Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Informasjonsteknologi – Automatisering og signalbehandling	Vårsemesteret, 2016.... Åpen / ~~Konfidensiell~~
Forfatter: Jørn Tore Ørsland	…… (signatur forfatter)
Fagansvarlig: Morten Tengesdal Veileder(e): Morten Tengesdal Åge Andersen	
Tittel på masteroppgaven: Posisjonering i elektromagnetisk felt 2 Engelsk tittel: Positioning in electromagnetic fields 2	
Studiepoeng: 30	
Emneord: CAN,FFT,ADC,ARM,Embedded	Sidetall: 176 + vedlegg/annet: 118 Stavanger, 15/06-2016 dato/år

Sammendrag

Formålet med denne oppgaven er (*i*) å sammenligne og velge egnet magnetfeltsensor, (*ii*) konstruere og bygge et dataloggesystem som måler og logger magnetfeltmålingene og (*iii*) verifisere invariantene A_θ og Z_θ. Invariantene ble utforsket av Kristian Stangeland i hans oppgave [25] og der kom han frem til at disse er uavhengig av fasestrømmen og at de dermed kan benyttes for beskrive posisjonen i et magnetfelt.

Det er gjennom denne oppgaven designet et målesystem for å måle magnetfeltstyrken til et magnetfelt, produsert av en høyspentmast. Ved komponentvalg til dette målesystemet er det fokusert på komponenters egenskaper og ikke faktorer som kost og kompleksitet. Når magnetfeltsensoren HMC2003 ble valgt, var det tydelig at den ville kreve et betraktelig mer komplekst kretskort en hva som innledningsvis var skissert. Sensoren er en analog 3-akse magnetfeltsensor som krever betydelig analog signalbehangling for å opprettholde ønsket presisjon. Dette medførte at omfanget av den delen av oppgaven som omhandlet konstruksjon og bygging av dataloggesystemet økte betraktelig.

Det utviklede målesystem består av flere sensornoder og en prosesseringsnode. Det er valgt å knytte disse sammen via en felles galvanisk isolert CAN-buss. Magnetfeltmålingene vil utføres av sensornodene og prosesseringsnoden utfører frekvensanalyse av målingene. Basert på frekvensanalysen, beregner prosesseringsnoden invariantene til hver sensornode. Antall sensornoder som kan inngå i det komplette målesystem, vil kun begrenses av datalast på buss, samt begrensninger definert i standard CAN2.0B [9].

Det er gjennom eksperimenter vist at invariantene A_θ og Z_θ synes å gjennspeile en posisjonsendring bra. Standardavvikene til invariantentene ble basert på målinger i et eksperiment beregnet til $\sigma_{A_\theta} < 0.01°$ og $\sigma_{Z_\theta} < 0.4°$, ved magnetfeltmåling under en høyspentmast. Omregnet i prosent representerer standardavviket $\sigma_{A_\theta} \approx 0.6\%$ av invarianten A_θ og standardavviket $\sigma_{Z_\theta} \approx 0.18\%$ av invarianten Z_θ.

Basert på eksperimentene ble det tydelig at invariantene kan benyttes for å estimere den faktiske posisjonen i et magnetfelt. Algoritmer som Kalman-filter eller CMA-ES er begge godt egnet til å utføre posisjonsestimering basert på invariantene. Utviklingen av dette hører naturlig til sammen med utviklingen av prosesseringsnoden og er dermed foreslått til videre arbeid.

Forord

Jeg ønsker først å takke min samboer Veronica,mine 3 barn Julie, Jonas og Frida og mine foreldre for støtte, motivasjon og forståelse, gjennom 6 år med videreutdanning.

Videre ønsker jeg å takke Morten Tengesdal for god veiledning og støtte gjennom studiet ved UiS og som veileder av denne oppgaven.

Til slutt vil jeg takke Verico og da spesielt Åge Andersen og Kristian S.Stangeland, for oppgaven og hjelp underveis.

Innhold

Forkortelser

ADC Analog-to-digital converter

AMR Anisotropic Magneto Resisive

CMSIS Cortex Microcontroller Software Interface Standard

DAC Digital to Analog Converter

FFT Fast Fourier transform

IDE Integrated Development Environment

IIR Infinite impulse response

IWDG Independent Watchdog

SAR Successive-Approximation-Register

SPI Serial Peripheral Interface Bus

Kapittel 1

Introduksjon

Denne rapporten tar for seg design og konstruksjon av en magnetfeltsensor som danner grunnlaget for navigasjon i et magnetfelt, generert av 3 ledere i en høyspentmast. Det teoretiske grunnlaget for navigasjon i et magnetfelt ble grundig utforsket i Kristian S.Stangeland sin masteroppgave [25]. I oppgaven kom han frem til to invarianter som kan benyttes for å navigere en drone i et tids- og posisjons-varierende magnetfelt. Denne oppgaven bygger videre på dette arbeidet og har som hovedmål å bekrefte eller avkrefte at invariantene kan benyttes for presis navigasjon i et magnetfelt.

1.1 Bakgrunn

I dag utføres visuell sjekk av komponenter i en høyspentmast med bemannet helikopter. Som følge av utviklingen av droner er det ønskelig å kunne benytte en slik for denne inspeksjonen. Dette vil føre til reduserte kostnader og det vil forenkle jobben.

Dersom posisjon fra GPS benyttes alene, blir forventet avvik fra ønsket posisjon for stor. Det er derfor i masteroppgaven [25] utforsket de teoretiske mulighetene som finnes, ved å navigere i magnetfeltet generert av strømførende ledere i en høyspentmast.

Som følge av den teoretiske ligningene er det ønskelig å forsøke om dette lar seg gjøre i praksis. Denne oppgaven skal utforske om ligningene fra [25] kan brukes i praksis.

1.2 Mål for oppgaven

Mål som er ønskelig å oppnå med denne oppgaven kan deles inn i følgende deler

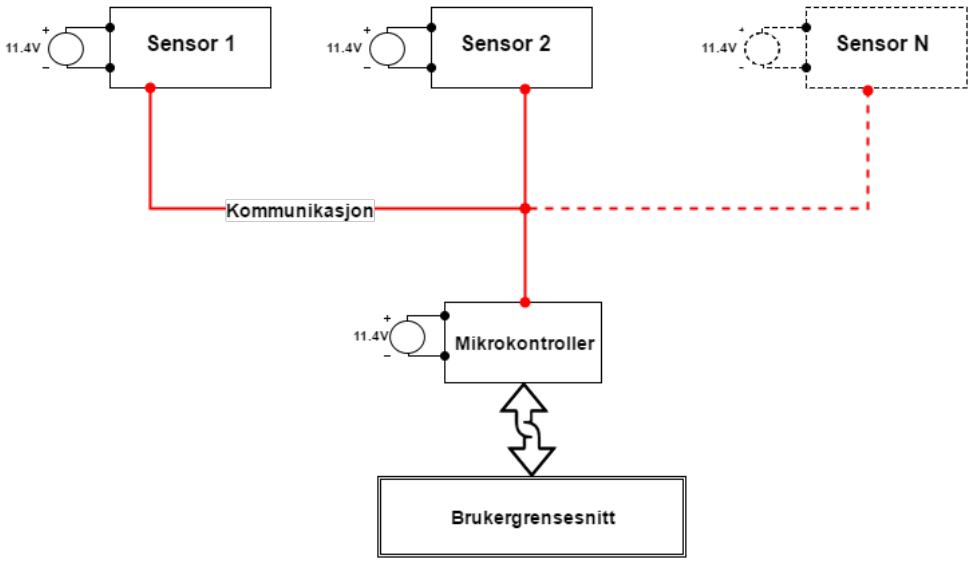
1) - Sammenlign sensorteknologier og deres nøyaktighet,linearitet og sensitivitet. På bagrunn av sammenligning, velges sensor.

2) - Konstruer og bygg et dataloggesystem som måler og logger magnetfelt.

3) - Verifiser ligningene fra Kristian S.Stangeland sin masteroppgave [25].

1.3 Overordnet systemstruktur

Systemstrukturen som illustreres i figur 1.1 viser en overordnet skisse over hvordan måle og posisjoneringssystemet er planlagt. Systemet som vises i figuren er tiltenkt å kunne monteres på en drone, for navigasjon i et magnetfelt. I design av dette systemet vil det derfor være sentralt å fokusere på både vekt og strømforbruk til komponentene som skal inngå.



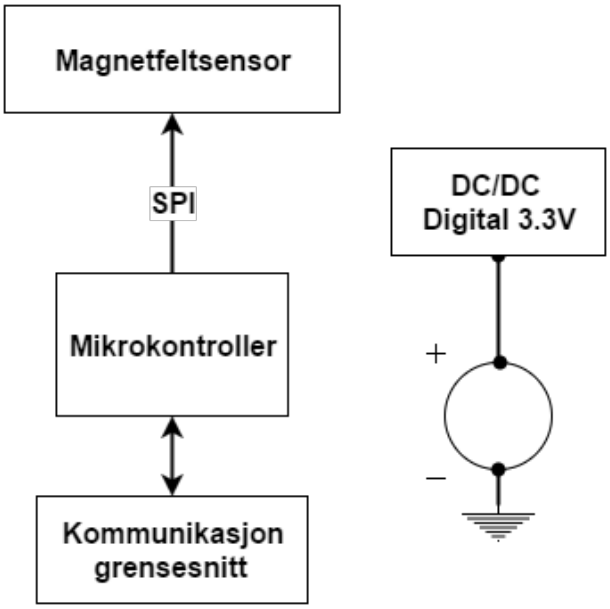
Figur 1.1: Overordnet systemstruktur.

I målesystemet er det tenkt at hver sensornode måler et forskjøvet magnetfelt i forhold til de andre sensornodene i målesystemet. Målingene sendes over valgt kommunikasjonsgrensesnitt, til en mikrokontroller med støtte for flyttallsberegning. Basert på forskjellen i de målte magnetfelt kan mikrokontrolleren utføre nødvendig prosessering av mottatte data og estimere den faktiske posisjonen, basert på ligninger i [25]. Det eksisterer utallige estimeringsalgoritmer som kunne vært egnet for løpende posisjonsestimering i denne oppgaven. Oppgaven [25] ble basert på CMA-ES algoritme og denne kunne vise til gode resultater. Bakdelen ved denne algoritmen er at den viste seg også å være svart ressurskrevende, selv for en moderne PC. I et mikrokontrollerbasert posisjoneringssystem vil det fokuseres på å finne en lettere estimeringsalgoritme som er egnet for å kjøre på systemer med begrensede ressurser.

Fra [25] vises det at presisjonen til posisjoneringssytemet er forventet å forbedres ved å øke antall sensorer. Systemet som skal designes bør dermed ha støtte for fremtidige utvidelser av antall sensorer som inngår i nettverket.

Måle og posisjoneringssytemet må også være i stand til å kommunisere med eksternt utstyr og programvare. Blokken brukergrensesnitt er tiltenkt å inneholde nevnt overgang mellom systemet som utvikles i denne oppgaven og øvrig utstyr.

Sensornode



Figur 1.2: Blokkskjema sensornode.

Figur 1.2 illustrerer blokkskjematisk hvordan hver sensornode er tenkt designet. Som det går frem av illustrasjonen, er sensoren delt inn i flere logiske grupperinger.

Øverst i illustrasjonen er sensor som måler magnetfeltstyrken i tre akser plassert. Denne skal med høy presisjon og tilstrekkelig båndbredde, måle det genererte magnetfeltet. Målingene sendes til intern mikrokontroller over en digital SPI eller I^2C bus. Målte magnetfeltdata sendes ubehandlet videre til en kraftigere mikrokontroller med støtte for flyttall, via valgt kommunikasjonsgrensesnitt. Mikrokontrolleren i målesystemet blir dermed tilegnet svært enkle operajoner og dette tillater valg av en enkel mikrokontroller med lavt strømforbruk.

Siden den planlagte sensornoden består utelukkende av digitale kretser, kreves kun en digital strømforsyning. Det er ikke nødvendig å ta hensyn til analoge kretser og krav til regulatorstøy er dermed begrenset. Svitsjet regulator med høyere virkningsgrad kan derfor med fordel benyttes.

1.4 Oppbygning av oppgaven

Stukturen til denne rapporten er som følger.

I kapittel 2 oppsummeres de mest sentrale ligningene fra masteroppgaven til Kristian S. Stangeland [25]. Dette gir en grunnleggende teoretisk forståelse for videre lesing.

I kapittel 3 blir ulike sensorteknologier sammenliknet og grunnleggende måleteori gjennomgås. Til slutt argumenteres det for valg av sensor.

I kapittel 4 blir komponenter som skal inngå i kretskort presentert, sammenliknet og argumentert for valg.

I kapittel 5 blir sentrale deler av programvaren gjennomgått og forklart.

I kapittel 6 blir ulike eksperimenter av magnetfeltmålinger utført og analysert.

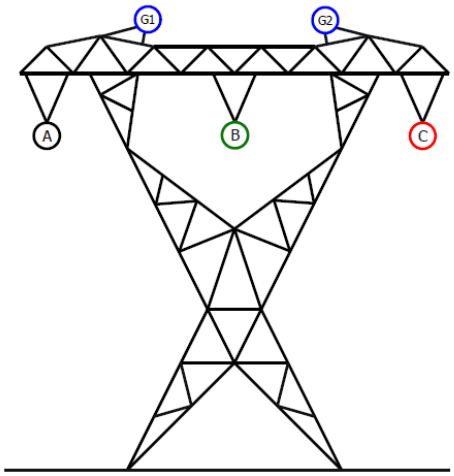
I kapittel 7 blir viktige resultater fra oppgaven analysert. Det gis også en beskrivelse av arbeidet som er lagt ned i oppgaven og forslag til videre arbeid.

I kapittel 8 blir oppgaven oppsummert og konkludert.

Kapittel 2

Teori

Vi skal i dette kapittelet se på teorien som ligger til grunn for å beregne posisjonen i et magnetfelt, generert av en høyspentmast. Denne teorien ble grundig utforsket av Kristian Stangeland i hans masteroppgave [25]. De mest sentrale resultatene og ligningene fra oppgaven hans, skal her oppsummeres. For å oppnå en dypere forståelse av invariantene og tilhørende teori, henvises det til [25].



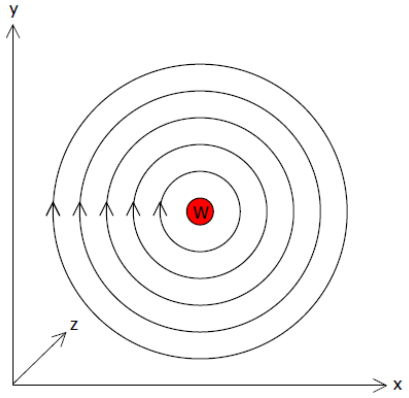
Figur 2.1: Eksempel på en høyspentmast med 3 ledere (A,B,C)
i et trefasesystem, med to jordledere på topp (G1,G2) [25].

Som innledningsvis nevnt, var målet for Kristian S. Stangeland sin oppgave å undersøke om det var mulig å beregne posisjonen til en drone, ved hjelp av magnetfelt generert av en høyspentmast. Et eksempel på en slik mast vises i figur 2.1. Oppgaven hans gikk videre ut på å argumentere for et nødvendig antall senorer og hvilken presisjon det kan forventes av de ulike konfigurasjoner.

Til slutt her skal vi se på mine simuleringer av forventet magnetisk feltstyrke. Fra disse vil vi finne disse svært nyttig for å kunne tallfeste forventet magnetfelt som befinner seg i nærheten av en høyspentmast. Simuleringene vil senere i oppgaven danne grunnlaget for flere sentrale valg.

2.1 Magnetfelt

Vi skal nå se på de ligninger som ligger til grunn for å beregne forventet magnetfelt.



Figur 2.2: Magnetisk felt av en uendelig lang ledning (W) parallell med z-akse [25].

I oppgaven [25] er det valgt å benytte Biot-Savartlov, ligning 2.1, som grunnlag for å beregne magnetfeltet. Ved hjelp av ligning 2.1 beregnes den magnetiske fluxtettheten,$\mathbf{B}$, i et punkt $\mathbf{p}$.

$$\mathbf{B}_n(\mathbf{r}) = \frac{\mu_0 I_w \hat{\mathbf{z}} \times (\mathbf{r} - \mathbf{r_w})}{2\pi |\mathbf{r} - \mathbf{r_w}|^2} \tag{2.1}$$

Hvor $\mu_0 = 4\pi \cdot 10^{-7} N/A^2$, I_w er den konstante strøm, $\hat{\mathbf{z}}$ er enhetsvektoren i strømretning, $\mathbf{r}$ posisjon til punktet, $\mathbf{r_w}$ er skjæringspunktet mellom kabel og x-y plan. Dette ble definert til $\mathbf{r} = (\mathbf{p_x}\hat{x} + \mathbf{p_y}\hat{y})$ og $\hat{\mathbf{z}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$

2.1.1 Magnetfelt rundt en leder

Det ble valgt å utvikle ligning 2.1 til kartesisk form. Dette ble gjort ved hjelp av følgende:

$$\hat{\mathbf{z}} \times (\mathbf{r} - \mathbf{r_w}) = \begin{bmatrix} r_y - rwy \\ -(r_x - r_w x \\ 0 \end{bmatrix}$$

$$\mathbf{d} = \mathbf{r} - \mathbf{r_w} = \begin{bmatrix} r_x - r_w x \\ r_y - y_w y \\ 0 \end{bmatrix}$$

Ved å kombinere disse, endte Stangeland opp med å kunne uttrykke den magnetiske flux på

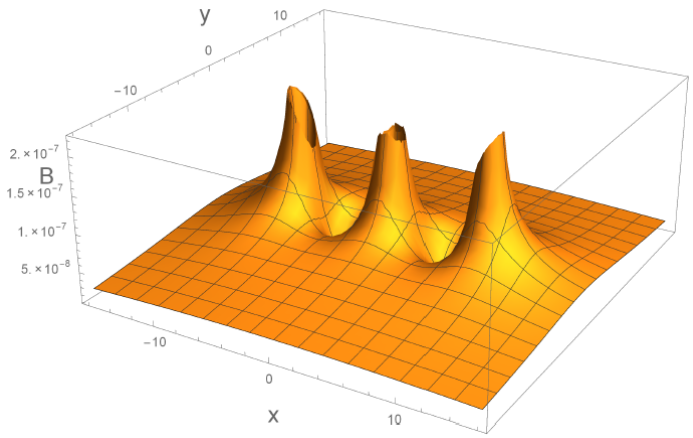
kartesisk form.

$$B_x = \frac{\mu_0 I_w d_y}{2\pi(d_x^2 + d_y^2)} \tag{2.2}$$

$$B_x = \frac{\mu_0 I_w d_x}{2\pi(d_x^2 + d_y^2)} \tag{2.3}$$

Ligningene 2.2 og 2.3 kan nå benyttes for å beregne den magnetiske flux rundt en enkelt leder.

2.1.2 Magnetfelt rundt N ledere



Figur 2.3: Feltstyrke rundt 3 ledere [25].

Magnetfeltet som skal måles består imidlertid av magnetfelt generert av flere ledere med faseforskyvning. Eksempel på et slikt felt illustreres i figur 2.3. Dette medførte at det var nødvendig å tilpasse ligningene slik at de også var gyldige i magnetfelt bestående av N ledere.

$$B_{nx} = \sum_{i=1}^{N} \frac{\mu_0 I_i d(i)_y}{2\pi(d(i)_x^2 + d(i)_y^2)} \tag{2.4}$$

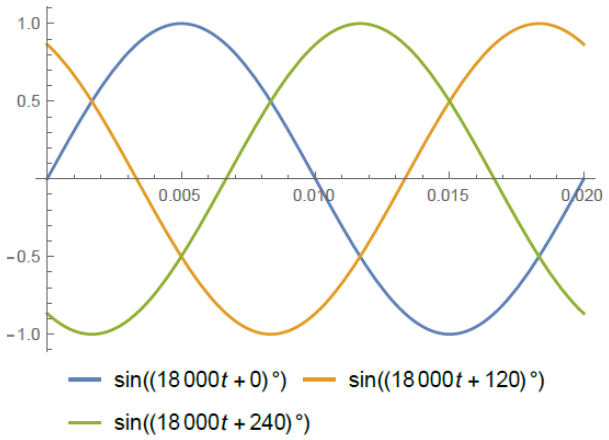
$$B_{yx} = -\sum_{i=1}^{N} \frac{\mu_0 I_i d(i)_x}{2\pi(d(i)_x^2 + d(i)_y^2)} \tag{2.5}$$

Ved hjelp av ligningene 2.4 og 2.5 kan feltstyrken beregnes i et gitt punkt, basert på N bidrag.

2.1.3 Vekselspenning

Det antas kjent at det norske forsyningsnettet benytter vekselspenning med frekvens på 50Hz. Strømmen i en leder kan dermed modelleres som:

$$I_i = A_i sin(\omega t + \varphi_i)$$



Figur 2.4: Strøm i 3 ledere i et trefasesystem [25].

Modellen for en leder kan videre utvikles til å også passe til et trefasenett. Fasestrømmen i et slikt nett illustreres i figur 2.4. I et slikt nett benyttes en faseforskyvning mellom hver enkelt leder på 120°. En modell basert på ligning ovenfor kan dermed defineres til:

$$
\begin{aligned}
I_1 &= I_p sin(\omega t + 0°) \\
I_2 &= I_p sin(\omega t + 120°) \\
I_3 &= I_p sin(\omega t + 240°)
\end{aligned}
$$

2.1.4 Vektorsum

Ved å erstatte I_i i ligningene 2.4 og 2.5 med modellen ovenfor ble det funnet et viktig resultat som benyttes videre i oppgaven.

$$
\begin{aligned}
B_{nx} &= \sum_{i=1}^{n} \frac{\mu_0 d(i)_y}{2\pi(d(i)_x^2 + d(i)_y^2)} I_p sin(\omega t + \varphi_i) \\
&= \sum_{i=1}^{n} A_{ix} sin(\omega t + \varphi_i) \\
&= A_x sin(\omega t + \delta_x) \qquad (2.6) \\
B_{ny} &= \sum_{i=1}^{n} \frac{\mu_0 d(i)_x}{2\pi(d(i)_x^2 + d(i)_y^2)} I_p sin(\omega t + \varphi_i) \\
&= \sum_{i=1}^{n} A_{iy} sin(\omega t + \varphi_i) \\
&= A_y sin(\omega t + \delta_y) \qquad (2.7)
\end{aligned}
$$

Ved å benytte *Harmonic Addition Theorem* er det mulig å beregne resulterende amplitude og fase når N sinussignal summeres.

$$\Psi = \sum_{i=1}^{N} A_i cos(\omega t + \delta_i) = Acos(\omega t + \delta)$$

Resulterende amplitude,A, etter å ha benyttet teoremet blir:

$$\begin{aligned} A^2 &= \sum_{i=1}^{N}\sum_{j=1}^{N} A_i A_j cos(\delta_i - \delta_j) \\ &= \sum_{i=1}^{N} A_i^2 + 2\sum_{i=1}^{N}\sum_{j>i}^{N} A_i A_j cos(\delta_i - \delta_j) \end{aligned} \tag{2.8}$$

Resulterende fase etter summasjonen blir:

$$tan\delta = \frac{\sum_{i=1}^{N} A_i sin(\delta_i)}{\sum_{i=1}^{N} A_i cos(\delta_i)} \tag{2.9}$$

Ved å benytte ligning 2.8 kan amplituden til $N = 3$ ledere beregnes ved:

$$A_x^2 = \sum_{i=1}^{3} A_{ix}^2 + 2\left[A_{1x}A_{2x}cos(\delta_1 - \delta_2) + A_{1x}A_{3x}cos(\delta_1 - \delta_3) + A_{2x}A_{3x}cos(\delta_2 - \delta_3)\right]$$

$$A_x^2 = A_{1x}^2 + A_{2x}^2 + A_{3x}^2 - A_{1x}A_{2x} - A_{1x}A_{3x} - A_{2x}A_{3x} \tag{2.10}$$

Tilsvarende som for x-aksen, kan amplituden tilhørende y-aksen finnes ved:

$$A_y^2 = A_{1y}^2 + A_{2y}^2 A_{3y}^2 - A_{1y}A_{2y} - A_{1y}A_{3y} - A_{2y}A_{3y} \tag{2.11}$$

Hvor amplituden til magnetfeltet A_i, er definert ved:

$$A_{ix} = \frac{\mu_0 d(i)_y I_p}{2\pi(d(i)_x^2 + d(i)_y^2)}$$

$$A_{iy} = -\frac{\mu_0 d(i)_x I_p}{2\pi(d(i)_x^2 + d(i)_y^2)}$$

Ved å sette inn i ligning 2.9 kan resulterende fase beregnes ved:

$$tan\delta_x = \frac{A_{1x}sin(\Psi) + A_{2x}sin(\Psi + 120°) + A_{3x}sin(\Psi + 240°)}{A_{1x}cos(\Psi) + A_{2x}cos(\Psi + 120°) + A_{3x}cos(\Psi + 240°)} \tag{2.12}$$

$$tan\delta_y = \frac{A_{1y}sin(\Psi) + A_{2y}sin(\Psi + 120°) + A_{3y}sin(\Psi + 240°)}{A_{1y}cos(\Psi) + A_{2y}cos(\Psi + 120°) + A_{3y}cos(\Psi + 240°)} \tag{2.13}$$

2.2 Invarianter

Kristian S. Stangeland kom frem til to invarianter som kan benyttes for å bestemme posisjonen i et magnetfelt. Invariantene er

- **Vinkel til amplitude: -** A_θ
- **Faseforskjell: -** Z_θ

I Stangeland sin oppgave kom han frem til at begge disse invariantene er uavhengige av linjestrømmen i lederene.

Parameterne som inngår i å beregne invariantene $A_x, \delta_x, A_y, \delta_y$, vil i praksis finnes fra frekvensanalyse av målt magnetfelt. For å få en litt dypere forståelse av hva som ligger til grunn for invariantene, er det i tillegg valgt å også ta med fremgangsmåten for teoretisk beregning av disse.

2.2.1 Vinkel til amplitude

Den første invarianten er vinkelen på vektoren $\begin{bmatrix} A_x & A_y \end{bmatrix}^T$. Denne vinkelen er i oppgaven definert som A_θ og beregnes ved:

$$A_\theta = atan2(A_y, A_x) \tag{2.14}$$

Ved å benytte *Harmonic Addition Theorem* er A_x og A_y definert ved:

$$A_x = \sqrt{N_{1x}^2 + N_{2x}^2 + N_{3x}^2 - N_{1x}N_{2x} - N_{1x}N_{3x} - N_{2x}N_{3x}}$$
$$A_y = \sqrt{N_{1y}^2 + N_{2y}^2 + N_{3y}^2 - N_{1y}N_{2y} - N_{1y}N_{3y} - N_{2y}N_{3y}}$$

Hvor N_{ix} og N_{iy} er magnetfeltstyrken i hver sin respektive retning, med strømmen $I_p = 1A$. Disse størrelsene defineres ved:

$$N_{ix} = \frac{\mu_o d(i)_y}{2\pi(d(i)_x^2 + d(i)_y^2)}$$
$$N_{iy} = -\frac{\mu_o d(i)_x}{2\pi(d(i)_x^2 + d(i)_y^2)}$$

I ligningene ovenfor er $d(i)_x$ og $d(i)_y$ avstanden fra punktet,p, som magnetfeltstyrken skal be-

regnes for, til sentrum av leder u_i. Fra denne informasjonen kan det defineres:

$$d(i)_x = u_{ix} - p_x$$
$$d(i)_y = u_{iy} - p_y$$

2.2.2 Faseforskjell

Den andre invarianten, Z_θ, er faseforskjellen mellom x-akse og y-akse. Denne faseforskjellen beregnes ved hjelp av definisjonen *AngleDiff*:

$$Z_\theta = AngleDiff(\delta_y, \delta_x)$$

I oppgaven ble *AngleDiff* definert som:

$$AngleDiff(a,b) = \begin{cases} 2\pi - a + b & , \quad a > b \\ b - a & , \quad a < b \end{cases} \tag{2.15}$$

Også her er *Harmonic Addition Theorem* anvendt for å beregne variablene som benyttes i beregning av invariantene. Som kjent er faseforskjellen 120° mellom lederene i et trefasesystem. Denne informasjonen ble benyttet for å definere hjelpefunksjonene *f()* og *g()*.

$$f(d,\Psi) = N_{1d}sin(\Psi) + N_{2d}sin(\Psi + 120^\circ) + N_{3d}sin(\Psi + 240^\circ)$$
$$g(d,\Psi) = N_{1d}cos(\Psi) + N_{2d}cos(\Psi + 120^\circ) + N_{3d}cos(\Psi + 240^\circ)$$

Hvor d representerer aksen og Ψ er den ukjente felles faseforskyvning som følge av at magnetfeltmålingene kan starte ved vilkårlig posisjon i signalperioden. Funksjonen *AngleDiff()* vil kansellere dette biddraget og det kan dermed settes lik 0.

Fasen på magnetfeltet kan nå beregnes til:

$$\begin{aligned} \delta_x &= atan2(f(x,0), g(x,0)) \\ &= atan2(\sqrt{3}(N_{2x} - N_{3x}), 2N_{1x} - N_{2x} - N_{3x}) \\ \delta_y &= atan2(f(y,0), g(y,0)) \\ &= atan2(\sqrt{3}(N_{2y} - N_{3y}), 2N_{1y} - N_{2y} - N_{3y}) \end{aligned}$$

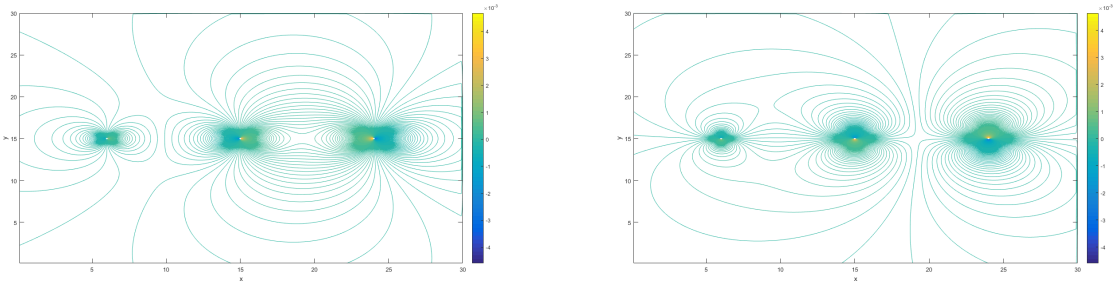
2.3 Simulering av magnetfelt

Vi skal i dette delkapittelet se på hvilke magnetfelt som kan forventes å befinne seg omkring lederene i en høyspentmast. Det er valgt å beregne dette ved å implementere ligningene 2.4 og 2.5 i MATLAB.

For å få et visuelt bilde av magnetfeltet rundt lederene er det valgt å se på konturlinjene til feltstyrkene ved følgende konfigurasjon:

$$I_1 = 2400sin(\omega t + 0°)$$
$$I_2 = 2400sin(\omega t + 120°)$$
$$I_3 = 2400sin(\omega t + 240°)$$

I simuleringen er de respekive lederene plassert med 9 meter avstand til nærmeste leder. Dette tilsvarer hva vi finner i figur 2.1.



B_x B_y

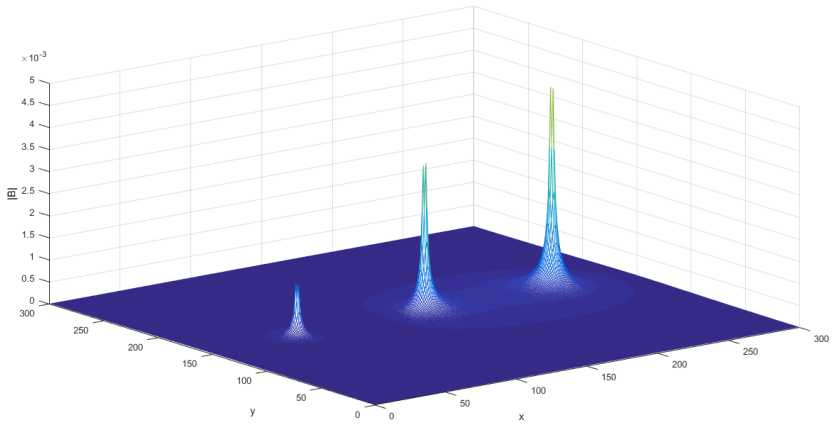
Figur 2.5: Konturlinjer rundt 3 ledere.

Som forventet viser figur 2.5 størst endring i feltstyrken i umiddelbar nærhet av lederene. Figuren viser også at det genererte magnetfeltet strekker seg betydelig lengre enn 10 meter, som i simuleringen er definert til ytre grense. Dette innebærer at det er mulig å detektere og da også navigere etter magnetfeltet et stykke bort fra lederene.

Fra figur 2.6 blir det tydelig at de høyeste feltstyrkene befinner seg i nærheten av hver enkelt leder. Når det på et senere tidspunkt skal velges egnet magnetfeltsensor, er det viktig å ha informasjon vedrørende høyeste og laveste felt som skal måles. Det er derfor valgt å utføre en simulering hvor disse parameterene tallfestes.

2.3.1 Forventet magnetfelt

For å beregne forventet magnetfelt er det også her benyttet MATLAB, med en implementasjon av ligningene 2.4 og 2.5. I simuleringene er det valgt å ikke ta med feltstyrkene i umiddelbar



Figur 2.6: Magnitude til magnetfelt rundt 3 ledere.

nærhet av faselederene. I implementasjonen er dette gjort ved å ikke beregne feltstyrke for posisjoner nærmere enn 2.5 meter fra faseleder. Dette resulterer i et rektangulært område hvor alle 4 sider er 2.5 meter fra nærmeste faseleder. Dersom ikke annet er nevt, er alle målingene utført på samme tidspunkt. Resultatene fra simuleringene oppsummeres i tabellene 2.1 og 2.2. Disse tabellene vil gjennom oppgaven danne grunnlaget for flere viktige valg.

I tabell 2.1 vises en oversikt over typiske feltstyrker,$\mathbf{B}$, ved ulik strøm som befinner seg i nærheten av en mast. Tabellen viser også beregnet forventningsverdi og standardavvik tilhørende de ulike linjestrømmene.

I tillegg til oversikten over størrelsen på feltstyrker som må kunne måles, er det også viktig å etablere en oversikt over hvor små endringene i feltstyrken er, ved endring i posisjon. Informasjonen som dette gir, danner grunnlaget for krav til sensitivitet til sensor og tilhørende signalbehandling.

Informasjonen vedrørende forventningsverdien og standardavviket, til en posisjonsendring på 1.0cm i magnetfeltet finnes i tabell 2.2.

Strøm	$B_{x,max}$	μ_{B_x}	σ_{B_x}	$B_{y,max}$	μ_{B_y}	σ_{B_y}
100	6.06μ	0.862μ	$3.92n$	3.35μ	0.811μ	$5.34n$
200	12.1μ	1.72μ	$7.84n$	6.70μ	1.62μ	$10.7n$
300	18.2μ	2.59μ	$11.8n$	10.0μ	2.43μ	$16.0n$
400	24.2μ	3.45μ	$15.7n$	13.4μ	3.25μ	$21.4n$
500	30.3μ	4.31μ	$19.6n$	16.7μ	4.06μ	$26.7n$
600	36.3μ	5.17μ	$23.5n$	20.1μ	4.87μ	$32.1n$
700	42.4μ	6.03μ	$27.4n$	23.4μ	5.68μ	$37.4n$
800	48.4μ	6.90μ	$31.3n$	26.8μ	6.49μ	$42.8n$
900	54.5μ	7.76μ	$35.3n$	30.1μ	7.30μ	$48.1n$
1000	60.6μ	8.62μ	$39.2n$	33.5μ	8.11μ	$53.4n$
1100	66.6μ	9.48μ	$43.1n$	36.8μ	8.93μ	$58.8n$
1200	72.7μ	$10.3\ \mu$	$47.0n$	40.2μ	9.74μ	$64.1n$
1300	78.7μ	11.2μ	$50.9n$	43.5μ	10.5μ	$69.5n$
1400	84.8μ	12.1μ	$54.8n$	46.9μ	11.4μ	$74.8n$
1500	90.8μ	12.9μ	$58.8n$	50.2μ	12.2μ	$80.2n$
1600	96.9μ	13.8μ	$62.7n$	53.6μ	13.0μ	$85.5n$
1700	103μ	14.7μ	$66.6n$	56.9μ	13.8μ	$90.9n$
1800	109μ	15.5μ	$70.5n$	60.3μ	14.6μ	$96.2n$
1900	115μ	16.4μ	$74.4n$	63.6μ	15.4μ	$102n$
2000	121μ	17.2μ	$78.4n$	67.0μ	16.2μ	$107n$
2100	127μ	18.1μ	$82.3n$	70.3μ	17.0μ	$112n$
2200	133μ	19.0μ	$86.2n$	73.7μ	17.9μ	$118n$
2300	139μ	19.8μ	$90.1n$	77.0μ	18.7μ	$123n$
2400	145μ	20.7μ	$94.0n$	80.4μ	19.5μ	$128n$

Tabell 2.1: Typiske feltstyrker i nærheten av en mast.

	B_x				B_y			
Strøm	μ_x	σ_x	μ_y	σ_y	μ_x	σ_x	μ_y	σ_y
100	$1.68n$	$0.428p$	$1.77n$	$0.583p$	$1.77n$	$0.583p$	$1.68n$	$0.428p$
200	$3.37n$	$0.856p$	$3.54n$	$1.17p$	$3.54n$	$1.17p$	$3.37n$	$0.856p$
300	$5.05n$	$1.28p$	$5.31n$	$1.75p$	$5.31n$	$1.75p$	$5.05n$	$1.28p$
400	$6.74n$	$1.71p$	$7.09n$	$2.33p$	$7.08n$	$2.33p$	$6.74n$	$1.71p$
500	$8.42n$	$2.14p$	$8.86n$	$2.92p$	$8.85n$	$2.91p$	$8.42n$	$2.14p$
600	$10.1n$	$2.57p$	$10.6n$	$3.50p$	$10.6n$	$3.50p$	$10.1n$	$2.57p$
700	$11.8n$	$3.00p$	$12.4n$	$4.08p$	$12.4n$	$4.08p$	$11.8n$	$2.99p$
800	$13.5n$	$3.42p$	$14.2n$	$4.66p$	$14.2n$	$4.66p$	$13.5n$	$3.42p$
900	$15.2n$	$3.85p$	$15.9n$	$5.25p$	$15.9n$	$5.25p$	$15.2n$	$3.85p$
1000	$16.8n$	$4.28p$	$17.7n$	$5.83p$	$17.7n$	$5.83p$	$16.8n$	$4.28p$
1100	$18.5n$	$4.71p$	$19.5n$	$6.41p$	$19.5n$	$6.41p$	$18.5n$	$4.71p$
1200	$20.2n$	$5.13p$	$21.3n$	$7.00p$	$21.2n$	$7.00p$	$20.2n$	$5.13p$
1300	$21.9n$	$5.56p$	$23.0n$	$7.58p$	$23.0n$	$7.58p$	$21.9n$	$5.56p$
1400	$23.6n$	$5.99p$	$24.8n$	$8.16p$	$24.8n$	$8.16p$	$23.6n$	$5.99p$
1500	$25.3n$	$6.42p$	$26.6n$	$8.75p$	$26.6n$	$8.74p$	$25.3n$	$6.42p$
1600	$27.0n$	$6.85p$	$28.3n$	$9.33p$	$28.3n$	$9.33p$	$26.9n$	$6.84p$
1700	$28.6n$	$7.27p$	$30.1n$	$9.91p$	$30.1n$	$9.91p$	$28.6n$	$7.27p$
1800	$30.3n$	$7.70p$	$31.9n$	$10.5p$	$31.9n$	$10.5p$	$30.3n$	$7.70p$
1900	$32.0n$	$8.13p$	$33.7n$	$11.1p$	$33.6n$	$11.1p$	$32.0n$	$8.13p$
2000	$33.7n$	$8.56p$	$35.4n$	$11.7p$	$35.4n$	$11.7p$	$33.7n$	$8.56p$
2100	$35.4n$	$8.99p$	$37.2n$	$12.2p$	$37.2n$	$12.2p$	$35.4n$	$8.98p$
2200	$37.1n$	$9.41p$	$39.0n$	$12.8p$	$38.9n$	$12.8p$	$37.0n$	$9.41p$
2300	$38.7n$	$9.84p$	$40.7n$	$13.4p$	$40.7n$	$13.4p$	$38.7n$	$9.84p$
2400	$40.4n$	$10.3p$	$42.5n$	$14.0p$	$42.5n$	$14.0p$	$40.4n$	$10.3p$

Tabell 2.2: Forventningsverdi og standardavvik til en posisjonsendring på 1.0cm

Kapittel 3

Måle magnetisk feltstyrke

I dette kapittelet vil jeg presentere og sammenlikne ulike sensorer for kvantifisering av det magnetiske feltet **B**, som gitt i [25]:

$$\mathbf{B_n}(\mathbf{r}) = \sum_{i=1}^{N} \frac{\mu_0 I_i \hat{\mathbf{z}}_\mathbf{i} \times (\mathbf{r} - \mathbf{r_i})}{2\pi |\mathbf{r} - \mathbf{r_i}|^2} \tag{3.1}$$

Som sammenlikningen i tabell 3.1 viser, er det kun et lite utvalg av sensorene som tilfredsstiller kravene. I kapittel 3.2 gis det en forenklet presentasjon av de sensorteknologiene som kan synes å tilfredsstille kravene til sensor.

I jakten på egnet sensor er det viktig å ta hensyn til hva sensoren er ment for. Er den designet for å være i en mobiltelefon/nettbrett er det gjerne lavere krav som stilles en hva som synes naturlig i denne oppgaven.

3.1 Krav til sensor

For å kvantisere magnetfeltet generert av en strømmast, er det flere sentrale krav til sensor som må oppfylles. Disse kravene belyses nedenfor. Vi starter med å se på hvordan de ulike teknologiene kvantiserer magnetfeltet samt typisk båndbredde.

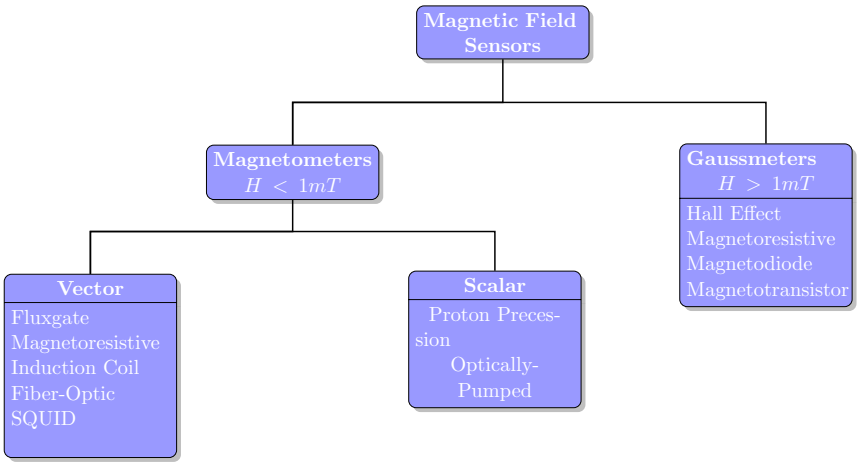
a) Måler vektorfelt

Dersom det ses bort fra oppløsning og måleområde kan magnetfeltsensorer deles inn i to logiske kategorier[16]. Hvordan sensoren måler og presenterer magnetfeltet bestemmer hvilken av de to kategoriene sensoren tilhører. Nedenfor blir kategoriene forklart nærmere.

1) - **Vektor**. Dersom det ønskes informasjon om feltstyrken til de ulike aksene, benyttes en magnetfeltsensor som måler vektorfelt. Vi vil med denne kunne måle separate verdier for feltstyrkene B_x, B_y, B_z .

2) - **Skalar**. Ønskes kun informasjon om feltstyrken til det totale magnetfeltet kan det benyttes sensor som måler feltets magnitude. Denne kategorien sensorer gir en verdi som representerer den totale feltstyrken B, der $B = \sqrt{B_x^2 + B_y^2 + B_z^2}$.

I figur 3.1 deles også de ulike teknologiene ytterligere inn i to kategorier,magnetometer og gaussmeter. Disse kategoriene representerer hver sensors typiske måleområde. Magnetometere er her de sensorer som måler felt mindre enn $1mT$ og gaussmetere måler felt større enn $1mT$.



Figur 3.1: Kategorier av magnefeltsensorer

Som vi ser fra figur 3.1 vil ikke Hall-effekt teknologien bli vurdert i denne oppgaven. Denne typen sensor måler total fluxdensitet og på grunn av deres lave sensitivitet og metningsproblemer fra jordens eget magnetfelt, blir de skjelden benyttet til å måle magnetfelt fra høyspentmaster [8]. Hall-effekt blir derimot i stor grad benyttet til å måle kraftigere magnetfelt.

Videre ser vi av figur 3.1 at det er flere ulike teknologier å vurdere før det kan velges sensor teknologi. I denne oppgaven er det kun de sensorene som kategoriseres som vektorfeltsensor som tas med i betraktning. Dermed begrenses utvalget av teknologier til 5 stk som må vurderes, for så å finne den optimale sensor. Disse 5 teknologiene er Fluxgate, magnetoresistor, induction coil, fiberoptikk og SQUID.

b) Båndbredde

Det anses som allment kjent at forsyningsnettet i Norge har en normalfrekvens på 50Hz, f_s. Som vist i [25] har også magnetfeltet denne frekvensen. Når dette magnetfeltet skal kvantiseres må, ifølge samplings teoremet [4], båndbredden f_b til sensoren være $f_{signal} < 2f_{b,sensor}$. Når vi leter

etter egnet sensor blir dette ett av det viktigste kravet som må oppfylles.

Teknologi	Båndbredde [Hz]
Flux Gate	dc to $2 \cdot 10^3$
Magnetoresistive	dc to 10^7
Search Coil	10^{-1} to 10^6
SQUID	dc to 5

Tabell 3.1: Typisk sensorkarakteristikk

Tabell 3.1 viser at det er 3 sensorteknologier som har høy nok båndbredde til å benyttes i denne oppgaven. Disse 3 er Flux Gate, Magnetoresistive og search coil. Da det ikke er ønskelig å designe selve magnetfeltsensoren, blir ikke sistnevnte med i videre vurderinger. Vi har da tilgjengelig to sensorteknologier som det må ses nærmere på for å komme frem til den optimale magnetfelt sensor. For å få en bedre forståelse av virkemåten til disse sensorene, blir grunnleggende teori behandlet i kapittel 3.2.

3.2 Sensorteknologi

Felles for disse er at det skilles mellom to ulike kategorier sensorer, vektor og absoluttverdi eller skalar. Magnetfeltet som skal kvantiseres i denne oppgaven krever førstnevnte teknologi. Teoretisk grunnlag hovedsakelig hentet fra dokumentet [16]. Jeg vil her behandle de sensorteknologiene som anses som nyttige i forhold til problemstillingen i denne oppgaven.

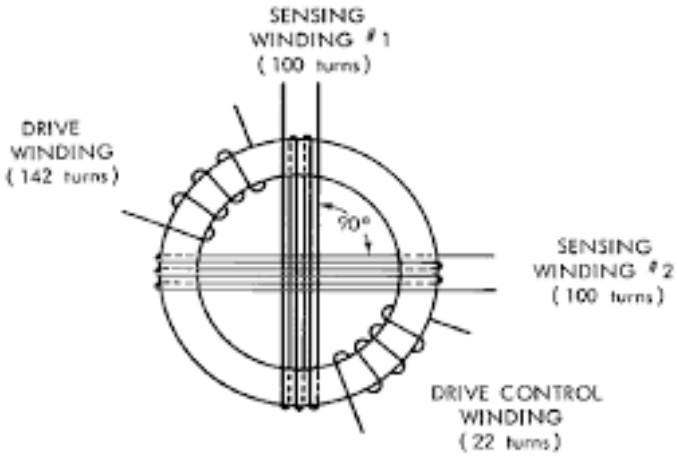
3.2.1 Fluxgate magnetometer

Figur 3.2: Fluxgatesensor fra Autonnic Research Limited

Fluxgate magnetometeret er en utprøvd sensor for å måle det magnetiske feltet. Denne teknologien benyttes i mange applikasjoner både på jorda så vel som i verdensrommet. Sensoren krever generelt lite energi og anses som robust og pålitelig og samtidig er den i stand til å måle påtrykket magnetfelt som en vektorkomponent. Denne sensorteknologien er oppgitt med et re-

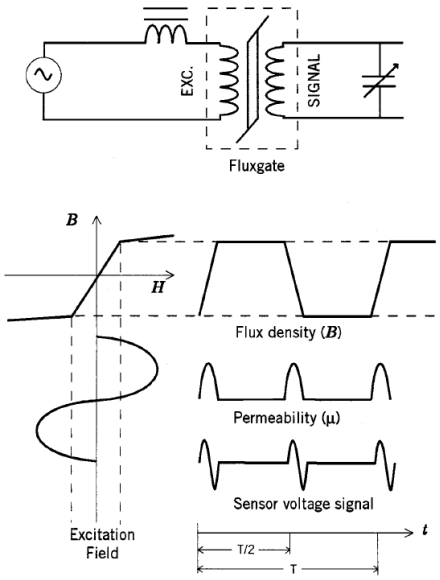
lativt stort arbeidsområde som kan spenne seg fra 0.1nT til 1mT, med en båndbredde fra DC til flere kHz.

Virkemåte



Figur 3.3: Design av et 2-D fluxgate magnetometer

Hjertet i magnetometeret er selve fluxgaten som konvereterer et magnetisk felt til en elektrisk spenning. Ringkjernen er konstruert med en tynn ferromagnetisk kjerne som lett lar seg gå i metning, figur 3.3. Denne kjernen kan være fast eller flytende, se figur 3.2. Fordelen med sistnevnte er at denne konstruksjonen kompenserer noe for bevegelse i det horisontale plan når en ønsker å måle et horisontalt magnetfelt. Som vises i figur 3.3 og 3.2 er sensorens eksiteringsspole plassert innerst rundt selve den ferromagnetiske kjernen. Spolene som måler magnetfeltet er plassert ytterst og ortogonalt på hverandre for å måle xy-planet .



Figur 3.4: Eksitasjonsfelt til et fluxgate magneometer

Figur 3.4 viser en vekselspenning som påtrykkes eksitasjonspolen og et magnetfelt settes opp. Dette magnetfeltet får fluxen i den ferromagnetiske kjernen til å periodisk gå i metning. Når kjernen er i transient tilstand mellom metning, har den en mye høyere permabilitet enn luft. Derimot når kjernen er i metning har den en permabilitet tilsvarende luft. På utsiden av kjernen er selve sensorviklingene plassert og en spenning som representerer påtrykket magnetfelt vil induseres i disse. Dersom det ikke påtrykkes noe eksternt magnetfelt vil flux endringen i disse være null. Derimot hvis sensoren påtrykkes et magnetfelt vil det hver gang kjernen er i transienttilstand, vil fluxen i kjernen indusere en spenning på sensorens utganger. I følge Farady's law, vil en fluxendring indusere en spenning i sensorene. Denne spenningen er proporsjonal med fluxendringen i kjernen.

For lavfrekvente magnetiske felt er denne spenningen gitt av:

$$e(t) = nA\frac{d(\mu_0\mu_e H)}{dt} = nA\mu_0 H\frac{d(\mu_e(t))}{dt} \tag{3.2}$$

Hvor

H - Amplitude av målt magnetfelt.

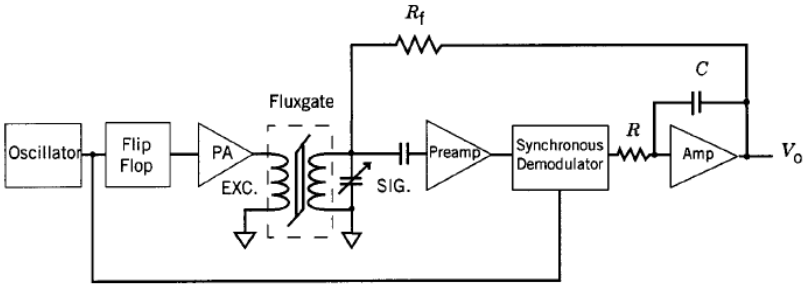
n - Antall vindinger.

A - Areal.

$\mu_e(t)$ - Effektiv relativ permabilitet til kjernen.

