Utvikling av system for innlesing, bearbeiding og informasjonsutveksling mellom håndholdte enheter og Logit D2D

av

Jon Peder Saxe
Morten Trydal

Hovedoppgave til mastergrad i informasjons- og kommunikasjonsteknologi

Høgskolen i Agder
Fakultet for teknologi
Grimstad, mai 2006

Sammendrag

LogIT Systems AS leverer løsninger for planlegging av transportoperasjoner av gods fra utgangspunkt til destinasjon. Deres produkt Logit D2D skal booke nødvendige ressurser og gi mulighet for å overvåke transporten på en oversiktelig og enkel måte. I forbindelse med dette systemet ønskes en mulighet for å registrere og rapportere gods til sentral server ved hjelp av en trådløs håndholdt enhet. Rapporten omhandler forarbeidet og utvikling av en prototyp for å dekke dette behovet.

Kommunikasjon med oppdragsgiver, litteraturstudie og avgrensninger ble tidlig utført for å fastsette oppgavens mål og omfang. Vi har gått i dybden og kikket på dagens AIDC (Automated Identification and Data Capture) teknologier, og også forsøkt å kikke på hva fremtiden har å by på. Her har teknologiene blitt vurdert ut i fra deres fremtid i forbindelse med logistikk og bruk mot Logit D2D. Ulike alternativer for struktur på løsingen har blitt vurdert. Arbeidet har blitt nøye planlagt, noe som har ført til en enkel og oversiktelig struktur.

Valg av hardware ble en omfattende og tidkrevende oppgave da dette gjaldt funksjonalitet, vel så mye som kompatibilitet og brukervennlighet. Mange produkter finnes på markedet i dag, og det var viktig å finne en kombinasjon som knyttet sammen de beste. Både med tanke på OS, software og hardware.

En betydelig del av prosjektet har bestått av planlegging og selve programutviklingen. Mye arbeid har bestått av å utvikle en fungerende prototyp. Denne basert på utstyr som finnes på markedet i dag, men med spesialutviklet programvare tilpasset den håndholdte enheten og Logit D2D softwareløsningen. Videre har det blitt foretatt en enkel usabilitytest, og vi har sett på løsningens fremtidige muligheter.

Løsningen består av en robust mobil enhet med svært omfattende innebygd funksjonalitet. Dette gir muligheter for dagens ID teknologier, men åpner også døren for fremtidige. Enheten inneholder også en rekke funksjoner som kan være praktiske i forbindelse med løsningens bruksområder.

Resultatet har blitt en fungerende prototyp LogIT kan benytte ved presentasjoner og forhandlinger. Ikke minst er dette et meget godt utgangspunkt for videreutvikling mot et ferdig og salgbart produkt.

Forord

Prosjektet er utført som en avsluttende mastergradsoppgave for femårig mastergradsstudium ved HiA, Høgskolen i Agder avdeling Grimstad. Oppgaven er utført over vårsemesteret 2006 av Jon Peder Saxe, mastergradstudent med fordypning i Mobil datakommunikasjon og Morten Trydal, mastergradstudent med fordypning i Systemutvikling.

Oppgaveteksten ble publisert av LogIT Systems AS i forbindelse med Bachelor og Masteroppgaver ved HiA i Grimstad. Vi retter en takk til LogIT Systems for å gi studenter muligheten til å arbeide med reelle oppgaver og systemer som er i bruk i dag. Vi ønsker også å takke for deres veiledning og samarbeidsvilje for at resultatet skulle bli til det beste for dem som oppdragsgiver, og oss som studenter.

Videre ønsker vi å takke vår veileder Frank Reichert som har vært en ressursperson for oss gjennom hele oppgaven. Hans tips, anbefalinger og veiledning har hjulpet oss å se nye muligheter. Som veileder med god erfaring på området har han vært med på å peke ut den beste veien mot et godt resultat. Vi satt stor pris på hans hjelp med utvikling, planlegging og kommunikasjon med oppdragsgiver.

Til sist fortjener også Strekmenn AS(leverandør av utstyr) og Mobile Compia(produsent av utstyr) en takk for god support, og ikke minst OpenNETCF.org for å presentere mye nyttig kode og informasjon.

Grimstad, 29. mai 2006

_____	_____
Jon Peder Saxe	Morten Trydal

Innholdsfortegnelse

Figur-, tabellister og kodeeksempler

Figurer:

Tabeller:

Kodeeksempler:

Forkortelser

- AIDC – Automated Identification and Data Capture
- Auto-ID – Automatic Identification
- C++ – Navn på programmeringsspråk
- C# – Navn på programmeringsspråk
- CF – Compact Framework
- DPI – Dots Per Inch
- Ghz – Gigahertz
- GPRS – General Packet Radio Service
- GSRN – Global Service Relation Number
- GUI – Graphical User Interface
- HiA – Høgskolen I Agder
- ICC – Integrated Circuit Card (smartkort)
- ISBN – International Standard Book Number
- ISSN – International Standard Serial Number
- J2ME – Java 2 Platform, Micro Edition
- Kbps – Kilobit per second
- Mb/sek – Megabit per second
- MD5 – Message-Digest algorithm 5
- OCR – Optical Character recognition
- OMC – Optical Memory Card
- PB – Provider Booking
- PDA – Personal Digital Assistant
- RS232 – Recommended standard-232
- SDK – Software Development Kit
- SGLN – Serialized Global Location Number
- SGTIN – Serialized Global Trade Item Number
- SSCC – Serial Shipping Container Code
- SIM – Subscriber Identity Module
- TZD – Time Zone Designator
- UPC – Universal Product Code
- VS – Visual Studio
- WEP – Wired Equivalent Privacy
- WLAN - Wireless Local Area Network
- WPA – Wi-Fi Protected Access
- XML – Extensible Markup Language

1. Innledning

Prosjektet vil i hovedsak gå ut på å utvikle et system for informasjonsutveksling mellom håndholdte enheter og Logit D2D. Vi skal her kikke nærmere på LogIT Systems AS [1], elementene rundt utviklingen av systemet og på relevante teknologier som er tilgjengelige i dag. Vi vil se på hvordan oppgavene skal utføres og på hvilke utfordringer vi muligens vil støte på når vi kommer i gang med arbeidet.

1.1 Bakgrunn

LogIT Systems utvikler og leverer løsninger for transport operasjoner av last fra utgangspunkt til destinasjon. Det skal booke de nødvendige ressursene og gi mulighet for å overvåke transporten så enkelt som mulig. Nøkkelfunksjoner i systemet er å utnytte transportressurser effektivt og gi riktig informasjon til de riktige personene på riktig tidspunkt. Et slikt system er avhengig av å være oppdatert til enhver tid. For at dette skal skje må systemet mates med data fra alle områder det dekker. Jo større systemet blir, jo mer omfattende blir det å melde tilbake og verifisere at informasjon og data er korrekt.

Figur 1 Offisielle logoer for LogIT Systems og Logit D2D.

Automatisk identifikasjon brukes for å identifisere objekter, samle data om dem, og overføre dataene direkte til datasystemer. Det finnes mange forskjellige teknologier innen AIDC(Automated Identification and Data Capture). F.eks. strekkoder, RFID , magnet striper, smart kort og stemmegjenkjenning. Disse teknologiene kan brukes i kombinasjon med håndholdte mobile enheter, som f.eks. Pocket PC, PDA eller avanserte mobiltelefoner. For å nyttiggjøre seg av informasjonen man får fra AIDC enheten utvikler man programvare ved hjelp av et programmeringsspråk(f.eks. Java, C# eller C++) som kjøres på den håndholdte mobile enheten. Denne programvaren kan så tolke dataene, for å oversende de til LogIT Systems løsning, kalt Logit D2D(door-to-door).

Det er ønskelig å kunne benytte en håndholdt enhet som er koblet opp mot LogIT Systems sitt eksisterende system. Via et trådløst nettverk skal man ha tilgang til systemet hvor enn en måtte befinne seg. Skal det være raskt og enkelt i bruk bør det også være minimalt med manuell inntasting. Derfor er det ønskelig at enheten selv skal kunne lese inn og tolke data fra for eksempel strekkoder eller RFID brikker på forsendelser. På denne måten behøver brukeren kun å verifisere dataene på en liten skjerm, utføre eventuelle endringer, og så sende dataene inn i systemet. En slik enhet vil også ha mulighet til å utføre andre funksjoner. For eksempel å holde styr på flerkollisendinger. Personen som registrerer vil da straks kunne se om noe mangler. Dette kan forhindre oppdelte forsendelser, noe som ville ført til unødvendige utleveringer. Det er ønskelig å få utviklet en fungerende prototyp. Dette medfører at alle hovedelementer av løsningen må inkluderes i arbeidet. Eventuelle avgrensninger må ikke hindre at systemet kan testes fra ende til ende. Dette innebærer at følgende ting må gjøres eller tas hensyn til:

- Kommunisere med LogIT Systems eksisterende database, som kjører på en Jboss applikasjonsserver.
- Ved innlesing til håndholdt enhet må data håndteres og klargjøres for lagring i det sentrale systemet.
- Utenforstående skal ikke ha tilgang til intern informasjon eller nettverk.
- Utviklingsspråket som velges må være stabilt, raskt og kunne tilby de nødvendige egenskapene som trengs for å utvikle en fullgod prototyp.
- For at løsningen skal aksepteres av brukeren, må brukergrensesnittet være intuitivt, presentere riktige data for bruker og gi riktige input muligheter.
- Håndtere kommunikasjon og inndata fra leseenheten og overføre disse korrekt til serverens databasesystem.
- Vurdere teknologier og protokoller som egner seg best for kommunikasjon mellom den håndholdte enheten og applikasjonsserveren?

Det er håp om å utvikle en løsning som kan brukes til testing og demonstrering for potensielle kunder. Avgrensninger vil derfor i første omgang dreie seg om å kutte ned på eventuelle ekstrafunksjoner. Dette vil vi kikke nærmere på i kapittelet avgrensninger.

Figur 2 Illustrerer en mulig struktur i Logit D2D systemet

Figuren illustrerer et alternativ av hvordan vi ser for oss en sammensetning av de produktene som er tilgjengelige i dag. En løsning med strekkodeleser tilkoblet en mobil enhet. Denne er koblet opp mot Internet via en WLAN ruter. Dette gir tilgang til LogIT Systems webservice som igjen gir tilgang til Logit D2D database på server. På denne måten vil all informasjon lagres sentralt, men være tilgjengelig lokalt. For å utvikle et slikt fungerende system vil vår forskning bestå av:

- Utforske elementenes rekkevidde, stabilitet og funksjonalitet
- Utvikle effektive applikasjoner som benytter begrenset datakraft
- Utvikle en intuitiv og effektiv brukerapplikasjon
- Utforske fremtidige løsninger og legge til rette for disse

1.2 Oppgaven

LogIT Systems AS utvikler programvare for planlegging og styring av avanserte logistikkoperasjoner. Logit D2D er en løsning for planlegging av multimodale transportkjeder. Programvaren er programmert i Java, basert på J2EE-plattformen og kjører på Jboss applikasjonsserver.

For å rapportere inn hendelser og statuser til Logit D2D, trenger tilbydere av transport- og terminaloperasjoner utstyr som på en enkel måte kan sende informasjon til transportstyringssystemet.

Prosjektet vil forsøke å identifisere og velge mulige håndholdte systemer, skissere utvidelser til arkitekturen for å utvide det nåværende systemet for mobil datainnsamling og oversendelse til back-end systemet. En "proof-of-concept" prototyp skal kunne demonstrere de tekniske mulighetene en slik løsning kan gi.

Det finnes i dag flere enheter på markedet, og en del av oppgaven vil være å finne en egnet enhet, eller en sammensetning av enheter.

Det vil også være nødvendig å utvikle programvare for enheten(e) for lesing, tolking og sending, og også et brukergrensesnitt for manuell inntasting av data direkte inn på den håndholdte enheten. Utover dette kan det bli aktuelt å kikke på mottak av data fra server til håndholdte enhet om tiden skulle tillate det.

Vi vil kartlegge andre mulige identifikasjonssystemer (både dagens og fremtidige) for last, lasting og lossing av varer og lastbærere i tillegg til strekkoder og RFID. Det vil også kartlegges hva slags data som lagres på slike identifikasjonsmerker i de forskjellige bransjene. En oppgave vil være å finne ut av hvordan Logit D2D kan nyttegjøre seg den informasjonen som finnes i identifikasjonsmerkene.

Oppgaven vil bestå av:

- Utarbeide kravspesifikasjon
- Velge teknologi
    - Finne egnet enhet eller sammensetning av enheter
    - Innlesing av identifikasjonsmerker
    - Oversending av data til sentral server
- Designe, utvikle og implementere programvare
- Integrere og teste mot Logit D2D

Prosjektet vil utføres i nært samarbeide med LogIT Systems. Alt teknisk utstyr vil skaffes til veie av selskapet.

1.3 Teoretisk rammeverk

Vi vil her gi en kort beskrivelse rundt de elementene som har vært innblandet under utviklingen av denne oppgaven. Denne teksten inneholder ord som kan være viktige for å forstå helheten, og selv om mange fagfolk kjenner dem alle, så tar vi en kort gjennomgang her.

Emulator: En emulator gjør det mulig å la en datamaskin kunne lese data som er laget for an annen type datamaskin. I utviklingen av prosjektoppgaven kan det brukes en PDA emulator som gjør det mulig å simulere applikasjonen vi utvikler direkte på PC'en vi jobber på, istedenfor å måtte overføre den til PDA hele tiden.

GPRS – General Packet Radio Service. En teknikk for å overføre data på mobiltelefonnettet GSM. GPRS kan overføre data i relativt beskjedne mengder, og betegnes gjerne som 2.5G, en teknologi midt i mellom andre(2G) og tredje generasjons(3G) mobiltelefoni.

localhost: localhost henviser til maskinen man jobber på. Man kan kjøre servertjenester og lignende lokalt på localhost. Ved å henvise til localhost når man jobber mot disse vil arbeidet utføres som mot en ekstern server.

Opennetcf - Open Net Compact Framework: Dette er en spesiallaget CF kun for PDA. Denne inneholder flere, og noen forbedrede funksjoner enn .NET CF. Opennetcf tilbyr også åpen kildekode.

PDA –Dette er en håndholdt liten datamaskin som kan kombinere data, telefon, faks, Internett, og nettverksfunksjoner. PDA går ofte også under navnene palmtops, håndholdte datamaskiner og lomme PC. Ulikt en bærbar PC er det på PDA vanlig å bruke en penn hvor man trykker direkte på skjermen.

Figur 3 PDA av merket TDA Recon

Språk: Når vi snakker om språk i forbindelse med programmering er det gjerne programmeringsspråk det er snakk om. Programmeringsspråk er et slags kodespråk, i de fleste tilfeller basert på engelsk, ment for å kunne styre en maskins oppførsel.

> **C#** - C# er et Microsoft utviklet objekt orientert programmeringsspråk. En slags hybrid av C og C++ som er utviklet for å kunne konkurrere med Java språket fra Sun. Syntaksen likner mye på Java sin.
>
> **C++** - Et kraftig objektorientert programmeringsspråk. Et av de kraftigste utviklingsspråkene, spesielt for store applikasjoner. Selv om språket er effektivt er det komplekst og kan være vanskeligere enn andre å mestre.
>
> **J2ME** – Java 2 Micro Edition. Kompakt versjon av Java. Tilpasset for å kunne utvikle små applikasjoner for enheter som PDAer og mobiler.

VS – Visual Studio: Kraftig programmeringsverktøy. En utviklingsplattform som støtter en rekke språk, og som kan kompilere koden til bruk i flere ulike formål.

WLAN –Forkortelsen betyr på norsk trådløst lokalt nettverk. Blir også noen ganger betegnet som LAWN. Dette er en type lokalnettverk som bruker høyfrekvente radiobølger istedenfor ledninger for kommunikasjon mellom de ulike maskinene på nettverket.

Figur 4 Eksempel på tilkoblingsmuligheter i et trådløst nett [#223]

.NET Framework: Dette rammeverket er utviklet av Microsoft som en del av Windows operativsystemet. Dette består av en rekke ferdigkodede løsninger. Dette rammeverket kan benyttes i utvikling av applikasjoner mot Windows plattformen. .NET CF(Compact Framework) er en kompakt versjon av .NET Framework og brukes til utvikling av applikasjoner for bruk på enheter med begrenset datakraft som for eksempel mobiler og PDAer.

1.4 Motivering og nytteverdi

Prosjektgruppen består av to master studenter innen hvert sitt område. Henholdsvis ”Mobil kommunikasjon” og ”Systemutvikling”. Oppgaven er derfor ikke bare ideell for at begge parter skal kunne utforske sitt kunnskapsfelt, men også hverandres. På denne måten får vi muligheten til å utforske og utvikle oss innen aktuelle teknologier som også vil være relevante for oss i fremtidige sammenhenger. Bruken av mobile enheter og teknologien innen dette området har skutt fart de siste årene. Ved å benytte teknologier blant de nyeste på markedet både innen hardware og software gjør det oppgaven meget fremtidsrettet. Erfaring innen utvikling av slike løsninger er derfor svært relevant for oss som fremtidige jobbsøkere og arbeidstakere.

Problemstillingen er både spennende og interessant på flere måter. LogIT Systems ønsker å satse tid og ressurser på å finne egnete enheter samt utvikle programvare for disse, noe som tyder på at dette er noe de satser på i tiden som kommer. LogIT Systems ønsker i dette tilfellet å utvikle en prototyp. En gang i fremtiden ønsker de at dette skal utvikle seg til et produkt for å komplettere deres nåværende produkter, og gi dem muligheten til å tilby en totalløsning innen transport og logistikksystemer.

Det at Posten Norge har signert en kontrakt med ErgoGroup(norges 3. største IKT selskap), der løsningen baseres på produktet Logit D2D, er selvsagt også veldig spennende. Det finnes omtrent ikke noe mer motiverende, enn å vite at dette faktisk er noe oppdragsgiver vil kunne dra nytte av.

Løsningen skal baseres på aktuelle teknologier som er spennende å arbeide med. På denne måten vil vi lære hvordan utviklingen kan dra nytte av dem på nye måter. Gruppemedlemmene har erfaringer med enkelte av teknologiene fra sin Bachelor oppgave, dette gir oss tro på at vi vil kunne utvikle en fungerende prototyp.

1.5 Litteraturstudier

Selv om oppgaven har vært en utviklingsoppgave har det krevd mye forarbeid. Vi har beveget oss innenfor teknologier og systemer som har vært nye for oss, og da er det viktig med et forarbeid som gir oss god innsikt i hva vi begir oss ut på, og som hjelper oss å foreta de riktige valgene.

Som oppvarming til ”IKT-590 Masteroppgaven” hadde vi i forkant faget ”IKT-505 Metodeseminar”. Dette faget var lagt opp for å forberede oss til masteroppgaven, og her fikk vi utviklet vår forståelse for vitenskapsteori og forskning [2, 3]. Vi har også vært igjennom en rekke temaer som har vært relevante for denne oppgaven, blant annet har vi studert temaene SE(software engineering) [4] og design av IS(Informasjons Systemer) [5].

Ved utvikling av systemer basert på det nyeste av teknologi på markedet i dag, kreves det oppdatert kunnskap om temaene. Vi har også vært ute etter mye produktspesifikk data. Bøker og artikler fra biblioteker og arkiver er ikke alltid oppdatert til å kunne tilby denne type informasjon rundt de mest moderne teknologier. Internet har derfor også vært en effektiv kilde til å søke frem oppdaterte artikler om de aktuelle temaer. Internet er i dag verdens største kilde til informasjon, men også til feilinformasjon. Vi har derfor vært kritiske til de skriv og artikler vi har funnet, og kun plukket ut de artikler vi anså som pålitelige.

Under programmering og utvikling har vi benyttet en rekke bøker for å søke fremgangsmåter, muligheter og løsninger. Noen bøker sitter vi igjen med som lærebøker fra relevante fag. Vi har også benyttet noen bibliotekbøker, og en del private bøker [6-11].

Som utgangspunkt for søk på Internett har vi i mange tilfeller benyttet oss av søkemotoren Google.com, og Wikipedia.org, ved å følge deres linker til utallige utviklere, applikasjoner og enheter. Siden ikke alle linker er like pålitelige har vi forsøkt å kvalitetssikre alt vi har ønsket å benytte.

Under forarbeidet trengte vi å sette oss inn i de ulike produkter som eksisterte på markedet, og hva disse hadde å tilby. Det mest relevante her for innhenting av informasjon rundt disse produktene er produsentenes egne offisielle websider. Vi har blant annet benyttet websiden til Mobile Compia [12], produsent av enheten vi til slutt valgte å benytte ”M3”. Her ble det presentert fullstendig spesifikasjoner. Utover dette innehold disse sidene også SDK [13] for styring av en rekke av funksjonene enheten har å by på. Dette SDKet var svært nyttige for oss i utviklingsfasen. På samme måte har også OpenNETCF.net – ”The Premier .NET Compact Framework Shared Source Site” [14] vært til stor hjelp. Disse sidene tilbyr en rekke løsninger for utviklere på CF, og alt dette i åpen kildekode.

En mer direkte, men vel så viktig kilde til kunnskap og hjelp under utvikling av applikasjoner finner man i foraer. Det finnes en hel rekke fora som omhandler ulike programmeringsmiljøer. Vi har i perioden blant annet benyttet oss av utviklingsforaene ”The Code Project” [15] og ”C# Corner” [16].

Utover dette er alle artikler og tekster vi har benyttet under arbeidet med oppgaven referert til ved de aktuelle tema. Hver enkelt referanse presenteres i kapitlet ”Referanser”.

1.6 Omfang

Vi skal gå i dybden og kikke på de teknologier som finnes i dag. Dette omfatter både aktuelle software og hardware teknologier som er på markedet, og dagens samt mulige fremtidige AIDC teknologier. Vi vil ta for oss de ulike enhetene som kan benyttes for å utføre de ulike oppgavene og studere deres spesifikasjoner. I rapporten vil vi presentere en gruppe av de mest aktuelle. Software og operativsystem på enhetene vil i noen tilfeller avgjøres under valget av

enhet, men ulike OS vil likevel studeres. Programmeringsspråk vil vi gå nærmere inn på for å se om de kan tilby hva vi har behov for.

Vi vil gå i dybden på dagens AIDC teknologier, og samtidig forsøke å se hva som kan komme i fremtiden. Vi vil se nærmere på hvordan de ulike teknologiene fungerer, og hvordan, eller om de kan benyttes i forbindelse med transport, logistikk, og Logit D2D.

Strukturen skal vurderes for å utvikle et mest mulig stabilt og funksjonelt system. Dette gjelder hele veien fra innlesing til data er lagret i sentral database. Minst en ende til ende forbindelse skal utvikles og kunne benyttes for innlesing og overføring av data.

Vi vil utvikle en funksjonell prototyp for innlesing, bearbeiding og sending av data til sentral server. Viktige funksjoner skal være ferdigutviklet, og løsningen skal fungere på en slik måte at man enkelt ser mulighetene for bruk i praksis. Den skal være ferdigutviklet og brukbar i den grad at den skal kunne brukes som et demonstrasjonsverktøy for blant annet interessenter og investorer.

Systemet skal være brukervennlig og intuitivt. Det skal utvikles et fornuftig brukergrensesnitt beregnet for bruk av lite datakyndige mennesker. Knapper og viktige funksjoner skal være tydelige og lett leselige. Standard oppgaver i applikasjonen skal være mulig å utføre med få klikk direkte med fingeren på skjermen. Brukervennligheten vil også evalueres i en liten brukervennlighetstest(usabilitytest) hvor eventuelle feil oppdages og forsøkes forbedret.

1.7 Ressurser

LogIT Systems har sagt seg villige til å stille med nødvendig utstyr for å utføre prosjektoppgaven. Hovedelementene som vil være nødvendige er listet opp her:

- Mobil enhet
- Strekkode og/eller RFID enhet(er)
- Relevante testobjekter
- Software og utviklingsverktøy
- Webservice mot Logit D2D
- WLAN tilknytning til nett
- GSM/GPRS tilknytning til nett
- Lokaler
- Ytterligere nødvendig utstyr

LogIT Systems vil stå for innkjøp av en håndholdt enhet, strekkodeleser og eventuell(e) RFID enhet(er). Relevante testobjekter vil bli fremskaffet samt tilgang til deres webservice og testdata. Software og utviklingsverktøy vil vi delvis kunne dekke selv og ved tilgjengelige programvare fra HiA. Utover dette blir det også oppdragsgivers oppgave å stå for nødvendig software. Tilgjengelig WLAN vil ikke være noe problem da dette finnes både hos LogIT, HiA og oss privat. Om oppgaven skal omfatte bruk av GPRS vil også utstyr og abonnement for dette være nødvendig. LogIT Systems har stilt et kontor til disposisjon for prosjektet, og eventuelt ytterligere nødvendig utstyr vil bli fremskaffet. PCer og grunnleggende programvare for å utføre generelt arbeid stiller vi med selv.

1.8 Avgrensninger

I og med at målet er å lage en mest mulig funksjonell prototyp, en løsning så utviklet at den kan brukes til testing og demonstrering for potensielle kunder, bør avgresninger være å kutte ned på eventuelle ekstrafunksjoner. Vi tillater derfor at løsningen på noen områder vil bære preg av at det er en prototyp, og ikke et ferdig produkt.

Feil oppstår i alle applikasjoner, og god feilhåndtering er et av leddene i en god løsning. Under applikasjonsutvikling vil man alltid prøve å unngå feil og å håndtere dem på en best mulig måte. Vi vil i vårt prosjekt ikke prioritere feilhåndtering i utvidet grad slik en ville gjort ved utvikling av et ferdig produkt. Dette er et svært tidkrevende arbeid og tidsforbruket vil neppe veie opp for forbedringen når det gjelder utvikling av en prototyp under slike tidsbegrensninger en masteroppgave gir.

Sikkerhet er alltid viktig, spesielt i trådløse applikasjoner. Applikasjonen kommer ikke til å håndtere meget sensitiv data. Likevel er det ikke ønskelig at andre skal ha tilgang til dem. En viktigere del ved sikkerheten er ikke om uvedkommende kan lese data, men om de får tilgang til å utføre endringer. Ved slik tilgang kan en hacker raskt gjøre store skader på systemet, noe som i de fleste tilfeller også vil ha store økonomiske følger. Likevel vil vi ikke gå i dybden for å utføre sikkerhetstiltak. Systemet er ment som en prototyp for å illustrere en fremtidig løsning. Prototypen vil i første omgang ikke håndtere reell data, og vil i svært liten grad være utsatt for dataangrep. Sikkerhet vil derfor ikke være prioritert i løsningen.

Vi vil i prosjektet kikke litt på brukervennlighet og temaer rundt dette slik at applikasjonen utformes på en måte brukerne forstår. En effektiv metode innen brukervennlighetsdesign er usabilitytesting. En god usabilitytest innebærer ganske omfattende arbeid. Dette vil vi sannsynligvis ikke ta oss tid til i prosjektperioden. Vi vil likevel forsøke å få tid til å utføre deler av en usabilitytest til fordel for fremtidig utvikling.

Det er ønsker om å flette inn flere teknologier i systemet. Blant annet bruk av både WLAN og GPRS, og leseenheter for RFID og strekkode. Vi vil i førsteomgang begrense oss til å kun utvikle én løsning fra innlesning og til Logit D2D server. Dette er det som er nødvendig for å få klar en fungerende prototyp. Hvilke alternativer som skal prioriteres først vil bestemmes i samarbeid med LogIT og spesifikasjoner rundt utstyret. Videre vil det være ønskelig å se på parallelle løsninger som for eksempel både WLAN og GRPS om tiden gir mulighet for dette.

1.9 Mål

Hensikten med utviklingen av systemet er å kunne effektivisere transportsystemer, forenkle arbeidsdagen til transportarbeiderne, forbedre kontroll og sporing av gods og å gjøre sentral data mer korrekte. Målet er å gi LogIT Systems muligheten til å utforske hvordan et slikt system på best mulig måte bør ta i bruk ulike teknologier, samt å utvikle en fungerende prototyp. Arbeidet vil benyttes for videre vurdering av satsning på området, og vil forhåpentligvis skape grunnlag for et nytt produkt i LogITs tilbudsspekter.

1.10 Forutsetninger

Det behøves et nært samarbeid med oppdragsgiver, og det kreves at en del teknisk utstyr er tilgjengelig for å kunne utvikle løsningen. Oppdragsgiver har lovet å skaffe til veie det nødvendige utstyret. Hvilket utstyr som er nødvendig vil komme frem ved nærmere

undersøkelse av hva som er tilgjengelig fra utgangspunktet, og i diskusjon og samarbeid med veileder og LogIT Systems.

Det vil være nødvendig med innsikt i noen av deres løsninger, og tilgang til noen av deres eksisterende produkter. Vi må også gjøres kjent med kommunikasjonsprotokoller i dagens system. Mulighet for et eget testområde i systemet hvor endringer og eventuelle feil ikke får konsekvenser vil også behøves.

Det vil være nødvendig med god kommunikasjon med prosjektveileder hos LogIT Systems. Vi har her blitt fortalt at LogIT Systems vil stille med en veileder og en kontaktperson, samt at vi har fått utnevnt en veileder innenfor relevant fagområde ved Høgskolen i Agder, HiA.

1.11 Rapportens oppbygning

Rapporten har blitt utarbeidet underveis i utviklingen av prosjektet. Helt i starten ble en grunnleggende mal fastsatt for å enklere bevare og presentere arbeidet i det øyeblikket det ble utført. På denne måten blir rapporten mest mulig korrekt og detaljert. Vi har forsøkt å dokumentere arbeidet detaljert slik at oppdragsgiver og fremtidige utviklere så enkelt som mulig kan sette seg inn i prosjektet. Dette har ført til en omfattende rapport, noe vi håper kan være en god ressurs for LogIT Systems for videre utvikling.

Vi starter med et lite sammendrag for å la leserne raskt få innsyn i rapportens innhold. Dette er så fulgt av innholdsfortegnelse, tabell- og figuroversikt for å gi leserne god oversikt.

Videre kommer et innledningskapittel hvor vi går nærmere inn på bakgrunnen for oppgaven og betingelser, forutsetninger og mål som er satt rundt prosjektet. I dette og de påfølgende kapitlene, en og to, ønsker vi å sette leser inn i hva oppgaven innebærer.

Kapittel tre, forarbeid og planlegging, dreier seg om de temaene det var nødvendig å gå i dybden på, og de avgjørelser som måtte tas før vi tok fatt på selve utviklingsarbeidet. Her ser vi nærmere på strukturen i systemet og de aktuelle teknologiene. Videre går vi ytterligere inn på teknologier for identifikasjonssystemer i kapittel fire.

Før vi går i dybden på utviklingen av systemet går vi raskt gjennom de teknologiene som har blitt valgt, og hvilke kriterier som førte til disse valgene. Så kommer kapittelet ”Utvikling” som omhandler utviklingen av systemet. Kapittelet er delt opp i tre naturlige deler; server, softwareløsning og design.

Mot slutten ser vi tilbake på arbeidet som er blitt utført. Her drøftes resultatet som har blitt oppnådd. Vi ser også på mulighetene til forbedringer og videreutvikling, og gir til slutt vår konklusjon for hvordan arbeidet har gått og resultatet er blitt. Her ser vi i hvilken grad vi har nådd våre mål, delmål og i hvilken grad oppgaveteksten er blitt besvart.

2 Beskrivelse av oppgaven

LogIT Systems har utviklet en løsning for planlegging av multimodale transportkjeder. Systemet har for øyeblikket ingen løsning for innlesing og innrapportering av hendelser og statuser trådløst. LogIT Systems ønsker derfor å se på mulighetene for å utvikle en løsning som på en enkel måte kan sende informasjon til transportstyringssystemet. Løsningen skal benytte seg av utstyr som allerede finnes på markedet. Det skal utvikles programvare som skal kunne utnytte en kombinasjon av utstyr og teknologier mot LogIT-systems sine eksisterende systemer.

Oppgaven vil i stor grad bestå av å finne riktig hardware løsning, vurdere/velge utviklingsplattform/-språk og selve utviklingen av programvare. Alt av hardware må være kompatibelt, ha god nok ytelse og ikke være for kostbart. Ved valg av utviklingsplattform må pris og ytelse tas hensyn til, samt hvilke muligheter den kan tilby. Applikasjonen må være brukervennlig, enkel og effektiv i bruk.

Kan systemet lages? Det finnes i dag teknologi til å utføre det meste om man utnytter riktig teknikk og fremgangsmåte. Det vil i startfasen være ekstremt viktig å gjøre grundig research rundt de ulike elementene som skal inngå i systemet. Feilvurderinger kan føre til at hele prosjektet stopper opp, eller at store endringer og omlegginger må til. Ved riktig valg av teknikk, utstyr og teknologi vil utvikling av ønsket løsning høyst sannsynlig la seg gjøre. Vi har her kikket nærmere på noen av de delproblemene planlegging og utviklingen vil innebære.

Programmeringsspråk: Valg av utviklingsspråk kan være kritisk for om oppgaven vil bli vellykket eller ikke. Kjernen av Logit D2D systemet er en javabasert JBoss server. Valg av Java som programmeringsspråk kan derfor høres ut som den selvfølgelige løsning. Dette behøver likevel ikke være tilfellet. Java kan ha kritiske mangler der andre språk kan tilby bedre løsninger. Å velge programmeringsspråk uten nærmere undersøkelse kan etter hvert vise seg å bli svært ødeleggende for prosjektets sluttresultat.
Avhengig av hvilket utviklingsspråk som velges, kan det tenkes at problemer oppstår med kommunikasjonen mellom den håndholdte enheten og Jboss applikasjonsserver. Vi har kommet frem til aktuelle programmeringsspråk i samtaler med oppdragsgiver. Java, C++ eller C# er ønsket da disse språkene antas å best kunne tilfredsstille kravene man har til en slik applikasjon. Det vil her være nødvendig å gå dypere inn i spesifikasjonene rundt disse tre språkene. En sammenligning av mulighetene språket tilbyr med tanke på mobile enheter og trådløs kommunikasjon vil være nødvendig.

Datalagring: Det skal utvikles en applikasjon for bruk på mobile enheter med kommunikasjon mot en sentral database. Vi vet før oppgaven veldig lite om databasestrukturen til LogIT Systems. Pga dette er det meget vanskelig å anslå om det vil oppstå vanskeligheter ved input av data. Når alle data lagres sentralt er det også viktig at formatering og lagring fungerer slik spesifikasjonen tilsier. Problemer med lagring av data kan oppstå om formatering av inndata ikke stemmer overens med formatering i den eksisterende databasen. Det er her viktig å gå i dybden i LogIT Systems spesifikasjoner for lagring, kommunikasjon og håndtering av data.

Tilleggsfunksjoner: Hva som er standard og hva som kan utføres er to ulike ting. Det finnes ytterligere problemer rundt valg av programmeringsspråk. Hva kan språket tilby, og hvilke kostnader gir det oss? Vil det være nødvendig med dyre tilleggsfunksjoner. Det kan også tenkes at det valgte utviklingsspråket ikke inneholder alle de nødvendige funksjonene som

trengs. Det kan tenkes at til noen språk vil det være nødvendig å kjøpe kostbare tilleggspakker, noe som kan være uheldig/uaktuelt for oppdragsgiver.

Mobil enhet: Applikasjonen som skal utvikles skal kjøres på en håndholdt enhet for registrering av gods. I fremtiden ser LogIT Systems for seg muligheten for å utvikle spesialenheter for dette formålet, men i et slikt tidlig stadie i prosessen vil det likevel være nødvendig å basere seg på eksisterende teknologi.
Det finnes i dag utallige mobile enheter og en rekke produsenter på markedet. Her er det viktig å utforske hva som finnes av enheter og hva de kan tilby. Kompatibilitet er viktig, samt hvilke utviklingsmuligheter man kan se for seg i fremtiden. Enheten bør være lagt til rette for å bruke tredjeparts applikasjoner. Den bør også ha støtte for de aktuelle teknologiene som er ønsket å benytte i prosjektet.

Trådløs kommunikasjon: Trådløs kommunikasjon mellom mobil enhet og nett med forbindelse til JBoss server er kritisk for at prosjektet skal bli vellykket. Det er viktig å finne passende trådløs teknologi. Oppdragsgiver har i oppgavebeskrivelsen nevnt teknologier som WLAN, GPRS og SMS. Det finnes en lang rekke eksisterende og kommende mobile teknologier. Her må behov, funksjonalitet og kompatibilitet vurderes og veies opp mot hverandre. Mengden data systemet skal transportere vil være beskjeden, så her vil stabilitet være en mer kritisk faktor.

Identifikasjonssystemer: Den håndholdte enheten skal automatisk identifisere gods ved hjelp av en allerede utviklet teknologi. De mest aktuelle AIDC(automatisk identifikasjon) teknologiene i dette tilfellet er strekkoder og RFID. Oppdragsgiver ønsker her også å se frem i tid. RFID er relativt nytt og fremadstormende, men andre nye teknologier er kanskje også på vei. Her er utfordringen å få et innblikk i dagens og fremtidige teknologier, og hvordan man kan legge til rette for at også disse kan benyttes i den utviklede applikasjonen.

2.1 Dagens situasjon

LogIT har spesialisert seg på å utvikle programvare for planlegging og optimalisering av transportkjeder. Deres systemer er spesielt tilpasset transportkjeder mellom vei-, båt- og togtransport. Det finnes i dag store aktører som benytter seg av systemer fra LogIT Systems AS. Blant annet benyttes deres løsninger hos Posten Logistikk [17] og Broströms AS [18].

Deres programvare er likevel kun en del av en totalløsning som i dag er i bruk. Det tilbys i dag bare grensesnitt mot dette systemet via vanlige datamaskiner. For å utvide horisonten for deres produkt kan det nå være aktuelt med en håndholdt enhet som kan levere og hente data med direktekommunikasjon mot hovedserver. På denne måten kommer LogIT Systems nærmere det å tilby en totalløsning, ved at dette også blir et produkt som kan benyttes i felten. En slik spesialtilpasset løsning for deres systemer finnes ikke i dag, men planlegging og utvikling er nå i gang. Dette i samarbeid med oss i forbindelse med avvikling av Masteroppgaven ved Høgskolen i Agder våren 2006.

3 Forarbeid og planlegging

Vi er to i gruppen som arbeider med prosjektoppgaven. Fra tidligere er vi kjent med hverandre både privat og i skolesammenheng. Det at vi har et godt forhold fra tidligere ser vi på som en fordel da det jobbes med et prosjekt basert på samarbeid og tillit. Vi har også ved tidligere anledninger jobbet på prosjektgrupper med liknende oppgaver, og har erfaring med hverandres fremgangsmåter og arbeidsmetoder.

Vi tok sikte på å ha like roller i gruppen, slik at begge fikk god oversikt fra prosjektperiodens start. Hele prosjektet var som tidligere nevnt delt inn i ulike hoveddeler. Vi har begge hatt innblikk i alle delene, men vi har fordelt rollene på de ulike elementene slik at vi begge har fått erfaringen med å ha hovedansvaret. På denne måten fikk vi innsikt i alle ledd i prosessen, samtidig som vi fikk fordypet oss i vår del og sørget for at den ble utført tifredsstillende og innenfor sine tidsrammer.

