UNIVERSITETET I OSLO
Fysisk Institutt

Prototyputvikling av digital solsensor for sonderaketter

Martin Sollien

31. mai 2006

Abstract

This thesis deals with prototype development of a digital sun sensor for use in sounding rockets. Different possible principles of operation is reviewed, and also how this particular instrument operates. The design is thoroughly explained, including the on board digital logic.

Basically, this instrument operate by letting light pass through a filter and then a pin-hole. On the image plane behind the pin-hole the location of the light beam is detected with a CMOS line camera. The sun angle can then be determined from the location of the light beam using trigonometry. Some processing is done on board the sounding rocket. To completely reconstruct the rocket's orientation the data has to be post-processed together with data from other orientation instruments. From the sun sensor alone only information about the sun angle, spin frequency, coning frequency and coning angle can be acquired.

The goal of this thesis was to make a sun sensor (reference sensor) as a part of a complete orientation system for use in sounding rockets.

The result is a sun sensor with a 120°field of view with a resolution of 0.06°. The readout speed is 6510 Hz. The cost of making one copy of this instrument is approximately 4500 NOK, excluding work hours.

Sammendrag

Denne oppgaven omhandler prototypeutvikling av en digital solsensor for sonderaketter. Det gåes igjennom noen mulige måleprinsipper og hvordan dette instrumentet opererer. Konstruksjonen av instrumentet blir grundig gjennomgått, inklusive den digitale logikken.

Instrumentet virker ved at lyset passerer gjennom et filter og et pin-hole (knappenålshull). På baksiden registreres det, på et CMOS-linjekamera, hvor lyset treffer. Ut i fra hvor lyset treffer kan man, ved hjelp av trigonometri regne ut hvilken vinkel lyset treffer sensoren. Noe databehandling gjøres i instrumentet. For å rekonstruere rakettens orientering må dataene post-prosesseres sammen med data fra andre orienteringsinstrumenter. Fra solsensoren alene kan man kun få informasjon om solvinklene, spinnfrekvensen, koningsfrekvens og koningsvinkel.

Målet med denne oppgaven var å lage en solsensor (referansesensor) som en del av et komplett orienteringssystem for bruk i sonderaketter.

Resultatet er en solsensor med ca 120°synsfelt med en oppløsning på ca 0.06°. Utlesningshastigheten er på 6510 Hz. Prisen for å lage en kopi er på ca 4500 NOK pluss arbeidstimer.

Forord

Arbeidet med denne oppgaven har vært svært interessant, variert og lærerikt. Jeg har vært med på utviklingen av et instrument fra idé til ferdigstillelse, og testing av instrumentet i rakettkampanje. Oppgaven har blitt utført ved Plasma- og Romfysikkgruppa ved Fysisk Institutt på Universitetet i Oslo.

Den første og største takken går til min hovedveileder Jan Kenneth Bekkeng. Du har hjulpet meg på en veldig god måte gjennom hele oppgaven, og vært en meget god veileder og støtte. Uten en så god veileder og kollega som deg hadde det vært betraktelig vanskeligere å gjennomføre denne oppgaven. I tillegg så takker jeg min andre veileder Jøran Moen. Ikke bare fordi du er en meget god veileder, men også for alt arbeidet du gjør for Plasma- og Romfysikkgruppa. Lars Lyngdal bør også takkes. Du har, med din lange erfaring innen elektronikk, kommet med en rekke gode råd og tips som har fått meg til å se lyset en del ganger. Karl H. Haugholt fra Optikkgruppen på SINTEF må også takkes. Du har guidet meg i riktig retning på et par veldig sentrale temaer.

Det er flere ved forskingsgruppen som bør takkes. Espen Trondsen er en av de som bør nevnes ved navn. Du har vært til mye hjelp, spesielt med tekniske problemer. I tillegg bidrar du veldig mye for at arbeidsmiljøet på denne forskningsgruppa er så bra som det er. I tillegg så takker jeg Ellen Osmundsen og Anette Borg. Det var dere, med deres skrivekurs for masterstudenter, som fikk meg i gang med skrivinga. Dere er også to lyspunkter på gruppen som bidrar mye til det gode miljøet her, både på grunn av det sosiale arbeidet dere gjør, og på grunn av det alltid-gode humøret deres som lett smitter over.

Alle masterstudentene på gruppa må også takkes. Njål Gulbrandsen, Knut Stanley Jacobsen, Andreas Quamme Nilsen, Woiciech Miloch, Xu Cong "Jon" Qui, Yvonne Rinne, Arvind Sharma og Åsmund Skjæveland: Dere har alle bidratt til at det har vært et trivelig, gøy og lærerikt miljø her. I tillegg vil jeg takke Henning Øye. Du har vært en meget hyggelig og kompetent medstudent som det har vært veldig gøy og interessant å studere sammen med.

Verkstedet ved Fysisk Institutt, og spesielt Steinar Skaug Nilsen og Thor-Arne Agnalt, skal også ha en stor takk. Dere er utrolig kompetente, og fan-

tasktiske personer å ha jobbet sammen med.

Jeg takker også Dag Langmyhr. Både på grunn av de lokale veiledningene du har laget for LaTeX (tekstkompilatoren som er brukt for å generere denne oppgaven), men også for ivrig og grundig ha løst og besvart alle mine spørsmål om / problemer med LaTeX.

Til slutt vil jeg takke min samboer Heidi Jorem Olsen. Du er et fantastisk menneske, og har vært veldig forståelsesfull mens arbeidet med masteroppgaven har pågått.

Et par kommentarer om skrivningen, stilen og figurene: Oppgaven er skrevet i LaTeX, og så og si alle figurene (bortsett fra de som omhandler CPLD'en) er også tegnet i LaTeX. Det er helt bevisst brukt "." som desimaltegn i steden for "," i gjennom hele oppgaven, selv om dette ikke er i henhold til Språkrådet sine retningslinjer. Grunnen til dette er blant annet at oppramsing av desimaltall blir meget uoversiktlig med bruk av "," som både desimaltegn og skilletegn. Når det gjelder mengden tekst i oppgaven så er den forsøkt holdt til dameskjørt-prinsippet: Ha den så lang at den dekker de viktige elementene, men så kort at det er interessant.

Innhold

Figurer

Tabeller

Kapittel 1

Innledning

For alle romfartøy er det viktig å kunne bestemme orienteringen i rommet. Satellitter må for eksempel orienteres slik at kommunikasjonsantenner og lignende rettes mot mottager. For å analysere de vitenskapelige dataene fra instrumentene ombord i sonderaketter trenger man informasjon om rakettens orientering i alle tre akser. Den nødvendige orienteringsnøyaktigheten varierer, men for sonderaketter vil en nøyaktighet på 1°- 2°trolig være tilstrekkelig i de fleste tilfeller. Prosjektet "Sounding Rocket Attitude Determination System" (SRADS) tar sikte på å lage et miniatyrisert og rimelig system for orienteringsbestemmelse av sonderaketter. SRADS vil bestå av treghetssensorer (gyroskoper) av MEMS-typen (MicroElectroMechanical System), et treakse-magnetometer og en digital solsensor. Orienteringen bestemmes ikke i sanntid; sensordataene postprosesseres for å rekonstruere rakettens orientering. Rekonstruksjonen av rakettens orientering gjøres ved å fusjonere alle sensorenes måledata sammen med kalibreringsdata og matematiske modeller.

Målsettingen for denne masteroppgaven er å lage et prototypeinstrument av en digital solsensor for bruk i SRADS-prosjektet. Det var ønskelig med et synsfelt på om lag 120°, og på grunn av en høy spinnhastighet på spinnstabiliserte sonderaketter (opptil 6 Hz) må instrumentet arbeide på en høy utlesningshastighet. For å tilfredsstille SRADS krav til nøyaktighet bør solsensoren ha en nøyaktighet på 1°eller bedre. Instrumentet skulle være ferdigstilt slik at det kunne bli testet sommeren 2005 på raketten IMEF/-HP2.

Kapittel 2

Sensorer for orienteringsbestemmelse

Det finnes mange sensortyper man kan bruke til orienteringsbestemmelse, alle med sine fordeler og ulemper. Jeg vil i dette kapittelet gå raskt igjennom de mest brukte sensortypene, og ulike måleprinsipper for solsensorer.

2.1 Måleprinsipper og sensorer

Det finnes i hovedsak to forskjellige klasser av sensorer som benyttes til orienteringsbestemmelse: Treghetssensorer og referansesensorer.

De to hovedtypene av treghetssensorer er akselerometere og gyroskoper, og måler henholdsvis translatorisk akselerasjon og rotasjon i forhold til en treghetsramme. Slike sensorer har som regel meget høy utlesningshastighet, men vil være ustabile over tid. Ustabiliteten skylles det som kalles drifting og bias, og kommer av at sensorene ikke er perfekte. For eksempel er friksjon, ulinearitet og produksjonsfeil kilder til dette. Dyrere sensorer har mindre drift, men de har allikevel ikke null drift.

Referansesensorer er sensorer som detekterer referansepunkter som vi kjenner plassering på / verdien av på forhånd. Med *kjenner* menes at vi har allerede en modell av referansen som er godt innenfor nøyaktigheten vi trenger. For eksempel så har vi mange meget gode modeller av solas posisjon og kan bruke denne som referansepunkt. Vi har også gode modeller av jordas magnetfelt nær jorda, og dette kan også brukes som referanse. Andre mulige referansepunkter kan være jordhorisonten eller stjerner. Denne typen sensor er stabil over tid, men kan ha dårlig oppdateringsfrekvens, og ved bruk av optiske sensorer er ikke referansepunktet alltid synlig.

I tabell 2.1 er det listet opp en del nøkkelinformasjon om eksisterende referansesensorer som finnes. Det er viktig å merke seg at det som er listet opp er nøyaktigheten til sensoren, og det er derfor viktig å bruke en modell som er mer nøyaktig enn sensoren for å utnytte dens fulle potensial.

Sensor	Nøyaktighet (grader)	Masse (kg)	Effekt (W)
Solsensor	0.005 - 3	0.05 - 2	0 - 3
Jord- (horisont-) sensorer:			
Pulsgeneratorer	0.1 - 0.5	0.05 - 1	2
Passive skannere	0.5 - 3	1 - 10	0.5 - 14
Aktive skannere	0.05 - 0.25	3 - 8	7 - 11
Stjernesensor	0.0003 - 0.1	1.5 - 10	1,5 - 20
Magnetometer	0.5 - 5	0.6 - 2	0.5 - 2
GPS (fasemålinger)	0.1	2 - 10	15

Tabell 2.1: *Nøkkelinformasjon over eksisterende orienteringsbestemmelsessensorer for romfartøy. Hentet i fra Helvajian (1999).*

2.1.1 Stjernesensor

Stjernesensorer (Wertz 1978) måler posisjoner til stjerner og sammenligner dette med data fra stjernekataloger. Generelt kan man si at stjernesensorer er de mest nøyaktige sensorene, men de har et par kritiske ulemper. Blant annet så er de dyre, store og har stort strømtrekk. I tillegg så er strølys fra kilder som Sola, Jorda og Månen et stort problem som kan ødelege både måledata og sensoren selv. De egner seg heller ikke til hurtigroterende raketter / satellitter fordi bildet som dannes av stjernehimmelen smøres ut ved rask rotasjon.

2.1.2 Jord- / Horisont-sensor

Horisontsensorer (Wertz 1978) finnes i mange forskjellige utgaver, men prinsippet er det samme for alle - detekter endringer i mottatt stråling. Ved å se på endringer i strålingen på instrumentet kan man se når instrumentet er rettet mot et legeme (Jorda eller måne) eller mot tomt rom. I overgangen mellom et legeme og tomt rom vil man se en kraftig endring i stråling. På denne måten kan man få en referanse.

Denne metoden har et par store problemer. For det første så er ikke overgangen mellom Jorda og tomt rom "hard". Det vil si Jorda har en atmosfære og denne gjør overgangen mer glidene og det blir vanskelig å detektere akkurat hvor Jordas horisont er. Et annet stort problem er at Jorda på ingen måte er en punktkilde for jordsatelitter og sonderaketter, i motsetning til Sola og stjernene. Disse to elementene vanskeliggjør designet, men en måte å minske disse problemene på er å bruke månen som kilde isteden.

2.1.3 Magnetometer

Magnetometere (Wertz 1978) er sensorer som måler magnetfeltet (både retning og styrke). De er som regel meget pålitelige, lette, bruker lite strøm,

og kan jobbe i et stort temperaturområde. For å bestemme orientering fra magnetometerdata må man sammenlikne det målte feltet med et referansefelt beregnet ut i fra en modell. Det betyr at modelleringsfeil i feltmodellen kan gi et signifikant feilbidrag, i tillegg til sensorfeilene, siden feltmodellene ikke inkluderer faktorer som for eksempel strømmer i ionosfæren eller lokale felter i jordas skorpe.

Et element som skaper problemer for satellitter, men ikke sonderaketter, er at Jordas magnetfelt avtar som $1/r^3$ og dette vanskeliggjør nøyaktige og pålitelige målinger et stykke vekk fra Jorda. Generelt kan man si at under 1000 km er det mulig å bruke magnetometer, men alt over dette kan være svært vanskelig, spesielt i nærheten av nordlysovalen.

Magnetometere finnes i mange utgaver som utnytter forskjellige måleprinsipper. Search-coil, "fluxgate", magnetoresistive sensorer og proton magnetometer er fire forskjellige magnetometertyper som blir mye brukt i romapplikasjoner.

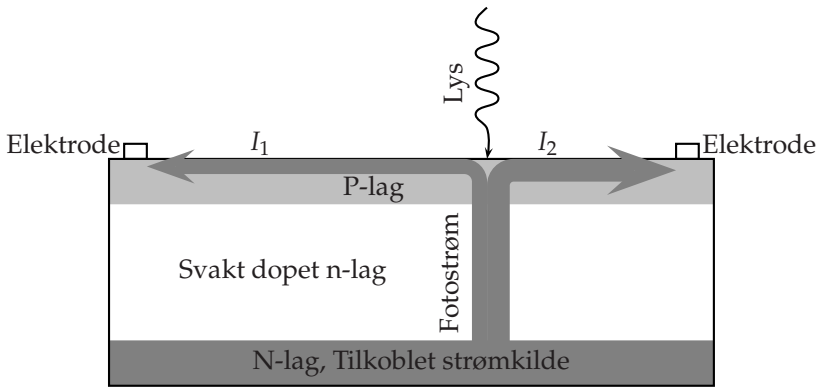
2.1.4 Solsensor

Solsensorer (Wertz 1978) er den mest brukte referansesensoren og det skylles en rekke faktorer. Blant annet så er sola betydelig mer lyssterk enn noe annet objekt nær Jorda, noe som gjør at det er enkelt å avgjøre om det er sola eller noe annet som detekteres. I tillegg så er solas diameter (0.53°) så og si konstant for sonderaketter og jordsatelitter, og den kan oppfattes som en punktkilde. Andre punkter som gjør at det har vært attraktivt med solsensorer er at mange satellitter har soleksperimenter og veldig mange bruker sola som strømkilde, og da er det viktig å rette satellitten inn mot sola. Det finnes en rekke forskjellige typer solsensorer, men vi kan dele de opp i tre hovedgrupper: Analoge, digitale og solnærværdetektorer (engelsk *Sun presence detectors*).

Analoge sensorer gir et signal ut som er en kontinuerlig funksjon av solvinkelen, mens digitale har diskrete, kodede verdier som utgangssignal. Solnærværdektektorer gir utslag når sola er i synsfeltet (engelsk *Field Of View* eller FOV), og brukes til å beskytte andre sensorer, og posisjonering. Denne typen brukes nesten utelukkende i satellitter og er uaktuell å bruke i denne masteroppgaven.

Sensorerbrikker / prinsipper som har vært aktuelle å bruke er:

Position Sensing Detector (PSD) er meget nøyaktige, ekstremt lineære, og veldig raske sensorer. Kort fortalt består detektoren av n-type silisium med et resistivt p-typelag på toppen og et ledende n-lag under (On-Trak Photonics Inc 2004). Når sensoren belyses vil det gå en strøm fra n-laget under til p-laget over og videre til de to elektrodene på endene på topplaget (se figur 2.1). Posisjonen til hvor lyset treffer sensoren er da gitt av følgende



Figur 2.1: *Skisse av PSD, fritt etter On-Trak Photonics Inc (2004). Det vi ser er at når lyset treffer detektoren går det en strøm der lyset treffer, fra n-laget opp i p-laget og ut til elektrodene. Siden p-laget er resistivt vil det gå mer strøm til elektroden som er nærmest (I_2 i dette tilfellet). Da kan vi regne ut hvor lysets treffpunkt er ut i fra strømmen på de to elektrodene.*

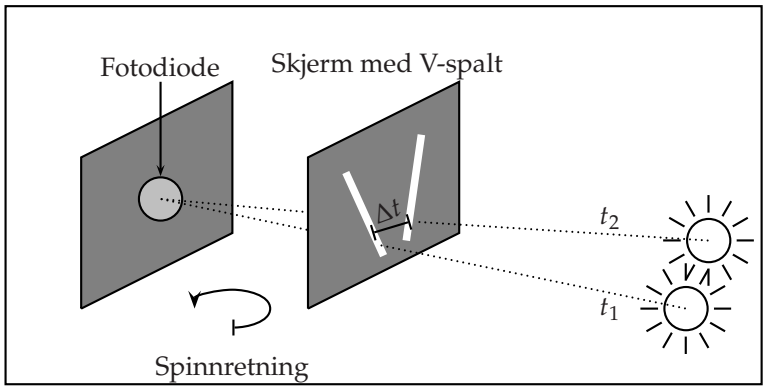
formel:

$$X = \frac{L}{2}\frac{I_1 - I_2}{I_1 + I_2}$$

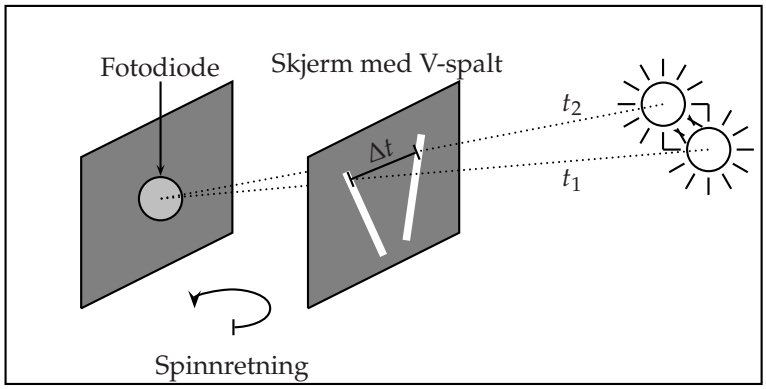
hvor L er lengden på sensoren, og I_2 og I_2 er strømstyrken på elektrode 1 og 2 henholdsvis.

Optikkgruppa på SINTEF Oslo har eksperimentert med denne typen sensor i posisjonsbestemmelse, og har hatt dårlige erfaringer med dette. Grunnen er at denne type detektor har store problemer dersom det skulle komme flere enn én lysstråle. Da vil strømmen kunne gå igjennom brikken på flere steder og det blir umulig å avgjøre hvor lyssenteret er. Dette er grunnen til at denne sensorentypen ikke er blitt brukt i denne masteroppgaven.

V-spaltesensor har vist seg å være god og stabil, men gir ut analoge signaler. Måleprinsippet er som følger (Larsen, Samuelsen & Thoresen 2001): Man har en fotodiode plassert bak en skjerm. I denne skjermen er det to striper som slipper lyset i gjennom, formet som en V. Når sensoren spinner vil sollyset treffe fotodioden to ganger per omdreining, og tiden mellom de to pulsene fra fotodioden vil, sammen med spinnhastigheten, avgjøre solvinkelen. Spinnhastigheten kan finnes ut ifra avstanden mellom to slike pulspar. Se figur 2.2 for en grafisk fremstilling av oppsettet, og figur 2.3 for hvordan spinnhastighet og solvinkel kan finnes ut i fra måledataene.

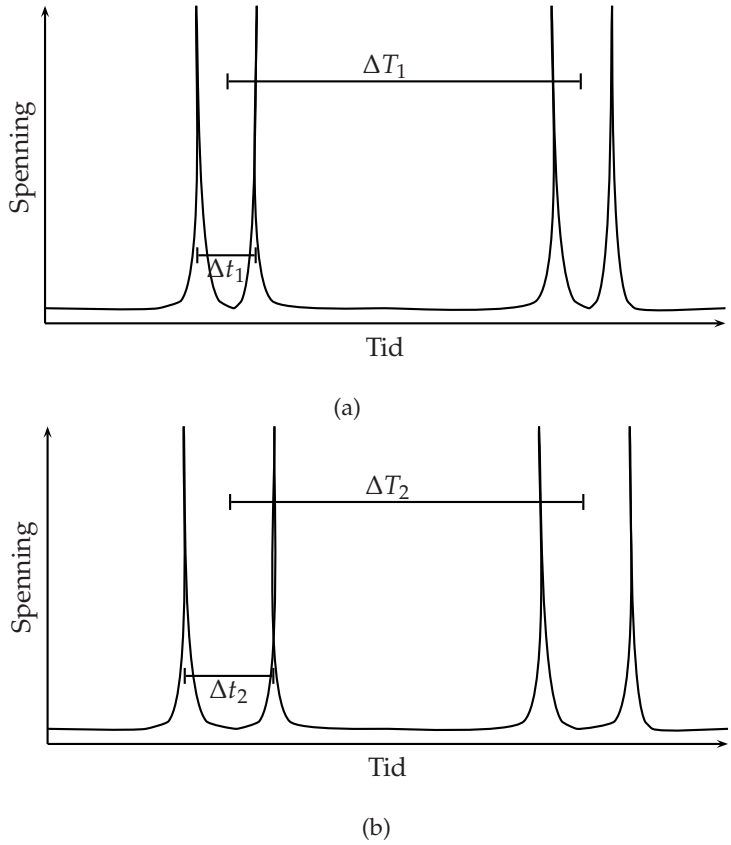


(a)



(b)

Figur 2.2: *Skjematisk fremstilling av V-spaltsensor. Det vi ser er en tidsmessig utvikling når sensoren (raketten) spinner. Sola blir først detektert ved t_1 og så litt seinere ved t_2. Vi ser da at i (a) så er $\Delta t = t_2 - t_1$ liten, mens i (b) så står sola høyere og dermed blir Δt større. Hvis vi vet spinnhastigheten kan vi finne ut vinkelen sola har ut i fra Δt.*



Figur 2.3: *Typisk signal fra V-spaltsensor. Figur (a) svarer til signalet man får hvis raketten er orientert som i figur 2.2a, mens (b) svarer til 2.2b. Vi ser i (a) at tiden, Δt_1, mellom de to solpulsene er kort, mens i (b) er tiden, Δt_2, lengre. Tiden mellom to slike pulspar gir spinnhastigheten, og vi ser at spinnhastigheten i (a), ΔT_1 er den samme som i (b),ΔT_2.*

CMOS / CCD-kamera er de samme som brukes i dagens forbruker-digitalkameraer. Men mens forbrukerkameraer må ha fargegjenkjenning, trenger man kun å se på lysintensiteten (sort-hvitt) for posisjonering. Hovedprinsippet her er at man har en maske foran en slik sensor, og ved hjelp av databehandling gjenkjenner man mønsteret i masken. Grunnen til at det er valgt denne sensortypen, skyldles blant annet at de er lette å jobbe med, finnes tilgjengelig kommersielt, og det er lett å gjøre databehandling av de digitale signalene ombord i en rakett. Hele dette prinsippet blir grundig forklart i seksjon 2.2, så jeg går ikke nærmere inn på det her.

2.2 CMOS bildesensor

En CMOS bildesensor er bygd opp som en rad (en-dimensjonal) eller matrise (to-dimensjonal) av lysfølsomme detektorer. I CMOS-teknologi brukes lysfølsomme detektorer som kalles fotodioder. Veldig enkelt fortalt så "konverterer" en fotodiode lys om til elektriske signaler slik at vi kan gjøre beregninger.

2.2.1 Hvordan en fotodiode virker

En fotodiode kan virke fordi et foton (en energipakke av lys) kan bli absorbert i et atom, og slå løs et elektron (og dermed generere et hull i elektronets opprinnelige plass). Dette avhenger av bølgelengden på lyset og matrialkonstanter i matrialet som lyset treffer. I en halvleder er den kritiske nedre grensen gitt ved

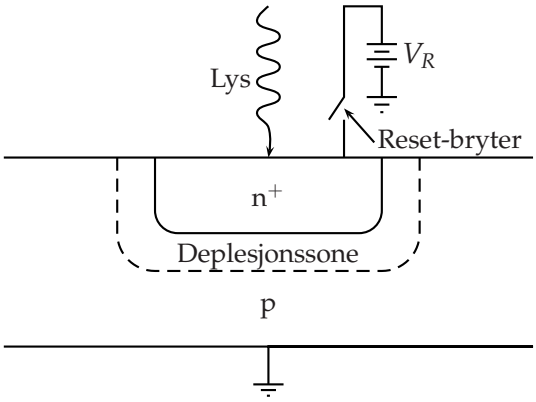
$$E_{foton} = h \cdot \nu = \frac{h \cdot c}{\lambda} \geq E_g$$

der E_{foton} er energien til et foton, h er Plancks konstant, ν er fotonets frekvens, c lysets hastighet, λ fotonets bølgelengde, og E_g båndgapsenergien til halvlederen. Hvis fotonets energi E_{foton} overstiger denne grensen E_g så er antall genererte elektron-hull-par proporsjonalt med den innfallende fotonfluksen. Båndgapsenergien E_g i silisium er på 1.1 eV, og derfor vil lys med bølgelengder kortere enn 1100 nm bli absorbert og generere elektron-hull-par.

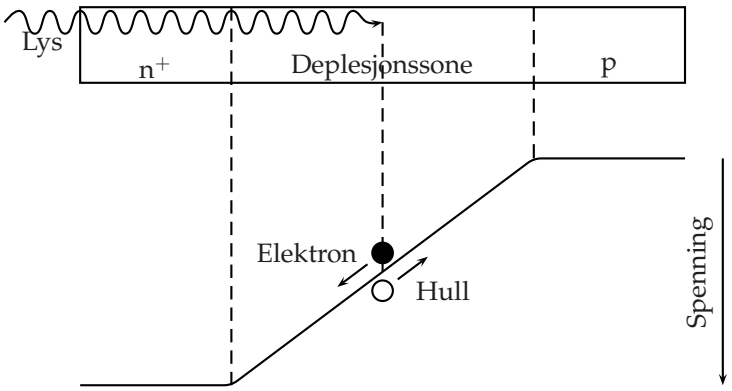
Disse elektronene (eller hullene avhengig av produksjonsprosessen) må telles på en eller annen måte, og det gjøres ved å måle den samlede spenningen de skaper. Se figur 2.4 for en illustrativ fremstilling.

En CMOS-sensor er bygd opp av en matrise eller en rad av slike fotodioder, og man kan ved å se på lysintensiteten på hver fotodiode få et bilde. Det finnes andre teknologier enn fotodioden for bruk i bildesensorer, men alle følger det samme prinsippet med absorbsjon av et foton for å generere et elektron-hull-par.

Det finnes en del feilkilder som kan ødelegge bildet. Blant annet så er ingen piksler eksakt like og man vil få ut forskjellige spenninger fra forskjellige piksler selv om alle har samme belysning (ikke-uniform respons). En annen effekt som er en feilkilde er den såkalte mørkestrømmen. Den kommer av at elektroner og hull oppstår også på grunn av den indre energien sensoren har. Hvis det oppstår et elektron-hull-par i deplesjonssonen vil det være umulig å skille den i fra et par dannet av et foton. Jo høyere temperatur jo større er mørkestrømmen, siden temperatur egentlig er et måltall på den indre energien et system har. Typisk vil mørkestrømmen fordobles for hver 6 - 10°C. En tredje feilkilde er at det kan være døde piksler, og disse vil enten gi ut ingen spenning eller full spenning, avhengig av



(a) Utsnitt av en fotodiode



(b) Energidiagram for en fotodiode

Figur 2.4: *Fotodiode i sperreretningen (Nakamura 2006). I figur (a) ser vi funksjonsfigur av en fotodiode: Et n-dopet område i p-silisium. Når n^+-området blir ladet med en positiv spenning (reset-bryter kortslutter n^+-området til V_R slik at det får V_R som spenning) i forhold til p-området oppstår det en deplesjonssone. Spenningsprofilen sees i figur (b). Her ser vi at hvis et foton blir absorbert i deplesjonssonen og slår løs et elektron fra et hull vil elektronet "falle" til det positivt ladde n^+-området og bli der, mens hullet trekkes til p-området og jord. Etter som tiden går blir mange slike elektron-hull-par skapt og spenningen i n^+-området synker. Etter en viss tid leser vi ut hva spenningen er på n^+-området i forhold til reset-nivået. Da finner vi ut hvor mange elektroner som er akkumulert i løpet av integrasjonstiden (tiden mellom reset og utlesning), og måler dermed fotonfluksen (lysintensiteten). Denne metoden ble først foreslått av Weckler (1967).*

hvordan død pikslene er. I tillegg så finnes det andre feilkilder, og ønsker man å lese mer om disse anbefales Nakamura (2006).

2.2.2 Forskjeller på CMOS og CCD

Både CMOS og CCD (Charge-coupled device) sensorer er i bruk i dag, og de aller fleste sensorer blant begge teknologier bruker fotodioden som sensorelement. Men det er en grunnleggende og viktig teknisk forskjell mellom de to teknologiene. Begge bruker som nevnt fotodioden som akkumulerer lyset og gjør det om til elektroner, men når det gjelder å lese ut denne verdien gjør de to teknologiene det på veldige forskjellige måter. CMOS-sensorer har en liten forsterker som sitter sammen med hver fotodiode, og denne forsterkeren er koblet til en felles analog til digital omformer (engelsk: *Analog to Digital Converter* eller ADC). Det er viktig å merke seg at vanligvis så befinner denne ADC'en seg internt på CMOS-brikken, men sensoren jeg bruker har ikke en slik innebygd. Det har derfor vært nødvendig med en ekstern ADC.

CCD-sensorer gjør det på en helt annen måte: Her er det ingen forsterker ved hver fotodiode, men kun en felles forsterker rett før ADC'en. I tillegg så går signalene fra fotodiodene gjennom de andre fotocellene. Eller med andre ord så leser man først ut ladningen på piksel 1, og så tømmes denne. Deretter så flyttes alle ladningene "ned ett hakk" (piksel 2 går til 1, piksel 3 til 2 og så videre), og piksel 1 leses ut på nytt. Denne prosessen gjentas til alle pikslene er lest ut. En annen måte å se det på er at CMOS-sensorer "dytter" ladningen ned til en felles ADC, mens i en CCD "dras" ladningen ned til en forsterker og videre til ADC. Dette gjør at CMOS-sensorer kan kalles aktiv-piksel-sensor, mens CCD kalles passiv-piksel-sensor (Nakamura 2006). Det finnes riktignok passiv-piksel-CMOS-sensorer (altså CMOS-sensorer uten individuell forsterker til hver fotodiode), men dette gir en meget dårlig sensor, ihvertfall hvis sensoren er av en viss størrelse.

En annen viktig forskjell mellom de to teknologiene er produksjonsprosessen. CMOS-sensorer kan produseres på nær sagt en hvilken som helst silisium prosessfabrikk, siden produksjonsteknikken er den samme som for de fleste andre mikrobrikker som datamaskinprosessorer, minnebrikker og lignende. CCD-produksjonsteknikk er derimot en spesialisert teknikk. Dette kommer blant annet av kravene som må stilles til nøyaktighet og veldig lavt tap i overføringslinjene, siden det ikke er en forsterker ved hver piksel. På grunn av den spesialiserte teknikken er CCD-sensorer som regel en god del dyrere en tilsvarende CMOS-sensor. CCD-sensorene har tradisjonelt vært av høyere kvalitet og vært mindre påvirket av støy enn CMOS-sensorer, men i de siste årene har det kommet store fremskritt på CMOS-fronten som har så å si vasket ut denne forskjellen.

Når det det gjelder strømforbruk så bruker CMOS langt mindre strøm

enn CCD. CMOS kan også klare seg med lavere spenning enn CCD.

Andre viktige forskjeller er X-Y-adresseringen som er mulig med CMOS, og integrerte funksjoner i CMOS-sensorer. X-Y-adressering betyr at man kan adressere en bestemt piksel, og lese ut kun én. Mens på en CCD, hvis man skal ha ut en bestemt piksel, så må man lese ut alle sammen. Angående integrerte funksjoner så kan man lett bygge inn ekstra funksjonalitet i CMOS-sensorer, og få et heldigitalt grensesnitt. Å gjøre det samme i CCD-sensorer er svært vanskelig og dyrt. Det betyr at man som regel må ha en del mer ekstra komponenter for å lese ut av en CCD-sensor enn for å lese ut en CMOS-sensor.

Kapittel 3

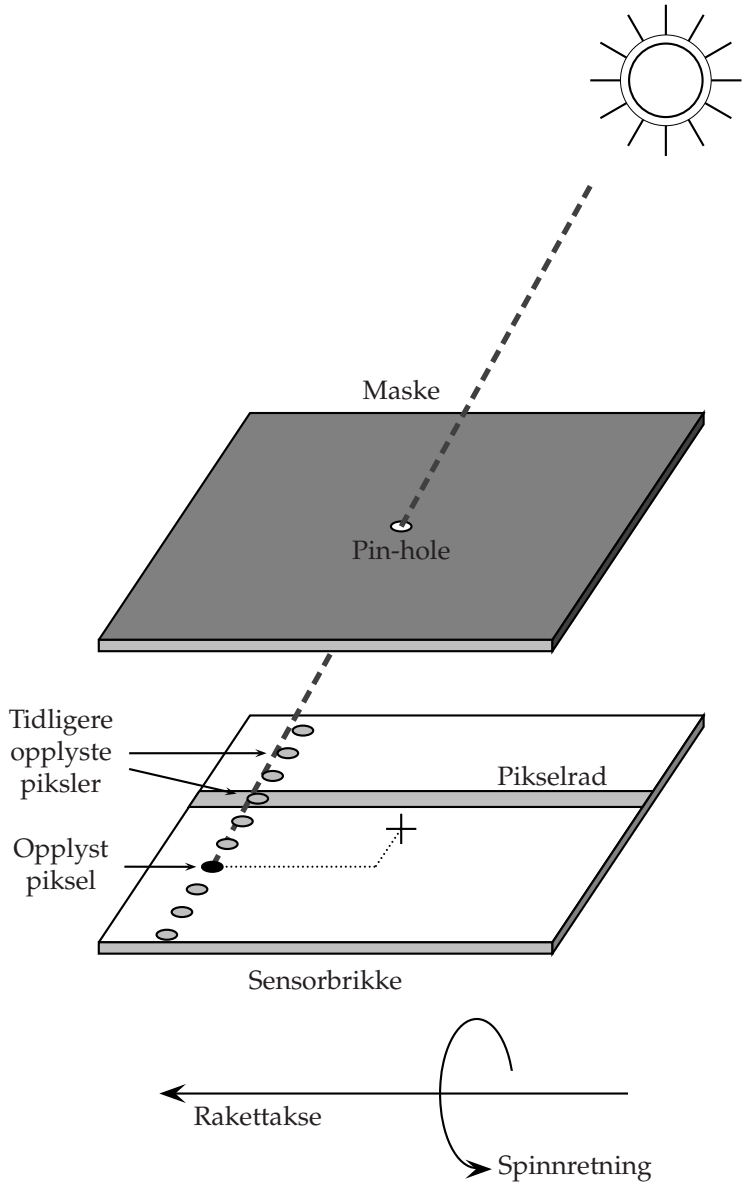
Den nyutviklede solsensorens virkemåte

3.1 Måleprinsipp

For å kunne avgjøre vinkelen til sola kan man bruke måleprinsippet som er vist i figur 3.1. Sola har en åpningsvinkel på 0.53° sett i fra Jorda, og for nær-jord applikasjoner kan vi se på sola som et objekt med uendelig stor avstand. Derfor kan vi se på solstrålene som parallelle stråler. Som vi ser av skissen brukes det en maske med et meget lite sirkulært hull (også kalt "pin-hole") montert foran en sensor (lysfølsom detektor). Dette vil begrense lysfluksen (mengden av innfallende lys), og danne et bilde av solen på detektoren. Detektoren er delt opp i et rutenett av lysfølsome punkter (piksler), og avhengig av solvinkelen (vinkelen mellom rakettens spinnakse og retningsvektoren til sola), vil sola lyse opp ulike piksler på detektoren. Fra detektoren leses det ut hvor stor lysintensiteten er på hvert enkelt piksel. Hvis det er kun ett piksel som er opplyst er det veldig lett å lese av hvor sollyset treffer, men hvis det er flere opplyste piksler kan man ved hjelp av databehandling se hvilke område som er opplyst, og dermed også avgjøre hvor senter av det opplyste området er. Når man vet posisjonen til senter av lyset og avstanden mellom pin-hole og detektorplan, kan man ved bruk av trigonometri regnet ut solvinkelen.

Dette prinsippet kan forenkles ytterligere. Siden denne sensoren skal monteres i en rakett som spinner, trengs ikke en to-dimensjonal detektor. Hvis man isteden har han en én-dimensjonal detektor montert parallelt med rakettaksen vil denne dekke 360°av himmelen ettersom raketten spinner (men synsfeltet langs rakettaksen økes ikke). Dette kan vi se av tidligere nevnte prinsippskisse; Solstrålene treffer alle pikselradene i tur og orden etter som raketten spinner, og vi kan lese av hvor på raden strålene treffer, for så å avgjøre solvinkelen.

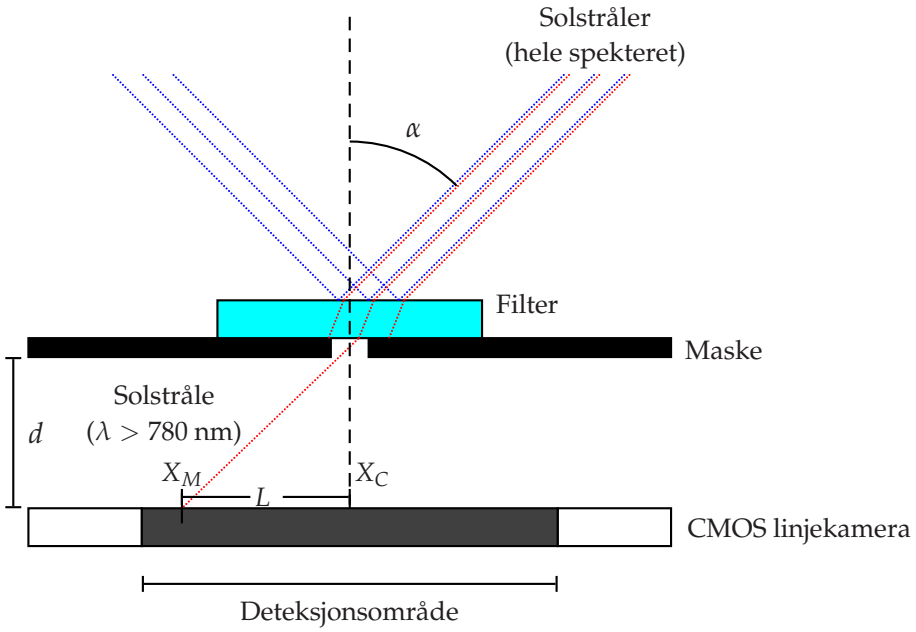
Vi får altså informasjon om solvinkelen bare en gang per omdreining.



