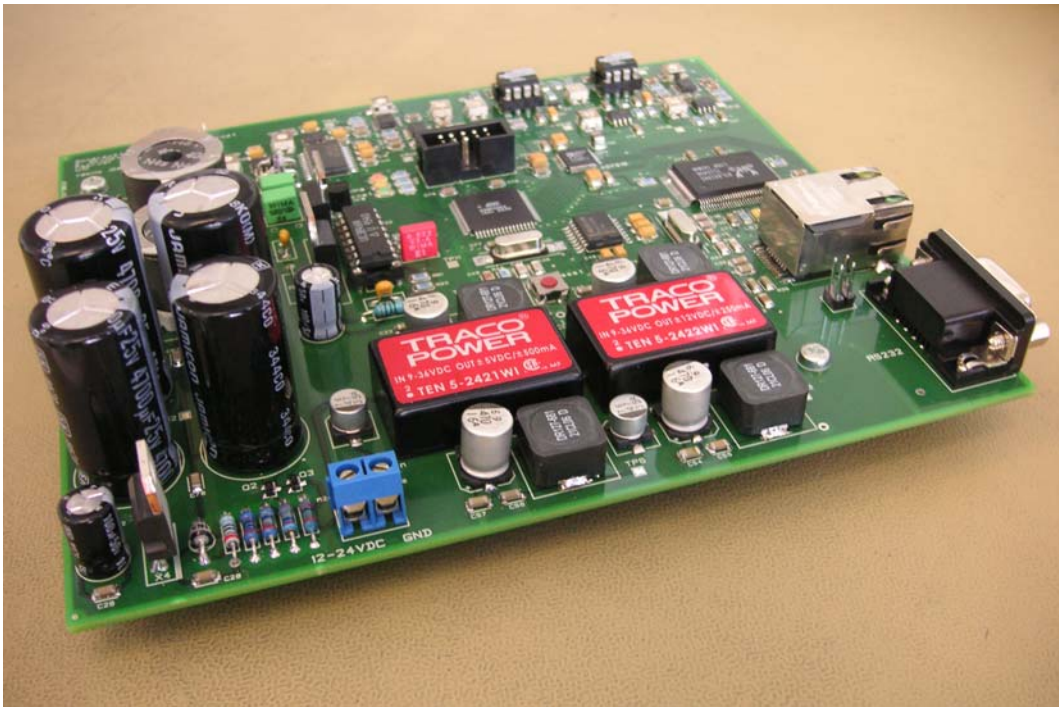


Masteroppgave:
Utvikling av sonarsystem

Knut Skumsvoll

Master i elektronikk og datateknologi
Studieretning: Instrumentering og måleteknikk

Fysisk Institutt

Universitetet i Oslo

Juni 2008

Forord

Denne oppgaven avslutter min mastergrad ved Fysisk Institutt, Universitetet i Oslo. Oppgaven som helhet har vært givende og variert. Jeg føler oppgaven har gitt meg en faglig kompetanse som jeg håper og tror jeg vil dra nytte av i arbeidslivet.

Først ønsker jeg å takke 1.amanuensis Torfinn Lindem som har vært min veileder under masteroppgaven. I tilegg til å tildele meg et spennende prosjekt har han avgitt tid til møter når dette har vært aktuelt. Jeg har også fått frie tøyler til å legge opp arbeidet selv. Å få lov til å styre mitt eget prosjekt etter eget ønske har vært veldig lærerikt.

Jeg ønsker også å rette spesiell takk til senioringeniør Stein Lyng Nielsen for tålmodighet og gode råd under mitt arbeid med oppgaven. Spesielt vil jeg fremheve hans vilje til å dele sin store og imponerende kunnskap med meg. Jeg har lært mye under samtaler vi har hatt.

Jeg vil også takke min tidligere medstudent Halvor Strøm som jeg delte mitt kontor med det første skoleåret. I tilegg til generelle diskusjoner har det vært fint å kunne slenge alle mine frustrasjoner til noen andre enn min samboer.

Til slutt ønsker jeg å takke min samboer Kristin Five for å holde ut disse årene jeg har vært så lite tilgjengelig.

Knut Skumsvoll
Juni, 2008

1.0 Sammendrag

Under arbeid med prosjektet har jeg som innledende delprosjekt arbeidet med analoge kretsløsninger, herunder likeretting og omhylningskurve av ekkosignal. Konklusjon og erfaring fra innledende prosjekt vil gi meg utgangspunkt for videre utvikling av sonarsystemet. Jeg har gjort dette da den analoge krets er det bærende element i systemet. Jeg anser det som hensiktsmessig å bygge digital løsning ut fra de krav og muligheter som den analoge krets setter. Digital krets tilpasses den analoge kretsløsning.
Etter arbeid med analog krets vurderte jeg forskjellige muligheter for behandling og justering av signalflyt. Dette innebærer vurdering av forskjellige metoder for sampling, TVG, kommunikasjon mellom komponenter samt ønskede tilkoblingsmuligheter. Jeg legger også opp til en kretsløsning med størst mulig fleksibilitet og justeringsmuligheter, da det kan være tilfeller der forventet funksjonalitet til delkrets ikke fungerer som forventet. Dette da det vil være økt risiko for feil om ny kretsløsning benyttes. Om krets har alternativ funksjonalitet eller mulighet for å kunne isoleres, anser jeg dette som gunstig. Forøvrig tester jeg så mange av delkretsene som mulig før total kretsløsning utarbeides, men hvor tid tilgjengelig i prosjektet som helhet vil påvirke omfanget av uttestingen.
Under mitt arbeid med krets for TVG-justering fattet jeg interesse i komponenten LM1972. Dette er et SPI kontrollert potmeter med forventet lav støy. Enhetens dempning er dB skalert, noe som vil være egnet for bruk etter hydroakustikkens karakter. Med disse og øvrige egenskaper ble enheten, etter enkel uttesting, valgt å implementere i total kretsløsning. Videre valgte jeg å benytte en 16-bit ADC for sampling av analogt signal. Det var ytret ønske fra kravspesifikasjonen i å benytte 10- eller 12-bit oppløsning ved prosjektstart. Da Ethernet kontroller kan kjøres i 8- eller 16-bit, valgte jeg å oppgradere ADC oppløsningen til 16-bit. Forøvrig er det ikke tidligere erfart å benytte Ethernet kontrolleren i 16-bit modus samt at det ikke er funnet informasjon om dette på internett. Med fare for å ikke være i stand til å få Ethernet kontroller til å fungere i 16-bit modus ble ADC med mulighet i å ”byte swappe” valgt. Denne funksjonaliteten gjør det mulig å sette MSB eller LSB som første bit fra den parallelle ADC utgangen. Dette åpner for å kunne kjøre Ethernet i 8-bit modus med ADC i 16-bit oppløsning. Ved enkel modifikasjon i programkode vil dette gi en økt fleksibilitet og sikkerhet. Jeg valgte også å kjøre µC, ADC og Ethernet kontroller i en felles bus. På denne måten lagres samplet data direkte til Ethernet kontrollerens bufferminne, uten å måtte mellomlagres i µC. µC vil av dette opprettholde kontroll over Ethernet kontroller og samtidig avlastes ved sampling av data. Fordelen vil gi seg gjeldende i en raskere samplings rate og økt hastighet på systemet generelt samt en økt nøyaktighet i tidsaktiverte rutiner i mikrokontrolleren. Jeg implementerer også mulighet for å kunne kontrollere sendereffekt på transduser via mikrokontroller. Bruker kan da styre ønsket sendereffekt via UDP pakker over Internett. Testkort og til slutt den endelige kretsløsning realiseres i CADStar. Endelig kretskort realiseres i et firelags kort.
Etter programmering av µC lages brukerprogram. Dette lager jeg med programmet Delphi. Datakommunikasjon sendes i UDP pakker mellom Delphi program og kretskort. Kommunikasjon er et toveis system mellom bruker og kretskort. Samtlige kommandoer fra bruker bekreftes satt fra kretskortet før systemstatus oppdateres i Delphi programmet. Etter ferdigstillelse er systemets funksjonalitet verifisert med test over nettet internt på fys-nettet ved UiO.

Innholdsfortegnelse

2.0 Innledning

2.1 Problemstilling

På slutten av 90-tallet ble det i samarbeid med Havforskningsinstituttet utviklet et SONAR-system med hensikt i å overvåke fisk i oppdrettsmerder. Utstyret ble benyttet som del av system for foringskontroll og atferdsundersøkelse.
Utstyret var i hovedsak basert på analoge kretsløsninger, med unntak av 2 stk. "Complex Programmable Logic Device", CPLD, som styrte henholdsvis senderen og den tidsvariable forsterkningskontrollen TVG, "Time Varied Gain". Utstyret kommuniserte med en IBM-kompatibel PC over printerkabel. Det var utviklet et eget kommunikasjonskort basert på ISA-bus som ble satt inn i PC.

Det kom nå ønske fra Havforskningsinstituttet om en oppgradering som gjorde det mulig å kjøre systemet under Windows operativsystem, samtidig som kommunikasjonen til/fra sonarsystemet skulle basere seg på Ethernet. En slik løsning vil gjøre det mulig å distribuere data og kontrollere utstyret over internett. Havforskningsinstituttet ønsket i tillegg til dette å kunne laste det analoge signalsystemet med 120kHz, en frekvensøkning i forhold til den tidligere begrensningen på 70kHz.

Utfordringen i denne oppgaven vil være å vurdere forbedring av den analoge delen (Rx-mottager) slik at man kan få en optimal løsning mot nye digitale styringskretser (mikrokontroller eller FPGA – "Field Programmable Gate Array"). Det er ønske om implementering av Ethernet kommunikasjon både på SONAR- og PC-siden. Forespørsel setter derfor ønske om revurdering av både hardware- og softwaredelen.

2.2 Overordnet funksjonalitet

Prosjektet omhandler deteksjon av akustiske signaler under vann for på denne måten å kunne bestemme avstand, størrelse og mengde fisk i det aktuelle området. Ved prosjektstart er ekkoloddsystem installert i stor-merder, med en eldre teknologi som anses som moden for utskiftning. Ved denne oppgaven drøftes analog krets for modifisering av detektert signal samt eventuelle nye måter å konstruere systemet som helhet. Det er spesifisert i kravspesifikasjonen å benytte mikrokontrolleren ATmega64 fra Atmel og integrere denne i et nytt design. Ved tidligere design er det benyttet 50kHz og 70kHz som utgangssignal fra transduser. Dette er begrensning tilknyttet etterfølgende analog mottaker krets. Det er ønskelig å øke denne operasjonsfrekvensen til 120kHz. Det vil være vesentlig å knytte flest mulig funksjoner til mikrokontroller og av dette kunne minimere antall komponenter. Datakommunikasjon mellom bruker og kretskort er ønsket gjort over internett via UDP-pakker.
Med dette som grunnfundament vil prosjektet omhandle utvikling av nytt system knyttet til hydroakustisk overvåkning og styring av fiskeoppdrett.

2.3 Hydroakustikk

Ved bruk av systemet benyttes en transduser for både å gi ut påtrykket generert signal fra μC og detektering av analoge ekkosignaler. Generering av signal innebærer å påtrykke en frekvens med et gitt antall svingninger med en gitt frekvens. Transduser benyttet tidligere og som også skal benyttes i dette prosjektet har oppgitt brukerfrekvens på 50kHz. Ved deteksjon av reflektert signal kan styrke og tid fra reflektert signal måles og gi informasjon til bruker. Karakteristikken for utbredelsen avhenger av flere komponenter, herunder trykk, saltinnhold og temperatur. Typisk hastighet for lyd under vann er målt til c = 1450 m/s og benyttes som utgangspunkt ved konstruksjon og programmering av systemet. Justering av dette bør gjøres ved praktisk erfaring på det aktuelle sted sonarsystemet skal benyttes.
I innledende fase av prosjektet ble det benyttet tid til konkretisering og drøfting av prosjektarbeidet. Det ble de drøftet teori med hydroakustikk og hva dette vil gi av konsekvenser for sonarsystemet. Krav og forventet signalverdier estimeres etter gjennomgang av boken 'Fhisheries Acoustics, David N. MacLennan og E. Jhon Simmonds'. Jeg kan dermed knytte teori om hydroakustikk til utvikling av sonarsystem og tilpasse spesifikt etter kravspesifikasjonen.
Fra oppgavens spesifikasjon er det satt ønske om å kunne detektere signalstyrke, TS (Target Strength), i området -30dB til -60dB. Ønsket rekkevidde til systemet er i området 5m til 100m. Dette er et utfordrende ønske, som gjør seg gjeldende i et høyt krav til krets for detektering av signalet. Transduser har oppgitt et 'Source Level' på 219dB ved en sendereffekt på 100W. Sonarens respons, Srt, er oppgitt til -185dB. For kalkulering av forventede signalverdier ved bruk av gjeldende transduser, tilpasses formel og verdier plottes i Excel for utregning. For eksempel av TS = -30dB se Figur 1 og Vedlegg 1.

$$EL = SL - 2TL + TS$$

$$VRTdB = SL + SRT - 2TL + TS$$

$$Vin\left[dB \Rightarrow V\right] = \left[V_0 * 10^{\frac{L\_in\_dB}{20}}\right] * \left[V_0 * 10^{\frac{Srt\_in\_dB}{20}}\right]$$

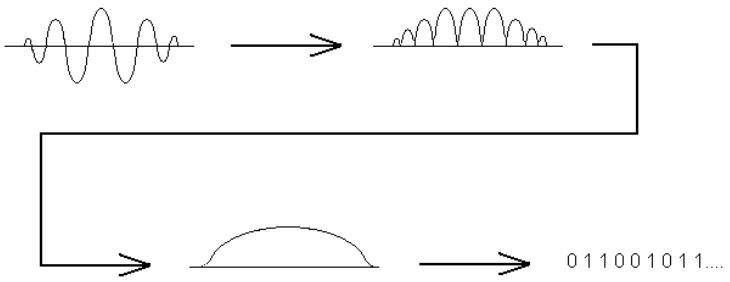
Figur 1 Beregning av forventet signalstyrke fra transduser
Se Vedlegg 2

Ved TS på -60dB estimeres signal fra transduser til å være 5,5mV ved 3m og 0,005mV ved 100m. Ved TS på -30dB, forventes det å måle 176mV ved 3m og ca 0,16mV ved 100m. Av dette fremkommer et ønske om detekterbart signalområde i spennet 0,005mV til 176mV. Hvorvidt signalet bør forsterkes før videre behandling eller om det vil være best egnet å justere analogt signal så raskt som mulig, drøftes i arbeid knyttet til kretsløsning. Utregning av estimert signalstyrke med og uten forsterkning er vist i Vedlegg 2.
Med dette som grunnlag utarbeides hardware- og softwareløsning til sonarsystem.

3.0 Analog krets

3.1-1 Likeretter og omhylningskurve

Prosjektet innledes med arbeid knyttet til analog krets med krets for likeretting av innkommende signal til transduser som kjerneproblematikk. Arbeidet er skrevet som egen del, da mye av arbeidet har konsentrert seg om nettopp dette. Etterfølgende arbeid er knyttet til påbygging på hva jeg kom frem til under dette arbeidet.
Den analoge signalflyt kan deles i tre deler. Første del mottar ekko og TVG-justerer denne. Dernest likerettes signalet før omhylningskurven av disse avtegnes. Det er denne omhylningskurven som samples og avleses av bruker. Da dette systemet ikke er et multibeam system, men kun baserer seg på effekt av mottatt ekkosignal, vil fase og frekvens av ekko være urelevant. Energi av omhylningskurve vil gi informasjon om objekt det aktuelle ekko er avgitt fra. Se Figur 2 for illustrasjon av signalflyt.



Figur 2 Illustrasjon av signalflyt for analogt ekkosignal

3.1-2 Overordnet funksjonalitet og arbeidsrutine for analogt kretssystem

Som del av masteroppgave, gjøres arbeid med delkrets tilhørende en kretsløsning som har til oppgave å styre og behandle en transduser brukt i hydroakustikk. Kretsen som arbeidet her konsentrerer seg om, omhandler behandling av mottatt ekkosignal fra transduser. I dette signalet ligger informasjonen brukeren vil ha av interesse, noe som gjør kretsen svært viktig og sentral. En krets er tidligere utarbeidet, men er noe treg i forhold til dagens ønske.
Kretsen som konstrueres deles i tre deler. Første del anser jeg som gitt og refereres til som 'Spennings tilførsel'. Denne kretsen har til oppgave å tilføre den spenningen som kretsen måtte ha behov for. Hvilke belastninger dette vil være, vil bli utarbeidet etter de kretser som finner seg egnet. På denne måten vil jeg kunne samle spenningsbehovet på en ryddig måte. Dette fremkommer også under simulering og design av kretsen.
Neste del skal transformere innkommende ekkosignal til absoluttverdi. Innkommende signal vil ha en svingende karakter. Signalet må likerettes slik at omhylningskurven kan måles over dette. Her ligger forøvrig svakheten til kretsen som helhet, med reaksjonstid til komponentene som sentral problematikk.
Omhylningskurven gjøres i tredje del, som vil være signalet som samples og benyttes som data for videre opptegning av det mottatte ekkosignalet.

Som samlet funksjon skal kretsen motta ekko signal, likerette signalet og gi ut omhylningskurven til det mottatte signal. Frekvensen på ekkosignalet varierer med 50KHz, 70KHz og 120KHz. Tidligere krets har hatt en begrensning opp mot 70KHz.

For strukturering av arbeidet, deler jeg arbeidet opp i 6 moduler.

Modul 1,-	Absoluttverdi krets
Modul 2,-	Komponentvalg
Modul 3,-	Omhylnings krets
Modul 4,-	Simulering
Modul 5,-	Design
Modul 6,-	Test

Jeg ønsker etter gjennomføring av dette delprosjektet, erfaring fra dagens tilgjengelige komponenter. Om nyere komponenter vil være det utslagsgivende moment, eller om man bør benytte andre kretsløsninger. Ved gjennomført test av overnevnte argumenter, vil videre arbeid med forbedring kunne gjøres på grunnlag av hva som kommer frem under dette delprosjektet. Som innledende verktøy benyttes PSpice for simulering av aktuelle kretser og komponenter. De simulerte data kan være noe misvisende, men for å kunne ha nytteverdi av verktøyet anses dette som neglisjerbart. Resultater fra simuleringer anses å være retningslinjer med sterk indikasjon på den reelle situasjon.
Delprosjektet vil være begrenset, da det ikke er avsatt mye tid for gjennomføring.

3.1-3 Problem, analogt kretssystem

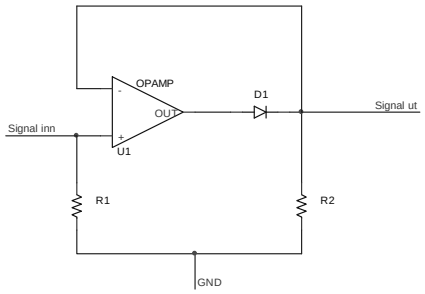
Kretsen som tidligere er benyttet har begrensninger i underkant av hva bruker ser det som gunstig. Begrensningen er knyttet til treghet i systemet, som resulterer i tap av verdifull data. Det er erfart en øvre opprasjons frekvens opp mot 70kHz ved det nå eldre systemet. Det vil derfor være av interesse å avdekke en bedre og raskere oppførsel ved å benytte dagens nyutviklede komponenter. Det vil også være nyttig å vurdere nye kretsløsninger om disse er bedre egnet. En bedre kretsløsning i samsvar med nytt komponentvalg vil gi verdifull erfaring for videre utvikling av transduser systemet.

Prosjektets del 1 anses som nyttig erfaring og vil ikke vektlegges mer enn hva som vil være relativt med hensyn på masteroppgavens arbeidsoppgaver som helhet. Komponentvalg, kretsløsninger, design av kretskort og praktisk erfaring gjøres så effektivt som mulig for på denne måten å kunne starte med øvrige arbeidsoppgaver knyttet til prosjektet. Fokus vil ligge i bruk av nyere komponenter og eventuelle nye kretsløsninger.

3.2 Kretsløsning for absoluttverdi

Krets for konvertering av innkommende ekko påbegynnes ved vurdering av forskjellige løsninger for konvertering til absoluttverdi. Å likerette for å kunne gi ut absoluttverdi av signalet kan gjøres ved veldig mange kretsløsninger, men hvor respons og nøyaktighet varierer svært mye. Som kriterium setter jeg ønske om løsning der avvik ved tidsforsinkelse og spenningsfall fra signalet, er så lavt som mulig. Tidsforsinkelse er relevant, særlig for responstiden. Treg respons vil kunne gi ut en lavere amplitudeverdi en hva som viser seg å være tilfellet for det innkommende signalet. Om responsen er tregere enn signalvariasjonen vil dette gi misvisende data. Ved et spenningsfall i en gitt deteksjonskrets, må signalet overstige et gitt nivå for å detekteres. Dette vil innvirke på både reaksjonstid og videre nøyaktigheten. Det vil også være positivt å benytte så få komponenter som mulig. Argumentet ved at dette vil gi en lavere produksjons kostnad, tas hensyn til av prisnipp, men da komponenter generelt er kilde til støy bør færrest komponenter benyttes. Relevant ved en absoluttverdi krets, vil være bruk av dioder. Mye av arbeidet med valg av krets blir derfor innledet med problematikk knyttet til denne komponenten.

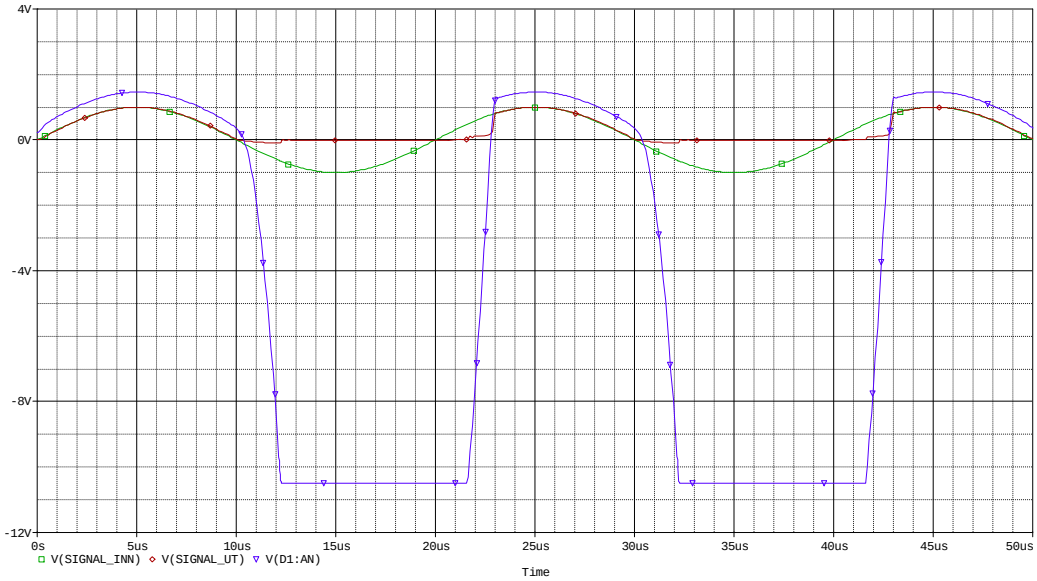
Den enkleste form for likeretting er bruk av operasjonsforsterker mot diode. Tilbakekobling på operasjonsforsterkeren vil gi sammenheng mellom innkommende og utgående signal, hvor dioden stenger eller åpner for signalet. Her vil dioden typisk påvirke signalet med et diodedropp. Dette bør tas hensyn til under kretssammensetning. Den enkleste form for likeretter krets er vist i Figur 3. Kretsen fungerer relativt godt, men setter strenge krav til operasjonsforsterkeren.



Figur 3 Enkleste form for likeretting

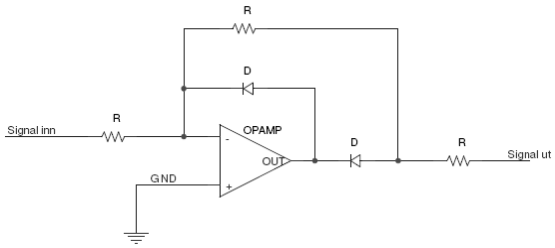
Ved den positive delen av et gitt sinus signal, vil negativ tilbakekobling resultere i at operasjonsforsterkeren følger det innkommende signalet nærmest perfekt. Når signalet videre passerer null og ned mot negativ spenning, vil dioden sperre. Dette bryter tilbakekoblingssløyfen til operasjonsforsterkeren slik at den nå vil gi sitt maksimale negative signal for å utjevne forskjellen mellom V-positiv og V-negativ. Operasjonsforsterkeren jobber nå på sin minimalverdi. Når nå signalet går over i den positive fasen igjen, vil operasjonsforsterkeren måtte skifte fra meget lavt signal, over til positivt signal. Det er bl.a. her et høyt krav til operasjonsforsterkeren settes, med krav til reaksjonshastigheten, slew rate. Ved signalforsinkelse knyttet til operasjonsforsterkerens maksimale spenningsvariasjon per tidsenhet, vil det kunne oppstå en forsinkelse i signalet

etter diodekoblingen. Dette vil kunne gi tap av data og en upresis gjengiving av signalet nær null. Simulering av fenomenet vises i Figur 4.



Figur 4 Simulering av krets Figur 3
Grønn farge er inngangssignal.
Blå farge er operasjonsforsterker utgang.
Rød er kretsutgangen.

Ved 21us til 23us, samt 41us til 43us i Figur 4, observeres tap av signal. I tillegg til dette vil det ved praktisk test typisk ligge et høyfrekvent støysignal på utgangen av operasjonsforsterkeren ved periodene 10u til 20u, og ved 31u til 40u i Figur 4. Den oscillerende effekten kommer av at operasjonsforsterkeren blir stående i open-loop tilbakekobling. Men denne støyen vil være av svært liten amplitude i forhold til signalet, som gjør støy effekten neglisjerbar i dette tilfellet.
Mye av overnevnte problematikk løses ved endring av kretsen vist i Figur 3. For eksempel ved at jeg endrer tilbakekobling og bruker en ny diode. Nytt eksempel av kretsløsning vises i Figur 5.

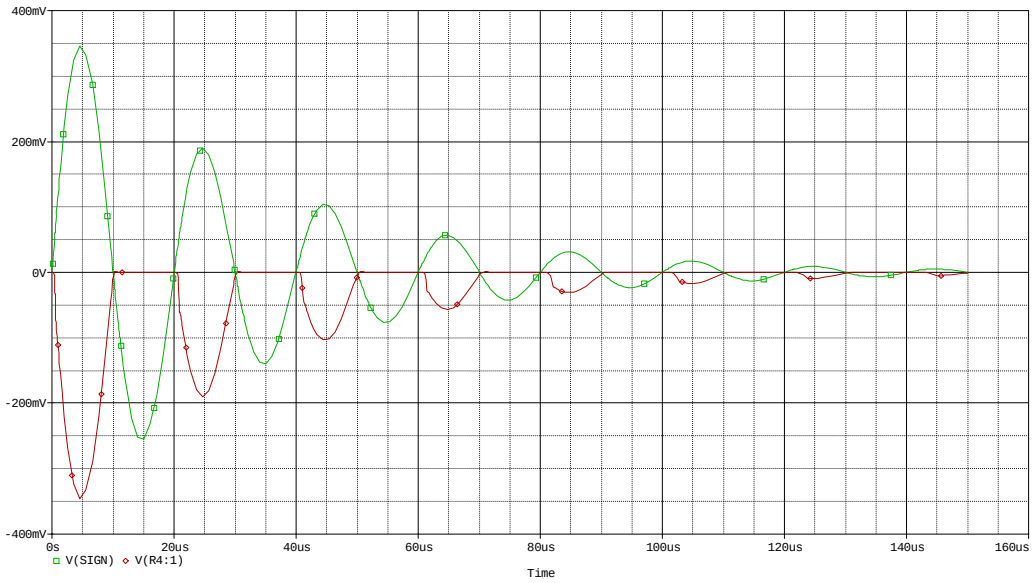


Figur 5 Forbedret likeretter krets

Kretsen i Figur 5 benytter ny diode i tilbakekoblingen. Denne vil hindre operasjonsforsterkeren i å trekke sitt maksimale negative nivå ved negativ inngangssignal. Følgen blir et bedre utgangspunkt for 'slew rate' kravet, samt at linearitetet ved lave verdier øker. Kretsen gir forøvrig krav til lav kildeimpedans, da impedansen til kretsen styres av inngangsmotstanden. Kretsen har også definert

utgangsmotstand i tilbakekoblingen. En eventuell kapasitans etter utgangen på kretsen vil derfor kunne lades ut via denne motstanden. Dette bør tas hensyn til ved bruk av denne løsningen.
Det gis inntrykk av at det er denne kretsløsningen det er tatt utgangspunkt i ved den tidligere benyttede kretsen. Forskjellen ligger i 180° dreining av diodene. Forøvrig, etter opptegning og simulering, bekreftes en noe begrenset operasjonshastighet ved den eldre benyttede modellen. Likeretterkretsen henger ikke lenger med fra ca 70mV amplitude. Et filter benyttet etter likeretter krets forverrer kvaliteten på utgangssignalet, særlig ved høyere frekvenser.
Som erfaring fra simulering, samt tidligere erfaringer gjort med kretsen, er løsningen noe avvikende fra ønskede resultat. Kretsen benytter også komponenter som kan være interessant å skifte ut.

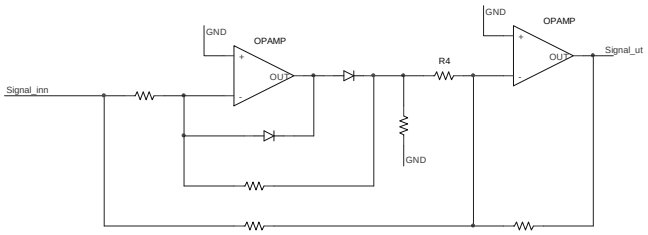
For erfaring designer jeg derfor et kretskort med denne løsningen. Kretsen har referanse i den tidligere benyttede kretsen, men hvor komponentene byttes ut til ny diode samt at ny operasjonsforsterker implementeres. For å kunne gi en bedre drøfting ønsker jeg å implementere nytt kretsalternativ i det samme designet. Dette vil gi et interessant svar på hvorvidt kretsen bør redesignes, eller om komponentene i dag vil gjøre merkbar forskjell. Modul 2 (komponentvalg) påbegynnes derfor allerede her med søk etter nye egnede komponenter til kretsen. Det blir derfor et parallelt arbeid med modul 1 og 2. Test av realisert kretskort er beskrevet senere i oppgaven (test og erfaring ved kretskortet er diskutert i 3.5-1 Test, ”*3.5-1-1 Kretskort I*”).
Kretsen som tidligere er benyttet gjør bruk av dioden 1N4148, samt operasjonsforsterkeren LT1058. I konklusjoner fra ”3.3 Komponentvalg”, ønsker jeg erfaring med operasjonsforsterkeren AD845 og diodene BAS16 samt 1N914. Dioden 1N914 er etter datablad tilnærmet lik den tidligere benyttede dioden 1N4148, men skiller seg noe ved bl.a. kapasitans. De ytterst få forskjeller i datablad ønskes erfart. Jeg ser det som interessant å erfare om diodene vil oppføre seg identisk.
Kretsen simuleres med dioden BAS16 og operasjonsforsterker LT1058. Jeg observerer ved simuleringen en mindre forbedring ved lave spenninger. Ved ca 40mV mister kretsen mye av signalet. Spenninger lavere enn dette har mindre innvirkning på likeretterkretsen. Dette er en svak forbedring, men ikke innenfor av hva som er ønskelig. Ny operasjonsforsterker benyttes, for å erfare om dette vil ha betydning. Simulering kjøres, vist i Figur 6.



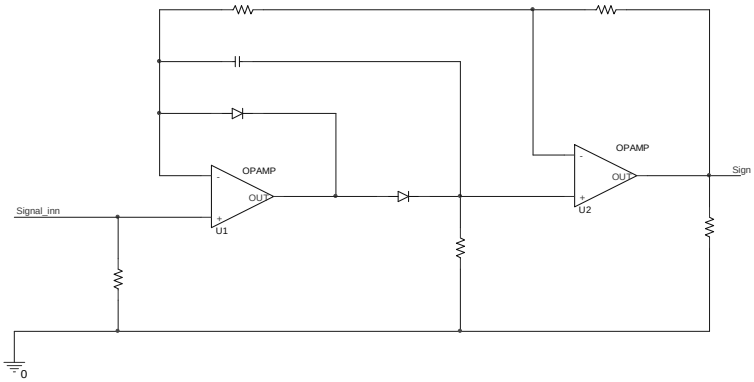
Figur 6 Simulering av tidligere benyttet krets
Diode BAS16, operasjonsforsterker AD845, 50KHz, 400mV amplitude med dempning

Som fremkommer av Figur 6, er dette en merkbar forbedring enn hva tidligere benyttede komponenter presterte. Etter hva simuleringen viser til, følger kretsen signalet godt. Da ned til signaler mot 15mV. Ved simulering ved høyere frekvenser vedvarer avviket. Det er i tillegg til overnevnte erfaringer, ikke fullverdig likeretting ved denne kretsløsningen. Positiv signal gjengis, men negativt signal blokkeres. Kun halvparten av inngangssignalet nyttegjøres. Dette er en negativ egenskap jeg ser potensial i å forbedre.

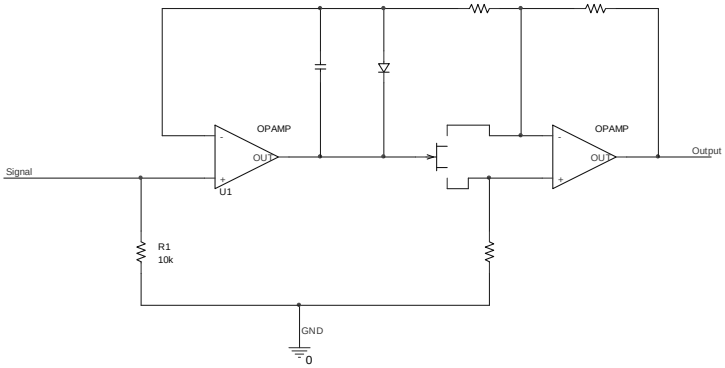
Etter vurdering av flere kretstyper, plukket jeg ut fire løsninger. Kretsene refereres til videre som krets ”A”, ”B”, ”C” og ”D”. Kretsene opptegnes i OrCAD for simulering i PSpice, for på denne måten å knytte simulert erfaring til kretsene. De fire løsningene ”A”, ”B”, ”C” og ”D” vises i hhv. Figur 7, Figur 8, Figur 9 og Figur 10. Som videre vurdering av kretsene benytter jeg arbeid fra ”3.3 Komponentvalg. For å kunne teste kretsløsninger på en relevant måte, anser jeg det som viktig å simulere med komponenter som vil være aktuelle å benytte i en endelig krets. Kretsene testes med ideell operasjons forsterker ved valg av kretsløsning.



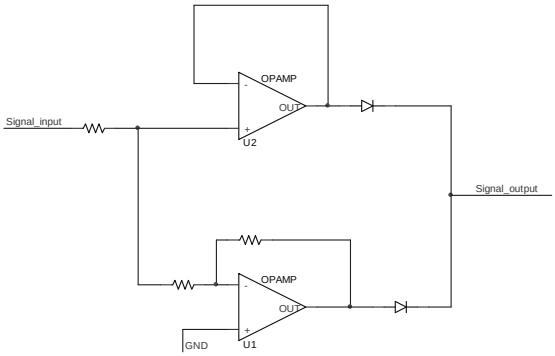
Figur 7 Kretsløsning ”A” for absoluttverdi



Figur 8 Kretsløsning ”B” for absoluttverdi

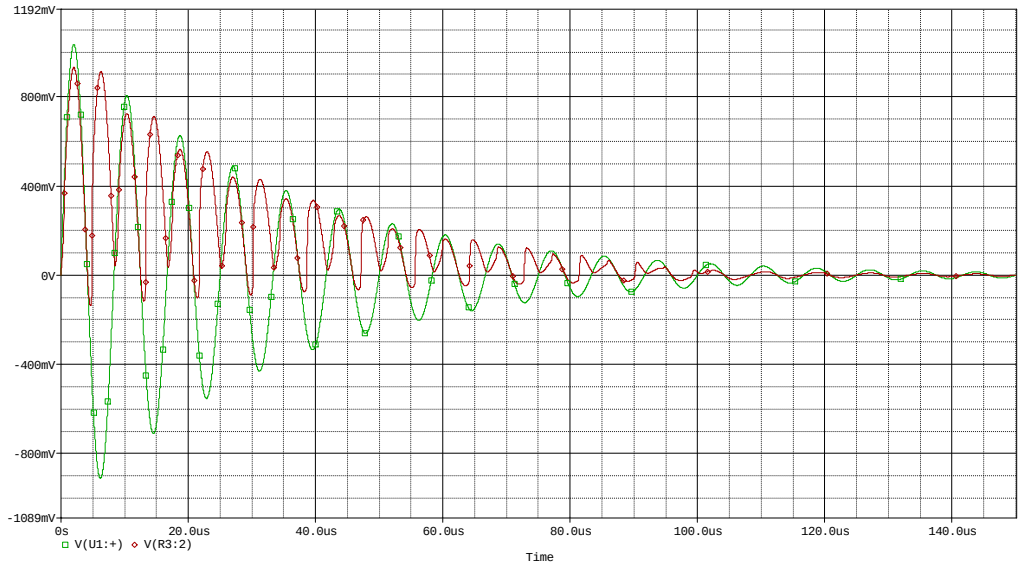


Figur 9 Kretsløsning ”C” for absoluttverdi



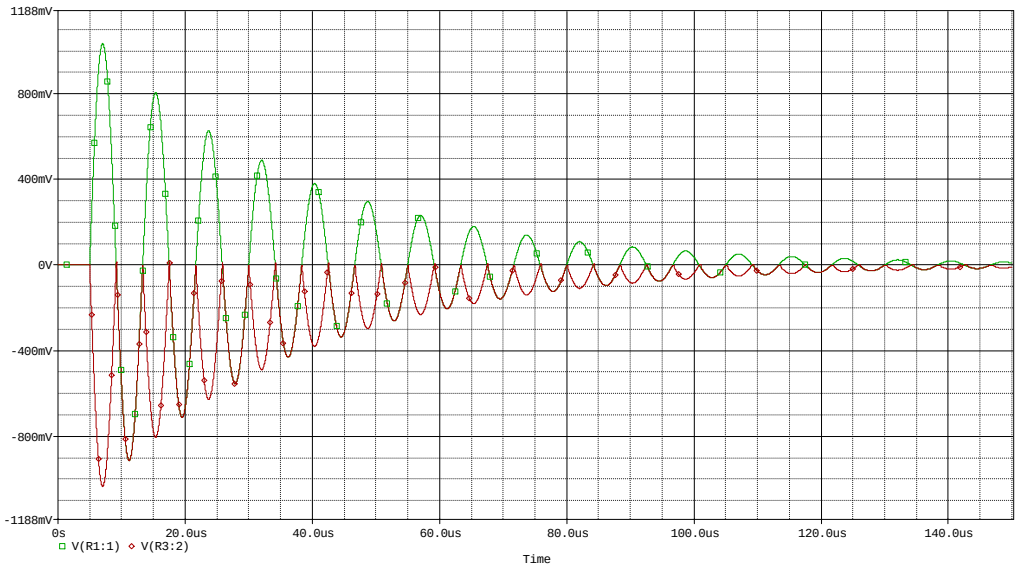
Figur 10 Kretsløsning ”D” for absoluttverdi

Løsning ”C”, i Figur 9, viste seg lite egnet. Transistor ga store avvik, særlig ved lave signaler. Som fremkommer av Figur 11, gir kretsen større avvik fra ca 30mV amplitude. Jeg observerte også at absoluttverdi varierer i amplitudeverdi og er noe ustabil allerede fra starten av i simuleringen. Signalet har også en noe lengre tid i området nær null, i forhold til hva den reelle tiden mot null vil være. Diodedropp endrer signalflyten fra det optimale. Produktet av denne effekten gir tap av data, samt en lavere middelverdi av det konverterte signalet. Dette anses som uheldig, da det vil gi en negativ innvirkning for den videre håndtering av signalet etter at absoluttverdi er oppnådd. Det gjøres oppmerksom på at kretsen kjøres mot et ”verst tenkelig” signal. Frekvensen er 120KHz og med en amplitude på 1V, som raskt faller i styrke. I løpet av 90u sekund er signalet falt fra 1V til ca 10mV amplitude.

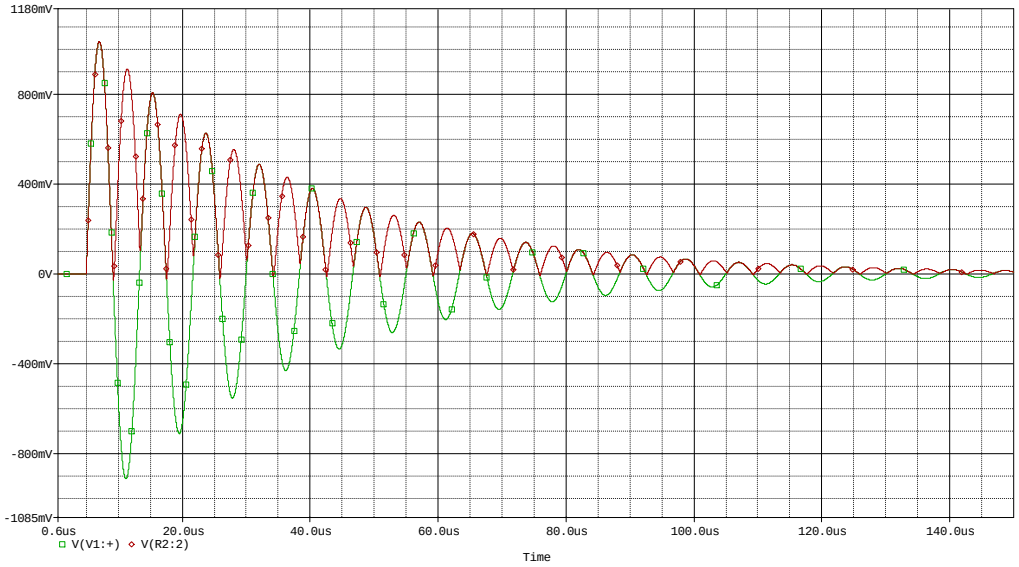


Figur 11 Simulering av krets ”C”
Diode BAS16, ideell operasjonsforsterker, 120KHz, 1V amplitude med dempning
Simulering viser her et lite presist system

Som fremkommer av Figur 7 og Figur 8, er disse kretsene relativt like, men det kan her nevnes at etter simulering viser egenskapene seg forskjellige. Ved tilbakekobling over første operasjonsforsterker benytter krets ”A” en motstand, mens ved ”B” benyttes en kondensator. Kretsene simuleres og er vist i Figur 12 og Figur 13.



Figur 12 Simulering av krets ”A”
Diode BAS16, ideell operasjonsforsterker, 120KHz, 1V amplitude med dempning
Jeg observerer en bedre simulert evne til å følge signalet nær null i denne kretsen

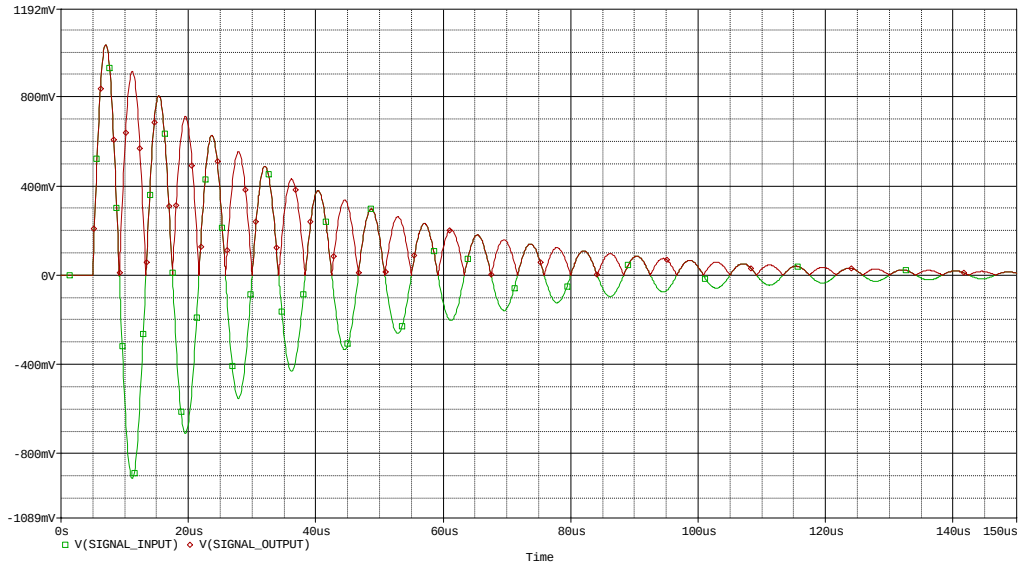


Figur 13 Simulering av krets ”B”
Diode BAS16, ideell operasjonsforsterker, 120KHz, 1V amplitude med dempning
Jeg observerer en simulert bedre evne til å følge de svakeste signalvariasjoner nær null

I utgangspunktet vil det være negativt å innføre kapasitans i denne type kretsløsninger. Forøvrig vil det videre være reell toppverdi ved omhylningskurven som vil være av interesse. Ved tilbakekobling mellom operasjonsforsterkere som vist i Figur 8, kretsløsning ”B”, vil kondensatoren gi avvik i verdi nær null. Som samlet gjennomsnittlig verdi, vil denne løfte snittet i liten grad. Avviket anses som positivt, da etterfølgende omhylnings krets typisk legger seg noe under toppverdi av amplitudeverdien. Ved simulering av forskjellige dioder, viser krets ”B” å være noe mindre kravstor til diodekapasitans, enn hva krets ”A” viser seg å være.

Jeg anser det som positivt å benytte signalet fult ut ved å omforme negativt signal til positivt signal, i motsetning til blokkering som tidligere kretsløsning gjør. Kretsene ”A” og ”B” anses som aktuelle erstatninger for den tidligere eldre likeretter kretsen.

Kretsen ”D”, vist i Figur 10, er enklere enn hva de tidligere drøftede kretser er. Operasjonsforsterkerne mates begge med innkommende signal direkte. Dette kan minne om en parallellkobling av operasjonsforsterkerne, i motsetning av en mer serie rettet kobling i krets ”A” og ”B”. Som negativ effekt ved krets ”A” og ”B” vil dioden i tilbakekoblingen over første operasjonsforsterker kunne gjøre annenhver kurvetopp lavere enn foregående, da lavere enn det faktiske signal. Dioden vil være et svakt ledd i kretskoblingen. Ved disse koblingene er det forøvrig tilbakekobling fra utgang tilbake til første operasjonsforsterker. Dette gir en korreksjon og forbedrer utgangsamplitudene. Ved krets i Figur 10, krets ”D”, er operasjonsforsterker koblet i ikke-inverterende og inverterende kobling. Som likeretterledd kobles diode på utgangen av operasjonsforsterkerne. Her vil diodene jobbe likt, uten forskjell i belastning. Kretsen simuleres og viser til gode resultater. Simulering er vist i Figur 14.



Figur 14 Simulering av krets ”D”
Diode BAS16, ideell operasjonsforsterker, 120KHz, 1V amplitude med dempning

Som fremkommer av Figur 12, Figur 13 og Figur 14, følger absolutt verdi av signalet, helt opp til 120KHz, med høy amplitude. Figur 12 og Figur 14 gjengir likerettet signal noe bedre, da krets ”B” simulert i Figur 14 har et lite avvik i området nær null, dannet av kondensator i tilbakekoblingen. Med simulering ved bruk av operasjonsforsterker AD845 fra modul 2, gjengis de simulerte egenskapene også der. Disse kretsene gir en klar og tydelig indikasjon på forbedring av den tidligere kretsløsning.

Med overnevnte resultater, ønsker jeg reell erfaring ved å produsere kretskort av aktuell krets. Den fysiske produksjon av PCB gjøres på universitetet. Jeg bestemmer meg for å teste krets ”B” med to typer dioder, for erfaring ved bruk av forskjellige komponenter. Kretsen viste til lavere krav til komponentparametere. Det vil her være av interesse å kunne se sammenheng mellom simulerte og målte resultater. Som innledet ønsket jeg

erfaring i krets med utgangspunkt i den tidligere benyttede enheten, men da med nye komponenter. Implementert i nytt kretskort inntegnes krets ”D”, vist i Figur 10. I alt tegnes og fremstilles det her to kretskort.

Simuleringene gjort under denne modulen har vært et meget nyttig verktøy. Men jeg tar det i betrakting at dette er en simulering med modeller som kan avvike fra de relevante verdier i forhold til hva en reel krets vil kunne gjøre. Unøyaktigheter i presisering av komponenter benyttet i PSpice vil kunne gi feil ved de små detaljer. Datablad må studeres og tas med i avgjørelser under de forskjellige løsninger og moduler. Det blir interessant å sammenligne dette fenomenet.

3.3 Komponentvalg

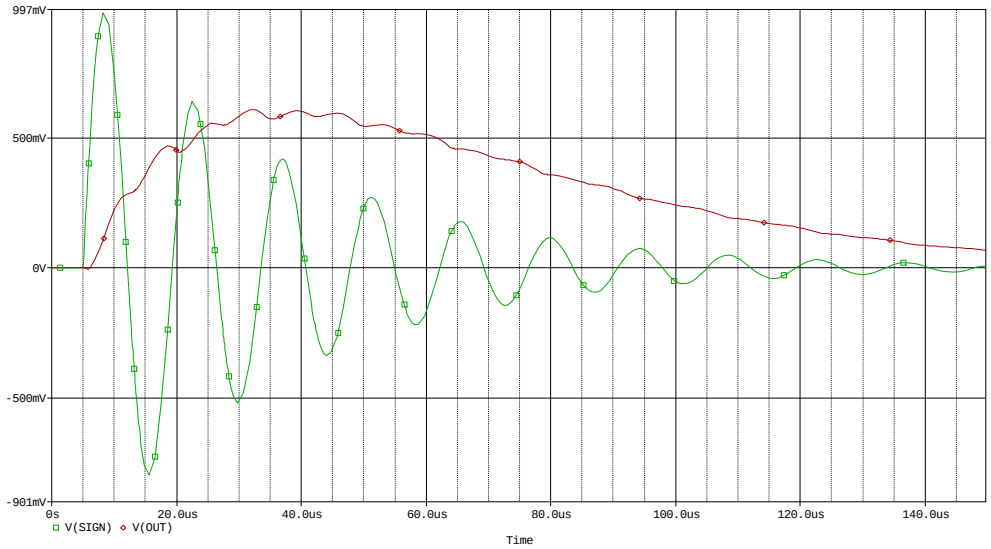
Ved valg av kretsløsning har det vært nyttig å gjøre simuleringer med kretsalternativer og komponenter som vil kunne være aktuelle. Jeg gjør gjennomgang av dioder som første del av komponent søk.
En egnet diode er en rask og relativt moderat strømsterk diode. Dioden bør ha liten lekkasje og være rimelig i pris. Signalene som skal kjøres over dioden er svak spenning og strøm. Det settes derfor lite krav til diodens "breakdown voltage". Problematikk knyttet til diodens diodedropp legges til utforming av kretsløsning. Diodedropp kan variere noe i størrelse avhengig av dioden. Under valg av diode tas dette derfor hensyn til kun som fenomen. Kretsen bør kunne kompensere for dette. Som relevant egenskap er diodens kapasitans. Denne verdien bør være lav. For en rask oppførsel ved høyere frekvenser, vil en kapasitans i dioden begrense gjengivelse av raskt varierende inngangssignal. Det er ønske om operasjonsfrekvens på 120KHz i det mottatte signalet. Etter en absoluttverdi konvertering av dette signalet vil dioden måtte jobbe med den doble frekvensen, dvs. 240KHz. Høy kapasitans vil påvirke diodens egenskap til å følge denne frekvensen. I tillegg til kapasitans, vil t_{rr} [(1)] tiden være relevant. Endring av spenning pr. tidsenhet bestemmer reaksjon og endringshastigheten av denne. Temperatur vil også være nyttig å ta hensyn til mot parametrene til dioden. En rask egenvarme vil kunne endre egenskapene til dioden. Dette med flere parametere undersøkes opp mot øvrige dioder. Jeg anser det som prioritet å benytte RoHS godkjent komponent. Tidligere benyttede diode er 1N4148. Dioden har en kapasitans på 4pF og en t_{rr} på 4ns, ved $I_{forward}$ på 10mA og R_{last} = 100Ω. Dioden er typisk benyttet i mange kretser og er relativt rask. Med betraktning i at delprosjektet har begrenset tid til rådighet, har jeg tatt Elfas sortiment som utgangspunkt for valg av dioder, med komponenter fra Farnel som en eventuell siste løsning.
Det eksisterer mangfoldige forskjellige dioder på markedet. Elfa alene kan tilby rundt 1000 typeaktuelle dioder. Her føres derfor dioder fortløpende opp i egen tabell i Excel, for på denne måten senere kunne luke ut dioder ved stadig å øke krav til enkeltparametere. Dette viste seg noe tidkrevende. Enkelte dioder plukkes til slutt ut og rask simulering gjøres i PSpice. Det viste seg også tidkrevende å konfigurere for simuleringsmodeller ved flere av diodene. Etter gjennomgang faller valget på dioden BAS16, i SOT23 kapsel. Denne har en kapasitans nede i 1.5pF. Denne parameteren skiller seg noe ut fra øvrige dioder. Det ble i tillegg observert diode 1N914. Denne viser seg tilnærmet identisk med den tidligere benyttede dioden 1N4148, men med få parameter forskjeller. Ved simulering viser den seg noe bedre egnet. Jeg anser det som sannsynlig at dette kommer fra en noe misvisende simuleringsmodell i PSpice. Det vil kunne være interessant å bekrefte dette ved praktisk erfaring.
Som simulert i Figur 6, forbedres signalgjengivelsen ved bruk av bedre egnet operasjonsforsterker. Operasjonsforsterkeren LT1058 er tidligere benyttet. Den har en slew rate på 100V/µs og en offset på 180µV til 600µV. Den har en CMRR på ca 80dB, og en settling time på 1.3µs. Datablad oppgis å være fra årstallet 1989.
Valg av operasjonsforsterker vil inneha noe varierende krav avhengig av krets den implementeres inn i. Her gjelder bl.a. båndbredden til forsterkningen. Avhengig av operasjonsfrekvens, vil forsterkningen endres noe. Jeg sammenligner derfor her med

t_{rr} [(1)] ,- reverse recovery time

aktuell krets fra ”3.2 Kretsløsning for absoluttverdi”, samt sannsynlig brukerfrekvens. Som ved valg av dioder, tas det utgangspunkt i Elfa sitt sortiment. Operasjonsforsterkere føres opp i tabell i Excel for så å filtreres med økende krav til enkeltparametere. Kretsen som anses som aktuell å benytte bruker ikke forsterkning og har lavt krav til båndbredde. Slew rate ønskes høy, men en for høy slew rate kan også virke mot sin hensikt. Systemet kan bli for følsomt og operasjonsforsterkeren variere raskere en hva som er ønskelig. Det tas også hensyn til en god CMRR og liten offset volt drift. Aktuelle operasjonsforsterkere evalueres og testes i PSpice, der modeller for simulering implementeres.
Etter gjennomgang faller valget mitt på operasjonsforsterkeren AD845. Denne er RoHS godkjent, samt relativt lik den tidligere benyttede operasjonsforsterkeren i pris. Den har en slew rate på 100V/µs og en god ”settling time” på 350ns ved 0,01 %. CMRR er på 94dB, en forbedring fra tidligere benyttede operasjonsforsterker. Dette og flere parametere utmerker seg ved denne operasjonsforsterkeren, som samlet anses som gode egenskaper ved bruk i aktuell krets.

3.4 Krets for omhylningskurve

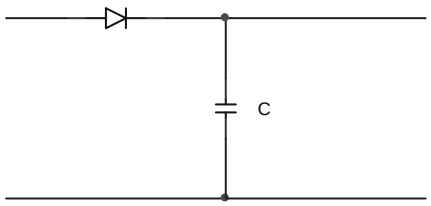
Ved tidligere kretsløsning er omhylningskurven dannet ved integrasjon over den positive del av inngangssignal. Ved simulering viser denne seg med relativt mye avvik sammenlignet med en ideell omhylningskurve. En noe treg stigetid og mindre presis evne til å følge den faktiske toppverdi observeres ved simulering. Ved å påtrykke 70KHz signal, med amplitude fra 1V med dempning, illustreres den avvikende oppførselen tydelig. Simulering er vist i Figur 15.



Figur 15 Simulering av tidligere benyttet krets
Diode 1N4148, opamp. LT1058, 70KHz, 1V amplitude med dempning

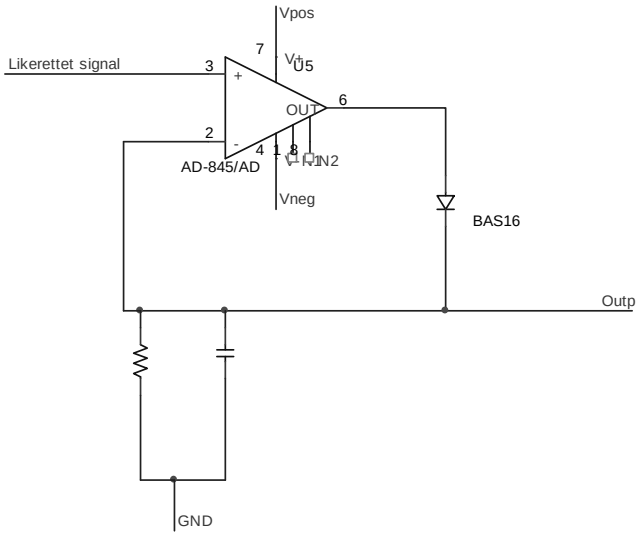
Simulering i Figur 15 er gjort med 70KHz, en frekvens lavere enn hva som i utgangspunktet er ønskelig maksimal frekvens. Ved å øke frekvensen ytligere, er det fra tidligere simuleringer erfart avvik i den innledede likeretterkretsen. Krets for likeretting av inngangssignal henger ikke med ved lave spenninger eller høyere frekvenser. Jeg anser det som sannsynlig at avvik oppstår med årsak i både kretskonstruksjon og trege komponenter. Ved videre å benytte omhylningskrets som ikke egner seg for høyere frekvenser, vil sum av feilkilder gi utilfredsstillende utgangssignal. Det kan her nevnes at eldre omhylningskrets trolig er beregnet med hensyn på likeretter kretsens begrensninger. Å sammenligne omhylningskretsen med krav om høyere operasjonsfrekvenser kan derfor her virke noe søkt. Men som fenomen vil en svakhet ved en aktiv integratorkrets være treghet knyttet til stige og fall tid.
Som forbedring vil det være god egenskap med rask stigetid. En rask stigetid vil nyttegjøre likerettet signal raskt og på denne måten gjøre bruk av signal så effektivt som mulig. Argument med konstruksjon av rask likeretterkrets vil virke lite hensiktsmessig om omhylningskretsen er treg. En for rask fall tid vil forøvrig gjøre perioden mellom hver toppverdi svært ustabil og på denne måten kunne gi en dårlig omhylningskurve. Fall tid bør være i likevekt mellom å ikke skape ustabilitet mellom toppverdier, og å falle raskt nok til å følge en svakere etterfølgende toppverdier eller ved endt inngangssignal.