Arbeidet var i realiteten delt opp i en rekke mindre prosjekter. Den første delen av prosjektperioden besto av valg av utstyr og software. Vi forsøkte å få alt dette klart før oppgavedefinisjonen skulle fastsettes i starten av februar. Dette fordi valg av utstyr og software kunne sette begrensninger som ville få konsekvenser for hvilke mål man kunne forvente å nå innen prosjektperiodens slutt. Dette har vi blant annet kikket nærmere på i kapittelet ”Vurderinger før valg av enheter”. De ulike elementene ville gitt fordeler og ulemper, samt skape krav i forhold til hverandre. Vi ville i denne perioden forsøke å finne den ideelle sammensetningen ut ifra hva som finnes av utstyr og teknologier i dag.

Den mobile enheten måtte ha støtte for den aktuelle programmeringsplattform, og plattformen måtte tilby riktige funksjoner. Den måtte også ha riktige tilkoblingsmuligheter og egenskaper for sitt bruk. På samme måte måtte valgt AIDC utstyr ha riktige tilkoblingsmuligheter i forhold til den mobile enheten. Etter en grundig gjennomgang av de ulike mulighetene valgte vi utstyr og kunne bevege oss videre til neste steg i utviklingen.

Her så vi det som naturlig å starte arbeidet med å lese inn data til mobil enhet. Dette valgte vi likevel å utsette til senere grunnet frykt for sen utstyrsleveranse. Vi velgte derfor å konsentrere oss om den andre enden av kommunikasjonen, mot server og Logit D2Ds webservice. Vi startet arbeidet utenfor Logit D2D systemene med egen testserver. På denne måten kunne vi arbeide uavhengig, samtidig som eventuelle vanskeligheter umulig kunne påføre deres systemer kritiske feil.

LogIT Systems hadde tilgjengelig programvare for å sette opp en testserver. I første omgang ville vi sette opp en egen testserver hvor vi kan arbeide mot denne lokalt. Om nødvendig kunne vi også utvikle en egen webservice å jobbe mot. Dette ville gjøre det lettere for oss å ha full kontroll og tilgang i startfasen, slik at vi kunne bli bedre kjent med systemet samtidig som vi jobbet mot det. Server ble satt opp på en tilgjengelig PC som vi i første omgang ville plassere på lokalnettet.

Når server var oppe startet utviklingen av programvare for kommunikasjon med nødvendige webservices. Arbeidet og programmering foregikk både på emulator og direkte på mobil enhet. I første omgang innebar dette å få tilgang til tjenestene, for så å utvikle toveis kommunikasjon. På dette tidspunktet forsøkte vi å få til kommunikasjon via WLAN. Eventuell tilgang via GPRS ville vi forsøke å få kikket nærmere på senere i prosjektperioden. Når kommunikasjon er oppnådd vil vi også få tilgang til webservice på lokal server.

Tilgangen skal utnyttes for å skape meningsfylt kommunikasjon. For å utvikle dette måtte vi bli kjent med nødvendige protokoller for kommunikasjon og formater for inn- og utdata. Dette så vi se nærmere på i LogIT Systems sine spesifikasjoner.

Vi var nå godt ute i prosjektperioden og alt nødvendig utstyr var på plass. Det var derfor på tide å se nærmere på innlesing av data. Leseenhet var innebygget i den mobile enheten og vi startet arbeidet med å kommunisere med denne for å innhente data. Dette krevde testelementer for strekkoder som inneholder relevant data.

De ulike applikasjonselementene var frem til dette utviklet i separate applikasjoner. Det var nå på tide å flette dette sammen til en applikasjon. Arbeidet besto av å få en fungerende ende til ende kommunikasjon hvor innleste data kunne nå systemet på testserveren. Når denne fasen var nådd hadde vi fullført hovedstrukturen i prosjektet, men det var likevel et stykke igjen til en fullverdig prototyp. Arbeidet videre besto dermed i å bygge på dette ”skjelettet” både med ytterligere funksjonalitet, design og brukervennlighet.

Applikasjonen var fremdeles mangelfull. Innhold med tanke på presentasjon og bearbeiding av data var på dette tidspunktet nesten fraværende. Dette var derfor det neste vi vil ta fatt på. Applikasjonsdelen skulle presentere brukeren for en rekke data, men før dette måtte disse bearbeides. Arbeidet vil derfor nå fokuseres på å tolke og bearbeide inndata både fra leserenheter og FTP. Disse dataene måtte systematiseres og presenteres på en oversiktelig og hensiktmessig måte for brukeren av systemet. Applikasjonen skulle gjøres i stand til å hente inn data fra Logit D2D server. Under registrering av elementer skulle den kunne utføre sammenligning og avmerking i listene med forventede produkter. Når dette var utført skulle applikasjonen så melde tilbake om resultatet til D2D server. Dette innebar en omfattende programmeringsdel. Vi arbeidet også her med implementering og utnyttelse av kommunikasjonsapplikasjonene vi tidligere har laget.

Selv om vi nå skulle ha en applikasjon med ende til ende kommunikasjon, og med de nødvendige funksjonene til å håndtere denne, så skulle den også kunne brukes av en hvilken som helst person. Design og brukervennlighet var derfor viktig å ta med som en del av utviklingen. For eksempel en lastebilsjåfør er ikke nødvendigvis den som er best kjent med slike applikasjoner, men det er sannsynlig at han er den som blir nødt til å håndtere dem. Et godt GUI (Graphical User Interface) er derfor viktig. Vi tok i bruk elementer rundt usability for å lage et godt og brukervennlig brukergrensesnitt. Skal et system eller en applikasjon bli godtatt og tas i bruk er det nødvendig å ta hensyn til selve brukeren. Det var her hovedsakelig de fem punktene under ”Usability” i Jakob Nielsensmodell [19] vi benyttet oss av.

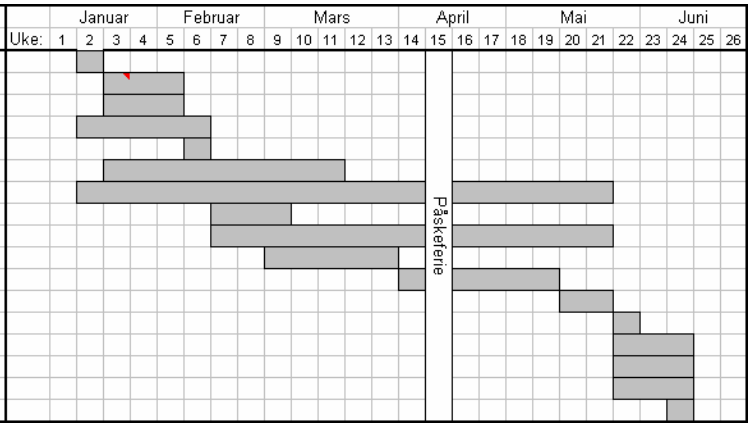
Applikasjonen nærmet seg ferdigutviklet og vi ville ta fatt på arbeidet med å se om ting fungerte slik de var ment. Reell testing mot LogIT sin testserver utgjorde en stor del av dette arbeidet. Dette var ikke live testing inn mot systemet som i dag er i bruk, men mot server som var tilsvarende.

Etter å ha kommet så langt hadde vi utviklet det vi hadde satt oss som mål. Da tiden tillot det så vi på mulighetene for videre utvikling. Vi forsøkte også å tilfredsstille noen av oppdragsgivers ønsker utover oppgavedefinisjonen.

3.1 Tidsperspektiv

Vi forsøkte å legge for oss et bestemt tidsperspektiv slik at arbeidet ble jevnt fordelt i prosjektperioden. Ved å følge denne planen slapp vi opphopning av arbeid og eventuelle overskridelser av deadline for oppgaven.

Oppgave	Fra	Til
Startdato for oppgaven	09.jan	09.jan
Valg av softwareløsning	uke 3.	uke 5.
Valg av hardwareløsning	uke 3.	uke 5.
Avgrense og definere oppgave	09.jan	06.feb
Tittel og oppgavedefinisjon fryses	06.feb	06.feb
Litteratursøk	uke 3.	uke 11.
Arbeid med rapport	09.jan	29.mai
Planlegge systemstruktur	uke 7.	uke 9.
Utvikling av applikasjon	uke 7.	uke 21.
Kommunikasjon med D2D server	uke 9.	uke 13.
Kommunikasjon med AIDC ustyr	uke 14.	uke 19.
Usability og design på applikasjon	uke 20.	uke 21.
Innlevering	29.mai	29.mai
Planlegging av presentasjon	29.mai	13.jun
Arebeide med poster	29.mai	13.jun
Arbeide med video	29.mai	13.jun
Presentasjon	14.jun	14.jun



Figur 5 Gant-diagram illustrerer planlagt arbeid og deadlines.

- 9. Januar var satt som den offisielle startdatoen for prosjektoppgaven. Det første vi da startet med var grundig litteratursøk i forbindelse med valg av software og hardwareløsning. At disse valgene ble riktige var kritiske for prosjektet, samtidig som det var viktig at de ble gjort relativt raskt, slik at det videre arbeidet kunne påbegynnes.

- Parallelt med dette arbeidet vi med å avgrense og definere oppgavedefinisjon og oppgavetittel. Dette ble utført parallelt siden valg av utstyr kunne innvirke på hvordan oppgaven kunne defineres, og oppgavedefinisjonen ville direkte stille krav til utstyr som ble valgt.

- Arbeidet med selve rapporten gikk igjen gjennom hele prosjektperioden. Ved jevnlig å dokumentere arbeidet effektiviserte vi utviklingen av en god rapport og forhindret opphopning av skrivearbeid mot slutten av prosjektperioden.

- Etter at tittel og oppgavedefinisjon ble satt var det viktig å fastsette målet man skulle jobbe mot. Planlegging av systemstrukturen ble utført i startfasen slik at man enkelt hadde kontroll over hvilke elementer man skulle forholde seg til.

- Videre ville hovedelementene i systemet være kommunikasjon med D2D server og kommunikasjon og innlesing fra AIDC utstyr. Her ville vi i første omgang arbeide med kommunikasjon med server. Dette skyldes mistanke om at levering av AIDC utstyr kunne drøye i tid. Vi ønsket å være sikre på at alt var på plass før vi tok fatt på denne delen.

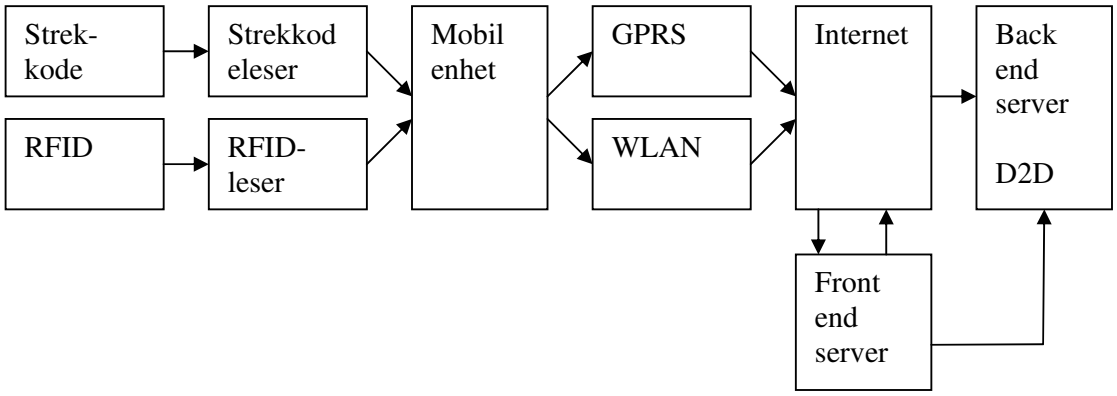
- Mot slutten av prosjektperioden ville vi også sette av litt tid til usability og design på applikasjonen. Dette var ikke av høyeste prioritet da det kun var en prototyp som skal produseres, men det krevde likevel et noenlunde fornuftig brukergrensesnitt(GUI).

- Etter at selve innleveringen er utført 29.mai vil vi starte arbeidet med å forberede fremføring, poster og videopresentasjon av prosjektoppgaven.

3.2 Kommunikasjonsstrukturen

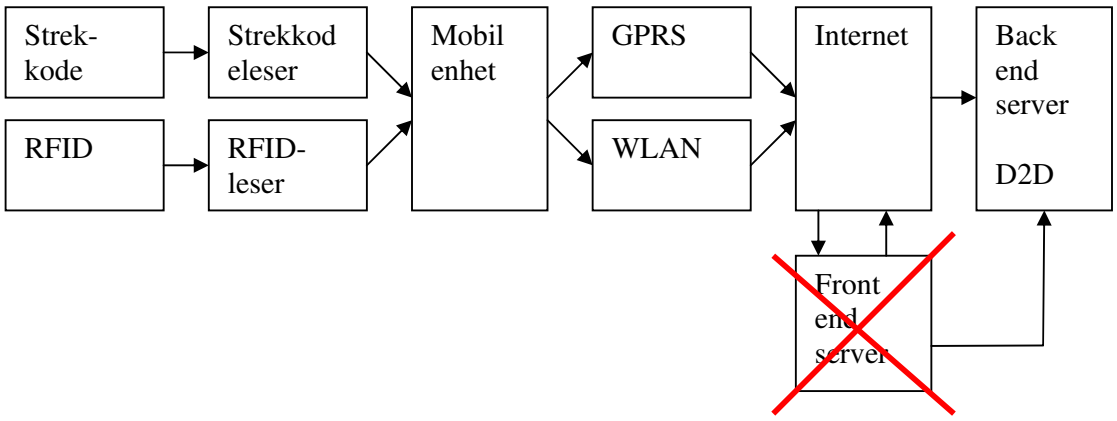
Vi skal her se på kommunikasjons-strukturen i systemet. Vi har kikket nærmere på hvilke enheter og muligheter som finnes på markedet i dag, samt på hva dette skal arbeide opp mot. Ut i fra dette har vi kommet frem til følgende struktur. Vi går fra kant og skriver om de ulike elementene og hvilke rolle de spiller i systemet.

Modellen illustrerer i første omgang en ID tag. Enten en strekkode eller en RFID brikke. Neste ledd er innlesingsenheter for disse teknologiene. Disse skal fange opp data fra ID tagene og videreføre dette til den mobile enheten. Her vil dataene i begrenset grad bli bearbeidet og sendt videre sammen med en rekke bestemte data fra den mobile enheten. Sendingen vil her foregå enten over WLAN eller GPRS. Via Internet vil dette skape forbindelse til en front end server. Her kan, om nødvendig, dataene bearbeides ferdig. Når dette er gjort, avhengig av serverens posisjon, kan signalene videreføres via Internet, eller direkte til D2D back end serveren.



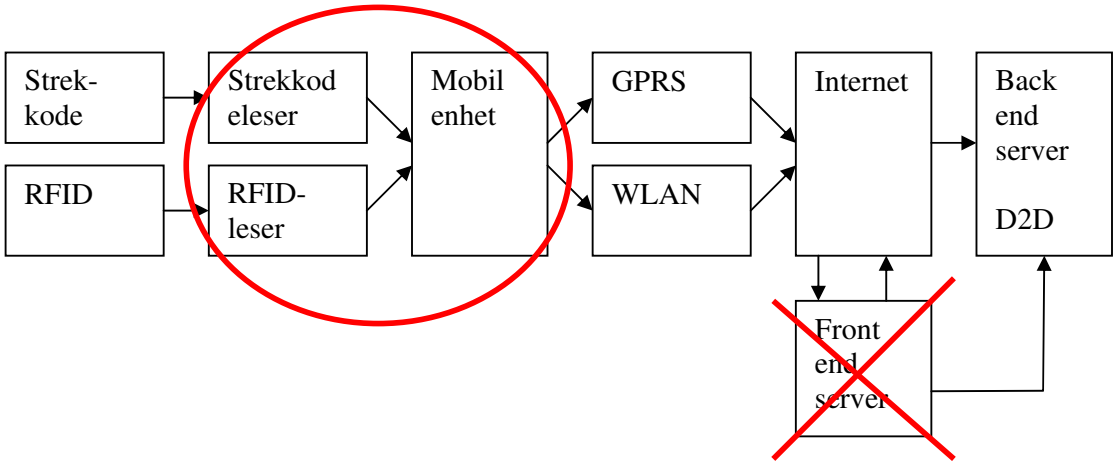
Figur 6 Kommunikasjonsstrukturen 1

Etter samtaler med LogIT har vi kommet frem til at bruken av en front end server ikke vil være nødvendig. Innlest informasjon skal ikke bearbeides før den leveres til Logit D2D server. D2D serveren har en webservice som kan kommuniseres med, og her vil det da være praktisk å kommunisere direkte mellom den mobile enheten og D2D serveren.



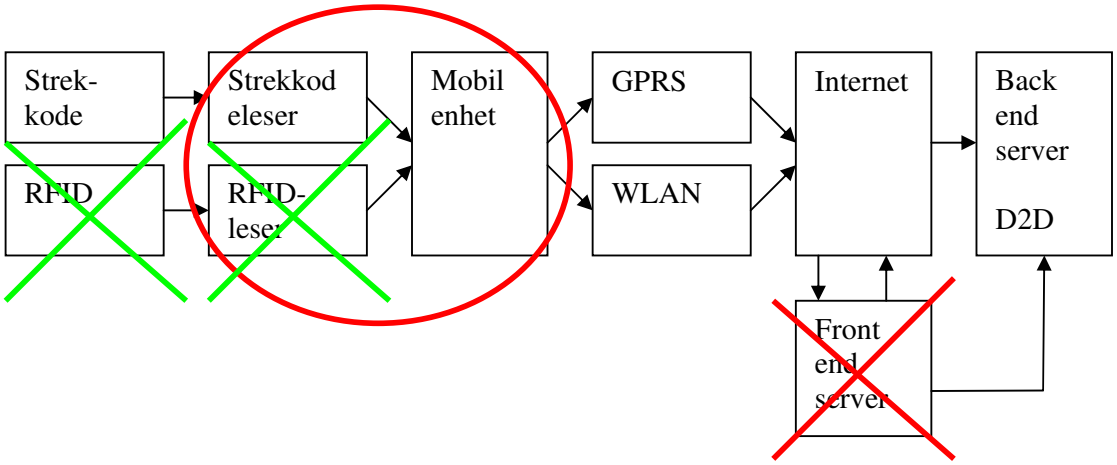
Figur 7 Kommunikasjonsstrukturen 2

Når det gjelder lese-enheter har vi etter nøye vurdering kommet frem til at den beste løsningen vil være en mobil enhet med innebygde leseenheter. Ved å ha disse funksjonene i en og samme enhet eliminerer dette flere punkter i kommunikasjons-strukturen.



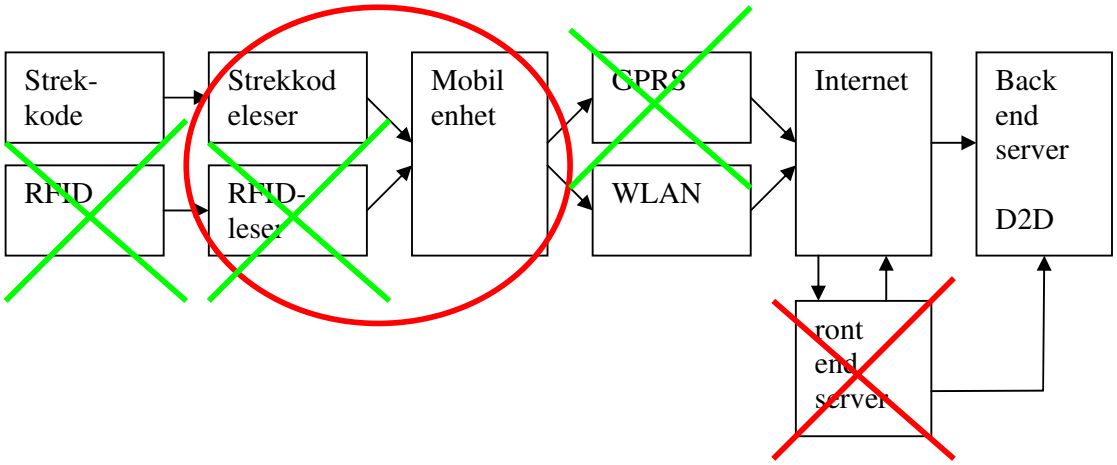
Figur 8 Kommunikasjonsstrukturen 3

Strekkode er ID teknologien som i dag er i bruk der hvor registreringsløsningen skal benyttes. Strekkode har derfor en klar førsteprioritet, mens RFID er prioritert bort. RFID vil ligge med i en fremtidig struktur, men kan utelukkes når det gjelder å illustrere vårt mål for denne oppgaven.



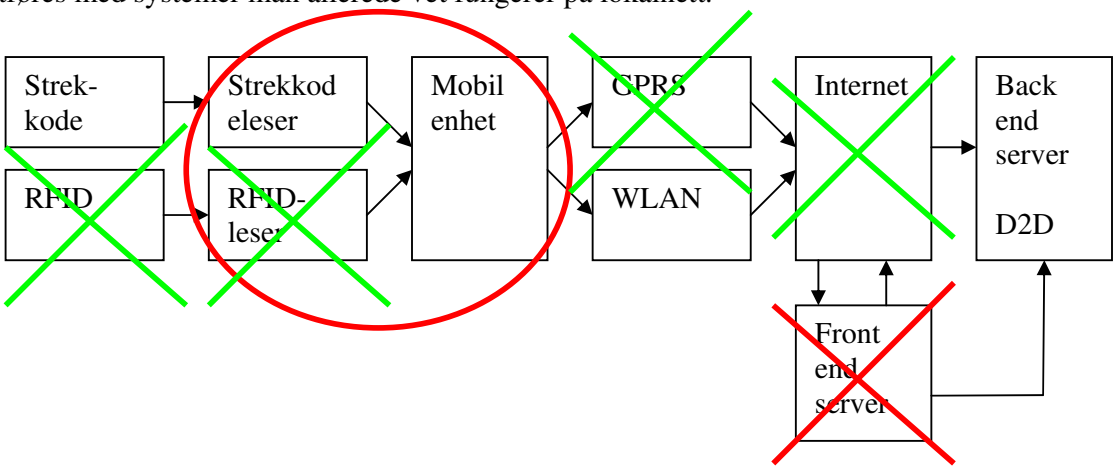
Figur 9 Kommunikasjonsstrukturen 4

Her gjelder det samme som med RFID. GPRS og WLAN utfyller hverandre, og en komplett løsning bør ha støtte for begge. Likevel vil det her i en demonstrasjons løsning kun være prioritert med WLAN, og GPRS blir derfor også utelukket. Dette skyldes ikke at det vil være større behov for den ene muligheten enn den andre, men at man rett og slett må ta en ting av gangen.



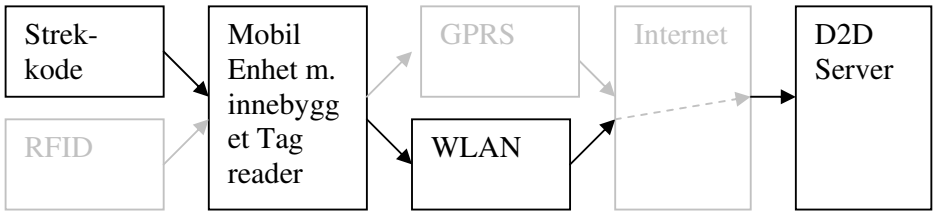
Figur 10 Kommunikasjonsstrukturen 5

Testing og utvikling vil foregå på lokalnett. Internet vil derfor kunne fjernes i den foreløpige strukturen. Utprøving på lokalnett i motsetning til Internet burde ikke utgjøre noen betydelige forskjeller. Kommunikasjon via Internet kan likevel føre til komplikasjoner, og arbeid på lokalnett vil derfor være en fordel og trygghet. Testing med data via Internet kan enklere utføres med systemer man allerede vet fungerer på lokalnett.



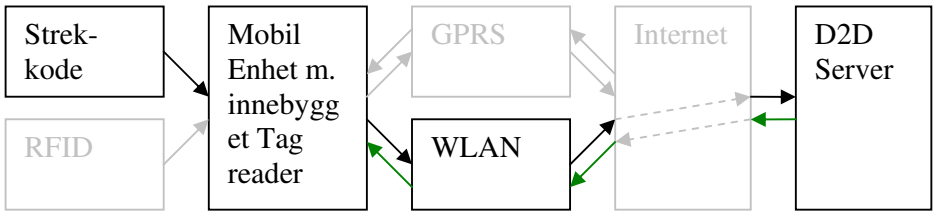
Figur 11 Kommunikasjonsstrukturen 6

Etter å ha gått igjennom den opprinnelige modellen ledd for ledd så sitter vi igjen med følgende modell. Denne illustrerer hvordan vi ser for oss strukturen i en ferdig prototyp. Her er elementene i svart det som skal prioriteres i førsteomgang, mens det som er grått er andreprioritet.



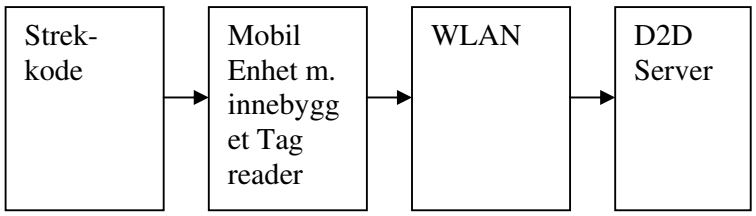
Figur 12 Kommunikasjonsstrukturen - Forenklet 1

Utover oppgavebeskrivelsen kom det også ønsker om muligheten for å formidle arbeidsoppgaver fra D2D server til den mobile enheten. Kommunikasjon mellom håndholdt enhet og server kunne derfor revurderes noe. Skulle serveren "pushe" oppdraget på enheten, eller skulle enheten selv velge oppdrag? Vi kom frem til at det siste var den beste løsningen. Hadde vi ønsket at oppdraget skulle "pushes" til enheten kunne vi benyttet SMS. Men bruk av SMS blir tungvindt når brukeren selv skal hente oppdrag, og da allerede har åpnet en kommunikasjonskanal. Vi vil derfor velge å overføre oppdragene via den samme stien som forespørselen kom fra. I og med at vi har prioritert WLAN for overføring av data til serveren, vil dette også bli benyttet for overføring av oppdrag til enheten. Skulle man benyttet GPRS ville dette bli benyttet i begge retninger.



Figur 13 Kommunikasjonsstrukturen - Forenklet 2

Til slutt sitter vi nå igjen med strukturen som illustrerer det vi ønsker å oppnå, og som vi skal jobbe mot i første omgang. Strukturen starter nå ved strekkoden. Denne skal leses av en mobil enhet med innebygget strekkodeleser. Heretter skal informasjonen overføres til D2D server via WLAN. Utover dette håper vi selvfølgelig også å kunne ta fatt på noen av de andre spennende elementene i strukturen for å utvikle et mest mulig tilfredsstillende system. Det ideelle hadde vært å ferdigstille strukturen vist i Figuren " Kommunikasjonsstrukturen - Forenklet 2", men å få i havn alle disse muligheten vil sannsynligvis være for tidkrevende innenfor prosjektperiodens tidsrom.



Figur 14 Kommunikasjonsstrukturen - Forenklet 3

3.3 Vurderinger før valg av enhet(er)

Valg av enhet eller sammensetning av enheter vil være svært avgjørende for resultatet av oppgaven, og er derfor viktig å legge vekt på. Vi har valgt å utføre et grundig forarbeid for å slippe å støtte på problemer grunnet begrensninger i utstyret. Det er også viktig at det ferdige produktet har realistiske fremtidsutsikter. Det må kunne konkurrere med lignende produkter på markedet i dag, og helst være åpent for implementering av fremtidige verktøy og teknologier innen sitt felt.

Jakten etter riktig hardwareløsning startet ved å diskutere behov og muligheter, for så å skissere nødvendig funksjonalitet med alternativer og ulike kombinasjoner. Hva skal

produktet brukes til, og hvordan? Hva innebærer dette? Hvordan kan det utnyttes og videre utvikles i fremtiden. Kan løsningen tilrettelegges for dette?

Helt overfladisk blir oppgaven å lese av en ID tag og levere denne informasjonen til sentral server sammen med lokasjon eller lignende data. RFID-leseren eller strekkode-leseren skal jobbe mot en håndholdt enhet, og det er viktig at denne enheten tilbyr nødvendig funksjonalitet, har god nok ytelse og har de nødvendige tilkoblingsmulighetene som trengs. Det er også viktig at enheten er forholdsvis robust, da brukeren gjerne befinner seg utendørs og kanskje i dårlig vær. Tripod Data Systems [20] tilbyr et eksempel på en slik håndholdt enhet, nemlig Recon. Den har grei ytelse og er vanntett(med noen forutsetninger). Den oppfyller MIL-STD-810F [21] standarden når det gjelder fall, vibrasjon og temperaturforskjeller. Ikke alle elektroniske enheter tåler temperatursvingninger og røff behandling, og kan fort ta skade av det. Batterilevetiden er og også viktig, og kan være avgjørende om enheten tas i bruk. Det er også avgjørende hvilke valgmuligheter man har av tilbehør og tilkoblingsmuligheter. Vi vil her se nærmere på hvordan vi har tenkt, og hvilke hensyn har vi tatt under vurdering av de ulike enhetene.

Tag reading:

Utfordringer: I dag er det hovedsakelig klassiske strekkoder som er aktuelt for oppdragsgiver LogIT Systems. Men også 2D strekkoder er svært utbredt, og nye teknologier er på trappene. RFID er for eksempel allerede i bruk på mange områder, og sprer seg raskt. Hva vil være standard om noen år? Hva vil konkurrenter og samarbeidspartnere benytte? Hvilke fordeler gir det?

Vurdering: En løsning bør som et minimum kunne dekke de teknologier som er aktuelle i dag. Dette gjelder da i første rekke strekkode og 2D strekkode, men vil nok også innebære RFID. Enheten som skal benyttes behøver derfor mulighet for innlesing av data fra disse. Når det gjelder fremtidige teknologier er dette vanskelig å ta høyde for på annen måte en å tilby tilkoblingsmuligheter, enten dette gjelder fysisk eller trådløst. Videre implementeringsmuligheter vil da avhenge av software og gå utenfor temaet valg av enheter. Som et ledd i utviklingen av en prototyp vil det likevel være nok med innlesing av en teknologi, og hovedfokus vil falle på strekkode som i dag er mest utbredt og også ønsket av LogIT Systems i første omgang.

Robusthet:

Utfordringer: Hvilket miljø skal utstyret utsettes for? På hvilke måter kan dette gå utover funksjonaliteten?

Vurdering: Temperatursvingninger, fuktighet, slag og støv kan fort sette elektroniske komponenter ut av spill. Det vil derfor være nødvendig med en enhet som er beregnet for å tåle de påkjenninger den kan bli påført av for eksempel lastebilsjåfører og bryggesjauere i de omgivelsene de befinner seg i. Dette vil kreve en robust enhet. I samråd med oppdragsgiver er det likevel kommet frem til at i første omgang er det funksjonalitet som teller, og at robusthet kommer i andre rekke. Dette skyldes at det kun er en prototyp som skal utvikles. Skal løsningen benyttes videre kan dette forbedres ved å få spesialutviklede enheter, eller tilpasse løsningen andre alternativer. Det er sannsynlig at det i fremtiden vil bli utviklet flere og bedre enheter tilpasset oppgaven. Robusthet vil derfor ikke være et fellende kriterium for valg av enhet i denne omgang, men være et betydelig pluss.

Nett-tilkobling:

Utfordringer: Innlesningsenhet vil behøve kommunikasjon med sentral server. Hvordan skal denne utføres? Hvor stabil må den være? Vil en teknologi være tilstrekkelig? Hvilket dekningsområde behøves? Hvor skal enheten benyttes, og hvilke forbindelser finnes der?

Vurdering: I første omgang trengs det her én forbindelse. Her kan det blant annet brukes GPRS, SMS eller WLAN. Avhengig av valg vil det variere om det er nødvendig med en eller flere teknologier. Brukes WLAN, noe som er svært aktuelt, vil dette fungere kun innenfor begrensede dekningsområder. Brukes GPRS vil dette fungere over et mye større område. En kombinasjon av disse ville gi god dekning takket være GPRS og en kraftig forbindelse vha. WLAN innen begrensede områder.

En eller flere enheter:

Utfordrnger: Bør løsningen bestå av en samling enheter, eller alt i en? Finnes det enheter som inneholder nok funksjonalitet og har en tilfredsstillende og hendig størrelse?

Vurdering: Begge deler har fordeler og ulemper. En enkel enhet blir lettere å holde styr på enn flere og har derfor en praktisk fordel. Størrelsen kan derimot gjøre dette til en ulempe igjen. Ved bruk av flere enheter kan en hovedenhet ligge trygt i en lomme eller i lastebilen mens mindre lese-enheter f.eks for strekkode og RFID kan benyttes tilkoblet med bluetooth. En fordel med dette er at man, om RFID ikke er aktuelt, kan kjøpe kun hovedenhet og strekkodeleser, eller motsatt. En ulempe med en hovedenhet er at data bør kontrolleres ved innlesing, og dette kan være vanskelig om enheten ligger på et annet sted. Valget her vil bli vurdert litt ut fra hva som befinner seg på markedet, men alt i en enhet vil nok være foretrukket.

Inntasting:

Utfordringer: Hvordan skal inntasting av kommentarer, rettelser og lignende fungere for brukeren? Hvor mange knapper skal en bruker forholde seg til? Ved direkte inntasting av tekst vil en form for tastatur forenkle jobben, men hvilke konsekvenser får dette?

Vurdering: Det visuelle førsteinntrykket og forståelsen av funksjonene vil være enklere med fysiske knapper. Men kan man kontrollere og deaktivere ønskede knapper? Eller risikerer man at de blir trykket på ved en tilfeldighet? F.eks i lommen. Mange knapper vil også gjøre enheten større, tyngre, og det kan kreve opp plass slik at man sitter igjen med et mindre display. Alternativet til fysisk tastatur er et virtuelt tastatur på display ved behov. Dette kan gjøre enheten mindre i størrelse og mer oversiktelig for bruker. Inntasting derimot kan være mer utfordrende for en som ikke er vant til denne teknikken. Virtuelt tastatur er trolig den beste løsningen i dette tilfellet.

Figur 15 TDS Ranger: Eksempel på enhet hvor fysisk tastatur opptar mye plass. Dette går også utover størrelse på skjerm.

Hastighet:

Utfordringer: Hvor tung blir applikasjonen å kjøre? Hva krever systemet av prosessorkraft?

Vurdering: Trolig vil applikasjonen ikke bli veldig tung å kjøre. De fleste enheter i dag har 200MHz eller 400MHz prosessor. Begge alternativer er nok tilstrekkelig. Likevel vil det alltid være en fordel med en kraftig prosessor. Med tanke på fremtidig bruk av enheten vil dette også gi flere muligheter for oppgraderinger og utvidelser av funksjonalitet.

Plattform/OS:

Utfordringer: På de håndholdte enheter, som er mest aktuelle for vårt prosjekt, er det Win CE .NET og Windows Mobile som er mest utbredt. Windows Mobile 2003 var basert på Win CE .NET 4.2. Siste utgave av begge OS'er er versjon 5, og det er støtte for flere nye API'er.

Vurdering: Det blir viktig å velge et OS der man lett kan utvikle applikasjoner, og som støtter de teknologier som man ønsker å benytte seg av.

J2ME og .NET CF er de to rammeverkene som er aktuelle. Det finnes forskjellige utviklingsverktøy for disse, f.eks. Netbeans for J2ME og Visual Studio .NET for .NET CF. Det finnes i tillegg diverse rammeverk som utvider funksjonaliteten til disse, eller forbedrer/forenkler eksisterende funksjonalitet.

De aktuelle programmeringsspråk er C++, Java og C#. Vi har litt erfaring med alle språk, dog er innsikten i C# bedre enn i Java og C++. Vi har utviklet programvare for .NET CF rammeverket tidligere. Velger vi .NET CF og C# vil nok produktiviteten være større enn ved bruk av J2ME og Java eller C++ grunnet erfaring.

Tilgang til emulatorer er også avgjørende, da man kan simulere en håndholdt enhet, og dermed kutte ned på kostnader og utviklingstid.

Kostnad:

Utfordringer: Hva vil dette koste en kjøper? Kan enheten tilpasses ulike behov? Blir prisen forsvarlig med tanke på funksjonaliteten?

Vurdering: Enheten blir en del av et stort system for å utvide mulighetene. Dette skal være salgbart og må derfor holde seg innenfor akseptable prisrammer. I vurderingen av utstyr vil pris ha en avgjørende rolle.

Funksjonalitet:

Utfordringer: Er enheten fremtidsrettet? Utover de absolutte krav, inneholder enheten funksjonalitet som kan vise seg nyttig? Aktuelle funksjoner: Telefoni, CF slots, Bluetooth, kamera og GPS.

Vurdering: Funksjoner gir muligheter. Jo flere man på en fornuftig måte kan få inn i enheten jo bedre er det. Selvfølgelig ikke til enhver pris. De nevnte funksjoner kan diskuteres bruken av. Telefoni er en svært aktuell ekstrafunksjon. Dette utvider enheten til et multiverktøy for

brukeren. På denne måten behøver arbeidsgiver å stille med kun én enhet til disposisjon for å dekke alle kommunikasjonsbehov. CF slots og Bluetooth gir fremtidsmuligheter ved tanke på tilkobling av ytterligere enheter og funksjonalitet. Kamera kan også utfylle en rolle i ulike situasjoner. Om systemet også er tilpasset bruk av bilder kan det for eksempel bli brukt ved rapportering av skadet gods under transporten og lignende. GPS kunne også gi flere muligheter i en utvidet løsning. Under registreringer ved lasting, lossing og utleveringer kunne en GPS mottaker automatisk registrere nøyaktig posisjon og rapportere til systemet. Det kunne også blitt brukt i forbindelse med kartplotting ved utlevering av gods. Mange funksjoner er egentlig ikke nødvendig i løsningen som her skal utvikles. Enheter som likevel inneholder disse funksjonene vil ha en fordel av dette videre i vurderingen med tanke på fremtidsmuligheter..

Brukervennlighet:

Utfordringer: Hvem skal bruke dette? Vil det bli nødvendig med opplæring? Hvor går grensene for hva brukeren er villig til å sette seg inn i og ta i bruk?

Vurdering: I første omgang må enheten ha en størrelse som gjør den praktisk. Er den for liten kan den fort rotes bort, og knapper og display kan bli for smått. Er den for stor derimot kan den bli klumsete å ha med å gjøre, og upraktisk å ha med seg. Når det gjelder funksjonalitet i software så må dette legges opp på en brukervennlig måte slik at det er så selvbeskrivende som mulig. Dette faller for så vidt utenfor feltet vi diskuterer her, valg av enhet, men vil likevel ha en viss forbindelse. Antall inndatataster på selve enheten vil nemlig ha en direkte innvirkning på hva som vil bli nødvendig av knapper og inntastingsmuligheter i programvaren. Her vil det trolig være en fordel å ha få knapper på selve enheten. Dette gir et ryddig og enkelt førsteinntrykk. Videre kan man overlate brukervennligheten til softwaredelen med tanke på enkel inntasting og navigering.

3.3.1 Kravspesifikasjon

Ut fra våre drøftinger rundt valg av utstyr har vi kommet frem til fem krav vi vil se på som det viktigste når vi velger enhet. De er som følger:

- Innebygd og/eller mulighet for tilkobling av ulike AIDC lesere
- Muligheter for trådløs tilkobling
- Mulighet for tilkobling av ytterligere utstyr
- God prosessorkraft
- Fornuftig pris i forhold til funksjonalitet

Et ytterligere krav for et ferdig produkt vil være som følger.

