Bachelorprojekt

Dataopsamling i forbindelse med analyse af nakke-skulder-arm problemer

af:
Andreas Rune Fugl [140983] anfug03@student.sdu.dk
Frederik Kvistgaard Ellehøj [230774] ththy03@student.sdu.dk

Datateknologistuderende ved Syddansk Universitet og IOT
Forårssemesteret 2006

Start: December 2005
Slut: Maj 2006

Vejleder: Kasper Støy

Resumé

Målet med dette bachelorprojekt, har været at designe og implementere måleapperatur til brug ved målinger i forbindelse med arm-skulder-nakke problemer opstået ved ensidigt gentaget arbejde. På længere sigt er det håbet at sådan apperatur vil kunne hjælpe til at forbygge disse typer af skader, ved at advare brugeren om potentielt farlige arbejdsvaner.
Det indledende analysearbejde afdækkede behovet for at finde en eller flere sensorteknologier, der kan benyttes til denne form for målinger. Det blev derfor besluttet at dreje projektet i en retning hvor fokus flyttes til at udvikle en hardware og software platform, der på simpel vis skal muliggøre praktiske tests af en række forskellige sensorer.
Ved projektets afslutning forligger der en fungerede prototype på en sådan sensor testplatform. Prototypen er en ekstrem fleksibel platform der tilbyder interfacing til stort set alle typer sensorer, idet den benytter en programmerbar analog frontend. Samtidig er den meget brugervenlig, og kan opsamle data over en meget lang periode. Det er dog stadig kun en prototype, med plads til forbedringer, omend den kan benyttes i sin nuværende form.

Forord

Denne rapport tager udgangspunkt i et ønske om at forebygge skader opstået som følge af ensidigt gentaget arbejde, f.eks. museskader. Som sådan henvender rapporten sig til alle med en interesse i dette felt, men rapporten henvender sig også til folk med en generel interesse i dataopsamlingssystemer, og embedded hardware/software.

Forfatterne af rapporten ønsker at rette en tak til følgende læger fra ortopædkirurgisk afdeling OUH: Overlæge PhD Lars Henrik Frich, overlæge Søren Larsen og overlæge Jens Lauritsen. Disse har alle en andel i at bringe forfatternes ringe lægelige viden op på et niveau som muliggjorde dette projekt.

Andreas Rune Fugl

Frederik Kvistgaard Ellehøj

Indhold

Del I

Bachelor Projektrapport

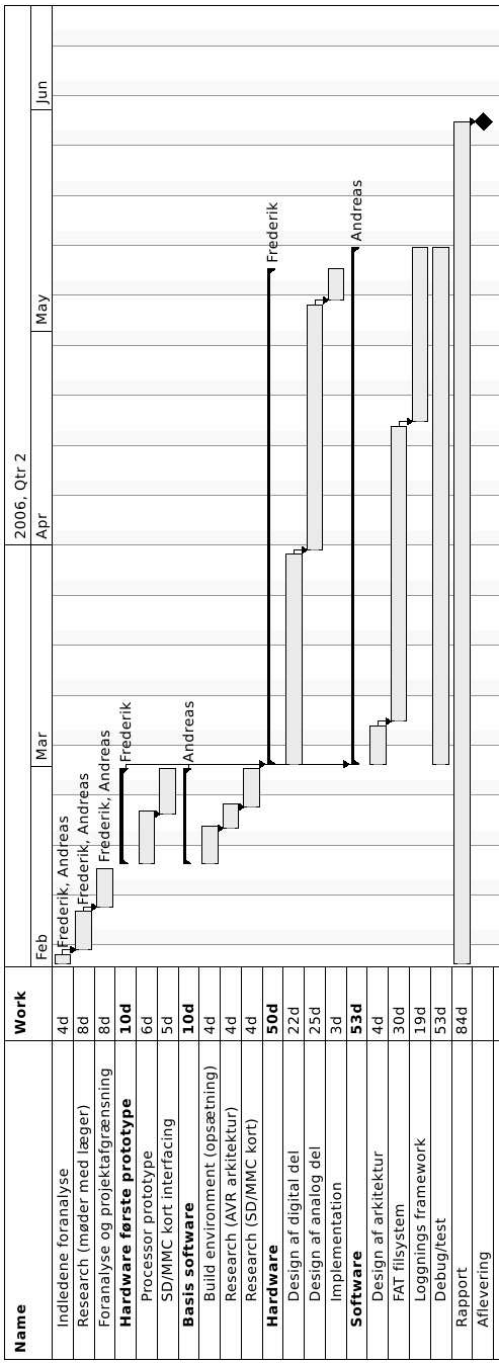
Kapitel 1

Indledning

1.1 Læsevejledning

Rapporten er delt op fire afsnit: Et indledende afsnit indeholdende problemformulering og foranalyse, et afsnit beskrivende hardwareplatformen, et afsnit beskrivende den udviklede software og endelig en afsluttende del med en samlende beskrivelse af systemet i funktion.
På den medfølgende CDROM kan kildekoden til systemets software samt datablade og diagrammer findes.
De enkelte dele kan læses hver for sig, men det anbefales dog at følge rapportens opdeling.

1.2 Tidsplan



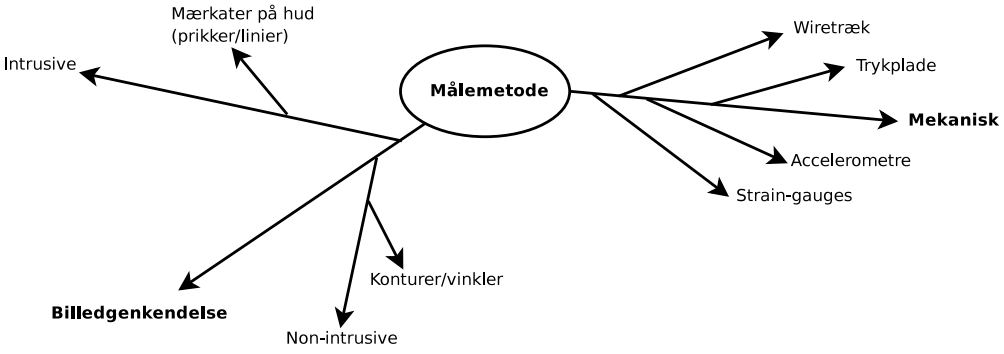
Figur 1.1: Fordeling af arbejdsopgaver.

1.3 Problemformulering

I forbindelse med brug af personlige computere, opstår der ofte tilfælde af EGA (Ensformigt gentaget arbejde) med nedsat arbejdsevne til følge.
Målet med projektet er at designe dataopsamlingsudstyr der kan assistere analysen af ergonomien ved computerarbejde og således hjælpe til forståelsen af nakke-skulder-arm problemer.

1.4 Foranalyse

Fra projektets start stod det klart at gruppen ikke havde den fornødne viden omkring ergonomien ved computerarbejde. På grund af dette var der dermed ingen muligheder for at kunne opstille konkrete måleparametre for et dataopsamlingssystem. Det blev dog forsøgt evalueret hvilke målemetoder på en menneskekrop, der fra et hardware- og softwaremæssigt synspunkt kunne realiseres. Dette fremgår af følgende figur 1.2



Figur 1.2: Brainstorm over målemetoder

Som det kan ses af figuren, er der mange forskellige målemetoder, overordnet set mekanisk eller billedgenkendelse. De har dog hver deres fordele og ulemper og et specifikt valg kunne derfor ikke tages uden måleparametre og ønskede nøjagtigheder.

Der blev derfor taget kontakt til Ortopædkirurgisk afdeling på OUH, for at søge viden om, hvilke konkrete måleparametre og nøjagtigheder der måtte være relevante i en undersøgelse.

1.4.1 Undersøgelser af håndled

Projektgruppen fik lov til at observere og stille spørgsmål ved en efterundersøgelse i ergoterapien på OUH af patienter som har haft håndledsbrud (Colles frakturer) i 2004. Dette indebar bla. måling af underarms og håndledsbevægelighed.

Formålet med ergoterapiens undersøgelse var at analysere den gavnlige virkning af forskellige behandlingsmetoder, som f.eks operativt indgreb kontra gips, træning ved en ergoterapeaut og lign. Et forventet resultat af undersøgelsen var at få afklaret om det kunne betale sig at starte tidligt i forløbet med genoptræningen, samt nytten af de udførte operationer.
Fokus for projektgruppen var primært at være åbne over for de anvendte målemetoder, deres eksisterende nøjagtighed og præcision, tidsforbrug af forløbet samt indvirkning på patienterne.

1.4.2 Undersøgelsen

Den samlede efterundersøgelse bestod af 3 hovedpunkter

1. Tilsendt spørgeskema (DASH[1])
2. Spørgeskema udleveret ved ankomst
3. Målinger, udvalgt fra lægens skøn

Det tilsendte spørgeskema (1) havde primært fokus på hverdagsgøremål, f.eks dreje nøgler, tage tøj på, åbne syltetøjsglas mm.
Det udleverede spørgeskema (2) havde fokus på de nuværende smerter på dagen for undersøgelsen. Smerten blev skaleret på en 1-10 skala.

1.4.3 Målinger

Målingerne blev udført af overlæge Søren Larsen og den tilstedeværende ergoterapeaut. Ca. 4 personer var inde ad gangen. Alt afhængigt af antal spørgsmål og personer tog hvert hold ca. 15-20 minutter.
Følgende målinger blev udført

1. Vinkelmåling af håndleddets bevægelighed (Måling på selve håndleddet med brug af oniometer[2] lagt ind til huden, bevægelse af underarm undgås)
   (a) Op/ned (extension/flexion (Skala +/- 90 grader)
   (b) Til siden (radial/ulnar deviation)
2. Underarms bevægelighed (skulder låst)
   (a) Supination/pronation (Tommelfinger væk fra/imod kroppen)
3. Tommelfingers bevægelighed
   (a) Objektivt skøn på at tommelfingeren kan nå håndfladen på modsatte side

[1]Disabilities of Arm, Shoulder and Hand - Spørgeskema baseret på amerikansk model
[2]Speciel vinkelmåler

4. Krumning af fingre
    (a) Alle fingerspidser når den øverste linje i håndfladen
    (b) Tommelfinger kan nå base af lillefinger

Følgende målinger blev udført af en ergoterapeaut

1. Grebsstyrke
    (a) Patient fatter om et måleinstrument og presser så at fingerstyrken kan måles

De forskellige bevægeligheder eller styrker summes typisk sammen for en akse for at give en samlet bedømmelse af bevægeligheden.

1.4.4 Observationer

Igennem forløbet blev der gjort følgende observationer:

1. Målinger ved en klinisk undersøgelse baseres ofte på manuelle aflæsninger af f.eks. oniometre, der foretages med en læges subjektive skøn.
2. Lægen der udfører undersøgelsen af folks bevægelighed i håndleddet kan typisk ikke udføre vinkelmålingen med mere end 10 graders nøjagtighed, grundet forskellighed i folks anatomi.
3. Lægen udtalte at de anvendte målinger var valgt ud fra hans egen vurdering om hvilke parametre der kunne være relevante i undersøgelsen. Han havde ikke nogen anerkendte metoder at rette sig efter. Derfor var undersøgelsen også et forsøg på at finde en metode til vurdering af bevægeligheden.
4. Ved undersøgelserne udfører patienterne ofte bevægelser ud i ekstremer der kunne medføre gene for dem. Når de blev blev adspurgt, svarede den langtovervejende del af personerne, at de normalt ikke ville udføre sådanne bevægelse i dagligdagen.
5. Det var meget forskelligt, hvor meget at folk anstrengte sig for at udføre bevægelserne. Enkelte ville ikke udføre bevægelserne til deres ekstremer, mens andre tydeligvis gjorde deres yderste. Desuden indgik der et konkurrencemoment, da flere personer sad i samme lokale samtidig til undersøgelsen.

Ud fra ovenstående blev der gjort følgende delkonklusioner

- Et kompakt, bærbart logningsapparatur til anvendelse på patienten, vil kunne give et indblik i hvordan patienters hverdag er påvirket af eventuelle gener. Dette vurderes som at give et langt bedre indblik i om successkriteriet for behandlingerne bliver opfyldt.

- Set ud fra de første tre observationer er det dog klart, at der ikke findes én anerkendt parameter der kan måles og vurderes. Det vil derfor være uhensigtsmæssigt at designe målrettet logningsapparatur, før at det er fuldstændig afklaret hvilke parametre der skal måles.

Selvom patienterne i den omtalte undersøgelse ikke har direkte relation til projektets målgruppe (folk der lider under EGA fra computerbrug), har den observerede undersøgelse dog givet et indblik i mange af de problemstillinger der opstår ved måling på den menneskelige anatomi. Disse problemstillinger omfatter blandt andet forskellighed i folks anatomi, vigtigheden af at måle de rigtige steder samt mange andre faktorer.

1.4.5 Konklusion på foranalyse

Da der fra den ovenstående analyse er blevet klart, at der ikke findes én sikker måde at kunne opsamle relevante data på, er det blevet valgt ikke at satse på en bestemt målemetode eller sensortype.
Istedet vil der blive udviklet en fleksibel evalueringsplatform, hvorpå der kan afprøves forskellige sensortyper, primært rettet imod mekanisk/elektrisk baserede typer som vist på figur 1.2.
Evalueringsplatformen skal gøre det muligt at imødekomme de alsidige krav som forskellige sensorer har med hensyn til signaltilpasning, timingskrav, logningsperioder mm.
Platformen skal desuden være brugervenlig, så at personer uden indgående kendskab til logningssystems hardware og software kan udføre målinger.
Endelig skal vejen fra planlægning og design af en sensorkonfiguration over til konkret datalogning kunne foregå på en hurtig og effektiv måde. Opsummeret er de overordnede krav til evalueringsplatformen:

1. Fleksibel mht. sensorvalg
2. Brugervenlig
3. Fleksibel konfiguration
4. Understøttelse for rapid prototyping

Med en evalueringsplatform der opfylder disse krav, vil det være muligt hurtigt at kunne evaluere en lang række sensorer i forbindelse med kliniske undersøgelser og i begrænset omfang personers dagligdag.
Ud fra dette vil der med høj sandsynlighed kunne konkluderes på hvilke sensortyper og konfigurationer, der kan anvendes i analysen af nakke-skulder-arm problemer samt andre lignende undersøgelser.

Del II

Hardware

Kapitel 2

Indledning

Denne del af rapporten beskriver design og implementation samt test af systemets hardware.

Skrevet af Frederik Kvistgaard Ellehøj

I denne del af rapporten beskrives den udviklede hardware. I første kapitel beskrives de overordnede valg der er truffet i forbindelse med design af en hardwareplatform til dette projekt. Andet kapitel beskriver den digitale del af hardwaren, mens tredie kapitel beskriver den analoge del af hardwaren.
Der vil i beskrivelsen af hardwareplaformen, blive lagt vægt på de overvejelser der ligger til grund for de valgte løsninger. Der vil således kun i det omfang det er skønnet nødvendigt forkomme teknisk baggrundsmateriale.

2.1 Valg af løsningsmodel

Den hardwareplatform der ønskes opbygget, skal understøtte de i foranalysen opstillede krav:

- Fleksibel mht. sensorvalg.
- Brugervenlig.
- Fleksibel konfiguration.
- Understøttelse for rapid prototyping.

For at reflektere disse krav i den udviklede hardware, opdeles denne i en analogdel og en digitaldel for derved at gøre det lettere at identificere kravenes indflydelse på disse. Overordnet set skal digitaldelen af hardwaren tilbyde følgende funktioner:

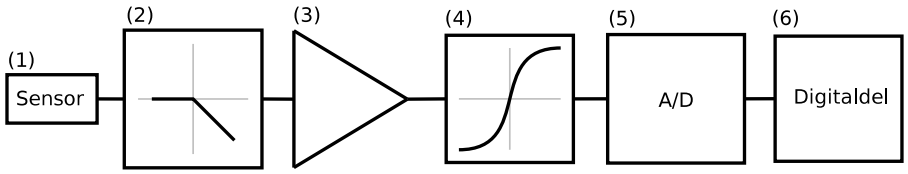
- Overordnede kontrolfunktioner for systemet.
- Datastorage til opsamlede data.
- Real time clock (RTC) til tidsstempling af opsamlede data.

- Kommunikationsinterface til omverdenen.
- Diverse I/O funktioner.

Den analoge del af hardwaren skal tilbyde følgende funktioner:

- Signalkonditionering af signaler fra sensorer.
- Båndbreddebegrænsning af signaler fra sensorer.
- Analog til digital konvertering af signaler fra sensorer.
- Funktioner til at skifte mellem forskellige sensorer.

Kravet om båndbreddebegrænsning af signalerne opstår som en konsekvens af Nyquist-Shannon-Kotelnikov sampling teoremet, idet vi sampler signalerne ved A/D konverteringen. Ud over de ovenfor beskrevne funktioner, skal systemet naturligvis også indeholde kredsløb til spændingsforsyning, resetstyring mv.
Den digitale del af hardwaren kan forholdsvis let realiseres vha. en microcontroller og tilhørende periferikomponenter. Derved kan kravene til denne del af hardwaren imødekommes uden de store designmæssige problemer. I det følgende forudsættes derfor at denne del af hardwaren allerede eksisterer og er funktionsdygtig. For en nærmere beskrivelse af den faktiske opbygning af den digitale hardware, henvises til kapitel 3. Da analoge sensorer kan have vidt forskellige karakteristika, er opbygningen af den analoge hardware en del sværere, hvis det ikke på forhånd vides hvilke sensorer der skal interfaces. Der er derfor nødvendigt at designe analogdelen således at den er fleksibel nok til at kunne interface forskellige typer af sensorer på en nem og brugervenlig måde.



Figur 2.1: Principdiagram for analogdel.

Figur 2.1 viser et principdiagram for analogdelen. Bemærk at hvis flere sensorer skal understøttes, kræver dette en multiplekser. Denne er dog udeladt her af hensyn til overskueligheden. Herunder er de enkelte blokkes funktion beskrevet:

1. Sensoren skal omsætte en ydre påvirkning til et elektrisk signal.
2. Et lavpasfilter sørger for at båndbreddebegrænse signalet så aliasing undgåes, når signalet senere samples af A/D konverteren.
3. En forstærker sørger for at tilpasse signalet til A/D konverterens udstyringsområde.

4. Det ville være at fortrække, hvis en arbitrær funktion kunne indsættes i analogdelen. En sådan funktion kunne f.eks. være en lineariseringsfunktion, eller en peak detektor.

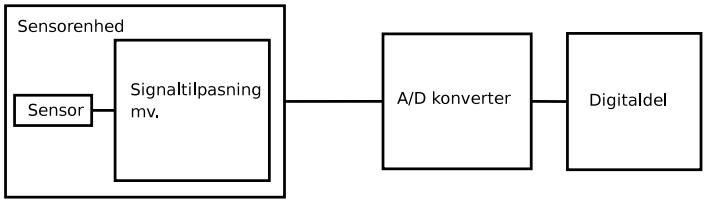
5. Samplingen af signalet, foretages af en A/D konverter.

6. Det samplede signal overføres til digitaldelen for videre forarbejdning og logning.

Det er klart at blokkene 1 til 5 skal være konfigurerbare eller udskiftelige for at opnå en tilfredsstillende fleksibilitet i analogdelen. I de følgende tre undersektioner vil tre forskellige muligheder for at opnå denne fleksibilitet blive beskrevet.

2.1.1 Signaltilpasning ved sensor

Den letteste løsning til at opnå fleksibilitet, er at kræve at sensorerne selv tager sig af al signalkonditionering osv. Herved opnåes at analogdelen i praksis blot består af A/D konverteren. På figur 2.2 ses et principdiagram for denne løsningsmodel.

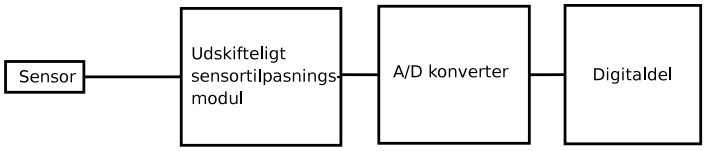


Figur 2.2: Principdiagram for signaltilpasning ved sensor.

Det ses umiddelbart at selve logningsenheden bliver simpel, tilgængæld bliver sensorenheden relativt kompliceret. Fordelen ved denne fremgangsmåde er at det er op til designeren af sensorenheden at leve op til de krav om signalniveauer mv. dataloggeren måtte stille. Ulempen er selvfølgelig at sensorenheden bliver mere kompliceret og dermed også fysisk større, en klar ulempe når sensorerne tænkes anvendt på den menneskelige krop. Et yderligere problem med løsningen er at tilpasningshardwaren skal duplikeres hvis flere sensorer af samme type skal anvendes, idet en multiplekser skal placeres centralt for at kunne vælge mellem de enkelte, fysisk adskilte, sensorer. Endeligt betyder løsningen at der skal udvikles og bygges hardware for hver type sensor der skal interfaces.

2.1.2 Modulopbygget signal tilpasning

En anden løsning, som overkommer problemet med at duplikere sensor tilpasningshardwaren, er at opbygge en platform med en række udvidelses- porte. Disse porte kan bestykkes med moduler som kan udføre den nødvendige signaltilpasning for en specifik type sensor. Figuren 2.3 viser et principdiagram for denne løsning.

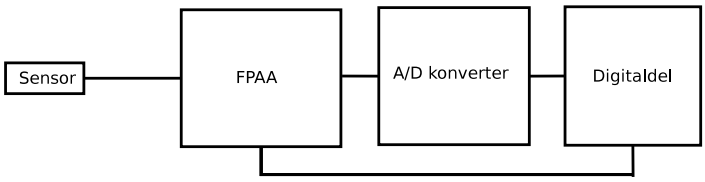


Figur 2.3: Principdiagram for modulopbygget signal tilpasning.

Den modulopbyggede system kan gøres mere eller mindre avanceret. I sin simpleste form, består modulerne af simpel analog signaltilpasning. En mere avanceret opbygning kunne opnåes ved at inkludere f.eks. en EEPROM på modulerne. Denne kunne så tilgåes af den centrale processor, for derved at kunne bestemme typen af modul. Ligeledes kunne lookup tabeller placeres i EEPROM'en på modulet, så eksempelvis liniarisering af det målte signal kunne foretages af den centrale processor ved at slå op i disse tabeller. Fordelen ved dette er naturligvis at disse ting kan udføres uden at skulle ændre i programmet til den centrale processor. Ulempen ved løsningen er at der stadigt skal udvikles og bygges hardware til hver enkel type sensor. En fordel er derimod at selve sensoren ikke behøver hardware tilkoblet direkte, hvilket er med til at reducere den fysiske størrelse af denne.

2.1.3 FPAA baseret signaltilpasning

Den sidste løsning involverer en relativ ny og ukendt teknologi: FPAA [1]. Denne teknologi vil blive beskrevt i større detalje senere i rapporten. Her skal blot nævnes at FPAA er en analog ækvivalent til en FPGA[2]. Med andre ord en universel programmerbar analog kreds, som gør det muligt dynamisk at opbygge diverse analoge kredsløb, blot ved at indlæse en bitstrøm i FPAA'en. På figur 2.4 ses princippet for denne løsning.



Figur 2.4: Principdiagram for FPAA baseret signal tilpasning.