Når kjernepermabiliteten veksler fra høy til lav, induseres spenningspulser i sensorspolene som har amplitude proportional til magnituden av magnetfeltet og en fase som indikerer retningen

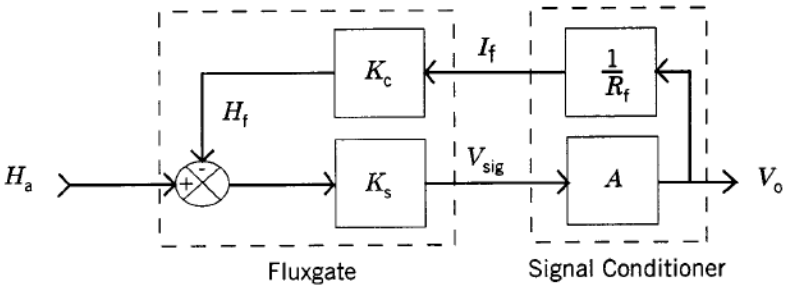
til feltet. En kan legge merke til på figur 3.4 at signalfrekvensen er den dobbelte av eksiteringsfrekvensen. Dette skyldes at kjernen går fra metning-til-metning to ganger hver eksiteringsperiode,T.



Figur 3.5: Typisk tilbakekobling av fluxgate

Presise og stabile magnetometere tilbakekobler det magnetisk felt. Et forenklet skjema er vist i figur 3.5. På venstre side for fluxgaten er eksitasjonskrets. Denne kretsen består av en oscillator som er tunet til den dobbelte eksitasjonsfrekvensen, en flip-flop som halvverer frekvensen før signalet forsterkes tilstrekkelig for å drive eksitasjonsspolen til sensoren. På høyre side av sensoren blir signal fra sensoren først forsterket for deretter å bli synkron demodulert ved hjelp av oscillatorens signal som referanse. Det forsterkede signalet blir deretter integrert og forsterket, før det gjennom resistor R_f blir ført tilbake til sensorens utgang. Å tilbakekoble signalet bidrar til at feltet i sensoren holdes nær null og dermed i den lineære delen av magnitiseringskurven til den ferromagnetiske kjernen.

Flytskjema for fluxgate magnetometeret er gitt i figur 3.6 og ved hjelp av denne forklares tilbakekoblingen.



Figur 3.6: Blokk diagram

Differansen mellom feltet H_a fra sensoren og tilbakekoblingen H_f blir konvertert til en spenning V_{sig}, gitt av transferfunksjon K_s. Spenningen V_{sig} blir så forsterket gjennom blokken A som gir utgangssignalet V_o fra sensoren. Utgangssignalet V_o blir så konvertert til en strøm,I_f, gjennom resistoren R_f for så bli tilbakekoblet til sensorens utgang.

Transferfunksjonen for magnetometeret er:

$$\frac{V_o}{H_a} = \frac{AK_s}{1 + \frac{K_c A K_s}{R_f}} \tag{3.3}$$

Forsterkningen, A, i blokkdiagrammet er generelt svært høy og en kan dermed forenkle transferfunksjonen 3.3 til:

$$\frac{V_o}{H_a} = \frac{R_f}{K_c} \tag{3.4}$$

3.2.2 Magnetoresistiv sensor

Magnetoresistans er evnen til å endre resistans, i en halvleder eller leder, som følge av påtrykt eksternt magnetfelt. Polariteten til det påtrykte magnetfelt vil bestemme hvorvidt resistansendringen blir positiv eller negativ. Videre skilles det mellom transversal og longitidinal magnetoresistans. Forskjellen mellom disse er at strømmen går på tvers av magnetfeltet ved førstnevnte og parallellt ved sistnevnte. Sensorer basert på denne teknologien kan oppnå høy grad av linearitet.

Virkemåte

De fysisk underliggende egenskapene til magnoresistivitet i en komponent eller leder med en ladning q, er gitt av Lorentz krafta 3.5.

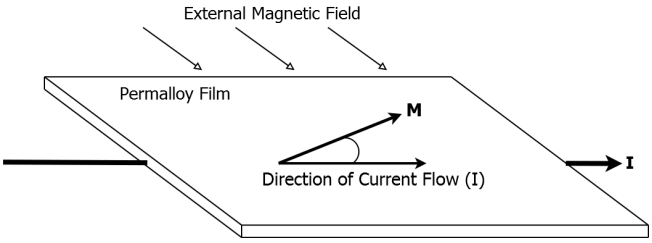
$$\mathbf{F} = q\left[\mathbf{E} + (l \times \mathbf{B})\right] \tag{3.5}$$

Dersom lederen er rett og med kjennskap til definisjon av elektrisk strøm, kan vi utvikle likning 3.5 til likning 3.6.

$$\mathbf{F} = \mathbf{I}l \times \mathbf{B} \tag{3.6}$$

En Lorentzkraft er en kraft som virker på ladninger i bevegelse i nærvær av et magnetfelt. Denne kraften tvinger elektronene i en leder til å bevege seg i kurvede baner justert etter påtrykt magnetfelt. Når ladningene forskyves til en mindre del av komponenten, vil komponentens ledningevne avta. Denne forskyvningen medfører en økt resistans i komponenten og nettop dette benyttest for å bestemme det påtrykte feltet **B**.

For å påvirke sensorens egenskaper til å måle et gitt magnetfelt med høy grad av linearitet,lav hysterese og med høy sensitivitet benyttes som oftest en legering av ulike metaller. Det eksisterer flere ulike megnetoresistive effekter, men mye brukt er Anisotropic Magneto Resisive (AMR) effekt som inntreffer i ferromagnetiske materialer. Oftes benyttes en legering som betegnes permaalloy, denne består av 80 % nikkel and 20% jern . Permalloy har gunstige egenskaper som lav sensitivitet til mekanisk stress og høy sensitivitet til magnetiske felt.

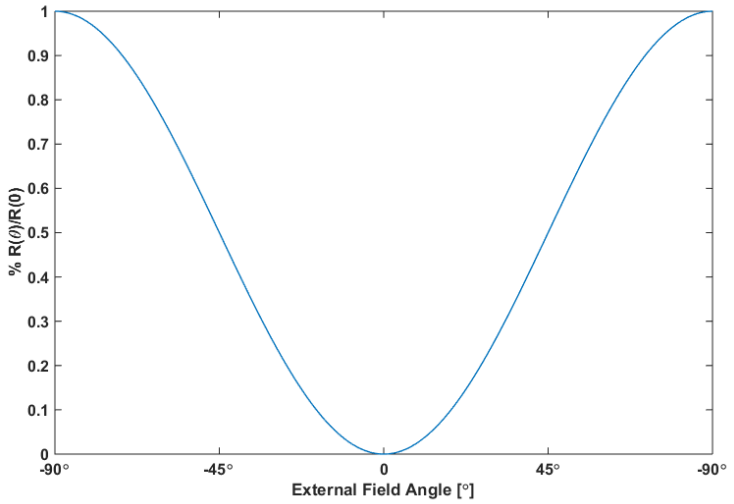


Figur 3.7: Konseptuelt design av en 2-D AMR sensor [12]

Som vist i figur 3.7 består en magnetoresistor (AMR) av en tynn rektangulær formet film. Den ene aksen til filmen kalles *easy axis* og denne er mye mer påvirkelig for magnetisering enn de andre aksene. Aksen *easy axis* er plassert langs lengden til filmen og *hard axis* er plassert ortogonalt på denne. Aksen normalt på filmen er på grunn av filmens tykkelse så marginal at den ikke er mottakelig for magnetisk felt.

Resistansendringen i permalloy film er en funksjon av vinkelen θ mellom påtrykt magnetfelt og strømmen **I** gjennom sensoren. Motstanden i sensoren er på sitt laveste når retningen på strømmen i sensoren og magnetfeltet er paralellt og på sitt høyeste når strøm og magnetfelt er vinkelrett på hverandre. For å beregne motstanden ρ i AMR elementet kan ligning (3.7) benyttes.

$$\rho(\theta) = \rho_{90^\circ} + (\rho_{0^\circ} - \rho_{90^\circ})cos^2\theta \tag{3.7}$$



Figur 3.8: Resistans som følge av påtrykt magnetfelt [12]

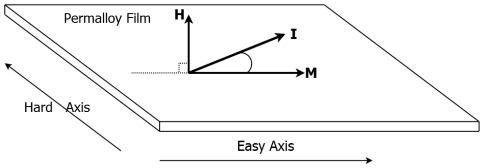
I figur 3.7 vises hvordan resistansen i ett element endres etter vinkel til eksternt magnetfelt. Her kan en også observere at resistansen i sensoren er polaritetsusensitiv, det vil si at responsen til et positivt felt er den samme som for et negativt felt. En kan også legge merke til at likningen 3.7 er periodevis tilnærmet lineær. Som et eksempel kan det i [12] leses av et lineært område fra 7.5° til 17.5°, men dette er parametere som er forventet at vil variere fra ulike produsenter og

sensortyper.

Som figur 3.8 viser, er området rundt 0° ulineært og med noe hysterese. For å kompensere for disse effektene kan en flytte det lineære området til permalloy ved å legge på striper med metall 45° på filmen. Disse stripene er et metall som innehar egenskaper som lavohmig og ikke ferromagnetisk. Mye benyttede materialer er gull eller aluminium.

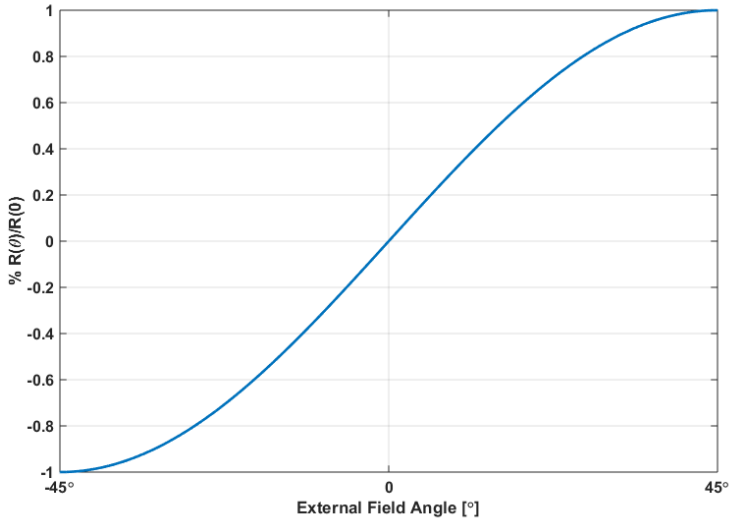
Figur 3.9: AMR sensor med ledende umagnetisk metall [12].
Permalloy(grå) og ledere (grønn)

Strukturen som vises i figur 3.9 er et eksempel på hvordan nevnte striper med ledende metall kan plasseres på en sensor for å forbedre egenskapene. Denne strukturen tvinger strømmen til å gå med 45° vinkel på *easy axis*.



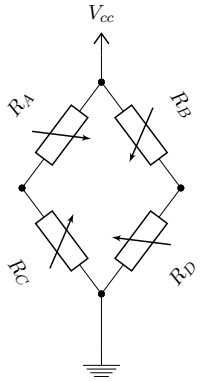
Figur 3.10: Konseptuelt design av av en 2-D AMR sensor [12]. Med ledende umagnetisk metall.

Ved å tvinge strømmen til å gå med 45° vinkel innføres en forskyvning i det målte feltet og derav flyttes måleområdet til en av de lineære delene av sensoren.



Figur 3.11: Forskyvning av måleområdet [12]

Ved å sammenlikne figur 3.8 med figur 3.11 kan det legges merke til flere forbedringer i sensorens respons. Den lineære responsen til felt med lav intensitet er forbedret og en har nå lite hysterese. Som nevn ovenfor måles ikke polariteten til magnetfeltet i en permalloy sensor, men ved å innføre metallstripene kan sensoren nå tolke polariteten til feltet. En endring i magnetfeltet, vil få vinkelen som strømmen går i til å avvike fra 0-punktet (45°) og magnetfeltet kan måles med polaritet.



Figur 3.12: AMR sensorer koblet i målebro

En typisk AMR sensor som måler en akse består av fire AMR elementer koblet i en Wheatstone bro [11] som vist i figur 3.12. For å øke utspenningen til hvert av elementene kan element R_A og R_D plasseres motsatt retta i forhold til R_B og R_C. En oppnår da et 4 ganger høyere sensorsignal fra hvert element i sensoren. Transferfunksjonen til sensoren kan da beskrives av:

$$v = IR_s \frac{\Delta\rho_m}{\rho} cos2\Delta\varepsilon h_a \sqrt{1 - h_a^2} \tag{3.8}$$

$$cos2\Delta\varepsilon = \frac{H_{k0}^2 + H_k^2 - (NM_s)^2}{2H_{k0}H_k} \tag{3.9}$$

$$h_a = \frac{H_a}{H_k + H_b} \tag{3.10}$$

Hvor

H_{k0} - Normalisert eksternt påtrykt felt.

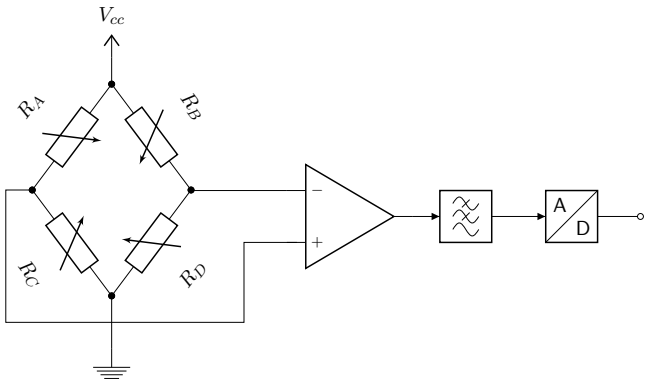
H_k - Retningsorientert felt.

R_s - Nominell resistans.

$\frac{\Delta\rho_m}{\rho}$ - Maksimal endring i resistivitet.

N - Demagnetiseringfaktor.

M_s - Magnetiseringmetning.



Figur 3.13: Prinsippskisse over magnetfeltmåling

Siden disse sensorene har et analogt utgangstrinn, der selve senorene er koblet i en Wheatstone bro, kan tradisjonell teori benyttes i signalbehandlingen. For å oppnå høy ytelse fra broa er det viktig med vel regulert forsyning uten drift. Det er også som vist i figur 3.13 et krav at signaljord har separat leder og ikke deler felles jordforbindelse. I figur 3.13 ser vi en prinsippskisse over hvordan sensoren kan kobles til en analog til digital omformer. Foruten selve broa består oppkoblingen av en differensialforsterker som forsterker forskjellen mellom inngangen samt et lavpassfilter. Siden sensoren er i stand til å måle langt høyere frekvenser enn samplefrekvensen til analog til digitalomformeren må et lavpassfilter fjerne de frekvenser som ville medført aliasing.

3.3 Valg av sensor

Vi har nå sett på den grunnleggende teorien som ligger bak de to sensorteknologiene som oppfylte de innledende krav. I forrige delkapittel så vi også på hvordan disse to måler den magnetisk feltstyrke på litt ulikt vis. Før det kan konkluderes med hvilken av de to sensorteknologiene som skal benyttes i oppaven, er det naturlig å se på noen siste krav.

I sammenlikningen av sensorer er det valgt å ta med 4 ulike sensorer. Disse 4 er listet i tabell 3.2. Som vi vil se i dette avsnittet kan flere av disse gi gode resultater i å måle magnetfeltstyrken generert av en høyspentmast, men det skal her forsøkes å komme frem til den best egnede sensoren.

Type	Produsent	Model	Rev	Akser	Pris
AMR	Honeywell	HMC2003		X,Y,Z	325$
	Honeywell	HMC1043L		X,Y,Z	18$
Flux Gate	Stefan Mayer Instuments	FLC3-70		X,Y,Z	499€
	Autonnic	A1015		X,Y	6£

Tabell 3.2: Oversikt over aktuelle sensorer

Vi starter med å se på kravet til **båndbredde**. Som nevnt innledningsvis, er kravet til båndredden bestemt av frekvensen til signalet den skal måle. For å unngå dempning av de frekvenser som

			Feil			
Type	Båndbredde	Måleområde	Linearitet	Hysterese	Sensitivitet	Minste feltstyrke
HMC2003	dc to $10^3 Hz$	$\pm 200\mu T$	0.5%FS	0.05%FS	$1V/100\mu T$	$4nT$
HMC1043	dc to $5 \cdot 10^6 Hz$	$\pm 600\mu T$	$\pm 0.1\%FS$	0.06%FS	$1mV/V/100\mu T$	$12nT$
FLC3-70	dc to $10^3 Hz$	$\pm 200\mu T$	$\ll 1\%$	Negligible	$2.857V/100\mu T$	$3nT$
A1015	Variable	$20\mu T$	5%	Negligible	Variable	$1nT$

Tabell 3.3: Sensor data hentet fra datablad

inneholder informasjon som vi ønsker å måle, må sensorene ha en båndbredde større enn den frekvensen som skal måles. Signalet som ønskes målt i denne oppgave, består hovedsaklig av en frekvenskomponent på 50Hz. Dette innebærer at valgt sensor bør ha en båndbredde større enn 100Hz.

For å komme frem til krav til måleområde og laveste feltstyrke som kan måles, er det nå på tide å se litt på selve magnetfeltet som skal måles. I tabell 2.1 er det en oversikt over typiske feltstyrker ved ulike strømmer som befinner seg i nærheten av en høyspentmast. De mest sentrale punktene fra denne tabellen oppsummeres her i tabell 3.4 og danner grunnlag for de neste kravene til sensor.

Strøm	$B_{x,max}$	μ_{B_x}	σ_{B_x}	$B_{y,max}$	μ_{B_y}	σ_{B_y}
100	6.06μ	0.862μ	$3.92n$	3.35μ	0.811μ	$5.34n$
200	12.1μ	1.72μ	$7.84n$	6.70μ	1.62μ	$10.7n$
1200	72.7μ	$10.3\ \mu$	$47.0n$	40.2μ	9.74μ	$64.1n$
2300	139μ	19.8μ	$90.1n$	77.0μ	18.7μ	$123n$
2400	145μ	20.7μ	$94.0n$	80.4μ	19.5μ	$128n$

Tabell 3.4: Simulerte verdier

Når vi nå har verdier på forventet magnetfelt i tabell 3.4, er det nå naturlig å se på krav til sensorens **måleområde**. For å finne et fornuftig krav til måleområde leter vi i tabell 3.4 etter den maksimale feltstyrken som ønskes målt. I denne tabellen finner vi at feltet er på sitt maksimale ved den høyeste linje strømmen. I nederste rad finner vi den høyeste feltstyrken ved $I = 2400A$ til å være $B_{x,max} = 145\mu T$ og $B_{y,max} = 80.4\mu T$. Dette betyr at hver av sensorens akser må ha et måleområde større enn $\pm 145\mu T$.

Ett av de innledende kriteriene for å velge sensorteknologier, var evnen til å måle lave feltstyrker. I figur 3.1 ble de ulike teknologiene gruppert inn i to hovedgrupper, avhengig av hvor lav feltstyrke som kunne måles. Flere kjente teknologier ble avvist på grunn av manglende evne til å måle lave feltstyrker. Nå skal vi her se på hvordan sensorene i tabell 3.3 overholder ønsket krav. For å kunne måle svært lave feltstyrker, er det viktig å se på den **minste feltstyrken**, B_{min}, som de valgte sensorer er i stand til å måle. Det ble tidlig i prosjektet i samråd med veileder Morten

Tengesdal besluttet at B_{min} bør være omtrent 1 ‰ av minste forventede feltstyrke,$\mu_{B_{min}}$. Fra tabell 3.4 leser vi av verdi for $\mu_{B_y} = 0.811\mu T$. Kravet til B_{min} blir da omtrent $0.8nT$. Når vi sammenlikner kravet med sensorene i tabell 3.3 finner vi at ingen av sensorene er i stand til å oppfylle dette kravet. Det kan da konkluderes med at kravet er noe høyt og bør reduseres noe. Ved å studere tabell 3.3 ser vi at 3 av sensorene er relativt i nærheten av det opprinnelige kravet. Det kan derfor virke fornuftig å sette kravet slik at disse er innenfor. Modifisert krav til B_{min} blir dermed satt til $B_{min} = 4nT$.

For å kunne skille mellom små variasjoner i feltstyrken er det også viktig at sensoren er sensitiv nok til å måle selv små endringer i feltstyrken. Det siste krav som stilles, er krav til sensorens **sensitivitet**. For at dette kravet skal kunne fastsettes på korrekt grunnlag, er det også her naturlig å se på forventede variasjoner i feltsyrken. For å finne denne variasjonen benyttes tabell 3.5. Denne tabellen inneholder forventningsverdien til endring i feltstyrken ved en 1cm posisjonsendring.

	$B_x[T]$				$B_y[T]$			
Strøm [A]	μ_x	σ_x	μ_y	σ_y	μ_x	σ_x	μ_y	σ_y
100	$1.68n$	$0.428p$	$1.77n$	$0.583p$	$1.77n$	$0.583p$	$1.68n$	$0.428p$
200	$3.37n$	$0.856p$	$3.54n$	$1.17p$	$3.54n$	$1.17p$	$3.37n$	$0.856p$
300	$5.05n$	$1.28p$	$5.31n$	$1.75p$	$5.31n$	$1.75p$	$5.05n$	$1.28p$
1200	$20.2n$	$5.13p$	$21.3n$	$7.00p$	$21.2n$	$7.00p$	$20.2n$	$5.13p$
2300	$38.7n$	$9.84p$	$40.7n$	$13.4p$	$40.7n$	$13.4p$	$38.7n$	$9.84p$
2400	$40.4n$	$10.3p$	$42.5n$	$14.0p$	$42.5n$	$14.0p$	$40.4n$	$10.3p$

Tabell 3.5: Simulerte verdier

Tabellen viser at forventet endring i feltstyrke varierer betraktelig som funksjon av strøm i lederne. Når det er så stor variasjon kan det være en utfordring å beholde presisjonen for lave og høye feltstyrker.

Det hele oppsummeres i tabell 3.6 og på grunnlag av denne tabellen velges sensor.

Sensor	Båndbredde	Måleområde	B_{min}	Sensitivitet	Pris
HMC2003	√	√	√	√	325$
HMC1043L	√	√	−	√	18$
FLC3-70	√	√	√	√	$499e$
A1015	√	−	√	√	$6£$

Tabell 3.6: Sensorkrav oppsummert

Tabell 3.6 viser at det er to sensorer som skiller seg ut ved å oppfylle kravene. Det kan videre forventes at de begge produserer tilsvarende målinger for feltstyrken. Når det allikevel skal velges kun en sensortype, vektlegges kriteriene pris og linearitet. Ved å sammenlikne lineariteten til de to sensorene, ser vi fra tabell 3.3 at HMC2003 har en marginalt bedre linearitet og er den

rimeligste.

Basert på kriterier nevnt ovenfor, velges HMC2003 som sensor i denne oppgaven.

Det er imidlertid viktig å notere seg at valgt sensor bryter fullstendig med designet skissert i figur 1.2. Den analoge sensoren HMC2003 vil kreve et langt mer komplekst kretskort en hva som innledningsvis ble lagt til grunn for design av sensornode. For å være i stand til å opprettholde presisjon i de målte signaler blir det ved valg av kretser og design av kretskort spesielt viktig å ta hensyn til analoge signaler med lav amplitude.

Kapittel 4

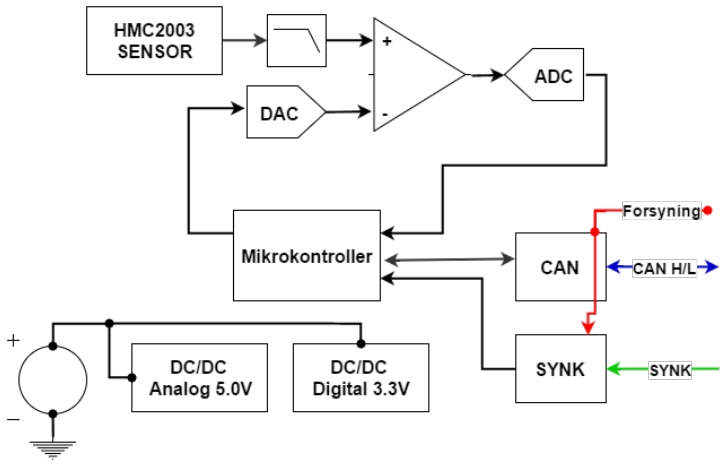
Maskinvaredesign

I dette kapittelet skal vi se hvordan det er valgt å designe kretskortet som måler de magnetiske feltstyrkene og det som mottar målingene for videre behandling. Underveis blir det forklart de viktigste og mest sentrale problemstillenger som dukker opp. Der det anses informativt vises utsnitt av skjema. For en fullstendig oversikt anbefales det å studere skjema i vedlegg D.1.

Det eksisterer mange ulike verktøy som kan benyttes for å legge ut kretskort. Flesteparten av verktøyene er imidlertid enten kostbare eller har så store begrensninger at de ikke kan anses egnede i denne oppgaven. På grunnlag av tidligere erfaringer er det valgt å benytte det vederlagsfrie programmet Design Spark PCB, levert av RS. Dette er et program som er fullstendig uten begrensninger og er godt egnet til å designe nødvendige kretskort i.

4.1 Sensornode

Figur 4.1 viser et oppdatert diagram over hvordan sensornoden nå er tenkt designet. Som det går frem av illustrasjonen, er sensoren delt inn i flere ulike deler. De mest sentrale delene oppsummeres nedenfor og grundigere forklaring gis i forbindelse med valg av den enkelte komponent.



Figur 4.1: Blokkskjema sensor.

Øverst i illustrasjonen finner vi den delen som måler magnetfeltstyrken, med tilhørende signalbehandling. Målt signal vil etter signaltilpassning bli kvantisert av ADC og sendes til mikrokontrolleren.

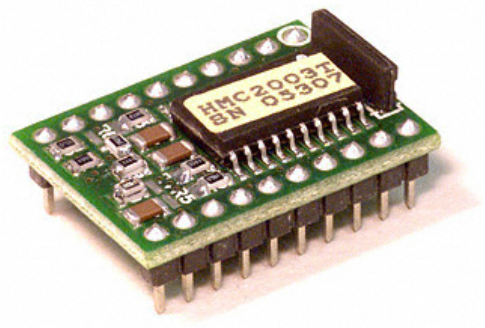
Mikrokontrolleren i dette systemet har som hovedoppgave å motta målte data fra ADC, korrigre DAC spenning for så å sende magnetfeltmålinger over canbus. Dette tillater valg av en enkel mikrokontroller med lavt strømforbruk.

Nederst til venstre i figuren finner vi batteri og spenningsregulatorer. For å unngå støy fra den digitale delen av sensorkortet, separeres strømforsyningene i en ren digital- og en analog -forsyning.

Til høyre i illustrasjonen finner vi inntegnet grensesnittene mellom sensornode og øvrige enheter. Grensesnittene er galvvanisk isolert fra det kablede nettverket og det benyttes ekstern forsyning for signaler som går til og fra kretskortet. Dette er med på å unngå at jordstrømmer og ledningsbunden støy skal kunne påvirke presisjonen til sensornoden.

4.1.1 Magnetfeltsensor HMC2003

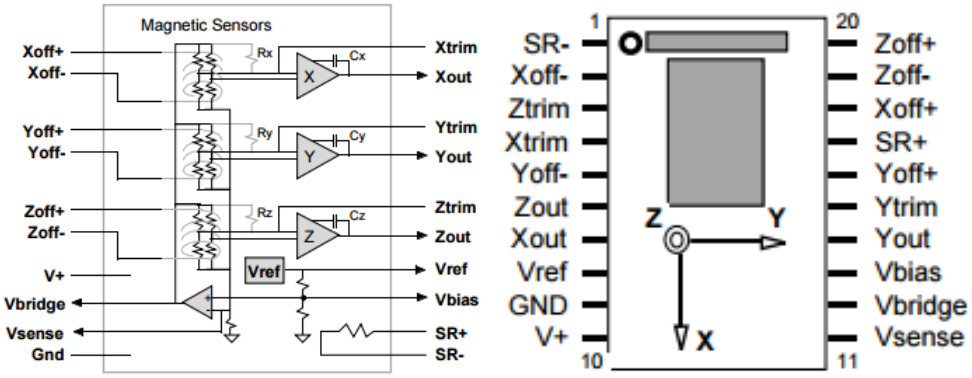
Vi skal nå se på de viktigste egenskaper til den valgte sensor, med vekt på de funksjoner og egenskaper som anses nyttige i denne oppgaven. Honeywell HMC2003 [13] er en høysensitiv AMR 3-akset magnetfeltsensor designet for å måle svake magnetfelt. For å kunne måle 3 akser, er sensoren sammensatt av to Honeywell sensorer integrert på samme kretskort. Hvordan sensoren måler magnetfeltet dekkes av gjennomgått teori i kapittel 3.2.2.



Figur 4.2: Foto av HMC2003,fra www.digikey.com

Som nevnt ovenfor består hvert enkelt kretskort av to magnetfeltsensorer som sammen måler de tre aksene (xyz). For å måle z-aksen benyttes en-aksesensoren HMC1001, de neste to aksene måles ved hjelp av to-aksesensoren HMC1002.

Ved å sammenlikne prinsippskissen for en AMR sensor som vises i figur 3.12 med den vi finner i figur 4.3, ser vi at valgte sensor er bestykket med ulike komponenter. Monterte resistanser bidrar til å redusere målefeil som følge av temperatur feil, forsterkning og offsett drift. For at



Figur 4.3: Kretsskjema og pinout som illustrerer sensorens akser [13].

hver sensormodul skal overholde spesifikasjoner gitt av datablad, testes og justeres hver enkelt sensors resistorverdier av produsent. Med denne test og trimmingen av komponentverdier kan vi være sikre på at hver solgte modul er innenfor gitte spesifikasjoner.

Fra datablad tilhørende HMC1001 og HMC1002 [11] finner vi båndbredden til sensorene å være 5MHz. For å redusere den høye bådbredden i HMC2003 sensormodul, har Honeywell valgt å legge til et lavpassfilter per kanal, med knekkfrekvens $f_{-3dB} = 1kHz$. Siden vi i denne oppgaven skal måle en frekvenskomponent på 50Hz er 1KHz båndbredde mer enn tilstrekkelig for å unngå dempning av signalet som skal måles.

For at sensorene skal kunne operere i henhold til gitte spesifikasjoner i hele sin levetid, har Honeywell konstruert to patenterte løsninger. Løsningene er at hver akse har hver sin offset- og set-/reset-strap. I figur 4.3 benyttes betegnelse *Xoff+ Xoff-* for offsett strapper og *SR+ SR-* for set/reset strap. Strappene er lavohmige ledere plassert i nærheten av målebroene og som vist i neste avsnitt, er disse i stand til å påvirke sensorens målinger. Strappene er orientert slik at offset strappen er plassert parallellt med easy-axis og set/reset er plassert ortogonalt på easy-axis. For dypere forståelse av dette henvises det til kapittel 3.2.2.

Offset straps

Som vist i kretsskjema i figur 4.3, har hver akse sin separate offset strap. Denne kan stappen har flere bruksområder, men dersom den benyttes, er det gjerne for å eliminere et eller flere uønskede bidrag til den målte feltstyrken. De uønskede bidragene kan her være alt fra jordens eget magnetfelt til faste magnetfeltkilder i nærheten av sensoren. I denne oppgaven skal det imidlertid måles et tidsvarierende magnetfelt og faste statiske magnetfelt vil dermed ikke påvirke amplituden til det målte signalet. Det gis allikevel en rask introduksjon til virkemåten til offset straps.

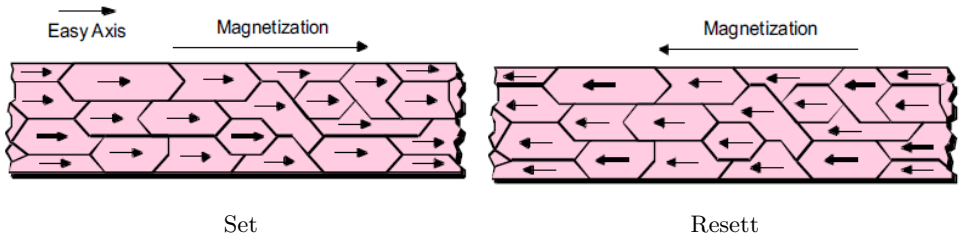
Virkemåten til offset funksjonen er at en metallspiral er plassert parallellt med sensorelementets sensitive akse. Ved å tilføre en kontrollert strøm gjennom denne spiralen, vil det genererte magnetfeltet påvirke målebroen og på denne måten kan et felt legges til eller trekkes fra. En strøm som sendes fra + terminal til - terminal vil legge til et magnetfelt og motsatt vil en trekke fra et felt ved å sende strøm fra - terminal til + terminal.

$$B_{Offset} = 0.5mA/\mu T \tag{4.1}$$

Ligning 4.1 viser at det kreves en relativt høy strøm for å eliminere selv et lite magnetfelt, på hver av aksene. Som kjent medfører et høyt strømtrekk, varmeutvikling i selve sensoren og dette introduserer gjerne nye uønskede effekter. Siden vi i denne oppgaven ser på å konstruere en mobil sensorløsning, er det viktig å fokusere på vekt og strømforbruk i alle ledd. Ved å benytte denne muligheten vil vi som nevnt ha behov for høyere batterikapasitet, noe som medfører økt kostnad og vekt på den ferdige løsningen.

Set og reset strap

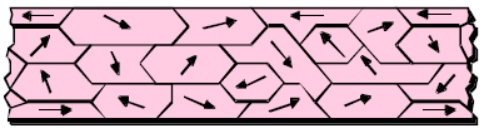
I løpet av produksjon av magnetfeltsensorer fra Honeywell, blir easy-aksis etablert langs lengden av AMR filmen. Denne prosessen orienterer AMR filmen slik at den har størst følsomhet i aksen den er tiltenkt å måle.



Figur 4.4: AMR film i set og reset tilstand [11].

Figur 4.4 viser en illustrasjon av en AMR film som er i set og en i reset tilstand. I disse tilstandene er sensoren i stand til å måle et magnetfelt med den høyeste presisjonen som sensoren tillater. Forskjellen mellom tilstandene set/reset er polariteten til hvordan feltet måles. Med det menes at det byttes fortegn til det målte magnetfelt ved å gå fra tilstand set til reset og motsatt.

Dersom sensoren blir påvirket av et magnetfelt over 1mT kan dette i følge [11], påvirke sensorens karakteristikk og degradere dens fremtidige egenskaper som presisjonssensor. Sensorens orientering vil nå bli som vist i figur 4.5. Det er tydelig å se en mer vilkårlig orientering. Sensoren har mistet noe av sin presisjon og vil heretter gi lavere utslag ved målinger.



Figur 4.5: AMR film med vilkårlig orientering [11].

Det er nå viktig å kunne gjennopprette sensorens opprinnelig karakteristikk. Dette gjøres ved å påtrykke set/reset strap med en høy strømpuls. Da gjennopprettes sensorens orientering og høye presisjon, se figur 4.4. Styrken til strømpulsen blir bestemt av ønsket sensorpresisjon og hvor lave felt det er ønskelig å måle.

Ved å benytte set/reset funksjonen under normal drift kan feilkilder som temperaturdrift,ulinearitet og kryss-akse effekt reduseres eller elimineres. Det er i [11] gitt en fremgangsmåte for å eliminere nevte feilkilder. Denne er:

1) - En strømpuls, I_{Set}, påtrykkes terminalen SR+. Når sensor gir en stabil utgangsverdi lagres denne som V_{Set}.

2) - En motsatt rettet strømpuls, I_{Reset}, påtrykkes terminalen SR+. Når sensor gir en stabil utgangsverdi lagres denne som V_{Reset}.

3) - Det kan nå beregnes en offset spenning som er gitt av:

$$V_{Offset} = \frac{V_{Set} - V_{Reset}}{2}$$

Nå kan spenningen V_{Offset} benyttes som et korrigerende ledd i endelig beregning av den målte feltstyrken. Fremgansmåten som er listet i punktene ovenfor vil bidra til å kansellere offset- og temperatur-effekter forårsaket av drift i sensor og tilhørende komponenter.

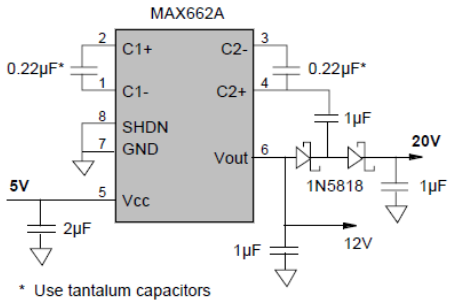
Kravet til strømpulsene, I_{Set} og I_{Reset}, varierer etter hvor lavt felt det er ønskelig å måle. Dersom minste felt som skal måles er $\approx 50nT$, er det tilstrekkelig med en puls på 3A [11]. Da det i denne oppgaven er ønskelig å måle feltstyrker betydelig lavere enn dette, kreves en strømpuls på over 4A i $2\mu s$ for å sette eller resette sensoren fullstendig [11].

Fordi HMC2003 sensoren har en samlet resistans i set/reset lederne på typisk 4.5Ω, kreves en relativt høy spenning. Ved hjelp av Ohms lov finner vi minste spenning til å være:

$$V_{min} = 4A \cdot 4.5\Omega = 18V$$

Som vi beregnet må spenningen være minst 18V for å være i stand til å sende ønsket strøm gjennom lederne. For å oppnå dette må forsyningsspenningen transformeres opp til ønsket nivå.

Foruten selve spenningsnivået, er det også viktig å dimensjonere et strømbuffer som er i stand til å levere minimum 4A i 0.2μs. Det er også ønskelig at kretsen som transformerer spenningen skal kunne startes og stoppes fra et styresignal. Styresignalet ønskes styrt fra en en prosessor og bør derfor ha en 3.3V tolerant inngang.

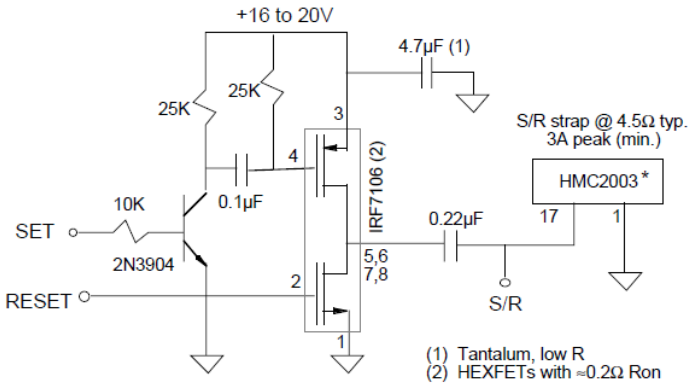


Figur 4.6: Forslag til krets for å transformere spenning [11].

Den største utfordringen med den foreslåtte krets er at den synes ikke å være tilpasset ønskede forsyningsspenninger på kretskortet. Dersom vi studerer kretsen i figur 4.6, kan det også legges merke til at den krever flere komponenter for å transformere spenningen. Spørsmålet er da om dette kan konstrueres med enklere og rimeligere komponenter, som enkelt kan tilpasses det aktuelle kretskortet?

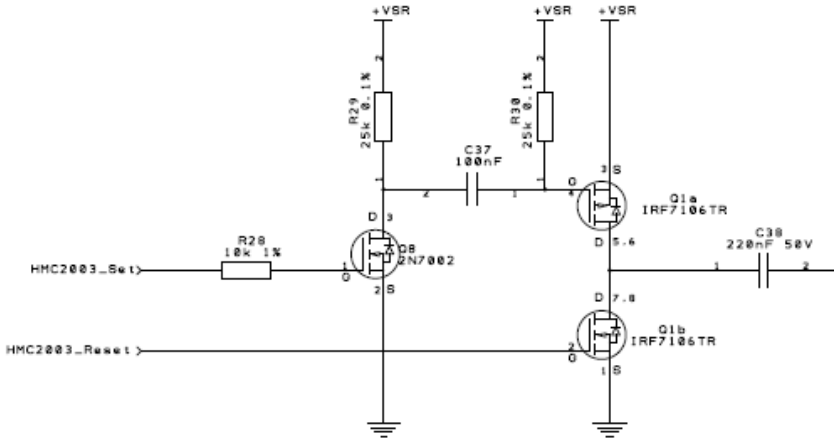
Hvordan dette er løst vises i delkapittel 4.1.7.

Nå som vi har sett på hvordan strømforsyningen til set/reset funksjonen er konstruert, er det nå på tide å se hvordan set/reset pulsen utføres. I design av utlegg er det valgt å benytte en anbefalt kretsløsning fra [11].



Figur 4.7: Anbefalt kretsløsning for å drive set/reset [11].

Ved å sammenlikne den anbefalte løsningen med den valgte, kan det legges merke til at de er tilnærmet identiske.

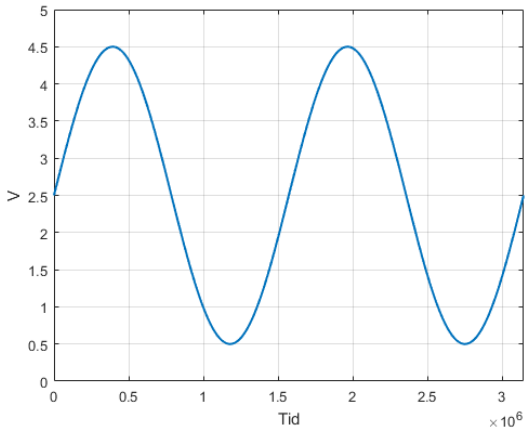


Figur 4.8: Valgt kretsløsning for å drive set/reset.

4.1.2 Analog signalbehandling

I forrige delkapittel så vi på hvordan sensoren måler magnetfelt og hvordan det oppnås best mulig presisjon og sensitivitet. I dette delkapittelet skal vi se på den analoge signalbehandlingen som må utføres for å oppnå pålitelige målinger av signalene fra sensor.

Fra datablad [13] er det opplyst at sensor HMC2003 måler feltstyrker i området $\pm 200\mu T$, med laveste målbare feltstyrke $B \approx 4nT$. Magnetfeltsignalet fra sensor er oppgit til å være sentrert rundt et nullpunkt i 2.5V, med endring $1V/100\mu T$.



Figur 4.9: Simulert utgangssignal fra HMC2003.

Figur 4.9 viser et plott over forventet utgangssignal fra HMC2003 sensor. Dette viser tydelig signalets sving rundt senter og med amplitude på $2V$. Videre i delkapittelet kommer dette plottet til å danne grunnlag for flere analyser.

Nå skal krav til analog signalbehandling vurderes, men det må først på plass en ligning som gir forholdet mellom signalspenning og Tesla. Dette forholdet er gitt av:

$$V_{\Delta B} = 10mV/\mu T \tag{4.2}$$

Som ligning 4.2 viser, er signalene fra sensor relativt lave ved lave felt. Tabell 3.5 viser verdier for forventet endring i feltstyrke ved en posisjonsendring på 1cm. Grenseverdiene settes inn i tabellen og finner forventet spenningsendring:

Feltstyrke, μ_B	$V_{\Delta B}$
$1.68nT$	$16.8\mu V$
$42.5nT$	$425\mu V$

Tabell 4.1: Endring i utsignal ved grenseverdiene μ_{min} og μ_{max}.

Som tabell 4.1 viser, er det ved μ_{min} en endring i utgangssignalet fra sensor på $16.8\mu V$. For å være i stand til å måle endringen kreves det at vi benytter en analog til digitalomformer med minst 20bit oppløsning. En analog til digital omformer med så høy oppløsning er som oftest både dyr, har høyt strømtrekk og er fysisk stor.

Legg merke til figur 4.9 hvor det aktive signalområdet fra sensoren befinner seg mellom 0.5V og 4.5V. For å kunne konvertere dette til en digital verdi, kreves en analog til digital omformer med refreansespenning, $V_{ref} = 5.0V$, og kommunikasjon med mikrokontroller, $V_{IO} = 3.3V$. Ved et søk finner vi fort ut av at dette er relativt uproblematisk, men må med som vilkår når valg av ADC gjøres.

Før vi ser på hvor mye signalet bør forsterkes, ser vi litt på hvor fin oppløsning som oppnås ved å benytte analog til digital omformer med noen færre bit.

ADC_{bit}	Oppløsning $V_{ref} = 3.3V$	Oppløsning $V_{ref} = 5.0V$
8	$12.89mV$	$19.53mV$
10	$3.22mV$	$4.88mV$
12	$805.66\mu V$	$1.22mV$
14	$201.41\mu V$	$305.05\mu V$
16	$50.35\mu V$	$76.63\mu V$

Tabell 4.2: ADC oppløsning ved ulike bit og V_{Ref}.

Dersom vi sammenlikner resultatene i tabell 4.1 med tabell 4.2 ser vi tydelig at det kun er analog til digital omformer med **minst** 16 bit som innehar tilstrekkelig oppløsning. Det må allikevel legges merke til at å benytte en 16 bit ADC uten å gjøre smarte valg i signalbehandlingen, vil medføre reduksjon i sensitivitet og små endringer i magnetfeltet vil gå tapt. Basert på dette vil det i denne oppgaven, hovedsaklig bli fokusert på analog til digitalomformere med 16 bit eller høyere oppløsnig.

Nå er det vist hvordan sensorsignalet endrer seg ved posisjonsendring og de utfordringene dette innebærer. Vi skal nå se litt på feltstyrken som måles og ikke bare endringen av den målte som funksjon av posisjonsendring. Tabell 3.4 viser den forventede målte feltstyrke samt maksimal feltstyrke som kan forventes målt. Vi setter disse grenseverdiene inn i en tabell og beregner forventet sensorspenning.

Parameter	Feltstyrke	Sensor spenning V_B
$\mu_{B,min}$	$0.811\mu T$	$2.5V \pm 8.11mV$
$\mu_{B,max}$	$20.7\mu T$	$2.5V \pm 207mV$
B_{max}	$145\mu T$	$2.5V \pm 1.45V$

Tabell 4.3: Utsignal ved grenseverdiene $\mu_{B,min}$,$\mu_{B,max}$ og B_{max}.

Tabell 4.3 viser at det er forventet at signalet svinger over hele det aktive området. For å beholde hele informasjonen i det målte magnetfeltet, må det konstrueres et analogtrinn som er i stand til å måle hele sensorens utslag med god presisjon.

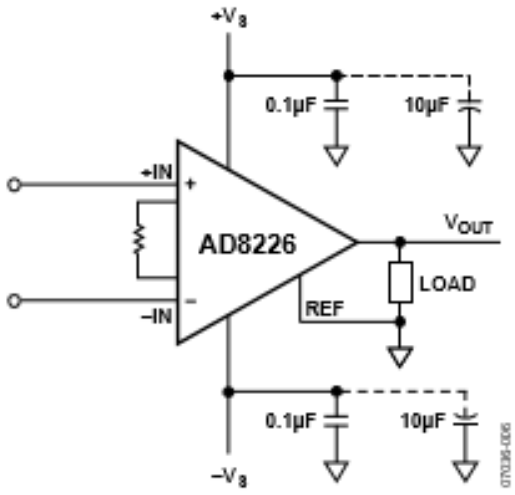
Tabellen viser også at høyeste forventede spenning fra sensor er $\approx 4V$. Nivået til signalspenningen vil dermed sterkt begrense muligheten til å forsterke opp signalet uten tap av informasjon. Vi kommer frem til at høyeste tillatte forsterkning uten forvrengning er 1.25. Ved så lav forsterkning vil vi ikke kunne måle de laveste feltstyrkeendringene med en 16bits ADC, uten at viktig informasjon går tapt.

For å kompensere for dette kan det enten benyttes en ADC med høyere oppløsning, eller vi kan utforske mulighetene vi har ved å utføre analog signalbehandling. Siden det ikke er ønskelig med høyere oppløsning skal vi nå se på sistnevnte og starter med å vise hvilke muligheter en instrumentforsterker introduserer.

Instrumenteringsforsterker

Vi skal nå se på hvilke muligheter en instrumenteringsforsterker introduserer for oss og forhåpentligvis vil den kunne bidra til at det oppnås en akseptabel forsterkning.

Det eksisterer utallige gode instrumentforsterkere. Jeg vil i denne oppgaven velge på grunnlag av tidligere erfaringer fra faget *Analog og Digital Elektronikk* ved UiS. I dette faget ble det benyttet instrumentforsterker AD623 fra Analog Devices og denne viste gode egenskaper. AD623 er nå erstattet med en forbedret men tilsvarende variant, AD8226 [6]. På dette grunnlag velger jeg å benytte AD8226 i denne oppgaven.



Figur 4.10: AD8226 [6].

Figur 4.10 viser instrumenteringsforsterkeren AD8226 illustrert med de innganger og utganger den er bestykket med samt en anbefalt oppkobling [6]. Vi vil i kommende delkapittel se at komponenten er i stand til å utføre den ønskede signaltilpassning.

Uttrykk for transferfunksjonen til instrumentforsterkeren:

$$V_{out} = G \cdot \big(V_{In+}(t) - V_{In-}(t)\big) + V_{Ref}(t) \tag{4.3}$$

Hvor

$$G = 1 + \frac{49.4k\Omega}{R_G} \tag{4.4}$$

Der R_G beregnes ved hjelp av:

$$R_G = \frac{49.4k\Omega}{G - 1} \tag{4.5}$$

Dersom V_{Ref} kobles til jord, endres ligning 4.3 til

$$V_{out}(t) = G \cdot \big(V_{In+}(t) - V_{In-}(t)\big) \tag{4.6}$$

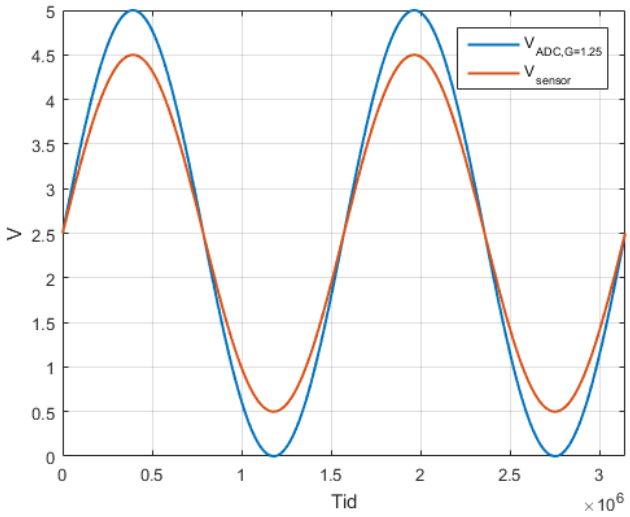
Ligning 4.6 viser at utgangsspenningen V_{out} er funksjon av forsterkningen G og differansen mellom inngangene V_{In+} og V_{In-}. Dette kan vise seg å bli svært nyttig funksjon og åpner for å redusere deler av signalet, samtidig som den resterende delen kan forsterkes opp.

- **Første ide, trekke fra en konstant spenning**

For å flytte signalet ned i nivå, kan en positiv spenning legges til på inngangen V_{In-}. Fra ligning 4.6 ser vi da at spenningen på V_{In-} terminalen trekkes fra signalet før det forsterkes.

For å utforske den mulige løsningen på problemstillingen, velger jeg å kjøre en simulering i Matlab. I simuleringen legges det inn et test sinussignal, som vist i figur 4.9. Testsignalet er en sinus med amplitude 2V og senter i 2.5V. Dette er ment for å kunne tilsvare et realistisk sensorsignal, samtidig som øvre og nedre grenseverdier tydelig fremkommer ved simulering. I simuleringen settes nivået til V_{In-} lik nedre grenseverdi fra sensor, dvs $V_{In-} = 0.5V$. Nå vises en realistisk simulering av responsen til instrumentforsterkeren og resultatene fra simuleringen vises i figur 4.11.

Figuren viser at ved å benytte en forsterkning på 1.25 med $V_{In-} = 0.5V$, oppnås det at signalet varierer mellom hele analog til digitalomformerens måleområde. Nå kan det opprinnelige signalet måles med større presisjon, men som vi kom frem til tidligere, er ikke denne forsterkningen tilstrekkelig. Derimot, hvis vi kunne forvente at høyeste målte felt var betydelig lavere enn hva som vises i tabell 4.3, kunne denne fremgangsmåten gitt gode resultater.



Figur 4.11: Signal V_{In+}(rød) og V_{Out}(blå) fra instrumentforsterker, der $V_{In-} = 0.5V$.

- **Modifisert ide**

Det er til nå utforsket ulike muligheter med en fast verdi som trekkes fra i en instrumentforsterker og kan konkludere med at dette ikke løser våres utfordringer med signaltilpassningen. Jeg ønsker allikevel ikke å forkaste ideen om at en instrumentforsterker kan tilpasse signalet slik at ønsket om å kunne benytte en enklere ADC, samtidig som hele signalets informasjon beholdes.