- Robust og tolerant mot slag, fukt og temperatursvingninger

Dette vil likevel ikke være avgjørende under valg av utstyr for utvikling av prototyp:

3.3.2 Utstyrs spesifikasjoner

Etter å ha gjort oss opp en mening om våre kriterier og krav for valg av utstyr har vi satt dette opp mot de enheter vi kunne finne på markedet i dag. Vi laget først en opplisting av de teknologier vi så for oss kunne håndtere de ulike elementene av oppgave. Vi sammenlignet så

dette med spesifikasjonene til de ulike alternativene. Vi ser her nærmere på noen av enhetene som var oppe til vurdering.

Tabell 1 Hvilke teknologier kan betjene de ulike elementene av systemet.

Tag reader	Overføring		Enhet	Overføring	Metode	Front end	Overføring	Backend d2d server
OCR	Trådløs:	Bluetooth	Pocket PC	WLAN	Webservice	JBOSS	Webservice	JBOSS server
RFID		IR	PC/laptop	GPRS	Mail	.net Framework	XML Service	
		GPRS	Mobiltelefon	SMS	HTTP			
		WLAN			XML Service			
	Fysisk:	RS232						
		USB						
		CF						

Baracoda Road runner:

En håndholdt strekkodeleser for klassisk og 2D strekkode. Leseren kan benyttes med Bluetooth for trådløs overføring med en rekkevidde på 100 meter. Leseren er laget robust og tåler en støyt. Kan benyttes sammen med for eksempel Qtek 9100.

Tabell 2 Spesifikasjoner Road runner

Tag reader	Overføring	
OCR	Trådløs:	Bluetooth

[22]

TDS Recon:

Recon er en solid enhet som inneholder det grunnleggende, og med tilkoblingsmuligheter til ytterligere utstyr. Den innebygde funksjonaliteten er likevel noe begrenset. Prosessoren er kun 200MHz.

Tabell 3 Spesifikasjoner TDS Recon

Overføring		Enhet
Fysisk:	RS232	Pocket PC
	USB	
	CF	

[23]

M3:

Dette er en enhet som er spesiallaget for logistikkbruk. Denne enheten inneholder en lang rekke nyttige funksjoner, og har muligheter for enda flere. Den har innebygget strekkodeleser, telefoni, GPRS og kamera. Utover dette har den 400MHz prosessor og blant annet RFID og GPS som options [12].

Tabell 4 Spesifikasjoner M3

Tag reader	Overføring		Enhet	Overføring	Metode
OCR (1D & 2D)	Trådløs:	Bluetooth	Pocket PC	WLAN	Webservice
RFID (option)		WLAN		GPRS	Mail
		GPRS/GSM		SMS	HTTP
		IrDA			XML Service
	Fysisk:	CF			
		USB/serieport			

Qtek 9100:

Qtek er designet for å være et kraftig verktøy for forretningsmenn. Denne er utviklet rundt det å være en mobil med alt! Minus med denne i forhold til prosjektet er at den ikke er designet for spesielt røffe forhold og at prosessoren er kun 200MHz. Fordeler er at den er liten og hendig, og at den inneholder god telefonfunksjonalitet. Den har også et uttrekkbart tastatur [24].

Tabell 5 Spesifikasjoner Qtek 9100

Overføring	Enhet	Overføring	Metode
Bluetooth	Pocket PC	WLAN	Webservice
WLAN		GPRS	Mail
GPRS		SMS	HTTP
			XML Service

Symbol MC 9000:

Dette er en enhet som er spesiallaget for logistikkbruk. Enheten har en 400MHz prosessor og gode tilkoblingsmuligheter. Den har også mulighet for telefonbruk, men designet er ikke tilrettelagt for dette. Pistolhåndtaket kan være praktisk ved effektiv bruk, men kan være klumsete og i veien ved oppbevaring og frakt [25].

Tabell 6 Spesifikasjoner Symbol MC 9000

Tag reader	Overføring		Enhet
OCR	Trådløs:	WLAN	Pocket PC
		Bluetooth	
	Fysisk:	RS232	
		USB	

3.4 Vurdering før valg av utviklingsmiljø

Når det gjelder utvikling av programvare for håndholdte enheter, er det to utviklingsmiljøer som passer meget godt. J2ME(Java 2 Micro Edition) [26] og .NET CF(.NET Compact Framework) [27]. Begge disse er beregnet til utvikling av programvare til forbruker elektronikk som mobiltelefoner, PDA'er og liknende produkter. De inneholder begge emuleringsmuligheter, noe som kan være meget kostnadsbesparende i startfasen av et prosjekt. Man har da muligheten til å utvikle deler av programvaren og kan på den måten undersøke om enkelte funksjoner er mulige å implementere. Slik kan man slippe å handle inn enheten før man er overbevist om at prosjektet er hensiktsmessig. En annen stor fordel med en emulator er mulighetene for enklere feilsøking kontra å feilsøke på den fysiske enheten. Dersom det er proprietære ting som f.eks. en strekkodeskanner må man dog feilsøke på enheten, da dette ikke vil fungere i emulatoren.

Emulatoren skal i teorien oppføre seg likt som om man hadde kjørt programvaren på en virkelig enhet. Vi vet erfaringsmessig at dette ofte ikke er tilfellet, og man støter relativt ofte på små problemer med at f.eks. GUI ikke blir seende helt lik ut.

Programmeringsspråket i J2ME er Java, i .NET kan man f.eks. bruke C#, C++ eller andre språk i .NET familien. Hvilket språk man velger, kommer an på flere faktorer. Tidligere erfaring med et eller flere av programmeringsspråkene vil være en stor fordel kontra programmeringsspråk man har liten eller ingen tidligere erfaring med. Hvis man kjenner språket kommer man hurtigere i gang med det produktive arbeidet og sluttresultatet bør følgelig være bedre. Dessuten løses som regel eventuelle problemer underveis enklere dersom man har en bedre forståelse for programmeringsspråket man jobber med, og man er mer mottakelig for hjelp fra andre.

De forskjellige programmeringsspråkene har sterke og svake sider, og det kan derfor hende at et programmeringsspråk egner seg bedre for den løsningen som blir forsøkt implementert. I tillegg til de funksjonene som rammeverkene J2ME og .NET CF tilbyr, finnes det andre rammeverk som kan gi ekstra funksjonalitet eller forbedre eksisterende funksjonalitet. OpenNETCF [14] Smart Device Framework er et slikt rammeverk for .NET CF. Dette open-source prosjektet gir tilleggsfunksjonalitet, og forbedrer allerede eksisterende muligheter i .NET CF som f.eks. serial port kommunikasjon(RS232), bluetooth med mer. I skrivende stund er versjon 2.0 Beta 1 den nyeste tilgjengelige versjonen, denne har støtte for .NET Compact Framework 2.0. Den siste stabile versjon er imidlertid 1.4 for .NET Compact Framework 1.x.

Håndholdte enheter utviklet spesielt for logistikkbruk, inneholder gjerne en innebygget strekkodeskanner, WLAN kort og andre ting. For å kunne benytte seg av disse tingene utvikler leverandørene programvare, et såkalt SDK(Software Development Kit). På denne måten kan utvikleren, i dette tilfellet oss, kontrollere disse enhetene fra programvaren vi utvikler. Hva slags SDK leverandøren tilbyr og hvilket programmeringsspråk dette støtter er en vesentlig faktor for hvilket utviklingsmiljø man bør velge for sitt prosjekt. Dersom SDK'et er utviklet i Java, med tilhørende kodeeksempler, vil det være lite hensiktsmessig å utvikle programvaren i f.eks. C# og vice versa.

Det er også viktig å få med at LogIT Systems er interessert i å kikke nærmere på .NET plattformen, da de stort sett benytter seg av Java i deres daglige arbeid. Benyttes dette kan vi forhåpentligvis vise nytteverdien av .NET plattformen og den lave terskelen for å utvikle applikasjoner til mobile og stasjonære enheter.

En viktig sjekkliste for valg av utviklingsmiljø vil være:

- Støttes enheten man skal utvikle programvaren for?
- Vil sluttresultatet i stor grad påvirkes av hvilket utviklingsmiljø man velger?
- Hvilken innsikt/erfaring har man med de forskjellige miljøene?
- Kan man enkelt søke hjelp dersom man står fast?
- Hvis det finnes et SDK, hvilket utviklingsspråk støttes av dette?
- Spesielle ønsker fra oppdragsgiver?

3.5 Vurdering før valg av trådløs teknologi

Teknologiene vil benyttes for å kunne hente oppdrag fra Provider Booking server til PDA og for å oversende de klargjorte XML filene fra PDA og til LogITs D2D system. Det er viktig å finne passende trådløs teknologi. Oppdragsgiver har i oppgavebeskrivelsen nevnt teknologier som WLAN, GPRS og SMS.

WLAN og GPRS er nok de mest aktuelle teknologier for prosjektet. SMS kan muligens benyttes som en form for varslingsmelding om nye oppdrag og lignende, men den lille datamengden SMSer kan overføre begrenser dens videre muligheter. En SMS meldings maksimale størrelse er på 140bits, som tilsvarer 160 7-bits tegn. WLAN derimot bruker radiobølger og 802.11G på 2.4Ghz er veldig vanlig på markedet i dag. Dette gir en teoretisk overføringshastighet på opptil 54Mb/sek. I trådløse nettverk er sikkerhet viktig, men dette har vært en svakhet. WEP(Wired Equivalent Privacy) var den første løsningen som ble presentert, men denne har vist seg å ha store svakheter. WPA(Wi-Fi Protected Access) ble laget for å rette opp i de svakhetene WEP hadde. Det finnes to forskjellige typer trådløse nettverk, ad-hoc og infrastruktur. Ad-hoc vil si at to trådløse enheter kommuniserer direkte med hverandre, uten innblanding fra sentrale aksesspunkter. Infrastruktur lager en ”bro” mellom det trådløse og det trådbaserte ethernettet, og det vil typisk være en ruter som deler ut IP adresser til WLAN klienter.

GPRS er en mobil data service som tilbys brukerne av GSM mobiltelefoner. GPRS er pakkesvitsjet, noe som betyr at flere kan bruke samme transmisjonskanal. En sannsynlig bitrate på ca 40kbps. Man kan sende og motta filer, og f.eks. ”surfe” på Internet osv.

Et krav til denne oppgaven er å kommunisere med server i en eller annen form. Til dette ser vi WLAN som et praktisk utgangspunkt. Kommunikasjon ved hjelp av WLAN er en meget godt egnet teknologi for bruk i mindre områder, da rekkevidden er forholdsvis begrenset. Dessuten har WLAN en stor fordel kontra GPRS når det gjelder overføringshastigheter. For en prototyp vil det være en veldig god løsning med tanke på utvikling, testing og demontrasjoner for potensielle kunder.

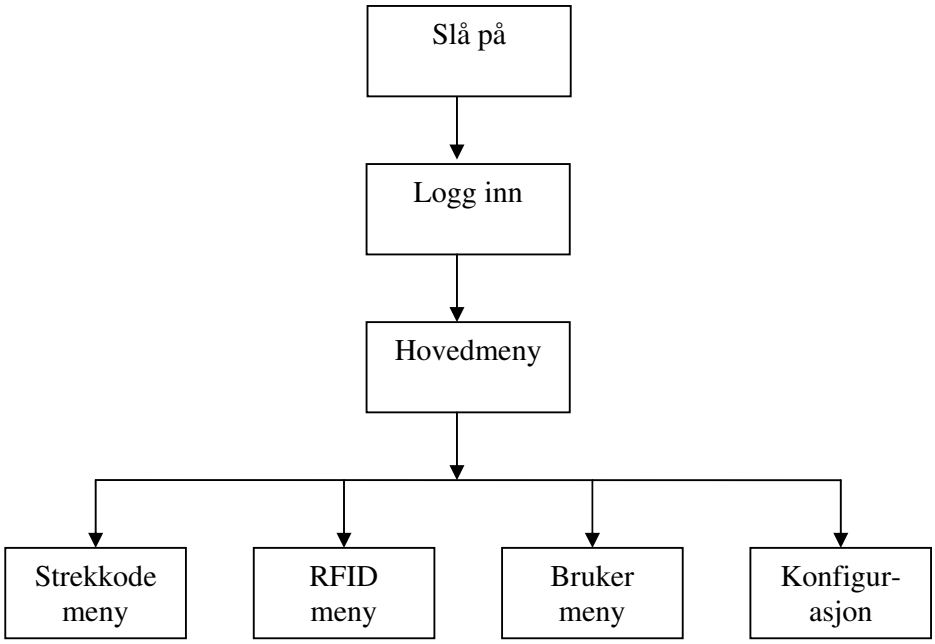
Neste steg etter implementasjon av WLAN kommunikasjon er å kunne benytte seg av GPRS kommunikasjon. Dette for å kunne dekke de områdene som ikke har WLAN dekning. Ulempen med GPRS er hastigheten og kostnadene forbundet med overføring av data.

Å finne en mobil enhet som har innebygd støtte vil være svært fordelaktig med tanke på fremtidig bruk. Dette vil derfor påvirke valget av mobil enhet, da det er svært ønskelig at denne skal inneholde støtte for både WLAN og GPRS, og da fortrinnsvis innebygd i selve enheten.

3.6 Menystruktur

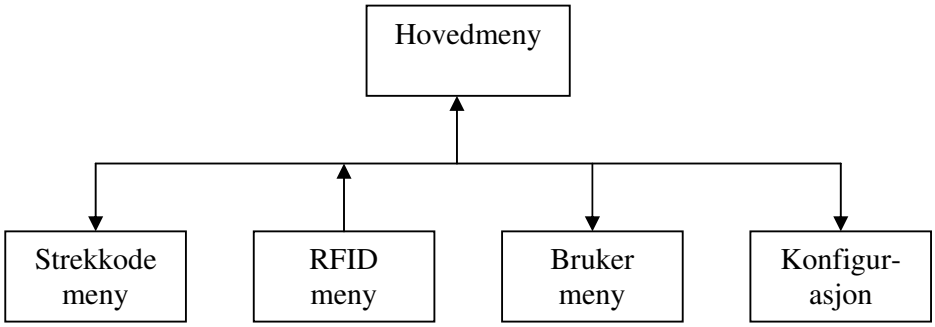
Det er alltid viktig at navigeringen er enkel å forstå. Dette krever logiske menyvalg og logiske samlinger av funksjoner. Dette gjelder i aller høyeste grad også her. Det er stor sjanse for at personene som skal ta løsningen i bruk til daglig overhodet ikke har dataerfaring. Oppsettet vil derfor gjøres så intuitivt som mulig.

Bruken vil i alle tilfeller starte med at maskinen slås på(ut av dvale). Man vil da bli presentert med et logg inn vindu. Her må brukeren logge seg inn med sitt brukernavn og passord for å kunne identifisere seg selv under arbeidet. Videre vil man da bli presentert med en hovedmeny. Her vil brukeren bli presentert med de valg som er tilgjengelige for øyeblikket. I illustrasjonen lenger nede på siden vil du se mulighetene ”Strekkode meny”, ”RFID meny”, ”Bruker meny” og ”konfigurasjon”(konfigurasjon bør muligens være passordbeskyttet, slik at kun administratorer har tilgang til å utføre endringer). Dette er valgene vi ser for oss at bruker skal presenteres for, og være ”ankerpunkt” for hvor man befinner seg, og hvilke oppgaver man utfører i applikasjonen.



Figur 16 Navigasjons strukturen 1

Navigasjon mellom de ulike hovedvalgene skal kunne skje på ethvert tidspunkt så lenge man er innlogget. På denne måten er det alltid en rask vei for å få utført en oppgave. Neste illustrasjon viser hvor man for eksempel kan navigere fra RFID menyen. Her har man tilgang tilbake til hovedmenyen, men også til de andre hovedmenyvalgene direkte. På denne måten har man hele tiden frihet til å utføre de oppgaver man ønsker.



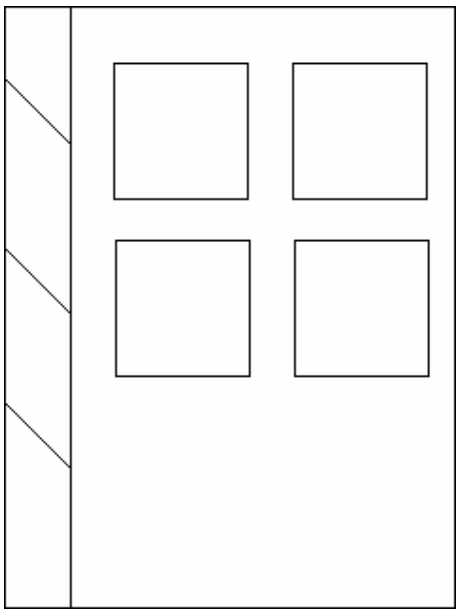
Figur 17 Navigasjons strukturen 2

En del av menyene vil i sin tur ha egne undervalg og alternativer. Disse vil være mer detaljerte og spesifikke, og vi vil ikke gå nærmere inn på utformingen av dem her. Disse vil likevel ikke dekke til eller ødelegge funksjonaliteten til hovedmenyen, som fortsatt vil gi brukeren friheten til å bevege seg rundt i applikasjonen.

3.7 Planlegging av design

Vi har gjort oss en del vurderinger rundt design av applikasjonen. Vi ser for oss at den skal dekke hele skjermbildet. Dette for å gjøre andre, uvesentlige funksjoner utilgjengelige for brukeren.

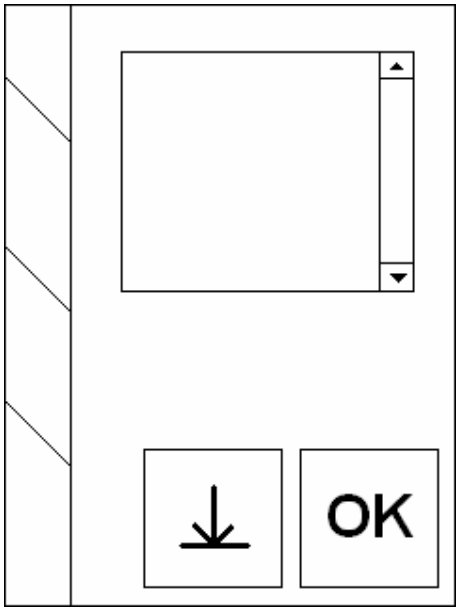
Applikasjonen skal benyttes av blant annet transportarbeidere og bryggesjauere. Vi ser derfor for oss at store knapper som er lett tilgjengelige er en god løsning. Vi vil gjerne eliminere bruken av skrivestift så mye som mulig. Hovedmenyen vil derfor bestå av store ikoner med skrift under i vanlig "windows-stil". På denne måten er det enkelt å trykke og treffe selv med store fingre. Ikonene er illustrert med fire store firkanter i displayet t.h.



Figur 18 Skisse av skjermbilde. Hovedmeny.

For at hovedmeny og andre menyer skal være lett tilgjengelige til enhver tid har vi valgt en struktur med tabs. Disse er lagt som menyvalg oppover langs venstre side av skjermbildet. Disse meny-tab'ene vil være tilgjengelige til enhver tid i alle vinduer i applikasjonen. Tab'ene vil også illustrere hvor man befinner seg, ved at de "lyser opp" med en annen fargerkombinasjon der man er.

Designet videre må tilpasses de ulike oppgavene. På dette tidspunkt er det vanskelig å se for seg det ferdige designet da det vil være nødvendig med en rekke knapper og valgalternativer som enda ikke er klarlagt. Vi har likevel sett for oss, og tegnet en rask skisse av hvordan vi skjermbildet for innlesing av ID tager. Her ser man til venstre tab-menyen som vil være i alle skjermbilder. Øverst har vi tegnet inn et vindu som er tiltenkt ID tag-informasjon. Ved innhenting av registrerings oppdrag vil de ulike pakkene listes opp her. Ved enkeltregistreringer vil også tag-id presenteres for å illustrere vellykket innlesing.



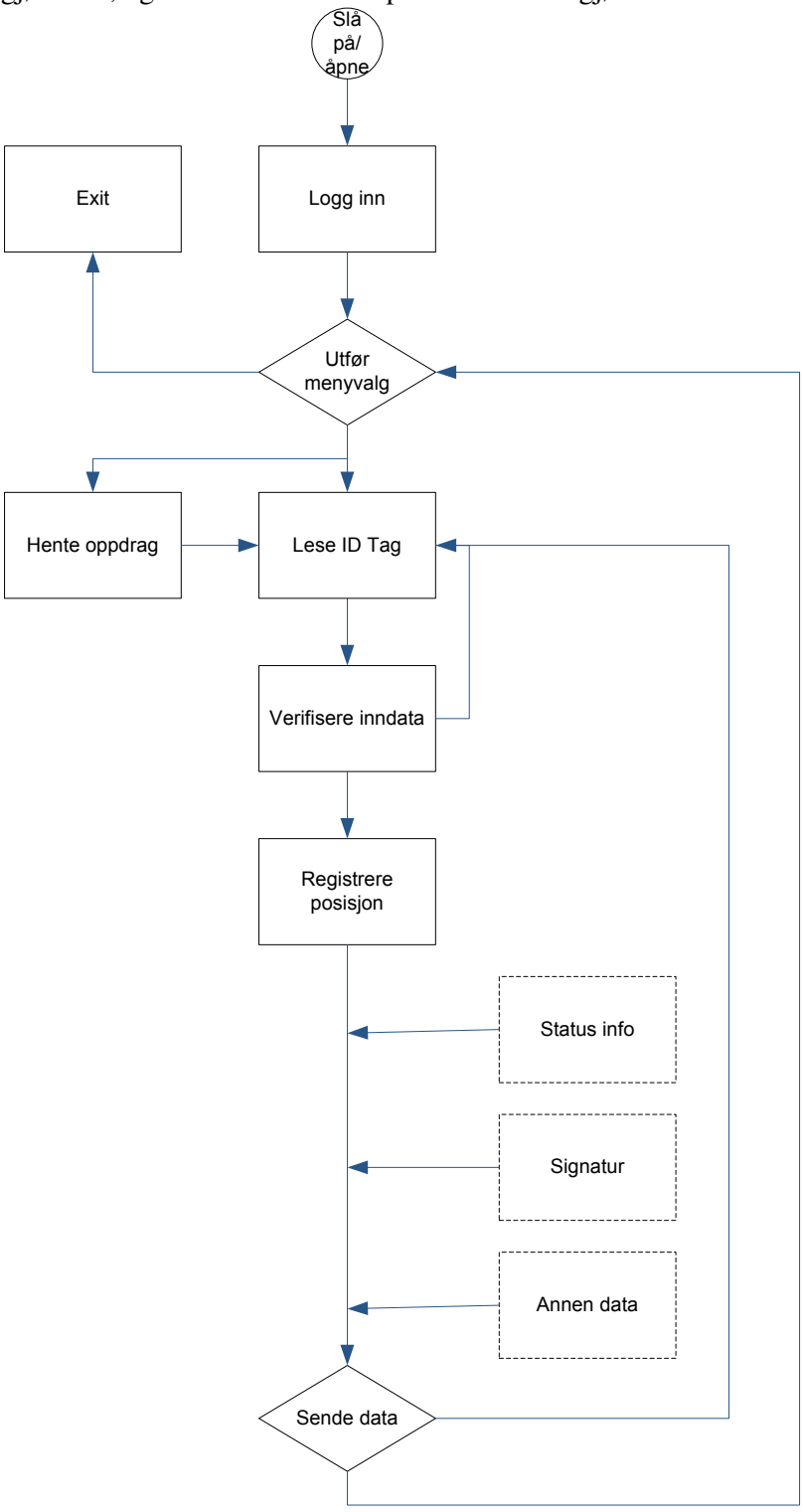
Figur 19 Skisse av skjermbilde for innlesing.

Også her er knappene laget store. Begrunnelsen for dette er samme som tidligere: Man skal enkelt kunne utføre jobben kun vha. fingrene. Dette vil likevel utgjøre et problem. Skjermstørrelsen er svært begrenset, og større knapper vil fort utgjøre en stor del av skjermbildet. En kombinasjon av store knapper, oversiktelig design og all nødvendig informasjon kan bli vanskelig å presse inn på et så lite display.

Når det gjelder designet utover dette vil vi være beskjedne i fargebruken. Lys bakgrunn med store kontraster mot tekst vil gjøre informasjonen lett leselig. Vi vil forsøke å være konsistente med navngiving og plassering av knapper og alternativer. Vi vil se nærmere på designen mot slutten av oppgaven. Frem til da vil prioriteten være funksjonalitet.

3.8 Lesing av ID Tags

Vi ser for oss at applikasjonen startes ved oppstart. Dette begrenser brukerens muligheter for å gjøre feil, og fokuserer arbeidet på det som skal gjøres. En uerfaren bruker kan fort forville seg inn på konfigurasjoner og lignende som kan være kritiske for at systemet skal virke. Vi har her laget et førsteutkast av hvordan vi ser for oss at lesing av ID tags skal foregå. Flytdiagrammet illustrerer handlingene i bruk fra start til slutt ved registrering av gods.



Figur 20 Flyt diagram for tag-reading.

Når applikasjonen åpnes vises logg inn skjermbildet.

Logg inn: Her må brukeren registrere seg med brukernavn og passord. På denne måten kan ikke hvem som helst ta applikasjonen i bruk, og registrering av gods kan knyttes opp mot en bestemt person. Brukeren vil bli logget ut automatisk etter en bestemt tidsperiode om applikasjonen er innaktiv.

Utføre menyvalg: Etter innlogging ender man opp på hovedmenyen. Her møter brukeren en rekke valg. I dette tilfellet er det valgt lesing av ID tag (strekkode eller RFID).

Hente oppdrag: Denne funksjonen vil benyttes for å hente ned arbeidsoppgaver fra en sentral server. Arbeidsoppgaven er en samling strekkoder som ønskes registrert ved en bestemt forflytning, lasting eller lossing.

Lese ID tag: Dette utføres ved enten ved hjelp av en egen fysisk knapp på den mobile enheten. Data fra den aktuelle ID tag vil så bli lest inn i applikasjonen. Det er også lagt inn en knapp i skjermbildet som kan benyttes for å utføre samme jobben. Denne er gjort stor og tydelig slik at brukeren enkelt kan klikke på den direkte med fingeren.

Verifisere inndata: Dette innebærer å kontrollere at datainnlesing er vellykket utført.

De tre neste oppgavene var tidlig i arbeidet meget uklare. Det krevdes ytterligere innsikt i hva som skulle registreres og oversendes. Vi kikket her på følgende muligheter, men forventet forandringer utover i utviklingen.

Status info: Ved registrering av gods er det i mange tilfeller nødvendig med statusinformasjon. Med dette mener vi f.eks ”avhentet”, ”videresendt”, ”stoppet for importfortolling”, ”utlevert” osv. Et alternativ for å løse dette kan være en dropdown meny som tilbyr de ulike alternativene.

Signatur: Ved for eksempel utlevering av varer kan signatur fra mottaker være nødvendig. Dette kan foregå digitalt på display og oversendes sammen med statuspakke til sentral server.

Annen data: Det vil høyst sannsynlig også være behov for annen data i både denne og fremtidige versjoner. Dette legges til rette for.

Exit: Brukeren kan velge å avslutte applikasjonen for å ha mulighet til å benytte enheten på andre måter. Om dette skal være passordbeskyttet eller ikke er en avgjørelse oppdragsgiver må ta.

4 Identifikasjonssystemer

Det finnes i dag en rekke identifikasjonssystemer for automatisk innlesing av data. Kjente betegnelser på dette er ”Automated Identification and Data Capture” (AIDC) og ”Automatic Identification” (Auto-ID). Vi vil her se nærmere på en rekke av de teknologiene som er i bruk i dag. Vi har også forsøkt å kikke litt på hva som kommer i fremtiden og se hva utviklere og leverandører har å by på.

Teknologiene vi i utgangspunktet prioriterte å studere nærmere var strekkode og RFID. Videre finnes det AIDC teknologier som biometri, magnetstriper, OCR, smartkort og stemmegjenkjenning [28]. Alle teknologiene er benyttet på en rekke områder og finnes i ulike varianter. Vi vil her se nærmere på hva de kan tilby, og vurdere om de vil være relevante overfor Logit D2Ds systemer i dag og i fremtiden. Hvilke datatyper og datamengder er de ulike løsningene i stand til å håndtere, og hvilke data lagres på slike identifikasjonsmerker i dag?

Vivek Argarwel [29] legger i sin rapport frem fire punkter som alle disse systemene for automatisk identifikasjon deler.

> – There is a product, part, component, package, pallet, tote box, barrel, etc. Accurate identification of this item, while moving into or through production, warehousing, or the distribution pipeline, will contribute to the benefits case.
>
> – A label, tag or coding device is affixed to the item so that it can be automatically read to identify what the item is, where it came from, where or to whom it is going, or whatever else might be needed by the user.
>
> – An automatic or hand-held bar code reader, optical character reader, magnetic stripe reader, vision system, or radio frequency interrogator will read the code, validate it, and convert the content into system-meaningful control and information output.
>
> – The code reader transmits the output to networked PC's, mini-computers, relays, solenoids, microprocessors, programmable controllers, diverters, counters, video displays, horns, bells, whistles, etc. for data manipulation or communication.
>
> [29]

4.1 Strekkode:

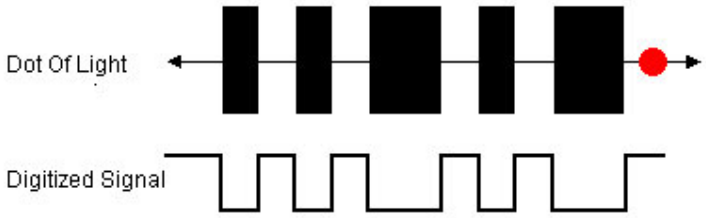
Strekkoder har vært i bruk i en årrekke. ”*Fem milliarder strekkoder blir skannet hver eneste dag i 140 ulike land* [29].” Dette viser noe av omfanget strekkoder har fått i dag. I 1952 ble en strekkodeløsning for første gang patentert, mens i 1966 ble det brukt kommersielt for første gang. Det ble etter hvert utviklet en standard for strekkodenes utforming UPC, og i 1974 ble den første UPC skanneren installert på et supermarked i Ohio, USA [30].

Det finnes forskjellige typer strekkodelesere [31] – [32] på markedet. De bruker alle hver sin metode for å lese og dekode strekkoden. Hva slags strekkodeleser man velger kommer an på formålet de skal brukes til. Det finnes laserbaserte skannere, som benytter seg av lys sendt ut

fra skanneren for å kunne identifisere og verifisere en strekkode. CCD(Charged Coupled Device) bruker lyssensorer i strekkodene som skanneren samler inn og dekoder for identifikasjon. Den andre typen er kamerabaserte skannere. Disse tar bilder av strekkodene, analyserer fargeskjærene og kontrastene i bildet og på den måten identifiserer strekkodene.

Strekkode skannere er vanligst med to forskjellige typer interface mot en PC, en variant som kan plugges rett i tastatur porten og en med RS232 [33] kommunikasjon. På en håndholdt enhet, vil strekkode-skanneren typisk benytte RS232 porten eller CompactFlash tilkobling. Varianten som plugges rett i tastaturporten gjør innlesingen meget enkel. Dataene går til PC'en akkurat som om noen hadde tastet det inn manuelt via tastatur. Denne metoden har til gjengjeld en del ulemper. Man har liten kontroll over hvor dataene havner når de skannes inn, de riktige feltene i applikasjonen må ha fokus, da dataene havner i vinduet og feltet med fokus. Det er heller ikke muligheter for å endre på dataene underveis da tastaturet blir utilgjengelig. RS232 kommunikasjon er mer kompleks, men gir til gjengjeld mer fleksibilitet med tankte på datamanipulasjon og kontroll over dataene.

Strekkoder leses som oftest av med en laser-leser. Man trenger derfor ikke fysisk kontakt for lesing, men behøver "klar sikt" mellom leser og strekkode. Lesere kan normalt håndtere avstander fra ca 0-20cm. Spesial-lesere kan også håndtere mye større avstander.



Figur 21 Digitalt signal ved skanning av strekkode

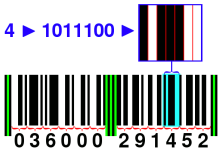
Strekkoder finnes i dag i en rekke varianter. Den mest utbredte er den klassiske endimensjonale løsningen. Videre har det blitt utviklet todimensjonale strekkoder, noe som tilbyr mange nye muligheter. Å gå i dybden på alle de ulike typene strekkode ville være et omfattende arbeid, vi har derfor valgt ut noen av de typene som skiller seg ut, og/eller vi ser som mest relevante.

4.1.1 Endimensjonale strekkoder

Figur 22 Teksten "Logit D2D" kodet med Code-128 (Alphanumeric) [34]

Endimensjonale strekkoder kan kun lagre relativt små datamengder. Normalt benyttes dette til produkt-IDer eller andre korte identifikasjonsnumre bestående enten av tall, eller kombinasjoner av tall og nummer.
Disse verdiene sier som oftest lite i seg selv, men knyttet opp mot sentrale systemer og databaser kan en leserenhet raskt hente informasjon rundt produktet. Dette er i dag i bruk i utallige sammenhenger som for eksempel kolonialforretninger, logistikk og produksjonssystemer.

Universal Product Code
Universal Product Code (UPC) er utviklet for å standardisere utforming og innhold i strekkoder brukt på produkter i butikker. Koden består av 95bits og inneholder en start, midt og slutt for å forenkle lesingen. Koden består kun av tall, og de ulike sifrene i koden har bestemte betydninger og må leses på følgende måte.



Figur 23 UPC-A Barcode [35]

Strekkode: S**LLLLLL**M**RRRRRR**E

- ”S” står for start og presenteres med den faste bittstrømmen ”101”
- Første ”L” siffer står for; 0 – vanlig produkt; 5 – kupong; osv…
- De fem neste ”L” sifrene er en produsentkode.
- ”M” står for midten med bittstrømmen ”01010”
- De fem første ”R” sifrene er produktkoden satt av produsenten
- Siste ”R” siffer er en sjekksum

S, M og E utgjør garder og forenkler lesingen av koden.

ISBN og ISSN
Ikke ulikt UPC brukes ISBN og ISSN for unikt å identifisere henholdsvis bøker og blader. Disse kodene finner man i dag bak på de aller fleste publiserte bøker og blader i form av en strekkode. ISBN er på samme måte som UPC bygget opp under bestemte regler, men disse skal vi ikke gå nærmere inn på her.

4.1.2 Stablede strekkoder

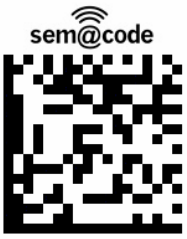
Stablede strekkoder (Stacked barcodes) lages ved å benytte endimensjonale strekkoder og legge i høyden over hverandre. På denne måten får man benytte flere ”linjer” inne i strekkoden. Dette blir derfor en mellomting mellom ekte 2D strekkoder og lineære strekkoder.

4.1.3 Todimensjonale strekkoder


Figur 24 Teksten ”Logit D2D” kodet med Datamatrix (alphanum) [34]

Ved å benytte todimensjonale strekkoder utvides lagringskapasiteten og mulighetene betraktelig. Strekkodens størrelse kan også reduseres da endimensjonale strekkoder blir bredere etter datamengde, mens todimensjonale også kan utnytte den vertikale plassen. 2D strekkoder har blitt mulige å ta i bruk etter hvert som mer avansert leserutstyr har kommet på markedet. Ulike utviklere har kommet opp med en rekke løsninger på hvordan denne teknologien kan benyttes. To eksempler på todimensjonale strekkoder som finnes i dag er ”Semacode” og ”DataGlyphs”.

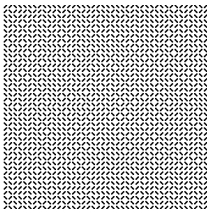
Semacode
Semacode er utviklet for å kunne presentere webadresser i todimensjonale strekkoder. Denne løsningen er utviklet i forbindelse med at mer og mer avanserte mobiltelefoner kommer på markedet. Ved hjelp av leseenheter eller kamera skal man kunne lese denne koden og bli sendt direkte til en webadresse på sin mobil. Teknologien kan benyttes på en rekke av dagens avanserte mobiltelefoner. Software og detaljer er tilgjengelige på http://semacode.org/ [36]



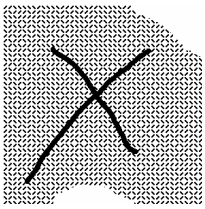
Figur 25 Semakodelink til http://semacode.org/ [36]

DataGlyphs

Ved å utnytte de muligheter todimensjonale strekkode har gitt oss har blant annet "Parc DataGlyphs" lagt til rette for en rekke nye bruksområder. DataGlyphs er fleksible i form og størrelse og kan gjemmes inn i bilder, benyttes på kurvede overflater eller gjøres usynlige på hvite ark. Egnete bruksområder er for eksempel å forhindre forfalskninger og svindel, gods sporing, ID kort, merking av reservedeler og produkt merking [37].

1. 

2. 

3. 

Figur 26 Oppgavebeskrivelsen kodet med DataGlyphs

DataGlyphs har robusthet og feilkorreksjon på høyt nivå. På denne måten kan selv skadede koder leses. På deres websider kan man selv generere koder. Om man ønsker, kan man i etterkant manipulere dem vha. en bilde editor, for så å forsøke å lese resultatet.

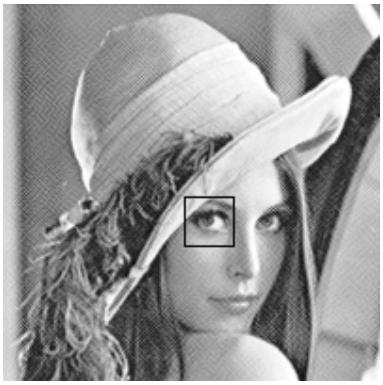
Nærmere forsøk viste at relativt store feil og mangler faktisk kunne håndteres. Oppgavebeskrivelsen "Utvikling av system for innlesing, bearbeiding og informasjonsutveksling mellom håndholdte enheter og Logit D2D" ble kodet med DataGlyphs. Resultatet kan sees på bilde 1. Bilde 2 viser et nærbilde av et lite felt av strekkoden.

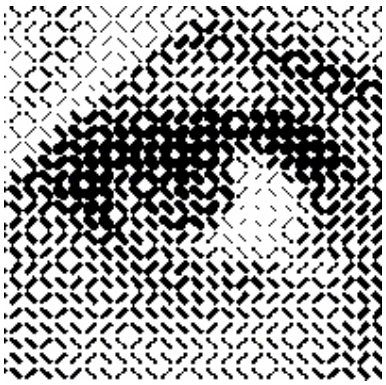
Strekkoden ble så åpnet opp i en bildeeditor hvor biter ble klippet bort og et stort svart kryss ble tegnet tvers over.(se bilde 3.) Etter dette ble bildet forsøkt lest. Resultatet var vellykket. Teksten ble gjengitt 100% korrekt.

Etter dette ble bildet forsøkt lest etter ytterligere ødeleggelser. Da med mislykket resultat.

Dette viser at også strekkoder med relativt store datamengder kan håndtere skader og mangler uten at kode går tapt. Selvfølgelig vil det likevel være en grense hvor gjenværende kodemengde blir for liten til å rekonstruere de opprinnelige dataene.

DataGlyphs kan også inneholde former, logoer og bilder. Nedenfor vises et eksempel hvor en DataGlyph er presentert i form av et bilde. Bildet t.v. har orginal størrelse på 3.3 ganger 3.3 tommer og en oppløsning på 600dpi. Bildet t.h viser nærbilde av det markerte øyet og hvordan gråtonen er presentert i form av DataGlyphs [37].





