LMON

Projektrapport

World Wide Medico

1 Resume

1.1 Dansk

Opgavens formål er at lave et Local Monitor System (LMON). LMON er et patientmonitorsystem, beregnet til brug i hospitalssektoren, som modtager signaler fra en patient, og en pumpe.

Fra patienten modtages ECG, EDR og puls signaler, der efterfølgende vises på LMON's display. LMON skal holde øje med patientens grænseværdier, og give alarm hvis disse overskrides.

Fra pumpen modtages indsprøjt raten og, en alarm hvis der opstår en fejl fra pumpen. Raten kan justeres både på pumpen og på LMON´s display. Ligeledes kan pumpen startes og stoppes fra LMON.

Opgavens problemstilling var, at udvikle et indlejret realtidssystem hvor der tages højde for de problemstillinger der er, ved udvikling til realtidssystemer. Under udviklingen er der især lagt vægt på dokumentationen og design mønstre. Systemet er lavet, så der tages højde for at det senere nemt kan udvides, til at indgå i et større distribueret system. Dokumentationen er lavet ved hjælp af use case teknikken og UML. Til at vælge design mønstre har vi taget udgangspunkt i bogen "Design Patterns Elements of Reusable Object-Oriented Software" [GOF-94].

LMON er udviklet til at køre på realtid operativsystemet RTKernel. RTKernel er installeret på en SBC686, som blandt andet kan kommunikerer med de to input/output enheder PV2019 og PPI.

Vi har stræbt efter at lave LMON så operativsystem uafhængigt som muligt, for at det skulle være muligt at skifte operativsystem specifikke ting ud, hvis LMON ønskes flyttet til et andet operativsystem. Operativsystem uafhængigheden har vi løst ved at udvikle vores eget framework, som vi har kaldt RTOS, hvilket medfører at systemet kan flyttes til et andet operativsystem uden større ændringer. I første omgang har vi udviklet LMOM til både at kunne køre på Windows og RTKernel.

Grafikken til systemet er lavet ved hjælp af RTPEG. Hele LMON er lavet med C++, udviklet i Microsoft Visual Studio .NET.

Det er lykkes at udvikle systemet, opfylde alle krav, og sample signaler med en frekvens på 500Hz.

1.2 English

The purpose of this assignment is to create a Local Monitor System (LMON). LMON is a patient monitoring system aimed for usage in the hospital sector, and receives signals from a patient and a pump.

ECG, EDR and pulse signals are received from the patient that subsequently is shown on LMON's display. LMON must watch the border limits for the patient, and give an alarm if these are exceeded.

From the pump the infusion rate is received, or an alarm message if an error occurs. The rate can be adjusted both on the pump and on LMON. The pump can be started and stopped from LMON.

The assignment was to develop a embedded real-time system where the problems related to real-time are taken into consideration. During development a special attention is paid to documentaton and design patterns. The system should be designed so that future extensions should be easy, and to be part of a larger distribueted system. The documentation is created using the use cases and UML. Design pattern inspiration has been taken from the book "Design Patterns Elements of Reusable Object-OrientedSoftware" [GOF-94].
LMON is developed to be executed on the realtime OS RTKernel. RTKernel is installed on a SBC686 that among other, communicates with two input/output units; PV2019 and PPI.

We have aimed at developing LMON as platform independent as possible, and to make it possible to change platform specific implementation code out, if LMON is portet to another operatingsystem.

We have solved the operatingsystem indenpendency by developing our own framework, called RTOS that makes it possible to move the application from one operating system to another without major changes. In the first time round we have developed LMON to be able to run under Windows and RTKernel.

The user interface is developed using RTPEG library. The whole LMON is written in C++, using Microsoft Visual Studio .NET.

We succeeded in developing a system that fulfils all requirements and samples signals with a frequency of 500Hz.

