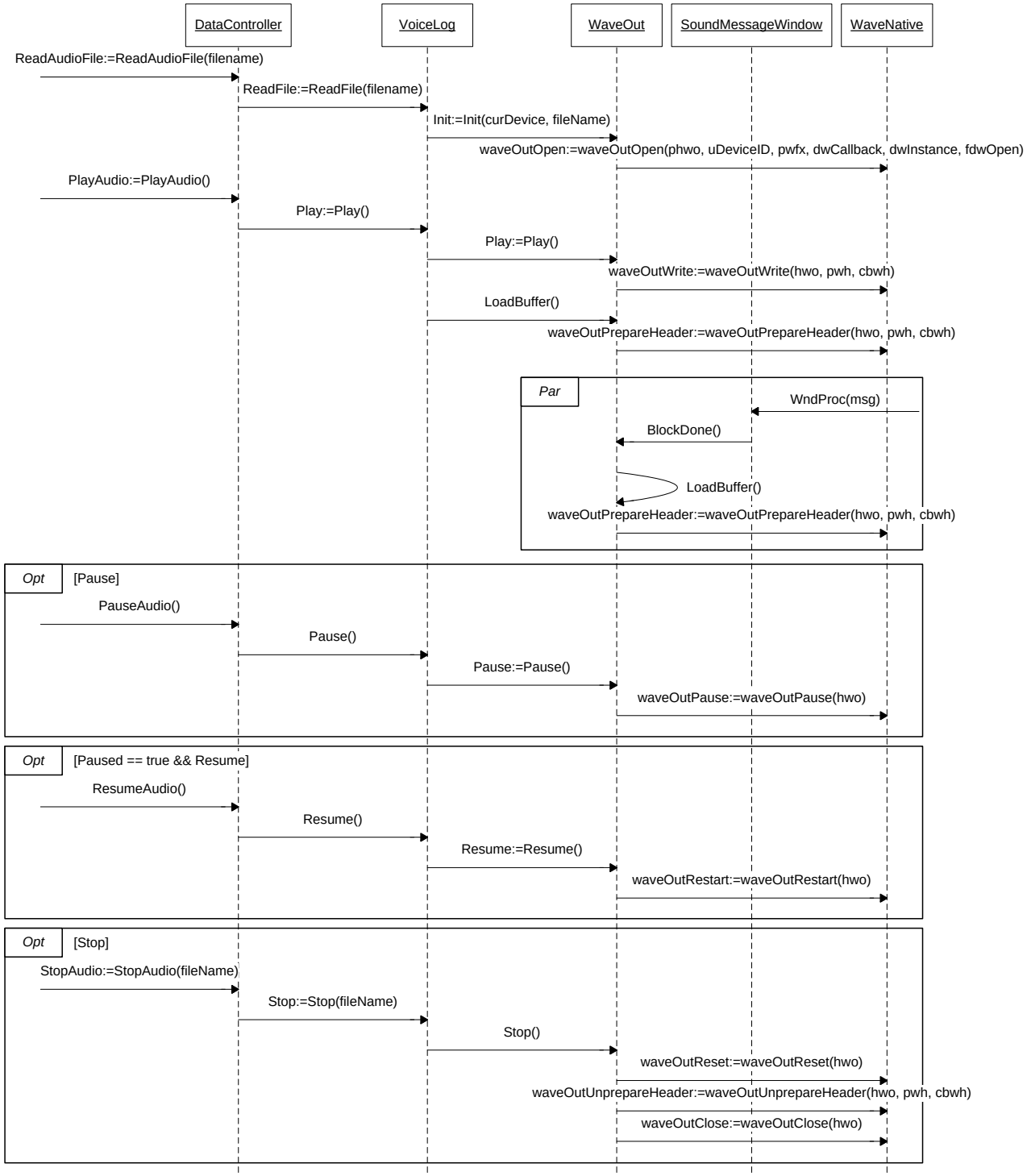
Figur 66 - Sekvensdiagram: Oprettelse af data log

§ 0010 – Optagelse af voice log



Figur 67 - Sekvensdiagram: Optagelse af voice log

§ 0011 – Afspilning af voice log



Figur 68 - Sekvensdiagram: Afspilning af voice log

9.3. Use Case Test

9.3.1. Use case § 0002 – Initialiser kommunikation

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	15-03-2007	2.0	1.0	GL
Titel:	Use case § 0002 – Initialiser kommunikation.			
Formål:	At teste funktionaliteten af use case §0002.			
Beskrivelse:	Efter at der er oprettet forbindelse til en Bluetooth enhed vil systemet forsøge at initialisere kommunikationen mellem PDA'en og enheden.			
Forudsætninger:	At der er 2 Bluetooth enheder inden for rækkevidde, en med forbindelse til et DISCOS system og en uden.			
Test cases:	**Hovedforløb:** 1. Der oprettes forbindelse til Bluetooth enheden med tilslutning til DISCOS systemet. 2. Forbindelsen valideres/initialiseres. 3. Der vises en bekræftelse på at forbindelsen er klar. **Alternativforløb:** 2a) Initialiseringen fejler 1. Der oprettes forbindelse til Bluetooth enheden der ikke har tilslutning til DISCOS systemet. 2. Initialiseringen fejler a. Forbindelsen afbrydes. b. Der vises en fejlbesked. c. Mulighed for at foretage en ny søgning. d. Mulighed for at oprette forbindelse til en ny enhed. e. Mulighed for at afslutte programmet.			
Tilladte afvigelser/fejl:	Ingen.			

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	15-03-2007	2.0	1.0	GL
Titel:	Use case § 0002 – Initialiser kommunikation.			
Resultater:	**Hovedforløb:** Kommunikationen blev initialiseret og der blev vist en bekræftelse. **Alternativforløb:** 2a) Forbindelsen blev afbrudt og der blev vist en fejlbesked. Det var dernæst muligt at fortage en ny søgning, at oprette forbindelse til en ny enhed samt at afslutte programmet.			
Konklusion:	Testen forløb succesfuldt.			

9.3.2. Use case § 0003 – Genetabler forbindelse

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	16-03-2007	3.0	1.0	GL
Titel:	Use case § 0003 – Genetabler forbindelse.			
Formål:	At teste funktionaliteten af use case §0003.			
Beskrivelse:	Efter at kommunikationen mellem PDA'en og DISCOS systemet er blevet afbrudt, forsøges der at genetablere den tabte forbindelse.			
Forudsætninger:	At der har været oprettet forbindelse til en Bluetooth enhed, men at denne er blevet afbrudt.			
Test cases:	**Hovedforløb:** 1. Bluetooth enheden aktiveres og placeres indenfor rækkevidde. 2. Der vælges at genetablere forbindelsen. 3. Der etableres forbindelse til enheden. 4. Der vises en meddelelse omkring succesfuld oprettelse af forbindelse. **Alternativforløb:** 3a)Der kan ikke oprettes forbindelse til den valgte enhed. 1. Bluetooth enheden deaktiveres. 2. Der vælges at genetablere forbindelsen. 3. Der kan ikke oprettes forbindelse til enheden. a. Fejlbesked vises. b. Mulighed for at forsøge at genetablere forbindelsen. c. Mulighed for ny søgning.			
Tilladte afvigelser/fejl:	Ingen.			