Vanlig kretskonstruksjon gjør gjerne bruk av integrasjon med det beste kompromiss mellom overnevnte fall og stigetid. Om en rask stigetid kan oppnås uten bekostning av falltid, vil dette gi stor forbedring. Etter noe tid med vurdering av problematikken, kom jeg på ide hvor det kan benyttes diode mot kondensatorens egenskap ved å lagre spenning. Kretsen gjør bruk av teori bak en toppverdimåler, vist i Figur 16. Prinsippet er å la signalspenningen lade opp kondensatoren gjennom dioden. Er signalspenning større enn kondensatorspenningen, lades kondensatoren opp. Om signalspenning faller lavere enn kondensatorspenning, sperrer dioden slik at kondensator beholder spenningen. Kretsen vil av dette "huske" toppverdien av signalet. Dette forutsetter stor utgangsimpedans.



Figur 16 Prinsipp ved toppverdi måler

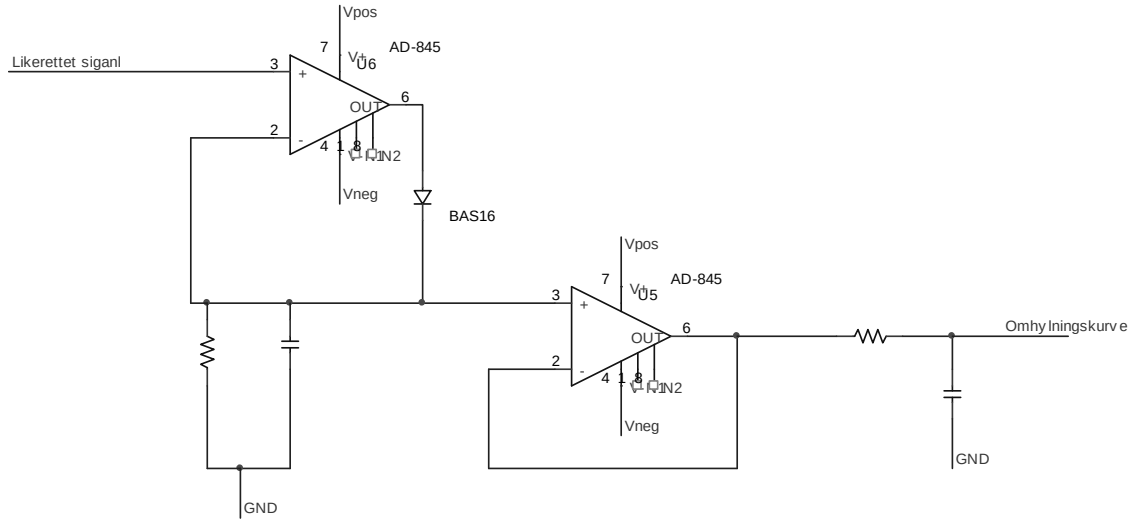
Rask stigetid oppnås ved kobling vist i Figur 16. Men ved denne kretsløsningen vil toppverdi forbli lengre enn hva som er ønskelig ved en omhylningskurve. Det vil derfor være behov for en kontrollert falltid. Jeg implementerer derfor videre en motstand parallelt med en kondensator. RC leddet vil da kunne utlades kontrollert over motstanden. Ved å koble som vist i Figur 17, vil responstiden være forbedret. En ujevn omhylningskurve vil kunne glattes ut ved bruk av lavpass filter.



Figur 17 Prinsipp ved krets for omhylningskurve

En svakhet ved denne løsningen vil være et større krav til operasjonsforsterkerens slew rate. Ved signal under siste toppverdi, vil kondensatoren opprettholde høy spenning i

tilbakekoblingen, som videre setter operasjonsforsterkerens utgang på sin minimale verdi. Når nå likerettet signal til operasjonsforsterkerens inngang når sin toppverdi, er kondensatoren noe ladet ut. Dette vil gi høyt signal på operasjonsforsterkerens utgang. Operasjonsforsterkeren må av dette endre spenning fra minimal verdi og opp over til positiv spenning lik inngangsspenning. Dette kan gi treghet i kretsen og vil være stressende for operasjonsforsterkeren. For øvrig kan dette forbedres ved å begrense negativ tilførselsspenning til spenning nærmere null. Ideelt vil jord være god tilkobling, men operasjonsforsterkeren vil da typisk ha spenningsfall som gir minimalverdi noe over null volt. Negativ tilførselsspenning bør av dette være noe lavere enn jord, en svakt negativ spenning.
Ved utgangen "Outp" i Figur 17, vil signalet ligge som sagtann signal, med middelverdi på likerettet signaltopp. Ved nå å legge inn spenningsfølger etterfulgt av et lavpassfilter, vil sagtann signal forbli upåvirket og glattet. Et lavpassfilter vil da ligge med verdi lik toppverdi av likerettet signal. Belastning av sagtann signalet vil endre impedansen i kretsen om lavpass filter legges ubufret på dette signalet. Det bør derfor gjøres bruk av spenningsfølger, slik at denne forblir ubelastet. Fullstendig krets for omhylningskurven er vist i Figur 18.



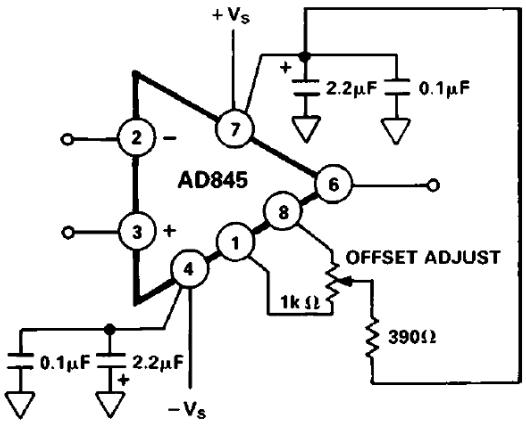
Figur 18 Kretsskjema for omhylningskurve

Lavpassfilter utregnes med hensyn på operasjonsfrekvensenes ytterpunkter, nemlig 50Khz og 120KHz. Ved likeretting av signalet, vil effektiv frekvens være tilnærmet den doble inngangsfrekvens, henholdsvis 100KHz og 240KHz. Lavpassfilteret må tilpasses dette.
Jeg ønsker å teste kretsen ved realisering i testkort og må skalere for beste midling slik at varierende frekvens kan benyttes. Ved et reelt kretsdesign, vil kretsen tilpasses den enkelte frekvens og dermed bli mer presis. I dette designet vil det derfor være et noe avvik i presisjonen.
Det kan eventuelt gjøres forbedring ved kontrollert jord i RC leddet under første operasjonsforsterker. Ved å benytte en tredje operasjonsforsterker med Schmitt-trigger kobling, vil utladningen av toppverdi kontrolleres for en raskere og kontrollert falltid. Ved å justere grense for lav verdi i Schmitt-trigger kobling, vil falltid, enten dette er ved

etterfølgende lavere verdi av likerettet signal eller at likerettet signal er endt, fortløpende justeres.
Det er ikke prioritert arbeid ved utvikling av denne ideen, da avsatt tid til dette delprosjektet ikke vil være tilstrekkelig. Krets vist i Figur 18 implementeres i kretskort sammen med krets ”A”. Det gjøres derfor kun en rask gjennomgang av kretsen, uten å stjele for mye tid tiltenkt øvrige kretser.

3.5 Design og kretsproduksjon

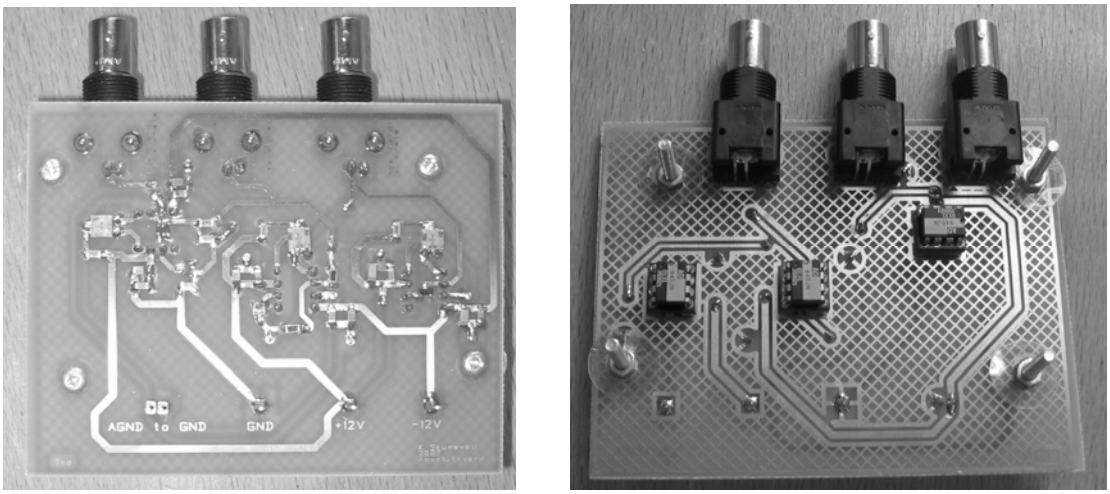
Som nevn har jeg bestemt å produsere totalt to kretskort i denne innledende delen av prosjektet. Et kort produseres med basis i eldre benyttede krets og krets vist i Figur 10 (”kretskort I”) samt å produsere et kretskort nummer to med krets vist i Figur 8 (”kretskort II”). Som nevnt implementeres kretsidé vist i Figur 18 i det sistnevnte kretskortet.
Fra datablad for operasjonsforsterkeren AD845, anbefales avkoblingskondensatorer ved inngangene for tilførselsspenning. Dette velges benyttet for en så stabil kretskonstruksjon som mulig. Det kobles i tillegg opp for justering av offset justering. Anbefalt kobling er vist i Figur 19.



Figur 19 Anbefalt Operasjonsforsterker avkobling og offset justering
Kopi fra datablad utgitt fra Analog Devices; Precision, 16MHz JFET Op Amp

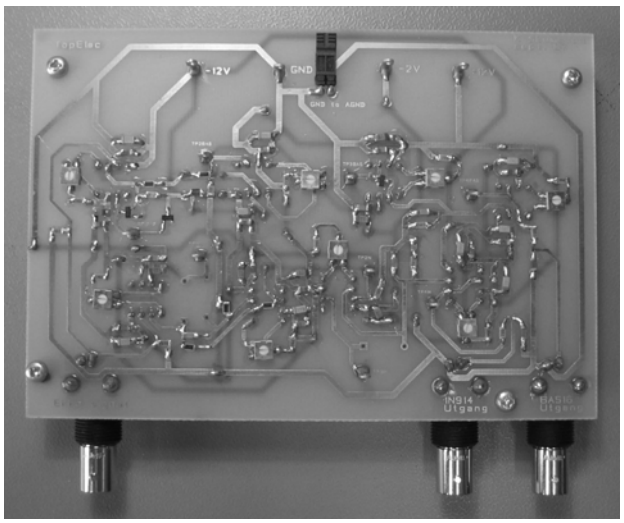
Plassering av avkoblingskondensatorer gjøres så nært operasjonsforsterkeren som mulig. For å unngå jordsløyfe i kretskort, kobles jord og analog jord sammen nær inngangsjord fra ekstern spenningskilde. Jeg plasserer testpinner der det finner seg egnet for på denne måten å kunne analysere resultatene på en tilfredsstillende måte. Da produserte kretskort vil være testkort, legges tilkobling av varierende tilførselsspenninger til pinner på kretskortet. Krav om varierende drivspenninger legges på eksterne spennigskilder. Som designverktøy benytter jeg CADStar, levert fra Zuken. Krets tegnes opp i skjemategning, som videre overføres til PCB verktøy for routing av kretskortet. Overnevnte koblinger gjøres ved samtlige produserte testkort. Produksjon av testkort gjøres ved UiO, hvor jeg følger opp med manuell plassering og pålodder komponentene. Pålodding av komponenter gjøres ved bruk av loddebolt. Jeg benytter derfor komponenter med størrelse ned til minimalt 1206. Typisk gjelder dette motstander og kondensatorer. Dioden BAS16 er i forpakning SOT-23.
Ved første kretskort, kretskort I, benyttes som nevnt likeretter krets vist i Figur 10, parallelt med den eldre kretskoblingen for sammenligning. Her implementerer jeg ikke krets for omhylningskurve, da dette ikke er kjernen i delprosjektets mål. Det er i utgangspunktet ønskelig med erfaring i effekten av nyere komponenter, målt opp mot de

tidligere benyttede komponenter. Respons og presisjon er interessante parametere å erfare.



Figur 20 Kretskort I, krets fra Figur 10 og krets basert på eldre likeretter

Ved det neste kretskortet, kretskort II, med krets vist i Figur 8, er det planlagt bruk av to typer dioder. Dette er dioden BAS16 og dioden 1N914. Dioden 1N914 er svært lik den tidligere benyttede dioden 1N4148. Det ønskes erfaring med kretsens avhengighet av varierende diodekapasitans samt å teste tillit til simuleringer gjort i PSpice. Det benyttes derfor operasjonsforsterker AD845 i begge kretsene i kretskortet. Krets vist i Figur 18 for omhylningskurve ønskes også erfart og implementeres i skjemategningen. Kretskort II er vist i Figur 21.
Skjemategning og PCB tegning for begge kretskortene er vist i Vedlegg 3.



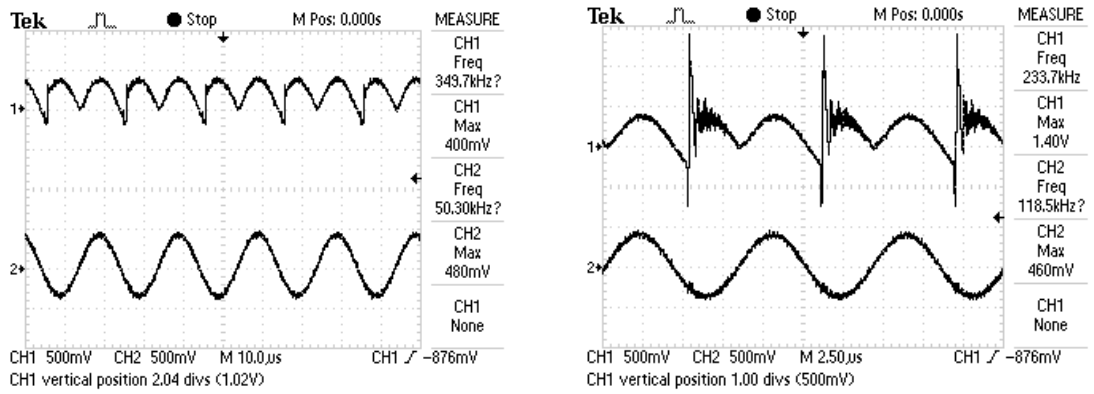
Figur 21 Kretskort II, krets fra Figur 8 og Figur 18

3.5-1 Test

Etter produksjon av testkort, oppkobles testkort med spenningskilder og signalforsyning. Jeg utfører enkel test for verifisering og utprøving av presisjon og nøyaktighet. Under testen tar jeg hensyn til de støyelementer som vil være relevante, enten dette er støy fra testverktøy eller testkortet. Det benyttes to spenningskilder, oscilloskopet Tektronix TDS1002 og signalgeneratoren GFG-813 fra GW, samt signalgenerator 33120A fra Hewlett Packard. Sistnevnte for generering av burst signal.

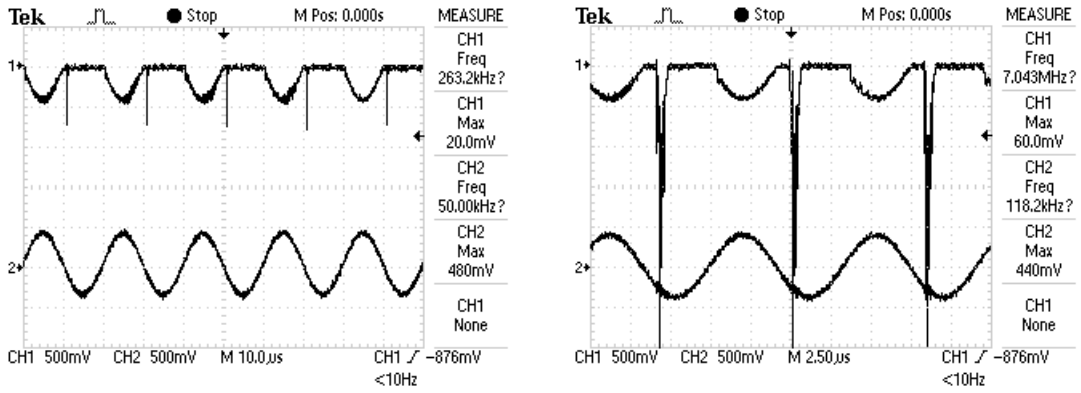
3.5-1-1 Kretskort I

Etter fremstilling og påmontering av komponenter, kobles spenning til testkortet. Jeg verifiserer at kortet ikke trekker mer strøm enn forventet og at det er spenning der det skal være spenning. Testkort viser seg å virke som forventet.
Signalgenerator påkobles testkortet og signalfrekvens 50kHz, 70kHz og 120kHz påtrykkes kretsene, med 400mV amplitude. Resultat måles og avbildes. Det avdekkes rask en utilfredsstillende oppførsel ved krets ”D”, fra Figur 5. I grenseområdet nær null opptrer brytningen som kraftig støy og del av signalet går tapt. Måleresultater ved 50kHz og 120kHz inngangsfrekvens er vist i Figur 22.



Figur 22 Likerettet signal ved 50KHz og 70KHz

Som observeres fra måling, øker feil med økende frekvens. Simulering i PSpice viste seg forøvrig bedre enn hva som her viste seg gjeldende ved fysisk måling. Ved test av nye amplitudeverdier viste avviket å vedvare. Jeg anser kretsen som lite egnet til bruk i sonarsystemet.
Ved krets basert på eldre versjon, gjøres tilsvarende test, med tilsvarende amplitudeverdi 400mV. Måleresultat er vist i Figur 23.



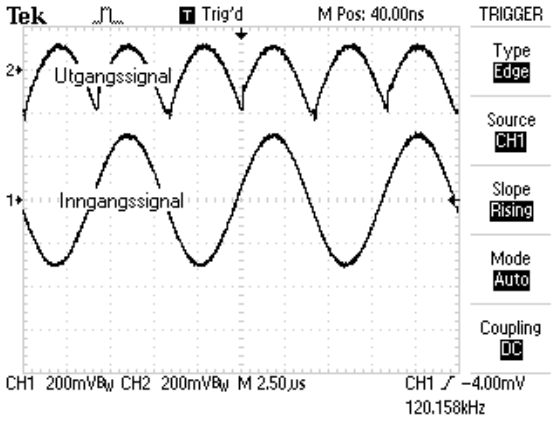
Figur 23 Likerettet signal ved 50KHz og 120KHz

Som fremkommer av målingene, er denne kretsen bedre egnet en hva krets fra Figur 5 viser seg å være. Men test viser også her utilfredsstillende resultat, da særlig ved høyere frekvenser. Ved implementering av et filter som etterfølgende krets anser jeg kretsen som tilfredsstillende ved lave frekvenser. Forsøket viser også at nye komponenter ikke ser ut til å gi tydelig forbedring av kretsen.

3.5-1-2 Kretskort II

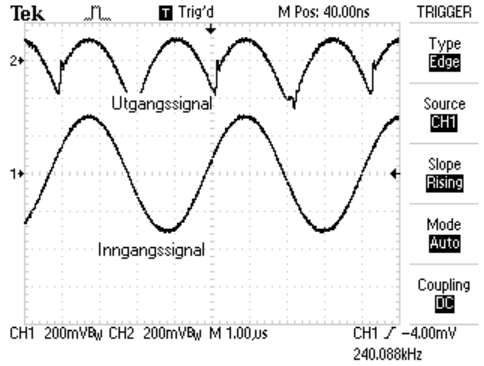
Som første steg ved utprøving av krets, utfører jeg funksjonstest. Lave spenninger påtrykkes kretsen og verdier måles. Testkortet viste seg å fungere tilfredsstillende. Videre påtrykker jeg signal fra signalgeneratoren GFG-813. Signalet justeres til sinus signal, med de aktuelle frekvenser og amplitude som varierende parametere. Det erfares å være uhensiktsmessig å gjøre tester med lave spenninger grunnet støy. Det observeres et dominerende støyforhold ved påtrykte lave amplitude verdier. Jeg observerer svak støy på jord.

Første måling gjøres ved testpinne 2, utgangen til likeretterkrets (se Vedlegg 3) ved kretsen som benytter dioden BAS16. Ved dette punktet måles likerettet signal. Ved etterfølgende krets vil signalet passere selvkomponert krets for omhylningskurve. Dette ønskes målt ved et senere tidspunkt. Frekvensen justeres til 50KHz, 70KHz og 120KHz. Signalet viser seg å gjengis som ved simulering i PSpice. Dioden følger inngangsfrekvensen godt. Måling vises i Figur 24.



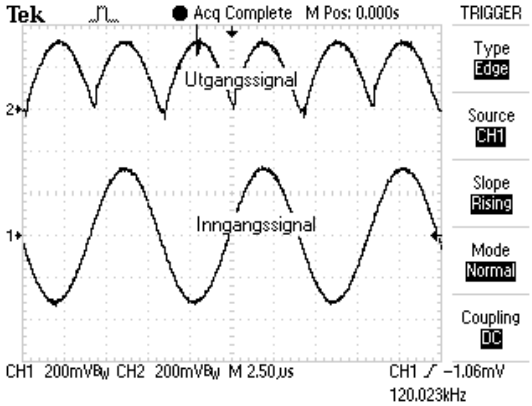
Figur 24 Likerettet signal ved 120KHz

Ved å videre øke frekvensen på inngangssignalet utover de ønskede 120KHz, viser kretsen seg å følge også raskere frekvenser. Ved frekvenser over ca 250KHz øker avviket rundt null, men da krets for omhylningskurve baserer seg på likerettet toppverdi, vil ikke dette påvirke utgangssignalet i nevneverdig grad. Måling av likerettet signal ved 240KHz er vist i Figur 25.



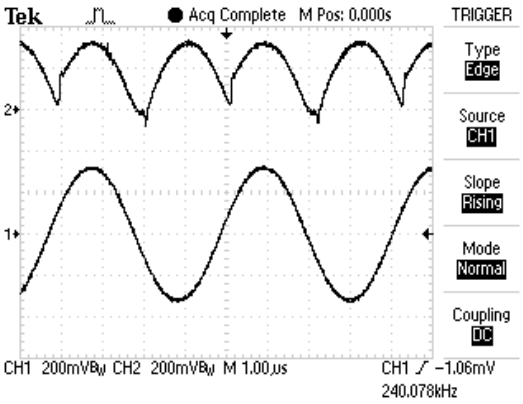
Figur 25 Likerettet signal ved 240KHz, 300mV amplitude

Krets med diode 1N914 velges og tilsvarende måling utføres. Kretsen belastes med 50KHz, 70KHz og 120KHz. Dioden 1N914 har en noe høyere kapasitans (4pF), enn hva dioden BAS16 har (1,5pF). Utgangssignal ved 120kHz inngangsfrekvens er vist i Figur 26.



Figur 26 Likerettet signal ved 120KHz

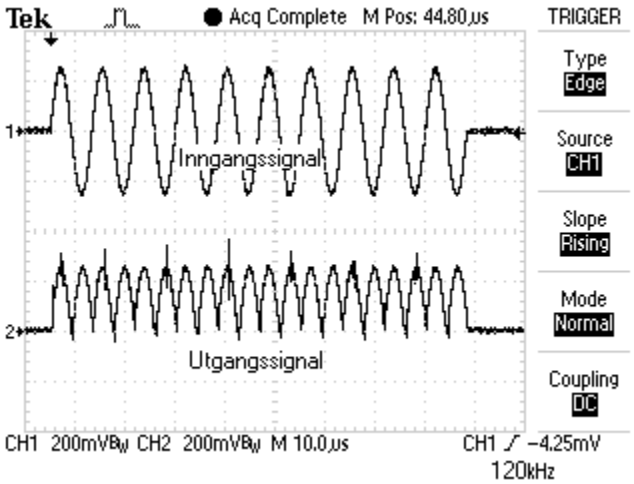
Som fremkommer av Figur 26, viser kretsen seg å være relativt uavhengig av diodekapasitansen. Sett opp mot forsøk vist i Figur 24, gir avbilding relativt likt utgangssignal ved samtlige frekvenser. Ved også her å øke frekvensen over beregnede operasjonsfrekvens, viser også krets med dioden 1N914 å begrense seg til ca 240KHz. Måling er vist i Figur 27.



Figur 27 Likerettet signal ved 240KHz, 300mV amplitude.

Ved videre vurdering av kretsen viser krets for likeretting å være noe unøyaktig med hensyn på varierende amplitude. Men etter gjennomgående måling, viser feil i målt signal å oppstå ved støy på innkommende signal. Med de ledninger og eksterne oppkoblinger jeg benytter, mottar kretskortet støy ved eksterne enheter. Dette viser seg å forplante seg videre i kretsen. Ved gitte amplitudeverdier skaper dette støyende svingninger. Støy er påvist ved inngangssignalet.

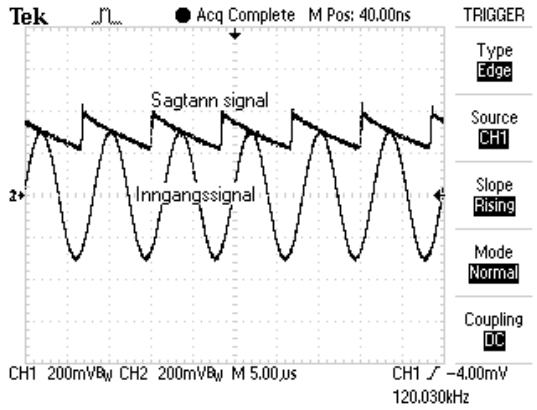
Ved praktisk bruk av krets koblet mot transduser, vil inngangssignal mottas som et "burst"- signal. Jeg ønsker å erfare denne type signal påtrykt kretsen. Det gjøres derfor test med signalgenerator 33120A fra Hewlett Packard. Signalgenerator settes med 10 perioder burst. Resultat etter påtrykt 120kHz signal er vist i Figur 28.



Figur 28 Burst ved frekvens 120KHz
Diode BAS16

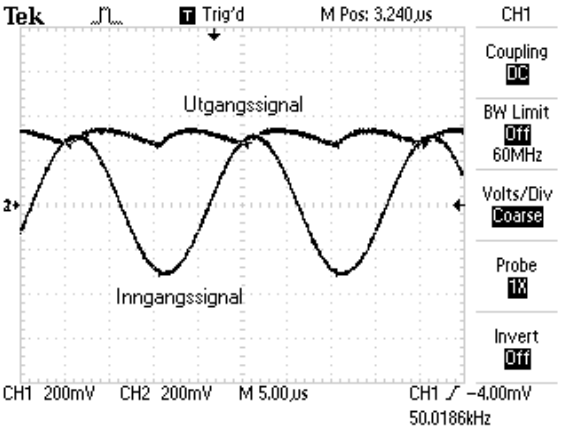
Det observeres her enkelte peak ved likerettet signal. Disse spores forøvrig tilbake til støy ved innkommende signal på kretskortet. En noe avvikende presisjon bekreftes i området rundt null. Dette er som ved simulering, og er forventet avvik. Da jeg benytter kondensator i tilbakekobling vil denne skape forstyrelse ved økende frekvens. Forøvrig vil denne begrense varierende toppverdier, da det er benyttet diode tilbakekobling over første operasjonsforsterker. Aktuell diode er opptegnet i kretsskjema i Figur 8.
Jeg gjør tilsvarende måling med diode 1N914. Denne dioden har en noe svakere evne til å følge raske signaler. Men som vist i Figur 26 har kretskonstruksjonen som egenskap å være relativt tolerant med hensyn på diodekapasitansen. Burst signal påtrykkes krets med diode 1N914 for måling, på tilsvarende måte som vist i Figur 28. Likerettet signal ved 50KHz, 70KHz og 120KHz med bruk av dioden 1N914 viser å gi tilsvarende utgangssignal som ved bruk av dioden BAS16.

Ved krets for omhylningskurve benyttes diode mot RC ledd for generering av sagtann signal. Ved måling ved dette leddet, observerer jeg forventet signal, med middelverdi om likerettet toppverdi. Måling av sagtann signal er vist i Figur 29.



Figur 29 Sagtann signal ved 120KHz, 300mV amplitude

Videre måler jeg utgangssignalet ved 50kHz, 70kHz og 120kHz. Utgangssignal er, som nevnt, et signal dannet av krets designet for bruk av samtlige nevnte frekvenser. Det vil derfor være avvik i stabilitet på utgangssignalet. Dette kan endres ved tilpassing av RC-ledd i krets for omhylningskurve for én bestemt frekvens. Se Figur 30 for måling av signal målt ved 120kHz.



Figur 30 Inngangs og utgangssignal for kretskort
Diode BAS16 120KHz inngangssignal

3.6 Konklusjon og erfaring for analog krets

Etter målinger ved de konstruerte kretser viser simulering i PSpice å være gode. Simulering har vært overens med de målte verdier ved realisering av krets, som av dette øker tilliten til valg av krets for test. Arbeidet innledningsvis baserte seg mye på PSpice og en feil representasjon av kretsoppførsel ville gitt feil grunnlag for å velge de valgte kretsløsningene. Det velges å benytte kretsløsning "B", vist i Figur 8. Etter test viste kretsen seg best egnet for den endelige kretsløsningen, med bl.a. økt uavhengighet i komponentparametere.
Det er i tillegg til valg av kretsløsning vist seg nyttig å benytte nye komponenter ved realisering av analog krets. Dioden BAS16 viste seg pålitelig og jeg ser den som egnet for bruk i det videre prosjektet. Det anses som positivt at dioden er overflatemontert. Jeg velger også å benytte operasjonsforsterker AD845 ved krets for likeretting.
Krets for omhylningskurve implementert i kretskort II, gir indikasjon på gode resultater. Resultater ved målinger gjort i ettertid viser til at stigetiden er forbedret i forhold til en integrator krets. Forøvrig anses det å være for liten tid til å gjøre flere erfaringer med denne kretsen. Jeg velger derfor å nedprioritere denne kretsen og heller benytte enkel lavpasskrets.

Slutt kapittel 3

4.0 Kretsløsning og kretskort design

Prosjektets andre fase omhandler kretsløsning og kretskortdesign av det helhetlige system. Mye av kretsdesign omhandler digital kretselektronikk og kommunikasjon mot internett. Krets baserer og bygger videre på den analoge krets som ved dette tidspunktet er erfart fra del 3 om analog kretselektronikk.

4.1 Kretsløsning og kretskort design

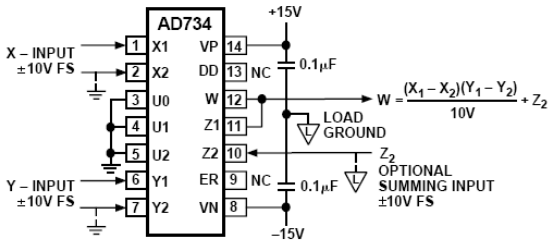
Som innledende forberedelse har jeg gjort vurdering av enkeltmoduler. Modulere er delt opp etter type krets, samt funksjonalitet av kretsen. Jeg har før dette gjort vurdering av systemsammensetning med et helhetlig angrep av problematikk, før forslag til krets er utarbeidet. Dette forarbeidet er endt opp i følgende moduler.

Moduler

- ✓ Spenning
- ✓ Transduser
- ✓ Analog krets
    - TVG
    - Filter
    - Likeretter
    - Omhylning
- ✓ Digital krets
    - Mikrokontroller
    - ADC
    - Ethernet
    - (RS232, JTAG)

Ved bruk av ekkolodd, vil det settes krav til egenskaper ved den elektroniske kretsen. Som del av mottagerkretsen vil det derfor være verdifullt å bruke noe tid på utvikling og valg av nytt kretsdesign for en best mulig utnyttelse av inngangssignalet. Krets som tidligere er utviklet benytter enheten EP910. Dette er en EPLD, Erasable Programmable Logic Device, med I\O arkitektur for mulighet for over 50 programmerbare konfigurasjoner. Enheten har 36 innganger og 24 utganger, og kan enkelt programmeres etter brukers definisjon. Intern logisk krets tilpasses ved å foreksempel benytte programmerings verktøyet A+PLUS. Det benyttes 2 stk. EP910 i den eldre kretsløsning. Dette ved generering av burst-signal samt ved digitalt tellersignal til DAC.
For generering av burst-signal, programmeres EP910 til å generere signal ved hjelp av ekstern krystall tilkoblet to av dens innganger. To enheter er altså her benyttet, krystall og EP910. Det medberegnes da ikke de tilhørende motstander og kondensatorer enheten benytter for funksjonalitet. Denne del-kretsen vil kunne erstattes av en mikrokontroller, hvor burst generering vil være en av flere oppgaver mikrokontrolleren vil kunne håndtere. Videre i den eldre kretsen benyttes ny EP910 i serie med en D/A konverter for generering av signal til TVG kretsen. Med en NE555 timer chip, genereres klokkefrekvens, som videre klokker EPLD kretsen. Denne er programmert til å telle digitalt inn til den digitale D/A konverteren. Stigende spenning genereres av dette med stigningstall avhengig av konfigurering av NE555 for valg av klokkefrekvens. Ved TVG justering av signal benyttes multiplyer AD734. Dette er en analog multiplyer, med flere funksjoner, avhengig av konfigurasjon av eksterne komponenter. I det tidligere benyttede kretskortet er AD734 konfigurert til å multiplisere inngangssignalet med stigende TVG spenning fra

nevnte krets, for så å divideres på 10V. Med inngangs signal X og Y, er funksjon og tilhørende oppkobling fra datablad oppgitt som;



Figur 31 AD734, enhet for TVG-justering

Z_2 kobles da til jord. Denne enheten er en god løsning, men vurderes til å være priset noe i overkant av hva som er ønskelig. Januar.08 oppgir Farnel en kostnad på 417kr for AD734. Ved bruk av mikrokontroller vil kun deler av TVG kretsen kunne erstattes. Jeg anser det som lite egnet å erstatte AD734 med en mikrokontroller. Forøvrig vil mikrokontroller kunne erstatte NE555 timerkretsen, samt EP910 for generering av digital verdi. To komponenter kan altså her forkortes.
Ved det tidligere produserte kretskortet er konvertert og filtrert signal ledet direkte til utgang. Det settes derfor krav til eksterne enheter for nyttegjøring av signalet. Typisk ekstern tilkobling vil kunne være tilkobling til PC, hvor signalet må samples. Jeg vurderer det som hensiktsmessig å integrere digital omforming i kretskortet. ATmega64 har integrert A/D Converter som vil kunne være en måte å realisere dette på. Begrensningen vil for øvrig her være en lav samplingshastighet ved bruk av den maksimale oppløsning på 10-bit. ATmega64 bruker suksessiv tilnærming ved ADC, med maksimale ADC-klokkefrekvens ved 10-bit oppløsning på 200kHz. Det er oppgitt at en konvertering vil ta minimalt 13 klokkepulser, dette om ADC allerede er påskrudd og at aktuell konvertering er en etterfølgende konvertering. En første konvertering er oppgitt til å ta 25 klokkepulser. Ved en konvertering på 13 klokkepulser, vil mikrokontrolleren bruke;

$$\frac{1}{200kHz} * 13 = 65\mu s$$

Figur 32 Konverteringstid ADC ved µC

Ved å benytte en burst frekvens på 120kHz, vil et likerettet signal opptre som den doble frekvens. Ved å sample en gang hver periode av 120kHz, vil annenhver likerettet signaltopp falle inn under et sampel. Da omhylningskurven til det likerettede signalet er det faktiske signal som ønskes samplet, vil denne sampelfrekvensen med stor sannsynlighet være tilstrekkelig. Altså bør et sampel gjøres med intervall;

$$\frac{1}{120kHz} \approx 8{,}33\mu s$$

Figur 33 Utregning av sampelfrekvens

Som fremkommer av Figur 32 og Figur 33, gir ATmega64 inntrykk av å være tregere en hva som er ønskelig for konvertering av analogt signal ved 10-bit oppløsning. Det kan derfor drøftes hvorvidt mikrokontrolleren er egnet for å erstatte en A/D Converter. Krav til oppløsning og samplings hastighet vil her være avgjørende argumenter.

Som videre utvikling er det med dette benyttet tid til vurdering av nytt kretsdesign og ny konfigurasjon med eventuelle nye komponenter. En forbedring ved minimalt bruk av komponenter vil bidra i å fornye kretsløsningen. Om det viser seg at endret kretsdesign gir bedre ytelse eller nøyaktighet anser jeg dette som god forbedring. Men det gjøres oppmerksom på at ved kun å benytte mikrokontroller, for på denne måten å kunne fjerne enkelte komponenter i den tidligere kretsløsning, vil dette alene være god forbedring i seg selv. Ytligre arbeid vil derfor være et steg videre i prosessen ved modifisering av sonarsystemet.
Løsningen ved tidligere kretsløsning fungerer godt, men har forbedringspotensial i å implementere nyere kretser. Krets som helhet vil kunne modifiseres ved kun å implementere mikrokontroller og med dette erstatte flere av de benyttede komponenter. Forøvrig har jeg ønske om å drøfte andre nye løsninger for en bredere erfaring av kretsløsninger. Jeg har derfor strebet etter ny tankegang med nye måter å detektere og justere ekkosignalet. Jeg vektlegger det å kunne kontrollere kretsen i størst mulig grad via Ethernet. Det er her balansegang mellom å utnytte de muligheter dagens teknologi tillater, samt å ikke overdrive med unødvendig mange funksjoner. Jeg setter også ønske om kun å benytte én mikrokontroller. Jeg anser det som sannsynlig at det vil være mer egnet å heller benytte FPGA enn flere mikrokontrollere i samme krets. Det er benyttet tid ved prosjektets startfase til å finne ny egnet kretsløsning for justering av TVG. Jeg har ønske om å gjøre dette digitalt, da med kontroll fra mikrokontroller. AD734 som tidligere er benyttet er vist seg godt egnet, med svært liten støy. Negativt anses det at kretsen gjør bruk av analogt styringssignal for justering av TVG. Dette setter krav til flere komponenter. Det er videre skissert ide om å dempe signalet maksimalt ved start, for videre å minke dempningen som TVG-funksjon. Jeg ser her mulighet for heller å løfte signalet noe, avhengig av maksimale amplitudeverdi den påfølgende krets kan håndtere. Dette og andre relevante argumenter innsamles og gir grunnlag for idè-løsning for ny krets. Om det ikke viser seg å kunne oppdrive ny bedre krets, vil også dette anses som en god erfaring for den videre utvikling av systemet i senere tid. Jeg føler innfallsvinklingen vil være relevant i forhold til prosjektets kravspesifikasjon, da denne spesifiserer at ny krets gjerne kan benytte ”dagens nye” komponenter for oppdatering av sonarsystemet. Større deler av de aktuelle kretser er, som nevnt tidligere i rapporten, uttestet for verifisering ved et tidligere tidspunkt. Typisk gjelder dette analog kretsutvikling. Utesting av krets har da vært prosess som innebærer komponent søk, kretskonstruksjon, simulering og i flere tilfeller en verifisering i form av produsert kretskort ved UiO. Videre valg av krets er gjort med vekt i funksjonalitet, kravspesifikasjonen og den tid som er til rådighet før prosjektet skal stå ferdig. Ved dette prosjektet har jeg benyttet CADStar som designverktøy. Her tegnes kretsene inn skjematisk, for så å designes som kretskort med grunnlag i dette. Allerede ved skjemategning er det vektlagt å sammenkoble kretsen på et tidlig tidspunkt. Dette skaper oversikt, i tillegg til en forenklet designprosess og signalgang i PCB. Den modulbaserte innfallsvinklingen beholdes også i denne prosessen og er å kjenne igjen ved det endelige kretskort. Som inntegnet krets har jeg benyttet komponenter fra et bibliotek hentet fra server på UiO. Dette er komponent med skjemafigur og tilhørende footprint. Enkelte av komponentene som er benyttet i

prosjektet har ikke vært å oppdrive i dette biblioteket som jeg derfor har måttet lage underveis. Verktøy for å konstruere ny komponent er gjort i CADStar Library Editor. Her er skjemafigur og footprint tegnet etter komponentens spesifikasjon i datablad. Footprint med tilhørende pad`s og avtegninger samkjøres mot skjemafigur i tekstfil. Eksempel fra eget bibliotek er vist i Vedlegg 4.

4.2 Delkretser

4.2-1 Spenning

Modulen inneholder kretsens inngangsspenning og de omforminger som vil være aktuelle for realisering av etterfølgende kretser. Fra veileder er det gjort ønske om å kunne benytte 12 eller 24 volt. Dette har jeg tatt hensyn til i utviklingen av denne kretsen. Det vil også være relevant å benytte en så støysvak spenningsregulator som mulig, for ikke å forstyrre den øvrige krets mer enn hva som er nødvendig. Som en forbedring av støynivået innlegges filteravkobling ved samtlige spenningskilder.
Totalt er det benyttet tre spenningsomformer kretser. En spenningsomformer er knyttet til kretsens transduser. Jeg har satt ønske om å kunne variere denne slik at sendereffekten fra transduser kan velges av bruker. Det er her benyttet spenningsregulatoren LM317T fra ST Microelectrinics. Dette er en positiv justerbar spenningsregulator med justerbar utgangsspenning 1,2V til 37V. Det konfigureres kobling mot mikrokontroller for å kunne velge ønsket senderfrekvens. Tre effekter vil være tilgjengelig, 10W, 87W og 300W. Oppkobling for variasjon av spenningsregulator gjøres etter datablad, samt å benytte to transistorer, BC846B. Transistorens base kobles opp mot mikrokontroller slik at spenningsregulatorens motstandsnettverk kan varieres for ønsket utgangsspenning. Oppkobling er vist i Vedlegg 5.
Den øvrige krets gjør krav om +/-12V og +/-5V. Dette til respektive analog krets og den digitale krets. Jeg har her valgt å benytte switch regulatorer fra TracoPower. Enhetene viser seg egnede med hensyn på friheten ved inngangsspenningen. TEN5WI serien tillater 9-36V inngangsspenning. Dette øker brukervennligheten og gjør enheten fleksibel. Veileders ønske om mulighet for tilkobling av 12 volt og 24 volt vil av dette være innfridd. Regulatoren har bl.a. et innebygget filter på inngangen samt en høy switchfrekvens. Switchfrekvensen er på 300KHz, en frekvens som er høyere enn hva som vil være aktuelt for kretsens maksimale senderfrekvens fra transduseren. En eventuell støy fra regulatoren vil av dette forventes å være utenfor ønsket målefrekvens i kretsen og derfor blande seg minimalt inn i signalet. Det er ikke vært vellykket å innhente informasjon fra produsent vedrørende anbefalt avkobling av enhetens inn- og utganger. Etter drøfting har jeg derfor valgt på eget initiativ å benytte kapasitans og induktans ved utgangene. Her vil induktansen være noe stor i den fysiske størrelse, men jeg legger vekt på at denne må kunne lastes med den maksimale strøm som spenningsregulator kan gi. Det er valgt skjermet induktans fra CooperBussmann på 680uH. Switchregulatorens ”common”-utganger avkobles med avkoblingskondensator mot den positive og negative spenningsutgang. Det er i tilegg til dette valgt å benytte polarisert kondensator mellom regulatorens forsyningsspenning og forsynings jord. Ved kontakt for tilslutning av inngangsspenning har jeg implementert lysdiode for verifisering av elektrisitet til kretskortet. Jeg har ikke valgt å benytte crowbar eller annen sikring av kretskortet, da jeg ikke anser dette som nødvendig.

4.2-2 Transduser

Det er benyttet transduserkrets som fra tidligere er utviklet ved UiO. Kretsens innganger er burst og shutdown. Disse har jeg lagt til to utgangsporter ved mikrokontrolleren. Mikrokontroller vil av dette kunne enable, samt å bestemme senderfrekvens og envelope. Jeg kan da programmere inn kode slik at bruker kan variere disse via Ethernet. Kretsens tilførselsspenning styres også av mikrokontroller, som nevnt i tekst om spenningsmodulen. Burstsignal fra mikrokontroller mates til PWM-kontrolleren UC3846 via transistoren BC546. Ekkosignal til transduser påtrykkes to BUZ21 transistorer og føres ut gjennom kretsens Tx-utgang. Tx-kretsens effektbehov hentes fra fire tilknyttede kondensatorer på 4700UF. Tx og Rx kretsen har begge komponent produsert ved UiO. Tx bruker trafo viklet ved UiO. Rx benytter induktans viklet også den ved UiO.
Ved Rx delen mottas signalet av transduser og ledes gjennom resonanskrets og over et antall kondensatorer, med antall avhengig av ønsket resonanskrets. For 70kHz benyttes to kondensatorer. Jeg har inntegnet to øvrige kondensatorer i pcb for mulighet til å kunne endre ønsket resonansfrekvens i ettertid. Rx kretsens to dioder begrenser inngangssignal styrken for på denne måten å ikke skade videre etterfølgende elektronikk. Rx kretsens resonansfrekvens benyttet videre i prosjektet settes til 70KHz. Test av transduserkretsen er beskrevet i ”4.2-3-1 Test av LM1972, TVG med RX-Tx krets”. Kretsen er her testet ved produksjon av testkort.

4.2-3 Analog krets, TVG

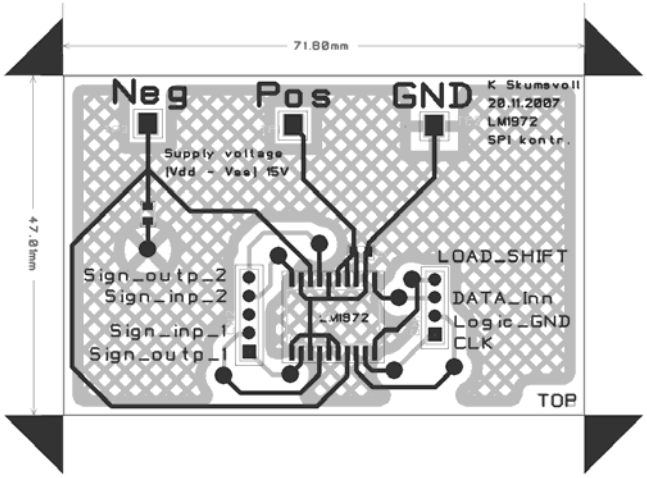
Ved sendt ekkosignal fra transduser, vil det mottatte ekkosignal måtte justeres etter tap av signalstyrke p.g.a. tilbakelagt avstand, temperatur, saltkonsentrasjon i vannmassen m.m. En tidsvariabel forsterkning bør benyttes, ved å kontinuerlig justere inngangssignalet med hensyn på tid. Jeg har vurdert flere løsninger for valg av krets for TVG-justering. Endelig valg er endt på enheten LM1972, en SPI kontrollert 78dB audio demper. Etter å ha produsert to testkretskort, har jeg funnet grunnlag i å utprøve enheten i et endelig design. Test er beskrevet i etterfølgende kapittel.
Mottatt ekkosignal fra resonanskretsen påtrykkes operasjonsforsterker, OPA228 fra BurrBrown. Signalet kobles som ved test av LM1972, ved å benytte LM1972`s ene kanal som kontrollert forsterkning ved tilbakekobling mot operasjonsforsterker. Utgangssignal påtrykkes LM1972`s andre kanal for TVG-modifisering av ekkosignal. Hensikten med dette er å kunne modifisere ekkosignal så tidlig som mulig, ved helst å forsterke signalet. Første kanal kan enten benyttes som en konstant forforsterker sammen med operasjonsforsterkeren, eller som en del av TVG justering i samarbeid med den andre kanalen. TVG-justeringens yttergrenser kan av dette justeres ved enten å løfte det totale signalet, eller ved å dempe det. Ved at jeg kobler kontroll av LM1972 mot bruker via Ethernet, vil dette gi en stor fleksibilitet ved den enkelte situasjon og brukersted. Er det ønsket fokus på f.eks. en gitt fiskestørrelse eller ønsket avstand, vil signalstyrken her kunne justeres etter hva bruker finner mest hensiktsmessig. LM1972 lar seg programmere til via MOSI, samt å leses fra via MISO. For økt kontroll, legger jeg inn mulighet for både MOSI og MISO kontroll fra mikrokontroller. Jeg forventer i hovedsak å benytte MOSI konfigurasjon fra mikrokontroller.

4.2-3-1 Test av LM1972, TVG med RX-Tx krets

Etter søk og drøfting av kretsløsninger, har jeg fattet interesse i enheten LM1972 fra National Semiconductor. Enheten virker lovende ved kontroll av TVG justering for det innkommende signal fra transduser. Enheten har to signalinnganger og to signalutganger. Kanalene styres serielt via SPI, med kontrollerbar dempning fra 0.0dB til 78dB, hvor enheten som egenskap i tillegg til dette kan dempe signalet 100dB (”mute”). LM1972 er en av tre modeller i serien. LM1971 har én signalinngang med 62dB som maksimale dempning. LM1971 har ”mute”-funksjon med dempning 102dB. LM 1973 har tre kanaler for signalgang, med 76dB som maksimale dempning. LM1973 har ”mute” funksjon med 104dB dempning. Jeg har valgt å benytte LM1972 ved utvikling og test av enheten. Det velges å benytte transduser frekvens på 70KHz ved utvikling av kretskort, men hvor det settes krav til mulighet for bruk av 50KHz og 120KHz ved bytte av komponent. LM1972 og LM1973 har begge en frekvensrespons på 100KHz. Det kan her gjøres oppmerksom på at LM1971 har en frekvensrespons større enn 200KHz. Denne vil kunne være egnet ved bruk av 120KHz i transduser. Det finnes også argument i å benytte to kanaler i dette forsøket slik at tilbakekobling til operasjonsforsterker kan forsterke Rx signal om ønskelig. Tilbakekoblingen gjøres da via ledig kanal i LM1972.

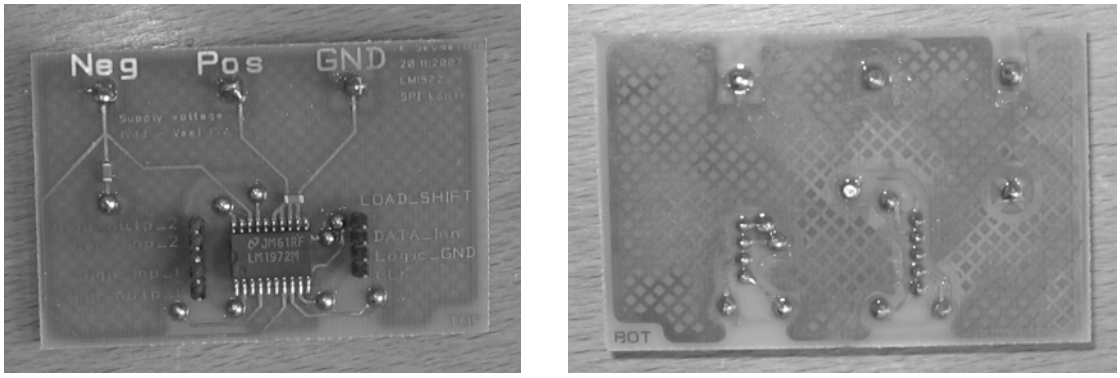
Ved å kunne kontrollere TVG og et eventuelt forsterker ledd, stiller det totale kretskortet seg mer fleksibelt enn hva som tidligere har vært mulig. Med digital kontroll vil disse funksjonene kunne kontrolleres gjennom mikrokontroller og over Ethernet via UDP-pakker. Bruker kan da justere etter behov og tilpasse for en best mulig utnyttelse av signalene. Jeg anser dette som verdifullt.

Som innledende forsøk ønsker jeg å teste enheten LM1972 for funksjonsverifisering. Det ønskes å produsere testkort for utprøving og bruk. Jeg tegner testkort som sendes til produksjon ved Elab. UiO. Jeg velger å benytte "hatch" jordingsplan for skjerming av enheten. Avkoblings kondensatorer benyttes mellom spenningskilde og jord. Enkle pinner inntegnes som tilkoblings punkt mot mikrokontroller. Layout vises i Figur 34.



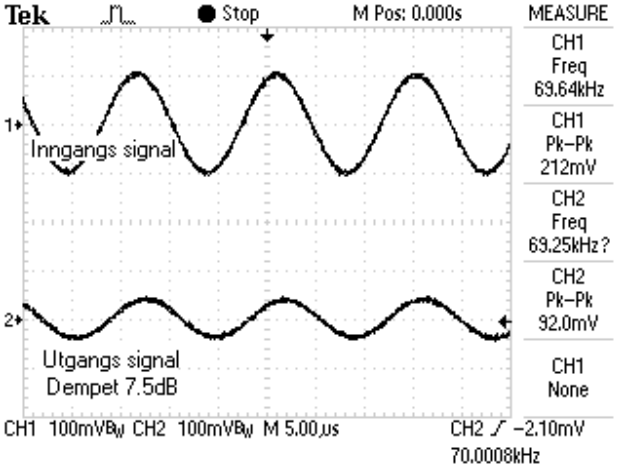
Figur 34 PCB av kretskort for funksjonstest av LM1972

Etter kretskort fremstilling benytter jeg 'Smart Dispense' fra MARTIN for påføring av lodde paste, for så å kunne plassere komponentene. Det benyttes ovn for smelting av paste slik at komponenter loddes fast. Via`s og koblingspinner loddes i ettertid for hånd. Ferdigstilt kretskort vises i
Figur 35.

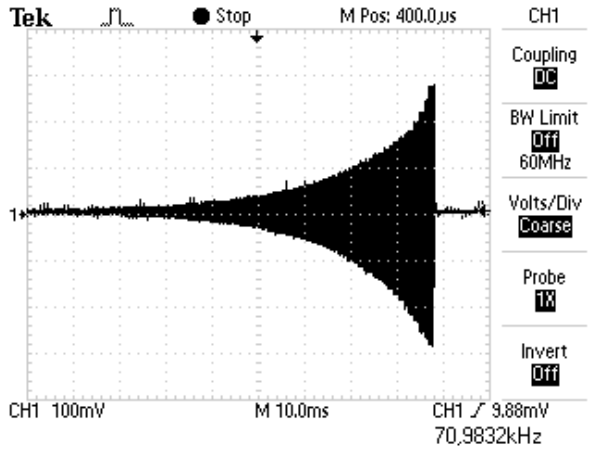


Figur 35 Produsert testkort for funksjonstest av LM1972

For enkel test av LM1972 benytter jeg kretskortet STK500 med mikrokontroller ATmega16, fra Atmel. Det benyttes signalgenerator 33120A fra Hewlet Packard. Målinger leses via oscilloskop TDS1002, fra Tektronix. Variabel spenningskilde til LM1972 benyttes for å kunne observere oppførsel ved forskjellige drivspenninger.
Jeg utvikler software program med manuell kontroll av dempning, samt to automatiske dempnings sykluser via knapper på STK500 kortet. LM1972 lar seg da kontrollere ved manuelt stepp i begge retninger av skala. Automatisk aktivering kan med dette styre LM1972 på to måter. Første modus varierer mellom maksimale og minimale dempning i sinus syklus. Den andre modusen demper som sagtann syklus etter reell TVG oppførsel. Dette er oppførsel ved utgangssignalet, ved at dette dempes maksimalt og steppes fortløpende ned til minimal dempning, for så å igjen starte direkte med maksimal dempning igjen. Dette er syklus som vil bli benyttet i det endelige produkt. Det programmeres inn mulighet for resetting av LM1972, hvor enheten resettes til null dempning. Samtlige valg gjøres ved å trykke knapper påmontert STK500 kittet. Programkode i "C" er vedlagt i Vedlegg 6.
Ved enkel test av LM1972 virker enheten aktuell for bruk i den endelige kretsløsning. I Figur 36 vises dempning på 7.5dB. I Figur 37 vises måling med TVG-justering av konstant 70kHz inngangssignal. Signalet dempes maksimalt for så å steppes ned mot null dempning. Jeg har ikke observert synlig rippel ved overgang mellom demper steg. Frekvens som vil være aktuelt for en kommende krets, gir inntrykk av å være innenfor hva enheten kan håndtere. Med de kabler og testapparater tilgjengelig ved dette forsøket, er det knyttet usikkerhet til LM1972`s egenskaper ved generering av støy og enhetens oppførsel ved minimale inngangssignaler. Måling av disse parametere er ikke kunnet fullføres da testutstyr tigjengelig og oppsett av dette, ikke innehar egenskaper som gjør slik test mulig. Dette var forøvrig kjent før avgjørelse om gjennomføring av testen ble avgjort.