2 Indholdsfortegnelse

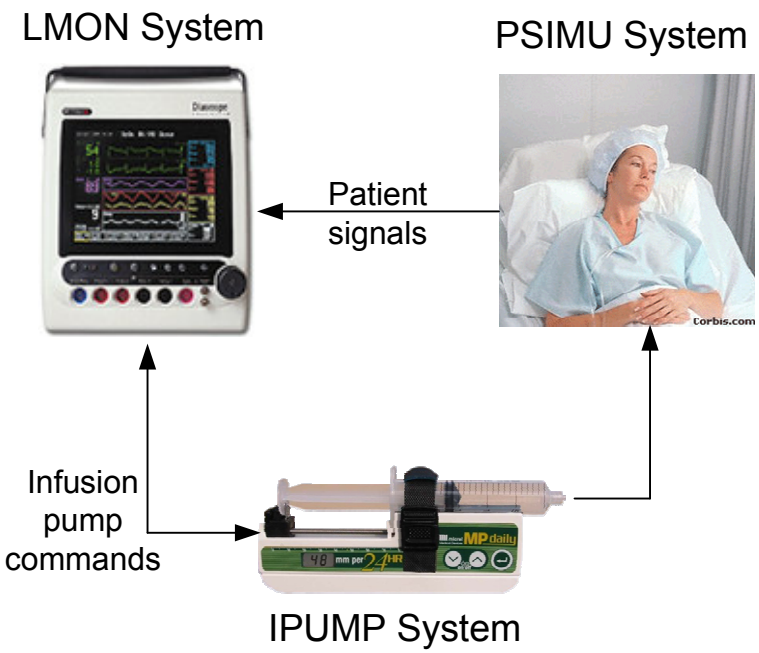
3 Indledning

Denne projektrapport beskriver projektforløbet til det første projekt på Teknisk IT linien. Projektets formål var, at udvikle et apparat til brug i hospitals sektoren, med fokus på realtid; modellering, diagrammering, samt programmering.

Projektet var opdelt i tre del-projekter:

- Et patientmonitor projekt (LMON, bilag #1).
- Et patient simulations projekt (PSIMU, bilag #2).
- Et projekt til styring af en pumpe (IPUMP, bilag #3)

Disse tre projekter, blev uddelt til tre forskellige projektgrupper. Alle tre projekter danner til sammen et stort system, der skal kunne arbejde sammen.



Figur 1 Oversigt over hele systemet.

Vores del af projektet var patientmonitorsystemet (LMON). Dette system skulle læse værdier fra patienten og vise dem på et display, samt styre pumpen.

Under udviklingen af LMON er der lagt vægt på dokumentering, realtid, design mønstre og operativsystem uafhængighed. Dokumenteringen er lavet ved hjælp af use case tekniken og UML.

Under udviklingen af LMON, har vi nøje undersøgt de problemer der kan forekomme, når der udvikles til et realtidssystem. Vi har ofte taget stilling til realtids problemmer som:

- Skal der samples, polles, eller skal der bruges interrupt rutiner.
- Hvilke prioriteter skal de forskellige tråde have.

- Hvor hurtigt skal der samples.

Hvilket er ting som ikke er så vigtige ved udvikling af systemer der ikke er så tids kritiske, men som her har en afgørende rolle for tidskritiske systemer.

Vi har under udviklingen hele tiden taget stilling til, hvilke design mønstre der var anvendelige til netop LMON. Vi har forsøgt at lave LMON operativsystem uafhængigt. Dette er gjort for at det skal være muligt at flytte LMON til et andet operativsystem, og så dele af det kan testes på windows. Operativsystem uafhængigheden er lavet ved at indkapsle den kode der er operativsystem afhængig, så som tråde, postkasser, semaforer, COM port m.m.

Rapporten består af to dele. En projektrapport der beskriver projektet og den valgte arbejdsproces, og en projekt dokumentation, der detaljeret beskriver udviklingen af produktet.

Projektrapporten består af resume, indledning, beskrivelse af udviklingsforløbet, samt en konklusion.
Projekt dokumentationen består af kravspecifikation, design og test.

3.1 Læsevejledning

Litteraturhenvisninger er skrevet på følgende format [forfatter – piblicerings år].
Et eksempel på dette er : [GOF-94].
For en komplet liste over referede bøger, se Litteraturlisten, fane 6.