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	16-03-2007	3.0	1.0	GL
Titel:	Use case § 0003 – Genetabler forbindelse.			
Resultater:	**Hovedforløb:** Forbindelsen blev genetableret og der blev vist en bekræftelse. **Alternativforløb:** 3a) Der kunne ikke oprettes forbindelse til enheden – der blev vist en fejlbesked. Det var dernæst muligt at fortage en ny søgning, samt at afslutte programmet.			
Konklusion:	Testen forløb succesfuldt.			

9.3.3. Use case § 0005 – Læsning fra EEPROM

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	29-03-2007	4.0	1.0	GL
Titel:	Use case § 0005 – Læsning fra EEPROM.			
Formål:	At teste funktionaliteten af use case §0005.			
Beskrivelse:	De enkelte Opti indstillinger indlæses fra EEPROM'en.			
Forudsætninger:	At der er oprettet forbindelse til et DISCOS system, at kommunikationen er initialiseret og at der er fundet mindst en Opti.			
Test cases:	**Hovedforløb:** 1. DC Service værdien for linie 1 indlæses. 2. Systemet verificerer at værdien er blevet læst. **Alternativforløb:** 1a) Der kan ikke læses fra EEPROM'en, da forbindelsen er blevet afbrudt. 1. DC Service værdien for linie 1 indlæses. 2. Forbindelsen afbrydes. 3. Værdien kan ikke læses fra EEPROM'en. a. Fejlbesked vises. b. Mulighed for at genetablere forbindelsen. c. Mulighed for at gå tilbage til enhedssøgningen. 2a) Værdien er ikke blevet læst fra EEPROM'en, da der er en fejl i CAN kommunikationen. 1. CAN kommunikationen til Opti enheden afbrydes. 2. DC Service værdien for linie 1 indlæses. 3. Værdien er ikke blevet læst fra EEPROM'en. a. Brugeren informeres om fejlen.			
Tilladte afvigelser/fejl:	Ingen.			

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	29-03-2007	4.0	1.0	GL
Titel:	Use case § 0005 – Læsning fra EEPROM.			
Resultater:	**Hovedforløb:** Værdien blev skrevet til EEPROM'en. **Alternativforløb:** 2a) Der forekom en exception og programmet blev afsluttet. 3a) Værdien kunne ikke skrives til EEPROM'en, da forbindelsen til DISCOS systemet blev afbrudt – der blev vist en fejlbesked. Det var dernæst muligt at forsøge at genetablere forbindelsen, samt at gå tilbage til enhedssøgningen. 4a) Værdien blev ikke skrevet til EEPROM'en – der blev vist en fejl meddelelse.			
Konklusion:	Testen forløb succesfuldt.			

9.3.4. Use case § 0006 – Skrivning til EEPROM

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	29-03-2007	5.0	1.0	GL
Titel:	Use case § 0006 – Skrivning til EEPROM.			
Formål:	At teste funktionaliteten af use case §0006.			
Beskrivelse:	Der ændres ved konfigurationen af Opti enheden ved at skrive en ny værdi i EEPROM'en.			
Forudsætninger:	At der er oprettet forbindelse til et DISCOS system, at kommunikationen er initialiseret og at der er fundet mindst en Opti.			
Test cases:	**Hovedforløb:** 1. DC Service værdien for linie 1 ændres til et tilfældigt heltal. 2. Systemet validerer den nye indstilling. 3. Værdien skrives til EEPROM'en. 4. Systemet verificerer at værdien er blevet skrevet i EEPROM'en. **Alternativforløb:** 2a) Den nye indstilling kan ikke valideres. 1. DC Service værdien for linie 1 ændres til et komma tal. 2. Den nye indstilling kan ikke valideres. a. Fejlbesked vises. 3a) Der kan ikke skrives til EEPROM'en, da forbindelsen er blevet afbrudt. 1. DC Service værdien for linie 1 ændres til et tilfældigt heltal. 2. Systemet validerer den nye indstilling. 3. Forbindelsen afbrydes. 4. Værdien kan ikke skrives til EEPROM'en. a. Fejlbesked vises. b. Mulighed for at genetablere forbindelsen. c. Mulighed for at gå tilbage til enhedssøgningen. 4a) Værdien er ikke blevet skrevet til EEPROM'en, da der er en fejl i CAN kommunikationen. 1. DC Service værdien for linie 1 ændres til et tilfældigt heltal. 2. Systemet validerer den nye indstilling. 3. CAN kommunikationen til Opti enheden afbrydes. 4. Værdien skrives til EEPROM'en. 5. Værdien er ikke blevet skrevet til EEPROM'en. b. Brugeren informeres om fejlen.			
Tilladte afvigelser/fejl:	Alternativforløb 2a er ikke blevet implementeret og vil derfor ikke blive varetaget.			

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	29-03-2007	5.0	1.0	GL
Titel:	Use case § 0006 – Skrivning til EEPROM.			
Resultater:	**Hovedforløb:** Værdien blev skrevet til EEPROM'en. **Alternativforløb:** 2a) Der forekom en exception og programmet blev afsluttet. 3a) Værdien kunne ikke skrives til EEPROM'en, da forbindelsen til DISCOS systemet blev afbrudt – der blev vist en fejlbesked.			

	Det var dernæst muligt at forsøge at genetablere forbindelsen, samt at gå tilbage til enhedssøgningen. 4a) Værdien blev ikke skrevet til EEPROM'en – der blev vist en fejl meddelelse.
Konklusion:	Testen forløb som forventet.

9.3.5. Use case § 0007 – Foretag Master målinger

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	29-03-2007	7.0	1.0	GL
Titel:	Use case § 0007 – Foretag Master målinger.			
Formål:	At teste funktionaliteten af use case §0007.			
Beskrivelse:	Der foretages målinger af Master enheden.			
Forudsætninger:	At der er oprettet forbindelse til et DISCOS system, at kommunikationen er initialiseret og at der er fundet mindst en Master.			
Test cases:	**Hovedforløb:** 4. Start målingen af Master enheden. 5. Systemet foretager målinger af RMS værdier. 6. De målte værdier vises. **Alternativforløb:** 2a) Der modtages ikke nogen målinger, da forbindelsen til DISCOS systemet er blevet afbrudt. 7. Start målingen af Master enheden. 8. Forbindelsen afbrydes. 9. Der kan ikke foretages målinger. d. Fejlbesked vises. e. Mulighed for at genetablere forbindelsen. f. Mulighed for at gå tilbage til enhedssøgningen. 2b) Der modtages ikke nogen målinger, da der er en fejl i CAN kommunikationen. 1. Start målingen af Master enheden. 2. CAN kommunikationen til Master enheden afbrydes. 3. Der kan ikke foretages målinger. a. Fejlbesked vises.			
Tilladte afvigelser/fejl:	Ingen.			

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	29-03-2007	7.0	1.0	GL
Titel:	Use case § 0007 – Foretag Master målinger.			
Resultater:	**Hovedforløb:** Alle målinger foretages og vises til brugeren. **Alternativforløb:** 2a) Målingerne kunne ikke foretages, der blev vist en fejlbesked. Det var dernæst muligt at forsøge at genetablere forbindelsen, samt at gå tilbage til enhedssøgningen. 2b) Målingerne kunne ikke foretages, der blev vist en fejlbesked.			
Konklusion:	Testen forløb succesfuldt.			