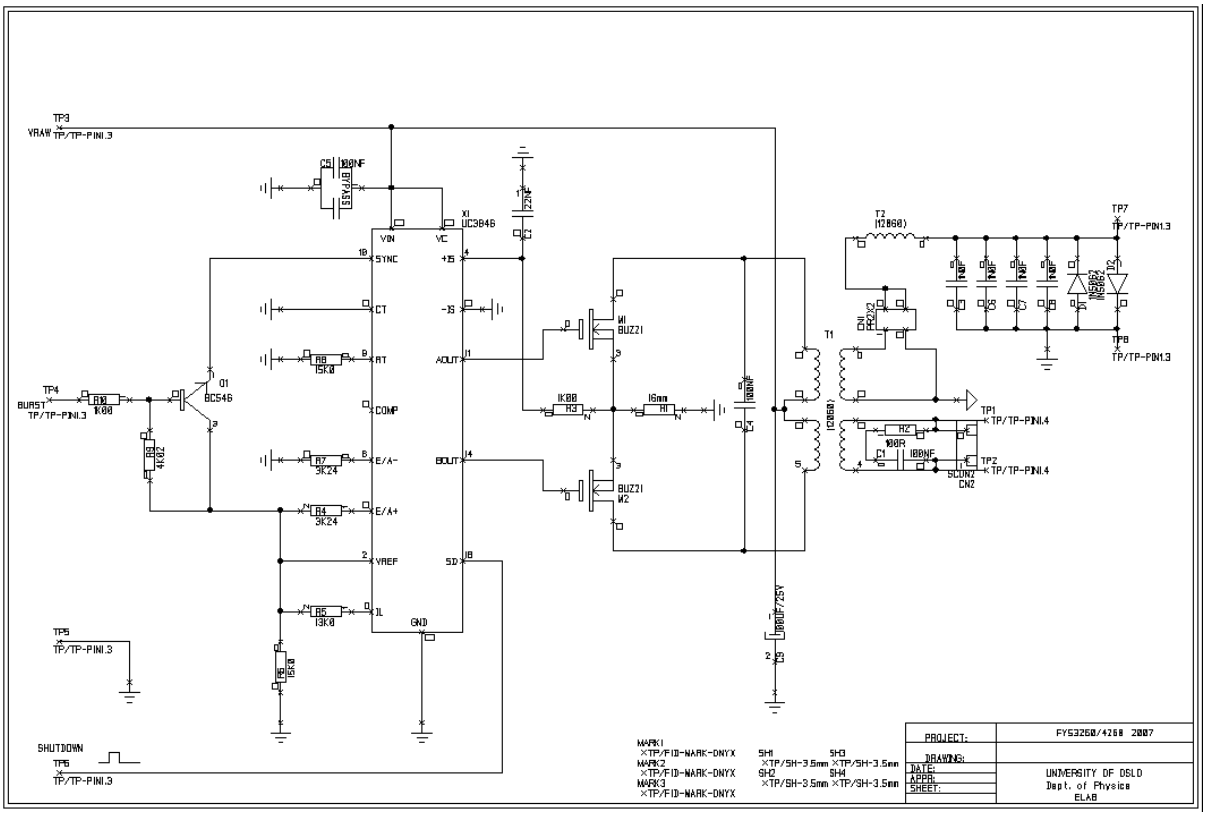


Figur 36 Test av LM1972 med 7,5dB dempning



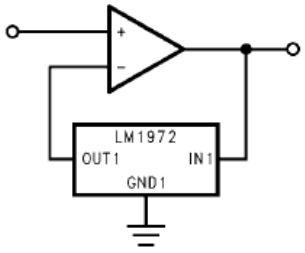
Figur 37 Måling av TVG-justert signal

Videre ønsker jeg erfaring med enheten tilknyttet Tx-Rx krets for transduseren. Transduserkrets er tidligere utvikles ved UiO. Denne kretsen kopieres inn i nytt design, hvor det gjøres enkle modifiseringer for tilpasning i nytt design.
Transduser kretsen som er vist i Figur 38 benytter PWM kontrolleren UC3846 som styrende generator ledd for transduser. Kretsens ”burst” inngang ledes via transistor til PWM kontrolleren. ”Burst” signal sendes videre til to effekttransistorer. Signalet sendes med dette til transduserens Tx utgang. Ved bruk av PWM kontrolleren vil signalet sendes ”glitch” fri og ryddig mot effekttransistorene. Sendersignal vil av dette forsterkes fra signalgenerator til Tx utgangen på transduser. Signalgenerator vil i mitt endelige kretsdesign være mikrokontroller. PWM kontrolleren har inngang ”shutdown” som setter UC3846 enten i dvalemodus med automatisk restart, eller latche inngangssignal av. Denne legges til pin for manuell inn- og utkobling ved testkortet. Ved Rx inngangen i transduserkretsen vil signalet justeres noe i form av forsterkning. Designet med induktans og kapasitans, vil fremheve mottatt signal ved kretsens resonansfrekvens. Resonansfrekvens til inngangskretsen justeres lik senderfrekvens brukt mot transduser.

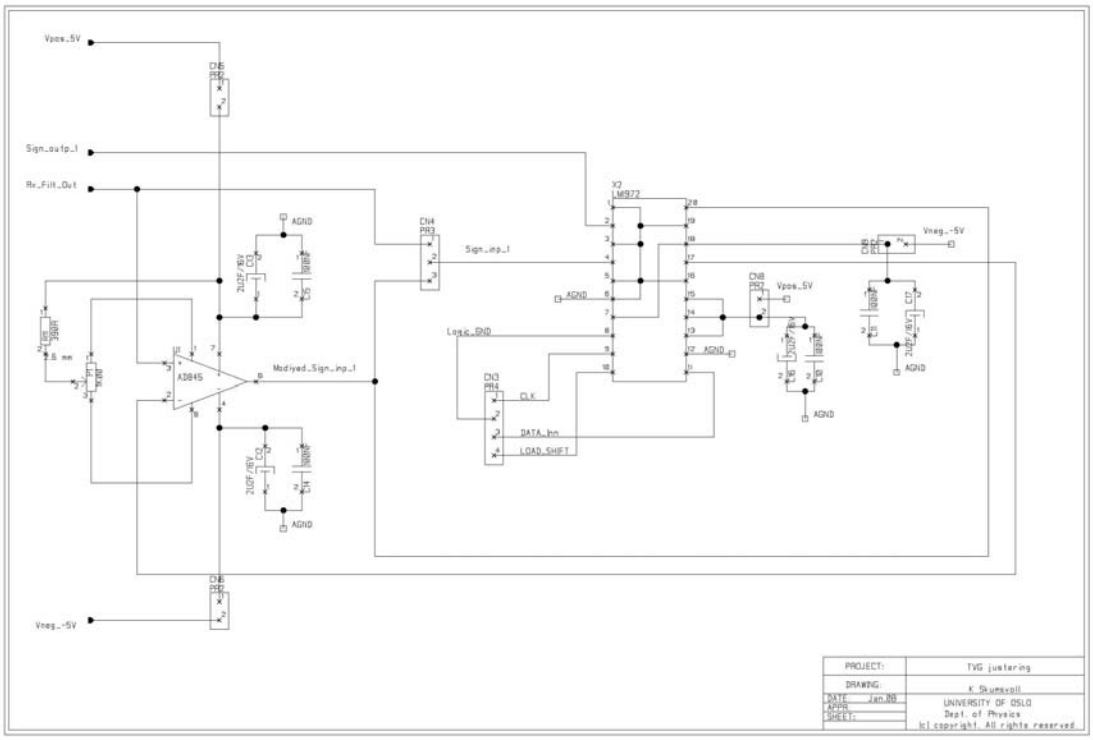


Figur 38 Skjemategning av transduserkrets

Jeg implementerer LM1972 i transduserkretsen med bruk av begge kanalene i µ-potmeteret. Én kanal for kontrollert dempning av TVG og én kanal for kontroll av forsterkningen til inngangssignalet. Det designes kontakt slik at bruker kan koble inngangssignal direkte til kanal én, eller koble inngangssignal via tilbakekobling mot operasjonsforsterker. Inngangssignalet vil da kunne forsterkes via LM1972`s ledige kanal, før det føres til første kanal i LM1972 for TVG-justering. Det implementeres for bruk av to typer operasjonsforsterkere. Operasjonsforsterkeren LT1037, som ble benyttet i den eldre kretsløsning, samt en nyere operasjonsforsterker, AD845. LT1037 er en meget støysvak operasjonsforsterker fra Linear Technology. Den har gode egenskaper egnet for kretsen og vurderes derfor også ved dette nye designet. AD845 er utviklet av Analog Devices og innehar også gode egenskaper. Operasjonsforsterkeren er benyttet under utvikling av likeretter krets tidligere i oppgaven. Tilbakekobling mot operasjonsforsterker gjøres etter henvisning fra datablad LM1972. Prinsipp vises i Figur 39. Mitt inntegnede koblingsskjema er vist i Figur 40.



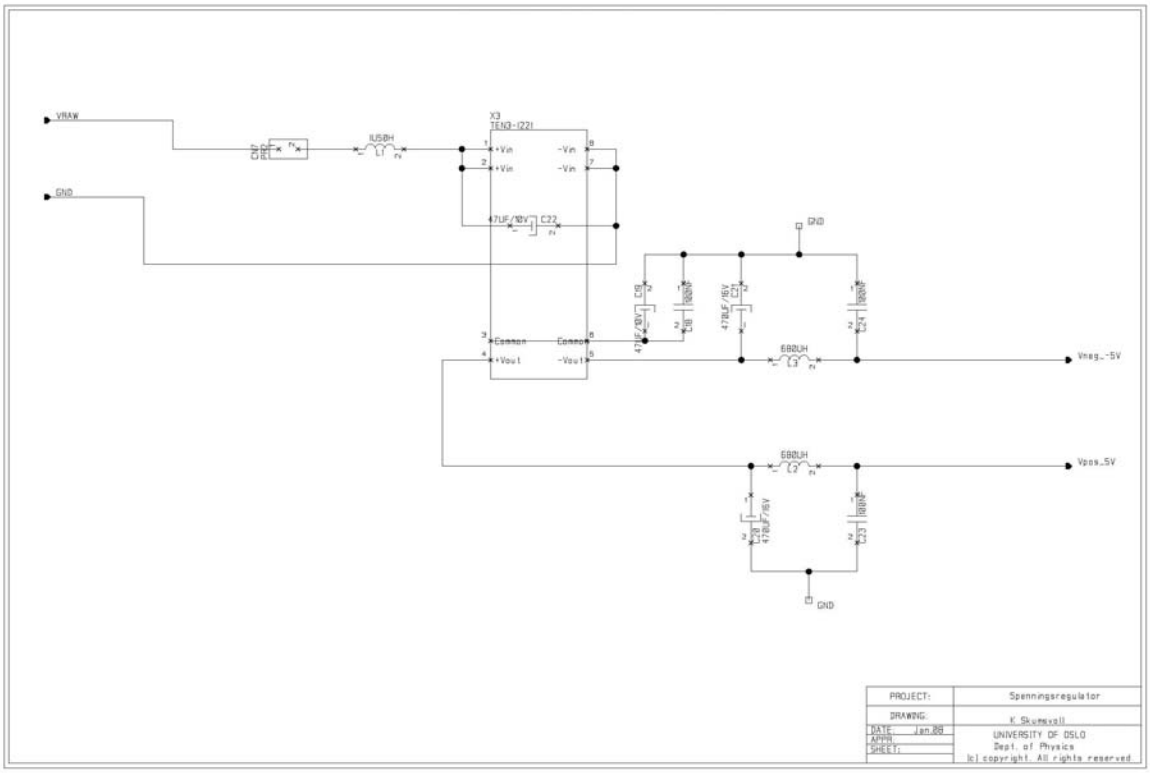
Figur 39 Kontrollert tilbakekobling med LM1972



Figur 40 Koblings skjema for krets med bruk av LM1972

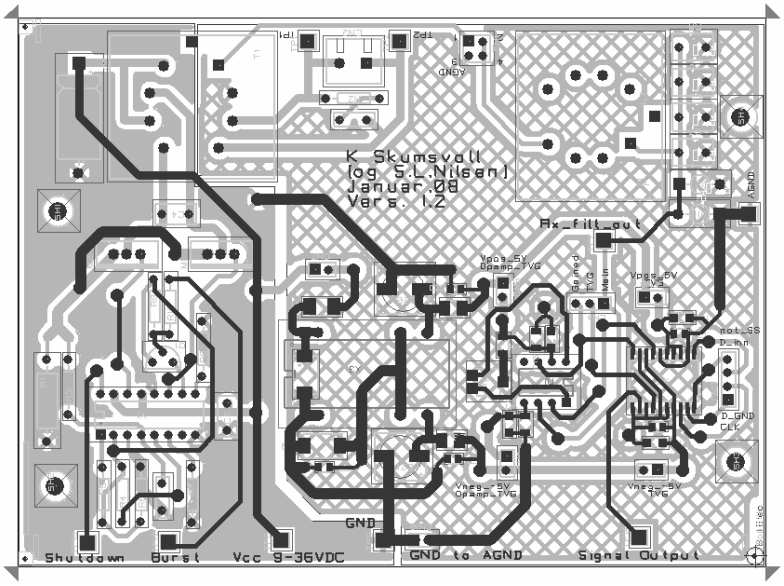
Som inngangsspenning til et endelig kretskort vil spenningen være 12VDC eller 24VDC. Jeg gjør derfor arbeid for implementering av en spenningsregulator for spenningsforsyning til operasjonsforsterker og LM1972. Dette setter for øvrig krav til en støysvak spenningsregulator, med egnede tilkoblinger for ytligre minimering av en eventuell støy. Det velges +/-5VDC som ønsket nedregulert spenning for drift av LM1972 og operasjonsforsterker. Etter søk, viser som nevnt en switchregulator å kunne være et egnet alternativ. Jeg ønsker denne testet.
En høy switchfrekvens vil være gunstig, og to alternativer vurderes. TEN3 og TEN5 fra Traco Power. TEN3 er en 3W spenningsomformer med switchfrekvens på 300KHz. Ved valg av denne, settes krav til inngangsspenning. Om inngangsspenning er 12VDC er TEN3-1221 aktuell, med +/-5VDC på utgangen. Er inngangsspenning 24VDC er TEN3-2421 et alternativ, også denne med +/-5VDC. Valget faller derfor på TEN5-2421WI, med +/-5VDC utgangsspenning. Jeg har tatt dette valget med grunnlag i at denne kan tilkobles inngangsspenning 9VDC til 36VDC. Ved å kunne benytte både 12VDC og 24VDC, vil dette gi en god fleksibilitet. Dette er en egenskap som anses som viktig. Jo større fleksibilitet og robusthet, jo bedre. TEN5-2421WI er en 6W spenningsregulator, med switchfrekvens på 300KHz. For minimering av støy avkobles utgangene med to kondensatorer og en spole. Èn tantal kondensator og èn keramisk kondensator mot common, med induktans mellom disse i signalgang. Ved inngangen har jeg benyttet spole i signalgang, med avkoblingskondensator mellom inngangsspenning og GND. Dette anses som noe i overkant av hva som er nødvendig, da TEN5 har innebygd filter på

inngangen. Det benyttes også induktans ved inngangsspenningen. Kobling av TEN5-2421WI vises i Figur 41.

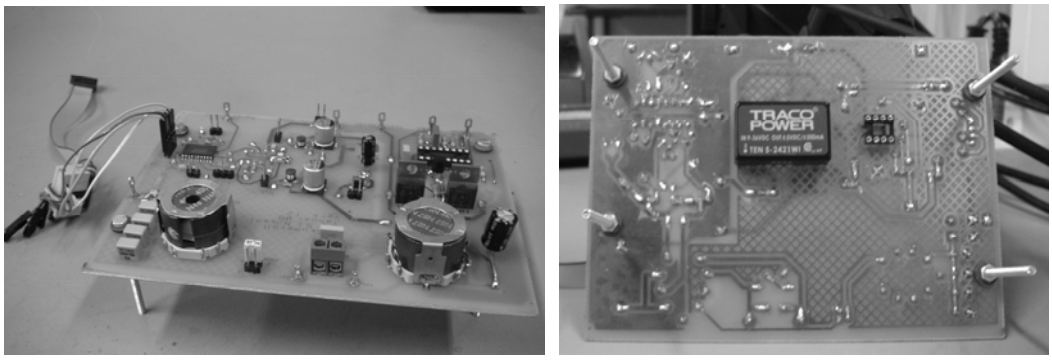


Figur 41 Oppkobling av switchregulator

Om spenningsregulator ikke viser seg å være tilstrekkelig støysvak, vil dette kunne hindre videre test av kretsen. Jeg legger derfor inn valg for bruk av TEN5-2421WI switchregulator eller tilkobling av ekstern støysvak spenningskilde. Mulighet for total avkobling og isolering av spenningsregulator er dermed tatt hensyn til. Valgmulighet av spenningskilde legges også inn i kobling mot operasjonsforsterker og LM1972.
Testkort tegnes og produseres ved UiO. Utlegg er vist i Figur 42. Påmontering av komponenter gjøres med manuell påføring av loddepasta. Overflatemonterte komponenter legges på kretskort og stekes i ovn for smelting av loddepasta. Avslutningsvis loddes øvrige komponenter for hånd. Endelig kretskort er vist i Figur 43. Test av kretskortet utføres etter utarbeidet plan.



Figur 42 Testkort med LM1972 og switchregulator



Figur 43 Produsert kretskort fra PCB vist i Figur 42

Testplan

Test av TEN5-2421WI

For verifisering av TEN5 utføres enkel måling. Det påtrykkes økende spenning på inngangen, og utgangsspenning måles. Det noteres om enkelte spenninger fremprovoserer støy over hva jeg definerer som over akseptabelt nivå.

Følgende måles:

	9V	10V	15V	20V	24V	30V	36V
V pos	*5,05V*	*5,05V*	*5,05V*	*5,00V*	*5,03V*	*4,95V*	*5,60V*
V neg	*-5,01V*	*-5,01V*	*-5,01V*	*-4,90V*	*-4,96V*	*-5,00V*	*-5,52V*
Støy	*ca 20mV*	*ca 20mV*	*15mV*	*15mV*	*20mV*	*20mV*	*10mV*

Tabell 1 Tabell med oppførte målte verdier

Kommentarer:
Ved 36VDC observeres spenningen avvikende fra ønsket utgangsspenning. Som fremkommer av Tabell 1, klatrer spenningen. Stigningen er ikke kritisk høy, men er nær maksimale spenning mikrokontrolleren kan tåle. Ved å la probene fra oscilloskop måle mot egen GND uten eksternt inngangssignal eller innblanding fra krets, måles støy til ca 20mV. Oppførte støyverdier i Tabell 1 er av dette ikke relevante for kretsens egenskaper. Det kan ikke vises til generert støy fra spenningskilde utover de målte 20mV i jord.

Konklusjon;
TEN5 fungerer tilfredsstillende etter de måleinstrumenter som er benyttet. Inngangsspenning anbefales ikke høyere enn 35VDC. Dette med argument i erfart avvik ved påtrykt 36VDC.

Om spenningsregulator viser seg å fungere tilfredsstillende, kobles inn LM1972. Videre test av testkortet vil utføres ved 5V fra spenningsregulatoren. Fungerer ikke TEN5 etter ønsket krav, kobles inn ekstern støysvak spenningsforsyning.
For videre vurdering av LM1972, benyttes testkort STK500. Mikrokontroller påfestet kortet, ATmega16, er ferdigprogrammert for kontroll av LM1972.

Test av LM1972.

Ved det produserte kretskortet er 3 stk. pin implementert ved Rx utgang mot LM1972, Main-TVG og Main-Gained. ”Main” refererer til hovedinngangens signal fra Rx. Signal passerer da LM1972 via kanal 1 og videre ut til utgangen på kortet. Main-Gained kobler signal fra Rx via operasjonsforsterker, med LM1972 som demperledd i negativ tilbakekobling, som vist i Figur 39. Verifisering av utgangssignal avleses på pin ”Signal Output”.
Kanal 1 benyttes først som innledende test. Denne vil ikke kunne inneha støy fra kanal 2.

Kommentarer:

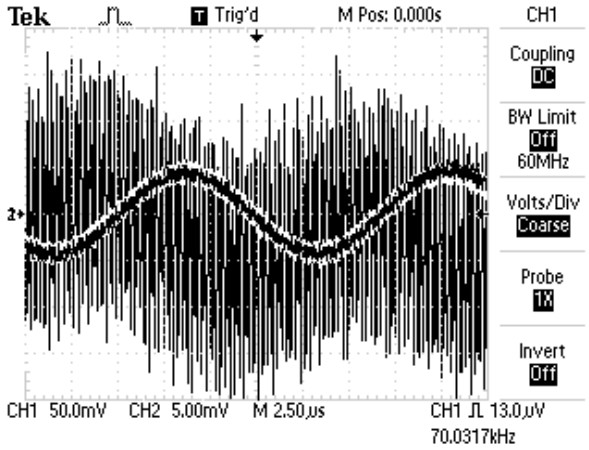
Test startes med rask kontroll av Rx leddet. Jeg ønsker å erfare egenskaper og oppførsel, før signal via denne kretsen påtrykkes LM1972. Etter kontroll av resonansfrekvens testes LM1972.

Det påtrykkes signal fra signalgenerator slik at funksjonstest kan utføres, da uten modifisering fra enheten LM1972. Resonanskrets imponerer med gode egenskaper knyttet til gjengivelse av ønsket signal.

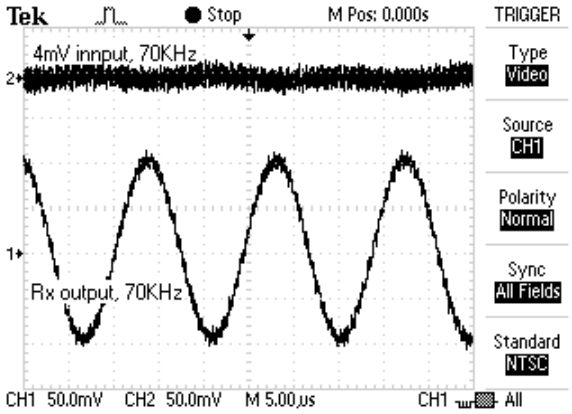
Signal gjennom LM1972 virker ikke å bidra med støy. Det kan ikke påvises et høyere støybilde på signal fra LM1972 enn hva som måles fra signalgenerator. Noe støy skyldes de testapparater som er tilgjengelig. Som erfart ved første testkort med LM1972, kan det ikke observeres synlig rippel eller andre forstyrelser under drift av enheten.

Ved videre test modifiserer LM1972 inngangssignalet. Pin valg retter inngangssignal direkte til kanal 1. Signalet vil dermed dempes før det ledes til måle pin. Det påtrykkes maksimal verdi 180mV, samt minste tilgjengelige generatorsignal på 4mV. Signal påtrykkes Rx inngangen. På denne måten testes LM1972 opp mot Rx kretsen på en mer reell måte.

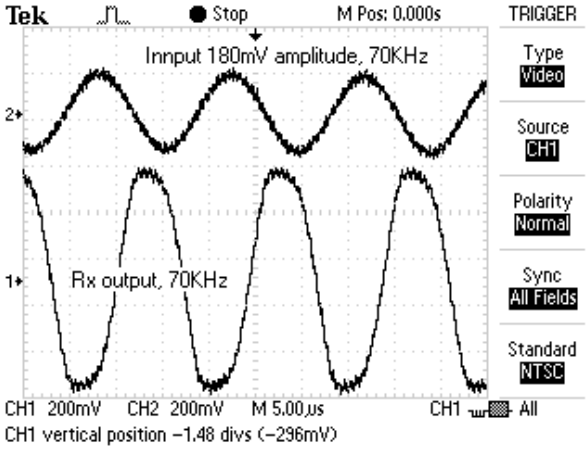
Det kan ikke måles endring i støybildet mellom Rx utgang og LM1972. Rx kretsen imponerer med sin evne til å fjerne støy og fremheve ønsket signal. Eksempel er vist i Figur 44. Ved å påtrykke forskjellig inngangssignal amplituder, erfares egenskap knyttet til Rx-kretsen. Eksempel på måling er vist i Figur 45 *og* Figur 46.



Figur 44 Støyende inngangssignal og filtrert utgangssignal gjennom Rx ledd



Figur 45 Inngangssignal 4mV, Rx utgangssignal ca 100mV, 70Khz



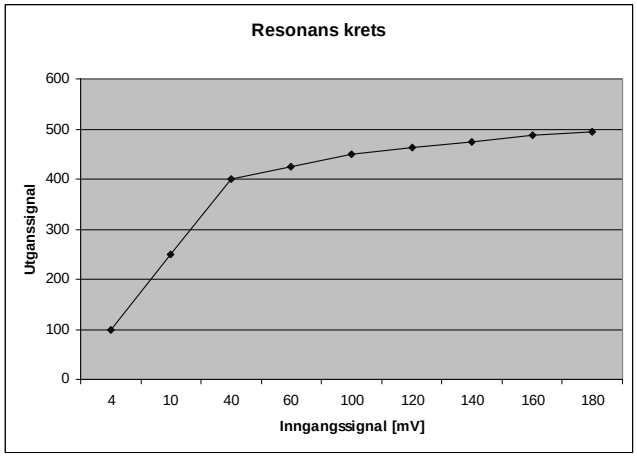
Figur 46 Inngangssignal 180mV, Rx utgangssignal ca 495mV, 70KHz

For verifisering av kanal 2 med tilbakekobling mot operasjonsforsterker, kobles signalvelger til Main-Gained. Signalet forsterkes etter økende dempning av LM1972. Da signalet etter utgangen av operasjonsforsterkeren føres inn til kanal 1 igjen, testes med dette kanalseparasjonen. Datablad oppgir dette til å være 100dB. Det benyttes operasjonsforsterker AD845 / LT1037 ved denne testen.
Kommentarer:
Ved denne målingen ledes signal fra Rx kretsen inn til gjennom LM1972 og til operasjonsforsterker. Som antydet tidligere er dette i negativ tilbakekobling for forsterkning av signalet. Utgangen til denne ledes inn gjennom LM1972 første kanal igjen og ut til pin for måling. Denne kobling er gjort for å strebe etter å fange opp støy fra LM1972. Signal vil med denne kobling passere LM1972 to ganger, en gang for hver kanal. Det er i tilegg til dette spekulert i om dette vil være en mest gunstig måte å koble inngangssignalet på. Å forsterke signalet maksimalt med hensyn på hva første kanal kan

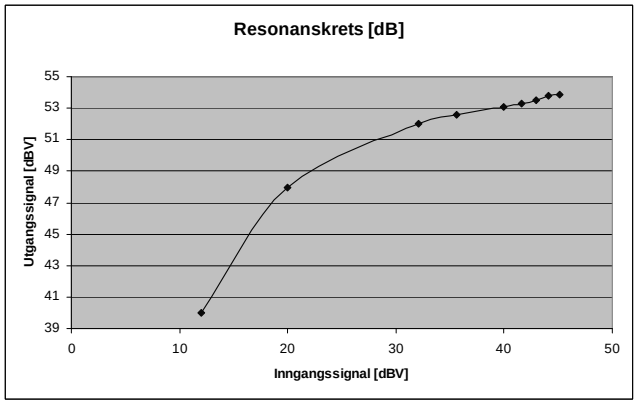
motta av maksimal signalverdi. Det kan ikke måles endring i støybildet etter denne koblingen.

Ved fullførte innledningstest med tilfredsstillende resultat, testes kretsen som helhet.
Kommentarer:
Det ønskes en noe større erfaring med Rx kretsen. Det påtrykkes derfor spenning fra 180mV til 4mV på Rx inngangen, som videre måles etter utgangen etter resonanskretsen. 4mV er laveste signal tilgjengelig fra signalgenerator. Etter utregninger vil det genereres 176,1mV fra transduser, om TS=-30dB og objekt er 3m fra transduseren. Det velges derfor 180mV som maksimal verdi. Rask måling føres i Excel og plottes i graf, vist i Figur 47.

In		Ut Rx	
4	[mV]	100,0	[mV]
10	[mV]	250,0	[mV]
40	[mV]	400,0	[mV]
60	[mV]	425,0	[mV]
100	[mV]	450,0	[mV]
120	[mV]	462,5	[mV]
140	[mV]	475,0	[mV]
160	[mV]	487,5	[mV]
180	[mV]	495,0	[mV]



In dB		Ut dB	
12,04	[dBV]	40,00	[dBV]
20,00	[dBV]	47,96	[dBV]
32,04	[dBV]	52,04	[dBV]
35,56	[dBV]	52,57	[dBV]
40,00	[dBV]	53,06	[dBV]
41,58	[dBV]	53,30	[dBV]
42,92	[dBV]	53,53	[dBV]
44,08	[dBV]	53,76	[dBV]
45,11	[dBV]	53,89	[dBV]



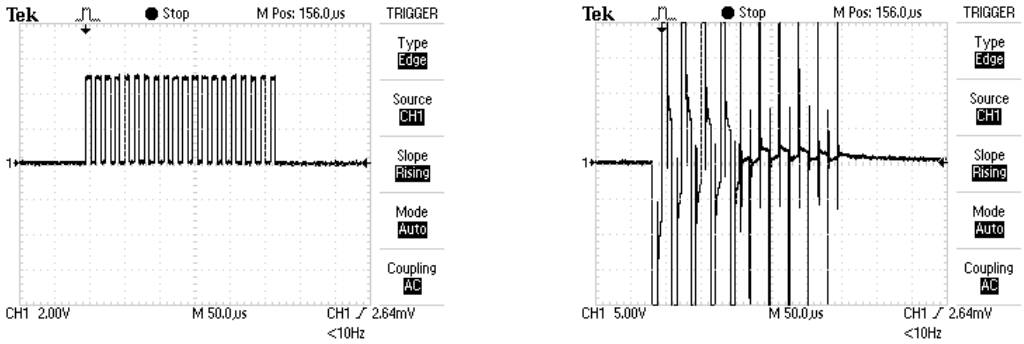
Figur 47 Måling av Rx-inngang og resonanskrets utgangen
Tabell viser inngangssignal og utgangssignal
Verdier er plottet i tilhørende grafer

Signal fra transduser til Rx krets, forsterkes via resonanskretsen. Denne vil være avgjørende for maksimal inngangsspenning for TVG. Ved å forsterke inngangssignalet før justering mot TVG, vil signalet forsterkes så tidlig som mulig. Dette vil redusere krav til kretskortets egengenererte støy. Ved et lavt inngangssignal, vil støy kunne blandes inn i signalet om signalgang passerer komponenter i kretskortet eller signalbaner forøvrig. Signalet av interesse vil med dette kunne gå tapt. Etter gjennomført test med testkortet, gir testen en avbildning av kretsens forsterkning. Dette gjøres med referanse til utregnede data forventet fra transduser ved gitte TS (target strength). Eksempel på utregnede verdier er oppgitt i Vedlegg 2. *De utregnede data gir indikasjon av styrken på signalet som mottas fra transduser etter ekko fra objekt i vannmassen. Tallverdier er satt med sendereffekt 100W. Dette er estimerte data og vil kunne være en kilde for unøyaktighet sammenlignet mot de faktiske forholdene ved en reell situasjon. Sanne verdier må måles i sjø med påmontert transduser. Da antatt objektstørrelse kan variere, anses det som gunstig å kunne gi bruker mulighet for avbalansering ved å selv definere forsterkningen. Ved å fritt kunne justere forsterkningen, vil feilmarginen justeres av bruker under drift. Forhold eller fiskestørrelse vil her kunne være relevante argumenter. Ved utført test via det utviklede testkortet, benyttes spenningskilde +/-5V. Dette gir begrensning i maksimale forsterkning. I oppsettet vil maksimale konstantforsterkning ved maksimal inngangssignal til LM1972`s andre kanal være ca 18,5dBV. Total forsterkning vil da være egenskap ved resonanskretsen, samt forsterkning ved LM1972. Måling mellom Rx inngang og resonanskrets utgangen, er vist i Figur 47. Data er plottet i tilhørende tabell. Utstyret tilgjengelig setter her begrensning i målebredde. Målebredden begrenses med svakeste tilgjengelige påtrykte signalstyrke 4mV på Rx inngangen, samt støy generert i utstyret med de tilhørende komponenter.*

Det observeres en ulineær sammenheng mellom inngangsamplitude og forsterkning i Rx-kretsen.

Etterfølgende justering av signalet vil være å TVG justere signalet for kompensering av transmisjonstap i vannmassen. Maksimal dempning ved 3m vil være ca 62dB. Ved den utregnede maksimale signalstyrke ca 176mV, og konstant forsterkning under +/-5V spenningstilførsel, vil utgangssignal da være ca 4mV. Ved å starte med en forsterkende tilbakekobling via operasjonsforsterkeren, for snart å overta dempning via første kanal, vil TVG-justering via begge LM1972`s kanaler, sammen kunne løfte inngangssignalet til best egnet forsterkning. Dette kan videre varieres etter ønske fra bruker. Metoden vil fungere som en "offsetjustering" av inngangssignalet.

Det er ikke avsatt mye tid til test av Tx utgangen. Oscilloskopbilde med påtrykt burst er vist i Figur 48. Kretsen virker her å fungere etter ønsket funksjonalitet.



Figur 48 Påtrykt 'Burst' fra µC og målt signal på Tx-utgangen

Konklusjon;

Etter gjennomført test gir spenningskilden tilfredsstillende resultater etter hva som er mulig å måle med tilgjengelig utstyr. Svak støy virker ikke å påvirke LM1972 eller operasjonsforsterker, da målt med noe høyere signaler en hva som vil være minimal verdi ved Rx-inngangen.

Transduser kretsens Tx fungerer med tilfredsstillende resultater.

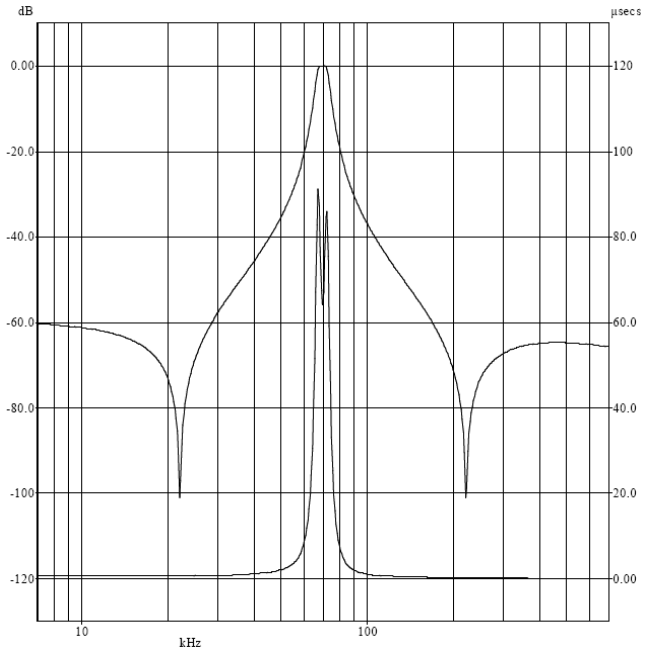
Rx krets med resonanskretsen, gir inntrykk av å fungere tilfredsstillende. Det er indikert en ulineær amplitudeforsterkning som bør testes noe mer ved et endelig kretskort eller et nytt og bedre testoppsett. Støydempning fungerer svært godt i denne kretsen.

LM1972 ser ikke ut til å generere målbar støy. Det er her relevant å gjøre oppmerksom på at de tilgjengelige måleapparater og tilhørende materiell ikke har gjort det mulig å måle de svakeste signaler. Signalgenerator har svakeste amplitude 4mV. Støy i overgang fra testkort til måleapparat vil være begrensende faktor for måling av de faktiske støynivåer i kretsen.

Med grunnlag i overnevnte erfaringer, velger jeg å benytte LM1972 i sonarsystemet. Dette for en reell erfaring. Det fryktes en noe svakere egenskap ved de svakeste ekkosignalene, men enheten overbeviser med mulighet for en god fleksibilitet for det totale systemet. Jeg velger også å benytte switchregulator fra Traco Power.

4.2-4 Analog krets, Filter

Ved tidligere design er det benyttet filter i tilknytning til inngangssignal. Dette er også ved denne oppgaven vurdert, men hvor jeg anser filteret som noe overflødig. Etter test gjort ved produsert testkort, er det gjort erfaring i Rx-kretsen og dens egenskaper. Jeg vurderer kretsen som svært god. Øvrig filtrering av inngangssignalet anses som egnet for filtrering av egenindusert støy av den bakenforliggende krets i forhold til Rx-kretsen. Det er ikke observert støy fra LM1972, men skulle enheten vise seg å avgi forstyrrelser, vil et filter begrense dette.
Jeg har vurdert filterløsning fra Linear Technology, ved simulering i FilterCAD Variasjon i parametere varieres for simulert erfaring av egenskaper. Jeg fatter bl.a. interesse i enheten IC1562. Enheten konfigureres ved enkle ytre komponenter for ønsket filtertype og parameterverdier. Simulering som helhet er vist i Figur 49, med graf av både dempning og responstid.



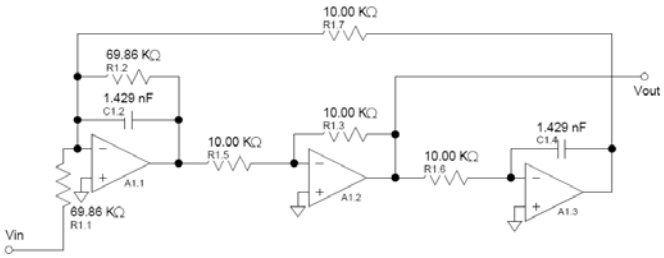
Figur 49 Simulering i FilterCAD med både dempning og responstid
Komponenten IC1562 er simulert.

Kretsen er her simulert med båndbredde 4kHz. Båndbredden er utregnet etter generelt prinsipp, bl.a. nevnt i "Fisheries Acoustics", ISBN 0-412-33060-1". Generell regel setter krav til filtrering uten tap av signal, ved at båndbredden bør være slik at $B\tau = 3$. Er ekkodeteksjonen begrenset av bredbåndet støy, kan båndbredden reduseres til $B\tau = 1$. Her vil noe av signalet bli borte, men hvor signal-til-støy forholdet forbedres. Erfaringsmessig er denne verdi satt til $B\tau = 2$. I kretsen vil τ være 0,5ms. Av dette utregnes båndbredden;

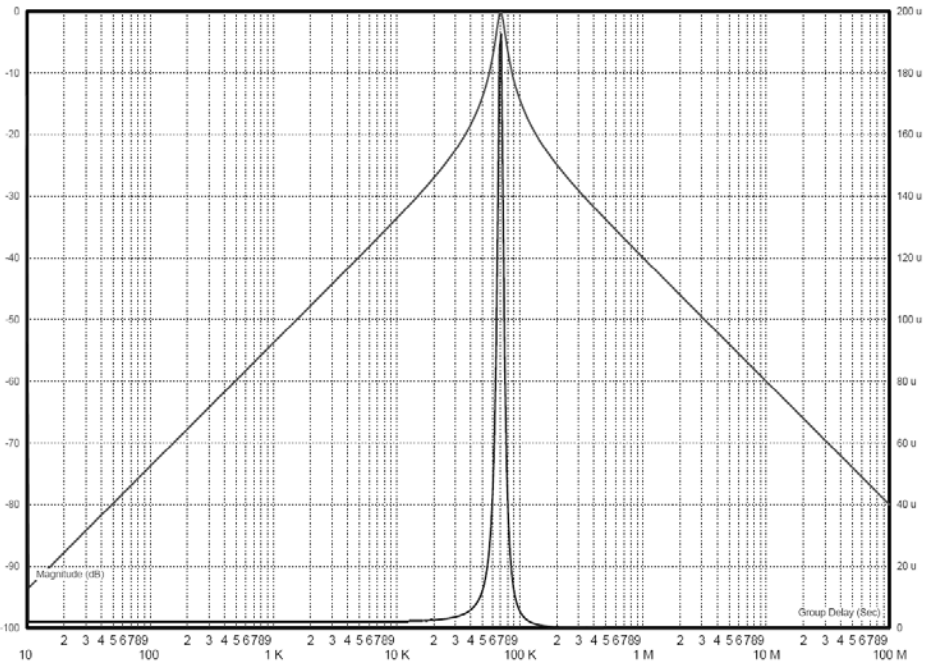
$$B = \frac{2}{\tau} = \frac{2}{0.5ms} = 4kHz$$

Lignende krets med krystall som styrende for filterets egenskap, vil også være alternativ. Filter vil eventuelt kunne kontrolleres fra mikrokontroller som erstatning av krystall, og på denne måten justeres av bruker. Men dette vil okkupere resurser fra mikrokontroller og kunne gjøre krav i å måtte benytte flere enn en mikrokontroller.

Jeg har også vurdert å benytte aktivt SallenKey eller Chebyshev filter. Etter gjenomgang ble Chebyshev filter funnet som egnet og vurdert implementert i kretsen. Filteret har en god respons og er etter justering stabilt med Q=6,9. Som vist i Figur 50 benyttes tre operasjonsforsterkere med et fåtall av ytre komponenter. Simulering av disse er vist i Figur 51.



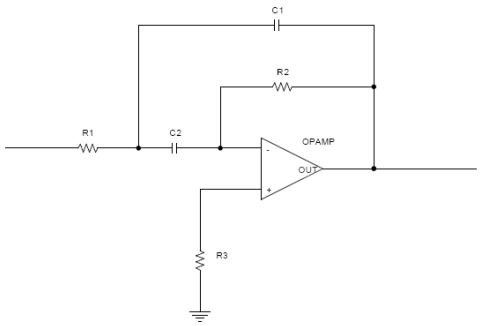
Figur 50 Chebyshev filter



Figur 51 Simulering av krets vist i Figur 50

Etter simulering i PSpice var dette filteret planlagt benyttet i kretsløsningen. I ettertid ble det gjort erfaring med resonanskretsen i Rx-delen, noe som resulterte i at jeg forkastet den planlagte filterkretsen. Rx-kretsens resonanskrets filtrerte inngangssignalet svært godt. Jeg forventer heller ikke å oppleve signifikant mengde støy i de påfølgende kretser. Jeg velger med disse argumenter, et enkelt aktivt båndpassfilter med feedback. Filter oppkobling er vist i Figur 52. Filteret er benyttet for å fjerne eventuell støy fra TVG-

justering, men er som nevnt tidligere noe overflødig. Filteret forventes å kunne elimineres ved en eventuell ny versjon av kretsløsningen.



Figur 52 Båndpass filter

Filteret justeres etter senterfrekvens $f_0 = 70kHz$

$$B = \frac{2}{\tau} = 4kHz$$

$$70kHz - \frac{4kHz}{2} = 68kHz$$

$$70kHz + \frac{4kHz}{2} = 72kHz$$

$$f_0 = \sqrt{68kHz \times 72kHz} \approx 70kHz$$

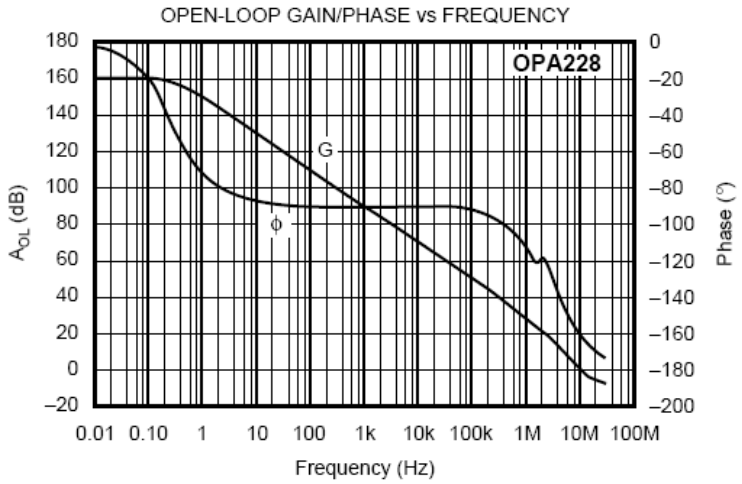
Filterets senterfrekvens velges til 70kHz. Med egenskaper knyttet til filteret, velger jeg en stor båndbredde, avvikende fra generell regel med Bτ = 2. Etterfølgende kretser vil være medvirkende til at avviket er akseptabelt. Med Q = 1,79 forsterkes signalet ca 6,5 ganger. Ved tidligere filterkretser er det benyttet operasjonsforsterker LT1037 fra Linear Technology. Operasjonsforsterkeren har lav støy og gode egenskaper ved implementering i kretser med svake signaler. Operasjonsforsterkeren er i dag ikke å oppdrive, slik at ny enhet må oppsøkes. Etter søk for erstatning av operasjonsforsterker LT1037, er enheten OPA228 valgt som ekvivalent. Enhetens egenskaper vil være avgjørende for valgt Q-verdi. Med forsterkning A = 6,5 utregnes krav til OPA228 i tilknytning til ønsket senterfrekvens. Verdi multipliseres med 10 for å forsikre om stabil funksjonalitet;

$$A = 6,5$$

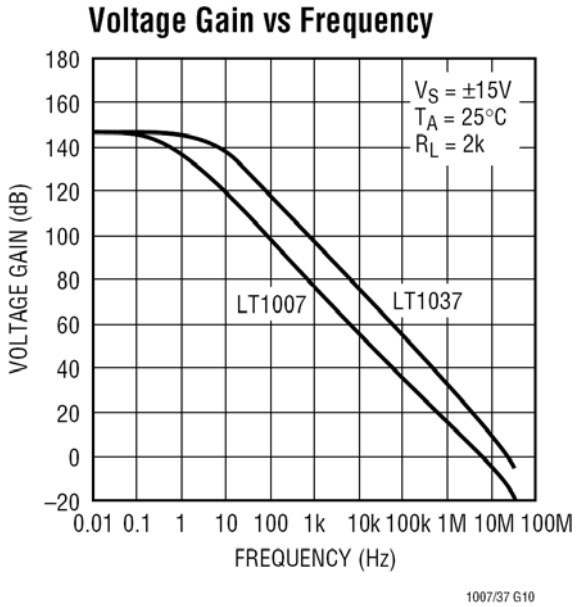
$$f_{gain-opamp} \geq 10 \times A \times f_{senter}$$

$$f_{gain-opamp} \geq 10 \times 70kHz \times 6,5 \approx 4,5MHz$$

Som vist i Figur 53 er dette innenfor OPA228 spesifikasjonen. En høyere forsterkning vil passere de grenseverdier enheten kan prestere. Som observeres i Figur 54, innehar LT1037 noe sterkere egenskaper i forholdet forsterkning - frekvens.



Figur 53 OPA220 karakteristikk hentet fra datablad



Figur 54 LT1037 karakteristikk hentet fra datablad

Med utreget verdi for Q, gir dette en stor båndbredde. Dette vil være avvikende fra generell regel om $B\tau = 2$. Som nevnt er resonanskretsen ved inngangen svært effektivt, samt at påfølgende kretser vil virke dempende på forstyrrende signaler. Med disse egenskaper vil den store båndbredden være relativt irrelevant.

$$A = -6{,}44 = -2Q^2$$
$$Q = 1{,}79$$

$$Q = \frac{f_{senter}}{BW}$$
$$BW = 39kHz$$

Videre utregnes filterets skaleringsfaktor;

$$\omega_0 = 2 \times \pi \times f_{senter}$$
$$\omega_0 = 439822{,}97 rad/s$$

Kretsens kondensatorverdi utreges av dette til å være;

$$C = \frac{1}{\omega_0}$$
$$C = 2{,}27\mu$$

Filterets grunnkapasitans er nå utregnet. Videre utregnes verdier for motstandene, referert i Figur 52.

$$R_1 = \frac{1}{2Q} = 0{,}2786\Omega$$
$$R_2 = 2 \times Q = 3{,}5897\Omega$$
$$R_3 = 2 \times Q = 3{,}5897\Omega$$

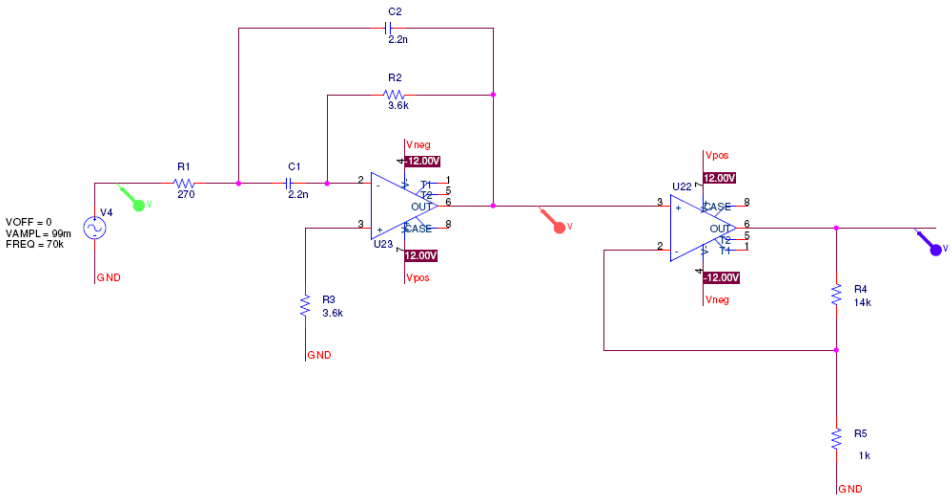
For en fornuftig skalering, justeres verdiene. For å beholde filterets egenskaper multipliseres motstandverdi med 10^3, mens kapasitansverdi divideres med10^3. Verdier mot komponenter i Figur 19 velges med dette til;

$$R_1 = 278\Omega$$
$$R_2 = R_3 = 3{,}6k\Omega$$
$$C_1 = C_2 = 2{,}3nF$$

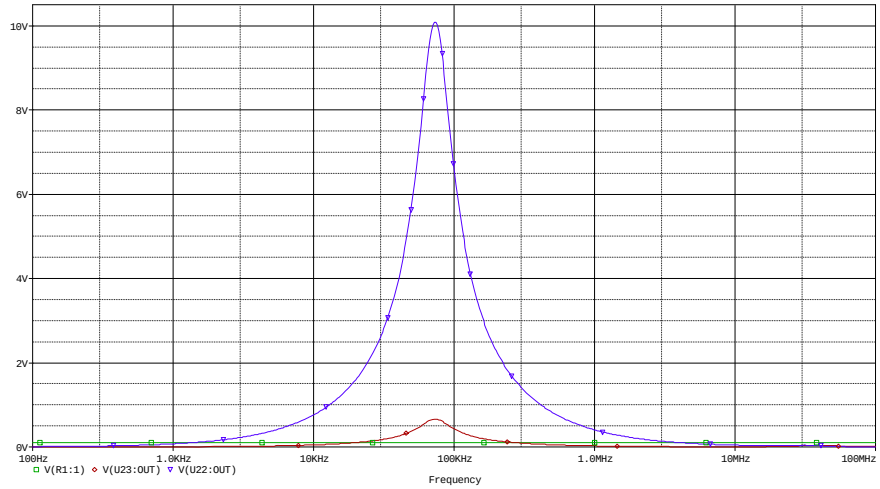
Jeg justerer signalet videre for maksimering av amplitudeverdi. Jeg har skalert påfølgende likeretterkrets til å kunne jobbe med 10V som maksimale amplitudeverdi. Jeg velger operasjonsforsterker OPA228 med ikkeinverterende forsterkning. Etter teoretisk utledning av forventet maksimale inngangssignal fra transduser, utregnes inngangssignalet til å være ca 99mV. Jeg har utregnet signalet fra objekt med TS = -30dB, posisjonert 4m fra transduserflaten med 100W sendersignal. Etter filter vil da signalet teoretisk være løftet til ca 640mV i filteret.

$$signalstyrke = 99mV \times 6{,}44 = 638mV$$

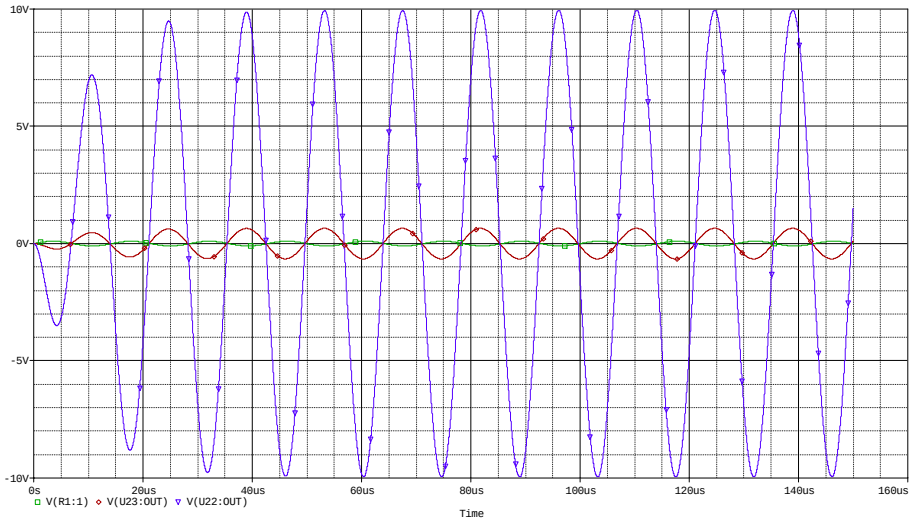
Filterets forsterkning A = 6,44 ønsker jeg å forsterke signalet ytligre. Med forsterkning A_{tot} = 15,6, vil 99mV justeres til tilnærmet 10V. Jeg implementerer ikke inverterende forsterkning med justerbar potmeter for kalibrering av bruker.
Kretsen inntegnes videre i PSpice for simulering. Det har ikke vært vellykket å oppdrive gyldig OPA228 komponent i PSpice. Etter dialog med Texas Instruments, ble jeg anbefalt fra produsenten å benytte deres egenutviklede simuleringsprogram. Etter test av programmet, anser jeg dette som mindre egnet for vurdering av kretsen. Jeg benytter derfor OPA637 i PSpice for simulering i PSpice. Operasjonsforsterkeren har noe svakere egenskaper enn hva OPA228 innehar med bl.a. svakere båndbredde. Dette anses som positivt, da det setter et høyere krav til kretsen under simulering. Jeg simulerer AC Sweep, fra 1Hz til 10MHz. Inntegnet krets er vist i Figur 55. Simuleringsresultat er vist i Figur 56. Fargekode i disse figurene korresponderer med hverandre. Dette er også tilfelle i transientanalysen vist i Figur 57. Figuren viser simulering av inngangssignal på 99mV som tilsvarer TS = -30dB, målt ved 4m avstand fra transduser. Minimalt inngangssignal utregnes til 0,0005mV og tilsvarer TS = -60dB, målt ved 100m avstand fra transduser. Ved begge utregninger er sendereffekt fra transduser satt til 100W. Teoretisk er disse verdier nå detekterbare. Justering av kretsens egenskap vil være mulig via TVG-kretsens justeringsegenskaper over SPI. Jeg legger opp til at bruker skal kunne velge dette over Ethernet. Jeg implementerer potmeter ved forsterkerkobling for manuell kalibrering. Etter hva simuleringen viser til, følger filteret inngangssignalet allerede etter ca 2 perioder. Ved de tidligere vurderte filterkretser er denne innsvinger tiden vært lengre.
Jeg implementerer potmeter ved samtlige operasjonsforsterkere for justering av offset. Flere av disse forventes å kunne fjernes ved en eventuell ny versjon av kretskortet. Feilkilde i uttesting vil bl.a. være at det ikke er benyttet identisk operasjonsforsterker modell i PSpice, men hvor den benyttede modell vurderes å ha noe dårligere egenskaper enn hva ønsket operasjonsforsterker vil inneha. Feilkilde vil også være at dette er en ren simulering, som kan avvike fra de faktiske fysiske forhold. Fra tidligere arbeid i prosjektet viser PSpice å gjengi gode simuleringer i forhold til faktiske reelle målinger. Rask oppkobling på veroboard verifiserer filterkretsen.
Jeg vurderer med dette filterkretsen som tilfredsstillende og implementerer denne i skjematisk kretstegning i CADStar.



Figur 55 Krets for omhylningskurven



Figur 56 Simuleringsresultater med AC-sweep, gjort etter krets vist i Figur 55



Figur 57 Transientanalyse av største forventede signalstyrke for krets vist i Figur 55 70kHz inngangssignal

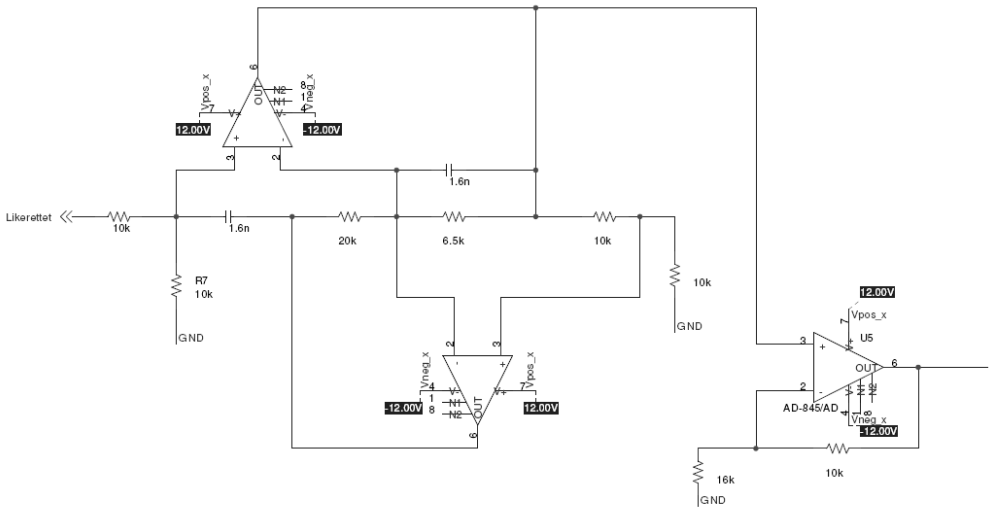
4.2-5 Analog krets, Likeretter og omhylningskurve

Krets for likeretting og omhylningskurve er ved prosjektets innledende fase utarbeidet. Jeg har der valgt å benytte operasjonsforsterker AD845 i likeretterkrets, en rask enhet fra Analog Devices. Ved uttesting av likeretterkretsen har jeg produsert testkort ved UiO for verifisering og funksjonalitets test. Det ble der benyttet operasjonsforsterker i DIL pakke. Denne komponentpakken velges benyttet ved utlegg av nytt kretskort bestilt fra Elprint da jeg ved dette tidspunktet innehar komponenten. Det er verifisert en maksimal operasjonsfrekvens ved likeretterkretsen på ca 250kHz. I dette designet er det valgt å benytte 70kHz som senderfrekvens, med 50kHz som alternativ senderfrekvens. Det er ikke vært tid til arbeid med fullføring av påbegynt ny krets for omhylningskurve. Jeg har derfor av tidsmessige årsaker valgt å benytte andreordens lavpass Besselfilter. Filteret har passbånd frekvens 15kHz, med Q = 0,58. Koblingsskjema med komponentverdier er vist i Figur 58. Jeg verifiserer filterets poler fra systemets transferfunksjon, med resultat i stabilt system. Systemets dempningsfaktor viser seg relativt høy, som vil være fornuftig med hensyn på filterets formål og stabilitet.

$$H(s) = \frac{2{,}25 \times 10^8}{s^2 + s \times 2{,}60 \times 10^4 + 2{,}25 \times 10^8}$$

$$p_{1-2} = -\zeta\omega_0 + /- \; j\sqrt{1-\zeta}\,\omega_0$$
$$p_{1-2} = -12990 + /- \; j7{,}5 \times 10^3$$

$$K = 1$$
$$\zeta = \frac{2{,}6 \times 10^4}{2\omega_0}$$
$$\zeta = 0{,}87$$



Figur 58 Krets for omhylningskurve

For fult ut å nyttegjøre amplitudeverdi, maksimeres signalamplituden med etterfølgende forsterkning. Dette vil opprettholde maksimale amplitude 10V, ved 99mV inngangssignal i transduser. Jeg benytter også her operasjonsforsterker OPA228 ved krets for omhylningskurve og forsterkning. Det implementeres potmeter ved samtlige operasjonsforsterkere for kalibrering. Flere av disse forventes å kunne fjernes ved en eventuell ny versjon av kretskortet.