For yderlig information omkring krav til systemet og use cases, se ”LMON Requirement Specification”, fane 2.

En detaljeret gennemgang af systemets design, realiseringer af use cases og implementation af design mønstre findes i ”LMON Design”, fane 3.

En test af test af use cases samt af det samlede system kan læses i ”LMON Test”, fane 5.

4 Projekt beskrivelse

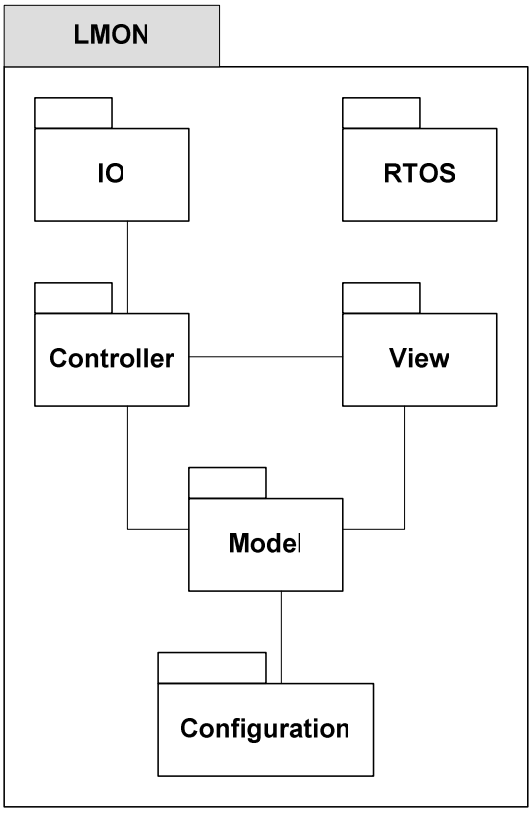
4.1 Projektgennemførelsen

Der er brugt to iterationer til udviklingen af systemet. I første iteration blev der fokuseret på at få hul igennem til patienten, sample dennes data, og udskrive dem ved hjælp af en simpel waveform på skærmen. I anden iteration blev kommunikationen med patienten mere forfinet, kommunikationen med pumpen blev tilføjet, samt displayet til brugeren blev lavet færdigt, og der blev tilføjet flere design mønstre.

Udviklingen af systemet blev delt op i flere delsystemer, som kunne laves uafhængigt.

- IO, som stod for kommunikation med patient og pumpe.
- View, som stod for kommunikation med brugeren af systemet igennem displayet.
- Controller, som stod for event- og tilstands styringen i systemet.
- Model, som opbevarede og behandlede alt data.
- RTOS som indeholder den operativsystem specifikke kode
- Configuration der indeholder de forskellige indstillinger brugeren kan sætte.

Figur 2 viser et pakkediagram af LMON.