9.3.6. Use case § 0008 – Foretag Opti målinger

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	29-03-2007	6.0	1.0	GL
Titel:	Use case § 0008 – Foretag Opti målinger.			
Formål:	At teste funktionaliteten af use case §0008.			
Beskrivelse:	Der foretages målinger af Opti enheden.			
Forudsætninger:	At der er oprettet forbindelse til et DISCOS system, at kommunikationen er initialiseret og at der er fundet mindst en Opti.			
Test cases:	**Hovedforløb:** 1. Start målingen af Opti enheden. 2. Systemet foretager målinger af de harmoniske værdier. 3. Systemet foretager målinger af DC værdierne. 4. Systemet foretager målinger af RMS værdier og faseforskelle. 5. De målte værdier vises. **Alternativforløb:** 2,3,4a) Der modtages ikke nogen målinger, da forbindelsen til DISCOS systemet er blevet afbrudt. 1. Start målingen af Opti enheden. 2. Forbindelsen afbrydes. 3. Der kan ikke foretages målinger. a. Fejlbesked vises. b. Mulighed for at genetablere forbindelsen. c. Mulighed for at gå tilbage til enhedssøgningen. 2,3,4b) Der modtages ikke nogen målinger, da der er en fejl i CAN kommunikationen. 1. Start målingen af Opti enheden. 2. CAN kommunikationen til Opti enheden afbrydes. 3. Der kan ikke foretages målinger. a. Fejlbesked vises.			
Tilladte afvigelser/fejl:	Ingen.			

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	29-03-2007	6.0	1.0	GL
Titel:	Use case § 0008 – Foretag Opti målinger.			
Resultater:	**Hovedforløb:** Alle målinger foretages og vises til brugeren. **Alternativforløb:** 2,3,4a) Målingerne kunne ikke foretages, der blev vist en fejlbesked. Det var dernæst muligt at forsøge at genetablere forbindelsen, samt at gå tilbage til enhedssøgningen. 2,3,4b) Målingerne kunne ikke foretages, der blev vist en fejlbesked.			
Konklusion:	Testen forløb succesfuldt.			

9.3.7. Use case § 0009 – Oprettelse af data log

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	13-04-2007	8.0	1.0	GL
Titel:	Use case § 0009 – Oprettelse af data log.			
Formål:	At teste funktionaliteten af use case §0008.			
Beskrivelse:	De målte data gemmes til en XML fil.			
Forudsætninger:	At der er oprettet forbindelse til et DISCOS system, at kommunikationen er initialiseret og at der er blevet foretaget målinger af en Master eller Opti enhed.			
Test cases:	**Hovedforløb:** 1. Gem målinger. 2. Systemet gemmer de enkelte modtagende målinger data i en log. 3. Log'en skrives til en XML fil. 4. En bekræftelse vises.			
Tilladte afvigelser/fejl:	Bekræftelsen kommer til udtryk ved at den benyttede knap bliver tilgængelig igen.			

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	13-04-2007	8.0	1.0	GL
Titel:	Use case § 0008 – Oprettelse af data log.			
Resultater:	**Hovedforløb:** De målte data gemmes til en XML fil og systemet vender tilbage til samme tilstand som før testen.			
Konklusion:	Testen forløb succesfuldt.			

9.3.8. Use case § 0010 – Optagelse af voice log

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	13-04-2007	10.0	1.0	GL
Titel:	Use case § 0010 – Optagelse af voice log.			
Formål:	At teste funktionaliteten af use case §0010.			
Beskrivelse:	Der indtales en voice log besked.			
Forudsætninger:	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret.			
Test cases:	**Hovedforløb:** 1. Start optagning. 2. Systemet initialiserer og starter optagelsen. 3. Indtal en test besked. 4. Stop optagelsen. 5. Systemets stopper optagelsen og gemmer de optagede data til en midlertidig wav fil. 6. Gem optagelsen. 7. Systemet omdøber den midlertidige fil det ønskede filnavn. **Alternativforløb:** 2a) Der forefindes ikke nogen audio input enhed i PDA'en. Benyt en PDA uden mikrofon. 1. Start optagningen. 2. Der vises en fejlmeddelelse.			
Tilladte afvigelser/fejl:	Det er ikke muligt at teste det alternative forløb, da der ikke er en PDA uden mikrofon til rådighed.			

Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	13-04-2007	10.0	1.0	GL
Titel:	Use case § 0010 – Optagelse af voice log.			
Resultater:	**Hovedforløb:** Optagningen fuldføres succesfuldt og der oprettes en wav svarende til det specificerede filnavn. **Alternativforløb:** 2a) Blev ikke testet.			
Konklusion:	Testen forløb som forventet.			

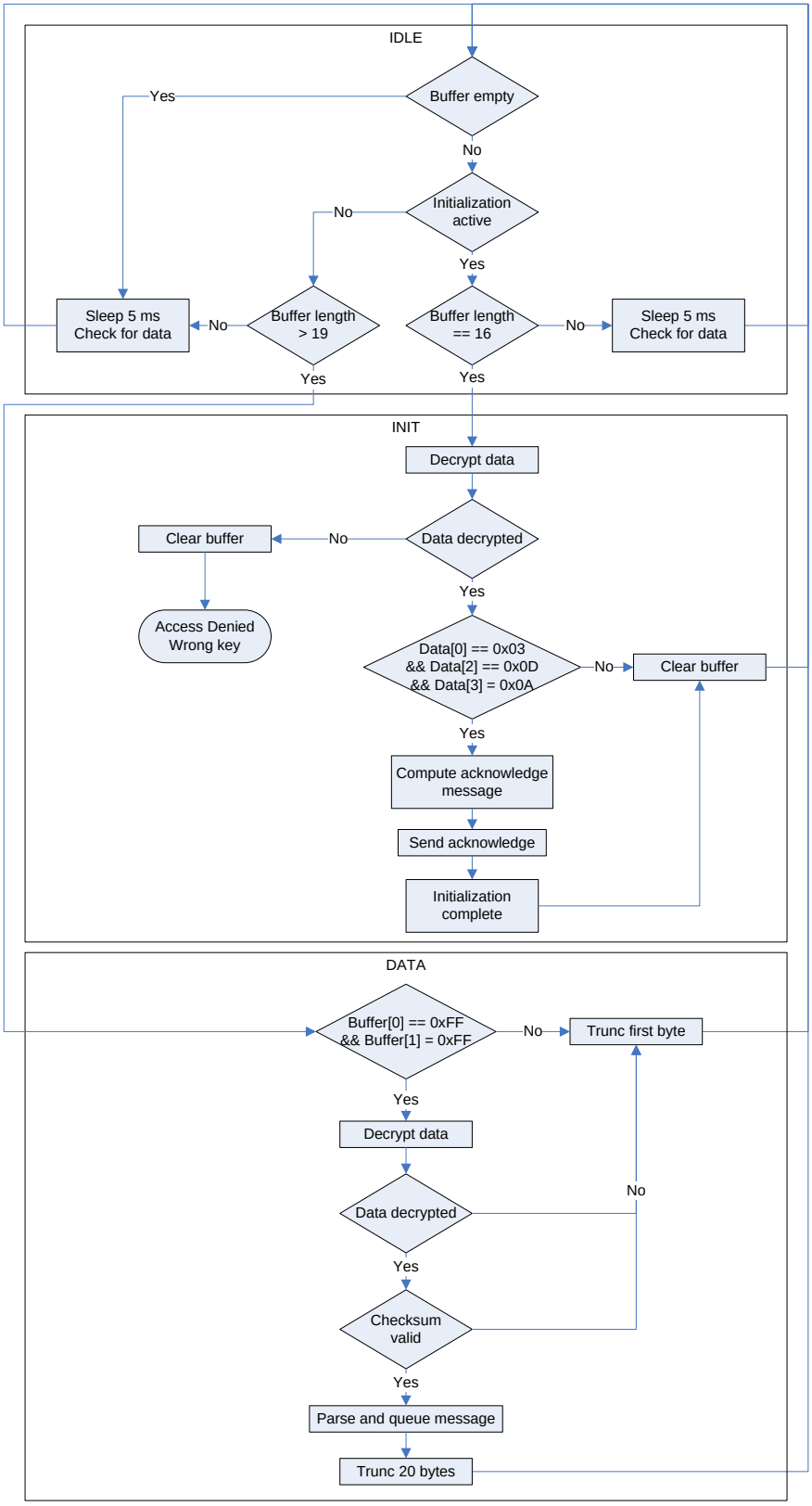
9.3.9. Use case § 0011 – Afspilning af voice log