Figur 3.1: *Prinsippskisse av en sensor med to-dimensjonalt kamera.*

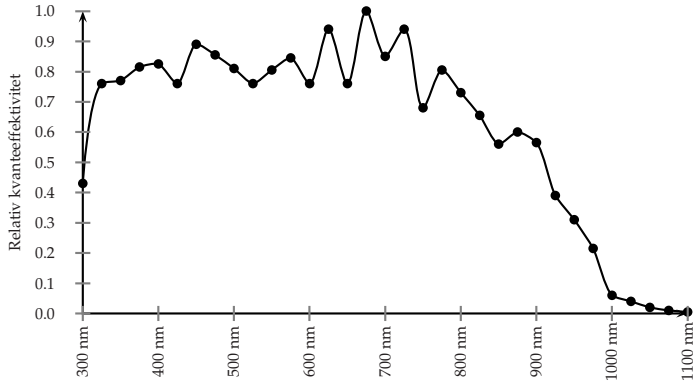
Sensoren skal benyttes sammen med andre orienteringsinstrumenter, som magnetometere og gyroskoper, og disse instrumentene sammen vil kunne gi nøyaktig orienteringsbestemmelse av raketten til et hvert tidspunkt.

Det ble tidlig bestemt at solsensoren skulle være en linseløs sensor. Det var flere grunner til dette, blant annet at det å lage en sensor med linsesystem ville ført til at sensoren ville blitt tyngre og mer omfattende å lage. Isteden ble det valgt å gå for et såkalt pinhole-kamera. En skisse av måleprinsippet som ble brukt er vist i figur 3.2.

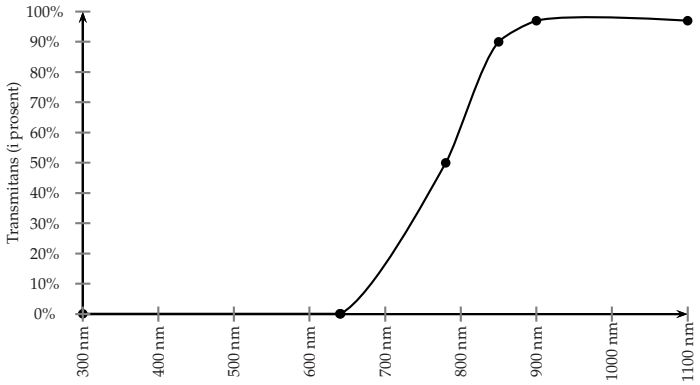


Figur 3.2: *Skjematisk tegning av virkemåten til den digitale solsensoren. Solvinkelen regnes ut i fra følgende formel* $\alpha = \tan^{-1}\left(\frac{L}{d}\right)$*, der* $L = X_M - X_C$.

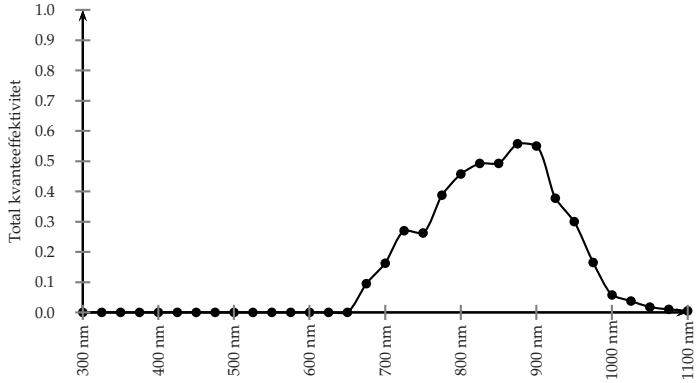
Det vi ser i skissen er parallelle solstråler som faller inn på sensoren. Strålen går gjennom et filter og så en maske med et sirkulært pin-hole slik at strålen som treffer detektoren er de vi ønsker med tanke på både bølgelengde og størrelse / form på strålen. Bølgelengden λ (> 780 nm) på filteret er valgt fordi vi ønsker å unngå problemer med refleksjoner av synlig lys (fra atmosfæren / jorda, månen, selve raketten og eventuelt andre kilder), og kun se på den infrarøde strålinga fra sola. Siden CMOS-sensorer mer eller mindre er transparente for bølger med bølgelengder større enn 1100 nm, så har vi en båndbredde på ca 780 til 1100 nm. I figur 3.3 kan vi se hvordan sensor med filter responderer avhengig av frekvensen på lyset.



(a) Relativ kvanteeffektivitet til CMOS-sensoren SLIS2048. Fritt etter Panavision Imaging (2005)



(b) Filtererkarakterestikk til filteret. Fritt etter Edmund Optics Ltd (2004)



(c) Samlet respons med filter og kvanteeffektivitet til sensor. Fullt utslag er satt til maks kvanteeffektivitet uten filter.

Figur 3.3: *Respons til sensor med filter. Her ser vi hvordan den relative kvanteeffektiviteten til CMOS-sensoren er i figur (a), og filterkarakteristikken til filteret (Edmund Optics Ltd 2004) i figur (b). Den totale responsen kan sees i figur (c)*

Avstanden d mellom detektor og maske bestemmes ut i fra hvor stort synsfelt man ønsker, og størrelsen på detektoren. Som vi kan se i kravene i seksjon 4.1 er størrelsen på synsfeltet ønsket til $\geq 120^\circ$ ($\pm 60^\circ$). Detektoren som ble valgt er SLIS2048 (se seksjon 4.7 for grunner for dette valget). Denne har et deteksjonsområde på 14.3 mm. Ut i fra dette blir avstanden

$$d = \frac{14.3 \text{ mm}}{\tan 60^\circ}$$
$$d = 4.1 \text{ mm}$$

Altså litt over fire millimeter.

Denne avstanden, sammen med med bølgelengden, bestemmer den optimale størrelsen på masken for at det skal bli minst mulig "utsmørning" av lysstrålene. Heald & Marion (1995) angir den optimale maskestørrelsen til

$$D_{\text{optimal}} \approx \sqrt{4\lambda d}$$
$$\approx 113 \text{ µm}$$

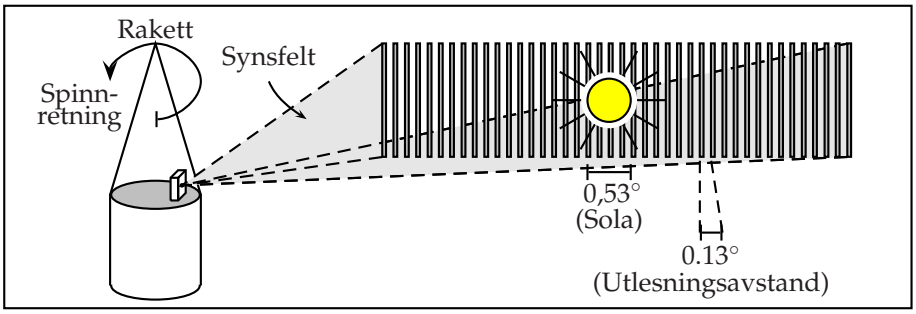
100 µm og 200 µm var de to nærmeste størrelsene som var tilgjengelig fra en av pin-hole-leverandørene, så begge disse ble kjøpt inn for testing. Et alternativ var å spesialbestille et pin-hole fra MINAlabben hos SINTEF for å få akkurat den ønskede størrelsen, men siden beregninga av størrelsen kun er en omtrentlig utregning, og det ikke er veldig kritisk om størrelsen endres litt, så ble det valgt å gå for et kommersielt tilgjengelig pin-hole.

Kameraoppløsningen er $\theta/N = 120^\circ/2048 \approx 0.06^\circ$, der θ er synsfeltet og N er antall piksler. Ut i fra dette ser vi at vi øker oppløsningen til kameraet ved å minske synsfeltet og visa versa.

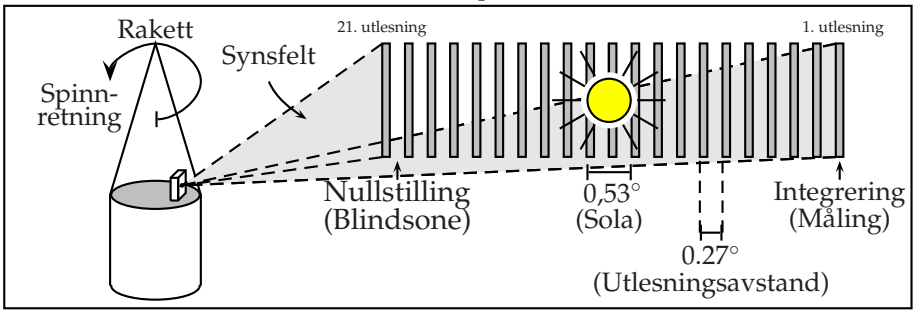
3.2 Hvordan sensoren opererer

Hvordan sensoren fungerer, forklares best ved hjelp av en figur. På figur 3.4 sees prinsippet, men legg merke til at dette er en tidsmessig fremstilling, ikke en geometrisk.

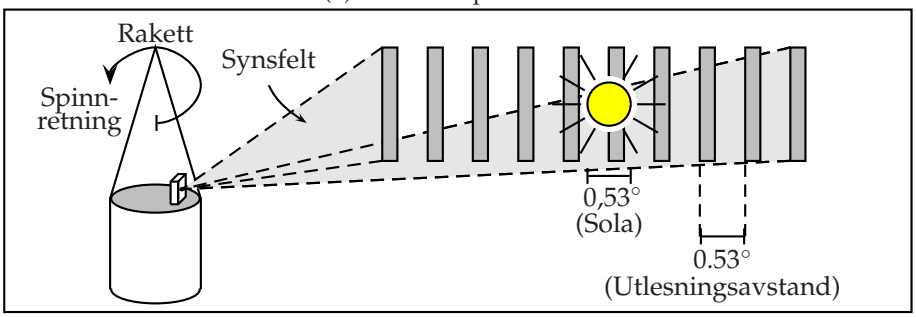
Sensoren i eksempelfiguren sveiper himmelen fra høyre mot venstre, altså mot klokka, (på grunn av rakettrotasjonen), og integrerer opp lyset som treffer den. Denne tida er markert med mørkegrå farge. Så leses informasjonen ut av sensoren, og kretsen nullstilles. På denne tida blir det *ikke* detektert noe, og er de hvite områdene mellom de mørkegrå feltene. Ved tiende utlesning i figur 3.4b ser sensoren sola for første gang, og sola blir detektert tre ganger ved denne spinnhastigheten. Ved lavere spinnhastighet, men samme utlesningshastighet (se figur 3.4a), ser vi at sola blir detektert flere ganger. I figur 3.4c ser vi derimot at raketten spinner så fort at sola detekteres ca bare en gang per omdreining. I figur 3.4d er spinnhastigheten så stor at vi risikerer å ikke se sola i det hele tatt.



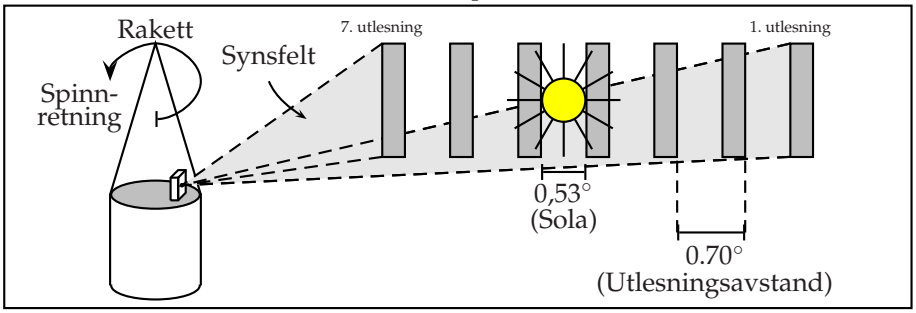
(a) 2.45 runder per sekund



(b) 4.9 runder per sekund



(c) 9.8 runder per sekund



(d) 16.4 runder per sekund

Figur 3.4: *Skisse av hvordan sensoren "ser" ved forskjellige spinnhastigheter. Legg merke til at dette er en tidsmessig fremstilling, ikke en geometrisk. Sensoren detekterer bare langs rakettaksen (opp og ned på skissen), men siden raketten spinner blir det en sidelengs sveiping over tid. I figur (b) spinner raketten med 4.9 runder per sekund, og vi ser at sensoren detekterer sola tre ganger per omdreining. I figur (a) spinner raketten halvparten så fort og vi får dermed dobbelt så mange utlesninger.*

Kapittel 4

Design av solsensoren

Flere faktorer har vært avgjørende for designløsningene som ble valgt. De to mest fremtredende har vært tid og tilgjengelighet. Det var en rimelig kort tidsramme på prosjektet, fra november 2004 til mai 2005. Dette har ført til at det ikke alltid ble valgt den lureste implementasjonen, men den som var mest opplagt, eventuelt den som var raskest å implementere.

Den andre faktoren som har hatt mye å si for de grunnleggende valgene som er gjort er tilgjengelighet. Det er altså valgt teknologi, komponenter og løsninger som var lett tilgjengelig. Spesielt var det problemer med å finne en CMOS-bildesensor som svarte til kravene som ble stilt. På et tidspunkt ble det vurdert om det skulle brukes en CCD-sensor i stedet siden denne teknologien var lettere tilgjengelig. Men til slutt ble det funnet frem til en produsent i USA som kunne levere en CMOS brikke med ønsket spesifikasjon.

Denne solsensoren er en del av et større prosjekt der et av hovedmålene er å lage en lav-kost orienteringsbestemmelsessystem for sonderaketter. Det er derfor forsøkt å finne så billige komponenter som mulig, og gjøre fremstillingen så enkel og billig som mulig. Men som tidligere nevnt var det bare en CMOS-sensor å velge i, og de andre delene er rimelig standardkomponenter. Derfor har momentet med å lage en billig løsning egentlig ikke hatt så mye å si, på grunn av manglende valgmuligheter.

En ting som er viktig å notere seg er forskjellen mellom oppløsning og nøyaktighet (Bentley 1995). Oppløsning er definert som den største endringen i inngangssignalet (solposisjonen i mitt tilfelle) uten at utgangssignalet (data fra hele systemet) endrer seg. Nøyaktighet er definert som den største forskjellen på målt verdi (solsensorens måling av solvinkel) og den sanne verdi (den faktiske solvinkelen). Nøyaktighet inneholder med andre ord alle feilkilder som finnes i systemet; feil i CMOS-sensor, feil plassering av sensorboksen i raketten, og så videre.

4.1 Generelle krav

Ønsket nøyaktighet på sensoren var satt til 1°, mens kravet til synsfelt var satt til 120°. Grunnen til dette som tidligere nevnt, er at denne sensoren er en del av et større system for orienteringsbestemmelse. Det ferdige systemet tar sikte på å ha en nøyaktighet på 1-2°. Solsensoren bør derfor ha en oppløsning som er minst en faktor 10 bedre enn ønsket nøyaktighet, som altså vil tilsi en sensoroppløsning på bedre enn 0.1°. Sensoren som ble brukt hadde 2048 piksler, og dette gir en oppløsning parallelt med rakettaksen som er på $\frac{128}{2048} \approx 0.06^\circ$. I den ferdige sensoren ble synsfeltet ca 128°.

Nøyaktigheten fører til et annet krav til instrumentet: Hastigheten den må jobbe på. Siden raketten spinner, må vi lese av kameraet med en viss hastighet for at feilbidraget ikke skal bli for stort. Det er tatt utgangspunkt i en rakett som har et nominelt spinn på ca 4 rps (runder per sekund) med $\pm 50\%$ variasjon. Gitt at vi har en spinnhastighet på 4 rps og ikke ønsker et feilbidrag på mer enn 10% av ønsket nøyaktighet må instrumentet m leses ut med en frekvens på minst $4\text{ Hz} \times 360^\circ / 0.1^\circ = 14400$ Hz. Dette krever en rimelig stor båndbredde, og man må derfor vurdere båndbreddeforbruket opp mot feilbidraget. For IMEF / HP2 (se kapittel 6.1) ble det valgt å lese ut med 6510 Hz. Denne utlesningshastigheten gir en oppløsning på $\frac{4\text{ Hz} \times 360^\circ}{6510\text{ Hz}} = 0.22^\circ$. Raketten kan ha et spinn på opptil 6 rps og da blir oppløsningen på 0.33°.

Kravet til den fysiske størrelsen til instrumentet ble satt til at det maksimalt kan være 75 mm $\times$55 mm $\times$35 mm. Disse målene er satt på bakgrunn av tidligere erfaringer angående hvor mye plass som kreves for å plass til alle komponentene, og hvor liten den må være for å kunne være med på de fleste rakettene. Det er disse målene det er tatt utgangspunkt i når instrumentet ble designet.

4.2 Operasjonsmiljø og hensyn til dette

Siden dette instrumentet skal sendes opp med en sonderakett er operasjonsmiljøet veldig anderledes enn for "normale" instrumenter. Dette gjør at man må ta en del spesielle hensyn når man designer instrumentet.

En ting som er veldig viktig å tenke på er at det trykket blir betydelig lavere jo høyere opp man kommer og lavt trykk kan skape en del problemer.

Et problem med lavt trykk som kan være veldig ille er det som kalles utgassing (engelsk *outgassing*). Utgassing er fenomenet der faste stoffer i vakuum sakte slipper ut stoffer. Dette er spesielt et problem for optiske instrumenter da gassen fra denne utgassingen kan kondensere på de optiske sensorene og skape store problemer og diffuse bilder. Dette problemet løses i satellitter og langt-levende romprober med å varme opp sensoren

slik at den kondenserte væsken igjen blir til gass og fordamper.

Oppvarming kan være et problem for alle elektroniske applikasjoner. Grunnen til dette er, som tidligere nevnt, at mørkestrømmen øker med temperaturen, og dataene fra sensoren får større og større feil.

Et annet problem med varme er at selve raketten blir kraftig varmet opp når den farer meget hurtig gjennom atmosfæren. På grunn av friksjonen mellom lufta og raketten genereres det mye varme, og dette gjelder spesielt i nesekonen hvor solsensoren er lokalisert. Det er ikke så mye å gjøre med dette problemet annet enn å ta hensyn til det, og passe på at instrumentet opererer riktig ved temperaturene som kan oppstå i nesekonen.

Andre elementer som er viktig å tenke på når man designer instrument for sonderaketter er at raketter kan ha en voldsom akselerasjon, kan til tider vibrere ganske kraftig, og i tillegg så vil den kjappe rotasjonen av raketten kunne skape store krefter. Det er derfor viktig å velge elektriske komponenter som kan operere under slike forhold. I tillegg så må man ha et design som tåler kreftene og som ikke blir deformert. Spesielt viktig er det for optiske instrumenter, da en endring på en tidels millimeter kan bety en del for hvor lysstrålen treffer (en forflyttning av CMOS-sensoren i denne oppgaven på en tidels millimeter vil bety at man får en offset-feil på ca 14 piksler). Det er også viktig å ha en boks som tåler påkjenningene og festeanordninger til raketten som ikke bøyer / gir seg.

Elektromagnetisk stråling er også et problem som man kan oppleve i satellitter / sonderaketter. Jorda har en atmosfære som beskytter mot noe av strålingen fra sola, men kommer man høyt nok opp (ionosfæren og høyere) mister man (noe av) denne beskyttelsen. Det kan derfor være viktig å undersøke hvor mye denne strålingen påvirker instrumentet i høyden det skal operere og hvordan man kan begrense strålingen hvis det er nødvendig.

Det kan også være viktig å ta hensyn til partikkelstrålingen (Solvinden). Solvinden er en kontinuerlig flyt elektroner, protoner og α-partikler fra sola.

Romskrap og partikler er to ting man må være klar over, uten at man kan få gjort så mye med det. Romskrap kan ha hastigheter på opp til 70 km/s og selv knøttsmå partikler kan i denne hastigheten slå hull i massive aluminiumsblokker.

4.3 Mekanisk utforming av instrumentet

Det er tatt utgangspunkt i målene nevnt i seksjon 4.1 når den fysiske utformingen av instrumentet ble gjort. Veggtykkelsen ble satt til 4 mm i alle sidevegger, mens platene på fram- og baksiden har 2 mm. 4 mm er brukt i sideveggene for to hovedgrunner. For det første at det skal være mulig å lage 2.5 mm skruegjenger i veggene. På den måten trengs det ingen and-

re festeanordninger enn de man kan lage i boksen selv. For det andre kan man da lage spor til kretskortene i veggene, og på den måten lage en enkel, men god og sikker festeanordning for disse. Man unngår da å kaste bort plass på kretskortene til festeanordninger, og man får god termisk kontakt til boksen slik at varmeoverføring fra komponentene blir bedre. Sporene er 2 mm dype.

I tillegg så gir 4 mm vegger en stabil og sterk boks, slik at den vil tåle påkjenningene i en rakett. Med så tykke vegger er det også mulig å skru 4 mm skruer inn i boksen og bruke dette som en god og stabil festeanordning til raketten.

Det er brukt 2 mm i bak- og frontplata fordi disse kun skal virke forsterkende / avstivende til sideveggene, og som beskyttelse mot stråling og lys, i tillegg beskyttelsen den gir mot berøring, støv og lignende. Frontplata har riktignok en oppgave til: Å ha en festeanordning for både pin-hole og filter, men dette fører ikke til at hovedplata trenger å bli tykkere. Festeanordningen som er valgt er å montere en plate over frontplata. I begge disse platene er det frest ut sirkulære rom for både filter og pin-hole.

For å maksimere arealet det er mulig å plassere elektronikkomponenter på er det valgt å ha tre kretskort i denne boksen.

En tegning av sammensetning av boksen vises i figur 4.1, mens alle tegningene av boksen er i appendiks C.

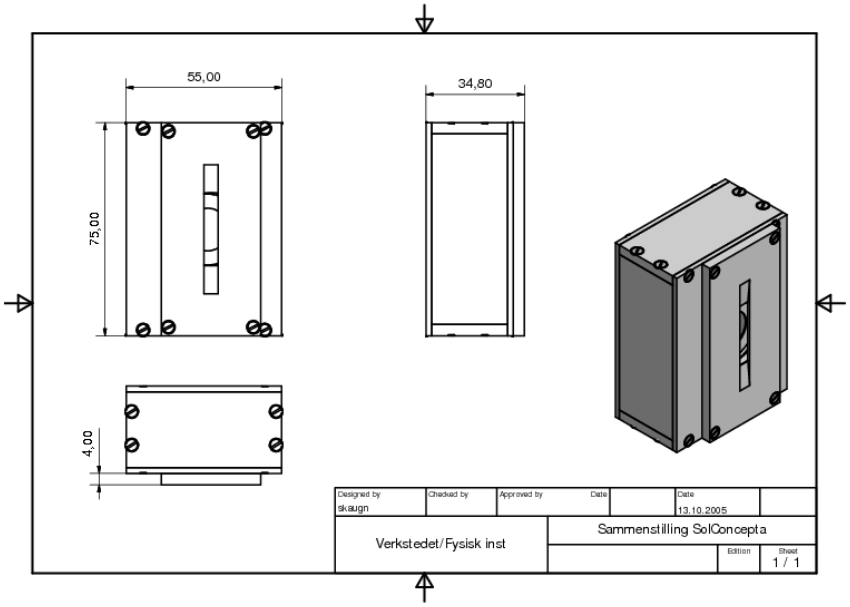
4.4 Elektrisk utforming av instrumentet

Kretskortene som er brukt er av type tolags *printed circuit board* (PCB). Tolagskort er valg først og fremst på grunn av at det er veldig enkelt å jobbe med under prototypeutvikling. Å feilsøke på multilagskort kan være svært vanskelig, og dette er hovedgrunnen til at det ikke er valgt. Prismessig så ligger multilagskort noe høyere enn tolagskort, men denne forskjellen er såpass liten at kostnadene ikke har vært det styrende her. Ved å bruke multilagskort kunne det blitt hele jordplan i instrumentet, og dette ville ha redusert støyen. Det er i steden lagt ned en del arbeid i komponent- og baneplasseringer for å minste støyen.

Når det gjelder komponenter så er det hovedsakelig brukt overflatemonterte komponenter (engelsk *surface mounted devices* eller SMD). Disse er plassbesparende å jobbe med i forhold til hullmonterte komponenter, i tillegg til at de som regel har en bedre høyfrekvenskarateristikk.

4.4.1 Jord, jordplan og elektrisk støy

Elektrisk jord er et spenningsnivå som vi definerer som 0 V. Dette spenningsnivået er referansenivået for alle deler av systemet. Siden jord er referanse for alle andre deler av systemet er det veldig viktig at spenningsni-



Figur 4.1: *Utforming av boks. Her sees hvordan den sammensatte boksen ser ut med mål.*

vået for jord holdes mest mulig fast og er likt over alt i systemet, slik at de enkelte komponentene er "enige" om hva som er 0 V. Spesielt viktig er det for de analoge komponentene å ha en støysvak jord. Man velger derfor ofte å ha et separat jordplan som definerer hva jord er.

For å ha dette jordplanet mest mulig "rent", altså fritt for støy, er det viktig å tenke på hvordan strømmene i systemet går når man designer. Komponentene som støyer mest er de digitale kretsene. Enkelte mener derfor at man skal splitte jordplanet opp i en analog og en digital del. Dette fører til at de digitale kretsene forstyrrer de analoge kretsene mye mindre. Denne løsningen gjør imidlertid at man ikke kan legge baner over grensen på denne splitten siden dette vil føre til at returstrømmene kan få en meget lang returvei og skape støy (som spalten var ment til for å forhindre).

Problemet med elektrisk støy er i all hovedsak digital elektronikk som støyer for analog. Hvis man derfor sørger for at returstrømmene *fra* den enkelte brikke kan gå rett under signalstrømene *til* brikken, så vil returstrømmen gå her, siden dette er minste motstands vei (lavest induktans) for høyfrekvente signaler (for lavfrekvente signaler er minste motstands vei den med lavest ohmsk motstand). Derfor mener Ott (2001) at man ikke bør splitte opp jordplanet i en digital og en analog bit. Man bør heller lage en digital og analog del på kortet *uten* å ha en splitt i jordplanet. Dette er

det viktig å tenke på når man gjør komponentplasseringen på kortet, slik at man plasserer analoge brikker på den ene siden og digitale på den andre. Til slutt passer man på å legge ledningsbanene slik at de også er adskilt.

I dette instrumentet er det valgt å bruke to-lagskort. Dette gjør at det ikke er mulig å ha et separat jordplan fordi det er nødvendig å ha signalbaner både på undersiden og oversiden av kretskortet. Dette gjør at det blir spalter i jordplanet. Hvis man ikke passer på, kan man lage en slik spalt som gjør at returstrømmer må gå store omveier. Dette vil gi både økt støy og økt utstråling.

4.4.2 Avkobling og elektrisk støy

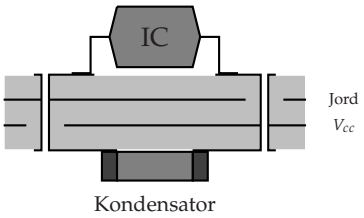
Hensikten med avkobling (kondensatorer) er primært å sikre en feilfri operasjon, sekundert å minske strålingen. Digitale brikker har en pulsaktivitet ("av-på-av-på") som medfører transiente strømstøt på minst to forskjellige måter. Det ene er et kortslutningstøt mellom V_{cc} og jord igjennom den digitale brikken (et fint lite eksempel på dette er en CMOS NOT-gate: Den vil kortslutte når inngangssignalet bytter verdi). Det andre strømstøtet er når den digitale brikken endrer utgangsverdi: Da må lasten drives til et nytt spenningsnivå. Begge disse strømstøtene trekker en del strøm fra strømforsyningen. Strømforsyningens spenning varierer avhengig av strømtrekket, og dermed vil disse strømstøtene føre til to ting: Transiente spenningsfall i V_{cc}, og generering av støy (på jord).

Avkobling bøter på dette problemet ved å virke som en buffer. Ladningen som de digitalene brikken sluker i pulsaktiviteten kan tas fra avkoblingen, mens avkoblingene selv trekker ladning mye "roligere" fra strømforsyningen. Avkoblingen fordeler med andre ord strømstøtene utover i tid, slik at stabiliteten til strømforsyningens spenning økes. Eller med andre ord: Avkoblinger fylles langsomt, og tømmes raskt. Transientstrømmene fjernes altså ikke, men de styres vekk fra følsomme områder.

Man skiller mellom to typer avkoblinger; høyfrekvent (HF) og lavfrekvent (LF). HF-avkobling er for problemet beskrevet tidligere i denne seksjonen, nemlig pulsaktiviteten. Kondensatorverdiene for HF-avkobling varierer noe, men en ofte brukt verdi er 100 nF. LF-avkoblinger benyttes for å stabilisere de lavfrekvente spenningsvariasjonene fra strømforsyning eller andre deler. Verdien på LF-kondensatorene ligger normalt mellom 1 og 100 µF. Disse verdiene er på ingen måte eksakte, og man må regne med å eksperimentere seg fram til hva som er riktig for hvert enkelt design.

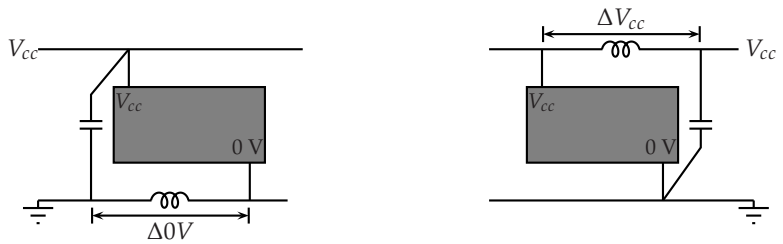
Alle elektriske komponenter som skaper transientstrømmer trenger avkobling, analoge så vel som digitale. Dette inkluderer alle digitale kretser, og de fleste analoge. Plasseringen av disse HF-kondensatorene er veldig viktig for funksjonaliteten til hele systemet. For å bevare mest mulig "ren" jord og stabil spenning fra strømforsyningen, plasseres kondensatorene så nær komponentene de skal avkoble som mulig. De skal helst plasseres mel-

lom strømforsyning / jord og komponenten, men det også mulig å la kondensatoren sitte på baksiden av kortet og la via'en (hullet gjennom kortet) koble til jord og V_{cc} (se figur 4.2).



Figur 4.2: *Kondensator på motsatt side av en IC (Integrated Circuit eller integrert krets). Vi ser at kondensatoren ligger i nærheten av IC'en, men at tilkobling til både jord og V_{cc} er "før" kondensatoren. Dette er ikke ideelt, men andre grunner (som for eksempel ledningsbaner ut fra IC'en, plassproblemer og lignende) kan gjøre at plasseringen av kondensator er best slik som vist.*

Det er også viktig hvordan kondensatoren plasseres ved kretsen. Målet er å ha en renest mulig jord, og en mest mulig stabil spenning fra strømforsyningen. Men når man må velge én av disse, så er det ren jord som normalt blir sett på det viktigste. Det er derfor viktig å plassere avkoblingen så nært 0V-tilkoblingen til kretsen som mulig (se figur 4.3).



(a) Feil plassering (b) Korrekt plassering

Figur 4.3: *Plassering av avkobling ved IC, fritt etter Grødal (1997). På grunn av at alle ledere har en viss induktans, vil det ved transientstrømmer oppstå en viss spenningsforskjell i lederne. Det er derfor viktig å ha disse lederne så korte som mulig. I figur (a) er kondensatoren er plassert nærmest V_{cc} tilkoblingen på brikken. Her blir spenningsvariasjonen på V_{cc} "null", mens det oppstår støy / spenningsvariasjon på jord. I figur (b) er kondensatoren plasser nærmest jordtilkoblingen på brikken. Her kommer støyen fra transientstrømmene på V_{cc}, mens jord unngår dette.*

LF-kondensatorene brukt i dette designet er av elektrolytt typen, nærmere bestemt tantal. Elektrolyttkondensatorer har meget gode lavfrekven-

segenskaper. Når det gjelder HF-kondensatorene så er det brukt keramiske, siden disse har meget gode høyfrekvensegenskaper.

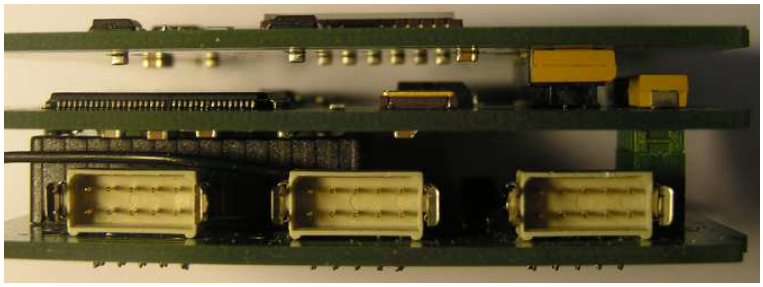
4.4.3 Blokkskjema over instrumentet

Instrumentets virkemåte beskrives i denne seksjonen. Et blokkskjema over instrumentet er vist i figur 4.5.

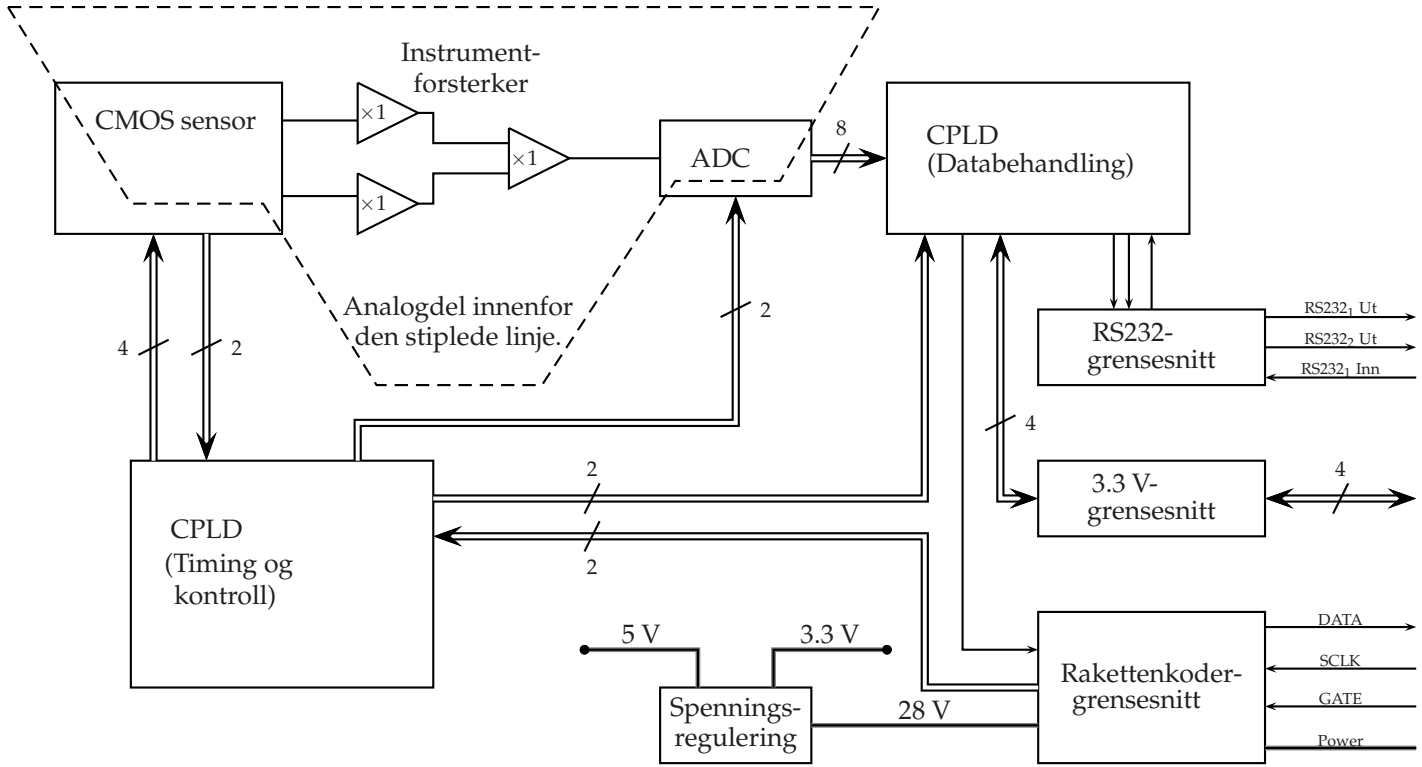
Hele instrumentet kontrolleres av en CPLD (Complex Programmable Logic Device). Den tar seg av alle timing og kontrollsignal, i tillegg til å gjøre databehandling av signalene man får fra CMOS-sensoren.

Hele prosessen begynner med at et lyssignal integreres på CMOS-sensoren. Når dette signalet er ferdig integrert sendes verdien av de 2048 pikslene analogt og serielt ut. Dette signalet går igjennom en instrumentforsterker før det sendes til en ADC og videre til CPLD'en for databehandling. I CPLD'en beregnes hvor senterpunktet for lyskilden er (se seksjon 4.8.3 for algoritmen). Det utregnede solsenteret sendes så til RS232-grensesnittet og rakettenkoderen når GATE signalet fra enkoder tilsier at det skal sendes. 3.3 V-grensesnittet er først og fremst for programmering av CPLD'en, men det er også mulig å bruke dette til feilsøking og lignende.

Ren fysisk er det tre kretskort montert på hverandre med kort-til-kort-kontakter mellom. Disse tre kretskortene har har hver sin oppgave: Strømforsyning og grensesnitt, databehandling, og sensorkort. Et bilde av de faktiske kortene sammensatt er vist i figur 4.4. Det øverste kortet er sensorkortet, i midten databehandlingskortet, og nederst strømforsynings- og grensesnittkort.