Figur 2 Oversigt over systemets pakker

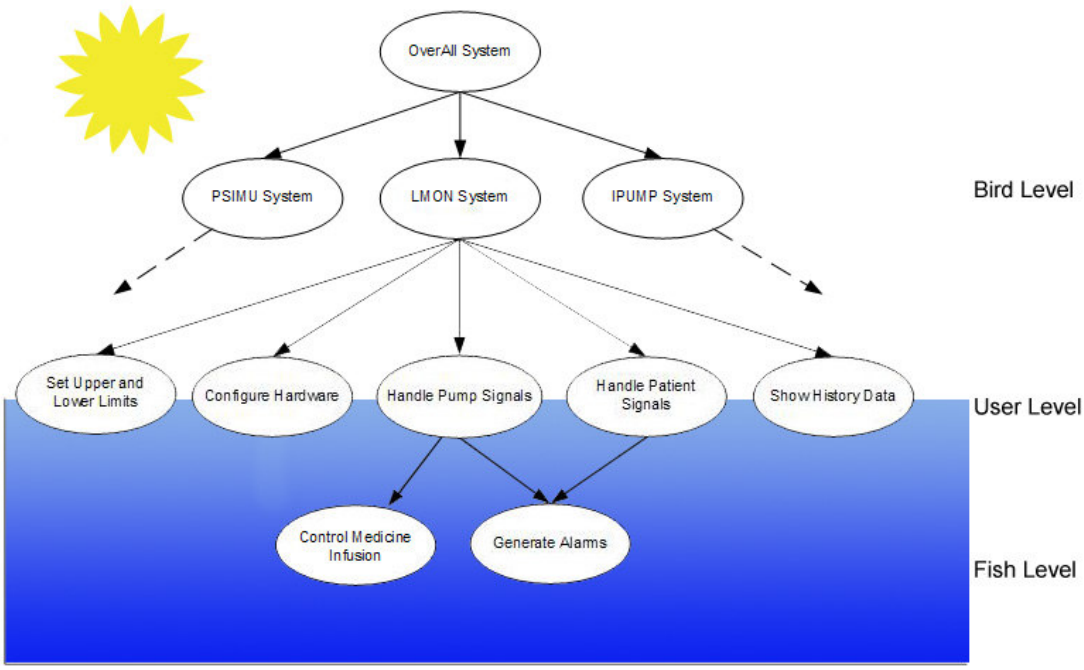
4.2 Metoder

Systemet er lavet ved hjælp af forskellige OOP teknikker. Use Case teknikken er brugt til at beskrive de funktionelle krav i systemet. UML blev brugt til at diagrammere systemet. Design mønstre er brugt til at modellere systemet.

4.3 Specifikations- og analysearbejdet

Til at specificere og analysere systemet har vi primært brugt use cases. Vi har delt vores Use Cases op, så de enten er bruger-, eller system initierede. Brugerinitierede use cases initieres af en aktør, og systeminitierede initieres af systemet.

Use casene er delt op i tre niveauer[Cockburn-01]. 'Bird level', 'user level' og 'fish level'.
'Bird level' beskriver de meget abstrakte use cases. Det vil sige, at det kan være ting som LMON delsystemet, eller en af de andre to del systemer.
'user level' beskriver, kort sagt, use cases som en primær bruger kan benytte sig af, og bagefter gå tilfreds væk. En 'user level' use case kan have en eller flere 'fish level' use cases under sig.
'fish level' er mere detaljeret, og her kan beskrives detaljerede ting, som er meget system kritiske. En 'fish level' use case har en eller flere 'user level' use cases over sig.
Vores use cases er lavet, så de primært ligger på 'user level'. Det vil sige at de er mere abstrakte end hvis de f.eks. lå på 'fish level'. Figuren herunder illustrerer hvorledes vores use cases ligger.



Figur 3 Oversigt over use case niveauerne

Det har været nemt at specificere kravene ved hjælp af use case teknikken, da man hurtigt får et overblik over hvad hele systemet skal kunne.

4.4 Designprocessen

Vi har anvendt ROPES designprocessen til udvikling af systemet. ROPES er en iterative udviklings proces, der anvender use cases i centrum for udviklingen. Vi har i LMON prcessen gennemløbet 2 iterationer, og løbene forfinet vores arkitektur. Vi har anvendt 4+1 view modellen [se bilag #6], til at afspejle systemet fra flere vinkler.

Allerede i første iteration har vi overveget opdellingen af pakker og ansvar i sytstemet, disse overvejelser resulterede i at mønstret '*Model View Controller* (MVC) [Buschmann-96 side 125]' blev valgt for at have en passende ansvarsfordelling i systemet. Dette mønster bruges til at opdele systemet i velkendte pakker og består af:

- Et view, som viser systemets tilstand og data til brugeren.
- En controller, hvor alle events/inputs bliver udført.
- En model, hvor alt data bliver opbevaret, og behandlet.

Til at modellere og beskrive systemet er anvendt UML og kendte design mønstre. Begge dele gør systemet mere struktureret, og nemmere at arbejde videre på, hvis det senere skal videreudvikles af os selv eller af andre. Design mønstrene har også givet os et højere abstraktions niveau, hvorved det er nemmere at diskutere hvorledes softwaren skal opbygges.