Figur 27 T.v. DataGlyph presentert i form av et bilde t.v., t.h. nærbilde

4.2 Biometri og stemmegjenkjenning

Biometri og stemmegjenkjenning er to AIDC teknologier som er skapt for å gjenkjenne personer. Biometri innebærer teknologier som fingeravtrykk, irisskanning, ansikts mønster og håndmål. Stemmegjenkjenning er mer selvforklarende, og er utviklet for å kunne gjøre akkurat det. Også signatur og skrivemønster kan nevnes i denne sammenhengen. Dette er alle teknologier som benyttes for å identifisere en bestemt person. De faller derfor utenfor hva vi er ute etter i forbindelse med Logit D2D, håndtering av gods, og vi velger derfor ikke å gå nærmere inn på disse temaene.

4.3 Magnet stripe

Magnetstriper fantes på markedet allerede på -50 og -60 tallet. Men selv om kortet ble brukt bl.a. som billett og betalingsmiddel allerede på denne tiden, var det ikke før på 1970 tallet at bruken skjøt fart. I dag finner man vanligvis magnetstripen presentert i form av kort. Dataene lagres på magnetstripen ved å modifisere magnetismen på små magnetiske partikler på båndet, omtrent på samme måte som på båndet på videokassetter og på disketter.

Bruk av magnetstriper er i dag svært utbredt. De aller fleste bruker en eller annen form for magnetstripe daglig. Kredittkort og tilgangs-/nøkkel-kort er blant de mest utbredte her i landet.

En magnetstripe-leser fungerer ved at et leserhode beveger seg over magnetstripen og registrerer polariteten på magnetpartiklene i magnetbåndet. Polariteten vil oppfattes som henholdsvis ”0” og ”1”, hvor man igjen vi kunne tolke kortets innhold binært. Det finnes to former for lesere. Den ene fungerer ved at kortet dras igjennom et spor som inneholder en leser. Den andre metoden er en leser hvor hele kortet dras inn, leses og returneres.

Figur 28 Illustrasjon hentet fra www.magtek.com

Enhver teknologi har en rekke fordeler og ulemper. Vi vil her presentere en punktliste over generelle fordeler og ulemper i forbindelse med magnetstriper. Senere kommer vi til å kikke på fordeler og ulemper ved bruk i forbindelse med Logit D2D.

Fordeler:

- Data kan bli modifisert og overskrevet
- Stor datakapasitet i forhold til f.eks strekkoder
- Gir sikkerhet ved å være uleselig for mennesker
- Immun mot søl fra skitt, vann, olje, fuktighet osv.
- Ingen bevegelige deler. Robust
- Veletablerte standarder
- Ingen forbruksvarer nødvendig for skriving og omskriving

Ulemper:

- Fungerer ikke på avstand. Krever nær kontakt med leser.
- Data kan bli skadet ved magnetiske omgivelser
- Kan være et problem i noen situasjoner at det ikke er leselig for mennesker

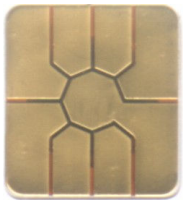
[38]

Magnetstripe er en teknologi som nå har vært på markedet i mange år, og har blitt standardisert under en rekke ISO standarder. (*For nærmere detaljer: [39]*) I de fleste tilfeller befinner magnetstripen seg i en plastikkfilm 5,66mm fra nederste kant av kortet og er 9,52mm bredt. Magnetstripen inneholder tre spor på 2,70mm hver. Spor en og tre er 8,3kbit/m, mens spor to har en datatetthet på 3,0kbit/m. Hvert spor kan inneholde 7-bit alfanumeriske tegn, eller 5-bit numeriske tegn [40].

4.4 Smartkort:

Smartkort ble oppfunnet på 1970 tallet og går også under navnet "Integreted Circuit Card" – (ICC). Dette er derfor en av de nyere AIDC teknologiene i bruk på markedet. Kortet har metall kontaktpunkter som benyttes av en leser ved bruk av kortet. Det finnes i dag tre ulike typer smartkort: Minnekort, Mikroprosessorkort og Optiske kort.

Minnekort
Minnekort benyttes for å lagre og lese data i forbindelse med for eksempel spillkonsoller, telekort, håndholdte PCer osv.



Figur 29 Illustrasjon hentet fra http://www.moneymuseum.com/

Mikroprosessorkort [41]
Mikroprosessorkort kan prosessere data på kortet og kan legge til, slette og manipulere informasjonen. Disse kortene brukes i en rekke løsninger, spesielt de som har kryptering innebygd, noe som krever håndtering og manipulering av store tall. Her er noen eksempler hvor slike kort er i bruk i dag:

- Kort som inneholder penger ("lagret verdi kort")
- Kort som gir sikker tilgang til et nettverk
- Kort som sikrer mobiltelefoner (SIM)
- Kort i forbindelse med TVsendinger. (Parabol, kabel)

Optiske minne kort
"Optical Memory Card" – (OMC) ser rett og slett ut som et kort hvor det er limt en bit av en CD på toppen, og det er omtrent det det er. Optiske minnekort kan lagre opp til 4MB data. Data kan skrives i flere omganger, men det som er skrevet kan ikke endres eller fjernes. OMC kort tilbyr en meget god sikkerhet og tar ca. 100 ganger datakapasiteten til et smartkort. Kortet er meget holdbart mot både søl og magnetisk stråling. Foventet levetid er på over ti år i motsetning til vanlige smartkorts tre år. Disse kortene har ingen prosessor innebygd selv om dette i dag er under utvikling. Kortleserne er enda preget av at dette foreløpig er en lite utbredt teknologi og mangler standardisering. Produksjonsprisen på et OMC kort er omtrent som et chip-kort, men leserne er fortsatt relativt dyre.

Figur 30 OMC er i bruk på over 7 millioner "GreenCards" i USA i dag [41].

Et smartkort kan selv være bærer av sin egen informasjon. En leser vil dermed ikke nødvendigvis være knyttet opp mot en sentral database, noe man må for eksempel med strekkoder. I nyere tid har det også kommet smartkort på markedet som kommuniserer trådløst via radiobølger. Dette kommer vi nærmere inn på i kapittelet RFID.

4.5 RFID:

I de siste årene har RFID [43] (Radio Frequency Indentification) fått mer og mer oppmerksomhet, og mange mener dette vil bli strekkodenes arvtaker. RFID-brikker kan sees på som en avansert variant av strekkoder. Det finnes tre forskjellige typer. Passive, semi-passive og aktive brikker.

- **Passive**:
  De passive har ikke innebygget batteri, og er derfor avhengige av å få strøm fra det inngående signalet for å kunne gi respons. Fordelen med de passive utgavene er at de kan være ekstremt små, siden de ikke har batteri innebygget. Smartcode Corporation [44] har utviklet en brikke på 0.25mm^2(se figur), og er tynnere enn papir. Disse har en praktisk rekkevidde fra ca. 2mm opptil noen meter, avhengig av valgt frekvens. Siden de ikke har batterier, er levetiden i teorien uendelig.
- **Semi-passive:**
  Semi-passive skiller seg fra passive ved at de har et batteri, slik at de ikke behøver å få strøm fra det innkommende signalet, kun til det utgående. De er dessuten raskere enn passive brikker.
- **Aktive:**
  Aktive brikker har en egen strømkilde som forsyner både inngående og utgående sending. De har som regel større rekkevidde og mer minne enn passive brikker, og kan i tillegg lagre informasjon fra senderen. Disse kan være på størrelse med en mynt, har en rekkevidde på titalls meter og har batterilevetid på opptil 10 år.

De fleste brikkene i bruk er passive, av den enkle grunn at de er billigere å produsere og ikke behøver batteri. I store volum koster slike brikker mindre enn 10 cents per stykk. Det var først da Wal-Mart-kjeden og det amerikanske forsvaret satset på RFID at det skjedde et lite gjennombrudd. De var de samme organisasjonene som tidlig satset på strekkkoder i sin tid, og førte til at folk ble mer klar over hvor nyttig dette var. I den senere tid har flere store kjeder tatt i bruk RFID teknologi. NATO [45] vil ta i bruk RFID for å kunne spore militære forsendelser mellom Europa og Afganistan, og denne kontrakten viser at det er et økende behov for RFID-systemer. USAs myndigheter vil trolig øke med 120% mer på det de benytter på RFID de neste fem årene(fra 2005-2009) [46]. Dagens Autopass løsning er basert på RFID, der man ved hjelp av en liten brikke kommer seg fort og enkelt gjennom bompengeringer, uten å måtte stå i kø eller ha noen form for menneskelig kontakt.

Figur 31 RFID-brikke fra Smartcode Corp

Bruksområdene for RFID er omtrent uendelige, de kan for eksempel erstatte strekkoder når det gjelder prising og vareinformasjon, og ikke minst føre til store innsparinger innen logistikk og transport. Fordelen med RFID kontra skanning, manuell opptelling og strekkoder er at det ikke trengs menneskelig berøring. Hvor enn det er menneskelig innblanding, er sjansene store for at ting ikke alltid går som planlagt. RFID avleserne kan lese opptil 96 tagger per sekund og tilbyr derfor en meget effektiv registrering av gods. RFID-tagger kan også inneholde større mengder informasjon som modell, konfigurasjon, versjon, lokalisering og så videre, avhengig av sammenhengen.

Veksten av RFID har blitt hemmet av diverse faktorer [47] og en del ”barnesykdommer” [48]. Radiostøy og interferens med andre systemer er hindringer som må løses. Så langt har kostnadene ved en overgang til RFID teknologi ikke vært hensiktsmessige. Standardisering av blant annet frekvenser og sendestyrke har hjulpet utviklingen mye, da man tidligere bare kunne benytte seg av teknologien i lukkede systemer. Nå kan man bruke RFID på tvers av organisasjoner, noe som gir nye anvendelsesmetoder og er med på å presse prisene ned.

RFID er fortsatt et forholdsvis nytt produkt, men er godt på vei til å erobre markedet på en rekke felter. Etter hvert som teknologien tas i bruk oppdages det stadig nye bruksområder. Blant annet ble RFID benyttet under Vasaloppet for å holde styr på tidtaking til 40.000 skiløpere både ved målgang, og syv mellomtider [49]. Teknologien er nå også brukt i forbindelse med merking/identifisering av dyr. Den internasjonale sammenslutningen av postverk, IPC, har inngått en avtale med IBM Business Consulting Services, som vha. et Internett basert system skal måle kvaliteten og effektiviteten av internasjonale postforsendelser [50]. Blant annet har det svenske [51], finske [52], tyske, britiske og australske [53] postverket tatt i bruk RFID for å få en oversikt over flaskehalser i systemet, og for generell sporing av post. Fordelene med RFID i denne sammenheng kontra strekkoder, er at personell ikke kan se brikkene slik de kan med strekkoder. Undersøkelsene blir derfor mer reelle. Systemet vil automatisk registrere posten når den passerer sentrale punkter i nettverket, og vil på den måten kunne spores hele veien fra avsender til mottaker.

Et annet viktig aspekt er sikkerhet, da disse brikkene kan tenkes å inneholde sensitive data. I og med at brikkene kan kommunisere trådløst blir dette noe helt annet i forhold til tidligere AIDC teknologier. Forskere [54] - [55] og studenter ved John Hopkings University i USA har oppdaget en svakhet med Texas Instruments Registration and Identification System, et RFID system som benyttes over hele verden. De har ved hjelp av moderne datateknologi dekryptert koden i brikkene, og dermed er det fritt frem for kjeltringer som eventuelt klarer det samme. Anslagsvis skal mer enn 150 millioner biler fra bl.a. Ford og Toyota benytte denne teknologien, og i USA brukes den til trådløs betaling av bensin. Utstyret som ble brukt til å knekke koden er kommersielt tilgjengelig, så i prinsippet kan hvem som helst misbruke dette. Med den knekte koden kan man f.eks. simulere bilnøkler og på den måten både låse opp og starte biler. Det bekymringsverdige er at man ikke engang trenger å være i kontakt med personen for å kunne få tak i informasjon nok til å kunne knekke koden. Man trenger nemlig ikke fri sikt til brikken, og de kan avleses fra større avstander.

Man mente tidligere at RFIDs grense på 128 tegn ville eliminere muligheten for virus. Men nå har forskere fra Vrije, Universitetet i Amsterdam, oppdaget og bevist at det er mulig å utvikle virus for RFID brikker. Følgende er deres scenario av hvordan et slikt virus kan plantes.

Scenarioet er som følger: Den kriminelle går inn i et supermarked og velger seg en helt normal dagligvare som er utstyrt med rfid-brikke. Dette kan for eksempel være et glass med syltetøy. Den kriminelle betaler for varen og tar den med seg hjem. Vel hjemme byttes den gamle radiobrikken ut med en ny, som er lastet med destruktiv kode ved hjelp av en ordinær pc og kommersielt tilgjengelig (og billig) utstyr for å skrive til rfid-brikker.

Deretter er det bare å gå tilbake til butikken med syltetøyglasset og betale for den samme varen en gang til. Når varen denne gangen skannes av butikksystemet kan et potensielt virus infiltrere hele supermarkedets produktdatabase og for eksempel endre prissettingen på ulike varer eller gjøre annen skade [56]. Ytterligere informasjon finnes her: [57]

4.6 Lagrede data:

Både strekkode og RFID er et medium for å kunne identifisere en bestemt artikkel og ved hjelp av dette kunne sette den inn i en sammenheng. Dette skjer normalt ved identifikasjonsnummer som kan gjenkjennes etter ulike standarder. En ID uten kunnskap om standard kan fort være like lite verdt som et hvilket som helst annet tall. Det er derfor viktig med globale samarbeid som kan fastsette standarder som kan benyttes til formål innen logistikk og handel. Det er også viktig med et system slik at ikke mange kolli fraktes rundt med samme ID-nummer, noe som raskt kunne ført til full forvirring.

Selv om RFID kan lagre data i større mengder enn strekkode ser det ut til at hovedegenskapen for RFID både innen logistikk og dagligvare vil være det å trådløst tilby en vareID. Den bestemte IDen vil være oppbygd i stor grad på same måte som på strekkoder, men muligheten for større mengder data gir også muligheten for diverse kontrollsiffer, tilleggsdata o.l. I mange systemer benyttes RFID i systemer som er utviklet for strekkode. For å skille RFID IDene og strekkode IDene i systemet er det da vanlig å sette ”00” foran RFID Ider.

Innholdet i disse IDene varierer fra standard til standard. Noe av det som gjentar seg er bestemte siffer som tilsvarer land, produsent og produktserie. For å kunne benytte seg av disse standardene må man dele opp de ulike kodene etter sin fastsatte standard, for så å benytte de oppdelte dataene til å identifisere deres betydning. Et eksempel på dette er for eksempel at hvis de tre første siffrene i UPC kodede strekkoder er mellom 700 og 709, så betyr det at varen er fra Norge. Dette resultatet får man ved å vite at strekkoden er en UPC kode, for så å sjekke opp de tre første sifrene mot EAN13 prefiks listen.

Disse dataene kan være nyttige på flere måter, da de for eksempel kan være med på å bidra til å opplyse om et produkts opprinnelse. De kan også benyttes til statistikk om gods og produkter. Det er da viktig for systemene som skal håndtere disse, at de vet hvilke standarder som er i bruk. Ofte benyttes IDene i systemene kun som en tall-ID. I slike forbindelser har IDen ingen betydning før den benyttes mot en database i f.eks et logistikksystem som kan opplyse om godsets innhold, opprinnelse og destinasjon. I de følgende delkapitlene skal vi kikke nærmere på standarder rundt strekkode og RFID.

4.6.1 Data i strekkoder

Transportbransjen: Alle de store transportørene bruker GS1 systemet til å strekkode merke fraktbrev og transportkolli [58]. Opprinnelig ble det utviklet til identifisering av produkter og pakningsvarianter(GTIN 8 og 13), men etterhvert oppstod det behov om å registrere mer informasjon om forsendelsene. GS1-128 [59] ble utviklet som et resultat av dette. Dette er den mest vanlige standarden som brukes i denne sektoren i dag. Det er også vanlig med en ”Felles transportetikett” [60] som plasseres på hvert enkelt kolli av avsender.

Figur 32 Tranportetikett som blant annet Nor-Cargo bruker [59]

Dagligvarebransjen: Var de første til å ta i bruk GS1 [61] standarden her til lands, tidlig på 1980-tallet. Alle produkter i detaljistleddet må merkes med EAN artikkelnummer [62]. De dominerende matvarekjedene i Norge krever dessuten at produktene merkes etter ”Standard for merking av D-pak og pall i dagligvarebransjen” konseptet [63]. De samme kjedene krever også utstrakt bruk av elektroniske meldinger definert i DEDIP2 [64].

Byggvarer: Produktene som selges over disk er merket med EAN artikkelnummer. Elektroniske meldinger brukes også, og da særlig elektronisk faktura [65]. Ved bruk av elektroniske meldinger brukes EAN for å identifisere kjøper, selger, adresser etc.

Elektro: Alle varer i Elnummerbanken skal nummereres med EAN artikkelnummer.

Energibransjen: Energibedriftenes Landforening og GS1 har inngått en avtale om bruk av Global Service Relation Number (GSRN) [66]

4.6.1 Data i RFID

Det GS1 [67] og GS1 US sameide selskapet EPCglobal Inc.[68] er skapt for å bidra til en global spredning av RFID. De har begynt arbeidet på standardiseringsjobben av RFID under navnet Electronic Product Code (EPC). Dette skjer på lignende fremgangsmåter som GS1 tidligere har skapt standarder for strekkoder. EPCglobal mener de i dag er den eneste standardiseringsorganisasjonen når det gjelder RFID.

Skal RFID ha noen verdi må det kunne benyttes av alle de ulike partene i en forsyningskjede. For at det skal la seg gjøre behøves en standard. Det er viktig at en standard utvikles for at RFID skal bevege seg ut av laboratoriene og inn i den virkelige verden. EPCglobals mål er å utvikle standarder som fyller firmaenes behov.

Figur 33 GS1 EPCglobal logo

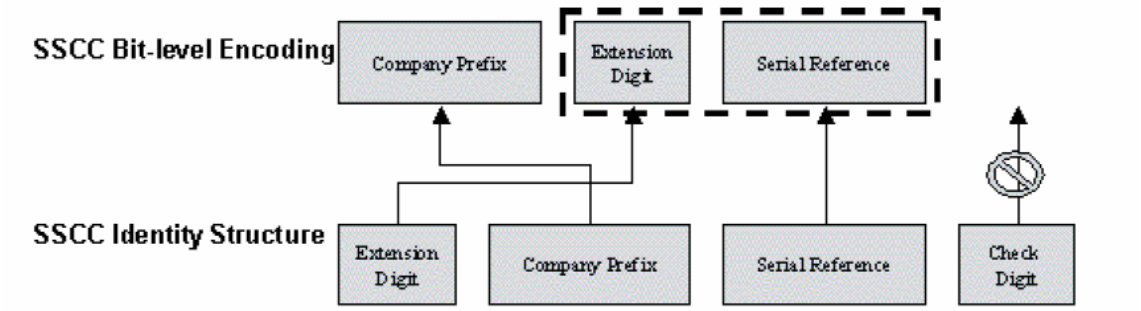
"Standards and specifications provide the common definitions, functionality and language for the hardware and software components of the EPCglobal Network. They help advance the EPCglobal community toward a common objective, namely, implementing the EPCglobal Network™ to improve visibility and efficiency in today's global, multi-industry supply chain" [69]

EPC bygges opp på samme måte som dagens GS1-nummer (EAN-nummer), men med et serienummer som adderes i tillegg. EPC nummerene skal benyttes for å gi varene unike identifikasjonsnumre i transportkjeden. For at brukerne skal være klar over at deres produkter er merket med denne type teknologi, skal de presenteres for en av to EPC logoer. Disse skal benyttes på alle produkter som benytter denne type teknologi.

Figur 34 EPC logo skal plasseres på produkt

- EPCglobal Spesifikasjoner/ikke fastsatte Standarder [70]
- EPCglobal fastsatte standarder [71]

Under de fastsatte standardene finner man blant annet "EPC Tag Data Standard Version 1.1 rev 1.27" [72] hvor man omtaler blant annet SGTIN (Serialized Global Trade Item Number), SSCC (Serial Shipping Container Code) og SGLN (Serialized Global Location Number). Vi ser nærmere på SSCC standarden i følgende modell.



Figur 35 " How the parts of the decimal are extracted and rearranged for encoding." [72]

Kinesiske myndigheter og industri arbeider med å utvikle sin egen standard for RFID. I denne forbindelse har det blitt uttalt følgende: "*China will use EPCglobal and ISO standards, but with some modifications to satisfy special needs in China* [73]." Kina ønsker likevel å bidra til å utvikle en internasjonal standard, hvor også deres egen standard vil være kompatibel. Men de ønsker å ha sin egen for å slippe eventuelle avgifter ved bruk av en internasjonal standard.

Selv om RFID ikke er fullstendig standardisert, så finnes det i dag en rekke retningslinjer for hvordan det bør benyttes. RFID er i dag presentert i ISO Standardene: ISO 11784, 11785, 14223/1, 10536, 1444, 15693 og 18000.

4.7 Sammenligning og konklusjon:

Vi har nå sett nærmere på de ulike AIDC teknologiene. I forbindelse med LogIT Systemsog deres produkt er det nok en rekke som kan direkte utelukkes. Når det gjelder fremtiden så er det aldri godt å vite hva som dukker opp av nye teknologier eller hvilke eldre teknologier som kan nyvinnes eller implementeres på nye måter. Vi må likevel forholde oss til det som eksisterer i dag og trekke konklusjoner ut fra dette.

Følgende er en tabell med en sammenligning slik at man, på en enkel og oversiktlig måte, kan se hva de ulike teknologiene kan tilby.

Tabell 7 Sammenligning mellom de omtalte identifikasjons medium

Teknologi	Kan lese uten fysisk kontakt	Kan leses uten "line of sight"	Gir mulighet for dynamiske data	Sikker mot virus og manipulering
Strekkode	JA	NEI	NEI	JA
Biometri	JA	NEI	NEI	JA
Stemmegjenkjenning	JA	JA	NEI	JA
Magnetstripe	NEI	NEI	JA	NEI
Smartkort	NEI	NEI	JA	NEI
RFID	JA	JA	JA	NEI

Det er mange spennende teknologier på markedet i dag. Mange av dem ville nok også kunne utføre oppgaven ved å merke gods, men den praktiske delen rundt både skriving og lesing er avgjørende. Noen av teknologiene kan utelukkes øyeblikkelig. For eksempel biometri er basert på menneskelige trekk og er derfor ikke aktuell i forbindelse med transport av gods. For stemmegjenkjenning gjelder det samme.

Når det gjelder smartkort vil de absolutt være egnede for å lagre de nødvendige data. Men lesing vil være tungvindt da dette vil kreve fysisk kontakt. Det kreves også at denne kontakten er nøyaktig for at en leser skal berøre de riktige kontaktpunkter. Dette vil være svært tungvindt i en stressende sammenheng under lasting og lossing av gods. Vi ser derfor også denne teknologien som uaktuell.

Magnetstripe stiller ikke så mye sterkere enn smartkort, men den har en viktig fordel. Innlesing krever mye lavere nøyaktighet. Her kan lesehodet benyttes i fart og kun dras over stripen for innlesing av data. Teknologien er også meget billig og brukes i dag på tog og fly billetter. Men selv om innlesingen er enklere enn smartkort er den likevel underlegen strekkoder. Magnetstriper krever fysisk kontakt med lesehodet ved lesing. Lesehastigheten er også viktig da man må bevege leseren med en bestemt fart for å utføre lesing. Bevegelsen må også foregå i riktig retning. Et siste problem er også avmagnetisering. Som beskrevet tidligere består de lagrede data av positivt og negativt ladde partikler. Ved transport i lasterom og containere vil gods bevege seg blant mye metall. Dette kan føre til avmagnetisering og skade av data på magnetstripen. Hva om man skal laste en pall med magneter? Slike problemer vanskeliggjør magnetstripens muligheter innen merking av gods.

Vi sitter nå igjen med de to teknologiene strekkoder og RFID. Det finnes i dag en rekke varianter av dem begge. Klassisk strekkode er i dag mest utbredt innen logistikk og er også meget godt egnet. Dessverre inneholder den en del begrensninger. Nyere varianter gir nå mulighet for lagring av større datamengder og beskyttelse mot skader. Et problem er likevel at også nye strekkoder må leses på samme måte som de eldre. Her kommer RFID på banen. RFID sine muligheter for innlesing kan nok føre til at teknologien vil ta over mye av markedet etter hvert.

Konklusjonen er derfor at dagens og fremtidens logistikksystemer vil være basert på de to teknologiene strekkode og RFID. Om RFID noen gang skal ta helt over vil det ta tid, men mye tyder på at strekkode også vil beholde sin andel av markedet. Muligens vil strekkode og RFID operere side om side fremover, eller kanskje det dukker opp nye og bedre teknologier.

Vil noen gang de nye teknologiene kunne ta over for strekkoden? Det er ikke til å se bort fra at begge teknologier har sine fordeler. Følgende sitat presenterer klart en mening om at både strekkode og nyere intelligente AIDC teknologier har sine områder å betjene, og at ikke nødvendigvis er noen stor trussel mot hverandre.

> Dagens konkurranse mellom intelligente merkelapper og strekkoder kan sammenlignes med en konkurranse mellom en ubåt og en sykkel *[28, Bert More, Juni 1999]*. Ubåten er ingen stor trussel på tørt land, og sykkelen blir ikke spesielt brukbar i vannet.
>
> [29]

5 Valgt utstyr og teknologi

Etter nøye vurderinger følte vi oss til slutt trygge på valg av utstyr og teknologier. Valgene ble fastsatt, utstyr kunne bestilles og arbeidet kunne begynne. Vi skal her ta en liten titt og bli kjent med det utstyr som ble valgt og blir benyttet videre i rapporten.

5.1 Mobil enhet

Jo mer vi vurderte og kikket på data og spesifikasjoner til de ulike modellene, ble M3 fra Mobile Compia [13] en klarere og klarere favoritt. Denne fremstilles fra den norske leverandøren Strekmenn AS [74] som ”Terminalen som har alt!”, og i vårt tilfelle så er den også det. M3 leveres med en rekke innebygde funksjoner. Man kan også spesialbestille med eller uten noen av alternativene, samt at man har en rekke tilkoblingsmuligheter mot ytterligere utstyr.

1. M3 har satset på et stort display. Dette gjør utviklingen av et brukervennlig og godt verktøy enklere.

2-3. Ekstra funksjonstaster. Disse kan programmeres etter ønske. Gir mange muligheter. Orginalt var to av de brukt til kamera og strekkodeleser. Dette kan imidlertid overstyres om ønskelig.

4. Tilkoblings-mulighet til base for ladning og tilkobling med active sync.

5. Høyttaler. Praktisk plassert med tanke på telefonbruk.

Figur 36 M3 fra Mobile Compia

6. Strekkodeleser plassert slik at man kan utføre lesing og samtidig ha oversikt over resultatet på skjermen. Praktisk plassert sammen med "håndfeste" på baksiden og de fire hurtigknappene 2 og 3 på sidene.

7. Kamera er plassert foran på enheten. Føles noe uvanlig ved fotografering, men er en praktisk plassering i forhold til skjerm og PDAens plassering i håndflaten.

8. CF-slot for tilkobling av ytterligere elementer.

9. 3x4 taster lagt opp på samme måte som de fleste mobiltelefoner. Hovedsakelig for å presentere tallene fra 0-9, men gir også muligheten for inntasting av tekst ved at tre bokstaver er presentert på hver tast.

10. Navigasjonstast gir mulighet for bevegelse i alle fire retninger.

11. En samling av seks funksjonstaster. Første tast aktiverer telefonfunksjonalitet. Andre tast bytter mellom tall- og bokstavmodus på tastene(9.). Tredje tast er backspace. De to siste knappene er enter og av/på.

Utover hva som er mulig å presentere på bildet så tilbyr også M3en innebygget WLAN og mulighet for Bluetooth, GPRS, mobiltelefoni, SMS og GPS. Alt dette åpner for en rekke fremtidige muligheter og dekker på en kompakt og praktisk måte de behov vi har i dag. På neste side har vi tatt med en liste over de viktigste data Mobile Compia presenterer i forbindelse med sin M3.

Figur 37 M3 fra Mobile Compia

Spesifikasjoner for Mobile Compia M3:

- OS Windows CE.NET 4.2
- CPU Intel Xscale PXA-255(400MHz)
- ROM FlashROM 64MB (Expandable up to 256MB)
- RAM SDRAM 64MB
- LCD 240 x 320, 3.5¡±, Color TFT LCD(65KColor) Front Light LED
- Dimension 78.6 X 160 x 24mm
- Weight 250g
- I/O USB/Serial , IrDA, Microphone, Speaker, Stereo Ear-mic Jack
- Battery Lithium Polymer Rechargeable/Removable 2000mA
- Anti-shock 1.5m
- Workable degree -20C ~ 50C
- Headset Speaker, Mic, Call Key
- Adaptor 1) HOME (input : AC 220V , output : DC 4.2V 880mA)
- Adaptor 2) CAR (input : DC 9~30V, output : DC 4.2V 800mA)
- Skanner 2D Laser Type (Opticon: 38400bps serial)
- WWAN CDMA 2000 1x wireless module
- Uart 2 + 1
- Camera CMOS Type 640x480 (USB or SPI)
- Camera 1 PC - 4 Slot USB
- KEY 3 x 4 dial, 4direction, 7 function (5front,2side), 1 Power

For mer detaljerte spesifikasjoner se norsk presentasjonsark fra Strekmenn AS:
http://www.strekmenn.no/default.asp?WCI=file&WCE=234

For mer detaljert informasjon besøk Mobile Compia sine websider på følgende link:
http://www.mobilecompia.co.kr/en/

5.2 AIDC leser

Ved valg av mobil enhet var vi heldige å ende opp med en med innebygget strekkodeleser. På denne måten og i forbindelse med at strekkoder var oppdragsgivers hovedprioritet, ble ikke lenger valg av AIDC leser et problem. Med M3en har vi strekkodeleseren på plass, og vi vet at det finnes muligheter for RFID leser og tilkoblingmuligheter for diverse andre leserenheter.

5.3 Kommunikasjon

For utvikling har kommunikasjon via WLAN blitt prioritert. Dette var en teknologi M3en hadde innebygd ved levering, noe Bluetooth og GPRS også var. M3en hadde derfor alle de kommunikasjonsteknologiene vi hadde vurdert å bruke. WLAN blir benyttet under utviklingen, videre får vi se om tiden tillater oss å implementere GPRS kommunikasjon.

5.4 Server

I og med at dette kun er ment for utvikling og testing av en håndholdt enhet stilles det ikke spesielt store krav til server. Vi benytter derfor en eldre maskin som utviklingsserver. Denne er satt opp med følgende spesifikasjoner

- Compaq Presario
- Microsoft Windows 2000 5.00.2195
- Servicepack 3
- Java 2 Runtime Environment
- JBoss 4.0.2
- MS SQL Server
- Logit D2D server
- LogMeIn Remote Desktop
- FileZilla FTP Server [75]
- Internet tilknytning

5.5 Operativsystem

Valg av OS ble tatt i forbindelse med valg av enhet, da Mobile Compia leverer M3 med Windows CE .NET 4.2. Videre var det likevel flere mulighet for programmering, og valgene stod mellom JAVA og .NET plattformen.

I og med at Mobile Compia leverer et SDK med drivere og eksempler for C++ og C#, ble det fort avgjort at Java ikke var akuelt. C++ ble valgt bort fordi vi har mye mer erfaring med C# både gjennom vår utdannelse og privat. SDK'et kom som nevnt med eksempler på hvordan man kunne styre WLAN kortet, kameraet, strekkodeskanneren, telefonen med tilhørende SMS og GPRS funksjonalitet og Bluetooth. Dette var et meget stort pluss, da vi ikke trengte å bekymre oss for at ting skulle være inkompatible eller at vi skulle måtte bruke ekstremt mye av utviklingstiden på å få de forskjellige enhetene til å kommunisere med M3en.

Visual Studio .NET har Windows CE .NET 4.2 emulator og OSet har støtte for bruk av OpenNetCF rammeverket. Dette gir ytterligere funksjonalitet utover det offisielle rammeverket fra Microsoft.

Ingen av gruppemedlemmene hadde tidligere erfaring med programmering for Windows CE .NET 4.2 og vi visste straks at dette kom til å bli en spennende og lærerik oppgave.

6 Utvikling

Planleggingen var tidkrevende, men av erfaring visste vi at utviklingen ville kreve enda mer. Etter planleggingen kom vi derfor i gang med å programmere så raskt som mulig. Tabellene på denne siden viser hvilke elementer i systemet vi skulle arbeide med.

Vi har måttet forholde oss til en server side og en applikasjonsdel. Selv om server siden stort sett var ferdig satt opp, så var det også her noen elementer vi måtte ta hånd om. På applikasjonsdelen kunne arbeidet deles opp i en rekke mindre oppgaver. Vi har på illustrasjonen gruppert dette i fire hovedgrupper: Design, kommunikasjon, applikasjon og datahåndtering. Disse gruppene inneholder igjen de mer spesifikke utfordringene vi har hatt. Stikkordene forklarer hva temaet dreier seg om, etterfulgt av en kapittelindeks.

Tabell 8 Tabell over utviklingselementer på server

Serverkommunikasjon	
Webservice .NET 1.1 > .Net 2.0	kap 6.1.1
FTP Server FileZilla	kap 6.1.2

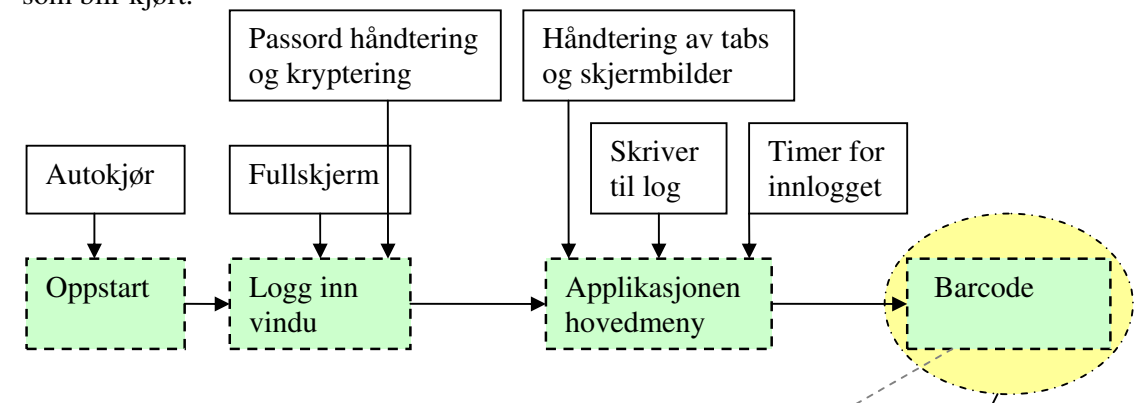
Vi har forsøkt å fordele på samme måte som tabellen illustrerer. Ved å dele oppgaven i en rekke mindre utfordringer har det vært enklere å fokusere på arbeid og fremgang. Oppgavene har vi fordelt mellom oss litt etter hvilke kompetanse og interesse vi hadde innenfor de ulike temaene. På denne måten har vi sikret at vi jobber med noe vi føler er interessant, og at vi hele tiden presterer et best mulig resultat. Mot slutten av prosjektperioden har vi flettet sammen de ulike elementene av applikasjonen.

Tabell 9 Tabell over utviklingselementene i applikasjonen

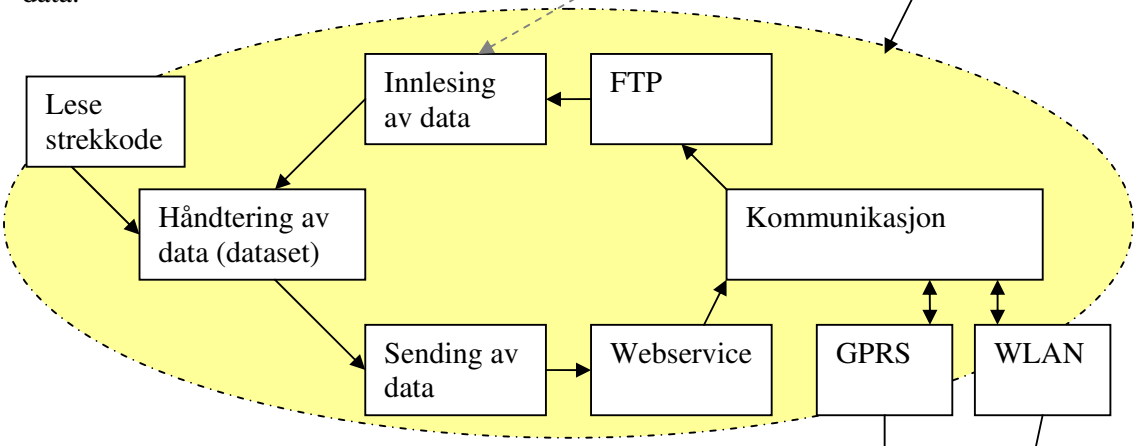
Design		Kommunikasjon	
Utseende	kap.6.3.1	WLAN	kap.6.2.11
Vinduer	kap.6.3.2	Webservice	kap.6.2.8
Menystruktur	kap.6.3.3	FTP	kap.6.2.7
Usability	kap.6.3.4	Telefoni	kap.6.2.5
Applikasjon		**Datahåndtering**	
Autokjør	kap.6.2.1	Skanner	kap.6.2.11
Fullskjerm	kap.6.2.2	XML Håndtering	kap.6.2.6
Passord	kap.6.2.4	Dataset	kap.6.2.6
Batteristatus	kap.6.2.3		

Vi vil nå, i de følgende kapitlene, gå igjennom de ulike temaene for å komme nærmere inn på hvordan vi har løst dem.

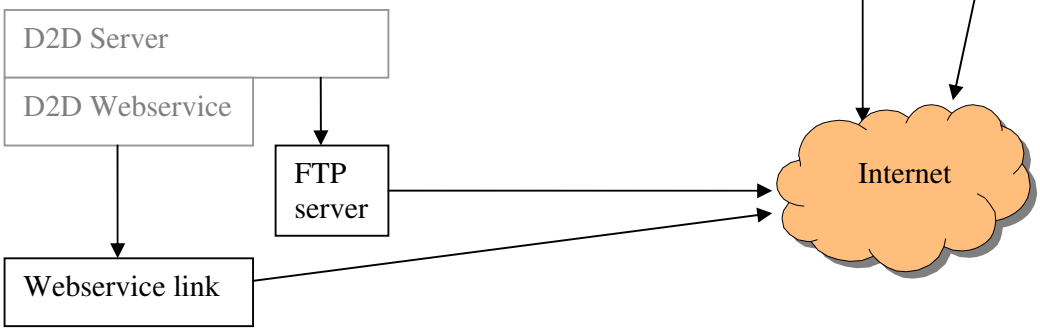
Vi har her presentert de ulike elementene applikasjonen består av. De grønne feltene er en illustrasjon av gangen i applikasjonen, mens de hvite viser viktige utviklede elementer som blir kjørt.



Vi ser her nærmere på de ulike elmentene som er utviklet for strekkodelesing og håndtering av data.



Til slutt tar vi med det som har blitt gjort på serversiden hvor en webservice link er utviklet og en FTP server er satt opp. De grå feltene illustrerer D2D serverløsningen som benyttes på serveren.