Testplan				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	13-04-2007	11.0	1.0	GL
Titel:	Use case § 0011 – Optagelse af voice log.			
Formål:	At teste funktionaliteten af use case §0010.			
Beskrivelse:	Der indtales en voice log besked.			
Forudsætninger:	Der er oprettet forbindelse til et DISCOS system og kommunikationen er blevet initialiseret. Derudover er der blevet optaget en voice log besked.			
Test cases:	**Hovedforløb:** 1. Start afspilning. 2. Systemet initialiserer wav filen. 3. Systemet afspiller optagelsen. **Udvidelser:** 3a) Brugeren ønsker at pause afspilningen. 1. Start afspilning. 2. Systemet initialiserer wav filen. 3. Systemet afspiller optagelsen. 4. Sæt afspilningen på pause et par sekunder. 5. Genoptag afspilningen. 3b) Brugeren ønsker at stoppe afspilningen. 1. Start afspilning. 2. Systemet initialiserer wav filen. 3. Systemet afspiller optagelsen. 4. Stop afspilningen. **Alternativforløb:** 2a) Der forefindes ikke nogen audio output enhed i PDA'en. Benyt en PDA uden lydkort. 1. Start afspilningen. 2. Der vises en fejlmeddelelse.			
Tilladte afvigelser/fejl:	Det er ikke muligt at teste det alternative forløb, da der ikke er en PDA uden lydkort til rådighed.			

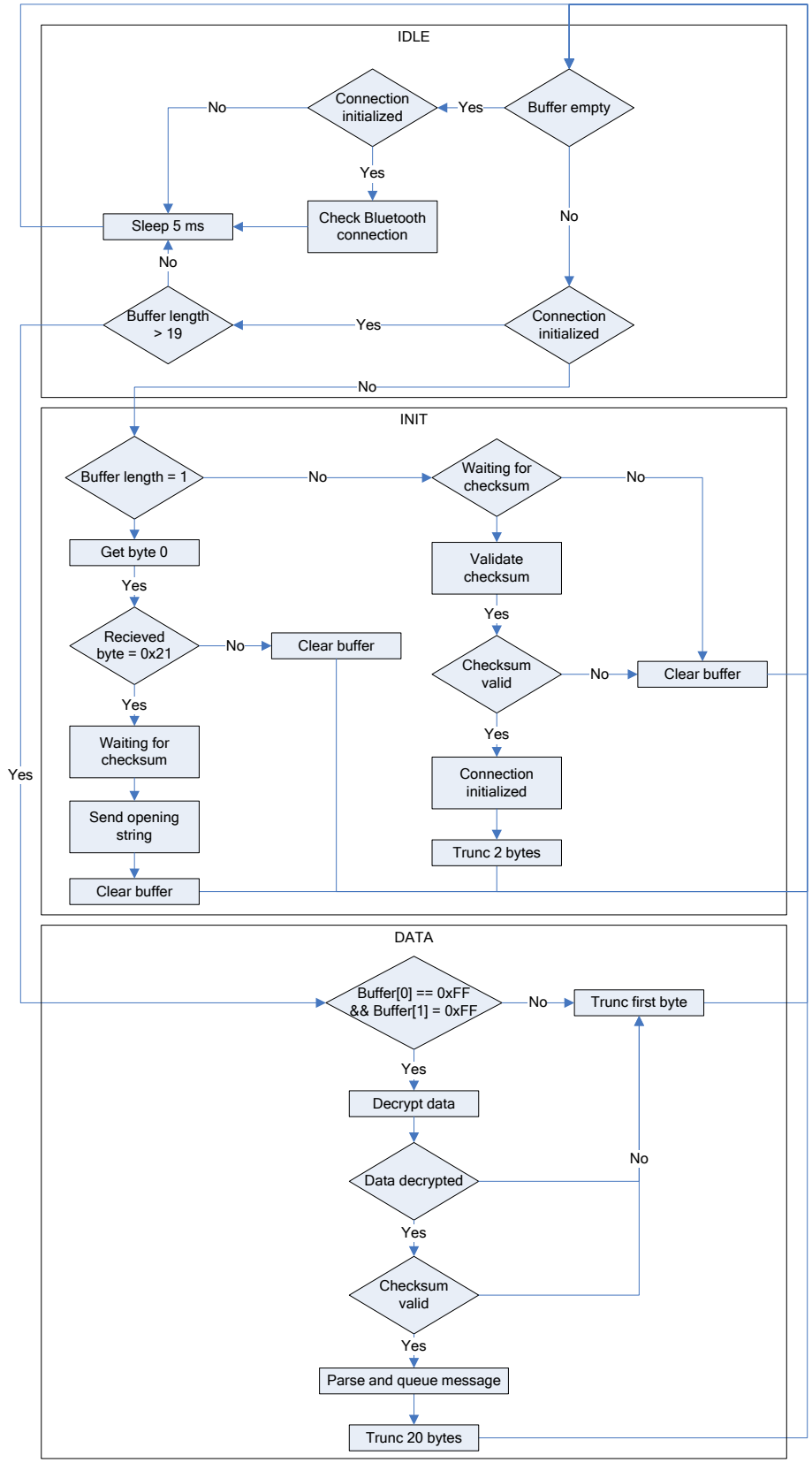
Testrapport				
Testtype:	**Dato:**	**Test nr.:**	**Version:**	**Udført af:**
Use case test	13-04-2007	11.0	1.0	GL
Titel:	Use case § 0011 – Afspilning af voice log.			
Resultater:	**Hovedforløb:** Afspilningen startede og stoppede svarende til den optagede besked. **Udvidelser:** 3a) Det var muligt at pause og genoptage afspilningen. 3b) Det var muligt at stoppe afspilningen før den endte. **Alternativforløb:** 2a) Blev ikke testet.			
Konklusion:	Testen forløb som forventet.			