Under udviklingen, har vi taget højde for at systemet både skal indgå i det samlede patient monitor system, hvor vi kommunikerer både med en pumpe og en patient, samt at systemet senere skal kunne udvikles til at indgå i et større distribueret system, hvor det skal være nemt at hente data ud fra LMON.

Vi har brugt UML til at beskrive softwaren, samt systemets funktionalitet. De arkitektursignifikante use cases er beskrevet ved hjælp af sekvensdiagrammer. Softwaren er modeleret ved hjælp af pakke- og klassediagrammer. Systemets tilstand er modelleret ved hjælp af et tilstandsdiagram, som beskriver alle systemets tilstande, og actions der provokerer tilstandsskift.. Systemets processer er modeleret ved hjælp af et proces/task view, som beskriver de forskellige tråde i systemet, og kommunikationen imellem dem.

4.5 Udviklingsværktøjer

4.5.1 Microsoft Visio 2003

Til at lave use case diagrammer og UML diagrammer er brugt Microsoft Visio 2003(Visio). Visio er et program som kan bruges til at tegne UML og use case diagrammer. Det kan ikke generere kode for en, det kan ikke løbe en sekvens igennem og teste det, og det kan ikke opdatere diagrammer løbende ud fra den kode der udvikles.

Vi har valgt at bruge Visio, da det er det vi kender bedst, og det opfylder de krav vi har til et sådan værktøj.

Alternativer er Rhapsody 5.0 og Rational Rose UML tool. Begge er mere avancerede end Visio, og har ikke de svagheder, som vi har nævnt ved Visio.

4.5.2 Microsoft Visual Studio .NET 2003

Til udviklingen af systemet, har vi brugt C++. Vi har udviklet systemet i Microsoft Visual Studio .NET 2003(VS).

For at effektivisere vores udviklingsproces, har vi udviklet en VS skabelon der gør det muligt at anvende RTKernels udviklingsmiljø fra VS. Skabelonen gør det bl.a. muligt at bruge RTKernel´s emulerings biblioteker og afvikle RTPEG i Windows. Denne funktionalitet har gjort udvikling af brugergrænsefladen (RTPEG) betydeligt mere effektiv, da udvikleren ikke er bundet til at teste sin applikation på en SBC686. En anden vigtig funktionalitet skabelon stiller til rådighed er metoder til at oversætte og linke sit program til både dos og diskette.

Ved at starte VS gennem RTKernel´ s debug shell, kan vi forbinde udviklingsmaskinen direkte til SBC686, og overføre og debugge applikationen gennem VS. Det er således muligt at indsætte breakpoints i sin kode og steppe gennem for at finde fejl.

4.6 Resultater

Det er lykkedes at udvikle LMON, så det opfylder alle de krav vi har stillet til det.
Der kan kommunikeres med pumpen, og sample hastigheden til patienten, på 20hz, er længe overholdt. Det er sågar lykkes at sample patient signalet med en frekvens på 500 Hz.
LMON kan vise de samplede EDR og ECG signaler, i nogle passende grafer der viser patientes tilstand. Pulsen vises med et lille hjerte som et digital tal, og beskeder fra pumpen vises som en status tekst. Brugeren har mulighed for at gå tilbage i tiden vha. en slider, der samtidig viser hvor lang tid der er gået siden maskinen er startet.
Brugern har også mulighed for til hvert signal der samples, at indstille en minumum og en maximum værdi signalet skal overskride for at generere en visuel og audio alarm.

Figur 4: Screenshot af det kørende system

Vi har fået udviklet et framework der tilbyder platform uafhængighed. Dette framework er designet vha. mønstret *abstract factory* pattern [GOF-94 side 87], og gør det muligt at tilføje flere operativ systemer uden større ændringer i koden. Vi har fjernet den operativsystem specifikke kode fra LMONs klasser, og placeret det i egene klasser. Båndet mellem LMON og de specifikke klasser oprettes ved at benytte abstrakte definitioner. For mere information omkring frameworket se i design dokumentet, afsnit 6.2.2.