Figur 4.4: *Bilde av kretskortene sett fra siden.*

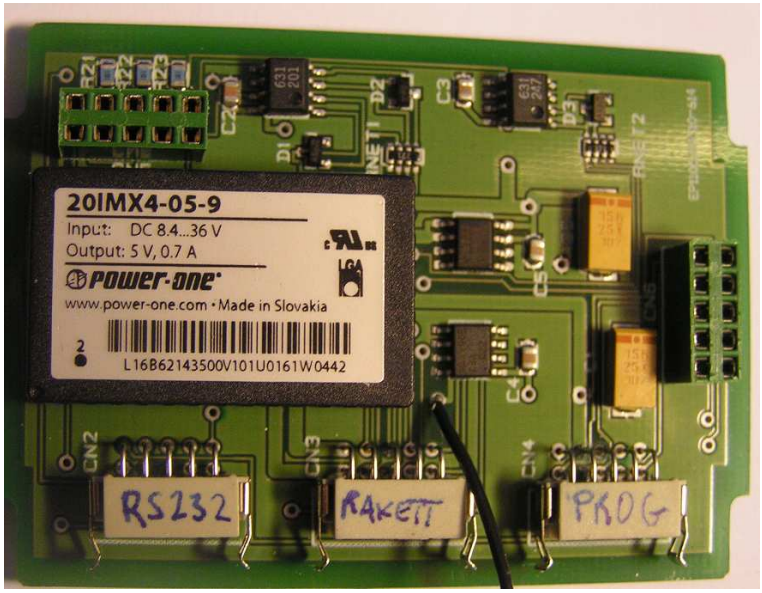


Figur 4.5: *Blokkskjema over instrumentet. I figuren ser det ut som det er to forskjellige CPLD'er. Det er det* ikke*, skjemaet er kun lagd slik for å lettere kunne skille mellom kontroll / timinglinjer og datalinjer. Grunnen til at det er to spenningsfølgere etter CMOS-sensoren er at sensoren har et ekstra sett med piksler. Dette ekstra settet er skygget for slik at man kan lese ut mørkestrømmen ut av ekstra settet, og dermed minske mørkestrømsfeilen.*

4.5 Strømforsynings- og grensesnittkort

Dette kretskortet sørger for spenningsregulering og grensesnittet mot andre instrument. Det har en blanding av hullmonterte og overflatemonterte komponenter. Alle komponentene er normalt montert på oversiden av kortet, men hvis kontaktene til omverden skal monteres slik at koblingene blir på baksiden av instrumentet, blir disse kontaktene på baksiden av kortet.

Skjemategning og PCB-utlegg kan sees i appendiks B. Et bilde av kortet er vist i figur 4.6.



Figur 4.6: *Bilde av strømforsynings- og grensesnitt kortet.*

4.5.1 Grensesnitt mot rakettenkoder

Dette instrumentet har tre forskjellige grensesnitt mot eksterne instrumenter: RS232, 3.3V digitale linjer, og grensesnittet mot rakettenkoder.

Rakettenkoder har følgende signaler til instrumentene:

SCLK Systemklokke. 833 kHz med 50% dutycycle.

GATE Kontrollsignal for utsending av data. Høyt signal når data skal sendes fra instrument til enkoder.

MINF Minor Frame. Puls som kan brukes for å synkronisere instrumentet mot enkoders rammerate (1085 rammer per sekund)

MAJF Major Frame. Puls som kan brukes for å synkronisere instrumentet mot enkoders formatrate (16.95 format per sekund).

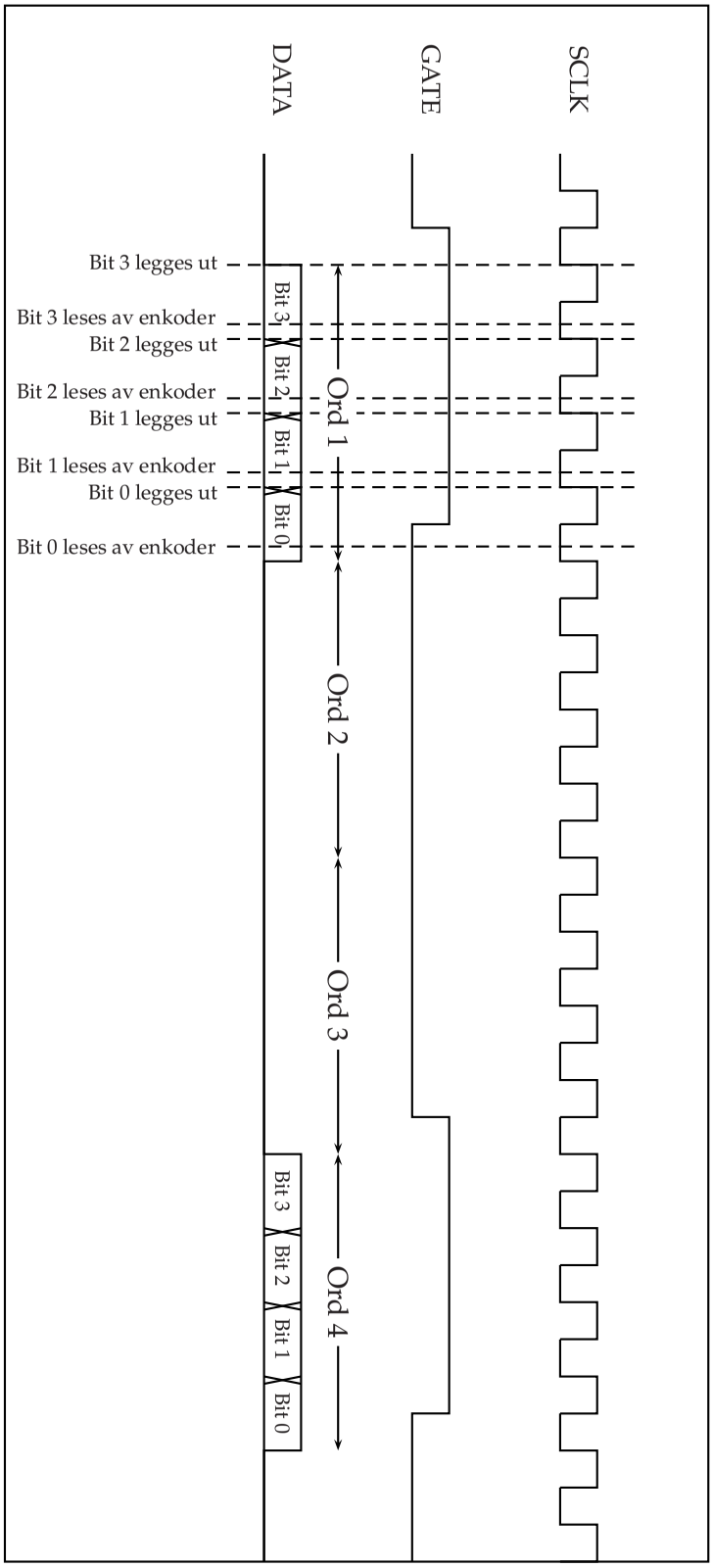
DATA Datalinje for sending av data fra instrumentet til enkoder.

Strømforsyningen er på 28V.

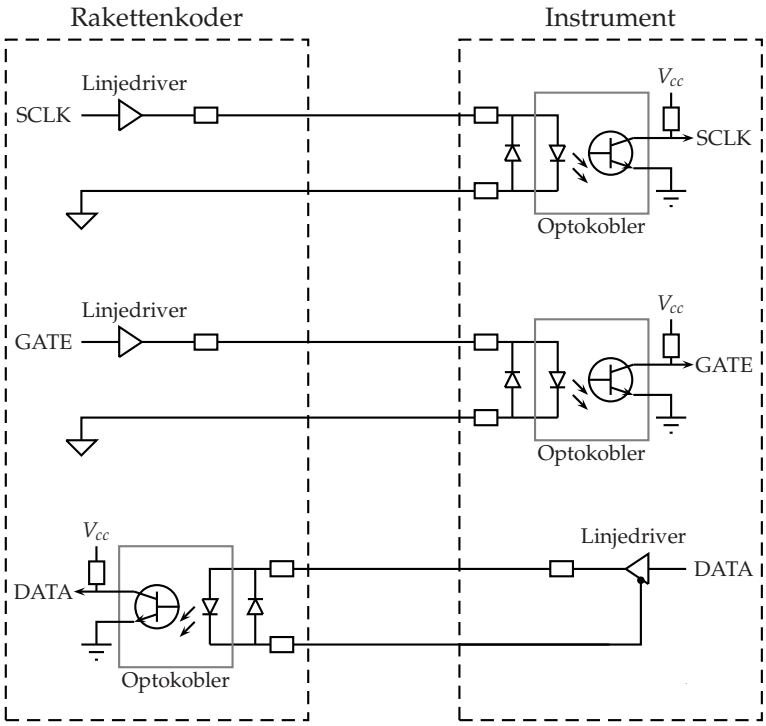
I dette instrumentet brukes kun signallinjene SCLK, GATE og DATA. Et timingdiagram for disse tre signalene kan sees i figur 4.7. I diagrammet er det kun brukt 4 bit, i mot 8 som det egentlig er. Dette er kun gjort for å få plass til alt i diagrammet og gjøre det mer oversiktlig.

Det vi ser i timingdiagrammet er at GATE-signalet endrer fra lav til høy på fallende flanke på SCLK. Deretter ved første stigende flanke på SCLK legges det mest signifikante bit'et (engelsk *most signifikant bit* eller MSB) ut på DATA-linja. Dette leses så av enkoder litt før neste stigende flanke på SCLK. Så legges neste bit ut. Dette gjentar seg til minst signifikante bit (engelsk *least signifikant bit* eller LSB). Etter at LSB er lagt ut settes GATE lav på første fallende flanke på SCLK, og LSB leses av enkoder etter at GATE har gått lav.

Det elektriske grensesnittet mellom instrumentet og rakettenkoderen er vist i figur 4.8. Det er brukt samme elektriske grensesnitt som i E-felt eksperimentet til Bekkeng (2002). Grunnen til at det brukes optokoblere er for å få et elektrisk skille mellom instrumentet og rakettenkoder. Dette bryter potensielle jordsløyfer, slik at støy på felles jord elimineres. I tillegg så sørger optokoblerne for at instrumentet og rakettenkoder tåler potensialforskjell på jordplanet på mange volt. Størrelsene på motstandene (instrumentsiden) er satt slik at strømmen gjennom optokobleren er optimal.



Figur 4.7: *Eksempeltimingdiagram for grensesnitt. Instrumentet legger ut MSB på DATA-linjen ved første positive flanke på SCLK etter at GATE-signalet har gått høyt. Deretter skiftes hvert bit ut på linjen etter som SCLK går. Så ligger GATE lav i to ord før instrumentet igjen skal sende. MERK: Det er kun brukt fire bit i dette diagrammet mot åtte som det egentlig er. Dette er gjort for å gjøre diagrammet mer oversiktlig.*



Figur 4.8: *Det elektriske grensesnittet mellom instrument og rakettenkoder.*

4.5.2 Grensesnittelektronikk

For grensesnittelektronikken er det brukt optokoblerne HP0631 fra HP. Disse optokoblerne har åpen-kollektor utgang, og det er derfor nødvendig med pull-up motstander (Hewlett Packard 1996). Drivspenningen for kretsen er 5 V, men siden CPLD'en skal maks ha 4 V på inngangene er pull-up-motstanden koblet til 3.3 V. Erfaringer gjort i hovedoppgaven til Bekkeng (2002) tilsier at det bør brukes 220 Ω på både inngangs- og returlinjene, da dette gjør at den lysemitterende dioden (engelsk *Light Emitting Diode* eller LED) opererer i sitt gunstigste område. Det er også satt på en signaldiode på optokoblerens innganger for å forhindre at feilkoblinger ødelegger LED'en i optokobleren. Det er viktig å merke seg at optokobleren inverterer signalet.

For sending av data er det brukt MAX490, en 8-pins RS422 krets. Denne har mulighet både til å sende og motta (Maxim Integrated Products 2003), men kun sendedelen skal brukes. Ingen eksterne komponenter er nødvendig, bortsett fra avkobling.

4.5.3 Grensesnittkontakter

Grensesnittet fra instrumentet til omverdenen er tre 2×5 Micronector 200 kontakter. Disse kontaktene tåler strømmer på 1.5 A, 120 V DC (Direct Current). De kan operere i -55° C til $+125^\circ$ C, og akselerasjon på 50 g (490 $\mathrm{m/s^2}$). De har i tillegg en låsemekanisme som holder kabelen som blir tilkoblet fast. Mekanisk så er det oppgitt at kontaktene tåler 500 operasjoner, og låsemekanismen tåler minst 10 N (ITW McMurdo Connectors 2004). Hvis vi antar at raketten har en maks akselerasjon på 25 g (245 $\mathrm{m/s^2}$) betyr dette at massen av kablene (inkludert kontakten selv) maksimalt kan være

$$\begin{aligned} m &= \frac{F}{a} \\ &= \frac{10\ \mathrm{N}}{245\ \mathrm{m/s^2}} \\ &\approx 41\ \mathrm{g} \end{aligned}$$

Dette er nok hovedankepunktet mot å bruke denne typen kontakter. DSUB-kontakter tåler betydelig mer siden disse har en låsemekanisme der man skrur fast kabelen, men er til gjengjeld en del større. Ved å bruke RTV (room temperature vulcanization, en type festemiddel) eller avlaste kontakten på annen måte så vil koblingene allikevel holde. Det må nevnes at det ikke er spesifisert i databladet i hvilken retning kontaktene tåler 10 N. Mest sannsynlig gjelder dette kun i utdragningsretningen (og ikke i alle retninger som det er regnet med her) , og ved å montere kontaktene slik at utdragningsretningen er vinkelrett med rakettaksen vil kontaktene tåle betydelig større laster. Erfaring etter noe bruk med disse kontaktene tilsier også at

det er veldig variasjon om kontakten er ny, eller om den har blitt montert noen ganger. Ved førstegangsmontering er det nærmest umulig å få løsnet kontakten igjen, men bare etter et noen få titalls monteringer er styrken svekket betraktelig.

Disse kontaktene er plassert på kretskortet slik at det er mulig å ta ledningene ut på den ene langsiden eller gjennom bakplaten av boksen. Det er viktig å merke seg at ved å ta kontaktene gjennom bakplaten av boksen så vil kontaktene stikke litt ut av boksen. I tillegg så vil nummereringen som er vist i tegningene i appendiks B på dette kortet endres (pinne 1 og 2 bytter plass, 3 og 4 og så videre).

Det er tre kontakter ut fra instrumentet med ti pinner hver. Den ene er for RS232-grensesnittet, nummer to for rakettgrensesnittet og den tredje er en kombinert programmering- og 3.3 V-grensesnitt.

4.5.4 Spenningsregulering

Til dette designet trengs det to spenninger: 5 V og 3.3 V. Fra raketten er kun 28 V tilgjengelig. Derfor er det behov for intern spenningsregulering. For å få spenningen ned fra 28 V til 5 V er det benyttet en svitsjregulator, mens for å få spenning fra 5 V til 3.3 V er det benyttet en lineærregulator. Svitsjregulatorer har en mye bedre effektivitet enn lineærregulatorer og det skapes derfor mye mindre varme i en svitsjregulator, i tillegg til at strømforsyningen belastes mindre. Ankepunktet mot bruk av svitsjregulatorer er at de skaper en liten rippelspenning på utgangen, men dette kan bøtes på ved bruk av kondensatorer ved både regulatoren og lasten. Lineærregulatorers store fordel er at de har lav rippelspenning.

Et kjapt overslag viser at kretsen maksimal bruker 3 W: CMOS-sensoren bruker 1.15 W ved 120 MHz klokke (og det brukes kun 40 MHz i dette designet), CPLD bruker under 0.2 W ved 40 MHz. Disse er de klart største forbrukerne, så vi kan anta at det tilsammen brukes det dobbelte av dette, altså 2.5 W. For å være på den sikre siden er det valgt en svitsjregulator som kan levere 3.5 W. Svitsjregulatoren er fra Power-one og har serienummer 20IMX4-05-9. Den har output på 5 V, 0.7 A og tåler innspenning fra 8.4 V til 36 V (Power-one 1999). Rippelspenningen peak-to-peak ut er maksimalt på 80 mV. Man trenger ingen eksterne komponenter for denne svitsjreguleringen, men om man ønsker å minske rippelspenningen så er det anbefalt å sette ha en 4.7 µF keramisk kondensator rett ved utgangen. Det er det ikke gjort i dette designet.

For å få spenningen ned til 3.3 V er det brukt TPS76733QD, en 8 pins SOIC (Small-Outline Integrated Circuit) lineærregulator. Denne krever en ekstern kondensator mellom utgang og jord på minst 10 µF (Texas Instruments 2002). I tillegg er det anbefalt, men ikke påkrevd, med en 100 nF keramisk kondensator mellom inngang og jord. Det er brukt en 15 µF tantalkondensator mellom utgang og jord, og en 100 nF mellom inngang og jord som

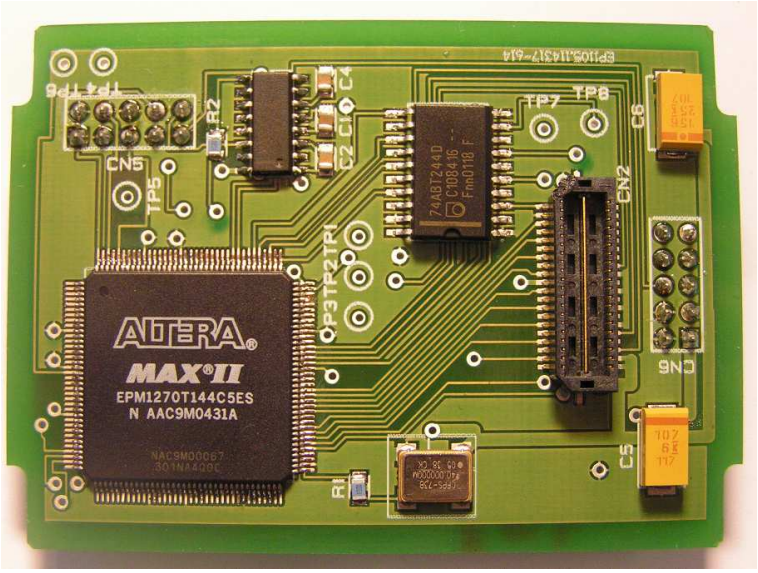
anbefalt.

4.5.5 Kort-til-kort-kontakter

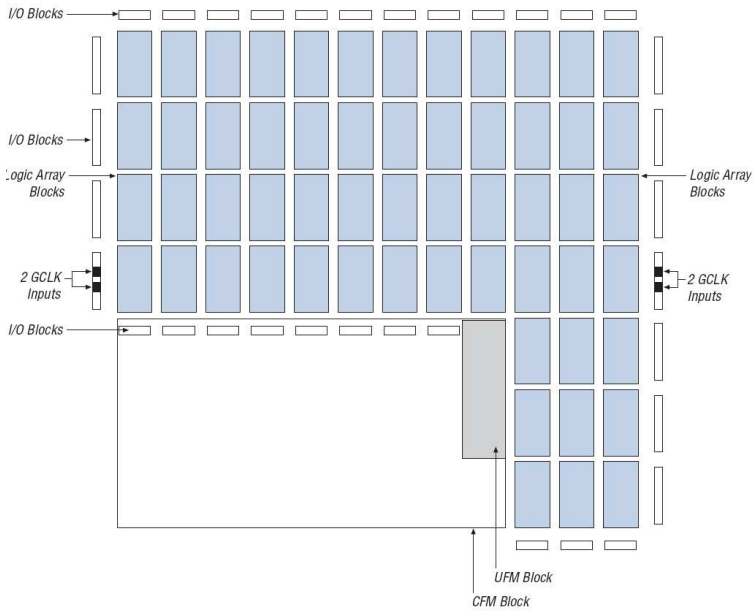
Svitsjregulatoren har en bygghøyde på 8.5 mm og der dermed den komponenten på kretskortet som har størst bygghøyde. Som kort-til-kort-kontakt er det derfor valgt en stiftlist med gull-pletering (og tilsvarende hylslist på databehandlingskortet). Dette gir en kortseparasjon på 10 mm.

4.6 Databehandlingskort

Dette kortet er plassert i mellom de to andre kretskortene, og gjør all databehandling, samt tar seg av timing / kontroll. Det er nesten bare overflatemonterte komponenter på dette kretskortet, bortsett fra de hullmonterte stiftlistene som kobles til strømforsyningskortet. Det er montering både på over- og undersiden av kortet, men det er bare kondensatorer og motstander i 0805 størrelse på baksiden av kortet. Selv om svitsjregulatoren på kortet under er 8.5 mm høy, er komponenter i 0805 størrelse ikke høyere enn 0.6 mm, slik at det har ikke vært nødvendig å ta hensyn til plasseringen av disse komponentene. Et bilde av kortet er vist i figur 4.9.



Figur 4.9: *Bilde av databehandlingskortet.*



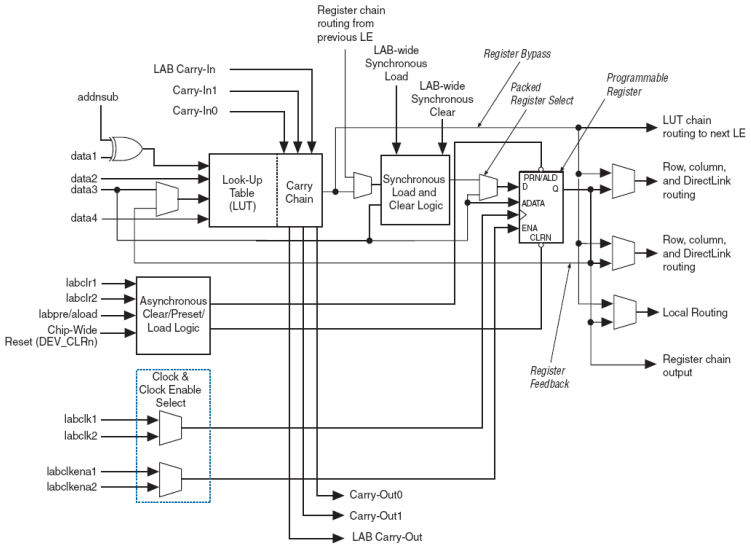
Figur 4.10: *Illustrasjon av utlegget til MAX II CPLD'en. Flashminnet er lokalisert nederst til venstre, mens resten av brikken består av såkalte Logic Array Blocks (LAB). Mellom hver LAB er det programmerbare koblinger.*

4.6.1 CPLD

CPLD'en (Complex Programmable Logic Device) som er brukt i dette designet er en Altera MAX II EMP1270 i 144-pins TQFP (Thin Quad Flat Pack) størrelse. Denne kretsen har 1270 LE (Logical Elemets eller logiske elementer på norsk), som tilsvarer ca 980 makroceller. Det er tilsammen 116 I/O-pinner for generelt bruk på kretsen. EMP1270 har en kjernespenning på 1.8 V, men drivspenningen skal være på 3.3 V eller 2.5 V. En intern lineærregulator i CPLD'en regulerer spenningen ned til riktig spenning.

EMP1270 har internt flash minne. Dette er delt opp i to seksjoner: En CFM-blokk (Configuration Flash memory) og en UFM-blokk (User Flash Memory). I CFM-blokken lagres konfigurasjonen for kretsen, slik at når kretsen skrus på laster flashminnet inn logikken i brikken og man får en "instant-on" operasjon. UFM-blokken tilbyr 8192 bit for generell lagring. Dette gjør at man kan lagre informasjon som overlever en powerdown. En oversikt over CPLD'ens utlegg er vist i figur 4.10.

Som vi ser i oversikten av CPLDen i figur 4.10 består den av såkalte LAB'er (Logic Array Blocks). En LAB består av 10 logiske elementer (LE). En LE er den minste byggeblokken i CPLD'en. Hvordan en LE er bygd opp er vist i figur 4.11. Hvis man ønsker en grundig gjennomgang av hvordan CPLD'en er bygd opp, bør Altera (2006) leses.



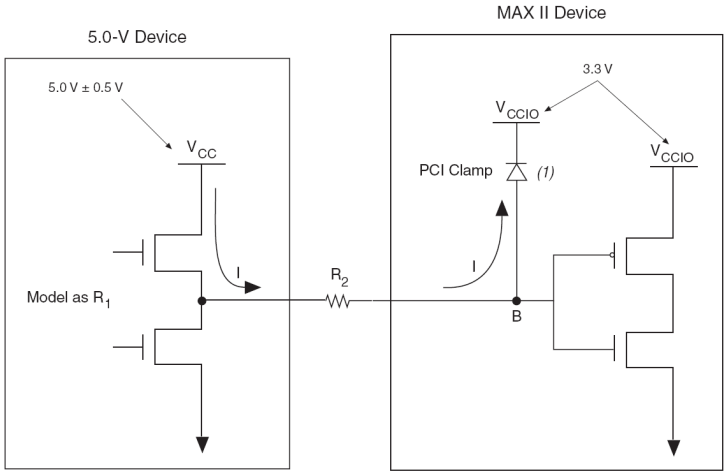
Figur 4.11: *MAX II Logisk Element (LE).*

Det er satt en avkoblingskondensator per powerpinne på CPLDen, tilsammen tolv stykker (tre på hver side). Disse er alle plassert på baksiden av kretskortet for å gjøre routingen enklere. I tillegg så er det satt en 100 µF tantalkondensator på 3.3 V for LF-avkobling, slik som Altera (2006) anbefaler.

CPLD'en støtter JTAG (Joint Test Action Group) programmering. Det er derfor lagt ut signallinjer for dette på programmeringskontakten. CPLDen har interne pull-up motstander på programmeringslinjene som trenger det. Altera anbefaler allikevel at man har eksterne pull-up motstander, da det kan være andre JTAG eneheter i JTAG-kjeden som ikke har det. Siden det er garantert at dette er den eneste komponenten i JTAG-kjeden er disse eksterne pull-up motstandene droppet.

De ubrukte pinnene på CPLD'en er ikke tilkoblet hverken jord eller V_{cc}. Grunnen til dette er at dette ikke har noe å si for støyegenskapene til kretsen, da alle ubrukte pinner programmeres til å være tilkoblet jord. Dette krever ingen ekstra resurser.

MAX II er 5.0 V TTL (Transistor-Transistor Logic) kompatibel, siden utsignalene kan levere 3.3 V spenning og høyt nivå i TTL er definert som 2.4 V - 5 V. CMOS-sensoren deffinerer minimum av høyt nivå til 3.3 V, slik at for den så kan man benytte seg av en funksjon CPLD'en har til å gi 4 V ut (krever bruk av eksterne pull-up motstander og en intern diode). Men ADC'en som er valgt definerer minimum av høyt nivå til 4 V. Dette er på grensen av hva CPLD'en kan gi ut. Det er derfor satt inn et buffer som løser problemet med utgangssignaler fra CPLD'en til CMOS-sensoren (se seksjon 4.6.3).



Figur 4.12: *MAX II 5.0 V innsignal.*

Når det gjelder 5.0 V inngangssignaler til CPLD'en løses dette på en annen måte. Man setter på den interne PCI klampdioden på den aktuelle pinnen som CPLD'en har, i tillegg til å sette på en ekstern seriemotstand (for å begrense strømmen); se figur 4.12. Motstandensverdien er regnet ut slik at maks strøm inn på pinnen er i henhold til det som blir programert i CPLD'en (16 mA):

$$\begin{aligned} R_2 &= \frac{V_{5\text{V}} - V_{3.3\text{V}} - V_{\text{diode}}}{I_{\max}} \\ &= \frac{(5 - 3.3 - 0.7)\ \text{V}}{16\ \text{mA}} \\ R_2 &= 62.5\ \Omega \end{aligned}$$

Man må altså velge en motstand på minst 62.5 Ω. Hvis man velger en motstand som er mye større, vil det gå utover stigetiden til signalet og man risikerer at signalet "kommer for seint". Det er derfor valgt å bruke 100 Ω motstander da dette tillater små endringer i spenningene i tillegg til en rask stigetid.

CPLD'ens logikk er i sin helhelt programert i VHDL ((VHSICHDL) Very High Speed Integrated Circuit Hadware Description Language). Kildekoden finnes i appendiks D.1, og logikken og algoritmene gjennomgås i seksjon 4.8. Tilsammen brukte logikken 542 LE av totalt 1270 LE som utgjør 43 % av kretsens kapasitet.

4.6.2 Klokke

Systemklokken er valgt til 40 MHz. Dette er noe høyere frekvens enn det som er absolutt minimum for å klare å sende ned data med en frekvens

på 6510 Hz. Absolutt minimum frekvens er nedsendingsfrekvens ganger antall klokkesykler en utlesning fra CMOS-sensor tar. En utlesning er på 2048 klokkepulser pluss 146 i det som kalles "line overhead", tilsammen 2194 klokkepulser. I tillegg så gjør CMOSsensoren en klokkedeling på 2 i forhold til klokka den får inn, slik at det tilsammen blir $(2048 + 146) \cdot 2 = 4388$ klokkepulser. Tilsammen gir denne en minimumsfrekvens på

$$f_{\text{min}} = 6510 \text{ Hz} \cdot (2048 + 146) \cdot 2$$
$$f_{\text{min}} = 28.57 \text{ MHz}$$

Absolutt minimum er altså 28.57 MHz. Det hadde altså vært mulig å klare seg med en 30 MHZ klokke, men det ble valgt å gå for 40 MHz i steden. Dette kommer av at det dukker alltid opp noe som trenger ekstra tid, i tillegg til at det er veldig greit å ha muligheter for justeringer.

Klokkeoscillatoren som ble valgt heter CFPS-72 og er i fra C-MAC. Dette er en overflatemontert krystalloscillator i keramisk forpakning (C-MAC 2005). Det ble imidlertid gjort en liten tabbe når det gjelder valg krystalloscillator: Den som er valg er for 5.0 V CMOS/TTL nivåer, mens den som burde vært valgt er CFPS-73 som har 3.3 V nivå. Denne tabben ble ikke oppdaget før kretskortene var bestilt, og da var det for seint å gjøre noe med det. Det er imidlertid ikke noe stort problem, og den eneste konsekvensen er at det blir en ekstra motstand på kretskortet (mellom klokka og CPLD'en). Uansett så har CFPS-72 og CFPS-73 samme pin-out, så et bedre alternativ til senere er å bruke CFPS-73 med 3.3 V strømforsyning og sette inn en null-ohms motstand.

4.6.3 Buffer / driver

Det var behov for en buffer / driverkrets. Grunnen til dette er at ADC'en som er brukt har 4.0 V som minimum av høyt nivå, men CPLD'en normalt gir ut 3.3 V. Ved å sette på en ekstern pull-up motstand og aktivere en PCI-clamping-diode i CPLD'en kan man få utspenningen til å komme opp til 4.0 V. Muligens ville dette holdt, men for å være på den sikre siden ble et buffer / nivåskifter satt inn.

Bufferet som opprinnelig var valgt var 74AHCT244 fra Phillips Semiconductors. Grunnen til at dette bufferet er valg er at høyt nivå inn (V_{IH}) er definert som over 2.0 V, høyt nivå ut (V_{OH}) er 4.4 V og at signalforsinkelsen ("propogation delay") er på typisk 3.5 ns (Phillips Semiconductors 2006). Bufferet som har blitt brukt er derimot 74ABT244. Det er veldig likt, men har noen forskjeller: signalforsinkelsen er på typisk 2.9 ns og (V_{OH}) er oppgitt til å være minimum 3.0 V (Phillips Semiconductors 1998). Grunnen til at det er blitt brukt ABT (Advanced Bi-CMOS) isteden for AHCT (Advanced High-Speed CMOS logic) er at ABT var tilgjengelig fra leverandør, mens AHCT var det ikke.

Det som er oppgitt om ABT-utgaven er at den har en (V_{OH}) på 3.0 V ved et strømtrekk på 3 mA. Kretsen som trenger 4.0 V som innspenning (ADC'en) trekker maksimalt 5 µA. Det ble derfor testet hva utspenning på ABT-utgaven var med en belastning på 5 µA. Spenningen ble målt til 4.1 V på testeksemplaret og det ble derfor antatt at ABT-utgaven holdt. Det ble derfor heller ikke laget mulighet for pull-up motstander på kretskortet. Dette har i ettertid vist seg å være uheldig. 74ABT244-pakken som var i det første instrumentet leverte 4.1 V til ADC'en, men pakken som var med i det andre instrumentet har vist seg at leverer 4.0 V (som altså er ingen forskjell i forhold til hva man kunne klart uten buffer, men med pull-up-motstander i stede). Dette er på grensen av hva databladet til ADC'en angir som minimum av høyt nivå, og det er ikke spesielt heldig å ha spenninger som ligger helt på grensen. Det bør derfor i fremtiden benyttes AHCT-utgaven og / eller pull-up motstand. Størrelsen på pull-up motstanden bør være på 5 V /10 mA $= 500\Omega$ eller noe større (ABT kan levere / trekke minst 32 mA og AHCT 20 mA). Instrument nummer 2 viste seg allikevel å operere stabilt, så det ble ikke satt på pull-up motstander.

4.6.4 RS-232

Det ble inkludert et RS-232 grensesnitt med to utganger, og en inngang. RS-232 grensesnittet gjør at man med letthet kan kommunisere med en datamaskin, og var tiltenkt brukt under testing og kalibrering.

Å ha med en RS-232 mottaker på instrumentet har vist seg å være meget nyttig under uttesting, da dette har tillatt at variabler (blant annet integrasjonstid) har vært lett å justere uten å måtte reprogrammere CPLD'en. RS-232 senderne har blitt brukt til å sende de samme dataene som sendes til rakettenkoder. Dette har gjort testing / kalibrering veldig lett. Mer om dette i kapittel 5.

RS-232 brikken som er brukt er MAX202-kretsen fra Maxim. Denne har en overføringshastighet på 115200 bps (bit per sekund) og krever fire eksterne kondensatorer på 100 nF i tillegg til avkoblingskondensator (Maxim Integrated Products 1996). Mimimumspenningen på inngangene til kretsen for høyt nivå (V_{IH}) er oppgitt til å være 2.0 V, så det trengs ingen driver / pull-up fra CPLD til RS-232. V_{OH} (høyt spenningsnivå fra utgangene) er derimot oppgitt til å være mellom 3.5 V og V_{cc} (5.0 V). Det trengs derfor en motstand mellom utganger på RS-232 og CPLD (for å begrense strømmen). Det er tatt utgangspunkt i at denne kommer opp i 5.0 og derfor er det satt på en 100 Ω motstand som for de andre 5.0 V signalene som går til CPLD.

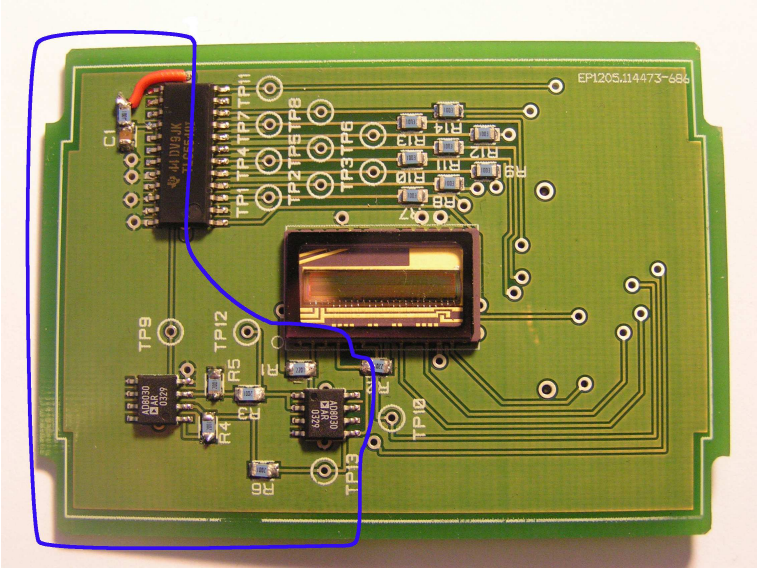
4.6.5 Kort-til-kort-kontakter

Kort-til-kort-kontaktene til strømforsyningskortet er som tidligere nevnt et stiftlist-hylsepar. Til sensorkortet derimot trengs bare 5 mm, og det måtte

derfor velges en annen kort-til-kort-kontakt. Det er valgt å bruke Qstrip fra Samtec. Disse passer til høydeforskjellen mellom kortene (5 mm), har 40 pinner og en frekvens rating på 275 MHz (Samtec 2001). Det er lagt en jordpinne i mellom hver signalpinne slik at returstrømmen kan gå nærmest mulig signalet, samt minske overhøring mellom signalene.

4.7 Sensorkort

Sensorkortet er det øverste av de tre kortene, naturlig nok siden sensorbrikken trenger fri sikt ut av boksen. Det er kun overflatemonterte komponenter på dette kortet. Komponentene er hovedsakelig montert på oversiden av kortet, mens avkoblingskondensatorene og kort-til-kort-kontakten er plassert på baksiden. Dette kortet har både analoge og analog/digitale kretser. Det er derfor viktig å skille de digitale signalbanene fra de analoge slik at den høyfrekvente digitale støyen påvirker de analoge signalene minst mulig (se figur 4.13)



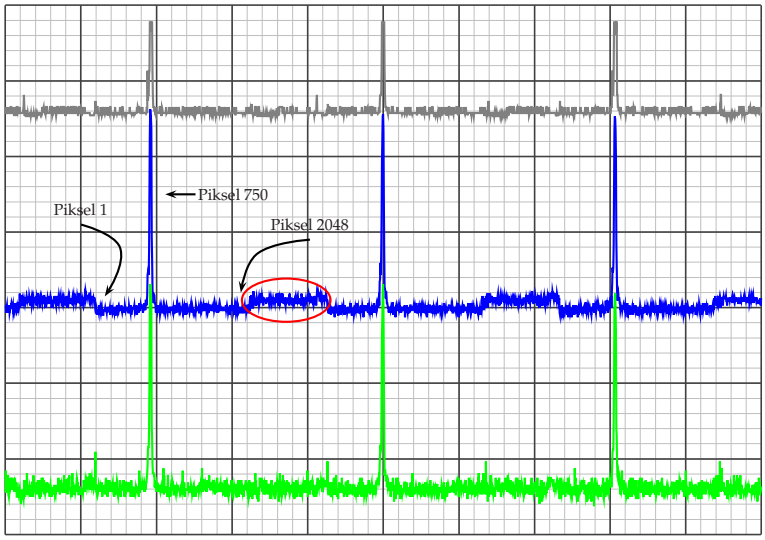
Figur 4.13: *Sensorkort med analogdel skisset inn. Området markert med blått er den analoge delen, mens resten er digitalt.*

4.7.1 Linjekamera (SLIS-2048)

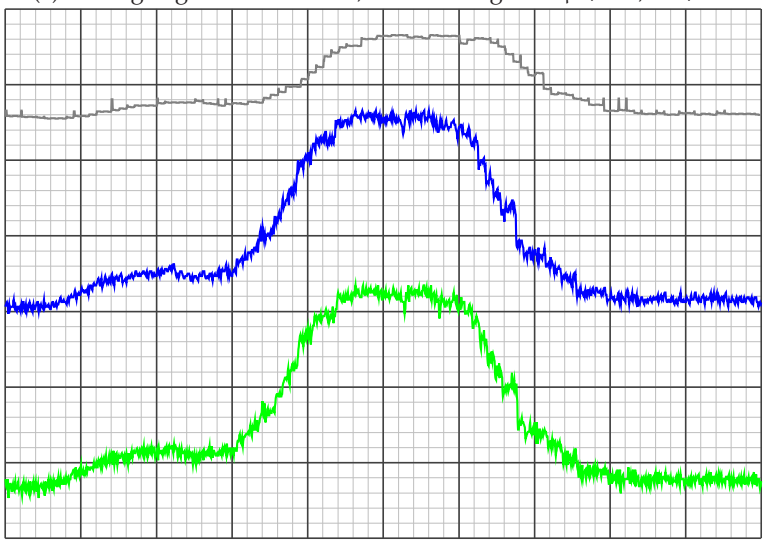
Bildesensoren som benyttes i instrumentet er CMOS-brikken SLIS-2048 fra Panavision SVI. Den har 2048 piksler som hver er på 7 µm $\times$ 7 µm. Fillfactor er på over 99%, og den har en maks utlesningshastighet på 56 millioner piksler per sekund (Panavision Imaging 2005).