9.4. Protokol

9.4.1. Flowchart



Figur 69 - Flowchart: Parser tilstandsmaskine (Discman PDA)



Figur 70 - Flowchart: Parser tilstandsmaskine (DISCOS kommunikations emulator)

9.4.2. DISCOS Beskeder

DISCOS CAN protokollen er baseret på det udvidede CAN format, hvilket består af et 29 bit id felt, et 8 byte datafelt og framing (rammestrukturen).

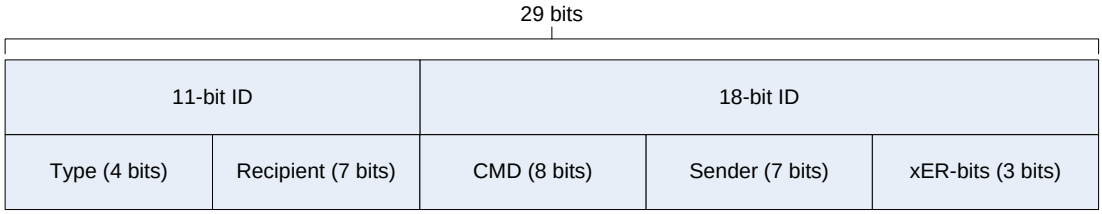
Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier A	11	First part of the (unique) identifier for the data
Substitute remote request (SRR)	1	Must be recessive (1)Optional
Identifier extension bit (IDE)	1	Must be recessive (1)Optional
Identifier B	18	Second part of the (unique) identifier for the data
Remote transmission request (RTR)	1	Must be dominant (0)
Reserved bits (r0, r1)	2	Reserved bits (it must be set dominant (0), but accepted as either dominant or recessive)
Data length code (DLC)	4	Number of bytes of data (0-8 bytes)
Data field	0-8 bytes	Data to be transmitted (length dictated by DLC field)
CRC	15	Cyclic Redundancy Check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

Tabel 17 - CAN extended frame format[15]

Da det er netværkslaget der varetager al framing vil dette ikke blive uddybet yderligere, og der vil i stedet blive fokuseret på identifikator og data feltet.

[15] Reference: [5]

Identifikator felt



Figur 71 - Specifikation af CAN pakkernes identifikator felt

Type

Angiver besked typen og specificerer prioriteten, jo lavere værdi jo højere prioritet).

0011: ID konfigurations beskeder.
1000: Normal drift beskeder.
1001: Kalibrerings beskeder.
1010: Data beskeder.
1011: Debug beskeder.

Recipient

CAN id'et på modtager enheden.

0: Udsender til alle enheder.
1: PC.
4-126: Andre enheder (Master og Opti).
127: Ikke konfigurerede enheder.

CMD

Angiver kommando type. En liste med de relevante kommandoer er vedhæftet i bilag 9.4.2.

Sender

CAN id'et på sender enheden.

1: PC
4-126: Andre enheder (Master og Opti).

xER bits

Speciel operation bits med følgende indkodning:

Bit x:

Reserveret til fremtidig brug.

Bit E:

End Of Text (ETX) bit. Denne bit benyttes til at indikere at en given pakke er den sidste i en række af pakker.

-0-: Beskeden er del af en serie beskeder og ikke den sidste.
-1-: Beskeden er enten en enkelt besked eller den sidste i en række.

Bit R:

--0: Beskeden er ikke en fjern anmodning.

--1: Beskeden er en fjern anmodning. Modtager skal svare med det passende data.

ID Konfigurations beskeder (0011)

Kommandoer

Navn	Recipient	CMD	Sender	xER	Data	Len	Beskrivelse
CNF_REQIDS	0000000	00000001	xxxxxxx	000	-	0	Anmoder alle enheder om at sende deres ID og type.
CNF_REPID	xxxxxxx	00000010	xxxxxxx	000	QQQTMP	6	Respons til CNF_REQIDS.
CNF_NEWUNITREQ	xxxxxxx	00000011	xxxxxxx	000	-	0	Anmoder ukonfigurerede enheder om at sende deres ID.
CNF_NEWUNITREP	xxxxxxx	qqqqqqqq	qqqqqqq	qqq	XT	2	Respons til CNF_NEWUNITREQ.

Vigtigt: Det er kun CNF_NEWUNITREP der må indeholde en data længde på 2 bytes, da disse bytes benyttes til at identificere kommandoen.

Signaturforklaring

Bytes:

Q: Unikt serienummer.

T: Enhedstype: 0 = Master, 1 = Opti.

X: Benyttes ikke.

M: Master position.
Master: Dets egen position. Opti: Positionen af den Master de er tilkoblet.

P: Opti position.
Master: Benyttes ikke. Opti: Dets position under Master enheden.

N: Master enhedens ID.

Bits

x: ID'et på den enhed der modtager eller sender beskeden.

q: Unikt serienummer.

Normal drift beskeder (1000)

Kommandoer

Navn	CMD	xER	Data	Len	Beskrivelse
NOP_SERVICE_ON	00000011	000	-	0	Start service tilstand.
NOP_SELECT	00001010	000	B	1	Enheden starter/stopper med at blinke.

Signaturforklaring

Bytes:

B: Blink status: 0 = Off, 1 = On

S: Angiver om strøm niveauet er under EE_10KVMIN (True/False).

Kalibrerings beskeder (1001)

Kommandoer

Navn	CMD	xER	Data	Len	Beskrivelse
CAL_RMS_PDIFSS	00000100	000	Z	1	Start/Stop med at sende RMS/faseforskelle.
CAL_GAINLVLLOCK	00001100	000	C	1	Åben/Lås forstærker niveau.
CAL_REQHARM	00001101	000	H	1	Anmoder om harmoniske værdier.
CAL_REPHARM	00001110	000	HNNMM	5	Respons til CAL_REQHARM.
CAL_SERVICE_OFF	00001111	000	-	0	Stop service tilstand.
CAL_REQDCVAL	00010001	000	-	0	Anmoder om DC værdier.
CAL_REPDCVAL	00010010	000	W1W2W3	6	Respons til CAL_REQDCVAL.

Signaturforklaring

Bytes:

Q: Oprindelse for x-times. 0 = spænding, 1 = strøm.

C: 0 = åben, 1 = lås.

Z: Linie oprindelse. 0 = V1/I1, 2 = V2/I2, 4 = V3/I3, 6 = stop.

S: Kilde til kalibrering. 0 = V1, 1 = I1, 2 = V2, 3 = I2, 4 = V3, 5 = I3, 6 = stop kalibrering.

L: Forstærknings niveau for kalibrerings konstanter (1-4, 1 = lavest, 4 = højest).

H: Harmonisk oprindelse. 0 = L1, 1 = L2, 2 = L3.

N: Niveau 1 grænseværdi, little-endian (gennemsnit over 10min * H-faktor).

M: Niveau 2 grænseværdi, little-endian (sidst målte værdi).

W1: DC værdi for linie 1 (2 bytes), little-endian.

W2: DC værdi for linie 2 (2 bytes), little-endian.

W3: DC værdi for linie 3 (2 bytes), little-endian.