4.7 Diskussion af opnåede resultater

Til trods for at vi har opfyldt alle kravene vi har stillet til systemet, er vores visning af ECG og EDR noget grynede og firkantede. Dette er til trods for at vi overholder Nyquist's samplings påstand, som siger at samplingshastigheden skal være dobbelt så stor som det samplede signals højeste frekvens.
Da den højeste frekvens vi kan sample med er 500 Hz, kan vi ikke vise et signal præcist, hvis det har en frekvens der er større end 250 Hz. For at finde ud af hvor stort et problem det er, kræver det en støre test, hvor vi sampler signaler fra en rigtig patient, eller fra PSIMU projektet.

4.8 Opnåede erfaringer

Gennem udviklingsforløbet af LMON her vi opnået at få forståelse for brug af design mønstre, realtids programmering samt fået udviklet et system der overholder de stillede krav.
Vi har modeleret systemet ved hjælp af use case teknikken, og benyttet flere forskellige design mønstre til at gøre koden mere flexibel og letlæselig.
Under udviklingen skulle vi være opmærksomme på at det var et realtidssytem vi skulle udvikle. Dette resulterede i, at der skulle tænkes lidt anderledes med hensyn til f.eks oprettelse og nedlæggelse af

objekter i forhold til udvikling af ikke realtids applikationer. Løsningen til at udgå at oprette objekter var at anvende singleton pattern, til at sørge for at kun 1 instans af objektet findes.
Under implementationen løb vi et par gange ind i problemer med RTKernel. Det første problem var vores kommunikation med COM porten. Under udvikling af IO pakken fungerede kommunikation til pumpen, men efter at have samlet pumpen med RTPEG opstod fejlen. Problemet viste sig at være i COM port driveren der fulgte med RTKernel biblioteket. Mere information om dette problem kan findes i design dokumentet afsnit 6.2.1.2.

Et andet problem vi løb ind i, var med de graf klasser der blev anvendt til at tegne EDR og ECG grafer. Vi ville benytte nogle funktioner som frameworket stillede tilrådighed, men det viste sig, at disse funktioner ikke var implementerede. Dette problem kunne kun løses på en enkelt måde; vi måtte selv implementere de manglende funktioner. Dette resulterede i en ekstra klasse i View, der indeholder vores implementation af funktionerne.

4.9 Projektets fortræffeligheder

Der er dele af LMON som vi har lagt ekstra arbejde i, og derfor er ekstra stolte af.

Vi syntes vi har fået lavet en god og intuitiv brugergrænseflade til systemet. Vi har især lagt væk på at brugergrænsefladen skal kunne bruges til en touchscreen, at den skal være flot, at den skal give et overskueligt overblik over hvordan patienten har det, samt at den skal kunne visse alle de ting som systemet kan, altså ikke skjule funktionalitet for brugeren.

Vi har taget hånd om realtids problemstillingerne, og anvender en preemptiv skedulering, til at tilbyde en blød real tid.

Vi er stolte af at vi har lavet systemet operativsystem uafhængigt. Det har vi gjort ved at lave vores eget framework, som tager hånd om alle operativsystem specifike kald. Som et resultat af det, er den resterende kode blevet en del pænere, da den ikke er fyldt med disse kald. Det gør også, at LMON er meget nemt at flytte til et andet operativsystem end RTKernel, da det kun er vores framework der så skal ændres i.

LMON er lavet ved hjælp af velvalgte design mønstre, og er godt dokumenteret ved hjælp af UML, så det vil være nemt for en ekstern programmør at få et overblik over koden, og udvide eller ændre den.

4.10 Forslag til forbedringer af projektet eller produktet

Systemet skal senere kunne indgå i et større distribueret realtids system, hvor data skal stilles til rådighed for andre enheder i det distribuerede system.

For at undgå de kantede grafer, skal systemet kunne håndtere højere samplingsfrekvenser. En mulighed kunne vare at benytte detikeret hardware til at sample data.

Endelig ville det også være en forbedring af systemet at fokusere mere på realtid. Vi har naturligvis overvejet hvordan de forskellige dele bedst kunne implmenteres, men en mere fokuseret analyse af hvor der skulle sættes ind for at skabe bedre realttime ville være en fornuftig forbedring.