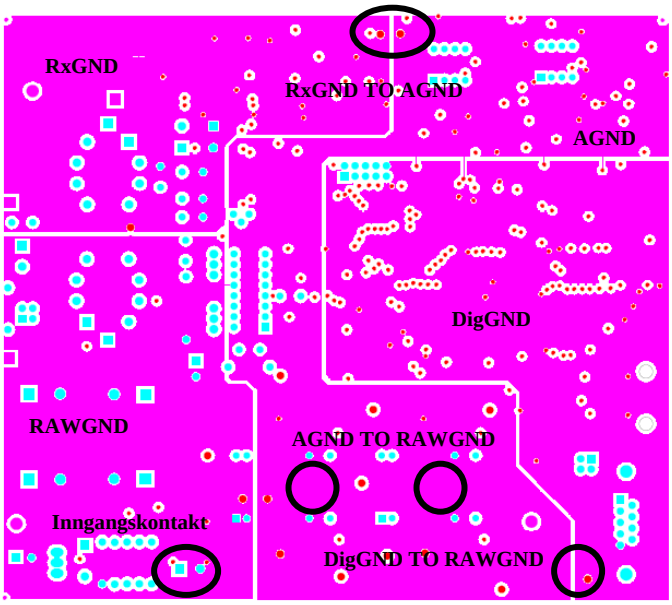
4.2-6 Digital krets, Mikrokontroller, Ethernet kontroller og ADC

Den digitale kretsen består i hovedsak av mikrokontroller, ADC, Ethernet kontroller og RS232 interface. Jeg implementerer JTAG til mikrokontrollerens predefinerte pinner. Som gitt av kravspesifikasjonen, skal det benyttes mikrokontrolleren ATmega64 fra Atmel. Kontrolleren er en 8-bit kontroller, med 64KB flash minne. Jeg velger å benytte maksimale operasjonsfrekvens på 16MHz ved mikrokontrolleren. Det benyttes ekstern krystall for oscillator mot mikrokontroller. For TVG kontroll tilkobles mikrokontrollerens SPI pinner. Det forventes å kun sende data til LM1972 for kontroll av TVG, men hvor jeg allikevel velger å tilkoble MISO til busslinjen for økt fleksibilitet for den kommende programkode. Funksjonaliteten av denne kan eventuelt knyttes til kontroll via Ethernet. I tilegg til kontroll av TVG, benyttes mikrokontrolleren mot 16-bit ADC. ATmega16 har intern ADC, med enten 8- eller 10-bit oppløsning. Da jeg ønsker å benytte Ethernet kontroller i 16-bit modus, vil verken 8- eller 10-bit være optimal for denne. Forøvrig viser 10-bit modus i mikrokontroller å være noe treg i forhold til ønsket samplingsrate. For en mer effektiv flyt i signalgang, velger jeg ikke å mellomlagre samplet data i mikrokontroller. Det vil derfor kun være kontrollsignal mellom ADC og mikrokontroller.
Jeg legger kommunikasjon mot Ethernet kontroller både til og fra mikrokontroller. Bus linje for registervalg og registerverdi implementeres, samt to interrupt linjer fra Ethernet kontroller til mikrokontrollerens external interrupt linjer. Jeg velger å benytte tre

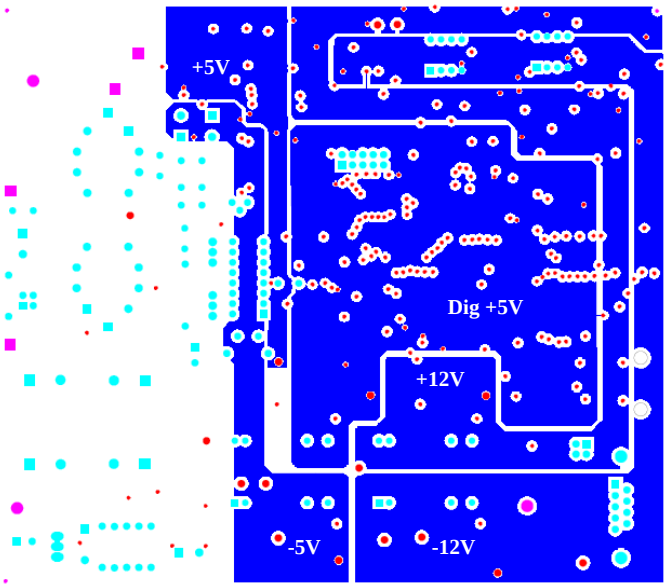
lysdioder tilknyttet mikrokontrolleren, samt reset knapp for manuell reset. Etter oppkobling er det 5 ledige pinner på mikrokontrolleren.
ADC som jeg velger å bruke i prosjektet, AD7610, er en 16-bit konverter fra Analog Devices. Det er fra kravspesifikasjonen ytret ønske om 10-bit ADC. Ved 10-bit vil det være mest effektivt å benytte Ethernet kontroller i 16-bit modus. Om 8-bit ADC benyttes anser jeg det som best egnet å benytte mikrokontrollerens ADC og med Ethernet kontroller i 8-bit modus. Da kravspesifikasjonen har satt ønske om høyere oppløsning, velger jeg Ethernet kontroller til 16-bit modus. Med grunnlag i at Ethernet kontroller vil være i 16-bit modus, anser jeg som det uhensiktsmessig å kun benytte 10 av de 16 tilgjengelige bit ved et sampel. Det velges derfor å benytte 16-bit ADC. Dette er langt høyere en de ønskede 10-bit, men å gå tilbake til 8-bit vil forplante seg videre i kretsen og sette nye krav til de kretser som jeg til nå har utviklet. Dette anses som uheldig. Valget om å benytte AD7610 har jeg gjort i hovedsak med argument i at de 16 datautgangene kan byteswappes. Med kontrollsignal fra mikrokontroller, kan rekkefølgen byttes mellom MSB og LSB. Årsak til dette ligger i at Ethernet kontrolleren ikke tidligere er benyttet i 16-bit modus. Tidligere prosjekt har koblet i 8-bit modus som anvist i boken ”Networking and Internetworking with Mikrokontrollers”. Jeg har ikke klart å oppdrive eksempler der Ethernet kontrolleren er valgt å operere i 16-bit modus. Det er derfor knyttet usikkerhet til oppkobling og programmering av Ethernet kontrolleren i denne modusen. Datablad til Ethernet kontroller er kortfattet, med begrenset informasjon knyttet til denne problematikken. Ved å benytte konverteren AD7610, vil det være mulig å operere i både 16- og 8-bit modus. Enkel modifikasjon av Ethernet kontroller må gjøres for endring av oppløsning. Det skal med dette være mulig i ettertid å gå over til 8-bit modus om ikke 16-bit modus viser seg egnet.
For en økt forbedring av programflyt, har jeg lagt mikrokontroller, ADC og Ethernet kontroller til felles databuss. Ved denne konfigurasjonen forventes et lavere resursbehov av mikrokontroller, noe som vil øke nøyaktigheten og samplingsraten. Ved å aktivere eller deaktivere ADC med signal fra mikrokontroller, kan dataflyt til Ethernet kontroller kontrolleres. For konfigurasjon av Ethernet kontrollerens registerverdier, benyttes bus fra mikrokontroller. Når kretsen er i sampling modus, klokkes den samplede data direkte inn i Ethernet kontroller fra ADC. Begrensende faktor ved direkte innklokking av data til Ethernet kontroller vil være tiden mikrokontroller bruker for å konfigurere Ethernet kontrolleren etter sendt UDP pakke. Dette må i ytterste fall gjøres mellom to sampels om flere UDP-pakker skal beskrive samme sampelvindu. Jeg har her forøvrig estimert en øvre samplingsrate på ca 22µs ved et slikt tilfelle, som tilsvarer en sampelfrekvens på ca 46kHz. Begrensningen ligger i oppbygning av UDP-header som må gjøres mellom et sampel om ett signal består av flere en én UDP-pakker. Det er knyttet håp om at det skal kunne være mulig å unngå å lage header for hver UDP-pakke ved å laste inn data etter ferdig laget header i Ethernet kontrollerens minne. Lar dette seg gjøre vil sampelfrekvens øke betydelig. Dette er svar som vil vise seg under arbeid med programmering av kretskortet og som det ved dette tidspunktet ikke kan gis svar på. Det er forøvrig satt som ønske fra kravspesifikasjonen om å sample med samplingsfrekvens på minimum 10kHz. Dette er sampel hvert 100µ som er tregere en utregnet forventet maksimale samplingsrate ved denne nye kretsløsningen. Det er ikke prioritert f.eks. dual line FIFO for økt sampelrate da samplingsraten allerede vil være sterkt forbedret.
RS232 og JTAG implementeres mot mikrokontrolleren. Dette er tilkoblinger som påvirker valg av busstilkobling fra mikrokontroller mot ADC og Ethernet kontroller og øker kompleksiteten til pcb-tegning.

4.2-7 PCB tegning

Ved inntegning i CADStar schematic sammenkobles kretsene for en total enhet. Kretsen overføres mot pcb, for realisering av kretskort. Jeg har på et tidlig tidspunkt, planlagt plassering av kretsmodulene. Det legges vekt på signalgang og signaltype, samt at GND og spenningstilførsel bør tilpasses disse. Kretskortet planlegges som fire lags kort, med signallag ved øvre og nedre lag, henholdsvis lag 1 og lag 4. Jeg benytter lag 2 som jordplan, og lag 3 som Vcc plan. Det er i dette designet operert med fire GND kategorier og fire spennings kategorier. GND er oppdelt til DigGND, AGND, RxGND og RAWGND. Samtlige GND plan sammenkobles i egnede tilkoblingspunkter. Det er derfor ikke flytende GND i kretsen. DigGND er tilknyttet den digitale delen av kretskortet. Tilsvarende er 5V tiltenkt digital spenningsforsyning, atskilt fra den øvrige kretsen. Sammenkoblingspunkt for felles jordplan, gjøres ved kretskortets kontakt av ekstern spenningskilde. DigGND er med dette ønsket i minst mulig grad å påvirke det analoge system. DigGND er implementert som flate under den digitale kretsen i kretskortets andre lag. Digital tilførselsspenning hentes fra utgangsfilteret til spenningsregulator. Den analoge jord legges under den analoge kretsen, henholdsvis forsterker, likeretter og krets for omhylningskurve. Analog GND tilkobles felles GND ved spenningsregulator. Ved transduserkretsens Rx-inngang er det opprettet RxGND. Dette forventes å være den del i kretsen med lavest støy. Inngangssignalet er her på sitt svakeste og mest sårbare nivå. Jeg velger derfor prinsipielt å tilslutte RxGND til felles jordreferanse via AGND. Koblingspunkt velges med maksimal avstand til kretsens støykilder. Også RxGND legges som heldekkende flate under Rx og TVG kretsen. Analog tilførselsspenning legges under tilhørende analog krets. Signallag plasseres som nevnt i øvre og nedre lag. Kretskortet er med dette et firelags kretskort. I Figur 59 vises GND lag, og i Figur 60 vises spenningslaget. Det er her inntegnet tilkoblingspunkter for jord og spenningslag type. Kretskortet som helhet med samtlige lag er vist i Vedlegg 8.

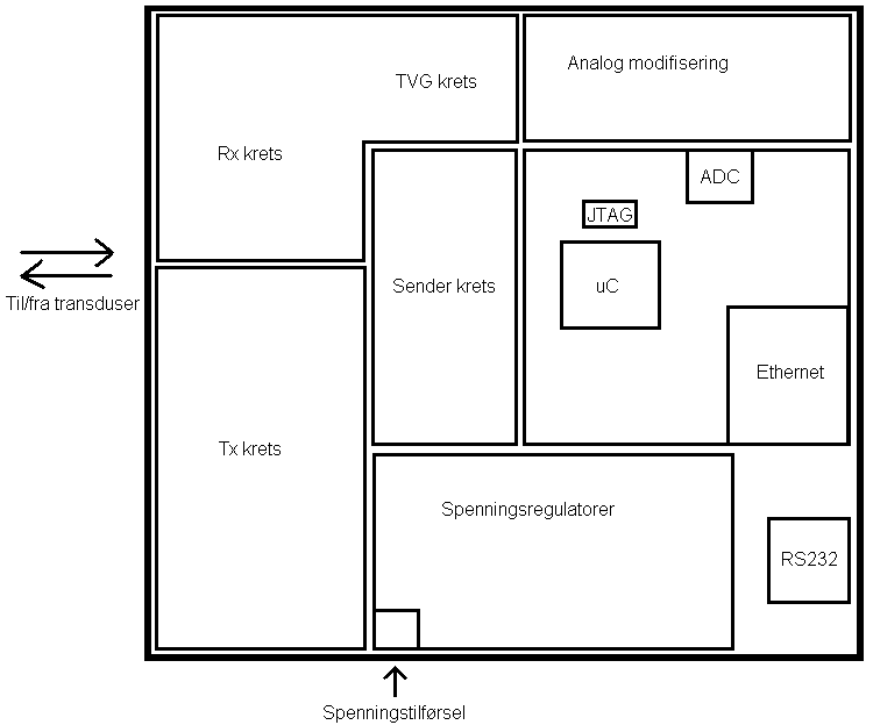


Figur 59 GND lag i kretskortløsning



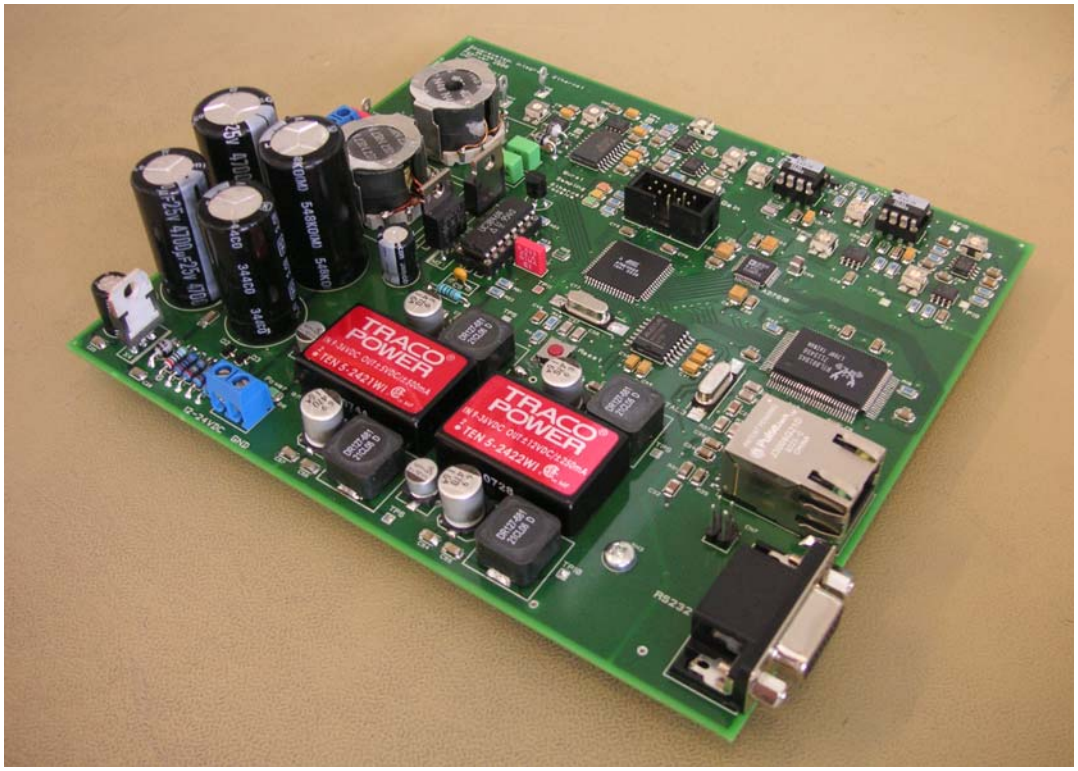
Figur 60 Spennings lag i kretskortløsning

Plassering av komponenter gjøres med første prioritet etter type signal og sårbarhet for støy. Her er spesielt viktig å tilslutte TVG krets så nære inngangssignalet som mulig. Orientering av komponenter som f.eks. mikrokontroller vil være relevant i prosessen med komponentplassering. I Figur 61 er det opptegnet skisse for min plassering av kretsens moduler. Ved sammenligning mot Figur 59 og Figur 60 kan det observeres likhetstrekk som forhåpentlig vis gir leser forståelse for mine valg av jord- og spennings lag.



Figur 61 Modulplassering ved kretskortdesign

Etter komponentplassering, defineres banebredder og øvrige relevante parametere. For korrekt justering, defineres verdier etter leverandøren hvor kretskortet skal bestilles fra. I dette tilfellet skal kortet bestilles fra Elprint. Eksempel på parameterverdier er vist i Vedlegg 10, hentet fra hjemmesiden til Elprint. Det benyttes 4101 standard til Elprint, vist i Vedlegg 11.
Parameterverdier for kretskortet velges med hensyn på produsentens begrensninger, produsentens pris og kretstypens signaltype.
Etter endt pcb tegning i CADStar, genereres gerber- og drill filer. Det genereres filer for hvert lag av kretskortet. Filene implementeres videre i software programmet CAM350 for inspeksjon og verifisering av det faktiske genererte designet. Det er disse filene som er grunnlaget for vedlagt transparentvedlegg i rapporten, Vedlegg 9. Avslutningsvis sendes designet til Elprint for produksjon. Dette gjøres via internett og deres innloggings tjeneste kalt Macaos. Kretskort lastes inn og spesifiseringer som for eksempel pris, antall, leveringstid m.m. føres. Avbildning av Macaos er vist i Vedlegg 11.



Figur 62 Ferdig produsert kretskort med påloddede komponenter

Slutt kapittel 4

5.0 Programmering

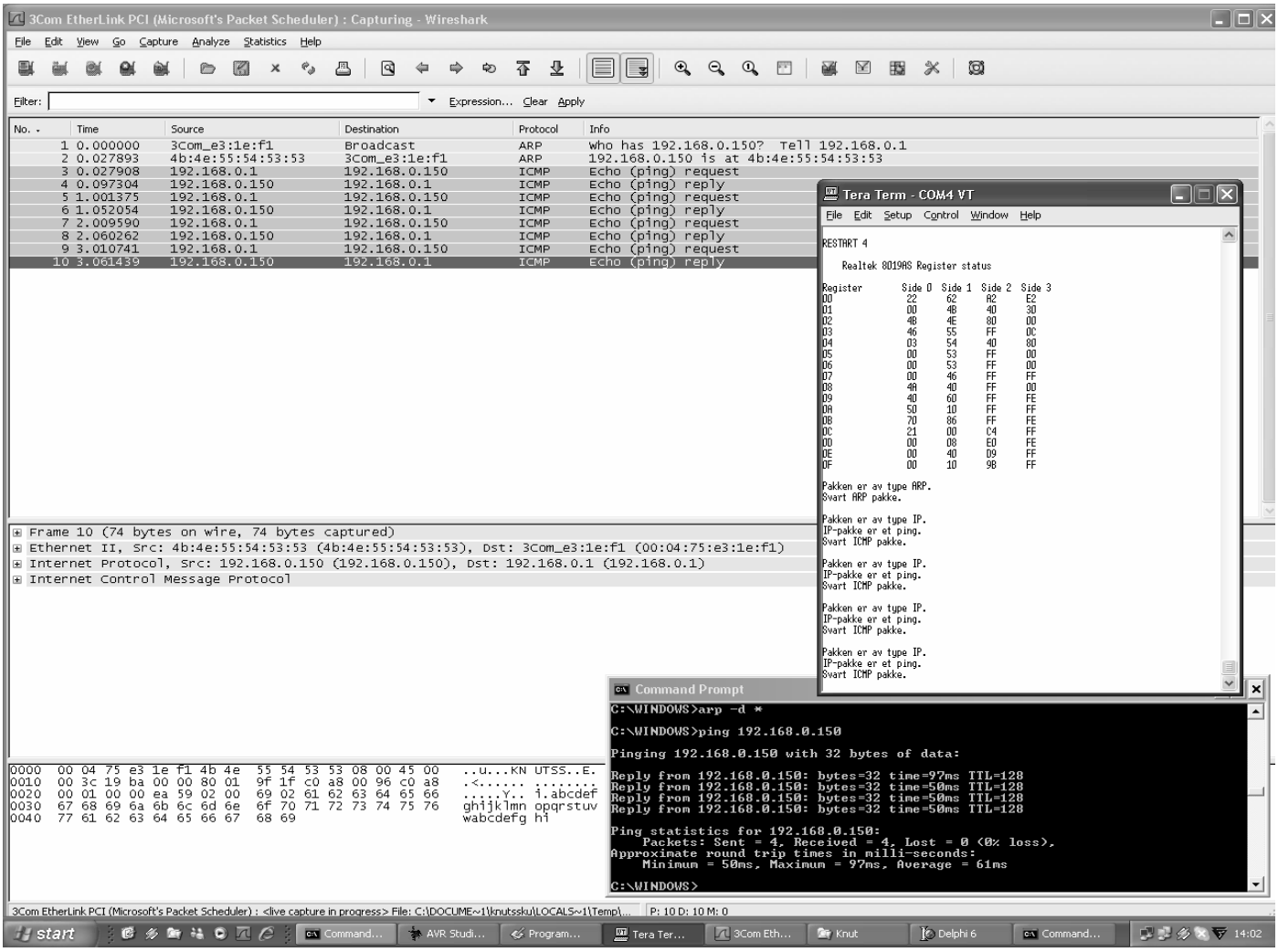
Prosjektets tredje fase omhandler programmering av mikrokontroller og programmering av brukerprogram i Delphi. Ved programmering settes produkt fremstilt fra mitt arbeid tidligere i prosjektet i drift. Dette er derfor en spennende fase med verifikasjon av funksjonalitet. Noe arbeid er gjort i perioden jeg ventet kretskortet ferdig etset, men det er krevende å lage kode uten å fortløpende kunne teste denne.
Som hjelpemiddel i arbeidet har jeg benyttet gratis software fra Atmel ved programmering av mikrokontroller. Programmering og feilsøking er gjort via JTAG.

5.1 C programmering

Programmering av mikrokontroller ble igangsatt så raskt som mulig etter komponentplassering på kretskortet. Som første del av programmeringsfasen laget jeg struktur og flytplan. Jeg forventet en relativt stor C-kode og jeg planla derfor en modulbasert programmering for på denne måten å lettere kunne holde oversikt over det totale programmet. Modulene ble delt opp i følgende moduler:

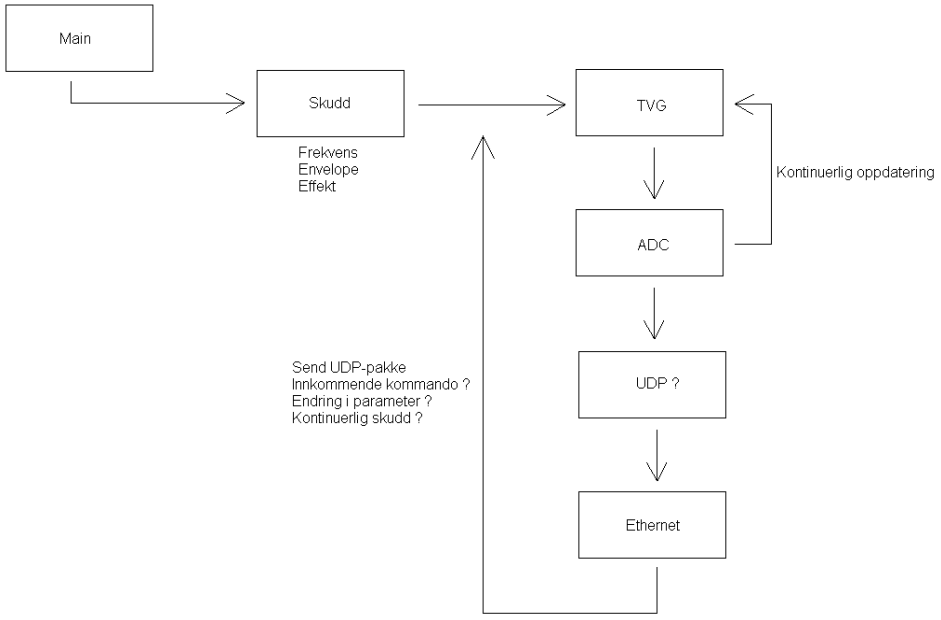
- ✓ Main
- ✓ Initiering
- ✓ Ethernet (RTL8019AS)
- ✓ ADC (AD7610)
- ✓ TVG (LM1972)
- ✓ Skudd
- ✓ RS232

For utvikling av systemet utviklet jeg først signalflyt over RS232 som jeg knyttet til programmet Tera Term. Det ble via dette systemet verifisert verdi- og funksjonskontroll under utvikling av software. Bl.a. kunne tabell over Ethernet kontrollerens parameterverdier skrives ut for verifisering. Jeg benyttet også Wire Shark, et snifferprogram, for oversikt over signalflyt gjennom PC`ens nettverkskort. Under oppbygning av ARP- og IP pakker viste verktøyet seg verdifullt. I Figur 63 vises avbildning av PC-skjerm under arbeid med utvikling av kommunikasjon over Ethernet. Som kan observeres er det der sendt ”ping” til kretskort, hvor kretskort har kunnet svare på korrekt måte. Til høyre vises Tera Term med utskrift via RS232 med oversikt over Ethernet kontrollerens register i topp.



Figur 63 Avbildning av PC-skjerm under utvikling av internett funksjonalitet for kretskortet Skjermbildet viser WireShark, Teraterm, bekreftelse på sendt og mottatt "pig"

Jeg legger opp til å kjøre metodekall ved innkommende pakke via internett og i tilfeller signalet skal samples eller justeres ved TVG. Skjematisk opptegnet programflyt er vist i Figur 64.



Figur 64 Programflyt for mikrokontroller

Jeg koblet interrupt pin fra Ethernet kontroller til µC for signal om innkommende pakke er mottatt fra internett. Jeg benytter også interrupt rutine ved trigging av sampel eller TVG justering. For visuell bekreftelse av funksjonalitet er det ved design av kretskort, inntegnet tre lysdioder. Disse indikerer innkommende pakke til kretskort fra internett, burst signal til transduser og sampling av innkommende signal.
Ved igangsetting av kretskortet startes sekvens for initiering av både interne og eksterne funksjoner. Dette er for eksempel SPI, JTAG osv. samt initiering av RTL8019AS Ethernet kontrolleren og ADC. Etter oppstartsekvens vil kretskortet fungere etter kommandoer fra bruker, med UDP pakker sendt over nettet.
Som valg kan bruker velge å kjøre ett enkelt skudd med valgt frekvens og envelope, for så å motta reflektert ekkosignal som opptegnet graf i Delphi. Alternativt kan bruker velge å kjøre i kontinuerlig drift med nytt skudd like etter mottatt ekko fra foregående skudd. Da systemet er designet til å dekke område på ca 100m, vil nytt skudd skytes etter ca 140ms. Ved dette systemet har jeg testet å opptegne graf av ekko via tre UDP-pakker for en høyere oppløsning og på denne måten praktisere systemets økte ytelse. Hver av de tilhørende pakker sendes fortløpende over Ethernet og sorteres ut av Delphiprogrammet ut ifra data innpakket i UDP-pakken. Av valgmuligheter kan bruker endre sendereffekt, forsterkning av signalet før ekkosignal påtrykkes analog krets i kretskort, frekvens i burst, antall perioder i ett burst (envelope) samt valg av port for både bruker og for kretskort. Flere parametere kan velges implementert i programmet, men er valgt bort av tidsmessige årsaker. Programkode for Delphi er vedlagt i Vedlegg 15.

5.2 Delphi

Som brukergrensesnitt har jeg laget program i Delphi. Programmet skal gi bruker kontroll over funksjoner i kretskortet samt å presentere de data som sendes fra kretskortet. Data sendt fra kretskortet er enten status og verdi i kretskort parametere, eller samplet data. Jeg har valgt å lage et så enkelt og oversiktlig program som mulig for en enkel og oversiktlig brukervennlighet.
Ved endring av parameterverdi, for eksempel sendereffekt, vil ikke Delphi programmet godkjenne valget før bekreftelse fra kretskort er mottatt. På denne måten vil bruker kunne forsikre seg om at kretskort er satt etter valgte premisser. Ubekreftet endring varsles med rødt lys og tekst. Er kretskort bekreftet satt endres farge til grønn og bekreftende tekst vises. For endring av senderfrekvens, envelope eller port, må knapp merket ”Options” velges. Ved å trykke ”Lagre og send” lukkes ”Options” vinduet og UDP-pakke med ny konfigurasjonsvalg sendes automatisk til kretskortet. Verifiserende bekreftelse i Delphi settes ikke før UDP-pakke fra kretskort er mottatt, da med identiske verdier i samtlige parametere. Om pakke ikke mottas fra kretskort kan bruker trykke ”Send” knapp for på ny å sende de valgte parametere og på denne måten motta nytt forsøk på bekreftende UDP-pakke fra kretskortet.
Om kretskortet er valgt å kjøre i kontinuerlig modus med fortløpende nye skudd, er det ikke mulig å velge enkelt skudd. For å velge enkelt skudd må kontinuerlig drift skrus av. Bruker får automatisk beskjed om ugyldig valg gjøres. Avbildning av programmet med eksempel på feilmelding er vist i Vedlegg 16. For fremvisning av samplet data benyttes delprogram inkludert i Delphi kalt TeeChart Standard 4.04. Her presenteres samplet data grafisk, hvor jeg har valgt å vise avstand i x-akse og styrke på signalet i y-aksen. Maksimale styrke på innkommende ekkosignal samplet av ADC er på 10V. Ved å øke signalbredden i tillegg til den økte oppløsningen i ADC, vil systemet gi betydelig forbedring i detaljoppløsningen.
Ved å benytte UDP-pakker har jeg valgt å bruke to typer pakker i systemet. En type inneholder samplet data med størrelse avhengig av ønsket oppløsning samt antall UDP-pakker som er ønsket i et sampelvindu. Den andre type pakke er en ren kommandopakke og er så liten som protokollen tillater, henholdsvis 32Byte. Som standard har jeg ikke benyttet kontrollsum i UDP-pakken, da samplet data klokkes rett inn i Ethernet kontrolleren og bygger pakken opp fortløpende. Da kontrollsummen skal ligge i pakkens header, lar den seg ikke utregne da sampeldata på det tidspunktet header bygges opp ikke er kjent. Forøvrig benyttes kontrollsum av Ethernetpakken, kontrollsum kalkulert med hensyn på totale header og pakkens lengde. Når det gjelder brukervariabler, er de første åtte databyte benyttet til kontroll eller kommando for parameterverdier. Hver UDP-pakke, enten det er sampel- eller kommandopakke, inneholder denne kommandosekvensen. Avhengig av om pakken er til kretskort eller PC, eller om pakken sendes eller mottas, kodes disse for avlesning. Også pakker med samplet data inneholder status for kretskortparametere. Dette har jeg lagt inn for å øke tilliten til kommunikasjonen. Oversikt over UDP-kommandoer er vist i Vedlegg 17.

Slutt kapittel 5

6.0 Praktisk erfaring

Etter gjennomgang av prosjektet har jeg opparbeidet erfaring knyttet til analog kretsproblematikk, erfart valg gjort ved kretsrevurdering og endring, erfart kretskortets evne til dialog via Ethernet, samt erfaring knyttet til kontroll og innhenting av data fra kretskortet via brukerprogram laget i Delphi.
Når det gjelder analog krets, anser jeg det som betydelig å bruke ny kretsløsning og nye komponenter i forhold til hva som er benyttet i den eldre kretsløsning. Etter gjennomgang virker kretsløsningen jeg kom frem til, å være raskere og mer nøyaktig. Ønske fra kravspesifikasjonen om operasjonsfrekvens på 120kHz er godt innenfor hva kretsløsningen kan håndtere. Kretsløsningen er implementert i det nye kretskortet. For økt nøyaktighet kan det være relevant å finne ny egnet krets for omhylningskurve av det likerettede ekkosignal. Som et sidesprang har jeg forøvrig testet kretsløsning også ved denne delkretsen, men hvor mangel på tid har hindret fullføring av arbeidet. Resultater fra utførte test av påbegynt krets for omhylningskurve gir inntrykk av å gi forbedring med raskere responstid.
Det var stilt spørsmål om nye komponenter ville gi tilstrekkelig forbedring av den eldre kretsløsning. Som generell erfaring fra prosjektets del 1 med analog kretsdrøftning, viser nye komponenter å ha mindre innvirkning på den eldre kretsløsningen. For forbedring og økt ytelse bør ny kretsløsning benyttes. Forøvrig er det erfart en forbedret ytelse i ny kretsløsning med bruk av dioden BAS16 og operasjonsforsterker AD845. Jeg har valgt å benytte disse komponentene i ny krets i det nye kretskortet.
Under revurdering av den totale analoge kretsløsning fant jeg det interessant å se nærmere på en komponent jeg oppdaget, henholdsvis komponenten LM1972. Jeg ønsket å se på mulighet for å benytte komponenten i TVG-justering av det innkommende ekkosignal. Dette for å øke kontrollen av systemet samt å minimalisere det totale antall komponenter. Etter test virker enheten å fungere etter ønsket funksjonalitet og ble derfor implementert i kretsløsningen min. Det er ikke gjort reell test i sjø grunnet problemer med spenningsregulator på kretskortet, men etter programmering viser enheten seg å fungere som ønsket. Det er implementert enhet med to kanaler som gir rom for videre optimalisering via programmkode. Ved å smelte sammen kanalene kan bruker justere konstantforsterkningen til innkommende signal. Med dette har jeg kunnet oppfylle ønsket om å gjøre systemet mest mulig fleksibelt.
Jeg tok sjansen på å realisere krets med Ethernet kontroller konfigurert til drift i 16-bit modus. Jeg var usikker på om dette lot seg realisere da dette ikke tidligere er gjort. Mye av usikkerheten var knyttet til den manglende dokumentasjonen om komponenten. Etter programmering av enheten viser dette seg som vellykket. Kretskort fungerer som ønsket. Arbeidet jeg gjorde med tilrettelegging av funksjonalitet for også å kunne bruke Ethernet kontrolleren i 8-bit modus har derfor vært en betryggende reserveplan som ikke har vært nødvendig å gjøre bruk av. Erfaring med Ethernet kontrolleren viser til en god stabilitet, også ved drift over tid.
For kontroll av enheten kan bruker benytte program jeg har laget i Delphi. Bruker kan der avlese ekkosignal via grafisk fremstilling, eller velge parameterverdier for kretskortet. Jeg har med argument i å ”presse” systemet noe, valgt å dele ett sampelvindu opp i 3 UDP-pakker. Jeg ønsker da å teste grense for sampelhastighet og oppløsning. Ved ca 45kHz sampelhastighet opptegnes graf som ønsket. Under drift viser programmet å fungere godt med stabil oppførsel, da også over tid.

Kretskoret som fremstår etter endt prosjekt oppfyller de krav som kravspesifikasjonen satt. Kretskortet lar seg benytte ved 120kHz burstfrekvens ved enkel modifisering av resonanskretsen. I tillegg til dette er systemet forbedret med økt fleksibilitet som gjør det mulig å forbedre videre ved programmering. Som ny funksjonalitet kontrolleres kretskortet over Ethernet. Som eksempel på realisert parameterkontroll kan nevnes brukerkontrollert burstfrekvens, brukerkontrollert envelope og brukerkontrollert transdusereffekt. Systemet lar seg også programmere til å gi bruker kontroll over sampelhastigheten samt mulighet for brukerkontrollert inngangsforsterkning for ekkosignal.
Avslutningsvis kan nevnes at kretskortet som er utviklet er det første i serien. Dette betyr at kretskortet innehar komponenter som kan fjernes ved neste versjon. Eksempler er krets for RS232 tilknytning, manuell offsetjustering av operasjonsforsterkere samt at filterkretsen trolig kan fjernes. Dette vil gjøre kretskortet mindre i fysisk størrelse.

7.0 Konklusjon

Prosjektet ble innledet med problematikk knyttet til om den analoge krets kunne forbedres, da enten ved bruk av nye komponenter eller eventuelt ved å implementere ny kretsløsning. Etter gjennomføring erfarte jeg at kombinasjonen å både benytte nye komponenter i tillegg til å benytte ny kretsløsning ga en tydelig forbedring. Særlig gjelder dette krets for likeretting av inngangssignal. Likeretterkrets har i tillegg til økt frekvensytelse, en økt evne til å nyttegjøre inngangssignalet til det fulle uten å blokkere deler av signalet. Dette resulterer i en bedre nøyaktighet samt et lavere krav til etterfølgende krets for omhylningskurve.
Jeg implementerte digital krets for kommunikasjon over Ethernet. Jeg benyttet der Ethernet kontroller RTL8019AS, konfigurert til 16-bit oppløsning. Dette viser seg vellykket, tross de usikkerheter som var knyttet til denne konfigurasjonen. Jeg har ikke måttet bruke muligheten jeg implementerte der jeg kunne koble om til 8-bit modus og programmere om ADC i "byteswape"-modus. Jeg ser det som forbedring av systemet å bruke Ethernet kontrolleren i 16-bit modus.
ADC virker å fungere optimalt etter det ønskede behov. 16-bit oppløsning er en kraftig forbedring fra 8-bit som tidligere er benyttet. Kontroll knyttet til tilkobling av enheten mot μC viser seg også som god.
Ideen jeg hadde med sammenkobling mellom enhetene via felles bus er etter erfaring en effektiv måte å knytte de aktive elementene i kretsen sammen på. Resultat i økt hastighet og avlasting av mikrokontroller gir et mer stabilt og presist system. Jeg anser sammenkoblingen av μC, ADC og Ethernet ved felles bus som en viktig erfaring og er et håndgrep som forbedrer systemet betydelig.
Kretskortet som jeg utviklet består både av analog og digital krets. Eldre krets var ikke i stand til annet en generering av burst samt å TVG-justere det innkommende signal. Med dagens teknologi og de løsninger som er laget i dette prosjektet, fremstår enheten som mer selvstendig og komplett. Det nye kretskortet er mer selvstendig i den forstand at det kontrollerer dialog mot Ethernet selv samt at parametere kan kontrolleres fra bruker over UDP-pakker. Eldre krets satt krav til manuelt å velge sendereffekt for transduser om denne var ønsket endret. Det nye kretskortet muliggjør endring av flere parametere enn hva det eldre kretskortet ga mulighet til. Tross den drastisk økte kompleksitet observeres den fysiske størrelsen i det nyutviklede kretskortet å være mindre enn det gamle og mindre fullstendige kretskortet. Dette bekrefter første versjon av det nye kretskortet som en mer kompakt enhet. Jeg anser derfor at ønsket om forbedring av det eldre kretskortet som innfridd.
Ønske fra kravspesifikasjonen om brukerfrekvens i burstsignal fra transduser er også innfridd. Realisert kretskort er spesifisert for 70kHz, men hvor enkel modifisering tillater 50kHz og 120kHz.
Enheten er testet i tilknytning til nettet ved UiO i 25 timer i strek uten stans. Systemet fungerte feilfritt i denne perioden. Enheten er også testet med burst satt i kontinuerlig drift i 25 timer. Dette innebærer syklus med generering av burstsignal, sampling og TVG-justering, hvor data sendes fortløpende til bruker. Også ved denne uttestingen viser enheten å fungere feilfritt. Dette bekrefter en god stabilitet, også hva gjelder Ethernet kontrolleren RTL8019AS. Jeg anser derfor Ethernet kontrolleren som egnet for bruk i sonarsystemet.

8.0 Kildereferanser

1. The Circuits and Filters Handbook, Wai-Kai Chen
   ISBN 0-8493-8341-2
2. The Art of Electronics, Paul Horowitz, Winfield Hill
   ISBN 0-521-37095-7
3. Networking and Internetworking with Microcontrollers, Fred Eady
   ISBN 0-7506-7698-1
4. Embedded Ethernet and Internet complete, Jan Axelson
   ISBN-13 978-1-931448-00-0
   ISBN-10 1-931448-00-0
5. Operational Amplifiers and Linear Integrated Circuits: Theory and Application, James M. Fiore
   ISBN 0-314-90893-5
6. Applications of Operational Amplifiers: Third-Generation Techniques, Burr-Brown, Jerald G. Graeme
   ISBN 0-07-023890-1
7. Dynamiske Systemer; Modellering, analyse og simulering, Finn Haugen
   ISBN 82-519-1877-4
8. Analoge Kretser og Komponenter, Rolf Ingebrigtsen
   ISBN 82-7634-317-1
9. Principles and Applications of Electrical Engineering, Giorgio Rizzoni
   ISBN 0-07-121772-X
10. Prefag 2E, Specification for Double-Sided Plated-Trough Boards
    ISBN 87-571-1967-8
11. Introducing Dephi Programming, John Barrow, Linda Miller, Katherine Malan, Helene Gelderblom
    ISBN 0-19-578911-3
12. Building Delphi6 Applications, Paul Kimmel
    ISBN 0-07-212995-6
13. Mastering Delphi4, Marco Cantù
    ISBN 0-7821-2350-3
14. Op Amp Applications, Walter G.Jung
    ISBN 0-916550-26-5
15. Feedback Amplifiers, Theory and Designs. Gaetano Paulumbo and Salvatore Pennisi
    ISBN 0-7923-7643-9
16. Microstrip Filters for RF/Microwave Applications, Jia-Sheng Hong and M. J. Lancaster
    ISBN 0-471-22161-9
17. RTP: Audio and Video for the Internet, Colin Perkins
    ISBN 0-672-32249-8
18. Lineær Kretselektronikk, Helge Balk
19. Wikipedia, nettside
    www.wikipedia.com

9.0 Vedlegg

Vedlegg 1 Utregning av estimert signalstyrke

Avstand [m]		Vin [mV] (- 30dB)		Vin [mV] (- 60dB)		G=100 (- 30dB)	G=100 (- 60dB)
3		176,0992		5,5687		17,6099	0,5569
4		99,0558		3,1324		9,9056	0,3132
5		63,3957		2,0047		6,3396	0,2005
7,50		28,1759		0,8910		2,8176	0,0891
10		15,8489		0,5012		1,5849	0,0501
15		7,0440		0,2227		0,7044	0,0223
20		3,9622		0,1253		0,3962	0,0125
25		2,5358		0,0802		0,2536	0,0080
30		1,7610		0,0557		0,1761	0,0056
35		1,2938		0,0409		0,1294	0,0041
40		0,9906		0,0313		0,0991	0,0031
45		0,7827		0,0247		0,0783	0,0025
50		0,6340		0,0200		0,0634	0,0020
55		0,5239		0,0166		0,0524	0,0017
60		0,4402		0,0139		0,0440	0,0014
65		0,3751		0,0119		0,0375	0,0012
70		0,3234		0,0102		0,0323	0,0010
75		0,2818		0,0089		0,0282	0,0009
80		0,2476		0,0078		0,0248	0,0008
85		0,2194		0,0069		0,0219	0,0007
90		0,1957		0,0062		0,0196	0,0006
95		0,1756		0,0056		0,0176	0,0006
100		0,1585		0,0050		0,0159	0,0005

Vedlegg 2 Drøfting av hydroakustikk og signalverdier

Aktuelle TargetStrength vil være innenfor området -30dB til -60dB
SourceLevel er oppgitt til 219dB // uPa ved 100W
Sonar respons Srt er oppgitt til -185dB // 1V/uPa
Resonansfrekvensen er oppgitt til 50kHz

$$EL = SL - 2TL + TS$$

$$VRTdB = SL + SRT - 2TL + TS$$

$$Vin[dB \Rightarrow V] = \left[V_0 * 10^{\frac{L\_in\_dB}{20}} \right] * \left[V_0 * 10^{\frac{Srt\_in\_dB}{20}} \right]$$

SL =	219	dB
Srt =	-185	dB
Alpha =	0	dB/m
B =	2 / T	(T = 0,5 s)
c =	1470	m/s
R =	t / c	
t =	c * R	

Avstand	10 log (R)	[dBm]
Spenning	20 log (V)	[dBv]

Vin uten tap (TL)

TL =	20 log (R)+ Alpha*R	[dB]	-26
EL =	SL - 2*TL + TS	[dB]	
Vrt =	SL - 2*TL + TS + Srt	[dB]	
Vin =	EL+ Srt	[dB]	

Objekt signalstyrke

TS = -60 dB

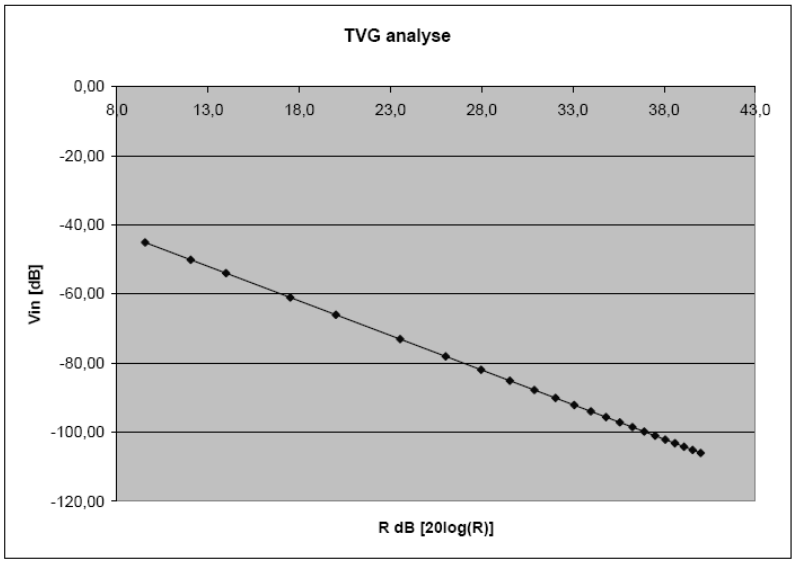
R avstand	TL [dB]	Vrt [dB]	EL [dB]
3	9,54	-45,08	139,92
4	12,04	-50,08	134,92
5	13,98	-53,96	131,04
7,50	17,50	-61,00	124,00
10	20,00	-66,00	119,00
15	23,52	-73,04	111,96
20	26,02	-78,04	106,96
25	27,96	-81,92	103,08
30	29,54	-85,08	99,92
35	30,88	-87,76	97,24
40	32,04	-90,08	94,92
45	33,06	-92,13	92,87
50	33,98	-93,96	91,04
55	34,81	-95,61	89,39
60	35,56	-97,13	87,87
65	36,26	-98,52	86,48
70	36,90	-99,80	85,20
75	37,50	-101,00	84,00
80	38,06	-102,12	82,88
85	38,59	-103,18	81,82
90	39,08	-104,17	80,83
95	39,55	-105,11	79,89
100	40,00	-106,00	79,00

Estimert signalstyrke

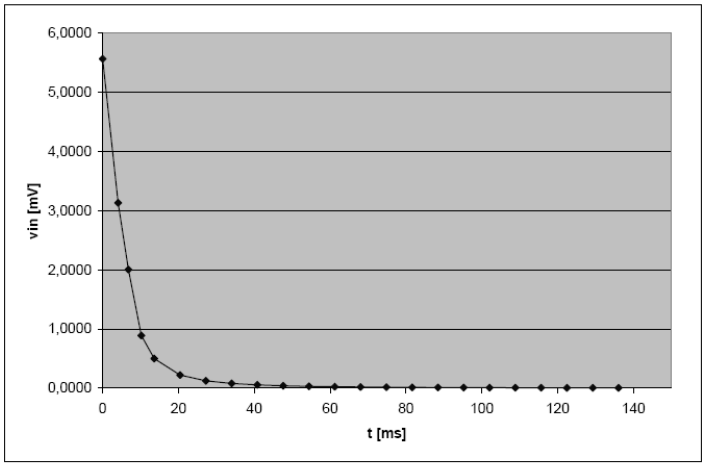
Vin [dB]	vin [mV]
-45,08	5,5687
-50,08	3,1324
-53,96	2,0047
-61,00	0,8910
-66,00	0,5012
-73,04	0,2227
-78,04	0,1253
-81,92	0,0802
-85,08	0,0557
-87,76	0,0409
-90,08	0,0313
-92,13	0,0247
-93,96	0,0200
-95,61	0,0166
-97,13	0,0139
-98,52	0,0119
-99,80	0,0102
-101,00	0,0089
-102,12	0,0078
-103,18	0,0069
-104,17	0,0062
-105,11	0,0056
-106,00	0,0050

Utregning av forsterkning av signal

vin [V]	G = 100	G = 101	G = 101*625	G=22,72
0,00556875	0,55687	0,56244	351,527157	0,126522
0,00313242	0,31324	0,31637	197,734026	0,071169
0,00200475	0,20047	0,20248	126,549776	0,045548
0,000891	0,08910	0,08999	56,2443451	0,020244
0,00050119	0,05012	0,05062	31,6374441	0,011387
0,00022275	0,02227	0,02250	14,0610863	0,005061
0,0001253	0,01253	0,01265	7,90936103	0,002847
8,019E-05	0,00802	0,00810	5,06199106	0,001822
5,5687E-05	0,00557	0,00562	3,51527157	0,001265
4,0913E-05	0,00409	0,00413	2,5826485	0,000930
3,1324E-05	0,00313	0,00316	1,97734026	0,000712
2,475E-05	0,00247	0,00250	1,56234292	0,000562
2,0047E-05	0,00200	0,00202	1,26549776	0,000455
1,6568E-05	0,00166	0,00167	1,04586592	0,000376
1,3922E-05	0,00139	0,00141	0,87881789	0,000316
1,1862E-05	0,00119	0,00120	0,74881525	0,000270
1,0228E-05	0,00102	0,00103	0,64566212	0,000232
8,91E-06	0,00089	0,00090	0,56244345	0,000202
7,8311E-06	0,00078	0,00079	0,49433506	0,000178
6,9368E-06	0,00069	0,00070	0,4378885	0,000158
6,1875E-06	0,00062	0,00062	0,39058573	0,000141
5,5533E-06	0,00056	0,00056	0,3505534	0,000126
5,0119E-06	0,00050	0,00051	0,31637444	0,000114



R avstand [m]	t [s]		t [s]	t [ms]
	0		*tur-retur*	0
3	0,002041		0,00408	4,08
5	0,003401		0,00680	6,80
7,5	0,005102		0,01020	10,20
10	0,006803		0,01361	13,61
15	0,010204		0,02041	20,41
20	0,013605		0,02721	27,21
25	0,017007		0,03401	34,01
30	0,020408		0,04082	40,82
35	0,02381		0,04762	47,62
40	0,027211		0,05442	54,42
45	0,030612		0,06122	61,22
50	0,034014		0,06803	68,03
55	0,037415		0,07483	74,83
60	0,040816		0,08163	81,63
65	0,044218		0,08844	88,44
70	0,047619		0,09524	95,24
75	0,05102		0,10204	102,04
80	0,054422		0,10884	108,84
85	0,057823		0,11565	115,65
90	0,061224		0,12245	122,45
95	0,064626		0,12925	129,25
100	0,068027		0,13605	136,05



R avstand [m]	R til 10log (R)		R til 2*10log (R)
3	4,771		9,5424
4	6,021		12,0412
5	6,990		13,9794
7,5	8,751		17,5012
10	10,000		20,0000
15	11,761		23,5218
20	13,010		26,0206
25	13,979		27,9588
30	14,771		29,5424
35	15,441		30,8814
40	16,021		32,0412
45	16,532		33,0643
50	16,990		33,9794
55	17,404		34,8073
60	17,782		35,5630
65	18,129		36,2583
70	18,451		36,9020
75	18,751		37,5012
80	19,031		38,0618
85	19,294		38,5884
90	19,542		39,0849
95	19,777		39,5545
100	20,000		40,0000

Drøfting av egenskaper for kretsløsning

Det vurderes å justere innkommende signal til spennet 0V til 10V.

Med å sette krav til å kunne observere fisk fra 4m, vil kraftigste signal (-30dB) ved G = 1000 være 100V.
Med å sette krav til å kunne observere -30dB fisk på ca 12,5m, vil kraftigste signal ved G = 1000 være 10V.
Med å sette krav til å kunne observere fisk på 100m, vil laveste signal (-60dB) med G = 1000 gi spenning 5mV.

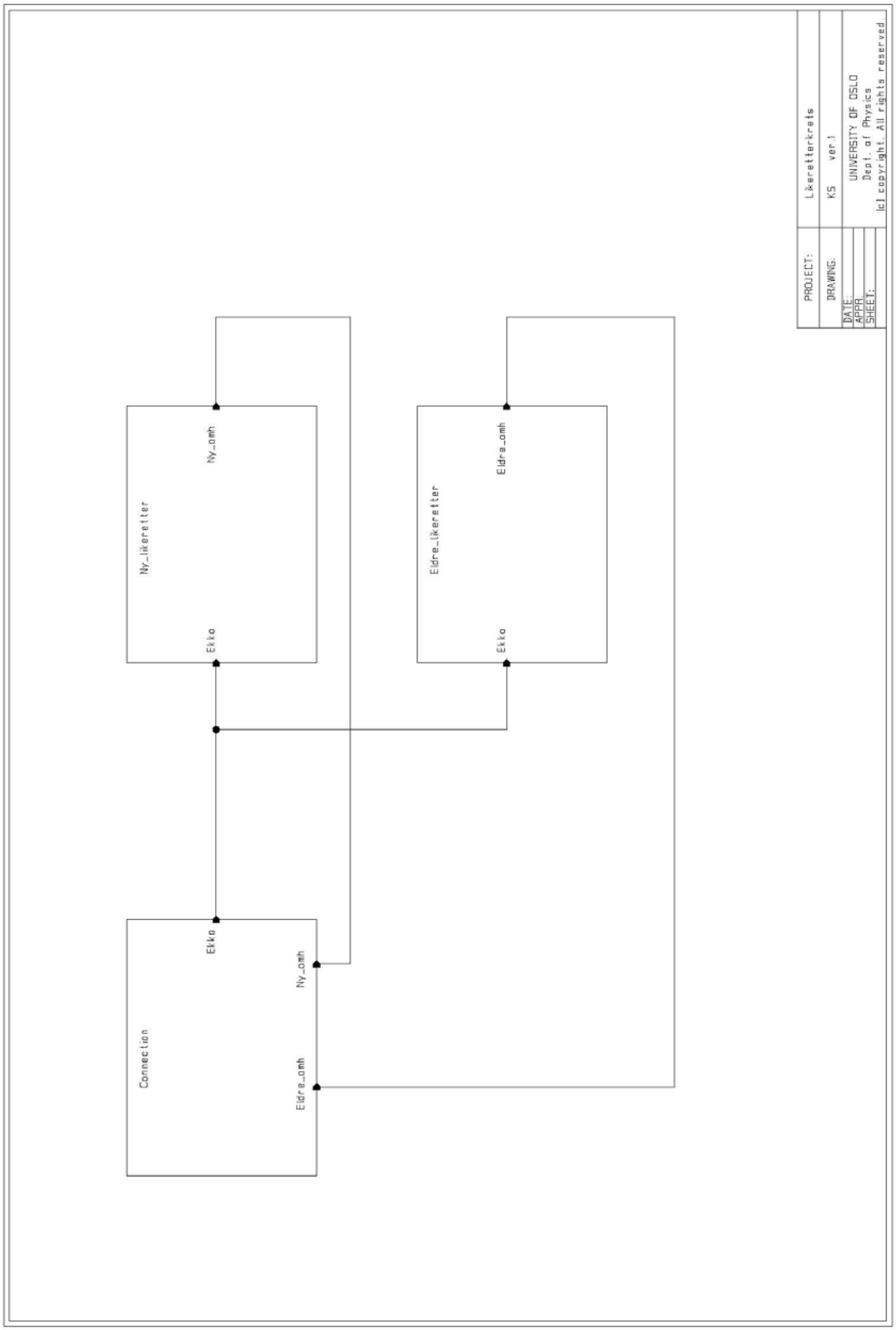
Altså,- -60dB signal i avstand 4m til 100m bør av dette være detekterbart.
-30dB signal i avstand ca 12,5m til 100m bør av dette være detekterbart.
Dette etter de forenklede verdier oppgitt om transduser.

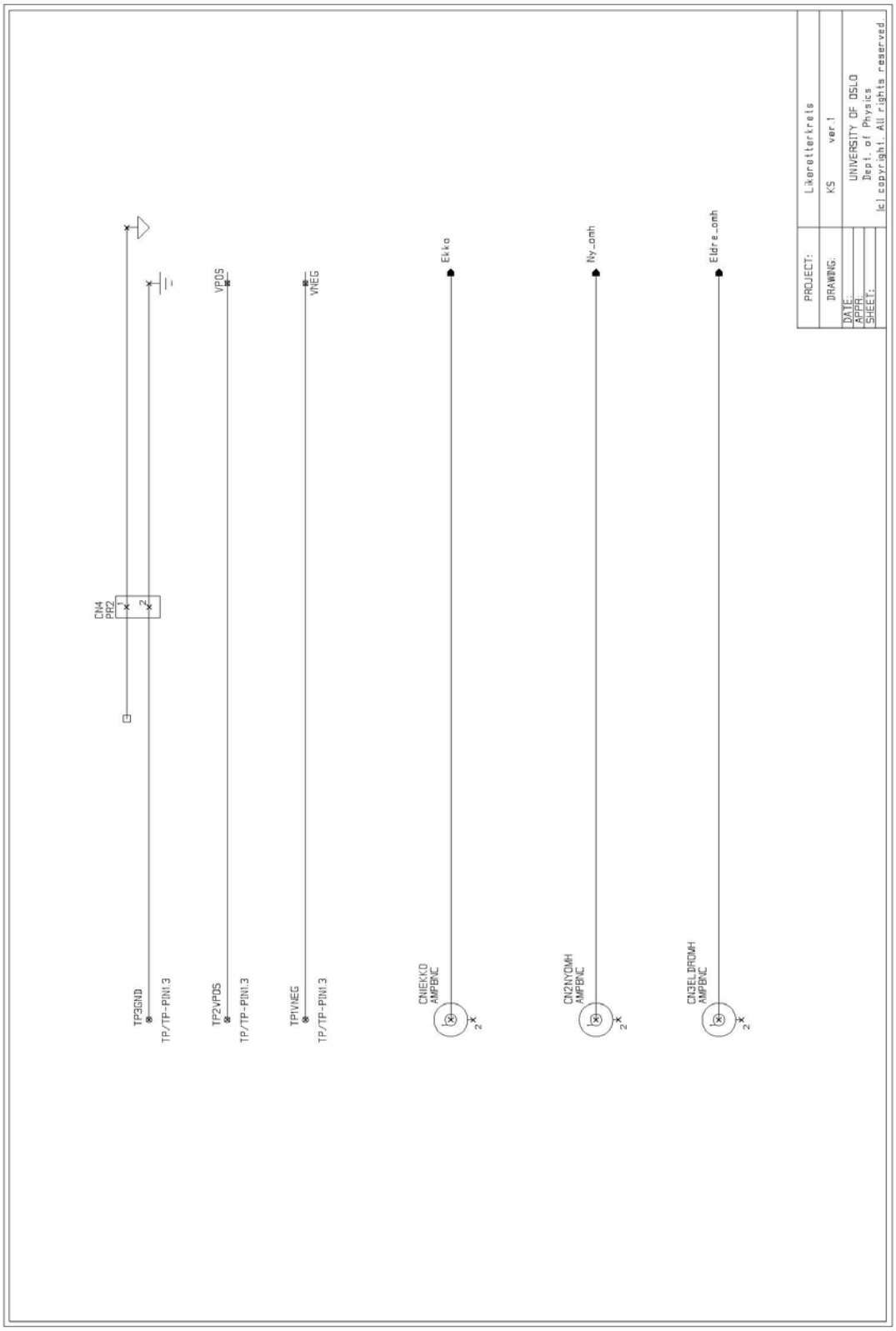
Om prioritering settes ved å kunne observere fisk fra 4m, innenfor signalstyrke, bør G = 100 benyttes.
Med å sette krav til å kunne observere -30dB fisk på 4m, vil kraftigste signal ved G = 100 være 9,9V.
Med å sette krav til å kunne observere fisk på 35m, vil laveste signal (-60dB) med G = 100 gi spenning cv 4mV.