Det var svært vanskelig å finne en sensor som tilfredsstilte kravene. Det finnes en god del CMOS linjesensorer, men disse har som regel en god del mindre aktivt område og med færre piksler. Ved å bruke sensorer med færre piksler synker oppløsningen. På et tidspunkt ble det vurdert å bruke en CCD-brikke i stedenfor, men tilslutt ble SLIS-2048 brikken funnet.

Det som er litt spesielt med denne brikken er at den ikke har en intern ADC. Det har derfor vært behov for å se nøye på de analoge signalet som sensoren sender ut. Det er heller ikke mulig å adressere ett piksel i sensoren, og dette gjør at den ligner veldig på en CCD-brikke, selv om den altså er en CMOS-brikke. Det analoge signalet fra brikken kan sees i figur 4.14.



(a) Analogt signal fra SLIS2048, tre utlesninger. 50 µs/div, 1 V/div



(b) Analogt signal fra SLIS2048, forstørret i tid. 50 ns/div, 1 V/div

Figur 4.14: *Analogt signal fra SLIS-2048 med laser som lyskilde, sammen med de digitaliserte signalene fra ADC'en TCL5540. Tiden går fra venstre til høyre i bildet, og spenningsnivået angis med høyden på grafen for de to analoge signalene (blå og grønn), mens for grå linje er det digitalnivået fra ADC'en som angis. Signalet som er markert med blå linje er signalet i fra SLIS-2048 til spenningsfølgerne, mens signalet som er markert med grønt er signalet fra differanseforsterkeren til ADC'en (TCL5540). En ting som er viktig å merke seg at signalet her er uten bruk av pin-hole, og dette signalet er derfor betydelig bredere enn det som er med bruk av pin-hole. Nivåene på de digitale signalene går fra 7 til 255.*

Det vi ser i figur 4.14a er hvordan signaler fra sensoren ser ut over tre utlesningsrunder. Utlesningen begynner med et lavt nivå litt uti andre rute fra venstre (piksel 1). Her detekteres det ikke nevneverdig lysintensitet. Etter ca en tredjedel av utlesningstiden (like før overgangen mellom rute to og tre) kommer en "spiker" i signalet. Her er lysintensiteten kjempehøy, og vi har detektert sol (piksel nummer ca 750). Så forsvinner lyset og sensoren faller ned på lavt intensitetnivå igjen. Litt uti fjerde rute er utlesningen ferdig og det blir rare verdier fra sensoren (markert med rød sirkel). Litt uti femte rute startes en ny utlesning. Figur (b) viser hvordan "spikeren" ser ut når den blir forstørret i tid. Man ser signalet er ganske grumsete med mye små variasjoner i figur (b).

Det er et par ting om SLIS-2048 som man bør merke seg. For det første så er ikke de første 16 og de 3 siste pikslene nødvendigvis innenfor spesifikasjonene. De kan altså gi ut rimelig ville verdier uavhengig om de er belyst eller ikke. Dette er både beskrevet i databladet og observert. Det andre er at det analoge signalet kan variere en del fra brikke til brikke i perioden da det ikke leses ut fra sensoren. Dette skal uansett ikke leses så dette er ikke spesielt viktig.

Noe som er ganske kritisk for sensoren er integrasjonstiden. Denne bestemmer hvor høy den integrerte intensiteten blir, og ved å velge for kort integrasjonstid kan det hende at intensiteten blir så lav at det ikke er mulig å detektere sola. Settes integrasjonstida for lang går hele sensoren i metning. Det ble eksperimentert fram til at 64 µs var en god integrasjonstid. Denne tida gjør at sensoren går så vidt i metning uten filteret. Det er normalt ikke ønskelig at sensoren går i metning, men med filteret foran senkes intensiteten betraktelig (50%). Riktignok så økes intensiteten betraktelig når man beveger seg opp i atmosfæren. Uansett så har det ikke så mye å si at sensoren går i metning, siden dette bare fører til en minimal utsmørning. Dersom det hadde blitt brukt en CCD-brikke kunne det å gå i metning vært helt ødeleggende for samtlige piksler.

Når integrasjonstiden er på 64 µs, så fører dette til at reset-tida er på 89.6 µs. Eller med andre ord: Sensoren er blind i 89.6 µs. Siden sola har en utstrekning på 0.53°betyr dette at instrumentet maksimal kan ha en spinnhastighet på $0.53° / (360° \cdot 89.6)$ µs $= 16.4$ rps for at vi kan være garantert å detektere sola.

Det ble vurdert å lage en digital automatic gain control til SLIS-2048. Denne ville justert integrasjonstiden slik at sensoren aldri gikk i metning. Problemet med en automatic gain control er at den alltid vil ligge litt på etterskudd, og problemet med at sensoren går i metning ble ikke sett på som stort nok til å bruke noe tid på å implementere en slik funksjon.

Det er to utgangssignaler fra denne kretsen, et signal fra pikslene som er belyst, og i tillegg et sett med piksler som er skjermet for lys. Ved å ta differansen mellom de belyste pikslene og de skjermede kan man fjerne mørkestrømmen. Det er det gjort i dette designet. Det er dog viktig at mør-

kestrømmen ikke blir så høy at den blir jevnstor med signalet generert av lys. Det er oppgitt at sensoren har en gjennomsnittlig mørkestrøm på 0.07% av maksimalt utslag ved 25°C. Hvis man antar at temperaturen i nesekonen blir 50 - 70°C, så er det i værste fall 45°C forskjell. Mørkestrømmen dobles for hver 6 - 10°C. Mørkestrømmen dobler seg altså inntil $45^\circ/6^\circ = 7.5$ ganger. Det vil si at den kan komme opp i $0.07\% \times 2^{7.5} = 12.7\%$. Dette er veldig høyt, men ikke høyt nok til å ødelegge målingene.

4.7.2 Hvorfor CMOS ble valgt

Det er tre hovedgrunner til at CMOS-teknologi ble valgt fremfor CCD: Strømforbruk, plass og enkelhet.

Strømforbruket er som sagt langt lavere på en CMOS-sensor enn for en tilsvarende CCD-sensor. I en rakett (eller satellitt for den saks skyld) er det begrenset hvor mye strøm man kan bruke, siden instrumentene opererer på batteridrift. Det er derfor viktig å velge komponenter som er så gjerrige i strømforbruket som mulig.

Plass og enkelhet henger mer eller mindre i sammen. CCD-sensorer må som tidligere nevnt som regel ha en del eksterne komponenter for å operere, mens med CMOS trenger man langt færre. Dette gjør at man kan spare en del plass (og vekt) ved å bruke CMOS-sensor. I tillegg så forenkles designet med færre komponenter, og hele designprossessen blir lettere. Med færre komponenter blir det også færre ting som kan gå galt.

I tillegg så er ikke CMOS plaget med feilen som kalles "smearing" slik som CCD er. Dette er en feil som oppstår når et piksel går i metning (fra langbølget lys). Denne vil da "smøre seg utover" ut i overføringskanalen, og dermed vil alle pikslene som bruker denne overføringskanalen se lyse ut.

4.7.3 Instrumenteringsforsterker (AD8030)

På utgangene av SLIS-2048 var det anbefalt å ha en instrumenteringsforsterker. Til dette så er det brukt operasjonsforsterkeren AD8030 fra Analog Devices. Dette er en hurtig rail-to-rail operasjonsforsterker med lavt effektforbruk (Analog Devices 1996). Denne kretsen er strengt tatt litt for treig i og med at den har en "settling time" til 0.1 % på 80 ns ved 2 V endring, i og med at tiden mellom to piksler er på 50 ns. Slew-raten er ikke begrensende. Den har allikevel vist seg å fungere tilfredsstillende, i tillegg til at det ikke var mulig å finne andre raskere rail-to-rail operasjonsforsterkere som fungerte tilfredsstillende (omkring sju forskjellige kretser ble testet).

Ved å ha komponenter mellom SLIS-2048 og ADC'en blir signalforsinkelsen merkbar, dog ikke stor. Den er målt til å være ca 10 ns.

4.7.4 Analog til digital omformer (TLC5540)

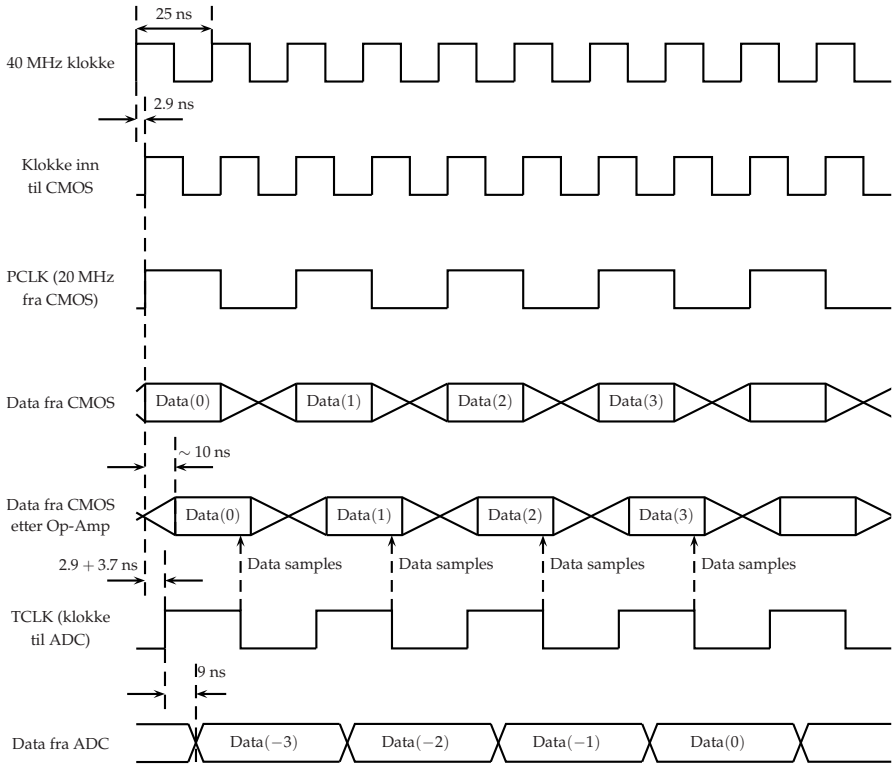
TLC5540 er ADC'en som er brukt. Dette er en 8-bits parallell ADC som klarer opp til 40 MSPS (Megasampler per sekund) (Texas Instruments 1999). Konvereringstiden er tre klokkepulser, og med en klokke på 20 MHz gir det en konverteringstid på 150 ns.

Referansespenningene til ADC'en kan justeres individuelt, så lenge forskjellen mellom de to er minst 1.8 V og maks V_{cc}. I dette designet er bunnreferansen satt til 0 V. Toppreferansen var opprinnelig satt til 2.28 V, siden maksimalt spenningsutslag var 2 V i følge databladet til CMOS-sensoren. Dette viste seg å være feil, og maks spenningsutslag ble målt til nesten 3.0 V. Dette var det ikke tatt høyde for på kretskortutlegget, og det måtte gjøres en "jukseløsning" for å få plassert motstanden som skulle justere referansespenningen. Denne "jukseløsningen" kan sees i figur 4.13. Det er den røde ledningen tilkoblet motstanden øverst til venstre som måtte til for å få løst dette problemet. Det denne løsningen gjør er å trekke den høye spenningsreferansen til kretsen opp fra 2.3 V til 3.0 V.

Utspenningen fra ADC'en er på 5.0 V. Dette er for høyt for CPLD'en og det settes derfor også her på 100 Ω motstander for å begrense strømmen, slik som det er gjort for de andre komponentene.

Det er viktig at timingen er korrekt for ADC'en, ellers sampler vi feil signal. Timingdiagram over klokker, data og sampling er gitt i figur 4.15.

Timingdiagrammet viser at data fra CMOS-brikken samples på synkende klokkeflanke av ADC'en. ADC'en bruker tre klokkepulser på å sample, digitalisere og legge dataene ut. Faktisk så bruker den litt lengre tid, nemlig 3.5 klokkepulser pluss 9 ns. Det er ikke vist i diagrammet, men CPLD'en sampler dataene fra ADC på synkende flanke på PCLK.



Figur 4.15: *Timingdiagram for TLC5540. I diagrammet kan de se ut som "Data fra CMOS" og "Data fra CMOS etter Op-Amp" er digitale, det er de* ikke. *De er kun skissert på denne måten for lettere å kunne vise tidsforsinkelsen som Op-Ampene gir.*

4.8 Instrumentets digitale logikk

Den digitale logikken er veldig sentral i dette designet. Det er utformingen og testing av denne som helt klart har tatt mest tid. Alt av den digitale logikken er i CPLD'en og er programmert i VHDL.

Et alternativ til CPLD er å bruke det som kalles for en mikrokontroller. En mikrokontroller kalles også computer-on-a-chip, og det er et veldig beskrivende navn. Mens en CPLD er programmerbar logikk, er en mikrokontroller et mye mer helhetlig system med in-put, out-put, minne, CPU, og en egen klokke. Bruk av en mikrokontroller kan redusere antall komponenter som trengs, siden den har mange av disse innebygd. Det store problemet med mikrokontrollere er at det kan være veldig vanskelig å gjøre en god timing, og man mister veldig mye av kontrollen av hvordan den opererer (på grunn av at man skriver i et høynivåspråk som for eksempel C). Det er dette som har gjort at det ikke er brukt en mikrokontroller i dette designet, siden både timing og presis virkemåte er svært kritisk.

Et annet alternativ til CPLD er det som heter FPGA (Field Programmable Gate Array). En FPGA er veldig lik en CPLD, men den har et par forskjeller. FPGA'er har som regel plass til mer logikk, men den viktigste forskjellen er at de har behov for en ekstern krets til konfigurering av FPGA'en. Dette gjør den mindre egnet siden det er ønskelig å lage et så lite instrument som mulig, samtidig som behovet for logikk ikke krever bruk av en FPGA.

4.8.1 VHDL som programmeringsspråk

Det er en del forskjeller på VHDL- (hardware) og software-programmering. En av de mer opplagte er at software-programmer jobber sekvensielt, mens hardware jobber parallelt.

Dette og andre ting gjør at det er en del feller å gå i og en del feil som er lett å gjøre. Disse er det ikke alltid lett å oppdage, og det er derfor viktig (og som regel tidsbesparende) å gjøre en så fullstendig simulering som mulig. Det er også noen feil som ikke kan oppdages i simuleringen, så det er også viktig å teste grundig at koden faktisk fungerer fysisk også.

Det har blitt oppdaget en god del feil både ved simulering og to feil ved fysisk testing. Feilene som ble funnet ved simulering var logiske feil / feil i programmeringen, mens feilene som ble funnet ved fysisk testing er faktisk feil som skyldes at CPLD'en ikke opererer som beskrevet. Den ene feilen er at kretsen som standard bruker det som kalles "one-hot-enkoding" for tilstandsmaskiner (det gås ikke nærmere inn på hva det er her, mer om dette kan leses i Skahill (1996)). Når denne enkodingen ble brukt førte dette ofte (ca 1% av tida) til at tilstandsmaskinen kom i en ugyldig tilstand, og derfor ikke opererte riktig. Når enkodingen ble bytta til "grey-enkoding" fungerte derimot tilstandsmaskinen fint.

Den andre feilen som ble funnet ved fysisk testing er det antakelig mulig å detektere ved hjelp av simulering, da det mistenkes at denne feilen kommer av dårlig kompilator og ikke kretsen selv. Det var imidlertid ikke mulig å se denne feilen under simuleringen som ble gjort. Denne feilen har også med tilstandsmaskin å gjøre. Detaljene er ikke viktige, men den gikk ut på at en av tilstandene ble hoppet over. Løsningen på dette var å gå fra en én-prosess-tilstandsmaskin til en to-prosess, i tillegg til å legge inn dobbel sjekk på tilstandsskiftingen.

4.8.2 Blokkskjema over digital logikk

Det er viktig å ha en god oversikt over hvordan den digitale logikken opererer, både for timingen og for å kunne skrive god VHDL-kode. Et blokkskjema over den digitale logikken er vist i figur 4.16.

For å bedre kunne forstå skjemaet gåes det kjapt igjennom hva den enkelte komponent gjør:

Timing og kontroll Denne komponenten tar seg av alt som har med timing og kontrollsignaler å gjøre. Den styrer alle eksterne komponenter og gir ut klokke til alt som trenger det.

SolFinner Denne komponenten leser inn data fra SLIS-2048 og avgjør om og eventuelt hvor sol detekteres. Solsenteret sammen med antall opplyste piksler overføres til SolBuffer ved endt utlesning.

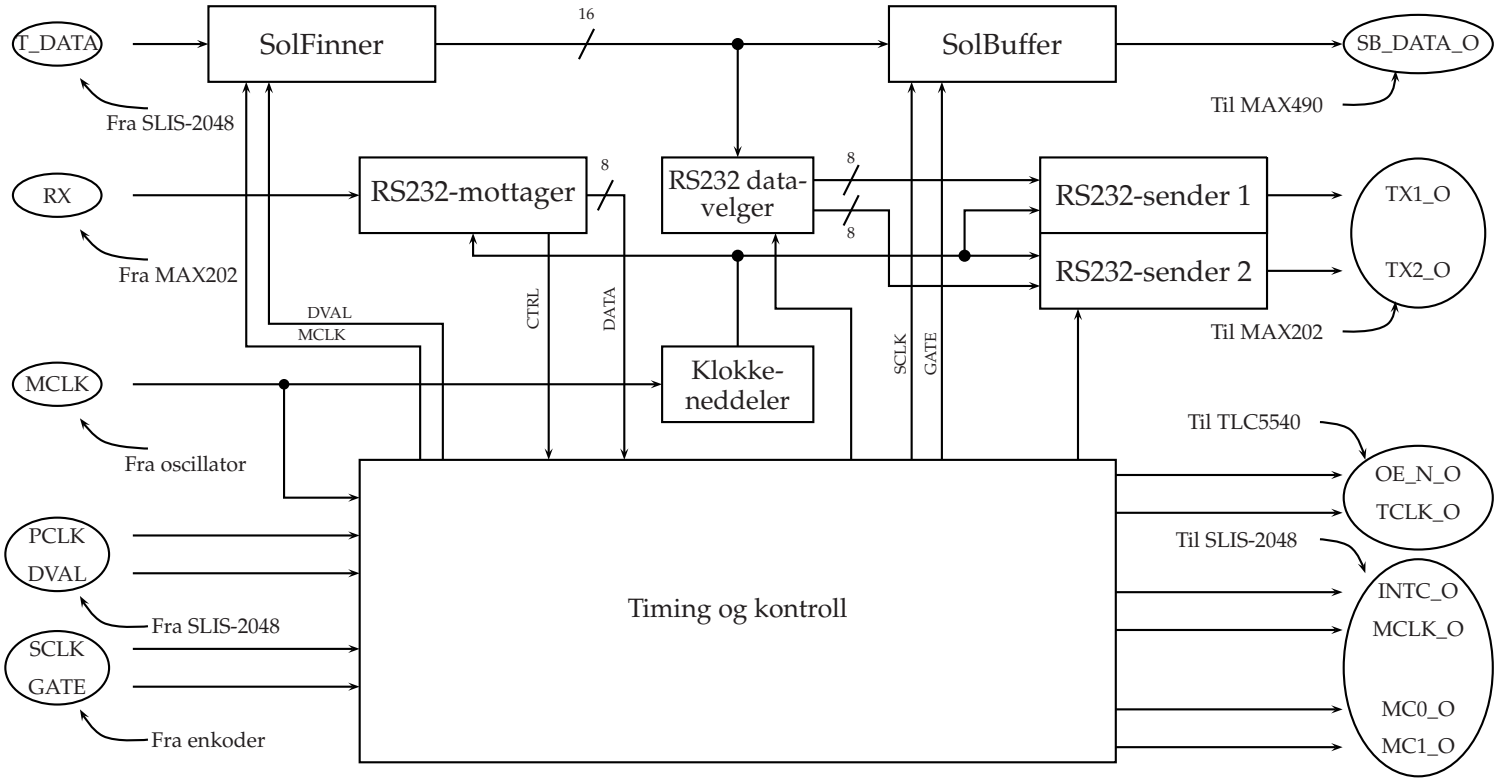
SolBuffer Denne komponenten bare holder data fra SolFinner og legger de ut til enkoder på riktig tidspunkt.

Klokkeneddeler Kun brukt for å få en klokke til RS232-komponentene.

RS232 mottager Brukes for å ta i mot data over RS-232 grensesnittet. Denne kan brukes til justering av instrumentet uten å omprogrammere (som for eksempel integrasjonstid).

RS232 datavelger Man man maks sende 9 bit per sending over RS-232 grensesnittet. Fordi data fra SolFinner er på 16 bit må det velges hvilke bit som skal sendes.

RS232 sender 1 og 2 Komponenter som sender ut data i henhold til RS-232 standarden.



Figur 4.16: *Blokkskjema over digital logikk. Elementer markert med ellipser er fra eller til eksterne komponenter. Elementer markert med rektangler er forskjellige komponenter i den digitale logikken.*

4.8.3 Utregning av solsenter

Utregning av solsenter kan gjøres på veldig mange måter. Men siden denne utregningen må gjøres i instrumentet er det svært begrenset hva som er tilgjengelig av regneressurser. Blant annet så er generell divisjon og multiplikasjon for krevende for en CPLD. Riktignok er det en smal sak å dele eller gange med 2 siden dette kun er å fjerne eller legge til et bit. Men i utgangspunktet er det kun addisjon og subtraksjon som kan brukes.

Det første problemet er uansett å definere hva som er sol. Det finnes mange forskjellige muligheter. Den enkleste er antakelig at hvis det blir detektert over en gitt intensitet, så er det sola som er detektert. Det er dette som er definert som sol i dette designet, og grensen er satt ved 50% av maks intensitet.

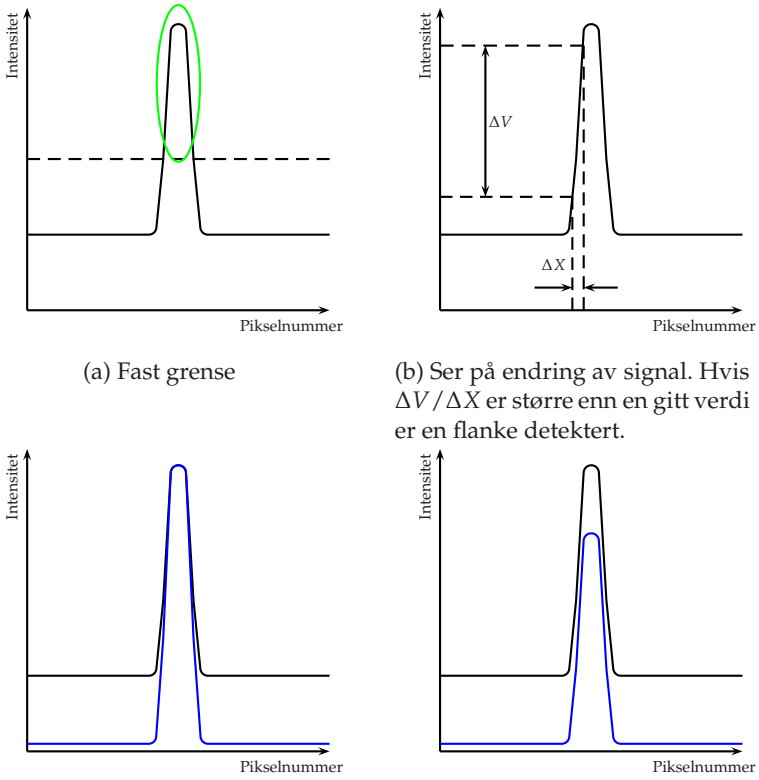
Det finnes mange andre gode og antakelig bedre løsninger. Blant annet så kan man se på endring av nivåene i stedet for å se på verdien av nivåene. Ved å bruke denne løsningen vil instrumentet klare å skille ut et lyssterkt objekt i en ellers lys bakgrunn. Denne løsningen krever imidlertid mye regneregnekraft og/eller minne.

En annen ting man kan gjøre er å se på hva gjennomsnittsverdien var forrige runde, og bruke dette som et slags nullnivå for bestemmelse om hva som er sol. Denne løsningen vil også klare å skille ut lyssterke objekter i en lys bakgrunn. Denne løsningen trenger også noe minne, og multiplikasjon- og divisjonoperasjoner.

En annen ting å gjøre er å bruke en form for feed-back for å justere integrasjonstiden, men det kan være vanskelig å finne en god algoritme for dette. I figur 4.17 er nevnte algoritmene illustrert.

Etter at det er blitt definert hva som er sol, er neste problem å avgjøre hva som er senter av sola. En veldig enkel metode er å ta første piksel som er over grenseverdien pluss siste piksel over grenseverdien, for så å dele svaret på to. Da finner man senter mellom disse to pikslene. Det er imidlertid et par problemer med denne. Det ene er hvis vi har et veldig "skeivt" signal, og det andre er hvis man har to lyskilder. Se figur 4.18. I (a) er det ikke opplagt at solsenteret er hvor det er regnet ut, eventuelt hvis middelplatået hadde ligget rett under grenseverdien, så kunne det hende at dette faktisk er sol, men at et ukjent objekt som skygger for sola senker intensiteten. Det er ikke tatt høyde for slike hendelser i dette designet, da det er svært usannsynlig at det skulle komme et objekt mellom sola og instrumentet. Dersom instrumentet skulle brukes på bakken mot en annen lyskilde er dette scenariet tenkelig, men i en sonderakett så er det ikke en reell mulighet at det skjer.

Hva man skal gjøre med hendelsen i figur 4.18b er derimot en større bekymring. At denne situasjonen oppstår er mulig da den andre lyskilden kan komme fra for eksempel refleksjoner fra raketten (intensiteten og bredde vil da antakelig være lavere, men kan fremdeles komme over ters-



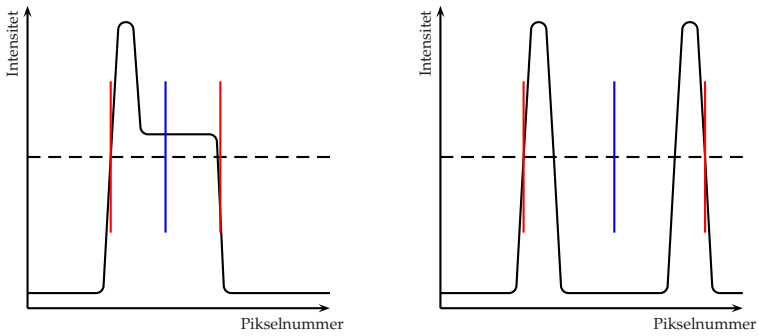
(a) Fast grense

(b) Ser på endring av signal. Hvis $\Delta V / \Delta X$ er større enn en gitt verdi er en flanke detektert.

(c) Skalering og nivåskifting av signal

(d) Endring av integrasjonstid

Figur 4.17: *Forskjellige løsninger for deteksjon av sol. I figur (a) behandles alt som er over en gitt grense som sol, mens i figur (b) er det endringen av intensitet som blir sett på. I figur (c) nivåjusteres og skalleres først signalet (fra svart graf til blå graf), før man enten ser på nivåverdier (som i (a)), eller nivåendringer (som i (b)). I figur (d) er det vist hva som skjer dersom man endrer integrasjonstiden.*



(a) "Skjevt signal (b) To lyskilder

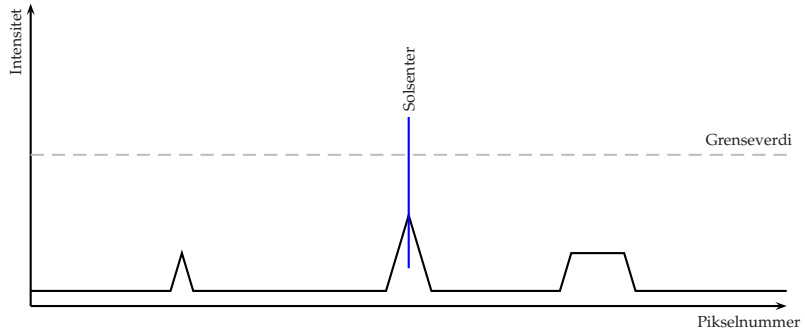
Figur 4.18: *Problemdeteksjoner. Første og siste piksel over grensenivået er markert med rød strek, mens det som blir utregnet som senter med den enkle algoritmen er markert med blå.*

kelverdi). En annen mulighet er at et eller flere av pisklene er / blir defekte og gir maks utslag uansett. Det er derfor satt opp en algoritme som også håndterer slike hendelser.

Algoritmen for å finne solsenter

Man kan se for seg i utgangspunktet tre forskjellige hovedtilstander: Ingen piksler kommer over terskelnivået, én sammenhengende gruppe av piksler kommer over terskelnivået, to eller flere ikke-sammenhengende grupper kommer over terskelnivået.

Ingen piksler over terskelniåvet Hvis dette skjer så er det mest sannsynlig ikke solstråler som treffer sensoren. Da settes antall opplyste piksler (de fem mest signifikante bit'ene) til 0. Solsenter settes til pikselet med høyest intensitet. På den måten vil instrumentet forhåpentligvis kunne gi ut fornuftige data selv om for eksempel terskelverdien er for høy. Se figur 4.19.



Figur 4.19: *Algoritme for å avgjøre hvor sol er (a): Ikke høy nok intensitet. Her ser vi et signal som ikke kommer over grensenivået. Det gjettes allikevel på at punktet med størst intensitet er solsenteret (markert med blått). Hvis det er to piksler med like høy intensitet, settes posisjonen midt mellom disse til senter.*

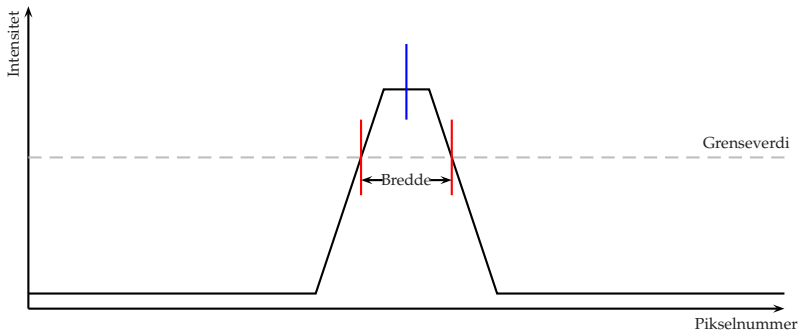
En sammenhengende gruppe av opplyste piksler Hvis man får én og kun én sammenhengende gruppe av en eller flere opplyste piksler settes solsenter X_M til

$$X_M = \frac{X_F + X_L}{2}$$

der X_F er første piksel over terskelnivå, og X_L er siste piksel over terskelniåvet. Solbredden X_W settes til

$$X_W = \frac{X_L - X_F}{2}$$

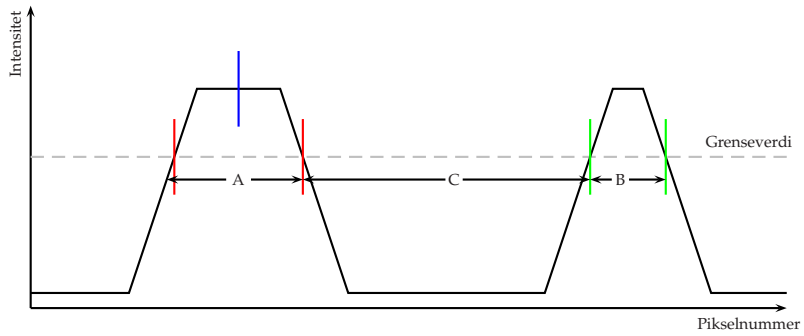
Det deles på to for å få et større område å sende ned bredde på. Slik det er nå så kan altså solbredden være 0 til 64 piksler bred med en oppløsning på 2 piksler. Man også tenke seg at man ønsker seg et enda større område, og da dele på 4. Man får da mulighet til å sende ned en bredde på 0 til 128 i oppløsning på 4 piksler. Man kan også tenke seg andre mer dynamiske metoder. Se figur 4.20 (det flate området er fullt utslag).



Figur 4.20: *Alogoritme for å avgjøre hvor sol er (b): Én gruppe over grensenivå. Første og siste piksel over grensenivået er markert med rødt. Pikselet midt i mellom disse to antas å være solsenteret (markert med blått).*

To eller flere ikke-sammenhengende grupper Hvis det er flere enn én gruppe så må man gjøre gjetninger på hva som har skjedd / hvilken som er lys fra sola. Hvis man først ser på hvordan man skal avgjøre hvis det er to grupper, er det tre forskjellige tilfeller som kan oppstå: Den første gruppen er bredest, den siste gruppen er bredest, eller de er like brede.

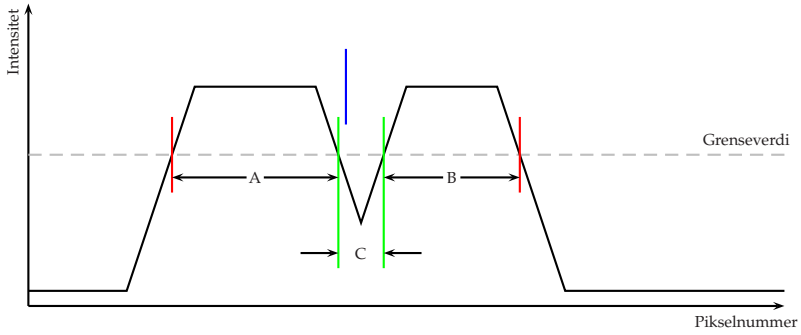
I det første tilfellet, der den første gruppen er bredest, ser vi på den som hovedgruppen. Det må da avgjøres hva den andre gruppa er. Hvis den smaleste gruppa (markert med B i figuren) er smalere enn avstanden mellom de to gruppene (markert med C), så blir det antatt at denne smaleste gruppa enten er døde / slitne piksler som gir ut feil verdi eller at det som er detektert er en refleksjon eller lignende (se figur 4.21). Man ser da bort i fra den andre og smaleste gruppa og behandler disse pikslene som de var ikke-opplyste.



Figur 4.21: *Alogoritme for å avgjøre hvor sol er (c): To grupper over grensenivå, avstanden mellom de er større enn den smaleste gruppen. Vi ser at bredden B er mindre enn avstanden C (avstand mellom de to opplyste områdene). Derfor blir den andre gruppen sett bort i fra, og senter blir midt i den første gruppa (markert med blått). Rødt markerer ytterpunktene på det som blir regna som sol, mens grønt markerer ytterpunktene på det som opprinnelig var detektert som sol, men som ble forkastet.*

Dette scenariet skjedde rimelig ofte før algoritmen for å avgjøre hvilke av to topper som er sol er ble implementert. Det ble da detektert mye verdier som opplagt var feil.

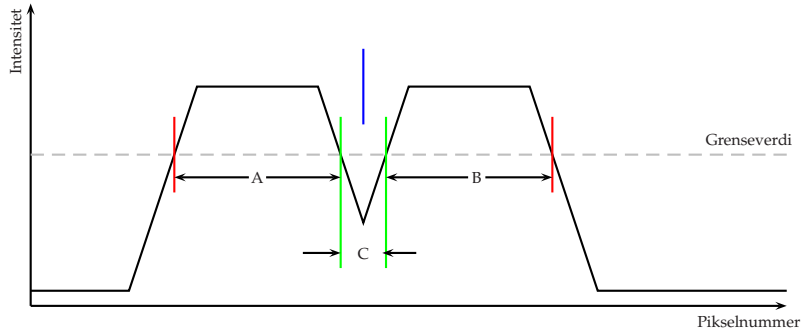
Hvis den smalete gruppa derimot er større en avstanden mellom de to gruppene antas det at de ikke-opplyste pikslene mellom de to gruppene er døde / slitne piksler som gir ut lavt nivå uansett, eller at det har kommet rusk eller tilsvarende på sensoren. Da blir de to gruppene sett på som én gruppe og de mørke pikslene i mellom blir sett på som opplyste (se figur 4.22).



Figur 4.22: *Alogoritme for å avgjøre hvor sol er (d): To grupper over grensenivå, avstanden mellom de er smalere enn den smaleste gruppen. Vi ser at avstanden C er smalere enn bredden B. Derfor blir det sett på som det mørke området er døde / slitne piksler, og at disse egentlig skulle vært lyse. Med andre ord så blir området i mellom de to grønne markørene sett på som høyt og dermed blir de rød markørene første og siste piksel over grensenivået. Senter i mellom disse to (sentersol) er markert med blått.*

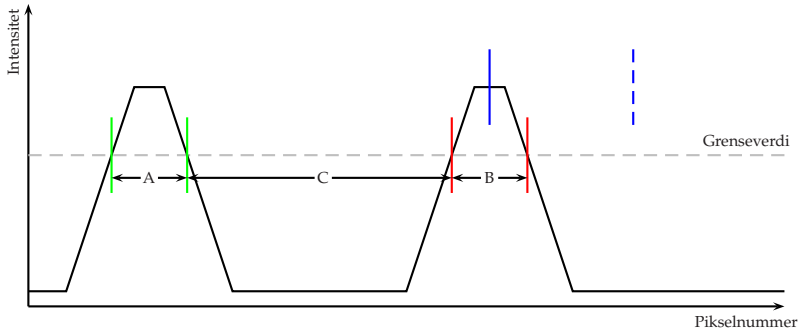
Helt analogt med det over, skjer hvis den siste gruppa er bredest. Hvis bredden av den smaleste gruppen er større enn avstanden mellom de, blir de sett på som én gruppe, hvis ikke ser man bort i fra den smaleste gruppen.

Hvis derimot gruppene er like store, har vi et nytt problem. Det vil si, hvis avstanden mellom gruppene er smalere enn størrelsen på gruppene så antar vi igjen at de mørke pikslene er døde / slitne piksler, og antar at de skulle vært opplyste (se figur 4.23).



Figur 4.23: *Alogoritme for å avgjøre hvor sol er (e): To like store grupper over grensenivå, avstanden mellom de er smalere enn bredden på gruppene. Vi ser at avstanden C er smalere enn bredden A og B. Derfor blir det sett på som det mørke området er døde / slitne piksler, og at disse egentlig skulle vært lyse. Med andre ord så blir området i mellom de to grønne markørene sett på som høyt og dermed blir de rød markørene første og siste piksel over grensenivået. Senter i mellom disse to (sentersol) er markert med blått.*

Hvis gruppene er smalere enn avstanden mellom de, så er det veldig vanskelig å gjette på hvilken som er riktig. Men i og med at en rakett har en saktevarierende koning, kan det gjettes på at den gruppen som er nærmest der forrige soldeteksjon var, er den riktige (figur 4.24). Man kan være så uheldig at forrige soldeteksjon ligger akkurat midt i mellom de to gruppene, så da er det i dette instrumentet gjort slik at hvis dette skjer så settes den første gruppen som gjeldene, mens den siste ser vi bort i fra.