Dersom vi nå modifiserer den opprinnelige ideen, der det benyttes en instrumentforsterker til signaltilpassningen og bytter ut den faste referansen V_{In-}, med en varierende referanse, vil dette gi noen forbedringer sammenliknet, i forhold til fast referanse? Vi skal nå undersøke nærmere om dette kan være en god ide og om den oppfyller gitte ønsker til signalkonvertering.

Ligning 4.6 modifiseres til:

$$V_{out}(t) = G \cdot (V_{In+} - A \cdot V_{In-}) \tag{4.7}$$

Som vist i ligning 4.7 er det innført en egen forsterkningsfaktor A til leddet V_{In-}.Ved hjelp av dette kan vi nå finne en smart måte å velge referansesignalet på, slik at dette trekker fra meste-parten av det statiske signalet fra sensor. Med dette introduserer vi en mulighet til å forsterke opp den resterende delen av signalet og dermed oppnå høyere sensitivitet for lave feltstyrker. Som vist i tabell 4.1, er det nettopp her informasjon om posisjonsendringene finnes.

For å bestemme høyeste forsterkning som oppnås uten forvrengning, må ligning for V_{out} studeres. Her kan det være smart å velge referansesignalet V_{In-} til å være lik sensorsignalet V_{In+}. Det oppnås da å kontinuerlig trekke fra en konstant del fra signalet. Ligning 4.7 modifiseres og vi finner et nytt uttrykk for V_{out}:

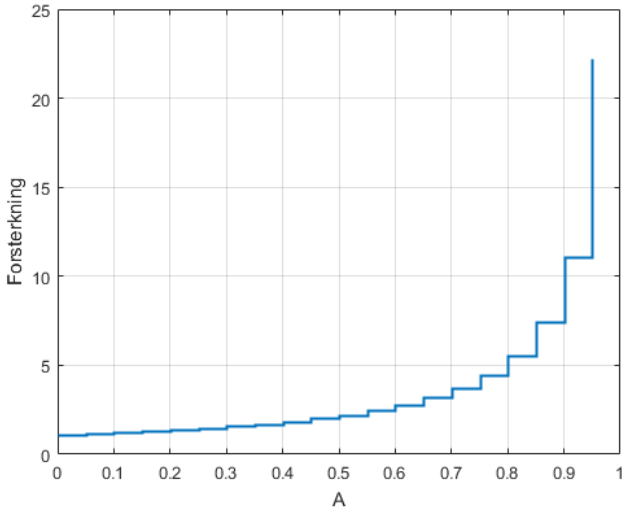
$$\begin{aligned} V_{out}(t) &= G \cdot \big(V_{In+}(t) - A \cdot V_{In-}(t)\big) \\ &= G \cdot \big(V_{In+}(t) - A \cdot V_{In+}(t)\big) \\ V_{out}(t) &= G \cdot (1 - A) \cdot V_{In+}(t) \end{aligned} \tag{4.8}$$

Når vi studerer ligning 4.8 som vi kom frem til ovenfor, legger vi merke til at faktorene som nå må bestemmes er A og G. Vi ordner så ligning 4.8 slik at den får et uttrykk for forsterkningen G.

$$G = \frac{V_{out}(t)}{(1 - A) \cdot V_{In+}(t)} \tag{4.9}$$

For å finne den optimale størrelse på A, velger jeg å plotte ligning 4.9. Plottet vises i figur 4.12 og som foreventet øker den mulige forsterkningen med økende faktor A.

For å plotte ligningen, må vi sette inn verdier for V_{out} og V_{In+}. Vi finner her største forsterkning som kan velges, ved å sette inn de største verdien som faktorene antas å kunne ha. For verdier større enn dette må det forventes forvrengning og da tapes deler av måleområdet. I plottet velges verdier $V_{out} = 5.0V$ og $V_{In+} = 4.5V$.



Figur 4.12: Plott av ligning 4.9. Innsatt $V_{out} = 5.0V$ og $V_{In+} = 4.5V$

Før forsterkningen G, kan bestemmes, må det først bestemmes en passende verdi til konstanten A. Her passer det bra å sammenlikne resultatet fra ulike verdier i en tabell. Sammenlikningen baseres på grenseverdier til sensorens forventede utgangssignal, der disse er hentet fra tabell 4.1.

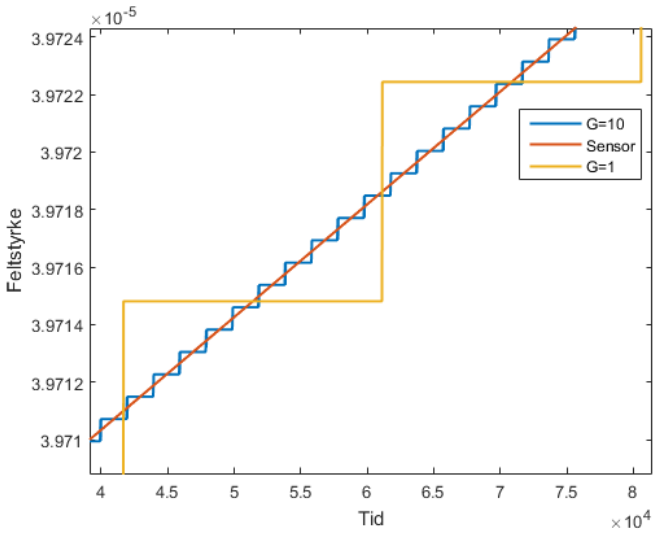
				ADC bit som representerer μ_{min} og μ_{max}			
A	G	μ_{min}	μ_{max}	$14bit_{min}$	$14bit_{max}$	$16bit_{min}$	$16bit_{max}$
0.00	1.11	$18.7\mu V$	$472\mu V$	0.06	1.55	0.24	6.19
0.05	1.17	$19.6\mu V$	$497\mu V$	0.06	1.63	0.26	6.52
0.10	1.23	$20.7\mu V$	$525\mu V$	0.07	1.72	0.27	6.88
0.15	1.31	$22.0\mu V$	$556\mu V$	0.07	1.82	0.29	7.28
0.20	1.39	$23.3\mu V$	$590\mu V$	0.08	1.93	0.31	7.74
0.25	1.48	$24.9\mu V$	$630\mu V$	0.08	2.06	0.33	8.25
0.30	1.59	$26.7\mu V$	$675\mu V$	0.09	2.21	0.35	8.84
0.35	1.71	$28.7\mu V$	$726\mu V$	0.09	2.38	0.38	9.52
0.40	1.85	$31.1\mu V$	$787\mu V$	0.10	2.58	0.41	10.32
0.45	2.02	$33.9\mu V$	$859\mu V$	0.11	2.81	0.44	11.25
0.50	2.22	$37.3\mu V$	$944\mu V$	0.12	3.09	0.49	12.38
0.55	2.47	$41.5\mu V$	$1.05mV$	0.14	3.44	0.54	13.75
0.60	2.78	$46.7\mu V$	$1.18mV$	0.15	3.87	0.61	15.47
0.65	3.17	$53.3\mu V$	$1.35mV$	0.17	4.42	0.70	17.68
0.70	3.70	$62.2\mu V$	$1.57mV$	0.20	5.16	0.82	20.63
0.75	4.44	$74.7\mu V$	$1.89mV$	0.24	6.19	0.98	24.76
0.80	5.56	$93.3\mu V$	$2.36mV$	0.31	7.74	1.22	30.95
0.85	7.41	$124\mu V$	$3.15mV$	0.41	10.32	1.63	41.26
0.90	11.11	$187\mu V$	$4.72mV$	0.61	15.47	2.45	61.90
0.95	22.22	$373\mu V$	$9.44mV$	1.22	30.95	4.89	123.79

Tabell 4.4: Utsignal ved grenseverdiene $\mu_{B,min}$,$\mu_{B,max}$ og B_{max}.

Tabell 4.4 viser at ved å modifisere ligning 4.6 til ligning 4.8, kommer det tydelig fram at det kan oppnås stor gevinst ved denne modifikasjonen. Videre viser tabellen at den tidligere maksimale forsterkningen nå har økt fra 1.25 til 22.2. Dette medfører at det oppnås inntil $\approx$ 18 ganger høyere signalforsterkning i forhold til opprinnelig.

Hvor mye signalet kan reduseres uten at det forringes, er så langt vanskelig å ta stilling til. For å kunne bestemme dette presist, kreves et ferdig design og en reell oppkobling. Det kan allikevel antas at det er rimelig å definere $A = 0.9$. Dette vil innebære en 90% reduksjon i signalets amplitude samt $\approx$ 11 ganger forsterkning av det resterende signal.

Ved å definere $A = 0.90$, kan vi se fra tabell 4.4 at den laveste forventede feltstyrkeendingen vil resultere i 0.61 bit endring i en 14 bit ADC og 2.45 bit endring i en 16 bit ADC. Siden en endring under 1 bit ikke er i stand til skille mellom de laveste endringene i magnetfeltet er 14 bit ADC ikke egnet i denne oppgaven. En ADC med 16 bit oppløsning gir derimot en endring på 2.45 bit og det kan konkluderes med at dette er en tilstrekkelig oppløsning og at en 16 bit ADC kan benyttes i denne oppgaven.



Figur 4.13: Plott av signal fra sensor, samt dekodet signal med ulik signalforsterlkning.

Plottet i figur 4.13 er ment for å illustrere forskjellene mellom resulterende signal når det benyttes forsterkning lik 1 og forsterkning lik 10. Som denne figuren viser, er det stor forskjell i oppløsning mellom de to dekodede signal og vi ser tydelig fordelen med dynamisk signaltilpassning av V_{In-}.

Det er også viktig å se litt på hvor stort standardavvikene mellom de dekodede- og det faktiske-signalet er. For å komme frem til dette benyttes:

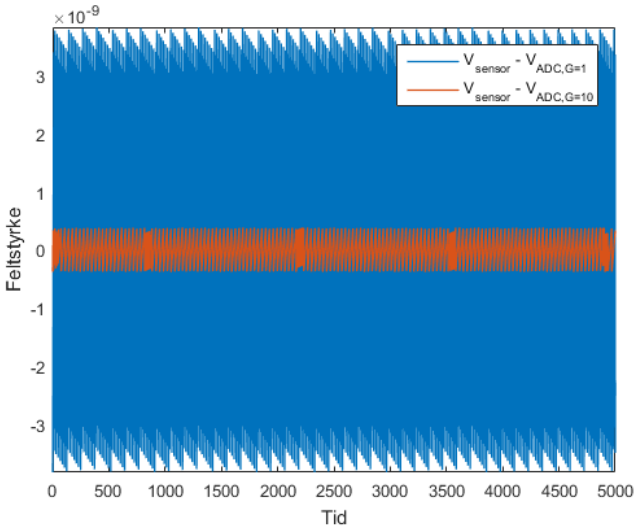
$$\sigma_G = SD\left(\sum(Signal - Dekodet)\right) \tag{4.10}$$

Vi setter inn verdier i ligning 4.10 og finner da standardavviket mellom dekodet og faktisk signal.

Videre ønsker vi å se på forholdet mellom avvikene og setter disse opp i ligning 4.11.

$$\begin{aligned} \sigma_{G=1} &= 2.21nT \\ \sigma_{G=10} &= 0.221nT \\ \gamma &= \frac{\sigma_{G=1}}{\sigma_{G=10}} = 9.88 \end{aligned} \tag{4.11}$$

Som ligning 4.11 viser, forbedres standardavviket omtrent 10 ganger ved å benytte forelåtte metode. Forskjellene illustreres i figur 4.14, der det plottes differansen mellom faktisk signal og dekodet.



Figur 4.14: Målefeil ved ulik forsterkning.

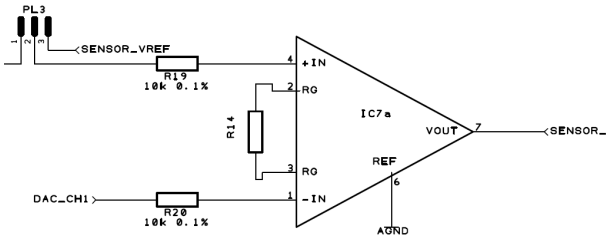
Som vi ser her har det dekodede signalet med 1 i forsterkning langt høyere avvik enn hva som er tilfellet med høyere forsterkning. Fra disse to figurene, samt resultatet fra ligning 4.11, kan det konkluderes med at ved å signaltilpasse V_{In+} forbedres sensitiviteten til det dekodede signalet, samtidig som målefeilen reduseres. Dette synes å være løsningen vi søkte da vi startet med å se på hvordan signalets amplitude og sensitivitet kunne økes. Jeg vil derfor benytte dette videre i oppgaven.

- **Tidsdiskret, modifisert ide**

Vi må nå se på hvordan referansesignalet V_{In-} kan produseres på best mulig måte. Til denne problemstillingen synes det å benytte en styrt spenningskilde som den mest optimale løsningen. Realiseringen av denne kilden kan velges å gjøres rent analogt med diskrete komponenter, eller ved hjelp av en DAC. Det er fordeler og ulemper med begge løsningene, men jeg anser presisjonen til en DAC som høyere enn hva som er mulig å forvente av et diskret analogt trinn. Det velges dermed å benytte en DAC for å produsere ønsket referansesignal.

For at ligning 4.8 skal være gyldig i et tidsdiskret system, må denne diskretiseres. Diskretisert ligning blir:

$$V_{out}(k) = G \cdot \big(V_{In+}(k) - A \cdot V_{In+}(k-1)\big) \tag{4.12}$$



Figur 4.15: Analog signaltilpassning.

Figur 4.15 viser et utdrag fra kretsskjema, hvordan den analoge signaltilpassningen er tenkt utført.

Når overnevnte fremgangsmåte benyttes, er det viktig å kjenne den faktiske forsterkningen som hver av aksene måtte ha. For å bestemme denne forsterkningen er det valgt å benytte kalibrering av hver enkelt instrumenteringsforsterker med tilhørende DAC. Denne kalibreringen utføres ved å måle en fast og kjent spenning som i figur 4.15 er *SENSOR_VREF* og samtidig variere spenningen fra DAC.

Programvaren kan deretter beregne den faktiske forsterkningen til hver kanal ved følgende ligning:

$$G_{Faktisk} = \frac{\Delta V_o}{V_{Ref} - \Delta V_{DAC}} \tag{4.13}$$

Når den faktiske forsterkningen, $G_{Faktisk}$, er beregnet kan vi i programvaren benytte den til å beregne det faktiske sensorsignalet. Hvordan dette løses forklares nærmere i kap 5.1.4

- **Beregne forsterkning**

Vi skal nå beregne forsterkningen G og tar da utgangspunkt i ligning 4.5. I figur 4.15 kan resistansen R_G gjennkjennes som $R14$.

$$\begin{aligned} R_G &= \frac{49.4k\Omega}{G-1} \\ &= \frac{49.4k\Omega}{11-1} \end{aligned}$$

$$R_G = 4.94k\Omega \tag{4.14}$$

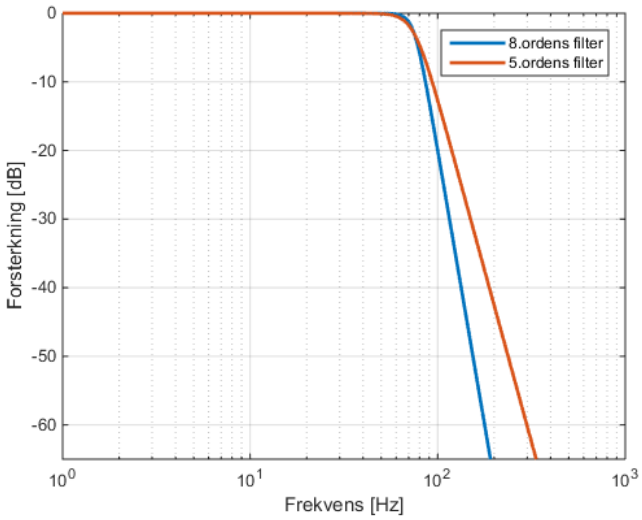
Standardkomponenter ønskes benyttet og finner nærmeste komponentverdi til å være $4.99k\Omega$ Før den nye verdien godkjennes, må det sjekkes hvilke endringer denne medfører. Det beregnes ny forsterkning ved hjelp av ligning 4.4 og finner resulterende forsterkning til å være 10.89. Dette er en svært liten endring i forhold til den tidligere beregnede forsterkning og anses som uproblematisk. For å få en forsterkning som er minst mulig avhengig av ytre påvirkninger som variajon i temperatur, er det her svært viktig å velge presisjons resistor med lavest mulig drift. I tilfeller der det inngår flere resistorer i serie/parallell er det viktig å også se etter at samtlige har lik drift.

Det er nå vist hvordan signalet kan tilpasses til å kunne måles med en 16bit ADC med $V_{Ref} = 5.0V$. I neste delkapittel er det nå naturlig å se på hvordan båndbredden til signalet fra sensor må tilpasses for å unngå uønsket aliasing.

Lavpassfiltrering av signal

Det anses kjent at forsyningsnettet i Norge har en normalfrekvens på 50Hz, med marginale avvik. Når vi ønsker å måle magnetfeltet som dette genererer, er det viktig at systemet vårt har en tilstrekkelig båndbredde, slik at alle frekvenskomponenter i signalet måles korrekt. Dette innebærer at den minimale båndbredden til systemet ikke bør være mindre enn $f_{bw} > 2 \cdot f_{Signal}$ for å unngå dempning.

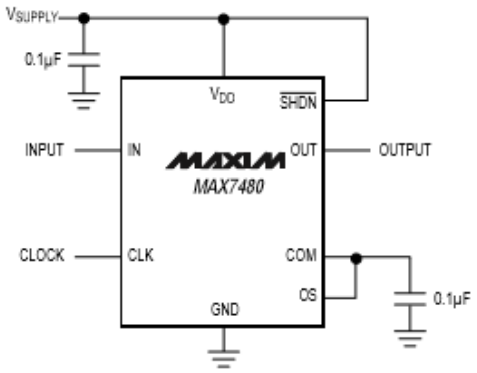
Vi starter med å se på systemets knekkfrekvens $f_c = f_{bw}$ og hvordan denne velges. Når systemets knekkfrekvens skal velges, er det viktig å velge en knekkfrekvens,f_c, som er tilpasset det gitte system. Det må fokuseres på å ha en tilstrekkelig høy knekkfrekvens for å unngå uønsket signaldempning i det frekvensområdet som anses som passbånd. Samtidig er det ønskelig å beholde samplefrekvensen, f_s, på et fornuftig nivå. Transisjonsområdet som spenner seg mellom f_c og $f_{s/2}$ er ønskelig å minimere.



Figur 4.16: Frekvensrespons ved ulike filterordner.

Figur 4.16 viser frekvensresponsen til to filtere, der begge har Butterworth karakteristikk, men er av ulik orden. Vi ser tydelig at transisjonsområdet er betydelig større for 5.ordens filter, sammenliknet med 8.ordens filter. Siden det er ønskelig å minimere det nevnte området velges det å benytte filter av 8.orden.

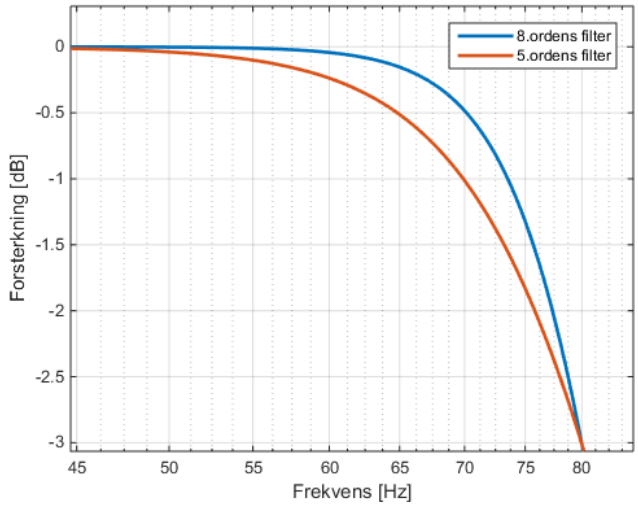
Hovedsaklig basert på tidligere gode erfaringer med svisjet kapasitansfilter fra Maxim, er det ønskelig å også her i denne oppgaven benytte å et slikt filter. Det er videre ønskelig å velge et filter som krever ensidig spenningsforsyning. Ved søk er det kun filter MAX7480 [18] som oppfyller spesifikasjoner, 8.ordens Butterworth-karakteristikk med ensidig forsyning. Det velges å benytte filteret MAX7480 i oppgaven.



Figur 4.17: MAX7480 [18].

Figur 4.17 viser en typisk oppkobling av filteret og her observeres filterets beskjedne krav til ytre komponenter. Filteret MAX7480 er svært fleksibelt og lar seg lett tilpasse ønsket knekkfrekvens. Dette gjøres ved å påtrykke et klokkesignal som er tilpasset knekkfrekvensen på teminal CLK. Med dette oppnås det å kunne styre knekkfrekvensen på et senere tidspunkt og denne muligheten medfører stor fleksibilitet. Dersom det ikke er ønskelig å påtrykke klokkesignal, kan den interne oscillator benyttes for å danne klokkesignalet internt i kretsen. Denne funsjonen krever en ekstra kondensator C_{osc} som kobles mellom CLK terminal og jord.

Begge mulighetene antas å gi fullgode resultater. Allikevel foretrekkes det å benytte intern oscillator og ekstern kondensator. Med denne løsningen unngås det å distribuere høyfrekvente klokkesignaler ut på kretskortet og det unngås dermed at klokkesignalene kan påvirke sensitive sensorsignaler.



Figur 4.18: Frekvensrespons ved ulike filterordner og $f_c = 80Hz$.

Når båndbredden og dermed knekkfrekvensen til systemet skal velges, vil det forsøkes å finne den laveste knekkfrekvens som ikke demper signalet i passbåndet. Det er valgt at 99% av signalet i passbånd skal bibeholdes. I decibel resulterer dette i at største dempning i signalets passbånd er $-0.087dB$. Dette setter imidlertid store krav til filter og valgt knekkfrekvens. Som vi vil se senere i oppgaven, er det viktig å finne den laveste knekkfrekvensen, da en liten endring i denne vil resultere i en stor endring i samplefrekvens. Knekkfrekvensen som benyttes i plottet i figur 4.18 er $80Hz$. Dette synes å være en passende knekkfrekvens dersom 8.ordens filter benyttes. Siden det allerede er besluttet å benytte 8.ordens filter kan det nå besluttes at $f_c = 80Hz$. Valg av knekkfrekvens medfører en signaldempning i passbåndet på $-0.047dB$. Ved å sammenlikne dette med kravet til største dempning, finner vi at det er godt innenfor krav.

Det neste som må avklares er amplituden til frekvenser lik $f_{s/2}$ og høyere, altså frekvenser i stoppbåndet. Fra [3] er det gitt at et fornuftig krav til amplitude til frekvenser over $f_{s/2}$ er

$$G(f_{s/2})_{dB} \leq 20 \cdot \log\left(\frac{1}{2}V_{ADC_{LSB}}\right) \tag{4.15}$$

Vi setter inn i ligning 4.15 og beregner ønsket dempning ved $f_{s/2}$

$$\begin{aligned} G(f_{s/2})_{dB} &\leq 20 \cdot \log\left(\frac{1}{2} \cdot \frac{3.3V}{2^{16}}\right) \\ G(f_{s/2})_{dB} &\leq -91.97dB \end{aligned} \tag{4.16}$$

I praksis kan det imidlertid bli vanskelig å oppnå kravet til dempning ved $f_{s/2}$, men ved å velge kvalitetskomponenter vil det allikevel tilstrebes å nærme seg dette.

Nå som filterets knekkfrekvens, orden og minste amplitude ved $f_{s/2}$ er bestemt, kan vi beregne

den minimale samplefrekvensen som kreves. Fra [3] finner vi ligning 5.9, som med noen modifikasjoner er nyttig i denne sammenheng.

$$G(f) = \frac{G_0}{\sqrt{1 + \left(\frac{f_{s/2}}{f_c}\right)^{2n}}}$$
$$G(f) \approx G_0 \cdot \left(\frac{f_c}{f_{s/2}}\right)^n \tag{4.17}$$

Hvor $G(f)$ er signalamplituden og G_0 er forsterkning i passbånd

Ligning 4.17 gir et utgangspunkt for å finne den nødvendige samplefrekvensen f_s.

$$G(f_{s/2})_{dB} = 20\log\left(G_0 \cdot \left(\frac{f_c}{f_{s/2}}\right)^n\right)$$
$$G(f_{s/2})_{dB} = 20\log\left(G_0\right) + 20n \cdot \log\left(f_c\right) - 20n \cdot \log\left(f_{s/2}\right)$$
$$\log\left(f_{s/2}\right) = \frac{G(f_{s/2})_{dB} - 20n \cdot \log\left(f_c\right)}{-20n}$$
$$f_{s/2} = 10^{\left(\frac{G(f_{s/2})_{dB} - 20n \cdot \log\left(f_c\right)}{-20n}\right)} \tag{4.18}$$

Fra ligning 4.18 kan det nå beregnes den laveste samplefrekvens som kan benyttes. Setter derfor inn tall i ligningen og kommer frem til:

$$f_{s/2} = 10^{\left(\frac{-92dB - 20\cdot 8\cdot \log(80)}{-20\cdot 8}\right)}$$
$$f_{s/2} \approx 301Hz$$

Vi har nå beregnet den minimale samplefrekvens som systemet må ha for å unngå aliasing. I utregningen ovenfor finner vi at minste samplefrekvens til systemet blir $2 \cdot f_{s/2}$, som tilsvarer $f_s \geq 602Hz$. Kravet til minste dempning av frekvenser $G(f_{s/2})_{dB}$, som beregnet ovenfor kan virke noe strengt. Vi skal derfor se på hvordan mer moderate krav påvirker systemets minimale samplefrekvens.

Som tabell 4.5 viser, har kravet til dempning $G(f_{s/2})_{dB}$ stor betydning for hvor ofte systemet må sample signalet. Den endelige realiseringen må tilpasses kravet slik at samplefrekvens avstemmes med mikrokontrollerens interne klokkefrekvens.

$G(f_{s/2})_{dB}$	Nyquistfrekvens, $f_{s/2}$	Samplefrekvens, f_s
$-20dB$	$106.68Hz$	$213.36Hz$
$-40dB$	$142.26Hz$	$284.52Hz$
$-60dB$	$189.71Hz$	$379.41Hz$
$-80dB$	$252.98Hz$	$505.96Hz$
$-92dB$	$300.67Hz$	$601.34Hz$
$-100dB$	$337.36Hz$	$674.71Hz$

Tabell 4.5: Krav til samplefrekvens, f_s.

Vi fortsetter nå med å vise hvordan filter kondensatoren C_{osc} beregnes.

Fra [18] er sammenhengen mellom knekkfrekvens f_c og filterets oscillator definert som:

$$f_c = \frac{f_{CLK}}{100} \tag{4.19}$$

Ligning 4.19 viser at filterets interne frekvens er 100 ganger signalets knekkfrekvens f_c. Vi ordner ligningen og finner et uttrykk for f_{CLK}:

$$\begin{aligned} f_{CLK} &= 100 \cdot f_c \\ f_{CLK} &= 100 \cdot 80Hz \\ f_{CLK} &= 8.0kHz \end{aligned} \tag{4.20}$$

Videre finner vi også sammenhengen mellom den interne oscillatoren f_{osc} og filterkondensatoren C_{osc} til å være definert som:

$$f_{osc_{(kHz)}} = \frac{53 \cdot 10^3}{C_{osc_{(pF)}}} \tag{4.21}$$

Vi ordner ligning 4.21 og finner et utrykk for $C_{osc_{(pF)}}$:

$$\begin{aligned} C_{osc_{(pF)}} &= \frac{53 \cdot 10^3}{f_{osc_{(kHz)}}} \\ C_{osc_{(pF)}} &= \frac{53 \cdot 10^3}{8.0} \\ C_{osc_{(pF)}} &= 6625pF \end{aligned}$$

Vi ønsker å benytte standardverdier og finner tilgjengelig to komponenter som ligger nært i verdi. Disse, samt endringen i knekkfrekvensen som medføres, oppsummeres i tabellen nedenfor.

For å unngå uønsket dempning i det frekvensområdet som skal måles, rundt 50Hz, er det som tidligere nevnt viktig at knekkfrekvensen blir plassert riktig. Dersom den plasseres for lavt, kan transisjonsområdet nærme seg 50Hz og dette kan da resultere i redusert amplitude i det

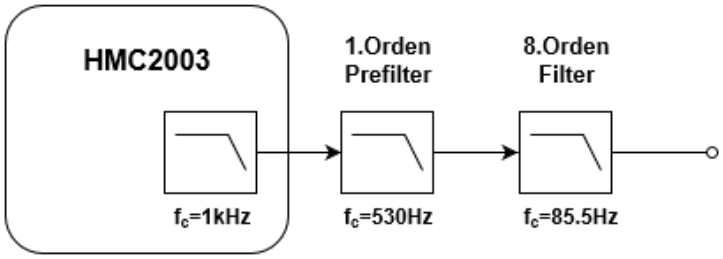
$C_{osc_{(pF)}}$	f_{osc}	Knekkfrekvens, f_c	Nyquistfrekvens, $f_{s/2}$	Samplefrekvens, f_s
$6625pF$	$8.0kHz$	$80.0Hz$	$300.57Hz$	$601.34Hz$
$6200pF$	$8.55kHz$	$85.5Hz$	$321.28Hz$	$642.56Hz$
$6800pF$	$7.79kHz$	$77.9Hz$	$292.93Hz$	$585.86.34Hz$

Tabell 4.6: Endring av knekkfrekvens og samplefrekvens som følge av kondensator valg.

frekvensområdet som informasjonen hentes fra. Hvis den derimot plasseres for høyt, vil eneste konsekvens være høyere samplefrekvens.

Når vi studerer konsekvensene av å måtte benytte standardverdier i tabell 4.6, finner vi kondensator med verdi $6800pF$ nærmest den ønskede. Men som vi ser, bidrar denne til en reduksjon i knekkfrekvens og basert på overnevte grunner, velges derfor ikke denne verdien. For å være på den sikre siden velges kondensator med verdi $6200pF$. Som forklart ovenfor vil denne bidra til en uproblematisk økning i knekkfrevens og samplefrekvens. Dersom vi ønsker å redusere samplefrekvensen på et senere tidspunkt, kan vi da gjøre vurderingene til amplituden $G(f_{s/2})_{dB}$ på ny. En reduksjon i kravet til dempningen $G(f_{s/2})_{dB}$ vil som tidligere forklart medføre lavere krav til samplefrekvens. For å unngå ustabiliteter og drift i filteret er det valgt å benytte kondensator av typen COG.

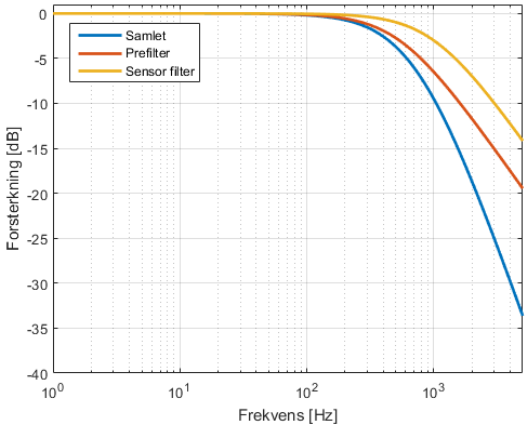
For å unngå aliasing i det svitsja kapasitansfilteret MAX7480 har også filteret en Nyquistfrekvens som må overholdes. Denne tilsvarer frekvensen $f_{s/2}$ til en ADC og blir her bestemt av $f_{osc/2}$. Vi vil derfor se på hvordan signalets båndbredde tilpasses filterets Nyquistfrekvens.



Figur 4.19: Analog filterbank.

Som figur 4.19 illustrerer, består den analoge filterbanken av flere ulike lavpassfiltere. For å finne en samlet knekkfrekvens til de to første filterne, starter vi med å se på lavpassfilteret i sensoren.

Fra [13] er det gitt at HMC2003 sensoren har realisert et lavpassfilter med knekkfrekvens $f_{-3dB} = 1kHz$. Filterets orden er imidlertid ikke oppgitt i tilhørende datablad. Det må derfor gjøres en kvalifisert gjettning av filterets orden og for å være på den sikre siden antar jeg at 1.ordens modell kan benyttes. Hvis det derimot er benyttet et høyere ordens filter i sensoren, vil eneste konsekvens være at det oppnås en større dempning av de uønskede frekvenser i forhold til først antatt.



Figur 4.20: Frekvensrespons til analog filterbank.

Det neste filteret vi må se på, er det som i figur 4.19 kalles prefilter. Av de to første filterne er det kun dette filteret som vi kan påvirke slik at det samlet oppnås ønskede egenskaper. Når prefilterets knekkfrekvens skal bestemmes er det valgt å fokusere på at prefilteret ikke skal medføre dempning i passbånd. Samtidig er det viktig at filterne samlet er i stand til å dempe de uønskede frekvenskomponenter tilstrekkelig. For at disse filterne ikke skal påvirke frekvenskomponentene i passbåndet er det ved hjelp av simuleringer, som vist i figur 4.20, kommet frem til en knekkfrekvens 6 ganger høyere enn f_c. Ved å benytte så høy knekkfrekvens på prefilteret vil det påvirke frekvenskomponenter i passbånd marginalt, samtidig som vi samlet oppnår en dempning på $-30dB$ ved $f_{osc/2}$. Det er valgt å opprettholde kravet at dempning i passbåndet ikke skal overstige $0.087dB$ og ved valgt knekkfrekvens overholdes kravet.

Det skal nå beregnes komponentverdiene som skal inngå i prefilteret. Fra simuleringen er det gitt at

$$f_{c,RC} = 530Hz$$

Knekkfrekvensen til et 1.ordens lavpassfilter er som kjent

$$f_{c,RC} = \frac{1}{2\pi RC} \tag{4.22}$$

Ligning 4.22 viser at kun R og C er ukjente verdier. Det velges å benytte en standard kondensator med verdi $C = 33nF$ og nå kan ligningen løses med hensyn på R.

$$R = \frac{1}{2\pi f_{c,RC} C}$$
$$R = \frac{1}{2\pi \cdot 530Hz \cdot 33 \cdot 10^{-9}}$$
$$R = 9099\Omega$$

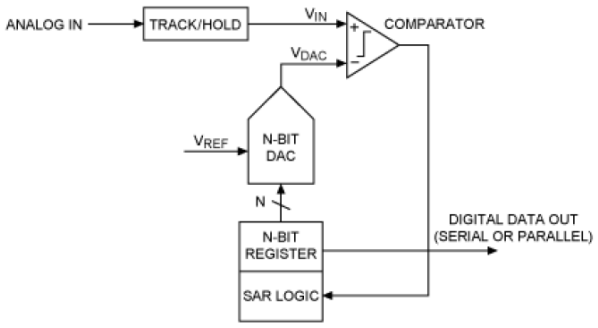
Fordi det er ønskelig å benytte standardverdier, benyttes resistor med nærmeste tilgjengelige verdi. Vi finner nærmeste verdi som er $R = 9.09k\Omega$ til å passe bra med beregninger ovenfor.

4.1.3 Analog til digitalomformer

Introduksjon

Det er fra [19] gitt at *Successive-Approximation-Register* (SAR) analog til digitalomformer er ofte det beste valget når det kreves medium til høy oppløsning av det analoge signalet. Oppløsningen der SAR benyttes er som oftest mellom 8 og 16 bit. SAR er sammenliknet med tilsvarende teknikker er å anse som strømgjerrig og ofte kan disse implementeres i en fysisk mindre krets. Denne kombinasjon av egenskaper fører til at denne type ADC er ideelle for en rekke applikasjoner.

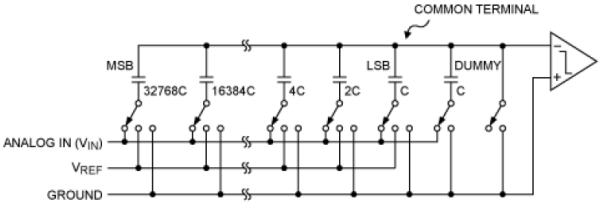
I denne oppgaven velges en ADC som benytter seg av metoden SAR. Vi starter dette delkapittelet med å gi en kortfattet introduksjon til hvordan denne teknikken konverterer et analogt signal til en digital verdi.



Figur 4.21: Forenklet SAR arkitektur [19].

Figur 4.21 viser blokkdiagram over en forenklet SAR arkitektur. Det skal nå forsøkes å forklare den mest grunnleggende teorien til sentrale blokker i figuren.

Vi starter med å se på blokken Track/Hold. Denne blokken kan anses som et 0.ordens filter. Filteret vil nå, ved hjelp av en styrt bryter, lade opp selve holdeelementet, som her er en kondensator. Dette filteret vil ved bestemte tider ta punktprøver av det tidsvarierende analoge sinalet som skal måles. Når en ny punktprøve foreligger, kan selve dekodingen av signalet starte.



Figur 4.22: Eksempel på en 16bit DAC [19].

Når vi skal omforme den analoge punktprøven fra *Track/Hold* blokken, benyttes en kombinasjon av DAC, komperator samt SAR logikk som vist i figur 4.21 og 4.22.

SAR logikken vil nå forsøke ved hjelp av en binær søkealgoritme å dekode det analoge signalet til en digital verdi. Fra figur 4.22 ser vi at DAC'en er konstruert av flere brytere som sender valgt signal til komperatoren. Ved hjelp av disse bryterne vil søkealgoritmen kunne velge hvilket bit som skal dekodes og resultatet hentes fra komperatorens utgang.

Prinsippet bak søkealgoritmen vises i algoritme 4.1.

Algorithm 4.1 SAR binært søketre

Data: V_{Ref}, V_{Analog}
Result: dekodet
begin
  $n \longleftarrow AntallBit$
  $dekodet \longleftarrow 0$
  for $(n \longleftarrow n-1) > 0$ **do**
    $V_{DAC} = \frac{V_{Ref}}{2^{(AntallBit-n+1)}}$
    if $V_{Analog} > V_{DAC}$ **then**
      $dekodet(n) \longleftarrow 1$
    else
      $dekodet(n) \longleftarrow 0$
    end
  end
end

Som vi ser fra algoritmen starter den binære søkealgoritmen med å bestemme MSB tilhørende den dekodede verdien. Dette gjøres ved å sjekke om det analoge signalet er større enn $V_{Ref/2}$. Dersom den analoge verdien er større, blir øverste bit i den dekodede verdien satt til 1. Hvis den derimot er mindre, blir verdien satt til 0. Søkealgoritmen vi så iterere seg gjennom samtlige tilgjengelige bit, der den ved hver iterasjon halvverer V_{Ref} for så å utføre nevnte sjekk. Når algortimen har søkt gjennom hele analog til digitalomformerens oppløsning, returneres den dekodede verdien til intern logikk som oppbevarer den dekode verdien til den leses av eller overskrives.

Som vi ser er det flere faktorer som direkte påvirker presisjonen til den dekodede verdien. Foruten presisjonen til den interne V_{Ref}, er DAC og komperator de mest kritiske komponentene i en ADC som benytter SAR teknologi. Feil eller avvik i DAC eller komperator vil i verste fall kunne medføre at vi havner på feil side av det binære søketreet. Dette vil da medføre uønsket bitfeil som ikke kan korrigeres senere. Når vi siden skal sammenlikne ulike analog til digitalomformere blir presisjonen som nevnt her et viktig kriterie for valg av komponent.

For at signalene skal dekodes mest mulig korrekt og ikke minst likt er det ønskelig å benytte en ADC med felles intern referanse. Vi oppnår da at en eventuell referansefeil vil gjennspeiles på alle 3 akser. Dersom en slik feil skulle forekomme vil jeg anta at siden dette ville resultert i en

felles dempning på alle akser, ikke vil kunne påvirke navigasjonsalgoritmer. Presisjonen til V_{Ref} anses dermed som et mindre viktig kriterie.

Krav til ADC

I forrige delkapittel argumenterte vi for å benytte en ADC med 16bit oppløsning og vi fant denne oppløsningen tilstrekkelig. Vi vil derfor kun sammenlikne analog til digitalomformere som innehar denne oppløsningen.

For å kunne sette flere krav til omformeren må vi først vurdere hva vi ønsker å måle. Feltstyrkene som ønskes målt fra HMC2003 sensor har på tross av forsterkningen, en relativt lav amplitude. Som nevnt ovenfor søker vi etter en ADC med felles intern referanse for å kunne dekode samtlige signaler med nøyaktig samme referanse. Som nevnt, vil derfor et avvik i referansen resultere i en felles feil på alle aksene og så lenge den gjenspeiles likt på samtlige akser, anses avviket som uproblematisk.

Som kjent er magnetfeltet som ønskes målt både tids og posisjonsavhengig og må nå vurdere hvordan dette måles på best mulig måte. Fra [25] er det gitt som et av resultatene i oppgaven, at en kan ved hjelp av fase og amplitude beregne nåværende posisjon i et magnetfelt. For å opprettholde nødvendig presisjon i beregning av fase og amplitude er vi helt avhengig av at feltstyrkene blir målt på nøyaktig samme tidspunkt. I søket etter ADC vil derfor dette danne grunnlag for krav om simultankonvertering av minst 3 kanaler.

Innledende krav til egnet ADC kan oppsummeres i følgende punkter:

1) - 16bit oppløsning.

2) - 4 separate kanaler.

3) - Parallellkonvertering med minimum 700 sampler per sekund.

4) - $V_{Ref} = 5.0V$ og $V_{IO} = 3.3V$

4) - Kretsen må kunne monteres uten hjelp av avansert utstyr.

Ved søk etter ADC som oppfyller de 4 innledende krav viser det seg at svært få komponeter er i stand til å oppfylle kravet med 4 separate kanaler med parallellkonvertering. Jeg lykkes imidlertid med å finne 2 ulike analog til digitalomformere som oppfyller kravene vi har definert ovenfor. Disse blir presentert i tabell 4.7.

Som tabell 4.7 viser, har vi funnet to nesten identiske analog til digitalomformere. For å bestemme hvilken som passer best for denne oppgaven vil jeg nå sammenlikne de viktigste feil parametere fra tilhørende datablader [22] og [21]. For en fullstendig oversikt over komponentenes egenskaper anbefales det å studere nevnte datablader.

[1] 08.03.16 - www.digikey.com

Fabrikat	Modell	Pakke	Pris
Maxim Integrated	MAX11047ECB+ -ND	64 TQFP	11.66$ [1]
Maxim Integrated	MAX11044ECB+	64 TQFP	12.25$ [1]

Tabell 4.7: Aktuelle analog til digitalomformere.

Modell	INL_{Typ}	DNL_{Typ}	$Offset_{Typ}$	$Gain_{Max}$	$SFDR_{Typ}$	I_{AVDD}	I_{DVDD}
MAX11047	±0.65	±0.7	±0.001	±0.0012	108	25mA	5.5mA
MAX11044	±0.4	±0.4	±0.001	±0.015	104	30mA	5.5mA

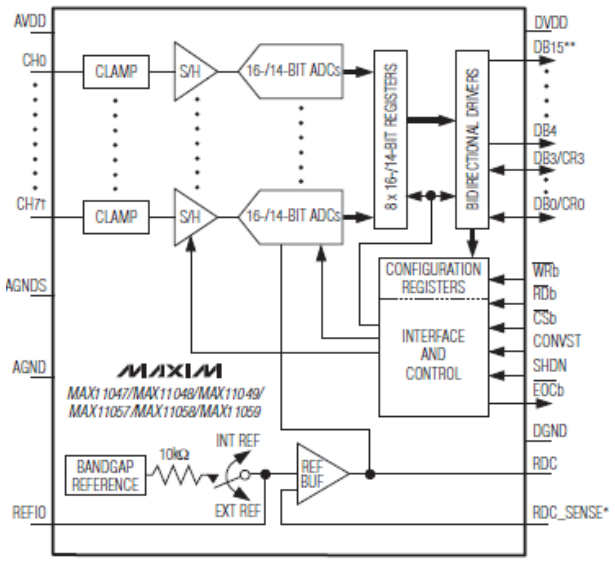
Tabell 4.8: Sammenlikning av analog til digitalomformere.

Ved å sammenlikne parametere tilhørende kretsene som listes i tabell 4.8 finner vi at kretsen MAX11044 har marginalt bedre spesifikasjoner. Tabell 4.7 viser at prisen for en MAX11044 er marginalt høyere. Differansen i pris er allikevel så lav at det ikke vil påvirke valg av krets. Det som derimot påviker valget av krets er hvordan kretsene måler signalet. Fra datablad [22] og [21] fremkommer det at MAX11044 måler et bipolart signal ($\pm 5.0V$) og MAX11047 måler et unipolart ($0..5V$). Da sistnevnte er mest egnet velges det å benytte analog til digitalomformeren MAX11047 [22].

Analog til Digitalomformer - MAX11047

Vi skal nå se litt på virkemåten til den valgte analog til digitalomformeren, MAX11047. Funksjoner og muligheter som ønskes benyttet i oppgaven vil her bli vektlagt og det henvises til datablad [21] for en dypere forklaring av kretsen.

Det er fra [21] gitt at kretsen MAX11047 er bestykket med 4 uavhengige SAR analog til digitalomformere som kommuniserer via felles digitalt grensesnitt. Samplefrekvensen til kretsen er oppgitt til maksimalt 250ksps med $3\mu s$ konverteringstid. Som ønsket har denne omformeren også mulighet for å benytte inten felles referanse.

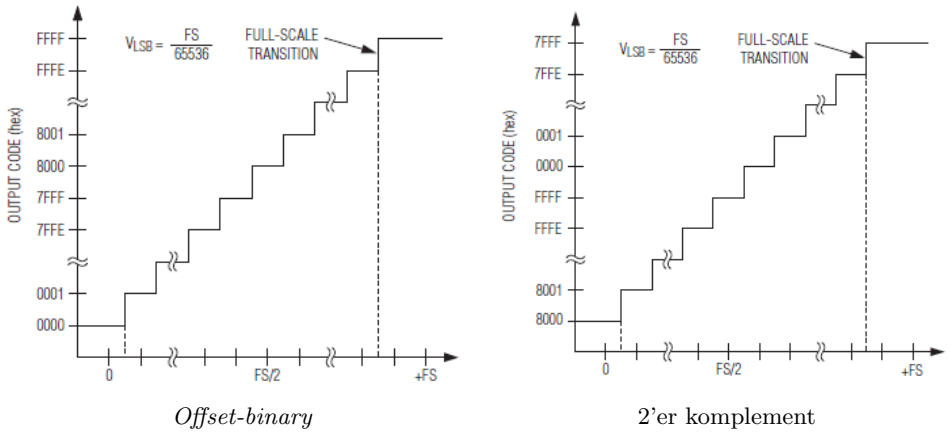


Figur 4.23: MAX11047 [21]. Figuren viser ulike varianter av en familie analog til digitalomformere. Valgt krets har 4 parallelle kanaler.

Figur 4.23 viser funksjonsdiagrammet tilhørende MAX11047. Vi skal nå studere de to grensesnittene som denne kretsen er bestykket med.

Analogt grensesnitt - Med det analoge grensesnittet, menes de blokkene i figur 4.23 som behandler det analoge signalet som påtrykkes terminal CH0 til CH3 (4 kanaler). I MAX11047 består dette av 4 separate kanaler, der hver tilsvarer SAR arkitekturen som vises i figur 4.21.

Analog til digitalomformeren MAX11047 dekoder det analoget signalet som unipolart, det vil si at den måler det analoge signalet fra $0..5V$. Kretsen kan representere de målte signaler som enten toerkompliment eller *offsett-binary*. Disse ulike måtene å representere det konverterte signalet på, vises i figur 4.24.



Offset-binary 2'er komplement

Figur 4.24: Alternative representasjoner av det dekodede signalet [22].

I denne oppgaven synes det å passe best med *offsett-binary* representasjon. Det konverterte signalet har da følgende transferfunksjon:

$$Dekodet = \frac{V_{IN} \cdot 2^{16}}{V_{Ref} \cdot \frac{5}{4.096}} \tag{4.23}$$

De analoge inngangene er kostruert slik at de ikke tar skade av signalspenninger mindre enn 7V. For å gjøre inngangstrinnene enda mer robust og dermed ikke ødelegges av signalspenning over 7V, er hver av kanalene utstyrt med over og underspeningsvern. I figur 4.23 illustreres denne beskyttelsen med blokken *CLAMP*. Vernet er i stand til å kompensere med $\pm 20mA$ for å redusere signalspenningen til et nivå som ikke er skadelig for den analoge inngangen. For at dette vernet skal fungere er det avhengig av en resistor,R_S, i serie med signalet. Fra datablad er det gitt at resistorer beregnes ved:

$$R_S = \frac{V_{FAULT,MAX} - 7V}{20mA} \tag{4.24}$$

Siden det ikke anvendes signalspenninger høyere enn 5V vil ikke noen av kretsene være i stand til å produsere en signalspenning over 7V. Denne funksjonen anses derfor ikke nyttig i forhold til å beskytte ADC mot for høy signalspenning, generert av en komponent montert på kretskortet. Funksjonen kan allikevel vise seg nyttig som transientbeskyttelse. Det settes inn litt romslige verdier i ligning 4.24 og finner:

$$R_S = \frac{8.2V - 7V}{20mA} = 60\Omega \tag{4.25}$$

R_S vil nå kunne begrense transienter noe, samtidig som resistoren ikke vil medføre uønsket dempning av signalet.

Digitalt grensesnitt - Det digitale grensesnittet for kommunikasjon med mikrokontrolleren, er et bidirektionalt parallellt grensesnitt. Dette grensesnittet benyttes for både konfigurering av ADC samt avlesning av dekodede verdier. En oversikt over dette grensesnittet finner vi i tabell 4.9.

Funksjon	DB0	DB1	DB2	DB3	DB4 $\cdots$ DB15
Konfigurasjon $\overline{WR} = 0$ $\overline{RD} = 1$	CR0	CR1	CR2	CR3	
	Konfigurasjonsregister				
Data $\overline{WR} = 1$ $\overline{RD} = 0$	Data ut				

Tabell 4.9: Parallellt digitalt grensesnitt.

I tillegg til det parallelle grensesnittet har kretsen også et digitalt grensesnitt som benyttes

for å styre kretsen. Dette grensesnittet består av:

Signal	Retning	Funksjon
$\overline{WR}$, *Write*	Inn	$\overline{WR}$ går fra fra lav til høy, skriver data til konfigurasjonsregister.
$\overline{RD}$, *Read*	Inn	$\overline{RD}$ går fra høy til lav, presentes neste kanals data på paralsellbuss.
$\overline{CS}$, *Chip Select*	Inn	$\overline{CS}$ lav, aktiverer kretsens grensesnitt.
$\overline{EOC}$, *End of conversion*	Ut	$\overline{EOC}$ høy, indikerer konvertering ferdig og nye data klar til avlesning.
$CONVST$	Inn	$CONVST$ går fra lav til høy, starter dekodingen av de analoge signalene.

Tabell 4.10: Grensesnitt for konfigurasjon.

Vi har nå sett på de tilgengelige grensesnittene og skal nå se litt på hvilke konfigurasjonsmuligheter kretsen har.

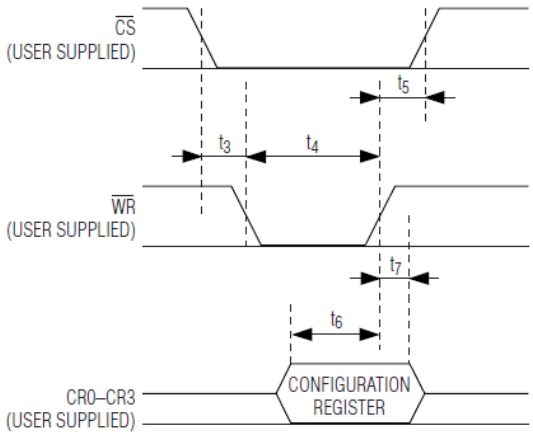
	Tilstand	
Register	0	1
CR0	$CONVST$ styrer når neste konvertering starter	Neste konvertering startes så snart som sist konvertering er ferdig.
CR1	Tilstand skal være 0	
CR2	Offset representasjon av dekodet signal	2'er komplement representasjon av dekodet signal
CR3	Intern referanse	Ekstern referanse

Tabell 4.11: Konfigurasjons register.

Ved å studere konfigurasjonsmulighetene som er listet i tabell 4.11 kan vi finne et bitmønster som er tilpasset det vi ønsker. I denne oppgaven velges det å konfigurere kretsen slik at programvaren kan styre når kretsen skal sample, det vil si CR0 = 0. Det er videre ønskelig å benytte muligheten som offset repersentasjon av det dekodede signalet tilbyr, det vil her innebære CR2=0. Til slutt velges det å benytte den interne referansen, som innebærer CR3=0.