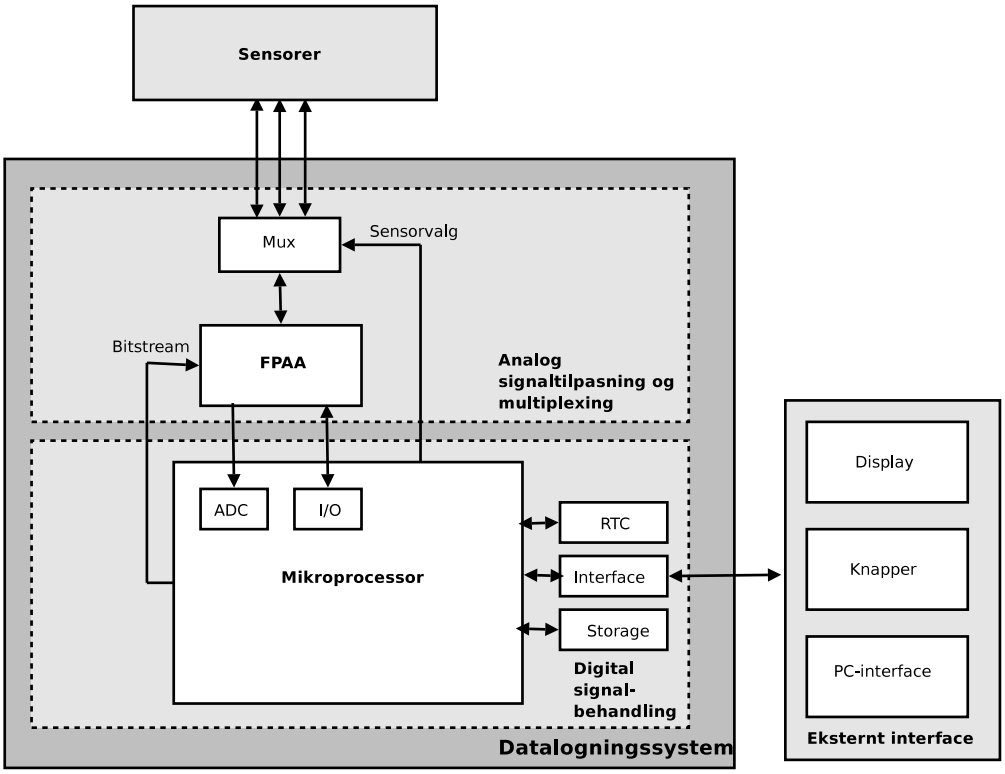
Med introduktionen af en FPAA i systemet bliver signaltilpasningen fuldt programmerbar. Således er der ikke behov for at udvikle og bygge hardware til hver enkelt sensor type. Istedet kan kredsløbet designes i software, og derefter indlæses i FPAA'en via den centrale processor. Herved opnåes et ekstremt fleksibel system, som tillader hurtig prototyping af forskellige sensorer. Ulempen ved metoden er at FPAA er en relativ ny og uprøvet teknologi, derfor bliver ud-

[1] Field Programmable Analog Array
[2] Field Programmable Gate Array

viklingen af selve dataloggeren mere kompleks. Yderligere er prisen for FPAA'er stadig relativ høj, pga. den begrænsede udbredelse. Endeligt er teknologien endnu ikke egnet til brug ved højere frekvenser, dette er dog ikke et problem i dette projekt.

2.1.4 Valg af løsning

Af de tre ovenfor beskrevne løsninger, kræver de to at der udvikles og bygges hardware for hver sensor typer der skal interfaces. Tilgengæld kan disse løsninger holdes på et forholdsvist billigt prisniveau, forusat at det er et begrænset antal sensorer der skal interfaces. For disse løsninger, taler også at de er baseret på kendt og gennemprøvet teknologi. Den første løsning kan dog udelukkes, idet dette projekt sigter på at sensorerne skal kunne placeres på den menneskelige krop. Det er derfor nødvendigt at sikre at den udviklede løsning resulteter i så små sensorer som overhovedet muligt, et krav som denne løsning ikke lever op til.
Den tredie løsning (FPAA løsningen), er den mest fleksible, men samtidigt også den dyreste. Yderligere er der risikoen ved at arbejde med en forholdsvis ny teknologi, hvor det kan være svært at finde hjælp til at løse specifikke problemer. Det er dog blevet vurderet at disse problemer ikke ville blive så store at de ikke ville kunne løses, idet der tilsyneladene er god hjælp at hente i de applicationnotes som producenten af FPAA'erne har på sin hjemmeside. Endelig er prisen ikke tillagt den store betydning, idet det ikke forventes at det endelige produkt skal produceres i store styktal. Det endelige valg falder altså på den tredie løsning: FPAA-baseret signaltilpasning.
Figur 2.5 viser et blokdiagram for det samlede system. Som det ses styrer en central processor hele systemet. Den sørger først og fremmest for at programmere FPAA'en via en seriel bitstream. Derefter vælger den den korrekte sensor vha. multiplekseren, så den kan udføre en måling med A/D konveretern. Processoren behandler herefter data fra A/D konverteren tidsstempler disse med hjælp fra Real Time Clocken, og gemmer slutteligt data på en form for storage. Systemet indeholder også et eller flere interfaces det gør det muligt at overvåge/styre systemet, via enten en PC eller et seperat kontrolmodul med display og knapper.



Figur 2.5: Blokdiagram for den valgte løsning.

Kapitel 3

Digital del

I dette kapitel beskrives den digitale del af systemes hardware. De enkelte dele vil blive beskrevet for sig, selv om der naturligvis eksisterer et overlap mellem dem.

3.1 Processor

Processoren er den centrale komponent i systemet. Kravene til processoren er som følger:

1. Skal have nok regnekraft til at understøtte systemets funktion.
2. Skal være en Single Chip processor, for at minimere systemets samlede størrelse
3. Skal indeholde RAM/ROM så eksterne komponenter undgåes.
4. Nok generelle I/O ben til at understøtte systemets funktion.
5. Timer, interruptcontroller mv.
6. Skal gerne indeholde følgende periferienheder:
   (a) UART, til pc kommunikation.
   (b) Real Time Clock.
   (c) A/D konverter.
   (d) Serielle interfaces, f.eks. SPI, til diverse kommunikation.
7. Processoren skal kunne loddes i hånden.

Ud over de ovennævnte krav, ville det være en absolut fordel hvis processoren understøtter et godt udviklingsmiljø, med mulighed for, på fleksibel vis, at debugge software. Evt. kan dette opnåes ved at benytte en simulator til den valgte processor. Helt optimalt ville det imidlertid være hvis der var adgang til en hardware emulator til processoren. Dette er dog nok uden for dette projekts rammer, og er derfor ikke et krav.

Da projektgruppen tidligere har arbejdet med en AM186ER processoren har det været naturligt at tage denne i betragtning til dette projekt. Adaptronics gruppen har imidlertid en række projekter hvor der benyttes AVR processorer, specielt ATMega128. Denne processor har derfor også relevans for projektet. I tabellen 3.1 ses de væsentligste egenskaber ved de to processorer.

	AM186ER	**ATMega128**
Producent	AMD	Atmel
Processor type	16 bit CISC	8 bit RISC
Arkitektur	Von Neumann	Harvard
Maks. clock	50 MHz	16 MHz
RAM	32 KB	4 KB
ROM	N/A	128 KB FLASH
EEPROM	N/A	4 KB
GPIO	32	53
Timer	3 á 16 bit	2 á 8 bit + 2 á 16 bit
Interrupt controller	ja	ja
DMA	ja	nej
Watchdog	ja	ja
UART	1	2
SPI	N/A	1
I2C	N/A	1
A/D	N/A	8 kanals 10 bit SAR
Debug interface	N/A	JTAG
Hustype	100 pin TQFP/PQFP	64 pin TQFP/MLF

Tabel 3.1: Sammenligning af processorer.

Som det ses af tabellen, er der tale om to vidt forskellige processorer. AM186ER processoren er den kraftigste af de to, med tilgengæld også den der har færrest indbyggede periferienheder. ATMega128 processoren har derimod stort set alle de indbyggede features som man kunne ønske sig, den er til gængæld ikke så kraftig. Bla. har den et indbygget JTAG interface som kan benyttes til at udføre avanceret debugging af software såvel som hardware, hvilket er en klar fordel. En anden væsentlig forskel er at selv om begge processorer har indbygget RAM, så er det kun ATMega128 som har indbygget en FLASH hukommelse til selve programmet. Det betyder at der skal forbindes en ekstern FLASH, eller lignende, til AM186ER processoren før den kan bruges. Alt i alt passer ATMega128 processoren bedst til projektet, idet det vurderes at den, selv om den er langsommere, trods alt stadigt er hurtigt nok til dette formål.

3.2 Storage

Formålet med storage delen af systemet, er at gemme de opsamlede data. Da systemet kan tænkes anvendt til at opsamle data i en længere periode af gangen, skal storagekapaciteten være stor nok til at imødekomme dette, uden behov for

at udlæse data midt i en måleperiode. Hvis det antages at en måling fylder 6 bytes (4 bytes til tidsstempel og 2 bytes til selve målingen), vil en times logning hvor der logges hvert 100 ms fylde: $6 \cdot 10 \cdot 60 \cdot 60 = 216000 bytes \approx 211 KB$. Med denne mængde data per time, koblet med det faktum at det er ønskeligt at kunne logge over en periode på adskellige timer, vil det betyde at et minimum af storagekapacitet på omkring 1 MB er ønskeligt. Et yderligere krav til storagesystemet er naturligvis at det skal være non-volatile, dvs. data skal ikke glemmes hvis strømmen bliver afbrudt. En række forkellige teknologier er blevet undersøgt, for at finde den bedst egnede. I tabel 3.2 ses typiske egenskaber for en række af disse.

Teknologi	**Kapacitet (typ.)**	**Interface**	**Filsystem krævet**
EEPROM	512 KB	SPI	nej
SRAM [a]	512 KB	Parallelt	nej
FLASH	1 MB	SPI	nej
DataFLASH [b]	8 MB	SPI	nej
Compact FLASH	256 MB	ATA	ja
SD/MMC card	256 MB	SPI	ja

Tabel 3.2: Sammenligning af storage teknologier.

[a] Kræver batteri backup
[b] Produceres af Atmel

Som tabellen viser er der et par teknologier som kræver mere end en enhed, f.eks. EEPROM. SRAM teknologien kræver som angivet et backup batteri for at kunne leve op til kravet om non-volatile storage. Disse to faktorer udelukker disse teknologier, da disse vil komplicere systemet. FLASH og DataFLASH er mere lovende teknologier, idet disse kan leveres i kapaciteter der er interessante for projektet. Nå disse alligevel er valgt fra skyldes det et ønske om at gøre systemet så brugervenligt som muligt. Hvis en af disse teknologier benyttes skal data nemlig overføres via en seriel (RS232) forbindelse fra systemet til den PC hvor data skal behandles. Dette er en process som er langsommelig, og som besværes af det faktum at mange moderne PC'er ikke længerer leveres med et RS232 interface.
Derfor fokuseres der på en løsning baseret på et memory kort, der kan tages ud af systemet, efter endt logging, og tømmes for data via en PC med en kortlæser. Denne løsning har dog den store ulempe at den kræver at systemet udstyres med software der kan håndtere et filsystem på kortet. Denne er problematik diskuteres nærmere i softwaredelen af rapporten.
De to memorykort teknologier der er blevet undersøgt er begge baseret på FLASH, og adskiller sig primært på deres interface. Compact FLASH benytter et parallelt interface baseret på ATA standarden. Et interface som er nemt at have med at gøre, men som dog har den ulempe at den kræver et stort antal I/O ben, i sin parallelle natur. SD/MMC card benytter derimod den serielle SPI standard, som ATMega128 processoren allerede understøtter i hardware. Det er derfor naturligt at vælge denne løsning. Dog har SD/MMC kort den ulempe at det er en 3,3 volts teknologi, mens den benyttede processor er 5 volt,

det kræver altså lidt levelshifter logik at interface de to, men da det drejer sig om ganske få forbindelser er dette ikke det store problem. Hastighedsmæssigt er der ikke forskel på de to Compact FLASH og SD/MMC kort i dette system pga. den begrænsede processorhastighed.

3.3 Real Time Clock

Real Time Clocken (RTC) har til opgave at holde styr på tiden, så data kan tidsstemples. ATMega128 processoren har en facilitet indbygget som muligører at implementere en RTC. Det eneste det kræver er at den udstyres med et 32,768 kHz krystal (ur-krystal). Hermed kan den selv i strømspare tilstand benytte en af de indbyggede timer til at holde en præcis RTC kørende. Ulempen ved denne løsning er dog at tiden skal indstilles hver gang strømmen har været afbrudt, idet der ikke kan tilsluttes et backup batteri. Derfor er det blevet valgt at benytte en ekstern RTC som tilbyder en batteri backup funktion. Da systemet allerede benytter SPI i forbindelse med SD/MMC kortet, ville det været naturligt at vælge en RTC som understøtter dette interface. Gruppen har i et tidligere projekt stiftet bekenskab med en RTC fra Dallas semiconductor ved navn DS1305. Denne RTC kan med et fåtal af eksterne komponenter bringes til at kommunikere med processoren via SPI. Den understøtter ydermere en række forskellige strømforsynings metoder, herunder batteri backup. Den har ligeledes den fordel at den understøtter alarm via interrupt, en feature som kan benyttes til at "vække" processoren på et bestemt tidspunkt, eksempelvis når der skal laves en måling.

3.4 I/O interfaces

I/O interfacet skal gøre det muligt at kommunikere med systemet. Den primære måde at kommunikere med systemet på er via et terminal program på en PC, alternativt via et "user interface" board (UI borad), bestående af et LC-Display og et antal knapper. Kommunikationen med systemet består hovedsageligt i start af logning samt overvågning af logning. Det skal dog være muligt at benytte systemet autonomt, dvs. uden brug af PC eller UI board.
ATMega128 processoren har indbygget 2 UART, hvoraf den ene benyttes til at kommunikere med PC'en via RS232. Det eneste denne løsning kræver, er at tilpasse de logiske niveauer som ATMega128 processoren benytter til de niveauer som RS232 standarden forskriver. Dette kan let løses ved at benytte en levelkonverter chip som eksempelvis MAX232 fra Maxim/Dallas.
UI boardet er en ekstra feature ved systemet, som ikke er nødvendig for systemets operation. Det er derfor blevet besluttet at denne del af systemet ikke nødvendigvis bliver en del af det færdige system, ved projektes afslutning. Det ville dog være hensigtsmæssig hvis systemet var forberedt til et sådan UI board. Dette er opnået ved at føre ATMega128 processorens I2C interface ud til et stik. Det er herefter en forholdsvis let sag at designe og bygge et UI board der kan forbindes til dette stik.

Kapitel 4

Analog del

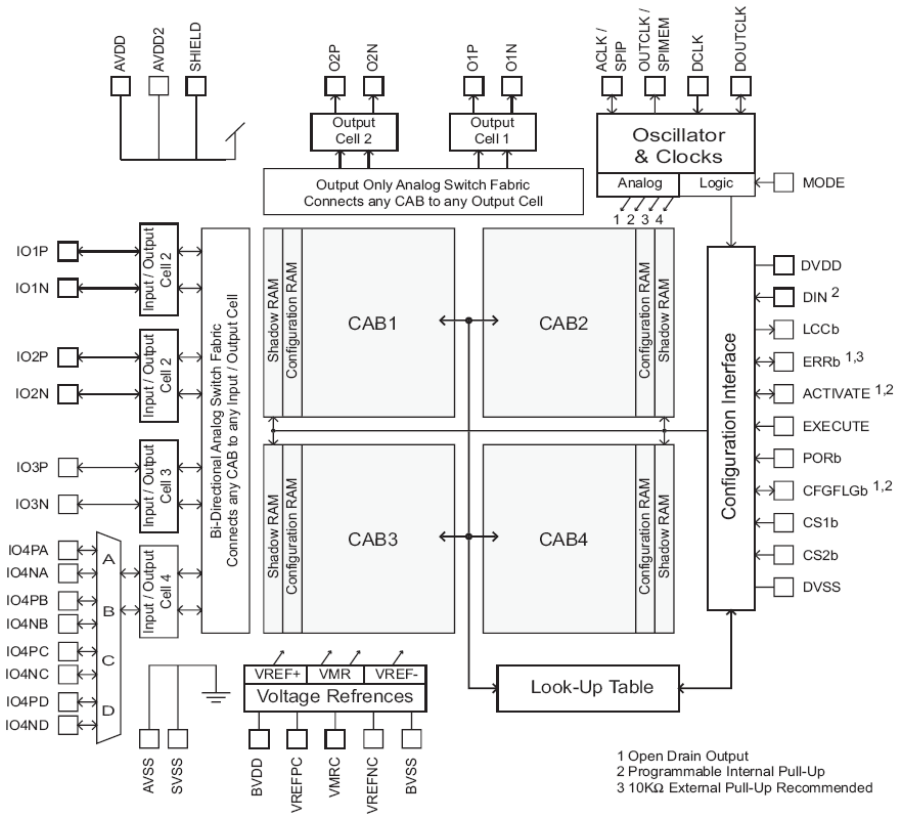
I dette kapitel beskrives den analoge del af systemet. Der vil blive lagt vægt på at beskrive hvorledes den valgte FPAA fungerer, samt hvordan den er koblet til den digitale del af systemet.

4.1 Virkemåde af FPAA

Den i projektet benyttede FPAA chip er fra firmaet Anadigm, som er det eneste firma som har specialiceret sig i denne type kredsløb. FPAA'er fåes i en række forskellige type, men de er dog grundlæggende ens. Det der primært adskiller dem er antallet af CAB's (se afsnit 4.1.2 for en nærmere forklaring om CAB) og om de kan rekonfigureres dynamisk eller ej. I projektet benyttes en FPAA ved navn AN221E04, som har 4 CAB's og kan rekonfigureres dynamisk.
Generelt set tillader en FPAA at man kan implementere en række forskellige analoge funktioner, blot ved at sende en bitstrøm til den. Teknologien er dog forholdsvis ny, og har derfor et par begrænsninger. Den største begrænsning er båndbredden, som i de nuværende FPAA'er begrænset til nogle få hundrede kilohertz. Dette er imidlertid nok til brug i audio applikationer, og ikke mindst dette projekt.
Figur 4.1 viser et overblik over "indmaden" i en FPAA af den type der bruges i projektet. FPAA er delt op i en række moduler:

- **I/O moduler**: Sørger for at analoge signaler kommer ind og ud af kredsen.
- **CAB** [1] **moduler**: Implementerer de egentlige analoge funktioner.
- **Configuration interface**: Sørger for at konfigurere kredsen, med data fra en ekstern kilde.
- **Oscilator & clocks**: Genererer clocksignaler til brug internt i kredsen.
- **Voltage references**: Genererer spændingsreferencer til brug internt i kredsen.

[1] Configurable Analog Block



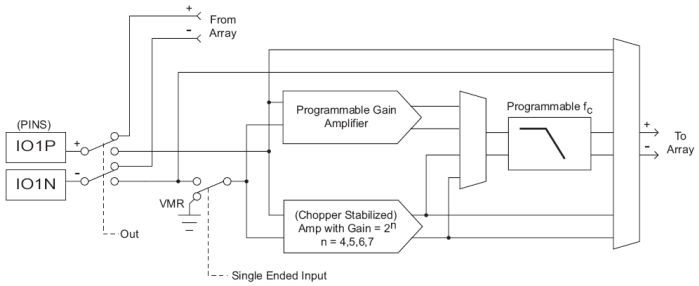
Figur 4.1: Blokdiagram for en FPAA type AN211E04.

Overordnet set behandles et analogt signal i FPAA på følgende måde: Signalet kommer ind via en I/O blok hvor det evt. forbehandles. Derefter routes det til en eller flere CAB's via en indbygget switch fabric, hvor signalet behandles. Endeligt routes signalet, igen via den indbyggede switch fabric, til en I/O blok hvor det evt. efterbehandles før det forlader kredsen. I de følgende afsnit beskrives de enkelte blokke i større detalje.

4.1.1 FPAA I/O blokke

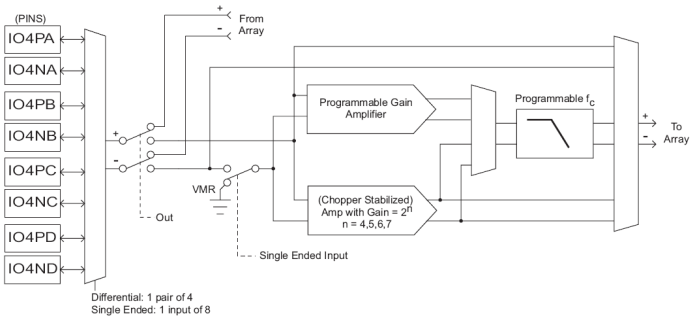
I/O blokke findes i tre varianter. Et modul som kan fungere som både Input og Output. Et næsten identisk modul som blot har en multiplekser indbygget. Og endeligt et modul som kun kan fungere som output modul. Fælles for alle typer af moduler er at de som udgangspunkt er differentielle, med reference til 2 volt, og et input/output område på 0-4 volt. Dette reducerer common mode støj væsentligt, men giver også et lidt mere komplekst design. På figur 4.2 ses et blokdiagram for et rent I/O modul. Til venstre på diagrammet ses selve input/output benene (IO1P og IO1N), mens udgangen til den indbyggede switch fabric ses til højre. Modulet kan skiftes mellem input og output med kontakten nær input/output benene. Når modulet fungere som output modul kommer signalet direkte fra CAB modulet, uden nogen form for mellemled.

Når modulet derimod fungerer som input, er der en række features som kan benyttes. Først og fremmest tilbyder modulet at konvertere fra single-ended input til differentiel input, i så fald kobles den negative ingangsterminal til VMR [2], som er på 2 volt. Derudover indeholder modulet to forskellige typer af forstærkere: En med en programmerbar gain på 16, 32, 64 eller 128 gange, og en med samme gain indstillinger, men som er chopper stabilised, hvilket betyder at offset fejl reduceres betydeligt. Tilgengæld bruger den mere strøm, samtidig med den støjer lidt mere. For en længere forklaring på denne type forstærker henvises til Anadigm dokumentation, eller [Analog, 2000]. Endelig indeholder modulet et anden-ordens antialiasing filter, der kan benyttes til at båndbreddebegrænse indgangssignalet.



Figur 4.2: Blokdiagram for FPAA I/O modul.

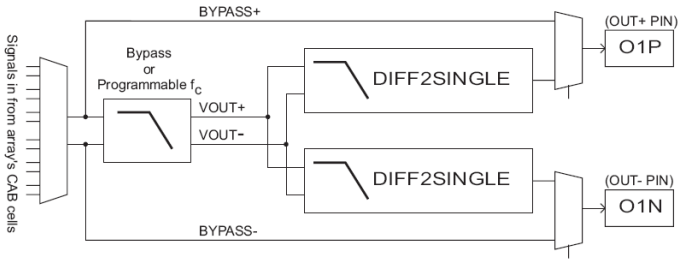
Figur 4.3 viser et I/O modul med indbygget multiplekser. Som det ses er modulet identisk med det føromtalte I/O modul. Blot er der indsat en multipleker der gør det muligt at vælge mellem flere sæt af input/output terminaler.



Figur 4.3: Blokdiagram for FPAA I/O modul med multiplekser.

Figur 4.4 viser et rent output modul. Modulet modtager et signal fra den indbyggede switch fabric og giver mulighed for at efterbehandle dette før det sendes til output terminalerne. I lighed med I/O modulerne er der mulighed for at føre signalet igennem et programmerbart anden-ordens rekonstruktionsfilter. Der er også muligt at føre signalet igennem en differentiel til single ended konverter, således at kun den ene udgangsterminal benyttes.

[2] Voltage Main Reference



Figur 4.4: Blokdiagram for FPAA out modul.

Alt i alt udgør I/O modulerne en særdeles fleksibel løsning. Modulerne frigør endda ressourcer i CAB modulerne, idet det er muligt både at for- og efterbehandle signalerne.