Data beskeder (1010)

Kommandoer

Navn	CMD	xER	Data	Len	Beskrivelse
DTA_RMS_PDIF	00000101	000	QAABBPPP	8	Beregnet RMS og faseforskelle.

Signaturforklaring

Bytes:

L: Oprindelse af X-times, 0 = spænding, 1 = strøm.

M: X-time fra linie 1, little-endian.

N: X-time fra linie 2, little-endian.

O: X-time fra linie 3, little-endian..

P: Faseforskel, little-endian.

Z: Linie oprindelse af beregnet værdi, 0 = V1/I1, 2 = V2/I2, 4 = V3/I3.

Q: vvvvcsss.

A: RMS for spændings linie.

B: RMS for strøms linie.

Bits

v: Forstærker niveau for spænding. 1 = niveau 1, 2 = niveau 2, 4 = niveau 3, 8 = niveau 4.

c: Forstærker niveau for strøm.

s: Oprindelse af målt data. 0 = V1, 1 = I1, 2 = V2, 3 = I2, 4 = V3, 5 = I3.

Debug beskeder (1011)

Kommandoer

Navn	CMD	xER	Data	Len	Beskrivelse
DBG_EEINOUT	00000010	000	AALDDDDD	8	Skriv data til EEPROM.
DBG_EEINOUT	00000010	001	AAL	3	Læs data fra EEPROM.
DBG_RESETMOD	00000100	000	M	1	Reset modul.

Signaturforklaring

Bytes:

A: Adresse i hukommelsen hvor der skal læses fra/ skrives til.

D: Data der skal skrives til hukommelsen.

L: Antal bytes der skal læses/skrives.

M: Nummeret på det modul der skal reset'es. 0 = "func".

9.5. EEPROM Memory map

Feltnavn	Adresse	Størrelse	Type og konv. faktor.	Beskrivelse
EE_DCVAL1	0x080	2 Bytes	Uint – x*1	DC værdi for L1.
EE_DCVAL2	0x082	2 Bytes	Uint – x*1	DC værdi for L2.
EE_DCVAL3	0x084	2 Bytes	Uint – x*1	DC værdi for L3.
EE_HARM_LONG_TIME	0x107	2 Bytes	Uint – x*1	Lang gennemsnits tid (sek.).
EE_HARM_TRIGGER_LEVEL_DIFF	0x109	2 Bytes	Uint – x*1	Forskellen mellem lang og kort gennemsnits tid, hvilket aktiverer en alarm. -80% af denne værdi deaktiverer alarmen igen (ampere).
EE_HARM_SHORT_TIME	0x10B	1 Byte	Float – x*10	Kort gennemsnits tid i (sek./10).
EE_BATT_CHARGE_TIME	0x10C	4 Bytes	Uint – x*3600000	Tidsinterval for hvor længe batteriet oplades i opladning/afladning cyklusen (ms).
EE_DCVAL_ALARM3	0x114	2 Bytes	Uint – x*1	DC alarm værdi for L3.
EE_DCVAL_ALARM2	0x116	2 Bytes	Uint – x*1	DC alarm værdi for L2.
EE_DCVAL_ALARM1	0x118	2 Bytes	Uint – x*1	DC alarm værdi for L1.
EE_DCVAL_SERVICE3	0x11A	2 Bytes	Uint – x*1	DC service værdi for L3.
EE_DCVAL_SERVICE2	0x11C	2 Bytes	Uint – x*1	DC service værdi for L2.
EE_DCVAL_SERVICE1	0x11E	2 Bytes	Uint – x*1	DC service værdi for L1.
EE_BOALARM1NAME	0x120	8 Bytes	String	Navnet på Binær Output alarm 1.
EE_BOALARM2NAME	0x128	8 Bytes	String	Navnet på Binær Output alarm 2.
EE_DEGREES_DISPLACE3	0x134	3 Bytes	Uint – x*(10000/18)	Forskydning af grader beregnet som 100 ns for L3.
EE_DEGREES_DISPLACE2	0x137	3 Bytes	Uint – x*(10000/18)	Forskydning af grader beregnet som 100 ns for L2.
EE_DEGREES_DISPLACE1	0x13A	3 Bytes	Uint – x*(10000/18)	Forskydning af grader beregnet som 100 ns for L1.
EE_TRANSFORM_FACT	0x13D	2 Bytes	Float – x*1024	Konverterings faktor mellem 4 KV og 10 KV værdier.
EE_RKI_T_LOW	0x13F	3 Bytes	Uint – x*(10000/18)	MTA - 90° i 100 ns.
EE_RKI_T_HIGH	0x142	3 Bytes	Uint – x*(10000/18)	MTA + 90° i 100 ns.
EE_RKI_ASCII_TRUE	0x145	1 Byte	Char – x*1	ASCII karakter til at indikere om T_low < T_x < T_high er sandt.
EE_RKI_ASCII_FALSE	0x146		Char – x*1	ASCII karakter til at indikere om T_low < T_x < T_high er falskt.
EE_VECTORDISPLACE	0x147	3 Bytes	Int – x*(10000/18)	Vektor forskydning i 100 ns.
EE_DEGREES_CONST3	0x14E	2 Bytes	Int – x*1	Grad konstant for L3.
EE_DEGREES_CONST2	0x150	2 Bytes	Int – x*1	Grad konstant for L2.
EE_DEGREES_CONST1	0x152	2 Bytes	Int – x*1	Grad konstant for L1.
EE_SECVOL	0x154	2 Bytes	Uint – x*100	Sekundær spænding.
EE_VECTORGROUP	0x156	1 Byte	Uint – x*1	Vektor gruppe.
EE_CURRENTSTEP	0x157	1 Byte	Uint – x*1	Aktuel transformer trin.
EE_STEPSIZE	0x158	1 Byte	Float – x*10	Trin størrelse.
EE_RATEDVOL	0x159	1 Byte	Float – x*10	Nominel primær spænding.