Vi skal nå se på hvilke definerte krav det er til pulstider, som vi må forholde oss til, når vi skal skrive eller lese data fra kretsen.

Først ser vi på definerte tidsmessige krav til pulser når vi skal programmere konfigurasjonsregisteret.



Figur 4.25: Tidsdiagram for programmering [21].

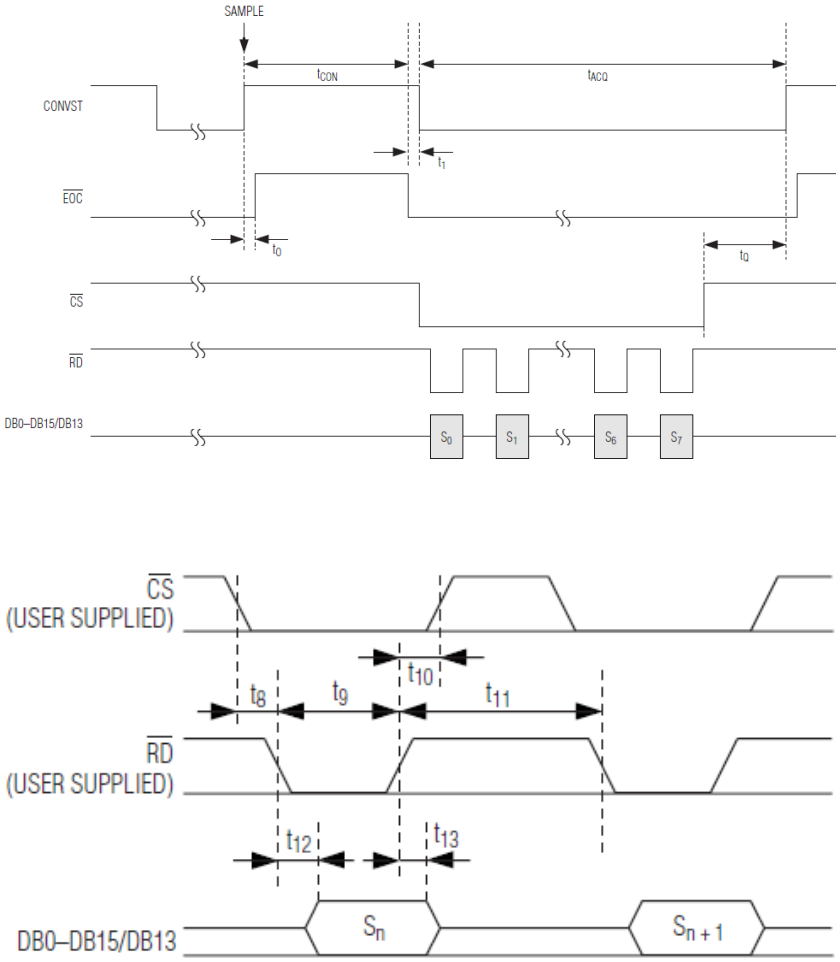
Hvor tidene er gitt i tabell 4.12

Parameter	Symbol	Min	Enhet
$\overline{CS}$ faller til $\overline{WR}$ faller	t_3	0	ns
$\overline{WR}$ lav	t_4	20	ns
$\overline{CS}$ går høy til $\overline{WR}$ går høy	t_5	0	ns
Stabiliser inngangsdata	t_6	10	ns
Inngangsdata holdetid	t_7	1	ns

Tabell 4.12: Definerte tider tilhørende figur 4.25 [21].

Vi ser nå fra tidsdiagram i figur 4.25 og tabell 4.12 at vi må forholde oss til en rekke tidskrav som må overholdes for å programmere kretsen. Dette må tas hensyn til når det senere skal skrive programvare som skal kommunisere med kretsen.

Vi har nå sett på tidskravene til kretsen når den skal programmeres og vi forstetter med å se på hvilke krav som gjelder når dekodede verdier skal leses fra ADC.



Figur 4.26: Tidsdiagram for leseoperasjon [21].

Hvor de definerte tidene er gitt i tabell 4.13

Parameter	Symbol	Min	Typ	Maks	Enhet
$CONVST$ går høy til $\overline{EOC}$	t_{CON}	0		3	μs
Måletid	t_{ACQ}	1		1000	μs
$\overline{CS}$ går høy til $CONVST$ går høy	t_Q	500			ns
$CONVST$ går høy til $\overline{EOC}$ går høy	t_0		47	140	ns
$\overline{EOC}$ går lav til $CONVST$ går lav	t_1	0			ns
$CONVST$ lav	t_7	1			ns
$\overline{CS}$ går lav til $\overline{RD}$ går lav	t_8	0			ns
$\overline{RD}$ lav	t_9	30			ns
$\overline{RD}$ går høy til $\overline{CS}$ går høy	t_{10}	0			ns
$\overline{RD}$ høy	t_{11}	10			ns
$\overline{RD}$ faller til gyldig data	t_{12}			35	ns
$\overline{RD}$ går høy, data holde tid	t_{13}	5			ns

Tabell 4.13: Definerte tider tilhørende figur 4.26 [21].

Vi ser nå fra tidsdiagrammet i figur 4.26 og tabell 4.13 at vi må forholde oss til en rekke tidskrav som må overholdes for å lese data korrekt fra kretsen. Også disse kravene må tas hensyn til når det senere skal skrives programvare som skal kommunisere med kretsen.

4.1.4 Digital til analogomformer

I kapittel 4.1.2 kom vi frem til at det er behov for DAC for å oppnå ønsket signalbehandling. Vi skal i dette delkapittelet forsøke å komme frem til en DAC som oppfyller kravene.

Krav til DAC

Vi ønsker en DAC som er i stand til å levere et signal med signalamplitude tilsvarende signal fra sensor. Vi ønsker videre at det digitale grensesnittet mellom DAC og mikrokontroller baseres på 3.3V. Det blir da tilsvarende utfordringer som ved valg av ADC og det kreves også her en krets som støtter ulike strømforsyninger.

Referansesignalet som det ønskes at DAC'en skal produsere har ingen direkte krav til oppløsning, men den må allikevel være i stand til å reprodusere 90% av sensorsignalets amplitude. Dette medfører at krav til oppløsning anses som moderat og begrenses til DAC med 10 bit oppløsning.

Som for ADC er det ønskelig med en DAC som har felles intern referanse for samtlige kanaler. Som nevnt vil da en eventuell referansefeil gjennspeiles likt over samtlige akser og det er da mulighet til å eliminere denne ved kalibrering. Dette medfører at vi søker etter en DAC med minst 3 separate kanaler.

Det siste kravet som må oppfylles er at kretsen må kunne monteres på kretskort uten spesialverktøy eller maskiner.

De fastsatte krav oppsummeres i følgende punkter:

- $V_{Ref} = 5.0V$ og $V_{IO} = 3.3V$
- 10 bit oppløsning
- 4 kanaler med intern referanse
- Kretsen må kunne monteres uten hjelp av avansert utstyr.

Ved søk etter egnede kretser finner jeg to aktuelle som oppfyller gitte krav. Disse skal vi nå se nærmere på.

Fra tabell 4.14 finner vi listet to kretser som oppfyller gitte krav. Ved å studere tilhørende datablad finner at eneste forskjell mellom disse to er grensesnittet mellom DAC og mikrokontroller. Det er gitt av tilhørende datablader at MAX5714 benytter SPI og MAX5814 benytter

[1] 03.04.16 - www.digikey.com

Fabrikat	Modell	Pakke	Pris
Maxim Integrated	MAX5714AUD+	14-TSSOP	2.85$ [1]
Maxim Integrated	MAX5814AUD+	14-TSSOP	2.85$ [1]

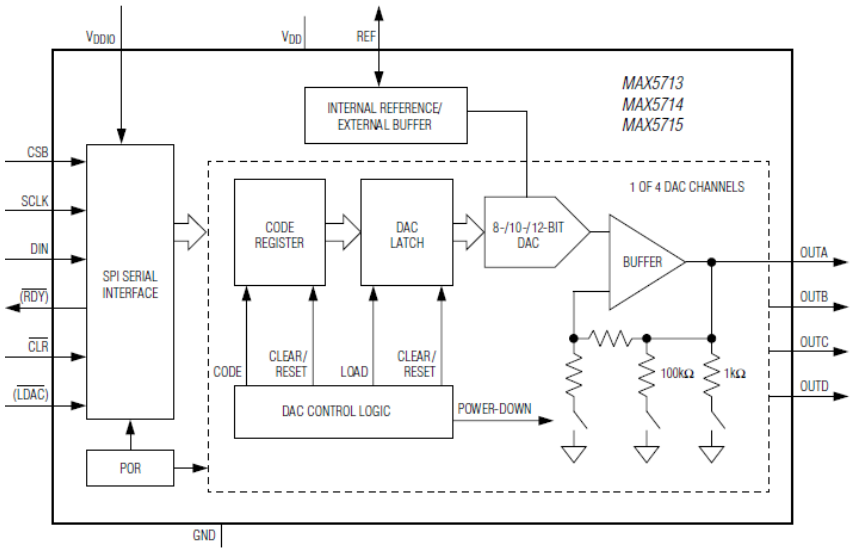
Tabell 4.14: Aktuelle digital til analogomformere.

I^{2}C som kommunikasjons grensesnitt mellom krets og mikrokontroller. Dette er begge veletablerte standarder som anses kjent for leser. Vi vil dermed ikke studere vikemåte til disse men kun konkludere med at det er ønskelig å benytte kretsen med SPI grensesnitt.

Digital til Analogomformer - MAX5714

Vi skal nå de på virkemåten til den valgte digital til analogomformeren MAX5714 [17] hvor de funksjoner som ønskes benyttet i denne oppgaven vil bli vektlagt.

Det er fra tilhørende datablad [17] gitt at kretsen MAX5714 er bestykket med 4 uavhengige digital til analogomformere som kommuniserer via felles digitalt grensesnitt. Som nevnt baseres det digitale grensesnittet på SPI kommunikasjon med en øvre hastighet på 50MHz.



Figur 4.27: MAX5714 [17].

Fra figur 4.27 ser vi funksjonsdiagrammet tilhørende MAX5714. Vi skal nå se på de to ulike grensesnittene som denne kretsen er bestykket med.

Analogt grensesnitt - Det analoge grensesnittet til hver av kanalene vil produsere en buffret analog spenning gitt av ligning 4.26

$$V_{out} = V_{Ref} \cdot \frac{Kode}{2^{10}} \tag{4.26}$$

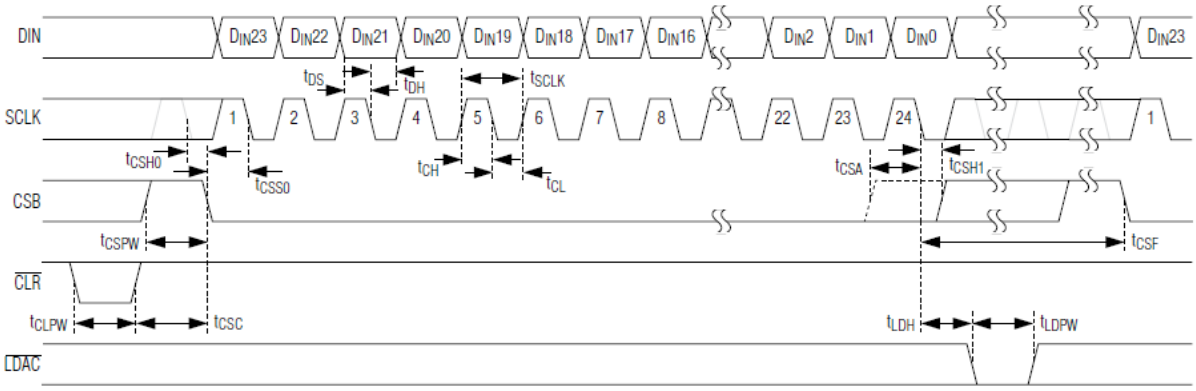
Hvor variabelen *Kode* representerer bitmønster gitt fra mikrokontroller. Foruten ekstern referanse har kretsen tilgjengelig 3 interne referansespenninger V_{Ref}. Ved hjelp av en definert SPI kommando kan ønsket V_{Ref} bestemmes fra programvaren. Det er videre gitt av datablad at hver av utgangene er buffret. Dette bufferet kan drive en resistiv last på minimum $2k\Omega$ parallellt med en kapasitans på $500pF$. Høyeste utgangsspenning bestemmes av kretsens analoge forsyning.

Digitalt grensesnitt - Det digitale SPI grensesnittet mellom mikrokontroller og DAC består av flere digitale inn og utganger som må settes korrekt for at begge kretsene skal være i stand til å dekode sendt informasjon. Vi starter med å se på de digitale inn og utganger og hvilken funksjon de ulike har.

Signal	Retning	Funksjon
$\overline{RDY}$	Ut	Benyttes for å sammenkoble flere kretser, ikke benyttet.
DIN	Inn	SPI data.
SCL	Inn	SPI klokke.
CSB	Inn	SPI velg krets.
$\overline{CLR}$	Inn	Sletter data og kontrollregistere.
$\overline{LDAC}$	Inn	Setter registere transperente, ikke benyttet.

Tabell 4.15: Digitalt SPI grensesnitt.

Fra tabell 4.15 får vi den nødvendige oversikt vi trenger for å analysere kretsens tidsdiagram. Lengden til hver melding eller kommando som sendes over SPI bussen til kretsen er definert i datablad [17] til 24 bit. Før vi ser på de ulike kommandoene skal vi først se på hvordan kretsen krever at mikrokontrolleren kommuniserer og hvilke pulstider som må overholdes.



Figur 4.28: MAX5714 Tidsdiagram [17].

Hvor de definerte tidene er gitt i tabell 4.16

Parameter	Symbol	Min	Typ	Maks	Enhet
SCLK periode	t_{SCLK}	20			ns
SCLK puls bredde høy	t_{CH}	8			ns
SCLK puls bredde lav	t_{CL}	8			ns
CSB faller til SCLK faller (setup tid)	t_{CSS0}	8			ns
CSB faller til SCLK faller (holde tid)	t_{CSH0}	0			ns
CSB stiger til SCLK faller (holde tid)	t_{CSH1}	0			ns
CSB stiger til SCLK faller	t_{CSA}	12			ns
SCLK faller til CSB faller	t_{CSF}	100			ns
CSB puls bredde høy	t_{CSPW}	20			ns
DIN til SCLK faller (setup tid)	t_{DS}	5			ns
DIN til SCLK faller (holde tid)	t_{DH}	4.5			ns

Tabell 4.16: Definerte tider tilhørende figur 4.28.

Vi ser nå fra tidsdiagram i figur 4.28 og tabell 4.16 at vi må forholde oss til en rekke tidskrav som må overholdes for å kommunisere med kretsen. Dette må tas hensyn til når vi på et senere tidspunkt skal utvikle driver til kretsen. Vi vil da samtidig studere de aktulle SPI kommanodene som denne kretsen benytter.

4.1.5 Kommunikasjon

For at sensornodene skal være i stand til å kommunisere med masternoden er det viktig at det velges en felles kommunikasjonsbus som er mest mulig egnet for problemstillingen. Vi vil nå forsøke å komme frem til den best egnede kommunikasjonsbus. Her vil det være hensiktsmessig å bruke den mest robuste. For å finne den mest robuste kommunikasjonsmetode er det naturlig å definere noen krav til egenskaper som må oppfylles for å velge metode.

Kravene til egenskaper er:

1. Høyest mulig grad av støyimunitet.
2. Fleksibel med tanke på fremtidig utvidelse av noder.
3. Kommunikasjonene må kunne anses som *realtime*, det vil si kommunikasjon uten forsinkende ledd.
4. Høy båndbredde.
5. Feilkorreksjon.
6. Mulighet for å gi ulike meldinger ulik prioritet.
7. Bidirektional bus der alle noder kan kommunisere.

Nå som flere krav er definert, skal vi i tabellen nedenfor sammenlikne kravene med egenskapene til de mest etablerte kommunikasjonsstandarder som benyttes i dag.

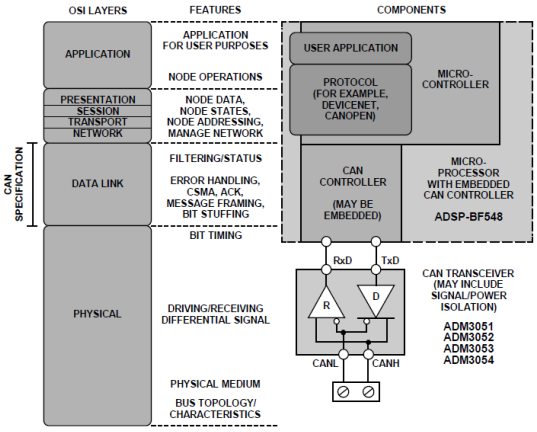
Standard	Støyimun	Fleksibel	*Realtime*	Høy båndbredde	Feilkorreksjon	Prioritet	Bidirektional
RS232	–	–	–	–	–	–	–
RS485	√	√	√	√	–	–	√ [2]
CAN	√	√	√	√	√	√	√

Tabell 4.17: Sammenlikning av kommunikasjonsstandarder.

Tabell 4.17 viser at *Controller Area Network* (CAN) er den standarden som innehar ønskede egenskaper og vi foretrekker dermed å benytte denne i oppgaven.

CAN-buss

Vi starter med å se litt på teorien som ligger bak CAN-buss og vil hovedsaklig vektlegge de deler som anses nyttige for problemstilingene i denne oppgaven. Det passer bra å begynne med å se på hvordan CAN er implementert ved hjelp av ulike lag, der hvert lag er tiltenkt definerte oppgaver.



Figur 4.29: CAN OSI lag [32].

I figur 4.29 finnes en oversikt over de ulike lag som CAN-buss er delt inn i og hvilke oppgaver og egenskaper de ulike lagene har. Vi starter med å se på det fysiske laget.

Det fysiske lag

Det fysiske laget tar seg av den elektriske signalomformingen som kreves for å kommunisere over et CAN-nettverk. Spesifikasjoner til dette laget er gitt av standard ISO-11892-2 og en grundigere beskrivelse kan finnes i denne.

[2] Krever 4 ledere

Figur 4.30: CAN signalnivåer [32].

Figur 4.30 er en illustrasjon som viser signalnivåene på en CAN-buss. Her ser vi det er definert to tilstander som CAN-kontroller(e) benytter når de skal sende og motta data. Det er valgt å la tilstand 1 være recessiv og tilstand 0 være den dominante tilstanden. Med dette oppnås det at meldinger med lavest id vil dominere over meldingen med den høyeste id. CAN-kontrollerene vil da være i stand til å prioritere meldinger på bitnivå ved å sjekke om deres bit er aktivt på nettverket.

Vi skal nå se litt på hvilke krav det typisk er gitt til signalnivåene i et CAN-nettverk. Tabell 4.18 viser typiske krav til signalnivåer som en CAN-kontroller må overholde for å kommunisere på et CAN-nettverk.

Nivå	CAN Tilstand	CAN Nivå
1	Recessiv	CAN_H - $CAN_L \leq 0.5$
0	Dominant	CAN_H - $CAN_L \geq 0.9$

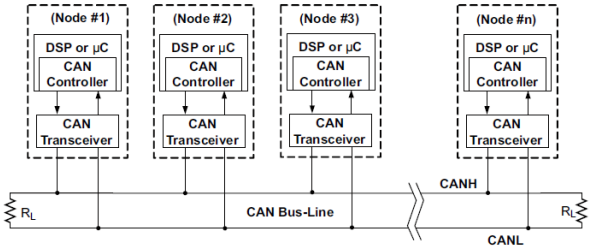
Tabell 4.18: Signalnivåer CAN-buss [32].

I standarden ISO-11892-2 er det gitt at det bør benyttes tvinnede kabler med nominell impedans på 120Ω. Det er videre spesifisert at EMC egenskaper forbedres ved å benytte skjermet kabel. Kvaliteten på signalkabelen samt kabellengde, vil direkte påvirke signalkvaliteten og dermed hvor høy bitrate som kan benyttes. Fra [10] er det gitt en tabell som viser maksimal bitrate ved ulike kabellengder. De 3 første radene i denne tabellen blir gjenngitt nedenfor.

Kabel lengde [m]	Bitrate [Mbps]
40	1.0
200	0.5
400	0.25

Tabell 4.19: Maksimal bitrate som funksjon av kabellengde [10].

Tabell 4.19 viser at dersom kabellengden begrenses til under 40 meter, kan den bitrate som ønskes benyttes. Siden det kan antas at dette systemet er tiltenkt å operere i støyete omgivelser kan et moderat valg av bitrate være fornuftig. På tross av dette velges bitrate til 1Mbit/s. Så høy bitrate vil naturligvis sette større krav til den fysiske delen av bus, men vil også tillate et høyere antall meldinger per tid.



Figur 4.31: Typisk CAN-nettverk [10].

Figur 4.31 illustrerer et typisk CAN-nettverk der flere noder er koblet sammen. I et slikt nettverk kan det lett oppstå uønsket signalrefleksjon. Selv om refleksjon er et problem som spesielt inntreffer med lengre signallinjer, anses det allikevel viktig å ta hensyn til dette i denne oppgaven.

For å opprettholde en pålitelig kommunikasjon i nettverket er det viktig at signalrefleksjoner blir holdt til på et minimalt nivå. Dette oppnås ved å terminere signalkablene med en termineringsresistor, R_L, i hver ende. Vi velger R_L slik at den har lik impedans som signallinjene, det vil si $R_L = 120\Omega$.

I et større system der det opereres med flere jordkilder, vil disse kildene forsøke å danne et felles nivå mellom dem. Når en node blir påviket av støy, vil det påvirke spenningsnivåene til denne. Dersom jordnivået til flere noder er sammenkoblet, vil denne støyen fra en node også kunne påvirke de andre nodene ved at de forsøker å utligne forskjellen i det felles jordnivået. Det introduseres da et uønsket fenomen som kalles jordstrømmer. For å unngå jordstrømmer, er det ønskelig at det fysiske laget til CAN kontrollerene er galvanisk isolert. Ved å galvanisk isolere kontrollerene elimineres fullstendig de problemer som felles jord kan medføre.

Meldingstruktur for CAN

For å kommunisere over CAN-buss, er krav til hvordan kommunikasjonen skal foregå definert i standarder. Disse standarder utvikles og vedlikehodes av Bosch GmbH. I dag anses CAN-standarder CAN 2.0A og CAN 2.0B som de mest utbredte og fleksible. Den største forskellen mellom CAN 2.0A og CAN 2.0B er hvordan standarden definerer bredden til *identifier*. CAN 2.0A støtter en *identifier* med 11 bit bredde og denne omtales som *standard identifier*. CAN 2.0B støtter en utvidet *identifier* som kan bestå av 29 bit og omtales som *extended identifier*. Moderne CAN-kontrollere vil som oftest støtte begge standarder og bredden til *identifier* velges av den som designer systemet. I denne oppgaven anses det som tilstrekkelig å benytte *standard identifier*, det vil si 11 bit.

Tabell 4.20 viser en oversikt over hvordan en standard CAN melding er strukturert. Denne oversikten viser at meldingen består av flere felt enn kun informasjonen vi ønsker å sende. Det fremkommer også av tabellen at kun noen av feltene som påvirkes direkte ved å velge blant annet melding identifikasjon samt data som ønskes sendt. De resterende felt muligjør feilsjekk

Standard CAN Melding										
Start	Identifier 11bit	RTR	IDE	Reserved	DLC	Data 0-8bytes	CRC	CRC Delim	ACK	ACK Delim	EOF
	Bestemmes				Bestemmes					

Tabell 4.20: Standard CAN-melding [32].

av mottatt melding, samt et bestemt start og avslutningsmønster. Disse feltene er merket grå og blir automatisk generert av CAN kontrolleren. I tabell 4.21 forklares de ulike feltene og vi ser også på bredden til de ulike feltene.

Felt	Antall Bit	Beskrivelse
Start	1	Initierer kommunikasjon
Identifier	11	En unik meldingsidentifikator som også representerer meldingens prioritet
Remote Transmission Request (RTR)	1	0 for data melding og 1 for *remote request*
IDE	1	Definerer standard (0) eller extended format(1)
Reserved	1	Reservert til fremtidig bruk.
DLC	4	Antall data byte i melding
Data	0-64	Data som skal sendes
CRC	15	*Cyclic Redundancy Check*
CRC delimiter	1	Skal være 1
ACK	1	Skal være 1
ACK delimiter	1	Skal være 1
EOF	7	Slutt på CAN-melding. Skal være 1

Tabell 4.21: Beskrivelse av felt i en CAN melding .

Vi har nå sett på hvordan CAN-buss kommuniserer over et multinode nettverk og hvilke krav det stilles til både nettverket og meldingenes struktur. Det eksisterer et stort utvalg av CAN-kontrollere som kunne passe i denne oppgaven, men jeg ønsker allikevel at denne kontrolleren er integrert i selve mikrokontrolleren. Dette blir en begrensende faktor når mikrokontroller skal velges, men jeg anser fordelene med en integrert CAN-kontroller viktigere enn stort utvalg mikrokontrollere. Vi skal avslutte delkapittelet med å sammenlikne ulike CAN-buss drivere, og bestemme oss for den som passer best til denne oppgaven.

Valg av CAN-buss driver

I følgende kapittel skal det velges den CAN-buss driveren som er best egnet i denne oppgaven. For å kunne bestemme oss for dette må vi først definere noen krav til driveren. Kravene som synes fornuftige er:

1. Ønsker forsyning $V_{DD} = 3.3$.
2. Galvanisk isolert fra nettverket.
3. Minimal bitrate 250kbps.
4. Monteres uten krav til spesialverktøy.

Ved søk etter kretser som oppfyller samtlige krav finner jeg to aktuelle CAN-drivere som vi nå skal se nærmere på. Disse to listes i tabell 4.22.

Produsent	Modell	Pakke	Hastighet	Pris
Texas Instruments	ISO1050	SOP-8, SOIC-16	1Mbit/s	43kr [3]
NXP	TJA1052i	SOT162-1	2Mbit/s	51kr [3]

Tabell 4.22: Egnede CAN-drivere.

Tabell 4.22 viser at komponentene er priset omtrent likt. Vi vil derfor se bort fra prisdifferansen videre i prosessen. Vi kan også legge merke til at de begge oppfyller kravet til hastighet men vi ser at kretsen TJA1052 [24] har mulighet for den dobbelte hastighet i forhold til kretsen ISO1050. Vedrørende kravet til kretsens fysiske utforming og mulighet for montering uten spesialvektøy, tilfredsstiller begge dette kravet. For å komme frem til den som passer best i denne oppgaven, blir vi nødt til å sammenlikne flere av de viktigste egenskapene til komponentene.

	Digital					CAN BUS							
	V_{DD}		I_{DD}(mA)			V_{DD}		$I_{DD,dom}$(mA)		$I_{DD,rec}$ (mA)			
Modell	Min	Maks	Min	Typ	Maks	Min	Maks	Typ	Maks	Typ	Maks	V_{CM}	V_{Maks}
ISO1050	3.3	5.25		1.8	2.8	4.75	5.25	52	73	8	12	±12	-27,+40
TJA1052i	3.0	5.25	2.6		5.6	4.75	5.25		70		10	±25	±58

Tabell 4.23: Sammenlikning av CAN-drivere.

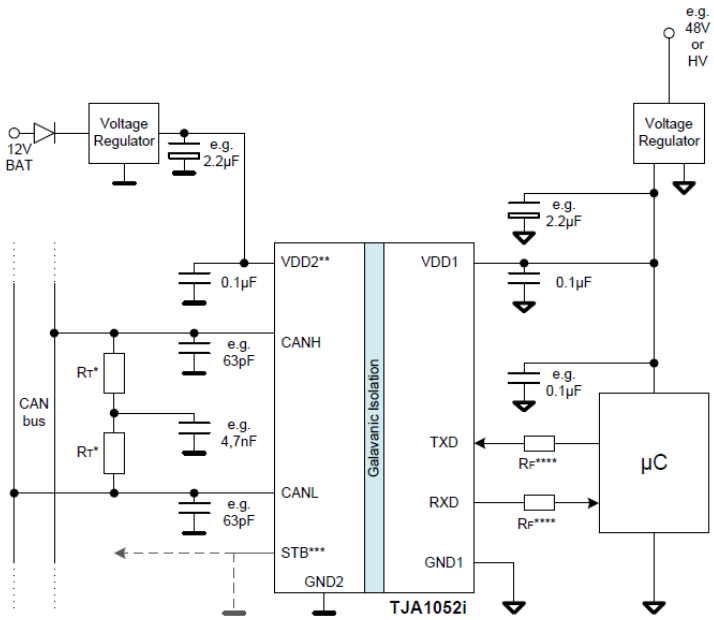
Tabell 4.23 viser at det er marginale forskjeller mellom de ulike kretsene. Videre ser vi at høyeste terminalspenning,V_{Maks}, som kretsene kan utsettes for uten å ta varig skade er ulik for de to kretsene. Det synes da fornuftig å anta at kretsen TJA1052i er designet mer robust på bus siden av kretsen. Da kan det forventes at kretsen i større grad er i stand til å håndtere eventuelle transienter som måtte forekomme på nettverket. Vi anser denne egenskapen for så viktig at den her vil være utslagsgivende for at vi velger kretsen TJA1052i som CAN-driver.

Vi skal nå se på hvordan kretsen TJA1052i kan benyttes og vi vil som før fokusere på de egenskapene som anses nyttige for oppgaven. Jeg vil basere dette på tilhørende datablad [24] samt teknisk rapport [23].

Figur 4.32 viser en anbefalt oppkobling av kretsen. Som figuren viser er selve kretsen galvansik skilt mellom inn og utgang og kretsen må dermed forsynes med 2 separate strømforsyninger uten felles jord. Med dette designet oppnås det å skille spenningspotensialene til CAN-buss fra

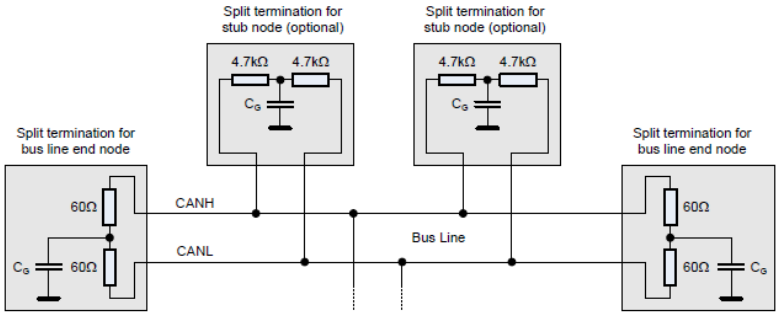
[3] 17.03.16 - www.farnell.com

resten av kretskortene og vi unngår dermed uønskede jordstrømmer. Selve dimensjoneringen av strømforsyningene blir nærmere omtalt i delkapittel 4.1.7.



Figur 4.32: TJA1052 typisk oppkobling [23].

Videre kan vi legge merke til at det ikke benyttes *common mode* spoler i figuren ovenfor. I [23] anbefales det å benytte et alternativt inngangstrinn i stedenfor spoler. Basert på eksperimenter er det kommet frem til bussterminering uten spoler, som i figur 4.33 forbedrer EMC egenskapene i forhold til ordinær bussterminering, som vises i figur 4.31.



Figur 4.33: Typisk *Split Termination* [23].

I denne oppgaven ønsker jeg å basere designet av CAN-driveren på applikasjonsnoten [23] og velger dermed å designe bussterminеringen som anbefalt.

4.1.6 Mikrokontroller

Når vi i dette delkapittelet skal bestemme oss for hvilken mikrokontroller som er best egnet for oppgaven, må vi starte med å definere noen krav og ønsker til egenskaper som denne bør

inneha. Jeg starter med å lage en oversikt over hvilke inn/utganger det er behov for til de ulike funksjonene. Til slutt summeres disse for å få oversikt over det minimale antall pinner en kontroller kan ha. Denne oversikten finnes i tabell 4.24 og viser at vi har behov for minst 35 GPIO pinner, hvor 2 er analoge innganger.

Grensesnitt	Funksjon	Antall pinner
DIGITAL	Synkroniser Klokke	1
	Ekstern ADC Data	16
	Ekstern ADC Kontroll	6
	CAN-buss	2
	SPI	5
	HMC2003 Set/Reset	2
	Pulstransformering	1
ADC	$V_{Batteri}$	1
	$V_{HMC2003,Set/reset}$	1
Totalt antall pinner		35

Tabell 4.24: Krav til tilgjengelige pinner.

Vi fortsetter med å definere flere sentrale krav til mikrokontrolleren. Vi starter med å påpeke at det er begrensede krav til hastighet. Allikevel er det ønskelig å basere valget på STM32 familien, da denne anses kjent fra tidligere prosjekter. Dette er en velutrustet mikrokontrollerfamilie der det finnes alt fra svært enkle til høyst komplekse mikrokontrollere. Felles for familien er at de alle er basert på kjent 32bit ARM teknologi og benytter samme utviklingsverktøy. Når det søkes etter egnet mikrokontroller blant utvalget i STM32 familien, vil faktoren lav effekt sammen med $V_{CC} = 3.3V$ være viktige parametere.

Fra kapittel 4.1.5 ble det besluttet å benytte CAN-buss til kommunikasjon mellom nodene. Det er dermed en stor fordel å finne en mikrokontroller med CAN-busskontroller integrert i kretsen. Ved å finne en krets med denne integrert, forenkles det endelige kretskort utlegg såvel som programvaren som skal skrives. Ved valg av mikrokontroller blir dette et viktig punkt som må tilfredsstilles.

Foruten selve sensormålingene er det ønskelig å måle to ulike sentrale analoge spenninger på kretskortet. Disse to er batterispening $V_{Batteri}$ samt $V_{HMC2003,Set/reset}$. Til dette formålet anses mikrokontrollerens integrerte analog til digitalomformer til å inneha tilstrkkelig oppløsning. Vi må dermed velge en mikrokontroller med denne funksjonen integrert.

Det er i [25] gitt at vi har behov for flere sensorer med en større fysisk avstand mellom sensorene enn hva som synes praktisk mulig på et kretskort. For å være sikker på at de ulike sensor nodene måler magnetfeltet til samme tid, er vi da nødt til å synkronisere de ulike interne klokkene. For denne synkronisering anses en løsning der de ulike sensornodene mottar en sykroniseringspuls som vi i programvaren kan benytte til å korrigere avvik i den interne klokken. Det blir da behov

for det som kalles *extern pin interrupt*. Denne funksjonen gjør mikrokontrolleren i stand til å generere et programavbrudd ved bestemt flanke. Ved valg av mikrokontroller blir også dette et viktig punkt som må tilfredsstilles.

Kravene til mikrokontrollerens egenskaper oppsummeres i følgende punkter:

1. Lavest mulig effekt
2. Ønsker forsyning, $V_{CC} = 3.3V$
3. Integrert CAN-buss kontroller
4. Integrert Analog til digitalomformer
5. Integrert SPI kontroller
6. Ønsker å velge fra STM32 serien
7. Minimum 35 GPIO pinner
8. Ekstern pinne avbrudd
9. Må kunne monteres uten krav til spesialverktøy

Ved søk etter mikrokontrollere som tilfredsstiller aktuelle krav, finner jeg følgende:

Modell	Kjerne	Pakke	CAN	ADC	Ekstern avbrudd	SPI	I/O	Pris
STM32F072CBT6	ARM Cortex-M0	LQFP-48	√	√	√	√	37	39.08kr [4]
STM32F091CBT6	ARM Cortex-M0	LQFP-48	√	√	√	√	37	27.46kr [4]
STM32F098CCT6	ARM Cortex-M0	LQFP-48	√	√	√	√	37	32.13kr [4]
STM32F103CBT6	ARM Cortex-M3	LQFP-48	√	√	√	√	37	57.75kr [4]

Tabell 4.25: Mikrokontrollere som tilfredsstiller fastsatte krav.

Som vist i tabell **??** finnes 4 mikrokontrollere som synes egnet for denne oppgaven. Vi vil nå sammenlikne flere parametere før vi velger den mikrokontrolleren som vi ønsker å benytte.

			Perefermoduler	
Modell	FLASH	RAM	Aktivert	Deaktivert
STM32F072CBT6[5]	128kB	16kB	24.1mA	13.5mA
STM32F091CBT6[5]	128kB	32kB	26.7mA	16.4mA
STM32F098CCT6[5]	256kB	32kB	26.1mA	16.8mA
STM32F103CBT6[6]	128kB	20kB	36mA	27mA

Tabell 4.26: Sammenlikning av egenskaper.

[4] 19.03.16 - www.farnell.com

Fra sammenlikning i tabell 4.26 finner vi marginale forskjeller mellom de ulike mikrokontrollerene. Siden dette skal forsynes fra batteri, er det som kjent ønskelig å benytte komponenter som forbruker minst mulig effekt. Vi velger dermed den mikrokontrolleren med lavest strømforbruk, det vil si STM32F072CBT6. Denne mikrokontrolleren oppfyller alle krav og den synes godt egnet i denne oppgaven.

4.1.7 Strømforsyning

Vi skal nå se på hvordan vi kan designe en strømforsyning som er i stand til å levere ønsket spenning og strøm til komponentene på kretskortet. Som tidligere delkapitler har vist, kreves det totalt to stømforsyninger til dette kretskortet. Forskjellen mellom disse to er at vi har behov for å forsyne den digitale delen av kretskortet med 3.3V mens den analoge delen krever 5.0V.

I designet av denne strømforsyningen forutsettes det at hver sensornode forsynes med 3-celler LiPo batteri med spenning 11.4V. Dette vil da gi en uavhengig støyfri kilde til hver sensornode.

Krav til kapasitet

Før vi kan søke etter egnet regulator, må vi sette opp en tabell som finner krav til strømstyrke til hver av forsyningene.

	Strømtrekk				
	Per stk			Total	
Modul	3.3V	5.0V	Antall	3.3V	5.0V
STM32F072	24mA		1	24mA	
TJA1052	5.6mA		1	5.6mA	
MAX11047	5.5mA	30mA	1	5.5mA	30mA
MAX7480		2.9mA	3		8.7mA
AD8226		350μA	3		1mA
MAX5714	1μA	1.5mA	1	1μA	1.5mA
			Sum	35.1mA	41.2mA

Tabell 4.27: Krav til strømforsyning

Fra tabell 4.27 kommer det frem at kravet til den minimale strømstyrken som en egnet strømforsyning antas å måtte levere, er $I_{Min,3.3V} = 36mA$ og $I_{Min,5.0V} = 42mA$. Når regulator skal velges bør det legges inn en feilmargin slik at valgt regulator gjerne har en kapasitet minst 2 ganger høyere enn hva vi kom frem til. Siden vi her ser på regulatorer som skal inngå i et kretskort der det måles svært lave spenninger er det sterkt ønskelig å benytte regulatorer som gir en støyfri kilde. Det er derfor naturlig å vurdere lineærregulatorer som i motsetning til svitsjet ikke genererer støy

[5]Ekstern klokke, $f_{HCLK} = 48MHz$
[6]Ekstern klokke, $f_{HCLK} = 72MHz$

av noen slag. Bakdelen ved lineærregulatorer er at de regulerer spennigen ved å selv absorbere overflødig effekt. Ved stor differanse mellom V_{in} og V_{ut} kan effektabsorbsjonen medføre uønsket høy temperatur og virkningsgraden,η, reduseres. Vi skal nå se på hvordan forventet effekttap og virkningsgraden kan beregnes for lineærregulator.

$$P = V \cdot I \tag{4.27}$$

$$P_{Tap} = (V_{Inn} - V_{Ut}) \cdot I \tag{4.28}$$

Vi modifiserer nå den generelle effekt ligningen 4.27 til ligning 4.28 som kan benyttes for å beregne tap i en lineærregulator, strøm gjennom regulator samt reguleringsspenning over komponenten. Vi starter med å beregne tapet og virkningsgraden som kan forventes til 3.3V lineærregulatoren.

$$\begin{aligned} P_{Tap,3.3V} &= (11.4V - 3.3V) \cdot 35.1mA \\ P_{Tap,3.3V} &= 0.284W \end{aligned} \tag{4.29}$$

Deretter beregnes effekten som benyttes ved hjelp av ligning 4.27.

$$\begin{aligned} P_{3.3V} &= 3.3V \cdot 35.1mA \\ P_{3.3V} &= 0.116W \end{aligned} \tag{4.30}$$

Virkningsgraden,η, beregnes ved:

$$\begin{aligned} \eta_{3.3V} &= \frac{P_{3.3V}}{P_{3.3V} + P_{Tap,3.3V}} \\ \eta_{3.3V} &= 0.289 \end{aligned} \tag{4.31}$$

Som ligning 4.31 viser, er det en virkningsgrad på 29%, noe som anses som svært lavt. Vi skal nå beregne det samme for 5V forsyningen.

$$\begin{aligned} P_{Tap,5.0V} &= (11.4V - 5.0V) \cdot 41.2mA \\ P_{Tap,5.0V} &= 0.26W \end{aligned} \tag{4.32}$$

Deretter beregnes effekten som benyttes ved hjelp av ligning 4.27.

$$\begin{aligned} P_{5.0V} &= 5.0V \cdot 41.2mA \\ P_{5.0V} &= 0.206W \end{aligned} \tag{4.33}$$

Virkningsgraden,η, beregnes ved :

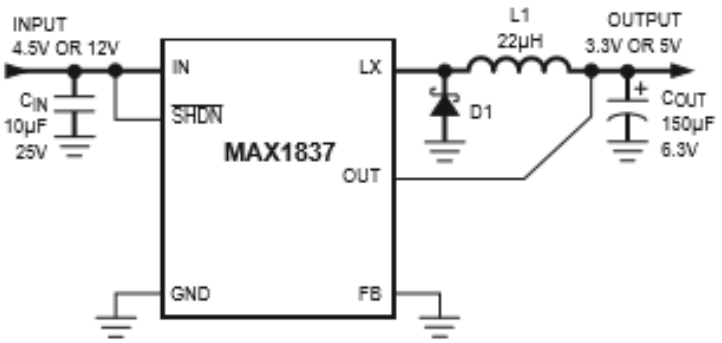
$$\eta_{5.0V} = \frac{P_{5.0V}}{P_{5.0V} + P_{Tap,5.0V}}$$
$$\eta_{5.0V} = 0.44 \tag{4.34}$$

Som vi ser av ligning 4.31 og ligning 4.34 oppnås en uakseptabel virkningsgrad ved å benytte lineærregulator. Som følge av den dårlige virknigsgraden må det nå sjekkes hva vi kan oppnå ved å erstatte den lineære med en svitsjet regulator.

Digital 3.3V-forsyning

Vi starter med å se på spenningen som forsyner den digitale delen av kretskortet, det vil si 3.3V. Av de to ulike spenningene er dette den minst følsomme for støy som en svitsjet regulator genererer.

Etter en vurdering av mulige alternativer velges Maxim Integrated MAX1837 [20]. Dette er en svitsjet regulator som krever minimalt med ekstra komponenter og blir dermed enkel å implementere samtidig som den krever liten plass på det endelige utlegget. Den leveres med fast utgangsspenning og vi velger naturligvis 3.3V modellen. Regulatoren er i stand til å forsyne 250mA med en virkningsrad $\eta_{3.3V} \approx 85\%$. Dette anses som tilfredsstillende i forhold til fastsatte krav.



Figur 4.34: MAX1837 [20].

Figur 4.34 viser typisk oppkobling av regulatoren MAX1837. Det er videre gitt anbefalte komponentverdier i simuleringsverktøyet EE-Sim fra Maxim Integrated [1].

Ref	Artikkelnummer	Produsent	Beskrivelse
U1	MAX1837ETT33-T	Maxim Integrated	Svitsja 3.3V regulator
C_{IN}	C4532X7R1C226M	TDK	Cap Ceramic 22μF 16V X7R 20 % SMD 1812 125C Plastic T/R
C_{OUT}	GRM32ER70J476K	Murata	Cap 47μF 6.3V
D1	EP05Q03L	Nihon	Diode Schottky 30V 785mA SOD-123
L1	7447798241 WE-PDF 1064	Wurth Elektronik	Ind 24μH 20% 4A

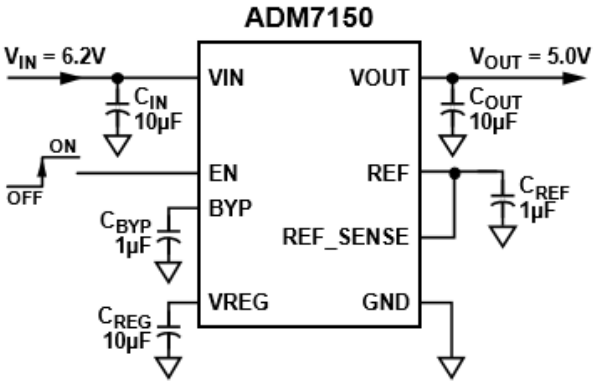
Tabell 4.28: Anbefalte komponenter.

Analog 5.0V-forsyning

Vi skal nå se på hvordan det kan dannes en stabil 5.0V analog spenningsforsyning. Som vi kom frem til ovenfor i ligning 4.34, er virkningsgraden til en lineærregulator uakseptabelt lav dersom $V_{in} = 11.4V$. Dersom inngangsspenningen derimot pre-reguleres,V_{in}, slik at den i størst mulig grad tilpasses den lineære regulatoren, finner vi svitsjregulatorens utgangsspenning til:

$$V_{ut,svitsj} = V_{in,lin} = V_{dropp} + V_{ut,lin} \tag{4.35}$$

For å kunne beregne $V_{ut,svitsj}$ i ligning 4.35, må det settes inn verdi tilhørende V_{dropp}. For å finne den manglende verdien må vi først bestemme oss for hvilken lineærregulator som skal benyttes.



Figur 4.35: ADM7150 [7].

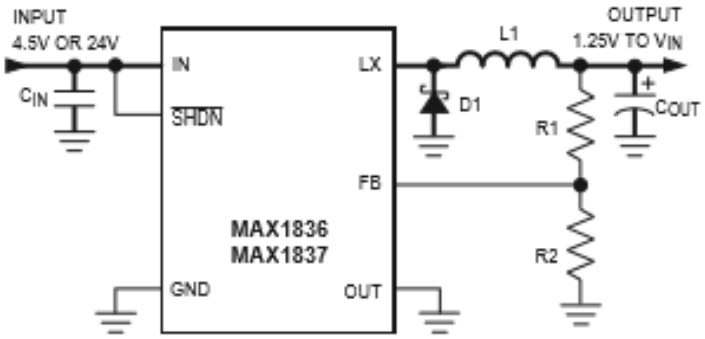
Etter en vurdering av mulige alternativer ble lineærregulatoren Analog Devices ADM7150ARDZ-5.0 [7] valgt. Fra datablad [7] blir det oppgitt at denne regulatoren kan levere inntil 800mA med en støy $1.0\mu V_{RMS}$. Dette anses svært bra sammenliknet med alternative regulatorer. Vi finner videre den maksimale reguleringsspenningen fra tilhørende datablad til å være $V_{dropp} = 0.5V$ ved 400mA. Setter dette inn i ligning 4.35 og bestemmer $V_{ut,svitsj}$.

$$V_{ut,svitsj} = 0.5V + 5.0V = 5.5V \tag{4.36}$$

Dersom den nye virkningsgraden nå beregnes, kommer vi frem til at denne er $\approx 90\%$. Dette anses

som en stor forbedring i forhold til den opprinnelige og vi ønsker å benytte denne løsningen.

Vi skal nå vise hvordan vi kan danne den regulerte spenningen $V_{ut,svitsj}$. Til denne jobben er det sterkt ønskelig å benytte samme regulator som benyttes for å regulere 3.3V spenningen. Når samme type regulator benyttes, oppnås hovedsaklig besparelser i arbeid med kretskort utlegg men også eventuell senere tuning av regulator. Vi skal nå se på en variant av kretsen MAX1837, med variabel utspenning.



Figur 4.36: MAX1837 med variabel utspenning [20].

Det er videre gitt anbefalte komponentverdier i simuleringsverktøyet EE-Sim fra Maxim Integrated [1].

Ref	Artikkelnummer	Produsent	Beskrivelse
U1	MAX1837ETT33-T	Maxim Integrated	Svitsja regulator
C_{IN}	C4532X7R1C226M	TDK	Cap Ceramic 22μF 16V X7R 20 % SMD 1812 125C Plastic T/R
C_{OUT}	C3216X5R1A335K	TDK	Cap 3.3μF 10V X5R 10% SMD 1206 85C
D1	EP05Q03L	Nihon	Diode Schottky 30V 785mA SOD-123
L1	7447798241 WE-PDF 1064	Wurth Elektronik	Ind 18μH 30% 980mA
R1			Resistor 34.8$k\Omega$
R2			Resistor 10.0$k\Omega$

Tabell 4.29: Anbefalte komponenter.

Kondensatorbank

I det endelige utlegget vil vi benytte produsentens anbefalte komponentverdier så langt som mulig. I tillegg til disse er det behov for ekstra kondensatorbanker som har til formål å stabilisere forsyningen slik at det unngås uønsket rippel. Disse bankene vil bli distribuert på kretskortet. Siden vi på det nåværende tidspunkt ikke er i stand til å avgjøre hvilke støykomponenter som skal filtreres, er det umulig å fastsette hvilken tidsrespons som ønskes til de ulike strømforsyningene. Vi blir derfor nødt til å gjøre en innledende gjetting og som i [14], baseres gjettingene på boka [33]. Den største tidsresponsen til forsyningene defineres innledningsvis som i [14], til å være

5ms. I det endelige utlegg ønsker vi også å benytte kondensatorer med ulike verdier. Fordelen med dette er at forsyningens båndbredde økes, noe som medfører høyere dempning av støy.

Vi skal nå se på hvordan vi kan beregne tidsresponsen til en strømforsyning. Til dette har vi behov for to elementære formler fra elektroteknikken. Vi starter med å definere forholdet mellom ladning, kapasitans og spennig.

$$Q = C \cdot V \tag{4.37}$$

Ved konstant strøm har vi også forholdet

$$Q = I \cdot T \tag{4.38}$$

Vi kombinerer nå ligningene 4.37 og 4.38.

$$C = \frac{I \cdot T_r}{V_{rippel}} \tag{4.39}$$

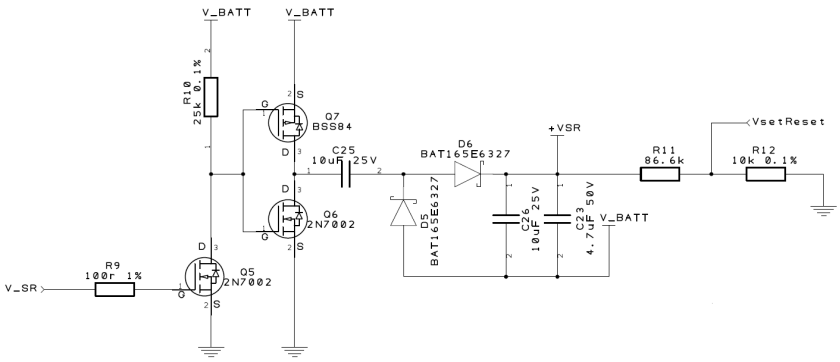
Den generelle ligningen 4.39 kan nå benyttes for å beregne kondensatorverdi, gitt strøm,tidsrespons og største akseptabel rippel. Ligningen er benyttet av simuleringsverktøyet EE-Sim [1] for å beregne nødvendige kondensatorbanker tilhørende de ulike regulatorene.

HMC2003 Set/Reset Puls Forsyning

Som forklart i delkapittel 4.1.1 har sensoren HMC2003 behov for en relativt høy strøm gjennom Set/Reset ledere for å orientere AMR filmen korrekt. Det er fra [13] oppgitt at disse ledere har en forventet resistans fra typisk 4.5Ω til 6Ω. For å oppnå største følsomhet kreves en $2\mu s$ strømpuls på over 4A gjennom disse. Den gitte informasjon settes inn i Ohms lov og finner nødvendig spenning:

$$V_{min} = 4.5\Omega \cdot 4A = 18V$$
$$V_{typ} = 4.5\Omega \cdot 5A = 22.5V$$
$$V_{maks} = 6\Omega \cdot 5A = 30V$$

For å orientere filmen korrekt viser beregningene ovenfor at det er behov for en spenning mellom 18V og 22.5V. Siden det ikke finnes noen spenningskilde med så høy spenning må den transformeres opp til påkrevd nivå. Det er i figur 4.6 vist et eksempel på hvordan dette kan løses. Denne løsningen anser jeg som ikke optimal med hensyn på plassutnyttelse og mulighet for å styre spenningen. Det skal derfor gjøres et forsøk på å designe denne kretsen med diskrete komponenter. Vi oppnår da at det blir større muligheter til å tilpasse kretsen til våres ønsker.