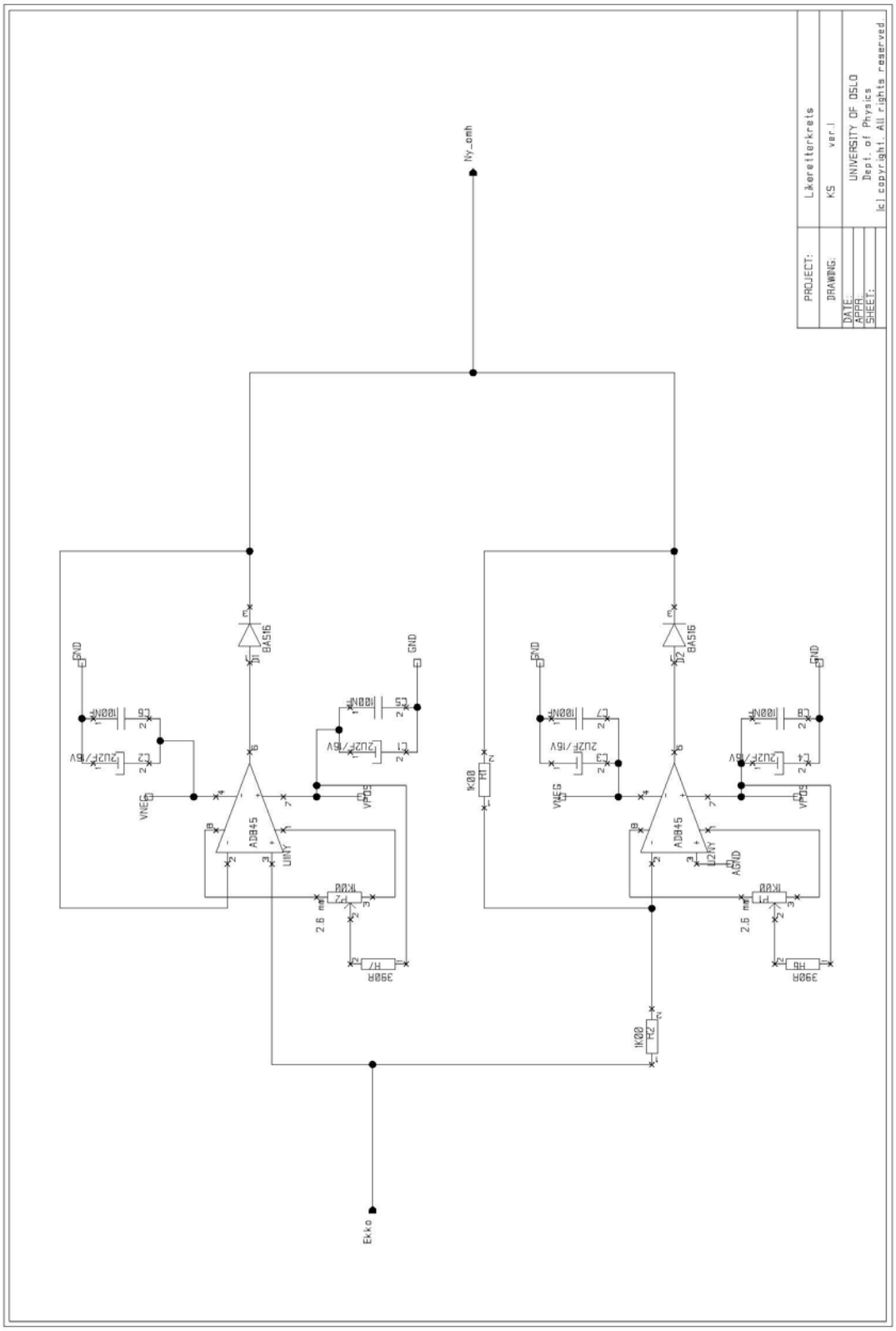
Altså,- -60dB signal i avstand 4m til 35m bør av dette være detekterbart.
-30dB signal i avstand ca 4m til 100m bør av dette være detekterbart.
Dette etter de forenklede verdier oppgitt om transduser.

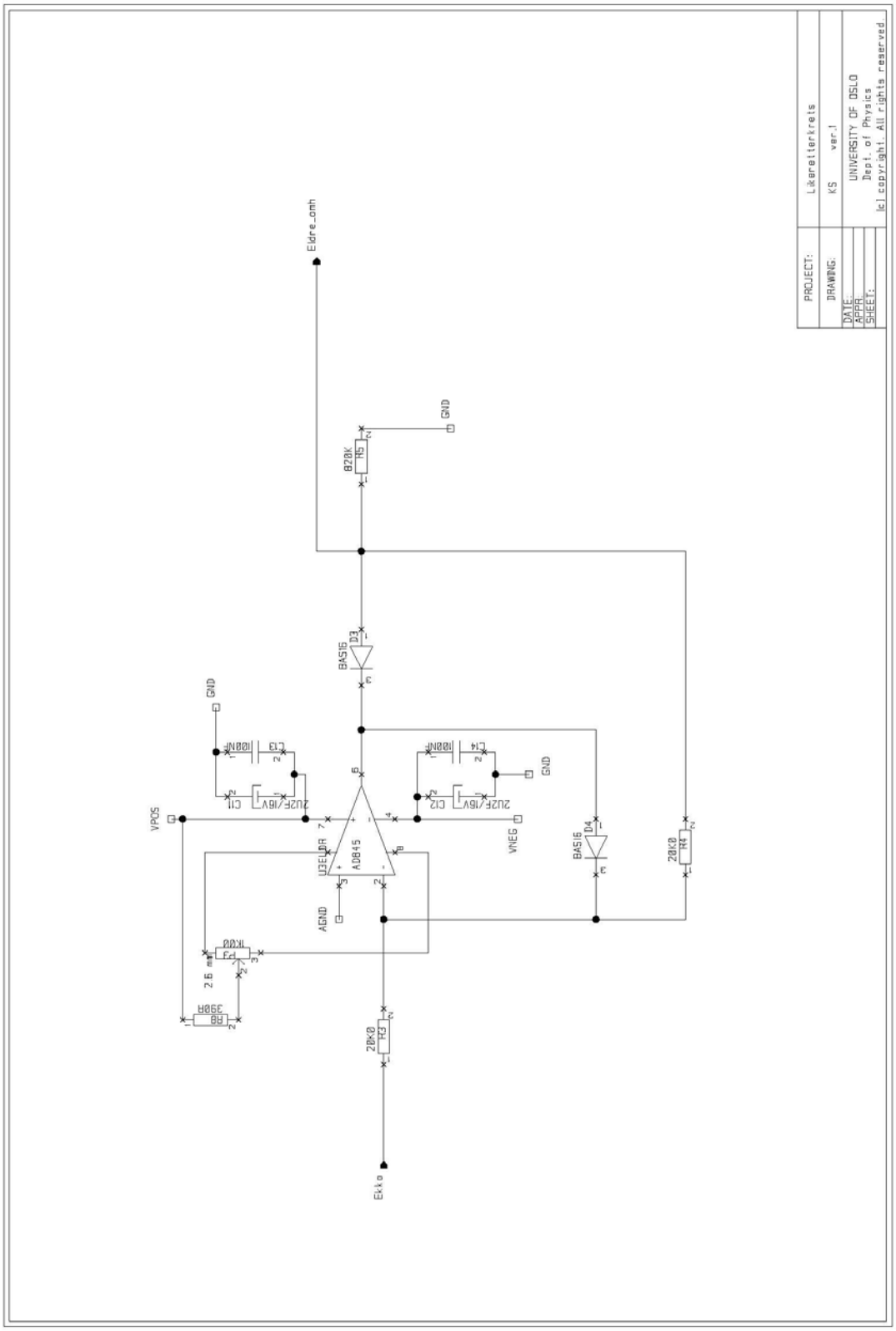
Det vurderes å integrere SPI styrt forsterker med ytligre forsterkning. Denne skal kunne styres fra uC, for variasjon i den totale forsterkning. Man kan da velge ønsket rekkevidde m.h.p. fiskestørrelse.

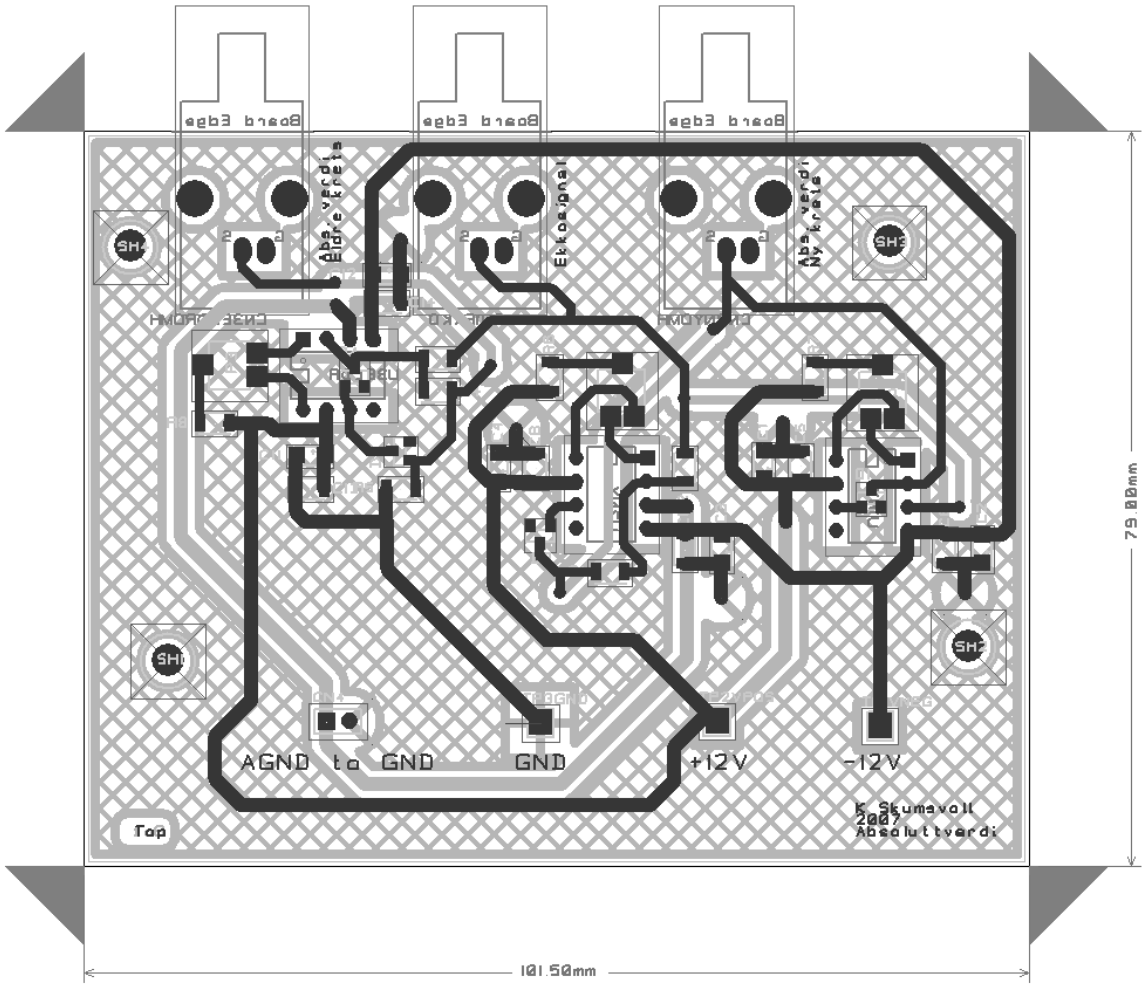
Vedlegg 3 PCB og skjemategning av likeretter krets og krets for omhylningskurve

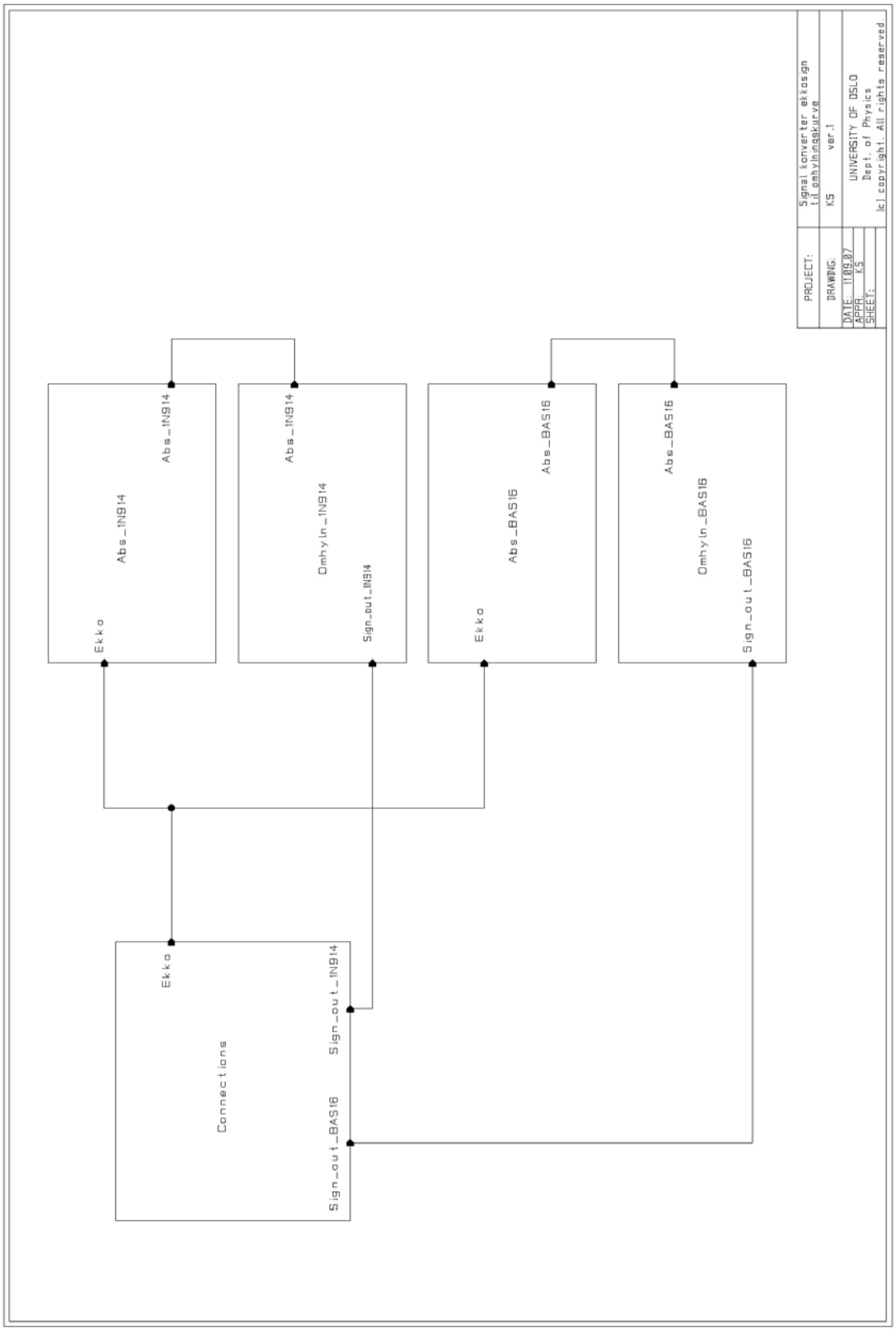


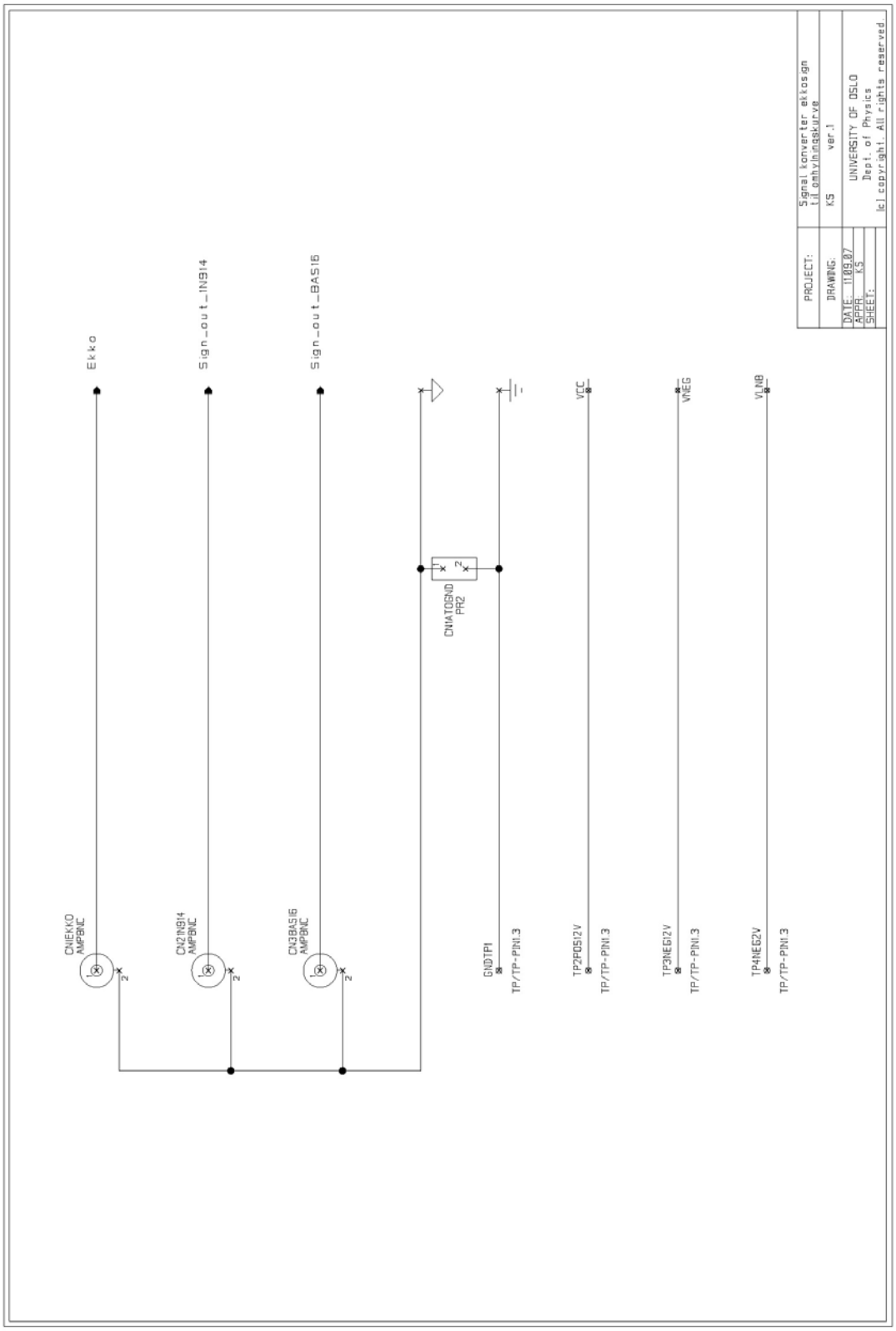

PROJECT: Likeretterkrets
DRAWING: KS ver 1
DATE:
APPR:
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
CN4
PR2
1
2
VPOS
VNEG
Ekko
Ny_omh
Eldre_omh
TP3GND
TP/TP-PIN13
TP2VPOS
TP/TP-PIN13
TP1VNEG
TP/TP-PIN13
CN1EKKO
AMPBNC
CN2NYOMH
AMPBNC
CN3ELDROMH
AMPBNC

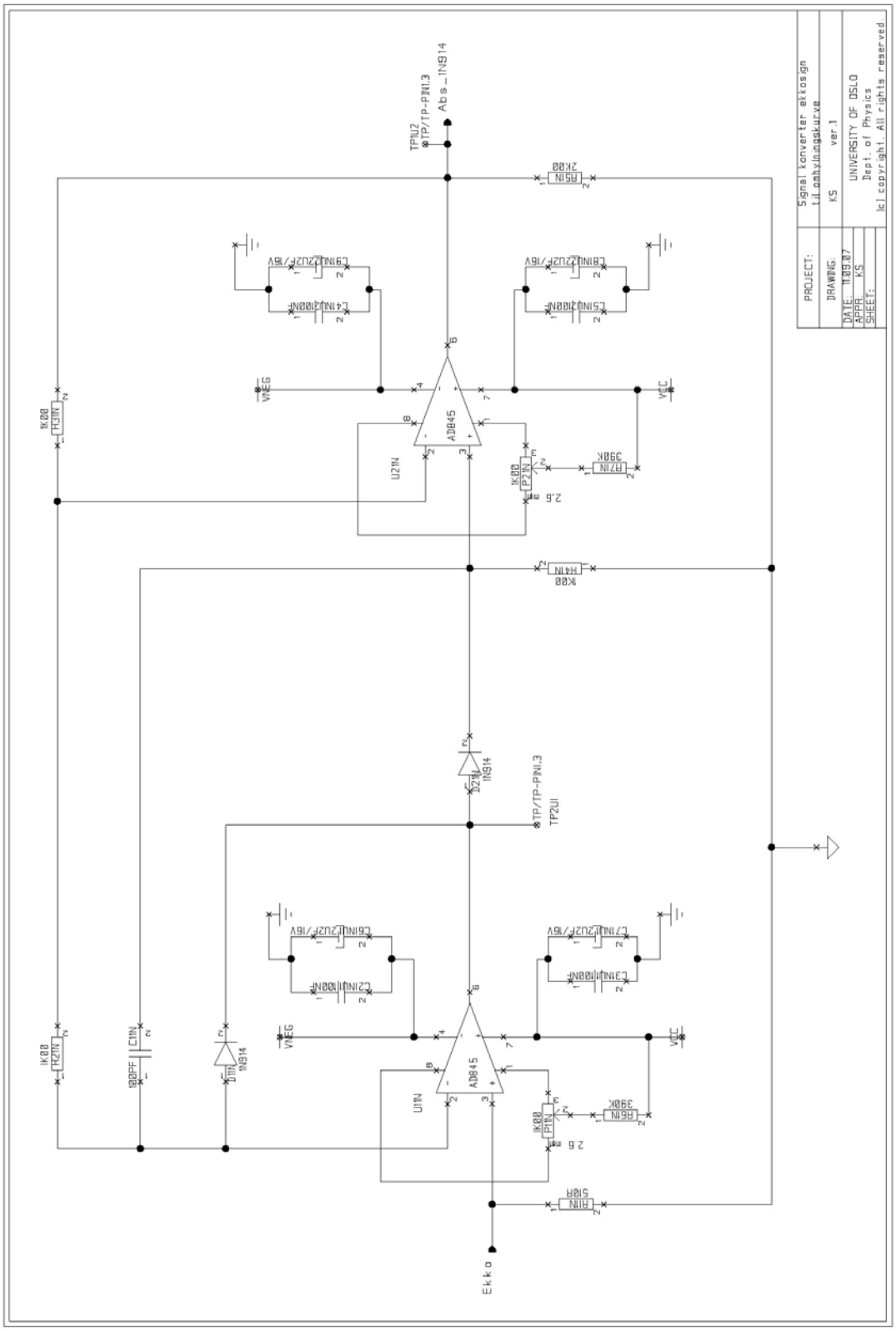

Likeretterkrets
KS
ver.1
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
PROJECT:
DRAWING:
DATE:
APPR:
SHEET:
Ny_omh
Ekko
GND
BAS16
D1
D2
100N
C5
C6
C7
C8
2U2F/16V
C1
C2
C3
C4
VNEG
VPOS
AD845
UINY
U2KY
AGND
1K00
R1
R2
R3
R4
R5
R6
2.6 mm
390R
H7
H8

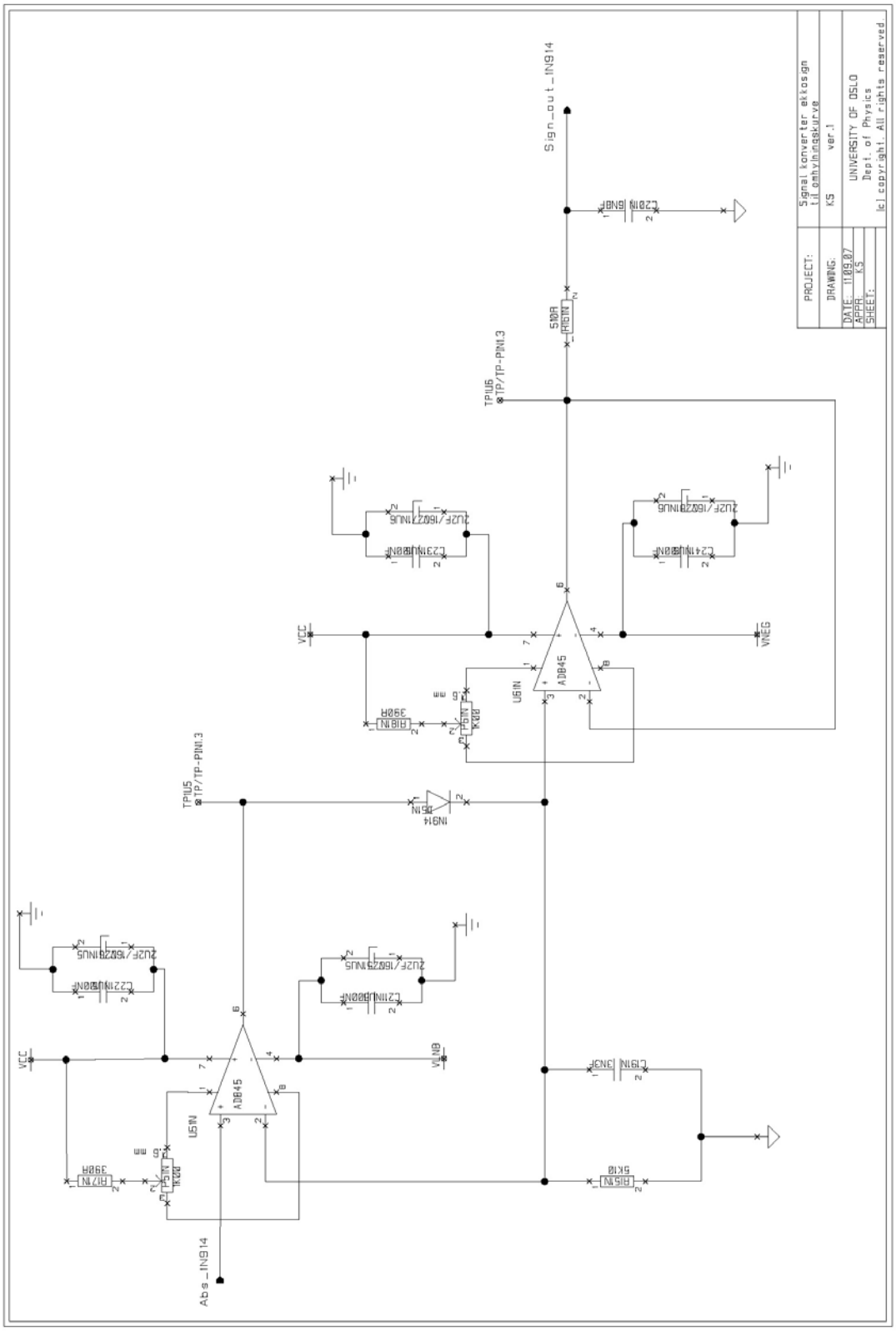

PROJECT: Likerettekrets
DRAWING: KS ver.1
DATE:
APPR:
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
Evdre_omh
GND
820K R5
BAS16 D3
GND
100NF C13
2U2F/16V C11
100NF C14
2U2F/16V C12
GND
VPOS
VNEG
U3ELMDR
AD845
AGND
BAS16 D4
20K0 R4
1K00 P2
2.6 mA
390R R8
20K0 R3
Ekte

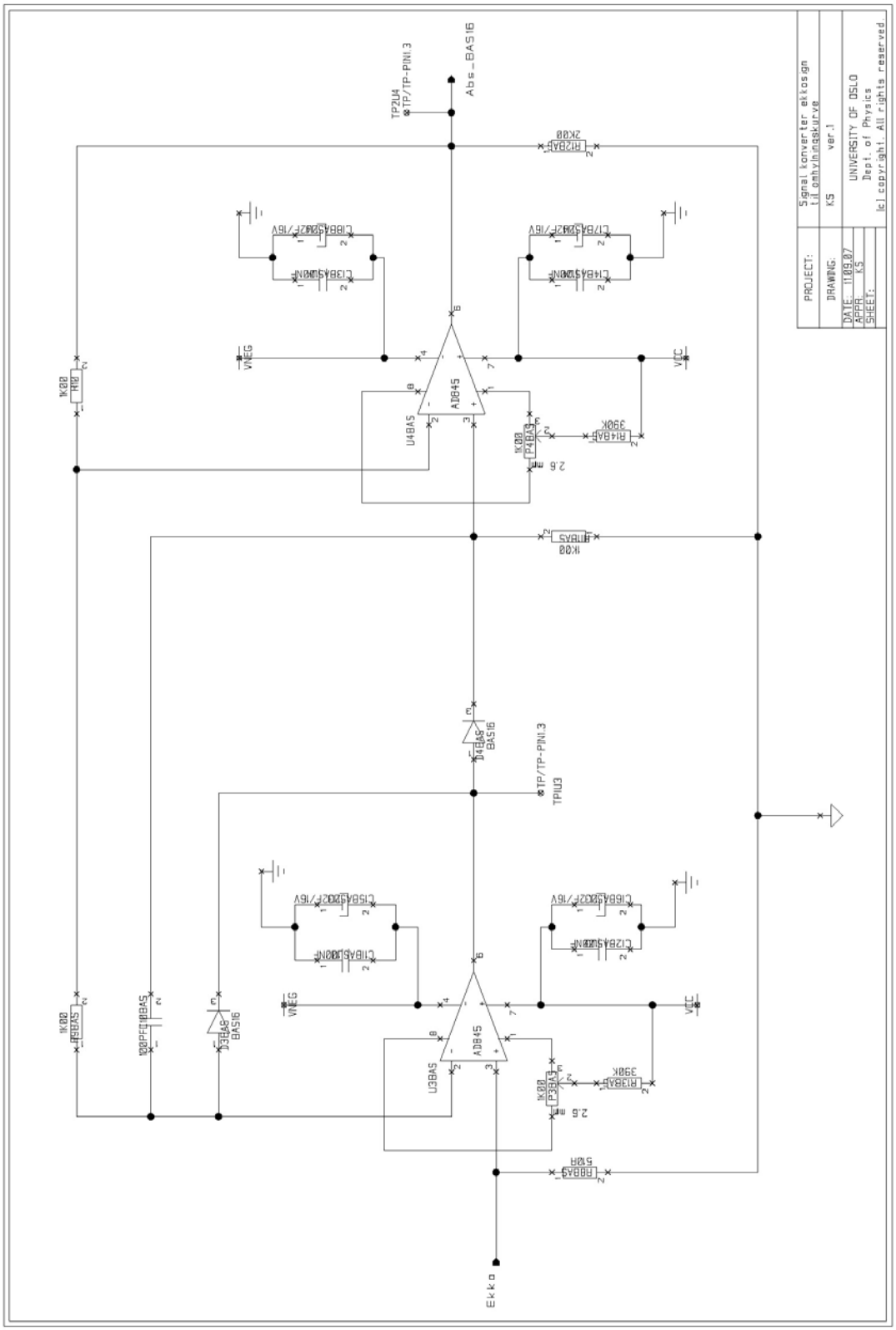

Board Edge
Abs verdi Eldre krets
Board Edge
Board Edge
Abs verdi Ny krets
SH1
SH2
SH3
AGND to GND
GND
+12V
-12V
K Skumsvoll 2007
Absoluttverdi
Top
79.00mm
101.50mm

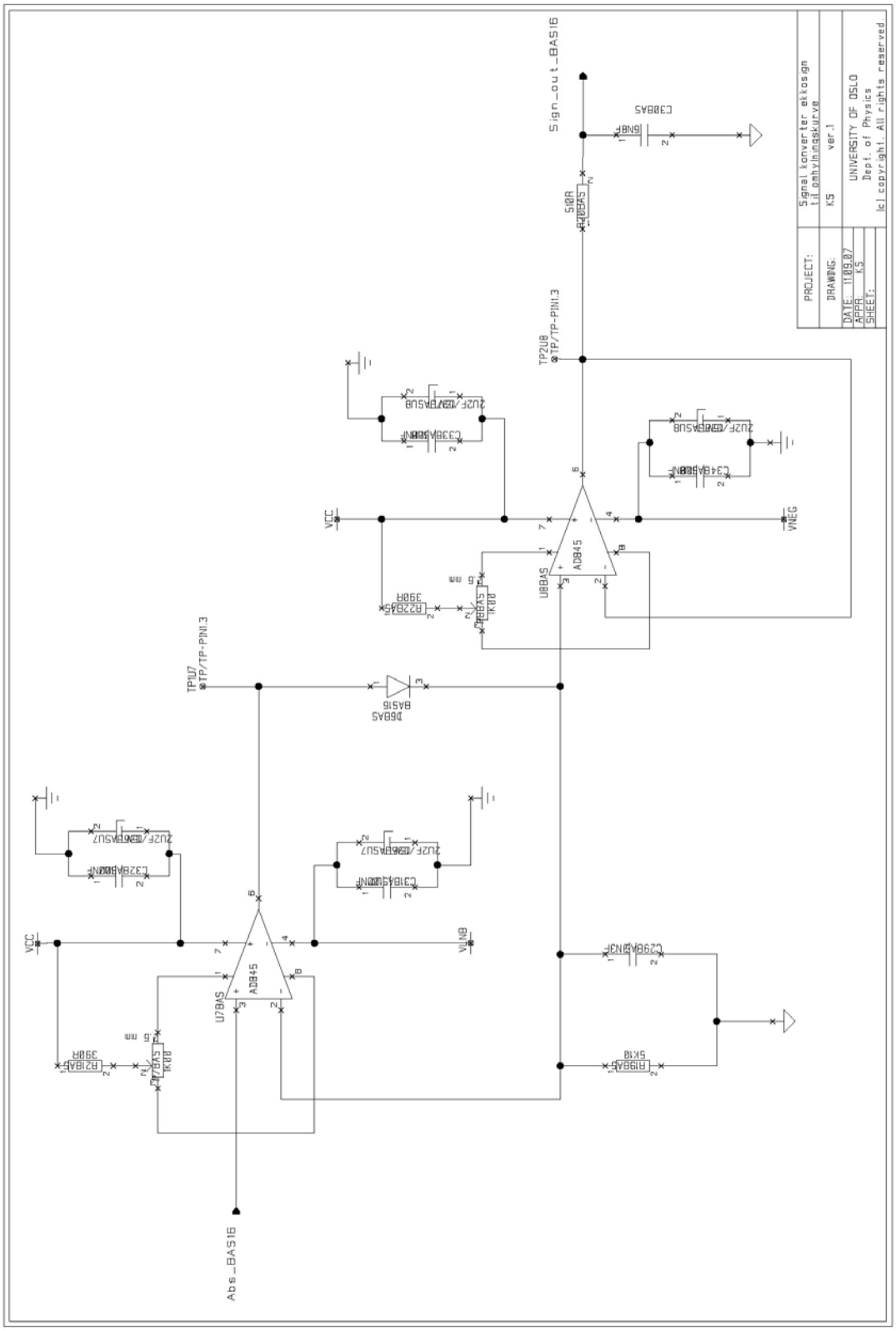

PROJECT:
Signal konverter ekkosign
til omhyllingskurve
DRAWING:
KS
ver.1
DATE: 11.09.07
APPR: KS
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
Abs_ING14
Abs_ING14
Ekko
Omhyll_ING14
Abs_ING14
Sign_out_ING14
Abs_BAS16
Abs_BAS16
Ekko
Omhyll_BAS16
Abs_BAS16
Sign_out_BAS16
Ekko
Connections
Sign_out_ING14
Sign_out_BAS16

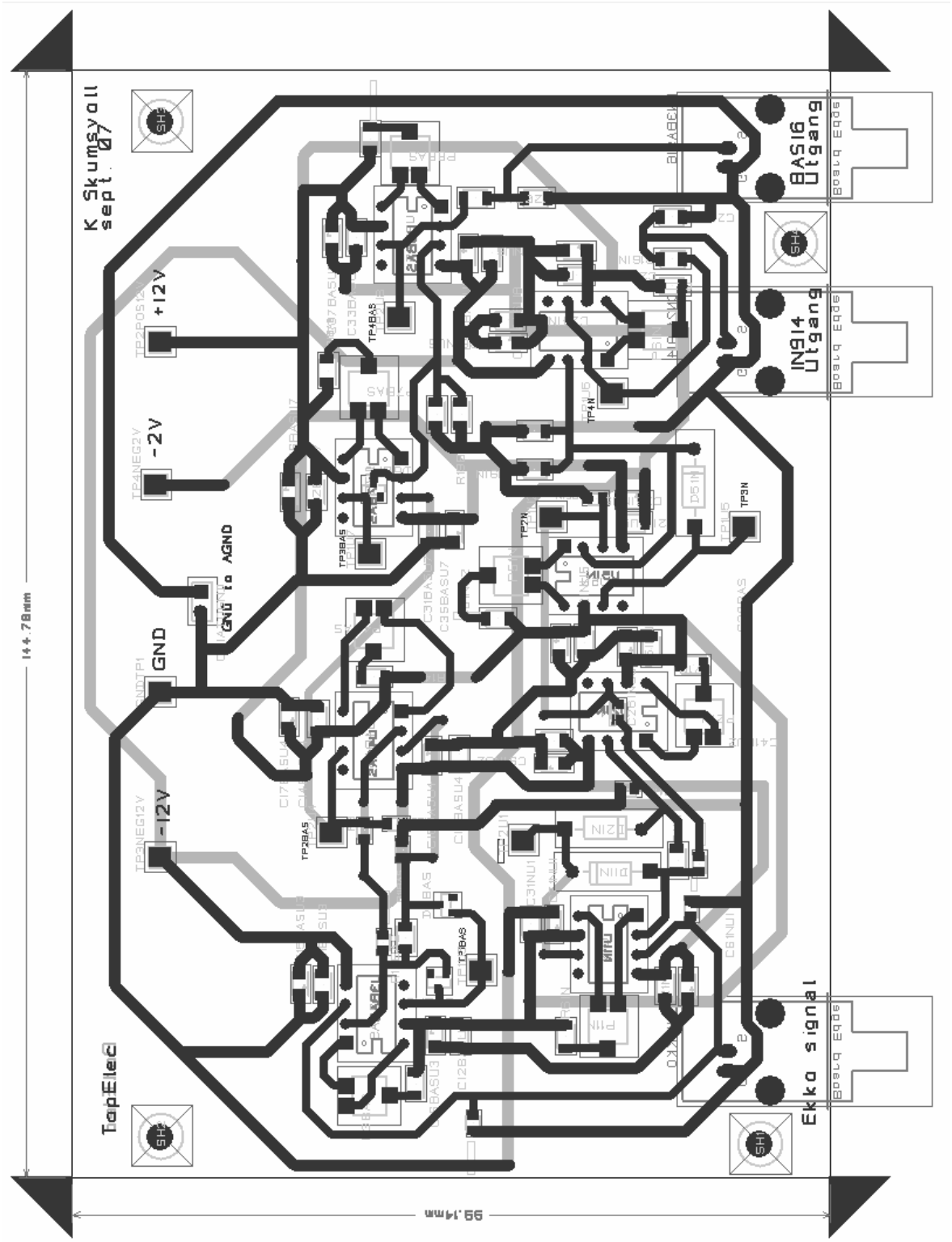

Ekko
Sign_out_IN914
Sign_out_BAS16
CN1EKKO
AMPBNC
CN2IN914
AMPBNC
CN3BAS16
AMPBNC
CN1ATOGND
PR2
GNDTP1
TP/TP-PIN1.3
TP2POS12V
TP/TP-PIN1.3
TP3NEG12V
TP/TP-PIN1.3
TP4NEG2V
TP/TP-PIN1.3
VCC
VNEG
VLNB
PROJECT:
Signal konverter ekkosign
til omhyllingskurve
DRAWING:
KS
ver.1
DATE: 11.09.07
APPR: KS
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved


Signal konverter ekkosign
til oshyningskurve
ver.1
KS
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
PROJECT:
DRAWING:
DATE: 11.09.07
APPR: KS
SHEET:
TPU2
@TP/TP-PIN1.3
Abs-IN914
2K00
R5IN
C8INU22UF/16V
C9INU22UF/16V
C4INU100NF
C5INU100NF
VNEG
VCC
U2IN
AD845
1K00
P2IN
2.5 m
39OK
R7IN
1K00
R6IN
2K00
R4IN
D2IN
IN914
TP/TP-PIN1.3
TP2UI
C6INU22UF/16V
C7INU22UF/16V
C2INU100NF
C3INU100NF
VNEG
VCC
U1IN
AD845
1K00
R2IN
10PF
C1IN
D1IN
IN914
1K00
P1IN
2.5 m
39OK
R8IN
51OH
R1IN
Ekko

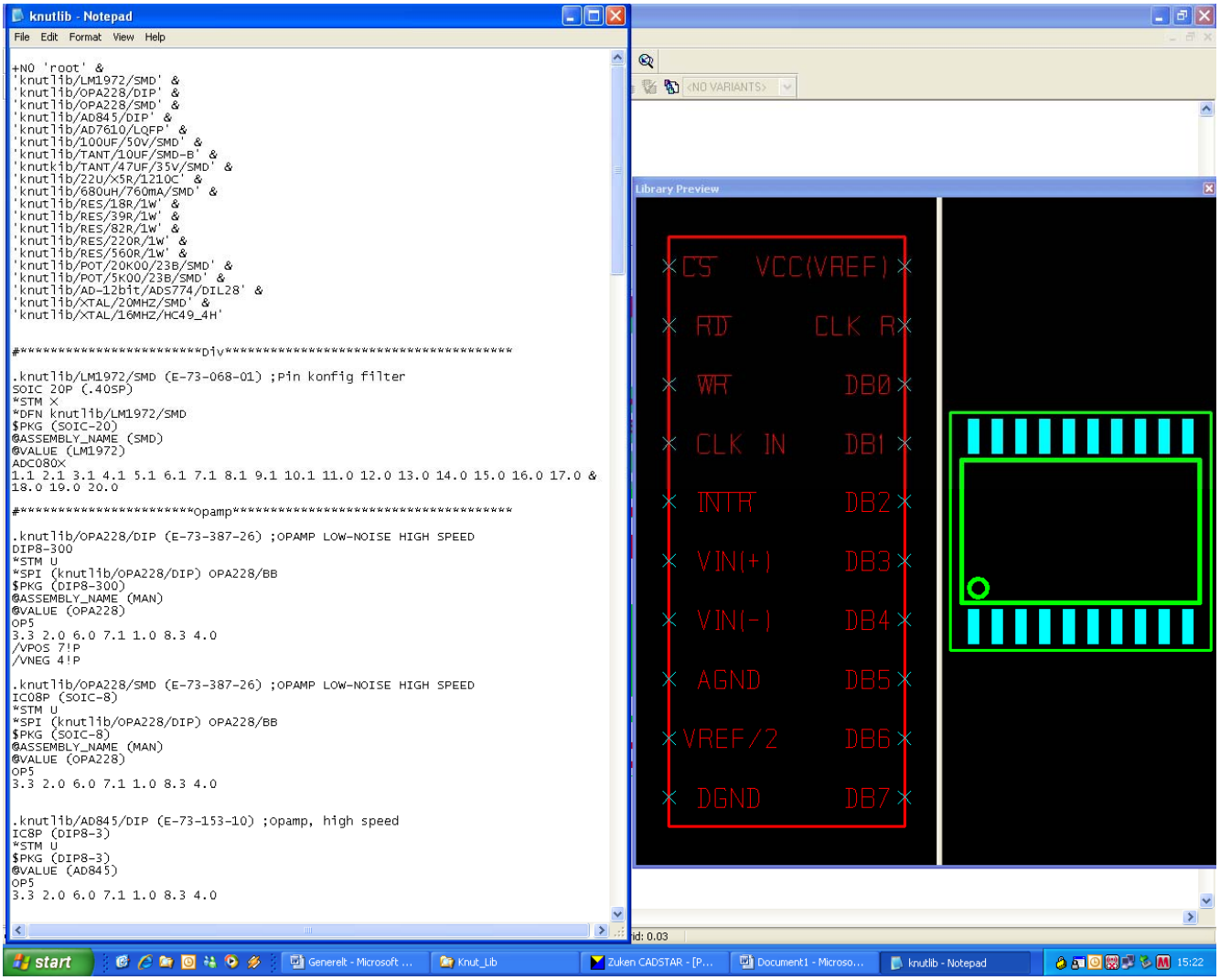

Signal konverter ekkosign
til oshyhnmgskurve
ver.1
KS
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
PROJECT:
DRAWING:
DATE: 11.09.07
APPR: KS
SHEET:
Sign_out_IN914
TPU6
TP/TP-PIN1.3
TPU5
TP/TP-PIN1.3
VCC
VNEG
VLNB
U5IN
ADB45
IN914
Abs_IN914


Signal konverter ekkosign
Lid oghyhnngskurve
ver.1
KS
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
PROJECT:
DRAWING:
DATE: 11.09.07
APPR: KS
SHEET:
Abs_BAS16
TP2U4
TP/TP-PIN1.3
AD845
U4BAS
U3BAS
VNEG
VCC
TP1U3
Ekko

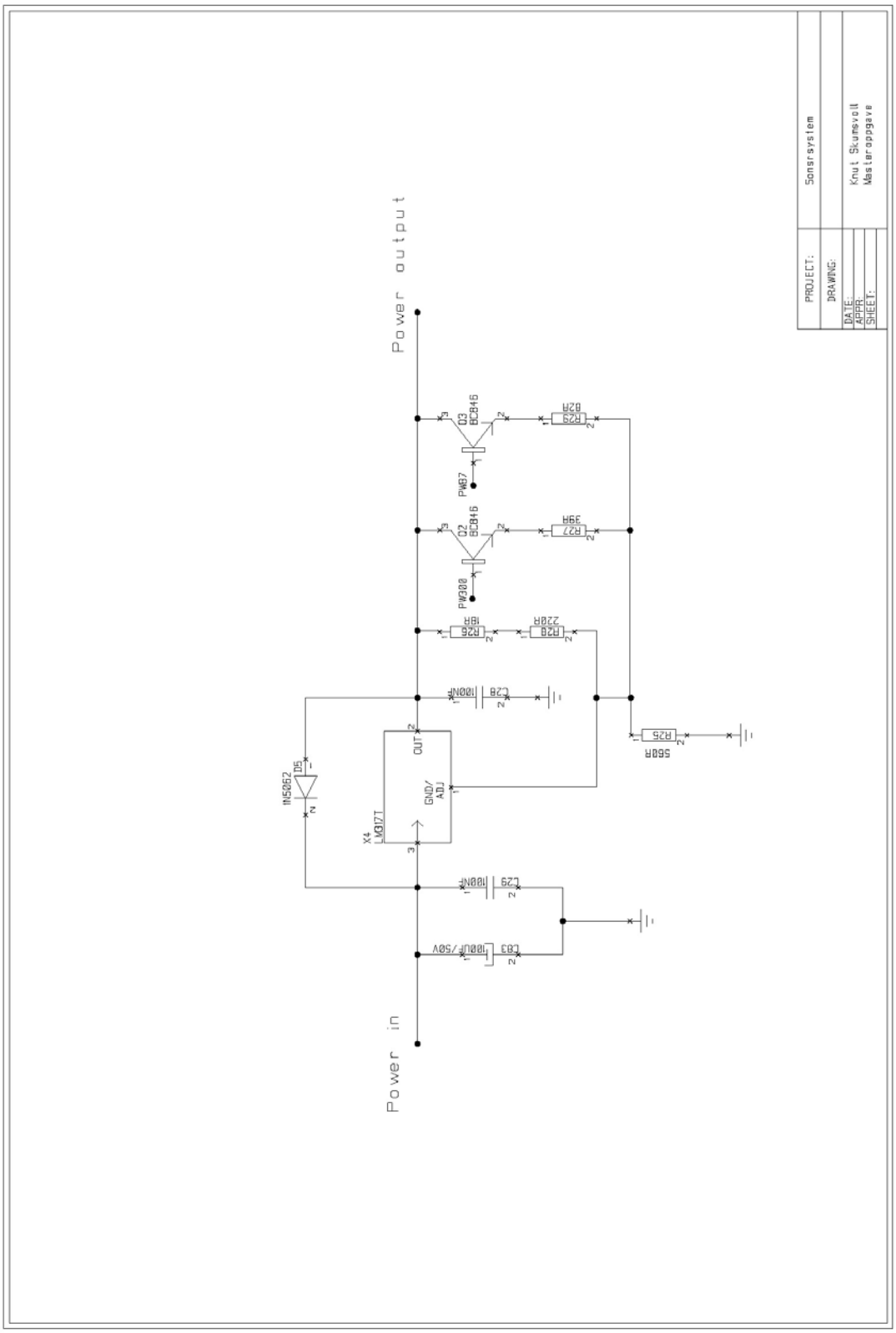

Sign_out_BAS16
C30BAS
5NB
50R
R40BAS
TP2U8
TP/TP-PIN13
2U2F/6V3BASU8
C33BA5
2U2F/6V3BASU8
C34BA5
VCC
VNEG
U8BAS
AD845
R228A5
390R
C35BA5
1K00
TP1U7
TP/TP-PIN13
D6BAS
BAS16
2U2F/6V3BASU7
C32BA5
2U2F/6V3BASU7
C31BA5
VCC
VNB
U7BAS
AD845
C29BA5
R19BA5
5K10
R21BA5
390R
C36BA5
1K00
Abs_BAS16
PROJECT:
Signal konverter ekkosign
tid oghyhnmgskurve
DRAWING:
ver.1
DATE: 11.09.07
APPR: KS
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved


K Skumsvoll
sept. 07
+12V
-2V
GND
-12V
GND to AGND
BAS16
Utgang
IN914
Utgang
Ekko signal
TopElec
144.78mm
99.14mm

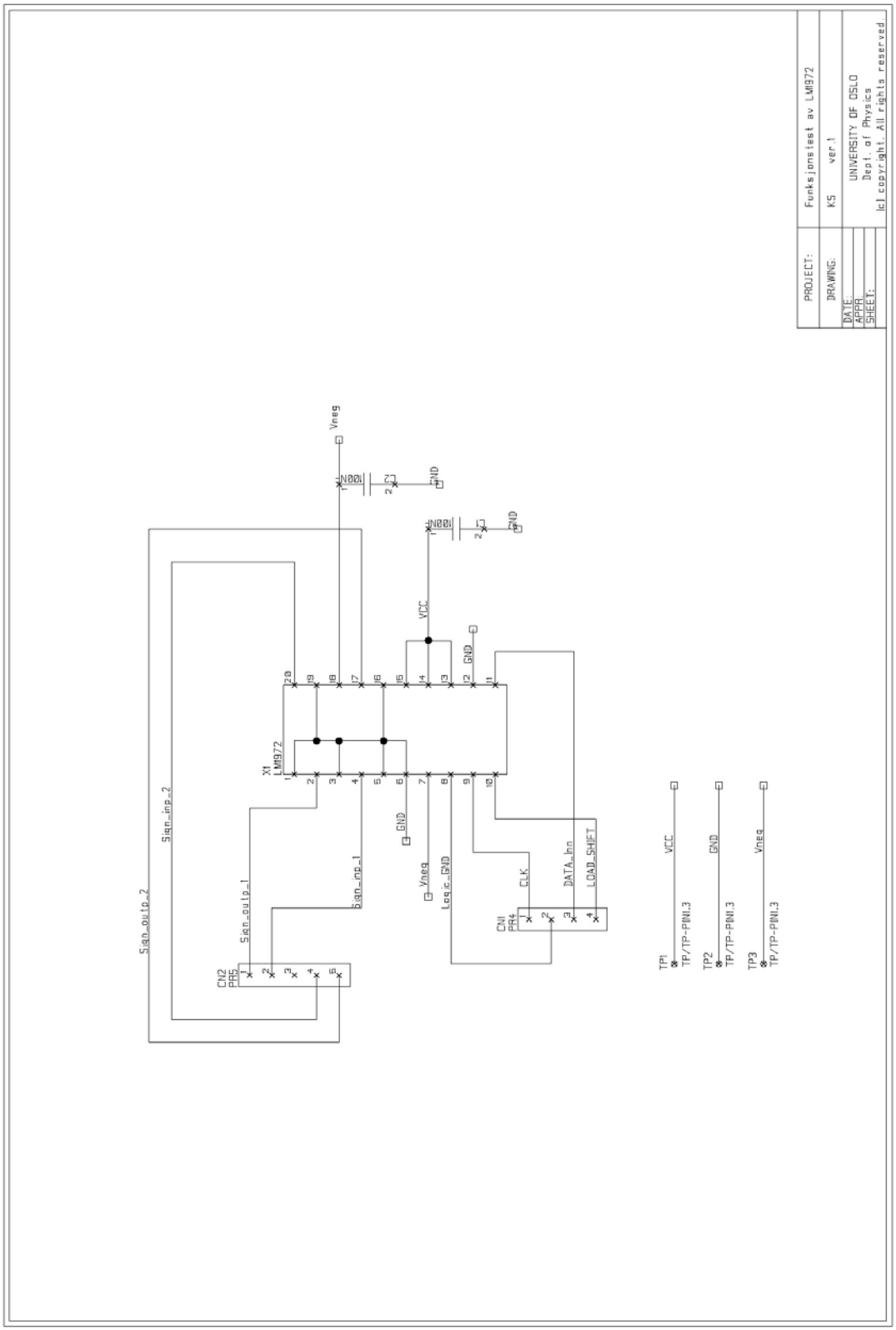
Vedlegg 4 Eksempel på egentegnet komponent for bruk i CADStar

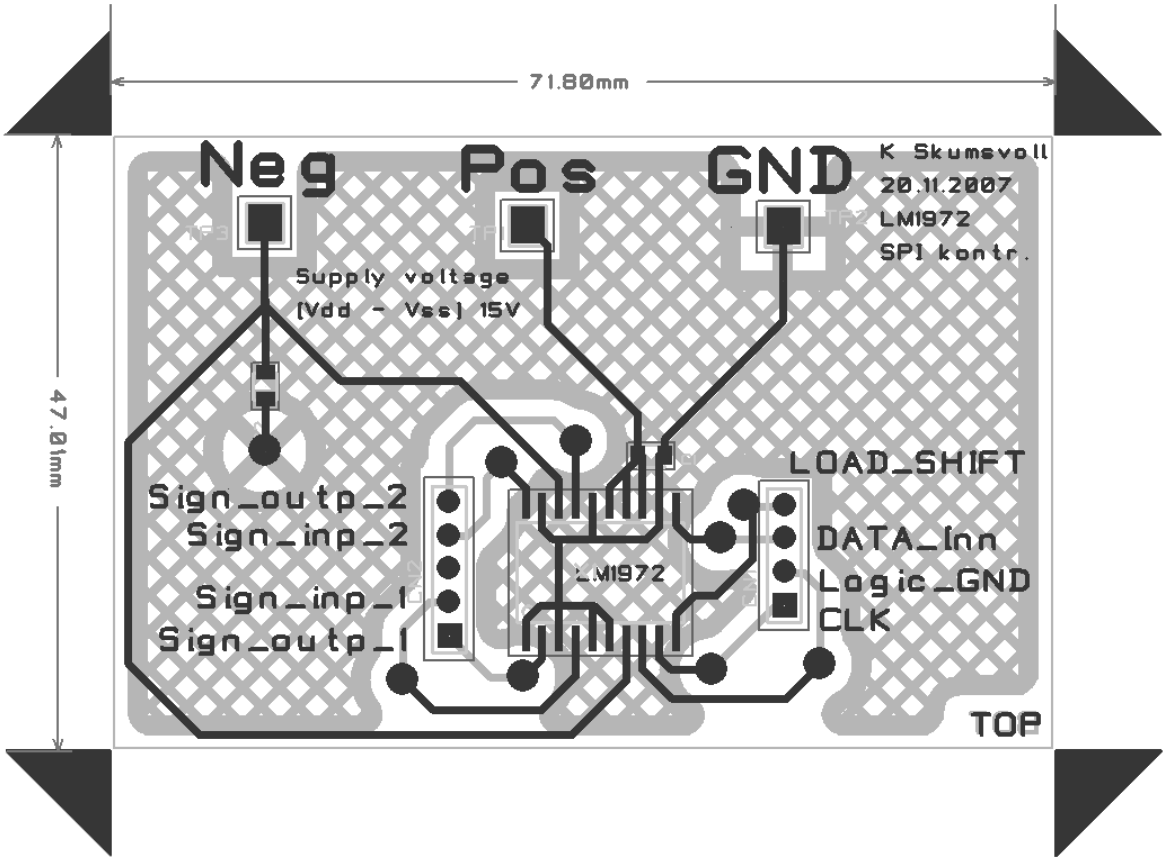


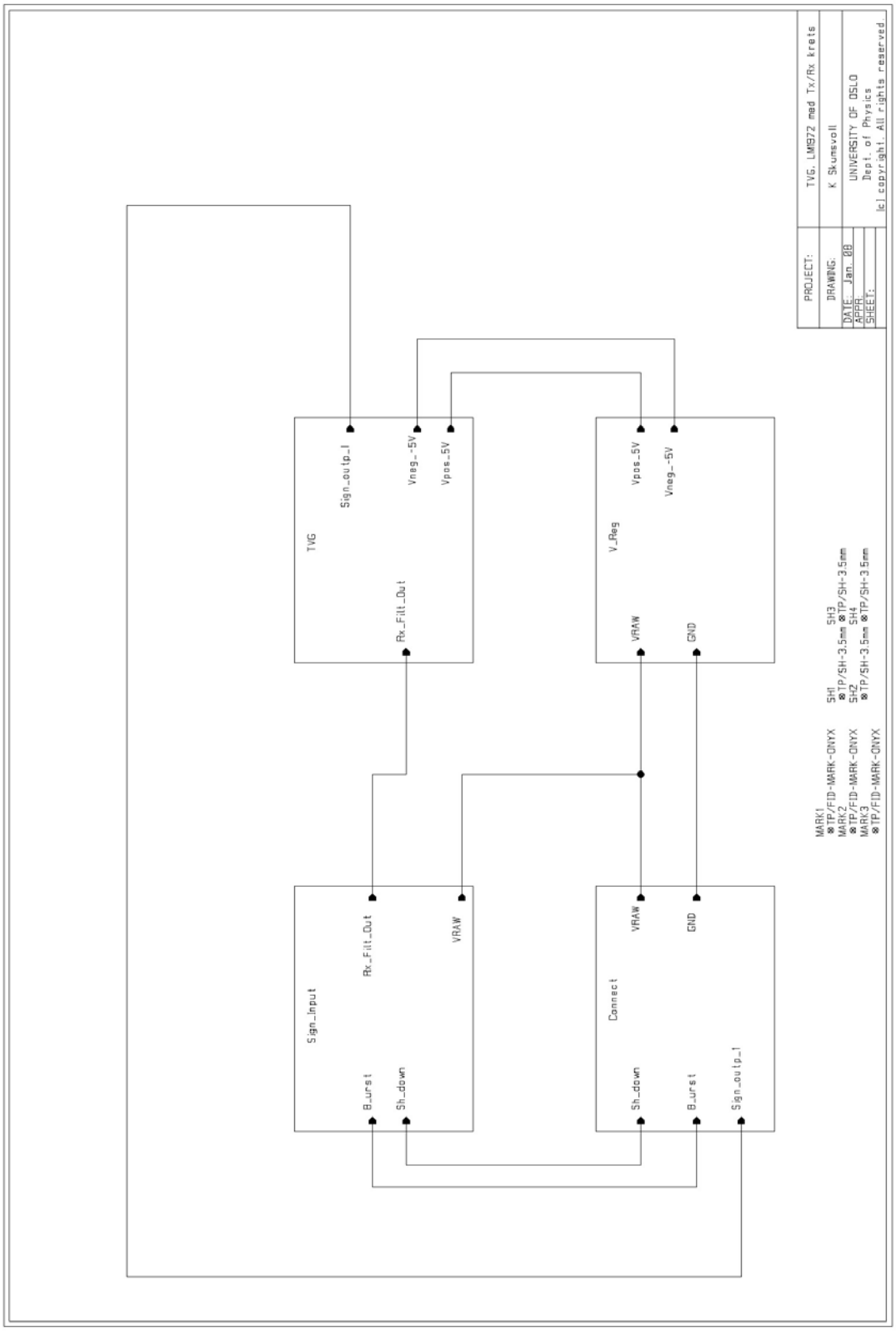
Vedlegg 5 Skjemategning av transistorkontrollert spenningsregulator