Figur 4.24: *Alogoritme for å avgjøre hvor sol er (f): To like store grupper over grensenivå, avstanden mellom de er større enn bredden på gruppene. Vi ser at avstanden C er større enn breddene A og B. Siden den siste gruppa er nærmere sist detekterte sol (forrige runde raketten var rettet mot sola, markert med stiplet blå linje), blir den første gruppa sett bort i fra, og den andre gruppa sett på som sol. Det gjør at de rød markeringene blir siste og første opplyste piksel, og den blå som sentersol.*

Hvis det er flere grupper enn to blir de to første gruppene først sammenlignet, og avgjort. Denne nye første gruppen blir så sammenlignet med neste gruppe, og så videre.

Det er fullt mulig med en algoritme som i tillegg til å se på bredden av de opplyste pikslene, ser på intensiteten. En slik algoritme har ikke blitt implementert da kompilatoren fra Altera hadde store nok problemer som det var med å kompilere den allerede implementere logikken. Det som imidlertid hadde vært mulig å implementere er en test som kontinuerlig kjører for å sjekke om man har døde piksler som gir ut høyt nivå.

4.8.4 Datainnsamling- og overføringstidskontroll

Å holde streng kontroll på alt som skjer til en hver tid er svært viktig. Det er faktisk bare 50 ns mellom utlesningen av to nabopiksler. Et timingdiagram som viser instrumentets operasjon i generelle trekk vises i figur 4.25.

Signalene i diagrammet er GATE, INTC, DVAL og DATA. GATE er signalet som angir når data skal sendes fra instrumentet og til enkoder (seksjon 4.5.1), INTC er integrasjontidskontroll, DVAL er når det er gyldige data inn til CPLD'en, og DATA er data som sendes ut fra instrumentet. Det som er viktig å legge merke til i diagrammet er at data som sendes ut fra instrumentet er 153.6 til 217.6 µs gamle (eller 0 til 64 µs hvis man skifter indeksen på dataene fra instrumentet én plass).

En annen ting som er viktig å notere seg er at integrasjonstida kan endres. Den er satt til 64 µs, men kan minskes til 0.1 µs, eller økes til 147.1 µs med 40 MHz klokkefrekvens (som instrumentet i dag bruker).

Det er også mulig å endre utlesningshastigheten til instrumentet. INTC kan endres veldig mye, så det som setter begrensningen her er utlesningen fra CMOS-sensoren. Den foregår så lenge DVAL er høy, og som vi ser av diagrammet er den lav i 51.2 µs. Allikevel så krever CMOS-sensoren en "hvileperiode" på 7.3 µs, pluss at ADC'en bruker 0.15 µs (man trenger strengt tatt ikke å vente på ADC'en og heller la den jobbe i parallell, men det kan gjøre ting mer komplisert enn nødvendig). Det gjør at man kan minske utlesningsintervallet med $(51.2 - 7.3 - 0.15)$ µs $= 43.75$ µs, og dette gir en utlesningsfrekvens på 9100 Hz, mot 6510 Hz som brukes i dag.

Det kan være mulig å øke klokkefrekvensen instrumentet jobber på fra 40 MHz til 120 MHz, og på den måten tredoble utlesningshastigheten. Dette kan imidlertid kreve et annet design av kretskortene, siden man generelt må se på ledningsbanene som transmisjonslinjer ved frekvenser over 50 - 100 MHz (Halbo & Ohlckers 1995), men dette tallet avhenger både av lengde på banene, materialet som er brukt i ledningsbanene og stige-/falltider til signalet. Mer spesifikt så kan man si at man må se på ledningsbanene som transmisjonslinjer når lengden av en komponent er større enn en tiendedel av bølgelengden til signalet (Ludwig & Bretchko 2003). Eller i symboler:

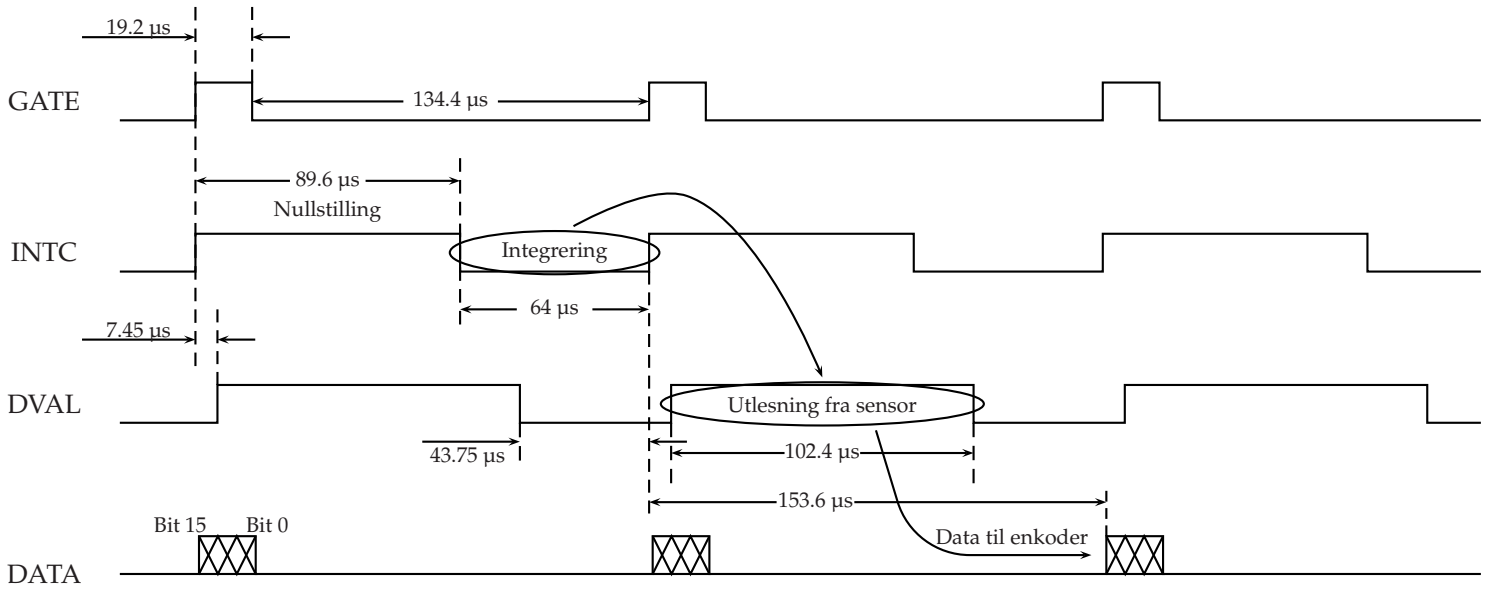
$$f = \frac{v_p}{10l} \tag{4.1}$$

der f er maksfrekvensen det er "trygt" å jobbe på, v_p er fasehastigheten i materialet, og l er lengden på komponenten. Lengden på banene i instrumentet kan vi anta at har en makslengde på 10 cm, og $v_p = 10$ cm/ns for aluminium. Det gir en frekvens på

$$f = \frac{10}{10 \cdot 10}\ \text{ns}^{-1} \tag{4.2}$$

$$= 100\ \text{MHz} \tag{4.3}$$

Vi ser at det ikke skal så mye til for å klare en klokke på 120 MHz, bare en liten reduksjon i banelengdene (banelengder på under 8.3 cm tilfredsstiller kravet).



Figur 4.25: *Timingdiagram for datainnsamling og utlesning. Det er to viktige ting å merke seg i dette timingdiagrammet. Det første er at dataene som sendes til enkoder er 153.6 til 217.6 µs gamle (eller 0 til 64 µs gamle hvis man skifter datarekka fra raketten én plass). Det andre er justeringsmulighetene: Man kan forkorte integreringstiden med inntil 64 µs, eller forlenge den med 83.1 µs (INTC må holdes høy i minst 130 klokkepulser, som er 6.5 µs ved klokkefrekvensen dette instrumentet jobber på).*

4.9 Overføringsprotokoll

Det er mye data fra dette instrumentet som er interessant å se på, spesielt med tanke på at dette er prototypeutvikling. Det er imidlertid en begrensning på at instrumentet kun kan sende 16 bit 6510 ganger i sekunder (se seksjon 4.1). 11 av disse bit'ene må brukes til å sende ned hvor solsenteret er. Man kan senke antall bit som brukes på å sende solsenteret, men da halveres oppløsningen for hvert bit som forkastes, og dette er ikke ønskelig. Man har altså 5 bit til andre data.

Andre data som kan være ønskelig å sende ned er gjennomsnitts-, maksimum- og minimumsnivået (for justering av integrasjonstiden til en senere oppskytning) eller hvor mange piksler som har et høyt intensitetsnivå. Intensitetsnivåene trenger 8 bit hver (siden ADC'en er på 8 bit), mens antall opplyste piksler trenger opp til 11. Siden instrumentet kun en brøkdel av tiden ser sola, ble det vurdert om man skulle sende disse dataene rett etter at man hadde detektert sol. Man varierer altså hva som sendes ned, og bruker de siste 5 bit'ene som kontrollsignal. Dette krever imidlertid mye logikk og minne, og gevinsten har ikke blitt sett på som stor nok til å bruke tid på dette.

De 5 siste bit'ene brukes derfor fast til å sende ned antall opplyste piksler, samt som en kontroll på at intensiteten er over grensenivået. Er de 5 bit'ene "00000" betyr dette at ingen piksel kom over intensitetsnivået. Ellers hvis de er ikke-null, så tar man binærkoden og gjør den om til desimaltall, ganger med 2 og trekker fra 1, så får man antall piksler som var opplyst. I mattematisk form:

$$N = M \cdot 2 - 1 \qquad \text{for } 1 \leq M \leq 32$$

der N er antall opplyste piksler og M er dataene man får fra instrumentet.

Det samlede bitmønsteret er altså $X_5X_4X_3X_2X_1Y_{11}Y_{10}Y_9Y_8Y_7Y_6Y_5Y_4Y_3Y_2Y_1$, der X_{5-1} er bredden av opplyste piksler og Y_{11-1} er senter av de opplyste pikslene. X_5 og Y_{11} er mest signifikante bit. Se forøvrig seksjon 4.8.3.

Kapittel 5

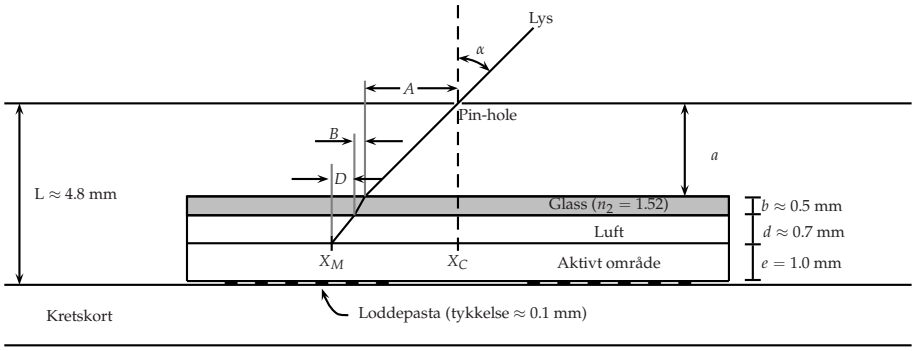
Kalibrering og testing

5.1 Kalibrering

Kalibrering av instrumentet må gjøres for å bestemme de ukjente / usikre parameterene. Grunnen til dette er at man må ha en nøyaktig modell av instrumentet for å kunne avgjøre hvilke solvinler man måler. Parameterene det er snakk om er X_C (hvilket piksel som er rett under pin-hole), b (tykkelsen på glasset i CMOS-sensoren), H (Avstanden mellom det aktive området på CMOS-sensoren og pin-hole'et, minus glasstykkelsen).

5.1.1 Modell

I figur 5.1 er det skissert en rimelig nøyaktig modell av en solstråles gang i gjennom instrumentet. Filteret, som er montert over pin-hole'et, er ikke tatt med i skissen da dette filteret ikke har noen betydning for solstrålenes gang.



Figur 5.1: *Fremstilling av lysets gang i sensoren. Merk at noen avstander er målte (markert med at de er tilnærmet lik en verdi), mens andre er i fra datablad.*

Figur 5.1 viser SLIS-2048 sensoren montert på kretskortet og pin-hole'et ovenfor. Alle avstander som er oppgitt trenger imidlertid en modifikasjon før de brukes; Pin-hole'et ligger nemlig ikke rett over det aktive område, men har en vinkel på ca $\beta = 12.25°$. Dette skyldes en misforståelse: Det aktive området på CMOS-sensoren ligger 1.32 mm vekk fra senter på brikken, og ikke midt i senter slik som det først ble forstått. Det ble forsøkt å rette opp dette ved å flytte pin-hole'et, men en perfekt justering ble det ikke. Dette gjør at høydeavstander må ganges med $1/\cos 12.25 \approx 1.023$.

Vinkelen lyset går med i glasset kan finnes fra Snells lov:

$$\begin{aligned} n_1 \sin\alpha &= n_2 \sin\theta \\ \sin\theta &= \frac{n_1 \sin\alpha}{n_2} \\ \theta &= \sin^{-1}\left(\frac{n_1 \sin\alpha}{n_2}\right) \end{aligned}$$

der n_1 er luftas brytningsindeks (≈ 1.0), n_2 er glassets brytningsindeks, α vinkelen i lufta, og θ vinkelen i glasset. Lysets treffer med en avstand fra senter på $X_M - X_C$ som er lik $A + B + D$. I fra figuren ser vi

$$\begin{aligned} X_M(\alpha) - X_C &= A + B + D \\ &= \frac{1}{\cos\beta}\left(a\tan\alpha + b\tan\theta + d\tan\alpha\right) \\ &= \frac{1}{\cos\beta}\left(H\tan\alpha + b\tan\left(\sin^{-1}\left(\frac{n_1\sin\alpha}{n_2}\right)\right)\right) \end{aligned} \tag{5.1}$$

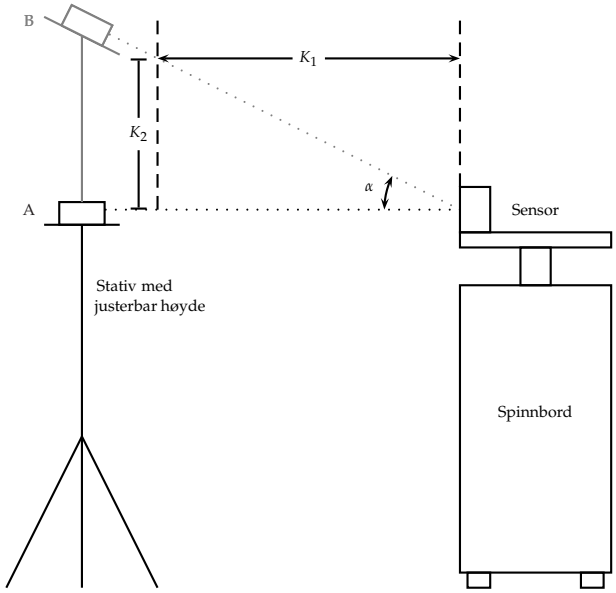
der $H = a + d$,

5.1.2 Kalibreringsoppsett

Kalibreringsoppsettet som er blitt brukt er veldig simpelt. Det ble undersøkt hva et såkalte "lav-kost" solemuleringssystem kostet, og det var snakk om fra 50 000 NOK og oppover. Dette var ikke innenfor prosjektet budsjett, og det ble derfor satt opp et eget system.

Det var vanskelig å finne en lyskilde som kan emulere sola, siden dette er en lyskilde med så og si parallelle stråler og har en åpningsvinkel på 0.53°. Det ble derfor valgt å bruke en laser (Light Amplification by Stimulated Emission of Radiation). Det ville antakelig vært optimalt å brukt en IR-laser (infrarød), siden instrumentet kun er sensitivt for IR-lys med filteret på. Isteden ble det valgt å bruke en rød laser (630 - 680 nm) og at kalibreringen skjer *uten* filteret montert. Det skyldes hovedsakelig at en rød laser var lettere tilgjengelig.

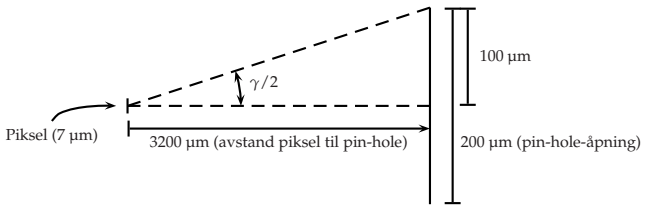
Kalibreringsoppsettet er vist i figur 5.2. Det vi ser i figuren er et stativ med justerbar høyde. På dette stativet er lyskilden (laser) montert. Denne



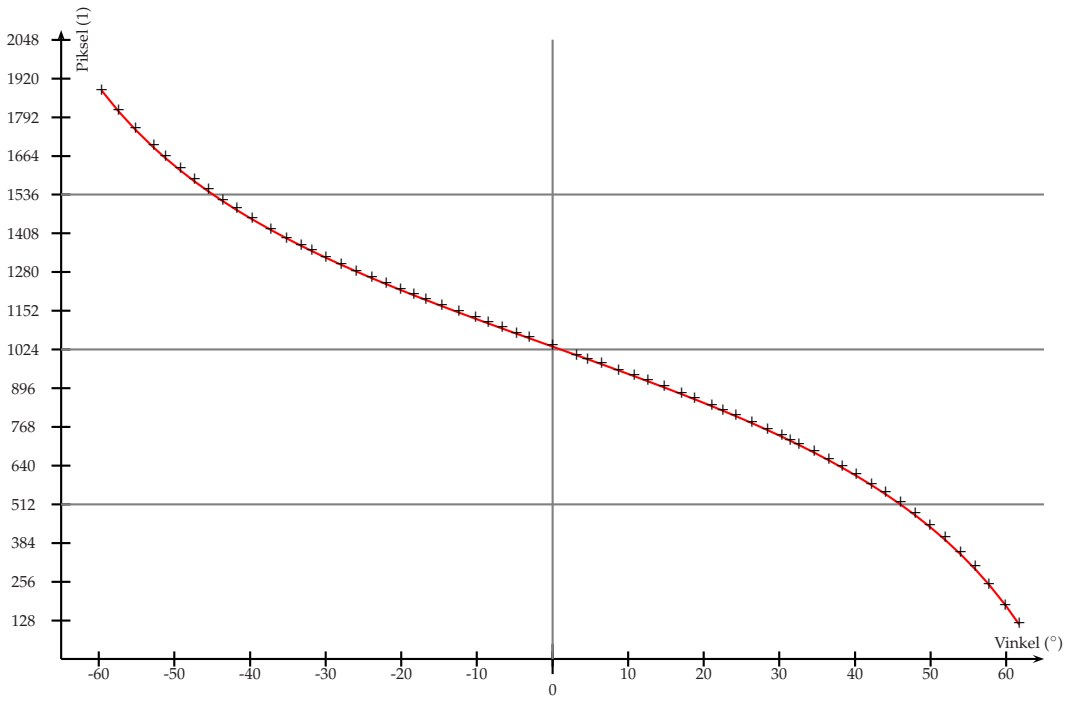
Figur 5.2: *Kalibreringsoppsett*

lyser på instrumentet som er i vater og fastmontert på et spinnbord. Når stativet er i posisjon A er laseren i vater, og man kan da lese ut nullpunktet (X_C) til sensoren. Når stativet er i posisjon B, justeres laseren slik at den treffer pin-hole'et. Så måles høyden K_2 ned til posisjon A (i vater) og avstanden K_1 til sensoren. Man kan da finne ut vinkelen som lyskilden faktisk har, og sammenligne den med data man får ut fra instrumentet. Ved å gjøre denne B målingen for veldig mange høyder får man et stort datasett til å kalibrere de to ukjente verdiene H og b i fra likning 5.1. Hvorfor det er blitt brukt et spinnbord forklares i seksjon 5.2.

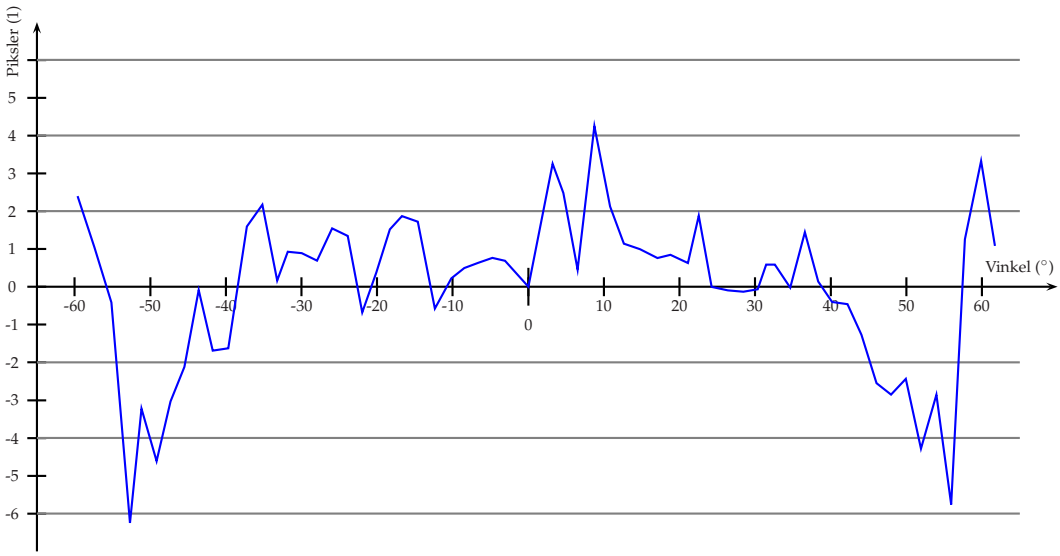
β ble funnet ved å undersøke for hvilke vinkler instrumentet detekterte sol. Resultatet ble at instrumentet detekterer sol fra 10.9°til 13.6°. Senter av dette er $\beta = 12.25°$. Grunnen til at det er såpass stor vinkel skyldes at et piksel har en synsvinkel på ca 3.6°, se figur 5.3.



Figur 5.3: *Synsvinkel til et piksel. Den totale synsvinkelen γ blir:* $\gamma = 2 \cdot \tan^{-1}(100/3200) \approx 3.6°$



Figur 5.4: *Graf over målte verdier og tilpasset kurve. De målte verdiene er markert med "+", mens den tilpassede kurven er vist med rød linje.*



Figur 5.5: *Avvik mellom målte verdier i og den tilpassede kurve.*

Et sett med måledata er vist i figur 5.4, sammen med en kurve som er tilpasset ved å bruke minste kvadraters metode for å bestemme de to ukjente H og b i likning (5.1). En kanskje mer interessant figur er figur 5.5. Den viser hvor stort avviket er mellom det tilpassede kurven og de målte dataene. Standardavviket er på 2.08 piksler, og man ser av figuren at avviket aldri er mer enn 7 piksler.

Det ble laget en MatLab-funksjon for å forenkle jobben med å regne på dataene. Dette programmet finnes i appendiks D.2, sammen med programmet for utlesning av data på RS-232 porten. Algoritmen til programmet er følgende (Golubitsky & Dellnitz 1999):

Vi antar at vi skal finne en funksjon på følgende form:

$$X_M(\alpha) = X_C + \frac{1}{\cos\beta}\left(H\tan\alpha + b\tan\left(\sin^{-1}\left(\frac{n_1\sin\alpha}{n_2}\right)\right)\right)$$

der X_C, H og b er ukjente. Vi deler da opp funksjonen i tre funksjoner for å få en lineær modell:

$$\begin{aligned} f_1(\alpha) &= 1 \\ f_2(\alpha) &= \frac{1}{\cos\beta}\tan\alpha \\ f_3(\alpha) &= \frac{1}{\cos\beta}\tan\left(\sin^{-1}\left(\frac{n_1\sin\alpha}{n_2}\right)\right) \end{aligned}$$

Måledata (målte vinkler) settes inn i funksjonene, og funksjonene settes inn i følgende matrise:

$$A = [f_1, f_2, f_3]$$

Matrisen har en størrelse på $n \times 3$, der n er antall målepunkter. Så finner vi resultatmatrisen ved å gjøre følgende operasjon:

$$\underline{\hat{X}} = \left(A^t A\right)^{-1} A^t \tilde{X}_M$$

der $\tilde{X}_M$ er en $n \times 1$ matrise med måldataene (fra instrumentet) . Da blir

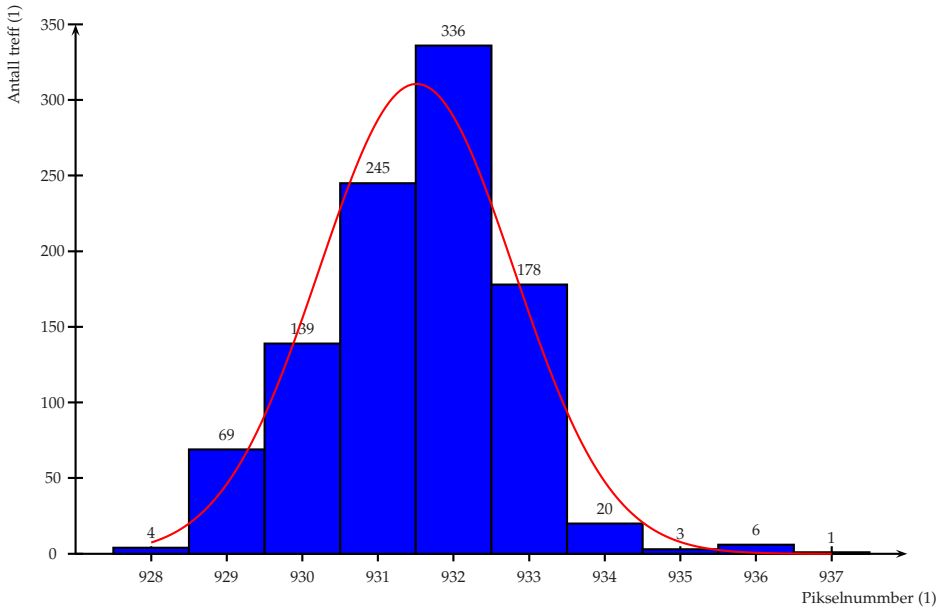
$$\begin{aligned} \hat{X}(1) &= \hat{X}_C \\ \hat{X}(2) &= \hat{H} \\ \hat{X}(3) &= \hat{b} \end{aligned}$$

Resultatet av dette programmet var at $\hat{H}$ (total lengde med luft mellom pin-hole og sensor) ble funnet til 3.2063 mm og $\hat{b}$ (tykkelsen på glasset i sensor) ble funnet til 0.4629 mm. $\hat{X}_C$ ble funnet til 1033.

5.2 Testing

Under kalibrering så var instrumentet påmontert et spinnbord. Det ble da gjort en spinntest av instrumentet, både for å se om det tålte spinnhastigheter opp på 2000°per sekund, og for å se om det opererte riktig.

Instrumentet tålte fint hastigheten, og det ble lagret serie av mottatte data via RS-232-grensesnittet når instrumentet spant med 4 rps på flere forskjellige vinkler. Ut i fra en av disse seriene er det laget et histogram som viser fordelingen av mottatte data (se figur 5.6). Standardavviket er regnet ut til 1.29 piksler, med et maks avvik på 6 piksler.



Figur 5.6: *Histogram over spinndata med normalfordelingen plottet inn* ($\mu = 931.51, \sigma = 1.29$)*. Antall treff per piksel er oppgitt, totalt 1001 treff.*

Instrumentet har også gjennomgått en ristetest. Denne ble foretatt på Forsvarets Forsknings Institutt på Kjeller. Det ble da testet med en sinus sweep på 4 g i frekvensområdet 50 Hz til 2 kHz, og en stokastisk sekvens med 6 g RMS (root means squared) i 20 sekunder. Begge testene ble gjort i alle tre akser. Testene var vellykkede, og instrumentet opererte riktig under alle testene. På samme lokasjon ble det gjennomført en varmetest, men ingen av instrumentene var i funksjon mens varmetesten pågikk. Temperaturtesten bestod av en sykling av temperaturen 0°- 70°C. Instrumentet opererte som normalt etter testen.

Vakuumtesting av instrumentet er ikke gjort.

5.3 Viktige resultater

Instrumentet har operert normalt under alle tester. Det er kalibrert og funnet at instrumentets utsignal følger likning (5.1) med $H = 3.2063$ mm, $b = 0.4629$ mm og med β målt til 12.25°.

Strømforbruket er målt til 90 mA ved 28 V, og instrumentet veier 187 gram.

Kapittel 6

Instrumentet i rakettkampanjer

6.1 IMEF/HP2-kampanjen

Denne kampanjen gikk sommeren 2005, og raketten hadde et oppskytningsvindu fra 1. til 7. juli. Med på kampanjen var FFI, Universitetet i Tromsø og Penn State University, i tillegg til Universitetet i Oslo.

Første integrering var 11. til 15. april 2005. Denne foregikk på Andøya. Her ble instrumentets digitale grensesnitt til rakettenkoder for første gang testet skikkelig. Tidligere var instrumentet kun testet mot en enkoderemulator som var konstruert ved Universitetet i Oslo. I tillegg ble det kontrollert at instrumentet faktisk passet i raketten rent mekanisk. Instrumentet fungerte som forventet, og plasseringen og mekaniske mål var korrekt.

Neste milepæl var vibrasjons- og temperaturtest på Forsvarets Forskningsinstitutt på Kjeller. Denne spente seg over fem dager, fra 9. til 13. mai. Som tidligere nevnt var alle testene vellykket for solsensoren sin del.

Siste punkt var avsluttende integrasjon og oppskytning. Avsluttende integrasjon begynte på Andøya den 23. juni. Frem til 1. juli ble raketten sammenstilt og alle instrumenter ble funksjonstestet.

Fra 1. juni var alt man ventet på gode vitenskapelige forhold og disse kom rundt halv ti den 4. juli. Raketten ble da skutt opp. Alt så ut til å gå bra frem til nesekonen skulle skytes av. På grunn av at fyringspulsene til sprengladningene ikke kom mens raketten var i lufta, ble nesekonen sittende på. Det ble derfor aldri mottatt fornuftige data fra solsensoren, siden den aldri fikk se sola. Av denne grunn finnes det ikke data å analysere, og dermed heller ikke noe endelig grunnlag for å si om instrumentet fungerer i en sonderakett.

Rakettmotoren som ble brukt er en én-trinns faststoffrakett (Improved Orion). På denne kampanjen hadde raketten en nominell høyde på 110 km.

6.2 HOTPAY1

Instrumentet skal også være med på HOTPAY1-kampanjen. Dette er en kampanje med oppskytningsvindu fra 26. juni til 7. juli 2006. Med på denne kampanjen er personer fra Leibniz Institute for Atmospheric Physics, Stockholm University, Alfred Wegener Institute for Polar and Marine Research, Imres GmbH, University of Greifswald, Bulgaria Academy of Sciences, Hebrew University of Jerusalem, Grax University og Technology, i tillegg til Universitetet i Oslo.

Det er bygget opp et nytt identisk instrument til denne kampanjen, siden IMEF/HP2 ikke var en rakett som skulle berges. Dette nye instrumentet har vært med på integrasjonstest og ristetest, men på grunn av mangel på både tid og penger har ikke oppgaveskriver hatt mulighet til å være med på disse testene. I stedet har veileder Jan Kenneth Bekkeng gjort disse testene.

Mens instrumentet ble regnet som et av eksperimentene i IMEF/HP2, er det sett som som en del av "housekeeping"-instrumenteringen i HOTPAY1. At den blir sett på som en del av housekeeping betyr at dette instrumentet er en standarddel av raketten.

Rakettmotoren i denne kampanjen er den samme som i IMEF/HP2-kampanjen, Improved Orion. Nominell høyde er også den samme.

Kapittel 7

Erfaringer, endringer og konklusjon

7.1 Kostnader

Kostnader for de elektriske delene var tilsammen ca 2300 NOK pr instrument. Produksjon av kretskortene kostet ca 1500 NOK for et sett. Aluminiumsboksen er det veldig vanskelig å avgjøre kostnadene på, men et anslag på 40 arbeidstimer er ikke så altfor galt. Pris for filter og pin-hole kom på tilsammen ca 550 NOK per sett. Tilsammen altså 4350 NOK pluss arbeidstimer.

7.2 Erfaringer og endringer

Det har vært mange "aha"-opplevelser under utviklingen av instrumentet. Spesielt under programmering av den digitale logikken, men dette var også egentlig forventet. Det er også en del andre elementer som burde vært gjort anderledes.

Et viktig element er testpinner. Det ble laget flere testpunkter på de enkelte kretskortene, men disse kan ikke brukes når kretskortene er montert inne i boksen. Det ble derimot lagt opp til svært få testlinjer ut via kontaktene. Det hadde vært svært nyttig under testing av instrumentet med flere testlinjer, og da spesielt de analoge signalene. Det er ikke ønskelig at disse er koblet til hele tiden, da dette vil føre til mye mer støy, men å kunne rute de analoge signalene ut av boksen ved behov hadde vært optimalt.

Et annet element som man bør undersøke i fremtiden er operasjonsforsterkerne. Den som er brukt i dag er har litt for lang "settling-time", og dette gjør at man kan observere litt ringing på signalene. Det har ikke vært mulig å finne noen bedre egnede operasjonsforsterkere, så det er ingen andre kretser å anbefale.

Som nevnt i seksjon 4.6.3, så er feil buffer brukt i instrumentene som er produsert. En ønskelig endring er å bruke riktig buffer, slik at man garantert unngår problemer. Det er også mulig å sette på pull-up-motstander på utgangen av bufferet, men dette er ikke en spesielt pen løsning, da det blant annet blir flere komponenter.

En komponent som kan være greit å bytte ut er RS-232 kretsen som er brukt. Den har en overføringsrate på 115200 baud. Dette er for lite til å kunne overføre samme mengde data som overføres til rakettenkoder. Det holder kun til halvparten. Det finnes kretser som er pinnekompatible med MAX202 som er brukt i dag, men med mulighet for å jobbe på høyere frekvenser. En av disse kan benyttes hvis man ønsker å få like stor båndbredde på RS232 som via datakanalen til enkoder.

Det er som tidligere nevnt brukt feil krystalloscillator. 3.3 V utgaven har samme pinnekonfigurasjon som den som er brukt, slik at det bare er å fjerne motstanden mellom CPLD og krystalloscillatoren, og bruke 3.3 V inngansspenning, så kan man benytte riktig klokke.

Et lite irritasjonselement er at bildesensoren ikke er plassert eksakt under pin-hole'et. Man kan løse dette "problemet" ved å enten flytte sensoren eller pin-hole'et.

Et svakt punkt ved instrumentet i dag er at instrumentet gjør signalbehandling ombord i raketten før data sendes ned *uten* at det er mulig å verifisere at signalbehandlingen har gjort jobben riktig. Dette er spesielt uheldig siden dette er en prototype. Man bør derfor sende ned alle rådata fra ca 1 sekund av flighten, men det er ikke teknisk mulig i dag på grunn av liten båndbredde.

En siste endring som bør gjøres er å benytte 4-lagskort i stedet for 2-lags som benyttes i dag. Dette gjør det vanskeligere å feilsøke, men gevinsten man får er såpass stor at det er verdt det: Mindre støy, bedre routing og kanskje viktigst av alt; instrumentet kan ta mindre plass.

7.3 Konklusjon

Instrumentet virker bra på bakken slik det er i dag. Det har enda ikke vært mulig å få en verifikasjon på at det opererer riktig under oppskytning, da den første raketten ikke fungerte riktig. Endelig bekreftelse om dette instrumentet opererer som tiltenkt i en sonderakett får man (forhåpentligvis) en måned etter at denne masteroppgaven er levert.

Bibliografi

Altera (2006), *MAX II Device Handbook*, Altera, California, USA.
*http://www.altera.com

Analog Devices (1996), *AD8029/AD8030/AD8040, Low Power, High Speed Rail-to-Rail Input / Output Amplifier*, Analog Devices, Maine, USA.
*http://www.analog.com/UploadedFiles/Data_Sheets/41420698100982AD8029_30_40_a.pdf

Bekkeng, Jan Kenneth (2002), Prototyputvikling av e-felt eksperiment for små sonderaketter, Master's thesis, Universitetet i Oslo.

Bentley, John P. (1995), *Principles of measurement systems*, 3 edn, Pearson Education Limited.

C-MAC (2005), *CFPS72*, C-MAC, Somerset, Storbritania.
*http://www.cmac.com/mt/databook/oscillators/smd/spxo/cfps-72,-73.pdf

Edmund Optics Ltd (2004), *Optics and Optical Instruments Catalog*, Edmund Optics Ltd, York, England.

Golubitsky, Martin & Michael Dellnitz (1999), *Linear Algebra and Differential Equation using Matlab*, Broooks / Cole Publishing Company.

Grødal, Agnar (1997), *Elektromagnetisk kompatibilitet for konstruktører*, Tapir forlag.

Halbo, Leif & Per Ohlckers (1995), *Electronic components, packaging and production*, Norwegian Metrology and Accreditation Service.

Heald, Mark A. & Jerry B. Marion (1995), *Classical Electromagnetic Radiation*, 3 edn, Saunders college publishing.

Helvajian, Henry, ed. (1999), *Microengineering aerospace systems*, The Aerospace Press, California.

Hewlett Packard (1996), *High CMR, High Speed TTL Compatible Optocouplers*, Hewlett Packard, USA.
*http://www.farnell.com/datasheets/4564.pdf

ITW McMurdo Connectors (2004), *Micronector 200*, ITW McMurdo Connectors, Hampshire, Storbritania.
*http://www.elfa.se/pdf/43/04395018.pdf

Larsen, May Aimee, Bent Chr. Samuelsen & Jon Thoresen (2001), Solsensor, Master's thesis, Høgskolen i Narvik.

Ludwig, Reinhold & Pavel Bretchko (2003), *RF Circuit Design, Theory and Applications*, Prentice-Hall Inc.

Maxim Integrated Products (1996), *+5V RS-232 Transceivers with 0.1 µF External Capacitors*, Maxim Integrated Products, California, USA.
*http://www.elfa.se/pdf/73/730/07302359.pdf

Maxim Integrated Products (2003), *Low-Power, Slew-Rate-Limited RS-485/RS-422 Transcievers*, Maxim Integrated Products, California, USA.
*http://pdfserv.maxim-ic.com/arpdf/MAX1487-MAX491.pdf

Nakamura, Junichi, ed. (2006), *Image sensors and signal processing for digital still cameras*, CRC Press.