Figur 4.37: Transformere spenning.

Kretsen som vises i figur 4.37 genererer nå en spenning inntil ca 22.8V fra batterispenningen på 11.4V. Virkemåten til denne kretsen er at mikrokontrolleren pulser transistor Q5 som igjen styrer tilstanden til *push/pull*-trinnet Q6,Q7. Når dette pulses vil en strøm gå gjennom kondensator C25 og ved hjelp av D5 og D6 likerettes og overlagres pulsene forsyningsspenningen. Resistansene R11 og R12 danner en spenningsdeler som gjør det mulig å måle spenningsnivået til C23 og C26. Med denne målingen oppnås å kunne kontrollere nødvendig ladning, samtidig som motstanden kan beregnes etter en puls i Set/Reset lederene.

Vi skal nå beregne resistansene R11 R12 og starter med å bestemme verdien $R12 = 10k\Omega$.

$$V_{ADC} = \left[\frac{R_{12}}{R_{11} + R_{12}}\right] \cdot V_{Inn} \tag{4.40}$$

Før vi kan beregne R11 ved hjelp av ligning 4.40 må vi bestemme hva den maksimale spenningen over R6 skal være. Det er nå viktig å bestemme denne spenningen slik at den ved alle driftsforhold vil være lavere enn den maksimale innspenning som mikrokontrolleren tillater. Vi definerer dermed de maksimale spenningene til $V_{ADC} = 3.0V$ og $V_{Inn} = 25V$. Deretter beregnes R11:

$$\begin{aligned} V_{ADC} &= \left[\frac{10k\Omega}{10k\Omega + R_{11}}\right] \cdot V_{Inn} \\ 3.0V &= \left[\frac{10k\Omega}{10k\Omega + R_{11}}\right] \cdot 25V \\ R_{11} &= \left[\frac{10k\Omega}{3.0V}\right] \cdot 25V - 10k\Omega \\ R_{11} &= 86k\Omega \end{aligned}$$

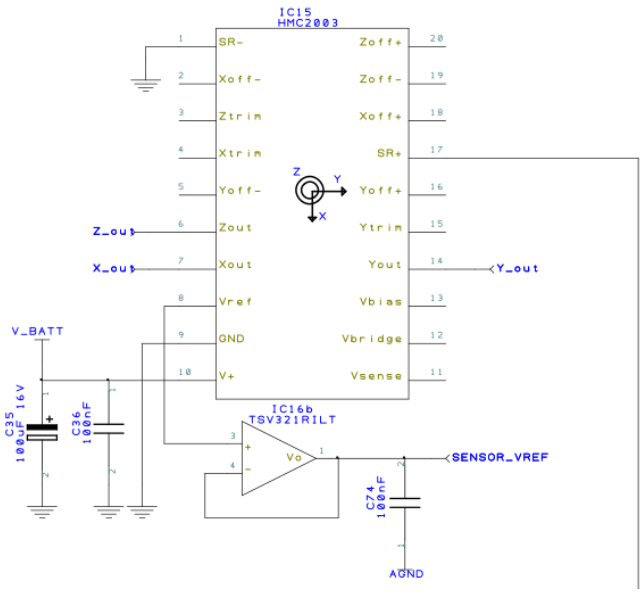
Verdien til C23 velges som anbefalt i [11], til minst $4.7\mu F$.

4.1.8 Skjema

Her skal de mest sentrale delene av skjemaet til sensornode presenteres og forklares. Fullstendig argumentasjon for valg av komponenter og løsninger finnes i kapittel 4.1.1 til 4.1.7. For en

fullstendig oversikt over skjema, henvises det til vedlegg D.1.

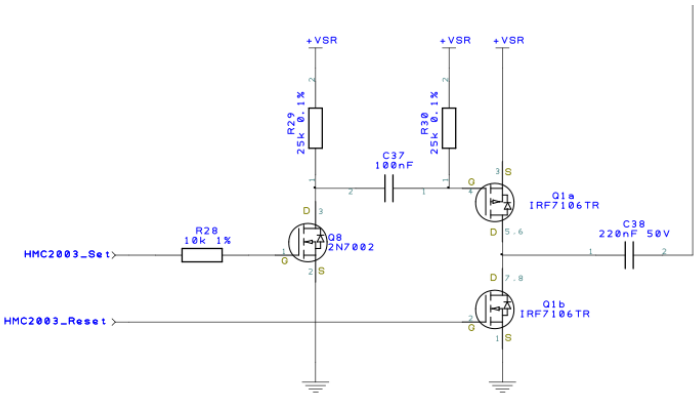
Magnetfeltsensor



Figur 4.38: Utsnitt fra kretsskjema, magnetfeltsensor HMC2003.

Øverst i figur 4.38 viser inntegnet magnetfeltsensor HMC2003. Denne måler magnetfeltet til de 3 aksene, hvor målingene gis som analoge signaler. For å begrense båndbredden til de målte signalene, er det implementert et 1.ordens lavpassfilter. Lavpassfilteret har en knekkfrekvens ved $1kHz$.

Videre har sensoren en integrert spenningsreferanse. I dette designet er det valgt å benytte denne referansen til å kalibrere forsterkningen tilhørende hver kanal. Det er i datablad [13] ikke oppgitt noe informasjon som sier noe om drivekapasiteten til referansesignalet. For å unngå at kretser påvirker signalet er det valgt å buffre dette med operasjonsforsterker IC16. Det oppnås da et lavohmig signal som kan distribueres på kretskortet og eventuell påtrykt støy vil ikke kunne påvirke selve referansen.

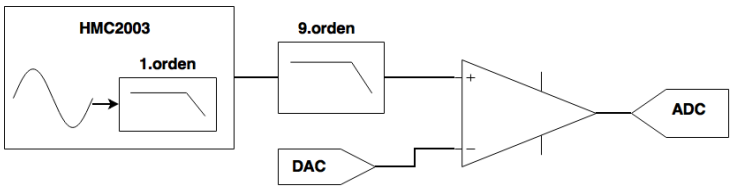


Figur 4.39: Utsnitt fra kretsskjema, set/reset av magnetfeltsensor HMC2003.

For å kunne utføre en fullstendig set/reset av sensor er det benyttet kretsløsning som anbefalt i [11]. Ved å sammenligne figur 4.39 med løsning i [11], kan det allikevel legges merke til at det er benyttet NMOS teknologi, i stedet for NPN, i dette designet.

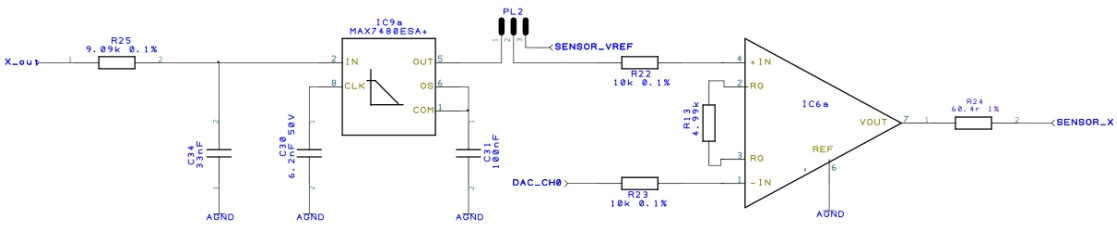
Analog signalbehandling

Den analoge signalbehandlingen består av flere logiske trinn. De ulike trinnene visualiseres ved hjelp av blokkskjema i figur 4.40. Sensoren HMC2003 er i figuren kun inkludert for å lettere se helheten. Denne komponenten blir derfor ikke nærmere forklart her.



Figur 4.40: Blokkskjema over analog signalbehandling.

De ulike trinnene skal nå forklares ved å referere til endelig skjema.



Figur 4.41: Utsnitt fra kretsskjema, analog signalbehandling.

Figur 4.41 viser hvordan den analoge signalbehandlingen er designet for x-aksen. Ved å studere skjema i vedlegg D.1, vil identiske kretsdesign gjennkjennes for y- og z-aksen.

Det målte magenetfeltet vil først filtreres av det interne 1.ordens lavpassfilter, med knekkfrekvens $1kHz$. Deretter blir signalet *prefiltrert* ved hjelp av RC leddet R25 og C34. Formålet med det 1. ordens *prefilteret*, er å dempe frekvenser som ville medført aliasing i det svitsja filteret MAX7480. Det vil si, frekvenser over $f_{osc}/2$. I dette designet er $f_{osc} = 8.55kHz$ og ved $f_{osc}/2$ er det valgt å ha en signaldempning på $-30dB$. Basert på denne informasjonen ble R25 og C34 beregnet slik at knekkfrekvensen til *prefilteret* er $530Hz$ og unngår dermed dempning i signalets passbånd.

Etter signalet er *prefiltrert*, blir det lavpassfiltrert ved hjelp av det 8.ordens svitsja filteret MAX7480 [18]. Valgt filter har mulighet til å tilpasse knekkfrekvensen, enten ved hjelp av et klokkesignal, eller ved å benytte en intern oscillator. I dette designet er det valgt sistnevnte og dette valget krever en kondensator som bestemmer oscillatorfrekvensen. Det ble argumentert for at knekkfrekvensen tilpasset signalet i dette målesystemet er $85.5Hz$. Valgt knekkfrekvens resulterte i en dempning på $\approx 92dB$ ved 300Hz, med en marginal dempning i passbånd.

For å kunne kalibrere forsterkningen til intrumenteringsforsterker IC6 er det valgt å måle referansesignalet fra magnetfeltsensoren HMC2003. Stiftlist PL2 som vises i figur 4.41 er implementert for at det skal være mulig å velge mellom magnetfeltsignal og referansesignal. I posisjon (1-2) velges signal fra sensor og i posisjon (2-3) velges referansesignal. Valgt signal sendes så til den positive terminalen på instrumenteringsforsterkeren IC6.

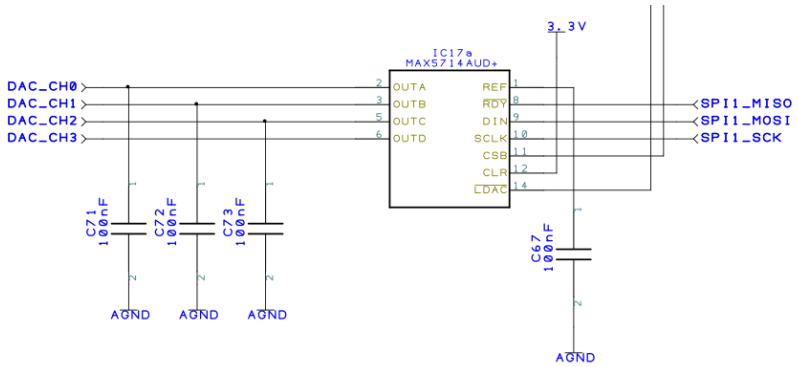
For å kunne benytte en ADC med 16 bit oppløsning og oppnå ønskelig sensitivitet, kreves det at signalet fra magnetfeltsensor forsterkes opp. Dette løses i designet ved å påtrykke den negative terminalen til instrumentforsterkeren med et signal som er 90% av forrige sampelverdi. Signalspenningen fra intrumenteringsforsterker blir da:

$$V_o(k) = G \cdot \big(V_+(k) - V_-(k)\big)$$
$$V_o(k) = G \cdot \big(V_+(k) - 0.9V_o(k-1)\big)$$

Hvor forsterkningen $G = 10.89$.

Dette gir en løsning som fjerner en kjent og vesentlig del av signalet, før detaljene i signalet forsterkes opp G ganger.

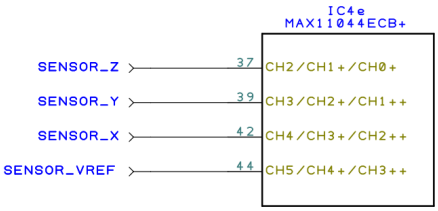
For å danne signalet V_- som inngår i ligningen ovenfor, kreves det imidlertid en pålitelig styrt spenningskilde. Det er her valgt å benytte en 4 kanals 10bits DAC for å danne dette signalet.



Figur 4.42: Utsnitt fra kretsskjema, DAC.

Figur 4.42 viser kretsen MAX5714 [17] inntegnet i skjema. Kretsen MAX5714 kommuniserer med mikrokontrolleren ved hjelp av SPI grensesnitt. Programvaren i mikrokontrolleren vil basert på den målte spenningen V_o sette ut en spenning som tilsvarer $0.9V_+$ for hver kanal. Med dette oppnås det å ha full kontroll over spenningen V_- til hver kanal og verdien til spenningen kan med høy presisjon inngå i ligningen som beregner faktisk signalspenning.

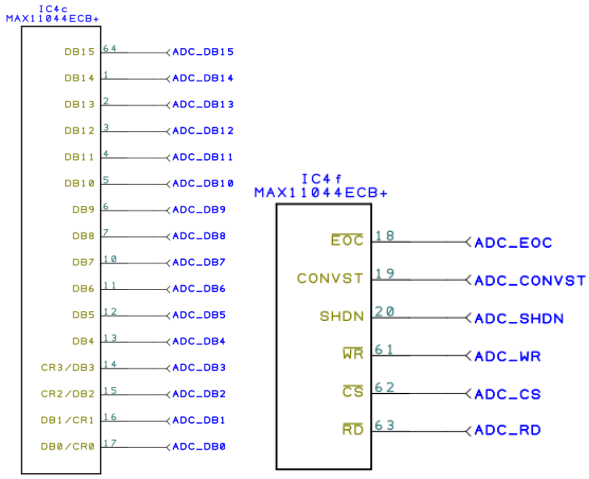
For å måle signalene med høy presisjon, er analog til digitalomformeren MAX1047 [21] valgt. Dette er en 16bit SAR analog til digitalomformer med 4 separate kanaler og $V_{ref} = 5.0V$. Videre støtter kretsen simultankonvertering av samtlige kanaler. Målinger av magnetfeltet vil dermed utføres samtidig på alle akser. Grensesnittet mellom mikrokontroller og kretsen MAX1047 baseres på kommunikasjon over parallelle datalinjer.



Figur 4.43: Utsnitt fra kretsskjema, ADC.

Figur 4.43 viser skjema der signalene kobles til de ulike kanalene i analog til digitalomformeren MAX1047 [21]. Her kan det legges merke til at signalene for z- og x-akse har byttet kanal i forhold til organisering ellers i maskinvare og programvare. Dette bytte av terminaler forenkler kretskortutlegget og kompenseres for i programvaren.

Ved å studere figuren ovenfor, kan det legges merke til at kanal 3 måler referansesignalet V_{Ref}. Målingen dette signalet inngår i beregning av hver akses kanalforsterkning.



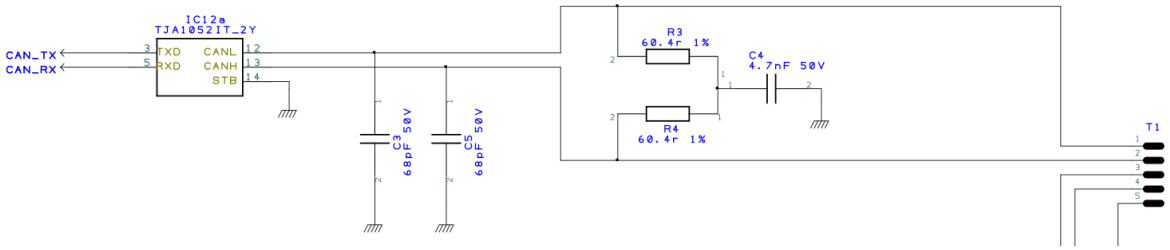
16 bit datagrensesnitt Konfigurasjon

Figur 4.44: Utsnitt fra kretsskjema, digitale grensesnitt til ADC.

Figur 4.44 viser grensesnittet mellom mikrokontroller og analog til digitalomformer MAX1047. Grensesnittet består av to logiske grensesnitt. Grensesnittet til venstre i figuren blir i hovedsak benyttet til å sende måleverdier til mikrokontroller. Grensesnittet til høyre i figuren benyttes for konfigurasjon av analog til digitalomformeren.

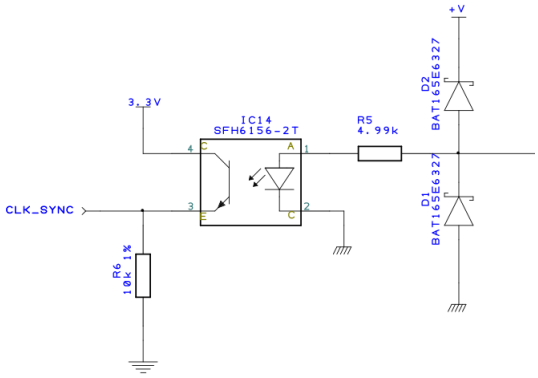
Kommunikasjon

Det er valgt å opprette to galvanisk isolerte kommunikasjonsgrensesnitt for kommunikasjon med eksternt utstyr. For å sende og motta meldinger grensesnittet basert på CAN 2.0B standarden [9].



Figur 4.45: Utsnitt fra kretsskjema, can.

Figur 4.45 viser hvordan candriver TJA1052i [24] er inntegnet i skjema. Driveren er galvanisk isolert og er i stand til å kommunisere over et CAN 2.0B nettverk med hastigheter inntil 2Mb/s. Videre kan det legges merke til at anbefalt busterminering er benyttet. Anbefalt terminering vises i datablad [24] og i figur 4.33.



Figur 4.46: Utsnitt fra kretsskjema, synkroniser.

For at prosesseringsnoden skal kunne synkronisere de ulike sensornodene, er det valgt å designe et eget grensesnitt som støtter dette. Figur 4.46 viser valgt kretsløsningen. For at dette grensesnittet skal være galvanisk isolert, er det valgt å benytte optokobler SFH6156 [31]. I skjema har denne komponenten referanse IC14.

For at optokobleren skal virke som tenkt, må resistansene R5 og R6 beregnes. Fra datablad [31] finner vi følgende nyttig informasjon:

Benevnelse	Beskrivelse	Verdi
$I_{C,max}$	Maksimal kollektorstrøm	50mA
V_{CEO}	Maksimal kollektor-emitterspenning	70V
V_{ECO}	Maksimal emitter-kollektorspenning	7V
V_{CEsat}	Typisk kollektor emitter metning	0.25V
$I_{F,max}$	Maksimal diodestrøm	60mA
V_F	Typisk spenningsfall over diode	1.25V
I_C/I_F	Typisk forhold mellom I_C og I_F	45%

Tabell 4.30: Viktige parametere fra datablad [31].

Basert på informasjonen i tabell 4.30 kan nå verdiene til resistansene beregnes. Vi starter med å beregne resistansen R5:

$$R5 = \frac{V_{bus} - V_F}{I_F}$$

For å kunne beregne resistansen R5, er det tydelig at det er behov for mer informasjon. Det er på et tidligere tidspunkt valgt at busspenningen $V_{bus} \approx 6.5V$. Strømmen gjennom dioden velges nå til 1mA. Denne informasjonen settes inn i ligningen ovenfor og R5 kan beregnes.

$$R5 = \frac{6.5V - 1.25V}{1mA}$$
$$R5 = 5250\Omega$$

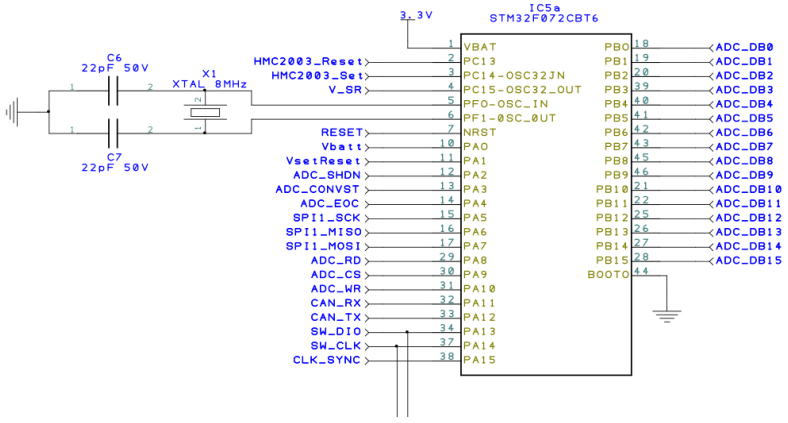
Beregnet verdi til R5 er ikke en standard verdi og det velges nærmeste standardverdi som er funnet til å være $4.99k\Omega$. Endringen dette måtte medføre anses som marginal og uten behov for konsekvensanalyse.

Resistansen R6 fungerer som en *pull-down* resistans. Dette innebærer at R6 ikke må påvirke signalnivået ved aktivt signal. Minste resistans som kan benyttes for å unngå at R6 påvirker det aktive signalet kan beregnes ved:

$$R6_{min} = \frac{V_{cc} - V_{CEsat}}{(I_C/I_F) \cdot I_F}$$
$$R6_{min} = \frac{3.3V - 0.25V}{0.45 \cdot 1mA}$$
$$R6_{min} \approx 6.8k\Omega$$

Basert på beregningen ovenfor velges R6 til $10k\Omega$. Valgt verdi er betydelig større en kravet til laveste resistans og dempning av signalet unngås.

Mikrokontroller

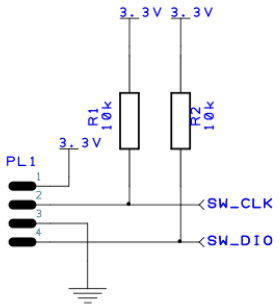


Figur 4.47: Utsnitt fra kretsskjema, mikrokontroller.

Figur 4.47 viser mikrokontrolleren STM32F072 og de ulike signaltilkoblingene til denne. Fra figuren ser vi at alle tilgengelige porter er benyttet til signaler. Det kan videre legges merke til at signalene som benyttes ved programmering av kretsen deles med ekstern DAC. Denne delingen

av grensesnitt, medfører at mikrokontroller må holdes i resatt tilstand ved programmering, under normal drift har dette ingen ulemper.

Klokkefrekvensen til mikrokontrolleren er 8MHz. Denne blir ved help av intern multiplikator, multiplisert opp 6 ganger og resulterende klokkefrekvens blir da 48MHz.

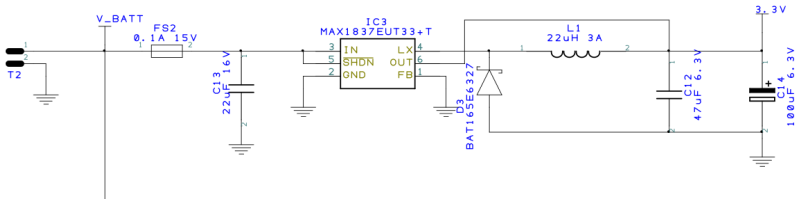


Figur 4.48: Utsnitt fra kretsskjema, programmering av mikrokontroller.

Figur 4.48 viser programmeringsgrensesnittet mellom en STLink V/2 kompatibel programmerer og mikrokontroller.

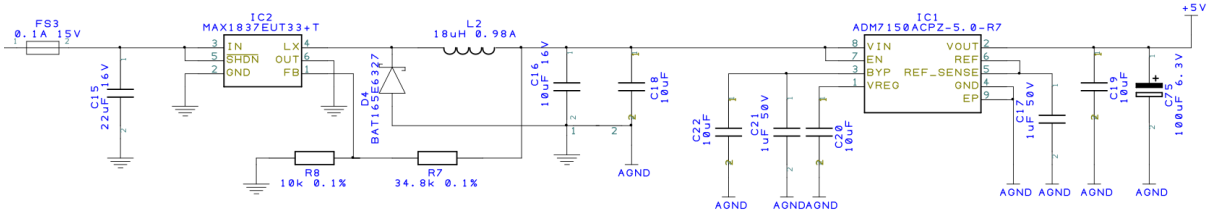
Strømforsyning

Hver sensornode er bestykket med 4 ulike strømforsyninger, hvor 3 av disse skal studeres i dette delkapittelet. Design av forsyningen som opptransformerer spenningen V_{SR}, anses som grundig forklart i kapittel 4.1.7.



Figur 4.49: Utsnitt fra kretsskjema, digital 3.3V forsyning.

Figur 4.49 viser den digitale spenningsforsyningen. Denne regulerer spenningen til den digitale delen av sensornoden ved hjelp av svitsjregulatoren MAX1837 [20]. Fra databladet tilhørende kretsen kan det observeres at regulatoren leveres med ulike faste utspenninger. I dette designet er regulator med utspenning lik 3.3V valgt. Beregning av komponentverdier er utført ved hjelp av designverktøyet EE-Sim fra Maxim Integrated [1]. I figuren er det valgt å plassere stjernepunktet for jording ved kondensator C14.



Figur 4.50: Utsnitt fra kretsskjema, analog 5.0V forsyning.

Figur 4.50 viser den analoge spenningsforsyningen. For å danne en stabil og mest mulig støyfri analogforsyning er det ønskelig å benytte en lineærregulator. Fra tapsberegninger i kapittel 4.1.7, ble det raskt avklart at spenningsnivået måtte pre-reguleres for å unngå uønsket tap i den lineære regulatoren.

I valgt løsning er kretsen MAX1837 [20] benyttet. Som nevnt ovenfor, leveres denne regulatoren med definerte utspenninger. Det er imidlertid mulig å overstyre dette ved å tilbakekoble den regulerte spenningen. Kretsen har da behov for to resistorer, som i figur 4.50 kan gjennkjennes som R7 og R8.

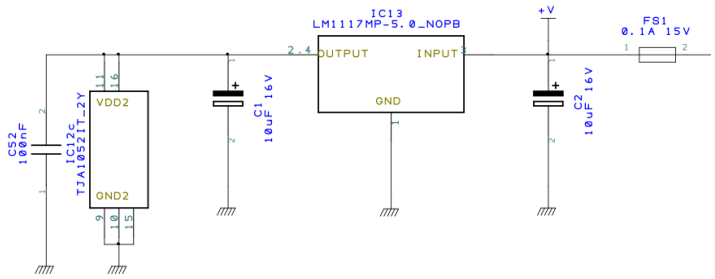
Verdien til resistansene er gitt av ligning 4.41 :

$$R_7 = R_8 \left[\left(\frac{V_{out}}{V_{FB}} \right) - 1 \right] \tag{4.41}$$

Hvor $V_{FB} = 1.25$ er gitt av datablad [20] og V_{out} er valgt til 5.5V.

Beregning av komponentverdier er også her utført ved hjelp av designverktøyet EE-Sim fra Maxim Integrated [1]. I figuren er det valgt å plassere stjernepunktet for jording ved kondensator C16. Broen mellom digital og analog jord er inntegnet mellom kondensatorene C16 og C18.

Pre-regulert spenning er nå tilpasset lineærregulatoren ADM7150 [7] og den analoge forsyningen kan dermed lineærreguleres, uten vesentlig tap i virkningsgrad. Fra datablad [7] er det oppgitt at kretsen finnes tilgjengelig med flere ulike utspenninger og i dette designet er 5.0V varianten valgt. For detaljer og argumentasjon rundt valg av lineærregulatoren ADM7150, henvises det til kapittel 4.1.7.



Figur 4.51: Utsnitt fra kretsskjema, galvanisk isolert 5.0V forsyning.

Figur 4.51 viser regulator LM1117MP-5.0 [30], som her benyttes for å regulere spenningen til den galvanisk isolerte delen av cankontrolleren TJA1052i. Den lineære regulatoren forsynes med spenningen $V_{bus} \approx 6.75V$, som reguleres av overgangen. For nærmere beskrivelse av spenningen V_{bus}, vises det til kapittel 4.4.1.

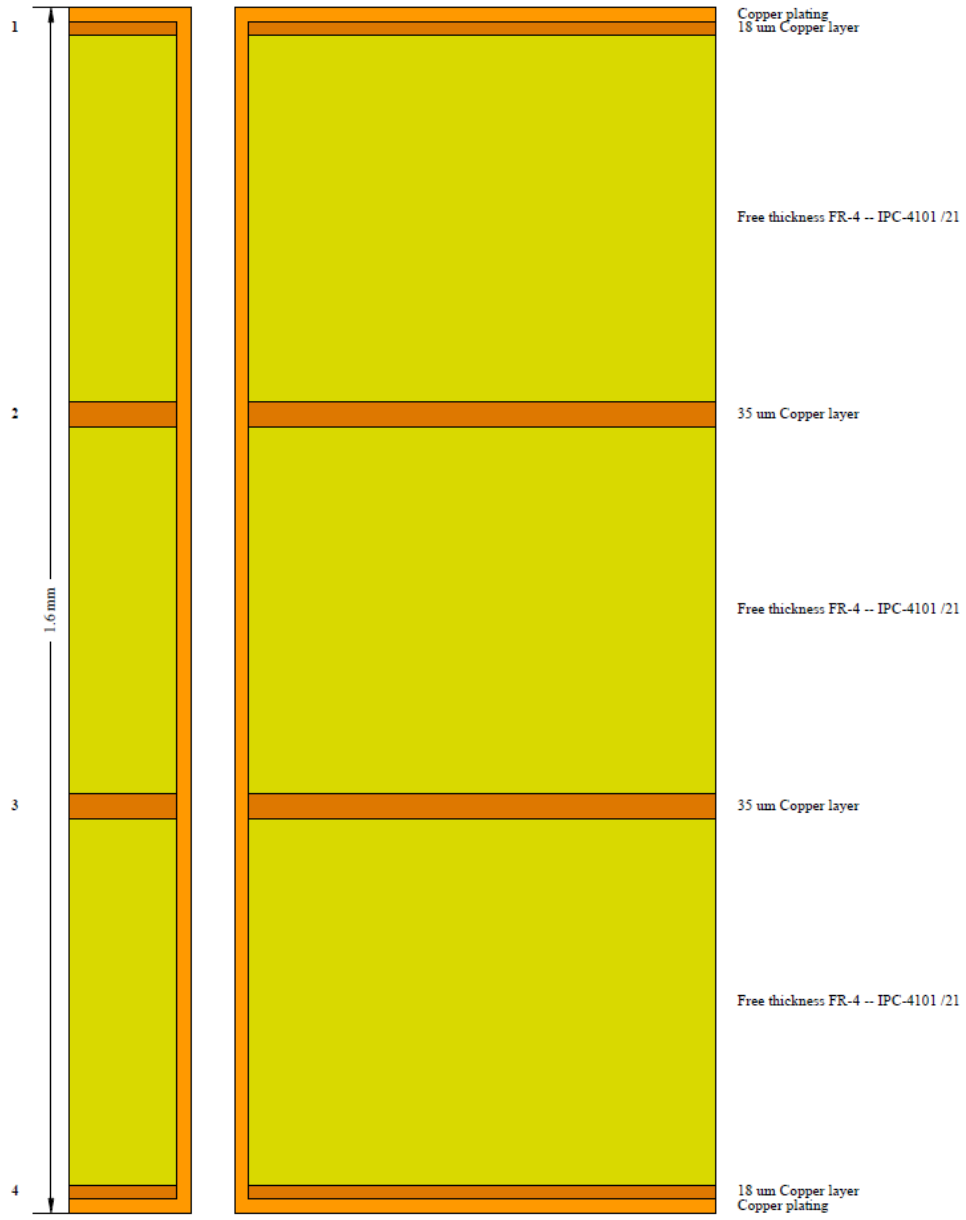
4.1.9 Kretskort

Som nevt i de foregående kapitler har vi i denne oppgaven høye krav til de ulike komponentene som inngår i den analoge kretsløsning. Når vi skal designe kretskortet blir det nå viktig å konstruere det slik at de analoge kretser og baner ikke påtrykkes støy fra digitale kretser. Hvor kretsene plasseres i forhold til hverandre og ikke minst hvor banene legges på kretskortet vil direkte påvirke kvaliteten til de analoge målingene. Det er derfor ofte nyttig å dele inn kretskortet i ulike soner hvor det forsøkes å unngå å plassere baner tilhørende andre strømforsyninger.

Vi skal nå se på hvordan sensor kretskortet er designet og hvordan de ulike problemstillingene rundt blandingen av digitale og analoge kretser er forsøkt løst.

Antall lag

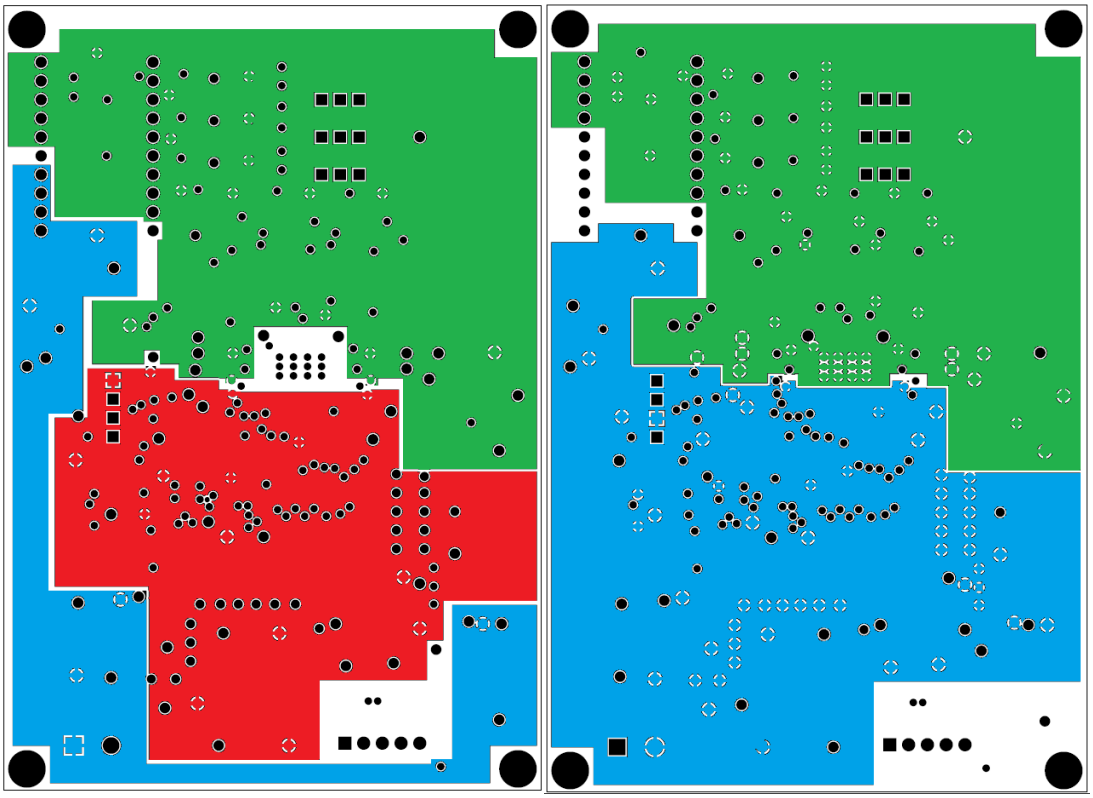
Det finnes mange konfigurasjonsmuligheter når det kommer til antall lag som vi kan benytte når kretskort skal designes. Det enkleste og rimeligste er ofte å benytte kretskort bestående av kun 2 lag. Dette gir en løsning der vi ikke har mulighet til å dedikere ulike lag til forsyningsspenning og medfører dermed at vi må trekke baner for å forsyne komponentene med både strøm og signal. Dersom kretskortet vi designer består utelukkende av digitale eller analoge signaler kan dette gi en tilfredsstillende løsning. Hvis vi derimot blander analoge og digitale signaler på samme kretskort, vil denne løsningen kunne bidra til en betydelig andel støy påtrykkes den analoge delen av kretskortet. Dersom vi ønsker å opprettholde presisjon i de analoge signalene er denne løsningen dermed mindre tilfredsstillende.



Figur 4.52: 4 lag kretskort.

Det er i denne oppgaven valgt å benytte kretskort inndelt i 4 lag. Denne inndelingen ser vi illustrert i figur 4.52. Lagene er konfigurert slik at øvre og nedre lag er hovedsaklig dedikert til digitale og analoge signaler og de midtre lagene består av strømforsyning. Med denne konfigurasjonen oppnår vi å forenkle utlegget ved at vi har tilgjengelig strømforsyning i de indre lagene. Vi unngår dermed å trekke baner til dette formålet og kretskortet kan dermed konstrueres mer kompakt. Foruten den nevnte forenklingen bidrar de indre lagene til at alle komponenter har en lavohmig tilgang til strømforsyning. Dette vil igjen bidra til å danne en mer stabil og et mer likt spennings-potensiale for alle komponenter. Vi oppnår dermed å redusere støy og potensialforskjeller indusert på kretskortet.

Forsyning - indre lag



Batteri(blå), Digital 3.3V(rød), Analog 5.0V(grønn) Jordplan. Digital(blå), Analog(grønn)

Figur 4.53: Kretskortets indre lag.

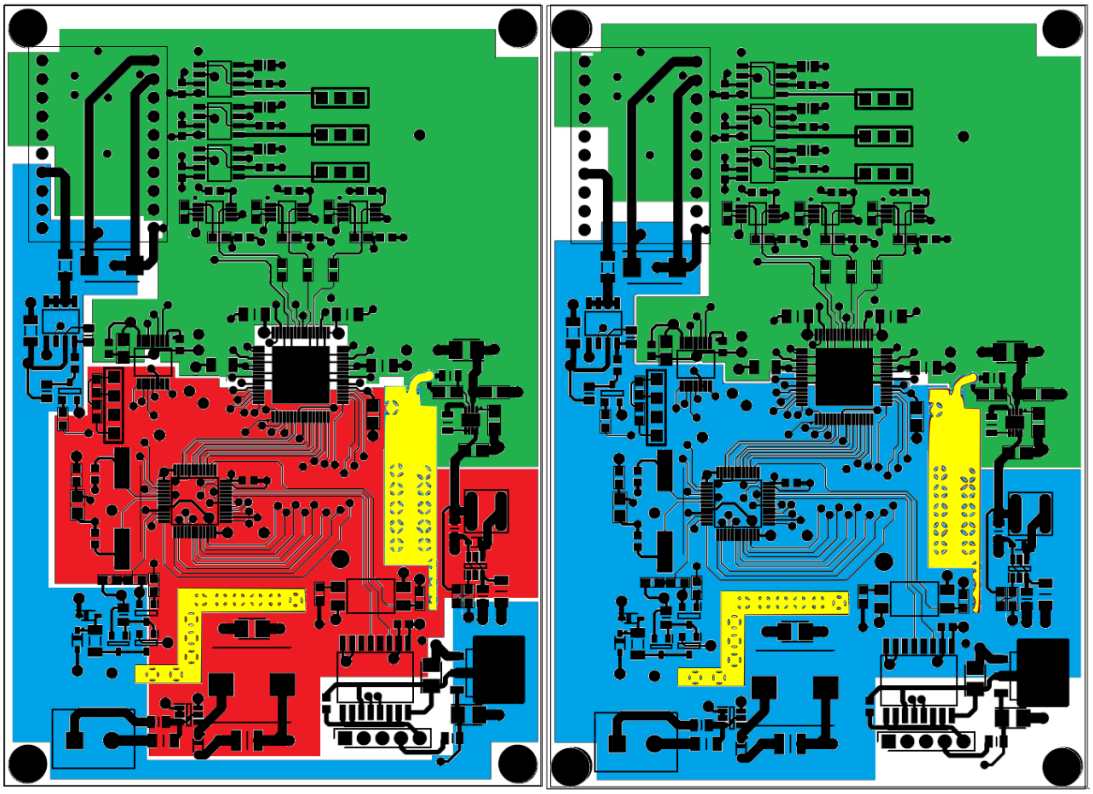
Fra figur 4.53 ser vi hvordan kretskortets indre lag er konstruert. Som vi så på i delkapittel 4.1.7 har vi behov for ulike strømforsyninger til den analoge og den digitale delen av kretskortet. I figuren til venstre vises dette som separate plan for de ulike strømforsyningene. Da oppnås det at hver krets har tilgjengelig den ønskede forsyningsspenning, samtidig som de ulike forsyningene ikke vil påvirke hverandre.

Ved å studere figuren til høyre, ser vi kretskortets ulike jordplan. Som vi ser fra figuren er også disse separert og vi unngår dermed blanding av digitale og analoge jordstrømmer på det analoge planet. Fra skjema ser vi at det i flere tilfeller er nødtvendig til å måle analoge signaler som forsynes av batterispenningen. Når dette skal gjøres er det viktig at disse signalene beholdes høyohmige og da reduserer dermed de uønskede jordstrømmer og medførende potensialendring i det analoge plan.

Signaler - ytre lag

De to ytre lagene på kretskortet består hovedsaklig av komponenter og signalbaner som trekkes mellom disse. Vi skal nå først se på kretskortets øvre lag og hvordan dette er konstruert for å

unngå at de ulike kretsene påvirker hverandre.



Ulike forsyninger Ulike jordpotensialer

Figur 4.54: Øvre plan vist i forhold til forsyningsplanene.

Fra figur 4.54 ser vi det øvre signallaget og hvordan dette er inndelt i forhold til de ulike strømforsyningene. Fargekodene til de indre lagene er som i figur 4.53.

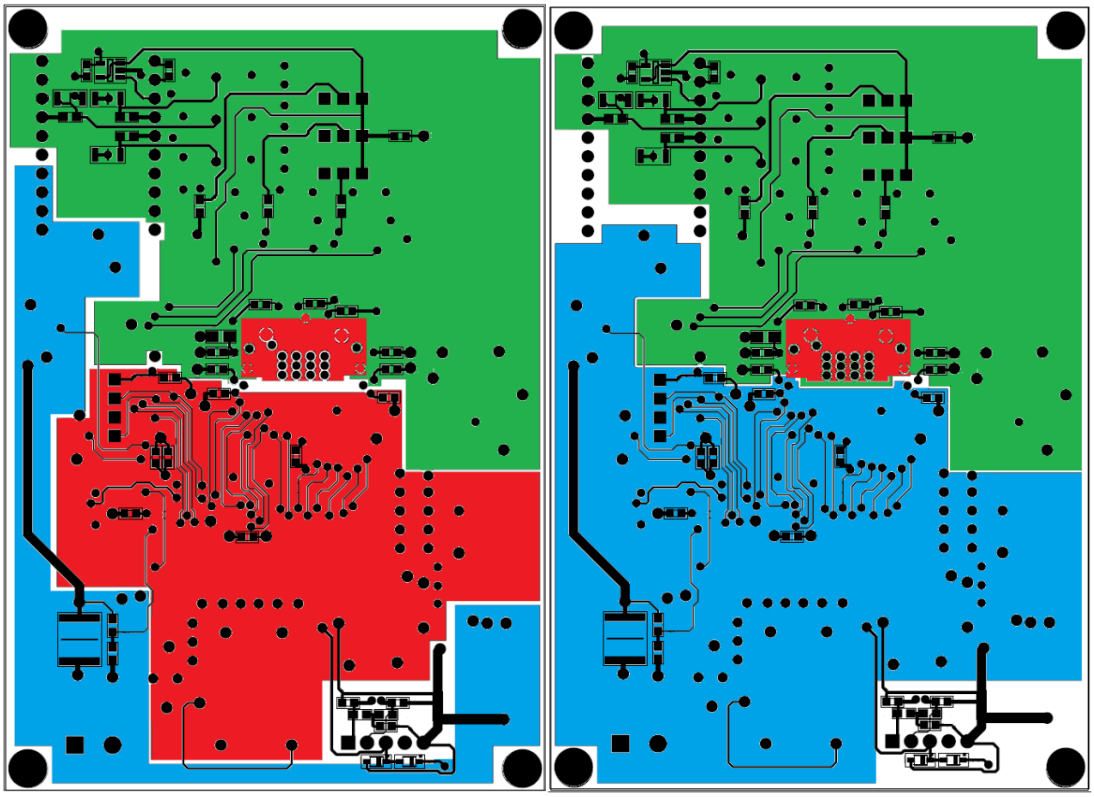
I designet av dette kretskortet har jeg valgt å plassere hele det analoge trinnet innenfor det analoge planet (grønn). De digitale kretser med tilhørende signaler er plassert innenfor det blå og røde plan. Denne måten å konstruere kretskortet på vil sørge for at de digitale signaler i minst mulig grad kan påvirke de analoge sensorsignaler.

De to gule planene er strategisk plassert for å skjerme øvrige komponenter og baner fra støy som produseres av de svitsja regulatorene. Disse planene danner en lavohmig forbindelse til digital jord. Dersom det på et senere tidspunkt oppleves problemer med utstrålt støy fra regulatorene, kan planene benyttes til å jorde en skjerm. Vi danner da et såkalt *Faradays bur* som fjerner størstedelen av den utstålte elektromagnetiske støy som måtte bli produsert av regulatorene. Foruten nevnte funksjon vil også planet til høyre i figurene knytte sammen det digitale og analoge jordplan i et felles stjernepunkt. Dette benyttes for både å danne et tilnærmet felles nullnivå, samtidig som det lar signalstrømmer fra de analoge signalene returnere.

Vi kan også legge merke til at begge forsyningslagene er rutet utenom nederste venstre hjørne. I dette området er den galvanisk isolerte CAN kontrolleren TJA1052 plassert sammen med

nødvendige kretser. For å unngå at eventuelle transienter skal kunne utlades via kretskortets srømforsyninger er det valgt å ikke plassere plan under kretsen.

Vi skal nå se på det nedre signalplan.



Ulike forsyninger Ulike jordpotensialer

Figur 4.55: Nedre plan.

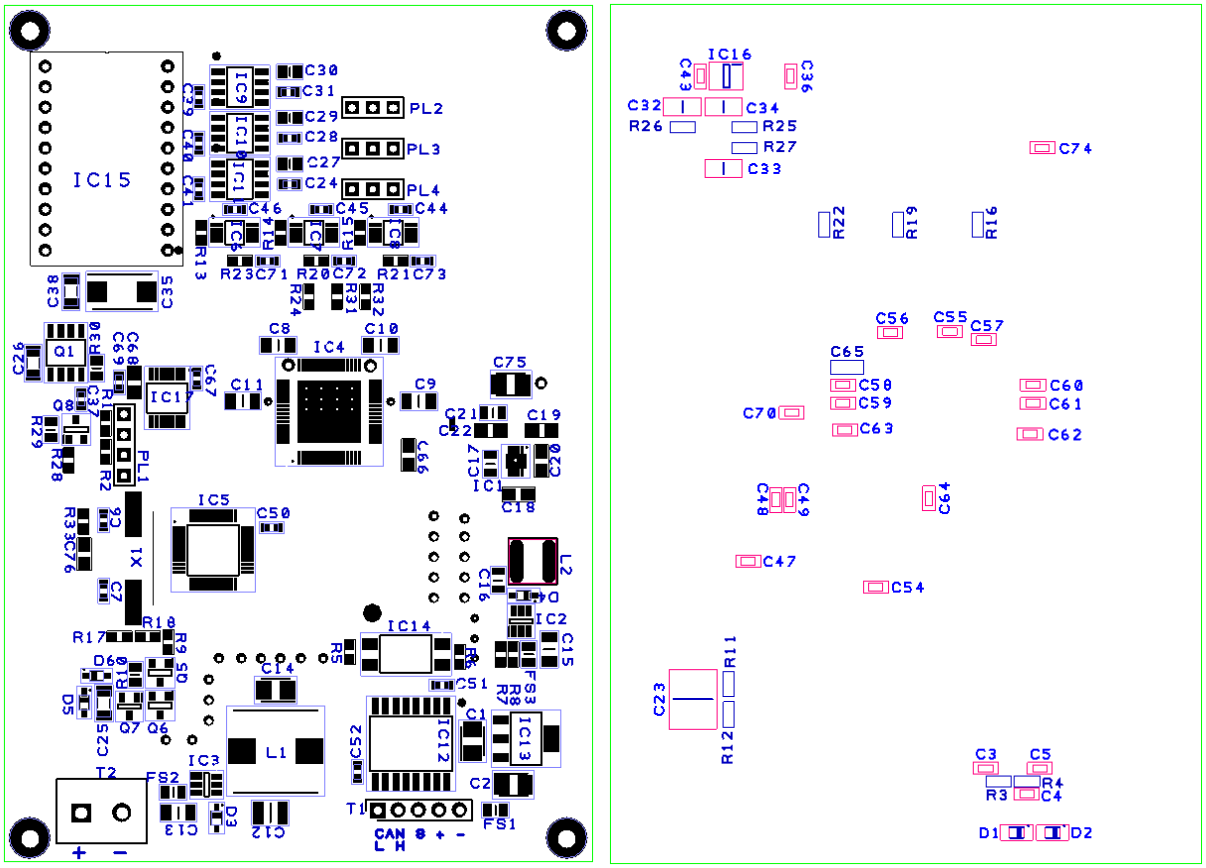
Fra figur 4.55 ser vi det nedre signallaget og hvordan dette er inndelt i forhold til de ulike strømforsyningene. Fargekodene til de indre lagene er som i figur 4.53.

Det nedre laget blir hovedsaklig benyttet som et hjelpelag for å rute baner som ville krysset andre baner på det øvre laget. Laget er også benyttet til å plassere komponenter som av ulike grunner ikke lar seg plassere på det øvre laget. Som for det øvre laget er det også her etablert soner på kretskortet som representerer de ulike strømforsyninger.

Vi kan også legge merke til et rødt lite plan omtrent midt på kretskortets nedre lag. Det er i datablad [21] tilhørende kretsen MAX11047 gitt at portene *RDC* og *RDC_SENSE* skal kobles til et separat plan, se figur 4.23, for å oppnå gitt presisjon. Planet skal så avkobles til analog jord med minst 2 avkoblingskondensatorer. I dette designet er det etter anbefaling fra [21], valgt å benytte 4 kjeramiske X7R kondensatorer på $22\mu F$ for avkobling.

Komponentplassering og tilkoblinger

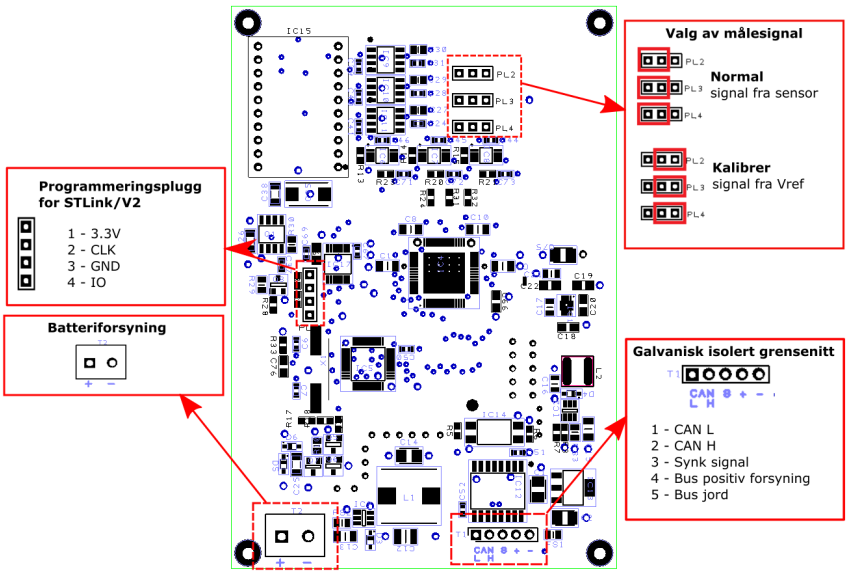
Det skal her gis en oversikt over hvor de ulike komponentene er plassert på kretskortet og hvilke tilkoblingsmuligheter som finnes.



Plassering topp Plassering bunn

Figur 4.56: Komponentplassering.

Figur 4.56 viser hvordan komponentene er plassert på topp og bunnlaget.



Figur 4.57: Tilkoblinger.

Figur 4.57 viser en oversikt over de tilkoblingsmuligheter som finnes på sensornoden og disse skal nå forklares.

Valg av målesignal - Ved hjelp av denne stiftlisten, velges hvilket signal som skal måles av ADC. Ved kalibrering må kortslutningsstiften settes i posisjon til høyre. Når signal fra sensor skal måles, settes kortslutningsstiften i posisjon til venstre.

Galvanisk isolert grensesnitt - For kommunikasjon med eksternt utstyr er det valgt å galvanisk isolere signalene. Grensesnittet har separat forsyning og denne forsyner den uisolerte delen av grensesnittet.

Pluggen kobles til signalene fra overgang mellom prosesseringsnode og det isolerte nettverket. Overgangen blir nærmere forklart i kapittel 4.4.

Programmeringsplugg for STLink/V2 - Denne pluggen benyttes for å programmere mikrokontrolleren ved hjelp av en STLink/V2 kompatibel programmerer.