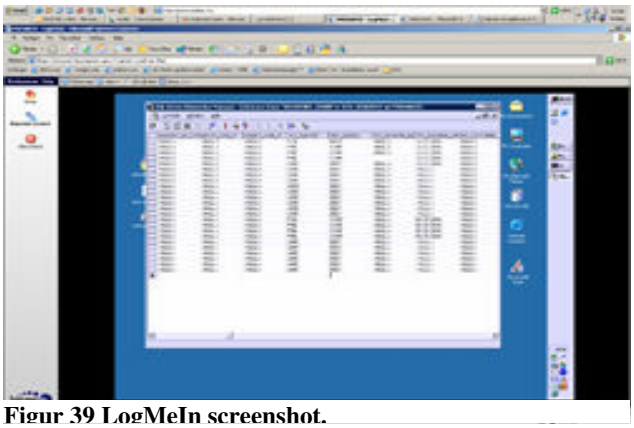
Figur 38 Kart over de viktigste elementene som applikasjonen består av

6.1 Server installasjon og oppsett

Under arbeidet har vi benyttet oss av en helt vanlig laptop som vi i samarbeid med LogIT Systems har satt opp som en D2D testserver. Utover deres standard oppsett har vi måttet gjøre noen tilpasninger for å kunne gjøre kommunikasjon med den mobile enheten mulig. Kommunikasjonen mellom mobil enhet og server skal foregå i to ulike tilfeller:

- Henting av oppdrag som innebærer automatisk
  - Oppkobling med FTP
  - Nedlasting av valgt XML fil
- Innsending av data som innebærer
  - Oppkobling mot webservice
  - Overføring av XML data

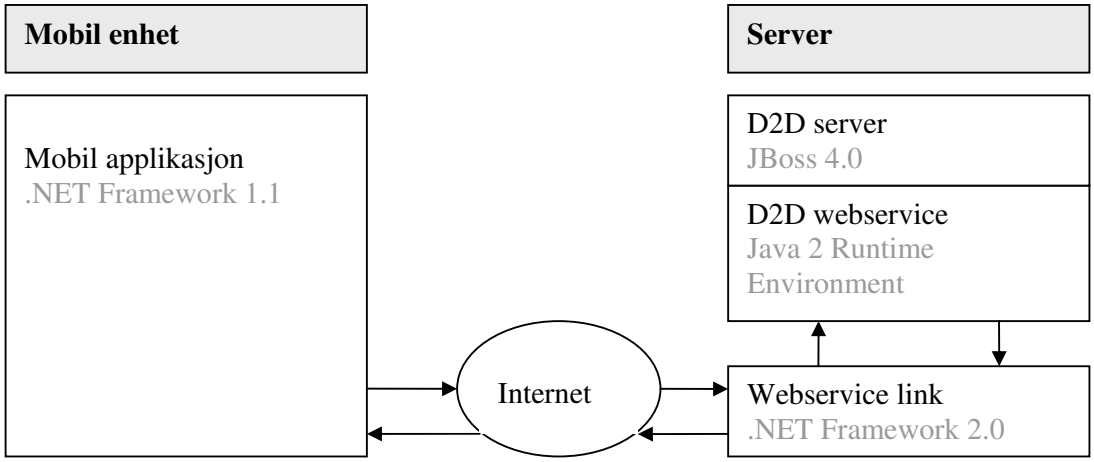
Serveren var i utgangspunktet satt opp med Microsoft Windows 2000 Professional og Microsoft SQL Server. Videre ble det benyttet JBoss 4.0 serverløsning og Logit D2D sin javabaserte applikasjon. For enkel og effektiv tilgang benyttet vi en Remote Desktop løsning. Serveren vi fikk utdelt av LogIT Systems til prosjektperioden var en Laptop med delvis defekt skjerm, satt opp som server. For å få et fullstendig skjermbilde og for at begge enkelt skulle ha tilgang til serveren under utviklingen installerte vi LogMeIn [76]. LogMeIn er en remote desktop løsning som tilbyr en gratis lettversjon. Denne krever kun en installasjon på webserver, og gir brukerne tilgang til maskinen via et webgrensesnitt. Dette har vært en meget praktisk løsning for oss under utviklingsperioden, og det har også gitt enkel tilgang til server for oppdragsgiver i de tilfeller det har vært nødvendig.



Figur 39 LogMeIn screenshot.

6.1.1 WebService link

I kapittel ”3.2 Kommunikasjonsstrukturen” avskrev vi behovet for en front end server. Dette behovet viste seg imidlertid å være til stede. Vi oppdaget tidlig at det oppstod et problem ved forsøk på kommunikasjon mellom applikasjonen basert på .NET Framework 1.1.4322 og den javabaserte webservicen på serversiden. .NET 1.1 klarte ikke å håndtere webservicen riktig og overføring av data lot seg da heller ikke gjøre. Etter nærmere utforskning av problemet kom vi frem til at dette problemet ikke var tilfellet ved applikasjoner basert på .NET Framework 2.0. Dette alene løste ikke problemet da Windows CE 4.2 platformen kun støttet .NET Framework 1.1. Problemet ble diskutert med oppdragsgiver, og vi kom frem til å utvikle en midlertidig løsning med en webservice link, en såkalt ”front end”. Avgjørelsen ble basert på antakelsen om at M3en sannsynligvis etter hvert vil komme med Windows CE 5.0 og .NET Framework 2.0, og at problemet da vil forsvinne.



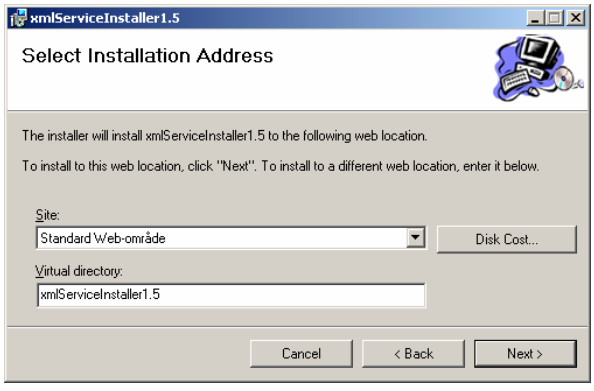
Figur 40 Webservicens posisjon i strukturen

For å omgå problemene som oppstod mellom deres javabaserte webservice og .NET 1.1 applikasjonen bestemte vi oss for å opprette en webservice som et mellomledd. Denne skal være basert på .NET 2.0, og kan derfor kommunisere både med den javabaserte D2D webservicen og .NET 1.1 applikasjoner. Webservicen skal ligge på samme serveren som den opprinnelige webservicen, og skal kun fungere som en forwarder. Det vil si at den kun skal videresende de data den mottar til den opprinnelige webservicen. Den vil så returnere om sendingen ble vellykket eller mislykket og eventuelle feilmeldinger.

For å kunne kjøre denne servicen kreves Internet Information Service, IIS, som følger med Windows versjonen. Denne var allerede installert på serveren og benyttet port 80. I og med at JBoss serveren benyttet port 8088 skapte det ingen komplikasjoner å kjøre begge samtidig. Videre lastet vi ned og installerte nyeste versjon av .Net Framwork 2, 2.0.50707, fra Microsoft sine websider [77].

Programmeringen av webservicen besto av få linjer kode og var grei å få på plass. Vi valgte å legge til en try/catch test på overføringen for å fange opp eventuelle feil. Responsmeldingen avhenger av disse og vil opplyse brukerapplikasjonen om sendingen har blitt utført korrekt eller ikke.

Selve servicen ble opprettet som en ASP.NET Web Service i Microsoft Visual Studio .NET på localhost på utviklermaskinen. For å sette opp servicen korrekt på serveren valgte vi å lage en Microsoft Installasjonsfil, *.msi, som kunne kjøres serveren. For å lage denne filen opprettet vi et nytt prosjekt i Visual Studio av typen Web Setup Project. Her la vi til webservicen og genererte installasjonsfila.



Figur 41 Installasjonsvindu for webserviceen

Vi overførte så installasjonsfila, XMLServiceInstaller1.5.msi, til serveren.

Figur 42 Innstallasjonsfil for Webservice

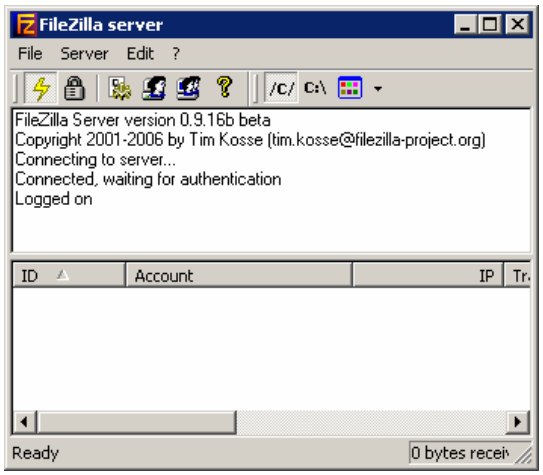
Ved å kjøre denne ble det opprettet en virtuell mappe, Virtual directory, på IIS og riktige filer ble lagt til slik at servicen ble kjørbar.

Servicen var nå klar til å benyttes. Ved å kalle denne servicen, istedenfor den opprinnelige webservicen, vil nå den samme jobben bli utført. Brukeren som programmerer mot servicen vil ikke merke noen forskjell.

For testing under utvikling har vi også laget en liten applikasjon for sending av XML data. Denne ligger med på CDen for vedlegg. [v1]

6.1.2 FTP Server

Når en bruker skal hente et oppdrag hentes den for øyeblikket fra D2D systemet som en XML fil som kalles PB(Provider Booking). Disse filene genereres og plasseres i en bestemt mappe automatisk i D2D serveren når oppdragene blir skapt. For å gi den håndholdte enheten tilgang til disse filene valgte vi å benytte oss av FTP. På serversiden måtte vi derfor kjøre en FTP server. Her benyttet vi FileZilla [75] som tilbyr gratis FTP server og klient software. FileZilla server versjon ble installert på serveren. Her var det viktig at ingenting kolliderte med D2D serversystemet. Ved å benytte standard FTP porter for kommunikasjon og dataoverføring, 20 og 21, oppsto det ingen komplikasjoner. Serveren ble satt opp med brukernavn og passord og riktig mappe med PB ble gjort tilgjengelig ved hjelp av serveren.



Figur 43 Skjermbilde fra FileZilla FTP server

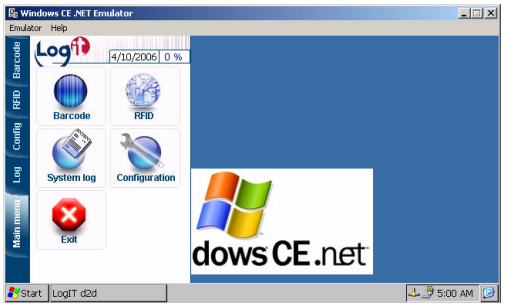
I en ferdig løsning vil ikke brukeren skulle hente PB meldingen fra D2D serveren, men fra tjenestetilbyderens(Provider) eget system. Serveren simulerer egentlig tjenestetilbyderens system som Logit D2D kommuniserer med via web services. Henting av provider booking filen man benytter for registrering av gods må derfor tilpasses tjenestetilbyderens system.

For henting av oppdrag må brukeren velge dette i applikasjonen på den håndholdte enheten, for at denne skal bli oppdatert med de for øyeblikket aktuelle oppdragene. I og med at henting av oppdrag var en oppgave utover oppgavedefinisjonen, og at det trolig er en midlertidig løsning, har vi gjort det på denne måten. Å benytte FTP til denne kommunikasjonen er nok ikke den optimale løsningen, men fungerer fint som et ledd i utviklingsfasen. Løsningen fungerer og tilfredsstillende for en prototyp for demonstrasjon av applikasjonens muligheter. Kommunikasjonsdelen er lagt i en egen klasse som man enkelt kan endre for å tilpasse fremtidige ønsker for kommunikasjon.

6.2 Softwareløsning

Utviklingen av softwareløsningen har foregått i Microsoft Visual Studio .NET 2003 hvor vi har utviklet en C#, OpenNETCF applikasjon. Ved å benytte et såpass kraftig utviklingsverktøy har vi hatt muligheten til å komme frem til det resultatet vi har i dag. Under

utviklingen har vi benyttet oss av en Windows CE .NET Emulator som har gjort at vi har kunnet kompilere og kjøre applikasjonen under arbeid på selve utviklings PCen. Dette er enklere og mindre tidkrevende enn å måtte overføre applikasjonen til den mobile enheten for hver eneste test av koden. Emulatoren gir også bedre tilbakemelding om eventuelle feil under kjøring av applikasjonen. En ulempe er at selv om koden som kjøres på emulator i utgangspunktet skal fungere på samme måte på den mobile enheten, er ikke dette alltid tilfellet.



Figur 44 Skjermbilde fra utvikling i emulator

Som nevnt i innledningen av kapittel 6, så har arbeidet vært fordelt i en rekke mindre elementer. Vi skal her gå nærmere i detalj rundt utviklingen og programmeringen av disse elementene.

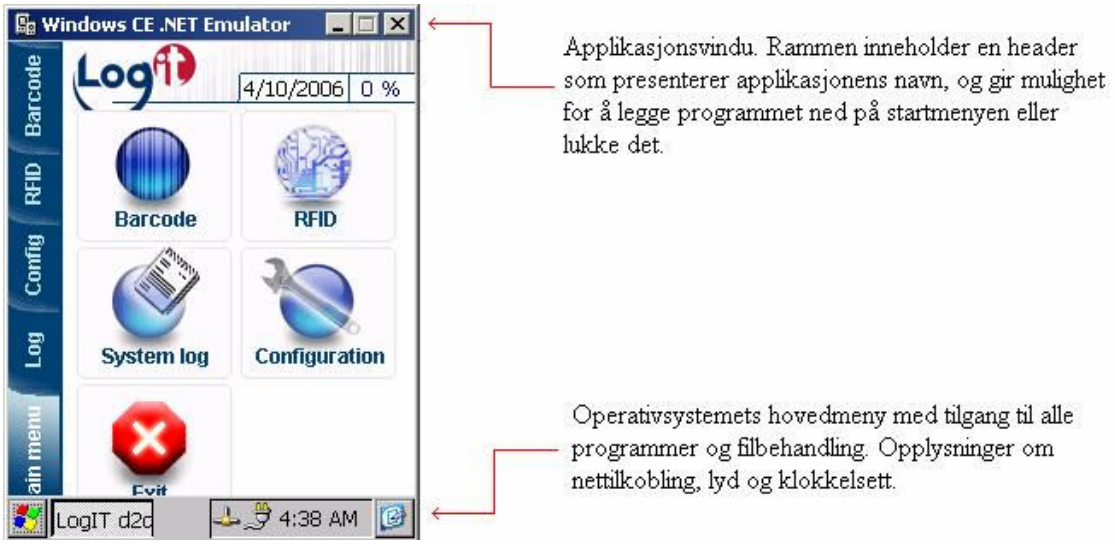
6.2.1 Autokjør

En ting som måtte tas hensyn til var oppstart av applikasjonen. Hvis PDA går tom for strøm eller slå seg av på annet vis. Når så PDAen slås på igjen, så må applikasjonen starte automatisk. PDAen er i vårt tilfelle ikke ment for annen bruk. Derfor bør applikasjonen starte automatisk med en gang, og ikke gi bruker mulighet for å gjøre noe annet på den.

Vi jobbet en del for å finne en måte å løse dette problemet på fra selve applikasjonen. Etter en del arbeid dukket det plutselig opp en lys idé. Først opprettet vi en snarvei til applikasjonsfilen. Så plasserte vi denne i oppstartsmappen for Windows. På denne måten blir applikasjonen startet automatisk når PDA slås på.

6.2.2 Fullskjerm

Vi ønsket å fjerne startmeny og applikasjonsramme for å gi vår applikasjon hele skjermbildet.



Figur 45 Hovedmenyene som viser på topp og bunn av en applikasjon ble fjernet.

Applikasjonen er ment i bruk av personer som ofte stiller med lite, eller kanskje ingen datakunnskaper. Den må derfor være enkel å forstå, enkel å bruke, og tilnærmet umulig å gjøre feil. Der støtte vi tidlig på et problem, da vanlige applikasjoner i Windows CE blir vist med en startmeny nederst i bildet og en meny øverst. Her finnes det knapper som en person fort kan komme borti, bevisst eller ikke. Og kan føre til at applikasjonen stopper, forsvinner fra skjermbildet eller lignende. Dette måtte vi forhindre at skulle kunne skje.

Det første vi kikket på her var å få applikasjonen til å dekke hele skjermbildet. På denne måten vil brukere kun ha tilgang til de knapper vi ønsker. Dette vil forhindre at noe kan gjøres galt. Selve det å fylle hele bildet var forholdsvis enkelt. Ved hjelp av noen få setning i klassens Load hendelse kunne man maksimere applikasjonen til å fylle ut hele skjermbildet.

```
// Del av formens load funksjon som maksimerer størrelsen
private void LogITd2d_Load(object sender, System.EventArgs e)
            {
                  fs = new FS();
                  fs.InitFullScreen();
                  fs.DoFullScreen(true);
                  this.WindowState = FormWindowState.Maximized;
                  this.FormBorderStyle = FormBorderStyle.None;
                  this.ControlBox = false;
                  this.Menu = null;

// Innstillingene nullstilles når formen lukkes
private void LogITd2d_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
            {
                  this.WindowState = FormWindowState.Normal;
                  this.FormBorderStyle = FormBorderStyle.FixedSingle;
                  this.ControlBox = true;
                  this.Menu = mm;
                  fs.DoFullScreen(false);
}
```

Figur 46 Kodeeksempel: Kode for å maksimere en form til å fylle skjermbildet.

Denne koden måtte legges inn i Load hendelse på hver nye form som skal kjøres i fullskjerm. Formen må også gjøres større, i og med at menyen på topp og bunn forsvant. På denne måten ble det også mer plass i skjermbildet.

Men da vi trodde at vi hadde kommet til en fin løsning, støtte vi på et forholdsvis stort problem. Da vi fjernet menylinjene fra applikasjonen vår fjernet vi også tilgangen til det virtuelle tastaturet. Dette behøves for å bruker skal kunne skrive i applikasjonen, og er meget nødvendig i en del tilfeller. Vi prøvde å løse dette problemet på ulike måter. Etter å ha utforsket litt nærmere hva som kunne la seg gjøre, fant vi en interessant løsning på Internet.

Ved hjelp av et ferdigutviklet klasse [78] som man kunne legge til i programmet sitt, så kunne man kalle opp tastaturet etter ønske. På denne måten kunne vi legge til events på `GotFocus` og `LostFocus` i de ulike tekstfeltene. Det vil si at når markøren kom i en tekstboks ble metoden `GotFocus` kjørt, og `LostFocus` kjørt når den forsvant. `LostFocus` ble også kjørt ved `Click` hendelsen på panelet, slik at man kunne trykke hvor som helst ellers i skjermbildet for å få bort tastaturet.

```
private void tbBarcode_GotFocus(object sender, System.EventArgs e)
{
      Sip.Show();
}

private void tbBarcode_LostFocus(object sender, System.EventArgs e)
{
      Sip.Hide();
}
```

Figur 47 Kodeeksempel: De 2 funksjonen som viser og skjuler tastatur når man manuelt skal taste inn strekkoder

Da det virtuelle tastaturet var på plass viste det seg at også dette skulle by på et lite problem. Tastaturet dekker nemlig ca. en firedel av skjermbildet. Det vil si at tekstbokser som ligger bak dette tastaturet, ikke er mulig å skrive til. Det kan derfor oppstå situasjoner der det er umulig å skrive til boksen uten å flytte det manuelt. Dette gjøres enkelt ved å holde skrivestiften på toppen av tastaturet og dra det til ønsket posisjon, men blir fort tungvindt. For da å løse dette på en enkel måte la vi alle tekstbokser som bruker skulle kunne endre så høyt i skjermbildet at tastaturet ikke dekket dem. Informasjon og knapper som ikke krevde tastatur, vil da dekkes av tastaturet, men kun mens det er i bruk.

Figur 48 Det virtuelle tastaturet var nødvendig for å kunne fylle inn i tekstboksene.

6.2.3 Batterivisning

Når det gjelder henting av batteristatus så finnes det en ferdig klasse for dette i OpenNETCF [14]. Denne kan brukes både for å få en prosent verdi av gjenværende batteri, eller man kan få verdien oppgitt i gjenstående tid. Problemet med disse løsningene er at de må være støttet av den aktuelle PDA. PDA'en vi har benyttet, Mobile Compia M3, støttet disse funksjonene.

```
// Oppretter grafisk overvåkning av batteristatusen
private OpenNETCF.Windows.Forms.BatteryLife bBatteryLife;

// Timer som bestemmer hvor ofte batteristatusen skal oppdateres
t = new Timer();
t.Interval = 30000;
t.Tick +=new EventHandler(t_Tick);
t.Enabled = true;

// Kjøres hver gang batterinivået skal oppdateres.
batteryLife.UpdateBatteryLife();
```

Kodeeksempel 1 Ulike setninger for å kontrollere batterinivå.

6.2.4 Passord

For at ikke hvem som helst skal få tilgang til applikasjonen har vi laget en passordløsning. Det vil si at ved oppstart må man logge inn. Innloggingen vil være gyldig så lenge applikasjonen er i bruk. Brukes derimot ikke applikasjonen vil det skje en timeout, og ny innlogging vil være nødvendig. Denne timeout tiden er satt med en standardverdi på 30 minutter.

Vi valgte å opprette en egen xml fil for å lagre innloggingsinformasjonen. Det vil si brukernavn og passord. Filen ga vi navnet ” loginInfo.xml”. Fra utgangspunktet vil denne fila inneholde standard brukernavn og passord som er kryptert med MD5 kryptering. Disse er kryptert sammen med en hemmelig nøkkel, for øyeblikket ”logit”, slik at dataene ikke uten videre kan byttes ut med nye krypterte brukernavn og passord.

```
<?xml version="1.0" encoding="utf-8" ?>
<LoginInfo>
      <Username>CE-AF-26-F6-D3-F3-5C-E2-79-73-6D-A9-7D-BE-0F-83</Username>
      <Password>0C-DC-44-89-D5-81-9D-C0-1C-4D-21-2D-63-76-44-FB</Password>
</LoginInfo>
```

Kodeeksempel 2 Innholdet i loginInfo.xml

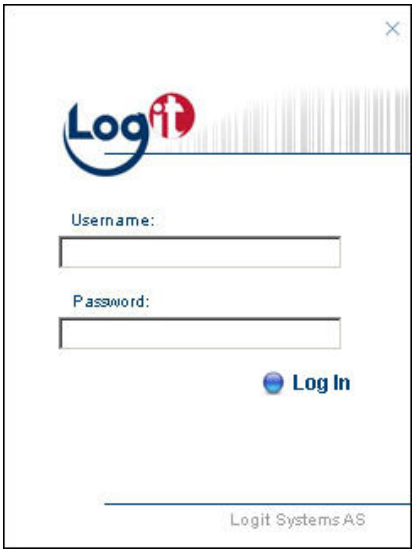
Da vi skulle programmere krypteringsløsningen støtte vi straks på et stort problem. MD5 kryptering var ikke del av det kompakte rammeverket .NET CF. Etter nærmere utforskning fant vi nok en gang løsningen hos OpenNETCF, som i sitt kompakte rammeverk hadde med sin egen MD5 krypterer. Ved å benytte denne kunne vi fortsette arbeidet med passordløsningen.

```
//kryptering
string kryptering(string ord)
{
      byte[] arrHashInput;
      byte[] arrHashOutput;
      MD5CryptoServiceProvider objMD5;
      objMD5 = new MD5CryptoServiceProvider();
      arrHashInput = Convert2ByteArray( ord + secretkey );
      arrHashOutput = objMD5.ComputeHash( arrHashInput );
      return BitConverter.ToString( arrHashOutput );
```

```
}
//krypterings slutt

//kryptereren
byte[] Convert2ByteArray(string strInput)
{
      int intCounter;
      char[] arrChar;

      arrChar = strInput.ToCharArray();
      byte[] arrByte = new byte[arrChar.Length];

      for (intCounter=0; intCounter <= arrByte.Length-1; intCounter++)
            arrByte[intCounter] = Convert.ToByte(arrChar[intCounter]);

      return arrByte;
}
//kryptereren slutt
```

Kodeeksempel 3 Kryptering

Når applikasjonen åpnes vil man få frem et innloggingsvindu. Her blir man nødt til å skrive inn et brukernavn og et passord. Når man så trykker på ”Log in” knappen vil disse to bli kryptert sammen med den hemmelige krypteringsnøkkelen. Man blir da sittende igjen med to krypterte strenger. Har man skrevet riktig brukernavn og passord skal disse stemme overens med de som er lagret i ” loginInfo.xml” fila. Disse blir kontrollert overens med hverandre. Er innloggingen korrekt kommer man direkte inn til applikasjonens hovedmeny, er innloggingen feil får man beskjed om dette og kan forsøke på nytt.



Figur 49 Vindu for innlogging

```
private void pbLogIn_Click(object sender, EventArgs e)
{
      //Kontrollerer brukernavn og passord
      bool detailsOk = checkUserDetails(tbUsername.Text, Password.Text);

      // hvis ok
      if(detailsOk)
      {
            tbUsername.Text = "";
            tbPassword.Text = "";

            showMainMenu();

            btnM_Exit.Image = new Bitmap("\\Program
            Files\\LogIt\\btn_w_mainmenu.gif");
            addToLog("Logged in");
      }
```

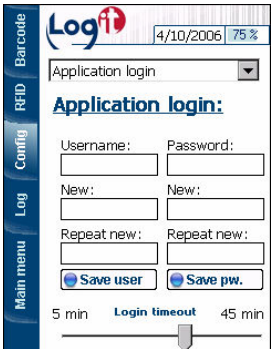
```
      // hvis feil
      else
      {
            tbUsername.Text = "";
            tbPassword.Text = "";
            MessageBox.Show("Wrong username or/and password.","Login
      error",MessageBoxButtons.OK,MessageBoxIcon.Hand,MessageBoxDefaultB
            utton.Button1);
      }
}
```

Kodeeksempel 4 Funksjon som kjøres når bruker trykker på "Log in" knappen:

```
// Timer som logger ut bruker dersom login timeout overstiges
loginTimeout = new Timer();
loginTimeout.Interval = 60000;
loginTimeout.Tick += new EventHandler(loginTimeout_Tick);
loginTimeout.Enabled = true;

// login timeout, hentes fra settings.xml filen
trackBarUpdateInterval.Value =
int.Parse(appSettings.Tables[0].Rows[0].ItemArray.GetValue(7).ToString());

// Regner ut antall minutter til timeout
minutesLeftToTimeout = (trackBarUpdateInterval.Value / 1000) / 60;

...

private void loginTimeout_Tick(object sender, EventArgs e)
{
      // Nedtelling for login timeout, sjekkes hvert minutt
      if(--minutesLeftToTimeout <= 0)
      {
            // logg ut
            reset_menu();
            pnlLogIn.Visible = true;
            pnlLogIn.Location = new Point(0, 0);
      }
}
```

Kodeeksempel 5 Kodeeksempel for login timer

Inne på kontrollpanel vil man ha et eget valg som heter "Application Login". Her vil man få tilgang til å endre brukernavn og passord. Her må man skrive inn opprinnelig brukernavn og/eller passord en gang og det nye to ganger for at en endring skal bli utført. Dette for å forsikre seg om at brukeren har skrevet inn korrekt før et eventuelt nytt brukernavn og passord tas i bruk. Her vil man også kunne justere ønsket tid for timeout på applikasjonen. Denne kan justeres fra 5 minutter til 45 minutter. Mindre enn fem minutter regner vi som unødvendig, og vil sannsynligvis føre til mer irritasjon enn nytte. Over 45 minutter tillater vi ikke brukeren å sette det til. Dette grunnet at brukeren muligens vil slippe å logge inn ved å sette timeouten til en meget høy verdi. Dette vil ødelegge sikkerheten denne funksjonen tilbyr.



Figur 50 Application login

6.2.5 Implementering av telefon

Med tilgang til SDKet til M3en hadde vi også mulighet til å kontrollere telefonifunksjonaliteten på enheten. Dette valgte vi å ikke gjøre. M3en har nemlig en egen fysisk knapp som aktiverer telefoniskjermbildet. Dette legger seg da automatisk over D2D applikasjonen. Ved å trykke på den samme knappen blir det fjernet. Dette fungerte såpass greit i forbindelse med vår applikasjon at vi ikke så noen hensikt i å begynne å utarbeide vår egen løsning, eller manipulere den eksisterende.

6.2.6 XML data struktur

Det finnes to typer XML data strukturer som vårt system skal kunne behandle, Provider Bookings og Loaded/Unloaded Goods Reports. Oppdragsgiver legger Provider Bookings på en server, i vårt tilfelle en FTP server, der man til enhver tid kan hente ned en liste over tilgjengelige oppdrag til den håndholdte enheten. Loaded/Unloaded Goods reports er filene som sendes fra den håndholdte enheten til Logit D2D systemet, etter at varene er skannet på den håndholdte enheten.

Vi vil nå vise et eksempel på hvordan en Provider Booking fil kan se ut og gå kort gjennom utvalgte deler for å forklare hva disse tag'ene beskriver.

```
<MessageHeader>
  <MessageId>15</MessageId>
  <MessageDate>2006-04-21T11:47:23.627+02:00</MessageDate>
  <MessageType>ProviderBooking</MessageType>
  <MessageTypeVersion>1</MessageTypeVersion>
  <MessageTypeRelease>1</MessageTypeRelease>
  <ControllingAgency>LOGIT</ControllingAgency>
  <MessageFunction>UPDATE</MessageFunction>
  <SenderId>tcm1</SenderId>
  <ReceiverId>p3</ReceiverId>
</MessageHeader>
```

Kodeeksempel 6 Provider Booking eksempel - MessageHeader

MessageHeader er en standardheader som inneholder parametere som skal være med i alle status pakker.

`<MessageId>15</MessageId>`

MessageId er en meldings identifikator som kun er for eget bruk. Denne blir ikke benyttet hos server. Id'en er ment for kunne benyttes til intern oversikt over pakkene. Her kan vi utvikle vår egen form for identifikator. Om det ikke skulle være nødvendig kan denne også stå tom.

`<MessageDate>2006-04-21T11:47:23.627+02:00</MessageDate>`

MessageDate inneholder tidspunkt for når filen er opprettet første gang og beskriver år, måned, dag og klokkeslett.

```
<MessageTypeVersion>1</MessageTypeVersion>
<MessageTypeRelease>1</MessageTypeRelease>
<ControllingAgency>LOGIT</ControllingAgency>
```

Disse tre parametrene vil i vår sammenheng ikke være i bruk. De vil derfor bare hardkodes og være med i statusmeldingen for å følge standarden.

`<MessageFunction>`**UPDATE**`</MessageFunction>`

Her beskrives hvilken type melding filen inneholder. Her kan status være ”new”, ”update” eller ”delete”. Statusmeldingene er for så vidt selvbeskrivende og betyr ”ny melding”, ”oppdater melding” og ”slett melding.

`<SenderId>`**tcm1**`</SenderId>`
`<ReceiverId>`**p3**`</ReceiverId>`

Disse tagene skal identifisere mottaker og avsender. De ulike organisasjonsnummer skal brukes til dette. SenderID vil kunne settes i konfigurasjon på den mobile enheten.

```
<ProviderBooking>
<ProviderBookingType>F</ProviderBookingType>
<References>
      <ProviderBookingNo>18</ProviderBookingNo>
      <RequestTime>2006-04-21T11:47:23+02:00</RequestTime>
      <ProviderContract>
            <ContractNo>3</ContractNo>
      </ProviderContract>
      <ProviderService>
            <ServiceNo>3</ServiceNo>
            <ServiceName>TR-NLRotterda-NOOslo-EPAL</ServiceName>
      </ProviderService>
</References>
<PlaceOfReceipt>NLRotterda</PlaceOfReceipt>
<FinalDestination>NOOslo</FinalDestination>
<Unit>
      <UnitIdentification>
            <UnitIdentificationName>UNIT_ID</UnitIdentificationName>
            <UnitIdentificationValue>B1</UnitIdentificationValue>
      </UnitIdentification>
      <UnitIdentification>
            <UnitIdentificationName>UNIT_ID</UnitIdentificationName>
            <UnitIdentificationValue>B2</UnitIdentificationValue>
      </UnitIdentification>
      <UnitLineNo>1</UnitLineNo>
      <Quantity>10.0</Quantity>
      <UnitType>
            <UnitTypeCode>EPAL</UnitTypeCode>
      </UnitType>
      <CargoType>
            <CargoTypeCode>FOTBALL</CargoTypeCode>
            <CargoTypeName>Fotball</CargoTypeName>
      </CargoType>
      <Party>
            <PartyRole>CUSTOMER</PartyRole>
            <PartyType>C</PartyType>
            <PartyCode>CUST1</PartyCode>
            <Name>Customer 1</Name>
      </Party>
      <DatesAndTimes>
            <D2DPickupTime>2006-06-23T09:00:00+02:00</D2DPickupTime>
            <D2DDeliveryTime>2006-07-02T05:00:00+02:00</D2DDeliveryTime>
      </DatesAndTimes>
      <CustomerReferences>
            <CustomerReference>CUST1-1</CustomerReference>
```

```
            <CustomerBookingNo>1</CustomerBookingNo>
      </CustomerReferences>
</Unit>
</ProviderBooking>
```

Kodeeksempel 7 Provider Booking eksempel – Hoveddel av statusmelding

Hoveddelen av statusmeldingen finnes inni `<ProviderBooking>`. Data om det som skal transporteres finner man her.

Inni `<References>` tag'en finner man igjen blant annet:

`<ProviderBookingNo>18</ProviderBookingNo>`
Dette er en identifikator for bookingen som er bestemt fra server siden. Denne beskriver unikt den bestemte bookingen.

`<PlaceOfReceipt>NLRotterda</PlaceOfReceipt>`
Hvor varen mottas.

`<FinalDestination>NOOslo</FinalDestination>`
Destinasjon.

`<Unit> </Unit>`
Alt inni disse tag'ene beskriver data om varene som sendes.

```
<UnitIdentification>
            <UnitIdentificationName>UNIT_ID</UnitIdentificationName>
            <UnitIdentificationValue>B1</UnitIdentificationValue>
</UnitIdentification>
      Det finnes en UnitIdentification tag for hver strekkode som er
      registrert i systemet. Strekkoden legges i UnitIdentificationValue.
```

`<UnitLineNo>1</UnitLineNo>`
Hvilken linje varen hører til i D2d systemet.

`<Quantity>10.0</Quantity>`
Antall kolli det er forventet av denne varetypen.

```
<UnitType>
      <UnitTypeCode>EPAL</UnitTypeCode>
</UnitType>
```
Forteller hva varene blir fraktet ved hjelp av, f.eks. paller eller container.

```
<CargoType>
      <CargoTypeCode>FOTBALL</CargoTypeCode>
      <CargoTypeName>Fotball</CargoTypeName>
</CargoType>
```
Beskriver hva slags varer som fraktes.

```
<DatesAndTimes>
      <D2DPickupTime>2006-06-23T09:00:00+02:00</D2DPickupTime>
      <D2DDeliveryTime>2006-07-02T05:00:00+02:00</D2DDeliveryTime>
</DatesAndTimes>
      Når varen ble hentet og når den blir forventet levert.
```

Vi vil nå vise et eksempel på hvordan en Loaded Goods Report fil kan se ut og gå kort gjennom utvalgte deler for å forklare hva disse tag'ene beskriver.

```
<MessageHeader>
      <MessageId>777</MessageId>
      <MessageDate>2001-12-17T09:30:47-05:00</MessageDate>
      <MessageType>Status</MessageType>
      <MessageTypeVersion>1</MessageTypeVersion>
      <MessageTypeRelease>1</MessageTypeRelease>
      <ControllingAgency>student</ControllingAgency>
      <MessageFunction>new</MessageFunction>
      <SenderId>p3</SenderId>
      <ReceiverId>tcm1</ReceiverId>
</MessageHeader>
```

Kodeeksempel 8 Loaded Goods Report eksempel – MessageHeader

MessageHeader er en standardheader som inneholder parametere som skal være med i alle status pakker.

<MessageId>**777**</MessageId>

MessageId er en meldings identifikator som kun er for eget bruk. Denne blir ikke benyttet hos server. Id'en er ment for kunne benyttes til intern oversikt over pakkene. Her kan vi utvikle vår egen form for identifikator. Om det ikke skulle være nødvendig kan denne også stå tom.

<MessageDate>2001-12-17T09:30:47-05:00</MessageDate>

MessageDate inneholder tidspunkt for når filen er opprettet første gang og beskriver år, måned, dag og klokkeslett.

<MessageTypeVersion>**1**</MessageTypeVersion>
<MessageTypeRelease>**1**</MessageTypeRelease>
<ControllingAgency>**student**</ControllingAgency>

Disse tre parametrene vil i vår sammenheng ikke være i bruk. De vil derfor bare hardkodes og være med i statusmeldingen for å følge standarden.

<MessageFunction>**new**</MessageFunction>

Her beskrives hvilken type melding filen inneholder. Her kan status være "new", "update" eller "delete". Statusmeldingene er for så vidt selvbeskrivende og betyr "ny melding", "oppdater melding" og "slett melding.

<SenderId>**p3**</SenderId>
<ReceiverId>tcm1</ReceiverId>

Disse tagene skal identifisere mottaker og avsender. De ulike organisasjonsnummer skal brukes til dette. SenderID vil kunne settes i konfigurasjon på den mobile enheten.

```
<Status>
<ProviderBookingNo>18</ProviderBookingNo>
<Trip>
<TransportMeans>
      <TransportMean>
            <Code>fastboat</Code>
      </TransportMean>
<Visits>
<ReportType>Loaded</ReportType>
<ArrivalDepartureTime>2006-07-01T00:00:00-05:00</ArrivalDepartureTime>
<ReportTime>2006-07-01T06:30:47-05:00</ReportTime>
<Location>NLRotterda</Location>
<Unit>
<UnitIdentification>
      <UnitIdentificationName>UNIT_ID</UnitIdentificationName>
      <UnitIdentificationValue>A1</UnitIdentificationValue>
</UnitIdentification>
<UnitLineNo>1</UnitLineNo>
<Quantity>
      <ReportedQuantity>10</ReportedQuantity>
      <OmittedQuantity>0</OmittedQuantity>
</Quantity>
<UnitType>
      <UnitTypeCode>EPAL</UnitTypeCode>
      <UnitTypeName>pallet</UnitTypeName>
</UnitType>
</Unit>
</Visits>
<Visits>
      <ReportType>Estimate</ReportType>
      <ArrivalDepartureTime>2006-07-02T05:30:47-
05:00</ArrivalDepartureTime>
<Location>NOOslo</Location>
</Visits>
</TransportMeans>
</Trip>
</Status>
```

Kodeeksempel 9 Loaded Goods Report eksempel - Hoveddel

Hoveddelen av statusmeldingen finnes inni <Status> tagene. Her finnes data om hvilke pakker som
skal håndteres, hva slags type pakker det er, og hvor de kommer fra og skal til.

<ProviderBookingNo>**18**</ProviderBookingNo>

Dette er en identifikator for bookingen som er bestemt fra serverside. Denne beskriver unikt den bestemte bookingen.

<TransportMean>
<Code>**fastboat**</Code>
</TransportMean>
<ReportType>**Loaded**</ReportType>

Dette er parametere for å beskrive forsendelsen. ”fastboat” beskriver fraktmetode, og ”Loaded” er status for forsendelsen.