4.1.2 FPAA CAB

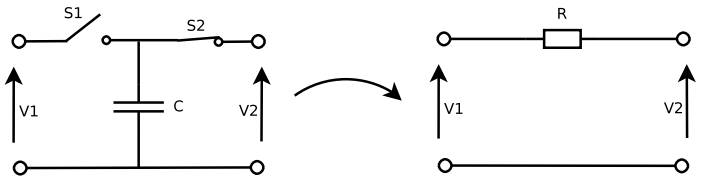
Hele hemmeligheden ved FPAA'erne ligger i CAB modulerne. Det er her det er muligt at realisere de forskellige analoge kredsløb, vha. af såkaldte CAM[3] blokke. En CAM er en analog byggeklods, som f.eks. en forstærker eller et filter. Et CAB modul kan implementere en eller flere CAM blokke, vha. switch capacitor teknologi. Formålet med disse CAM byggeklodser er at gøre det let at implementere analoge funktioner, ved blot at vælge forskellige CAM'er fra et bibliotek og sætte disse sammen til er færdigt kredsløb. Til at understøtte dette findes der software, som via et visuelt interface muligør denne opbygning af kredsløb. Herunder er et uddrag af de forskellige CAM typer som softwaren understøtter:

- Forstærkere.
- Hel- og halv-bølge ensrettere.
- Summations forstærkere.
- Filtre en- eller to-polede, høj-, lav- og bånd-pas eller båndstop.
- Differentiator/Integrator.
- Multiplier.
- Generator af periodiske signaler.
- Vilkårlig overføringsfunktion.
- Komperator.
- Sample and hold.
- Sinus oscillator.
- Spændingsreference.

[3]Configurable Analog Module

4.1.2.1 Switched capacitor teknologi

Grundstenen i switched capacitor teknologi er muligheden for at emulere modstande med kondensatorer. Dette er vigtigt da modstande er svære at lave med gode tolerancer på en chip. Kondensatorer (små kapaciteter) er langt lettere at lave, ydermere fylder de mindre. Derudover er der behov for at lave variable modstande, noget som er endog meget svært at lave på en chip. Dette kan dog løses ved at emulere disse modstande med kondensatore.
Figur 4.5 viser princippet for hvordan en kondensator kan benyttes til at emulerer en modstand.



Figur 4.5: Principdiagram for switched capacitor modstandsækvivalent.

Kontakterne `S1` og `S2`, er i virkeligheden implementeret med MOSFET transistore, men er blot vist som kontakter her, for at fremme overskueligheden. De styres af et fælles kloksignal, men i modfase, forstået på den måde at når `S1` er åben er `S2` lukket, og omvendt. Når `S1` er lukket oplades kondensatoren `C` til `V1`, svarene til en ladning på $Q = C \cdot V1$. Det samme sker når `S2` er lukket, blot med `V1` erstattet med `V2`. Der opnåes derfor samlet set en ladningsoverførelse på:

$$\Delta Q = C \cdot \Delta V \Rightarrow \tag{4.1}$$
$$\Delta Q = C \cdot (V1 - V2) \tag{4.2}$$

Hvis det fælles kloksignal til kontakterne har en frekvens på f_{clk}, bliver den overførte ladning per tidsenhed:

$$f_{clk} \cdot \Delta Q = f_{clk} \cdot C \cdot (V1 - V2) \Rightarrow \tag{4.3}$$
$$\frac{\Delta Q}{\Delta t} = f_{clk} \cdot C \cdot (V1 - V2) \tag{4.4}$$

Da strøm er det samme som ladning per tidsenhed, kan ovenstående skrives som:

$$I = f_{clk} \cdot C \cdot (V1 - V2) \tag{4.5}$$

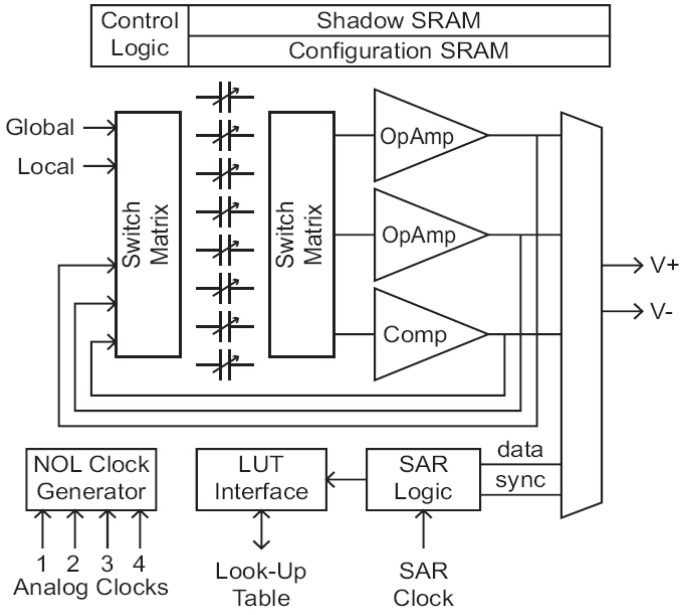
Ved at omarrangere dette fåes:

$$\frac{1}{C \cdot f_{clk}} = \frac{V1 - V2}{I} = R \tag{4.6}$$

Af ovenstående ses at det er muligt at emulere en modstand, vha. en kondensator, og at modstandens værdi er omvendt propertionel med kondensatorens værdi og frekvensen. Dette betyder at forholdsvis store modstande kan emuleres med små kapaciteter. Det er dog vigtigt at huske på at dette er et diskret tid system, der ikke udviser den kontinuert tid sammenhæng mellem strøm og spænding som en rigtig modstand gør. Dette er dog ikke noget problem sålænge det behandlede signal overholder $f_{signal} << f_{clk}$.

4.1.2.2 Opbygning af CAB

Figur 4.6 viser et blokdiagram over et CAB modul.



Figur 4.6: Blokdiagram for FPAA CAB modul.

Kernen i et CAB modul, er de 8 variable kondensatorer der ses i midten. Disse består af en masse ens kondensatorer som kan sættes sammen og på den måde fungere som en variabel kondensatorer. CAM byggeklodserne er ikke direkte afhængige af værdien af disse kondensatore, men snarere af forholdet mellem værdierne. Dette er en stor fordel fordi det er svært at fabrikere en kondensator med en præcis værdi, men det er til gengæld let at sikre at spredningen af værdier mellem kondensatorerne er meget lille typisk omkring 0.1%. Et CAB modul indeholder et par operationsforstærkere og en komperator. Disse kan forbindes på forskellig vis med hinanden og de variable kondenstore via to switch matricer, tilbagekobling kan ligeledes opnåes på denne måde. Hvordan tingene skal kobles sammen bestemmes af de bits som opbevares i konfigurations SRAM'en, disse bestemmer den aktuelle konfiguration. De bits som gemmes i den såkaldte Shadow SRAM, er den næste konfiguration, så for at skifte konfiguration skal disse blot kopieres over i konfigurations SRAM'en, en procedure

som kan fortages på 1 klokcyklus. Der er med andre ord muligt at skifte konfiguration dynamisk, idet en ny konfiguration kan indlæses i shadow SRAM uden at forstyrre den aktuelle konfiguration. Ud over de omtalte blokke indeholder et CAB modul også et interface til en central Lookup tabel (LUT) samt et succesiv approksimation register (SAR). Disse giver tilsammen mulighed for at implementere specielle funktioner, som f.eks. vilkårlige overføringsfunktioner og signalgeneratorer.

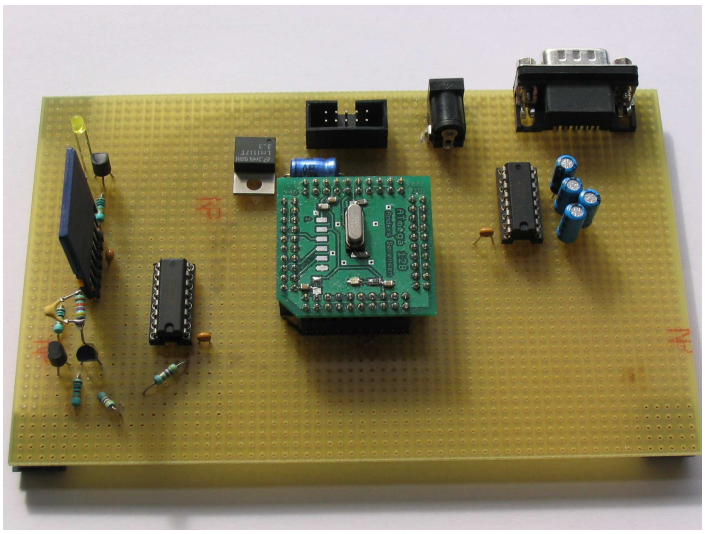
4.2 A/D konverter

Den anvendte processor ATMega128 har som tidligere nævnt allerede en A/D konverter indbygget. Denne konverter har en del features. Den har grundlæggende 8 singleended kanaler på 10 bit, men disse kan også benyttes som differentielle kanaler, hvilket er ganske nyttige i dette projekt, da udgangene fra FPAA'en er differentielle. Konverteren har en maksimal samplingsfrekvens på omkring 70 ksps, hvilket er rigeligt til denne applikation. Det er derfor belevet besluttet at benytte denne konverter i den første prototype. Ikke mindst fordi der kun er mulighed for at lave print i to lag, og dette kan vise sig at være afgørende for hvor meget støj der bliver på signalerne. Det kan være at konklusionen på prototypen, bliver at støj ikke er noget problem, og at der derfor godt kan anvendes en bedre konverter med en større opløsning.

Kapitel 5

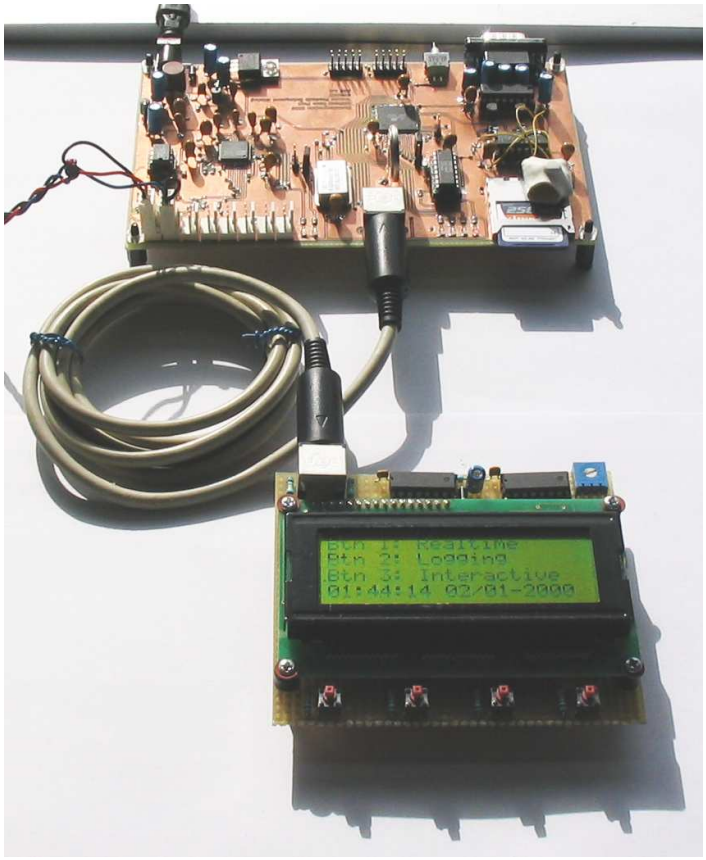
Implementation

I dette kapitel beskrives den egentlige implementation af systemets hardware. Hardwaren er designet og bygget i tre etaper, første etapes mål var at opbygge en prototype på "fumleprint" med en ATMega128 processor og den elektronik der skal til at interface et SD/MMC kort, se figur 5.1.



Figur 5.1: Prototype opbygget på fumleprint.

Denne prototypes formål var at skabe et udgangspunkt for softwareudviklingen, idet det blev vurderet at en af de største softwaremæssige udfordringer var at få "hul igennem" til SD/MMC kortet og filsystemet. Med denne prototype på plads kunne udviklingen af en mere komplet prototype påbegyndes. Denne prototype indeholder alle systemets komponenter, og målet med den var at verificere systemest samlede design, samt at fungere som test platform, se figur 5.2.
Den sidste etape i projektet var at benytte den udviklede hardware som test platform og komme med eventuelle korrektioner til designet, så disse kunne danne basis for en ny prototype, eller et endeligt design. I de følgende afsnit vil det blive beskrevet hvordan de enkelte dele af hardwaren er implementeret, i



Figur 5.2: Den endelige prototype med UI board.

bilagene kan de enkelte diagrammer studeres.

5.1 Spændingsforsyning

Systemet kræver en række forskellige spændinger for at kunne fungere. 5 volt kræves til mange digitale kredsløb, herunder processor, FPAA og RTC. Processoren og FPAA'en kræver både 5 volt digital og analog forsyning, disse adskilles med et filter bestående af et par kondensatorer og et par ferrit perler. De 5 volt bliver forsynet direkte til systemet via en netadapter. SD/MMC kortet kræver 3.3 volt, en LM1117T benyttes til at fremskaffe denne spænding. Endeligt kræver visse dele af den analoge del af kredsløbet -5 volt. Denne spænding fremskaffes ved hjælp af en switched capacitor voltage converter af typen LM2663, koblet i en standard kobling, dog med lidt ekstra afkobling.

5.2 Systemclock

Da både processoren og FPAA'en kræver et clocksignal for at fungere, og da begge kan køre på samme frekvens, er det naturligt at konstruere et samlet

clockkredsløb til disse komponenter. Processorens maksimale clockfrekvens er 16 MHz, samme frekvens som Anadigm anbefaler at benytte til FPAA'en. Clocksignalet fremskaffes derfor helt enkelt vha. en krystaloscillator på 16 MHz, som blot kræver 5 volt for at fungere. Dette design har yderligere den fordel at systemet vil støje mindre, end hvis der var benyttet to clockkredsløb.

5.3 Processor og supportkredsløb

Processoren kræver forbavsende lidt for at kunne fungere. Faktisk behøver den kun et clockkredsløb og en spændingsforsyning. Processoren indeholder selv resetkredsløb og brownout detektor, en ekstern resetknap er dog inkluderet i designet for at lette softwareudviklingen. Ligeledes for at assistere softwareudviklingen er der inkluderet et JTAG interface i form af et JTAG stik. JTAG interfacet er implementeret således at systemet ikke skal strømforsynes via JTAG, men via sin egen forsyning. Reset er ført igennem til JTAG stikket, således systemet kan styres fuldstændig fra JTAG. Selv om systemets primære programmeringsinterface er JTAG, er det også implementeret en ISP port, da denne kan benyttes til at redde systemet, hvis JTAG funktionaliteten ved et uheld bliver slået fra.

5.4 SD/MMC interface

SD/MMC kort har en række interface options, eller modes. Den hurtigste og mest avancerede af disse er SD mode (kun SD kort understøtter denne mode naturligvis). Desværre er specifikationerne for denne mode ikke offentligt tilgængelige, hvorfor den ikke er interessant i denne sammenhæng. Heldigvis understøtter MMC kort SPI interfacet, og da SD kort er kompatible med MMC kort understøtter de også denne standard. SPI er kort fortalt en synkron seriel master/slave busforbindelse med tre forbindelser: MISO [1], MOSI [2] og SCLK [3]. Derudover kræves et chipselect til hver enhed på bussen, idet addressering ikke understøttes som det f.eks. er tilfældet med I2C standarden. SPI masteren i systemet er naturligt nok processoren, og slaverne udgøres af SD/MMC kortet, RTC'en og FPAA'en. En række generelle I/O (GPIO) ben på processoren udgør chipselect til de enkelte enheder. Mens FPAA'en og RTC'en begge er 5 volts enheder, er SD/MMC kortet en 3.3 volts enhed, hvorfor levelshifting er nødvendig på den del af SPI bussen hvor denne er tilkoblet.

5.4.0.3 Levelshifting

Tre signaler, Chipselect, SCLK og MOSI, skal levelshiftes fra 5 volt til 3.3 volt, idet disse kommer fra processorens udgange og er forbundet til SD/MMC kortets indgange. Men kun et signal, nemlig MISO skal levelshiftes fra 3.3 volt til 5 volt,

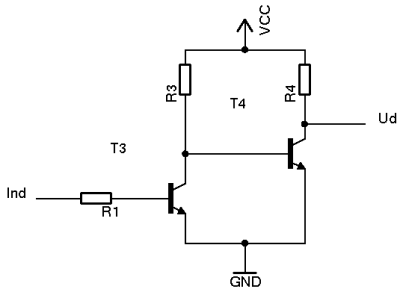
[1] Master Out Slave In
[2] Master Out Slave In
[3] System Clock

idet dette er data out fra kortet til processoren. I Databladet for henholdsvis ATMega128 processoren og SD/MMC kortet er angivet hvilke spændingsniveauer der opfattes som høj og lav, angivet i forhold til forsyningsspændingen. Tabel 5.1 sammenfatter niveauforskellene mellem de to komponenter i begge retninger, ved en V_{cc} på 5 volt og 3.3 volt for henholdsvis processor og SD/MMC kortet.

Fra ATMega128 til SD/MMC kort					
	$ATMega128_{ud}$		SD/MMC_{ind}		
	Min	Max	Min	Max	
Høj	4.2		1.98	3.6	[V]
Lav		0.7	-0.3	0.825	[V]
Fra SD/MMC kort til ATMega128					
	SD/MMC_{ud}		$ATMega128_{ind}$		
	Min	Max	Min	Max	
Høj	2.75		3	5.5	[V]
Lav		0.4125	-0.5	1	[V]

Tabel 5.1: Sammenligning af signalniveauer.

Tabellen viser tydeligt at der er behov for at konvertere niveauerne begge veje. Høj_{ud} spændingen fra processoren er helt klart for høj til SD/MMC kortet. Omvendt er Høj_{ud} spændingen fra SD/MMC kortet for lav til processoren. Niveaukonverteringen fra processoren er lavet med en 4050 buffer kreds, med seks buffere, hvoraf tre benyttes. Fordelen ved denne kreds er at den kan tåle højere spænding på sine indgange end dens forsyningsspænding. Det er derfor muligt at forsyne den med 3.3 volt og benytte 5 volt signaler på dens indgange. Dens udgangsspændinger for Høj_{ud} og Lav_{ud} ligger meget tæt (inden for 0.1 volt) på henholdsvis V_{cc} (3.3 volt) og 0 volt, hvilket ligger indenfor grænserne af SD/MMC kortets niveauer. Fra SD/MMC kortet til processoren er der kun et signal der skal niveaukonverteres, så derfor benyttes til simpelt transistortrin til at fortage denne konvertering. Denne løsning er dog ikke uden problemer som beskrevet i afsnit 6.1.3. Transistortrinnet består af to transistorer, som det ses på figur 5.3.



Figur 5.3: Transistortrin til levelshifting.

Forsyningsspændingen V_{cc} til transistortrinnet er på 5 volt, mens indgangsspændingen til trinnet kan forventes max at være på ca. 0.4 volt for et lavt signal, og min 2.75 volt for et højt signal, jævnfør tabel 5.1. For at dimensionere trinnet startes "bagfra". Første vælges R_c for den sidste transistor i trinnet, så den ønskede I_c opnåes. Hernæst beregnes I_b for denne transistor vha. formel 5.1.

$$I_b > \frac{V_{cc} - V_{ce}}{\beta \cdot R_c} \tag{5.1}$$

Herefter kan R_b beregnes vha. formel 5.2. Bemærk at denne formel ganger resultatet fra 5.1 med 10, dette er for at sikre at transistoren drives i saturation.

$$R_b = \frac{V_b - V_{be}}{10 \cdot I_b} \tag{5.2}$$

Den beregnede basismodstand bliver så kollektor modstand for den næste transistor der så kan dimensioneres efter samme principper. Her skal det blot huskes at indgangsspændingen til dette trin skal sættes til det laveste niveau som SD/MMC kortet giver som Høj niveau (2.75) volt, da dette er worst case. Lavt niveau fra SD/MMC kortet er ikke noget problem, da dette er under V_{be}, og dermed garantere at transistoren drives i cutoff.

5.5 Real Time Clock

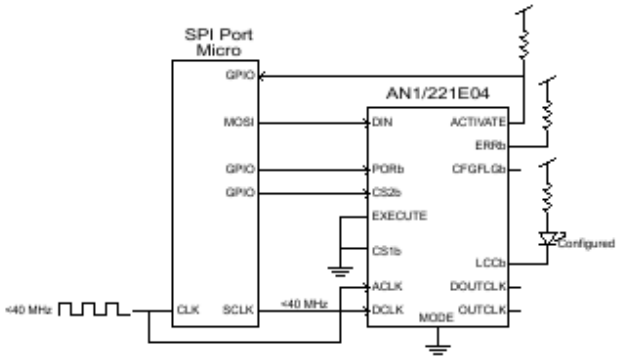
RTC kredsløbet er relativt simpelt opbygget omkring en DS1305 chip. Denne chip understøtter både SPI og I2C protokollen, her er SPI protokollen valgt ved at forbinde SERMODE benet til ground. RTC'en benytter et 32.768 kHz krystal til at holde tiden. RTC'en er forbundet direkte til SPI bussen, idet det er en 5 volts enhed som ikke behøver levelshifting. Som før omtalt understøtter RTC'en en alarmfunktion, via et interruptben, dette er derfor forbundet til processoren, med den fornødne pullup modstand. Endeligt understøtter RTC'en et backupbatteri, som tager over når primærforsyningen kommer under et vist niveau, på den måde fungerer RTC'en stadig selv om hovedforsyningen afbrydes.

5.6 I/O interface

I/O interfacet består af to dele: Et RS232 interface og et I2C interface. Som tidligere beskrevet benyttes det til henholdsvis PC tilslutning og UI board tilslutning. I2C interfaces er ganske simpelt implementeret ved at føre processorens I2C ben til et stik. RS232 interfacet derimod kræver en niveaukonvertering. Denne er udført med den kendte MAX232 chip, som indeholder alt nødvendigt isenkram, inklusiv charge pump til at producere de nødvendige spændinger. Alt der kræves er en håndfuld eksterne kondensatorer og et DB9 stik.

5.7 FPAA digitalt interface

FPAA'en er meget fleksibel hvad angår konfiguration. Den kan både boote fra en seriel EEPROM, og en FPGA boot EEPROM, desuden er det muligt at sende konfigurationsdata til den direkte fra en processor via SPI. Alle disse muligheder, koblet med det faktum at FPAA'en er designet til at arbejde sammen med andre FPAA'er, gør at det digitale interface bliver en smule kompliceret at arbejde med. Bla. har mange af benene mere end en funktion, nogle skifter endda fra at være input til at være output og omvendt, under konfigurationsprocessen. FPAA'en understøtter grundlæggende to typer af konfigurationer, en primær konfiguration, der skal indlæses efter Power On Reset, og skal indeholde alle bits til at beskrive et helt kredsløb. Den understøtter også en "update" konfiguration, hvor det kun er de ændrede bits mellem to forskellige kredsløb der skal overføres. Se software delen af rapporten for en mere uddybende diskusion af dette.



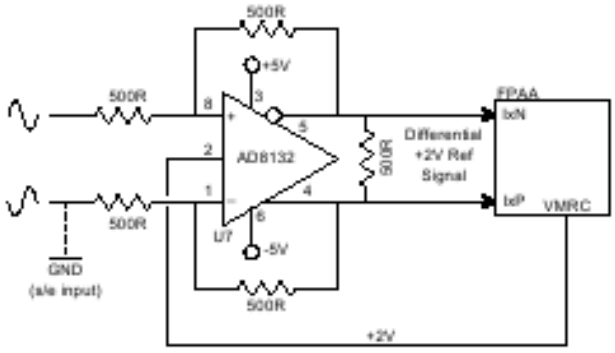
Figur 5.4: Typisk tilslutning af FPAA til hostprocessor via SPI.