Batteriforsyning - Plugg for å koble til ekstern forsyning. Tilkoblet kilde bør anses som støyfri og med en spenning fra 10V til 14V.

4.2 Prosesseringsnode

Det skal her gis en kortfattet introduksjon til hvordan prosseseringsnoden er tenkt designet, med tilhørende oppgaver. I denne oppgaven ble det valgt å utføre oppgavene til prosesseringsnoden i MATLAB. I et realisert system må prosesseringsnoden inngå i det endelige målesystemet.

I et realisert målesystem vil de mest sentrale oppgavene til prosesseringsnoden være:

Konfigurere sensornoder - Prosesseringsnoden bør ha mulighet for å starte og stoppe magnetfeltmålinger. Dersom det oppdages unøyaktige målinger, må det også være mulig å igangsette offsett beregning i hver sensornode.

Motta måledata - Noden mottar måledata fra sensornoder i nettverket og lagrer disse i tilhørende tabeller.

Beregne invarianter - Prosesseringsnoden skal på bakgrunn av måledata fra sensornodene beregne invarianter og estimert posisjon i et magnetfelt.

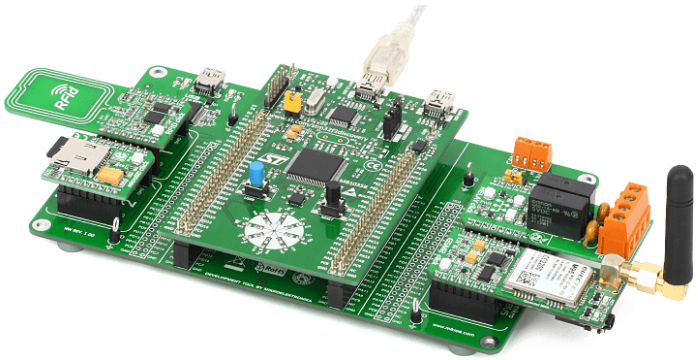
Oppdatere ekstern posisjonsregulator med posisjonsdata - Oppdaterte posisjonsestimat sendes til en ekstern posisjonsregulator.

Måle og korrigere målinger basert på orientering - Prosesseringsnoden skal måle og korrigere målingene for orientering i rommet og i forhold til faseledere.

Ved å oppsummere hva som kreves av en mikrokontroller for å være i stand til å utføre overnevnte oppgaver, er det mulig å fastsette noen krav til egenskaper som valgt mikrokontroller må ha.

Valgt mikrokontroller bør inneholde flere separate kommunikasjonsgrensenitt. Dette muliggjør at kommunikasjon mellom noder og ekstern posisjonsregulator kan foregå på isolerte nettverk og forsinkelser på grunn av prioritet eller bus last unngås. Begge nettene kan med fordel være basert på CAN-buss.

I beregning av invariantene kreves det at valgt mikrokontroller beregner en presis FFT tilhørende sensorenes akser. Beregning av FFT kan anses som svært ressurskrevende og det er dermed sentralt å velge en mikrokontroller med tilstrekkelig klokkehastighet. For å opprettholde presisjon i beregningene av både FFT og til slutt invarianter, må det velges en mikrokontroller med integrert støtte for flyttall. Med dette oppnås høyere presisjon og hurtigere beregninger.



Figur 4.58: STM32F3 skjold fra mikroElektronika.

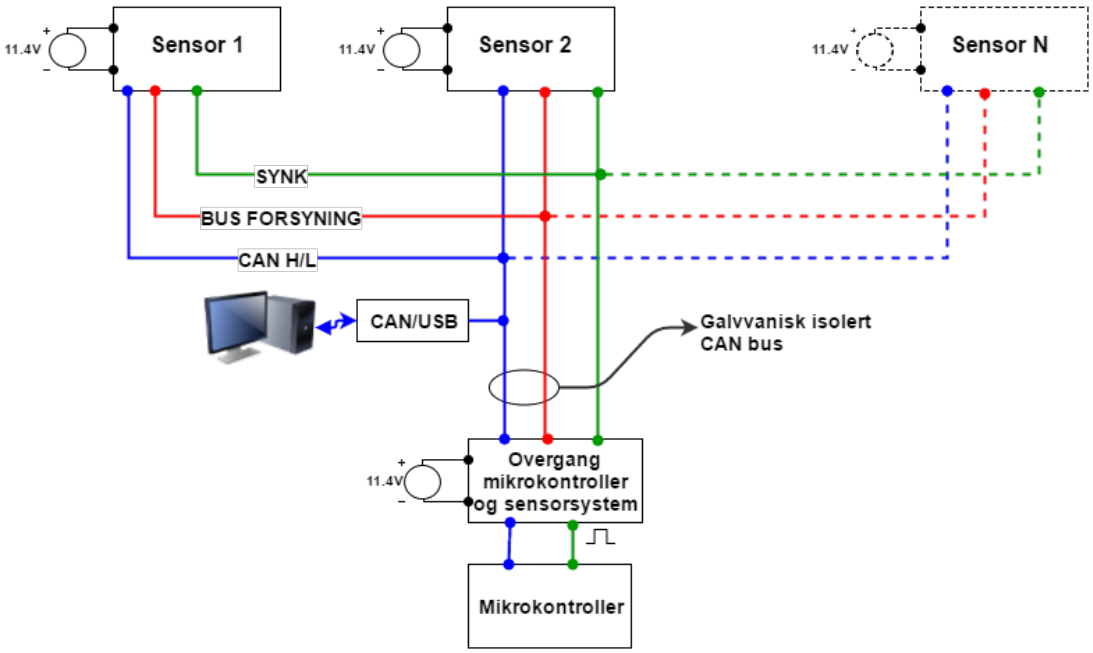
En aktuell mikrokontroller som oppfyller definerte krav er STM32F303 [29]. Denne mikrokontrolleren er basert på ARM®Cortex®-M4 kjernen og har dermed integrert støtte for beregning av flyttall. Mikrokontrolleren oppfyller også ønsket om flere separate kommunikasjonsgrensesnitt.

Ved å velge denne mikrokontrolleren kan utviklingskortet STM32F3discovery benyttes. Til dette kortet er har mikroElektronika utviklet et egnet skjold som kretskortet kan plasseres i. Skjoldet med montert utviklingskort vises i figur 4.58

Foreslått design krever imidlertid en overgang mellom utviklingkortet og den isolerte CAN-bussen. Denne overgangen og dens funksjoner beskrives nærmere i kapittel 4.4.

4.3 Målesystem

Systemstrukturen som illustreres i figur 4.59 viser en overordnet skisse over hvordan målesystemet er planlagt. Denne illustrasjonen viser at målesystemet er fleksibelt i forhold til antall sensorer som inngår i det samlede system. I målesystemet vil største antall sensorer begrenses av datalast på CAN-buss, samt kabellengde.



Figur 4.59: Overordnet systemstruktur.

Det er valgt å implementere et fullstendig galvvanisk isolert CAN-nettverk. Med dette valget oppnås det å fjerne potensielle feilkilder som jordstrømmer og kabelbunden støy introduserer. For å kunne opprettholde en fullstendig isolering under eksperimenter, ble det nødvendig å designe en overgang mellom mikrokontroller og nettverket. Funksjoner og virkemåte til denne overgangen forklares nærmere i kapittel 4.4.

Ved å benyttes separat batteriforsyning til hver enkelt sensor oppnås det samme fordeler som ved

å benytte galvanisk isolert nettverk. Samtidig vil et system basert på separat batteriforsyning være mer fleksibelt i forhold til senere utvidelser.

Det er ønskelig at samtlige sensorer i nettverket måler magnetfeltet ved samme tid. For å oppnå dette er det valgt å legge til et felles synkroniseringssignal. Ved hjelp av dette signalet kan hver sensor synkroniseres med mikrokontrolleren. Dette vises nederst i figur 4.59.

Mikrokontrolleren er tiltenkt å kontrollere den nevnet synkronisering og utføre nødvendige beregninger basert på motatt måledata. For å være i stand til å utføre beregninger med høy presisjon og hurtighet, må det her velges en modell som støtter beregning av flyttall. Funksjoner og virkemåte til denne forklares nærmere i kapittel 4.2.

For datafangst er det mulig å koble en datamaskin til nettverket ved å benytte en egnet overgang. Dette krever imidlertid at det utvikles et program som er i stand til å kommunisere med grensesnittet og dekode definerte CAN-meldinger.

4.4 Overgangskort

Dersom kretskortet til prosesseringsnoden hadde vært designet for dette prosjektet, ville overgangen blitt en naturlig del av kretskortet. Når det ble valgt å benytte en løsning som skissert i figur 4.58, er det behov for et ekstra kretskort som håndterer ovegangen mellom prosesseringsnode og sensornoder.

I målesystemet vil overgangen som må utvikles ha følgende oppgaver:

CAN-kommunikasjon - Danne ovegang mellom isolert buss og prosesseringsnode.

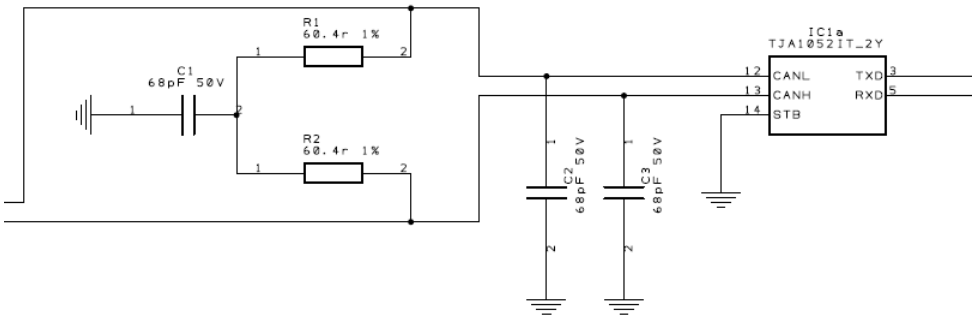
Bus forsyning - Kretskortet må forsyne den isolerte delen av sensorbus med regulert forsyning. Spenningsnivået til denne forsyningen bør være noenlunde tilpasset sensornodenes regulator og må kunne levere tilstrekkelig med strøm.

Buffre synkroniseringssignal - Overgangen skal buffre synkroniseringssignalet fra prosesseringsnode slik at alle sensornoder er i stand til å tolke dette korrekt.

4.4.1 Skjema

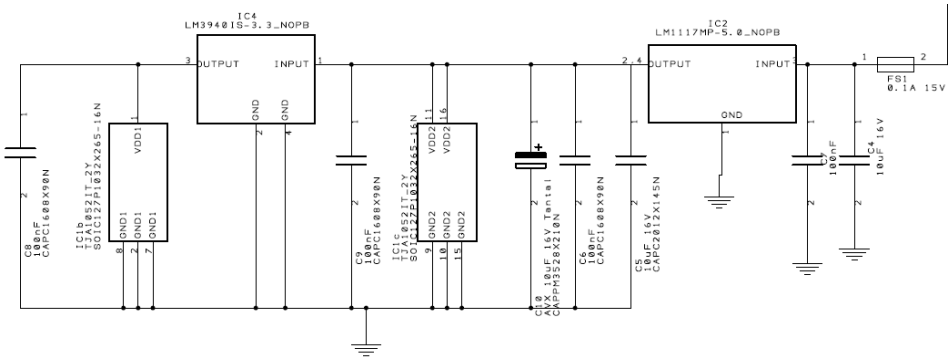
I design av dette kretskortet er det valgt å basere komponentvalg på valg fra sensornoden. Argumentasjoner for valg blir dermed de samme som for sensornoden og gjenntas ikke uten at løsninger benyttes på en annen måte. Å basere dette designet på tidligere valg vil gi fordeler som redusert komponentkost og redusert utviklingstid. For en fullstendig oversikt over skjema, henvises det til vedlegg D.2. Leser bør imidlertid gjøres oppmerksom på at signalretningen i skjemaet er fra høyre til venstre.

CAN-grensesnitt



Figur 4.60: Galvanisk isolert CAN-kontroller.

Skjema i figur 4.60 viser grensesnittet mellom prosesseringsnoden og det galvanisk isolerte nettverket er designet. Den valgte løsningen er fullstendig tilsvarende som for sensornoder.



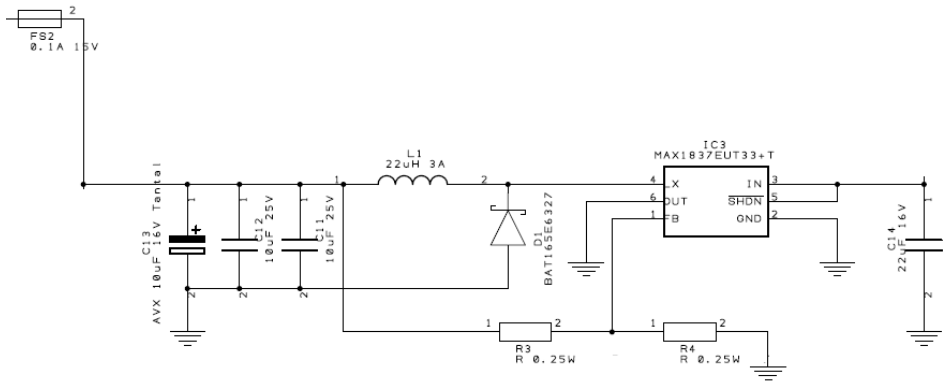
Figur 4.61: Trinnvis regulering av spenning til CAN-kontroller.

Fra datablad [24] er det gitt at kretsen TJA1052 krever to ulike spenninger, en for mottakerdelen (3.3V) og en til senderdelen (5.0V). For å oppnå større grad av fleksibilitet vedrørende forsyningsspenning, er det valgt å benytte to trinnvise reguleringer. Regulator IC2 regulerer fra forsyningsspenning til 5.0V, deretter vil regulatoren IC4 regulere spenningen fra 5.0V til 3.3V. Den maksimale inngansspenning blir da bestemt av IC2 og ved å studere tilhørende datablad [30], oppgis denne til $V_{max} = 15.0V$. Da målesystemet baseres på batteriforsyning på 11.4V er dette godt innenfor den maksimale forsyningsspenning til regulator.

Bussforsyning

Siden dette kretskortet kun er ment for å benyttes i forbindelse med eksperimenter med to sensornoder, er kravet til kontinuerlig strøm kretskortet må kunne levere, gitt av det maksimale strømtrekk av to TJA1052 can kontrollere. Fra datablad til TJA1052 [24] oppgis $I_{max,TJA1052} = 70mA$, dette gir et samlet krav til regulator $I_{min} = 200mA$. Fra datablad tilhørende regulator

MAX1837 [20] oppgis $I_{max,MAX1837} = 250mA$. Regulator MAX1837 kan dermed benyttes også i dette designet.



Figur 4.62: Regulering av bussforsyning.

Designet av regulatoren vises i figur 4.62. Ved å studere figuren ser vi tydelig at dette tilsvarer design benyttet i sensornoden. Den eneste forskjellen er at det her valgt å benytte en høyere regulert utspenning i forhold til hva som finnes på sensornoden. Spenningen ble valgt på grunnlag av innspenning som spenningsregulator LM1117 [30] krever for å danne en stabil forsyning til CAN-kontroller på hver sensornode. Det kan settes opp følgende uttrykk for å beregne denne:

$$\begin{aligned} V_{min} &= V_{drop} + V_{reg} \\ &= 1.25V + 5.0V \\ V_{min} &= 6.25V \end{aligned}$$

For å kompensere for unøyaktigheter i oppgitte data, velges reguleringsspenningen til 6.75V.

Motstandene R3 og R4 i figur 4.62 skal nå bergnes slik at utspenning blir 6.75V. Fra datablad til regulatoren MAX1837 [20] er ligning 4.42 gitt for å beregne regulert spenning, V_{out}

$$R_3 = R_4 \left[\left(\frac{V_{out}}{V_{FB}} \right) - 1 \right] \tag{4.42}$$

Hvor $V_{FB} = 1.25$ er gitt av datablad [20].

Ligning 4.42 inneholder for mange ukjente og en av motstandene må velges. Motstanden R_3 velges til $120k\Omega$ og ligningen kan dermed løses med hensyn på R_4.

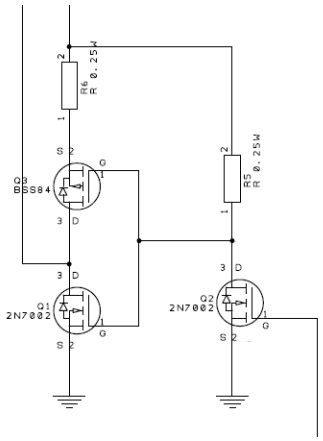
$$R_4 = \frac{R_3}{\left(\frac{V_{out}}{V_{FB}} - 1\right)}$$

$$R_4 = \frac{120k\Omega}{\left(\frac{6.75V}{1.25V} - 1\right)}$$

$$R_4 \approx 27.3k\Omega$$

Bufring av synkroniseringssignal

Synkroniseringssignalet fra prosesseringsnoden anses som for høyohmig til å sendes direkte til sensornodene. Signalet må derfor buffres og med dette oppnås det å konvertere det høyohmige signalet til lavohmig.

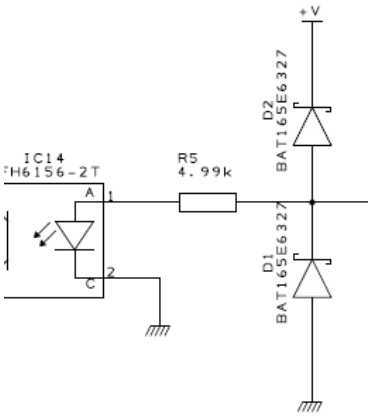


Figur 4.63: Signalbuffer.

Figur 4.63 viser at valgt design er et standard *push-pull*-trinn. Det som derimot ikke går frem av figuren, er forsyningsspenningen. Trinnet forsynes av bussforsyning og som det ble beregnet ovenfor, er $V_{bus} = 6.75V$.

Motstandene R_5 og R_6 skal nå beregnes. Siden motstand R_5 kun er med på å lage korrekt styrespenning med korrekt nivå til transistorene Q_1 og Q_3 er det moderate krav til komponenten. For å unngå unødvendig strømtrekk velges $R_5 = 1k\Omega$. Strømmen gjennom motstanden,I_{R_5}, kan da beregnes til $6.7mA$.

Når motstanden R_6 skal beregnes, må først lasten som trinnet skal drive analyseres.



Figur 4.64: Last.

Figur 4.64 viser lasten på hver sensornode, som *push-pull*-trinnet skal drive. Ved å se bort fra spenningsfallet i optokobleren IC14, kan det maksimale strømtrekket beregnes til:

$$I_{max,pp} = \frac{6.75V}{4.99k\Omega}$$
$$I_{max,pp} \approx 1.35mA$$

For å drive to laster kreves dermed en strøm gjennom R_6 på to ganger den beregnede $I_{max,pp}$. Minste krav til strøm gjennom motstanden blir dermed $\approx 3mA$. I beregninger av motstanden R_6 var kun resistive ledd tatt med. Som følge av at signalet skal sendes over ledninger med varierende lengde og forleggning, kan kapasitive og induktive elementer antas å påvirke signalet negativt. Det er derfor valgt å øke den maksimale strømmen til trinnet slik at signalet blir mindre påvirket av støy og nevnte elementer.

I endelig valg av komponentverdi er denne valgt basert på at $I_{max,pp} \gg 3mA$ og allerede innkjøpt komponent. Verdien som oppfyller disse krav er 100Ω og strømmen gjennom denne kan beregnes ved:

$$I_{R_6} = \frac{6.75V}{100\Omega}$$
$$I_{R_6} = 67.5mA$$

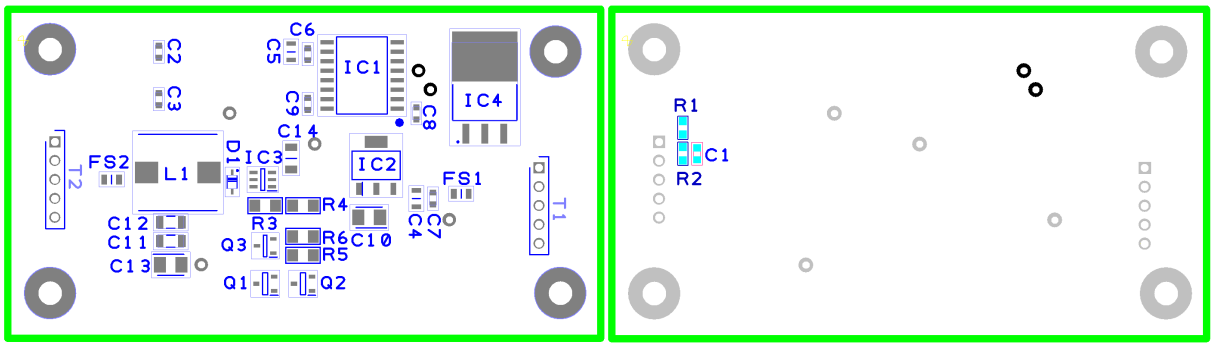
Push-pull-trinnet bør nå være i stand til å buffre synkroniseringssignalet slik at begge sensornoden er i stand til å tolke signalet korrekt.

4.4.2 Kretskort

Siden dette kretskortet er kun ment for å benyttes under eksperimenter, er det utviklede kretskort betydelig enklere enn kretskort til sensornoden. Ved utvikling av dette kretskortet har det

vært sentralt å fokusere på at det skal kunne produseres i en kretskortfres. Dette legger store begrensninger på hvordan kretskortet designes og det er viktig å fokusere på de begrensninger som en kretskortfres har.

Komponentplassering

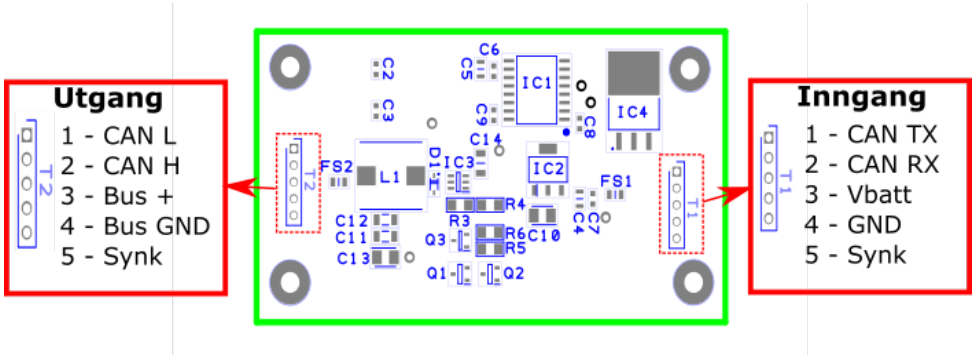


Plassering topp Plassering bunn

Figur 4.65: Komponentplassering.

Figur 4.65 viser hvor de ulike komponentene er plassert på topp og bunnlaget på det produserte kretskort.

Tilkoblinger



Figur 4.66: Tilkoblinger.

Figur 4.66 viser tilkoblingsmuligheter som kretskortet overgang er bestykket med. Til høyre i figuren, vises tilkoblingene hvor prosesseringsnoden kobles til. Til venstre vises tilkoblinger til den isolerte bus.

Kapittel 5

Programvare

5.1 Sensornode

Dette kapittelet vil gi et innblikk i hvordan programvaren til sensornoden er designet og tilhørende problemstillinger løst. Spesifike mikrokontrollermoduler og tilhørende registere vil ikke nødvendigvis bli forklart i detalj og det henvises til datablad [27][26][28] for en dypere forståelse.

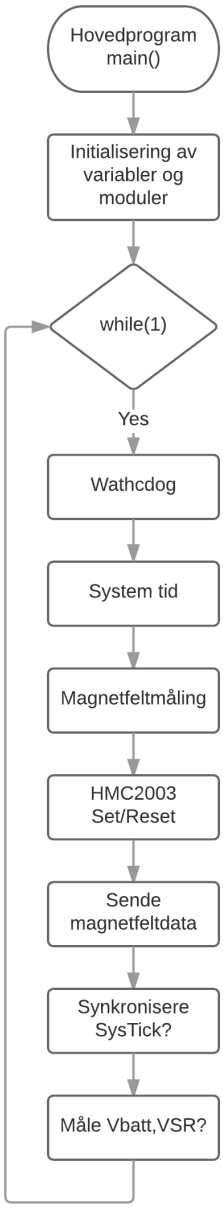
Når det skal utvikles programvare til en STM32F mikokontroller, eksisterer det flere ulike programmeringsspråk, kompilatorer og IDE. I denne oppgaven er det valgt å benytte programmeringsspråket C sammen med den vederlagsfrie IDE CooCox. Fra hjemmesiden til CooCox anbefales det å benytte kompilatoren *GCC-ARM-Embedded*.

For konfigurasjon av pereferimoduler er det valgt å benytte den anbefalte gresesnitt standarden CMSIS. Foruten å forbedre lesbarheten til koden, vil også CMSIS bidra til at koden bli mer portabel og enklere å modifisere i ettertid, av både meg selv og andre.

Kildekoden er valgt å stukturere slik hver modul er fordelt over 3 filer. Modulene har en fil for funksjoner som benyttes i programvaren, en for konfigurasjon av modulen og en headerfil. For å lett kunne skille mellom de ulike filene er det valgt å navngi konfigurasjonsfilene med modulnavn+Conf.c. Kildekoden presenteres i sin helhet i vedlegg C.1. Det anbefales å studere dette for å få en fullstendig oversikt over virkemåte.

5.1.1 Hovedprogram

Vi skal nå se på hvordan programvaren er designet og studere de ulike modulene som kreves for at programvaren kan utføre tiltenkte oppgaver.



Figur 5.1: Overordnet flytdiagram.

Flytdiagrammet i figur 5.1 viser et overordnet flytdiagram over hovedrutinen, med den uendelige løkken *while(1)*. Hovedrutinen main starter med å initialisere variabler og pereferimoduler som benyttes i programvaren. Etter initialisering går programmet over i en uendelig løkke. Denne løkka benyttes for å utføre rutiner som enten krever mye ressurser og dermed uønsket i et avbrudd, eller til rutiner som ikke er tidskritiske. Disse rutinene vil forklares grundigere nedenfor.

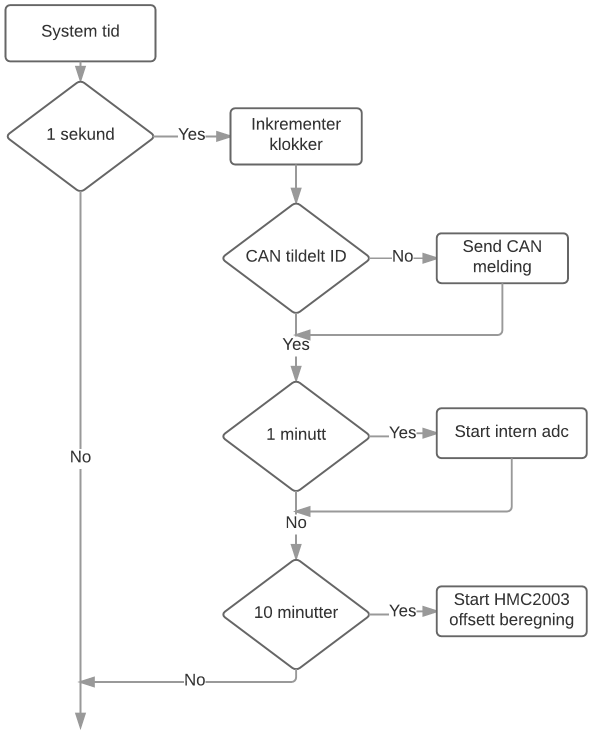
De rutiner som derimot er tidskritiske, har behov for en pålitelig kilde som avbryter program eksekveringen ved bestemte tider eller hendelser. Til dette formålet er det i STM32 implementert en modul som heter SysTick. Denne modulen vil virkemåte og valgt konfigurasjon forklares i delkapittel 5.1.3.

Watch Dog

Den føste blokken i flytskjema 5.1 er *watchdog*. Denne blokken har som funksjon å kvittere til *watchdog* modulen at programmet fortsatt kjører som forventet. I dette programmet er det valgt å implementere denne kvitteringen ved hjelp av en macro med navn *_klappHunden_*.

Systemtid

Blokken systemtid er implementert i hovedprogrammet for å holde orden på systemtid og starte ulike funksjoner ved gitte tider.

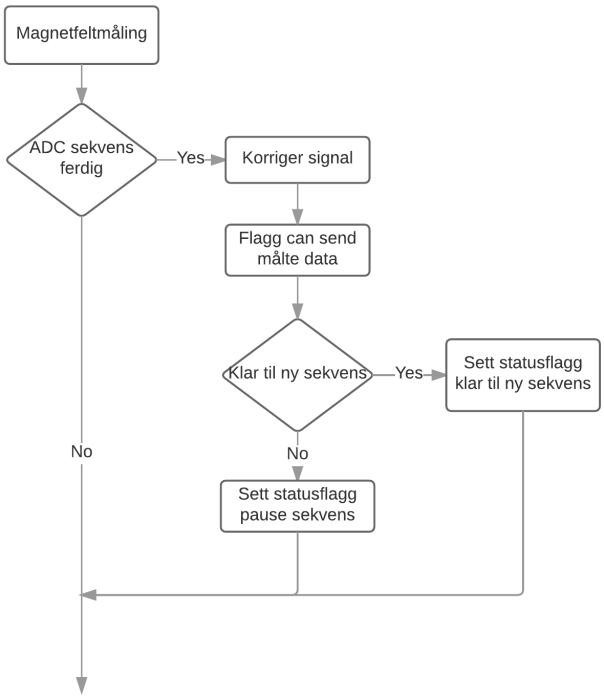


Figur 5.2: Overordnet flytdiagram.

Flytskjemaet tilhørende blokken ser vi i figur 5.2. Fra dette ser vi at blokken utføres hvert sekund og starter da med å inkrementere klokker. Etter at klokkene er inkrementerte sjekkes det først om enhet har fått tildelt en unik CAN-adresse. Dersom denne enheten ikke er tildelt unik adresse, vil den hvert sekund sende en forespørsel om tildeling av adresse. Videre vil det sjekkes om 1 minutt har passert siden sist den interne adc målte spenningene V_{batt} og +VSR. Dersom dette er tilfellet, vil flagg som starter den interne adc settes. Til slutt sjekkes det om 10 minutt har passert siden sist sensoroffsettberegning. Dersom dette er tilfellet, vil flagg som starter offsett beregning settes.

Magnetfeltmåling

Blokken magnetfeltmåling er implementert i hovedprogrammet for å fange opp flagget som settes når en magnetfeltmåling er utført.



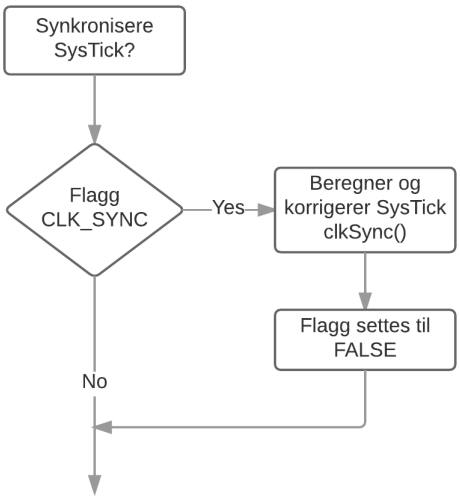
Figur 5.3: Overordnet flytdiagram.

Flytskjemaet tilhørende blokken vises i figur 5.3. Som dette viser, sjekkes det om flagget tilhørende den eksterne adc er satt til adc sekvens ferdig. Dersom dette er satt til denne tilstanden indikerer dette at en ny måling av samtlige kanaler er utført. Blokken vil deretter korrigere de målte magnetfelt ved hjelp av metoden *korrigerSignal()*, for så å sette flagg som indikerer at nye data er klar til overføring.

For å være i stand til å pause de kontinuerlige magnetfetmålingene er det lagt til en sjekk der det sjekkes om de kontinuerlige magnetfeltmålingene skal fortsette eller ta en pause. Denne muligheten anses nyttig der andre metoder skal behandle målingene.

Synkronisere SysTick

Ett komplett målesystem består av flere sensorer som kommuniserer med en enhet som analyserer målt magnetfeltdata. For at målingene til de ulike sensorene skal være synkroniserte er det valgt å legge til mulighet for å korrigere tellerverdien til SysTick.



Figur 5.4: Overordnet flytdiagram.

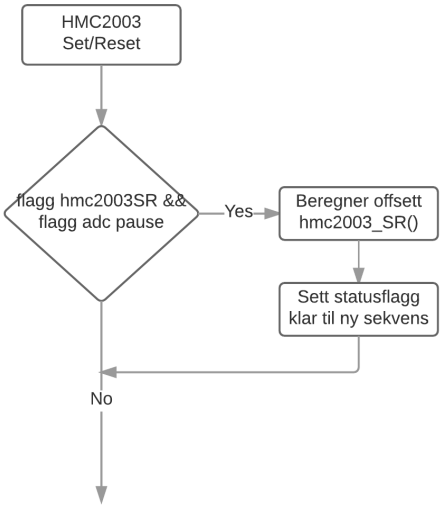
Flytskema i figur 5.4 viser hvordan statusflagget *CLK_SYNC* fanges opp i hovedprogrammet. Kilden som setter dette statusflagget er avbruddsrutinen *EXTI4_15_IRQHandler()*. Denne rutinen setter flagget etter et bestemt antall pulser er registrert på pinne PA15. Fra programlisting 5.1 ser vi hvordan algoritmen beregner korrigerende tiltak for å øke eller redusere telleverdien til SysTick.

Listing 5.1: Synkronisering av SysTick

```
1  void clkSync( void ){
2      uint8_t teller = 0;
3      int32_t avvik = 0;
4
5      /* Bergner avviket mellom målte tider og ønsket tid
6       * avvik = SUM[ (Tid(n+1) - Tid(n)) - forventetTid ]
7       * */
8      for( teller = 0; teller < ELEMENTER_I_CLKSYNC - 1 ; teller++ ){
9          avvik += ( clkSyncTid.tid[teller + 1] - clkSyncTid.tid[teller] ) - ...
               CLKSYNC_FORVENTET_TID;
10     }
11
12     /* Korrigerer systick */
13     if( avvik < (- CLKSYNC_AVVIK) ){
14         clkSyncTid.systickLoad++;
15         SysTick->LOAD = clkSyncTid.systickLoad;
16     }
17     else if( avvik > CLKSYNC_AVVIK ){
18         clkSyncTid.systickLoad--;
19         SysTick->LOAD = clkSyncTid.systickLoad;
20     }
21 }
```

HMC2003 Set/Reset

Det ble i delkapittel 4.1.1 klart at sensoren HMC2003 har behov for en periodisk set/reset av de magnetoresistive sensorene. Det ble ikke gitt noen klare retningslinjer på hvor ofte denne funksjonen bør kjøres og det er valgt at 10 minutter er et passende intervall. Samtidig som sensorens magnetoresistive elementer blir resatt for å opprettholde høy presisjon, kan målingene fra denne prosessen benyttes for å oppdatere sensorens offsett.

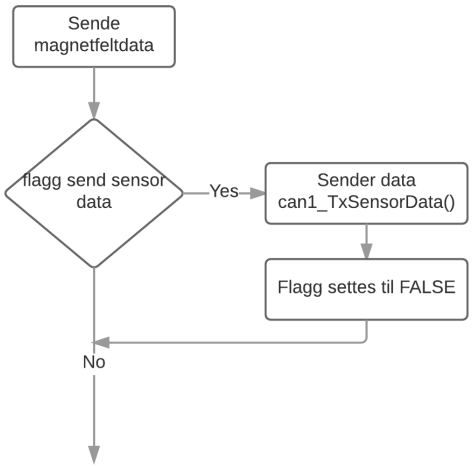


Figur 5.5: Overordnet flytdiagram.

Flytskjemaet i figur 5.5 viser hvordan virkemåte til blokken HMC2003Set/Reset er programmert. Blokken starter med å sjekke om flagget *hmc2003SR* er satt og den eksterne adc er i pauset fra magnetfeltmålinger. Dersom disse kravene er tilfredsstillt vil en ny offsettberegning starte. Etter offsettberegning vil statusflagget tilhørende den eksterne adc settes til ny sekvens.

Sende magnetfeltdata

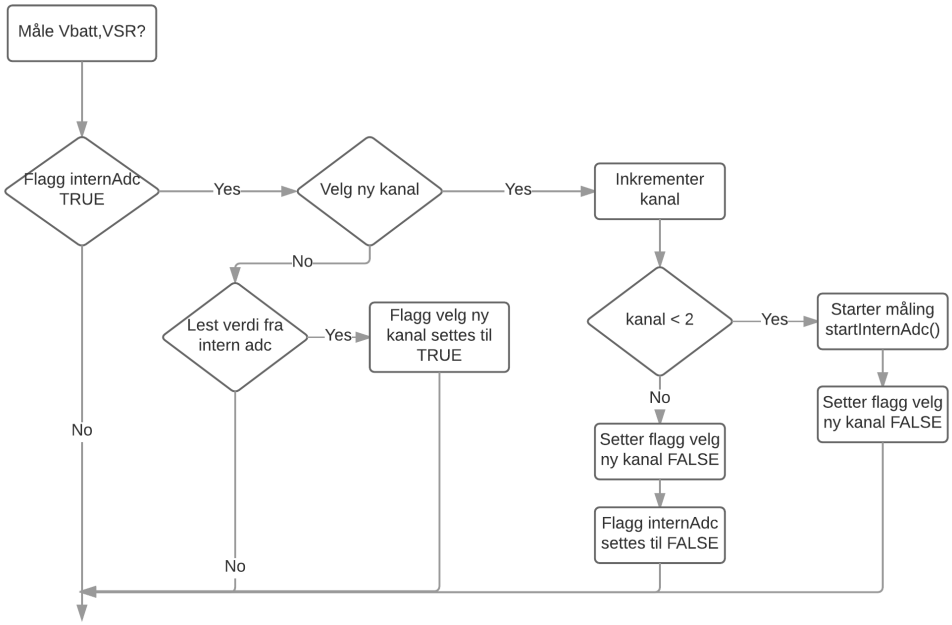
Blokken *sende magnetfeltdata* sørger for at målte magnetfeltdata sendes over CAN-buss. Som flytskjemaet i figur 5.6 viser, sendes sensordata så snart som flagget *send sensordata* settes. Etter data er sendt tilbakestilles flagget.



Figur 5.6: Overordnet flytdiagram.

Måle V_{Batt} og +VSR

I programvaren benyttes den interne adc for å måle den faktiske batterispenning. Med denne målingen gjøres programvaren i stand til å reagere dersom batterispenningen blir for lav. Den opptransformerte spenningen +VSR måles også av den interne adc og med denne målingen har programvaren mulighet til å gjøre korrigerende tiltak dersom spenningen blir for høy.



Figur 5.7: Overordnet flytdiagram.

Fra figur 5.7 ser vi flytskjemaet til hvordan måleprosessen er programmert. Som flytskjemaet viser startes måleprosessen ved at tilhørende statusflagg settes.

Listing 5.2: Intern adc målinger

```
1  /* Leser målingene som utføres av den interne adc */
2     if( statusFlagg.internAdc == FLAGG_ADCSEKVENS_Start){
3        if( internADC.velgNesteKanal == TRUE ){
4           can1_TxData(CAN_DEFAULT_ADR, 1,&internADC.kanalIdx );
5           if( internADC.kanalIdx < 2 ){
6              startInternADC();
7              internADC.velgNesteKanal = FALSE;
8           }
9           else{
10             internADC.velgNesteKanal = FALSE;
11             statusFlagg.internAdc = FLAGG_ADCSEKVENS_Ferdig;
12          }
13       }
14       if( ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC ) == SET ){
15          internADC.adcData[ internADC.kanalIdx ].verdi = ...
               ADC_GetConversionValue(ADC1);
16          internADC.velgNesteKanal = TRUE;
17          internADC.kanalIdx++;
18       }
19    }
```

5.1.2 WatchDog

For å oppnå en mer robust programvare er det valgt å implementere en modul som i datablad kalles *IWDG*. Denne modulen er en uavhengig timer som teller ned fra en bestemt verdi. Klokkefrekvensen til watchdog telleren forsynes fra den uavhengige RC-oscillatoren *(LSI)* som danner en klokkefrekvens på 40kHz. Denne deles ned til en laverer frekvens og sammen med størrelsen på tellerverdien bestemmes hvor lang tid det tar før telleren når 0. Dersom telleren når 0, indikerer dette at programmet henger og denne modulen vil da restarte programmet. For å unngå at telleren når 0 og dermed programreset, må telleren innen fornuftig tid tilbakestille telleren til opprinnelig verdi. Funksjonen fanger dermed opp programheng og uendelige løkker. I dette programmet er det valgt å la 4 sekunder være lengste tid uten kvittering av *watchdog*. Det oppnås da å kunne utføre flere ressurskrevende rutiner etterhverandre, uten at programmet resetter seg.

Basert på informasjonen ovenfor kan ønsket tellerverdi beregnes til:

$$\begin{aligned} WatchDog_{telleverdi} &= \frac{f_{LSI}}{divisjon} \cdot tid \\ &= \frac{40kHz}{16} \cdot 4s \\ &= 10000 \\ WatchDog_{telleverdi} &= 0x3E8 \end{aligned}$$

Listing 5.3: Initiering av watchdog

```
1 void initWatchDog( unsigned char prescaler ){
2     IWDG->KR = 0x5555;
3     IWDG->RLR = 0x3E8; /* Teller ned fra denne verdien */
4     IWDG->PR |= prescaler;
5     IWDG->KR = 0xCCCC;
6     IWDG->KR = 0xAAAA;
7 }
```

Listing 5.3 viser hvordan watchdog konfigureres. For å resette telleren i while løkka skriver vi 0xAAAA til watchdog register IWDG− >KR.

5.1.3 SysTick

SysTick modulen er en konfigurerbar teller som teller ned fra en verdi og når den kommer til 0 genererer den programavbrudd. I dette programmet er det naturlig å la ønsket samplefrekvens bestemme verdien som det skal telles ned fra. Verdien bestemmes ved:

$$SysTick_{teller} = \frac{f_{clk}}{f_{avbrudd}} \tag{5.1}$$

Vi vil først se på detaljene omkring sampling av magnetfeltet i delkapittel 5.1.4, men det synes naturlig å beregne verdien til systick her. Vi vil se i delkapittel 5.1.4 at det er ønskelig å sample magnetfeltet med 2kHz. Dette innebærer at den eksterne adc MAX11047 krever at inngangen *CONVST* pulses med en negativ flanke hver gang samplingen skal startes. Basert på ligning 5.1 og den gitte informasjon kan nå SysTick telleren beregnes.

$$\begin{aligned} SysTick_{teller} &= \frac{48MHz}{4kHz} \\ SysTick_{teller} &= 12000 \end{aligned} \tag{5.2}$$

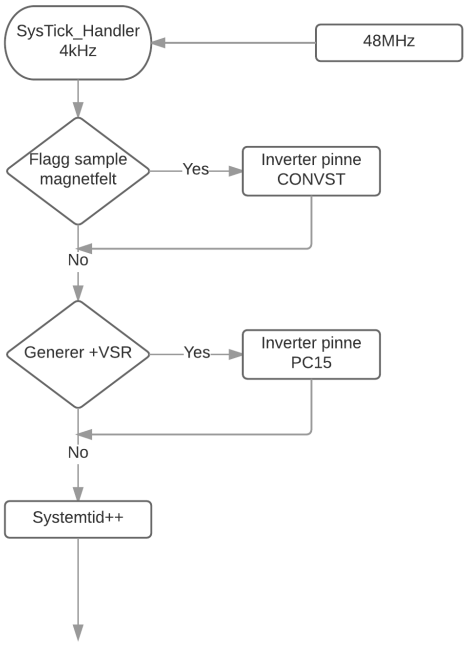
Beregnet verdi tilordnes systick teller ved hjelp av metoden gitt i listing 5.4. I denne metoden er $load = SysTick_{teller} = 12000$.

Listing 5.4: Initiere SysTick

```
1 void initSysTick( uint16_t load ) {
2     SysTick->CTRL = 0;
3     SysTick->LOAD = load;
4     SysTick->VAL = 0;
5     SysTick->CTRL = 0x07 ;
6     NVIC_SetPriority(SysTick_IRQn, 1);
```

```
7 }
```

Det er nå dannet en pålitelig tidsbase som kan benyttes til tidskritiske funksjoner samt holde orden på systemtiden.



Figur 5.8: Flytskjema SysTick avbrudd.

Fra flytskjema i figur 5.8 kan det legges merke til at SysTick kun har 3 oppgaver. Disse oppgavene deles inn i følgende grupper :

Flagg Sample magnetfelt - Dersom flagg *FLAGG_ADCSEKVENS_Start* er satt skal den eksterne adc konvertere de analoge signalene fortløpende. Som tidsdiagramet til den eksterne adc viser i figur 4.26, krever omformeren en fallende puls på inngang *CONVST* hver gang en ny konvertering skal starte. Dette er løst ved å invertere pinnen PA3 hver gang SysTick kalles. Det oppnås da å starte en ny konvertering hver andre gang SysTick kalles.

Flagg generer +VSR - Dersom flagg *pulseVSR* er satt, skal spenningen som benyttes i forbindelse med sensor set/reset opptransformeres. Dette er her løst ved å pulse PC15 med frekvensen som dannes ved å invertere PC15 hver gang SysTick kjøres.

Systemtid - Variabelen systemTid inkrementeres hver gang SysTick kjøres. Dette danner grunnlaget for systemets interne klokke og muligjør tidsstyring i programmet.

5.1.4 Magnetfeltmålinger

Signalet som denne programvaren er tiltenkt å måle, er et signal som hovedsaklig inneholder en 50Hz komponent. Frekvenskomponenter over dette kan i stor grad anses som påtrykt støy fra ulike kilder og inneholder ingen nyttig informasjon.

I delkapittel 4.1.2 ble det besluttet at minste samplefrekvens som hver kanal skal samples med er 642Hz og de analoge filterbanker ble tilpasset denne frekvens. Som nevnt ovenfor er vi kun interessert i å måle 50Hz komponenten og ønsker dermed å digitalfiltrere signalet for da å fjerne påtryktstøy og kvantiseringsstøy fra adc. Selve filtreringen av målingene ønskes utført på et system med støtte for flyttallsberegning og unngår med dette problemstillinger som flyttall i et heltallssystem ville introdusert. De ulike målingene kan dermed sendes som rådata over CAN-buss til en enhet som utfører den digitale signalbehandling. For å ha tilgjengelig et tilstrekkelig antall sampler for filter og senere FFT beregninger, er det ønskelig å oversample signalene. Det er eksisterer mange ulike filtreringsalgoritmer som kan benyttes, men valget av algoritme vil det først bli tatt stilling til når grundigere tester foreligger. På det nåværende tidspunkt anses en samplefrekvens på 2kHz som tilstrekkelig og samtidig tilpasset systemets interne klokke.

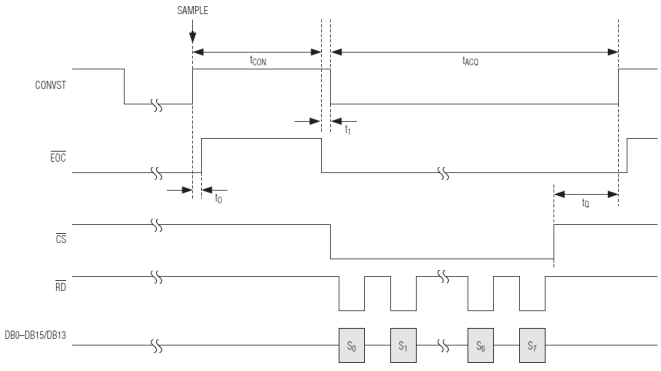
For å opprettholde høy presisjon i den valgte kresløsningen ble det klart i delkapittel 4.1.2 at det er nødvendig å beregne den faktiske forsterkning tilhørende hver instrumentforsterker. Hvordan dette løses forklares i underkapittel 5.1.4.

Fra datablad [11] kom det frem at sensoren har behov for periodisk resett av de magnetoresistive sensorene. En fremgangsmåte for dette ble skissert i delkapittel 4.1.1. Hvordan dette er løst forklares i underkapittel 5.1.4.

Måleprosess

Med måleprosessen menes de metoder som kreves for at programvaren skal kunne sende oppdaterte måleverdier til bestemte tider. De deler av prosessen som involverer direkte kommunikasjon med adc er realisert ved hjelp av lette avbruddsbaserte metoder som kommuniserer ved å sette statusflagg. Metoder som utfører mer ressurskrevende algoritmer kjøres fra hovedprogram når deres respektive flagg settes

Måleprosessen startes og stoppes ved hjelp av en definert can melding. Når programvaren mottar denne meldingen første gang, vil flagget tilhørende adc bli satt til *FLAGG_ADCSEKVENS_Start*.



Figur 5.9: Tidsdiagram for leseoperasjon [21].

Virkemåte og tidsdiagram 4.26 gjenntas her i figur 5.9. Fra dette tidsdiagrammet kan det identifiseres hvordan kretsen MAX11047 forventer at en konvertering utføres.

For å starte en ny paralellkonvertering av de analoge kanalene, krever adc kretsen at pinnen *COVST* går først fra høy til lav og deretter til høy igjen. Hvor selve konverteringen av signalene foregår ved sistnevnte flanke. For å oppnå en pålitelig samplerate er det valgt å implementere denne funksjonen i SysTick.

Listing 5.5: Utdrag fra SysTick

```
1 void SysTick_Handler(void) {
2    /* Starter og stopper ekstern ADC */
3    if( statusFlagg.adc == FLAGG_ADCSEKVENS_Start){
4       GPIOA->ODR ^= ADC_CONVST_Pin;
5    }
```

Når kretsen har konvertert alle kanalene indikeres dette ved at pinnen $\overline{EOC}$ går fra lav til høy. I programvaren vil et eksternt pinneavbrudd fange opp denne hendelsen. Avbruddet vil så starte metoden *lesEksternADC()*, som leser de dekodede verdiene som sendes over det paralelle grensesnittet. Fra figur 5.9 kan det legges merke til at pinnen $\overline{RD}$ krever en transisjon fra høy til lav for å gi ut verdier tilhørende neste kanal.

Listing 5.6: Hent konverterte verdier fra MAX11047

```
1 void lesEksternADC( void ){
2    uint8_t ch;
3    uint8_t kanal;
4    GPIOA->BRR = ADC_CS_Pin; /* CS = Aktiv */
5    for (ch = 0; ch < ADC_Kanaler; ch++) {
6       GPIOA->BRR = ADC_RD_Pin; /* RD = Lav */
7       kanal = kanalKorrigering( ch );
8       ekstADC[ kanal ].lesteADC_Data = GPIOB->IDR;
9       GPIOA->BSRR = ADC_RD_Pin; /* RD = Høy */
10   }
11   GPIOA->BSRR = ADC_RD_Pin; /* RD = Høy */
12   GPIOA->BSRR = ADC_CS_Pin; /* CS = Inaktiv */
13   statusFlagg.adc = FLAGG_ADCSEKVENS_Ferdig;
14 }
```

Etter en konvertering settes flagg som indikerer at en målesekvens er utført. Flagget fanges så opp av den uendelige løkka i hovedprogrammet, som starter metoden *korrigerSignal()*. Etter denne metoden er fullført settes flagg som indikerer at nye data skal sendes. Tilslutt sjekkes det om en ny sekvens skal startes.

Listing 5.7: Utdrag fra den uendelig løkka i hovedprogrammet

```
1 if( statusFlagg.adc == FLAGG_ADCSEKVENS_Ferdig ){
2          korrigerSignal();
3          statusFlagg.can = FLAGG_CAN_TxSensorData;
```

```
4
5           /* Sjekker om ADC skal starte ny sekvens */
6           if( statusFlagg.adcModus != FLAGG_ADCMODUS_Pause && ...
                statusFlagg.hmc2003SR != FLAGG_HMC2003_SetReset){ statusFlagg.adc ...
                = FLAGG_ADCSEKVENS_Start; }
7           else{statusFlagg.adc = FLAGG_ADCSEKVENS_Pause; }
8       }
```

Metoden *korrigerSignal()* beregner det faktiske signal basert på ligning 5.3. I implementasjonen benyttes de kalibrerte verdier for kanalforsterkningen og her utnyttes brøken teller og nevner slik at heltallsdivisjon kan benyttes.

$$V_s(k,n) = \frac{V_{ADC}(k,n) \cdot G_{Nevner}(k)}{G_{Teller}(k)} + V_{DAC}(k,n-1) \pm V_{offsett}(k) \tag{5.3}$$

Hvor **n** indikerer iterasjon og **k** indikerer kanal.