<ArrivalDepartureTime>2006-07-01T00:00:00-05:00</ArrivalDepartureTime>

Her beskrives tidspunkt for når f.eks et skip kommer eller går fra havn. Denne skal komme med melding fra server, men skal kunne endres manuelt ved avvik.

<ReportTime>2006-07-01T06:30:47-05:00</ReportTime>
Dette er tidspunktet når meldingen blir overført til den sentrale serveren.

Location>**NLRotterda**</Location>
Beskriver avsender av forsendelsen.

```
<UnitIdentification>
            <UnitIdentificationName>UNIT_ID</UnitIdentificationName>
            <UnitIdentificationValue>A1</UnitIdentificationValue>
</UnitIdentification>
      Det finnes en UnitIdentification tag for hver strekkode som er
      registrert i systemet. Strekkoden legges i UnitIdentificationValue.
```

<UnitLineNo>**1**</UnitLineNo>
Ved en forsendelse kan det beskrives flere units. Hvert unit vil ha sitt eget unit nummer.

```
<Quantity>
 <ReportedQuantity>10</ReportedQuantity>
 <OmittedQuantity>0</OmittedQuantity>
</Quantity>
```

Her beskrives antall kolli i den beskrevne unit. ReportedQuantity her beskrevet som 10 sier at dette inneholder ti paller. OmittedQuantity vil beskrive eventuelle avvik fra det forventede antall.

```
<UnitType>
  <UnitTypeCode>EPAL</UnitTypeCode>
 <UnitTypeName>pallet</UnitTypeName>
</UnitType>
```

Her beskrives form for gods. Varene kan for eksempel sendes som container, pall eller pappesker. Her er varene sendt som Europapal.

```
<Visits>
      <ReportType>Estimate</ReportType>
      <ArrivalDepartureTime>2006-07-02T05:30:47-5:00</ArrivalDepartureTime>
      <Location>NOOslo</Location>
</Visits>
```

Her beskrives mottaker av forsendelsen. Ved mottak vil units være presentert her, og ikke under avsender.

6.2.7 Henting av data – FTP

Innhenting av XML filer til M3'en foregår vha. FtpHandler klassen, som gjør det mulig å koble seg til en valgt FTP server og deretter motta en liste med alle XML filene på serveren. For brukeren av systemet blir disse presentert i en combobox under Barcode menyen. Hentingen av oppdrag skjer ved at bruker trykker på knappen til høyre for comboboxen som henter ned liste over tilgjengelige oppdrag. Ved igjen å velge en bestemt Provider Bookingene (oppdrag) blant disse lastes denne inn i applikasjonen på den håndholdte enheten. Brukeren kan så begynne med skanningen av varer.

Ved å benytte oss av OpenNETCF sin løsning kunne man forholdsvis enkelt utvikle FTP funksjonalitet i applikasjonen.

6.2.8 Sending av data – Webservice

Når varene er skannet, skal disse sendes videre i en ny XML fil til D2D systemet. Dette gjøres ved at vi først bygger opp en ny XML data struktur og sender XML dataene som en tekststreng til en web service som kjører på samme server som D2D systemet. Vi måtte lage vår egen web service, pga. kompabilitetsproblemer mellom LogIT Systems java web service og .NET Compact Framework 1.x. Vår web service er laget med .NET Framework 2.0 og fungerer som et mellomledd mellom vår applikasjon og deres web service. I og med at vår web service ligger lokalt i forhold til deres web service, ser vi det som lite sannsynlig at det vil oppstå feil pga. dette. Dersom det etterhvert viser seg at det blir mulig å oppdatere M3'ens OS til WinCE .NET 5.0, vil man kunne kjøre .NET Compact Framework 2.0 på den, og man kan da enkelt fjerne dette mellomleddet.

6.2.9 Håndtering av inndata – Scanner

Den innebygde strekkodeskanneren oppfyller en av oppgavens viktigste funksjoner. Ved hjelp av denne kan varenes strekkoder registreres på den håndholdte enheten for så å videresendes til D2D systemet sammen med annen relevant informasjon.

For å kunne styre den håndholdte enhetens strekkodeskanner benyttet vi oss av produsentens eget SDK. Da kunne vi enkelt importere driveren for skanneren, initialisere den og bruke alle funksjonene som var gjort tilgjengelige for oss.

Første steg var å legge til en referanse til driveren i vårt Visual Studio prosjekt. Dette gjorde vi ved å velge Project->Add reference, deretter fant vi frem til driver filen kalt MCSSLibNet.dll og trykket "ok". Da var alt klart slik at vi kunne benytte oss av driveren i vårt program.

Deretter opprettet vi et objekt i vår klasse som representerer strekkodeskanneren:

```
// Oppretter et objekt som representerer strekkodeskanneren
private MCSSLibNet.ScannerControl scan;
```

Kodeeksempel 10 Strekkodeskanner - oppretter objekt

I klassens konstruktør initialiserer vi scan objektet og legger til lyttere for to hendelser.

```
// Initialiserer scan objektet
scan = new MCSSLibNet.ScannerControl();

// Legger til lytter for hendelsen når skanneren mottar data
scan.ScannerDataEvent +=new
MCSSLibNet.ScannerDataDelegate(scan_scannerDataEvent);

// Legger til lytter for når applikasjonens "hovedbilde" aktiveres
this.Activated +=new EventHandler(LogITd2d_Activated);
```

```
```

Kodeeksempel 11 Strekkodeskanner - diverse kodeeksempler

Denne funksjonen kjøres fordi vi la til en lytter for klassens egne Activated hendelse.

```
private void LogITd2d_Activated(object sender, EventArgs e)
{
    // Reset the scan event handler for the Scanning object to the form's
    // delegate.
    scan.RegisterRecieveForm();
    scan.SetDefaultOption();
}
```

Kodeeksempel 12 Strekkodeskanner - Reset scan event

I klassens Load hendelse kjøres bl.a. denne koden, som forsøker å initialisere skanneren og eventuelt gir feilmelding dersom forsøket ikke er vellykket. Koden forsøker også å registrere S knappen på høyre side av M3en, slik at denne aktiverer strekkodeskanneren.

```
int nRet = scan.ScanInit();

if(nRet != 0)
      MessageBox.Show("Scanner Init Failed");
else
      scannerSetting();

scan.RegisterRecieveForm();
if(!scan.RegHotKey(0x0111, scannerButton,true))
      MessageBox.Show("RegHotKey failed");
```

Kodeeksempel 13 Strekkodeskanner – Initialisering

Denne funksjonen kjøres dersom strekkodeskanneren blir initialisert og setter bl.a. lyd som skal avspilles ved innlesing av en strekkode.

```
private void scannerSetting()
{
      // Lyd som avspilles når strekkode leses
      scan.UseDefaultSound(true,"\\Windows\\alarm1.wav");
      MCSSLibNet.MCModuleOption mdo = new MCSSLibNet.MCModuleOption(3,1,2);
      scan.SetModuleOption(ref mdo);
}
```

Kodeeksempel 14 Strekkodeskanner - Lydavspilling

scan_scannerDataEvent funksjonen kjøres hver gang en strekkode leses inn, og det er her man kan påbegynne bearbeidingen av disse. Som man ser har denne funksjonen et argument kalt MCSSLibNet.ScannerDataArgs e, og det er i dette objektet e at man finner verdiene som blir lest inn. Strekkoden hentes ut ved å skrive e.ScanData som returnerer en tekststreng.

```
private void scan_scannerDataEvent(object sender,
```

```
MCSSLibNet.ScannerDataArgs e)
{
      // Kodeeksempel ble for langt til å lime inn. Er vedlagt
}
```

Kodeeksempel 15 Strekkodeskanner - Hendelse ved innlesing av data

Når programmet avsluttes forsøkes det å fjerne registreringen av S knappen igjen, slik at denne er tilgjengelig for evt. andre applikasjoner. Dette gjøres også med selve strekkodeskanneren. Dette foregår i `LogITd2d_Closing funksjonen.`

```
if(!scan.UnRegHotKey(0x0111))
      MessageBox.Show("UnRegHotKey fail");

scan.ScanClose();
```

Kodeeksempel 16 Strekkodeskanner - ved lukking av applikasjon

6.2.10 WLAN

Her hadde vi valget mellom å lage en WLAN løsning helt fra bunnen av eller å la den innebygde ”connection manager” ta seg av søking etter trådløse nett, autentisering osv. Etter litt diskusjon kom vi frem til at vi ville benytte oss av den innebyggede ”connection manager”, og kun aktivere/deaktivere WLAN kortet fra vårt program.

Referansen til driveren ble lagt til på samme måte som for strekkodeskanneren, men filen vi nå la til heter WlanUtilCF.dll. Dessuten må vi kopiere en fil kalt WlanUtil.dll over på M3’en sin rot katalog, da driveren vi la til over kaller opp funksjoner i den sistnevnte dll filen.

Oppretter et objekt som representerer WLAN kortet:

```
// oppretter et objekt som representerer wlan kortet
private WlanUtilCF.WlanControl wlanControl;
```

Kodeeksempel 17 WLAN - Oppretter et objekt

Initialiserer dette objektet i klassens konstruktør.

```
// kjører wlan klassens konstruktør
wlanControl = new WlanUtilCF.WlanControl();
```

Kodeeksempel 18 WLAN – Initialiserer

Deretter kunne vi bruke de to linjene med kode der vi ønsket å slå på eller av WLAN kortet.

```
// aktiverer wlan kortet
wlanControl.SetWlanPwrOn();

// deaktiverer wlan kortet
wlanControl.SetWlanPwrOff();
```

Kodeeksempel 19 WLAN - slå av/på

6.2.11 GPRS

GPRS kommunikasjon var et av elementene på oppdragsgivers ønskeliste som ikke var med i oppgavedefinisjonen. Eller nærmere bestemt var det definert ønske om trådløs kommunikasjon. Konklusjonen ble senere fastslått til at både GPRS og WLAN var optimalt, men at WLAN skulle prioriteres i utviklingsfasen, mens GPRS ville være ønskelig å implementere hvis tiden kunne tillate dette. Da ting så ut til å falle på plass mot slutten valgte vi å sette oss inn i dette temaet for å presentere et så komplett resultat som mulig.

GPRS er svært effektivt for en slik mobil enhet som utvikles, spesielt hvis man beveger seg over store områder. I M3 SDKet fra Mobile Compia var det også presentert kodeeksempel for bruk av GPRS. Dette var til stor hjelp for oss. Dessverre var koden svært lite kommentert, så innføring i dens betydning krevde litt arbeid rundt tolkning og testing. For å benytte oss av GPRS på M3, måtte vi importere driverfilene. GSMCoreNet.dll ble lagt i applikasjonskatalogen, mens GSMCore.dll måtte legges i rotkatalogen.

Å sette riktige parametere for GPRS tilkoblingen var en utfordring. Dette er spesifikt for hver enkelt operatør. Etter mye søking på Internet kom vi frem til en liste presentert av Esato.com, ”Network parameters for GPRS connection” [79] som ga oss løsningen.

Tabell 10 Utdrag fra ” Network parameters for GPRS connections” [79]

The list works like this:
Network operator, Country, Modem properties:,"extra settings", Additional AT commands, Telephone number, TCP/IP settings,IP address, only if not dynamic, TCP/IP settings:DNS 1, only if not,dynamic, TCP/IP settings:DNS 2, only if not,dynamic, TCP/IP settings:IP header compression, Connection user name,(if na= not needed) Connection Password,(ifna = not needed)
…
Netcom, Norway, *99***1# ,dynamic, 212.45.188.43, 212.45.188.44, no ,n.a., n.a.
…
MOVISTAR MOVISTAR,Telenor Mobil, Norway,
AT+CGDCONT=1,"IP",,"internet",;+CGQREQ=1,0,0,0,0,0;+C, GQMIN=1,0,0,0,0,0,*99***1#, dynamic, dynamic, dynamic, no, 1111
…
[79]

Disse instillingene må settes spesifikt for hver enkelt operatør, og det bør derfor lages en form for konfigurasjonsløsning i applikasjonen. I og med at vi startet arbeidet av GPRS på ”overtid” har disse parametrene blitt hardkodet. Det bør lages et menyvalg for dette i config. En mulighet for dette er å presentere alle parametere, for alle operatører i en egen fil, for så å la bruker velge land og operatør i config. En annen løsning er at alle de ulike parametrene må tastes inn i config.

6.2.12 Dataset

Et dataset [80] er en kopi av data som ligger i minnet på enheten programvaren kjøres på. Et dataset består av flere DataTable objekter som inneholder dataene og man kan ved å aksessere disse tabellene legge til/slette/endre data. I vår applikasjon brukes DataSet klassen til å håndtere data fra XML filene som representerer Provider Bookings og Loaded/Unloaded

Goods Reports. Dataset objektene er også grunnlaget for dataene som presenteres i datagrid'ene for bruker under menyvalget "Barcode".

6.2.13 OpenNETCF 1.4

Installasjonen gjennomføres enkelt ved å kjøre 2 Windows Installer Package filer [81], som automatisk installerer filene, gjør nødvendige registerendringer osv. Etter at dette er fullført kan man opprette et nytt prosjekt i Visual Studio .NET kalt "OpenNETCF application" og benytte seg av den ekstra funksjonaliteten rammeverket tilbyr. Første gang man overfører programmet man har utviklet til den håndholdte enheten eller emulatoren, installeres også nødvendige filer på disse.

Selv om programvaren er åpen for bruk kommersielt og ukommersielt stiller OpenNETCF noen enkle krav rundt bruk av deres kode. Vi har lagt med deres "Shared Source License, July 23, 2004" som vedlegg [v6].

6.2.14 M3 SDK

Kort tid etter at vi mottok den håndholdte enheten, M3 fra Mobile Compia, henvendte vi oss til produsenten for å få tilgang til deres SDK(Software Development Kit). På denne måten kunne vi enkelt kontrollere strekkode skanner, WLAN kort osv. Vi hadde problemer med å få dette til å fungere i starten, da dokumentasjonen var omtrent ikke-eksisterende. Det eneste vi hadde var enkle eksempler på bruk av koden. Etter en del prøving og feiling fant vi til slutt ut hvor driverne skulle plasseres på M3en. En av driverne måtte plasseres i katalogen til programmet vi utviklet, mens den andre måtte plasseres i toppen av katalogstrukturen. Dette fordi driveren som lå i programkatalogen benyttet seg av funksjoner som befinner seg i den andre driveren. Ble ikke dette gjort riktig fikk vi en "MissingMethodException" når vi kjørte programmet vårt.

6.2.15 Timedate

I meldingfilene er det en rekke tager som inneholder ulike tidspunkt. Her benyttes en standard for tid og dato i formatet " YYYY-MM-DDThh:mm:ssTZD", ISO 8601 [82]. Dessverre fantes det ingen funksjon i C# som kunne returnere tid og dato i dette formatet. Vi måtte derfor lage en liten metode som kunne utføre denne jobben. Standarden inneholder også TZD (time zone designator). Vi ser for oss at TZD blir noe som kan settes i admin i applikasjonen, men midlertidig velger vi å hardkode dette. Ved å kalle opp ISOTime kan man så få returnert tid og dato i riktig ISO format. Koden her plukker ut hvert enkelt element; år, dato, dag osv av TimeDate funksjonen. Disse dataene returneres kun tosifret hvis tallet virkelig er tosifret. En returneres som "1" og ti returneres som "10". ISO standarden krever at tallene for eksempel for Mars måned returneres som "03". Det må derfor utføres en test på hvert eneste siffer, og om nødvendig legge til en "0" før tallet. Når hele ISO verdien er generert returneres den som en string.

```
public string ISOTime(string TZD)
{
      //YYYY-MM-DDThh:mm:ssTZD
      //2001-12-17T09:30:47-05:00
      string test= "";
      string ISO = "";
```

```
      ISO += DateTime.Now.Year;
      ISO += "-";

      test = DateTime.Now.Month.ToString();
      if(test.Length>=2)
      {ISO += test;}
      else{ISO += 0" + test;}
      ISO += "-";

//.........KODE FORKORTET........//
//..Vises fullstendig I vedlegg..//

      test = DateTime.Now.Second.ToString();
      if(test.Length>=2){ISO += test;}
      else{ISO += "0" + test;}
      ISO += "-";
      ISO += TZD;

      return ISO;
}
```

Kodeeksempel 20 TimeDate til ISO 8601

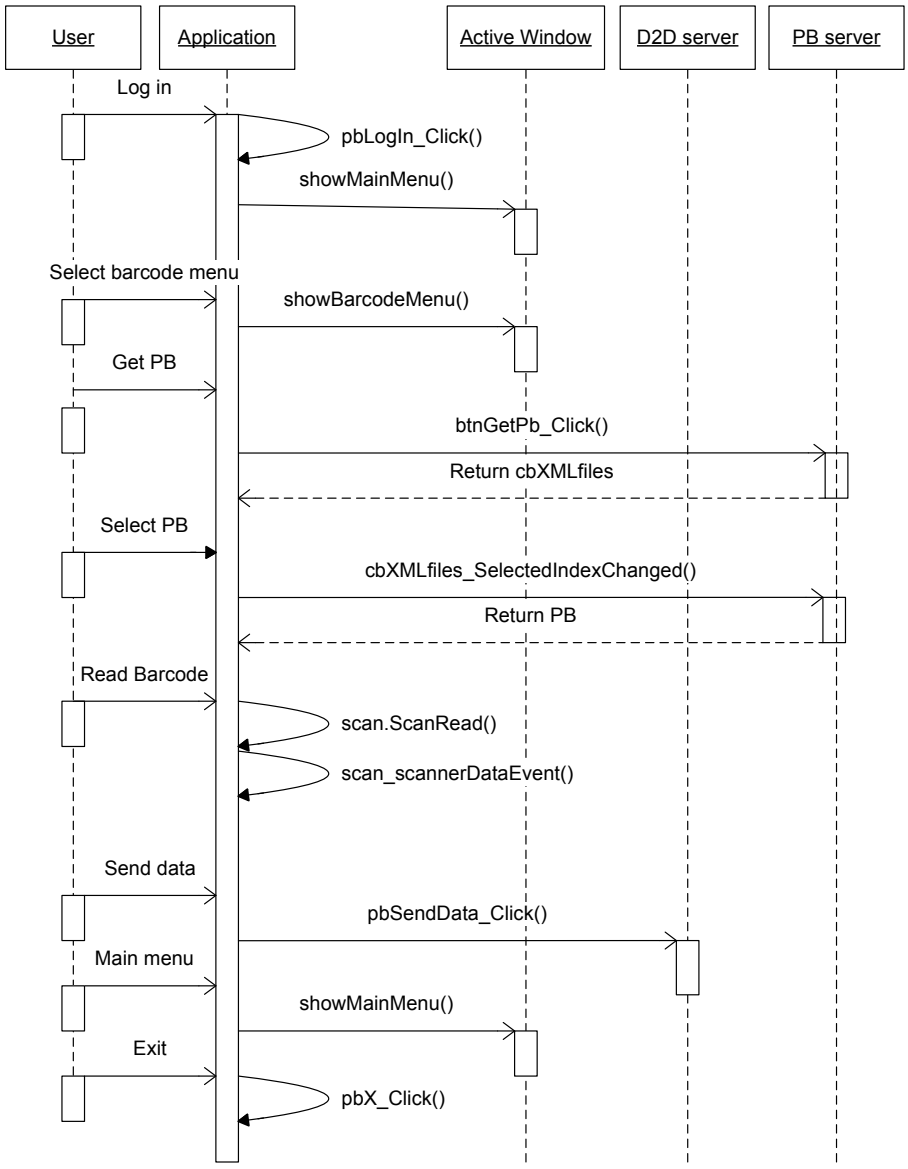
6.2.16 Kode og filstrukturen

I og med at prosjektet er utviklet for en ekstern oppdragsgiver er det viktig at resultatet er forståelig og mulig å arbeide videre på. Dette gjelder også kodematerialet slik at andre utviklere har mulighet til å sette seg inn i koden.

Vi har derfor lagt vekt på å kommentere koden godt. Dette er en svært effektiv måte å gi en fremtidig programmerer innsikt i hva som egentlig foregår i koden. Men selv med god kommentering kan koden virke uoversiktelig. Det er derfor viktig å splitte opp de ulike elementene på en fornuftig måte. I vår løsning kontrollerer hovedklassen det meste som skjer i applikasjonen. Denne inneholder også en rekke små funksjoner. Alle større oppgaver har fått sine egne klasser som kalles opp ved behov. Her er det for eksempel en egen klasse for FTP, for visning/skjuling av tastatur og en for fullskjerm.

Vi valgte å ikke ta med en fullstendig oversikt over de ulike klasser og metoder i som finnes i applikasjonen her i rapporten. Dette ble såpass omfattende materiale, og for den generelle leser heller ikke interessant. For de spesielt interesserte er det vedlagt fullstendig kildekode både i papirutgave og på CD. For å gi innblikk i koden og strukturen har vi der vært ekstra nøye med kommentering. Her i rapporten nøyer vi oss med å presentere et mer forenklet bilde av det som foregår under registrering av en strekkode. I sekvensdiagrammet på neste side ser vi for oss typisk bruk av applikasjonen, og presenterer de viktigste metodene som kjøres ved en typisk brukersesjon.

Sekvensdiagrammet viser en bruker som logger seg inn på applikasjonen og kommer til hovedmeny. Herfra velger brukeren å gå inn på menyvalget ”Barcode”. Her henter bruker inn potensielle oppdrag, for så å laste ned et bestemt. Bruker skanner deretter inn en strekkode, før han sender fila til den sentrale D2D serveren. Til slutt går han ut til hovedmenyen og avslutter applikasjonen derfra.



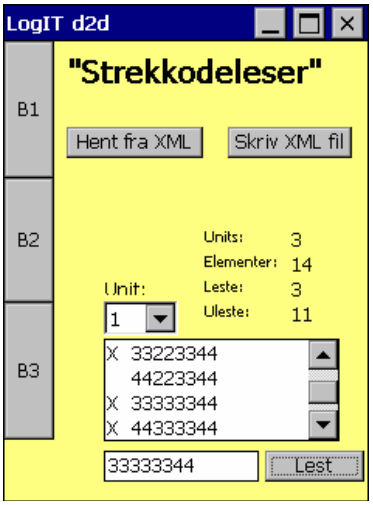
Figur 51 Sekvensdiagram illustrerer de viktigste metodene som kjøres ved lesing av strekkode

6.2.17 Vanskeligheter

Ved utvikling av et nytt produkt er det selvfølgelig vanlig at det oppstår problemer, og at prosjektet i tilfeller tar nye veier. Når vi nå har kikket nærmere på elementene i det ferdige systemet ønsker vi derfor å kikke litt på hvilke hindre vi har møtt på veien. Det er normalt at veien til målet er usikker ved utvikling av en prototyp, og noen ganger er prøving og feiling eneste vei til målet. Vi ønsker å nevne disse aspektene i rapporten da det har hatt betydning for vårt arbeid gjennom prosjektperioden.

Tidlig prototyp: Ivrige etter å lage et godt produkt satte vi i gang med utviklingen. Med få retningslinjer og relativt frie tøyler fra oppdragsgiver utviklet vi programvaren etter de

spesifikasjoner vi hadde tilgjengelig. Etter hvert som det begynte å ta form og vi hadde en prototyp av prototypen å vise frem, tok vi den med i møte med oppdragsgiver.



Figur 52 Tidlig prototyp av applikasjonen

I denne versjonen var datalagringen under registrering av identifikasjonsmerker basert på arrays. Fire arraylister med navnene Posisjon, ArrayUnitNo, ArrayStrekkode og ArrayStrekkodeLest inneholdt de nødvendige data. Posisjon inneholdt kun et nummereringssystem for å identifisere de ulike radene. ArrayUnitNo beskrev hvilken Unit id koden tilhørte, og ble eventuelt merket med "Ny" om dette var en ny strekkode. Hensikten med den siste kolonnen var å vise om en strekkode var blitt innlest eller ikke. Innleste strekkoder ble merket med en "X", eller "N" om de var nye. Tabellen lenger ned på siden illustrerer hvordan dataene var lagret i arrays.

Vi hadde også utviklet et brukergrensesnitt basert på dette systemet. Her ble dataene presentert i en ListBox. I boksen ble alle strekkodene listet nedover. Når noen ble merket som lest ble dette markert ved å plassere "X" foran IDen. Nye IDer ble lagt til i lista.

Tabell 11 Datalagring i Arrays

Posisjon	ArrayUnitNo	ArrayStrekkode	ArrayStrekkodeLest
0	1	931257772315	
1	1	9312573213253	X
2	1	9312577123222	
5	3	2111143266734	
6	3	2111243266734	
7	3	2111343266734	X
10	3	2111643266734	
11	Ny	3218492132134	N
12	Ny	3218492132135	N
13	Ny	3218492132136	N
---	---	---	---

Da dette arbeidet ble presentert for våre oppdragsgivere begynte tankene å svirre, og de fikk straks et klarere bilde av hva de egentlig var ute etter. Dessverre var dette ganske annerledes enn hva som var utviklet til nå. De ønsket nå at løsningen skulle være mest mulig lik deres PC baserte løsning med dataene presentert i datagrids. Vi tok selvfølgelig deres ønsker til etterretning. Mesteparten av koden så langt måtte dermed forkastes og vi satte i gang utviklingen av en ny versjon. Denne viste seg heldigvis å fungere på en mer praktisk måte med hensyn på datalagring, så endringene hadde også sine fordeler. Utviklingen videre presenteres nærmere i kapitellet "6.2 Softwareløsning".

Design: Brukergrensesnittet ble designet og tilpasset oppdragsgiver, LogIT Systems, sin logo. Designet baserte seg på logoens hovedfarge og var også tilpasset logoens form. Midt i prosjektperioden ble vi opplyst om at LogIT nå hadde fått en ny logo, og at denne helst måtte benyttes i applikasjonen. Resultatet ble en del dobbeltarbeid med grafikken til applikasjonen. Heldigvis var den nye logoen forholdsvis lik både i farger og utseende, så det bød ikke på så store problemer. For at den nye logoen ikke skulle oppta for stor del av skjermbildet valgte vi å kutte den nederste delen av buen på g'en.

Compact Framework: Å arbeide med CF uten å være godt kjent med det kan by på mange uønskede overraskelser. For å kunne lage et mer kompakt rammeverk er selvfølgelig en del funksjoner utelatt i CF. Under utvikling møter man derfor ofte på vanskeligheter hvor man overhode ikke hadde forventet det.

En stor begrensning vi smertelig oppdaget i CF var Datagrid klassen. Denne inneholdt svært få muligheter til manipulasjon. Enhver liten endring som ville vært en kort kommando i et vanlig rammeverk, ble raskt et forholdsvis stort hinder, og en skikkelig utfordring å løse. Et annet problem vi møtte på i CF var at FTP funksjonalitet var delvis utelatt. Dette gjorde programmering mot FTP server svært vanskelig. Her hadde heldigvis OpenNETCF publisert åpen kode for FTP klient, og dette gjorde arbeidet betydelig enklere for oss. MD5 kryptering var også utelatt, og det var generelt mange små begrensninger vi ikke hadde forventet.

Meldingsfiler: Det tok en stund før LogIT endelig fastsatte hvilke type meldingsfiler de ønsket sendt til og fra den mobile enheten. Dette var nok fordi de enda ikke hadde dette klart for seg, og ønsket å holde mulighetene åpne. For vår del førte dette til en del ekstra arbeid, da vi stadig måtte ta hensyn til endringer. Ikke før i overgangen April-Mai ble meldingsfilene vi kunne forholde oss til 100% fastsatt slik at arbeidet kunne ferdigstilles.

Disse problemene grunner nok i at selve prosjektoppgaven ble utlyst fra LogIT Systems for de manglet innsikt i hvilke muligheter som fantes. De ønsket å se på hva som kunne utvikles på en mobil enhet, men hadde ingen klare retningslinjer for hva de var ute etter. Dette har imidlertid løst seg utover i prosjektperioden. Vi har sett på dette som en fin erfaring med reell produktutvikling, med usikkerhet og problemer som til tider skaper mye ekstraarbeid og hodebry.

ActiveSync: Det største problemet i prosjektperioden kom da ActiveSync sluttet å fungere. Uten ActiveSync var det ikke mulig å overføre prosjektet til M3en, og dermed umulig å fortsette utviklingen av applikasjonen. Vi har beskrevet dette problemet og løsningen nærmere i kapittel ”7.1 Implementasjon på håndholdt enhet”.

Server: Å arbeide mot en server har i mange tilfeller bydd på store problemer. I starten var det mange løsninger som ikke var ferdig installert eller riktig satt opp på serveren. Logit D2D serveren har vi til tider hatt store problemer med. Denne har ved uheldige kombinasjoner av inndata låst seg og vært svært tidkrevende å få opp og gå igjen. Heldigvis har oppdragsgiver vært tilgjengelig for å hjelpe når det har vært nødvendig.

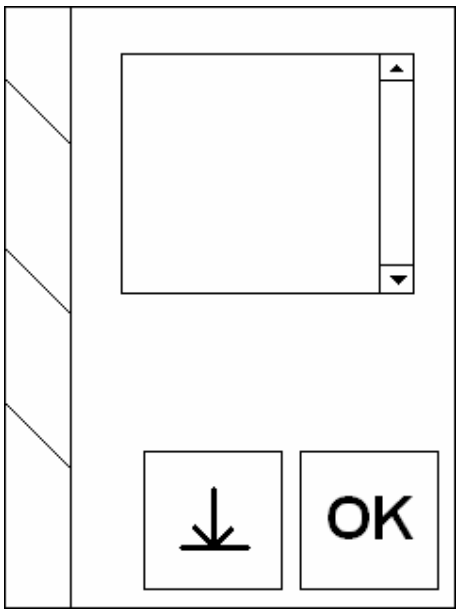
SQL server:
Microsoft SQL Server 2000 ble benyttet. I starten av prosjektperioden hadde vi en del problemer med installasjonen av denne. Vi forsøkte på 3 forskjellige PC’er med Windows XP installert. På 2 av dem feilet installasjonsprosedyren og den tredje hadde problemer med å lytte på porten den tar i mot forespørsel på. Løsningen ble til slutt å installere programvaren på en PC med Windows 2000.

6.3 Design og brukergrensesnitt

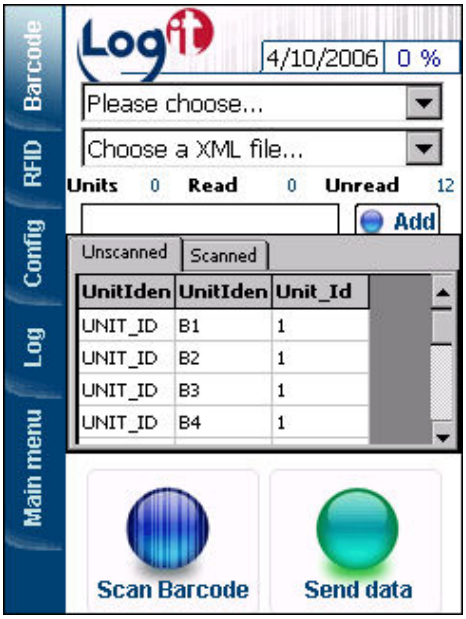
Design er en svært viktig del av et hvilket som helst system som en bruker skal benytte. Designet omhandler ikke bare farger og utseende, men brukervennlighet, plassering av knapper og funksjoner. Vi kaller dette brukergrensesnitt, eller GUI(Graphical User Interface),

den grafiske delen av applikasjonen som er bindeleddet mellom bruker og applikasjon. Et godt brukergrensesnitt gjør arbeidsoppgavene enklere for brukeren. Men design omhandler også utseende, og at det skal se bra ut. Et godt utseende gir applikasjonen "respekt" fra brukeren. Men dette krever at det ser profesjonelt ut. Design og brukergrensesnitt innebærer nøye avveininger rundt bruk av farger, menyer, knapper og ulike skjermbilder. Vi har forsøkt å få denne prototypen til å se mest mulig ut som et ferdig produkt. Tiden tillot oss å legge litt arbeid i grafikk og design. Vi skal her ser nærmere på hva som er gjort, og hvorfor.

I kapittel " 3.7 Planlegging og design" gjorde vi oss noen tanker rundt hvordan applikasjonens brukergrensesnitt kunne se ut. Her gikk vi ikke i detalj som tekstbokser og farger, men prøvde å skissere et grunnlag. Vi har nå arbeidet videre på dette grunnlaget for å utvikle design og plassere menyer, datalister og knapper i et mest mulig ferdig brukergrensesnitt. Et eksempel av resultatet ser du her hvor vi sammenligner en tidlig skisse med dagens prototyp.



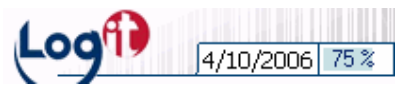
Figur 53 Tidlig designskisse



Figur 54 Prototypens design

I og med at dette skal benyttes på en håndholdt enhet satt vi oss en del regler. All informasjon skal være lett tilgjengelig. De aller fleste funksjoner skal kunne aktiveres ved bruk av fingeren. Altså må de være store og enkle å treffe. Fargekombinasjonen og tekst skal benytte store kontraster slik at de selv i sollys er enkle å tyde og lese.

Vi valgte opprinnelig å dele opp skjermbildet i fire deler: Header, meny, hovedbilde og bunn. Bunnen ble senere prioritert bort da vi følte denne løsningen ikke utnyttet skjermbildet godt nok. Dataene som skulle presenteres her finner man igjen andre steder i skjermbildet. Headeren er med på å gjøre skjermbildet luftig. Praktisk sett er ikke denne veldig viktig. Den kunne i det minste vært laget mye mindre. Men ved å presentere logo og gi litt plass her, så håper vi at brukeren skal oppleve litt "luft" i skjermbildet. Det er viktig at skjermen ikke oppleves som et kaos av knapper og datafelt, men at man får et overblikk over det hele. Vi valgte derfor å gi god plass til LogIT Systems sin nye logo her. I bakgrunnen presenteres en svak strekkode. Denne er der kun for utseendets del, men tar man seg tid til å tyde den, så skjuler



Figur 55 Applikasjons header

den faktisk ordene ”LogIT Systems AS”. Headeren vil være en del av skjermbildet hele tiden mens en bruker benytter applikasjonen. Vi valgte derfor også å presentere dato og batteri her. På denne måten vil bruker alltid ha full oversikt.

Menyen er plassert til venstre i skjermbildet. Her gjorde vi et valg om at menyen alltid skal være tilgjengelig og at knappene skal være forholdsvis store. Dette ”stjeler” en del av skjermbildet, men gir brukeren full tilgang. Ved at knappene er store er de enkele å benytte selv med fingrene. Man behøver ingen skrivestift for å treffe sine valg. Ved at menyen alltid er tilgjengelig vil brukeren hele tiden få tilgang til hvor han måtte ønske. Lysere skrift og en lys gradient over knappen gir en effektiv og mer diskret visning av hvor man befinner seg. I figuren ”Menyvalg” er ”Log” markert.

Figur 56 Menyvalg

Hoveddelen er den mest omfattende delen. Her skal store mengder data kunne presenteres. De er jo da også skilt ut som ulike ”ark” som kan presenteres. Hvordan disse er lagt opp vil vi se nærmere på under kapittelet ”Sideinnhold”.

6.3.1 Utseende

Utseende er viktig. Da tenker vi på inntrykket applikasjonen gir en bruker ved å se på det, uten nødvendigvis å bruke det. Hvordan vil en eventuell kjøper av LogIT Systems sine systemer reagere når han kikker på produktet for første gang. Virker det innbydende, profesjonelt og forseggjort, eller som slurvete hastearbeid? Utseende gir et førsteinntrykk som varer lenge, og kan være vondt å vende om man kommer galt ut. Hva som ligger bak i programvaren kan være akkurat det samme, men det som ser profesjonelt ut er enklere å selge.

- **Innbydende utseende:** Vi har arbeidet med å kunne presentere et innbydende utseende på applikasjonen Selv om vi utvikler en prototyp, så skal denne selges om applikasjonen skal ha mulighet til å leve og utvikles videre. Enten den skal selges til investorer, sluttbrukere, eller selve idéen skal presenteres til høyerestående innen LogIT Systems for videre satsning.

- **Velkjent:** Vi har forsøkt å presentere data og menyer på velkjente måter. Slik vil applikasjonen være mest mulig selvbeskrivende for brukeren.

- **Luftig og lyst:** Skjermbildet er lagt opp mest mulig luftig og lyst. Dette av flere årsaker. En lys bakgrunn er god kontrast til mørk skrift. Lyse farger er oppløftende i motsetning til mørke dystre farger, og et luftig oppsett gjør skjermbildet oversiktelig.

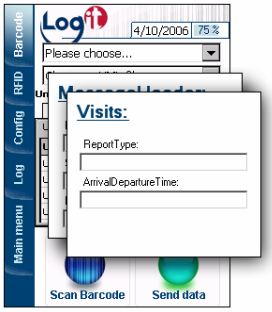
- **Farger:** Farger utgjør en stor del av inntrykket en applikasjon gir. Her er det viktig å ikke bruke for mange. Vi valgte å ta utgangspunkt i LogIT Systems sin logo. Denne inneholder blå og rød skrift på hvit bakgrunn. Vi valgte denne blåfargen inn i designet som hovedfarge. Ved å bruke denne blåfargen, samt noen nyanser av denne, mot gråtoneskalaen på tekst og bokser har applikasjonen fått et ryddig utseende. Rødfargen i logoen lot vi være da flere farger ville skapt et mer uryddig bilde, og fordi den i seg selv trekker en liten oppmerksomhet til utvikler av systemet.

- **Moderne:** For å gjøre produktet salgbart er det viktig at det oppfattes som nytt. Ved å være oppmerksom på trender og hva som er moderne på softwaremarkedet i dag har vi forsøkt å lage et moderne utseende på applikasjonen. Avrundete former med skygger og gradienter gjør bildet mykere, og fjerner inntrykket av et gammeldags firkantutseende. Blått er også meget populært innen applikasjon og webdesign, og passet derfor fint med at denne fargen ble hentet ut fra logoen. Vi har latt oss inspirere av Apple [83] hvor man ofte opplever store "myke" ikoner i blåfargeskalaen hvor skyggene gir en slags svevende effekt.

Figur 57 Hovedmenyen

6.3.2 Sideinnhold i vinduer

Funksjoner og valg har blitt fordelt i logiske samlinger og presenteres på en rekke små ark. På denne måten kan man oversiktelig få opp de relevante data ved å klikke seg frem i menyene. Man når de aller viktigste dataene direkte ved å klikke i hovedmenyen, eller tab menyen til venstre. Mer spesifikke data og funksjoner kan man nå ved å bruke dropdown menyene som finnes i noen av kategoriene. De bytter ut deler av skjermbildet ved å vise de ulike arkene man forespør.



Figur 58 "Barcode" med undermenyer

Dette er tilfellet for eksempel for siden "Barcode". Dette er siden hvor det er størst krav til å presentere store mengder data. På hovedsiden er det her presentert lister med strekkoder som skal scannes og er scannet, samt litt data om hvor mange som er registrert osv. Dette tar fort opp plass i skjermbildet. Videre er det en del spesifikke data som skal være mulig å manipulere i ulike tilfeller. Disse dataene finner man da ved å velge tema i den øverste dropdown menyen. Når man er ferdig kan man velge seg tilbake for å se det opprinnelige bildet.

For å gi brukeren oversikt over hvilke undersider de befinner seg på, så blir disse presentert med en overskrift. Dette gjøres ikke i hovedsidene av to årsaker. Man kan se hvilken hovedside man befinner seg i ut i fra hovedmenyen til venstre. Slike overskrifter tar mye plass, og bør derfor ikke brukes mer enn nødvendig.