Figur 5.4 viser hvordan en FPAA typisk forbindes til en hostprocessor. FPAA'en har brug for to clocksignaler, et digitalt clocksignal (DCLK) som benyttes i forbindelse med SPI overførslen, og som derfor forbindes til SPI bussens clocksignal. Det analoge clocksignal (ACLK), benyttes internt i FPAA'en til at drive det analoge switched capacitor kredsløb. Dette clocksignal er det samme 16 MHz clocksignal som ATMega128 processoren benytter, og kommer fra den centrale clockgenerator. Data in (DIN) er forbundet til SPI bussens MOSI, så data kan indlæses i FPAA'en. Bemærk at FPAA'en ikke har en Data out, og MISO på SPI bussen benyttes derfor ikke. Power On Reset (PORb) benet på FPAA'en er forbundet til er I/O ben på processoren, så det er muligt at resette FPAA'en før en primær konfiguration indlæses. FPAA'en har to chipselect ben (CS1b og CS2b), hvoraf det ene er forbundet permanent til ground, mens det andet benyttes til at aktivere kredsen via et I/O ben på processoren. Endeligt er benet Activate på FPAA'en forbundet til et I/O ben på processoren, dette ben går lavt når en konfiguration er indlæst og aktiveret, processoren kan hermed afgøre om FPAA'en er klar til brug. Det sidste ben som har interesse i denne forbindelse er MODE benet, hvis dette er højt, er FPAA'en en SPI master som

er istand til at boote fra en EEPROM, hvis det derimod er lavt er FPAA'en en SPI slave som skal styres fra en hostprocessor. Benet er derfor sat lavt.

5.8 FPAA analog interface

FPAA'en råder over syv analoge ind-/ud-gange, hvoraf de fire er er multipleksede, derudover har den to rene udgange. Alle ind- og ud-gange er som tidligere nævnt differentielle, med reference til VMR som er på to volt. Dette kan både være en fordel og en ulempe. En fordel hvis der kan benyttes differentielle signaler, da dette undertrykker indstrålet støj ganske betragteligt. Men en ulempe hvis de anvendte sensorerne ikke har differentielt udgangssignal. Dette kompliceres yderligere af at indgangene ikke har reference til ground, hvis dette var tilfældet kunne den negative indgangsterminal blot forbindes til ground for at skabe en single-ended indgang, men dette er altså ikke muligt. For at råde bod på dette, er to af indgangene udstyret med specielle differentielle operations forstærkere. De andre fem ind-/ud-gange er ført direkte ud på en række stik. Disse kan benyttes som differentielle indgange, men de kan også benyttes som differentielle udgange, eller single-ended udgange (ved kun at benytte den ene terminal). Dette kan være en fordel hvis det skal benyttes en sensor som kræver en form for signal for at kunne fungere.



Figur 5.5: FPAA indgangstrin med differentiel opamp.

Figur 5.5 viser princippet for indgangene der er udstyret med en operations forstærker. Den anvendte operationsforstærker (AD8132) er speciel på den måde at den er fuld differentiel, dvs. at den har differentiel udgang, og selvfølgelig også differentiel indgang. Den har dog en feature mere idet den har en commonmode indgang, som benyttes til at sætte et commonmode signal. I dette tilfælde benyttes denne indgang til at sætte et commonmode niveau til VMR (2 volt) på udgangen. Med andre ord levelshiftes signalet således at det nu har reference til ground. Dette er smart da det derved er muligt at forbinde single-ended signaler med reference til ground, ved blot at forbinde minus-terminalen til ground, og føre signalet ind på den positive-terminal. Anadigm anbefaler at man ikke benytter VMR udgangen fra FPAA'en direkte, derfor er der indskudt

en buffer for dette signal i form af en spændingsfølger implementeret vha. en operationsforstærker. Dimensionering af feedbackkredsløbet omkring AD8132 operationsforstærkeren er lidt anderledes end det kendes fra traditionelle operations forstærkere, da den har differentielle udgange. Analog devices anbefaler at man laver to ens feedbackkredsløb som vist på figuren. Der eksisterer et yderligere feedbackkredsløb inde i kredsen som bestemmer gain på commonmode indgangen. Denne commonmode gain er sat til 1 og er uafhængig af den differentielle gain som bestemmes af de eksterne komponenter. Hvis forholdet mellem de komponenter som sidder i de to feedbackkredsløb er ens fåes en forstærkning på $G = \frac{R_F}{R_G}$. Hvor R_G er modstanden som sidder i indgangen, og R_F er modstanden som sidder i tilbagekoblingssløjfen. Med andre ord giver fire ens modstande en gain på en. Værdien af dissen modstande er valgt ud fra Analog devices anbefalinger med hensyn til modstandsværdi og støj i kredsløbet, se figur 5.6.

Table II. Recommended Resistor Values and Noise Performance for Specific Gains

Gain	R_G (Ω)	R_F (Ω)	Bandwidth –3 dB	Output Noise AD813x	Output Noise AD813x + R_G, R_F
1	499	499	360 MHz	16 nV/Hz	17 nV/Hz
2	499	1.0 k	160 MHz	24.1 nV/Hz	26.1 nV/Hz
5	499	2.49 k	65 MHz	48.4 nV/Hz	53.3 nV/Hz
10	499	4.99 k	20 MHz	88.9 nV/Hz	98.6 nV/Hz

Figur 5.6: Tabel der viser sammenhæng mellem modstandsværdi og støj. Taget fra AD8132 kredens datablad.

Indgangsimpedansen af koblingen er afhængig af om den benyttes i differentiel eller single-ended mode. Hvis den anvendes i differentiel mode er indgangsimpedansen givet ved $R_{in,dm} = 2 \cdot R_G$, hvilket med de anvendte komponentværdier beløber sig til $1k\Omega$. Anvendes den i single-ended mode (den negative indgangsterminal forbundet til ground), er regnestykket imidlertid lidt mere kompliceret, se formel 5.3

$$R_{in,sm} = \frac{R_G}{1 - \frac{R_F}{2 \cdot (R_G + R_F)}} \tag{5.3}$$

Med de anvendte komponentværdier giver dette en indgangsimpedans på ca. 666Ω.
FPAA'ens to udgange er forbundet direkte til processorens A/D konverter med differentielle forbindelser. Da A/D konverteren kan fungere i både single-ended og differentiel mode kan det styres i software om den ene eller den anden mode skal benyttes, idet det ene input blot ignoreres hvis der benyttes single-ended mode. Blot skal det huskes at dette halvere udstyringsområdet. A/D konverterne kan enten benytte den analoge forsyningsspænding, sin egen interne ref-

erence spænding, VMR fra FPAA'en eller også kan en af udgangene fra FPAA'en benyttes som referencespænding. Den sidste mulighed kan benyttes til at generere en reference internt i FPAA'en hvis dette skulle ønskes. De forskellige muligheder kan vælges ved hjælp af en række jumpere.

5.9 UI board

Selv om UI boardet ikke oprindeligt var tænkt implementeret, er det dog opbygget som "proof of concept" på fumleprint. Dette består helt enkelt af et 4x40 tegns LC-Display og fire knapper. Det hele er forbundet til I2C bussen via to stk. PCF8574 kredse. Disse kredse er helt enkle 8bit IO extenders, som gør det muligt at opnå en 8 bit bidirektionel databus der kommunikere med processoren via I2C. LC-Displayet er koblet i 4 bits mode, hvilket betyder at hele displayet kan kobles til en PCF8574. De fire knapper er koblet til den anden PCF8574 via pullup modstande.

5.10 Print og afkobling

At designe et print med både analoge og digitale komponenter, er altid lidt af en udfordring. Denne udfordring bliver bestemt ikke mindre af at skolens udstyr kun tillader at fabrikere print i to lag. Anadigm anbefaler at fremstille i fire lag, og har en række anbefalinger til hvordan printlayoutet bør være. Dette inkluderer seperat digital og analog groundplane, samt en række anbefalinger til hvordan komponenter bør placeres i forhold til disse. Anadigm anbefaler at man ved to-lags print benytter så stor en del af det ubenyttede printareal som muligt til groundplane, da det ikke er muligt at have et dedikeret groundplane i dette tilfælde. Derudover anbefaldes det at benytte to afkoblingskondensatore ved hvert forsyningsben. En $10 - 47\mu F$ tantalkondensator og en $100nF$ keramisk kondensator type X7R. Tantalkondensatoren udmærker sig ved gode lavfrekvente egenskaber og har en lav ESR. X7R typen af keramisk kondensator har gode højfrekvente egenskaber, så tilsammen udgør disse to kondensatortyper en effektiv afkobling. Dog er det vigtigt at huske at placere afkoblingen tæt på forsyningsbenen. Printet er forsøgt designet i sektioner, så den digitale sektion findes i en side af printet, mens den analoge sektion findes i den anden side af printet. Clockgeneratoren er placeret midt imellem disse to sektioner, da begge sektioner benytter denne og det er vigtigt at holde clockbanerne så korte som muligt. Endeligt er der en sektion til spændingsforsyningen.

Kapitel 6

Test

I dette kapitel beskrives de test som er udført på den udviklede prototype, samt de problemer der er blevet konstateret i forbindelse med disse tests. Yderligere er der forslået ændringer til at udbedre de konstaterede fejl, hvor dette er muligt.

6.1 Test af den digital del af systemet

Den digitale del af systemet er forholdsvis enkel at teste, og består primært af en række "hul igennem tests".

6.1.1 Test af JTAG og generel processor funktionalitet

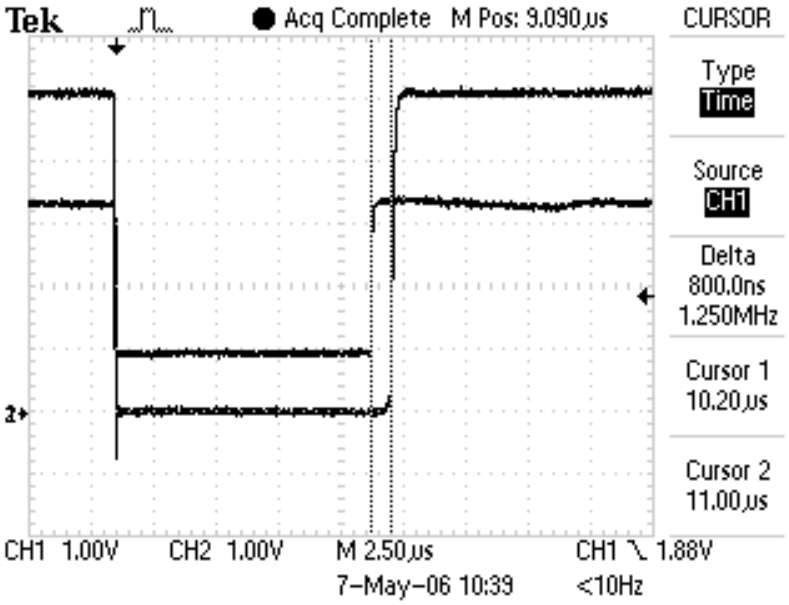
Den første og mest afgørende test af prototypen, var om det overhovedet var muligt at komme i kontakt med processoren via JTAG interfacet. Men før dette blev forsøgt, blev printet målt igennem for at konstatere om de forskellige forsyningsspændinger var iorden (5, -5 og 3.3 volt). Dette blev fundet iorden, og kontakt blev forsøgt via JTAG adapteren og AVRStudio IDE'et. Kommunikation med processoren fungerede uden problemer og den eksterne cloçkgenerator fungerede også efter hensigten. Desuden blev reset kredsløbet testet og fundet iorden. Desværre var det ikke muligt at testet ISP programmeringsinterfacet, da en ISP adapter ikke var til rådighed.

6.1.2 Test Real Time Clock

Real Time Clocken blev helt enkelt testet, ved at prøve om det var muligt at kommunikere med den via SPI bussen. Til dette formål blev der skrevet et lille stykke testkode, som skulle afklare dette. Kredsløbet fungerede efter hensigten, både hvad angår selve ur funktionaliteten, men også alarm funktionen via interrupts. Endeligt blev det konstateret at RTC'en holdt tiden selv om hovedforsyningen blev afbrudt, et tegn på at batteribackup funktionen virker. Slutteligt blev der fortaget en "speedtest", for at se om kommunikationen med RTC'en fungerer ved fuld hastighed på SPI bussen (8 Mbps), hvilket den gjorde.

6.1.3 Test af SD/MMC kort

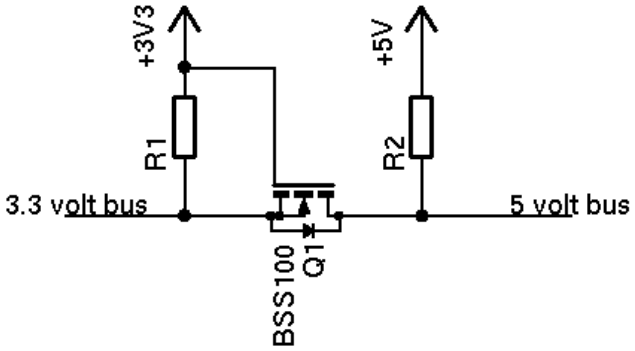
Før SD/MMC kortet blev forbundet blev det konstateret om levelshifterne fra processoren til SD/MMC kortet fungerede korrekt. Dette var for at sikre at kortet ikke fik en overspænding og blev ødelagt. Hernæst blev det testet om det var muligt at kommunikere med kortet via SPI. På den første prototype, bygget på fumleprint, blev der ikke konstateret nogle problemer med SPI kommunikationen. Da anden prototype var færdig, blev det dog konstateret at SPI kommunikationen fejlede, når hastigheden blev for høj. Noget som ikke var testet på den første prototype. Efter først at have udelukket en software fejl, blev SPI kommunikationen set efter med et oscilloscop. Første mistanke var at signalet blev ødelagt af ringning eller lignende. Dette kunne dog ikke konstateres. Istedet kunne det konstateres at der var et delay i transistor trinnet (dataout fra SD/MMC kortet), se figur 6.1.



Figur 6.1: Screenshot af transistor levelshift.

Figuren viser ind- og ud-gangsspændingen fra transistor trinnet. Som det ses er der ikke nogle problemer når der skiftes fra høj til lav, dette er et tegn på at trinnet drives "hårdt nok". Signalet kommer også fint igennem trinnet, uden nævneværdig ringning eller forvrængning. Problemet opstår når trinnet skal "slippe" igen, her er der et delay på 800nS fra indgangen på trinnet går høj, til det samme sker på udgangen. Dette propergationdelay, gør at dataout fra SD/MMC kortet kommer ud af sync med SPI bussens clocksignal. Da SPI bussen sampler data på enten rising eller falling edge af sit clock signal, betyder dette at når hastigheden bliver for stor nærmer clockens periodetid sig dette propagationdelay og data kan ikke længere aflæses korrekt.
Forklaringen på denne opførelse skal findes i opbygningen af en bipolar transistor. Denne består jo som bekendt af PN overgange, i dette tilfælde helt præcist

af NP og PN overgange (da de anvendte transistorer er NPN typer). Disse overgange har en lille uønsket parasitkapacitet, når transistoren styres "on" vha. en basisstrøm oplades basis-emitter "kondensatoren". Dette kan gøres hurtigt hvis basisstrømmen er tilstrækkeligt stor, hvilket den oftes er, da der ønskes hurtige skiftetider. Desværre bevirker denne parasitkapacitet, samt det faktum at transistoren drives hårdt, at transistorens holdetid forøges, netop det fænomen der observeres her. Holdetiden opstår fordi den omtalte parasitkapacitet skal aflades før transistoren lukker igen. Holdetiden kan nedsættes ved at parallelkoble en kondensator med basismodstanden, en såkaldt "speedup kondensator". Meningen med denne kondensator er at den skal optage den ladning der er opbygget i PN overgangen mellem basis og emitter når transistoren lukker, samtidig giver opladningen af kondensatoren et tilskud til basisstrømmen når transistoren åbner. Denne løsning er dog ikke blevet afprøvet, da der under research af problemet er blevet fundet en mere elegant løsning, der kan ses på figur 6.2.



Figur 6.2: MOSFET levelshifter.

Denne levelshifter har mange fordele, herunder:

1. Den er simpel, og bruger få komponenter.
2. Den er bidirektionel (det er dog ikke nødvendigt i denne sammenhæng).
3. Den er hurtig.
4. Kan benyttes til at adskille de forskellige dele af bussen.
5. Beskytter lavvolt delen af bussen mod transitienter fra højvolt delen af busse.

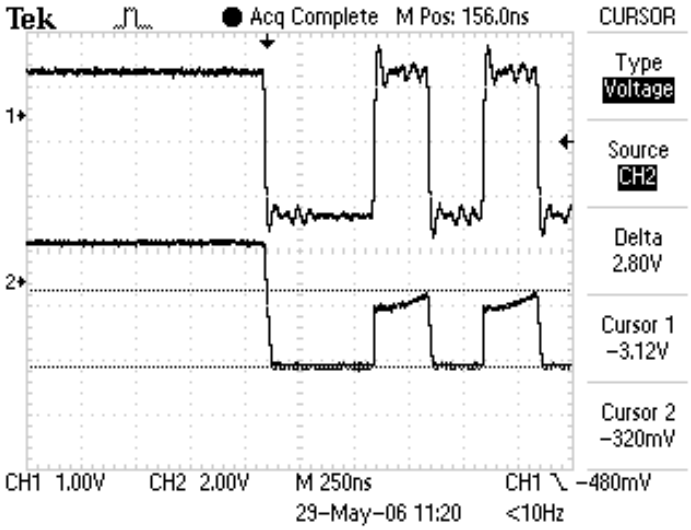
For at forstå virkemåden er det nyttigt at betragte de følgende tre tilstande:

1. **Ingen enhed trækker bussen lav:** Pullup modstanden R1 trækker, derved er både gate og source på MOSFET'en på 3.3 volt, som derved er lukket, da $V_{gs} = 0$ (Threshold på den anvende MOSFET er ca. 2 volt). Dette betyder at 5 volt delen af bussen bliver trukket høj af pullup modstanden R2.

2. **En 3.3 volt enhed trækker bussen lav:** Source på MOSFET'en er lav, og gate'en er på 3.3 volt, dette er over thresholdspændingen, og MOSFET'en åbner. Herved trækkes 5 volt bussen lav igennem MOSFET'en, vel og mærket ved samme niveau som på 3.3 volt bussen.

3. **En 5 volt enhed trækker bussen lav:** Da source på MOSFET'en i dette tilfælde er på 3.3 volt, og drain er tæt på 0 volt, vil den indbyggede diode begynde at lede. Dette vil trække sourcen ned, indtil thresholdspændingen nåes og MOSFET'en begynder at lede. Herved trækkes 3.3 volt bussen lav med samme niveau som 5 volt bussen.

Den ovenfor omtalte metode, er blevet afprøvet i praksis, ved at bygge et lille print med levelshifteren på. Den eksisterende transistor levelshifter blev sat ud af spillet ved at skære printbanerne til den over. Herefter kunne den nye leveshifter installeres.



Figur 6.3: Måling på MOSFET levelshifter.

Figur 6.3 viser en måling foretaget på MOSFET levelshifteren. Den øverste kurve er 3.3 volt bussen, mens den nederste er 5 volt bussen. Som det ses er denne levelshifter væsentligt hurtigere en transistorudgaven. Dog når den ikke at komme helt op på de krævede 3 volt som er minimum for ATMega128 processorens høj niveau, den kommer kun op på 2.8 volt. Dette skyldes dog at de brugte pullup modstande er for store på prototypen, ved at sænke disse til omkring $4.7k\Omega$ kan dette rettes. Alt i alt betyder denne ændring at SPI bussen kan benyttes ved fuld hastighed (8 Mbps), hvor den maksimalt kunne benyttes ved 500 kbps med transistor trinnet.

6.1.4 Test af I/O interfaces

Testen af RS232 og I2C interfacene blev ligesom andre test foretaget som en "hul igennem test". RS232 interfaces blev testet med et stykke kode som var

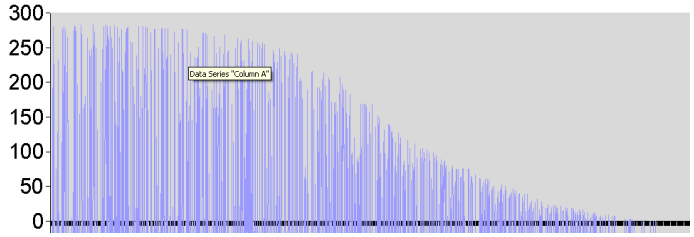
istand til at kommunikere med et terminal program på en PC, dette fungerede uden problemer. Spændingsniveauerne blev også målt med et scop, for at sikre at disse var i overensstemmelse med standarden.
I2C interfaces var som før omtalt blot tænkt som en udvidelsesport til senere brug sammen med et UI board. Dette board blev imidlertid bygget, og det kunne derfor konstateres at det fungerede uden problemer.

6.2 Test af den analoge del af systemet

Test af det analoge delsystem, består af to dele: Hul igennem test fra processoren til FPAA'en og en række tests af selve de analoge kredsløb.

6.2.1 Test af FPAA'ens digitale interface

Formålet med denne test er at bestemme om det er muligt at kommunikere med FPAA'en via SPI bussen, og derved indlæse konfigurationer i denne. Denne test blev udført ved at sende en konfiguration til FPAA'en, også observere om denne aktiverede sit ACTIVATE ben som tegn på at konfigurationen var forstået og aktiveret. Samtidig kunne der konstateres at VMR blev sat til 2 volt, denne referencespænding er nemlig kun tilstede når en konfiguration er aktiveret. Der blev dog fundet et enkelt problem, idet det viste sig at der var byttet om på to ben på det footprint af FPAA'en der var brugt til at tegne diagrammet. Disse ben var derfor også byttet om på printet. Dette blev dog klaret med en skarp kniv og et par stykker tynd ledning.



Figur 6.4: Test af 20 kHz lavpas filter implementeret i FPAA'en.

For at teste FPAA'ens funktionalitet, samt indgangskredsløbene og A/D konverteren, blev der konstrueret en konfiguration til FPAA'en indeholdene et lavpas filter, med en afskæringsfrekvens på 20 kHz. Dernæst blev en funktionsgenerator tilsluttet systemet, og et sweep fra 10 til 100 kHz blev gennemført, mens systemet loggede. Resultatet fra denne logning blev importeret i OpenOffice og en graf blev plottet. Resultatet kan ses i figur 6.4

6.2.2 Måling af støj

For at få en ide om hvor godt designet af printet er, og hvor effektiv afkoblingen i systemet fungerer, er der blevet udført et antal målinger af støjen i systemet.

Først og fremmest er støjen på de forskellige forsyningsspændinger blevet målt. Resultatet heraf kan ses i tabel 6.1.

Spænding	Støj
5 volt	3.3 mV RMS
3.3 volt	0.3 mV RMS
-5 volt	41 mV RMS
VMR (2 volt)	0.25 mV RMS

Tabel 6.1: Støj målt på de forskellige forsyningsspændinger.