Metoden vises i sin helhet i listing 5.8 og her kan det også legges merke til at etter korrigering av signalet, oppdateres også verdiene til dac.

Listing 5.8: Korriger signal

```
1  void korrigerSignal ( void ){
2     uint8_t teller = 0;
3     uint32_t dacVerdi = 0;
4
5     for( teller = 0 ; teller < ( ADC_Kanaler - 1 ) ; teller++){
6        dacVerdi = ((uint32_t)ekstADC[ teller ].dac<<15)/625;
7
8        ekstADC[ teller ].korrigert.verdi = (uint16_t)((( (uint32_t)ekstADC[ ...
             teller ].lesteADC_Data )* (uint32_t)ekstADC[ teller ].G_nevner )
9              / (uint32_t)ekstADC[ teller ].G_teller) + (uint32_t)dacVerdi;
10
11       /* Oppdaterer dac verdi */
12       dacVerdi = ((5625*(uint32_t)ekstADC[ teller ].korrigert.verdi)/10)>>15;
13       ekstADC[ teller ].dac = (uint16_t)(dacVerdi);
14       skrivEksternDAC(DAC_CMD_CODEn_LOADn|teller, dacVerdi );
15       /* Legger til/tekker fra offsett */
16       if(ekstADC[teller].fortegn == OFFSETT_FORTEGN_POS){
17          ekstADC[ teller ].korrigert.verdi +=ekstADC[ teller ].offsett;
18       }
19       else{
20          ekstADC[ teller ].korrigert.verdi -=ekstADC[ teller ].offsett;
21       }
22    }
23 }
```

Beregne forsterkning

Metoden som beregner den faktiske forsterkning tilhørende hver intrumentforsterker anses for stor til å inkludere i helhet og det henvises til vedlegg C.1 for å studere denne i detalj.

Målet til algoritmen er å beregne den faktiske forsterkning, hvor denne beregningen baseres på ligning 4.13. For å slippe å bla tilbake kan det passe å gjenta ligningen her:

$$G_{Faktisk} = \frac{\Delta V_{ADC}}{\Delta \left(V_{Ref} - V_{DAC} \right)} \tag{5.4}$$

For å beregne den faktiske forstekning ved hjelp av ligning 5.4, må det måles endringen i utspenning,V_o, som følge av endret DAC-spenning, V_{DAC}. I metoden har jeg valgt å definere endringen δV_{DAC} til:

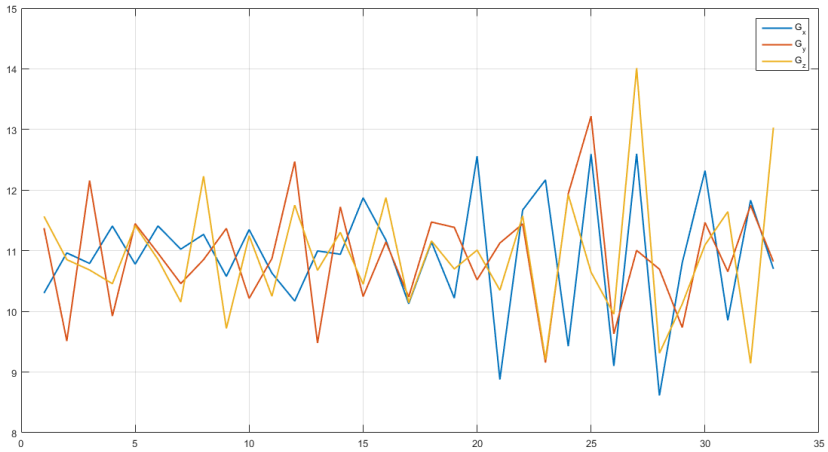
$$\delta V_{DAC} \equiv 1bit \tag{5.5}$$

I programvaren er det valgt at algoritmen skal benytte $N = 32$ målinger for å danne grunnlag for beregningene. For å fjerne tilfeldig støy, er det valgt at hver måling er gjennomsnittet av $K = 128$ målinger.

$$V_{DAC}(n) = V_{Ref} - n \cdot \delta V_{DAC}$$

$$V_{ADC}(n) = \sum_{k=1}^{K} V_{ADC}(k)$$

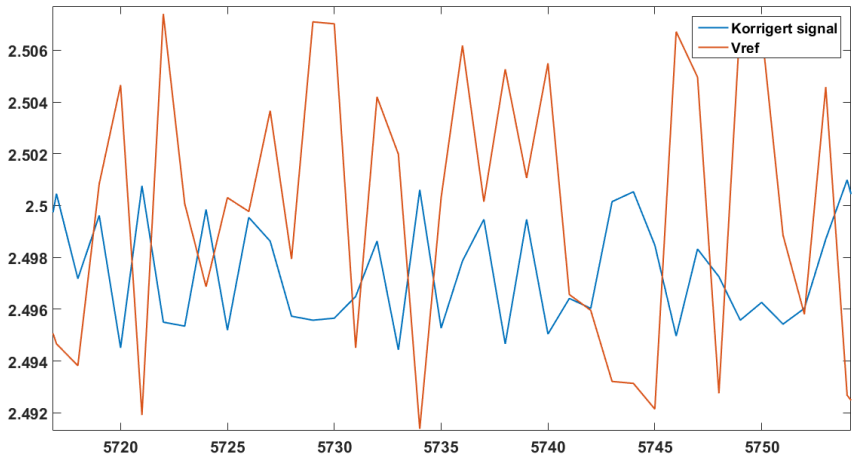
Så snart en måling av $V_{ADC}(n)$ er fullført, sendes denne målingen over CAN-buss slik at faktisk forsterkning kan beregnes ved hjelp av verktøy med støtte for flyttallsberegning. I denne oppgaven er det valgt å legge denne beregningen til en datamaskin og denne kan basert på målinger beregne korrekt forsterkning. Når forsterkning er beregnet sendes denne informasjonen via CAN-buss til sensor.



Figur 5.10: Målt kanalforsterkning.

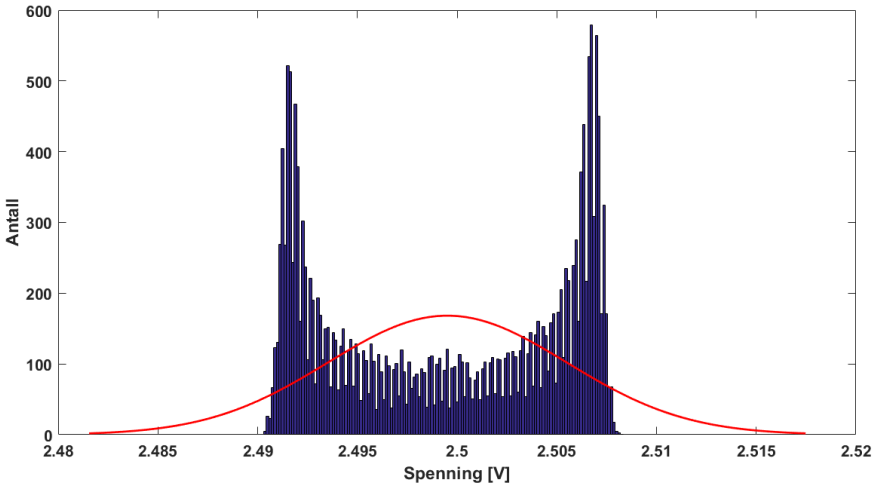
Figur 5.10 viser målt kanalforstekning ved å benytte metode nevnt ovenfor. Fra figuren er det tydelig at forstekningen er langt mindre konstant en opprinnelig forventet og grundigere analyser av variasjonene må dermed utføres.

Ved å studere ligning 5.4 er det tydelig at dersom referansen V_{ref} inneholder unøyaktigheter, så vil disse forplantes inn i beregning av forsterkningen.



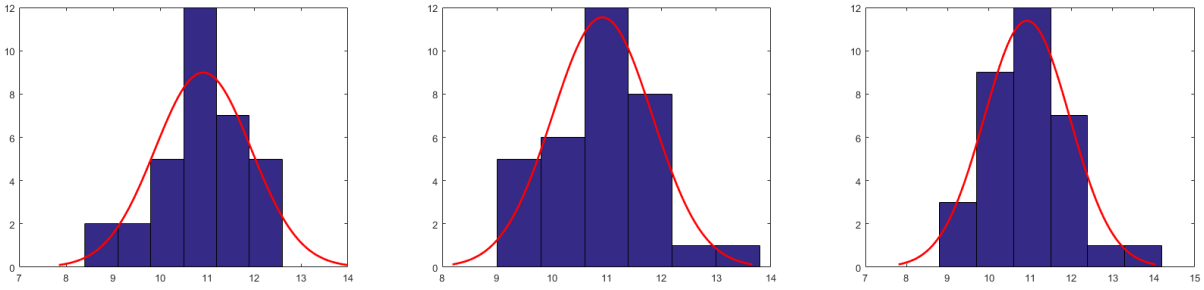
Figur 5.11: Målt V_{Ref} og beregnet signalverdi.

Figur 5.11 viser tydelig at spenningsreferansen fra sensor HMC2003 ikke er så stabil som ønsket. For å kunne si noe mer om referansen, er det naturlig å se på fordelingen til denne.



Figur 5.12: Fordeling av målt V_{Ref}.

Ved å studere fordelingen til den målte referansen i figur 5.12, er det tydelig at denne varierer betraktelig. Variasjonen i referansen vil dermed i høy grad påvirke stabiliteten til beregningen av forsterkningen i ligning 5.4. På tross av den vesentlige ustabilitet i referansen, skal det forsøkes å komme frem til en måte å beregne kanalforsterkningen på.



G_x G_y G_z

Figur 5.13: Fordeling av forsterkning.
(Forsterkning vises i horisontal akse og antall vises i vertikal akse.)

Av figur 5.13 er det tydelig at variasjonene i forstekningene er normalfordelt. Fra [15] er det gitt at sentralgrenseteoremet kan benyttes dersom datasett har størrelse $N \geq 20$, med uavhengige variabler med samme sannsynlighetsfordeling med forventning μ og standardavvik σ . Dette leder til følgende resultat:

$$\overline{X} = \frac{1}{n}(X_1 + X_2 + \cdots + X_n) \approx Normal\left(\mu, \frac{\sigma}{\sqrt{n}}\right) \tag{5.6}$$

Sentralgrenseteoremet i ligning 5.6 gir oss et verktøy til å beregne den forventede forsterkning til hver kanal.

Før sentralgrenseteoremet kan benyttes må den faktiske forsterkningen til hver måling beregnes. Dette kan beregnes ved hvelp av ligning 5.7.

$$G(k,n) = \frac{1}{K \cdot (V_{DAC}(n) - V_{DAC}(n-1)} \cdot \big(V_{ADC}(k,n) - V_{ADC}(k,n-1)\big) \tag{5.7}$$

Hvor **n** indikerer måling og **k** indikerer kanal.

Nå kan sentralgrenseteoremet benyttes for å beregne den forventede forsterkning tilhørende hver enkelt kanal.

$$E[G(k)] = \mu_G(k) = \frac{1}{N}\sum_{n=1}^{N} G(k,n) \tag{5.8}$$

Etter beregning er det ønskelig å benytte den beregnede forsterkning som faktorer teller og nevner. Med dette oppnås det å kunne benytte heltallsdivisjon i de beregninger der disse faktorene inngår.

Beregne sensoroffsett

I delkapittel 4.1.1 ble det presentert en teoretisk algoritme for å beregne faktisk offset til hver kanal. Det skal her forklares hvordan dette er løst i programvaren og hvilke problemstillinger som finnes. Selve kildekoden anses for stor til å presentere i sin helhet her og det henvises til vedlegg C.1 for å studere denne i detalj.

Metoden har som formål å beregne en korrigerende faktor som legges til eller trekkes i fra den beregnede måleverdien i ligning 5.3. Dette er en metode som det i [13], anbefales å utføre ved gjevne mellomrom.

Før en SET/RESET prosess av sensoren kan utføres, må spenningen $+VSR$ opptransformeres til et nivå som er i stand til å danne tilstrekkelig strøm gjennom sensoren. Fra [13] er det gitt at motstanden i senorens *strapper* anslås til å være mellom 4.5..6Ω og for en fullstendig resett kreves en strøm $4A < I < 5A$. Ved å sette denne informasjonen inn i Ohms lov beregnes nødvendig nivå på spenningen $+VSR$:

$$\begin{aligned} VSR_{Maksimal} &= 6\Omega \cdot 5A = 30V \\ VSR_{Typisk} &= 4.5\Omega \cdot 4.5A \approx 20V \\ VSR_{Faktisk} &= 5.5\Omega \cdot 4.5A \approx 25V \end{aligned}$$

Fra beregninger ovenfor ser vi at det er behov for en spenning mellom 20V og 25V for oppnå påkrevd strøm gjennom kretsen.

Når denne spenningen skal opptransformeres gjøres det ved å sette ett flagg. Dette flagget fanges opp av SysTick avbruddet og for hver gang dette startes, inverteres den tilhørende pinnen, *V_SR*.

Kretsen som transformere spenningen vises i figur 4.37. SysTick vil fortsette denne prosessen frem til flagget deaktiveres som følge av enten spenningen er på ønskelig nivå, eller tidsavbrudd som følge av for langsom oppladning av kondensator.

Når spenningen $+VSR$ er på ønsket nivå, kan prosessen sensor SET/RESET startes. Metoden starter med å sette pinnen PC14 med referansenavn *HMC2003_Set* til høy. Det vil da gå en strøm gjennom sensor med positiv retning. Signalet fra sensor kan da måles og det er valgt å benytte $N = 64$ målinger av sinalet. De målte verdiene lagres i tabell $V_{set}(k)$. Deretter settes pinnene PC14 til lav og PC13 til høy. Det vil da gå en strøm gjennom sensor men nå, motsatt rettet. Signalet måles fra sensor og det er valgt å også her oversample signalet med $N = 64$ målinger og lagres i tabell $V_{reset}(k)$.

Overnevnte målinger danner videre grunnlag for å beregne sensor offsett. I programvaren beregnes denne ved help av følgende ligning:

$$V_{Offsett}(k) = \frac{V_{Set}(k) - V_{Reset}(k)}{2N}$$

5.1.5 CAN-kommunikasjon

Det ble tidlig i prosjektet besluttet at kommunikasjon mellom de ulike nodene skulle baseres på CAN. Dette er en buss som gir høy støyimunitet sammen med både høy bitrate og deteksjon av kollisjon. En annen fordel med denne bussen er at meldingene inneholder informasjon omkring prioritet på buss og jeg kan dermed tildele de meldinger som anses viktigst høyest prioritet.

For at ikke mikrokontrolleren skal bruke tid på can meldinger som ikke er av interresse for programvaren, er det i STM32F072 implementert filtere. Filterene er realisert i hardware og kan ved korrekt konfigurasjon avvise de meldinger som anses å være uten nytteverdi. I et system med høy busslast vil denne funksjonen bidra til at mikrokontrolleren kan arbeide uhindret og kun motta de meldinger som er av interesse for det aktuelle program. Filter funksjonen anses nyttig i denne applikasjonen og jeg ønsker derfor å implementere denne i programvaren. Virkemåte og implementasjon blir nærmere behandlet i seksjon CAN Filterbank.

CAN-standard identifikator

Det ble i kapittel 4.1.5 besluttet å benytte CAN med 11bit standard identifikator. Dersom oppbyggning av identifikatoren velges smart kan denne benyttes for å sortere meldinger i mottaker.

I denne oppgaven er det valgt å stukturere identifikatoren slik at det tillates filtrering. Oppbyggingen er som vist bestående av 11 bit, hvor nedre bit [3:0] er reservert adresse til sender og de øvre [10:4] representerer type melding.

	MSB	9	8	7	6	5	4	3	2	1	LSB
	Type melding							Adresse			

Tabell 5.1: STDID Struktur

Konfigurasjon av CAN-modul

Metoden *initCAN()* initierer CAN-buss modulen,GPIO og nødvendige klokkesignaler slik at det tillates can kommunikasjon med 1Mbit/s. Koden er basert på eksempel fra [27] og tilpasset CMSIS som benyttes i de øvrige moduler.

Listing 5.9: Initiering av CAN buss

```
1  void initCAN( void ){
2     GPIO_InitTypeDef  GPIO_InitStructure;
3     NVIC_InitTypeDef NVIC_InitStructure;
4     CAN_InitTypeDef CAN_InitStructure;
5     CAN_FilterInitTypeDef CAN_FilterInitStructure;
6
7     RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN1, ENABLE);
8     RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);
9
10    GPIO_PinAFConfig(GPIOA, GPIO_PinSource11, GPIO_AF_4 );
11    GPIO_PinAFConfig(GPIOA, GPIO_PinSource12, GPIO_AF_4 );
12
13    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 | GPIO_Pin_12 ;
14    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
15    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
16    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
17
18    GPIO_Init(GPIOA, &GPIO_InitStructure);
19
20    /* CAN Kommunikasjon*/
21    CAN_DeInit( CAN1);
22    CAN_StructInit(&CAN_InitStructure);
23
24    CAN_InitStructure.CAN_TTCM = DISABLE;
25    CAN_InitStructure.CAN_ABOM = DISABLE;
26    CAN_InitStructure.CAN_AWUM = DISABLE;
27    CAN_InitStructure.CAN_NART = DISABLE;
28    CAN_InitStructure.CAN_RFLM = DISABLE;
29    CAN_InitStructure.CAN_TXFP = DISABLE;
30    CAN_InitStructure.CAN_Mode = CAN_Mode_Normal;
31    CAN_InitStructure.CAN_SJW = CAN_SJW_2tq;
32    CAN_InitStructure.CAN_BS1 = CAN_BS1_4tq;
33    CAN_InitStructure.CAN_BS2 = CAN_BS2_3tq;
34    CAN_InitStructure.CAN_Prescaler = 6;
35    CAN_Init(CAN1, &CAN_InitStructure);
36 }
```

CAN-filterbank

Som nevnt innledningsvis kan vi ha stor nytte av å implementere can filter som filtrerer bort de meldingene som ikke er av interesse. Siden dette anses som en nyttig mulighet, samtidig som den er relativt raskt beskrevet i datablad [27], skal det her gis en grundigere forklaring av hvordan denne modulen konfigureres.

CAN modulen implementert i STM32F072 har tilgjengelig 28 konfigurerbare filterbanker, hvor hver filterbank kan konfigureres ulikt både i bit bredde og filter type.

Bit bredden til en filterbank kan konfigureres som:

1) - Ett 32bit filter med mulige filtrerings kriterier STDID[10:0],EXTID[17:0] og IDR,RTR bit.

2) - To 16 bit filtere med mulige filtrerings kriterier STDID[10:0],EXTID[17:15] og IDR,RTR bit.

Vi har videre mulighet til å spesifisere to ulike filter typer:

Mask Mode - Med dette valget kan det spesifiseres en filtermaske som gir bit som må stemme (1) og bit som ikke har betydning (0) for at filteret skal akseptere eller avvise meldingen.

Identifier List Mode - Med denne typen filter kan vi detalj spesifisere hvilke meldinger vi ønsker å motta. Kun de meldinger som er spesifisert per filter vil aksepteres. Siden det ikke benyttes maske, kan antallet meldinger økes til det dobbelte per filter.

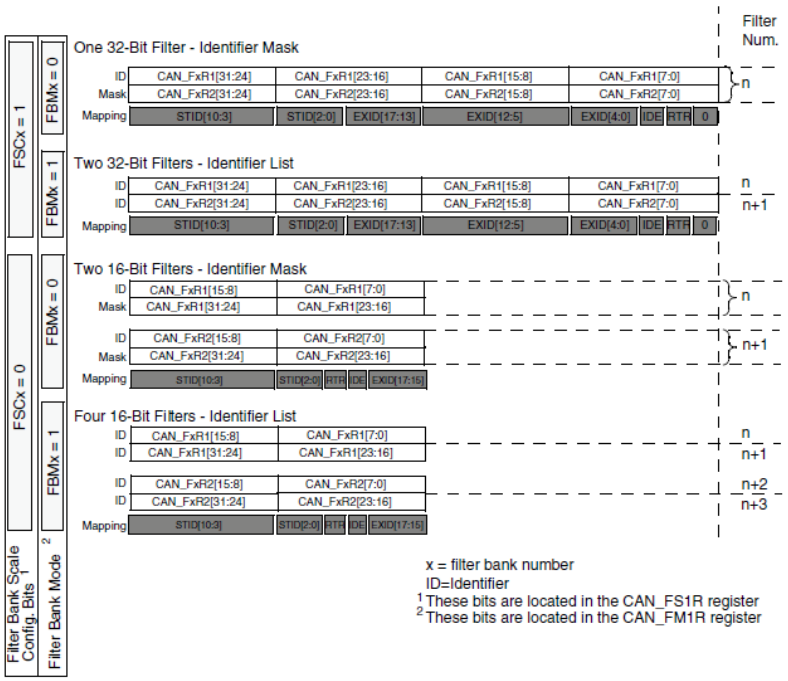
I denne oppgaven er det valgt å benytte ett filter som er konfigurert til 16 bit og type *Mask Mode*. Med denne konfigurasjonen sammen med struktur av meldinger oppnår jeg at sensor kun mottar meldinger fra en enkelt id. Dette vises i tabell 5.2. Ved å studere hvordan det er valgt å

	MSB	9	8	7	6	5	4	3	2	1	LSB
Filter ID								0	0	0	1
Filter Maske								1	1	1	1

Tabell 5.2: Filtrering av meldinger

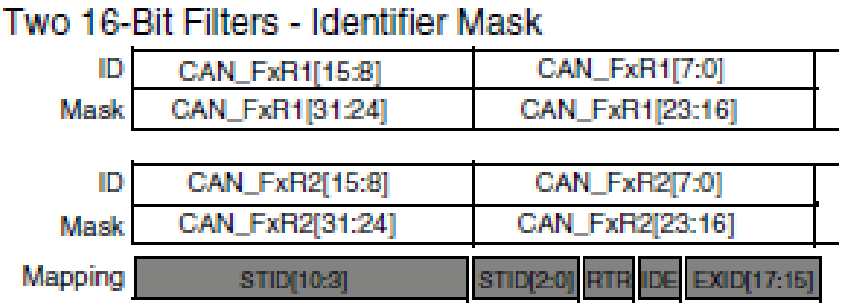
definere identifikator og sammenlikne dette med filterkonfigurasjonen i tabell 5.2 kan det legges merke til at hver sensor vil kunne motta alle meldinger som sendes med identifikator 0x1. Andre identifikatorer oppfyller ikke filterkriteriet gitt av masken og vil dermed avvises av filter.

For at filteret skal virke som forventet må de riktige registere konfigureres på en korrekt måte.



Figur 5.14: Konfigurasjon av filterbank [27].

For at filteret skal virke som forventet må tilhørende registere konfigureres korrekt. I figur 5.14 vises en fullstendig oversikt over hvordan modulen skal konfigureres og de ulike valg som finnes.



Figur 5.15: Konfigurasjon av filterbank [27].

Som nevnt ovenfor ønsker jeg å benytte can filter med 16 bit bredde og av typen *Mask Mode*. I figur 5.15 vises de aktuelle register for denne konfigurasjon. Fra figuren bør det legges merke til den spesielle inndeling av de to 32bit registere. Hvordan denne konfigurering utføres vises i kildekoden i programlisting 5.10. Her konfigureres CAN filterbank 0 til kun å akseptere meldinger fra en bestemt identifikator.

Listing 5.10: Konfigurasjon av CAN Filter

```
1    CAN_FilterInitStructure.CAN_FilterNumber = 0;
2    CAN_FilterInitStructure.CAN_FilterMode = CAN_FilterMode_IdMask;
```

```
3     CAN_FilterInitStructure.CAN_FilterScale = CAN_FilterScale_16bit;
4     CAN_FilterInitStructure.CAN_FilterIdHigh = (uint16_t)CAN_MASTER_ID<<5;
5     CAN_FilterInitStructure.CAN_FilterIdLow = (uint16_t)CAN_MASTER_ID<<5;
6     CAN_FilterInitStructure.CAN_FilterMaskIdHigh = (uint16_t)CAN_FILTER_MASKE<<5;
7     CAN_FilterInitStructure.CAN_FilterMaskIdLow = (uint16_t)CAN_FILTER_MASKE<<5;
8     CAN_FilterInitStructure.CAN_FilterFIFOAssignment = CAN_FIFO0;
9     CAN_FilterInitStructure.CAN_FilterActivation = ENABLE;
10
11    CAN_FilterInit(&CAN_FilterInitStructure);
```

Sende CAN-melding

For å sende data kan det i kildekoden programmert ulike metoder som er tilpasset de parameter som skal sendes. Vi skal nå se på hvordan data blir sendt og den mest generelle metoden velges.

Listing 5.11: Sende CAN melding

```
1  void can1_TxData( unsigned int msgId , unsigned char lengde, unsigned char ...
       *data ){
2     int i = 0;
3
4     can1TxMsg.StdId = msgId;
5     can1TxMsg.ExtId = 0x0;
6     can1TxMsg.IDE = CAN_ID_STD;
7     can1TxMsg.RTR = CAN_RTR_DATA;
8     can1TxMsg.DLC = lengde;
9
10    for( ; i < lengde ; i++ ){
11       can1TxMsg.Data[ i ] = *(i + data );
12    }
13    CAN_Transmit(CAN1, &can1TxMsg);
14 }
```

I listing 5.11 vises en generell metode for å sende en melding på CAN-buss. Metoden tar inn argumentene standard indentifikator, tabell lengde og data tabell. Disse variablene blir så benyttet for å bygge opp en can melding som sendes can kontroller ved hjelp av den ferdigdefinerte metoden *CAN_Transmit(CAN1, &can1TxMsg)*

Motta CAN-melding

Når can kontrolleren mottar en can melding som aksepteres av definert can filter vil denne meldingen bli plassert i noe som i datablad omtales som postboks. Samtidig som meldingen plasseres i denne postboksen settes det et flagg som indikerer ny melding. I programvaren er det valgt å knytte et avbrudd til dette flagget og ved mottak vil dermed avbruddsmetoden

CEC_CAN_IRQHandler() kalles. Denne metoden leser meldingen fra postboksen og på bakgrunn av *idetifier* velges riktig aksjon.

Listing 5.12: Mottak av CAN melding

```
1  void CEC_CAN_IRQHandler(void){
2     CAN_Receive(CAN1, CAN_FIFO0, &can1RxMsg);
3     switch ( can1RxMsg.StdId & 0x7F0 ) {
4
5        case CAN_MSG_START_STOPP_MEAS:
6           if( statusFlagg.adc != FLAGG_ADCSEKVENS_Start ){statusFlagg.adc = ...
                FLAGG_ADCSEKVENS_Start;}
7           else{ statusFlagg.adc = FLAGG_ADCSEKVENS_Pause; }
8           break;
9
10       case CAN_MSG_CAN_ID_TILDELING:
11          /* Lagrer den oppdaterte CAN-buss adressen */
12          flashLagring.param[ _FLASH_PARAM_CAN_ID_] = (uint16_t)can1RxMsg.Data[0];
13          flashLagring.param[ _FLASH_PARAM_CAN_Initiert_ ] = TRUE;
14          lagreTilFLASH( &flashLagring );
15          break;
16
17       case CAN_MSG_GAIN_KALIBRERING:
18          /* Venter til adc sekvens er ferdig */
19          statusFlagg.adcModus = FLAGG_ADCMODUS_Pause;
20          while( statusFlagg.adc != FLAGG_ADCSEKVENS_Pause);
21          beregneGain();
22
23          statusFlagg.adcModus = FLAGG_ADCMODUS_Run;
24          break;
25
26       case CAN_MSG_START_SET_RESET:
27          statusFlagg.hmc2003SR = FLAGG_HMC2003_SetReset;
28          break;
29    }
30 }
```

5.1.6 FLASH lagring

I programvaren anses det som sterkt ønskelig å ha mulighet til å lagre parametere til et permanent minne, for så å senere kunne hente disse tilbake. Ved å studere datablad [28] blir det raskt avklart at mikrokontroller STM32F072 ikke har integrert EEprom for lagring av variabler. Selv om mikrokontrolleren ikke har integrert EEprom er det mulig å lagre parametere direkte til FLASH. I denne oppgaven er det valgt å basere permanent parameterlagring på sistnevnte metode.

Når det skal lagres parametere i FLASH er det viktig at vi er klar over at også programkoden er

lagret nettopp her. Dersom programkoden overskrives av parameterlagringen kan det få uforutsigbare konsekvenser for programmet. Det er derfor svært viktig at området som skal benyttes, plasseres med tilstrekkelig avstand fra programkoden. Det unngås dermed at de ulike områdene kommer i konflikt med hverandre.

Det er valgt å benytte en felles struktur for de parametere som inngår i FLASH lagring. Denne strukturen vises i programlisting 5.13.

Listing 5.13: FLASH skruktur

```
1 struct flash{
2     uint32_t param[ _ANT_PARAM_I_FLASH_ ];
3     uint32_t baseAdresse;
4     uint8_t oppdatert;
5     uint8_t antallParametere;
6 };
```

For å oppdatere FLASH med verdier lagret i strukturen i programlisting 5.13, benyttes metoden *lagreTilFLASH()*. Metoden sletter først den gitte siden i FLASH, for så å lagre de ulike parameterene.

Listing 5.14: Parameterlagring i FLASH

```
1 void lagreTilFLASH( struct flash *data ){
2     uint8_t teller = 0;
3     FLASH_Unlock();
4     FLASH_ClearFlag(FLASH_FLAG_BSY | FLASH_FLAG_EOP | FLASH_FLAG_PGERR );
5     FLASH_ErasePage( data->baseAdresse );
6     FLASH_Unlock();
7     for (teller = 0; teller < data->antallParametere; teller++) {
8        FLASH_ProgramWord( (data->baseAdresse + teller*4) , data->param[ ...
             teller ] );
9     }
10    FLASH_Lock();
11 }
```

For å hente de lagrede parametere fra FLASH benyttes metoden *leseFraFLASH()*. Denne metoden oppdaterer strukturen som vises i programlisting 5.13 med verdier lagret i FLASH.

Listing 5.15: Lese parametere fra FLASH

```
1 void leseFraFLASH( struct flash *data ){
2     uint8_t teller;
3
4     for( teller = 0; teller < data->antallParametere; teller++){
5        data->param[ teller ] = (uint32_t *)( data->baseAdresse + 4*teller);
6     }
```

```
7 }
```

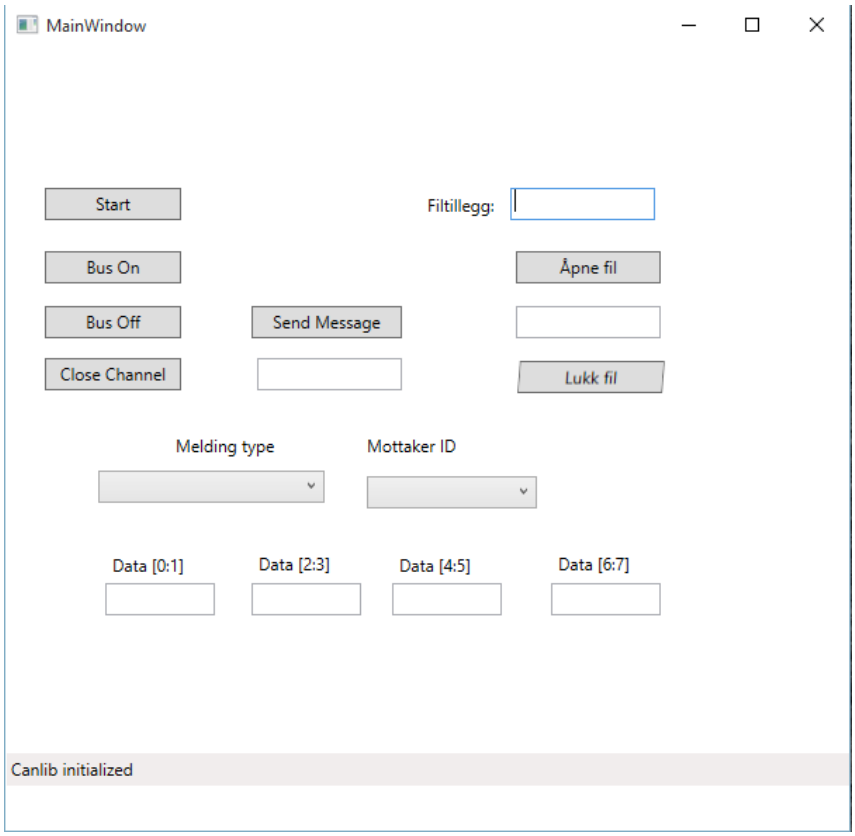
Ved omprogrammering av mikrokontrolleren er det også viktig å kun slette og reprogrammere det området som programkoden er plassert i. Dersom dette ikke tas hensyn til kan reprogrammeringen slette de lagrede parametere. I kildekoden er det derfor viktig å fange opp om FLASH er tom og da tilordne parametere med estimerte verdier, se listing 5.16.

Listing 5.16: initierer parametere fra FLASH

```
1  void initFLASHVariabler( struct flash *data ){
2     /* Legger standard eller kalibrerte verdier inn i struktur */
3     leseFraFLASH( &data );
4
5     if ( data->param[ _FLASH_PARAM_CAN_Initiert_ ] != TRUE ) {
6        data->param[ _FLASH_PARAM_CAN_Initiert_ ] = CAN_DEFAULT_ADR;
7     }
8
9     if ( data->param[ _FLASH_PARAM_GAIN_Kalibrert_ ] != TRUE ) {
10       /* Legger inn estimert forsterkning */
11       data->param[ _FLASH_PARAM_GAIN_X_t_ ] = 109;
12       data->param[ _FLASH_PARAM_GAIN_X_n_ ] = 10;
13
14       data->param[ _FLASH_PARAM_GAIN_Y_t_ ] = 109;
15       data->param[ _FLASH_PARAM_GAIN_Y_n_ ] = 10;
16
17       data->param[ _FLASH_PARAM_GAIN_Z_t_ ] = 109;
18       data->param[ _FLASH_PARAM_GAIN_Z_n_ ] = 10;
19    }
20 }
```

5.2 Grafisk brukergrensesnitt

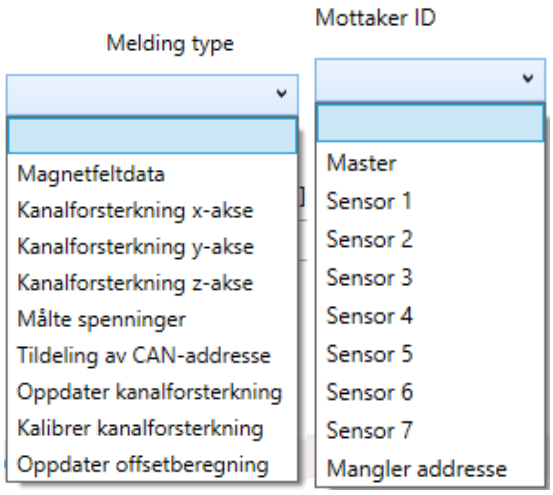
For å kunnne konfigurere sensornoder tilkoblet felles CAN-buss er det tilpasset et eksempel fra produsent av CAN til USB grensesnittet som er benyttet i denne oppgaven [2]. Kildekoden til dette programmet anses som for omfattende til å inkludere i sin helhet her. Det henvises til kapittel C.3 for å studere kildekoden som ligger bak programmet.



Figur 5.16: Grafisk brukergrensesnitt.

Dette programmet håndterer den mest nødvendige nødvendige kommunikajonen mellom sensornoder og datamaskin. Til venstre er det knapper for å initiere CAN til USB-grensesnittet. Til høyre i grensesnittet er det bokser for lagring av magnetfeltdata. Disse lagres som beskrevet i kapittel 6.1. I midten er det funksjoner for å sende meldinger til sensornoder.

For å sende en melding, må type melding og mottaker velges. Dette gjøres ved hjelp av boksene som vises i figur 5.17. Innholdet i meldingen legges inn i boksene merket *Data[0:1]*,*Data[2:3]*,*Data[4:5]* og *Data[6:7]*.



Figur 5.17: Menyvalg for sending av CAN-melding.

Kapittel 6

Eksperimenter

I dette kapittelet skal testmetoder og resultater for magnetfeltmålinger presenteres. Under kontrollerte forutsetninger, utføres innledende eksperimenter i kapittel 6.2. Etter de innledende eksperimenter, vil magnetfeltmålinger og resultater fra høyspentmast presenteres i kapittel 6.3.

6.1 Analyse av sensordata i MATLAB

Felles for alle eksperimenter er at magnetfeltmålingene analyseres i program *beregnInvarianter()*. Programmer er utviklet i MATLAB og fullstendig kode finnes vedlagt i vedlegg C.2. De mest sentrale deler av programmet vil forklares i kommende delkapittel.

6.1.1 Laste inn måledata

Programmet *beregnInvarianter()* starter med å hente inn måledata fra en tekstfil. Programmet forventer at målingene kan hentes fra følgende struktur | ID | X | Y | Z | , med første måling plassert øverst i filen. Feltene separeres med et enkelt mellomrom.

Listing 6.1: Lese inn måleverdier

```
1  function [ A_1, Z_1, A_2, Z_2, X1, Y1, Z1, X2, Y2, Z2 ] = beregnInvarianter( ...
       filnavn )
2  V = textread(filnavn);
3  for idx=1:length(V)
4     if(V(idx,1) == 3)
5      V_s1 = [V_s1; V(idx,2:4)];
6     elseif(V(idx,1) == 15)
7       V_s2 = [V_s2; V(idx,2:4)];
8     end
9  end
10
11 % Sensor 1
```

```
12 Xs1 = V_s1(200:end-10,1);
13 Ys1 = V_s1(200:end-10,2);
14 Zs1 = V_s1(200:end-10,3);
15 % Sensor 2
16 Xs2 = V_s2(200:end-10,1);
17 Ys2 = V_s2(200:end-10,2);
18 Zs2 = V_s2(200:end-10,3);
```

Når alle måleverdiene er lastet inn i tilhørende variabler, blir variablene omregnet til spenning. Dette synes å gi en mer informativ representasjon når variablene plottes. For omregning fra bitrepresentasjon til spenning, benyttes ligning 6.1.

$$V_{i,k} = I_k \cdot \frac{5.0V}{2^{16}} \tag{6.1}$$

Hvor i representerer aksen og k representerer sensor nummer.

Listing 6.2: Konvertere til spenning

```
1 %Konverterer til spenning
2 x_s1 = Xs1*(5/2^16);
3 y_s1 = Ys1*(5/2^16);
4 z_s1 = Zs1*(5/2^16);
5 x_s2 = Xs2*(5/2^16);
6 y_s2 = Ys2*(5/2^16);
7 z_s2 = Zs2*(5/2^16);
```

6.1.2 Frekvensanalyse

For å hente nødvendig informasjon fra måledataene er det nødvendig å utføre en frekvensanalyse på måledataene. Før vi ser på denne teorien og hvordan det løses i programmet, må antall målinger,L , som skal inngå i frekvensanalysen bestemmes. Fra teorien omtales dette som størrelsen på vinduet.

I en virkelig applikasjon vil størrelsen på vinduet og samplefrekvensen gitt av ligning 6.2,bestemme hvor hyppig frekvensanalyse av målte signaler oppdateres.

$$FFT_{Oppdatert} = \frac{F_s}{L}\,[Hz] \tag{6.2}$$

I dette systemet er det på forhånd bestemt at samplefrekvensen $F_s = 2kHz$, hvor stort vinduet velges, vil da alene bestemme tiden mellom hver frekvensanalyse.

Når vinduets størrelse skal defineres er det også viktig å oppnå at frekvenskomponenten med informasjon, i dette tilfellet 50Hz, fremkommer som et heltall. Det oppnås da at amplituden

til frekvenskomponenten er plassert i ett element, i stedet for spredd over flere element i den endelige frekvensanalyse. I våres tilfelle må dermed L velges slik at $L > 50$ og $\frac{Fs}{L} = \mathbb{N}_0$.

Ved å sette inn L = 400 målinger i ligning 6.2, kommer vi frem til at $\frac{Fs}{L} = \mathbb{N}_0$ og $FFT_{Oppdatert} = 5Hz$. Dette synes å oppfylle overnevnte krav og ønsker til valgt vindu.

For å konvertere de målte signalene fra tid til frekvens benyttes diskret Fourier transform. Den diskretet Fourier transform er i [5] definert til:

$$X(k) = \sum_{n=1}^{L} x(n)e^{-j2\pi kn/N} \tag{6.3}$$

Teorien bak transformen anses som kjent og vil dermed ikke forklares nærmere.

Listing 6.3: FFT av måledata

```
Fs = 2000;
T = 1/Fs;
L = 400;
f = Fs*(0:L)/L;
% Deler opp i blokker av lengde L
AntallBlokkerS1 = floor( length(x_s1)/L );
AntallBlokkerS2 = floor( length(x_s2)/L );
AntallBlokker = min(AntallBlokkerS1,AntallBlokkerS2);

% Utfører FFT beregning
for idx=1:(AntallBlokker)
   start= idx + (idx-1)*L;
   stopp = start+L;
   X1(idx,:) = fft((x_s1(start:stopp)-mean(x_s1(start:stopp))),2*L);
   X2(idx,:) = fft((x_s2(start:stopp)-mean(x_s2(start:stopp))),2*L);
   Y1(idx,:) = fft((y_s1(start:stopp)-mean(y_s1(start:stopp))),2*L);
   Y2(idx,:) = fft((y_s2(start:stopp)-mean(y_s2(start:stopp))),2*L);
   Z1(idx,:) = fft((z_s1(start:stopp)-mean(z_s1(start:stopp))),2*L);
   Z2(idx,:) = fft((z_s2(start:stopp)-mean(z_s2(start:stopp))),2*L);
end
```

I programlisting 6.3 vises matlabkoden som utfører FFT av måledata. Virkemåten til denne er først å finne antall vinduer tilgjengelig for hver sensor. Når dette er funnet, velges det minste antall vinduer.

Fra datablad [13] er det gitt at sensoren har nullpunkt i 2.5V. Det er videre gitt at eksterne magnetfelt vil medføre at sensoren endrer signalspenning proporsjonalt med påtrykt magnetfelt. Signalet fra sensor kan dermed uttrykkes som:

$$V_{Sensor} = 2.5V \pm 2V \tag{6.4}$$

Ved å studere ligning 6.4, er det tydelig at målte signaler inneholder en betydelig dc komponent. Dc komponenten inneholder her ingen informasjon og det er derfor ønskelig å trekkes fra de målte signalene. Dette utføres ved å benytte ligning 6.5.

$$x(n) = V_{Sensor}(n) - \frac{1}{N}\sum_{n=1}^{N} V_{Sensor} \tag{6.5}$$

Metoden vil videre iterere seg gjennom tilgjengelig antall vinduer og fortløpende utføre FFT beregning av disse.

6.1.3 Invarianter

Invarianter gitt i kapittel 2.2, beregnes her ved programlisting 6.4. I beregning av invariantene benyttes følgende ligninger:

$$A_x = \sqrt{\Re(X(\omega))^2 + \Im(X(\omega))^2}$$
$$A_y = \sqrt{\Re(Y(\omega))^2 + \Im(Y(\omega))^2}$$
$$A_\theta = atan2\left(\frac{A_y}{A_x}\right)$$
$$\delta_x = atan2\left(\frac{\Im(X(\omega))}{\Re(X(\omega))}\right)$$
$$\delta_y = atan2\left(\frac{\Im(Y(\omega))}{\Re(Y(\omega))}\right)$$

Listing 6.4: Beregne invarianter

```
1  Ay1 = abs(Y1(:,21));
2  Ax1 = abs(X1(:,21));
3  A_theta1 = atan2( Ay1, Ax1);
4
5  dy1 = atan2( imag(Y1(1:AntallBlokker,21)), real(Y1(1:AntallBlokker,21)) );
6  dx1 = atan2( imag(X1(1:AntallBlokker,21)), real(X1(1:AntallBlokker,21)) );
7  for idx=1:length(dx1)
8     Z_theta1(idx) = AngleDiff(dy1(idx),dx1(idx));
9  end
10
11 Ay2 = abs(Y2(1:AntallBlokker,21));
12 Ax2 = abs(X2(1:AntallBlokker,21));
13 A_theta2 = atan2( Ay2, Ax2);
14
15 dy2= atan2( imag(Y2(1:AntallBlokker,21)), real(Y2(1:AntallBlokker,21)) );
16 dx2 = atan2( imag(X2(1:AntallBlokker,21)), real(X2(1:AntallBlokker,21)) );
```

```
17
18 for idx=1:length(dx2)
19     Z_theta2(idx) = AngleDiff(dy2(idx),dx2(idx));
20 end
21 A_1 = mean(A_theta1);
22 A_2 = mean(A_theta2);
23 Z_1 = mean(Z_theta1);
24 Z_2 = mean(Z_theta2);
```

Når beregninger er utført, returneres følgende verdier fra programmet:

$$\overline{A}_{\theta,1} = \frac{1}{N}\sum_{n=1}^{N} A_{\theta,1}\ ,\ \overline{A}_{\theta,2} = \frac{1}{N}\sum_{n=1}^{N} A_{\theta,2}$$

$$\overline{Z}_{\theta,1} = \frac{1}{N}\sum_{n=1}^{N} Z_{\theta,1}\ ,\ \overline{Z}_{\theta,2} = \frac{1}{N}\sum_{n=1}^{N} Z_{\theta,2}$$

$$X_1(k)\ ,\ Y_1(k)\ ,\ Z_1(k)\ ,\ X_2(k)\ ,\ Y_2(k)\ ,\ Z_2(k)$$

6.2 Innledende eksperimenter

Før målinger på magnetfelt generert av en høyspentmast startes, er det ønskelig å måle magnetfelt generert under kontrollerte forutsetninger. Med dette oppnås full frihet til å bestemme både faseestrøm og posisjon til ledere og sensor.

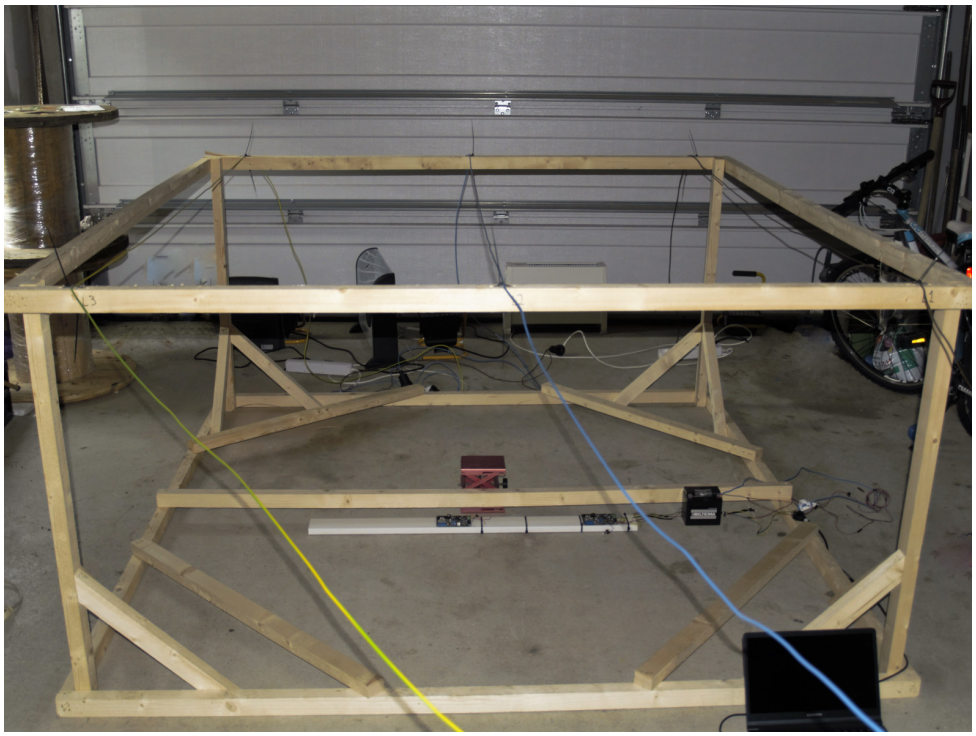
Figur 6.2 viser en illustrasjon over testoppsettet som benyttes i forbindelse med de innledende testene av å måle magnetfelt generert av vekselspenning. I testoppsettet kan som nevnt fasestrømmen varieres og dette utføres ved å endre effekten til lastene. Resistive laster som benyttes i eksperimentene er listet i tabell 6.1. Den innbyrdes posisjon til fasene L1,L2 og L3 er også mulig å endre. Ved å endre avstanden mellom lederene, vil det omkringliggende magnetfeltet også endres.

Type last	Fasestrøm
Ovn 2kW	8.6A
Halogen lampe 400W	1.64A
Ovn + lampe $\approx 2.4kW$	10A

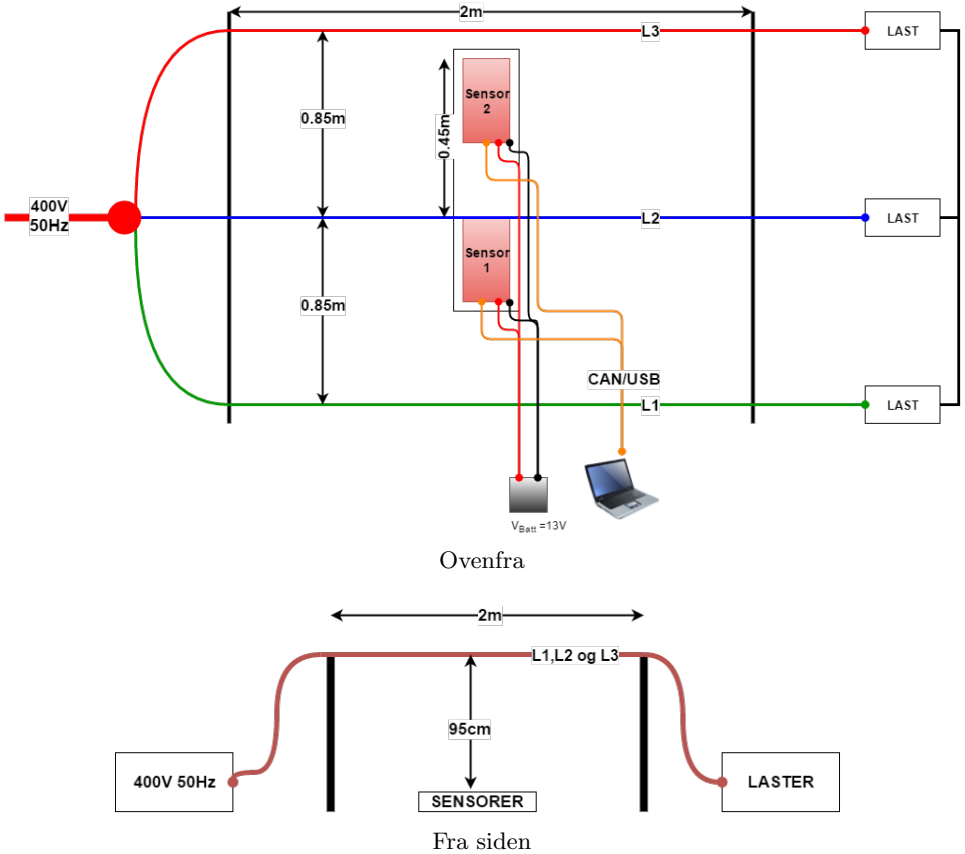
Tabell 6.1: Fasestrøm ved ulike resistive laster.

For å unngå støyproblemer som 50Hz og reguleringsstøy fra en strømforsyning, er det valgt å benytte eksternt batteri. De to sensorene blir dermed forsynt av en stabil og lavohmig kilde som ikke påtrykker støy.

Sensordata som måles, sendes som definerte meldinger over canbus. Disse meldingene leses og dekodes av en datamaskin ved hjelp av egnet canbus til usb grensesnitt. For analyse og videre



Figur 6.1: Testoppsett 3fase vekselspenning i garasje.

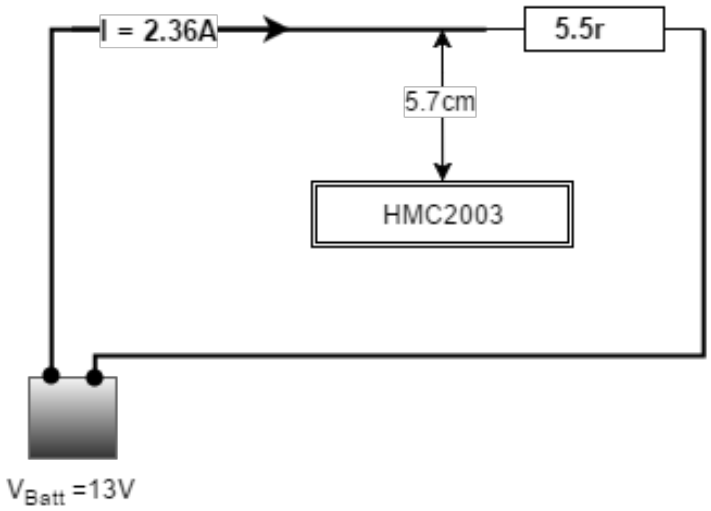


Figur 6.2: Testoppsett 3fase vekselspenning, med sentrale mål.

behandling av målingene er det valgt å benytte MATLAB.