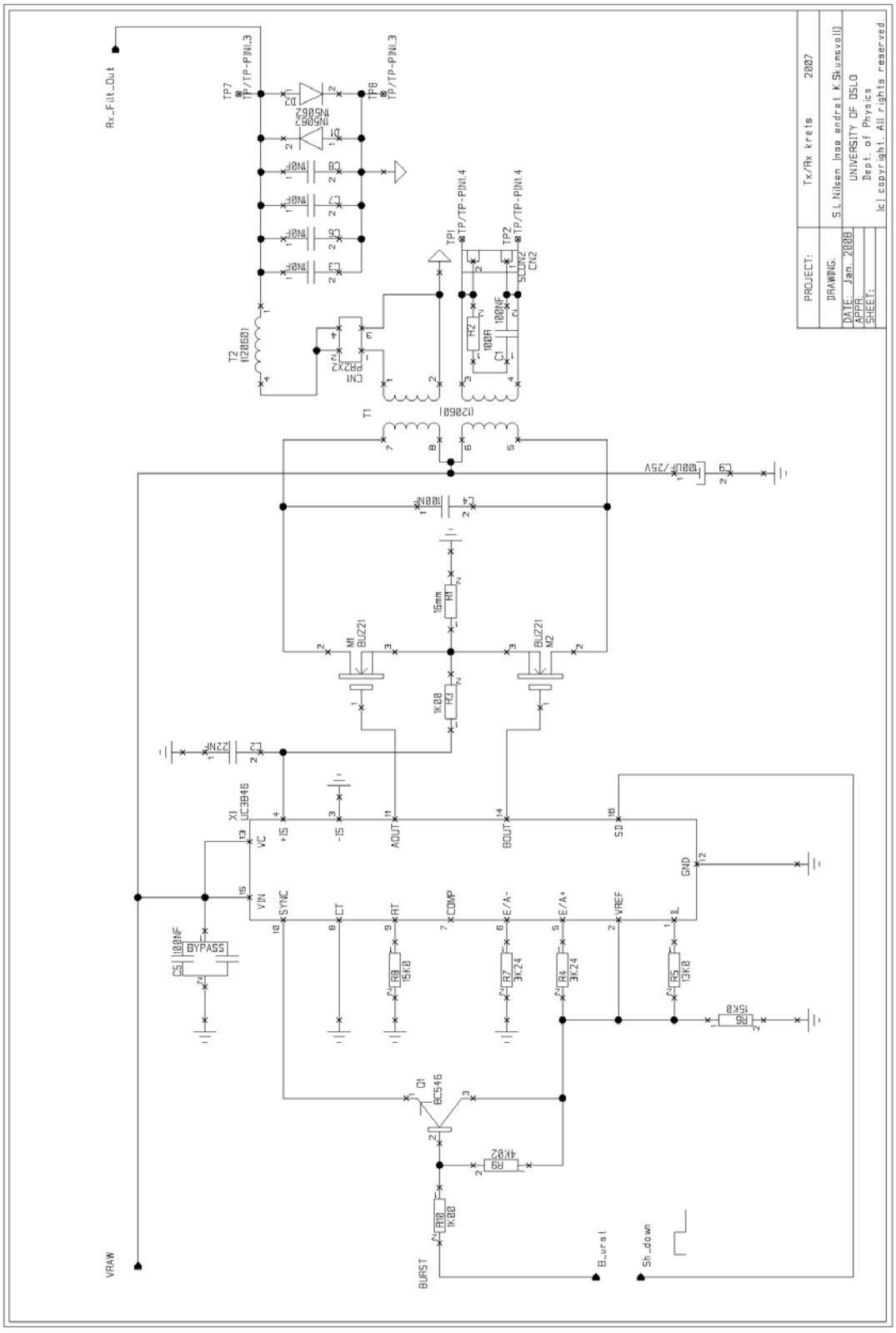


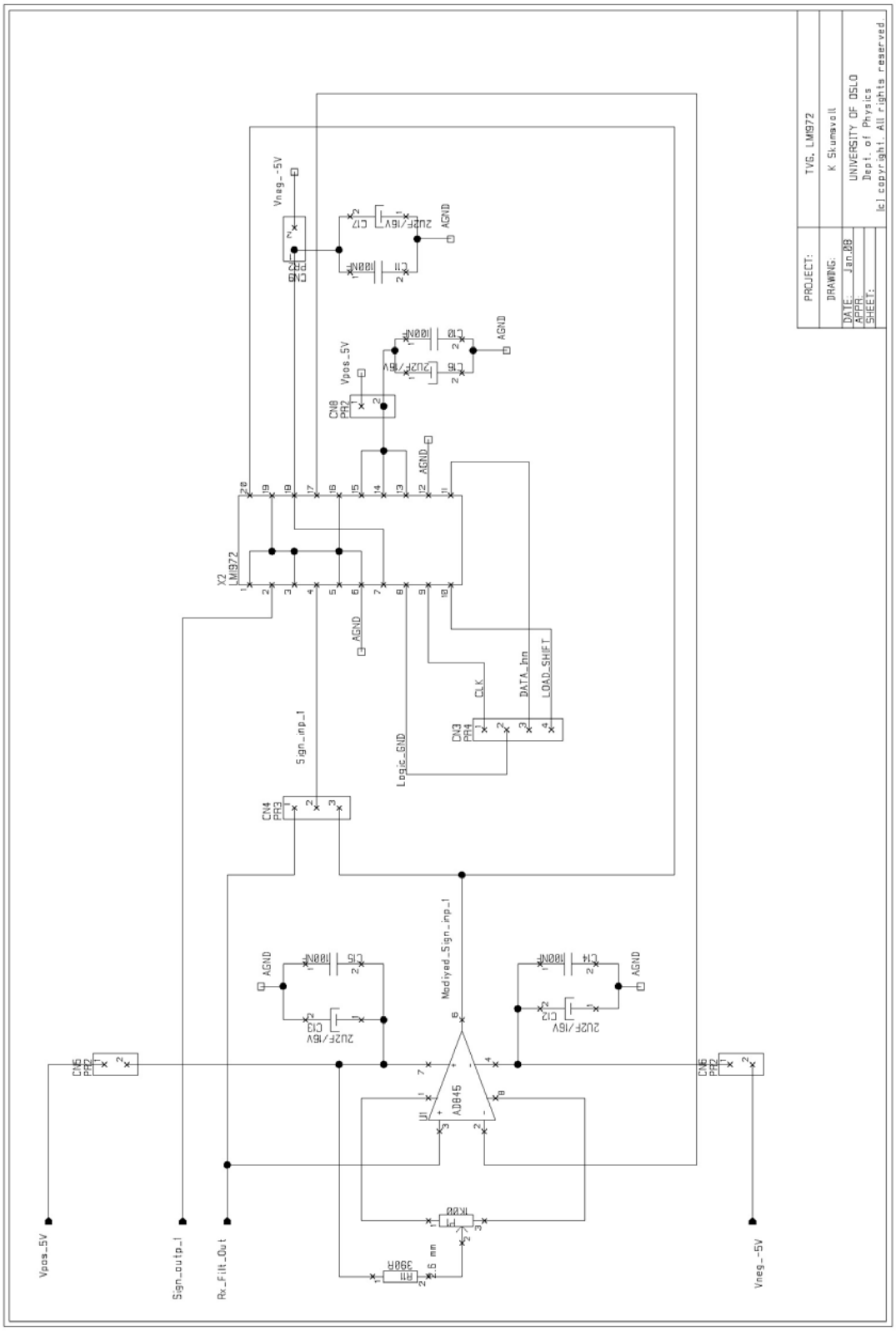
Vedlegg 6 Enkel PCB og skjemategning av testkort LM1972

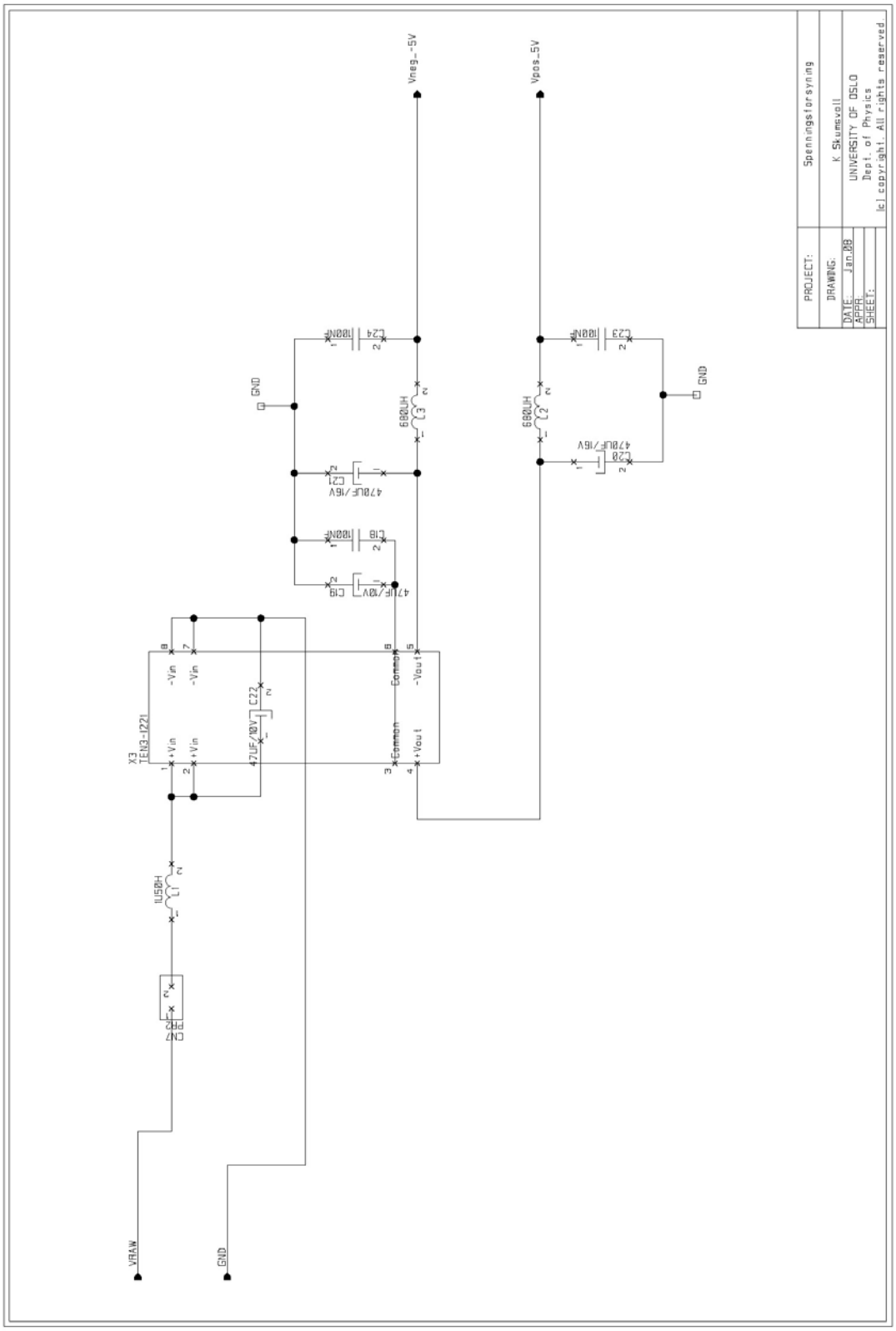


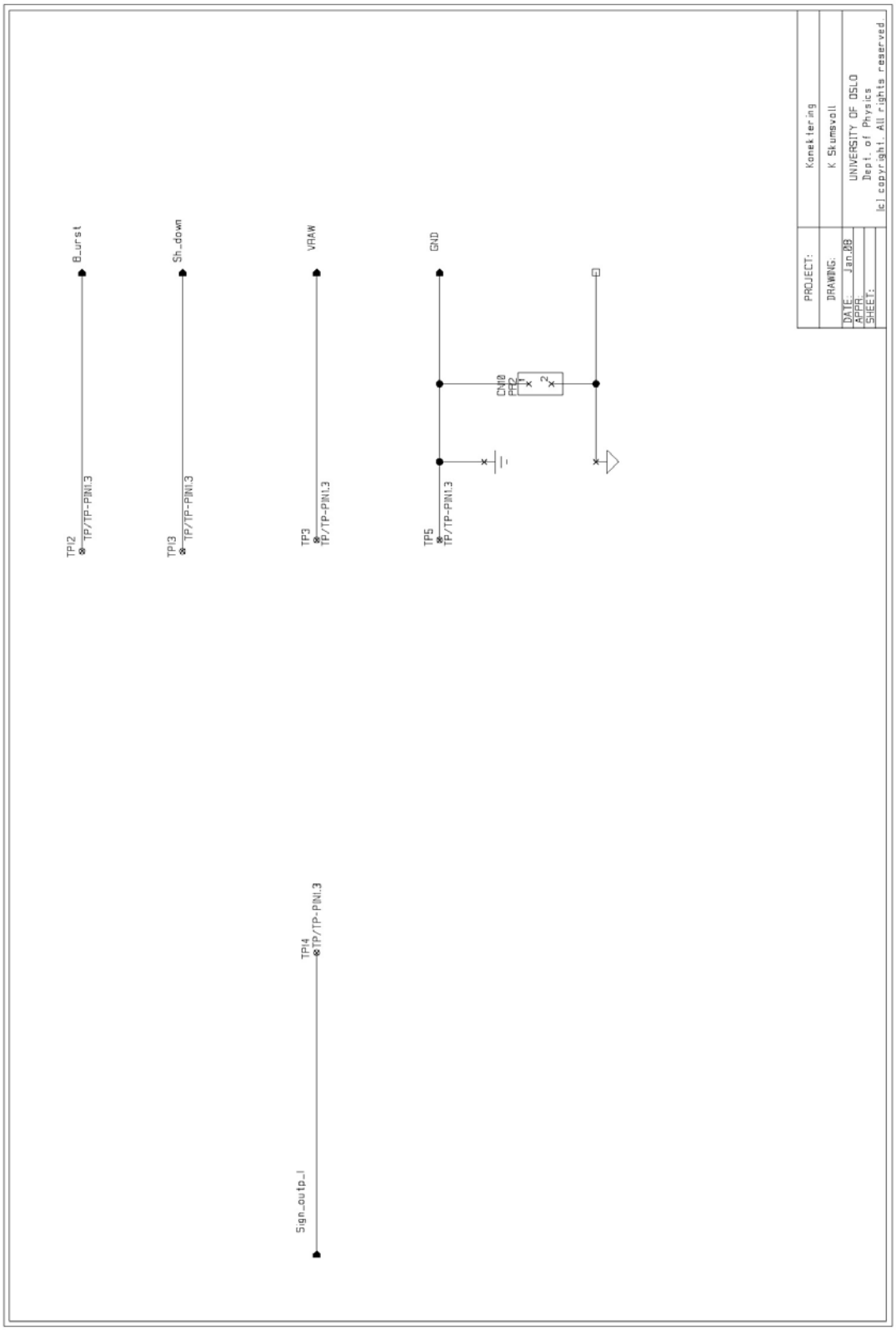

71.80mm
47.01mm
Neg
Pos
GND
K Skumsvoll
20.11.2007
LM1972
SPI kontr.
Supply voltage
(Vdd - Vss) 15V
LOAD_SHIFT
Sign_outp_2
Sign_inp_2
DATA_inn
LM1972
Logic_GND
Sign_inp_1
CLK
Sign_outp_1
TOP

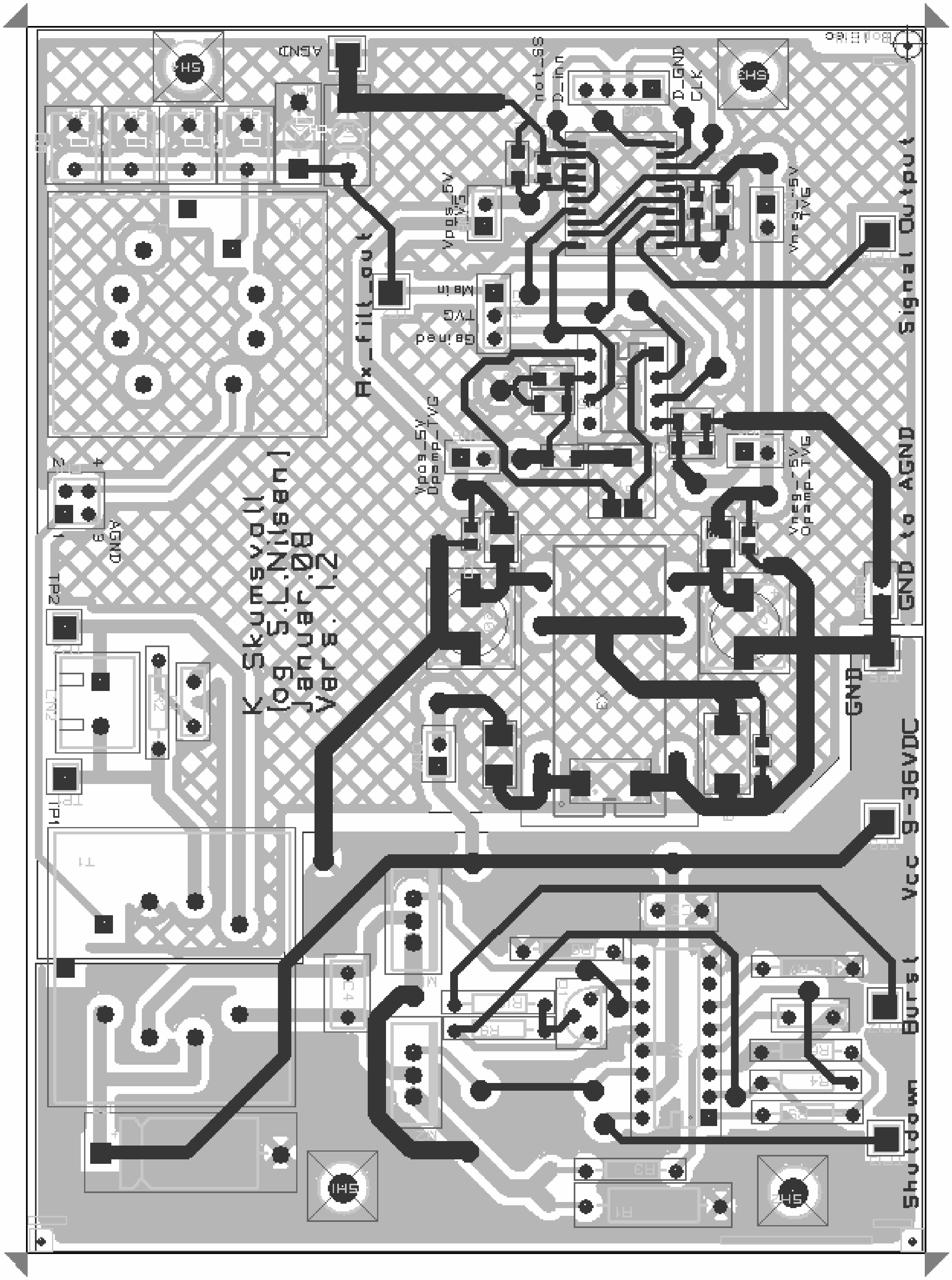

PROJECT:
TVG, LM972 med Tx/Rx krets
DRAWING:
K Skumsvoll
DATE: Jan 08
APPR:
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
TVG
Sign_outp_I
Vneg_-5V
Vpos_5V
Rx_Filt_Out
V_Reg
Vpos_5V
Vneg_-5V
VRAW
GND
Sign_Input
Rx_Filt_Out
VRAW
Burst
Sh_down
Connect
VRAW
GND
Sh_down
Burst
Sign_outp_I
MARK1
TP/FID-MARK-ONYX
MARK2
TP/FID-MARK-ONYX
MARK3
TP/FID-MARK-ONYX
SH1
TP/SH-3.5mm
SH2
TP/SH-3.5mm
SH3
TP/SH-3.5mm
SH4
TP/SH-3.5mm


Rx_Filt_Out
TP7
TP/TP-PIN1.3
TP8
TP/TP-PIN1.3
D2
D1
1N5062
C8
C7
C6
C5
100NF
TP1
TP/TP-PIN1.4
TP2
TP/TP-PIN1.4
CON2
R2
100R
C1
100NF
T2
M2060I
CN1
DB2X2
T1
M2060I
C9
100U/25V
C4
100NF
M1
BUZ21
M2
BUZ21
15mm
R1
1K00
R3
C2
22NF
X1
UC3846
VC
+IS
-IS
ADJT
BOUT
SD
GND
VIN
SYNC
CT
RT
COMP
E/A-
E/A+
VREF
UL
C6
100NF
BYPASS
R8
15K0
R7
3K24
R4
3K24
R5
13K0
R6
15K0
Q1
BC546
R9
4K02
R10
1K00
VRAW
BURST
Burst
Sh_down
PROJECT:
Tx/Rx krets
2007
DRAWING:
S.L.Nilsen Inga andrea K.Skumsvoll
DATE: Jan 2008
UNIVERSITY OF OSLO
Dept. of Physics
APPR:
SHEET:
(c) copyright. All rights reserved


PROJECT: TVG, LM972
DRAWING: K Skumsvoll
DATE: Jan.08
APPR:
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved
Vneg_-5V
Vpos_5V
AGND
X2 LM972
CN3 PR4
CN4 PR3
CN5 PR2
CN6 PR2
CN8 PR2
U1 AD845
Sign_inp_I
Logic_GND
CLK
DATA_hnn
LOAD_SHIFT
Modified_Sign_inp_I
Sign_outp_I
Rx_Filt_Out
2U2F/16V
100N
1K00
390R


Vneg_-5V
Vpos_5V
GND
GND
GND
GND
VRAW
680UH
680UH
ILSBH
X3
TEN3-1221
-Vin
-Vin
+Vin
+Vin
Common
Common
-Vout
+Vout
C22
47UF/10V
C24
100N
C23
100N
C18
100N
C21
47UF/16V
C19
47UF/10V
C20
47UF/16V
PROJECT:
Spenningsforsyning
DRAWING:
K Skumsvoll
DATE: Jan.09
APPR:
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved


B_urst
Sh_down
VRAW
GND
TP12
TP/TP-PIN1.3
TP13
TP/TP-PIN1.3
TP3
TP/TP-PIN1.3
TP5
TP/TP-PIN1.3
CN10
PIP2
1
2
TP14
TP/TP-PIN1.3
Sign_outp_l
PROJECT:
Konektering
DRAWING:
K Skumsvoll
DATE: Jan.09
APPR:
SHEET:
UNIVERSITY OF OSLO
Dept. of Physics
(c) copyright. All rights reserved


Signal Output
GND to AGND
GND
Vcc 9-36VDC
Burst
Shutdown
Ax_filt_out
AGND
not_SS
D_inn
D_GND
CLK
Main
TVG
Gained
Vpos_5V
Vneg_-5V
Opamp_TVG
TP1
TP2
T1
K Skumsvoll

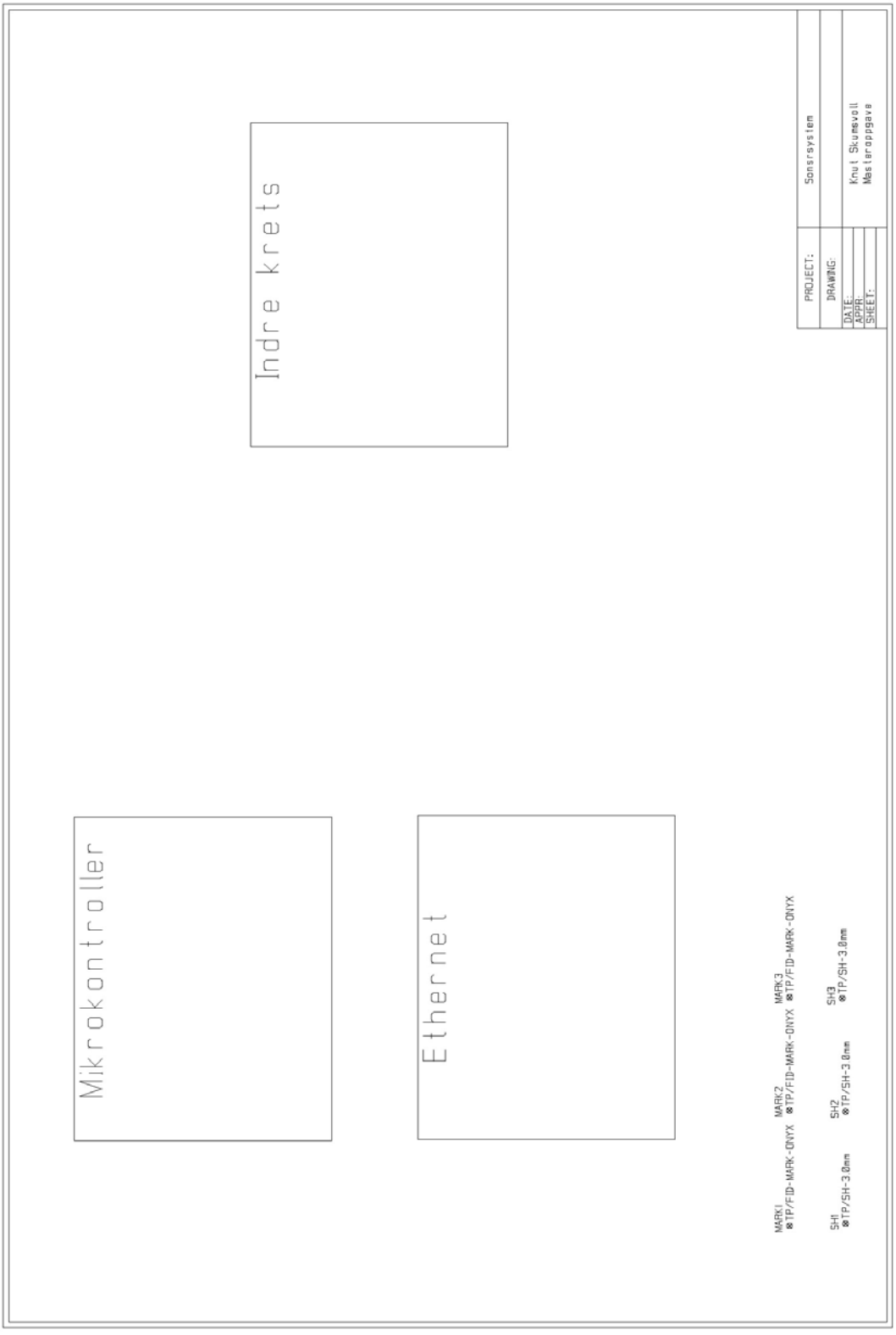
Vedlegg 7 C kode for programmering av testkort med LM1972

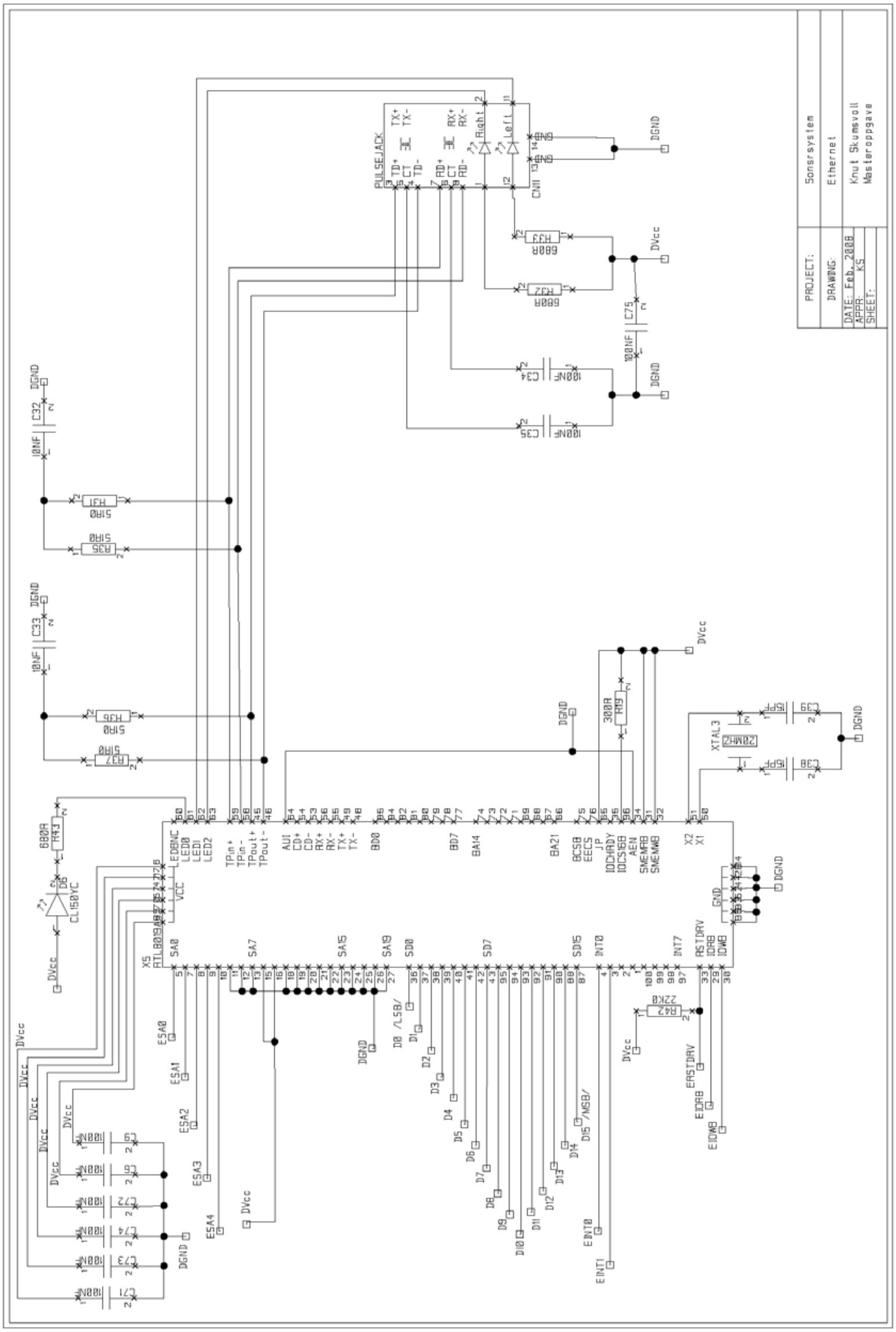
```
#include <avr/io.h>
#include <inttypes.h>
#include <avr/pgmspace.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "Spi.h"
#include <stdio.h>
#include <delay.h>

#define F_CPU = 8MHz

/*************************************************************************************************
ATmega16

SW til PINA
LED til PORTD
ISP,- PORTB 4(SS), 5(MOSI) og 7(SCK)
Ved intrykk av SW0, autoink-/dekrementer starter(autom. økning/minkning av dempning)
Ved intrykk av SW1, autoink-/dekrementer stopper(autom. økning/minkning av dempning)
Ved intrykk av SW2, autodekrementer starter(autom. økning av dempning)
SW6 øker dempningen ett stepp
SW7 minker dempning ett stepp
SW5 resetter dempning lik 0.0 (null)

Til trigger av skop ved enveisdempning, triggersignal PORTE 0

Sender PORTD (LED) til SPI. Led viser dataregister (dempning) for bekreftende verifisering
av den faktiske sendte (SPI) verdi.

*************************************************************************************************/
        int autom=0, okeverdi=0, stige=0, halv=0, helstep=0;
        unsigned char dataverdi=0x00;
        unsigned char adresse=0x00;


ISR (TIMER0_COMP_vect){

        extern int autom,stige,okeverdi,helstep;
        extern char adresse;
        int x=0;

                if(okeverdi==1 && autom!=1){                    //Dataverdi er endret, ikke automatisk
                x=1;
                okeverdi=0;                                     //Endring utføres, variabel resettes
                }

                if(autom==1){                                   //Automatisk inkrementer/dekrementer
                x=1;
                if(stige==1){
                PORTC |= (1<<0);                                //Trigger settes, PORTE0
                if(PORTD>0x9F && PORTD<=0xFE)PORTD++;           //Inkrement om er stigende, og mindre enn 0xFE (teller ned)

                else if(PORTD<=0x9F){                           //Da LM1972 øker m ett steg fra 48dB tom. 78.0dB (og 100dB)
                        if(helstep==1){
                                PORTD++;
                                helstep=0;
                                }
                                else helstep=1;
                                }
                else if(PORTD==0xFF)stige=0;                    //Snu teller retning
                        }
                if(stige==0 && halv==0){                        //Valgt maks til min til maks dempning syklus

                if(PORTD>0x9F)PORTD--;                          //Øker dempning 1/2 om større enn 0x9F
                if(PORTD<=0x9F && PORTD>=0x81){                 //Øker dempning annenhver gang om mondre enn 0x9F
                if(helstep==1){                                 //Da LM1972 øker med ett helt steg fra 48dB tom. 100dB
                        PORTD--;
                        helstep=0;
```

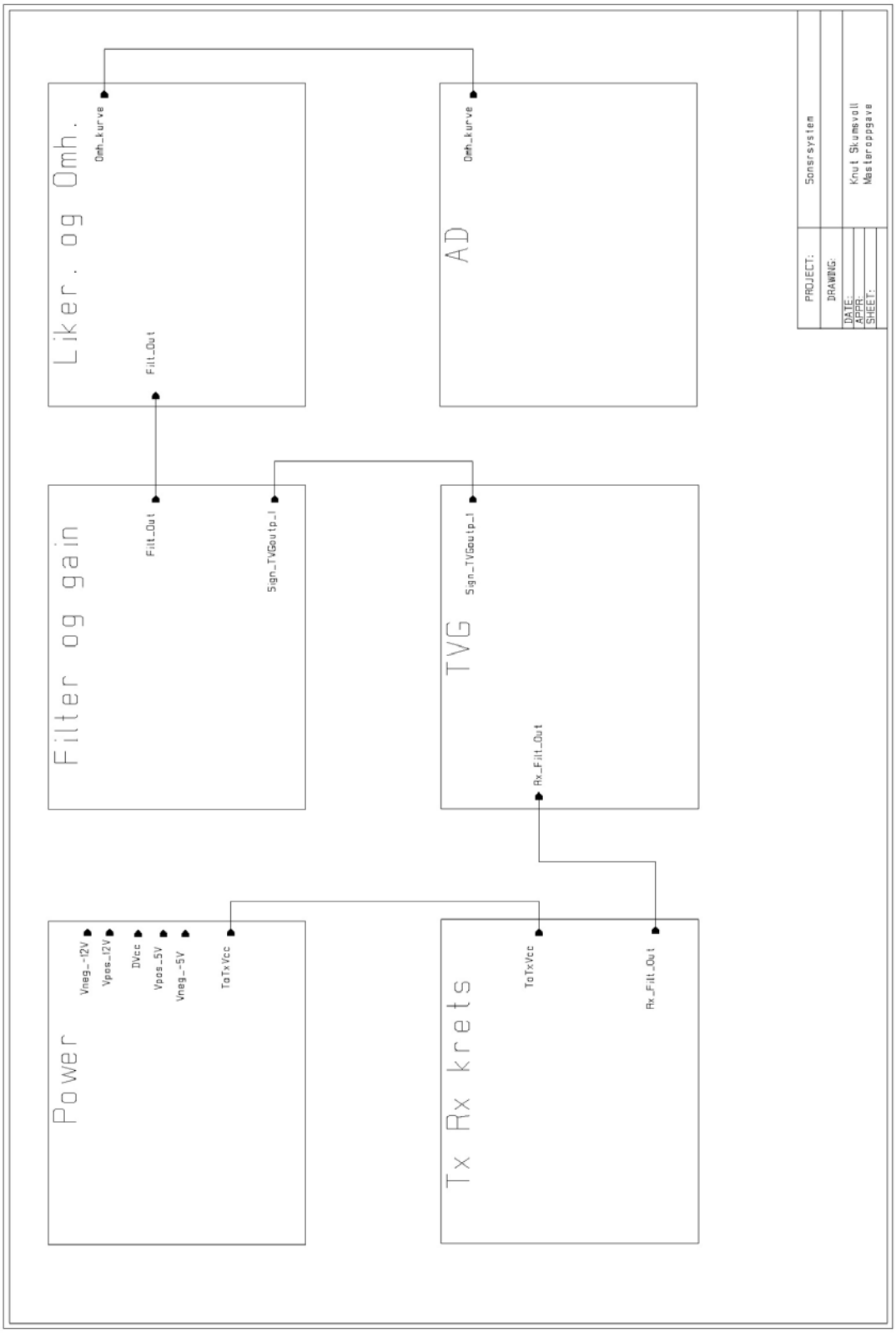
```
                                        }
                                        else helstep=1;
                                        }
                                        else if(PORTD==0x80)stige=1;                    //Vender tellesyklus og øker med 1
                                        }
                                        if(stige==0 && halv==1){                        //Valgt maks til min dempning syklus
                                                PORTD=0x80;                             //Setter maksimal dempning
                                                stige=1;                                //Start minimering av dempning
                                                PORTC &= ~(1<<0);                       //Trigger resettes, PORTC0
                                        }
                                }

                                if(x==1){                                               //Om dataverdi er endret
                                dataverdi= (~PORTD);                                    //Dataverdi har verdi av PORTB (LED)
                                send_Spi(dataverdi, adresse);
                                x=0;
                                }
        }

ISR (TIMER1_COMPA_vect){                                                                //Interrupt ca hvert sekund med 20 perioder a 70KHz (BURST)

        for(int q=0; q < 40; q++){
                _delay_us(12.8);
                PORTC ^= 0x02;                                                          //PORTC1 XOR (Togler)
                }
                PORTC ^=0x00;
                }


static void interrupt_init(void){                                                       //Metode initsierer interrupt,synlig i main
        TCCR0 |= (1<<WGM01)|(1<<COM01);                                                 //CTC(ClearеOnComp.Match)enabler,ClearUpSetDown
        TCCR0 |= (1<<CS01)|(1<<CS00);                                                   //CS00 satt,- CLK/64
        OCR0 = 0x8D;                                                                    //OutputCompareRegister teller 35 pulser.
                                                                                        //8M/256=31,25kHz*1,114ms=35 pulser.
                                                                                        //448,5us mellom hvert demper stepp,->
                                                                                        // 448,5us*31,25kHz=14,016 pulser
        TCCR1B |= (1<<WGM12)|(1<<CS12)|(1<<CS10);                                       //CTC, CS10 og CS12 = fclk/1024
        OCR1AH = 0x07;
        OCR1AL = 0xA1;                                                                  // 0x07A1 = 1953. 8MHz/1024 ca= 7812Hz
                                                                                        //7812Hz :4=1953 (hvert 0,15sek)
        TIMSK |= (1<<OCIE0)|(1<<OCIE1A);                                                //TimerCounterCompMatch Interrupt enabler Tim 0 og timer 1A
        }

        static void init_port_pin(void){                                                //Initsierer PIN og PORT,synlig i main
        DDRD = 0xFF;                                                                    //Port B settes til utganger
        PORTD = 0xFF;                                                                   //LED startes avskrud
        DDRA = 0x00;                                                                    //PINA settes til inganger
        DDRB |= (1<<DDB5)|(1<<DDB7)|(1<<DDB4);                                          //PORTB 4(SS), 5(MOSI) og 7(SCK) settes til ut-porter
        PORTB |= (1<<4);                                                                //PORTB4 settes høy (notSS)
        DDRC |= (1<<DDC0)|(1<<DDC1);                                                    //PORTC0 settes til ut port (trigger til oscilloskop)
                                                                                        //PORTC1 settes til ut port (Burst)
        }

static void init_Spi(void){
        SPCR |=(1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0);                                  //SPIenable,MASTERselect, (og MSB first)
        }

/*****************************************************/
static void test(void){
        PORTD=PORTD;
        }

void main(void){

        interrupt_init();                                                               //Kaller metode for initsiering av interrupt
        init_port_pin();                                                                //Kaller metode for initsiering av portB
        init_Spi();                                                                     //Kaller metode for initsiering av SPI

        sei();                                                                          //Global interrupt enable

        while(1){
        if((PINA==0x7F)&&(autom!=1)){                                                   //Om SW7 er trykket (minke dempning)
```

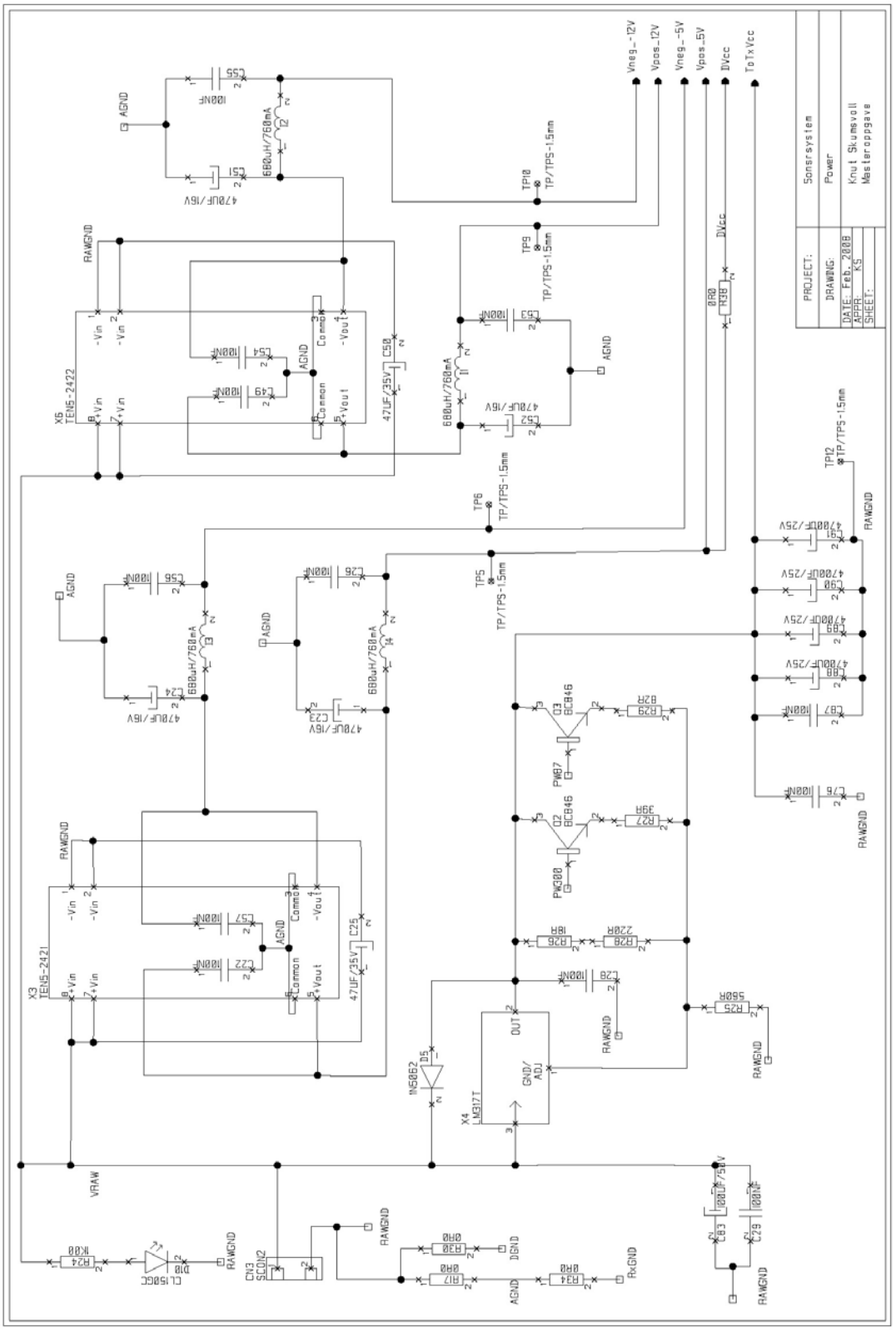
```
                while(PINA==0x7F){}                     //ikke endre før bryter er sluppet
                if(PORTD <= 0xFE){                      //Max økning til 0x7F,verdi over dette er også MUTE
                        PORTD++;                        //PORTB øker i verdi
                        okeverdi=1;                     //Indikerer justere satt verdi
                        }
                }
        if((PINA==0xBF)&&(autom!=1)){                   //Om SW6 er trykket (øke dempning)
                while(PINA==0xBF){}                     //ikke endre før bryter er sluppet
                if(PORTD >= 0x81){                      //Minimale verdi 0x00
                        PORTD--;                        //PORT minker i verdi
                        okeverdi=1;                     //Indikerer justere satt verdi
                        }
                }
        if(PINA==0xFD){                                 //Om SW1 er trykket (stopp auto inkrementer/dekrementer)
                while(PINA==0xFD){}                     //ikke endre før bryter er sluppet
                autom=0;                                //Disabler automatisk
                halv=0;                                 //Resette atu. parameter
                test();
                }
        if(PINA==0xFE){                                 //OM SW0 er trykket (auto inkrementer/dekrementer)
                while(PINA==0xFE){}                     //ikke endre før bryter er sluppet
                autom=1;                                //Enabler automatisk teller
                stige=1;                                //Teller variabel
                halv=0;                                 //Teller variabel,- teller opp og ned
                }
        if(PINA==0xFB){                                 //OM SW2 er trykket (auto inkrementer/dekrementer)
                while(PINA==0xFB){}                     //ikke endre før bryter er sluppet
                autom=1;                                //Enabler automatisk teller
                stige=1;                                //Teller variabel
                halv=1;                                 //Teller halv syklus,-> max dempning til min dempning
                }
        if((PINA==0xDF)&&(autom!=1)){                   //SW5, resette dempning til ingen dempning
                while(PINA==0xFD){}                     //ikke endre før bryter er sluppet
                PORTD=0xFF;                             //Setter PORD, LED av, ingen dempning
                dataverdi= (~PORTD);                    //Dataverdi har verdi av notPORTB (LED)
                send_Spi(dataverdi, adresse);           //Kaller metode for å sette LM1972
                }
        if(PINA==0xF7){                                 //Om SW3 er trykket (skifte adresse til kanal 1)
                while(PINA==0xF7){}                     //ikke endre før bryter er sluppet
                adresse=0x01;
                }
        if(PINA==0xEF){                                 //Om SW4 er trykket (skifte adresse til kanal 0)
        while(PINA==0xEF){}                             //ikke endre før bryter er sluppet
                adresse=0x00;
                }
        }
}
```

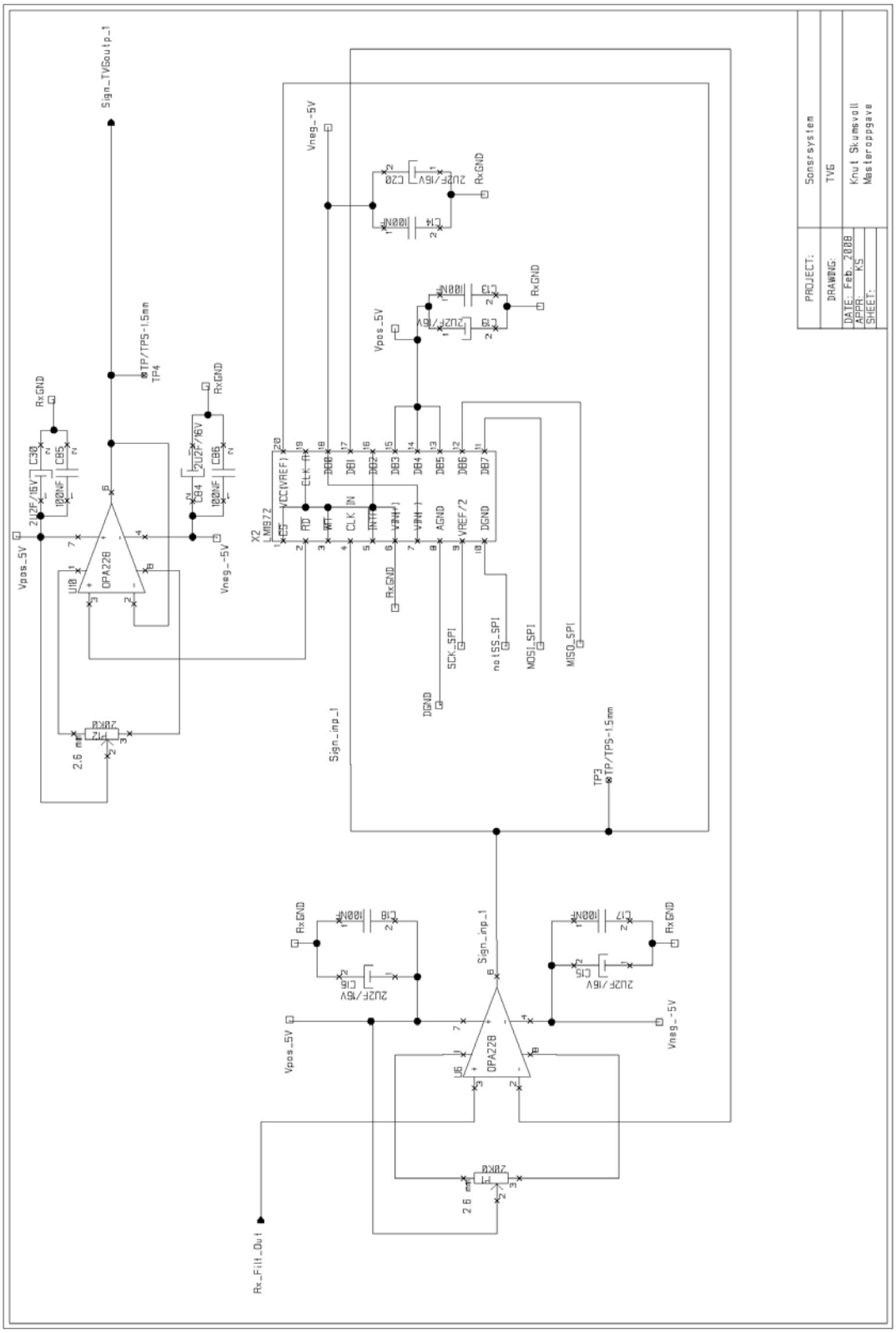
Vedlegg 8 PCB og skjemategning av hovedkretskort

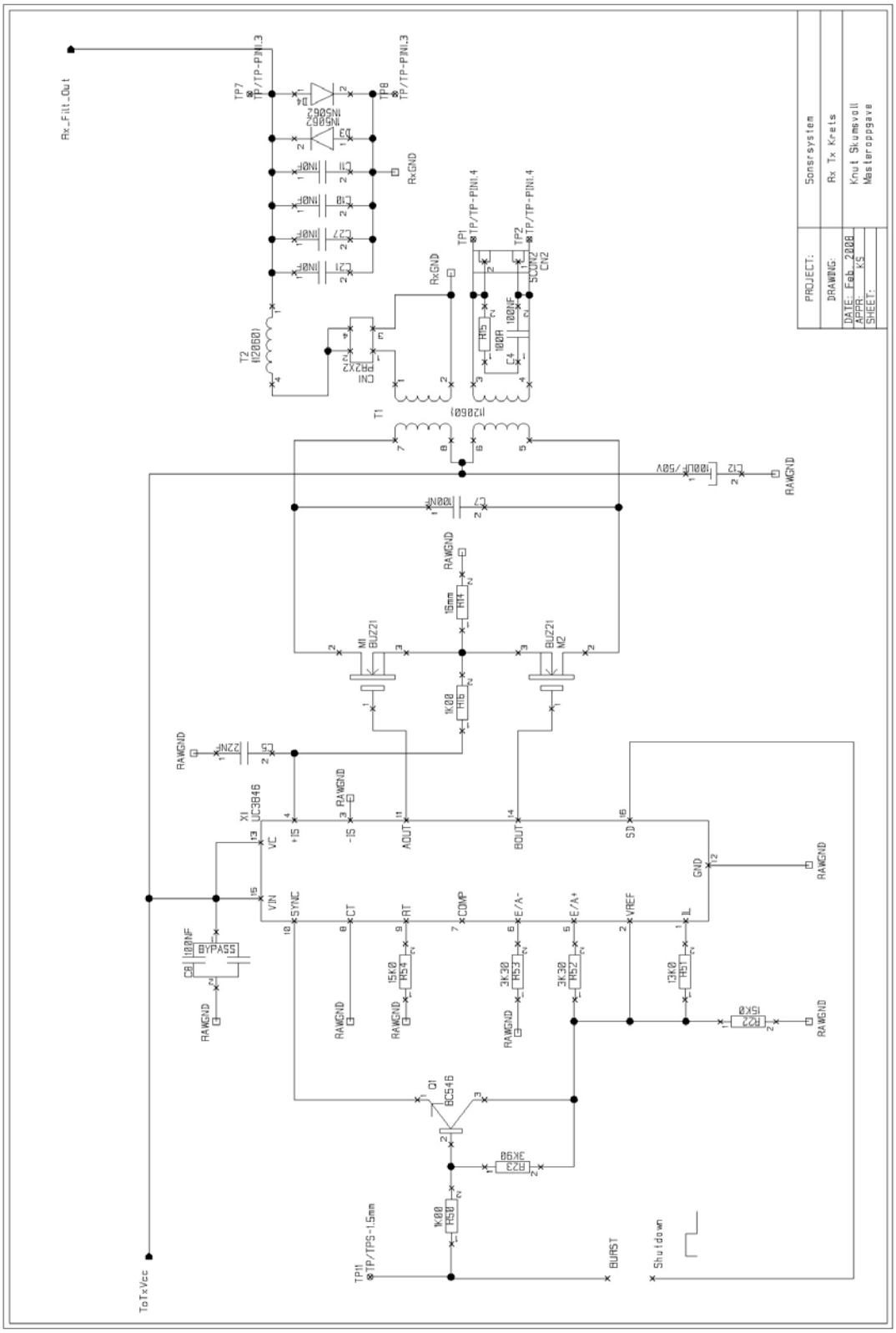


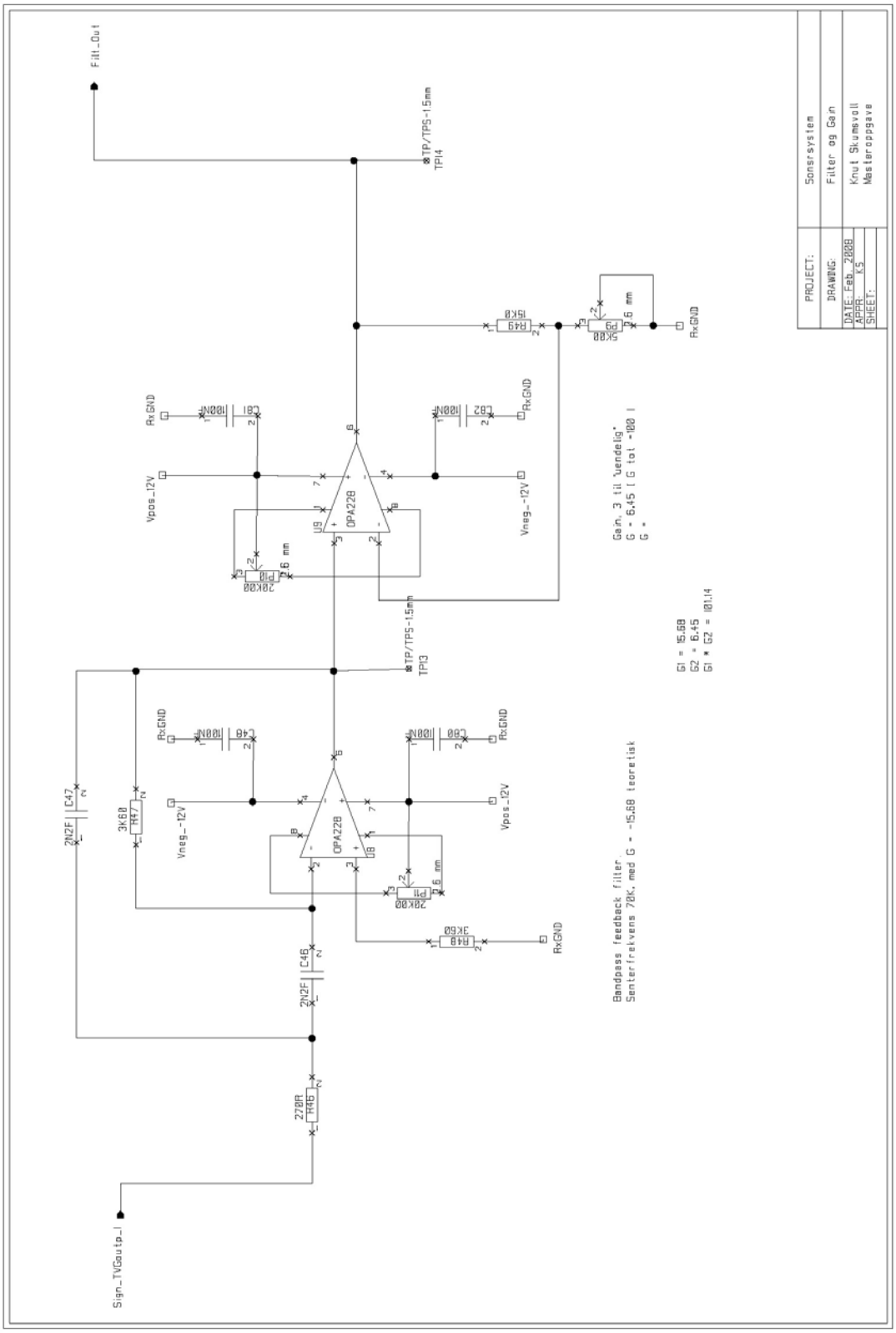

PROJECT:
DRAWING:
Ethernet
DATE: Feb. 2008
APPR: KS
SHEET:
Knut Skumsvoll
Masteroppgave
PULSEJACK