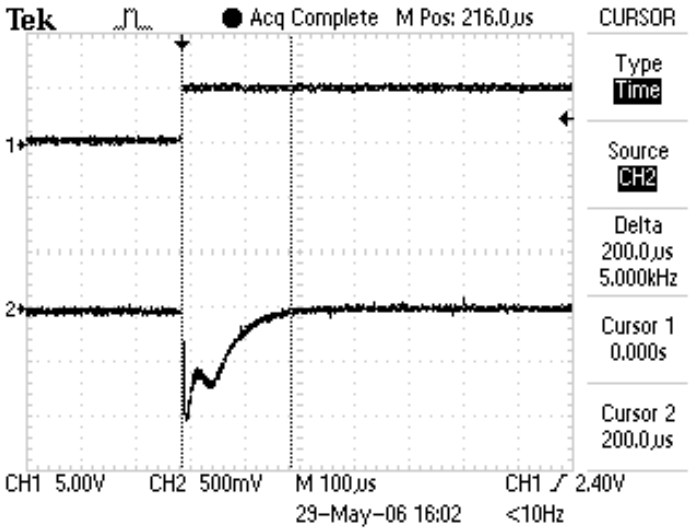
5 volt forsyningen er den primære forsyning forstået på den måde at den kommer direkte fra 220 volt adapteren, og den danner grundlag for alle andre forsyninger i systemet. Støjen på denne forsyningslinie ligger på omkring 3.3 mV RMS, dog er der rimelig stor variation, som følge af skrivningerne til SD/MMC kortet. Dette kunne tyde på at levelshifter logikken kunne trænge til at blive afkoblet bedre end den nuværende prototype er afkoblet. Som det ses i tabellen for 3.3 volt forsyningen er støjen på denne en del lavere en på 5 volt forsyningen. Dette skyldes at 3.3 volts regulatoren undertrykker en del af støjen. Den klart værste forsyning er -5 volt forsyningen, dette er dog som forventet, da denne kommer fra en chargepump. Støjen fra denne afhænger bla. af oscillatorfrekvensen og ESR i de anvendte kondensatorer. Dette er dog ikke det store problem da -5 volt forsyningen kun bliver brugt til at forsyne indgangstrinnene, som består af AD8132 operationsforstærkere, og disse har en power supply rejection ratio på -70dB.
Det mest interessante måling af støj er dog ikke på forsyningen, men derimod på selve udgangen fra FPAA'en, altså det signal som samples af A/D konverteren. Denne måling er dog lidt besværlig da der er tale om differentielle signaler. Først blev et analog oscilliscop brugt til at se om støjen havde samme udseende på de to differentielle ledninger. Da det var konstateret af dette var tilfældet, blev støjen målt med et fintfølende RMS voltmeter.
Da der ikke korelation mellem støjen på de to ledninger, kan disse ikke blot lægges sammen for at finde den samlede støj. Istedet regnes støjen som det samlede effektbidrag: $V_{noise} = \sqrt{V_-^2 + V_+^2}$. Med de målte værdier bliver dette: $V_{noise} = \sqrt{0.00375^2 + 0.00375^2} \approx 5.3mV$. Sammenholdes dette tal med med værdien af en LSB for A/D konverteren, som ved et udstyringsområde på 2.56 volt [1], er på 2.5 mV, svarer støjen altså til 2 LSB. Der er med andre ord plads til forbederinger, særligt hvis der senere skal benyttes en A/D konverter med en højere opløsning.

6.2.3 Test af settletime

Den sidste test af det analoge delsystem, er at måle settletime for FPAA'en, efter denne er blevet konfigureret . Denne måling er vigtig for at kunne bestemme hvornår signalet er stabilt nok til at det kan samples. Hvis der ikke ventes på

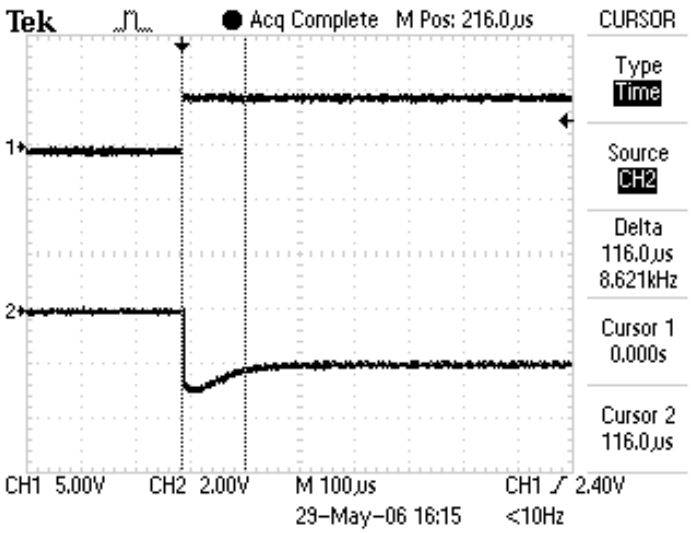
[1] Standard intern referencespænding for ATMega128

at signalt bliver stabilt efter en omkonfigurering af FPAA'en, vil den samplede værdi være et resultat en et mere eller mindre tilfældigt transientforløb. Figur 6.5 viser transitientforløbet efter et skift til et kredsløb indeholdende et lavpasfilter. Den øverste kurve er ACTIVATE signalet som indikere at FPAA'en er konfigureret og klar ved at gå høj. Den nederste kurve er udgangen på FPAA'en, som det ses er der et vist transient forløb efter ACTIVATE går høj. Dette forløb dør ud efter ca. 200μs, og det er herefter sikkert at sample signalet.



Figur 6.5: Transitientforløb efter skift til et lavpasfilter implementeret i FPAA'en.

Figur 6.6 viser samme situation, men istedet for et lavpas filter implementerer FPAA'en nu en unity-gain forstærker. Som det ses bliver det transiente forløb noget anderledes, men det har dog cirka samme længde. Settletime afhænger med andre ord, tildels af det implementerede kredsløb, og det kan derfor være nødvendigt at måle denne i visse situationer, for at bestemme hvor lang tid der skal ventes for signalet samples.



Figur 6.6: Transitientforløb efter skift til en unity-gain forstærker implementeret i FPAA'en.

Kapitel 7

Delkonklusion

Der er blevet udviklet en prototype på systemets hardware. Denne er baseret på en FPAA, hvilket giver en uovertruffen fleksibilitet, når det kommer til interfacing af forskellige sensorer, og giver mulighed for rapid prototyping. Data opsamlet fra sensorene gemmes på et SD/MMC kort, hvilket giver mulighed for hurtigt og nemt at flytte dataene til en PC til videre forarbejdning. Endvidere indeholder systemet en Real Time Clock med batteri backup, så det er muligt at tidstemple data. Endeligt har systemet en række interfaces der gør det muligt at styre det via et terminalprogram på en PC. Men der er også mulighed for at benytte systemet med et tilhørende UI board med LC-display og knapper.
Systemet er dog stadig en prototype, og der er plads til forbedringer. Derfor anbefales det at lægge vægt på følgende i den næste prototype:

- Forbedre støjforhold, ved at forbedre printlayout og afkobling.
- Udskifte hele levelshifting logikken med den omtalte MOSFET type.
- Lave et rigtigt print til UI boarded.
- Evt. benytte en ekstern A/D konverter for forbedret opløsning.

Del III

Software

Kapitel 8

Indledning

Denne del af rapporten beskriver design og implementation af systemets software

Skrevet af Andreas Rune Fugl

I denne del af rapporten beskrives den udviklede software til dataopsamlingssystemet.
Delrapporten er organiseret så der først beskrives det overordnede arkitekturvalg og dernæst design og implementationer af delsystemer samt overvejelser omkring forskellige designparametre.

Da mange af overvejelserne der er gjort for softwaren har været afhængig af hardwareplatformen, anbefales det at have læst del II der beskriver hardwaren i systemet.

Kapitel 9

Overordnet arkitektur

9.1 Indledning

Dette afsnit præsenterer de overvejelser der er blevet gjort ved design af den overordnede arkitektur for softwaren kørende på systemets mikrocontroller. To forskellige tilgangsvinkler beskrives og afvejes i forhold til det ønskede system.

9.2 Design

Softwaren i systemet eksekverers på en mikrocontroller og har overordnet set til ansvar at styre al funktionalitet, heriblandt analog-digital konvertering af data samt lagring på non-volatile medie.
Funktionaliten i systemet består overordnet af

1. A/D-konvertering
2. Sensorstyring igennem en FPAA
3. Tidsstyring
4. Lagring af data på medie
5. Brugerflade (RS232, LCD, input mm.)

A/D-konverteringen står for samplingen af det analoge signal i systemet.
Sensorstyringen omfatter styring af den FPAA der anvendes til sensortilpasning.
Tidsstyringen omfatter opdatering af tiden fra en RTC, samt styring af de tidspunkter hvorpå der skal udføres logninger.
Lagringen af måledata foregår på et SD-kort med et FAT-filsystem.
Brugerfladen lader brugere tilgå systemet enten via RS232 systemkonsollen eller det tilkoblede interface med LCD og knapper.

Timingerne i systemet kan generelt set deles op i følgende

- Strict (must complete on)
  - Opdatering af tid

- Deadline (must complete within)
  - RS232, LCD, input.
- Flexible
  - A/D-konvertering
  - Sensorstyring
  - Lagring af data på medie

Den centrale proces i systemet er datalogningen, der overordnet set er en sekventiel proces bestående af sensorkonfiguration, A/D konvertering og lagring. Sensorkonfigurationen må nødvendigvis udføres før A/D konverteringen påbegyndes. Til lagringen er det da blot et krav, at ingen målinger går tabt. Lagringen kunne f.eks. fungere igennem en skrivecache hvor der skrives til mediet når der er tid.
Selvom datalogningsprocessens elementer kan betegnes som havende fleksible timingskrav, er de tilsammen væsentlige for systemets logningshastighed. Valget af den overordnede systemarkitektur skal derfor facilitere disse krav.
I forbindelse med valg af den overordnede softwarearkitektur er der blevet gjort overvejelser mht. fordele og ulemper som det fremgår af tabel 9.1.

	RTOS		**Traditionel**
+	Driverbase	+	Større throughput
+	Processsamarbejde	+	Mindre ramforbrug
+	Mulighed for udvidelse	-	Skalering
+	Gennemsigtighed	-	Tidsforbrug ved større operationer
-	Ramforbrug	(+)	Mindre strømforbrug
+	Prioritering af processer		
+	Fejltolerance		
-	Øget kompleksitet		

Tabel 9.1: Sammenligning af overordnede arkitekturer.

Da den traditionelle fremgangsmåde antages kendt, vil følgende afsnit primært omhandle vurderingen af realtids operativsystemer.

De to realtids opereativsystemer, FreeRTOS og AvrX blev brugt i vurderingen da de kunne anvendes på systemets mikrocontroller.
Begge er af time-sharing typen og tilbyder preemptiv og frivillig schedulering, message queues samt semaphorer. Trådning foregår for begge igennem "tasks" eller opgaver med definerede prioriteter. Disse er specielt definerede funktioner indeholdende den kode der måtte ønskes kørt.
En mulig brug af disse operativsystemet i en preemptiv konfiguration kan ses af tabel 9.2. Funktionaliteten i systemet er fordelt på forskellige opgaver, alt efter om de kan adskilles eller om de har forskellige timingskrav:
Pga. prioriteringen der udføres af operativsystemet, vil en opgave med lavere prioritet altid træde i baggrunden til fordel for en opgave med højere prioritet.

Prioritet	Opgaver		
3	Opdatering af tid		
2	RS232	LED	Input
1	ADC	FPAA	Storage

Tabel 9.2: Opdeling i opgaver og prioriteter til RTOS.

Ved at vælge en egnet scheduleringsfrekvens vil det være muligt at imødekomme mange forskellige timingskrav. Lagringsopgaven kunne, jævnfør ovenstående ide, køre med en lav prioritet og lagring dermed udføres når der er tid fra de andre opgaver. Et umiddelbart problem er dog, hvis at der i en periode ikke er nok processortid til lagringsopgaven så at skrivebufferne til sidst fyldes op og data mistes.

Som det fremgik af tabel 9.1 betragtes øget kompleksitet og ramforbrug som væsentlige ulemper ved brug af et RTOS.
Ved indførelsen af de ekstra lag som operativsystemet tilbyder for at forbedre processamarbejdet, øges kompleksiteten på flere områder. Blandt andet vil senere brugere af systemet i en vis grad skulle sætte sig ind i operativsystemets virkemåde, udover den centrale del af koden som udgør funktionaliteten.
Grundet den større mængde kontrolstrukturer der bruges i operativsystemet er RAM-forbruget også forøget. Der kræves desuden en grundig evaluering af operativsystemet, for at sikre sig at timingskrav er opfyldt.

Vurderet i forhold til den centrale opgave i systemet har muligheden for processamarbejde mindre relevans, da datalogningsprocessen som tidligere nævnt er sekventiel af natur. Det vurderes derfor at brugen af et realtids operativsystem ikke vil bidrage med væsentlige fordele til systemet. Taget den øgede kompleksitet og ekstra tidsforbrug ved implementationen i betragtning, er det derfor blevet valgt at anvende den traditionelle løkke-metode til systemet og fokusere på funktionaliteten af systemets delkomponenter.

9.2.1 Implementation

Fordelingen af systemets funktionalitet i kildekode, kan ses af tabel 9.3.

Fil	Note	Indhold
a2d.c	(1)	Driver til A/D konverteren på ATMega128.
buffer.c	(1)	Generel byte-buffer funktionalitet, anvendt af bla. uart.
debug.c	(1)	Hjælperutiner til debug.
dosfs.c	(2)	DOSFS FAT16/32 filsystemsdriver.
ds1305.c		Driver til Dallas DS1305 Real-Time-Clock.
fpaa.c		Driver til Anadigm AN221E04 og generelle hjælperutiner.
i2c.c	(1)	Two-wire I2C driver.
lcd.c	(1)	HD44780 LCD driver, kun delvist i brug.
main.c		Programmets hovedløkke indeholdende brugerflade og logningsløkker.
mmc.c	(1)	MMC/SD-kort driver.
rprintf.c	(1)	printf rutiner.
spi.c	(1)	SPI driver.
storage.c		Hjælperutiner til skrivning og læsning fra FAT filsystemet.
timer.c	(1)	Funktioner til brug af timere på ATMega128.
timer128.c	(1)	Samme som ovenstående.
uart.c	(1)	Driver til seriel UART på ATMega128.
uart2.c	(1)	Samme som ovenstående.
uiboard.c		Eksternt brugergrænsefladeboard, med bla. LCD output og brugerinput over I2C.
vt100.c	(1)	Funktioner til VT100 emulation på den serielle terminal.
rtc.c	(3)	Rutiner til datostyring på mikrocontrolleren.
sensor.c		Hjælperutiner til til skrivning af logheadere og indlæsning af sensorkonfigurationer.

Tabel 9.3: Fordeling af funktionalitet.

(1) - Procyon AVRlib [Stang, 2006].
(2) - DOSFS FAT12/16/32 Filsystem [Edwards, 2006].
(3) - RTC-kode fra Atmel Application Note AVR064 [Atmel, 2006c].

Udover de tilhørende headere, findes der følgende specielle headere:

Fil	Note	Indhold
logger_hw_defs.h		Indstillinger specifikke for hardwareplatformen.
logger_settings.h		Generelle indstillinger for systemet.
logger_strings.h		Strengkonstanter brugt forskellige steder i systemet.

Tabel 9.4: Specielle headerfiler.

Kapitel 10

Lagring

10.1 Indledning

Lagringsmediet i systemet har til ansvar at gemme opsamlede data på en sikker måde, så at de bevares selv om systemet mister strømforsyning.
Der opstilles følgende krav til lagringsfunktionaliteten i systemet:

1. Lagringsmediet skal have fuld understøttelse på en PC.
2. Driver til lagringsmediet skal kunne afvikles på systemets mikrocontroller.
3. Filsystemet skal have læse- og skriveunderstøttelse på mikrocontrolleren.
4. Logningsformatet skal kunne genkendes af gængse analyseværktøjer, uden konvertering
5. Filsystemets driver skal kunne afvikles på systemets mikrocontroller.

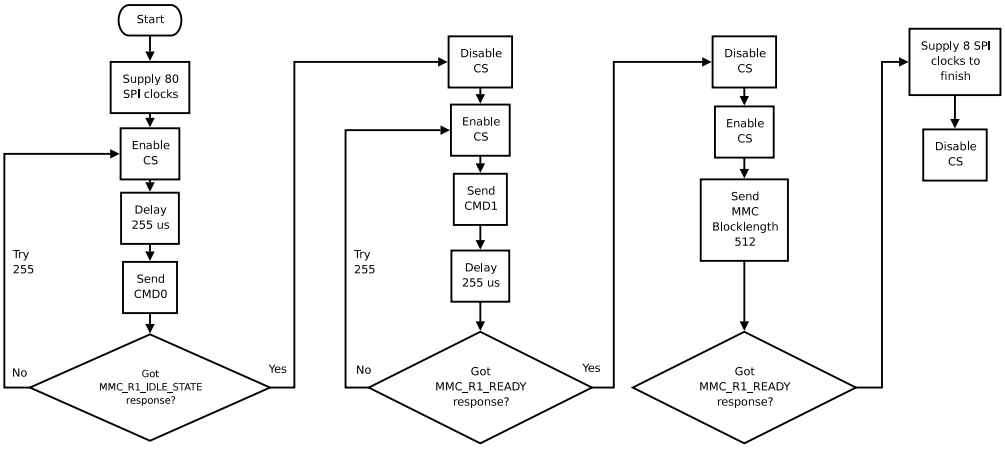
10.2 Medie

Hardwareplatformen har indbygget et MMC/SD interface der kan anvende MMC- og Secure Digital kort.
MMC- og især SD-kort bruges i stor grad til digitalkameraer, med en markedsandel på over 40% i 2005 [Wikipedia, 2006]. Populariteten af digitalkameraer har medført en stadig faldende pris på både medier og kortlæsere. Nyere versioner af styresystemer såsom Linux og Microsoft Windows har understøttelse for langt de fleste SD/MMC læsere og opfylder dermed krav 1.
SD-kort kan fås i mange forskellige størrelser fra 32MB op til 4GB. Dermed er der rig mulighed for at kunne vælge et kort, der opfylder en ønsket lagringskapacitet.

SD- og MMC-kort kan tilgås igennem et simpelt 1 bits serielt SPI interface, hvilket i høj grad egner sig til indlejrede systemer. På systemets hardwareplatform er interfacet tilkoblet mikrocontrollerens SPI-bus.
Til læsning og skrivning er der taget udgangspunkt i MMC/SD-rutiner fra Procyon AVRlib [Stang, 2006]. Disse vil blive skrevet i implementationsafsnittet.

10.2.1 Implementation

Før at at der kan læses og skrives til et MMC/SD kort er det nødvendigt at det først bliver initialiseret korrekt.
Funktionen fra Procyon AVRlib til at håndtere dette, `mmcReset` implementerede ikke forløbet korrekt og som følge deraf var initialiseringen upålidelig. Initialiseringensrutinen blev derfor skrevet om til at følge Sandisk's anbefaling (Se [Sandisk, 2006]) for initialisering af SD-kort, samt at være væsentlig mere konservativ mht. waitstates og gentagelser.
Initialisering af SD/MMC kort er på systemet gjort som vist på figur 10.1



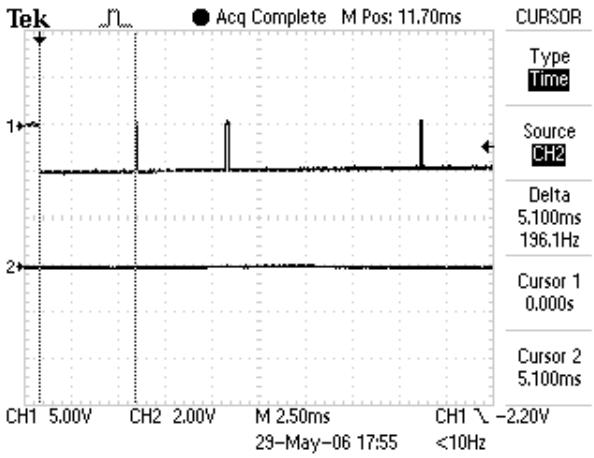
Figur 10.1: Systemets initialisering af MMC/SD kort.

Når der ses på om det rigtige svar kommer tilbage fra kortet, sendes der adskillige dummy bytes, for at give kortet tid til at behandle kommandoen.

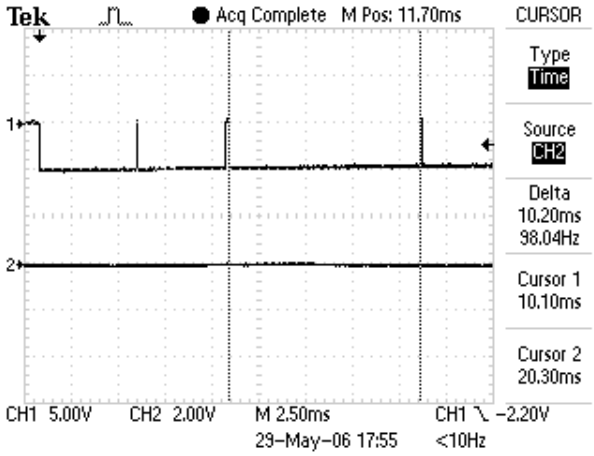
Læsning og skrivning til interfacet foregår igennem rutinerne `mmcRead` og `mmcWrite`, der henholdsvis læser og skriver en 512-bytes buffer. I forhold til de originale rutiner er disse to også blevet ændret for at forbedre pålidelighed og performance.

10.2.2 Test

Figur 10.2 viser en måling på skrivetiden af et SD-kort, udført ved at måle på chipselect af kredsen ved skrivning af 512-bytes med `mmcWrite`.
Som det kan ses af figuren tager skrivningen 5,1 ms og den efterfølgende er også på omkring 5 ms. Dette er dog ikke altid konstant som det kan ses af figur 10.3. I denne situation er det nødvendigt for interfacet at vente på at SD-kortet bliver færdig med at udføre sin skriveoperation og som resultat er ventetiden oppe på 10.2 ms.



Figur 10.2: Tid for skrivning til MMC/SD kort.



Figur 10.3: Tid for skrivning til MMC/SD kort når kortet ikke er klar med det samme.

10.3 Filsystem

File Allocation Table (FAT) i dens varianter er et af de mest udbredte filsystemer og stammer i formaliseret form tilbage fra 1977 hvor det blev anvendt af Microsoft Disk BASIC [Wikipedia, 2006].
Selvom det ikke mere anvendes som det primære valg til operativsystemers lagring på PC-siden, er det på grund af sit relativt ukomplicerede design blevet en de facto standard for mange transportable lagringsmedier.
Iblandt mange hobbyfolk på Internettet, har det været populært at lave egne embedded konstruktioner af f.eks. MP3-afspillere med anvendelse af bærbare medier kørende FAT-filsystemer. Som resultat deraf findes der adskillige open-source embedded implementationer af filsystemet, men næsten uden undtagelse er disse kun med læseunderstøttelse.

Efter længerevarende søgning blev der fundet frem til open-source implementationen DOSFS [Edwards, 2006], der er et FAT-kompatibelt filsystem. DOSFS har følgende interessante features[1]:

- Læse- og skriveunderstøttelse
- Understøtter FAT12, FAT16 og FAT32
- Understøtter underbibloteker
- Kan køre med en enkelt 512-byte sector buffer
- Uafhængig af memory-management
- Delvis understøttelse for søgning i filer

Set i forhold til de opstillede krav, er denne implementation særdeles interessant til brug i systemet. For at anvende filsystemet skal der blot angives funktioner der kan læse og skrive fra og til en buffer og videre til det fysiske medie.
Implementationen af dette, samt de nødvendige modifikationer til koden vil blive beskrivet i implementationsafsnittet.

10.3.1 Implementation

For at anvende filsystemet på et mediet, er to rutiner `DFS_ReadSector` og `DFS_WriteSector` defineret for læsning og skrivning. Disse er for systemet ganske simple, da MMC-rutinerne anvender samme sektorstørrelse på 512 bytes, som kræves af DOSFS's bufferhåndtering.
For at opnå en større abstraktion til håndtering af logfiler i systemet, er der blevet implementeret følgende ekstra funktioner:

- `mountFatVolume` - Sørger for at mounte FAT-filsystemet så efterfølgende funktioner kan anvendes.
- `printFile` - Udskriver indholdet af en fil til en terminal eller LCD.
- `appendToLog` - Skriver en buffer til slutningen af en åben fil.
- `writeToLog` - Skriver en buffer til en fil.
- `deleteFile` - Sletter en fil.