De ulike arkene er basert på paneler som vises eller ikke vises. Applikasjonen består derfor av en rekke paneler med de ulike skjermbildene. Bare et skjermbilde er aktivt av gangen. Klikker man for å åpne et nytt skjermbilde blir alle skjermbilder nullstilt med reset_menus(), mens det man ønsker åpnet blir satt visible = true, og posisjonert riktig i skjermbildet.

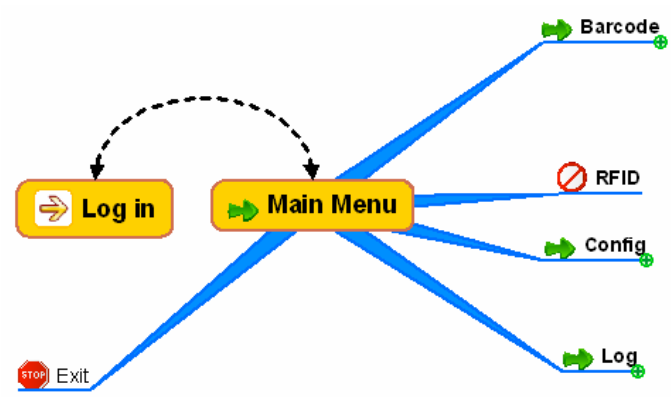
```
private void btnM_Barcode_Click(object sender, EventArgs e)
{
      showBarcodeMenu()
}

...

private void showBarcodeMenu()
```

```
{
      reset_menu();
      pnl_Barcode.Visible = true;
      pnl_Barcode.Location = pActive;
      btnM_Barcode.Image = new Bitmap("\\Program
      Files\\LogIt\\btn_w_barcode.gif");
}
```

Kodeeksempel 21 Koden som aktiveres ved klikk på barcode i hovedmenyen

Hovedmenyen og headeren er unntak fra disse sidevisningene med ulike ark. Disse to menyene vises permanent så lenge en bruker er innlogget. Hver av dem er også basert på paneler, men de skifter verken plassering eller synlighet.

6.3.3 Menystruktur



Figur 59 Applikasjonens hovedstruktur

På en håndholdt enhet har man som nevnt lite plass for å presentere data. Skjermen er liten og scrolling er sjeldent foretrukket. Hovedmenyen har vi forsøkt å gjøre enkel og oversiktelig. Til venstre viser en oversikt over de valgene man blir tilbudt. Man begynner automatisk ved ”Log in”(markert med gul pil). Her er det i utgangspunktet lagt opp for å kun ha en vei videre. Nemlig å skrive korrekt brukernavn og passord, for så å bli videresendt til hovedmenyen. Dette er en sikkerhet som hindrer uvedkommende tilgang til softwaren i maskinen. Dette fungerer som en sikkerhet i og med at applikasjonen åpnes automatisk ved oppstart og benytter hele skjermbildet. Man vil derfor ikke ha tilgang til noen av Windows applikasjonene mens denne applikasjonen kjører. Vi har imidlertid i dialog med oppdragsgiver kommet frem til at det kan være praktisk med en ”nødutgang” inntil videre. Dette kan være praktisk under utvikling og demonstrasjon hvor man f.eks må gjøre endringer på software i den håndholdte enheten.

Når man har skrevet riktig brukernavn og passord ved ”Log in” ender man opp i hovedmenyen ”Main Menu”. Valgene man finner her er de samme man til enhver tid, så lenge man er innlogget, får presentert i menyen til venstre i skjermbildet. Hva som befinner seg under dette nivået er presentert i illustrasjonen ”Applikasjonens struktur”.

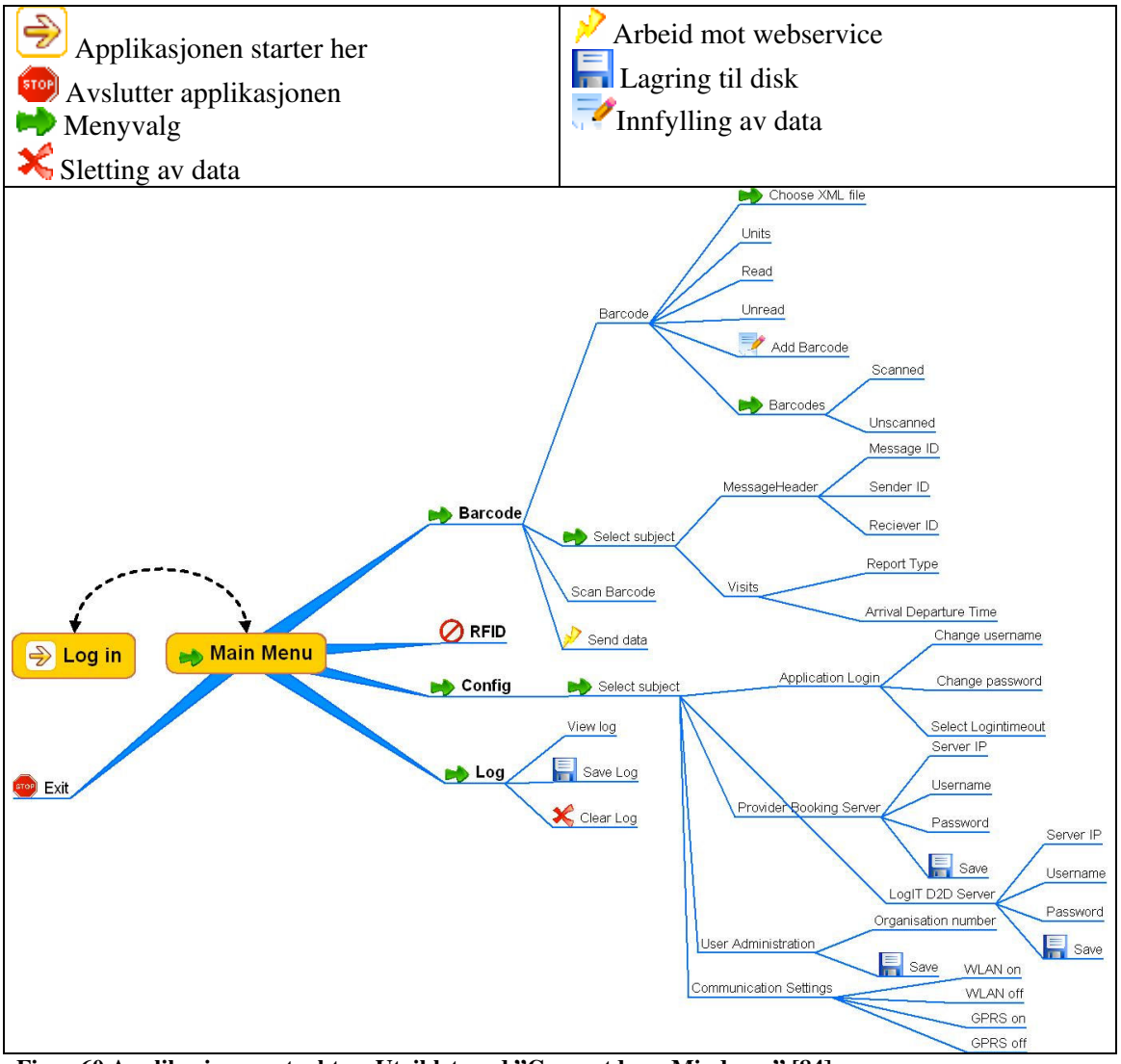
- **Barcode:** Her føres du til skjermbildet for registrering av strekkoder. Her får du tilgang til alt som er nødvendig for innlesing og håndtering av strekkoder. En undermeny kan åpne sidene ”MessageHeader” og ”Visits” som begge inneholder muligheter for å manipulere bestemte data i XML filen.

- **RFID:** I en fremtidig løsning vil sannsynligvis dette valget innebære mye likt til Barcode. I utviklingen av denne prototypen har RFID ikke vært prioritert å implementere.

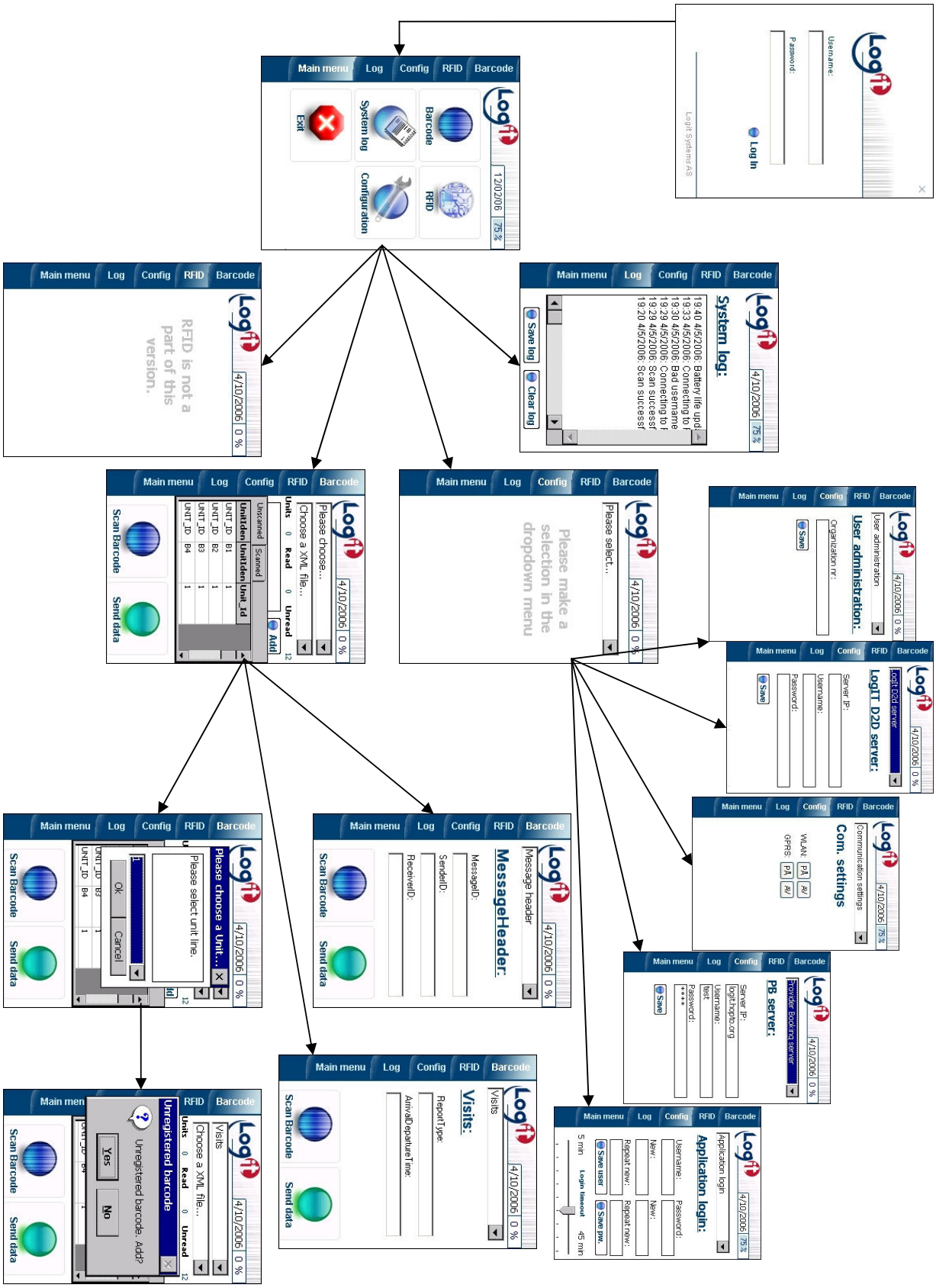
- **Config:** Under config får man tilgang til å utføre en del administrative endringer. Det første man får tilgang til er en dropdown meny med fem undervalg. Her er det lagt opp for å kunne utvide med ytterligere alternativer. Valgene som er der i dag er ”Communication settings”, ”Application Login”, ”Provider Booking Server”, ”Logit D2D server” og ”User Administration”.

- **Log:** Tekstvinduet i logen oppdateres kontinuerlig. Her kan man få en oversikt over hvilke oppgaver som er blitt utført. Logen kan om ønskelig tømmes eller lagres til fil.

- **Exit:** Ønsker man å avslutte applikasjonen fullstendig kan man bruke exit knappen. Man vil da ende opp på vanlig skrivebord i Windows CE.



Figur 60 Applikasjonens struktur. Utviklet med ”Conseptdraw Mindmap” [84].



Figur 61 Kart over applikasjonens skjermbilder.

6.3.4 Usabilitytesting

LogIT Systems AS har utlyst Masteroppgaven for utvikling av en prototyp for innlesing av AIDC på håndholdte enheter. Denne løsningen er ment for sluttbrukere og må derfor også kunne brukes av disse. Som utvikler er det fort å glemme brukeren og hvordan brukeren ser på produktet. En knapp eller løsning som er det mest innlysende og selvfølgelige i utviklingsøyeblikket kan være totalt misvisende eller helt kryptisk for en utenforstående. Som en del av utviklingen ønsker vi derfor å ha med en usabilitytest for å forsikre oss om at resultatet er noe en bruker kan håndtere.

Fra tidligere prosjekter har vi erfart at usabilitytesting er en svært effektiv måte å forbedre en applikasjon på. Den gangen utførte vi en usabilitytest på en nettbutikk i faget IKT-503 Koordinerings og samarbeidsteknologi ved HiA. Vi har valgt å benytte oss av fremgangsmåtene og teoriene vi kom frem til i vår tidligere rapport. For at ikke dette skal bli en for omfattende del av denne rapporten har vi her valgt å fokusere kort på hva en slik test innebærer og resultatene av denne. Vi har også bearbeidet mye av teorien rundt temaet fra vår tidligere rapport og lagt dette med som vedlegg [v3]. Til slutt har vi gjort noen tanker rundt usefulness, det om applikasjonen presenterer de funksjoner som er nødvendige.

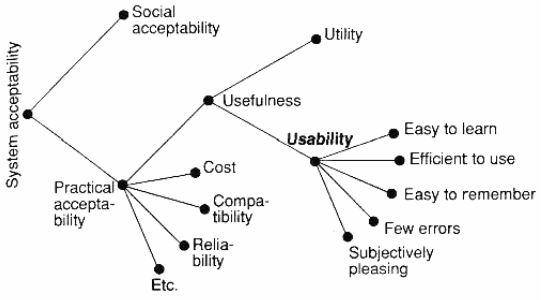
Usability testing

Det er mulig å teste brukervennlighet ved å gjøre såkalte usability tester. Meningen med usability testing er å finne problemer med et system, for å kunne gi anbefalinger til endringer for så å bedre bruksverdien til produktet gjennom en utviklingsfase. Usability kan undersøkes med en test på om produktet når noen forhåndsbestemte mål, målene blir oftest satt av oppdragsgiver. Det finnes i dag en rekke selskaper, for eksempel Usability Sciences Corporation [85], som har spesialisert seg på å utføre usabilitytester. Selv om usability testingen avslører feil eller mangler med et produkt så er det ikke sikkert dette blir utbedret med det første. Kunnskapen kan komme til nytte senere i andre systemer eller i nye versjoner av systemet. Testen kan bli utført av en typisk bruker, men med store systemer er det viktig at brukeren ikke trenger å teste hele systemet på en gang. Da kan brukeren gå lei og resultatene kan bli misvisende. Testen skal være av programmet, for en usability test er ikke en kunnskapstest for brukeren. Testingen bør skje gjennom hele utviklingen av systemet. En bruker kan lett komme til et problem som utvikleren aldri tenkte seg eksisterte og på denne måten kan de bli rettet før det får alvorligere konsekvenser. Slik systematisk testing er dessverre tidkrevende så det blir dessverre utført altfor sjeldent.

Kort teori:

Usability er en del av det å godta et system, en del av det store feltet acceptability. Usability kan deles inn i 5 deler: Lett å lære, effektivitet i bruk, lett å huske, få feil og brukertilfredshet. I rapportens vedlegg har vi under ”Usability” forklart de ulike feltene nærmere.

Tester fra Tom Landauer [86] og Jakob Nielsen [19] viser at antall problemer funnet i en usability test med n personer kan uttrykkes med formelen: N(1-(1-L)^n)



Figur 62 Modell av delene som omhandler system acceptability. Jacob Nielsen 1993

N er det totale antallet usability-problemer i designen og L er gjennomsnittsantallet av oppdagede feil ved testing fra en enkelt person. Det viser seg at ved store tester vil L typisk ligge på 31%. Ved å bruke L=31% får man kurven under. Kurven viser at allerede ved 3 testpersoner vil man ha oppdaget nærmere 2/3 av alle problemene og jo flere brukere man tester desto mindre nytt oppdager man i forhold til antall testpersoner. Dette fordi man vil etter hvert se de samme feilene om og om igjen. Landauer og Nielsen foreslår å bruke tilgjengelige ressurser på flere tester med 5 deltakere fremfor en stor test med mange deltakere, nettopp på bakgrunn av denne kurven. Med hold i denne teorien samt de praktiske grunnene ved å skaffe mange testpersoner velger vi å sette et mål med 3-6 testpersoner.

Figur 63 Hentet fra http://www.useit.com/alertbox/20000319.html

Oppgavene:
Tidsperspektivet for bruken av applikasjonen vil variere ut fra arbeidsoppgavene. Vi antar likevel at en gjennomsnittlig registreringsøkt vil vare rundt 10-15 minutter. Vi ønsker derfor at testperioden ikke skal overskride dette. Et annet argument for å holde testen forholdsvis kort er bruken av frivillige deltakere. Testen vil også inneholde et kort intervju av deltakeren i etterkant for å avklare usikkerheter, samt en enkel rangering av hvordan de opplevde vanskeligheten av oppgavene. Dette vil gi en total testtid på ca 15min.

1. ***Fyll inn brukernavn og passord test/test. Logg deg inn i applikasjonen.***
   Her ønsker vi å gi testbrukeren en myk start. Ved å gi en meget enkel oppgave til å begynne med vil testbrukeren føle seg mye mer komfortabel med produktet. Utover dette er også oppgaven tatt med for å sjekke om dette virkelig er så enkelt som forventet.

2. ***Bruk 10-20 sekunder på å gjøre deg kjent med de ulike elementene i skjermbildet.***
   Fortsatt gir vi brukeren tid til å gjøre seg komfortabel med produktet, samtidig som vi får mulighet til å observere hvilke fremgangsmåter brukeren velger.

3. ***Hent inn et oppdrag fra din "arbeidsgiver", slik at du vil være klar til registrering av gods.***
   Her begynner utfordringene for brukeren. Han vet ingenting fra utgangspunktet om hvordan dette gjøres. Formålet med oppgaven er å se om testpersonen opplever plasseringen som logisk. Om han klarer å finne frem og utføre oppgaven korrekt.

4. ***Les inn, eller fyll inn en hvilken som helst strekkode.***
   Dette er hva hele applikasjonen dreier seg om. Det er derfor viktig at dette kan utføres på en enkel og logisk måte for brukeren. Vi ønsker å se om brukeren er tilfreds med slik det er lagt opp på applikasjonen i dag.

5. ***Send inn dataene til sentral server.***
   Dette er en enkel oppgave om man kan den. Den krever kun ett trykk. Her ønsker vi å se om knappen er logisk plassert. Om brukeren nøler eller er usikker på informasjonstekst og tilbakemelding før og etter overføring av fil.

6. ***Endre tiden det tar for applikasjonen å bli logget ut om den er innaktiv.***
   Dette er en konfigurasjons brukeren selv skal kunne justere. For å få tilgang til dette må man inn to nivåer i menystrukturen. Vi ønsker å se om brukeren finner frem til dette, og om brukeren føler plasseringen logisk.

7. ***Kontroller i loggen at endringene er blitt utført.***
   Dette skal i utgangspunktet være en meget enkel oppgave. Men en log kan i mange tilfeller være kryptisk å lese. Det viktigste her er at brukeren finner frem til loggen. Skal den benyttes til feilsøking eller lignende vil det gjerne være erfarne brukere som skal kikke nærmere på den.

8. ***Er det noe du savner? Funksjoner du føler denne burde hatt, men som du ikke finner igjen på knapper eller i menyer?***
   Her gir vi brukeren muligheten til å tenke over hva han har gjort og hvordan han opplevde det. Vi ønsker at brukeren skal opplyse om eventuelle feil eller mangler de som bruker opplever.

Testresultater:
Når vi nå har utført brukervennlighetstesten er det viktig å behandle dataene riktig. Ved å systematisere resultatene sitter man igjen med et klarere bilde, og ser mer konkret hvor og hva problemene er. Det første vi gjorde når testene var ferdige var å gå gjennom notatene og lage oss en oversikt over hvilke oppgaver brukerne hadde problemer med. Tabellen viser de ulike testpersonene til venstre, mens hver kolonne bortover representerer en oppgave. I krysspunktene her presenteres antallet problemer den enkelte bruker har opplevd på den bestemte oppgaven.

Tabell 12 Problemer oppdaget av bruker pr oppgave

Problemer

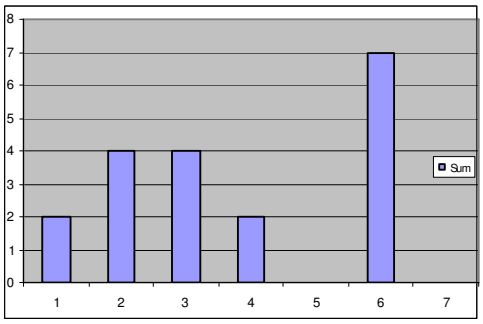
Oppgave	1	2	3	4	5	6	7
Testperson 1	0	1	2	1	0	2	0
Testperson 2	1	2	1	1	0	2	0
Testperson 3	1	0	0	0	0	1	0
Testperson 4	0	1	1	0	0	2	0
Sum	**2**	**4**	**4**	**2**	**0**	**7**	**0**

Selv om flere brukere har opplevd problemer på samme oppgave, så er det ikke dermed sagt at de har hatt samme problem. De ulike problemene blir derfor presentert her med en kort setning hver. Foran setningen betyr det første tallet nr. på testperson, og det andre oppgave.

1-2 Føler at Main Menu burde vært øverst til venstre, ikke nederst
1-3 Fant ingen logiske valg
1-3 Forstår ikke meningen av teksten.
1-4 Usikker på om strekkoden ble lagt til i lista
2-1 Er vant til å kunne åpne virtuelt tastatur nederst i skjermbildet.

2-2 Savnet litt en "X" eller OK knapp i hjørnet av de ulike vinduene for å lukke.
2-2 Var litt usikker på om tab linja og menyen var de samme i starten
2-3 Ble litt usikker på hva dataene som ble presentert betyr.
2-4 Trodde tekstfeltet måtte markeres ved innlesing
2-6 Slet med å finne riktig valg i menyen
2-6 Savnet en OK eller en eller annen bekreftelse på at endringen ble lagret.
3-1 Trodde Enter i det virtuelle tastaturet kunne brukes når man logget inn
3-6 Valgte feil i meny. I tvil om hvilket valg som var riktig.
4-2 RFID knappen i hovedmenyen fungerte ikke
4-3 Trykket først på den øverste boksen
4-6 Slet med å finne riktig meny
4-6 Hadde vanskeligheter med å finne ut hvor det kunne justeres

Som vi ser er det svært ulikt hvilke oppgaver brukerne har hatt problemer med. Oppgave 5 og 7 er det ingen som har opplevd betydelige problemer, og vi vil derfor ikke se nærmere på disse. Oppgave 1, 2, 3, 4 og 6 ble det derimot avdekket noen vanskeligheter. Oppgave 6 var verst med hele syv vanskeligheter oppdaget, så her er det helt klart problemer som kan forbedres. Heretter kom oppgave 2 og 3 med fire vanskeligheter hver, og 1 og 4 med to. Vi har gruppert vanskelighetene for å se hvilke som gjentok seg flest ganger. De som går oftest igjen vil være de viktigste å gjøre noe med for å bedre applikasjonens funksjonalitet.



Figur 64 Grafisk visning av problemer oppdaget av bruker pr oppgave

4	Sleit med å finne riktig valg i menyen
2	Usikker på om strekkoden ble lagt til i lista
2	Savnet litt en "X" eller OK knapp i hjørnet av de ulike vinduene for å lukke.
2	Fant ingen logiske valg for hvor man skulle "hente oppdrag"
1	Trodde tekstfeltet måtte markeres ved innlesing
1	Trodde enter i det virtuelle tastaturet kunne brukes når man logget inn
1	RFID knappen i hovedmenyen fungerte ikke
1	Main menu bør være øverst til venstre
1	Hadde vanskeligheter med å finne ut hvor timeout kunne justeres
1	Er vant til å kunne åpne virtuelt tastatur nederst i skjermbildet.
1	Var i starten litt usikker på om tab linja og menyen var de samme

Vi vil her beskrive nærmere de problemene mer enn en testbruker støtte på:

4 Slet med å finne riktig valg i menyen

Dette var et problem som gjentok seg for hver eneste testbruker. Problemet oppsto da de skulle utføre valg i dropdown menyen på config siden. Alternativene i denne menyen var ikke navngitt slik at brukerne forsto hva som skulle velges for å komme dit de ville.

2 Usikker på om strekkoden ble lagt til i lista

Ved innlesing av strekkode blir denne lagt til i listen ”Read”, mens man fortsatt så ”Unread” lista. Dette førte til at et par av brukerne var usikre på om innlesingen var vellykket.

2 Savnet litt en "X" eller OK knapp i hjørnet av de ulike vinduene for å lukke.

Brukerne som tidligere var kjent med bruken av håndholdte enheter savnet stadig en ”X” eller ”ok” knapp øverst i høyre hjørne for å bekrefte og/eller lukke et vindu. Ikke nødvendigvis for det var nødvendig, men fordi dette var noe de var vant til.

2 Fant ingen logiske valg for hvor man skulle "hente oppdrag"

Når brukerne ble bedt om å hente oppdrag fra sentral server oppsto det i to tilfeller problemer. Dette skyltes nok at teksten var mer spesifikk en nødvendig. Her sto det ”Hente XML fil”, noe brukere som ikke er datakyndige naturlig nok ikke vet hva er.

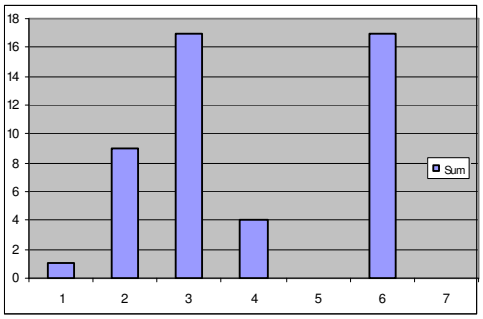
Etter testen lot vi også brukerne gradere de ulike oppgavene etter hvordan de opplevde vanskelighetsgraden. Her er 0 enkelt og 9 vanskeligst.

Tabell 13 Brukernes rangering av vanskelighetsgrad pr. oppgave

Vanskelighetsgrad

Oppgave	1	2	3	4	5	6	7
Testperson 1	0	2	9	2	0	2	0
Testperson 2	0	4	3	1	0	4	0
Testperson 3	1	1	0	1	0	2	0
Testperson 4	0	2	5	0	0	9	0
Sum	**1**	**9**	**17**	**4**	**0**	**17**	**0**

Sammenligninger mellom denne vanskelighetsgraderingen og statistikken for vanskeligheter oppdaget viser store likheter. Likevel gir denne statistikken en viktig pekepinn på hvor betydelige de oppdagede vanskelighetene var. Om en bruker støter på en vanskelighet, men likevel graderer den som svært enkelt, så er det ikke sikkert denne vanskeligheten nødvendigvis bør prioriteres forbedret.



Figur 65 Grafisk visning av brukernes rangering av vanskelighetsgrad pr. oppgave

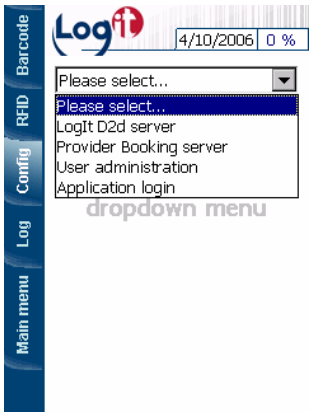
Statistikken viser også her at oppgave 5 og 7 ikke bød på noen problemer. Oppgave 1 viser derimot et avvik mot problemstatistikken. Her graderes dette som et svært lite problem. Oppgave 6 graderes som forventet som svært vanskelig, men noe overraskende er oppgave 3 gradert like vanskelig. Dette viser at brukerne la større vekt på problemene som her ble oppdaget.

Som en slags siste oppgave lot vi testbrukeren selv kommentere om det var noe de mente burde vært endret, eller om de savnet noe. I intervjuet etter testen pratet vi om oppgavene de hadde utført, og prøvde sammen å komme frem til mulige forbedringer. Utover det vi tidligere i testen har kommet frem til, kom vi frem til fem punkter som muligens kunne forbedres.

1. Kunne muligens gått rett til Barcode ved innlogging
2. Knappen kunne kanskje hett "Start barcode project", ikke bare Barcode
3. Muligens litt komplisert for en lastebilsjåfør/bryggesjauer. Forenkle?
4. Kunne vært samme type ikonmeny på configsiden som på hovedsiden
5. Ved scanning bør "scanned" taben vises istedenfor "unscanned" taben

Muligheter for forbedringer:
En applikasjonsutvikler kan aldri forhindre alle problemer. Alle brukere tenker på sin måte, og det vil alltid være noe som på en eller annen måte kan forvirre en bruker. Det er derfor i utgangspunktet viktig å ta fatt på det som brukerne oppfattet som den største hindringen, og det som flest brukere hadde problemer med.

- Menynavn: Alle brukerne hadde problemer med å velge riktig i dropdownmenyen på config sidene. Her bør det absolutt byttes ut til mer forståelige navn for brukeren, eller finne en mer fungerende menyløsning. Vi tror bedre navngiving vil løse problemet. I prosessen kan man gjerne ta med de aktuelle brukerne for å vurdere hvilke navn som kan være passende.

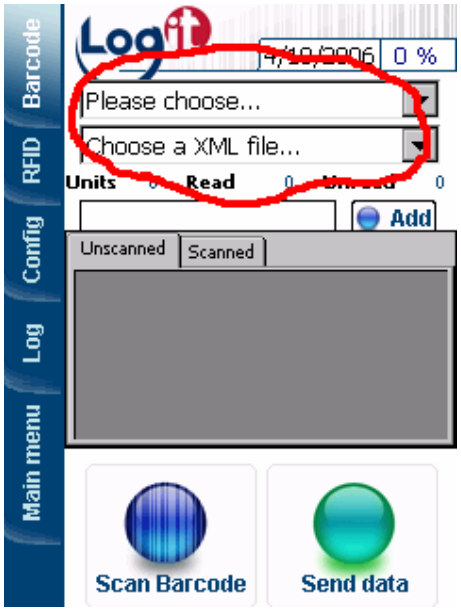


Figur 66 Dropdownmenu i config

- Strekkode visning: For å unngå forvirring bør nye innleste strekkoder vise mer igjen for brukeren. En begynnelse kan her være å skrifte fokus til ”Read” taben som viser de leste strekkodene. Videre kan en mulighet være å markere den nyeste strekkoden med farge og fokus.

- Lukking: Flere brukere savnet en ”X” eller ”ok” knapp øverst i hjørnet, f.eks for å komme tilbake til hovedmeny. En slik knapp kan enkelt legges inn i applikasjonsløsningen. Dette, hvis de kjenner seg igjen i forhold til andre applikasjoner, kan gjøre brukerne mer komfortable med bruken av også denne applikasjonen.

- Hente oppdrag: Dette problemet kan kanskje være vanskelig å løse kun i applikasjonen. For at brukeren skal forstå hvordan det hele fungerer behøver han en liten opplæring. I forbindelse med dette vil bruker også lære hvordan selve hentingen av oppdrag fungerer. Likevel ser vi mulighet for forbedringer her. Menyen man velger dette i bør ha en mer folkelig tekst som beskriver hva som blir utført. Kanskje også en egen knapp for å aktivere hentingen, istedenfor at dette skjer automatisk ved valg i dropdownmenyen. Kombinasjonen av dropdowmeny for å velge side, og for å velge oppdrag kan kanskje også forbedres(merket rødt).



Figur 67 Skjermbildet for innlesing av strekkode

Videre er det blitt oppdaget en rekke småproblemer som kun enkeltpersoner har oppdaget. Alle disse tilfellene kan trolig også forbedres, men vil ikke være av samme prioritet. Her må det gjøres en liten avveining rundt hvor store ressurser man vil bruke

på utvikling av brukergrensesnitt. Det å bestemme seg for å forbedre kontinuerlig til alle brukerproblemer er løst vil ikke la seg gjøre. I mange tilfeller ser en bruker svaret i det en annen ser som problem. En mellomting eller et annet alternativ vil alltid være en mulighet, men ingen løsning kan sikre mot alle brukerfeil.

Vi vil derfor i første omgang anbefale å utbedre de problemer som er oppdaget av minst to brukere. Når dette er gjort kan man kikke på de andre. Noen vil kunne løses meget enkelt og raskt, og bør da absolutt utføres. Direkte feil bør utbedres i alle tilfeller. Et eksempel på dette er: ”RFID knappen i hovedmenyen fungerte ikke.” Problemer som ikke nødvendigvis bør gjøres noe med er f.eks: ”Main menu bør være øverst til venstre”. Her er det en enkelt brukers mening om en enkelt utviklers avgjørelse. Problemet bør derfor tas opp med flere. F.eks en avstemning blant testpersoner. På denne måten vil flertallets mening gi en god pekepin.

Når alle forbedringer er utført er det på tide med usability test nummer 2. I denne skal fire nye, men mest mulig tilsvarende brukere utføre de samme oppgavene. Man kan måle hvor vellykket forbedringene var med en sammenligning av første og andre test. I noen tilfeller kan det vise seg at det man trodde var en forbedring, overhodet ikke var det. I de fleste tilfeller vil likevel den andre brukertesten avsløre at det er gjort forbedringer, men at kanskje andre problemer oppstår, eller bare får større fokus enn tidligere. Ut fra dette resultatet bør utviklerne avgjøre om det er behov for en tredje test.

Usefulness
I mange tilfeller faller usefulness i skyggen av usabilitytesting som egentlig kun er et underfelt under punktet usefulness. Ved usability snakker vi om brukervennlighet, og det om en bruker forstår hensikten og sammenhengen mellom elementene i det skjermbildet de blir presentert for. Usefulness er mer generelt for brukbarhet. Dette innebærer også at applikasjonen må inneholde de funksjoner som trengs for å utføre oppgavene den er ment for. Et fint ytre som er enkelt å bruke har ingen hensikt om det ikke kan utføre arbeidet det er ment for.

Det siste spørsmålet i brukervennlighetstesten var et ledd for å bli kjent med eventuelle mangler eller overflødige muligheter i applikasjonen.

9. ***Er det noe du savner? Funksjoner du føler denne burde hatt, men som du ikke finner igjen på knapper eller i menyer?***

Ved intervju og diskusjon med brukerne rundt produktet ble det blant annet diskutert om brukeren behøver tilgang til mer enn selve innlesningsskjermbildet ”scanning”. At funksjonaliteten i de ytterligere knappene var nødvendig var det ingen tvil om. Men behøver alle brukere tilgang til dem, eller bare noen overordnede? Dette er et tema som kan være verdt å diskuteres nærmere i fremtidig utvikling av applikasjonen. Utover dette kom det ikke frem noen ytterligere ønsker om funksjonalitet som manglet. For å studere dette nærmere vil det nok være nyttig å la testbrukerne bruke systemet over litt lengre tid.

7 Implementasjon

Implementasjonen mot Logit D2D har ikke foregått uten problemer. Under utviklingen har flere problemer oppstått, noe som har vært tidkrevende og frustrerende. Mye av problemene grunner i at LogIT Systems fra utgangspunktet ikke selv var helt klar over hva de var ute etter. Dette er forståelig og vanlig ved utviklingen av nye produkter, men selvfølgelig krevende for utviklerne. Men etter hvert som prosjektet har pågått har vi fått klarhet i hvordan systemet til slutt skal implementeres for å fungere korrekt.

7.1 Implementasjon på håndholdt enhet

Store deler av programvareutviklingen har foregått på emulator. Etter hvert som utstyret har dukket opp og software har blitt ferdigstilt har dette blitt implementert i softwareløsningen på den mobile enheten M3 fra Mobile Compia. I teorien skal det som fungerer på emulatoren også fungere på en mobil enhet, men det er ikke alltid slik i praksis. Ofte er noen elementer unike for de bestemte enhetene. I vårt tilfelle hadde M3en en rekke funksjoner utover det vanlige. Hos Mobile Compia [13] lå SDK tilgjengelig for nedlasting. Ved å ta dette i bruk i vår software fikk vi mange ny muligheter, men mistet muligheten til å kjøre denne koden på emulator uten å gjøre en del endringer.

Midt i prosjektperioden støtte vi på et enormt problem. Kommunikasjonen mellom utviklingsverktøyet Visual Studio .NET og M3en sluttet å fungere. Problemet førte til at arbeidet ble forsinket med en hel uke. Verken vår norske leverandør Strekmenn AS eller Mobile Compia hadde noen løsning på problemet. Eneste forslag var å foreta en hard reset og installere operativsystemet på nytt. Dette løste ikke problemet. Vi forsøkte også å installere Visual Studio .NET, .NET Compact Framework, OpenNETCF Smart Device Framework 1.4 og ActiveSync på nytt uten hell.

Etter mye feilsøking og søking på diverse fora viste det seg at flere hadde hatt liknende problemer. Det viste seg at feilen oppstod når vi oppgraderte til nyeste versjon av programmet ActiveSync som gjør kommunikasjon mellom PC og M3 mulig. Det er en feil som oppstår i visse tilfeller, slik at blant annet enkelte verdier i PCens register blir endret og følgelig skaper problemer med kommunikasjonen mellom M3 og PC.

Det ble skissert flere mulige løsninger [87] - [89] på problemet. Det som til slutt løste vårt problem var:

- Avinstallere ActiveSync.
- Kjør filen ProxyPorts.reg som finnes under C:\Program Files\Microsoft Visual Studio .NET 2003\CompactFrameworkSDK\WinCE Utilities[90]. Denne legger inn de rikige registerverdiene igjen, som ble ødelagt da vi installerte nyeste versjon av ActiveSync tidligere.
- Installere ActiveSync.
- Opprette en gjestekonto for ActiveSync, slik at M3 og PC kan kommunisere.
- Kopiere DelDesktopCryptoKeys.exe til M3, som finnes i samme katalogen som nevnes over, og kjøre denne.

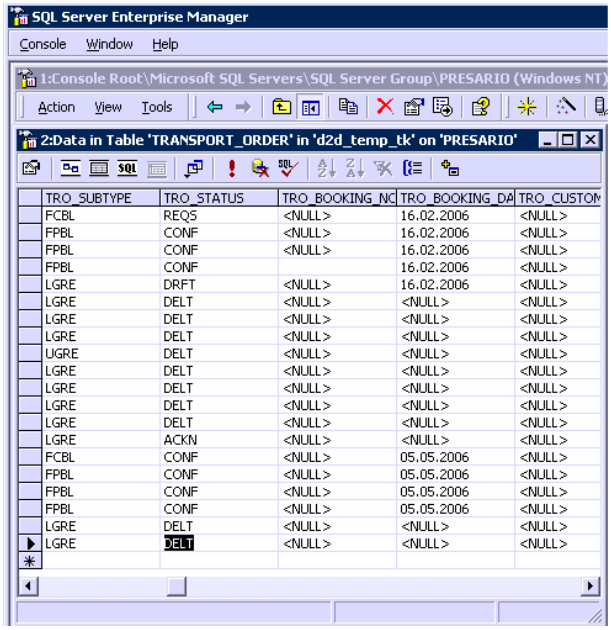
Utover dette har implementasjonen på den håndholdte enheten fungert forholdsvis greit.