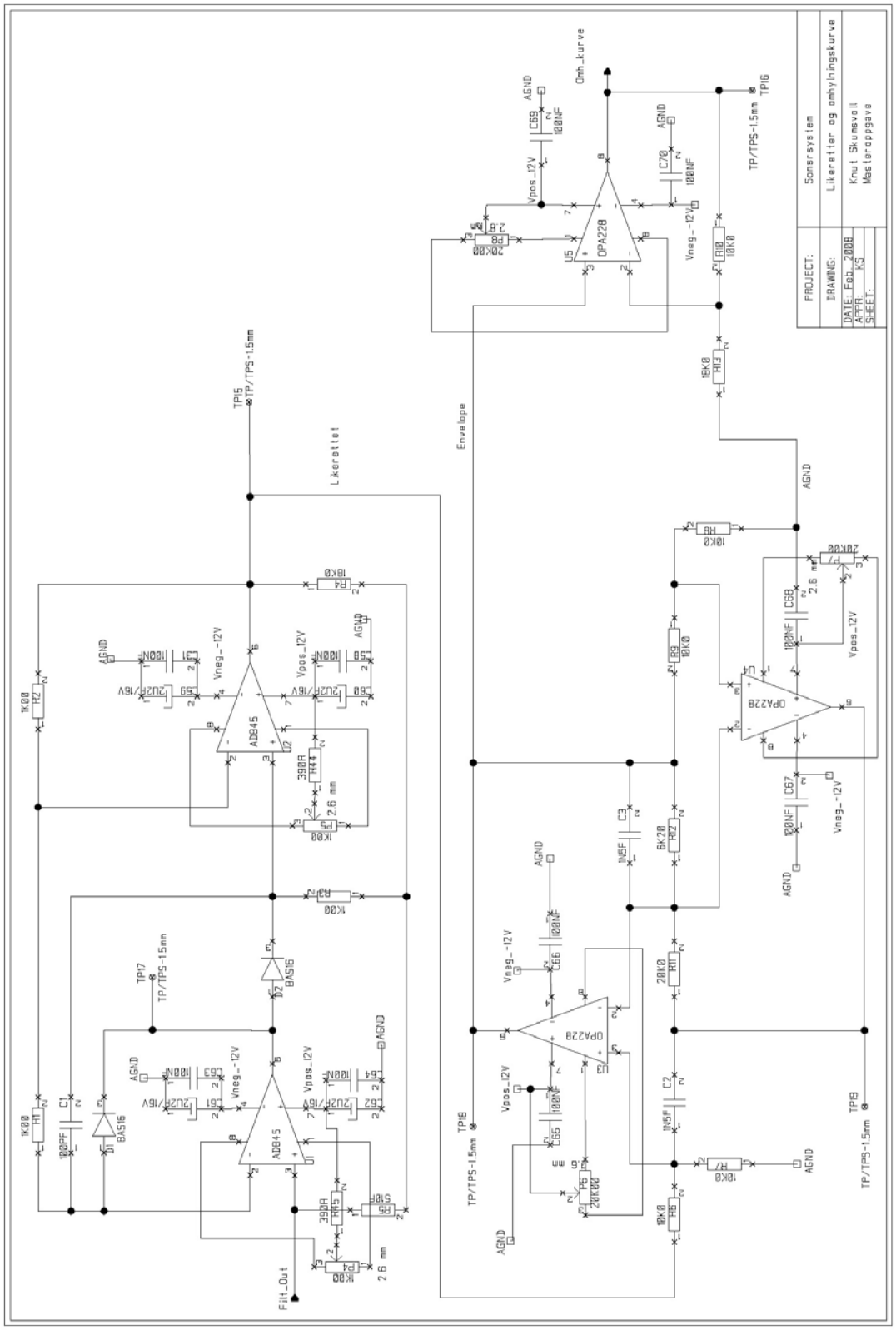

Liker. og Omh.
Omh_kurve
Filt_Out
AD
Omh_kurve
Filter og gain
Filt_Out
Sign_TVGoutp_I
TVG
Sign_TVGoutp_I
Rx_Filt_Out
Power
Vneg_-12V
Vpos_12V
DVcc
Vpos_5V
Vneg_-5V
ToTxVcc
Tx Rx krets
ToTxVcc
Rx_Filt_Out
PROJECT:
Sonarsystem
DRAWING:
DATE:
APPR:
SHEET:
Knut Skumsvoll
Masteroppgave

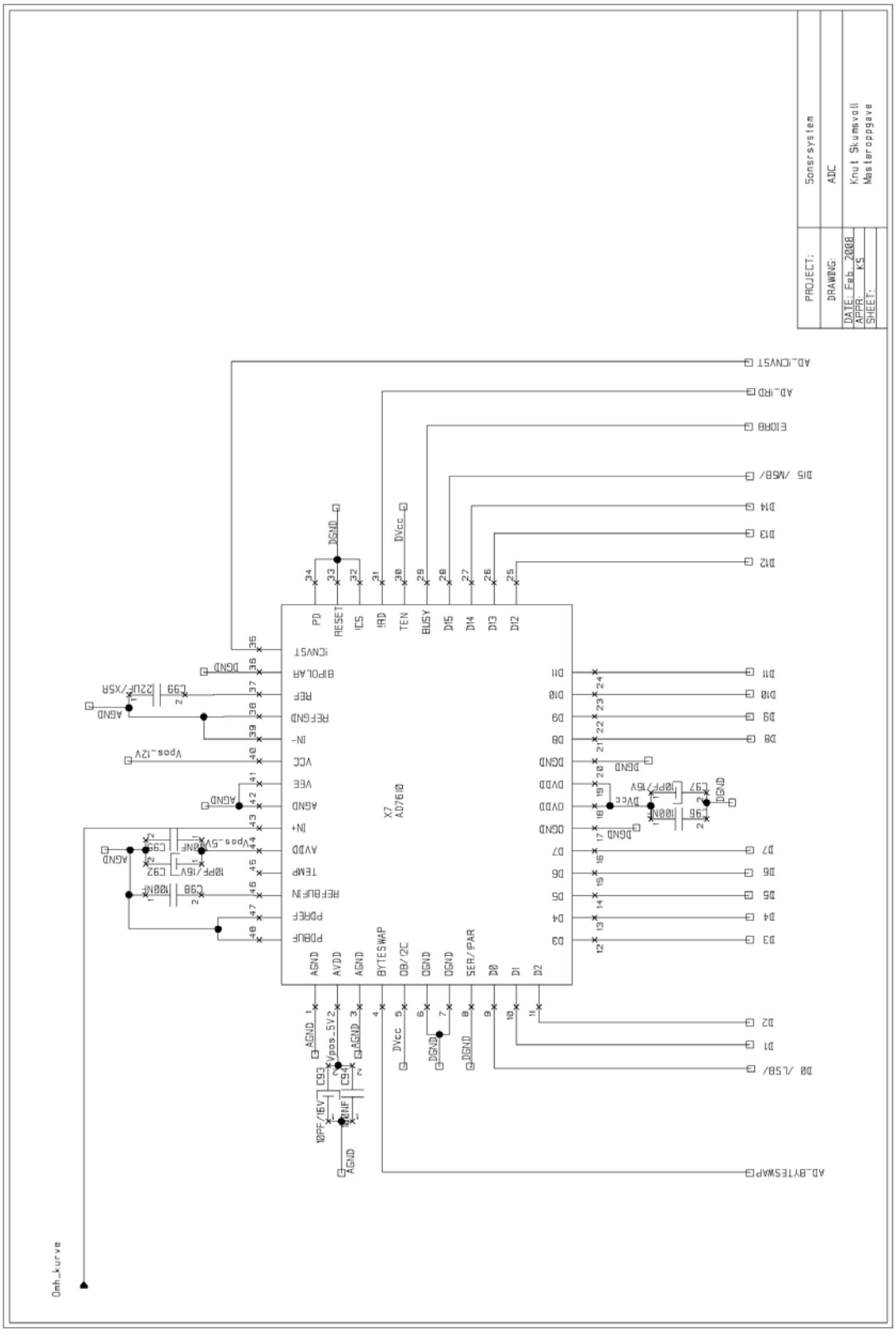

Vneg_-12V
Vpos_12V
Vneg_-5V
Vpos_5V
DVcc
ToTxVcc
Sonsrsystem
Power
PROJECT:
DRAWING:
DATE: Feb. 2008
APPR: KS
SHEET:
Knut Skumsvoll
Mesteroppgave
AGND
RAWGND
DGND
RxGND
VRAW
TP/TPS-1.5mm
680uH/760mA
47UF/16V
47UF/35V
100NF
470UF/25V
LM317T
1N5062
CLI50GC
SCON2
TEN5-2422
TEN5-2421

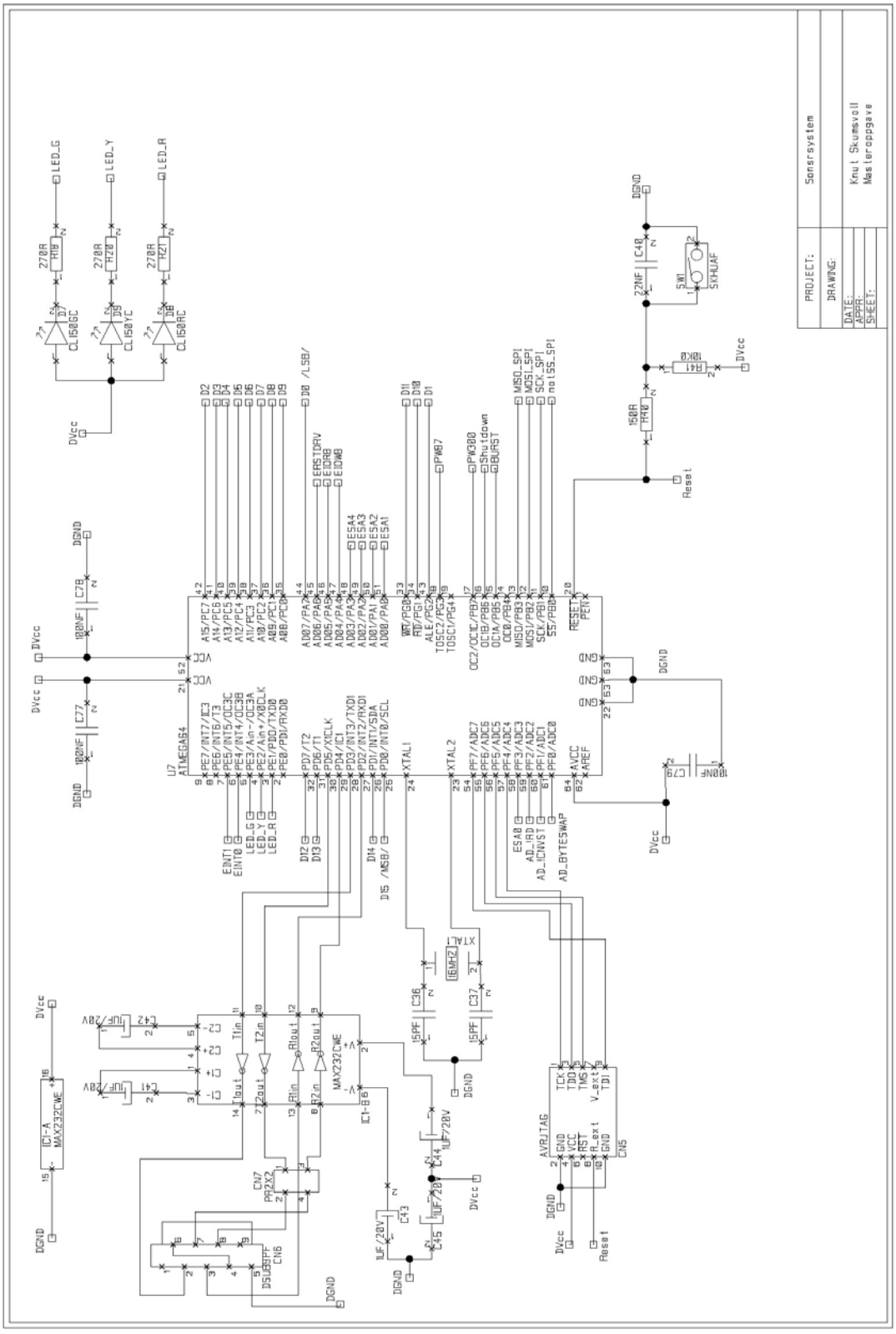

PROJECT: Sonarsystem
DRAWING: TVG
DATE: Feb. 2008
APPR: KS
SHEET:
Knut Skumsvoll
Mesteroppgave
Sign_TVGoutp_1
TP4
TP/TPS-15mm
Vneg_-5V
Vpos_5V
RxGND
OPA228
MISO_SPI
MOSI_SPI
SCK_SPI
DGND
Sign_inp_1
TP3
Rx_Filt_Out

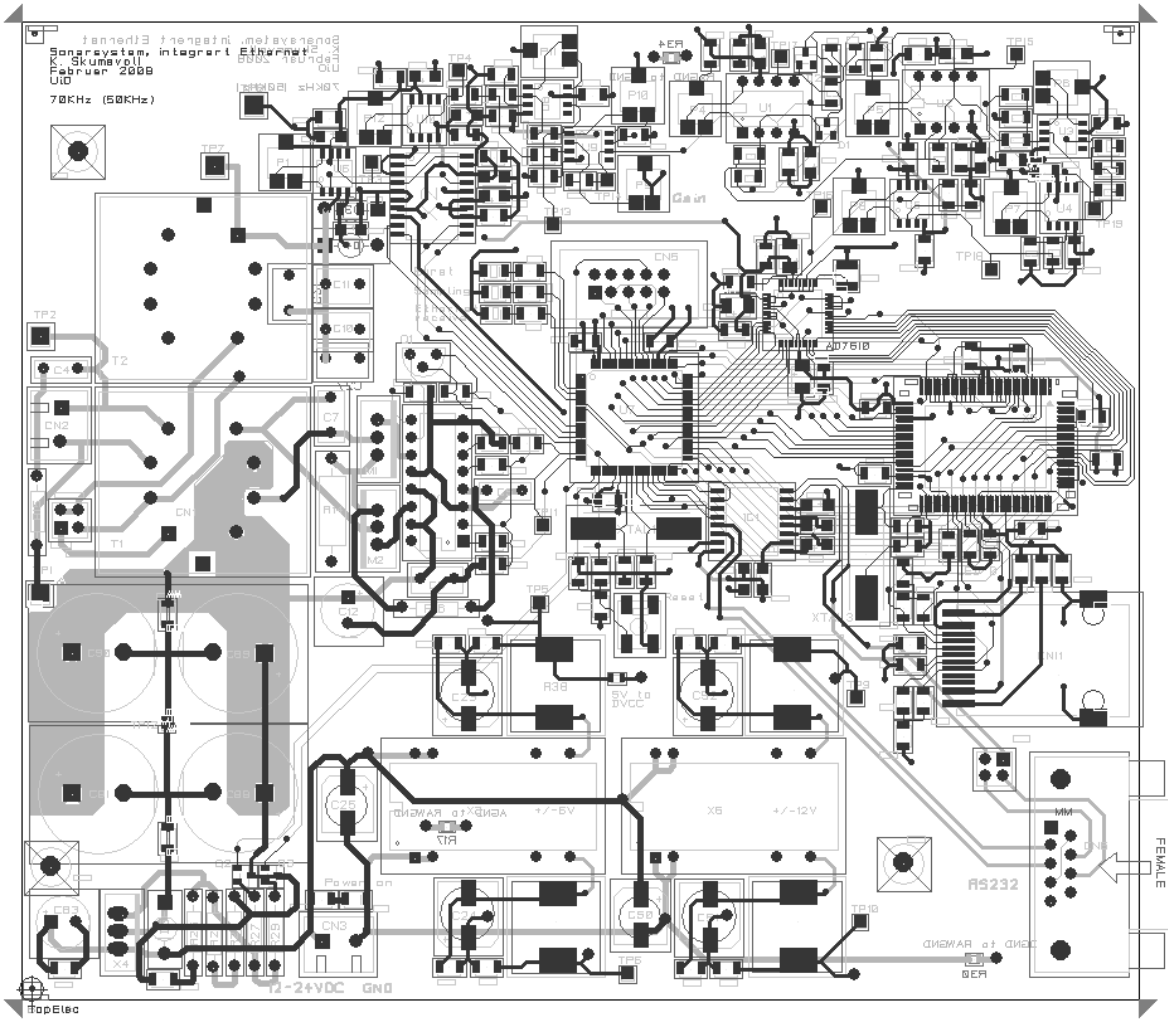

PROJECT: Sonsrysystem
DRAWING: Rx Tx Kretс
DATE: Feb. 2008
APPR: KS
SHEET:
Knut Skumsvoll
Mesteroppgave
Rx_Filt_Out
TP7
TP8
TP/TP-PIN1.3
RxGND
TP1
TP2
TP/TP-PIN1.4
SCON2
CN2
T2
M2060
T1
RAWGND
100UF/50V
C12
100NF
C7
M1
M2
BUZ21
15mm
R14
1K00
R16
22NF
C5
X1
UC3846
VC
+IS
-IS
ADJ
BOUT
SD
GND
VIN
SYNC
CT
RT
COMP
E/A-
E/A+
VREF
IL
100NF
BYPASS
C8
15K0
R54
3K30
R53
3K30
R52
13K0
R51
15K0
R22
Q1
BC546
3K90
R23
1K00
R50
TP11
TP/TPS-1.5mm
BURST
Shuddown
ToTxVcc


Filt_Out
TP/TPS-1.5mm
TP14
PROJECT:
Sonarsystem
DRAWING:
Filter og Gain
DATE: Feb 2008
APPR: KS
SHEET:
Knut Skumsvoll
Mesteroppgave
R49
15K0
P9
5K00
2.6 mm
RxGND
Vpos_12V
Vneg_-12V
U9
OPA228
C81
100N
C82
P10
20K00
TP/TPS-1.5mm
TP13
C48
C80
C47
2N2F
3K60
R47
U8
P11
R48
3K60
C46
270R
R46
Sign_TVGoutp_
Gain 3 til 'uendelig'
G = 6.45 i G tot =100 i
G =
G1 = 15.68
G2 = 6.45
G1 * G2 = 101.14
Bandpass feedback filter
Senterfrekvens 70K, med G = -15.68 teoretisk


PROJECT: Sonsrsystem
DRAWING: Likeretter og omhylningskurve
DATE: Feb 2008
APPR: KS
SHEET:
Knut Skumsvoll
Masteroppgave
Likeretter
Envelope
Omh_kurve
Fil_Out
AD845
OPA228
BAS16
AGND
Vpos_12V
Vneg_-12V
TP/TPS-1.5mm


PROJECT: Sonarsystem
DRAWING: ADC
DATE: Feb 2008
APPR: KS
SHEET:
Knut Skumsvoll
Mesteroppgave
X7
AD7640
AD_ICNVST
AD_IRD
EIORB
D15 /MSB/
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0 /LSB/
AD_BYTESWAP
AGND
DGND
DVcc
Vpos_12V
Vpos_5V
Omh_kurve

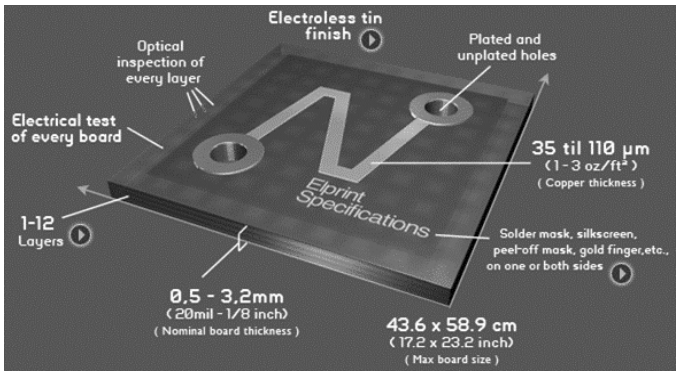

PROJECT: Sonsrsystem
DRAWING:
DATE:
APPR:
SHEET:
Knut Skumsvoll
Mesteroppgave
U7
ATMEGA64
MAX232CWE
IC1-A
IC1-B
CN7
CN6
CN5
AVRJTAG
DSUB9PF
SKHUAF
SW1
LED_G
LED_Y
LED_R
Reset
DVcc
DGND



(bildet av kretskort er her vist uten GND- og VCC-lag)

Vedlegg 9 Transparent av PCB for hovedkretskort

Vedlegg 10 Parameterverdier fra Elprint



Vedlegg 11 Elprint standard 4101

4101: 1.6mm 35/35µm Tg=180

Quantity	Material	Type	Thickness (mm)
1	copper plating		0.025
1	copper foil	12 um	0.012
2	prepreg	2116 370HR	0.230
1	inner layers	1.08mm 35/35 370HR	1.130*
2	prepreg	2116 370HR	0.230
1	copper foil	12 um	0.012
1	copper plating		0.025
		Total thickness:	**1.664**

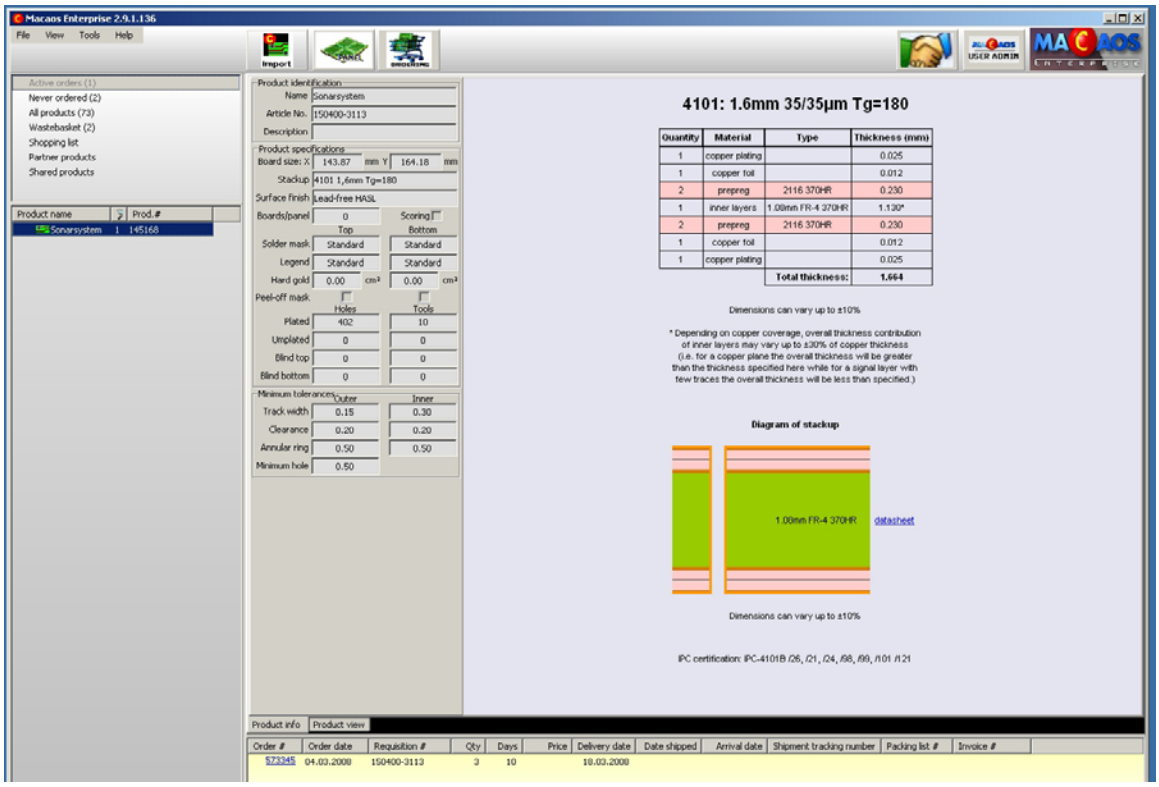
Dimensions can vary up to ±10%

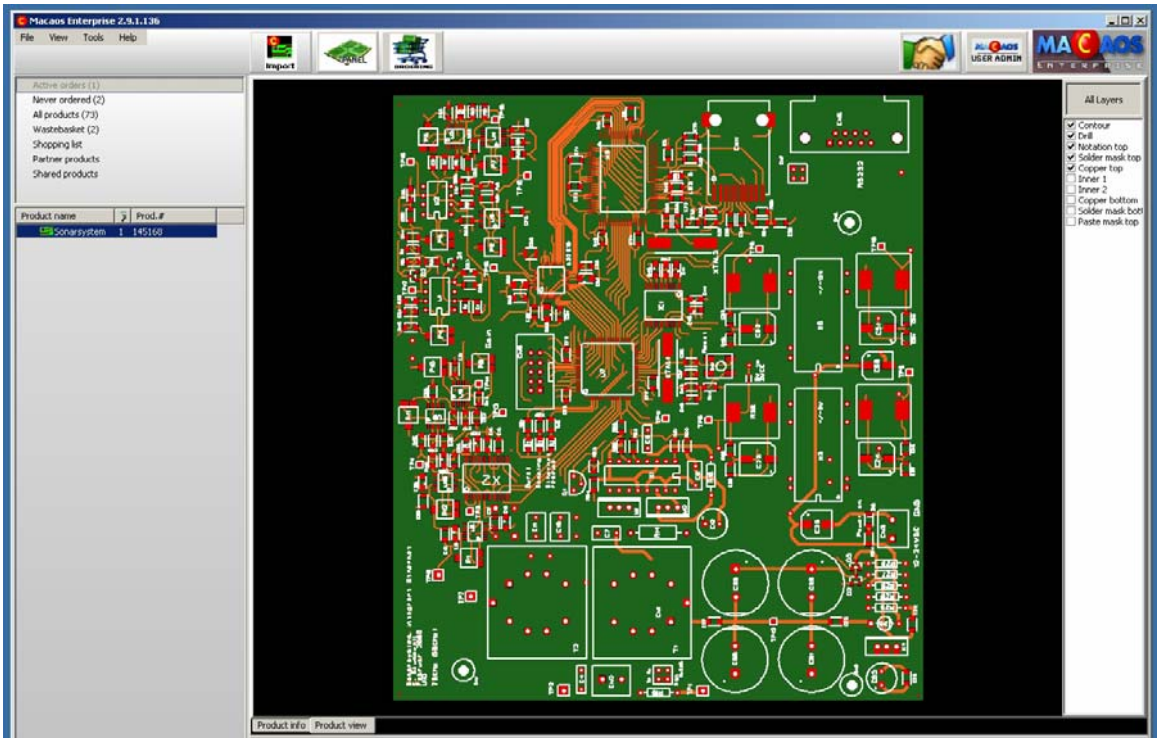
* Depending on copper coverage, overall thickness contribution of inner layers may vary up to ±30% of copper thickness (i.e. for a copper plane the overall thickness will be greater than the thickness specified here while for a signal layer with few traces the overall thickness will be less than specified.)