Det ville være interesant at undersøge forskellige skedulerings algoritmer, og således få undersøgt hvilke muligheder der fandtes, og hvilken løsning der ville passe bedst til systemet.

Det ville være en stor hjælp hvis der fandtes værktøjer til at undersøge real tiden i et system. Det kan være meget svært at beregne, hvor lang tid en operation tager.

5 Konklusion

Dette projektforløb har budt på en række af nye udfordringer. Vi har for første gang arbejdet med modellering og udvikling af realtidssystemer, hvor der er stillet større krav til samtidighed og resurser. Det har imidlertid ikke været nogen nem opgave at udvikle realtidssystemet, og det er svært at afgøre om vores realitid er overholdt i det endelige system. Det kunne være rart hvis der fandtes et værktøj der kunne hjælpe med at måle og analysere realtidssystemer.

Under udvikling af systemet har vi haft mest fokus på at bruge design patterns. Vi har fra start af overvejet forskellige muligheder, og er løbende blevet inspireret under kursus-forløbet, til at bruge flere patterns. Vi mener ikke at have overdrevet vores brug af patterns, da vi mener det er af stor betydning at kunne udvikle software på et højere abstraktionsniveau. Det vil være svært i fremtiden ikke at gøre brug af de patterns vi har lært.

Vi har i modeleringen benyttet os af flere forskellige design mønstre. Dette har været med til at skabe et ensartet flow gennem hele systemet. Ligeledes gør brugen af et eller flere mønstre koden mere genbrugelig, samt andre udviklere vil lettere kunne sætte sig ind i hvordan systemet virker.
Det design vi har valgt, er yderst flexibelt og istand til at blive tilpasset et andet miljø. På et senere tidspunkt skal LMON kunne integeres med et central patient monitor system, og når dette sker skal vores design muligvis laves lidt om. Men eftersom vi endnu ikke kender til det specifikke design af den central, har vi valgt et mønster der kan tilpasses nye omgivelser.

Vi har fået udviklet en brugervenlig brugergrænseflade, der på simpel vis giver overblik over den tilsluttede patient. Det er muligt for en bruger af systemet, at indstille forskellige alarm grænseværdier for de forskellige signaler der vises fra patienten. Brugeren har også mulighed for, at kunne se tidligere patient data. Endelig er det muligt for en tekniker at indstille apperatet til at benytte de indstillinger et bestemt hospital benytter.

Systemet kører blød realtid, da vi vi anvender preemptiv skedulering, og derfor ingen garantier kan få for at et task bruger den tid der er beregnet. En anden faktor er, at vi gemmer data til en log fil, og denne får en betydelig størrelse og dette peger også på et blødt realtids system.
Systemet opererer med en samplingsfrekvens på mindst 20ms, og er istand til at sample data fra tre forskellige datakilder; ECG, EDR og puls. Disse signaler bliver lagret på et persistens medie, så de er tilrådige efter brug.

Under implementationen gjorde vi stor brug af den operativsystem uafhængighed vores system har opnået. Det har blandt andet vist sig at være til stor hjælp da brugergrænsefladen skulle udvikles og testes. Ved at udvikle systemet så det er muligt at skifte operativsystemet ud, har vi forøget vores

muligheder for at kunne konkurrere på det rigtige marked, hvor virksomheder ofte stiller krav til, vitale dele af systmerne skal kunne skiftes ud.

Vi har testet LMON, og fået det til at kommunikere med en simuleret patient, samt en pumpe. Det var muligt at vise EDR og ECG signalerne i waveform grafer, dog var disse en smule hakkede. Vi testede med at sende et sinus signal ind, og på LMON kunne vi se signalet, men det var ikke helt glat. Vi kunne indstille forskellige grænseværdier for hvert signal, og hvis et signal overskred sin grænseværdi, blev alarmen sat igang.

Underskrifter

_____

Frank Henningsen

_____

Claus Kirkegaard Clausen

_____

Nick Lykke Nygaard

_____

Jens Peter Troelsen