Eftersom at det er forskelligt, hvordan MMC/SD-kort er er formateret, er det nødvendigt i `mountFatVolume` at probe om der findes en partitionstabel (MBR) eller ej som indeholder en partiton med et FAT16/32 filsystem. Hvis der ikke findes en partitionstabel, antages FAT-filsystemet at starte fra fysisk sektor 0 (mange kort er desværre formateret på denne relativt usikre måde).
Sidstenævnte metode er ikke speciel god eftersom at kortet kan være formateret med et andet filsystem end FAT. Det anbefales derfor at anvende kort som er

[1] Delvist gengivet og oversat fra [Edwards, 2006]

formateret med en partitionstabel.
Funktionen anvender som standard den første primære partition og kan mounte følgende typer

- Win95 FAT32
- Win95 FAT32-LBA
- FAT16 < 32MB
- FAT16
- Win95 FAT16-LBA

Det skal bemærkes at der ikke er understøttelse for lange filnavne og der skal derfor anvendes 8.3 filnavne[2].

DOSFS anvender ikke nogen form for caching af FAT eller filer af hensyn til RAM-forbrug og derfor er især skrivehastigheden lav, da der her skal udføres ændringer på FAT'en. Programmøren af DOSFS kommer med vejledning til, hvordan en eventuel caching kan udføres, men det blev vurderet at komplet implementation af dette ville kræve mange ressourcer. Der er istedet blevet lavet caching på et højere niveau, der er specielt tilpasset lagring af tekstdata i logfiler.
Cachen består af et array i RAM der indeholder et antal blokke, hver indeholdende en mængde tekstdata som er udestående for skrivning til det permanente lager. Hver blok indeholder ud over tekstdata, information om hvilken fil den har data til, samt et flag der indikerer om blokken er i brug.
Når et logningsforløb måtte ønske at lagre data, kaldes funktionen `appendToLogCache` istedet for den tidligere beskrevne `appendToLog`. Funktionen lagrer det ønskede tekstdata på den første frie blok i arrayet. Den brugte blok bliver så fyldt med tekstdata og den markeres som værende i brug af den aktuelle sensor.
Når arrayet med blokke er fyldt op, køres funktionen `flushCache`. Denne løber arrayet igennem, og for hver enkelt fil opsamler blokkene af tekstdata i samme rækkefølge som de blev skrevet. Disse samles til en stor tekststreng, som så kan skrives til til MMC-kortet med én enkelt filoperation. Derved undgås overheadet af flere funktionsgennemløb, samt at FAT ikke skal opdateres så mange gange. Tømning af cache kan også foretages manuelt af logningsforløbet ved selv at kalde `flushCache`. Dette foretages blandt andet når et logningsforløb terminerer.
Før at cache-funktionaliteten anvendes, skal den logfil og sensor man ønsker at logge til registeres. Dette gøres med `registerLogCache`, der sørger for at åbne filen og tildele denne et id, så der senere hurtigt kan refereres til filen. Den registrerede fil forbliver åben efter denne registrering. Dette betyder at skrivehastigheden forbedres betragteligt eftersom kald til åbning af filen undgås samt at det ikke er nødvendigt at søge til enden af filen når der skal skrives.

[2]Maks 8 tegn + 3 til filendelse

10.4 Logningsformat

I forbindelse med valg af logningsformat til systemet er der blevet set på to muligheder, kommaseparerede tekstfiler (CSV) og binære filer.
CSV-formatet[3] udemærker sig ved at have god understøttelse i en lang række analyseværktøjer, være platformsuafhængig samt let at implementere med brug af standard strengformateringsrutiner. En ulempe er at data først skal formateres til strenge i ASCII-format, før at de kan skrives til logningsfilen. Dette afhjælpes dog af at der i det anvendte C-libary findes en række af standardiserede strengformateringsfunktioner.
Alternativt kunne der have været anvendt et binært format. Fordelen ved dette, er at implementationen i systemet bliver simplere da data direkte kan skrives til filen. Denne fil vil dog ikke umiddelbart være genkendelig af gængse analyseværktøjer.
Ud fra ønsket om at lade importen til analyseværktøjet foregå uden brug af konvertering, vælges det derfor at lade logningsformat være et CSV-format lagret i flade tekstfiler.

Med hensyn til syntaksen i CSV-formatet findes der ikke en standard. Formatet har dog været i brug meget længe og en anerkendt beskrivelse findes i [Group, 2006].

For at kunne koble tidskonfigurationer og de tilhørende logningsfiler sammen, anvendes de første linier af logfilen til at skrive forskellige informationer. Primært er information omkring det brugte logningsinterval gemt, så at analyseværktøjet kan anvende den rigtige skalering på tidsaksen. Andre nyttige informationer er starttidspunkt og en beskrivende tekst for logfilen.
Udover datasamplingens resultat, skrives der for hver type datalogning forskellige data til en record. For langtidslogningen (beskrevet i afsnit 11.2) er der inkluderet et ekstra felt udover måleresultatet, der indeholder tidsoffsettet.

10.4.1 Implementation

Da der i det benyttede C-library [Non-gnu, 2006] findes standard strengformateringsrutiner, er anvendelsen af formatet ganske enkelt og derfor er der ikke lavet separat funktionalitet til at skrive CSV-formatets records.
Skrivning foregår ved at allokere en tekstbuffer med nok plads til den højest muligt recordlængde og anvende `snprintf` funktionen, f.eks. som vist

```
1 snprintf(textdata,sizeof(textdata),"%u,%u\r\n",adSample, timerTick)
```

hvilket resulterer i en logrecord som f.eks.

```
1 267,200
```

`snprintf` anvendes fremfor `sprintf`, da sidstnævnte ikke er sikret imod at skrive udover størrelsen af textbufferen.

[3] Comma-Separated Values

Kapitel 11

Tidsstyring

11.1 Indledning

Formålet med tidsstyringen at afgøre, hvornår systemet skal udføre en datalogning. En datalogning indebærer mange trin, bla. konfiguration af FPAA'en, opsamling fra A/D konverteren samt efterfølgende lagring til det permanente lager.
Følgende afsnit beskriver systemets logningsmetoder.

11.2 Fleksibel langtidslogning

Langtidslogningen har til formål at give den største fleksibilitet for brugeren i valg af logningsintervaller og konfigurationer for den analoge FPAA. Denne tilstand er rettet imod meget langsomt varierende signaler, der ønskes overvåget over flere dage.
Der er opstillet følgende krav til langtidslogningen i systemet:

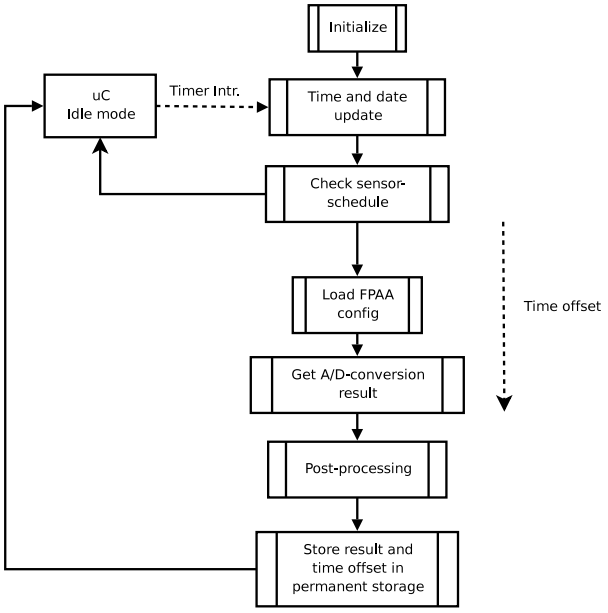
1. Logning for en vilkårlig sensor skal kunne udføres på alle ønskelige tidspunkter med en opløsning på 1 sekund.

2. Sensorerne skal kunne konfigureres med hver sin FPAA konfiguration.

3. Hvis en sensorlogning ikke kan udføres nøjagtigt til et ønsket tidspunkt skal tidsforsinkelsen registreres med opløsning på 1 ms.

Som det kan ses af figur 11.1 anvendes en stor del af systemets funktionalitet i denne tilstand.
For at kunne imødekomme kravet om fleksibelt valg af tidspunkter, anvendes en styring der baserer sig på den aktuelle tid og dato der gemmes permanent i RTC'en[1]. Hvert sekund sammenholdes den aktuelle tid med sensorernes tidspunkter og det afgøres om det er tid til at udføre en logning.
Konfigurationsformat af logningstidspunktet er inspireret af formatet fra Unix

[1] Real Time Clock



Figur 11.1: Fleksibel langtidslogning.

daemonen "cron", der på disse systemer anvendes til at eksekvere planlagte opgaver. Et entry af dette format består af en linie med tre felter. Felterne er separeret med mellemrum, og hvert felt er et heltal der beskriver følgende:

1. Sekund [0-59]
2. Minut [0-59]
3. Time [0-23]

Når der står et heltal i et felt, laves der en direkte sammenligning af tallet med den relevante nuværende tidspunkt. Hvis alle tre felter matcher med det nuværende tidspunkt er det det ønskede tidspunkt nået.
I hvert af disse felter kan der stå et jokertegn (*), der betyder at feltet ved sammenligning kan antage alle værdier, dvs. at tidspunktet vil matche for alle værdier af feltet.
Desuden kan et felt angives som $*/n$, der får feltet til at matche når at tallet n går op i tidsfeltet uden rest, dvs. $t\%n = 0$.
Eksemplerne i tabel 11.1 illustrerer brugen af formatet.
Hele funktionaliteten omkring den fleksible tidsstyring er vist på figur 11.1 som blokken "Check sensorschedule".

Efter at det er blevet afgjort om, hvorvidt der skal udføres en logning indlæses sensorens tilhørende FPAA-konfiguration (jævnfør afsnit 13). For den anvendte FPAA er der i databladet ikke nogen dokumentation for, hvor lang tid det kan tage for at kredsen er klar efter indlæsningen af en konfiguration. Istedet er der på kredsen ført et ben ud der kan polles for at afgøre om kredsen er klar til

Linie	Betydning
`*/1 * *`	Hvert sekund
`*/5 * *`	Hvert 5. sekund
`*/1 30 *`	Hvert sekund i det 30. minut af alle timer
`0 0 12`	12:00:00

Tabel 11.1: Eksempler på brug tidskonfigurationsformat.

brug.
Da det derfor ikke kan vides med sikkerhed, hvilken tid der går fra at logningen ønskes foretaget til kredsløbet er klar og måling reelt set foretaget (Samplingstidspunktet for A/D konverteringen), er det derfor blevet valgt at registre denne tid og gemme den sammen med målingen (illustreret som "Time offset" på figur 11.1). Analyseværktøjet der senere skal behandle data, kan så foretage en tidsmæssig justering af måleresultaterne.

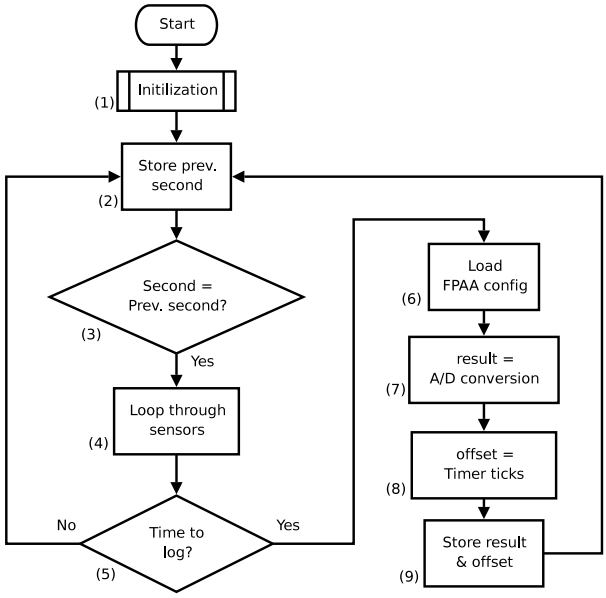
På figur 11.1 er det illustreret, hvordan at efterbehandling kan foretages i systemet (blokken "Post-processing"). Dette kunne f.eks. være forbedring af måleresultater og digitale filtre som beskrevet i afsnit 12.3.

Efter at data er blevet gemt til det permanente lager, lukkes systemets mikrocontroller ned i en energibesparende tilstand, og den sættes til at vente på at det næste sekund er gået.

11.2.1 Implementation

11.2.1.1 Logningsforløb

Systemets langtidslagring er implementeret i funktionen `loggingloop_flexible()`.
Flowet i funktionen kan ses på figur 11.2.
Ved initialiseringen i (1) indlæses konfigurationsfilerne til tidsstyringen fra filer på det permanente lager og disse gemmes i RAM så indlæsning kan foregå hurtigt under kørsel. Ved initialiseringen indlæses ligeledes konfigurationerne til FPAA'en og associeres til tidskonfigurationerne. Endelig fjernes gamle logfiler fra lageret, der skrives headere til nye logfiler og bufferen til lagring initialiseres.
Logningsforløbet starter fra (2), hvor det nuværende sekund gemmes i en midlertidig variabel. Dette har tilsammen med testen i (3) til formål at starte kontrollen af tidskonfigurationer, netop når der er gået et sekund. Når sekundet bliver inkrementeret pga. et timerinterrupt gås der videre til (4).
Løkken (4) løber igennem de initialiserede sensorer og der testes vha. (5) om den aktuelle sensor i gennemløbet skal udføre en logning. Hvis det ikke er tid, gås der tilbage til (2).
Hvis det er tid til at logge, loades FPAA konfigurationen i (6) og efter at kredsen melder klar kan A/D konverteringen (7) udføres.
Offsettet i (8) beregnes ud fra antallet af timer ticks siden det sidste timerinterrupt der kommer hvert sekund. Tidsoffsettet er dermed den tid der går fra



Figur 11.2: Flowchart over implementation af fleksibel langtidslogning.

at logningen ideelt set skulle have været udført (på selve timerinterruptet) og til at systemet udfører A/D konverteringen i (7), med en negligibel usikkerhed pga. en testløkkes turn-around tid.
Resultatet gemmes endelig på det permanente lager i (9) og der gås tilbage til indgangen på datalogningsforløbet.

Som det fremgår af figur 11.2 anvendes strømbesparelsesfunktionaliteten af mikrocontrolleren ikke som beskrevet i designafsnittet. Dette er udeladt, da det blev vurderet at strømbesparelse på dette punkt ikke ville nytte meget eftersom resten af softwaren og hardwareplatformen ikke tager hensyn til det.
Der er ligeledes ikke implementeret nogen form for efterbehandling af data, da der i de afprøvne applikationer ikke har været behov for dette.

Som sekundtæller anvendes ATMega128'erens 16 bits Timer1, der er sat op til at køre med en en clockdivisor på 256. Med hardwareplatformens 16 MHz eksterne clock, resulterer dette i at timeren kører med $16Mhz/256 = 62,5kHz$. Dette betyder at hvert timertick er på $\approx 15,99\mu s$.
For at kunne opfylde krav 3, anvendes som beskrevet timerens ticks for at afgøre tidsoffsettet. Hvert tick er som det kan ses en skæv værdi og skal først skaleres til en mere bekvem værdi i millisekunder.
Antal timerticks per millisekund kan beregnes til

$$\frac{1ms}{15,99\mu s} \approx 62,500 \text{ tics} \tag{11.1}$$

Dette betyder at antal timerticks skal divideres med 62,5 for at konvertere til antal millisekunder. Dette er dog en upraktisk størrelse at dividere med og det

er derfor valgt at runde op til 64 timerticks, hvilket er 1,024 ms. Dette resulterer i ca. 2% fejl, men gør det til gengæld muligt at udføre divisionen mere effektivt, da at dividere med 64 svarer til at skifte mod højre 6 gange. Offset i ms beregnes i praksis da som

$$\text{offset i ms} \quad = \quad \text{tics} >> 6 \tag{11.2}$$

11.2.1.2 Tidsstyring

Indlæsningen af konfigurationsformatet for logningstidspunktet foregår igennem funktionen `readConfData`, der tager sig af al indlæsning af konfigurationsdata. De forskellige felter indlæses af hjælpefunktionerne `readSecondField`, `readMinuteField`, og `readHourField`. Disse tager sig af at fortolke felterne i overensstemmelse med formatet som tidligere beskrevet og gemmer denne fortolkning i en konfigurationsstruktur der gør det nemt senere at aflæse hvordan et felt skal tolkes.

Når det skal afgøres om en sensor skal logge, anvendes funktionen `timeToLog`. Givet en sensorkonfigurations komplette tidsfelt som sat af ovenstående funktioner, afgør denne funktion om tidsfeltet stemmer overens med det nuværende tidspunkt.

11.3 Fastintervalslogning

Fastintervalslogningen er den traditionelle datalogningsmetode i den forstand, at den udfører målinger på én enkelt sensor med et fast sampleinterval igennem hele måleperioden.
Logningen med fast interval er designet ud fra følgende krav:

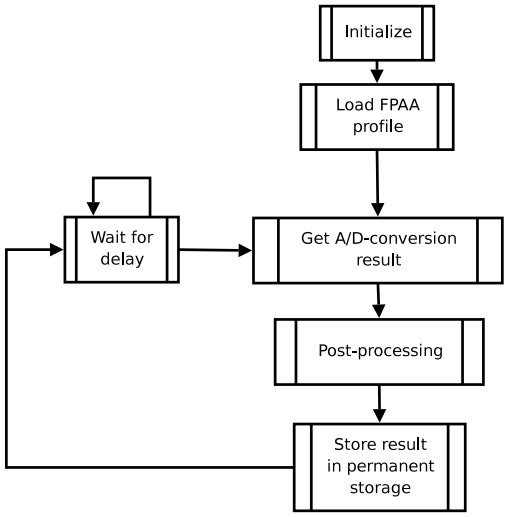
1. Logningen skal kunne indstilles til et fast interval i logningsperioden på mellem 1 sekund og en nedre grænse defineret af lagringssystemets maksimale throughput.

2. Dataloggeren er begrænset til kun at anvende én FPAA konfiguration i logningsperioden.

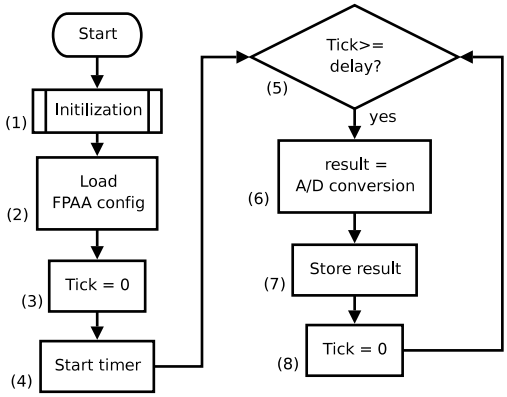
Før datalogningsforløbet initialiseres systemet og den ønskede FPAA konfiguration indlæses. Som det fremgår af figur 11.3 anvendes der ikke nogen speciel funktionalitet til at bestemme hvornår logningen udføres. Istedet anvendes der en fast ventetid inden logningsforløbet gentages. Efter at A/D konvertering er udført, er det muligt at lave efterfølgende post-processing f.eks. digital filtrering eller lign. En omfattende post-processering vil dog nedsætte systemets samlede throughput under datalogningen.

11.3.1 Implementation

Systemets hurtige logning er implementeret i funktionen `loggingloop_fixed`. Dens overordnede flow kan ses på figur 11.4



Figur 11.3: Hurtig logning med en sensor.



Figur 11.4: Flowchart over implementation af hurtig logning.

Ved initialiseringen (1), indlæses konfigurationen på samme måde som beskrevet for den fleksible logning i afsnit 11.2.1.1, men kun for én enkelt sensor. Da de samme funktioner anvendes indlæses også konfigurationsdata som ikke er relevant for fastintervalslogningen, men af hensyn til simpelhed indlæse alle data. Ligeledes indlæses alle FPAA konfigurationer i trin (2) men kun den første anvendes.
Før at logningsforløbet startes, nulstilles variablen der tæller antal timerticks og timeren startes i trin (3) og (4). Derefter startes selve logningsforløbet fra trin (5). Der afgøres her om det fornødne antal ticks er gået ved en sammenligning i en løkke.
Når den ønskede tid er gået, udfører systemet en A/D konvertering (6) og resultatet gemmes til det permanente lager i (7) og til sidst nulstilles ticks.

Til at styre tiden, anvendes 8-bits timeren Timer2 på mikrocontrolleren kørende

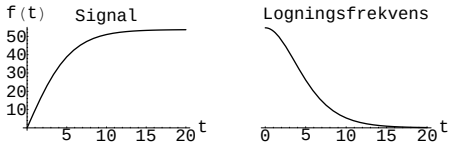
med en clockdivision på 64. Dette resulterer i at timerens overflow kommer hvert 1,02 ms. Med accept af en 2% fejl på tidsstyringen bruges timerens overflow interrupt til at at tælle variablen Ticks op. Derved opnås den ønskede granulitet

11.4 Variabel logning

Den variable logning har til formål at kunne lave logninger over længere tidsforløb, samtidig med at hurtige ændringer skal kunne registreres.
Den variable logning er blevet designet ud fra følgende krav:

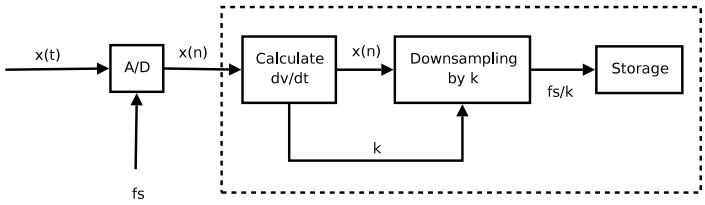
1. Hurtige ændringer i signaler skal kunne registreres
2. Kun datapunkter nødvendige for at beskrive signalet skal medtages.

Den begrænsende faktor for at kunne logge hurtigt varierende signaler består i lagringen til det permanente medie, da dette har et throughput væsentligt lavere end resten af systemet. Dermed bliver det maksimale permanente throughput for hele systemet begrænset af det permanente medie's hastighed.
Ved anvendelse af skrivebuffere er det muligt at komme over denne begrænsning, men grundet begrænset RAM-mængde kan dette kun gøres i et kort tidsrum. Den variable logning basere sig på dette, ved kun at anvende en høj lagringshastighed når det er nødvendigt for at beskrive signalet tilstrækkeligt.
Figur 11.5 illustrerer, hvordan logningsfrekvens justeres alt efter signalets udseende.



Figur 11.5: Justering af samplingsinterval ud fra signal.

Ved hurtigt varierende signaler (højere frekvens) med en stor ændring af signalet per tidsenhed, køres der med højere logningsfrekvens end hvis at signalet kun varierer langsomt. Ved at registrere den anvendte logningsfrekvens sammen med signalværdien er det muligt at anvende interpolation når logdata senere skal analyseres.
Til at bestemme om det er nødvendigt at køre med en høj logningsfrekvens, kan signalets variation $\frac{dv}{dt}$ eller for det diskret-tid tilfælde $\frac{\Delta v}{\Delta t}$ anvendes som udgangspunkt. Ved hele tiden at køre med med en høj samplingsfrekvens på A/D konverteren og gemme to målinger i hukommelsen kendes den nuværende variation. Ved en høj variation i den målte værdi imellem to eller flere målinger lagres der flere målinger til det permanente medie. Hvis der kun er en lav variation anvendes en lavere lagringshastighed.
Princippet i den variable logning kan ses på figur 11.6.



Figur 11.6: Datalogningssystem med variabel logning.

På systemet der fremgår af figur 11.6, bruges signalets variation $\frac{dv}{dt}$ [2] til at afgøre med hvilken frekvens der skal gemmes til det permanente lager. Downsamplingen der fødes med parameteren k smider ganske enkelt hvert k'ende sample væk. Frekvensen hvormed der gemmes til det permanente lager bliver så f_s/k.
Ved en implementationen bør der være en vis hysterese og skalering af downsamplingen med faktor k så at følsomheden kan justeres. Derudover vil det være nødvendigt at have en mekanisme som sørger for at der ikke skrives for hurtigt til det permanente lager over en for lang periode.