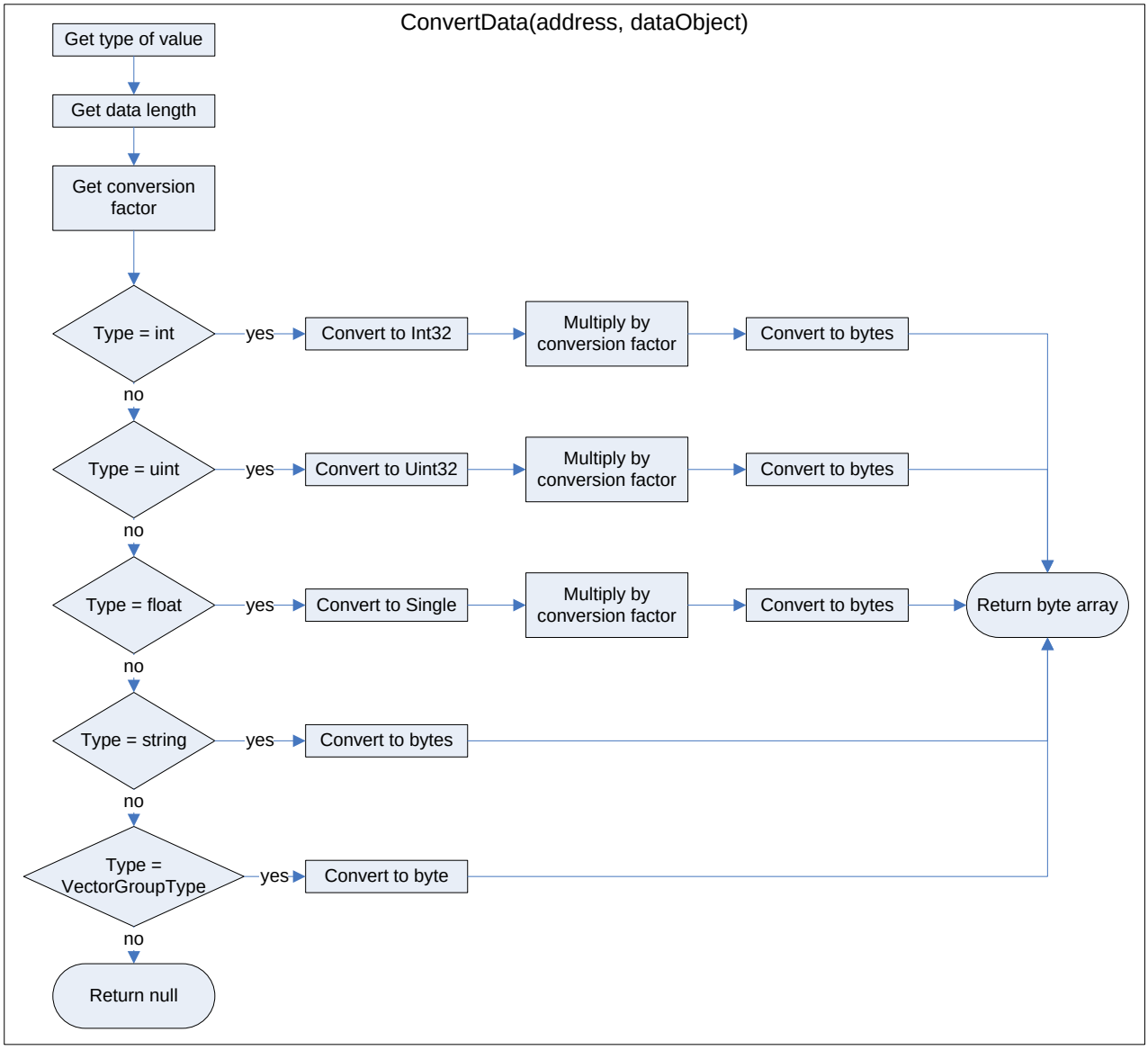
EE_SYSTEM_STATE	0x15A	1 Byte	Uint – x*1	System status nummer.
EE_SYSTEM_ERROR_CODE	0x15B	1 Byte	Uint – x*1	Fejlkode.
EE_TRAFOTYPE	0x15E	1 Byte	Uint – x*1	Binær transformer type.
EE_FORMAL_VECTORGROUP	0x15F	1 Byte	Uint – x*1	Indikerer dobbelt fase byt.
EE_SYSGROUNDING	0x161	2 Bytes	Uint – x*1	Type af system jording.
EE_MTA	0x163	2 Bytes	Uint – x*100	Maksimal drejningsmoments vinkel.
EE_SCIRCINDICLVL	0x165	3 Bytes	Uint – x*100	Kortslutnings alarm niveau.
EE_10KVMIN	0x16A	1 Byte	Uint – x*1	Grænseværdi for strøm (ampere).
EE_K_SYM	0x16B	1 Byte	Float – x*256	Forholdet mellem minimum og maksimum strøm faser.
EE_REMOTEDISC	0x16E	1 Byte	Uint – x*1	Automatisk afbrydnings værdi (0 = slået fra, 1 = slået til).
EE_FIELDTYPE	0x16F	1 Byte	Uint – x*1	Felt type (0 = slået fra, 1 = linie, 2 = transformer).
EE_PING_TIME	0x178	2 Bytes	Uint – x*1	Tidsinterval mellem hver modul ping (24 - 720 timer).
EE_RST_BUT_STATE	0x17A	1 Byte	Int – x*1	Genstart knappens tilstand ved nedlukning.
EE_OP_INTERVAL	0x187	3 Bytes	Float – x*60000	Åben fase interval (ms).
EE_VBATT_DIS_VOL	0x197	1 Byte	Uint – x*1	Det maksimale tilladte spændingsfald under afladning (%).
EE_SOF	0x198	1 Byte	Int – x*1	Specifikation af hvilke system funktioner der er aktiveret.
EE_VBATTNORM	0x199	2 Bytes	Uint – x*1	Nedre grænseværdi for normal batterispænding (0 - 1023).
EE_VBATTLOW	0x19B	2 Bytes	Uint – x*1	Nedre grænseværdi for lav batterispænding (0 - 1023).
EE_BIALARM1NAME	0x1BC	8 Bytes	String – x*1	Navnet på Binær alarm 1.
EE_BIALARM2NAME	0x1C4	8 Bytes	String – x*1	Navnet på Binær alarm 2.
EE_ANALARM2DL	0x1D3	3 Bytes	Uint – x*60000	Alarm forsinkelse for alarm 2 (ms).
EE_ANALARM2OU	0x1D6	1Byte	Uint – x*1	Alarm udløsnings niveau (0x01 = over eller 0x00 = under grænseværdi) for alarm 2.
EE_ANALARM2TH	0x1D7	2 Bytes	Float – x*1	Grænseværdi for alarm 2 (0 - 1023).
EE_ANALARM1DL	0x1D9	3 Bytes	Uint – x*60000	Alarm forsinkelse for alarm 1 (ms).
EE_ANALARM1OU	0x1DC	1Byte	Uint – x*1	Alarm udløsnings niveau (0x01 = over eller 0x00 = under grænseværdi) for alarm 1.
EE_ANALARM1TH	0x1DD	2 Bytes	Float – x*1	Grænseværdi for alarm 1 (0 - 1023).
EE_BO2ACTLVL	0x1DF	1 Byte	Uint – x*1	Alarm aktiverings niveau for alarm 2.
EE_BO1ACTLVL	0x1E0	1 Byte	Uint – x*1	Alarm aktiverings niveau for alarm 1.
EE_BIALARM2DL	0x1E1	3 Bytes	Uint – x*60000	Alarm forsinkelse for alarm 2 (ms).
EE_BIALARM2	0x1E4	1 Byte	Uint – x*1	Alarm udløsnings niveau (0x01 = over eller 0x00 = under grænseværdi) for alarm 2.