7.2 Implementasjon på server

Under utviklingen har vi ikke hatt tilgang til en aktiv server. Dette grunnet at eventuelle feil kunne fått alvorlige følger. For å kompensere for dette har det blitt satt opp en utviklingsserver av LogIT Systems med samme software som på deres operative servere. Arbeidet mot denne har ikke vært uproblematisk.

Serveren inneholder en software serverløsning som heter Logit D2D. Denne skal ta imot XML data i form av en string via en webservice. I forbindelse med dette har det vært to problemer som har vanskeliggjort arbeidet for oss. Serveren har i mange tilfeller ikke klart å håndtere feilinformasjon, noe som har ført til at den har sluttet å fungere. Dette er muligens grunnet lite kontrollering, feilhåndtering og feilkorreksjon av inndata på serversiden. Tilfeller hvor serveren har sluttet å fungere har vært svært tidkrevende. En restart av server eller programvare har sjeldent vært nok. I tilfeller har vi vært nøtt til å sette opp hele databasen og en rekke konfigurasjoner på serveren på nytt. Dette i samarbeid med vår kontaktperson hos LogIT Systems, som har vært meget hjelpsom.

Webservicen viste seg også å være en utfordring. Her var problemet at .net 1.1 ikke klarte å kommunisere med Javabaserte webservicer, slik som den på serveren. Dette kunne imidlertid .net rammeverket versjon 2.0. Men .net 2.0 kunne ikke kjøres på operativsystemet på den mobile enheten M3. Løsningen ble derfor å installere .net 2.0 rammeverket på serveren. Her kunne vi kjøre en webservice som sendte inngående data videre til java webservicen. På denne måten kunne vi sende data fra den mobile enheten med .net 1.1 til serveren, og dermed direkte inn i datasystemet til Logit D2D.



TRO_SUBTYPE	TRO_STATUS	TRO_BOOKING_NC	TRO_BOOKING_DA	TRO_CUSTOM
FCBL	REQS	<NULL>	16.02.2006	<NULL>
FPBL	CONF	<NULL>	16.02.2006	<NULL>
FPBL	CONF	<NULL>	16.02.2006	<NULL>
FPBL	CONF		16.02.2006	<NULL>
LGRE	DRFT	<NULL>	16.02.2006	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>
UGRE	DELT	<NULL>	<NULL>	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>
LGRE	ACKN	<NULL>	<NULL>	<NULL>
FCBL	CONF	<NULL>	05.05.2006	<NULL>
FPBL	CONF	<NULL>	05.05.2006	<NULL>
FPBL	CONF	<NULL>	05.05.2006	<NULL>
FPBL	CONF	<NULL>	05.05.2006	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>
LGRE	DELT	<NULL>	<NULL>	<NULL>

Figur 68 Skjermbildet fra manuell endring i databasen

Systemet er ikke laget for å motta samme meldingen om og om igjen. Under utvikling og testing har vi likevel vært nødt til å sendt samme meldingen fra M3en flere ganger. For at dette skulle kunne mottas av D2D serveren måtte vi manuelt inn i databasen og sette ACKN til DELT i "TRO_STATUS" kolonnen i "TRANSPORT_ORDER" tabellen. Dette måtte gjøres hver eneste gang en melding var sendt, for at serveren skulle kunne godta en ny.

Brukeren skulle også kunne hente oppdrag fra serveren. Når et oppdrag er lagt inn i D2D systemet så genereres en XML fil som plasseres i en bestemt mappe. For å gi de mobile brukerne tilgang til dette installerte vi en FTP server på serveren. Denne ga tilgang til den bestemte mappen med oppdrag. Ved å arbeide mot denne FTP serveren kan da applikasjonen på den mobile enheten kontrollere tilgjengelige oppdrag og laste ned nødvendige etter behov.

8 Resultat

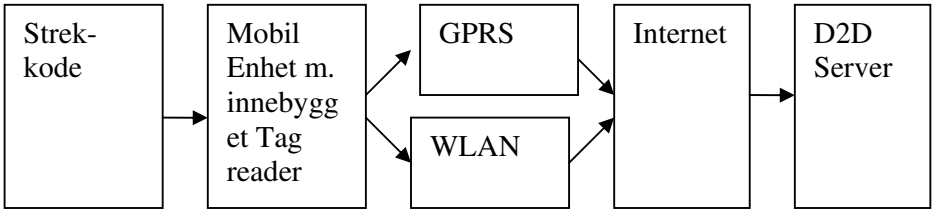
Resultatet vi sitter igjen med etter prosjektperioden er i all hovedsak to ting. Rapporten, og en prototyp. Rapporten dekker utfyllende de teoretiske spørsmål som ble stilt fra oppdragsgiver igjennom oppgavedefinisjonen. Videre dekker den planlegging, utvikling og resultat av den håndholdte enheten.

Figur 69 Det ferdige demoproduktet

Den håndholdte enheten har blitt et produkt av planlegging og utvikling over et semester. Etter fem måneder sitter vi igjen med en prototyp ferdig utviklet for demobruk. Denne består av det vi har kommet frem til er det beste alternativet av utstyr for å dekke dette formålet, og som finnes på markedet i dag.

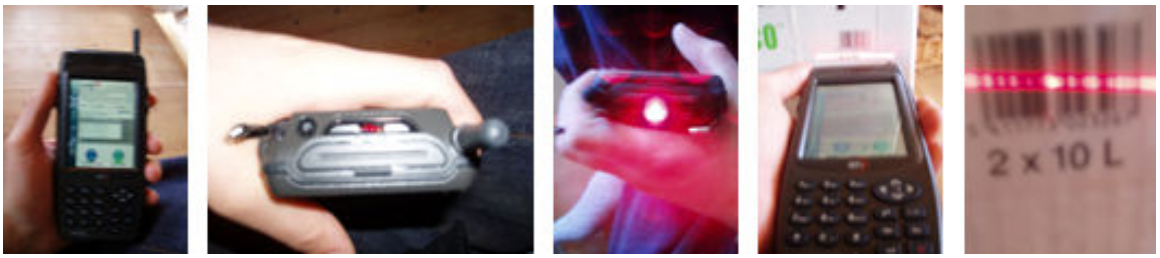
Prototypen er håndholdt og praktisk, og kan kommunisere via trådløst nett. Det er en robust og enkel enhet som skal tåle hard bruk på f.eks. lagre, omlastningsterminaler, togstasjoner og brygger.

Systemet håndterer i dag strekkoder som identifikasjonsmerker. Disse leses inn til den håndholdte enheten. Inndataene sammenlignes med data som er hentet ned fra Logit D2D serverer. De håndteres og settes sammen etter et XML oppsett fastsatt av Logit D2D. Når filene er klare for videresending utføres dette via enten WLAN eller GPRS. Filene sendes så over Internet til den sentrale D2D serveren hvor de blir behandlet i deres interne system.



Figur 70 Det ferdige systemet

Følgende bilder viser noen eksempler av M3en under testbruk. Her ser du først applikasjonen kjørende på enheten. Neste bilde viser lesehodet plassert foran. Ved å trykke på scan knappen på høyre side eller i applikasjonen starter leseren. M3en pekes så mot strekkoden som blir innlest til applikasjonen for videre håndtering.



Figur 71 Bilder fra M3en i bruk

Figur 72 Enheten i bruk ved skanning av strekkode

Den mobile enheten som ble benyttet var en M3 fra Mobile Compia. Denne har vi utviklet programvare på for å utføre de ulike oppgavene. Det hele er presentert med et gjennomarbeidet grensesnitt for enkel, oversiktelig og effektiv bruk.

Vi føler resultatet i dag besvarer de oppgaver vi er blitt gitt. Vi går nærmere inn på prosjektets deloppgaver og hvordan disse er blitt utført i kapittel ”8.2 Drøfting”, etterfulgt av vår konklusjon i kapittel ”9 Konklusjon”.

Som et siste resultat kan vi jo også nevne at denne prosjektperioden har latt oss utvikle oss innen programmering, utvikling og systemdesign. Vi har fått erfaring innen en rekke aktuelle teknologier som vil være nyttige for oss å ha med videre.

8.1 Videre muligheter

Slik vi ser det etter denne prosjektperioden, så har dette systemet en rekke fremtidige muligheter. Vårt begrensede tidsrom har ikke gjort det mulig for oss å utvikle alle de løsninger som kunne være aktuelle. Vi har fokusert på det som var spesifisert i oppgavebeskrivelsen. Tankene våre har likevel streifet andre muligheter som vi vil nevne her.

RFID:
Dette anbefaler vi å prioritere høyt videre. RFID er helt klart noe som kommer, og jo fortere en løsning er klar, jo lettere vil det nok være å få markedsandeler med et slikt produkt. RFID kan sannsynligvis implementeres forholdsvis enkelt. Vi har forsøkt å legge til rette for dette under vår utvikling, men uten å vite mer eksakt hva det vil innebære, så er det vanskelig å gjøre så mye rundt det nå. Det er viktig å huske på at Mobile Compia og deres norske leverandør Strekmenn AS leverer orginale RFID moduler til M3en.

Posisjon: De siste årene har salget av GPS mottakere skutt i været. M3en kan leveres med GPS innebygget. Dette gir helt nye muligheter. M3en kan i første omgang også benyttes som en kjøreguide. Om dette er nyttig vil avhenge litt av hvem brukerne av utstyret er. I dag er den mest ment for registrering av gods f.eks ved lasting eller lossing av skip. Skulle bruksområdet utvides til for eksempel ved utleveringer av varer, så kan en slik løsning vise vei for budbringeren. GPSen kunne også benyttes i godsrapportene hvor posisjon skal oppgis. Dette ville være mer nøyaktig enn havn eller by lokasjon som er benyttet i dag.

Push: Dagens løsning er laget slik at brukeren selv må sjekke mot en FTP server om det finnes nye oppdrag. M3en har støtte for telefoni og SMS. SMSen kan integreres inn i applikasjonen for å motta oppdrag. Med en slik løsning vil en arbeidsgiver kunne pushe på nye oppdrag til arbeiderne via SMS. Dette ville kunne effektivisere arbeidet i stor grad.

Signatur: Det lar seg gjøre å hente inn signaturer digitalt på M3en. Ved hjelp av en liten kodesnutt man for eksempel kan finne hos OpenNetCF, så kan signaturer som blir skrevet på touchskjermen lagres som bilde eller datafiler og behandles videre i systemet. Spesielt aktuelt ville dette være om man skal drive utlevering av varer slik som nevnt under overskriften ”Posisjon”. Underskrifter kan også være aktuelle i andre sammenhenger, så dette er relativt enkelt å implementere, men likevel en forholdsvis nyttig funksjon.

.NET Rammeverk: For at kommunikasjonen skulle kunne foregå mellom .Net framework 1.1, og Java webservicen, så måtte vi benytte en .Net 2.0 service som mellomledd. Dette mellomleddet vil være anbefalt å eliminere i fremtiden. Sannsynligvis vil det bli muligheter for dette i fremtiden.

Usabilitytesting: Under usability testen kom vi frem til en rekke mulige forbedringer. Ved fremtidig utvikling bør denne testen benyttes til å gjøre så mange brukervennlighetsforbedringer som mulig. De aller fleste problemene som ble nevnt her(6.3.4 Usabilitytesting) kan forbedres relativt enkelt. Vi har forsøkt å utføre noen av de mest fremtredende, men her er det fortsatt rom for en rekke forbedringer. Ved videre utvikling er det også anbefalt å utføre flere brukervennlighetstester.

Sikkerhet: Sikkerhet er i ethvert system viktig for at det skal fungere tilfredsstillende. For å beskytte sine data behøves derfor dette å gjennomgås fra ende til ende. Får gale personer tilgang til et system kan de fort skape mange komplikasjoner. Her hvor enheten er knyttet opp mot en sentral server med en rekke mer eller mindre sensitiv data, kan tilgang føre til totalt sammenbrudd. I og med at sikkerhet ikke har vært en del av vårt prosjekt har vi heller ikke jobbet spesielt med dette. Vi anbefaler derfor LogIT Systems å arbeide med dette under fremtidig utvikling av applikasjonen.

Terminologi: I og med at vi har arbeidet med utvikling av et produkt til en yrkesgruppe vi ikke tidligere har hatt noen spesielle forbindelser med, vil deres terminologi også være ukjent for oss. Vi har forsøkt å tilpasse ord og uttrykk så fornuftig og logisk det lar seg gjøre. Applikasjonen bør likevel sees over og bearbeides med menyvalg og knappetekster slik at den blir fullstendig tilpasset de tilsiktede brukeres terminologi. Dette kan være lurt å utføre i forbindelse med for eksempel en brukervennlighetstest, i fokusgrupper eller lignende.

GPRS konfigurasjon: I og med at GPRS kommunikasjonen ble ferdig helt mot sluten av prosjektperioden ble det ikke tid til utvikling av en konfigurasjonsløsning. Vi har i kapitlet ”6.2.11 GPRS” presentert et par alternativer til hvordan en slik løsning kan presenteres.

8.2 Drøfting

Arbeidet oss i mellom har gjennom hele prosjektperioden fungert meget godt. Vi har hele tiden klart å utfylle hverandre, slik at vi har fått dekket de behov det har vært for å ferdigstille en slik løsning. Ved å fordele arbeidsoppgavene oss i mellom har vi fått arbeidet med alle de deloppgaver oppgavedefinisjonen hadde å by på. Oppgavebeskrivelsen besto av en tekst hvor en rekke utfordringer var presentert. Vi har her forsøkt å plukke ut det vi har oppfattet som våre oppgaver, og beskrevet hvordan vi føler vi har besvart dem. Hver oppgave blir presentert med selve oppgave teksten. Deretter har vi fylt ut vår oppfattelse av om oppgaven er utført eller ikke, for så å beskrive situasjonen litt nærmere.

8.2.1 Krav fra oppgavedefinisjon:

- **Finne egnet enhet eller sammensetning av enheter**

  Opprinnelig Status: UTFØRT

  Det ble gjort et grundig forarbeid før valg av enhet ble utført. Vi begynte arbeidet med å gjøre oss en del vurderinger av hva vi var ute etter. Dette er presentert detaljert i kapittel ”3.3 Vurderinger før valg av enhet(er)”. Videre tok vi for oss de modellene vi fant på markedet i dag og vurderte de opp mot hverandre. Spesifikasjoner for noen av de mest relevante produktene er presentert i kapittel ”3.3.2 Utstyrs spesifikasjoner”. Til slutt ble det gjort en avgjørelse om innkjøp av utstyr. Dette utstyret er presentert i kapittel ”5 Valgt utstyr og teknologi”.

- **Utarbeide kravspesifikasjon**

  Status: UTFØRT

  I kapittel ” 3.3.1 Kravspesifikasjon” presenterer vi de krav vi etter våre vurderinger setter til utstyret som skal benyttes.

- **Velge teknologi**

  Status: UTFØRT

  Valg av teknologi har vært en stor utfordring. Feil valg kunne fort spolert hele oppgaven. Vi valgte derfor å gjøre et grundig forarbeid før noen avgjørelser ble tatt. Vi har valgt å beskrive denne prosessen i flere kapitler i rapporten. Kapittel ”3 Forarbeid og planlegging” omhandler i stor grad arbeidet vi utførte før selve utviklingen kunne starte. Her er underkapitlene ”3.3 Vurderinger før valg av enheter”, ”3.4 Vurdering før valg av utviklingsmiljø”, og ”3.5 Vurdering før valg av trådløs teknologi”. Nærmere beskrevet hvilket utstyr som til slutt ble valgt er presentert i kapittel ”5 Valgt utstyr og teknologi”.

- **Det skal designes og å utvikles programvare for enheten(e) for lesing, tolking, sending av data, og brukergrensesnitt for manuell inntasting av data**

  Status: UTFØRT

  Dette har vi lagt vekt på som selve kjernen i oppgaven, og utfordringen de fleste andre deloppgavene har hatt en forbindelse med. Applikasjonen er utviklet for å tilfredsstille disse kravene og skape et grensesnitt mest mulig klargjort for reell bruk.

- **Integrere og teste mot Logit D2D**

  Status: UTFØRT

  Det har i utviklingsperioden vært satt opp en Logit D2D testserver tilsvarende deres aktive servere. Utvikling og arbeid har blitt utført mot denne for å forsikre oss om at vår applikasjon fungerer mot deres systemer. Dette er presentert nærmere i kapitlene ”6.1 Server installasjon og oppsett” og ”7.2 Implementasjon på server”.

- **Skissere utvidelser til arkitekturen for å utvide det nåværende systemet for mobil datainnsamling og oversendelse til back-end systemet.**

  Status: UTFØRT

  I kapittel ”3.2 Kommunikasjonsstrukturen” går vi i detalj igjennom hvordan vi ser for oss utvidelsen av det eksisterende Logit D2D systemet. Vi kikker på hvilke elementer som bør og kan være med i en slik løsning, og etter hvert avgrenser det til hva vi tror vi kan utvikle i løpet av prosjektperioden.

- **En ”proof-of-concept” prototyp skal kunne demonstrere de tekniske mulighetene en slik løsning kan gi.**

  Status: UTFØRT

  Den utviklede prototyp kan benyttes som en demonstrasjonsenhet for de tekniske løsninger som er spesifisert. Dette kan leses nærmere på i kapittel ”8 Resultat”. Vi har også forsøkt å tilfredsstille noen ønsker utover oppgavedefinisjonen for å tilrettelegge resultatet for demonstrasjonsvisning. Se kapittel ”8.2.2 Ytterligere ønsker”.

- **Kartlegge andre mulige identifikasjonssystemer**

  Status: UTFØRT

  Vi har fordypet oss i dette temaet i kapittel ”4 Identifikasjonssystemer” hvor vi kikker på de systemer som finnes på markedet i dag, og har noen tanker om hvordan dette blir fremover.

- **Kartlegge hva slags data som lagres på slike identifikasjonsmerker i de forskjellige bransjene, og å finne ut av hvordan Logit D2D kan nyttegjøre seg den informasjonen**

  Status: UTFØRT

  Vi kommer inn på dette i kapittel ”4 Identifikasjonssystemer” hvor vi går inn på de ulike typene identifikasjonsmerker og hvordan disse fungerer. Mer i detalj kikker vi på temaet i kapittel ”4.6 Lagrede data”. Vi var i starten litt usikre på hva oppdragsgiver egentlig var ute etter her. Etter å ha diskutert dette nærmere i møter føler vi nå at rapporten omfatter og besvarer de temaer som var ønsket.

8.2.2 Ytterligere ønsker.

Oppdragsgiver hadde også noen ønsker utover oppgavebeskrivelsen. Dette er selvfølgelig naturlig da oppgavebeskrivelsen må tilpasses at arbeidet kun skal pågå i en relativt kort prosjektperiode. Vi tok oss likevel tid til å kikke nærmere på noen av ønskene som ble gitt. Følgende er en oversikt over disse ønskene, og beskrivelse av våre resultater.

- **Mottak av data fra server til håndholdte enhet**

  Status: UTFØRT

  Opprinnelig gikk oppgaven ut på å utvikle en applikasjon for innlesing og videresending av data. Videre om tiden skulle tillate det var det ønske om å kikke på muligheten for mottak av data fra server. Denne funksjonen ville bidra til å skape en mye bedre helhet i det endelige produktet. Dette var snakk om en forholdsvis omfattende ekstraoppgave. For at LogIT Systems skal få full nytte av sitt demo-produkt ordnet vi med denne muligheten i applikasjonen.

- **Utviklet webservice på webserver**

  Status: UTFØRT

  Grunnet kompatibilitets komplikasjoner oppstod et ønske om en webservice på server for å håndtere inndata. En webservice for mottak og videreføring av data til D2D server er blitt utviklet.

- **RFID**

  Status: IKKE UTFØRT

  Ved oppstart av oppgaven var ID teknologi uklart. At strekkode skulle bli den valgte ID teknologien ble raskt avklart da RFID ble valgt bort av oppdragsgiver. Ved kjøp av den håndholdte enheten bestemte de seg for ikke å inkludere RFID utstyr, og gjorde det dermed ikke mulig å utføre noe arbeid rundt dette.

- **GPRS**

  Status: UTFØRT

  Utover WLAN kommunikasjon har vi også utviklet mulighet for GPRS kommunikasjon mot Internet og sentral D2D server. Ved hjelp av dette vil enheten kunne benyttes stort sett overalt, de eneste krav er telefonforbindelse, og at enheten er utstyrt med et aktivt SIM kort.

- **Presentere dataene i datagrids etter deres standard**

  Status: DELVIS UTFØRT

  Under utviklingen kom det ønske fra oppdragsgiver om å presentere dataene i datagrid på den håndholdte enheten mest mulig identisk til deres opprinnelige systemer. Resultatet er at datene nå presenteres i datagrids, men ikke helt på samme måte som i deres opprinnelige systemer. CF gjør det dessverre svært arbeids- og tidkrevende å manipulere slike grids. I og med at oppgaven er forholdsvis enkel, men svært

tidkrevende har vi nedprioritert denne delen. Dette kan enkelt ferdigstilles på et senere tidspunkt.

- **Utføre usabilitytest på applikasjonen**

  Status: DELVIS UTFØRT

  Det har vært snakk om å utføre en usabilitytest på applikasjonen for å kunne presentere et best mulig resultat. Tiden har ikke tillatt å utføre en fullstendig ustabilitytest. Dette krever nemlig flere forsøk med unike testpersoner, noe som fort blir en tidkrevende operasjon. Vi har utført første del av testen med fire testpersoner, noe som påpeker vanskeligheter og potensial til forbedringer. Denne testen kan nå bygges videre på for å utvikle en best mulig applikasjon.

9 Konklusjon

Det skulle utvikles en trådløs registrerings og kommunikasjonsenhet. Et system som kan hente oppdrag fra server, og hvor AIDC merker kan leses inn. Her skal dataene taes med videre inn i applikasjonen for å bearbeides før de genereres til en statusfil som sendes tilbake til hovedserver.

Vel så viktig som å utvikle applikasjonen, var det at den skulle være brukervennlig. Farger design og struktur skulle settes opp fra bunnen av. Plassering av knapper og menystruktur er viktig, og det skal utføres en usabilitytest for å gjøre dette så godt som mulig. Brukervennlighet er en stor utfordring når man følger uttrykket: ”Alt en bruker kan gjøre feil, gjøres feil.”

Oppgaven ble gitt med informasjon om hva som var ønsket utviklet, ikke hvordan man ønsket det utviklet. Prosjektet var i stor grad en utviklings- og programmeringsoppgave, men bød også på spennende utfordringer som design, planlegging, og ikke minst det å tenke nytt. Fremgangsmåte for å løse oppgaven ble planlagt i forprosjektperioden, hvor vi også kom frem til å benytte C# på Windows CE operativsystem for å utvikle applikasjonen.

Oppdragsdataene kan nå hentes inn i XML format fra oppdragsserver, mens innleste ID informasjon tas imot som en string. XML dataene blir håndtert i et dataset som presenteres for brukeren i et datagrid. Når registreringene er utført kan PB fila sendes videre til server som XML via en webservice på serveren.

Vi har utviklet brukergrensesnitt og design på enheten for å lage en mest mulig brukervennlig versjon. Applikasjonen har også blitt brukervennlighetstestet med relevante testpersoner.

Det har blitt utført en del arbeid på serverside for å få systemet til å fungere som det skal. Oppgaven og ytterligere ønsker har gitt oss nok å sette fingrene i. Vi har forsøkt så godt vi kan å bidra til å tilfredsstille alt. Oppgavebeskrivelsen har selvfølgelig hatt høyeste prioritet, men vi har også fått løst noen av ekstraoppgavene.

Systemet er demotestet. Det er blitt utført vellykkede tester virtuelt fra emulator og fra håndholdt enhet mot testserver. Systemet leser inn, kan håndtere og videresende produkters identifikasjonsstrekkode. Vi sitter nå igjen med en demo-enhet som kan hente oppdrag, lese ID tager og sende resultatet til sentral server. Altså en fungerende prototyp som kan håndtere ende til ende kommunikasjon i systemet. Dette skulle tilfredsstille oppgavens tittel ”Utvikling av system for innlesing, bearbeiding og informasjonsutveksling mellom håndholdte enheter og Logit D2D”. Vi mener også å ha besvart oppgaveteksten, og dens underpunkter utfyllende.

Prosjektet har gitt oss et innblikk i hvordan det er å arbeide mot et mål, og mot en tidsfrist. Dette har gitt oss innblikk i hva vi kan forvente oss i arbeidslivet og fra fremtidige arbeidsgivere. Prosjektarbeidet har vært en sunn og spennende utfordring som har lært oss mer enn noen av oss hadde forventet.

Som prototyp vil produktet kreve ytterligere utvikling på vei mot en ferdig løsning. Vi har i kapittel ” 8.1 Videre muligheter” studert en del muligheter vi mener fremtidige utviklere kan ta fatt på. Kanskje kan vårt arbeid ha dannet grunnlaget for et nytt satsningsområde, og at prototypen kan bane vei for et etterhvert ferdig og innbringende produkt for LogIT Systems AS.

Vi håper og tror at vårt arbeid gjennom prosjektperioden, og resultatet vi sitter igjen med har svart til oppdragsgivers forventninger. Alt i alt føler vi at vi har besvart de spørsmål og oppgaver som var presentert i oppgavedefinisjonen, samt også å ha gjort vårt beste for å håndtere ytterligere ønsker.

Referanser

[1] LogIT Systems AS, sist lest Mai 2006, http://www.logit-systems.com

[2] Chalmers, A.F. (1999). What is this thing called Science?, Open University Press, Buckingham, UK (pp. 1-73).

[3] Grelland, H.H. (2004). En kort introduksjon til vitenskapsteori. Notat, Høgskolen i Agder, Grimstad, Norge.

[4] Glass, R.L., Vessey, I. and Ramesh, V. (2002). Research in software engineering: an analysis of the literature. *Information and Software Technology*, 44, 491-506.

[5] A. R. Hevner,S. T. March, J. Park and S. Ram, "Design Science in Information Systems Research", MIS Quarterly, p. 28, pp. 75-105, 2004

[6] E. Rubin og R. Yates, "Microsoft® .NET Compact Framework Kick Start", Sams Publishing, 2003

[7] D. Fox og J. Box, " Building Solutions with the Microsoft .NET Compact Framework: Architecture and Best Practices for Mobile Development", Addison Wesley, 2003

[8] J. Liberty og D. Hurwitz, "Programming ASP.NET", O'Reilly, 2003

[9] J. Liberty, "Programming C#", O'Reilly, 2003

[10] A. Troelsen, "C# and the .NET Platform", New York: Spring- Verlag, 2001

[11] L. Barfield, "Design for new media",Henry Ling , 2004

[12] Mobile compia, Your global partner, Mai 2006 http://www.mobilecompia.co.kr/en/

[13] Mobile Compia, "Data room > SDK", sist April 2006 http://www.mobilecompia.co.kr/en/bbs_list.asp?dbn=BBsd

[14] OpenNETCF.org, "The Premier .NET Compact Framework Shared Source Site", Mai 2006 http://www.opennetcf.org

[15] The Code Project, "Free Source Code and Tutorials", Mai 2006 http://www.codeproject.com/

[16] C# Corner, "C-Shart corner", Mai 2006 http://www.c-sharpcorner.com/

[17] LogIT System AS, " Norway Post Logistics", Mars 2006,

http://logit-systems.com/index.php?option=com_content&task=view&id=27&Itemid=45

[18] LogIT System AS, " Brostrøm - LogIT Sea is now in use", Mars 2006, http://logit-systems.com/index.php?option=com_content&task=view&id=37&Itemid=45

[19] J. Nielsen, Usability Engineering, 1993

[20] Tripod Data Ssystems, "TDS Recon: Overview", Mars 2006, http://www.tdsway.com/products/recon

[21] Department of Defense Test Method Standard, "MIL-STD-810F", Mars 2006 http://www.dtc.army.mil/navigator/

[22] Baracoda Inc., "Designd for applications where form factor, weight and autonomy are key", Februar 2006, http://www.baracoda.com/baracoda/products/p_23.html

[23] Tripod Data System, "TDS Recon: Overview", Mars 2006, http://www.tdsway.com/products/recon

[24] Qtek, "Qtek 9100", Mars 2006, http://www.qtek.no/norway/produkter/9100.aspx

[25] Symbol, "MC9000 Series Industrial Class Mobile Computer from Symbol", April 2006 http://www.symbol.com/category.php?category=171

[26] Sun Developer Network, "Java Platform, Micro Edition (Java ME)", sist lest Mai 2006 http://java.sun.com/j2me/

[27] Microsoft Corporation, ".NET Compact Framework", Mars 2006, http://msdn.microsoft.com/smartclient/understanding/netcf/

[28] Wikipedia – The Free Encyclopedia, "Automated identification and data capture", Mars 2006, http://en.wikipedia.org/wiki/Automated_identification_and_data_capture

[29] V. Agarwal, "Assessing the benefits of Auto-ID Technology in the Consumer Goods Industry", cambridge university auto-id centre institute for manufacturing, university of cambridge, UK, 2001

[30] About Inc, "Bar Codes", Mars 2006, http://inventors.about.com/library/inventors/blbar_code.htm

[31] TalTek, "How a Bar Code Reader Works", Mars 2006 http://www.taltech.com/TALtech_web/resources/intro_to_bc/bcpwork.htm

[32] Bar Code Readers, ”Choosing Tthe Right Bar Code Reader”, Mars 2006
http://www.barcodereader.info/Articles/Choosing_The_Right_Bar_Code_Reader.php

[33] Wikipedia – The Free Encyclopedia, “RS-232”, Mars 2006,
http://en.wikipedia.org/wiki/RS-232

[34] TEC-IT Datenverarbeitung GmbH, “Generate Bar Codes Online…”, Mars 2006,
http://www.tec-it.com/asp/main/startfr.asp?LN=1

[35] Wikipedia – The Free Encyclopedia, “Universal Product Code”, Mars 2006,
http://en.wikipedia.org/wiki/Universal_Product_Code

[36] Semacode Corporation, “Semacode”, Mars 2006
http://semacode.org/

[37] Parc Research, “DataGlyphs: Embedding Digital Data”, Mars 2006,
http://www.parc.com/

[38] Material Handling and Industrial Distribution Lab, SCHOOL OF INDUSTRIAL ENGINEERING PURDUE UNIVERSITY, WEST LAFAYETTE, U.S.A., “Magnetic Stripe”, Mars 2006,
http://gilbreth.ecn.purdue.edu/~tanchoco/MHE/ADC-is/Magnetic/main.shtml

[39] Luis Padilla Visdómine, ” Track format of magnetic stripe cards”, Mars 2006
http://www.gae.ucm.es/~padilla/extrawork/tracks.html

[40] Wikipedia – The Free Encyclopedia, “Magnetic stripe card”, Mars 2006,
http://en.wikipedia.org/wiki/Magnetic_stripe

[41] Sun Developer Network (SDN), “Smart Card Overview”, Mars 2006,
http://java.sun.com/products/javacard/smartcards.html

[42] LaserCard Ccorporation, Mars 2006
http://www.lasercard.com

[43] Wikipedia – The Free Encyclopedia, “RFID”, sist lest Mai 2006
http://en.wikipedia.org/wiki/Rfid

[44] Smartcode Corp., “SMARTCODE™ CORP. ANNOUNCES THE WORLD'S SMALLEST RFID CHIP”, MARS 2006,
http://www.smartcodecorp.com/newsroom/13-01-04.asp

[45] Computerworld, ” Nato ruster opp med RFID”, sist lest Mars 2006
http://www.computerworld.no/index.cfm?fuseaction=artikkel&id=3246DDC2-F1FF-9846-61C89FC1B94CEC3B

[46] RFID Weblog, “U.S. Government Market for RFID to Grow 120 Percent”, Mars 2006
http://www.rfid-weblog.com/50226711/us_government_market_for_rfid_to_grow_120_percent.php

[47] Computerworld, ” Standardisering gjør forskjellen”, sist lest Mars 2006, http://www.computerworld.no/index.cfm?fuseaction=artikkel&id=659B5DCA-C4E0-4472-FBC51BACC3B56774

[48] Computerworld, ”Finn en intern rfid-evangelist”, sist lest Mars 2006 http://www.computerworld.no/index.cfm?fuseaction=artikkel&id=90ECC234-CD09-4570-EB3F836D792F6DC5

[49] Digi.no, “RFID tok tiden på Vasaløpet”, Mars 2006, http://www.digi.no/php/art.php?id=294417

[50] Computerwolrd, ”Følger posten hele veien”, sist lest Mars 2006 http://www.computerworld.no/index.cfm/fuseaction/artikkel/id/41975/tema_id/E48EBF3C-EB74-8C5B-0E3479273BAC0A42

[51] Computerworld, ” Merker 70.000 brev”, sist lest Mars 2006 http://www.computerworld.no/index.cfm?fuseaction=artikkel&id=2B9A94BF-9222-98A6-07F4A196E5E0ADCA

[52] Posti - Finland Post Corporation, ” Finland Post to Pilot RFID Technology”, sist lest Mars 2006, http://www.posti.fi/english/current/archive2005/Finland_Post_to_pilot_RFID_Technology-150605.html

[53] RFID Journal, “Aussies Track Mail Service Via RFID”, sist lest Mars 2006 http://www.rfidjournal.com/article/articleview/2014/1/1/

[54] tu.no, ” Hackere knekker RFID-koder”, sist lest Mars 2006 http://www.tu.no/nyheter/ikt/article31897.ece

[55] Matthew Green, Adam Stubblefield, and Avi Rubin, The Johns Hopkins University Information Security Institute Baltimore, USA, “Analysis of the Texas Instruments DST RFID”, sist lest Mars 2006 http://rfid-analysis.org/

[56] PC World Norge, “RFID-brikker trues av datavirus”, Mars 2006 http://www.pcworld.no/index.cfm?fuseaction=artikkel&id=FDD69BD7-F1FF-9846-655D9A071A606EFA

[57] Faculty of sciences, Vrije Universiteit, Amsterdam, ”RFID Viruses and Worms”, Mars 2006, http://www.rfidvirus.org/index.html

[58] GS1 Norway, ”Transportbransjen”, Mai 2006, http://ean.imaker.no/cgi-bin/ean/imaker?id=139&visdybde=2&aktiv=139

[59] GS1 Norway, ”GS1-128 (tidl EAN/UCC 128)”, Mai 2006, http://www.gs1.no/cgi-bin/ean/imaker?id=321

[60] GS1 Norway, ”Felles transportetikett”, Mai 2006, http://ean.imaker.no/cgi-bin/ean/imaker?id=833

[61] GS1 Norway, ”GS1 General Specifications”, Mai 2006 http://ean.imaker.no/cgi-bin/ean/imaker?id=2307&visdybde=2&aktiv=2307

[62] GS1 Norway, “EAN Guiden om nummerering og merking med GTIN (EAN Artikkelnummer)”, Mai 2006 http://ean.imaker.no/cgi-bin/ean/imaker?id=1540

[63] GS1 Norway, ” Standard for merking av D-pak og pall i dagligvarebransjen”, Mai 2006, http://ean.imaker.no/cgi-bin/ean/imaker?id=846

[64] GS1 Norway, ” Dagligvare DEDIP2 brukerprofiler”, Mai 2006, http://ean.imaker.no/cgi-bin/ean/imaker?id=516

[65] Trelast og Byggevarehandelens Fellesorganisasjon, Mai 2006, http://www.tbf.no

[66] Trelast og Byggevarehandelens Fellesorganisasjon , “Global Service Relation Number (GSRN)”, Mai 2006, http://ean.imaker.no/data/f/0/07/06/3_2401_0/tildeling_Maalepunkt070206.pdf

[67] GS1 Sweden, Mai 2006, http://www.ean.se

[68] EPCglobal Inc., Mai 2006, http://www.epcglobalinc.org/

[69] EPCglobal Inc, ” Specifications & Ratified Standards”, Mai 2006, http://www.epcglobalinc.org/standards_technology/specificationsRatifiedStandards.html

[70] EPCglobal Inc, ”Specifications”, Mai 2006, http://www.epcglobalinc.org/standards_technology/specifications.html

[71] EPCglobal Inc, ”Ratified Standards”, Mai 2006, http://www.epcglobalinc.org/standards_technology/ratifiedStandards.html

[72] EPCglobal Inc, ” EPCTM Generation 1 Tag Data Standards Version 9 1.1 Rev.1.272”, 2005

[73] The RFID Weblog, ”China Creating its Own RFID Standard”, Mai 2006, http://www.rfid-weblog.com/50226711/china_creating_its_own_rfid_standard.php

[74] Strekmenn AS, Mars 2006 http://www.strekmenn.no/

[75] FileZilla FTP, Mai 2006

http://filezilla.sourceforge.net/

[76] LogMeIn, ”Remote Access and Desktop Control Software for your PC”, April 2006, https://secure.logmein.com/go.asp?page=home&lang=no

[77] Microsoft Corporation, “Microsoft .NET Framework Version 2.0 Redistributable Package (x86)”, April 2006, http://www.microsoft.com/downloads/details.aspx?FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5&displaylang=en

[78] Crhis Tacke, OpenNETCF.org - Forums**,** “Show/Hide the SIP”, Feburar 2006, http://www.opennetcf.org/forums/topic.asp?TOPIC_ID=57&SearchTerms=sip

[79] Esato.com, “Network parameters for GPRS connections”, Mai 2006, http://www.esato.com/archive/t.php/t-740

[80] Microsoft Corporation, “DataSet Class”, Mai 2006, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfSystemDataDataSetClassTopic.asp

[81] OpenNETCF.org, “File Download”, April 2006, http://www.opennetcf.org/download.asp?product=SmartDeviceFramework14

[82] W3C, Date and Time Formats”, Mai 2006 http://www.w3.org/TR/NOTE-datetime

[83] Apple Computer Inc., Mai 2006, http://www.apple.com/

[84] ConceptDraw, “MINDMAP 4 - Mind Mapping, Brainstorming and Project Planning”, Mai 2006, http://conceptdraw.com/en/products/mindmap/main.php

[85] Usability Sciences Corporation, “Delivering business impact through the science of usability”, Mai 2006 http://www.usabilitysciences.com/

[86] T Landauer, The Trouble with Computers: Usefulness, Usability, and Productivity, 1995

[87] OpenNETCF.org, “Connecting to Pocket PC thru VS .NET 2003”, April 2006, http://www.opennetcf.org/forums/topic.asp?TOPIC_ID=305&SearchTerms=activesync,bug

[88] Microsoft Corporation, “Windows CE Utilities for Visual Studio .NET 2003 Readme - I uninstalled ActiveSync and then reinstalled it. Now I can’t deploy my application. How do I fix this?”, April 2006, http://download.microsoft.com/download/c/d/b/cdbff573-73fb-4f9f-a464-c5adc890e1ae/Readme.htm#ASUninstall

[89] Microsoft Corporation, “BUG: Cannot deploy smart device application after you remove and reinstall ActiveSync”, April 2006, http://support.microsoft.com/default.aspx?scid=kb;en-us;813579

[90] Microsoft Corporation, ” Windows CE Utilities for Visual Studio .NET 2003 Add-on Pack 1.1”, April 2006, http://www.microsoft.com/downloads/details.aspx?FamilyId=7EC99CA6-2095-4086-B0CC-7C6C39B28762&displaylang=en

Vedlegg