11.4.1 Implementation

Den variable interpolerende datalogning er ikke blevet implementeret af tidsmæssige årsager i dette projekt. Det vurderes dog at det med det ovenstående design og de tilgængelige funktioner vil være muligt at lave en implementation ud fra ovenstående design.

[2] For det diskrete samplede signal er det egentlig $\frac{\Delta v}{\Delta t}$, men notationen misbruges for klarhedens skyld

Kapitel 12

Analog-digital konvertering

12.1 Indledning

Analog-digital konverteringen i systemet er ansvarlig for at oversætte de analoge signaler fra signaltilpasningsdelen til digitale repræsentationer der kan fortolkes af systemets mikrocontroller og lagres permanent til senere analyse.
Dette afsnit beskriver analog-digital konverteringen i systemet og de overvejelser der er gjort mht. til forbedring af måleresultater.

12.2 Opløsning og kalibrering

Systemets mikrocontroller, en ATMega128 har en integreret A/D konverter, jævnfør [Atmel, 2006a]. Denne er af SAR-typen[1] og har følgende generelle egenskaber:

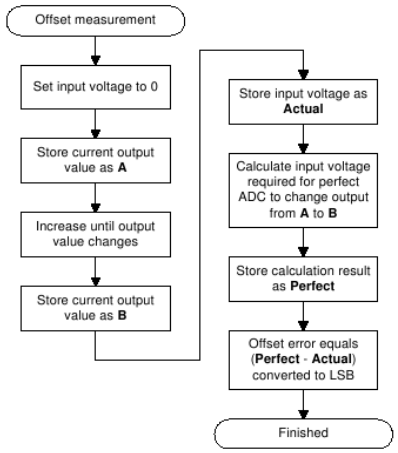
- 10-bits opløsning (max)
- Single conversion eller free-running.
- Single-ended og differentielt input
- Mulighed for interrupt ved færdig konvertering
- Mulighed for konvertering i støjreduktionstilstand.
- Variabel konverteringshastighed/opløsning

Før at A/D konverteren kan anvendes effektivt, skal den kalibreres for at opnå den bedste nøjagtighed. Dette gør sig især gældende ved anvendelse af det differentielle input, da der ud over selve A/D konverterens vil være en påvirkning fra den indbyggede operationsforstærker (der anvendes ved differentielt input). Ved differentielt input giver det sig til kende ved betydelige offset- og gainfejl, ifølge databladet resulterende i en samlet absolut fejl på op til 17 LSB. Gain- og offsetfejl kan dog på en relativ enkel vis bortjusteres.
Offsetfejlen er forskellen mellem det aktuelle skift af den første bitkode og $\frac{1}{2}$

[1] Successiv Approksimation.

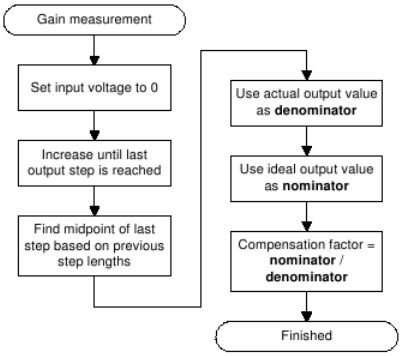
LSB (med andre ord, hvor meget A/D konverteren afviger fra værdien ved en udstyring 0 V). Gainfejlen er defineret som fejlen ved fuld udstyring (V_{FSR}), efter at offset er blevet bortjusteret.

Offsetfejlen kan i praksis findes ved følgende fremgangsmåde som vist på figur 12.1 (gengivet fra [Atmel, 2006b]



Figur 12.1: Flowchart for måling af single-ended offsetfejl.

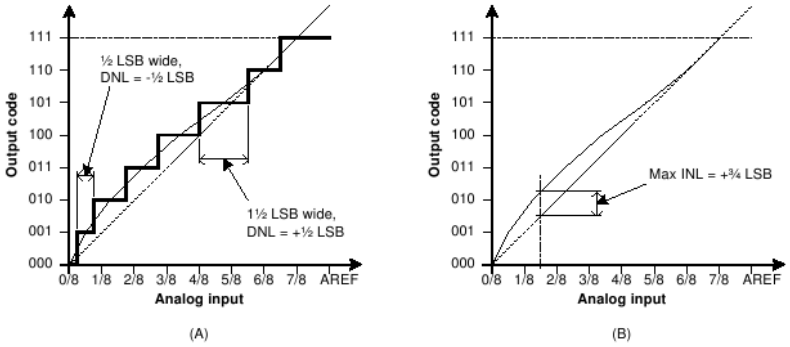
Ved differentielle målinger er det enklere at bestemme offset fejlen for den indbyggede operationsforstærker der forstærker differencen imellem de to inputs. Dette gøres ganske enkelt ved at forbinde de to inputs til det samme potentiale, og offsetfejlen for forstærkningen kan aflæses direkte. Desværre fås ikke nogen information omkring selve A/D konverterens transitionsfejl og en offsetfejl på $\frac{1}{2}$ til 1 LSB kan ikke undgås.
I begge tilfælde fås den korrigerede værdi ved at fratrække den beregnede offset fra de målte værdier.



Figur 12.2: Flowchart for måling af gainfejl.

Efter at have korrigeret for offsetfejlen kan gainfejlen findes. Dette kan gøres som vist på figur 12.2 ([Atmel, 2006b]).

Med korrigering af både offset- og gainfejl er der taget hensyn til de vigtigste fejlkilder. Tilbage er der integral og differentiel non-linearitet (INL og DNL) der kan betyde en ulineær afvigelse indenfor 0 V og V_{FSR}. Figur 12.3 illustrerer for en 3-bits A/D konverter, hvordan INL og DNL skal tolkes.



Figur 12.3: Eksemepl på en ikke-lineær A/D konverter karakteristik.

Integral non-linearitet (figur 12.3A) er defineret som den maksimale vertikale forskel imellem den aktuelle og den perfekte karakteristik.
Differentiel non-linearitet (12.3B) er defineret som den maksimale/minimale forskel mellem den aktuelle bredde af steps og hvad der ville være ideelt (1 LSB).
De er dog besværlige at kompensere for da de vil kræve en måling der kan udfylde en lookuptabel til kompensation over hele intervallet. For ATMega128'eren er INL typisk på 0.75 LSB for single-ended og mellem 1.5 og 5 LSB ved differentielt alt afhængig af valgt gain på operationsforstærkeren. Den maksimale differentielle non-linearitet ligger typisk på 0.5 LSB uafhængigt af hvilken tilstand der køres i.

A/D konverteren er af SAR-typen og kan indstilles til at køre med forskellige clockfrekvenser. For at have 10-bits opløsning med de ovenstående absolutte nøjagtigheder er det nødvendigt at køre med en frekvens på A/D konverteren på mellem 50 og 200 kHz. For systemets 16 MHz clock betyder dette at den eneste acceptable prescalerværdi er 1/128. Dette giver en clockfrekvens på 125 KHz for A/D konverteren.
Ved en normal konvertering, tager det 13 A/D clockcyklusser for at fuldføre en konverteringen. Frekvensen, hvormed der kan foretages 10-bits målinger (samplingsfrekvensen) bliver da så

$$\frac{125kHz}{13} = 9.615kHz \tag{12.1}$$

Ønskes der en højere samplingsfrekvens kan dette foregå på bekostning af en lavere opløsning. Databladet dokumenterer nøjagtigheden af målinger op til 1 MHz, og det kan derfor ikke anbefales at gå ud over dette. 1 MHz kan rammes

præcist ved at anvende en prescalerværdi på 16. Den maksimale samplingsfrekvens bliver da så

$$\frac{1MHz}{13} = 76.9kHz \tag{12.2}$$

12.3 Forbedring af måleresultater

12.3.1 Støjreduktion

Da A/D konverteren er indbygget i systemets mikrocontroller, er den ekstra sårbar overfor indstrålet støj fra mikrocontrollerens CPU-kerne og I/O systemer.
Den anvendte ATMega128 har dog en speciel tilstand benævnt "ADC Noise Reduction", der er specielt beregnet til at afhjælpe dette problem. Tilstanden standser CPU-kernen, flash og I/O og starter derefter en A/D konvertering. Efter at konverteringen er færdig fortsætter mikrocontrolleren som normalt. Dette forbedrer støjforholdende for A/D konverteren og en bedre præcision kan opnås. Systemer såsom den interne timer og interruptcontrolleren kan dog fortsat fungere, så at det er muligt at bringe mikrocontrolleren ud af tilstanden.

Kapitel 13

Sensorstyring (FPAA)

13.1 Indledning

Som beskrevet i hardwareafsnittet anvender hardwareplatformen en dynamisk rekonfigurerbar FPAA fra Anadigm, der gør det muligt at lave en lang række analoge funktioner i én kreds, f.eks. filtre, forstærkning, mm.
Firmaet Anadigm er ikke de eneste der udvikler disse kredse, men de skiller sig dog ud ved at tilbyde en række grafiske designprogrammer der muliggør design af analoge kredsløb på et højt abstraktionsniveau. Det er derfor ønskeligt at lade dataloggeren understøtte designs udført i værktøjet.
I dette afsnit beskrives styringen af FPAA-kredsen, samt processen i at overføre analoge design fra FPAA designværktøjet til brug i systemet.

1. Et design fra værktøjet AnadigmDesigner 2, skal kunne bruges af dataloggeren som en sensorkonfiguration.

2. Dataloggeren skal kunne skifte automatisk imellem de valgte sensorkonfigurationer, når den først er programmeret.

3. Forløbet fra design af det analoge kredsløb i designværktøj over til brug af konfigurationen på dataloggeren skal være så enkelt som muligt.

13.2 Design

Efter at have designet det analoge kredsløb i designværktøjet ([Anadigm, 2006a]), er det muligt at overføre konfigurationsdata til FPAA'en på overordnet set to måder.
Den primære konfiguration anvendes til at lave en fuld konfiguration af FPAA'en. Denne metode skal altid anvendes efter et reset af kredsen. Den primære konfiguration består af en header blok efterfulgt af en eller flere datablokke. Datablokken indeholder informationer om, hvordan designet skal realiseres i kredsen. For AN221E04 kredse [Anadigm, 2006b] bliver den fulde primære konfiguration på 590 bytes. Datablokke hvor der står 0x00 kan dog udelades[1] og en

[1] Ved reset af kredsen kommer den interne SRAM i en kendt tilstand (alle bits 0) og det er derfor kun nødvendigt at skrive de steder hvor der skal stå 1.

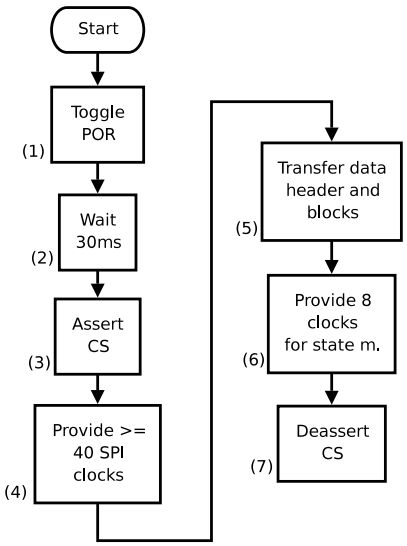
mindre størrelse opnås, ved at aktivere kompression i designværktøjet.
Når først en primær konfiguration er blevet overført til kredsen kan kredsen hurtig reprogrammeres vha. en update konfiguration, da det kun er nødvendigt at udskifte bits der er forskellige imellem konfigurationerne. AN221E04 kredsen har yderligere den fordel at det ikke er nødvendigt at resette kredsen ved ny konfigurationsindlæsning pga dens Shadow RAM. Denne metode at overføre konfiguration på benævnes som "Static Configuration".
For applikationer der måtte ønske at kunne stille på designparametre, f.eks en operationsforstærkers gain samt at loade nye konfigurationer så hurtigt som muligt, har designværktøjet to metoder til dette: "Algorithmic" og "State Driven". Disse to betegnes som "Dynamic Configuration". Begge anvender C-kode der automatisk genereres af designværktøjet og vil derfor kræve en omprogrammering af systemets mikrocontroller ved udskiftning af konfigurationer. Set i forhold til ønsket om at have en hurtig transition fra design af det analoge kredsløb til brug i systemet er de dynamiske løsninger derfor ikke velegnede, selvom de hver især har fordele mht. tidsforbruget på konfigurationen og fleksibilitet. Den statiske konfiguration er derimod en selvstændig enhed, som ikke er afhængig af specifik tilpasset programkode i systemets mikrocontroller og fortidige konfigurationer af FPAA'en. Brugeren behøver derfor ikke bekymre sig om, hvilken programkode mikrocontrolleren kører. Af hensyntagen til en hurtig transition fra design til ibrugtagen af systemet, er det derfor blevet valgt at bruge den statiske konfigurationsmetode.

For den statiske konfigurationsmetode er der flere muligheder for at overføre data til FPAA'en.
Første mulighed er at sende konfigurationen som en rå datastrøm til en serielport. Denne metode anvendes blandt andet til Anadigms egne evalueringsplatforme.
Anden mulighed er at gemme konfigurationen i en fil. Som filformat kan der vælges blandt andet Intel og Motorola hex, samt rå data i ASCII hex og binært format.
Af de nævnte muligheder, er en eksport af en konfigurationsfil den mest interessante. Dette lægger op til at den statiske konfiguration overføres direkte til systemets lagringsmedie, som i forvejen indeholder tidskonfigurations- og logningsfiler. Overførslen af den statiske konfiguration til systemet vil med denne fremgangsmåde derfor være begrænset til ganske få, hurtige trin.
Det er derfor med hensyntagen til krav 3 blevet valgt at lade de statiske konfigurationsfiler ligge sammen med tidskonfigurationsfilerne. Af de forskellige filformater er det rå, binære format blevet valgt. Da dette er ud over at være pladsbesparende har den fordel at det ikke kræver en senere konvertering i mikrocontrolleren når det skal sendes til FPAA'en. Mapning af de statiske konfigurationsfiler til tidskonfigurationsfilerne gøres på simpel vis ved nummereret navngivning af filerne.
Jævnført krav 2 er det et krav at systemet skal kunne skifte imellem de forskellige statiske konfigurationsfiler til FPAA'en. Det er derfor nødvendigt at systemets mikrocontroller har disse tilgængelige.
En oplagt, effektiv løsning ville være at ligge samtlige profilers konfigurations-

data over i RAM på et ikke-kritisk tidspunkt, f.eks. under opstart. Overførselshastigheden vil så i praksis under kørsel kun være begrænset af SPI-interfacets hastighed. Ulempen er at et område af RAM kun kan sættes af til dette formål under datalogning. For den aktuelle mikrocontroller, ATMega128 der har 4KB RAM, vil f.eks. 4 ukomprimerede konfigurationsfiler på hver 576 bytes optage over 50% af den samlede mængde hukommelse. Da filsystemet og lagringsmediet i forvejen kræver mere end 25% af RAM vil en selv meget optimeret kode have svært ved at passe i hukommelsen.
Et alternativ er at anvende ATMega128 mikrocontrollerens EEPROM. Denne har en kapacitet på 4KB, hvilket er nok til 7 ukomprimerede konfigurationsfiler. Skrivningen til EEPROM tager lang tid (ifølge databladet typisk 8.5ms), hvorimod læseadgangen tager 4 clockcyklusser. I modsætning til mikrocontrollerens RAM, kan konfigurationsdata opbevares i EEPROM selvom strømmen tages fra. Dette kunne f.eks. udnyttes til at have bestemte konfigurationer fast programmeret i systemet.
Pga. de ovennævnte begrænsninger mht. RAM er det blevet valgt at lægge konfigurationerne i EEPROM. En mulighed havde været at udvide hardwareplatformen og tilføje ekstern RAM til mikrocontrolleren, men dette lod sig ikke gøre af tidsmæssige årsager for hardwaredesigneren.

13.3 Implementation

Proceduren for at loade en primær konfiguration til FPAA'en er implementeret i `fpaa_loadConfig` og kan ses på følgende figur 13.1.



Figur 13.1: Primær konfiguration af FPAA.

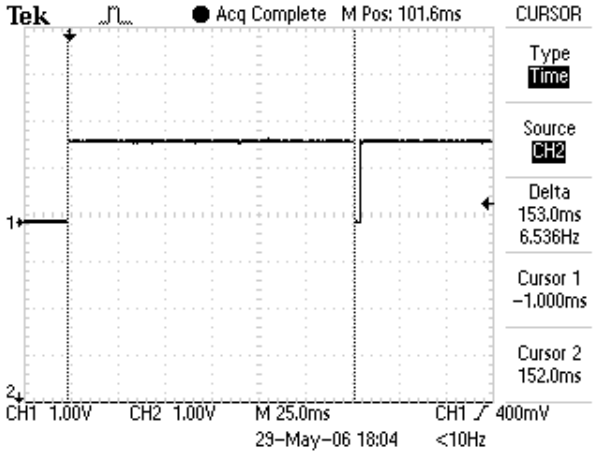
For at nulstille kredsen og sætte den i en tilstand, hvor den kan programmeres skal Power-On-Reset benet på kredsen toggles høj-lav-høj og efter en ventetid på 30 ms (2), kan kredsen chip-selectes.

Internt består konfigurationslogikken i kredsen af en tilstandsmaskine der skal færdiggøre sin interne opstartssekvens før at den kan programmeres. Dette gøres ved at sende 40 clockcyklusser (svarende til fem 8-bits overførsler på SPI-bussen. Kredsen er nu klar til at modtage primær konfigurationsdata, hvilket sendes i trin (5). Efter sidste databyte skal kredsen have 8 SPI clocks (6) til at færdiggøre programmeringen.
Kredsen kan nu deselectes og Activate udgangen på kredsens polles. Når denne går høj, betyder det at kredsen er klar til brug.
Det skal dog bemærkes, at det analoge kredsløb som FPAA'en realiserer, efter realiseringen vil have et transient forløb indtil at det når en steady-state. Dette vil især gøre sig gældende for f.eks. feedback kredsløb af højere orden. Det er derfor nødvendigt at være påpasselig med dette og vente til at realiseringen har stabiliseret sig, så transiente forløb ikke fejlagtigt medtages i en A/D konvertering.

Den anvendte metode til at loade konfigurationer er ikke optimal set ud fra et ønske om hurtig konfiguration, da der som minimum skal ventes i 30 ms efter Power-On-Reset. Metoden er dog simpel, og har ingen afhængigheder af foregående konfigurationer.
Ved at udnytte den benyttede FPAA's shadow-ram vil det være muligt at loade konfigurationen med kun en meget lille afbrydelse. For at udnytte det fuldt ud, skal det dog afgøres før tid, hvilken konfiguration der skal anvendes. Samlet set er det skønnet at dette ville være for omfattende at implementere i dette projekt.

13.4 Test

Testen er blevet udført ved at lade `fpaa_loadConfig` aktivere LED6 når der gås ind i funktionen, og deaktiveres når der gås ud og måle med et storagescop. Tiden for indlæsning af en 166 byte primær konfiguration kan ses på figur 13.2



Figur 13.2: Tid for primær konfiguration af FPAA.

Det kan ses af figuren at det tager 153 ms for at fuldføre konfigurationen. Af denne tid udgør 30 ms tiden der ventes efter POR (samt 5+166+1 SPI-overførsler hvilket med den anvendte bitrate på 4 Mbps tager kun en lille andel på $((5 + 166 + 1) \cdot 8)/4 \cdot 10^6 = 344\mu s$). Dette vil sige at kredsen er omtrent 120 ms om at færdiggøre sin konfiguration i denne tilstand.

Kapitel 14

Delkonklusion

Igennem dette projekt er der blevet udviklet software og drivere til dataopsamlingssystemets hardwareplatform.
Med det udviklede system er det muligt på hurtig vis at lave analog signaltilpasning for at kunne afprøve en række sensorer. Dette gøres ved at lægge konfigurationsfilen fra designværktøjet sammen med tidskonfigurationsfiler på et SD/MMC-kort formateret med et FAT filsystem. Ved kørsel sørger systemet automatisk for at anvende den ønskede analoge signaltilpasning.
Systemet kan ud over en traditionel fastintervalslogning udføre dataopsamling på basis af den aktuelle tid og dato vha. hardwareplatformens RTC. Igennem et fleksibelt format til at definere logningstidspunkter er det muligt at vælge imellem en lang række logningsintervaller. De opsamlede måledata bliver lagret i CSV-format på systemets SD/MMC-kort. Dette kan umiddelbart efter endt logning overføres til f.eks. en PC for videre analyse.
Til anvendelsen af systemet, er der blevet implementeret en begrænset brugerflade, der kan anvendes igennem enten RS232 eller et tilkoblet board med LCD og knapper. Denne har dog ikke været fokuseret på i projektet og denne er derfor med mest som proof-of-concept.
Der er følgende forslag til forbedringer som bør laves på en senere version af softwaren:

- Den nuværende brugerflade på RS232 og det tilkoblede board har forskellige uoverensstemmelser mht. funktionalitet, bla. kan ikke alle tilstande vælges via RS232.

- Konfigurationen af FPAA'en i systemet foregår på en pålidelig og nem metode men denne foregår langsomt. For at opnå bedre logningshastigheder anbefales det at gøre brug af updatekonfigurering og kredsens shadow ram.

- Filsystemets driver sætter ikke oprettelses- og modifikationstidspunkter korrekt. Dette kan gøres ved at modificere driveren til at bruge systemets tid. Sletning af store logfiler på systemet tager lang tid. Det anbefales derfor at slette gamle logfiler på en PC før at der logges.

- Grundet implementationen af systemets tid og dato der vedligeholdes af

mikrocontrollerens timer, er der en vis drift i tiden. Dette kan afhjælpes ved at anvende en mere avanceret styring af tiden eller ved at synkronisere med hardwareplatformens RTC flere gange under logningsforløbet istedet for kun at gøre det under opstart.

- A/D konverteren i systemet er ikke kalibreret, så størst mulig nøjagtighed realiseres ikke med den nuværende implementation.

Del IV

Samlet afslutning

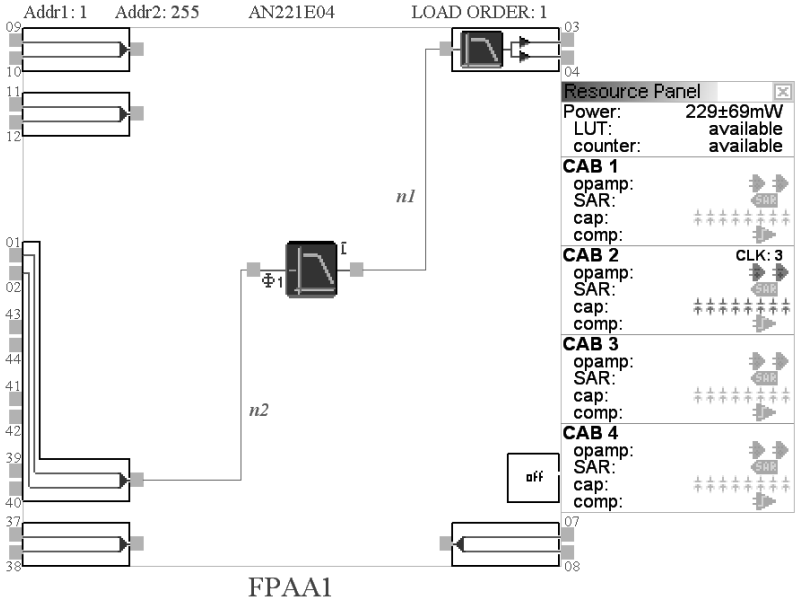
Kapitel 15

Brug af data loggeren

15.1 Brug af software

Dette afsnit gennemgår, hvordan en typisk datalogningsproces foregår fra design af den analoge signaltilpasning over til analyse af data.