On-Trak Photonics Inc (2004), *Position Sensing Detectors Theory Of Operation*, On-Trak Photonics, Lake Forest, California.
*http://www.on-trak.com/pdfs/PSDTO.pdf

Ott, Henry W. (2001), 'Partitioning and layout of a mixed-signal pcb', *Printed Circuit Design* .
*http://www.hottconsultants.com/pdf_files/june2001pcd_mixedsignal.pdf

Panavision Imaging (2005), *SLIS - 2048 High Speed 2048 x 1 Line Scan Image Sensor*, Panavision Imaging, Homer, New York, USA.
*http://www.panavisionsvi.com/PDS0005E.pdf

Phillips Semiconductors (1998), *74ABT244, Octal buffer/line driver (3-State)*, Phillips Semiconductors, California, USA.
*http://www.semiconductors.philips.com/acrobat/datasheets/74ABT244_2.pdf

Phillips Semiconductors (2006), *74AHC244; 74AHCT244, Octal buffer/line driver; 3-State*, Phillips Semiconductors, California, USA.
*http://www.semiconductors.philips.com/acrobat/datasheets/74AHC_AHCT244_4.pdf

Power-one (1999), *IMX4 DCDC Series Data Sheet*, Power-one, California, USA.
*http://www.powerone.com/resources/products/datasheet/imx4.pdf

Samtec (2001), *Qstrip Factsheet*, Samtec.
*http://www.elfa.se/pdf/43/04369302.pdf

Skahill, Kevin (1996), *VHDL for Programmable Logic*, Addison-Wesley Publishing.

Texas Instruments (1999), *TLC5540, 8-bit high-speed analog-to-digital converter*, Texas Instruments, Texas, USA.
*http://www.elfa.se/pdf/73/733/07339682.pdf

Texas Instruments (2002), *Fast-Transient-Response 1-A Low-Dropout Voltage Regulators*, Texas Instruments, Texas, USA.
*http://www.elfa.se/pdf/73/733/07330608.pdf

Weckler, Gene P. (1967), 'Operation of p-n junction photodetectors in a photon flux integrating mode', *IEEE Journal of solid-state circuits* **sc-2**(3).

Wertz, James R., ed. (1978), *Spacecraft attitude determination and control*, D. Reidel.

Tillegg A

Forkortelser

ABT Advanced Bi-CMOS (Bi-CMOS logikk-familie, kombinasjon av TTL og CMOS)

AC Alernating Current

AHCT Advanced High-Speed CMOS logic (CMOS logikk-familie)

ADC Analog to Digital Converter (analog til digital omformer)

ASIC Application Spesific Integrated Circuit

bps bit per sekund

CCD Charge-Coupled Decice

CMOS Complementary Metal-Oxide-Semiconductor

CPLD Complex Programmable Logic Device

DAC Digital to Analog Converter (digital til analog omformer)

DC Direct Current

FOV Field Of View (synsfeltet)

FPGA Field Programmable Gate Array

HF Høyfrekvent / High Frequency

HP Hotel Payload

IC Integrated Circuit (Integrert krets)

ICI Investigation of Cusp Irregularities (rakett)

IR Infrarød

JTAG Joint Test Action Group

LAB Logic Array Blocks

laser Light Amplification by Stimulated Emission of Radiation

LED Light Emitting Diode (Lysemitterende diode)

LF Lavfrekvent / Low Frequency

LSB Least Signifikant Bit (minst signifikante bit)

MSB Most Signifikant Bit (mest signifikante bit)

MSPS Megasamples per second (Megasampler per sekund)

PCB Printed Circuit Board (kretskort)

PSD Position Sensing Detector

rps runder per sekund

RTV Room Temperature Vulcanization (festemiddel)

SMD Surface Mounted Devices (overflatemonterte komponenter)

SOIC Small-Outline Integrated Circuit

SRADS Sounding Rocket Attitude Determination System (orienteringsbestemmelsessystem for sonderaketter)

TQFP Thin Quad Flat Pack

TTL Transistor-Transistor Logic

VHDL (VHSICHDL) Very High Speed Integrated Circuit Hadware Description Language

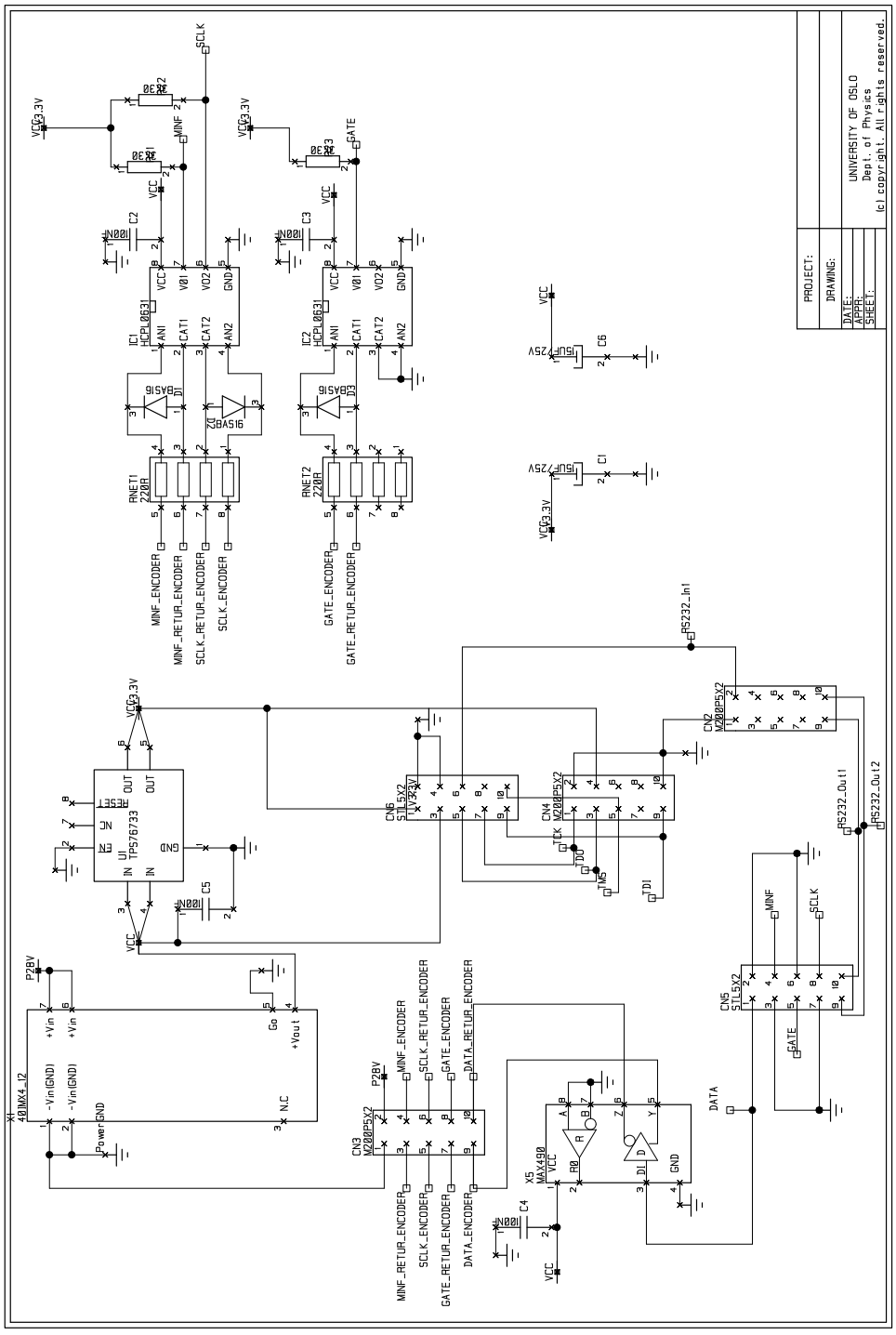
Tillegg B

Kretskortskjemaer og utlegg

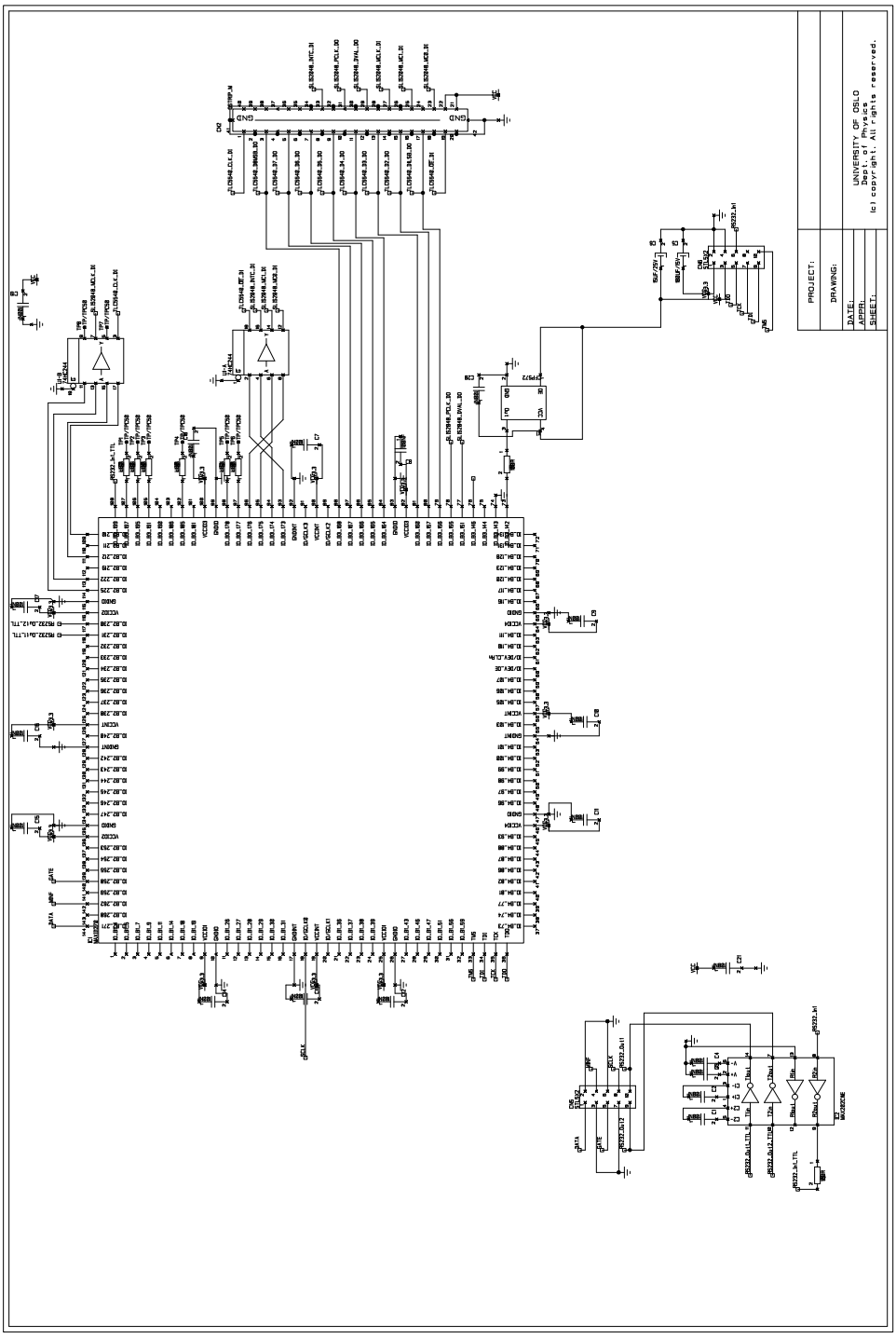
På de neste sidene er de elektriske skjemategningene og kretskortutleggene gjengitt. Disse er tegnet i Zuken CadStar 7.0 og 8.0.

Det er gitt to skjemaer over sensorkortet. Den første (figur B.3) er skjemategningen som opprinnelig var tegnet. Den andre (figur B.4) viser endringene som er gjort på kortene før de har blitt tatt i bruk. Kretskortutlegget er basert på den første.

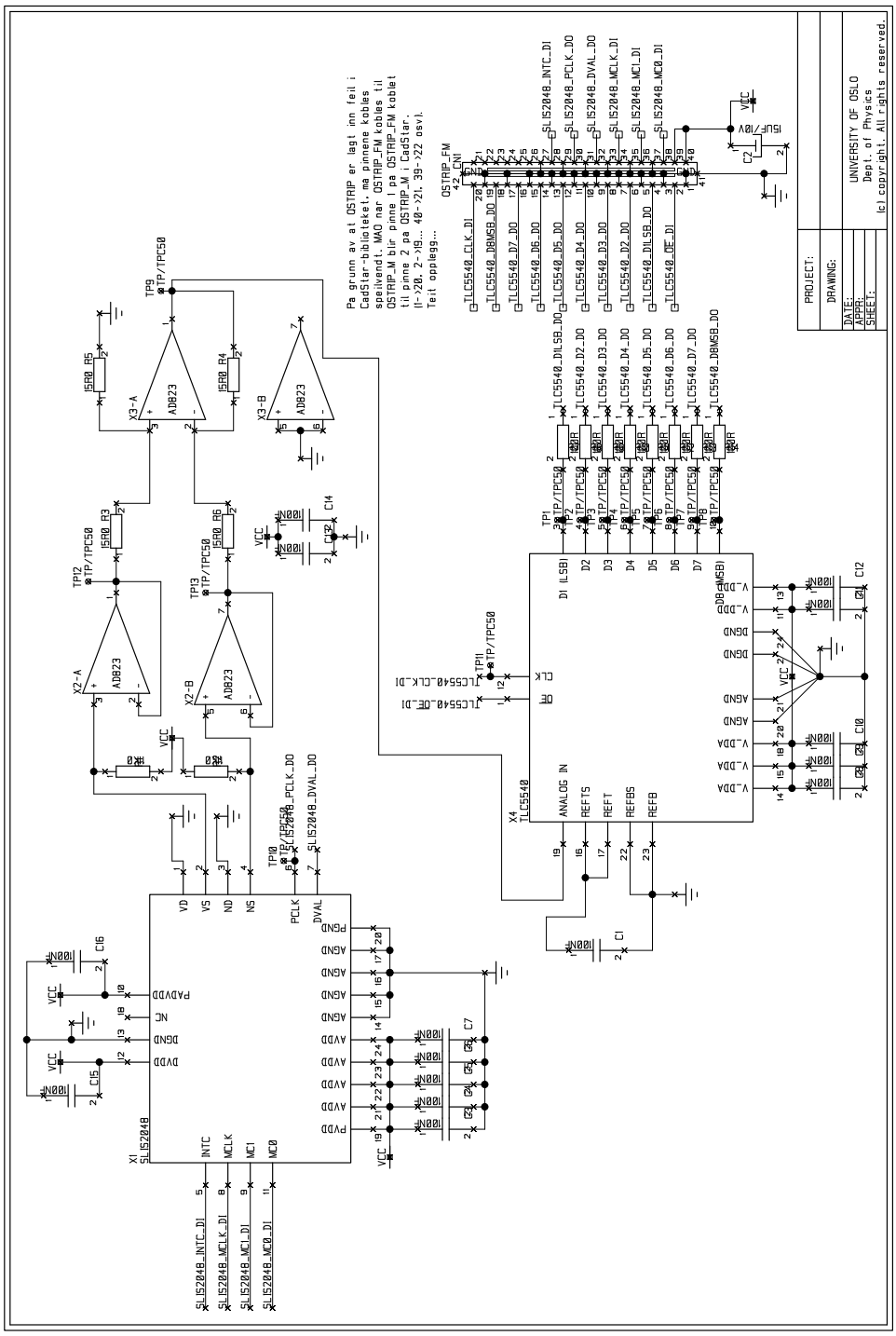
Alle filene som har vært brukt i forbindelse med produksjon av kretskortene finnes på http://folk.uio.no/martiso/solconcepta/cadstar.zip.



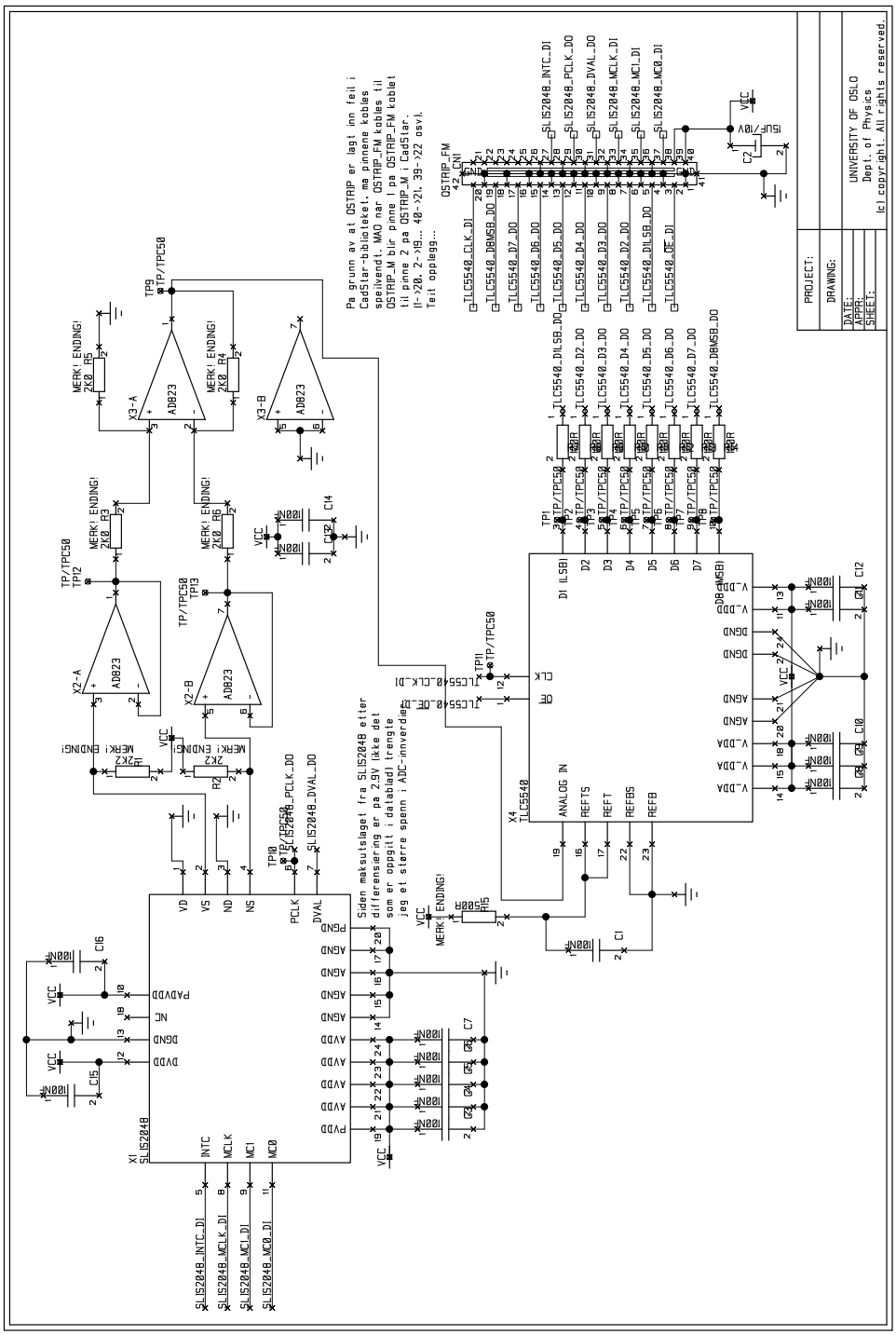
Figur B.1: *Skjemategning over powerkortet.*



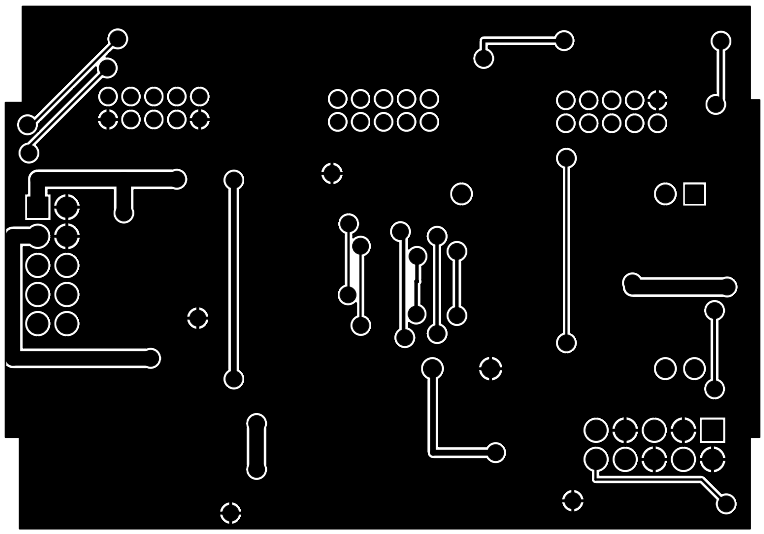
Figur B.2: *Skjemategning over databehandlingskortet.*



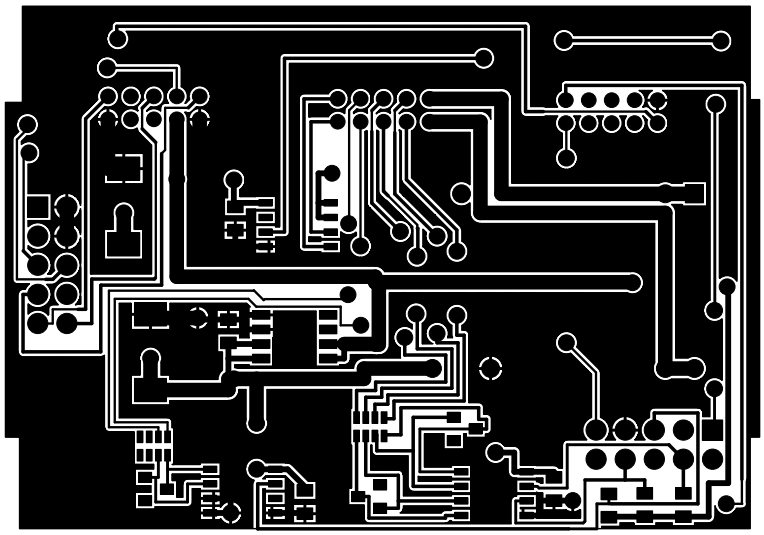
Figur B.3: *Skjemategning over sensorkortet.*



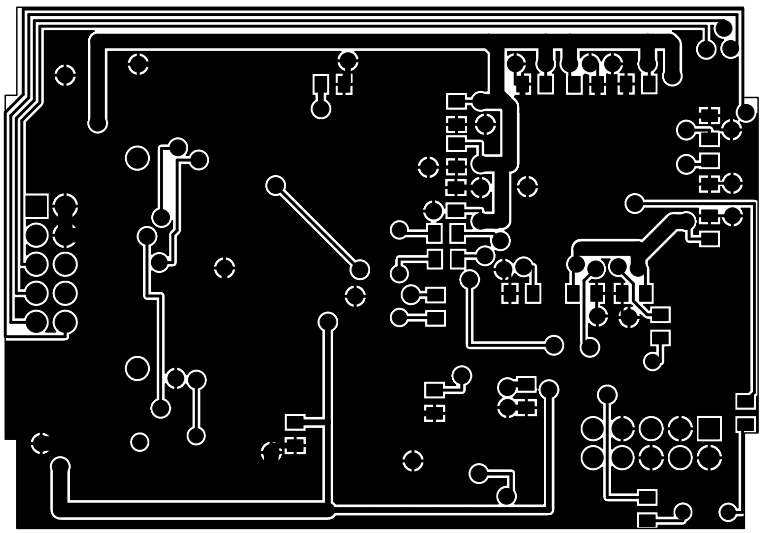
Figur B.4: *Skjemategning over sensorkortet etter endringer.*



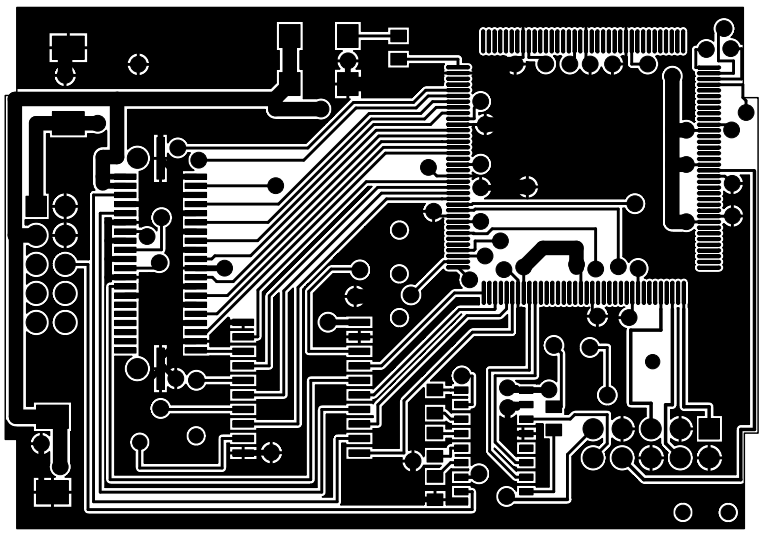
Figur B.5: *Utlegg av powerkortet, bunnelektrisk.*



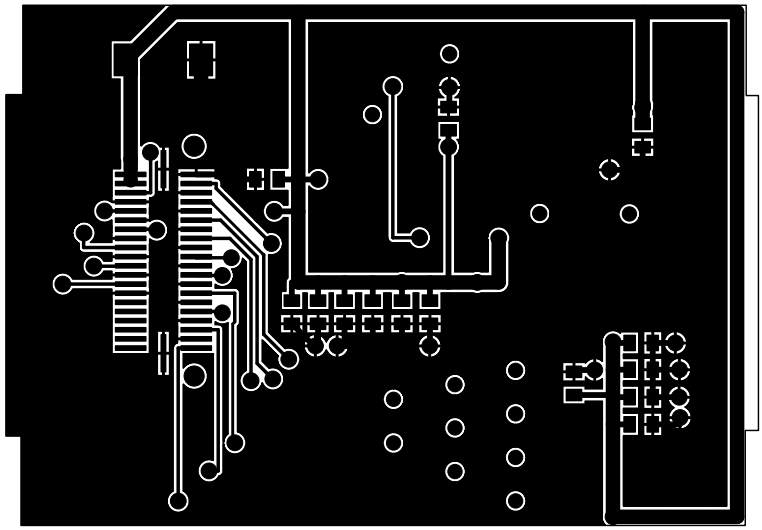
Figur B.6: *Utlegg av powerkortet, toppelektrisk.*



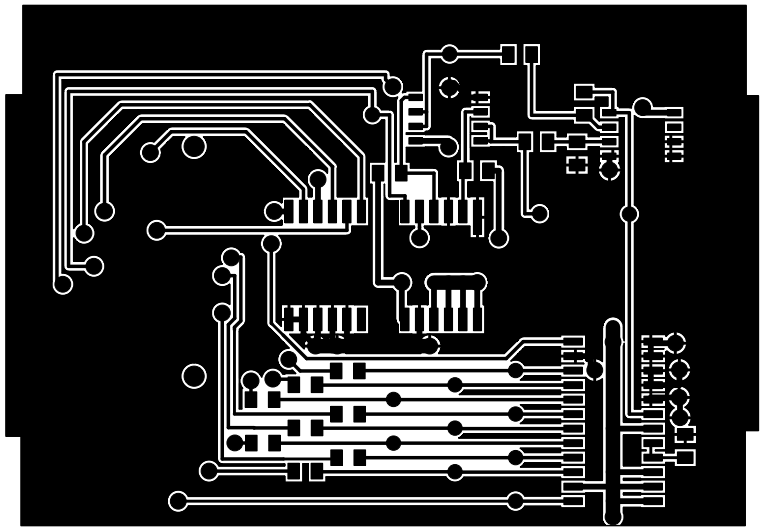
Figur B.7: *Utlegg av databehandlingskortet, bunnelektrisk.*



Figur B.8: *Utlegg av databehandlingskortet, toppelektrisk.*



Figur B.9: *Utlegg av sensorkortet, bunnelektrisk.*



Figur B.10: *Utlegg av sensorkortet, toppelektrisk.*

Listing: *Deleliste til powerkort.*

Parts List

CADSTAR Design Editor Version 8.0

Part Name/Number	Description	Qty.	Comps.
E-65-766-31	10% 50V 0805 X7R	4	C2-5
E-43-954-30	5X2 PINS MALE HORIZ. PCB-CONNECTOR	3	CN2-4
E-43-704-33	5X2 SCOTT ELEC. PINROW	2	CN5-6
E-70-300-34	DIODE SOT23	3	D1-3
E-73-228-29	FULL DUPLEX TRANCIEVER	1	X5
OPTO/HCPL0631/SMD	HIGH SPEED OPTOCOUPLER	2	IC1-2
F-909-040	DC-DC CON 4W VOUT +12V	1	X1
E-60-176-10	RESISTOR KOA 0805 1% 0.125W	3	R21-23
RESNET/220R/SMD	RESISTOR ARRAY 1206	2	RNET1-2
E-73-306-73	Voltage regulator 3.3V 1A	1	U1
E-67-731-88	TANTAL ELECTROLYTIC CAP LOW ESR	2	C1 C6

End of report

Listing: *Deleliste til databehandlingskort.*

Parts List

CADSTAR Design Editor Version 8.0

Part Name/Number	Description	Qty.	Comps.
ALTERA/MAXII1270/TQFP	ALTERA MAX II 1270 CPLD	1	IC1
E-65-766-31	10% 50V 0805 X7R	19	C1-4 C7-21
E-43-693-02	2X20 PINS MALE C2C-CONNECTOR 5MM CARDSPACING 300MHz	1	CN2
E-43-704-33	5X2 SCOTT ELEC. PINROW	2	CN5-6
E-73-904-12	OCTAL LINE DRIVER/RECEIVER	1	U1
X-XX-XXX-XX	DUAL RS-232 TX/RX	1	IC2
E-60-172-55	RESISTOR KOA 0805 1% 0.125W	8	R1-8
E-67-730-89	TANTAL ELECTROLYTIC CAP LOW ESR	2	C5
E-67-731-88	TANTAL ELECTROLYTIC CAP LOW ESR	2	C6
F-4246019	Crystall-Oscillator 1 - 160 MHz	1	X1

Listing: *Deleliste til sensorkort.*

Parts List

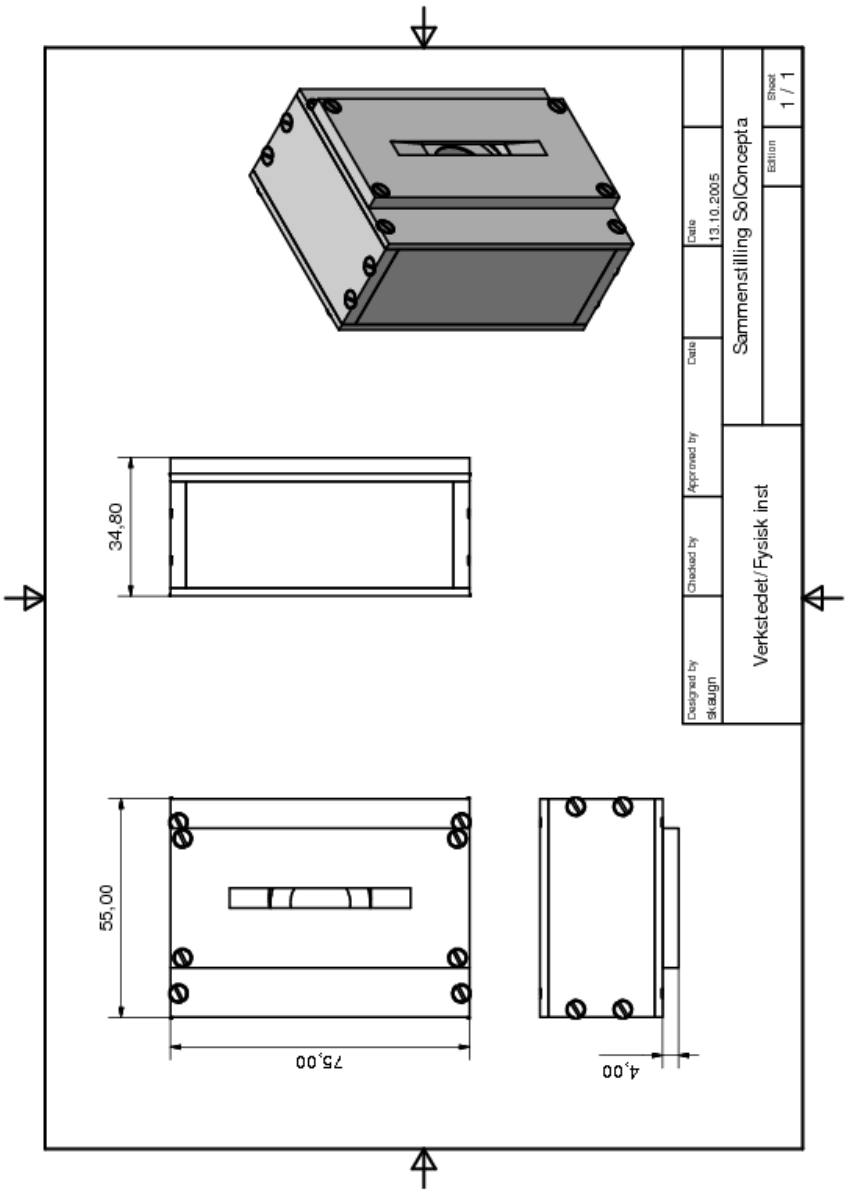
CADSTAR Design Editor Version 8.0

Part Name/Number	Description	Qty.	Comps.
E-65-766-31	10% 50V 0805 X7R	15	C1 C3-16
E-43-693-44	2X20 PINS FEMALE C2C-CONNECTOR 5MM CARDSPACING 300MHz	1	CN1
X-XX-XXX-XX	DUAL OPAMP	2	X2-3
E-60-172-55	RESISTOR KOA 0805 1% 0.125W	8	R7-14
E-60-175-78	RESISTOR KOA 0805 1% 0.125W	3	R1-2
E-60-175-60	RESISTOR KOA 0805 1% 0.125W	4	R3-6
E-60-174-20	RESISTOR KOA 0805 1% 0.125W	4	R15
SPES/SLIS2048/SMD	HIGH SPEED 2048x1 LINE SCAN IMAGE SENSOR	1	X1
E-73-396-82	ADC 8-Bit, high-speed, pipelined	1	X4
E-67-731-88	TANTAL ELECTROLYTIC CAP LOW ESR	1	C2

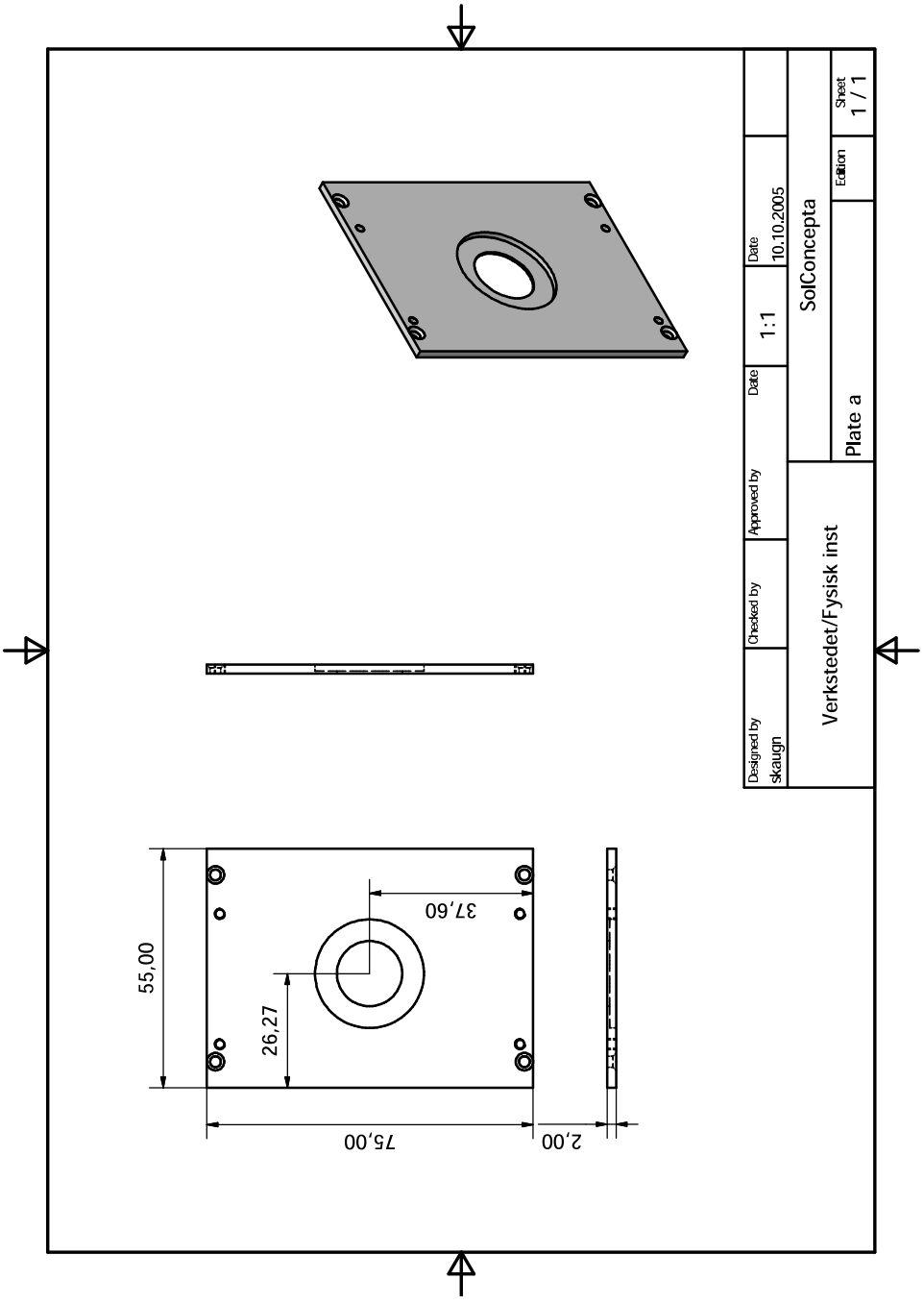
Tillegg C

Mekanisk boksutlegg

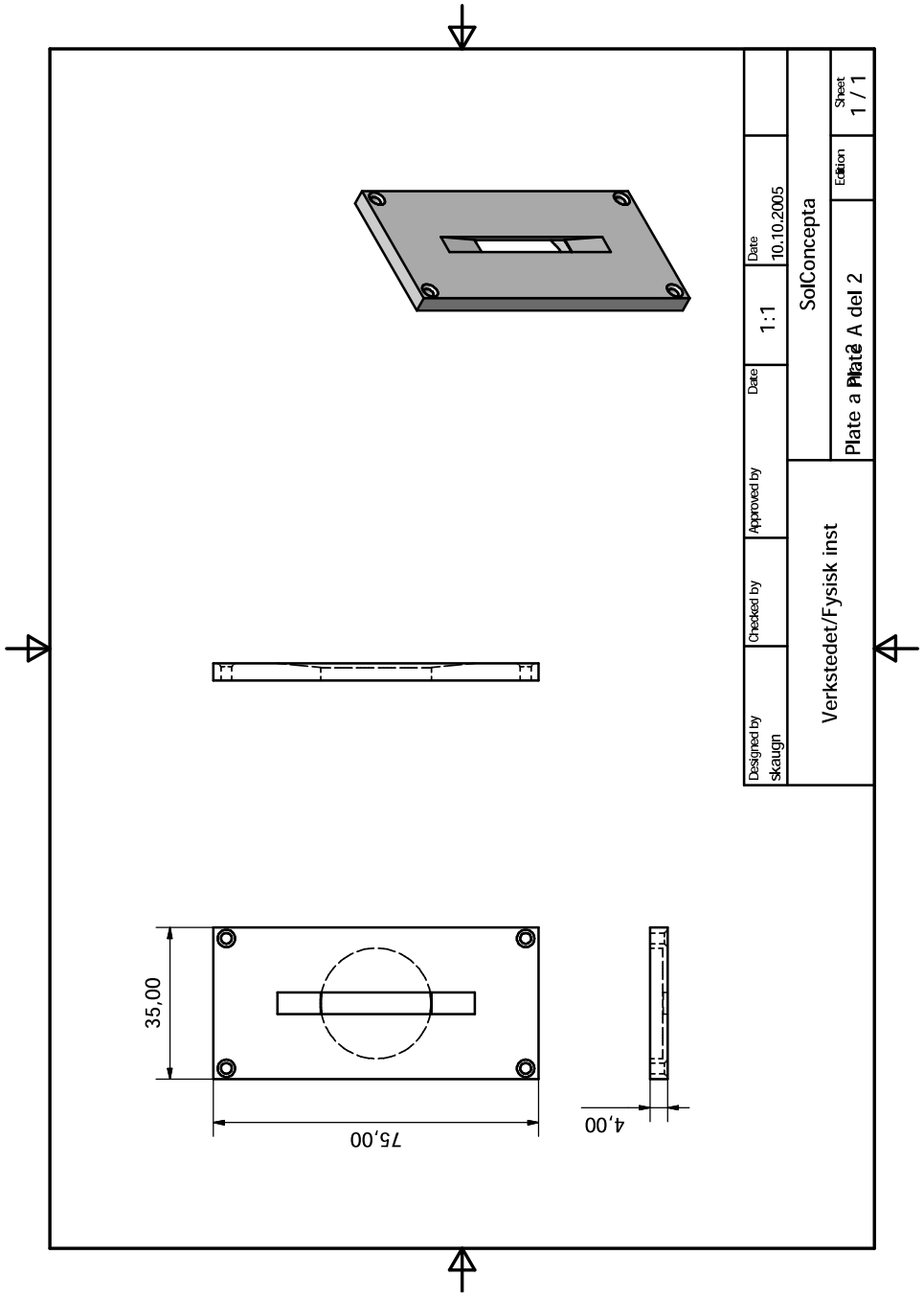
På de neste sidene er aluminiumsboksens konstruksjonstegninger gjengitt. Steinar Nilsen på Fysisk Institutt ved Universitetet i Oslo har tegnet disse.