4101: 1.6mm 35/35µm Tg=180

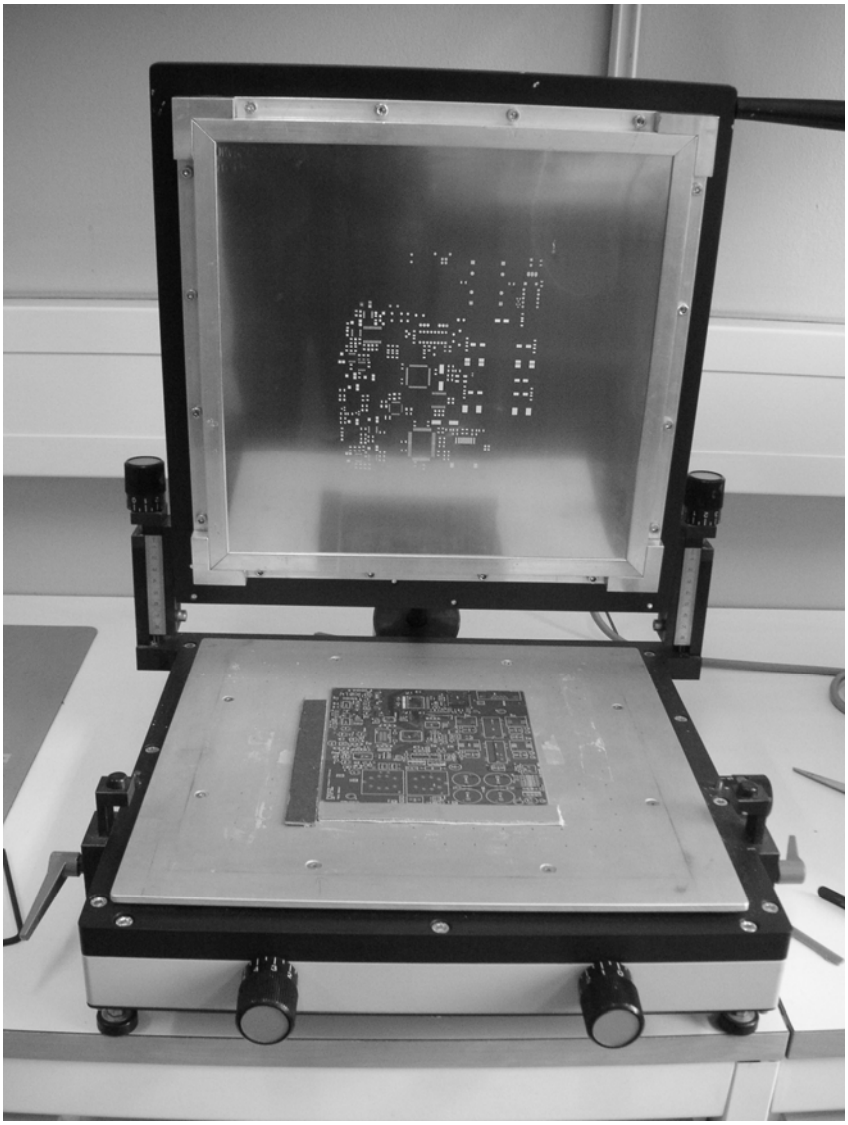
Dimensions can vary up to ±10%

Vedlegg 12 Macaos for bestilling av kretskort





Vedlegg 13 Påføring av loddepasta for påmontering av komponenter



Vedlegg 14 C-programkode for hovedprogram

```
/*****************************************************************************
*****************************************************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll
Masteroppgave, UiO

Mikrokontroller ATmega64
16MHz ekstern krystall
Ethernet kontroller RTL8019AS
20MHz krystall

Main
Programmet styrer sonarsystem med å kontrolleres via Ethernet. Kommunikasjon
sendes som UDP-pakker til eller fra kretskortet.
Systemet gjør bruk av ISP og RS232 for brukerkontroll og programmering
av mikrokontrolleren.
Kode er delt opp i deler for å lette forståelsen i programmflyt
Systemet sampler og sender data etter brukerens ønske. Tilleggsprogram for
kommunikasjon med mikrokontroller er laget i Delphi

*****************************************************************************
*****************************************************************************/
#include "Init.h"
#include "Eth.h"
#include "TVG.h"
#include "ADC.h"
#include "RS232.h"
#include "Skudd.h"

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <inttypes.h>
/*****************************************************************************
Baudrateverdier til UBRR-registeret
*****************************************************************************/
#define b9600   95
#define b19200  47
#define b38400  23
#define b57600  15
#define b115200  7

uint8_t dataverdi=0x00;
unsigned char adresse=0x00;
int SPIoppdatering=0;
int ADCsampling=0;
int ADCteller=0;
uint8_t variabel=0x00;
uint16_t variabel2=0x5555;
uint8_t inkPakke =0;
uint16_t avlest1=0,avlest2=0;
int teller,int4Teller;

void startInterrupt(void);

/******* Interruptrutiner *******/
/*ISR (INT4_vect){
        int4Teller=0x01;
        //Oppdaterer teller
        skrivSkjerm("\r\nInterrupt PINE.4 er aktivert !");
        }
ISR (INT5_vect){
        skrivSkjerm("\r\nInterrupt PINE.5 er aktivert !");
        }
*/
ISR (TIMER1_COMPA_vect){                        //Interrupt for sampling med ADC (interrupt 25kHz)
        kjorSample++;                           //Oppdaterer for sample-rutine i Eth.c
        PORTE&=~(1<<0);
```

```
        }

ISR (TIMER3_COMPA_vect){                                    //Interrupt for oppdatering TVG (interrupt 887Hz)
        dataverdi++;
        adresse=0x00;
        send_Spi(dataverdi,adresse);
        }



void startInterrupt(void){
        dataverdi=0x00;
        }

/****** Metode for manuell tastatur kommandoer *****/
ISR (USART1_RX_vect){
        switch(UDR1){
                case 'R': skrivSkjerm("\n Kaller register utskrift: \r\n\n");RTLregRS232(); break;
                case 'H': hentPakke(); break;
                case 'E': svarPakkeARP(); break;
                case 'S': udpHeaderToSend(); break;
                case 'Z': enableADC();konverter(); dataFraRTL();
                        skrivSkjerm("Samplet verdi er : ");
                        binTilHex(lagTil8((read_busData()),0x01));
                        binTilHex(lagTil8((read_busData()),0x00)); break;

                default: skrivSkjerm("\t default"); break;
                }
        }

/*********************************/
int samplingFrekvens = 50;                                  //Brukervariabel, samplingfrekvensen
int transduserFrekvens = 70;                                //Brukervariabel, transduserfrekvensen
int antPerioder = 20;                                       //Brukervariabel, antal perioder i burst

/***********Main***************/
int main(void){

        cli();                                              //Dissabler global interrupt
        Init();
        DIODG_ON;                                           //Oppstartrutine diode og uC kontroll
        _delay_ms(80);
        DIODY_ON;
        _delay_ms(80);
        DIODR_ON;
        _delay_ms(80);
        DIODG_OFF;
        _delay_ms(80);
        DIODY_OFF;
        _delay_ms(160);
        DIODY_ON;
        _delay_ms(80);
        DIODG_ON;
        _delay_ms(80);
        DIODR_OFF;
        _delay_ms(80);
        DIODY_OFF;
        _delay_ms(160);
        DIODY_ON;
        _delay_ms(80);
        DIODR_ON;
        _delay_ms(80);
        DIODG_OFF;
        _delay_ms(80);
        DIODY_OFF;
        _delay_ms(160);
        DIODY_ON;
        _delay_ms(80);
        DIODG_ON;
        _delay_ms(80);
        DIODR_OFF;
        _delay_ms(80);
```

```
DIODY_OFF;
_delay_ms(80);
DIODG_ON;
_delay_ms(80);
DIODG_OFF;

sei();                                          //Enabler global interrupt
burstModus=0x10;
udpNr=0x00;                                     //Resetter teller som startrutine
while(1){
        skrivTilRTL(CR,0x22);
        while(!(PINE&(1<<4))){
        }

        avlest1 = lesFraRTL(RISR);              //Leser verdi fra ISR (interrupt-) register
        if(avlest1 & OVW){
                overrun();
                skrivTilRTL(RISR,0x10);         //Resetter flagget
        }
        if(avlest1 & PRX){                      //Om pakke uten error er mottat
                DIODG_ON;
                hentPakke();                    //Kjør metode for å behandle pakke
        }
        skrivTilRTL(CR,0x0020);                 //Rutine for kontroll om samtlige pakker er hentet
        avlest1 = lesFraRTL(BNRY);
        skrivTilRTL(CR,0x0060);
        skrivTilRTL(CR,0x62);
        avlest2 = lesFraRTL(CURR);

        skrivTilRTL(CR,0x22);
        while(avlest1 != avlest2){              //Så lenge det er pakker som ikke er hentet ut av bufferet
                hentPakke();
                skrivTilRTL(CR,0x0020);
                avlest1 = lesFraRTL(BNRY);
                skrivTilRTL(CR,0x0060);
                skrivTilRTL(CR,0x62);
                avlest2 = lesFraRTL(CURR);
                skrivTilRTL(CR,0x22);
        }
                skrivTilRTL(RISR,0x00FF);       //Resetter flagg
        _delay_ms(50);
        DIODG_OFF;
        _delay_ms(50);

/********** Rutine sampling **********/
        kjorSample=0x00;                        //Resetter sampleteller, endres i interrupt
        TIFR   |= (1<<OCF1A);                   //Resetter flagg OCF1A
        ETIFR  |= (1<<OCF3A);                   //Resetter flagg OCF3A
        udpNr=0x00;                             //Teller for antall sampel() rutine kjørt
/********* Kontinuerlig skudd **********/
        while((udpNr<(0x03)) && (burstModus==0x00)){    //Så lenge kontinuerlig modus er valgt
                if(udpNr==0x00){
                        sendBurst();
                        DIODY_ON;
                        dataverdi=0x00;
                }
                udpHeaderToSend();
                udpSample();                    //Data-> datasampels
        }
        DIODR_OFF;
        DIODY_OFF;
/********** Kjør ett skudd **********/
        while((udpNr<(0x03)) && (burstModus==0x03)){    //Variabel for single burst
                if(udpNr==0x00){
                        sendBurst();
                        DIODY_ON;
                        dataverdi=0x00;
                }
                udpHeaderToSend();
                udpSample();                    //Data-> datasampels
        }
        if((burstModus==0x03) && (udpNr==0x03)){
```

```
                    burstModus=0X0A;                    //Stans burst om kjørt singel burst
                    DIODR_OFF;
                    DIODY_OFF;
                }
            }
}
```

```
/*************************************************************************
**********************************' .h '*********************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

Init.h
...
*************************************************************************
*************************************************************************/
/*************************************************************************
Definisjoner for kontroll av lysdioder
*************************************************************************/
#include <inttypes.h>

#define DIODG_ON PORTE &= ~(1<<3);
#define DIODG_OFF PORTE|= (1<<3);
#define DIODY_ON PORTE &= ~(1<<2);
#define DIODY_OFF PORTE|= (1<<2);
#define DIODR_ON PORTE &= ~(1<<1);
#define DIODR_OFF PORTE|= (1<<1);

void Init(void);
/*************************************************************************/




/*************************************************************************
*************************************************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

Init
Her initieres samtlige elementer på kretskortet. Hoved komponenter er her
uC og Ethernet kontroller RTL8019AS
*************************************************************************
*************************************************************************/

#include "Eth.h"
#include "ADC.h"
#include "Skudd.h"
#include "TVG.h"
#include "RS232.h"

#include <inttypes.h>
#include <avr/io.h>
#include <util/delay.h>

void InitIO(void){
        DDRF |= (1<<1)|(1<<2);                          //PORTF.1 til utport (ADC-!CNVST),PORTF.2 !RD utport
        PORTF |= (1<<1)|(1<<2);                         //Setter !CNVST (aktiv lav trigger konvertering ADC),setter !RD
        PORTF&=~(1<<2);                                 //Enabler ADC !RD, aktiv lav. Senere disablet under InitADC
        DDRE |= (1<<1)|(1<<2)|(1<<3);                   //Setter PORTE.1-2-3 til utganger (LED`s)
        DDRE&=~(1<<4)&~(1<<5);                          //PINE.4.5 til inganger (intterupt pinner)
        PORTE |= (1<<1)|(1<<2)|(1<<3);                  //LED ønskes slukket ved initsieringsrutine
        DDRE&=~(1<<4)&~(1<<5);                          //Inganger PORTE4,5 (Int0 og 1 fra RTL)
        DDRG |= (1<<3);                                 //Setter PORTG.3 til utgang (sendereffekt)
        DDRB |= (1<<6)|(1<<7)|(1<<5);                   //Setter PB6 (shutdown),PB7 utgang (sendereffekt),PB5
burstsignal
        PORTB|=(1<<6);                                  //Setter "shutdown" for disable senderkrets
        PORTB &=~(1<<7)&~(1<<5);                        //Resetter effektpin 300W, og burstfrekvens
        /***** Ethernetkontroller Init *****/
        DDRA=0xFF;                                      //PORTA0,1,2,3,4,5,6,7 til utganger
```

```
	PORTA|=(1<<4)|(1<<5);                          //PORTA4 (IOWB) og PORTA5 (IORB) settes høy
	PORTA&=~(1<<6);                                //Resetter PORTA6 (RSTDRV til RTL)
	DDRC=0xFF;                                     //PORTC0,1,2,3,4,5,6,.7 til utganger
	DDRD|=(1<<0)|(1<<1)|(1<<6)|(1<<7);             //PORTD0,1,6,7 til utganger
	PORTD|=(1<<0);
	DDRF|=(1<<3);                                  //PORTF3 til utgang
	DDRG|=(1<<0)|(1<<1)|(1<<2);                    //PORTD0,1,6 til utganger
	/************************************/
	}

void InitSPI(void){
	gain=0x00;                                     //Variabel "gain" settes til null, fra TVG.c
	SPCR |=(1<<SPE)|(1<<MSTR);                     //SPI enable,master select,first MSB,f(ocr)/4->SPR0-1 = 0,0
	send_Spi(0x00,0x00);                           //TVG adjust
	send_Spi(gain,0x01);                           //Signal input to TVG gain
	}

void InitUSART(void){                              //RS232 initsiering
	UCSR1B = (1<<RXCIE1)|(1<<RXEN1)|(1<<TXEN1);
	//RxCompleteInterruptEnable,ReceiverEnable,TransmitEnable
	UCSR1C = (1<<UCSZ11)|(1<<UCSZ10);              //Asynkron,8-bit karakterstørrelse,1-stopbit
	UBRR1L = 103;                                  //16MHz ekstern krystall
	}

void InitInterrupt(void){
	/******* ADC sampling, timer1 ******/
	TCCR1B |= (1<<WGM12)|(1<<CS10);                //CTC, OCR1A = TOP, No prescaling
	OCR1AH = 0x01;                                 //0x0320 -> 800 Dec.x = (50uHz/16Mhz) = 800,
	                                               //800/2 = 400 0x0190
	OCR1AL = 0x90;                                 //2*400perioder à 1/16MHz->interrupt pr50kHz
	                                               //(sampl.frekvens = 50kHz)
	/************************************/
	/********* TVG oppdatering *********/
	TCCR3B |= (1<<WGM32)|(1<<CS30);                //CTC mode, ikke prescaler
	OCR3AH = 0x46;                                 //teller 18038 perioder a 16MHz.Dette gir SPI-oppdatering
	OCR3AL = 0x76;                                 //ca 887Hz.18038(Dec) -> 0x4676(Hex)
	/************************************/
	//TIMSK |= (1<<OCIE1A);                        //TimerCounterComp.Match Interrupt Timer1A enabler
	//ETIMSK |= (1<<OCIE3A);                       //TimerCounterComp.Match Interrupt Timer3A enabler
	/************************************/
	/**********Int PINE.4.5 ***********/
	//EIMSK|=(1<<INT4)|(1<<INT5);                  //Aktiverer INT4 og INT5 (PINE.4 og PINE.5)
	//EICRB|=(1<<ISC41)|(1<<ISC40);                //INT4-> reisende flanke genererer intterupt request
	//EICRB|=(1<<ISC51)|(1<<ISC50);                //INT5-> reisende flanke genererer intterupt request
	/************************************/
	ETIMSK &=~(1<<OCIE3B)&~(1<<OCIE1C);
	}

void InitADC(void){
	extern uint8_t udpNr,udpH,kjorSample;          //Importerer eksterne variabler fra Eth.h

	disableADC();                                  //Disabler ADC (disabler bus fra ADC)
	udpNr=0x00;                                    //Resetter teller for UDPsamel-pakke nr.
	udpH=0x00;                                     //udpheader ved udp-sampels ikke lagret i txbuffer
	kjorSample=0x00;                               //Resetter teller for sampling
	}

void InitRTL(void){

	/********** Resetsyklus ***********/
	PORTA|=(1<<4)|(1<<5);                          //Setter IOR og IOW
	PORTA|=(1<<6);                                 //Resetter Ethernetkontroller (aktiv RSTDRV)
	_delay_ms(2);                                  //Krever min. 1.6ms for konfigurering
	PORTA&=~(1<<6);                                //Aktiverer RTL igjen
	_delay_ms(2);                                  //Krever min. 1.6ms for konfigurering
	/************************************/
	/******* Setter RTL register *******/
	set_busAdresse(CR);                            //Adresse CR
	set_busData(0x0021);                           //Set til page0, RD2, STP, Stop RTL, abortremote DMA
	writeStrobe();                                 //Skriver data inn i RTL
	_delay_ms(2);                                  //RTL kreven min 1.6ms for intern konfigurasjon
	set_busAdresse(DCR);                           //Data Configuration Register
```

```
		set_busData(0x0059);				//8B FIFO før sende til lokal DMA,send pacet
								//execute,normal loopback operation,WTS=1
		writeStrobe();					//Skriver data inn i RTL
		set_busAdresse(RBCR0);				//Remote byte cout register0
		set_busData(0x0000);				//Settes til null
		writeStrobe();
		set_busAdresse(RBCR1);				//Remote byte count register1
		set_busData(0x0000);				//settes til null
		writeStrobe();
		set_busAdresse(RCR);				//ReceiverConfigurationRegister
		set_busData(0x0004);				//AB,PRO.Aksepter broadc,dest.adresse må stemme med
PAR0-5
		writeStrobe();
		set_busAdresse(TPSR);				//Transmit Page Start Register
		set_busData(0x0040);				//16-bit modus-> 0x80-8Kb=0x60
		writeStrobe();
		set_busAdresse(TCR);				//Transmit Configuration Register
		set_busData(0x0002);				//Internal loopback,CRC Checker
		writeStrobe();
		set_busAdresse(PSTART);				//Definert i Eth.h, PSTART=0x01
		set_busData(0x0040);				//16KByte tilgj,da 16-bit modus
		writeStrobe();
		set_busAdresse(BNRY);				//Adresse til der motatt data skal leses fra
		set_busData(0x0046);				//Start ved receive 0x46
		writeStrobe();
		set_busAdresse(PSTOP);				//PSTOP definert som 0x80 fra datablad ved 16-bit modus
		set_busData(0x0080);				//Settes til 0x0080 etter datablad
		writeStrobe();
		set_busAdresse(CR);				//Command register
		set_busData(0x0061);				//Setter til p1 (side 1), STP->stop command
		writeStrobe();
		set_busAdresse(CURR);				//Peker til sideadresse til første motaksbuffer ved inkommende
pakker
		set_busData(0x0046);				//(=0x047)Motta 1518Byte,->0x05EE->0x0600(hele256Byte)
								//0x4000+0x0600=0x46
		writeStrobe();
		for(int i=0;i<6;i++){				//Løkke for å legge inn MAC: "KNUTSS"
			set_busAdresse(PAR0+i);
			set_busData(minMAC[i]);
			writeStrobe();
			}
		set_busAdresse(CR);				//Command Register
		set_busData(0x0022);				//Igangsetter Ethernetkontroller, RemoteWrite
		writeStrobe();
		set_busAdresse(RISR);				//Interrupt Status Register
		set_busData(0x00FF);				//Sette samtlige for å resette de
		writeStrobe();
		set_busAdresse(IMR);				//Interrupt Mask Register
		set_busData(0x001B);				//Enabler interrupt ved receivedNoError, transmittNoError,
								//transmittAbortExessiveCollisions, OverFlow receive buffer
		writeStrobe();
		set_busAdresse(TCR);				//Transmit Configuration Register
		set_busData(0x0000);				//Enable CRC kontroll og generator,ikke loopback
								,AutoTransmitDisable, //CollisionOffsetDisable
		writeStrobe();
		udpH=0x01;					//Variabel for sende udp. 0x01->lag header innhold.0x00-
>benytt forige
		}

void InitSkudd(void){
		shutdown();					//Metode disabler transduser senderkrets
		}
/*****************************************************************************/
void Init(void){
		InitIO();
		InitSPI();
		InitUSART();
		InitInterrupt();
		InitADC();
		InitRTL();
		InitSkudd();
}
```

```
/*****************************************************************************
*********************************' .h '*************************************
Sonarsystem
2008        vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

Eth.h
...
*****************************************************************************
*****************************************************************************/
#include <avr/io.h>
#include <inttypes.h>

/*** Definisjoner av RTL reg. verdier, Page0 om ikke annet er spesifisert ***/
            #define CR              0x00                    //Command register-
            #define PSTART          0x01                    //Page start - Startpage for receive buffer ring
            #define PAR0             0x01                   //p1 Physical address- Inneholder addresse for
                                                            //mottak av pakker (IP?)
            #define PSTOP           0x02                    //Page Stop- Stop page for receive buffer ring, ikke over 0x80
            #define BNRY            0x03                    //Boundary reg. - Boundary register, for å hindre overskrivinger
                                                            // av receive bufferet
            #define TPSR            0x04                    //Transmit Start Page register
            #define TBCR0           0x05                    //Transmit byte count register0
            #define TBCR1           0x06                    //Transmit byte count register1
            #define RISR            0x07                    //Interupt Status register
            #define CURR            0x07                    //p1 Current page reg- Current page i buffer
            #define RSAR0           0x08                    //Remote Start Address register0
            #define RSAR1           0x09                    //Remote Start Address register1
            #define RBCR0           0x0A                    //Remote Byte Count register0
            #define RBCR1           0x0B                    //Remote Byte Count register1
            #define RCR             0x0C                    //Receive Configuration register
            #define TCR             0x0D                    //Transmit Configuration register
            #define DCR             0x0E                    //Data Configuration register
            #define IMR             0x0F                    //Interrupt Mask register
            #define RDMA            0X10                    //Remote DMA port (dataport for RTL bufferet)
            #define PRX             0x01                    //ISR register definisjon
//*************************************************************************
//*          ARP Layout
//*************************************************************************
            #define ARP_hwtype      0x0100                  //Hardware type (0x0001=10Mb Ethernet)
            #define ARP_protoType   0x0008                  //Protocol type (0x0800=IP)
            #define ARP_hwprlen     0x0406                  //Lengde på hardware (MAC) addresse, lengde på protokoll
(IP)                                                         //addresse(IPv4=4byte lang)
            #define ARP_senderIP    0x0E                    //arp source ip address
            #define ARP_senderMAC   0x0B                    //arp target mac address
            #define ARP_destIP      0x13                    //arp target ip address
//*************************************************************************
//*          IP Header Layout
//*************************************************************************
            #define IP_versLen      0x07                    //IP version and header lengde
            #define IP_senderaddr   0x0D                    //IP address of source
            #define IP_destaddrIn   0x0F                    //IP addess of destination
            #define IP_destaddrUt   0x0D                    //IP adresse til den som pakken skal sendes til
            #define IP_senderMACUt  0x03                    //MAC adressen til den pakken skal sendes til
//*************************************************************************
/*          RTL receive og transmit minneseksjoner */
//*************************************************************************
            #define txStart         0x0040                  //transmittbufferet trenger kun å holde en UDP pakke som er
            //#define rxStart       0x0046                  //1518B, dette settes til 0x4600-0x4000=0x0600=1536B.
            //#define pStop         0x0080                  //Resten av minnet settes til receivebuffer (PSTOPmaks=0x60)
            //#define CURRstart     0x0047

                                                            //Peker til sideadresse til første motaksbuffer ved
                                                            //inkommende pakker
                                                            //Motta 1518Byte,->0x05EE->0x0600(hele256Byte)
                                                             //0x4000+0x0600=0x46

//*************************************************************************
/*          Interrupt Status Register bit */
//*************************************************************************
#define RDC                     0x40                        // 0100000 Remote DMA Completed
```

```
#define OVW                     0x10                    // 0000001 Recieve buffer overflow
#define PRX                     0x01                    // 0000001

#define NOP() asm volatile ("nop");                     //Delay en klokke

void set_busAdresse(uint8_t);
void set_busData(uint16_t);
uint16_t read_busData(void);
void writeStrobe(void);
void dataTilRTL(void);
void dataFraRTL(void);
void hentPakke(void);
void svarPakkeARP(void);
void ARPpakke(void);
void overrun(void);
void udpMottatt(void);
void udpHeaderToSend(void);
void udpSample(void);
void icmp(void);
void skrivTilRTL(uint8_t,uint16_t);
uint16_t lesFraRTL(uint8_t);
uint16_t lagTil16(uint16_t,uint8_t);

/*****************************************************************************
        Definisjon av kretskortets MAC adresse
*****************************************************************************/
extern char minMAC[6];                                  //MAC til kretskortet
/*****************************************************************************/
uint16_t packet[1514];                                  //Benyttes til UDP-pakke

extern uint8_t udpH;                                    //0x01->lagret i send buffer.0x00->ikke lagret i buffer
extern uint8_t udpNr;                                   //Teller.Pakke nr X i UDPsamling.'
extern uint8_t burstModus;                              //Variabel for enkelt eller kontinuerlig burst
extern uint8_t kjorSample;                              //Variabel for sampling.Styres i interrupt i Main
/*****************************************************************************
        Variabler ved ADC konvertering, med interrrupt
*****************************************************************************/
uint8_t kjorSample;
/*****************************************************************************/
/*** feilsøkvariabler***/
uint16_t temptemp,tmptmptmp;
void hentPakke(void);
void IPpakke(void);
uint8_t lagTil8(uint16_t,uint8_t);
uint16_t lesFraRTL(uint8_t);
```

```
/*****************************************************************************
******************************************************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall
Ethernet kontroller RTL8019AS
20MHz krystall

Eth (Ethernet)
Her mottas og sendes pakker til og fra kretskortet. Kontrolleren er satt
i 16-bit modus
******************************************************************************
******************************************************************************/
#include "Init.h"
#include "Eth.h"
#include "RS232.h"
#include "Skudd.h"
#include "ADC.h"
#include "TVG.h"

#include <avr/io.h>
#include <inttypes.h>
#include <util/delay.h>

#define SETIOR PORTA |= (1<<5)
#define CLRIOR PORTA &= ~(1<<5)
#define SETIOW PORTA |= (1<<4)
#define CLRIOW PORTA &= ~(1<<4)

uint8_t udpH;                                   //Deklarerer typevariabel udpHeader
uint8_t udpNr;                                  //Deklarerer typevariabel udp teller ved sampel udp
uint8_t burstModus;                             //Variabel benyttet ii Main, single,kont. eller stans burst fra
transduser
uint8_t kjorSample;                             //Variabel endret i interrupt i Main.Starter konvertering

uint8_t reSend;                                 //Til avsender "pakke ikke mottat, send på nytt"
uint8_t tallar8;                                //Benyttes i Eth.c ved adresse funksjon
uint16_t tallar16;                              //Benyttes i Eth.c ved skrive til RTL funksjoner
uint16_t tallar16ut;                            //Benyttes i Eth.c ved skrive fra RTL funksjoner
uint16_t identitet;                             //Benyttes ved svar kommando etterfulgt av sampel-udp`s
uint16_t identitetRDMA;                         //Benyttes ved svar kommando etterfulgt av sampel-udp`s
uint16_t destinasjonsPort;                      //Variabel for lagring av avsenderport ved
                                                //kommando til koretskort
uint8_t stoppKretskortKommando=0x00;            //Variabel for returUDP rutine ved stopp kommando fra Delphi.

uint8_t paType=0x00;                            //paType benyttet i IPpakke();

uint16_t lengde=0x00;                           //Variabel for  av mottatt pakke
uint16_t pgheader[2];                           //Variabel for "page header"
/*****************************************************************************
          Definisjoner av IP og MAC adresse
******************************************************************************/
uint8_t minIP[4] = {129,240,84,8};              //IPadresse til kretskort*/
char minMAC[6] = {'K','N','U','T','S','S'};     //MAC til kretskortet

uint8_t senderIP[4];                            //IPadresse til sender av pakken
char senderMAC[6];                              //MAC adresse til avsender

uint8_t staticSenderIP[4]={129,240,84,200};     //Min PC til internett
char staticSenderMAC[6]={0x00,0x04,0x75,0xe3,0x1e,0xf1};  //Min PC til internett
/*****************************************************************************/
/***** Adresse og data til RTL *****/

void skrivTilRTL(uint8_t addr, uint16_t data){
          dataTilRTL();
          NOP();
          set_busAdresse(addr);
          NOP();
          set_busData(data);
```

```
		NOP();
		writeStrobe();
		NOP();
		}
/************************************/
/********** Data fra RTL ***********/

uint16_t lesFraRTL(uint8_t Regnavn){

		uint8_t regNavn = Regnavn;
		uint16_t regGet;

		dataFraRTL();
		NOP();
		set_busAdresse(regNavn);
		NOP();
		regGet=read_busData();
		NOP();

		return regGet;
		}
/************************************/
/**** Metode setter data til bus****/

#define SET_BUS_PORT(port,pin,var,bitpos) \
port &= ~(1<<pin); \
if(var&(1<<bitpos)) port |= (1<<pin);

void set_busAdresse(uint8_t adrE){				//Setter adresseverdi til adressebus, 5-bit fra 8-bit value

		tallar8=adrE;

		SET_BUS_PORT(PORTF,3,tallar8,0); //A0
		SET_BUS_PORT(PORTA,0,tallar8,1); //A1
		SET_BUS_PORT(PORTA,1,tallar8,2); //A2
		SET_BUS_PORT(PORTA,2,tallar8,3); //A3
		SET_BUS_PORT(PORTA,3,tallar8,4); //A4
		}

void set_busData(uint16_t dataE){				//Setter dataverdi til adressebus, 16-bit

		tallar16=dataE;

		SET_BUS_PORT(PORTA,7,tallar16,0); //D0
		SET_BUS_PORT(PORTG,2,tallar16,1); //D1
		SET_BUS_PORT(PORTC,7,tallar16,2); //D2
		SET_BUS_PORT(PORTC,6,tallar16,3); //D3
		SET_BUS_PORT(PORTC,5,tallar16,4); //D4
		SET_BUS_PORT(PORTC,4,tallar16,5); //D5
		SET_BUS_PORT(PORTC,3,tallar16,6); //D6
		SET_BUS_PORT(PORTC,2,tallar16,7); //D7
		SET_BUS_PORT(PORTC,1,tallar16,8); //D8
		SET_BUS_PORT(PORTC,0,tallar16,9); //D9
		SET_BUS_PORT(PORTG,1,tallar16,10); //D10
		SET_BUS_PORT(PORTG,0,tallar16,11); //D11
		SET_BUS_PORT(PORTD,7,tallar16,12); //D12
		SET_BUS_PORT(PORTD,6,tallar16,13); //D13
		SET_BUS_PORT(PORTD,1,tallar16,14); //D14
		SET_BUS_PORT(PORTD,0,tallar16,15); //D15
		}

/************************************/
/**** Metode leser data fra bus ****/
#define RED_BUS_PIN(pinn,lesDt,bitpoz) \
lesDt &= ~(1<<bitpoz); \
if(pinn) lesDt |= (1<<bitpoz);

uint16_t read_busData(void){

		tallar16ut=0x0000;
		CLRIOR;
		NOP();
```

```
        RED_BUS_PIN((PINA&(1<<7)),tallar16ut,0);
        RED_BUS_PIN((PING&(1<<2)),tallar16ut,1);
        RED_BUS_PIN((PINC&(1<<7)),tallar16ut,2);
        RED_BUS_PIN((PINC&(1<<6)),tallar16ut,3);
        RED_BUS_PIN((PINC&(1<<5)),tallar16ut,4);
        RED_BUS_PIN((PINC&(1<<4)),tallar16ut,5);
        RED_BUS_PIN((PINC&(1<<3)),tallar16ut,6);
        RED_BUS_PIN((PINC&(1<<2)),tallar16ut,7);
        RED_BUS_PIN((PINC&(1<<1)),tallar16ut,8);
        RED_BUS_PIN((PINC&(1<<0)),tallar16ut,9);
        RED_BUS_PIN((PING&(1<<1)),tallar16ut,10);
        RED_BUS_PIN((PING&(1<<0)),tallar16ut,11);
        RED_BUS_PIN((PIND&(1<<7)),tallar16ut,12);
        RED_BUS_PIN((PIND&(1<<6)),tallar16ut,13);
        RED_BUS_PIN((PIND&(1<<1)),tallar16ut,14);
        RED_BUS_PIN((PIND&(1<<0)),tallar16ut,15);

        SETIOR;
        NOP();
        return tallar16ut;
        }
/************************************/

void writeStrobe(void){                                 //Leser fra uC til RTL
        CLRIOW;
        NOP();
        NOP();
        SETIOW;
        NOP();
        }

void dataTilRTL(void){                                  //Databus settes som utganger
        DDRA|=(1<<7);                                   //PORTA7 settes til utgang
        DDRC=0xFF;                                      //PORTC0,1,2,3,4,5,6,.7 til utganger
        DDRD|=(1<<0)|(1<<1)|(1<<6)|(1<<7);              //PORTD0,1,6,7 til utganger
        DDRG|=(1<<0)|(1<<1)|(1<<2);                     //PORTD0,1,6 til utganger
        }

void dataFraRTL(void){                                  //Databus settes som innganger
        DDRA&=~(1<<7);                                  //PINA7 settes til ingang
        DDRC=0x00;                                      //PORTC0,1,2,3,4,5,6,.7 til innganger
        DDRD&=~(1<<0)&~(1<<1)&~(1<<6)&~(1<<7);          //PORTD0,1,6,7 til innganger
        DDRG&=~(1<<0)&~(1<<1)&~(1<<2);                  //PORTD0,1,6 til innganger
        }
/*****************************************************************************************************************/
/*      Sende og motta til/fra RTL                                                                               */
/*****************************************************************************************************************/

uint16_t lagTil16(uint16_t stor, uint8_t liten){
        return ((stor<<8)+liten);
        }

uint8_t lagTil8(uint16_t variabel,uint8_t offset){      //Metode henter 8 første eller 8 siste bit av 16bit "variabel"
        return ((variabel>>(offset*8))&0xFF);
        }

void hentPakke(void){

        skrivTilRTL(CR,0x001A);                         //CR-> SendPacket kommando

        for(int i=0; i<2; i++){                         //For løkke to ganger
                pgheader[i] = lesFraRTL(RDMA);          //Henter verdi i RemoteDMA (to første->16bit)(leser 2 ganger)
                }

        uint32_t rxlen=pgheader[0x01];
        uint16_t tilPacket=0x00;
        for(int i=0;i<rxlen;++i){
        tilPacket = lesFraRTL(0X10);                    //Leser fra RDMAPORT

                if(i<0x30){
                        packet[i] = tilPacket;
                        }
```

```
                }

        uint16_t temp=0x00;
        while(!(temp & RDC)){
        temp = lesFraRTL(RISR);
                }

        skrivTilRTL(RISR,0x00FF);
        uint16_t pakkeType =0x00;
        pakkeType = packet[0x06];                                       //PakkeType tildeles verdi i pos 0x06

        if(pakkeType==0x0008){                                          //Om pakken er av type IP
                udpH=0x00;                                              //Lag ny udp-header ved udpHeaderToSend
                IPpakke();
                }
        else if(pakkeType==0x0608){                                     //Om pakken er av type ARP
                udpH=0x00;                                              //Lag ny udp-header ved udpHeaderToSend
                ARPpakke();
                }
        else{
                udpH=0x00;                                              //Lag ny udp-header ved udpHeaderToSend
                }
}


void IPpakke(void){

        senderIP[1]=lagTil8(packet[IP_destaddrUt],0x00);                //Henter senders IP fra mottatt pakke
        senderIP[2]=lagTil8(packet[IP_destaddrUt],0x01);
        senderIP[3]=lagTil8(packet[IP_destaddrUt+1],0x00);
        senderIP[4]=lagTil8(packet[IP_destaddrUt+1],0x01);

        senderMAC[0]=lagTil8(packet[IP_senderMACUt],0x00);              //Henter destinasjons-IP fra mottatt pakke
        senderMAC[1]=lagTil8(packet[IP_senderMACUt],0x01);
        senderMAC[2]=lagTil8(packet[IP_senderMACUt+1],0x00);
        senderMAC[3]=lagTil8(packet[IP_senderMACUt+1],0x01);
        senderMAC[4]=lagTil8(packet[IP_senderMACUt+2],0x00);
        senderMAC[5]=lagTil8(packet[IP_senderMACUt+2],0x01);

        uint8_t IP_dest0=lagTil8(packet[IP_destaddrIn],0x00);           //Henter destinasjons-IP fra mottatt pakke
        uint8_t IP_dest1=lagTil8(packet[IP_destaddrIn],0x01);
        uint8_t IP_dest2=lagTil8(packet[IP_destaddrIn+1],0x00);
        uint8_t IP_dest3=lagTil8(packet[IP_destaddrIn+1],0x01);


        if((senderIP[1]==192)&&
                (senderIP[2]==168)&&
                (senderIP[3]==0)&&
                (senderIP[4]==1)){
                }

        if((IP_dest0==minIP[0])&&
        (IP_dest1==minIP[1])&&
        (IP_dest2==minIP[2])&&
        (IP_dest3==minIP[3])){

                uint8_t ekkoReq=0x00;
                ekkoReq=lagTil8(packet[0x11],0x00);
                if(ekkoReq==0x08){                                      //Sjekker om icmp er ekkorequest
                        }
                paType =0x00;
                paType = lagTil8(packet[0x0B],0x01);                    //Henter pakke type fra motatt IP-pakke

                switch(paType){
                        case 0x01:  icmp();break;
                        case 0x11:  udpMottatt();               break;
                        default:    skrivSkjerm("\r\nIP pakke er ikke UDP-pakke.\r\n"); break;
                }
        }
}
```

```
void ARPpakke(void){

        senderIP[1]=lagTil8(packet[ARP_senderIP],0x00);                 //Henter senders IP fra mottatt pakke
        senderIP[2]=lagTil8(packet[ARP_senderIP],0x01);
        senderIP[3]=lagTil8(packet[ARP_senderIP+1],0x00);
        senderIP[4]=lagTil8(packet[ARP_senderIP+1],0x01);

        uint8_t ARP_dest0=lagTil8(packet[ARP_destIP],0x00);             //Henter destinasjons-IP fra mottatt pakke
        uint8_t ARP_dest1=lagTil8(packet[ARP_destIP],0x01);
        uint8_t ARP_dest2=lagTil8(packet[ARP_destIP+1],0x00);
        uint8_t ARP_dest3=lagTil8(packet[ARP_destIP+1],0x01);

        senderMAC[0]=lagTil8(packet[ARP_senderMAC],0x00);
        senderMAC[1]=lagTil8(packet[ARP_senderMAC],0x01);
        senderMAC[2]=lagTil8(packet[ARP_senderMAC+1],0x00);
        senderMAC[3]=lagTil8(packet[ARP_senderMAC+1],0x01);
        senderMAC[4]=lagTil8(packet[ARP_senderMAC+2],0x00);
        senderMAC[5]=lagTil8(packet[ARP_senderMAC+2],0x01);

        if((ARP_dest0==minIP[0])&&
                (ARP_dest1==minIP[1])&&
                (ARP_dest2==minIP[2])&&
                (ARP_dest3==minIP[3])){
                svarPakkeARP();
        }
}



void svarPakkeARP(void){

                skrivTilRTL(CR,0x0022);                                 //Abort/Complete remote DMA, (og STA)
                skrivTilRTL(TPSR,txStart);                              //Transmit Page Start Register,16-bit modus-> 0x80-8Kb=0x60
                skrivTilRTL(RSAR0,0x0000);                              //Remote Start Adress Register, startadresse for remote DMA
                skrivTilRTL(RSAR1,0x0040);                              //Remote Start Adress Register, startadresse for remote DMA
                skrivTilRTL(RISR,0x00FF);                               //Interrupt Status Register, resetter ved å skrive til 1
                skrivTilRTL(RBCR0,(lagTil8((pgheader[0x01]),0x00))-4);  //Remote Byte Count Register, datateller for remote DMA
                skrivTilRTL(RBCR1,lagTil8((pgheader[0x01]),0x01));      //Remote Byte Count Register, datateller for remote DMA
                skrivTilRTL(CR,0x0012);                                 //"Remote write" start

        for(int i=0;i<5;i++){                                           //Motager MAC adreasse
                skrivTilRTL(RDMA,lagTil16(senderMAC[i+1],senderMAC[i]));
                i++;
                }
        for(int i=0;i<5;i++){                                           //Avsender MAC adresse (min)
                skrivTilRTL(RDMA,lagTil16(minMAC[i+1],minMAC[i]));
                i++;
                }
        skrivTilRTL(RDMA,0x0608);                                       //Pakke er av type ARP
        skrivTilRTL(RDMA,ARP_hwtype);                                   //Hardware type, 10Mb Ethernet=0x0001
        skrivTilRTL(RDMA,ARP_protoType);                                //Protokoll type, IP
        skrivTilRTL(RDMA,ARP_hwprlen);                                  //Hardware size,Protocol size
        skrivTilRTL(RDMA,0x0200);                                       //ARP reply
        for(int i=0;i<5;i++){                                           //Avsender MAC adresse (min)
                skrivTilRTL(RDMA,lagTil16(minMAC[i+1],minMAC[i]));
                i++;
                }
        for(int i=0;i<4;i++){                                           //Avsender IP adreasse (min)
                skrivTilRTL(RDMA,lagTil16(minIP[i+1],minIP[i]));
                i++;
                }
        for(int i=0;i<5;i++){                                           //Mottager MAC adresse
                skrivTilRTL(RDMA,lagTil16(senderMAC[i+1],senderMAC[i]));
                i++;
                }
        for(int i=0;i<4;i++){                                           //Motager IP adreasse
                i++;
                skrivTilRTL(RDMA,lagTil16(senderIP[i+1],senderIP[i]));
                }
        for(int i=0;i<9;++i){                                           //Fyller til 60B pakke
                skrivTilRTL(RDMA,0x5555);
```

```
                }
        while(!(lesFraRTL(RISR) & 0x40));                       //Evig løkke til remote DMA er ferdig
        skrivTilRTL(TBCR0,(lagTil8((pgheader[0x01]),0x00))-4);  //Leser inn total byte som skal sendes
        skrivTilRTL(TBCR1,lagTil8((pgheader[0x01]),0x01));
        skrivTilRTL(CR,0x24);                                   //Send kommando
        }


//*********************************************************************
//*        Perform ICMP Function
//*   This routine responds to a ping.
//*********************************************************************
void icmp(void){

        uint8_t ekkoReq=lagTil8(packet[0x11],0x00);             //Henter om ekkoreq på mottatt pakke
        if(ekkoReq==0x08){                                      //Sjekker om icmp er ekkorequest
                skrivTilRTL(CR,0x0022);                         //Abort/Complete remote DMA, (og STA)
                skrivTilRTL(TPSR,txStart);                      //Transmit Page Start Register,16-bit modus-> 0x80-8Kb=0x60
                skrivTilRTL(RSAR0,0x0000);                      //Remote Start Adress Register, startadresse for remote DMA
                skrivTilRTL(RSAR1,0x0040);                      //Remote Start Adress Register, startadresse for remote DMA
                skrivTilRTL(RISR,0x00FF);                       //Interrupt Status Register, resetter ved å skrive til 1
                skrivTilRTL(RBCR0,(lagTil8((pgheader[0x01]),0x00))-4);  //Remote Byte Count Register, datateller for remote DMA
                skrivTilRTL(RBCR1,lagTil8((pgheader[0x01]),0x01));      //Remote Byte Count Register, datateller for remote DMA
                skrivTilRTL(CR,0x0012);                         //"Remote write" start

                for(int i=0;i<5;i++){                           //Motager MAC adreasse
                        skrivTilRTL(RDMA,lagTil16(senderMAC[i+1],senderMAC[i]));
                        i++;
                        }
                for(int i=0;i<5;i++){                           //Avsender MAC adresse (min)
                        skrivTilRTL(RDMA,lagTil16(minMAC[i+1],minMAC[i]));
                        i++;
                        }
                skrivTilRTL(RDMA,0x0008);                       //Pakke er av type IP
                skrivTilRTL(RDMA,packet[IP_versLen]);           //Versjon og header lengd
                skrivTilRTL(RDMA,packet[0x08]);                 //Totale lengde på pakken
                skrivTilRTL(RDMA,packet[0x09]);                 //Returnerer Identifier sendt av avsender
                skrivTilRTL(RDMA,packet[0x0A]);                 //Returnerer FragmentOffset sendt av avsender
                skrivTilRTL(RDMA,packet[0x0B]);                 //Returnerer TimeToLive og protokoll type sendt av avsender
                /*** Lager header kontrollsummen ***/
                uint32_t sum = 0x0;                             //Variabel for bruk for lagring av checksum
                packet[0x0C]=0x0000;                            //Setter avsender kontrollsum lik null(pga. egen utregning)
                for(int i=(0x07);i<0x11;i++){                           //Fra packet plass 0x08 til 0x10 (tom. destinasjon
IP)
                        sum=sum + packet[i];                    //Sumerer pakkene
                        }
                while(sum>>16){                                 //De siste 16 bit
                        sum = (sum & 0xFFFF) + (sum >> 16);     //Siste 16 bit legges til de øverste 16 bit
                        }
                sum = ~sum;                                     //En`s komplement
                /*********************************/
                skrivTilRTL(RDMA,sum);                          //Legger header kontrollsum inn i pakke
                for(int i=0;i<4;i++){                           //Avsender IP adreasse (min)
                        skrivTilRTL(RDMA,lagTil16(minIP[i+1],minIP[i]));
                        i++;
                        }
                for(int i=0;i<4;i++){                           //Motager IP adreasse
                        i++;
                        skrivTilRTL(RDMA,lagTil16(senderIP[i+1],senderIP[i]));
                        }
                packet[0x11]=0x0000;                            //Setter type ekko reply
                /******* Lager icmp kontrollsummen *******/
                uint16_t pkln = ((pgheader[0x01])/2)-2;         //Totale lengde på pakken
                packet[0x12]=0x0000;
                sum = 0x0000;                                   //Variabel for bruk for lagring av checksum
                for(int i=(0x11);i<pkln;i++){                   //Fra packet plass 0x11 til tom. siste datategn
                        sum=sum + packet[i];                    //Sumerer pakkene
                        }
                while(sum>>16){                                 //De siste 16 bit
                        sum = (sum & 0xFFFF) + (sum >> 16);     //Siste 16 bit legges til de øverste 16 bit
                        }
                sum = ~sum;                                     //En`s komplement
```

```
                    /**********************************/
                    packet[0x12]=sum;                                   //Legger kontrollsum pakkeposisjon
                    skrivTilRTL(RDMA,packet[0x11]);                     //Legger inn ekko reply
                    skrivTilRTL(RDMA,packet[0x12]);                     //Legger inn kontrollsum
                    skrivTilRTL(RDMA,packet[0x13]);                     //Legger inn identifier som mottatt pakke
                    skrivTilRTL(RDMA,packet[0x14]);                     //Legger inn sequence number som mottatt pakke
                    for(int i=0x15;i<0x3C;i++){
                            skrivTilRTL(RDMA,packet[i]);                //Legger inn data identisk til mottatt pakke
                            }
                    while(!(lesFraRTL(RISR) & 0x40));                   //Evig løkke til remote DMA er ferdig
                    skrivTilRTL(TBCR0,(lagTil8((pgheader[0x01]),0x00))-4);  //Leser inn total byte som skal sendes (60B)....74
                    skrivTilRTL(TBCR1,lagTil8((pgheader[0x01]),0x01));
                    skrivTilRTL(CR,0x24);                               //Send kommando
            }
}


/*************** Metode håndterer udp pakke fra PC (bruker kommandoer) ***************/
void udpMottatt(void){

            if((lagTil8(packet[0x15],0x00)==0x55) &&                    //Kontroll om bruker avslutter, stanser sampling
              (lagTil8(packet[0x16],0x00)==0x55)){
                    TIMSK &= ~(1<<OCIE1A);                              //TimerCounterComp.Match Interrupt Timer1A disables
                    ETIMSK &= ~(1<<OCIE3A);                             //TimerCounterComp.Match Interrupt Timer3A disables
                    stoppKretskortKommando = 0x55;
                    burstModus=0x0A;
                    }
            else{
                    burstFrekvens = lagTil8(packet[0x15],0x00);         //Henter verdi burstfrekvens fra udp-pakke
                    envelope = lagTil8(packet[0x15],0x01);              //Henter verdi envelope(antall perioder) fra udp-pakke
                    valgtEffekt = lagTil8(packet[0x16],0x00);           //Henter verdi valgtEffekt fra udp-pakke
                    gain = lagTil8(packet[0x16],0x01);                  //Henter verdi gain fra udp-pakke
                    burstModus = lagTil8(packet[0x17],0x00);            //Henter verdi burstModus fra dup-pakke(10,-ikke burst.
                                                                        //0,- kontinuerlig. 3,- single burst)
                    packet[0x17]=(lagTil16(0xCC,burstModus));           //Retur verdi til delphi burstmodus og type "confirm" (=0xCC)
                    identitet=(lagTil16(lagTil8(packet[0x09],0x00),
                    (lagTil8(packet[0x09],0x01))));                     //Identitet ved sampel-udp request til utregning kontrollsum
                    identitetRDMA=packet[0x09];                         //Identitet ved sampel-udp request til RTL
                    destinasjonsPort=packet[0x12];                      //Henter avsenders port
                    }
            /********** Rutine svarer avsender,bekrefter *********/
            skrivTilRTL(CR,0x0022);                                     //Abort/Complete remote DMA, (og STA)
            skrivTilRTL(TPSR,txStart);                                  //Transmit Page Start Register,16-bit modus-> 0x80-8Kb=0x60
            skrivTilRTL(RSAR0,0x0000);                                  //Remote Start Adress Register, startadresse for remote DMA
            skrivTilRTL(RSAR1,0x0040);                                  //Remote Start Adress Register, startadresse for remote DMA
            skrivTilRTL(RISR,0x00FF);                                   //Interrupt Status Register, resetter ved å skrive til 1
            skrivTilRTL(RBCR0,(lagTil8((pgheader[0x01]),0x00))-4);      //Remote Byte Count Register, datateller for remote DMA
            skrivTilRTL(RBCR1,lagTil8((pgheader[0x01]),0x01));          //Remote Byte Count Register, datateller for remote DMA
            skrivTilRTL(CR,0x0012);                                     //"Remote write" start
            for(int i=0;i<5;i++){                                       //Motager MAC adreasse
                    skrivTilRTL(RDMA,lagTil16(senderMAC[i+1],senderMAC[i]));
                    i++;
                    }
            for(int i=0;i<5;i++){                                       //Avsender MAC adresse (min)
                    skrivTilRTL(RDMA,lagTil16(minMAC[i+1],minMAC[i]));
                    i++;
                    }
            skrivTilRTL(RDMA,0x0008);                                   //Pakke er av type IP
            skrivTilRTL(RDMA,0x0045);                                   //Versjon og header lengd
            skrivTilRTL(RDMA,packet[0x08]);                             //Totale lengde på pakken
            skrivTilRTL(RDMA,packet[0x09]);                             //Returnerer Identifier sendt av avsender
            skrivTilRTL(RDMA,0x0000);                                   //FragmentOffset (flags)
            skrivTilRTL(RDMA,0x1180);                                   //76->TimeToLive,=118. 11->Type UDP pakke

            /********** Lager header kontrollsummen *********/
            uint32_t sum = 0x0;                                         //Variabel for bruk for lagring av checksum
            packet[0x0C]=0x0000;                                        //Setter avsender kontrollsum lik null(pga. egen utregning)
            for(int i=(0x07);i<0x11;i++){                               //Fra packet plass fom. 0x08 tom. 0x10 (tom. destinasjon IP)
                    sum=sum + packet[i];                                //Sumerer pakkene
                    }
            while(sum>>16){                                             //De siste 16 bit
                    sum = (sum & 0xFFFF) + (sum >> 16);                 //Siste 16 bit legges til de øverste 16 bit
```

```
                }
        sum = ~sum;                                             //En`s komplement
        /***********************************************/
        skrivTilRTL(RDMA,sum);                                  //Legger header kontrollsum inn i pakke
        for(int i=0;i<4;i++){                                   //Avsender IP adreasse (min)
                skrivTilRTL(RDMA,lagTil16(minIP[i+1],minIP[i]));
                i++;
                }
        for(int i=0;i<4;i++){                                   //Motager IP adreasse
                i++;
                skrivTilRTL(RDMA,lagTil16(senderIP[i+1],senderIP[i]));
                }
        skrivTilRTL(RDMA,destinasjonsPort);                     //Returnerer port kretskort
        skrivTilRTL(RDMA,packet[0x11]);                         //Returnerer til mottatt avsender port
        skrivTilRTL(RDMA,0x2800);                               //Lengde på returnert pakke
        skrivTilRTL(RDMA,0x0000);                               //Checksum settes til null
        if(reSend==0x01){                                       //om ovverrun er kjørt
                skrivTilRTL(RDMA,0x0101);
                skrivTilRTL(RDMA,0x0101);
                skrivTilRTL(RDMA,0x0101);
                reSend=0x00;                                    //Resetter reSend
                }
        else if(stoppKretskortKommando == 0x55){
                skrivTilRTL(RDMA,(lagTil16(envelope,0x55)));    //Retur av mottatte og satt styringsvariabler
                skrivTilRTL(RDMA,(lagTil16(gain,0x55)));        //Retur av mottatte styringsvariabler
                skrivTilRTL(RDMA,(lagTil16(0xCC,burstModus)));  //Retur av mottatte styringsvariabler
                stoppKretskortKommando=0x00;
                }
        else{
                skrivTilRTL(RDMA,(lagTil16(envelope,burstFrekvens)));   //Retur av mottatte og satt styringsvariabler
                skrivTilRTL(RDMA,(lagTil16(gain,valgtEffekt)));         //Retur av mottatte styringsvariabler
                skrivTilRTL(RDMA,(lagTil16(0xCC,burstModus)));          //Retur av mottatte styringsvariabler
                }
        for(int i=0;i<0x0D;i++){                                //Fyller pakken med 2*16=32 byte
                skrivTilRTL(RDMA,0x8585);
                }
        while(!(lesFraRTL(RISR) & 0x40));                       //Evig løkke til remote DMA er ferdig
        skrivTilRTL(TBCR0,(lagTil8((pgheader[0x01]),0x00))-4);  //Leser inn total byte som skal sendes XXX Byte
        skrivTilRTL(TBCR1,lagTil8((pgheader[0x01]),0x01));
        skrivTilRTL(CR,0x24);                                   //Send kommando
        effekt();
        send_Spi(gain,0x01);
}

/********** Metode bygger header udp om nødvendig ***************/
void udpHeaderToSend(void){
udpH=0x00;
        if(udpH==0x00){                                         //Om variabel "packet" er endret
                skrivTilRTL(CR,0x0022);                         //Abort/Complete remote DMA, (og STA)
                skrivTilRTL(TPSR,txStart);                      //Transmit Page Start Register,16-bit modus-> 0x80-8Kb=0x60
                skrivTilRTL(RSAR0,0x0000);                      //Remote Start Adress Register, startadresse for remote DMA
                skrivTilRTL(RSAR1,0x0040);                      //Remote Start Adress Register, startadresse for remote DMA
                skrivTilRTL(RISR,0x00FF);                       //Interrupt Status Register, resetter ved å skrive til 1
                skrivTilRTL(RBCR0,0x00D6);                      //Remote Byte Count Register, datateller for remote DMA
                skrivTilRTL(RBCR1,0x0003);                      //Remote Byte Count Register, datateller for remote DMA
                skrivTilRTL(CR,0x0012);                         //"Remote write" start

                for(int i=0;i<5;i++){                           //Motager MAC adreasse
                        skrivTilRTL(RDMA,lagTil16(senderMAC[i+1],senderMAC[i]));
                        i++;
                        }
                for(int i=0;i<5;i++){                           //Avsender MAC adresse (min)
                        skrivTilRTL(RDMA,lagTil16(minMAC[i+1],minMAC[i]));
                        i++;
                        }
                skrivTilRTL(RDMA,0x0008);                       //Pakke er av type IP
                skrivTilRTL(RDMA,0x0045);                       //Versjon og header lengd
                skrivTilRTL(RDMA,0xC803);                       //Totale lengde på pakken
                skrivTilRTL(RDMA,identitetRDMA);                //Returnerer egen identifier 0x4321
                skrivTilRTL(RDMA,0x0000);                       //FragmentOffset (flags)
                skrivTilRTL(RDMA,0x1180);                       //80->TimeToLive,=128. 11->Type UDP pakke
                packet[0x0A]=0x4500;                            //Versjon og header lengd
```

```
                packet[0x0B]=0x03C8;                                    //Totale lengde på pakken
                packet[0x0C]=identitet;                                 //Returnerer egen identifier 0x4321
                packet[0x0D]=0x0000;                                    //FragmentOffset (flags)
                packet[0x0E]=0x8011;                                    //80->TimeToLive,=128. 11->Type UDP pakke
                packet[0x0F]=lagTil16(minIP[0],minIP[1]);
                packet[0x10]=lagTil16(minIP[2],minIP[3]);
                packet[0x11]=lagTil16(senderIP[1],senderIP[2]);
                packet[0x12]=lagTil16(senderIP[3],senderIP[4]);
        /********** Lager header kontrollsummen ************/
                uint32_t sum = packet[0x0A];                            //Variabel for bruk for lagring av checksum
                for(int i=(0x0B);i<(0x13);i++){                         //Fra packet plass 0x08 til 0x10 (tom. destinasjon IP)

                        sum=sum + packet[i];                            //Sumerer pakkene
                        }
                while(sum>>16){                                         //De siste 16 bit
                        sum = (sum & 0xFFFF) + (sum >> 16);             //Siste 16 bit legges til de øverste 16 bit
                        }
                sum = ~sum;                                             //En`s komplement
        /*************************************************/
                uint8_t temp1 = lagTil8(sum,0x01);
                uint8_t temp2 = lagTil8(sum,0x00);
                sum = lagTil16(temp2,temp1);
                skrivTilRTL(RDMA,sum);                                  //Legger header kontrollsum inn i pakke

                for(int i=0;i<4;i++){                                   //Avsender IP adreasse (min)
                        skrivTilRTL(RDMA,lagTil16(minIP[i+1],minIP[i]));
                        i++;
                        }
                for(int i=0;i<4;i++){                                   //Motager IP adreasse
                        i++;
                        skrivTilRTL(RDMA,lagTil16(senderIP[i+1],senderIP[i]));
                        }
        udpH=0x01;                                                      //udpHeaderToSend udp-pakke-header er lagret i send
buffer,=0x01
        }
        skrivTilRTL(RDMA,destinasjonsPort);                             //Destination port mottat fra avsender
        skrivTilRTL(RDMA,0x8813);                                       //Kretskort port mottat fra avsender
        skrivTilRTL(RDMA,0xB403);                                       //Lengde på pakke fom. sourceport tom. siste sampelpakke
        skrivTilRTL(RDMA,0x0000);                                       //Checksum settes til null (data kommer etter denne lages)
        skrivTilRTL(RDMA,lagTil16(envelope,burstFrekvens));             //variabler fra Skudd.h
        skrivTilRTL(RDMA,lagTil16(gain,valgtEffekt));                   //Variabel effekt->Skudd.h. Variabel gain->TVG.h
        skrivTilRTL(RDMA,lagTil16(0xEE,(udpNr+1)));                     //udp pakke nr(sampel-pakke) og type sampel(=0xEE)
}

/********** Metode sampler og sender udp-pakke **************/
void udpSample(void){                                                   //Metode legger datasamples i udp-pakke

        TIFR   |= (1<<OCF1A);                                           //Resetter flagg OCF1A
        ETIFR  |= (1<<OCF3A);                                           //Resetter flagg OCF3A
        TIMSK  |= (1<<OCIE1A);                                          //TimerCounterComp.Match Interrupt Timer1A enabler
        ETIMSK |= (1<<OCIE3A);                                          //TimerCounterComp.Match Interrupt Timer3A enabler

        enableADC();
        PORTF &= ~(1<<2);
        writeStrobe();
        udpNr++;
        for(int i=0;i<0x01D2;i++){                                      //Kjør 342 (0x02AD sampels à 16Bit)

                                                                        //Kjører til komplett udppakke (733-header-delphikomandoer =
                                                                        //685 datasampel pakker)
                                                                        //Må deles på 2 da sampler 16bit.1466/2=733=0x02DD-
datakomandoer
        konverter();
        writeStrobe();
                while(PORTE&(1<<0)){}                                   //Venter på trigger interrupt i main (oppdatering av kjorSample).
                PORTE|=(1<<0);
                }DIODR_OFF;
        TIMSK  &=~(1<<OCIE1A);                                          //TimerCounterComp.Match Interrupt Timer1A disabler
        ETIMSK &=~(1<<OCIE3A);                                          //TimerCounterComp.Match Interrupt Timer3A disabler
        while(!(lesFraRTL(RISR) & 0x40));                               //Evig løkke til remote DMA er ferdig
        PORTF |= (1<<2);
        skrivTilRTL(TBCR0,0x00D6);                                      //Leser inn total byte som skal sendes 1466 Byte
```

```
        skrivTilRTL(TBCR1,0x0003);
        skrivTilRTL(CR,0x24);                                   //Send kommando
}

/********** Metode håndterer buffer overrun ***************/
void overrun(void){

        reSend = 0x00;

        uint16_t lestVerdi = lesFraRTL(CR);                     //Les status CR
        skrivTilRTL(CR,0x21);                                   //Kjører stopp komando,abort/complete DMA
        _delay_ms(2);                                           //Vent minst 1.6ms
        skrivTilRTL(RBCR0,0x00);                                //Resetter Remote Byte Counter0
        skrivTilRTL(RBCR1,0x00);                                //Resetter Remote Byte Counter1
        if(!(lestVerdi&(1<<2))){                                //Om RXE er null
                reSend=0x00;                                    //Ikke CRC error,Frame error eller pakke mistet
                }
                else if(lestVerdi&(1<<2)){                      //Om RXE er satt
                        lestVerdi = lesFraRTL(RISR);
                        if((!(lestVerdi&(1<<1))) || (!(lestVerdi&(1<<3)))){  //Om ikke PTX og TXE er satt(pakke sendt
                                                                //ikke error,aborted collitions)
                                reSend=0x00;
                                }
                                else{
                                reSend=0x01;                    //Bruker må sende pakke på nytt
                                }
                        }
        skrivTilRTL(TCR,0x0002);                                //Set CRC (Internal Loopback)
        skrivTilRTL(CR,0x0022);                                 //Initsierer start kommando
        skrivTilRTL(BNRY,0x0046);                               //Boundary Register
        skrivTilRTL(CR,0x0062);                                 //Resetter flagg, ikke RDC(Remote DMA completed)
        skrivTilRTL(CURR,0x0046);                               //Crerent Page Register
        skrivTilRTL(CR,0x0022);                                 //Igangsetter Ethernetkontroller, RemoteWrite
        skrivTilRTL(RISR,0x10);                                 //Resetter flagget OVW
        skrivTilRTL(TCR,0x00);                                  //Enable CRC kontroll og generator,ikke loopback,
                                                                //-> og AutoTransmitDisable,CollisionOffsetDisable

        }
/ ******************************************************************/
```

```
/*****************************************************************************
*********************************' .h '*************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

TVG.h
...
*****************************************************************************
*****************************************************************************/

void send_Spi(uint8_t, uint8_t);

extern uint8_t gain;
*****************************************************************************
*****************************************************************************
```

```
/*****************************************************************************
*********************************' *************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

TVG
Her er kode for kontroll og parameter verdier vedrørende TVG-justering
av innkommende ekkosignal
*****************************************************************************
*****************************************************************************/
#include <avr/io.h>
#include <util/delay.h>

uint8_t gain;                                   //Variabel fra Delphi for ingangsforsterkning.Brukt i Eth.c
                                                //variabel "gain" reflekterer verdi av forsterkning. Sendes i udp

pakkene.
void send_Spi(uint8_t dataverdi,uint8_t adresse){

        uint8_t kanal,data;
        data=dataverdi;
        kanal=adresse;

        PORTB &=~(1<<4);                        // LOAD/SHIFT settes lav
        char c=SPDR;
        SPDR=kanal;
        while(!(SPSR & (1<<SPIF)));             //SPI interrupt flagg settes ved overføring ferdig
        c=SPDR;
        SPDR=data;
        while(!(SPSR & (1<<SPIF)));             //SPI interrupt flagg settes ved overføring ferdig
        c=SPDR;
        PORTB |= (1<<4);                        // LOAD/SHIFT settes høy
}
```

```
/*****************************************************************************
*********************************' .h '*************************************
Sonarsystem
2008        vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

ADC.h
...
*****************************************************************************
*****************************************************************************/
void konverter(void);
void enableADC(void);
void disableADC(void);

/*****************************************************************************
*****************************************************************************
```

```
/*****************************************************************************
*********************************' ***'*************************************
Sonarsystem
2008        vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

ADC
Her kontrolleres og brukes ADC styrt av uC
*****************************************************************************
*****************************************************************************/
#include <avr/io.h>
#include <util/delay.h>

#include "Eth.h"

void konverter(void){                                   //Kjører rutine for å ta sample
        PORTF &= ~(1<<1);                               //Resetter !CNVST (aktiv lav trigger konvertering ADC)
        PORTF |= (1<<1);                                //Setter !CNVST (aktiv lav trigger konvertering ADC)
                                                        //Delay one cycle (1/16000000) =ca63ns
}

void enableADC(void){
        set_busAdresse(RDMA);                           //Data skal sendes til RDMA.Setter fast adresse RTL
        dataFraRTL();
}

void disableADC(void){
        PORTF |= (1<<2);                                //ADC disable data til bus (!RD aktiv lav)
}
```

```
/*****************************************************************************
*******************************' .h '*****************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

Skudd.h
...
*****************************************************************************
*****************************************************************************/
void burst(uint8_t);

void sendBurst(void);
void effekt(void);
void shutdown(void);

extern uint8_t burstFrekvens;
extern uint8_t envelope;
extern uint8_t valgtEffekt;
/*****************************************************************************
*****************************************************************************
```

```
/*****************************************************************************
*****************************************************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

Skudd
Setter variabler for skudd. Dvs. effekt, frekvens, envelope,shutdown. Her
kontrolleres transduseren
*****************************************************************************
*****************************************************************************/
#include "Init.h"
#include "Eth.h"
#include "ADC.h"

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <inttypes.h>

/*****************************************************************************
              Definisjoner for kontroll av sendereffekt
*****************************************************************************/
#define PW10          (PORTG&=~(1<<3));(PORTB&=~(1<<7));          //Setter sendereffekt 10W(3,8V)
#define PW87          (PORTG|=(1<<3));(PORTB&=~(1<<7));           //Setter PG3,sendereffekt 87W(12V)
#define PW300         (PORTG&=~(1<<3));(PORTB|=(1<<7));           //Stter PB7,sendereffekt 300W(24V)
/*****************************************************************************/
#define DIODY_ON PORTE &= ~(1<<2);
#define DIODY_OFF PORTE|= (1<<2);

uint8_t burstFrekvens,envelope,valgtEffekt;                       //Deklarerer variabler benyttet av metoder utenfor Skudd.c
uint8_t feq=0x00;

void burst(uint8_t frekvens){

              udpNr=0x00;                                         //Resetter teller for UDPsampel-pakke nr.X
              uint8_t nyFrek = frekvens;

              if(nyFrek==0x32){                                   //Rutine signaliserer med diodene
```

```
                DIODY_ON;
                _delay_ms(50);
                DIODY_OFF;
                _delay_ms(50);
                DIODY_ON;
                _delay_ms(50);
                DIODY_OFF;
                _delay_ms(50);
                DIODY_ON;
                _delay_ms(50);
                DIODY_OFF;
                _delay_ms(50);
                DIODY_ON;
                _delay_ms(50);
                DIODY_OFF;
                _delay_ms(50);
                }
        }

void effekt(void){

        uint8_t watt = valgtEffekt;
        if(watt==0x01){                                         //Sendereffekt 10W
                valgtEffekt=0x01;
                PW10;
                }
        if(watt==0x08){                                         //Sendereffekt 87W
                valgtEffekt=0x87;
                PW87;
                }
        if(watt==0x1E){                                         //Sendereffekt 300W
                valgtEffekt=0x30;
                PW300;
                }
        }

void shutdown(void){
        PORTB |= (1<<6);                                        //Setter "shutdown" for disable senderkrets
        }

void sendBurst(void){

        uint8_t freq=0x00;

        if(burstFrekvens==0x46){                                //70KHz transduserfrekvens
                freq=0x01;
                }
        else if(burstFrekvens==0x32){                           //50KHz transduserfrekvens
                freq=0x02;
                }
        else{
                burstFrekvens=0x00;
                }
        PORTB &= ~(1<<6);                                       //Resetter "shutdown" for enable senderkrets
        DIODR_ON;
        if(burstFrekvens!=0x55){                                //Send burst
                for(int i=0;i<envelope;i++){                    //Genererer skudd frekvens, avhengig av brukervariabel
                                                                //envelope(antall perioder)
                        PORTB |= (1<<5);                        //Setter port
                        _delay_ms((1/burstFrekvens)/2);         //Brukervariabel bestemmer senderfrekvens[KHz]((1/fek)=T,
                                                                //T/2=halv periode)
                        PORTB &= ~(1<<5);                       //Resetter port
                        _delay_ms((1/burstFrekvens)/2);         //Brukervariabel bestemmer senderfrekvens[KHz]((1/fek)=T,
                                                                //T/2=halv periode)
                        }
                }
        SFIOR |= (1<<PSR321);                                   //Resetter timere1-2-3.Teller med identisk
                                                                //starttidspunkt(ADC,TVG)
                PORTB |= (1<<6);                                //Setter "shutdown" for disable senderkrets

        }
```

```
/*****************************************************************************
*********************************' .h '*************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

RS232.h
...
*****************************************************************************
*****************************************************************************/
void RTLregRS232(void);
void binTilHex(uint16_t);
void visPakke(void);

/*til testkjøring*/
void skrivSkjerm(char s[]);

uint8_t lesRTL(uint8_t addr);
/*****************************************************************************
*****************************************************************************
*****************************************************************************
*****************************************************************************
Sonarsystem
2008          vers.1.0
Knut Skumsvoll

Mikrokontroller ATmega64
16MHz ekstern krystall

RS232
Her brukes kode som verktøy ved programmering av enheten. RS232 kan skrive
ut tabell over RTL8019AS paramerere eller enkeltverdier eller setninger. Det
benyttes TeraTerm som display av meldinger i arbeidet med masterprosjektet
*****************************************************************************
*****************************************************************************/
#include "Eth.h"
#include "RS232.h"

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <inttypes.h>

uint8_t RSvariabel;
uint16_t lestData;

void binTilHex(uint16_t verdi);
void skrivSkjerm(char s[]);

void RTLregRS232(void){

                set_busAdresse(CR);                                     //Adresse CR
                set_busData(0x21);                                      //Stop mottak/sende pakker
                writeStrobe();
                skrivSkjerm("\r\n    Realtek 8019AS Register status   \n\n\r");
                skrivSkjerm("Register\tSide 0\tSide 1\tSide 2\tSide 3\n\r");
                cli();                                                  //Dissabler global interrupt
                for(uint8_t i=0;i<16;i++){
   binTilHex(i);                                                        //Kaller metode binTilHex
                skrivSkjerm("\t\t ");                                   //Skriver nye linje til skjerm
   dataTilRTL();
                set_busAdresse(CR);                                     //Adresse CR
```

```
		set_busData(0x0020);				//Side 0
		writeStrobe();
		binTilHex(lesRTL(i));				//Leser fra RTL, omformer fra bin til til hex
		skrivSkjerm("\t ");				//Skriver ny linje til skjerm
		dataTilRTL();
		set_busAdresse(CR);				//Adresse CR
		set_busData(0x0060);				//Side 1
		writeStrobe();
		binTilHex(lesRTL(i));				//Leser fra RTL, omformer fra bin til til hex
		skrivSkjerm("\t ");				//Skriver ny linje til skjerm
		dataTilRTL();
		set_busAdresse(CR);				//Adresse CR
		set_busData(0x00A0);				//Side 2
		writeStrobe();
		binTilHex(lesRTL(i));				//Leser fra RTL, omformer fra bin til til hex
		skrivSkjerm("\t ");				//Skriver ny linje til skjerm
		dataTilRTL();
		set_busAdresse(CR);				//Adresse CR
		set_busData(0x00E0);				//Side 3
		writeStrobe();
		binTilHex(lesRTL(i));				//Leser fra RTL, omformer fra bin til til hex
		skrivSkjerm("\r\n");				//Skriver ny linje til skjerm
  }
  sei();						//Enabler global interrupt
}


void binTilHex(uint16_t verdi){				//Metode for konvertering binær til hexverdi

		uint16_t high, high_n, low, low_n;
		high_n = (verdi & 0x00F0) / 16;
		if(high_n > 0x0009){
			high = high_n + 0x0037;
			}
		else{
			high = high_n + 0x0030;
			}
		while(!(UCSR1A & (1<<UDRE1)));
		UDR1 = high;
		low_n = (verdi & 0x000F);
		if(low_n > 0x0009){
			low = low_n + 0x0037;
			}
		else{
			low = low_n + 0x0030;
			}
		while(!(UCSR1A & (1<<UDRE1)));
		UDR1 = low;
}

void skrivSkjerm(char s[]){				//Metode skriver txt ut til skjerm via RS232

		int i=0;
		while(s[i]!='\0'){
			while(!(UCSR1A & (1<<UDRE1)));
			UDR1=s[i];
			i++;
		}
}

uint8_t lesRTL(uint8_t addr){				//Metode leser verdi fra adresert plass

		RSvariabel=addr;

		set_busAdresse(RSvariabel);			//Setter adresse til adressebus
		dataFraRTL();					//Setter pin fra RTL på uC til ingang
		lestData = read_busData();			//Leser data fra databus
		return lestData;
}

void visPakke(void){					//Skriver ut datapakke i buffer
```

```
	uint8_t dataLengde = 0x00;

	skrivSkjerm("\x1b[2J");				//"Tømmer" vinduet for txt ved å fylle inn blank
	skrivSkjerm("\r\n");				//Skriver ny linje til skjerm med innrykk til marg
	for(int i=0; i<96; i++){
		binTilHex(packet[i]);			//Plass i i pakken i bufferminnet, skriver til skjerm
		dataLengde++;				//Øker variabel med èn
		if(dataLengde==0x10){
			dataLengde=0x00;
			skrivSkjerm("\r\n");		//Ny linje, flytt til marg
		}
	}
}
```

Vedlegg 15 Delphi kode for konstruksjon av brukerprogram

MAIN

```
unit Main;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, IdBaseComponent, IdComponent, IdTCPServer, StdCtrls,
  ComCtrls, Menus, TeEngine, Series, TeeProcs, Chart, DbChart,
  IdUDPBase, IdUDPServer,IdSocketHandle, Sockets;

type
  TForm_Main = class(TForm)
    Panel1: TPanel;
    ProgressBar1: TProgressBar;
    UpDown1: TUpDown;
    Bt_StartStop: TButton;
    Bt_toOptions: TButton;
    Label1: TLabel;
    SonarGraf: TDBChart;
    UDPServer: TIdUDPServer;
    RadioGroup1: TRadioGroup;
    Ed_MainGainverdi: TEdit;
    Sirkel: TShape;
    Bt_MainSendUDP: TButton;
    Label2: TLabel;
    Bt_EnkeltSkudd: TButton;
    Bt_KontSkudd: TButton;
    Memo1: TMemo;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series3: TLineSeries;
    procedure Bt_StartStopClick(Sender: TObject);
    procedure Bt_toOptionsClick(Sender: TObject);
    procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);
    procedure sendUDP();
    procedure brukerEndret();
    procedure FormCreate(Sender: TObject);
    procedure verdiBurst();
    procedure getEnvelopeVerdi();
    procedure getGainVerdi();
    procedure avsluttKretskort();
    procedure verdiRadioGroup();
    procedure Bt_MainSendUDPClick(Sender: TObject);
    procedure RadioGroup1Click(Sender: TObject);
    procedure Bt_EnkeltSkuddClick(Sender: TObject);
    procedure Bt_KontSkuddClick(Sender: TObject);
    procedure UDPServerUDPRead(Sender: TObject; AData: TStream;
      ABinding: TIdSocketHandle);
    procedure brukMottattUDP(Sender: Array of Byte; Datalengde : Integer);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Main: TForm_Main;
  SendBuf : Array[0..32] of Byte;
  MottaksBuffer : Array [0..1024] of Byte;
implementation

uses Properties;

{$R *.dfm}

procedure TForm_Main.Bt_StartStopClick(Sender: TObject);
begin
 if Bt_StartStop.Caption = 'Start' then
```

```
 begin
   Bt_StartStop.Caption:= 'Stop';

   UDPServer.Active:=True;
   verdiRadioGroup();
   getGainVerdi();
   SendUDP();
   Bt_MainSendUDP.Caption:= 'Sendt';
 end
 else if Bt_StartStop.Caption = 'Stop' then
 begin
   Bt_StartStop.Caption:='Start';
   avsluttKretskort();
   SendUDP();
   Sirkel.Brush.Color := clRed;
   Bt_MainSendUDP.Caption:= 'Send';
   Bt_KontSkudd.Caption := 'Kont. skudd';
 end;
end;

procedure TForm_Main.Bt_toOptionsClick(Sender: TObject);
begin
  Form_Option.Visible:=true;
end;

procedure TForm_Main.UpDown1Click(Sender: TObject; Button: TUDBtnType);
begin
  if Button = btNext then ProgressBar1.Position:=ProgressBar1.Position+1
  else if Button = btPrev then ProgressBar1.Position:=ProgressBar1.Position-1;
  Ed_MainGainverdi.Text := IntToStr(ProgressBar1.Position);
    brukerEndret();
end;

procedure TForm_Main.brukerEndret();
begin
 Bt_MainSendUDP.Caption := 'Send';
 Sirkel.Brush.Color := clRed;
 Label2.Caption := '   Kretskort ikke oppdatert';
end;

procedure TForm_Main.sendUDP();{Metode sender Data-> Burstfrek,Envelope,Effekt,Gain,SingleBurst?}
 var s : Integer;
begin
 s := StrToInt(Form_Option.Lb_OptionsDestPort.Text);
 UDPServer.SendBuffer('129.240.84.8',s,SendBuf,32);
end;

procedure TForm_Main.FormCreate(Sender: TObject);
begin
 SendBuf[0]:= 70;            //Burstfrekvens
 SendBuf[1]:= 10;            //Envelope(antall perioder)
 SendBuf[2]:= 1;             //Effekt [W]
 SendBuf[3]:= 0;             //Forsterkning
 SendBuf[4]:= 10;            //Single Burst (10,-ikke burst. 0,- kontinuerlig. 3,- single burst)
 SendBuf[5]:= 85;
 SendBuf[31]:=85;
end;

procedure TForm_Main.verdiBurst();
begin
 SendBuf[0]:= StrToInt(Form_Option.Lb_OptionsBurstFrek.Text);
end;

procedure TForm_Main.getEnvelopeVerdi();
begin
 SendBuf[1]:= StrToInt(Form_Option.Lb_OptionsEnvelope.Text);
end;

procedure TForm_Main.getGainVerdi();
begin
 SendBuf[3]:= ProgressBar1.Position;
end;
```

```
procedure TForm_Main.verdiRadioGroup();
begin
 case RadioGroup1.ItemIndex of
 0 : begin
       SendBuf[2] := 1;
     end;
 1 : begin
       SendBuf[2] := 8;
     end;
 2 : begin
       SendBuf[2] := 30;
     end;
 end;
end;

procedure TForm_Main.avsluttKretskort();
begin
 SendBuf[0]:=85;                        //Avslutt kommando
 SendBuf[2]:=85;                        //Avslutt kommando
 SendBuf[4]:=10;                        //Komando stans burst
end;

procedure TForm_Main.UDPServerUDPRead(Sender: TObject; AData: TStream;
  ABinding: TIdSocketHandle);
var
 Data: Integer;
begin
  Data:=Adata.Read(MottaksBuffer,AData.Size);
  brukMottattUDP(MottaksBuffer,Data);
end;

procedure TForm_Main.Bt_MainSendUDPClick(Sender: TObject);
begin
 if (Bt_StartStop.Caption = 'Stop') then
   begin
   Bt_MainSendUDP.Caption:= 'Sendt';
   verdiRadioGroup();
   verdiBurst();
   getGainVerdi();
   SendUDP();
   end;
end;

procedure TForm_Main.RadioGroup1Click(Sender: TObject);
begin
 brukerEndret();
end;

procedure TForm_Main.Bt_EnkeltSkuddClick(Sender: TObject);
begin
 if (Bt_StartStop.Caption = 'Stop') and (Bt_KontSkudd.Caption = 'Kont. skudd') then
  begin
   SendBuf[0]:= StrToInt(Form_Option.Lb_OptionsBurstFrek.Text);
   SendBuf[1]:= StrToInt(Form_Option.Lb_OptionsEnvelope.Text);
   verdiRadioGroup();
   getGainVerdi();
   SendBuf[4]:=3;                       //Setter til single burst
   sendUDP();
   SendBuf[4]:=10;                       //Resetter verdi for kontinuerlig burst
   brukerEndret();
  end
 else if Bt_StartStop.Caption = 'Stop' then
  begin
   showMessage('Ugyldig valg! Kontinuerlig skudd er i drift.');
////////Hvordan endre navn påå ramma til Warning ?
  end;
end;

procedure TForm_Main.Bt_KontSkuddClick(Sender: TObject);
begin
 if (Bt_StartStop.Caption = 'Stop') and (Bt_KontSkudd.Caption = 'Kont. skudd') then
  begin
```

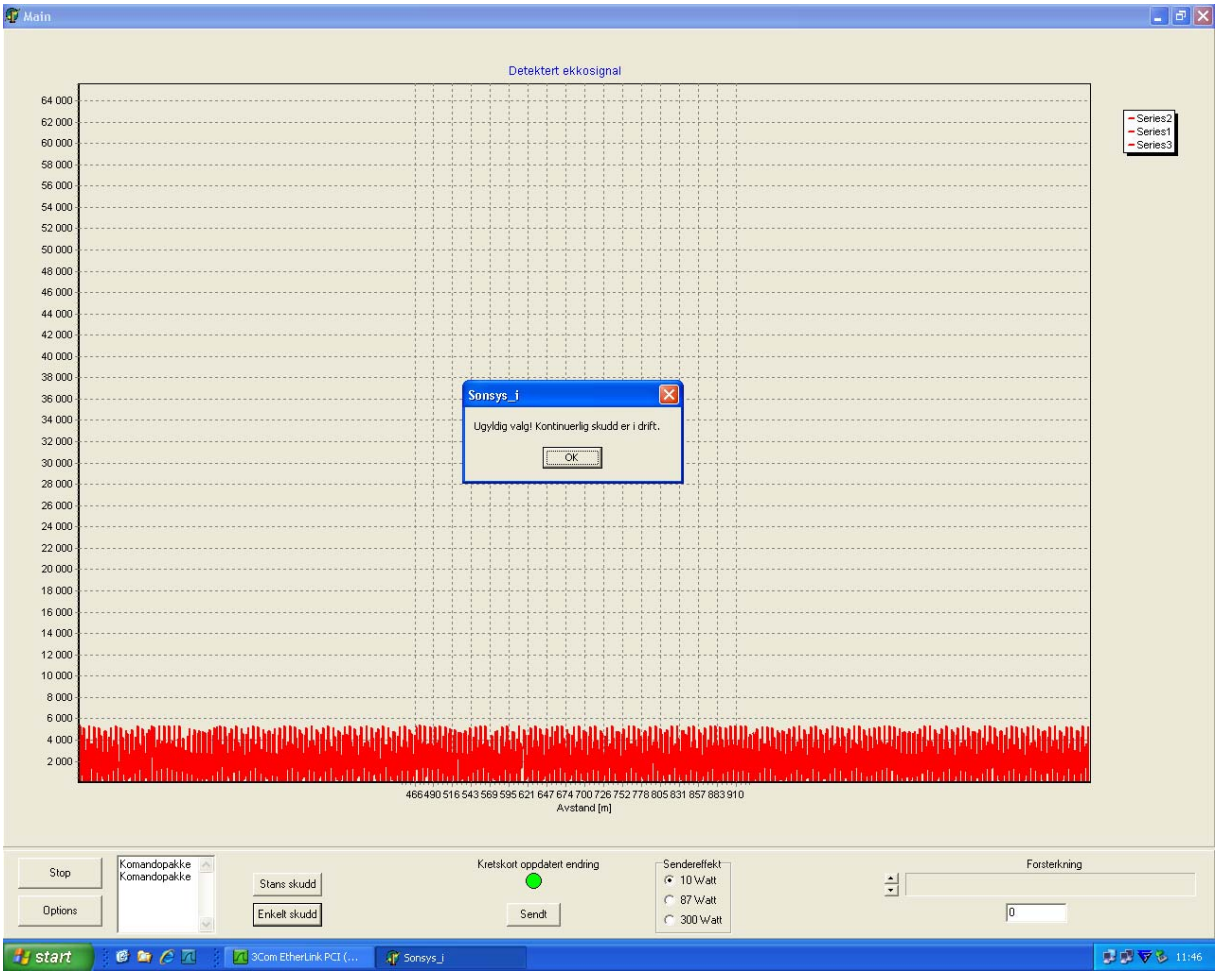
```
   brukerEndret();
   Bt_KontSkudd.Caption := 'Stans skudd';
   SendBuf[4]:=0;                          //Setter til kontinuerlig burst
   sendUDP();
  end
 else if (Bt_StartStop.Caption = 'Stop') and (Bt_KontSkudd.Caption = 'Stans skudd') then
  begin
    brukerEndret();
    Bt_KontSkudd.Caption := 'Kont. skudd';
    SendBuf[4]:=10;
    sendUDP();
  end;
end;

procedure TForm_Main.brukMottattUDP(Sender: Array of Byte; Datalengde : Integer);
var
 sampelArray : Array[0..2165] of Double;
 i,j:Integer;
 LowByte,
 HighByte: Byte;
 MergedWord1: Word absolute LowByte;
 MergedWord2: Word absolute HighByte;
 begin
 j:=6;
  if (MottaksBuffer[0]=sendBuf[0]) and
     (MottaksBuffer[1]=sendBuf[1]) and
     (MottaksBuffer[2]=sendBuf[2]) and
     (MottaksBuffer[3]=sendBuf[3]) and
     (MottaksBuffer[4]=sendBuf[4]) then
   begin
    Label2.Caption := 'Kretskort oppdatert endring';
    Sirkel.Brush.Color := clLime;
    Bt_MainSendUDP.Caption:= 'Sendt';
   end;
  if {(Datalengde=1450) and} (MottaksBuffer[5]=238) then
   begin
    Label2.Caption := 'Kretskort oppdatert endring';
    Sirkel.Brush.Color := clLime;
    Bt_MainSendUDP.Caption:= 'Sendt';
     if (MottaksBuffer[4]=1) then
      begin
       Series1.Clear;
      for i := 0 to (466) do {724}
       begin
        //Series1.Clear;
        HighByte := MottaksBuffer[j+1];
        LowByte := MottaksBuffer[j];
        sampelArray[i] := MergedWord1;
        sampelArray[i] := MergedWord2;
        j:=j+2;
        Series1.AddXY(i,sampelArray[i],(IntToStr(i)));
       end
      end
    else if (MottaksBuffer[4]=2) then
     begin
      Series2.Clear;
      for i := 466 to (932) do {724}
       begin
        //Series2.Clear;
        HighByte := MottaksBuffer[j+1];
        LowByte := MottaksBuffer[j];
        sampelArray[i] := MergedWord1;
        sampelArray[i] := MergedWord2;
        j:=j+2;
        Series2.AddXY(i,sampelArray[i],(IntToStr(i)));
       end
      end
     else if (MottaksBuffer[4]=3) then
      begin
       Series3.Clear;
       for i := 932 to (1398) do {724}
        begin
```

```
        HighByte := MottaksBuffer[j+1];
        LowByte := MottaksBuffer[j];
        sampelArray[i] := MergedWord1;
        sampelArray[i] := MergedWord2;
        j:=j+2;
        Series3.AddXY(i,sampelArray[i],(IntToStr(i)));
       end;
      end;
   end;
  if (MottaksBuffer[5] = 204) then
   begin
    Memo1.Lines.Add('Komandopakke');
    if(MottaksBuffer[0] = 85) and (MottaksBuffer[2] = 85) then {stopp kommando}
     begin
      Sirkel.Brush.Color := clBlue;
      Label2.Caption := '          Sonar stanset';
     end;
    end;
 end;

end.
```

PROPERTIES

```
unit Properties;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm_Option = class(TForm)
    Lb_OptionsSourcePort: TLabeledEdit;
    Bt_SaveAndClose: TButton;
    Lb_OptionsDestPort: TLabeledEdit;
    Lb_OptionsBurstFrek: TLabeledEdit;
    Lb_OptionsEnvelope: TLabeledEdit;
    St_OptionsKretskort: TStaticText;
    St_OptionsPC: TStaticText;
    procedure Bt_SaveAndCloseClick(Sender: TObject);
    procedure Lb_OptionsBurstFrekChange(Sender: TObject);
    procedure Lb_OptionsEnvelopeChange(Sender: TObject);
    procedure Lb_OptionsSourcePortChange(Sender: TObject);
    procedure Lb_OptionsDestPortChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Option: TForm_Option;
implementation
uses Main;
{$R *.dfm}

procedure TForm_Option.Bt_SaveAndCloseClick(Sender: TObject);
begin
 if(Lb_OptionsBurstFrek.Text='70') or (Lb_OptionsBurstFrek.Text='50') then
  begin
   Form_Main.verdiBurst();
   Form_Main.verdiRadioGroup();
   Form_Main.getGainVerdi();
   Visible:=false;
   if Form_Main.Bt_StartStop.Caption = 'Stop' then
   Form_Main.SendUDP();
  end
  else
   begin
    showMessage('Ugyldig valg! Velg 70 eller 50 KHz.')
   end;
end;

procedure TForm_Option.Lb_OptionsBurstFrekChange(Sender: TObject);
begin
 Form_Main.brukerEndret();
end;

procedure TForm_Option.Lb_OptionsEnvelopeChange(Sender: TObject);
begin
Form_Main.brukerEndret();
end;

procedure TForm_Option.Lb_OptionsSourcePortChange(Sender: TObject);
begin
Form_Main.brukerEndret();
end;

procedure TForm_Option.Lb_OptionsDestPortChange(Sender: TObject);
begin
Form_Main.brukerEndret();
end;

end.
```

Vedlegg 16 Brukerprogram med eksempel på feilmelding til bruker
Det kan observeres samplet signal opptegnet i rødt



Vedlegg 17 UDP-pakke konfigurasjon

UDP-pakke sendt fra Delphi	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Datakommandoer	Burstfrekvens	Envelope	Effekt	Forsterkning	Singel / Kont. Burst	Ikke i bruk
	[kHz]	[ant. Perioder]	[W]			
"Avslutt kretskort"	55	x	55	x	x	x
						cc = copyed and comfirmed
Mottatt UDP-pakke fra kretskort						
Satte verdier	Burstfrekvens	Envelope	Effekt	Forsterkning	Singel / kont. Burst	ee = ekko (sampel) pakke
	[kHz]	[ant. Perioder]	[W]			
Overrun rutine er kjørt	01	01	01	01	01	01