15.1.1 Design i AnadigmDesigner 2

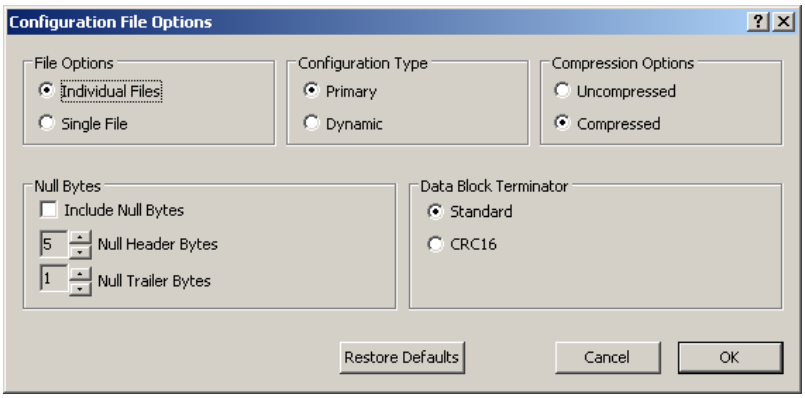


Figur 15.1: Design af et lavpasfilter i FPAA.

Figur 15.1 viser et analogt design i AnadigmDesigner2 værktøjet. Vist er et biquad lavpasfilter med afskæringsfrekvens på 20 kHz.
Inputsignalet tages differentielt fra benene I4PA og I4NA, ved at vælge dette i multiplexeren på input celle 4 (Input line select: A). Efter lavpasfilteret føres dette ud igennem et generelt lavpasfilter til output til A/D konverteren.

Figur 15.2: Valg af konfigurationsformat.

Dernæst vælges "Write Configuration Data to a File" fra menuen "Configure". Her skal configurationen stå som følgende figur 15.3



Figur 15.3: Options for valg af konfigurationsformat.

Her er det vigtigt at der vælges primær konfiguration, standard block afslutning, ingen inkludering af null bytes og komprimeret format.
Den binære primære konfiguration gemmes nu i en mappe "CONF" på et MMC/SD-kort, sammen med en tidskonfiguration. Da der her kun køres med én enkelt sensor, navngives filerne henholdsvis "fpaa1.bin" og "1.cfg". Under opstart af en datalogning, vil loggeren lede efter konfigurationsfiler navngivet fra 0 til 10. Det maksimale antal sensorer der kan bruges af gangen er 4.
Indholdet af tidskonfigurationen er følgende

```
1 Name sens1
2 LogInterval 110
3 TimeField */1 * *
```

Feltet "Name" er til at identificere sensoren efter at datalogning er udført, og kopieres over til headeren af den skrevne logfil. "LogInterval" angiver tidsforsinkelsen imellem hver logning når der køres i fastintervalslogning, her sat til 110 ms. "TimeField" angiver det fleksible tidskonfigurationsformat til den fleksible logningsmetode, her indstillet til logning hvert sekund.
Indholdet af konfigurationsbibloteket er da så som på figur 15.4
Udover bibloteket "CONF" skal der være et biblotek i roden af MMC/SD-kortet navngivet "LOGDATA" til lagring af logningsfiler, da dataloggeren ikke selv kan oprette bibloteket.

Kortet kan nu indsættes *i den slukkede datalogger* og strømmen kan nu tændes. Dataloggeren starter nu op og menuen præsenteres såfremt at der er tilkoblet

```
D:\CONF>dir
 Disken i drev D er NY ENHED
 Diskens serienummer er AC3B-CC46

 Indhold af D:\CONF

30-05-2006  00:35    <DIR>          .
30-05-2006  00:35    <DIR>          ..
27-05-2006  10:21                48 1.cfg
30-05-2006  20:09               130 fpaa1.bin
               2 fil(er)            178 byte
               2 mappe(r)      30.342.144 byte ledig

D:\CONF>
```

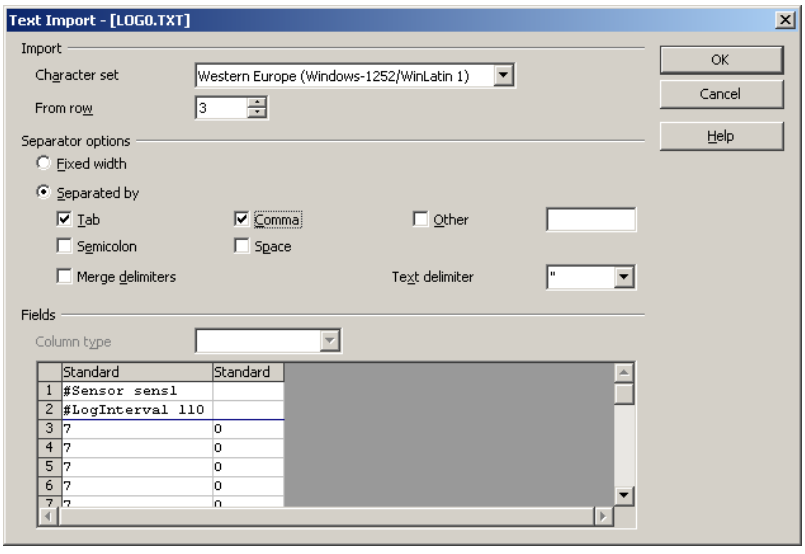
Figur 15.4: Indhold af 32 MB SD-kort med konfiguration.

et I/O board.

15.1.2 Fleksibel datalogning

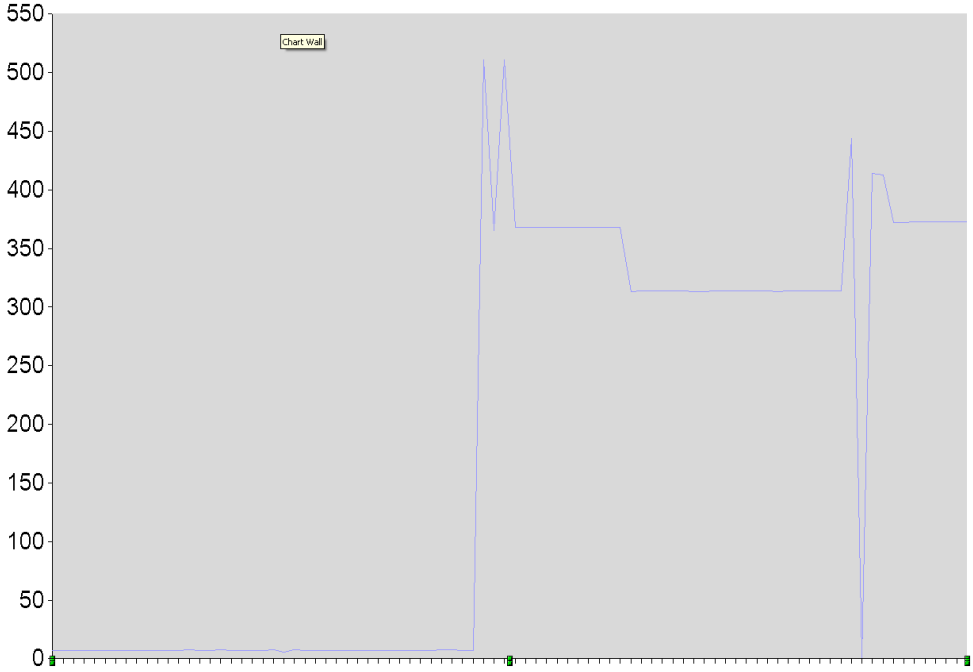
Fleksibel datalogning startes ved at vælge "Logging" og dernæst "Flexible" ved at holde de respektive taster nede indtil at dataloggeren begynder at konfigurere sensorerne.
Når konfigurationen er færdig, begynder logningen og starttidspunktet samt antallet af initialiserede sensorer vises. Løbende kan der følges med i hvor meget data der er skrevet til kortet.

Datalogningen afsluttes ved at holde den venstre tast nede indtil at dataloggeren stoppes. *Efter at strømmen er taget fra* kan kortet tages ud.
Data kan nu importeres i et analyseværktøj, f.eks. som det her på figur 15.5 viste OpenOffice Calc.



Figur 15.5: Import af logningsdata i OpenOffice Calc.

Som det kan ses af figuren, inkluderes der med selve måledata et ekstra felt. Som beskrevet i afsnittet omkring logningsformatet, bruges dette for den fleksible logning til at bestemme offsettet fra målingens ønskede start til den reelt set bliver udført i millisekunder. Da der her kun køres med én sensor, er offsettet mindre end et millisekund.
Figur 15.6 viser et plot af de målte data for en tilkoblet potentionmeter der varierer spændingen på indgangen fra 0-5 V.



Figur 15.6: Plot af logningsdata med fleksibel logning.

15.1.3 Fastintervalslogning

Fasterintervalslogning startes ved at vælge "Logging" og dernæst "Fixed" i lighed med den fleksible logning.

Med fastintervalslogningen inkluderes der kun måledata i logfilen.
Figur 15.7 viser et logningsforløb for fastintervalslogningen

15.1.4 Realtids tilstand

Udover de to datalogningstilstande, kan der vælges en "realtids"tilstand, der ikke udfører datalogning til filer, men derimod viser de målte værdier på systemets LCD samt til en tilkoblet terminal.
På den tilkoblede terminal præsenteres data i et CSV-format indeholdende alle konfigurerede sensorer. Det er derfor muligt at udføre datalogning direkte til f.eks. en PC.

Figur 15.7: Plot af logningsdata målt med fastintervalslogning.

Realtidstilstanden kører ikke med noget fast interval, men er derimod begrænset af skriving til LCD og RS232 interfacet.

15.1.5 Interaktiv tilstand

Ønskes dataloggeren anvendt uden uiboardet kan systemet gå en interaktiv tilstand hvor denne kan bruges igennem en simpel prompt på RS232 interfacet. Generelt set kan denne anvendes ved at tilkoble sig RS232 interfacet med 57600 bps 8-N-1 og VT100 terminal emulation. Tilstanden kan aktiveres ved enten at taste "i" over RS232 eller at vælge "Interactive" hvis uiboardet er tilkoblet.

15.1.6 Brug af flere sensorer

Ønskes der logget fra flere tilkoblede sensorer samtidig, kan dette gøres i fastintervalslogning og realtids tilstand. Dette gøres ved at placere fortløbende navngivne tids- og sensorkonfigurationer i "CONF" mappen, med overensstemmelse mellem filfornavnene som vist på figur 15.8
Den nuværende implementation kan køre med op til 4 sensorer samtidig.

15.1.7 Fortolkning af måledata

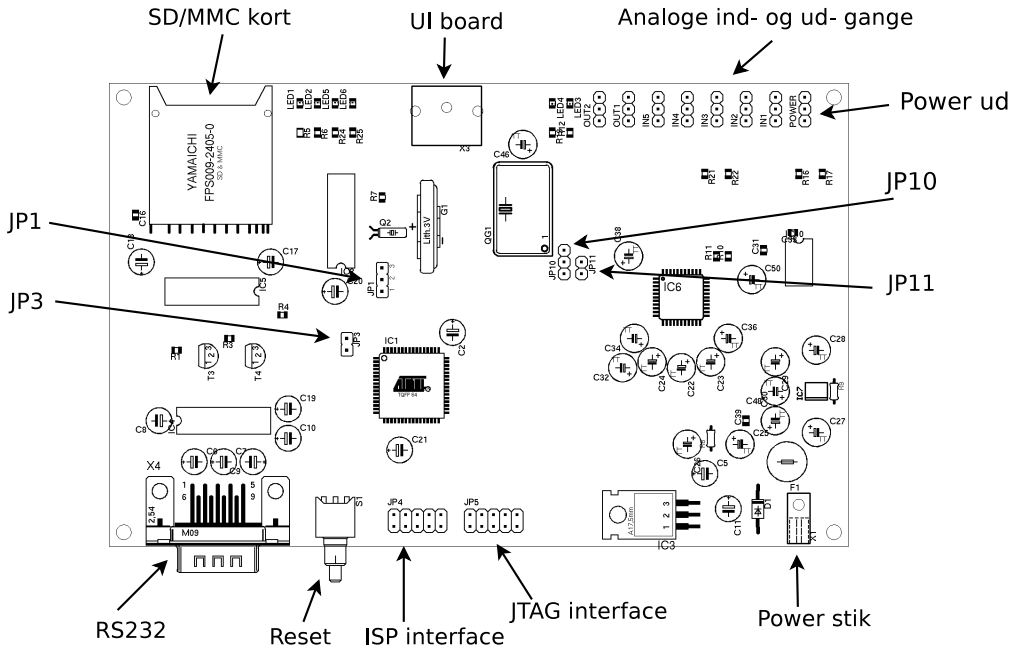
FPAA'en samt A/D konverteren i systemet kører i differentiel tilstand, hvilket betyder at data udlæses som et tal med fortegn på mellem -511 og 511. Hvad disse værdier er i spænding på A/D konverterenss inputs kan findes ved beregningen som angivet på side 242, "ADC Conversion Result" i ATMega128'erens datablad, se [Atmel, 2006a]

```
D:\CONF>dir
 Disken i drev D er NY ENHED
 Diskens serienummer er 2CD3-B3F0

 Indhold af D:\CONF

26-05-2006  17:27    <DIR>          .
26-05-2006  17:27    <DIR>          ..
14-05-2006  16:22               109 fpaa1.bin
14-05-2006  16:22               109 fpaa2.bin
14-05-2006  16:22               109 fpaa3.bin
14-05-2006  16:22               109 fpaa4.bin
27-05-2006  10:04                46 2.cfg
27-05-2006  10:04                46 3.cfg
27-05-2006  10:04                46 4.cfg
27-05-2006  10:21                48 1.cfg
               8 fil(er)            622 byte
               2 mappe(r)     254.025.728 byte ledig
```

Figur 15.8: Indhold af 32 MB SD-kort med flere konfigurationer.

15.2 Brug af hardware

I dette afsnit beskrives brugen af systemets hardware. Figur 15.9 viser en oversigt over systemets forbindelser og jumpere.



Figur 15.9: Oversigt over forbindelser og jumpere.

Herunder følger en beskrivelse af de enkelte forbindelser og jumpere:

15.2.0.1 Power stik

Her tilsluttes systemet strømforsyning. Denne skal være på 5 volt DC, og skal kunne levere mindst 1 A. Plus benet er i midten. Hvis forsyningsspændingen vendes forkert springer sikringen.

15.2.0.2 JTAG interface

Her tilsluttes JTAG adapteren. Stikket har standard Atmel layout, og alle JTAG adaptere der er kompatible med Atmels JTAG adapter kan benyttes. Systemet kan ikke spændingsforsynes fra JTAG adapteren, og den feature skal slåes fra på adapteren.

15.2.0.3 ISP interface

Dette interface kan benyttes i tilfælde af at JTAG bliver slået fra ved et uheld, eller på anden vis ikke kan benyttes. Inden interfaces kan benyttes skal det aktiveres vha. jumperen JP3.

15.2.0.4 Reset

Reset knap som resetter ATMega128 processoren, og dermed hele systemet. Forsigtighed skal udvises med at resette processoren mens der skrives til SD/MMC kortet, idet dette kan føre til datakorruption.

15.2.0.5 RS232

Standard RS232 interface. Kan benyttes til at styre systemet via et terminalprogram på en PC. Forbindes til PC'en via et null-modem kabel. Der benyttes ingen form for hardware flow control.

15.2.0.6 SD/MMC kort

Her indsættes SD eller MMC kortet som systemet benytter til at gemme loggede data. Der er ingen nedre eller øvre grænse for kortets størrelse.

15.2.0.7 UI board

Stik som tillader tilslutning af eksternt UI board.

15.2.0.8 Analoge ind- og udgange

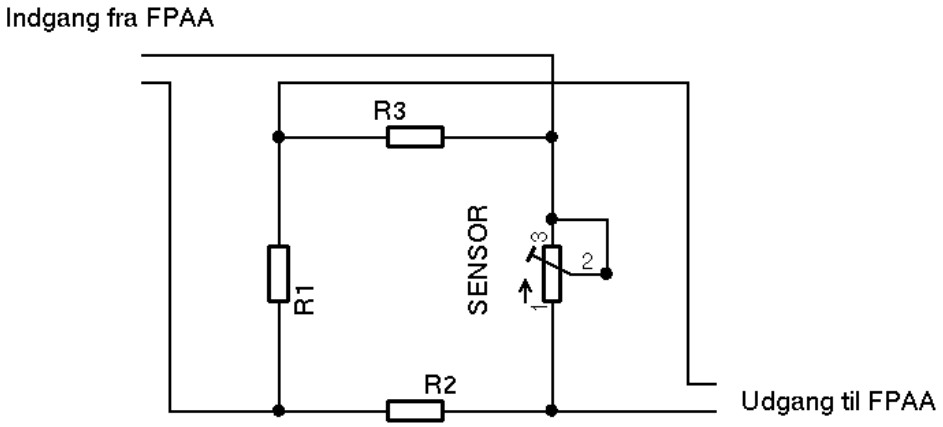
Her tilsluttes sensorerne. Betragtes stikkene fra venstre mod højere:

- **Udgang 2:** Er tilsluttet direkte til FPAA'ens IO2 ben.
- **Udgang 1:** Er tilsluttet direkte til FPAA'ens IO1 ben.
- **indgang 5:** Er tilsluttet direkte til FPAA'ens IO3 ben.
- **indgang 4:** Er tilsluttet direkte til FPAA'ens IO4D ben.

- **indgang 3:** Er tilsluttet direkte til FPAA'ens IO4C ben.
- **indgang 2:** Er tilsluttet til FPAA'ens IO4B ben, via en AD8132 operationsforstærker.
- **indgang 1:** Er tilsluttet til FPAA'ens IO4A ben, via en AD8132 operationsforstærker.

Ind- og ud- gange der er forbundet direkte til FPAA'en, skal generelt benyttes i differentiel mode. Stik der er forbundet via en AD8132 kan benyttes i differentiel mode, men kan også benyttes i single-ended mode, hvis den ene indgangsterminal forbindes til ground. Forbindelserne i stikkene er som følger:

- Ben 1 (længst ud mod kanten af printet): Negativ ind- eller ud- gangsterminal.
- Ben 2 (i midten): Ground.
- Ben 3 (ind mod midten af printet): Positiv ind- eller ud- gangsterminal.



Figur 15.10: Eksempel på sensor tilslutning.

Figur 15.10 viser et eksempel på hvordan en ohmsk sensor kan tilsluttes systemet, vha. en wheatstone bro. En udgang fra FPAA'en benyttes til at exciterer sensoren, mens en indgang benyttes til at måle resultatet.

15.2.0.9 Power ud

Dette stik kan benyttes til at spændingsforsyne sensorene, hvis dette er nødvendigt. Ben forbindelserne er som følger:

- Ben 1 (længst ud mod kanten af printet): -5 volt
- Ben 2 (i midten): ground.
- Ben 3 (ind mod midten af printet): +5 volt

15.2.0.10 Jumperen JP1

Denne jumper bestemmer om den ATMega128 processorens interne 2.56 volt reference spænding skal benyttes til A/D konverteren, eller om der skal benyttes en reference spænding fra FPAA'en. Hvis det sidste vælges, bestemmer jumperene JP10 og JP11 hvad denne reference er. Hvis jumperen placeres længst ned mod processoren benyttes den interne reference, hvis den placere længst oppe mod SD/MMC kortet benyttes den eksterne reference. Bemærk! Hvis softwaren sætter processoren op til at benytte den interne reference, må jumperen ikke stå i position "ekstern reference"!

15.2.0.11 Jumperen JP3

Denne jumper skal kun isættes hvis ISP interfaces skal anvende.

15.2.0.12 Jumperen JP10

Denne jumper bestemmer om A/D konverterens eksterne referencespænding (se JP1), skal være FPAA'ens VMR (2 volt), eller skal komme fra en af udgangene på FPAA'en. Sættes jumperen længst oppe mod ind- og ud- gangsstikken vælges VMR som reference. Sættes jumperen længst ned mod strømstikket, vælges FPAA'ens O2P udgang som reference, dette giver mulighed for at sætte en brugerdefineret referencespænding. Vælges denne mulighed skal jumperen JP11 fjernes.

15.2.0.13 Jumperen JP11

Fjern denne jumper hvis FPAA'ens O2P udgang benyttes som referencespænding til A/D konverteren.

Kapitel 16

Afslutning

16.1 Konklusion

Som det kunne læses i foranalysen, har projektet taget en drejning i forhold til den fra starten opstillede problemformulering. Der er istedet for datalogningsudstyr specifikt rettet mod analysen af nakke-skulder-arm problemer, blevet udfærdiget en fleksibel sensorevalueringsplatform.
Ved at anvende en FPAA til analog signaltilpasning er systemet ekstremt fleksibelt mht. sensorvalg og igennem et fleksibelt konfigurationsformat gør den brugere istand til på en fleksibel og hurtig måde at udføre datalogning.
Selvom der i projektet kun i et mindre omfang har været fokuseret på udvikling af brugerflade, er der blevet udviklet brugergrænseflader. Disse består af en VT100 terminal samt et lokalt uiboard.
Systemet anvender SD/MMC kort med et FAT filsystem til lagring af lognings-data. Det er dermed let og hurtigt at overføre data til PC'er for analyse.
Systemet er dog stadig en prototype som kan forbedres, især på følgende punkter:

- Forbedrede støjforhold
- Udskiftning af levelshifterlogik
- Forbedring af UI board
- Mere RAM til mikrocontroller
- Forbedring af brugerflade
- Hastighedsforbedring af FPAA programmering
- Forbedring af filsystem
- Forbedring af tidsstyring

På trods af ovenstående punkter er det dog vurderingen at prototypen kan anvendes til evaluering af forskellige sensortyper.

16.2 Perspektivering

Da de oprindelige mål for projektet ikke er nået, er det naturligvis håbet at det udfærdigede produkt i fremtiden ville kunne benyttes til at komme nærmere målet.
Man kunne håbe at projektet kan hjælpe til at identificere og bibringe en større forståelse, af de måleparametre der er væsentlige i tidlig identifikation af skader opstået, som følge af ensidig gentaget arbejde.
Den udviklede platform er så fleksibel og almen brugbar så at den ikke kun er begrænset til anvendelse for de i rapporten beskrevne formål. Man kunne f.eks. sagtens tænke sig systemet anvendt som en almen datalogger til opsamling af data i produktionsmiljøer.

Litteratur

[Anadigm, 2006a] Anadigm (2006a). Anadigmdesigner2 - user manual.

[Anadigm, 2006b] Anadigm (2006b). Field programmable analog arrays - user manual.

[Analog, 2000] Analog (2000). Chopper stabilization.

[Atmel, 2006a] Atmel (2006a). Atmel atmega128 datasheet.

[Atmel, 2006b] Atmel (2006b). Characterization and calibration of the adc on an avr, application note avr120.

[Atmel, 2006c] Atmel (2006c). A temperature monitoring system with lcd output, application note avr064.

[Edwards, 2006] Edwards, L. A. (2006). Dosfs free fat12/fat16/fat32 filesystem. [Online].

[Group, 2006] Group, N. W. (2006). Common format and mime type for comma-separated values (csv) files (rfc4180).

[Non-gnu, 2006] Non-gnu (2006). Avr libc. [Online].

[Sandisk, 2006] Sandisk (2006). Secure digital card - product manual.

[Stang, 2006] Stang, P. (2006). Procyon avrlib. [Online].

[Wikipedia, 2006] Wikipedia (2006). Secure digital card — wikipedia, the free encyclopedia. [Online; accessed 15-May-2006].

Figurer

Tabeller

Del V

Bilag

Bilag A

Hardware