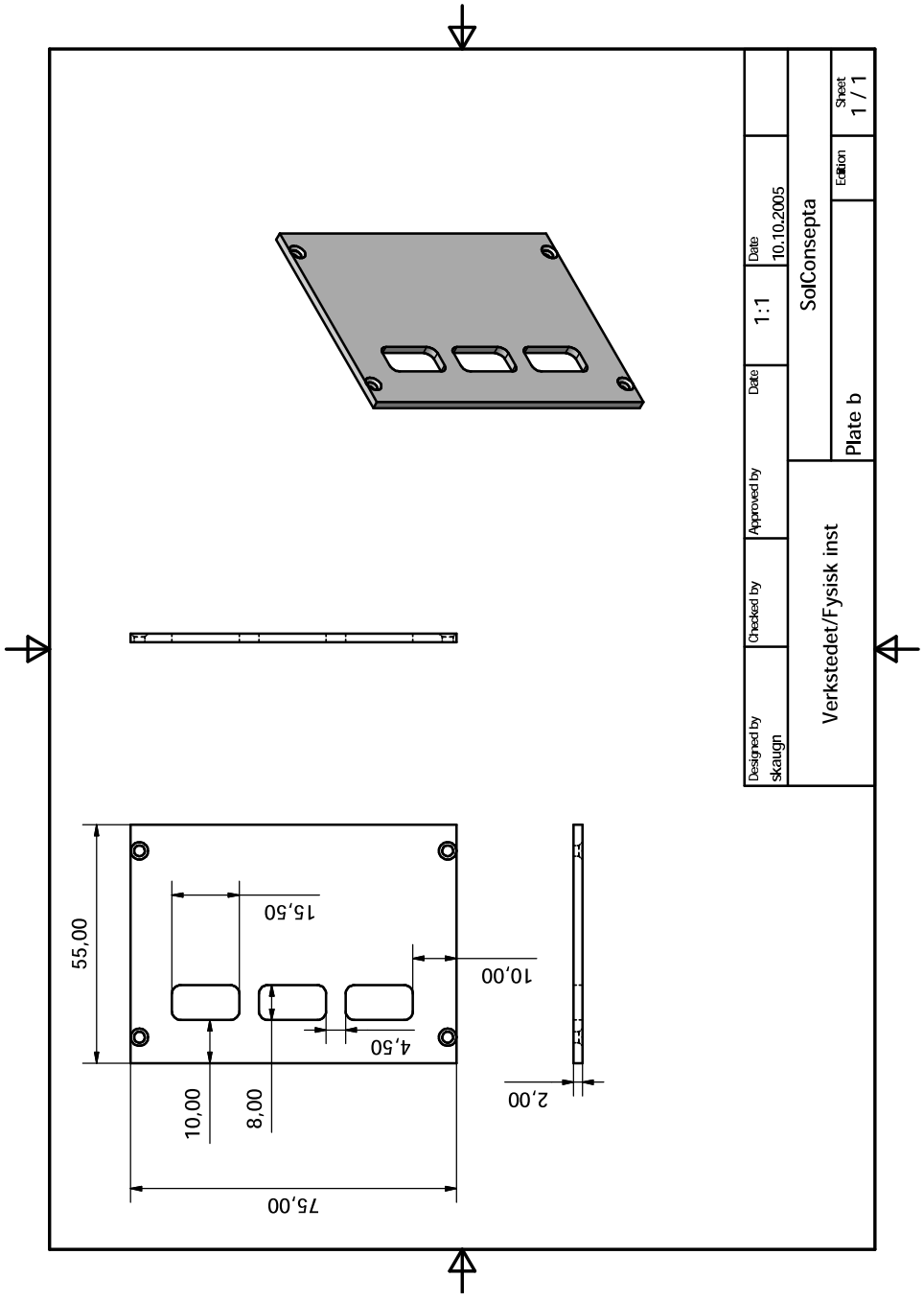
Figur C.1: *Boksen sammensatt.*



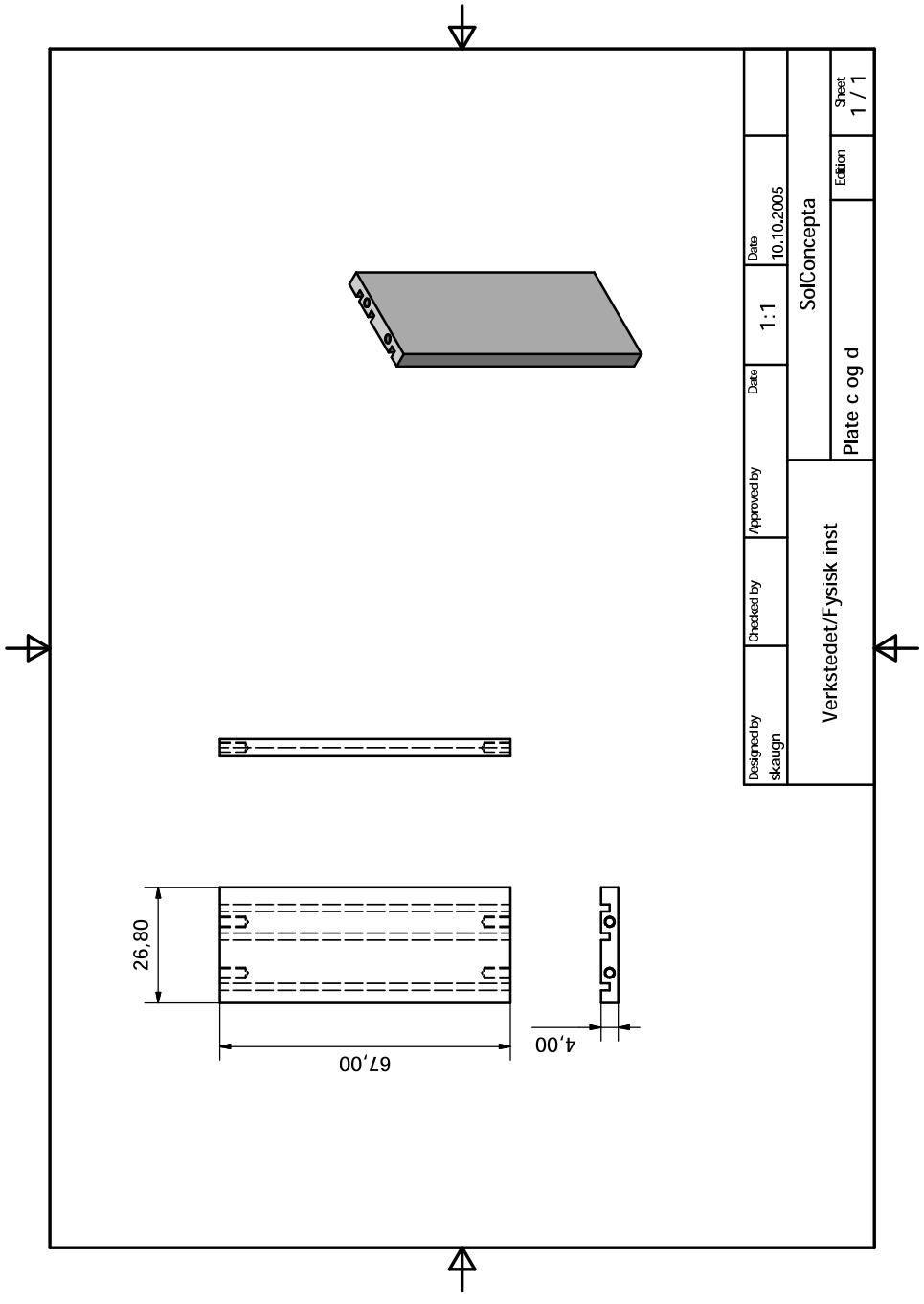
Figur C.2: *Nedre topplate.*



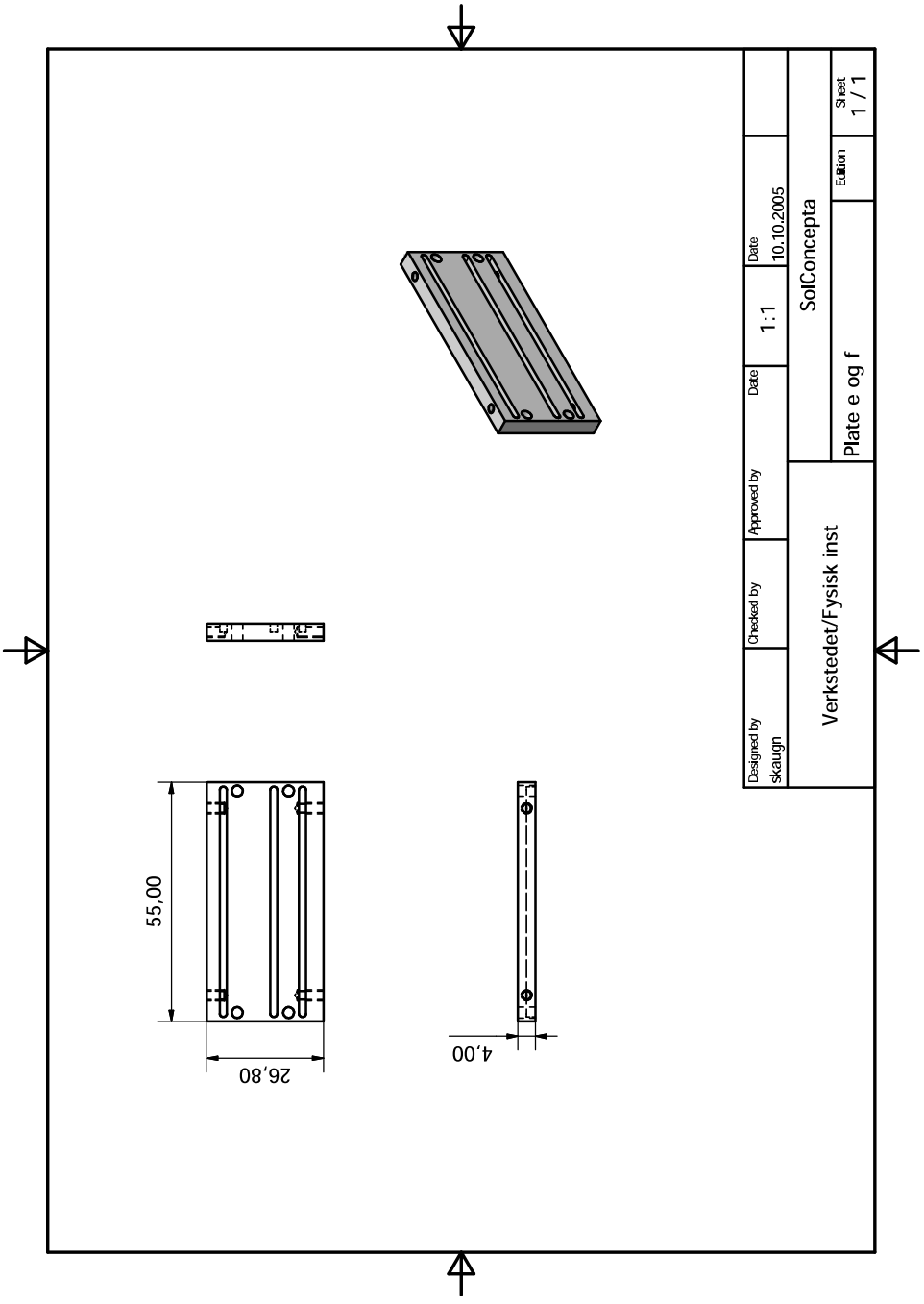
Figur C.3: *Øvre topplate.*



Figur C.4: *Bunnplate. Dersom kontaktene skal tas ut på siden skal firkatene ikke skjæres ut.*



Figur C.5: *Lange sideplater. Dersom kontaktene skal tas ut på siden, må det skjæres ut hull til dette på en av disse platene.*



Figur C.6: *Korte sideplater.*

Tillegg D

Programkode

D.1 VHDL-kode

VHDL-koden er simulert og kompilert i Altera Quartus II 5.1 Web Edition. Filene listet under, i tillegg til alle filene som trengs for å kompilere hele prosjektet finnes på http://folk.uio.no/martiso/solconcepta/vhdl.zip. Det er kommentert i toppen av hver fil hva den gjør.

Listing: *Soltiming.vhd.*

```
-------------------------------------------------------------------------------
--Filnavn                : Soltiming.vhd
--Påbegynt dato          : 21.03.2005
--Siste oppdatering      : 18.04.2006
--Av                     : Martin Sollien
--Kort beskrivelse       : Toppfila til SolConcepta-prosjektet. Holder kontrollen på
--                         timing av SLIS2048, TLC5540 og Solfinner.vhd.
-------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;


entity Soltiming is

  port (
    -----Globale inn og utsignaler------------------------------------------------
    CLK_I      : in  std_logic;          -- Hovedklokke. 40MHz
    TP1_O      : out std_logic;          -- Testpunkt 1
    TP2_O      : out std_logic;          -- Testpunkt 2
    TP3_O      : out std_logic;          -- Testpunkt 3
    TP4_O      : out std_logic;          -- Testpunkt 4
    TP5_O      : out std_logic;          -- Testpunkt 5
    TP6_O      : out std_logic;          -- Testpunkt 6

    -----Encodertilkobling--------------------------------------------------------
    SCLK_N_I   : in  std_logic;          -- Klokke fra encoder. 833,28 kHz
    MINF_N_I   : in  std_logic;          -- MINF fra encoder 1085 frames/s
```

```
    GATE_N_I   : in  std_logic;          -- GATE fra encoder. Går høy når jeg
                                         -- skal sende data
    -----SLIS2048 I/O-----------------------------------------------------------
    S_INTC_O   : out std_logic;          -- integrasjonstid for SLIS2048
    S_MCLK_O   : out std_logic;          -- Klokke til SLIS2048
      -- S_MC1_O og S_MC0_O er feil i koden lengre ned, men siden jeg har gjort
      -- feil med navngivningen i CadStar retter disse to feilene seg mot
      -- hverandre. . .
    S_MC1_O    : out std_logic;          -- Utlesningstype til SLIS2048
    S_MC0_O    : out std_logic;          -- Utlesningstype til SLIS2048
    S_PCLK_I   : in  std_logic;          -- Pixelklokke fra SLIS2048
    S_DVAL_I   : in  std_logic;          -- Datavalid-signal fra SLIS2048
    -----TLC5540 I/O------------------------------------------------------------
    T_OE_N_O   : out std_logic;          -- Output Enable til TLC5540. Aktiv lav
    T_CLK_O    : out std_logic;          -- Klokke til TLC5540
    T_DATA_I   : in  std_logic_vector(8 downto 1);
                                         -- Digitalisert signal fra TLC5540
    -----Solbuffer I/O----------------------------------------------------------
    SB_DATA_O : out std_logic;           -- Seriell øverføring til MAX490 og
                                         -- encoder
    -----Solfinner I/O----------------------------------------------------------

    -----RS232 I/O--------------------------------------------------------------
    RX_I       : in std_logic;           -- RS232-inngang
    TX_O       : buffer std_logic;       -- RS232-utgang
    TX2_O      : buffer std_logic        -- RS232-utgang 2
    );


end Soltiming;

architecture letthefunbegin of Soltiming is

  component Solfinner
    port (
      T_DATA_I       : in  std_logic_vector(8 downto 1);
                                         -- Det digitaliserte signalet fra
                                         -- SLIS2048
      T_CLK_I        : in  std_logic;    -- Klokka til TLC5540, brukes til å
                                         -- avgjør når data fra TLC5540 er
                                         -- gyldige
      SF_TMP_O       : out std_logic_vector(7 downto 0);  -- Testverdi
      SB_DATA_O      : out std_logic_vector(15 downto 0);
                                         -- Data som skal sendes ut, og legges i
                                         -- SolBuffer
      GATE_I         : in  std_logic;    -- Signal fra encoder som angir når
                                         -- SolConcepta kan sende
      CLK_I          : in  std_logic;    -- Hovedklokke. 40MHz
      S_DVAL_I       : in  std_logic;    -- Fra SLIS2048
      LRESET_I       : in  std_logic     -- Hvis Solfinner skal resettes.

    );

  end component;

  component Solbuffer
    port (
      SB_DATA_I      : in  std_logic_vector(15 downto 0);
                                         -- Digitalisert signal fra TLC5540
      SCLK_I         : in  std_logic;    -- Klokke fra encoder
      GATE_I         : in  std_logic;    -- Angir at man skal sende
      SB_DATA_O      : out std_logic     -- datalinje ut til MAX490
```

```
  );

end component;

component rs232
  generic (

    CLKTIMES_G     : integer range 2 to 6  := 4;
                                    -- Hvor mange ganger høyere frekvensen
                                    -- er inn enn RS232 skal være. Bør være
                                    -- satt til minst 16 ganger, altså 4.
                                    -- (2^CLKTIMES) = antall gangner

    ANTBIT_G       : integer range 5 to 8 := 8;
                                    -- Antall bit som RS232 er satt på.
                                    -- Default 8 bit
    ANTSTOPBIT_G   : integer range 1 to 2 := 1;
                                    -- Hvor mange stoppbit:
                                    -- ANTSTOPBIT_G = 1 tilsvarer 1 bit.
                                    -- ANTSTOPBIT_G = 2 tilsvarer 2 bit.
                                    -- Har ikke støtte for 1.5 stoppbit.
    PARITYCHECK_G : integer range 1 to 5 := 1
                                    -- Paritetsbitsjekk:
                                    -- 1 => Ingen sjekk
                                    -- 2 => Odde bit (Odd)
                                    -- 3 => Partalls bit (even)
                                    -- 4 => Høy (Mark)
                                    -- 5 => Lav (Space)
    );
  port (
    CLK_I          : in    std_logic;   -- Klokke. X-dobbel frekvens i forhold
                                        -- til RS232klokka
    RESET_I        : in    std_logic;   -- Global reset, aktiv lav
    TX_O           : out   std_logic;   -- Sende-kanalen
    RX_I           : in    std_logic;   -- Mottager-kanalen

    -- Datalinjer
    DATA_I         : in std_logic_vector(ANTBIT_G - 1 downto 0);
                                        -- Datakanal inn
    DATA_O         : out std_logic_vector(ANTBIT_G - 1 downto 0);
                                        -- Datakanal ut

    -- Handshake
    DATASEND_I     : in  std_logic;     -- Det som ligger DATA_I skal sendes
    SERIAL_RDY_O   : out std_logic;     -- Klar til ny sending
    DATARECV_O     : out std_logic;     -- Data er mottat og ligger klar på
                                        -- DATA_O
    ERRORRECV_O    : out std_logic      -- Feil under mottagning av data
    );
end component;

signal SCLK_S         : std_logic;      -- SCLK fra encoder
signal MINF_S         : std_logic;      -- MINF fra encoder
signal GATE_S         : std_logic;      -- GATE fra encoder
signal SB_DATA_S      : std_logic_vector(15 downto 0);
                                        -- Kobling mellom solfinner og
                                        -- solbuffer
signal T_CLK_S        : std_logic;      -- Klokke til TCL5540
signal SF_RESET_S     : std_logic;      -- Hvis Solfinner skal resettes
signal ST_RESET_S     : std_logic;      -- Reset til Soltiming
signal CLK_HALVERT_S : std_logic;       -- 20MHz klokke (samme som S_PCLK_I)
```

```
signal INT_TID_S      : std_logic_vector(12 downto 0);
                                    -- Integrasjonstid for SLIS2048.
                                    -- Bestemmer intensiteten.
                                    -- VELDIG VIKTIG: Denne bestemmer hvor
                                    -- lenge S_INTC_O skal holdes høy, slik
                                    -- at jo større INT_TID_S er jo kortere
                                    -- er integrasjonstida! (Hadde kanskje
                                    -- vært riktigere å kalle denne
                                    -- S_RESETTID, da den henviser på hvor
                                    -- lenge ladningskondensatorene i
                                    -- SLIS2048 holdes i resettilstand.)
                                    -- Merk: Denne er på 13 bit (et mer enn
                                    -- antall piksler) og det er fordi
                                    -- pikselklokka er halvparten av den
                                    -- globale klokka.

signal INT_TID_TELLER_S : std_logic_vector(12 downto 0);
                                    -- Teller som angir hvor lenge S_INTC_O
                                    -- er blitt holdt høy.

signal STARTVENT_S : std_logic;     -- Handshake til holdeteller
signal STARTVENTLENGENOK_S : std_logic_vector(5 downto 0);
                                    -- Teller i holdeteller som angir hvor
                                    -- lenge vi skal vente før vi starter
                                    -- ny utlesningssyklus


type SLISTILSTAND_T is (venting, start, integrering, tull);
                                    -- Tilstandsmaskin for SLIS2048 som
                                    -- setter i gang ny avlesning med mer
attribute enum_encoding : string;
attribute enum_encoding of SLISTILSTAND_T : type is "gray";
-- Å velge gray-encoding viste seg å være helt nødvendig for at kretsen
-- skulle virke. Kom veldig ofte inn i en udefinert tilstand ved å bruke
-- default encoding (one-hot).

signal SLISTILSTAND_S : SLISTILSTAND_T;

--RS232 signaler ------------------------------------------------------------
signal RS232RESET_S : std_logic;    -- Reset til RS232
signal RS232DRCV_S  : std_logic_vector(7 downto 0);
                                    -- Data som mottas på RX_I kommer hit,
                                    -- kan leses ut når RS232RX_OK_S ='1'
signal RS232RX_OK_S : std_logic;    -- Ny data ligger på RS232DRCV_I
signal RS232DSEND_S : std_logic_vector(7 downto 0);
                                    -- Data som skal sendes på TX_O. Leses
                                    -- av rs232 når RS232TXNOW_S = '1'
signal RS232TXNOW_S : std_logic;    -- Send det som ligger på datalinja
signal RS232RDY_S   : std_logic;    -- RS232 er klar for ny sending
signal RS232ERRTX_S : std_logic;    -- RS232 har mottat noe, men feil

signal RS232CLKDIV_S : std_logic_vector(3 downto 0); -- RS232klokkedeler
signal RS232CLK_S    : std_logic;   -- RS232klokke
signal SF_TMP_S      : std_logic_vector(7 downto 0); -- testsignal

signal RS232DSEND2_S : std_logic_vector(7 downto 0);
                                    -- Data som skal sendes på TX_O. Leses
                                    -- av rs232 når RS232TXNOW_S = '1'
signal RS232TXNOW2_S : std_logic;   -- Send det som ligger på datalinja
signal RS232RDY2_S   : std_logic;   -- RS232 er klar for ny sending
```

```
  signal SF_TMP2_S      : std_logic_vector(7 downto 0);  -- testsignal
  signal RS232DATAHOLDER_S : std_logic_vector(15 downto 0);  -- testsignal
  signal RS232VELGER_S : std_logic;      -- testsignal
  signal RS232DATATELLER_S : integer range 0 to 31;


  signal testeteller : std_logic_vector(4 downto 0);

  signal S_INTC_S : std_logic;          -- testsignal

begin  -- letthefunbegin

  --Klokker-----------------------------------------------------------------------
  S_MCLK_O <= CLK_I;
  T_CLK_S  <= S_PCLK_I;
  T_CLK_O  <= T_CLK_S;
  --Klokker slutt-----------------------------------------------------------------

  SCLK_S   <= not SCLK_N_I;
  MINF_S   <= not MINF_N_I;
  GATE_S   <= not GATE_N_I;

  T_OE_N_O <= '0';

  TP1_O    <= GATE_S;
  TP6_O    <= S_INTC_S;

  RS232RESET_S  <= '1';
  RS232TXNOW_S  <= GATE_S;                -- For å kunne se på hva som sendes ut
  RS232TXNOW2_S <= GATE_S;
  SF_RESET_S    <= '0';
  ST_RESET_S    <= '0';

  u1 : Solfinner port map (
    T_DATA_I      => T_DATA_I,
    T_CLK_I       => T_CLK_S,
    SB_DATA_O     => SB_DATA_S,
    GATE_I        => GATE_S,
    CLK_I         => CLK_I,
    SF_TMP_O      => SF_TMP_S,
    S_DVAL_I      => S_DVAL_I,
    LRESET_I      => SF_RESET_S);

  u2 : Solbuffer port map (
    SB_DATA_I     => SB_DATA_S,
    SCLK_I        => SCLK_S,
    GATE_I        => GATE_S,
    SB_DATA_O     => SB_DATA_O);

  u3 : rs232 port map (
    CLK_I         => RS232CLK_S,
    RESET_I       => RS232RESET_S,
    TX_O          => TX_O,
    RX_I          => RX_I,
    DATA_O        => RS232DRCV_S,
    DATA_I        => RS232DSEND_S,
    DATASEND_I    => RS232TXNOW_S,
    SERIAL_RDY_O  => RS232RDY_S,
    DATARECV_O    => RS232RX_OK_S,
    ERRORRECV_O   => RS232ERRTX_S);

  u4 : rs232 port map (
```

```
    CLK_I          => RS232CLK_S,
    RESET_I        => RS232RESET_S,
    TX_O           => TX2_O,
    RX_I           => '1',
    DATA_I         => RS232DSEND2_S,
    DATASEND_I     => RS232TXNOW2_S,
    SERIAL_RDY_O => RS232RDY2_S);



-- purpose: Styre timingen til SLIS2048
-- type   : sequential
-- inputs : CLK_I, GATE_S, INT_TID_S, STARTVENTLENGENOK_S
-- outputs: STARTVENT_S, S_INTC_O, S_MC1_O, S_MC0_O
slisTiming: process (CLK_I, GATE_S)
begin  -- process slisTiming
  if GATE_S = '1' then                  -- asynchronous reset (active high)
    S_MC0_O           <= '1';
    S_MC1_O           <= '0';
    SLISTILSTAND_S    <= venting;
    INT_TID_TELLER_S <= (others => '0');
    TP2_O         <= '0';
    TP3_O         <= '0';
    TP4_O         <= '0';
    TP5_O         <= '1';
    STARTVENT_S <= '0';
    S_INTC_O      <= '1';
    S_INTC_S      <= '1';

  elsif CLK_I'event and CLK_I = '1' then  -- rising clock edge
    S_MC0_O       <= '1';
    S_MC1_O       <= '0';
    TP2_O         <= '0';
    TP3_O         <= '0';
    TP4_O         <= '0';
    TP5_O         <= '0';

    case SLISTILSTAND_S is
      when venting =>
        TP2_O <= '1';
        STARTVENT_S <= '1';
        if STARTVENTLENGENOK_S > "001110" then  -- vente 16 klokkesykler på
                                                -- SCLK_S for å lese ut dataene
                                                -- nærmest mulig når jeg skal
                                                -- sende
          SLISTILSTAND_S <= start;
        end if;



      when start =>
        STARTVENT_S <= '1';             -- TEST! Holder denne høy for å være
                                        -- sikker på at ALT i denne prossessen
                                        -- får med seg at vi skal gå til neste
                                        -- tilstand.
        TP3_O <= '1';
        S_INTC_O <= '1';
        S_INTC_S <= '1';

        if INT_TID_S < INT_TID_TELLER_S then
          SLISTILSTAND_S <= integrering;
        end if;
```

```
          INT_TID_TELLER_S <= INT_TID_TELLER_S + 1;


        when integrering =>
          TP4_O <=  '1';
          S_INTC_O <= '0';
          S_INTC_S <= '0';

        when others =>
          SLISTILSTAND_S <= start;
          -- Var problemer med å bruke one-hot-encoding. Bruker derfor nå gray-
          -- encoding og for å være sikker på at jeg ikke havner i en udefinert
          -- tilstand så har jeg med denne.

      end case;
    end if;
  end process slisTiming;

   INT_TID_S <= "0100000000000";

  -- purpose: Utføre kommandoer fått i fra RS232
  -- type   : sequential
  -- inputs : RS232DRCV_S, RS232RX_OK_S
  -- outputs: aa
-------------------------------------------------------------------------------
-- VELDIG VIKTIG Å FJERNE DETTE FØR LAUNCH!!!!!!!
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--      rs232tolker: process (RS232RX_OK_S, RS232RESET_S)
--      begin  -- process rs232tolker
--        if RS232RESET_S = '0' then
--          INT_TID_S <= "0100000000000";
--        elsif RS232RX_OK_S'event and RS232RX_OK_S = '1' then  -- fått ny data
--          case RS232DRCV_S is

--            ---------------------------------------------------------------------
--            -- Husk at ordene doble, halvere, øke og minke henviser til INT_TID_S,
--            -- slik at integrasjosntida ikke dobles, men resettida.


--            when "00101010" =>              -- '*' (42)
--              -- "Doble" integrasjonstida, effektivt: Halvere resettida
--              if INT_TID_S < "0001001011000" then
--                INT_TID_S <= "0000100101100";       -- 150 (Må minimum holde
--                                                    -- S_INTC_O høy i 146
--                                                    -- klokkesykler)
--              else
--                INT_TID_S <= '0' & INT_TID_S(12 downto 1);
--              end if;


--            when "00101011" =>              -- '+' (43)
--              -- Øke integrasjonstida med "6,25% av maks"
--              if INT_TID_S < "0001100101100" then
--                INT_TID_S <= "0000100101100";       -- 150 (Må minimum holde
--                                                   -- S_INTC_O høy i 146
--                                                   -- klokkesykler)
--              else
--                INT_TID_S <= INT_TID_S - "0001000000000";
--              end if;
```

```
--            when "00101101" =>                -- '-' (45)
--              -- Senke integrasjonstida med "6,25% av maks"
--              if INT_TID_S > "0111100100100" then
--                INT_TID_S <= "1000100100100";       -- 2190 (Kan maks holde S_INTC_O
--                                                    -- høy i 2194 klokkesykler)
--              else
--                INT_TID_S <= INT_TID_S + "00010000000";
--              end if;


--            when "00101111" =>                -- '/' (47)
--              -- "Halvere" integrasjonstida, effektivt: Doble resettida
--              if INT_TID_S > "0100010010010" then
--                INT_TID_S <= "1000100100100";      -- 2190 (Kan maks holde
--                                                   -- S_INTC_O høy i 2194
--                                                   -- klokkesykler)
--              else
--                INT_TID_S <= INT_TID_S(11 downto 0) & '0';
--              end if;


--            when others =>

--          end case;
--        end if;
--    end process rs232tolker;



  -- purpose: Telle opp slik at SLIS2048 venter med å begynne på riktig
  --          tidspunkt
  -- type   : sequential
  -- inputs : SCLK_S, STARTVENT_S
  -- outputs: STARTVENTLENGENOK_S
  holdeteller: process (SCLK_S, STARTVENT_S)
  begin  -- process holdeteller
    if STARTVENT_S = '0' then           -- asynchronous reset (active low)
      STARTVENTLENGENOK_S <= (others => '0');

    elsif SCLK_S'event and SCLK_S = '1' then  -- rising clock edge
      STARTVENTLENGENOK_S <= STARTVENTLENGENOK_S + 1;
    end if;
  end process holdeteller;

  -- purpose: Dele ned klokka til RS232
  -- type   : sequential
  -- inputs : CLK_I
  -- outputs: RS232CLK_S
  RS232klokkedeler: process (CLK_I)
  begin  -- process RS232klokkedeler
    if CLK_I'event and CLK_I = '1' then  -- rising clock edge
      if RS232CLKDIV_S = "1010" then
        RS232CLKDIV_S <= (others => '0');
        RS232CLK_S <= not RS232CLK_S;
      else
        RS232CLKDIV_S <= RS232CLKDIV_S + 1;
      end if;
    end if;
  end process RS232klokkedeler;
```

```
  -- purpose: Lage en klokke som er på halve frekvensen av GATE_I
  -- type    : sequential
  -- inputs  : GATE_I
  -- outputs: RS232VELGER_S
  GATEneddeller: process (GATE_S)
  begin  -- process GATEneddeller
    if GATE_S'event and GATE_S = '0' then  -- falling clock edge
      RS232VELGER_S <= not RS232VELGER_S;
    end if;
  end process GATEneddeller;


  -- purpose: til testformål
  -- type    : sequential
  -- inputs  : RS232VELGER_S
  -- outputs:
  RS232dataplukker: process (RS232VELGER_S, testeteller)
  begin  -- process RS232dataplukker
    if RS232VELGER_S'event and RS232VELGER_S = '1' then
      if testeteller = "11111" then
        RS232DATAHOLDER_S <= "1110011111111100";
        testeteller <= "00000";
      else
        RS232DATAHOLDER_S <= SB_DATA_S;
        testeteller <= testeteller + 1;
      end if;

    end if;
  end process RS232dataplukker;

  -- purpose: til testformål
  -- type    : combinational
  -- inputs  : RS232VELGER_S
  -- outputs:
  RS232sendevelger: process (RS232VELGER_S, testeteller, RS232DATAHOLDER_S)
  begin  -- process RS232sendevelger
    if RS232VELGER_S = '1' then
      RS232DSEND_S    <= RS232DATAHOLDER_S(15 downto 8);
      RS232DSEND2_S   <= RS232DATAHOLDER_S(15 downto 8);
    else
      RS232DSEND_S    <= RS232DATAHOLDER_S(7 downto 0);
      RS232DSEND2_S   <= RS232DATAHOLDER_S(7 downto 0);
    end if;

  end process RS232sendevelger;

end letthefunbegin;
```