EE_BIALARM1DL	0x1EA	3 Bytes	Uint – x*60000	Alarm forsinkelse for alarm 1 (ms).
EE_BIALARM1	0x1ED	1 Byte	Uint – x*1	Alarm udløsnings niveau (0x01 = over eller 0x00 = under grænseværdi) for alarm 1.
EE_AVRGTIME	0x1F5	2 Bytes	Int – x*1000	Tidsinterval for måling af gennemsnits værdi, i forhold til ingen spændings Alarm niveau (ms).
EE_NOVOL	0x1F7	3 Bytes	Float – x*100	Grænseværdi svarende til ingen spænding.
EE_LOWVOL	0x1FA	3 Bytes	Float – x*100	Nedre alarm niveau for spænding.
EE_HIGHVOL	0x1FD	3 Bytes	Float – x*100	Øvre alarm niveau for spænding.
EE_BAY	0x240	5 Bytes	String – x*1	Bay id.
EE_MASTER_OF_UNIT	0x245	1 Byte	Int – x*1	Master position. Master: Dets egen position. Opti: Positionen af den Master de er tilkoblet.
EE_POSITION_OF_UNIT	0x246	1 Byte	Int – x*1	Opti position. Master: Benyttes ikke. Opti: Dets position under Master enheden.
EE_NID_OF_MASTER	0x247	1 Byte	Int – x*1	Master enhedens ID.
EE_UNIT_TYPE	0x248	1 Byte	Int – x*1	Enhedstype: 0 = Master, 1 = Opti.
EE_ID_NUMBER	0x249	1 Byte	Int – x*1	Enhedens CAN id (0 - 127).
EE_STATION_ID	0x24A	5 Bytes	String – x*1	Stations id.
EE_EQTY	0x24F	6 Bytes	String – x*1	Udstyrs type.
EE_RELAY_SEC	0x25D	1 Byte	Float – x*1	Antal sekunder et relæ er aktivt.
EE_RELAY_DEAD	0x25E	1 Byte	Int – x*1	Tidsinterval for hvor lang tid relæet skal være uspecificeret før der afsendes en SMS (sek.).
EE_CALI_CONSTANT	0x273	12 Bytes	Float – x*4096	Kalibrerings konstanter for spænding og strøm.
EE_FVER	0x280	4 Bytes	String – x*1	Firmware version.
EE_PVER	0x284	4 Bytes	String – x*1	Protokol version.

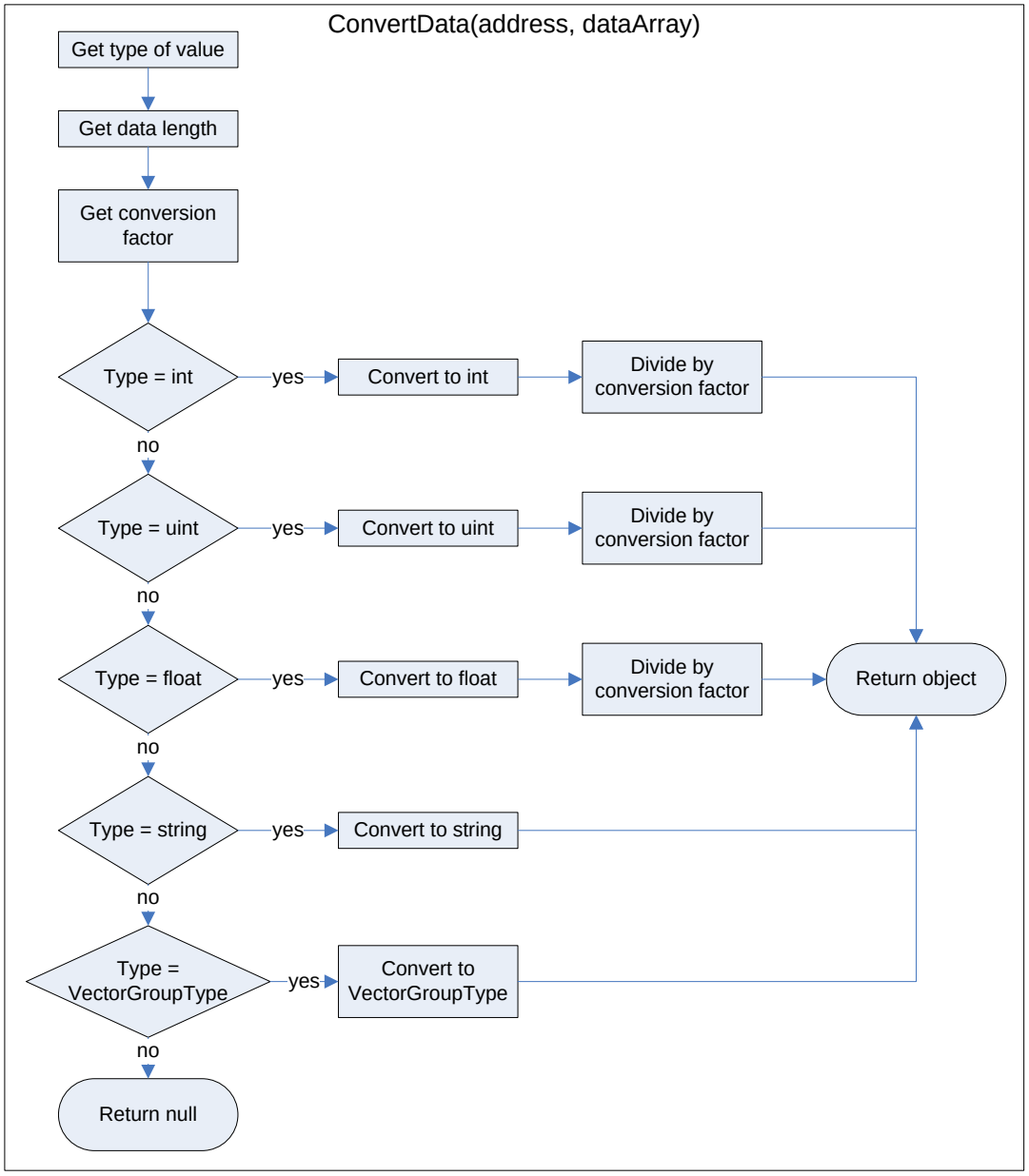
Tabel 18 - EEPROM Meta Map

9.5.1. Meta Map Konvertering

Nedenfor er der afbilledet 2 flowchart der illustrerer, hvordan EEPROM data konverteres til og fra DISCOS enhederne.



Figur 72 - Flowchart: Konvertering af data fra software objekt til firmware data



Figur 73 - Flowchart: Konvertering af data fra firmware data til software objekt.

9.6. CD Struktur

Dette afsnit beskriver de forskellige elementer der er placeret på den vedlagte CD:

- Visual Studio – Denne mappe indeholder den Microsoft Visual Studio solution, der samler de forskellige projekter, som er udviklet gennem projektforløbet.
    - APP – Denne mappe indeholder kildekoden til de klasser, der indgår i applikationslaget.
    - Common – Denne mappe indeholder kildekoden til de fælles elementer, der benyttes af de forskellige projekter.
    - Communication – Denne mappe indeholder kildekoden til de klasser, der indgår i kommunikationslaget.
    - Data – Denne mappe indeholder kildekoden til de klasser, der indgår i datalaget.
    - Device – Denne mappe indeholder kildekoden til de klasser, der indgår i enhedslaget.
    - GUI – Denne mappe indeholder kildekoden til de klasser, der benyttes i forbindelse med den grafiske brugerflade.
    - Protocol – Denne mappe indeholder kildekoden til de klasser, der indgår i protokollaget.
    - Install – Denne mappe indeholder installationsprojektet til Windows Mobile 5.0.
- Install – Denne mappe indeholder installationsfilerne til Discman PDA applikationen.
- Documents
    - DiscmanPDA_source.pdf – Dette PDF dokument er et udtræk kildekode dokumentationen.
    - DiscmanPDA.pdf – Dette PDF dokument er en kopi af denne rapport.