Listing D.1: *Solfinner.vhd*

```
-------------------------------------------------------------------------------
--Filnavn            : Solfinner.vhd
--Påbegynt dato      : 23.03.2005
--Siste oppdatering  : 20.04.2006
--Av                 : Martin Sollien ( martiso@fys.uio.no )
--Kort beskrivelse   : Mottar data fra ADC (TLC5540) og avgjør om det er blitt
--                     detektert sol.
-------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity Solfinner is

  port (
    T_DATA_I        : in  std_logic_vector(8 downto 1);
                                        -- Det digitaliserte signalet fra
                                        -- SLIS2048
    T_CLK_I         : in  std_logic;    -- Klokka til TLC5540, brukes til å
                                        -- avgjør når data fra TLC5540 er
                                        -- gyldige
    SF_TMP_O        : out std_logic_vector(7 downto 0); -- Testverdi
    SB_DATA_O       : out std_logic_vector(15 downto 0);
                                        -- Data som skal sendes ut, og legges i
                                        -- SolBuffer
    GATE_I          : in  std_logic;    -- Signal fra encoder som angir når
                                        -- SolConcepta kan sende
    CLK_I           : in  std_logic;    -- Hovedklokke. 40MHz
    S_DVAL_I        : in  std_logic;    -- Signal fra SolTiming som angir at nå
                                        -- begynner en ny utlesning
    LRESET_I        : in  std_logic;    -- Hvis Solfinner skal resettes.
    RSS_B           : buffer std_logic_vector(5 downto 0)
                                        -- RSS = Runder siden sol. Angir hvor
                                        -- mange runder SLIS2048 har gått siden
                                        -- sist gang intensiteten var over
                                        -- grenenivået. Kun lagt på i
                                        -- tillfellet man ønsker en grenseverdi
                                        -- på sol som varierer (automatic
                                        -- gain).
    );

end Solfinner;

architecture FinnerDuNoe of Solfinner is

  signal NIVAA_S                : std_logic_vector(8 downto 1);
                                        -- Angir hva som er nåværende nivå for
                                        -- soldeteksjon
  signal SOLREKKE_S             : std_logic_vector(2047 downto 0);
                                        -- Rekke som angir hvilke plasser i
                                        -- sensoren det er detektert at sola er

  type INNLESNINGTILSTAND_T is (idle, startstraks, ingenSol, aktivSol,
                                etterSol, avslutt);
                                        -- Tilstandsmaskin for innlesning av
                                        -- data
  attribute enum_encoding : string;
  attribute enum_encoding of INNLESNINGTILSTAND_T : type is "gray";
 -- Å velge gray-encoding viste seg å være helt nødvendig for at kretsen
```

```
  -- skulle virke. Kom veldig ofte inn i en udefinert tilstand ved å bruke
  -- default encoding (one-hot).

  signal INNLESNINGTILSTAND_S          : INNLESNINGTILSTAND_T;
  signal INNLESNINGTILSTAND_NESTE_S    : INNLESNINGTILSTAND_T;

  signal VENTETELLER_S           : integer range 0 to 255;

  signal DATAPEKER_S             : std_logic_vector(10 downto 0);
  signal STARTTELLER_S           : std_logic_vector(4 downto 0);

  signal SENTERSOLPEKER_S        : std_logic_vector(10 downto 0);
                                       -- Senter, opplyste piksler
  signal SOLBREDDE_S             : std_logic_vector(4 downto 0);
                                       -- Bredde på sola som er over NIVAA_S

  signal STARTSOLPEKER_S         : std_logic_vector(10 downto 0);
                                       -- Første opplyste piksel i
                                       -- nåværende deteksjon
  signal SLUTTSOLPEKER_S         : std_logic_vector(10 downto 0);
                                       -- Siste opplyste piksel i
                                       -- nåværende deteksjon
  signal STARTSOL_S              : std_logic;
                                       -- (Ny) Sol detektert
  signal STARTSOLPEKER_2_S       : std_logic_vector(10 downto 0);
                                       -- Første opplyste piksel i
                                       -- første deteksjon
  signal SLUTTSOLPEKER_2_S       : std_logic_vector(10 downto 0);
                                       -- Siste opplyste piksel  i
                                       -- første deteksjon
  signal SLUTTSOL_S              : std_logic;
                                       -- Har detektert sol, men ikke
                                       -- nå lengre

  signal MAXINTENSITET_S         : std_logic_vector(7 downto 0);
                                       -- testsignal som skal brukes
                                       -- for å finne maxintensitet
                                       -- (for kalibrering)
--  signal MININTENSITET_S         : std_logic_vector(7 downto 0);
                                       -- testsignal som skal brukes
                                       -- for å finne minintensitet
                                       -- (for kalibrering)

  signal SOLPEKERSISTRUNDE_S : std_logic_vector(10 downto 0);
                                       -- Hvor senter lå siste runde

  signal STARTPIKSLER_S : std_logic_vector(4 downto 0);
                                       -- Antall piksler offset vi får på
                                       -- grunn av at så mange piksler er
                                       -- dårlige.



begin  -- FinnerDuNoe

  -- Ikke i bruk i mitt nåværende design
  RSS_B    <= (others => '0');
  SF_TMP_O <= (others => '0');


  NIVAA_S         <= "01111111"; -- over 50% av saturation er sol
```

```
  STARTPIKSLER_S <= "01111";     -- 16 rare piksler på starten


  -- purpose: Bytte tilsander
  -- type    : sequential
  -- inputs : T_CLK_I
  -- outputs: INNLESNINGTILSTAND_S
  datainnlewsningTilsand: process (T_CLK_I, INNLESNINGTILSTAND_NESTE_S)
  begin  -- process datainnlewsningTilsand
    if T_CLK_I'event and T_CLK_I = '1' then  -- rising clock edge
      INNLESNINGTILSTAND_S <= INNLESNINGTILSTAND_NESTE_S;
    end if;
  end process datainnlewsningTilsand;

  -- purpose: Leser inn data på riktig tidspunkt og avgjør om det er sola eller
  --          ikke
  -- type    : sequential
  -- inputs : T_CLK_I, T_DATA_I
  -- outputs: SOLREKKE_S
  datainnlesning: process (T_CLK_I, INNLESNINGTILSTAND_S,
                           S_DVAL_I, STARTTELLER_S, T_DATA_I)
  begin  -- process datainnlesning
    if T_CLK_I'event and T_CLK_I = '0' then  -- falling clock edge

      case INNLESNINGTILSTAND_S is

        when idle =>
          if S_DVAL_I = '1' then
            INNLESNINGTILSTAND_NESTE_S <= startstraks;
          else
            INNLESNINGTILSTAND_NESTE_S <= idle;
          end if;

          STARTTELLER_S <= (others => '0');


        when startstraks =>

          SLUTTSOL_S         <= '0';
          STARTSOL_S         <= '0';
          SLUTTSOLPEKER_S    <= (others => '0');
          STARTSOLPEKER_S    <= (others => '0');
          SLUTTSOLPEKER_2_S <= (others => '0');
          STARTSOLPEKER_2_S <= (others => '0');
          DATAPEKER_S        <= "00000000000" + STARTPIKSLER_S;
          MAXINTENSITET_S    <= (others => '0');
--          MININTENSITET_S    <= (others => '1');

          if STARTTELLER_S > ('1' + STARTPIKSLER_S) then
            -- AD conv kommer på tre PCLK etter at
            -- DVAL er gått høy. I tillegg så gir de 16 første pikslene
            -- problemer, og må derfor vente bort de også (tar med et par
            -- ekstra for å være sikker)
            INNLESNINGTILSTAND_NESTE_S <= ingenSol;
          else
            STARTTELLER_S <= STARTTELLER_S + 1;
          end if;

        when ingenSol =>

          if DATAPEKER_S < "00000010000" then
            -- Må ha denne fordi Altera ikke klarer å lage en skikkelig
```

```
        -- kompilator. (Tilstandsmaskinen blir ikke riktig implementert)
        STARTSOLPEKER_S       <= (others => '0');
        SLUTTSOLPEKER_S       <= (others => '0');
        SLUTTSOLPEKER_2_S     <= (others => '0');
        STARTSOLPEKER_2_S     <= (others => '0');
        MAXINTENSITET_S       <= (others => '0');
        INNLESNINGTILSTAND_NESTE_S <= ingenSol;
      else

        STARTSOL_S <= '0';

        -- Kun for testformål. Finne ut hvor lav minimumsintensiteten er.
--      if T_DATA_I < MININTENSITET_S then
--        MININTENSITET_S <= T_DATA_I;
--      end if;

        -- Hvis intensiteten ikke kommer over NIVAA_S så sender jeg ned
        -- hvor det er høyest intensitet.
        if T_DATA_I > MAXINTENSITET_S then
          MAXINTENSITET_S <= T_DATA_I;
          SLUTTSOLPEKER_S <= DATAPEKER_S;
          STARTSOLPEKER_S <= DATAPEKER_S;
        end if;

        -- Hvis det er to punkter som har like høy intensitet og er maks,
        -- så sender jeg ned senter av disse to som solsenter.
        if T_DATA_I = MAXINTENSITET_S then
          SLUTTSOLPEKER_S <= DATAPEKER_S;
        end if;

        if T_DATA_I > NIVAA_S then
          -- Vi har nå detektert sol. Tar vare på hvor dette er og går inn
          -- i aktivSol-tilstanden (hvis da ikke dette er piksel 2043, for
          -- da går vi inn i avslutt-tilstanden).
          STARTSOLPEKER_S <= DATAPEKER_S;
          SLUTTSOLPEKER_S <= DATAPEKER_S;
          INNLESNINGTILSTAND_NESTE_S <= aktivSol;
        end if;

        if DATAPEKER_S > "11111111000" then
          -- Vi er nå ferdig med utlesning, og går til avslutt-tilstanden.
          -- Vi må også sette sentersol her, og at bredden på sola er null
          -- (betyr at vi ikke nådde solnivået som er satt).
          -- Grunnen til at jeg avslutter tidligere her enn i de andre
          -- tilstandene er fordi jeg trenger to klokkesykler for å komme
          -- igjennom de andre syklene. Det har jeg ikke tid til.

          -- Her bitshifter jeg STARTSOL og SLUTTSOL (deler med 2) og
          -- legger de sammen. Så legger jeg til 1 hvis det siste bit'et i
          -- begge to er 1. Da får jeg SENTERSOL. (Utregning av
          -- gjennomsnitt, men unngå at man går over maks antall bit)
          SENTERSOLPEKER_S <= ('0' & STARTSOLPEKER_S(10 downto 1)) +
                             ('0' & SLUTTSOLPEKER_S(10 downto 1)) +
                             ("0000000000" &
                             (STARTSOLPEKER_S(0) and SLUTTSOLPEKER_S(0)));
          SOLBREDDE_S <= "00000";
          INNLESNINGTILSTAND_NESTE_S <= avslutt;

        end if;

      end if;
```

```
    DATAPEKER_S <= DATAPEKER_S + 1;


  when aktivSol =>

    if DATAPEKER_S < "00000010000" then
      -- Må ha denne fordi Altera ikke klarer å lage en skikkelig
      -- kompilator. (Tilstandsmaskinen blir ikke riktig implementert)
      STARTSOLPEKER_S       <= (others => '0');
      SLUTTSOLPEKER_S       <= (others => '0');
      SLUTTSOLPEKER_2_S     <= (others => '0');
      STARTSOLPEKER_2_S     <= (others => '0');
      MAXINTENSITET_S       <= (others => '0');
      INNLESNINGTILSTAND_NESTE_S <= ingenSol;
    else

      SLUTTSOL_S <= '0';

      if DATAPEKER_S > "11111111001" then
        -- Hvis vi er i nest siste runde går vi til etterSol-tilstanden
        INNLESNINGTILSTAND_NESTE_S <= etterSol;
      end if;

      if T_DATA_I < NIVAA_S then
        -- Hvis det er slutt på soldeteksjon går vi til etterSol-
        -- tilstanden
        INNLESNINGTILSTAND_NESTE_S <= etterSol;

      else
        SLUTTSOLPEKER_S <= DATAPEKER_S;
      end if;

    end if;

    DATAPEKER_S <= DATAPEKER_S + 1;


  when etterSol =>

    if DATAPEKER_S < "00000010000" then
      -- Må ha denne fordi Altera ikke klarer å lage en skikkelig
      -- kompilator. (Tilstandsmaskinen blir ikke riktig implementert)
      STARTSOLPEKER_S       <= (others => '0');
      SLUTTSOLPEKER_S       <= (others => '0');
      SLUTTSOLPEKER_2_S     <= (others => '0');
      STARTSOLPEKER_2_S     <= (others => '0');
      MAXINTENSITET_S       <= (others => '0');
      INNLESNINGTILSTAND_NESTE_S <= ingenSol;
    else
      if SLUTTSOL_S = '0' then
        SLUTTSOL_S            <= '1';
        STARTSOLPEKER_2_S     <= STARTSOLPEKER_S;
        SLUTTSOLPEKER_2_S     <= SLUTTSOLPEKER_S;
      end if;

      if T_DATA_I > NIVAA_S then
        if DATAPEKER_S < "11111111001" then
          INNLESNINGTILSTAND_NESTE_S <= aktivSol;
          STARTSOL_S            <= '1';
          STARTSOLPEKER_S       <= DATAPEKER_S;
        end if;
```

```
end if;

if STARTSOL_S = '1' then
  -- Andre gang vi er over terkselnivå, og ned til under
  -- terskelnivå igjen. Her må vi
  -- avgjøre hva som er sol og hvem som er "fake". I
  -- tillegg så må vi nullstille ting slik at etter
  -- dette ser det bare ut som vi har hatt en deteksjon
  -- før (hvis vi får tre deteksjoner)
  -- Husk at _2_ er den første deteksjonen

  if ((SLUTTSOLPEKER_2_S - STARTSOLPEKER_2_S) >
      (SLUTTSOLPEKER_S - STARTSOLPEKER_S)) then
    -- Hvis den første bolken er størst
    if ((STARTSOLPEKER_S - SLUTTSOLPEKER_2_S) >
        (SLUTTSOLPEKER_S - STARTSOLPEKER_S)) then
      -- Hvis tomrommet mellom bolkene er større enn den
      -- minste bolken, så ignorerer vi den minste
      -- bolken. Eneste jeg gjør da er å si at vi ikke
      -- har detektert noe andre gang.
      STARTSOL_S         <= '0';
      STARTSOLPEKER_2_S  <= STARTSOLPEKER_2_S;
      SLUTTSOLPEKER_2_S  <= SLUTTSOLPEKER_2_S;
    else
      -- Hvis tomrommet mellom bolkene er mindre enn
      -- eller lik den minste bolken, antar jeg at
      -- tomrommet kun er en glipp (døde/slitne piksler
      -- eller lignede). Setter derfor de to blokkene,
      -- inklusive tomromett som en blokk.
      STARTSOL_S         <= '0';
      STARTSOLPEKER_2_S  <= STARTSOLPEKER_2_S;
      SLUTTSOLPEKER_2_S  <= SLUTTSOLPEKER_S;
--      STARTSOLPEKER_2_S <= "00111000000";
--      SLUTTSOLPEKER_2_S <= "00111000100";
    end if;
  elsif ((SLUTTSOLPEKER_2_S - STARTSOLPEKER_2_S) <
         (SLUTTSOLPEKER_S - STARTSOLPEKER_S)) then
    -- Hvis den andre bolken er størst
    if ((SLUTTSOLPEKER_2_S - STARTSOLPEKER_2_S) <
        (STARTSOLPEKER_S - SLUTTSOLPEKER_2_S)) then
      -- Hvis tomrommet mellom bolkene er større enn den
      -- minste bolken, så ignorerer vi den minste
      -- bolken. Jeg må da sette at siste bolk er den
      -- eneste detekterte.
      STARTSOL_S         <= '0';
      STARTSOLPEKER_2_S  <= STARTSOLPEKER_S;
      SLUTTSOLPEKER_2_S  <= SLUTTSOLPEKER_S;
    else
      -- Hvis tomrommet mellom bolkene er mindre eller
      -- lik den minste bolken, antar jeg at tomrommet
      -- kun er en glipp (døde/slitne piksler eller
      -- lignede). Setter derfor de to blokkene sammen,
      -- inklusive tomrommet, som én blokk.
      STARTSOL_S         <= '0';
      STARTSOLPEKER_2_S  <= STARTSOLPEKER_2_S;
      SLUTTSOLPEKER_2_S  <= SLUTTSOLPEKER_S;
--      STARTSOLPEKER_2_S <= "01100000000";
--      SLUTTSOLPEKER_2_S <= "01100000100";
    end if;
  else
    -- Hvis bolkene er like store.
    if ((SLUTTSOLPEKER_S - STARTSOLPEKER_S) <
```

```
                (STARTSOLPEKER_S - SLUTTSOLPEKER_2_S)) then
              -- Hvis tomrommet mellom bolkene er større enn
              -- bolkene så må vi gjette på hvilken av bolkene
              -- som er den riktige. Jeg velger da å gjette på
              -- den som er nærmest det som blei sendt forrige
              -- gang.
              if (SOLPEKERSISTRUNDE_S >
                  (('0' & STARTSOLPEKER_S(10 downto 1))
                   + ('0' & SLUTTSOLPEKER_2_S(10 downto 1)))) then
                -- Hvis den siste deteksjonen er nærest forrige
                -- setter vi den som gjeldene
                STARTSOL_S         <= '0';
                STARTSOLPEKER_2_S <= STARTSOLPEKER_S;
                SLUTTSOLPEKER_2_S <= SLUTTSOLPEKER_S;
              else
                -- Hvis den første deteksjonen er nærest forrige
                -- beholder vi den som gjeldene
                STARTSOL_S         <= '0';
                STARTSOLPEKER_2_S <= STARTSOLPEKER_2_S;
                SLUTTSOLPEKER_2_S <= SLUTTSOLPEKER_2_S;
              end if;

            else
              -- Hvis tomrommet mellom bolkene er mindre enn
              -- bolkene så slår vi dem sammen og gjetter på at
              -- pikslene i mellom var døde/slitne.
              STARTSOL_S         <= '0';
              STARTSOLPEKER_2_S <= STARTSOLPEKER_2_S;
              SLUTTSOLPEKER_2_S <= SLUTTSOLPEKER_S;
--              STARTSOLPEKER_2_S <= "11100000000";
--              SLUTTSOLPEKER_2_S <= "11100000100";
            end if;
          end if;                    -- slutt på testing av bolkene
          -- mot hverandre


        end if;


        if DATAPEKER_S > "11111111011" then
                        -- Ferdig med utlesning. VIKTIG!!!!
                        -- JEG TAR IKKE MED DE TRE SISTE
                        -- PIKSLENE DA DISSE IKKE
                        -- NØDVENDIGVIS ER INNENFOR
                        -- SPESIFIKASJONENE. (Dette står både
                        -- i databladet og er observert)
          SENTERSOLPEKER_S <= ('0' & STARTSOLPEKER_2_S(10 downto 1)) +
                              ('0' & SLUTTSOLPEKER_2_S(10 downto 1)) +
                              ("0000000000" &
                               (STARTSOLPEKER_2_S(0) and SLUTTSOLPEKER_2_S(0)));
          -- Her bitshifter jeg STARTSOL og SLUTTSOL (deler med 2) og
          -- legger de sammen. Så legger jeg til 1 hvis det siste bit'et i
          -- begge to er 1. Da får jeg SENTERSOL.

-------------------------------------------------------------------------------
-- HER MÅ TING BESTEMMES FØR OPPSKYTNING (gjort)
-- Må definere bredde (Er nå satt til 1 til 62 i steg på 2. Kan også gjøre det
-- dynamisk, f eks 1:2:32 33:4:128)
-------------------------------------------------------------------------------
          if (SLUTTSOLPEKER_2_S - STARTSOLPEKER_2_S) > "00000111101" then
            SOLBREDDE_S <= "11111";
          elsif STARTSOLPEKER_2_S = SLUTTSOLPEKER_2_S then
```

```
            SOLBREDDE_S <= "00001";
            SOLPEKERSISTRUNDE_S     <= SENTERSOLPEKER_S;
          else
            SOLBREDDE_S <= SLUTTSOLPEKER_2_S(5 downto 1)
                         - STARTSOLPEKER_2_S(5 downto 1)
                         - ((not SLUTTSOLPEKER_2_S(0)) or
                            STARTSOLPEKER_2_S(0));
            SOLPEKERSISTRUNDE_S     <= SENTERSOLPEKER_S;
          end if;

          INNLESNINGTILSTAND_NESTE_S <= avslutt;
        end if;                          -- Slutt på "hvis siste runde"



      end if;

      DATAPEKER_S <= DATAPEKER_S + 1;


    when avslutt =>
      SB_DATA_O(10 downto 0) <= SENTERSOLPEKER_S;

-------------------------------------------------------------------------------
-- Testing av intensitet. ENDRES FØR LAUNCH!!!!!!!!!!!!
-------------------------------------------------------------------------------
--        SB_DATA_O(15 downto 11) <= MAXINTENSITET_S(7 downto 3);

      SB_DATA_O(15 downto 11) <= SOLBREDDE_S;

      if VENTETELLER_S = 127 then  -- Vente litt slik at jeg er sikker på
                                   -- at S_DVAL_I er gått lav.
        INNLESNINGTILSTAND_NESTE_S <= idle;
      end if;

      VENTETELLER_S <= VENTETELLER_S + 1;

    end case;

  end if;
end process datainnlesning;


end FinnerDuNoe;
```

Listing D.2: *Solbuffer.vhd*

```
-----------------------------------------------------------------------
--Filnavn            : Solbuffer.vhd
--Påbegynt dato      : 21.03.2005
--Siste oppdatering  : 10.05.2005
--Av                 : Martin Sollien ( martiso@fys.uio.no )
--Kort beskrivelse   : Parallell til seriell omformer, og sender ut data til
--                     MAX490
-----------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;


entity Solbuffer is

  port (
    SB_DATA_I       : in  std_logic_vector(15 downto 0);
                                          -- Digitalisert signal fra TLC5540
    SCLK_I          : in  std_logic;      -- Klokke fra encoder
    GATE_I          : in  std_logic;      -- Angir at man skal sende
    SB_DATA_O       : out std_logic       -- datalinje ut til MAX490

    );

end Solbuffer;

architecture datasending of Solbuffer is

  signal SHIFTREG_S : std_logic_vector(15 downto 0);


begin  -- datasending

  -- purpose: Sende data til MAX490 og encoder
  -- type   : sequential
  -- inputs : SCLK_I, GATE_I, SB_DATA_I
  -- outputs: SB_DATA_O
  sending: process (SCLK_I, GATE_I, SB_DATA_I)
  begin  -- process sending
    if SCLK_I'event and SCLK_I = '1' then  -- rising clock edge
      if GATE_I = '0' then
        if SB_DATA_I = "1110101110010000" then  -- Skal ikke sende
                                                -- synksignal
          SHIFTREG_S <= "1110101110010001";
        else
          SHIFTREG_S <= SB_DATA_I;
        end if;
        SB_DATA_O  <= '0';
      else
        SB_DATA_O                <= SHIFTREG_S(15);
        SHIFTREG_S(15 downto 1) <= SHIFTREG_S(14 downto 0);
      end if;
    end if;
  end process sending;

end datasending;
```

Listing D.3: *rs232.vhd*

```
------------------------------------------------------------------------
--Hva                : RS232-tranciever driver
--Filnavn            : rs232.vhd
--Påbegynt           : 11. april 2005
--Siste oppdatering: 03. mai 2005
--Av                 : Martin Sollien (martiso@fys.uio.no)
--Kort beskrivelse : RS232-driver. Både sender og mottager. Kan velge antall
--                    bit, antall stopp bit, og paritetssjekk. Trenger ferdig
--                    neddelt klokke til den X-doble frekvensen som RS232 skal
--                    kjøres på. Antall ganger fordobling er angitt ved
--                   CLKTIMES_G, og må være minst fire, bør være 16 for god
--                    mottagning på RS232. Hvis man kun skal sende holder det
--                    med fire.
--Ressursforbruk    : Bruker mellom 57 og 100 logiske elementer. Default 81 LE
------------------------------------------------------------------------


------------------------------------------------------------------------
-- MERK !!!!
-- (FORELØPIG ER) FØLGENDE IKKE IMPLEMENTERT:
-- 1.5 STOPPBIT
------------------------------------------------------------------------


library IEEE;
use IEEE.Std_Logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;


entity RS232 is

  generic (
    CLKTIMES_G     : integer range 2 to 6  := 4;
                                        -- Hvor mange ganger høyere frekvensen
                                        -- er inn enn RS232 skal være. Bør være
                                        -- satt til minst 16 ganger, altså 4.
                                        -- (2^CLKTIMES) = antall gangner

    ANTBIT_G       : integer range 5 to 8 := 8;
                                        -- Antall bit som RS232 er satt på.
                                        -- Default 8 bit
    ANTSTOPBIT_G   : integer range 1 to 2 := 1;
                                        -- Hvor mange stoppbit:
                                        -- ANTSTOPBIT_G = 1 tilsvarer 1 bit.
                                        -- ANTSTOPBIT_G = 2 tilsvarer 2 bit.
                                        -- Har ikke støtte for 1.5 stoppbit.
    PARITYCHECK_G : integer range 1 to 5 := 1
                                        -- Paritetsbitsjekk:
                                        -- 1 => Ingen sjekk
                                        -- 2 => Odde bit (Odd)
                                        -- 3 => Partalls bit (even)
                                        -- 4 => Høy (Mark)
                                        -- 5 => Lav (Space)
    );

  port
  (
    CLK_I          : in     std_logic;      -- Klokke. X-dobbel frekvens i forhold
                                            -- til RS232klokka
    RESET_I        : in     std_logic;      -- Global reset, aktiv lav
```

```
    TX_O          : out   std_logic;      -- Sende-kanalen
    RX_I          : in    std_logic;      -- Mottager-kanalen

    -- Datalinjer
    DATA_I        : in std_logic_vector(ANTBIT_G - 1 downto 0);
                                          -- Datakanal inn
    DATA_O        : out std_logic_vector(ANTBIT_G - 1 downto 0);
                                          -- Datakanal ut

    -- Handshake
    DATASEND_I    : in  std_logic;        -- Det som ligger DATA_I skal sendes
    SERIAL_RDY_O  : out std_logic;        -- Klar til ny sending
    DATARECV_O    : out std_logic;        -- Data er mottat og ligger klar på
                                          -- DATA_O
    ERRORRECV_O   : out std_logic         -- Feil under mottagning av data
  );

end RS232;


architecture sendOgMotta of RS232 is

  signal RS232DATA_S : std_logic_vector(ANTBIT_G - 1 downto 0);

  type RS232_T is (rs232idle,startBit,trancieve,paritycheck,stopBit,
                   udef1, udef2, udef3);
  attribute enum_encoding : string;
  attribute enum_encoding of RS232_T : type is "gray";
  -- Å velge gray-encoding viste seg å være helt nødvendig for at kretsen
  -- skulle virke. Kom veldig ofte inn i en udefinert tilstand ved å bruke
  -- default encoding (one-hot).

  signal RS232TX_S : RS232_T;
  signal RS232RX_S : RS232_T;

  signal PARITYBIT_S  : std_logic;        -- Paritetsbit for TX
  signal PARITYBITR_S : std_logic;        -- Paritetsbit for RX
  signal STOPBITRX_S  : std_logic;        -- Stoppbitteller
  signal BITCOUNTTX_S : integer range 0 to ANTBIT_G; -- Bitteller til TX
  signal BITCOUNTRX_S : integer range 0 to ANTBIT_G; -- Bitteller til RX
  signal CLKCOUNTTX_S : std_logic_vector((CLKTIMES_G - 1) downto 0);
                                          -- Klokkedeler til TX
  signal RS232TXCLK_S : std_logic;        -- Klokka til RS232TX
  signal RXCOUNT_S    : std_logic_vector((CLKTIMES_G - 1) downto 0);
                                          -- Nullnivå for klokka på RX
  signal RXCLK_S      : std_logic_vector((CLKTIMES_G - 1) downto 0);
                                          -- Klokka til RX
  signal HALFRS232CLKTIME_S : std_logic_vector((CLKTIMES_G - 1) downto 0);
                                          -- Halvparten av maxverdi, brukes til
                                          -- RX

begin


  -- purpose: Dele ned klokka, slik at vi sender på riktig frekvens
  -- type   : sequential
  -- inputs : CLK_I, RESET_I, CLKCOUNTTX_S
  -- outputs: CLKCOUNTTX_S
  RS232CLKTX: process (CLK_I, RESET_I)
  begin  -- process RS232CLK
    if RESET_I = '0' then                 -- asynchronous reset (active low)
```

```
    CLKCOUNTTX_S  <= (others => '0');
  elsif CLK_I'event and CLK_I = '1' then  -- rising clock edge
    CLKCOUNTTX_S <= CLKCOUNTTX_S + 1;
  end if;
end process RS232CLKTX;

RS232TXCLK_S <= CLKCOUNTTX_S(CLKTIMES_G - 1);

-- purpose: Sende data over RS232
-- type    : sequential
-- inputs : RS232TXCLK_S, RESET_I, DATA_I
-- outputs: TX_O, SERIAL_RDY_O
RS232TX: process (RS232TXCLK_S, RESET_I)
begin  -- process RS232TX
  if RESET_I = '0' then
    RS232TX_S <= rs232idle;
    TX_O <= '1';
    SERIAL_RDY_O <= '1';
  elsif RS232TXCLK_S'event and RS232TXCLK_S = '1' then  -- rising clock edge
    case RS232TX_S is
      when rs232idle =>
        SERIAL_RDY_O <= '1';
        TX_O         <= '1';
        if DATASEND_I = '1' then
          SERIAL_RDY_O <= '0';
          RS232TX_S <= startBit;
          RS232DATA_S <= DATA_I;
          BITCOUNTTX_S <= 0;
        end if;


      when startBit =>
        TX_O <= '0';
        SERIAL_RDY_O <= '0';
        BITCOUNTTX_S <=  0 ;
        PARITYBIT_S  <= '0';
        RS232TX_S <= trancieve;

      when trancieve =>
        TX_O <= RS232DATA_S(BITCOUNTTX_S);
        PARITYBIT_S <= PARITYBIT_S xor RS232DATA_S(BITCOUNTTX_S);

        if BITCOUNTTX_S = ( ANTBIT_G - 1) then
          if PARITYCHECK_G = 1 then   -- Ikke paritetssjekk
            if ANTSTOPBIT_G = 1 then  -- Ett stoppbit
              SERIAL_RDY_O <= '1';
              RS232TX_S <= rs232idle;
            elsif ANTSTOPBIT_G = 2 then  -- Dobbeltstoppbit
              RS232TX_S <= stopBit;
            end if;

          else
            RS232TX_S <= paritycheck;
          end if;

        end if;
        BITCOUNTTX_S <= BITCOUNTTX_S + 1;


      when paritycheck =>
        if PARITYCHECK_G = 2 then      --Odd parity check
          TX_O <= PARITYBIT_S xor '1';
```

```
        end if;
        if PARITYCHECK_G = 3 then      --Even parity check
          TX_O <= PARITYBIT_S;
        end if;
        if PARITYCHECK_G = 4 then      --Mark bit
          TX_O <= '1';
        end if;
        if PARITYCHECK_G = 5 then      --Space bit
          TX_O <= '0';
        end if;

        SERIAL_RDY_O <= '1';

        if ANTSTOPBIT_G = 1 then  -- Ett stoppbit
          SERIAL_RDY_O <= '1';
          RS232TX_S <= rs232idle;
        elsif ANTSTOPBIT_G = 2 then  -- Dobbelt stoppbit
          RS232TX_S <= stopBit;
        end if;



      when stopBit =>
        TX_O <= '1';
        SERIAL_RDY_O <= '1';
        RS232TX_S <= rs232idle;

      when others =>
        RS232TX_S <= rs232idle;
    end case;


  end if;

end process RS232TX;


HALFRS232CLKTIME_S <= ('0','1', others => '1');

-- purpose: Motta data over RS232
-- type   : sequential
-- inputs : RS232CLK_S, RESET_I, RX_I
-- outputs: DATA_O, DATARECV_O, ERRORRECV_O
RS232RX: process (CLK_I, RESET_I)
begin  -- process RS232RX
  if RESET_I = '0' then                 -- asynchronous reset (active low)
    RXCOUNT_S <= (others => '0');
    ERRORRECV_O <= '0';
  elsif CLK_I'event and CLK_I = '1' then  -- rising clock edge
    RXCOUNT_S <= RXCOUNT_S + 1;
    case RS232RX_S is
      when rs232idle =>
        ERRORRECV_O <= '0';

        -- Verdi som settes av teller når vi er på starten av ny mottagning.
        -- I de andre tilstandene skal vi sjekke hva tellleren er i mot denne
        -- verdien og kun gå inn i de dersom de er identiske. MAO
        -- synkronisering (på første fallende flanke + halv klokkeperiode
        -- settes merket for når man skal lese ut bit'ene)
        if RX_I = '0' then              -- Det begynner nå en ny sending
          RXCLK_S <= RXCOUNT_S + HALFRS232CLKTIME_S;
          RS232RX_S <= startBit;
```

```
    end if;
    DATARECV_O <= '0';

  when startBit =>
    if RXCLK_S = RXCOUNT_S then
      ERRORRECV_O <= '0';
      DATARECV_O <= '0';
      if RX_I = '0' then
        RS232RX_S <= trancieve;
      else
        RS232RX_S <= rs232idle;
      end if;
      BITCOUNTRX_S <= 0;
      PARITYBITR_S <= '0';
      DATA_O <= (others => '0');
    end if;


  when trancieve =>
    if RXCLK_S = RXCOUNT_S then
      if BITCOUNTRX_S = (ANTBIT_G - 1) then
        if PARITYCHECK_G = 1 then
          RS232RX_S <= stopBit;
        else
          RS232RX_S <= paritycheck;
        end if;
      end if;
      DATA_O(BITCOUNTRX_S) <= RX_I;
      PARITYBITR_S <= PARITYBITR_S xor RX_I;
      BITCOUNTRX_S <= BITCOUNTRX_S + 1;
    end if;


  when paritycheck =>
    STOPBITRX_S <= '1';
    if RXCLK_S = RXCOUNT_S then
      case PARITYCHECK_G is
        when 2 =>                   -- Odd
          if PARITYBITR_S = RX_I then  -- Da er det feil
            ERRORRECV_O <= '1';
            RS232RX_S   <= rs232idle;
          else
            RS232RX_S   <= stopBit;
          end if;

        when 3 =>                   -- Even
          if PARITYBITR_S = RX_I then  -- Da er det riktig
            RS232RX_S   <= stopBit;
          else
            ERRORRECV_O <= '1';
            RS232RX_S   <= rs232idle;
          end if;

        when 4 =>                   -- Mark
          if RX_I = '1' then
            RS232RX_S   <= stopBit;
          else
            ERRORRECV_O <= '1';
            RS232RX_S   <= rs232idle;
          end if;

        when 5 =>                   -- Space
```

```
                if RX_I = '0' then
                  RS232RX_S    <= stopBit;
                else
                  ERRORRECV_O <= '1';
                  RS232RX_S    <= rs232idle;
                end if;

              when others =>              -- Skal aldri komme hit
                RS232RX_S <= stopBit;


            end case;
          end if;

        when stopBit =>
          if RXCLK_S = RXCOUNT_S then

            -- Sjekk antall stopbit
            if RX_I = '1' then
              if ANTSTOPBIT_G = 1 then     -- Kun et stoppbit
                DATARECV_O  <= '1';
                RS232RX_S    <= startBit;
              elsif STOPBITRX_S = '0' then  -- To stoppbit, andre bit
                DATARECV_O  <= '1';
                RS232RX_S    <= startBit;  -- To stoppbit, første bit
              else
                STOPBITRX_S <= '0';
              end if;
            else
              ERRORRECV_O   <= '1';
              RS232RX_S      <= rs232idle;
            end if;
          end if;

        when others =>
          RS232RX_S <= startBit;
      end case;


    end if;
  end process RS232RX;

end sendOgMotta;
```

D.2 MatLab-kode

MatLab er brukt for lesing fra insturmentet og kalibrering av det. Fila `leseSol.m` leser data fra instrumentet via RS232-grensesnittet. `SolFitting.m` brukes til kalibrering. Filene finnes på http://folk.uio.no/martiso/solconcepta/matlab.zip.

Listing D.4: *leseSol.m*

```
function [piksler,bredde] = leseSol(port,antallutlesninger)
% Funksjon for å lese ut data av SolConcepta.
% for å lese ut av COM1, angi port = 'COM1'
% antallutlesninger angir hvor mange punkter som skal leses ut.
%
% MERK: Hvis du avbryter denne midt i en utlesning VIL ting
%       krølle seg. Det som skjer at den angitte seriellporten
%       blir låst uten mulighet for å ta den tilbake. Omstart av
%       Matlab (og kanskje maskinen) blir påkrevd. Ved å bruke
%       lese3.m unngår du dette problemet (men den fila er ikke
%       like pent skrevet og du må selv åpne og lukke seriell-
%       porten ved behov).

antpkt = 64;
sistesynk = 0; %peker på andre byte i synkordet
antSidenSist = 0;
sisteord = 0;
synkord1 = 231;
synkord2 = 252;
halvarray=0;
synk2 = 0;
tic

if exist('s1')~=1
    s1 = serial(port, 'BaudRate', 115200, 'InputBufferSize',antpkt*2);
    s1.ReadAsyncMode = 'continuous';
    fopen(s1);
end

feilJa = 0;
gjPkt = 0;

while gjPkt < antallutlesninger;
%  feilJa = feilJa +1;
%  s1l = haha(feilJa,:);
  if(s1.BytesAvailable == 128)
    sprintf('Full buffer');
  end

  s1l = fread(s1,antpkt);
  synk1 = find(s1l == 231);
  synk2 = find(s1l == 252);
  if (length(synk1) == 1)
    if ((synk1 ~= 64) & synk1 ~= 1)
      if (s1l(synk1+1) == 252)
        synk = synk1;
      else
        synk = 0;
      end
    else
%       feilJa = 11;
%       display 'Første del av synkordet var siste eller første byte'
      synk = 0;
```

```
    end
  else
    if (length(synk1) > 1)
      if (length(synk2 == 1))
        if (s1l(synk2-1) == 231)
          synk = synk2 - 1;
        else
          synk = 0;
        end
      else
        %burde kanskje lagt inn sjekk om det er flere synk2ord, men da må
        %jeg bruke en for-løkke og ting vil ta mye lengre tid.
        synk = 0;
      end
    else
      synk = 0;
    end

  end

  if (synk > 0)
    if (sistesynk > 0)
      if (synk == sistesynk)
        letearray = halvarray;
        letearray(length(letearray)+1:length(letearray)+synk-1) = s1l(1:synk-1);

%        utslag = find(letearray(1:2:length(letearray)) > 31) * 2 - 1;
        utslag = find(letearray(1:2:length(letearray)) > 7) * 2 - 1;

        if (length(utslag) > 0)
          for i = 1:length(utslag)
            tiden = toc;
            fprintf('%6.4d %6.4d ', (bitand(letearray(utslag(i)),7)*256 ...
                                     + letearray(utslag(i)+1)), ...
                    bitand(letearray(utslag(i)),248)/4)
            fprintf(' %12.7f Hz\n', 1/tiden)
            gjPkt = gjPkt + 1;
            gjPiksel(gjPkt) = (bitand(letearray(utslag(i)),7)*256 ...
                + letearray(utslag(i)+1));
            gjBredde(gjPkt) = bitand(letearray(utslag(i)),248)/4;
            tic;
          end
        end

      else
%        display 'Synk error mot forrige innlesning'
        antSidenSist = antSidenSist + 62;
      end
    else
%      display 'Synk error mot forrige innlesning'
      antSidenSist = antSidenSist + 62;
    end
    halvarray = s1l(synk+2:length(s1l));
  else
%    display 'Synk error i denne innlesning'
%    synk1
%    synk2
    antSidenSist = antSidenSist + 62;

  end
  antSidenSist = antSidenSist + 2;
  sistesynk = synk;
```

```
end

fclose(s1);

piksler = gjPiksel;
bredde = gjbredde;
```

Listing D.5: *SolFitting.m*

```
function [senter,H,b] = SolFitting(beta,datasett)
%
% Funksjon for å finne avstand mellom pin-hole og sensor,
% og hvor tykt glasset i sensoren er.
%
% Inndata:
% --------
% beta:     Hvor skjevt pin-hole'et sitter i forhold til det aktive
%           området på sensoren.
% datasett: Datasettet på følgende form:
%
%  x_1 y_1
%  x_2 y_2
%   .   .
%   .   .
%  x_n y_n
%
% der x_1 er vinkelen til måling 1 i grader, og y_1 er sentersol
% gitt fra instrumentet.
%
% Utdata:
% -------
% senter: Mest sannsynelige senter for instrumentet
% H:      Hvor stor avstand med luft det er i mellom det aktive
%         området på sensor og pin-hole i mm.
% b:      Tykkelse på glasset i mm.
%
% Formel for plotting:
% utregnet = 1033
% + (1000/7)*(1/cos(beta*pi/180))*(H*tan(datasett(:,1)*pi/180)
% + b*tan(asin(sin(datasett(:,1)*pi/180)/1.52)))
%
% Av: Martin Sollien, 2006, for bruk i masteroppgave

kurve0 = ones(length(datasett),1);
kurve1 = tan(datasett(:,1)*pi/180);
kurve2 = tan(asin(sin(datasett(:,1)*pi/180)/1.52));
A = [kurve0 kurve1 kurve2];
svar = inv(A'*A)*A'*datasett(:,2);

senter = svar(1);
H = svar(2)*7*cos(beta*pi/180)/1000;
b = svar(3)*7*cos(beta*pi/180)/1000;
```