6.2.1 Måle statisk magnetfelt

Det første forsøket som er valgt å utføre, er måling av et statisk magnetfelt generert av en likespenningskilde. Det er valgt å benytte et standard 12V batteri som kilde i dette eksperimentet. Støyproblemer som følge av 50Hz nettforsyning og reguleringsstøy fra en ordinær strømforsyning elimineres dermed. Sensordata som måles, sendes som definerte meldinger over canbus. Disse meldingene dekodes av en datamaskin for videre behandling i MATLAB.



Figur 6.3: Testoppsett for eksperiment.

Figur 6.3 illustrerer hvordan testoppsettet i dette eksperimentet er utført. Fra figuren gjenkennes likespenningskilden nederst i høyre hjørne. Denne kilden danner en tilnærmet stabil forsyning til sensor og effektresistans.

Teoretisk beregning

Strømmen, I, som danner magnetfeltet som sensor skal måle, beregnes ved:

$$I = \frac{U}{R}$$
$$I = \frac{13.0}{5.5\Omega} = 2.36A$$

Ved å sette inn informasjon fra figur og beregning ovenfor kan forventet magnetfelt beregnes ved

hjelp av ligning 2.4. Parameterene som inngår i ligningen oppsummeres først:

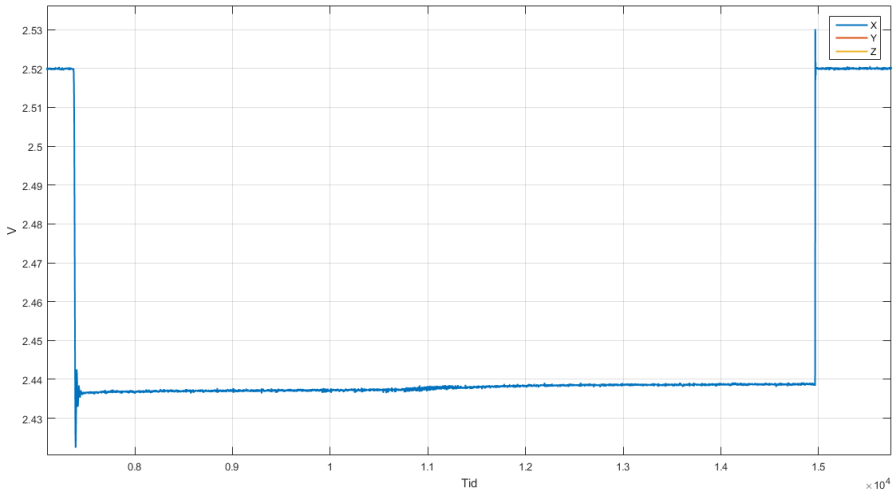
$$\begin{aligned} \mu_0 &= 4\pi \cdot 10^{-7} \\ d(i)_y &= 0.057m \\ d(i)_x &= 0.0m \\ I &= 2.36A \end{aligned}$$

Dette settes så inn i ligning og forventet feltstyrke beregnes:

$$\begin{aligned} B_x &= \frac{\mu_0 I_i d(i)_y}{2\pi(d(i)_x^2 + d(i)_y^2)} \\ B_x &= \frac{4\pi \cdot 10^{-7}(0.057)}{2\pi 0.057^2} \cdot 2.36 \\ B_x &\approx 8.28\mu T \end{aligned} \tag{6.6}$$

Måling fra sensor

Fra ligning 6.6 fant vi at forventet magnetfelt er $\approx 8.28\mu T$. Vi skal nå se på hvordan sensor måler det genererte magnetfeltet.



Figur 6.4: Dekodet signal fra sensor x-akse.

Figur 6.4 viser det målte spenningsnivået tilhørende x-aksen. Det målte magnetfeltet kan da beregnes ved:

$$\Delta V_{Sensor} = 2.52V - 2.437V = 0.083V \tag{6.7}$$

Sensoren HMC2003 har i datablad oppgitt sensitivitet til 1V endring, per Gauss. I denne oppgaven er det valgt å benytte Tesla som enhet og det må demed etableres en transferfunksjon som muligjør konvertering fra volt til Tesla. Transferfunksjonen finner vi til:

$$\begin{aligned} 1V &= 1Gauss = 0.0001Tesla \\ h &= \frac{0.0001Tesla}{1V} \\ h &= 0.00001\left[T/V\right] \end{aligned} \tag{6.8}$$

Nå kan den målte magnetfeltstyrken beregnes ved:

$$\begin{aligned} B_x &= \Delta V_{Sensor} \cdot h \\ B_x &= 0.083[V] \cdot 0.0001\left[T/V\right] \\ B_x &= 8.3\mu T \end{aligned} \tag{6.9}$$

Konklusjon

For å kunne si noe om hvor lik den målte feltstyrken er den teoretisk beregnede, velges det å beregne avviket,α, mellom de to.

$$\begin{aligned} \alpha &= 8.3\mu T - 8.28\mu T \\ \alpha &= 0.02\mu T \end{aligned}$$

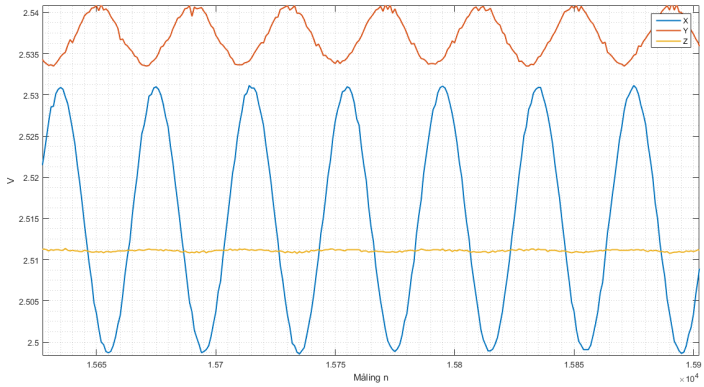
Fra den beregnede α, ser vi at det er et lite avvik mellom teoretisk beregnet feltstyrke og målt feltstyrke. Avviket kan ha flere kilder, men det kan antas at størstedelen av avviket skylles lav presisjon i posisjonsmåling mellom leder og sensor. Det er i datablad ikke oppgitt nøyaktig hvor i HMC2003 sensor AMR elementet er posisjonert. Posisjonen må antas og dermed tap i presisjon. På tross av et lite avvik, kan det konkluderes med at eksperimentet var vellykket og sensor er i stand til å måle lave magnetfelt.

6.2.2 Måle tidsvarierende magnetfelt

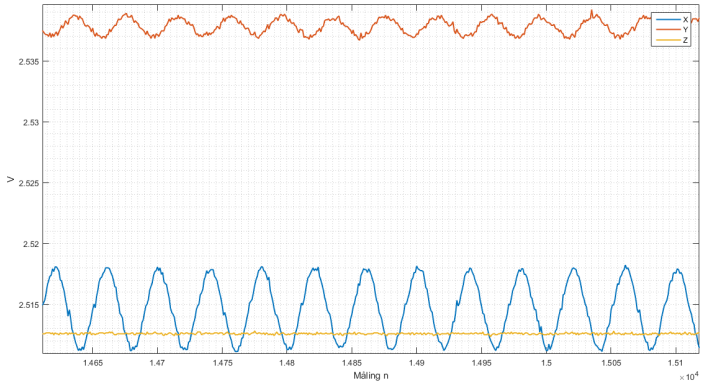
Formålet med dette innledende eksperimentet er å undersøke om sensornoden er i stand til å måle et tidsvarierende magnetfelt og om det er behov for digital filtrering av målingene.

Testutstyret som benyttes i eksperimentet er illustrert i figur 6.2, med laster definert i tabell 6.1. Målingene utføres i samme punkt i testutstyret og kurvene bør dermed ha lik fase, men ulik amplitude, for de ulike lastene.

Plott av tidsvarierende magnetfelt



Målinger ved 8.6A last.



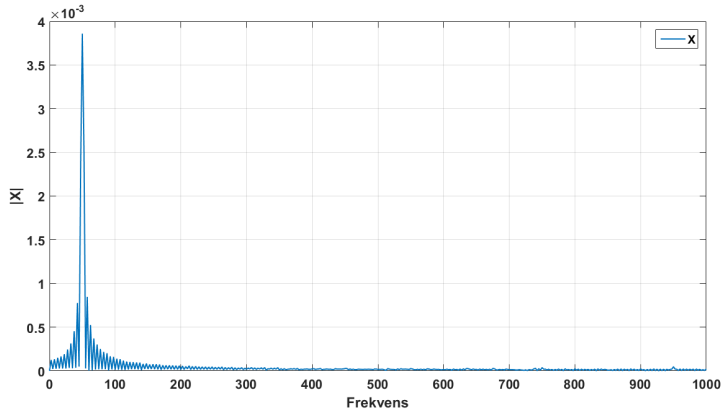
Målinger ved 1.6A last.

Figur 6.5: Målinger med ulik last, i samme posisjon.

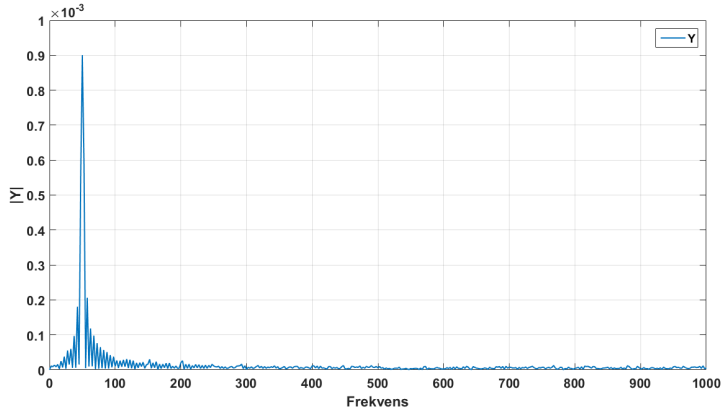
Målingene som vises i figur 6.5, viser målt magnetfelt i samme posisjon, med endret last. Fra figuren fremkommer det tydelig at målt magnetfeltet varierer med en fast frekvens og med stabil amplitude.

I plottet som viser målt magnetfelt tilhørende den laveste laststrøm, kan det allikevel synes at spesielt y-aksen er noe befengt av støy. Hvorvidt denne støyen er av en slik karater at den påvirker invariantberegninger, må undersøkes nærmere ved frekvensanalyse av målesignalene. Dersom dette er tilfellet, bør digitalfiltrering av måledata vurderes. I denne problemstillingen kan trolig et lett IIR-filter utføre en akseptabel filtrering.

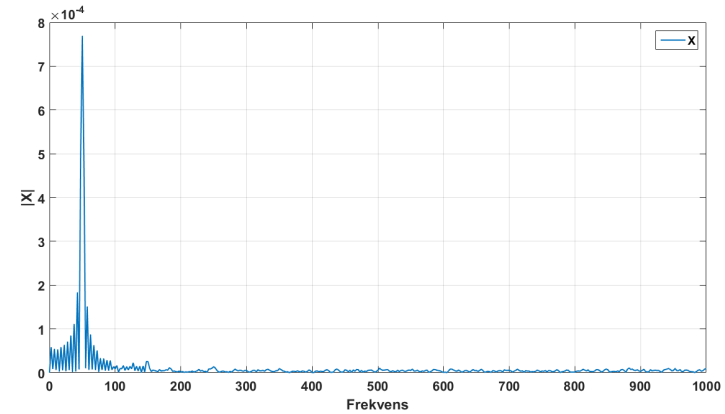
Frekvensanalyse



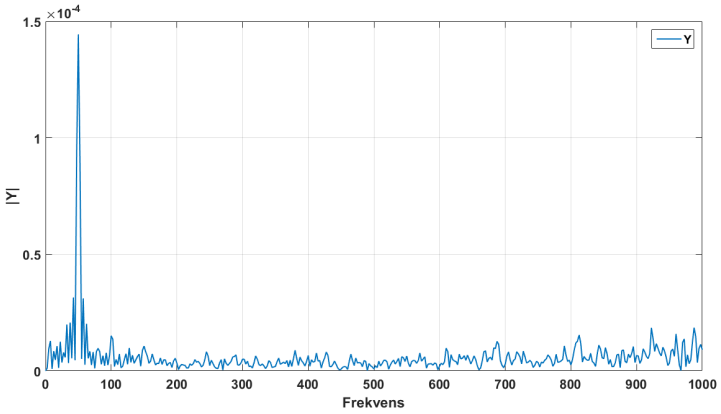
Figur 6.6: Frekvensanalyse av magnetfeltmålinger, FFT x-akse ved 8.6A last.



Figur 6.7: Frekvensanalyse av magnetfeltmålinger, FFT y-akse ved 8.6A last.



Figur 6.8: Frekvensanalyse av magnetfeltmålinger, FFT x-akse ved 1.6A last.



Figur 6.9: Frekvensanalyse av magnetfeltmålinger, FFT y-akse ved 1.6A last.

Figur 6.6 til figur 6.6 viser frekvensanalysen til magnetfeltmålingene vist i figur 6.5. Fra plottene i figuren, kan det legges merke til at amplituden ved 50Hz er dominerende, selv uten digital filtrering.

Konklusjon

I det 2. inledende eksperimentet, var målet å verifisere at sensorer er i stand til å måle tidsvarierende magnetfelter. Magnetfeltmålingene vil senere danne grunnlaget for beregning av invarianter. For å oppnå høy presisjon i nevnte beregning, er det avgjørende at frekvenskomponenten med i informasjon, fremkommer tydelig i beregning av FFT.

Basert på frekvensanalyse i figurene 6.6 til figur 6.6, kan det konkluderes med at målet ble oppnådd og det er ikke nødvendig å vurdere digital filtrering.

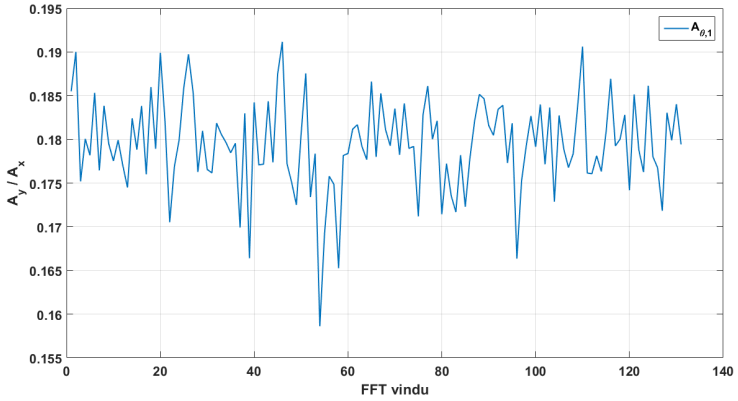
6.2.3 Invarianter basert på målinger

Invariantene fra [25], oppsummert i kapittel 2.2, skal i dette delkapittelet verifiseres at de er uavhengig av fasestrømmen. For å kunne verifisere dette, skal målinger av magnetfeltet i både det horisontale- og vertikale-plan utføres. Tilgjengelig testutstyr har ikke direkte mulighet for denne type test. Dette løses imidlertid ved å dele opp testen i en vertikal og en horisontal del.

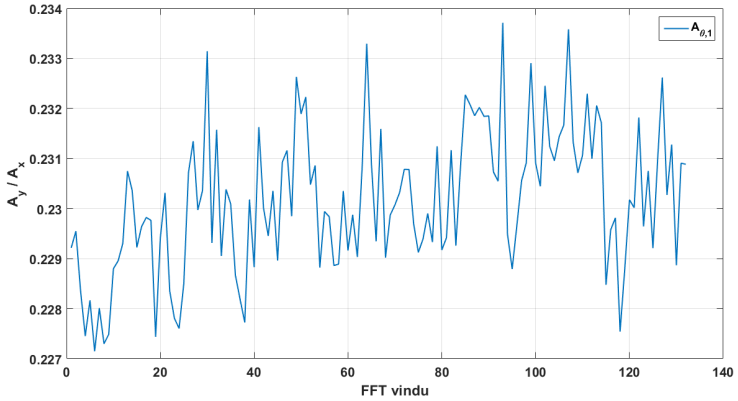
Eksperimentet startes med å verifisere stabiliteten til invariantene A_θ og Z_θ. Til verifiseringen benyttes samme datasett som ovenfor.

Invariant A_θ

For å få et visuelt bilde over beregnet A_θ, starter vi med å se på et plott over invarianten.



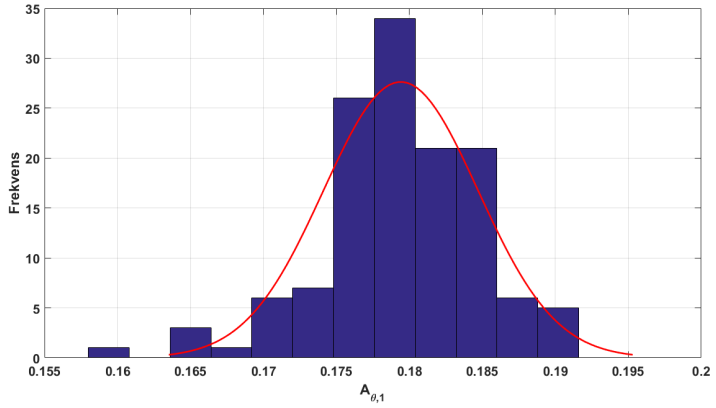
1.6A last.



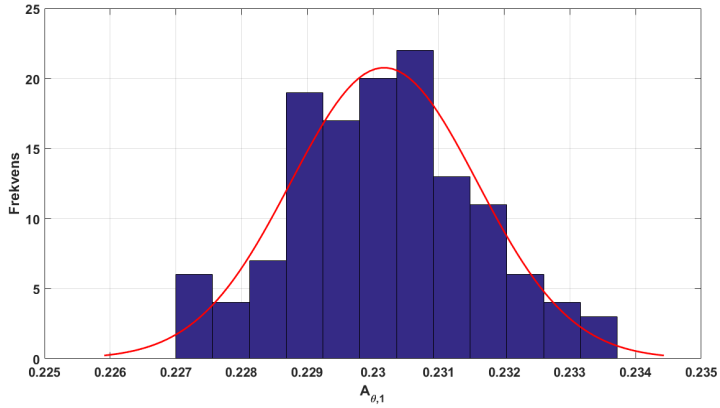
8.6A last.

Figur 6.10: Beregnet A_θ ved ulik last.

Fra plottene i figur 6.10 er det tydelig at invarianten A_θ ikke er så konstant som gitt av [25]. Før det er mulig å si noe mer om invarianten A_θ, er det naturlig å se på fordelingen til denne.



1.6A last.



8.6A last.

Figur 6.11: Fordeling til A_θ, ved ulik last.

Figuren 6.11 viser fordelingen til invarianten A_θ. Fra denne kan det legges merke til at fordelingen synes normalfordelt. Siden antall vinduer er større enn 20, kan sentralgrenseteoremet gitt i ligning 5.6 benyttes [15]. Forventningsverdien til A_θ kan dermed beregnes ved:

$$\overline{A}_\theta = E\left[A_\theta\right] = \mu_{A_\theta} = \frac{1}{N}\sum_{n=1}^{N} A_\theta \tag{6.10}$$

Med tilhørende standardavvik:

$$\sigma_{A_\theta} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(x_n - \mu_{A_\theta}\right)^2} \tag{6.11}$$

Ved hjelp av sentralgrenseteoremet kan invarianten A_θ, tallfestes med tilhørende standardavvik. Dette beregnes for begge sensorene og vises i tabell 6.2.

Det er nå tydelig at invarianten A_θ er mindre stabil en først antatt. Hvilke konsekvenser dette får for presisjonen til navigering i magnetfelt rundt en høyspentmast, er på nåværende stadiet

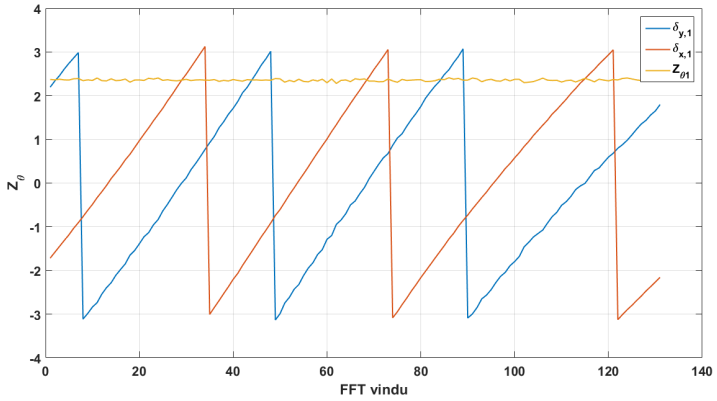
Fasestrøm	Invariant	$\overline{A}_\theta$ [rad]	σ_{A_θ} [rad]
1.6A	$A_{\theta,1}$	$179.8 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$
1.6A	$A_{\theta,2}$	$298.6 \cdot 10^{-3}$	$7.1 \cdot 10^{-3}$
8.6A	$A_{\theta,1}$	$230.2 \cdot 10^{-3}$	$1.4 \cdot 10^{-3}$
8.6A	$A_{\theta,2}$	$283.3 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$

Tabell 6.2: Forventningsverdi og standardavvik til A_θ.

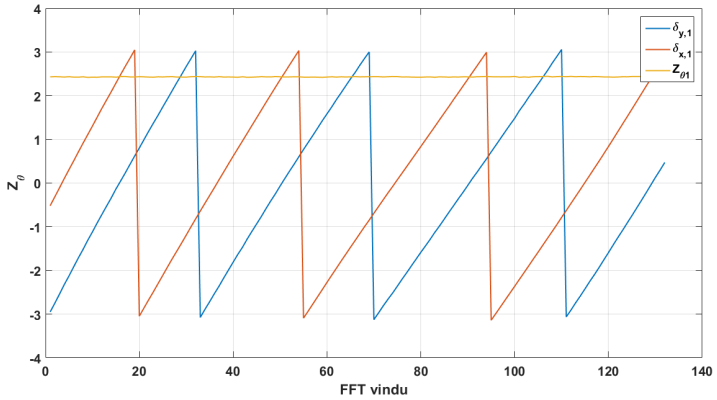
vanskelig å ta stilling til.

Invariant Z_θ

Vi starter med å se på plott av variablene δ_x og δ_y i figur 6.12. Fra figuren er det tydelig at variablene δ_x og δ_y, er periodiske sagtannspulser, med konstant amplitude. Dersom plottet for 1.6A last studeres, kan allikevel noe støy identifiseres.



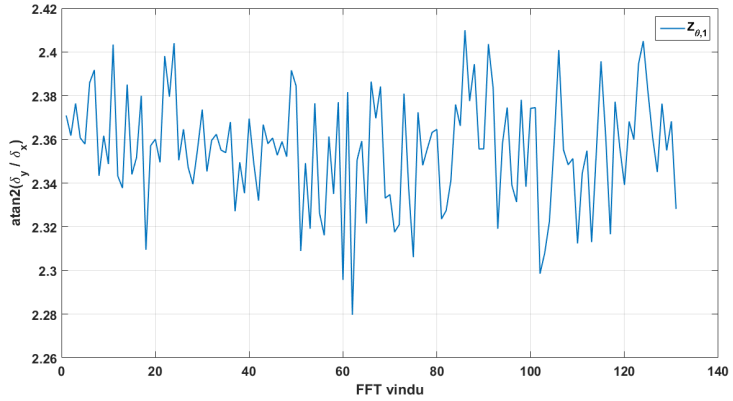
1.6A last.



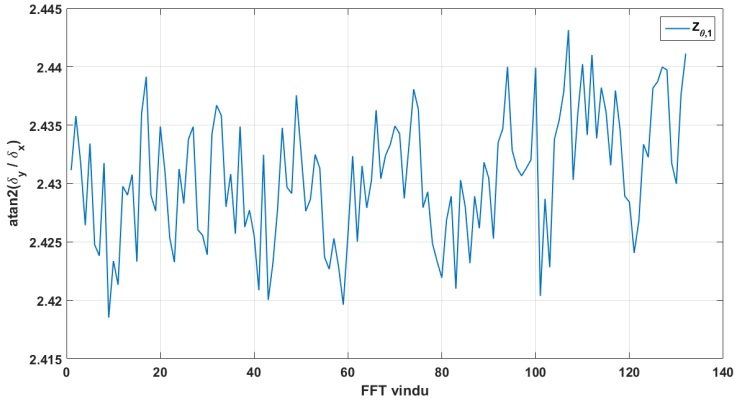
8.6A last.

Figur 6.12: Vinklene δ_x,δ_y og beregnet Z_θ, ved ulik last.

Variablene δ_x, δ_y, blir deretter benyttet for å beregne invarianten Z_θ. Plott av invarianten, ved ulik last vises i figur 6.13.



1.6A last.



8.6A last.

Figur 6.13: Beregnet Z_θ ved ulik last.

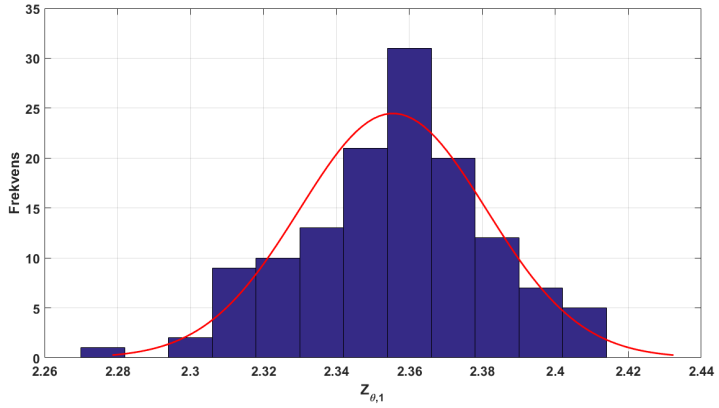
Fra plottene i figur 6.13 er det tydelig at også invarianten Z_θ ikke er så konstant som gitt av [25]. Før det er mulig å si noe mer om invarianten Z_θ, er det naturlig å se på fordelingen til denne.

Også her synes sentralgrenseteoremet å kunne benyttes for å beregne den forventede invariant, med tilhørende standardavvik. I beregningen benyttes ligning 6.10 og ligning 6.11. Verdiene beregnes for begge sensorene og vises i tabell 6.2.

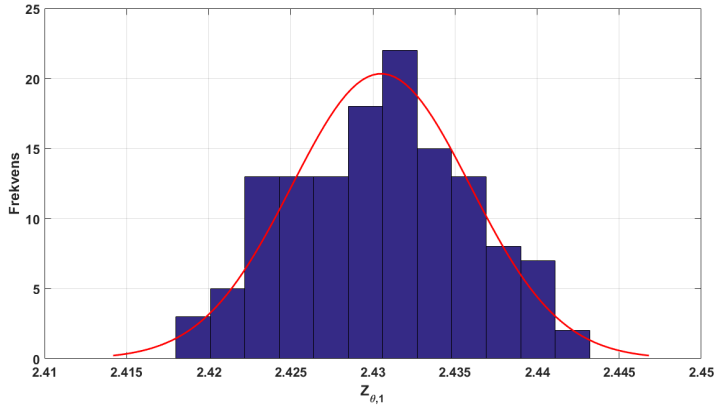
Fasestrøm	Invariant	$\overline{Z}_\theta$ [rad]	σ_{Z_θ} [rad]
1.6A	$Z_{\theta,1}$	2.3561	$26.6 \cdot 10^{-3}$
1.6A	$Z_{\theta,2}$	1.7074	$23.7 \cdot 10^{-3}$
8.6A	$Z_{\theta,1}$	2.4305	$5.4 \cdot 10^{-3}$
8.6A	$Z_{\theta,2}$	1.7005	$6.9 \cdot 10^{-3}$

Tabell 6.3: Forventningsverdi og standardavvik til Z_θ.

Det er nå tydelig at også invarianten Z_θ er mindre stabil en først antatt. Hvilke konsekvenser dette får for presisjonen til navigering i magnetfelt rundt en høyspentmast, er på nåværende tidspunkt vanskelig å ta stilling til.



1.6A last.



8.6A last.

Figur 6.14: Fordeling til Z_θ, ved ulik last.

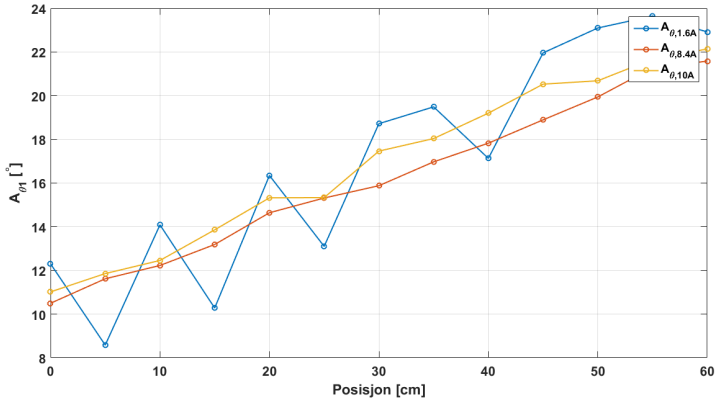
Horisontal posisjonsendring

I dette eksperimentet skal respons til invariantene analyseres ved posisjonsendring i det horisontale plan. Dersom invariantene virkelig er invarianter, er det naturlig å forvente lik respons ved ulik fasestrøm. For å teste dette, er det valgt å benytte de 3 definerte fasestrømmer. Magnetfeltet måles så i 13 ulike posisjoner, med 5 cm avstand mellom posisjonene.

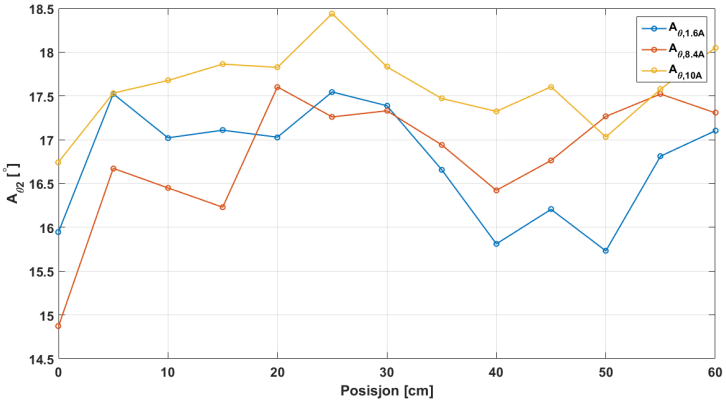
Figur 6.15 viser beregnet A_θ ved ulik fasestrøm og posisjon. I denne figuren kan det virke som spesielt $A_{\theta,2}$ er noe uforutsigbar. Siden $A_{\theta,1}$ virker betydelig mer forutsigbar, er det vanskelig å ta stilling til om dette kan relateres til fasestrøm eller måletekniske problemer.

Figur 6.16 viser beregnet Z_θ ved ulik fasestrøm og posisjon. Dersom figuren sammenlignes med figur 6.15, kan det legges merke til at invarianten Z_θ synes å være betydelig mer forutsigbar.

For å kunne si noe mer angående plottene av invariantene er det behov for en oversikt over standardavviket til hver enkelt invariant. Dette plottes i figurene nedenfor.

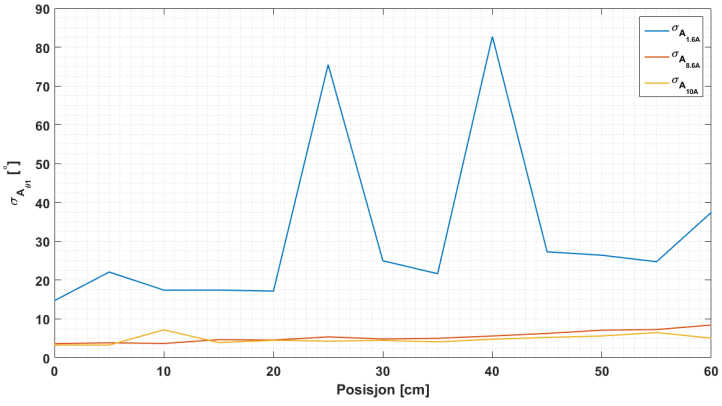


$A_{\theta,1}$

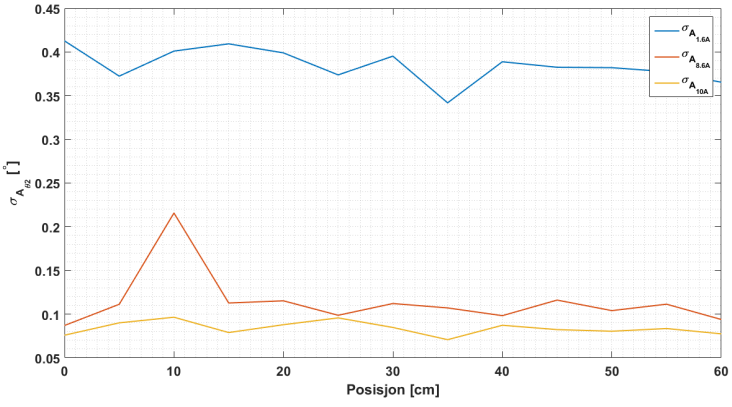


$A_{\theta,2}$

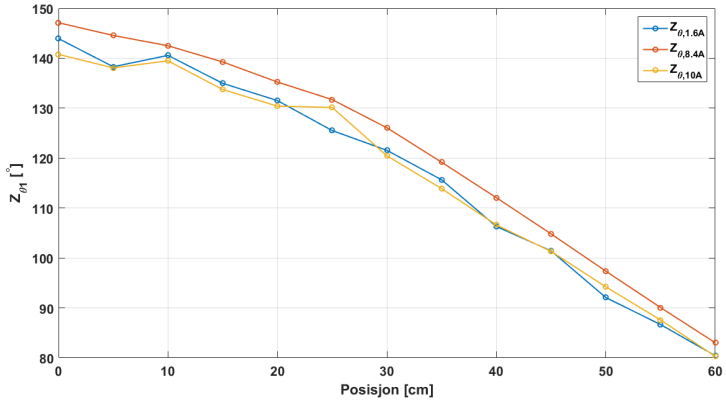
Figur 6.15: A_θ, ved ulik posisjon og last.



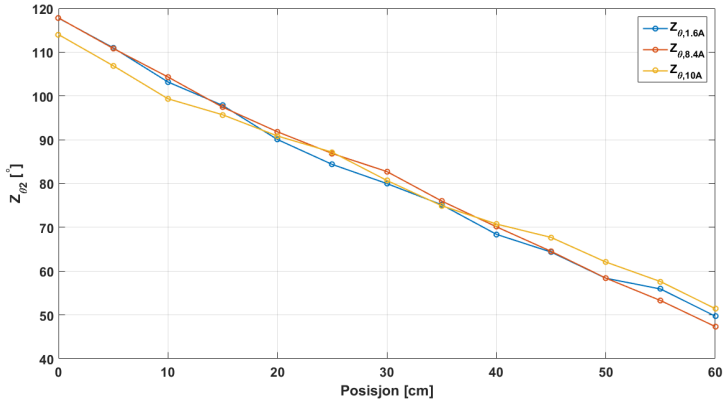
$A_{\theta,1}$



$A_{\theta,2}$



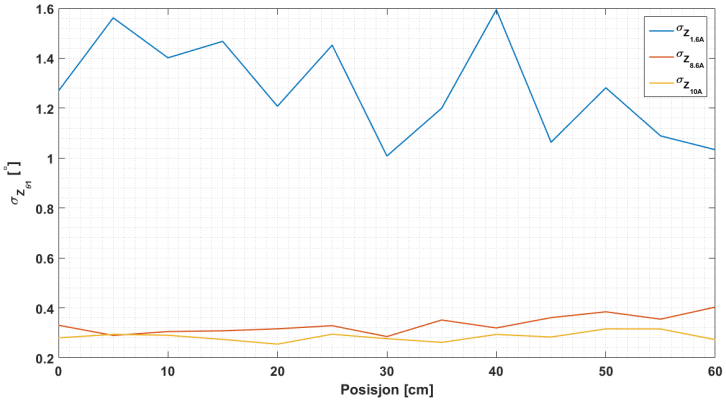
$Z_{\theta,1}$



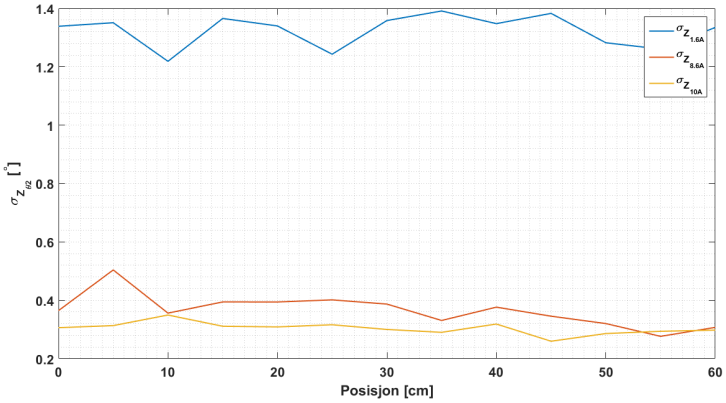
$Z_{\theta,2}$

Figur 6.16: Z_θ, ved ulik posisjon og last.

Figur 6.17 viser beregnet standardavvik tilhørende invarianten A_θ. Fra denne figuren er det tydelig at avviket er betydelig større for målingen med lav fasestrøm. For de andre to, kan det observeres en antatt måleteknisk feil i posisjon 10cm. Foruten denne måletekniske feilen, synes standardavviket ved fasestrøm 8.6A og 10A å være likt.



$Z_{\theta,1}$



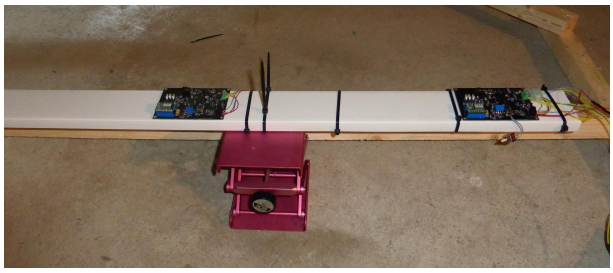
$Z_{\theta,2}$

Figur 6.18: Standardavviket til invarianten Z_θ, ved ulik posisjon og last.

Figur 6.18 viser beregnet standardavvik tilhørende invarianten Z_θ. Også for denne invarianten, er det tydelig at standardavviket er betydelig høyere for målingen med lav fasestrøm. De andre to, synes å svinge rundt et tenkt fast standardavvik.

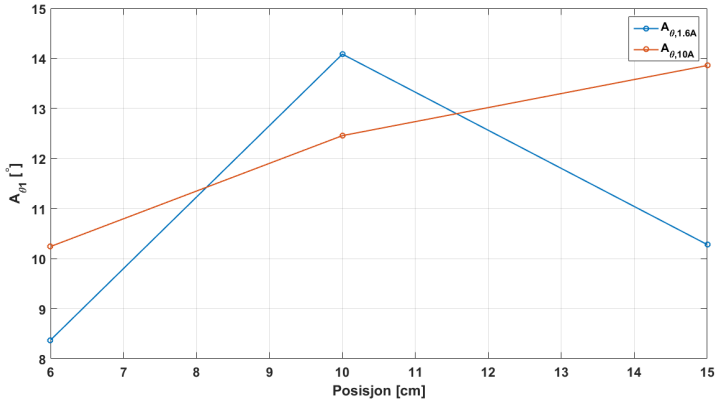
Vertikal posisjonsendring

For kunne måle vertikal posisjonsendring, ble det nødvendig å finne utstyr som trinnløst kunne flytte sensornodene oppover i høyden. Det ble raskt besluttet at en saksejekk av aluminium var godt egnet til dette formålet. Fotografi av måleutstyret vises i figur 6.19.

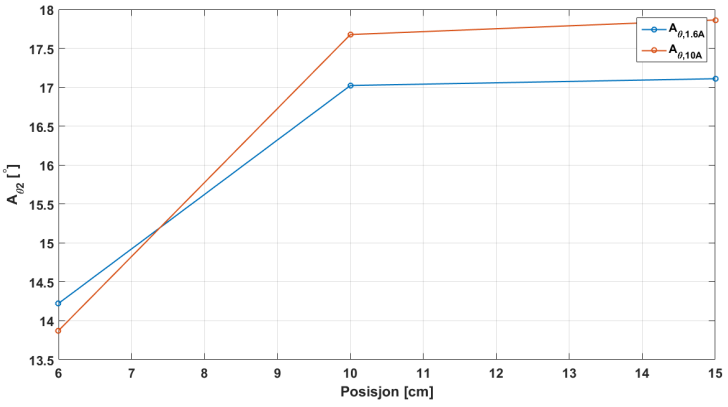


Figur 6.19: Fotografi av vertikal magnetfeltmåling i testutstyret.

Det ble videre valgt å teste med 2 ulike fasestrømmer i 3 ulike vertikale posisjoner. De valgte fasestrømmer er 1.6A og 10A. Nullpunktet i den vertikale posisjonsendringen er valgt til minste høyde av saksejekk, det vil si 6cm. De neste punktene som magnetfeltet måles i, er 10cm og 15cm. Dette anses som tilstrekkelig for å kunne identifisere om invariantene er i stand til å måle vertikal endring i magnetfeltet. Vi starter med å se på invariantene A_θ.



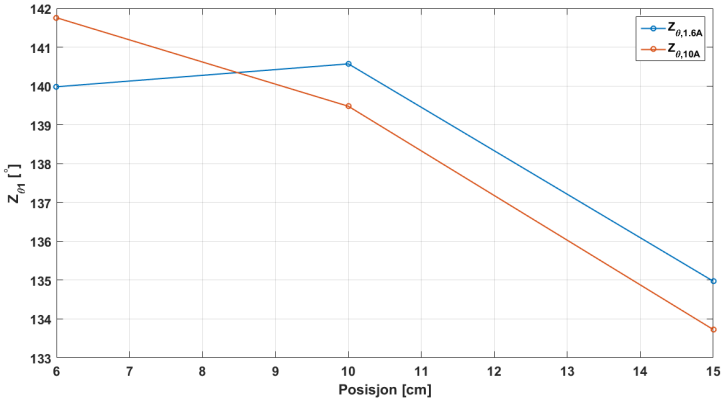
$A_{\theta,1}$



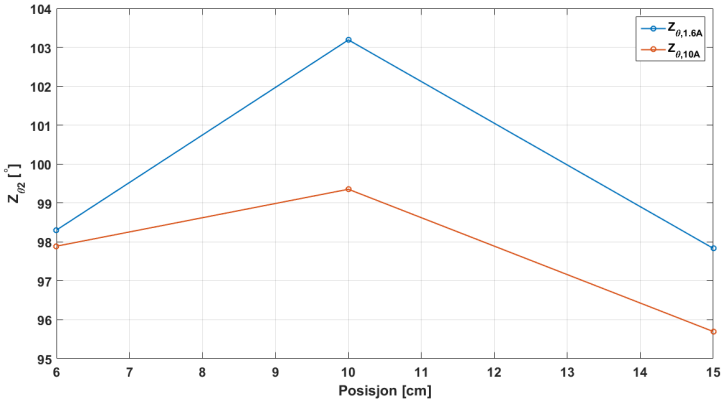
$A_{\theta,2}$

Figur 6.20: A_θ, ved ulik posisjon og last.

Figur 6.20 viser beregnet A_θ ved ulik fasestrøm og vertikal posisjon. Fra denne figuren er det tydelig å se at invarienten A_θ endres ved vertikal posisjonsendring. Som for de andre innledenende eksperimentene, er det også her indikasjoner på at beregnet A_θ varierer ved ulik fasestrøm.



$Z_{\theta,1}$

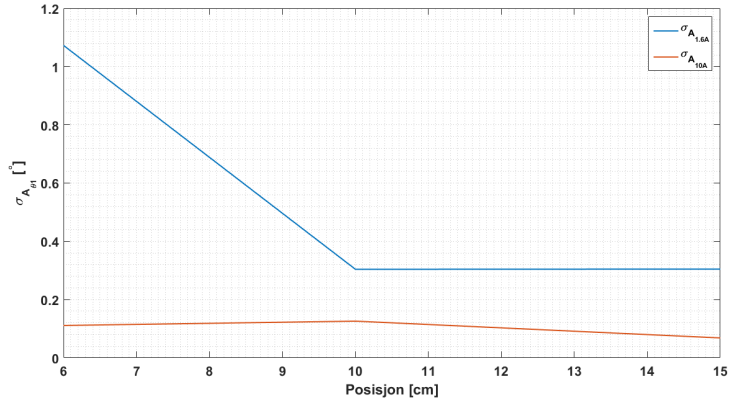


$Z_{\theta,2}$

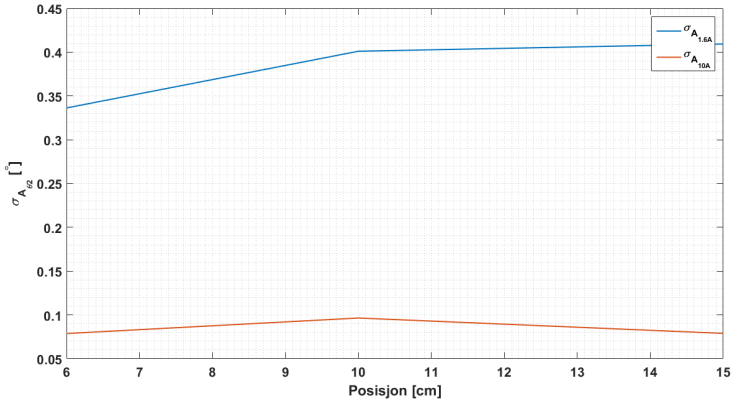
Figur 6.21: Z_θ, ved ulik posisjon og last.

Figur 6.21 viser beregnet Z_θ ved ulik fasestrøm og posisjon. Det fremkommer tydelig at også invarianten Z_θ endres ved vertikal posisjonsendring. Som for de andre innledenende eksperimentene, er det også her indikasjoner på at beregnet Z_θ varierer ved ulik fasestrøm.

For å kunne si noe mer angående invariantene, er det nå naturlig å se på tilhørende standardavvik.



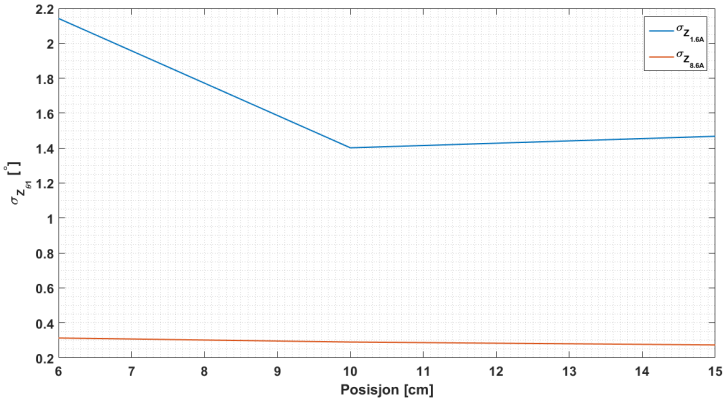
$A_{\theta,1}$



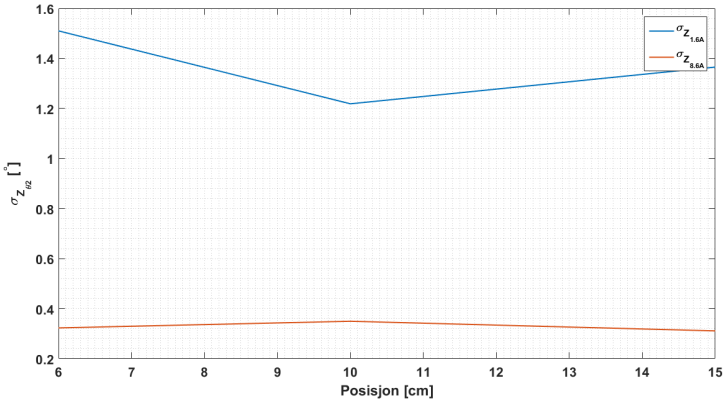
$A_{\theta,2}$

Figur 6.22: Standardavviket til invarianten A_θ, ved ulik posisjon og last.

Figur 6.22 viser standardavviket til invarianten A_θ. Fra denne figuren er det tydelig at en høyere fasestrøm gir en lavere variasjon i beregnet invariant. Standardavvikene tilhørende målingene for høy fasestrøm, synes også å være stabile. For lav fasestrøm synes standardavvikene å gå mot et standardavvik på $\approx 0.4°$.



$Z_{\theta,1}$



$Z_{\theta,2}$

Figur 6.23: Standardavviket til invarianten Z_θ, ved ulik posisjon og last.

Figur 6.23 viser beregnet standardavvik tilhørende invarianten Z_θ. Også i denne figuren synes standardavviket til invarianten basert på den høyest fasestrømmen å være mest stabil.

Konklusjon

Målet for dette eksperimentet var å verifisere at invariantene virkelig er uavhengig av fasestrøm.

Fra plottene er det tydelig at en høyere fasestrøm resulterer i en mer konsekvent invariant og dermed lavere standardavvik. Basert på funn i dette eksperimentet og fra figurene 6.16 og 6.16, er det ikke mulig å konkludere med at invariantene er fullstendig uavhengige fra fasestrømmen.

6.2.4 Konklusjon av innledende eksperimenter

De innledende eksperimentene startet med å verifisere at sensornoden er i stand til å måle et statisk magnetfelt med høy presisjon. Dette ble i høy grad bekreftet gjennom å sammenligne måleresultater med beregnet feltstyrke.

Siden den utviklede sensor er tiltenkt å måle et tidsvarirende magnetfelt, ble også dette verifisert. Analyser av både målinger og frekvensanalyse ble lagt til grunn for å konkludere med at sensornoden er i stand til å måle tidsvarierende magentfelt. Et viktig resultat av dette eksperimentet var, basert på frekvensanalyse av signalene, at det ikke synes å være behov for digital filtrering av måledata.

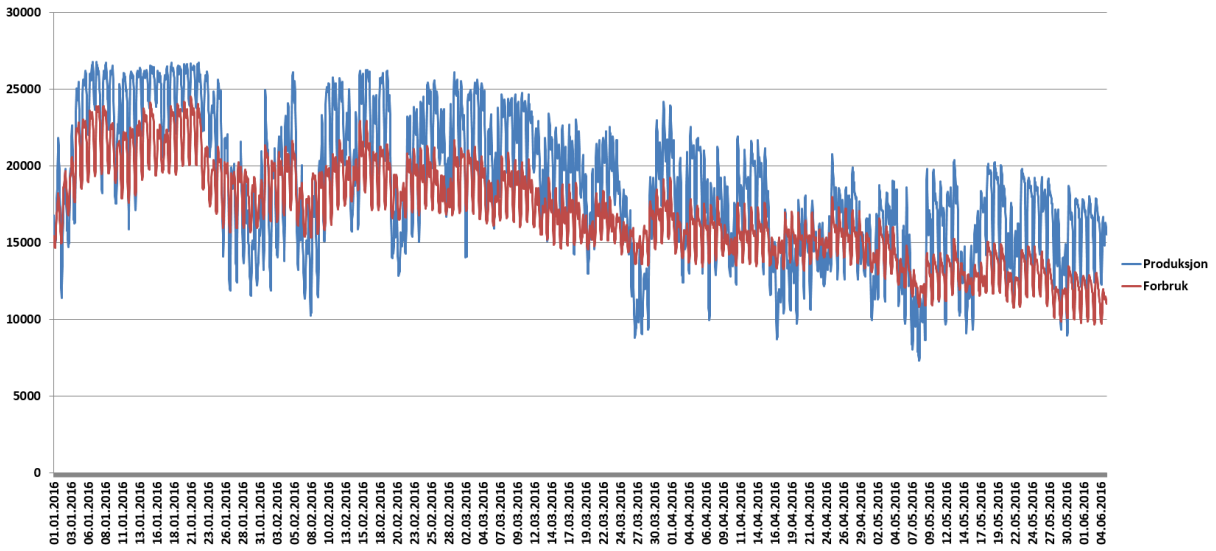
Det siste innledende eksperimentet hadde som formål å analysere hvorvidt invariantene var stabile. Eksperimentene som ble utført var basert på både målinger i samme punkt og ved forflytning. Fra disse ble det funnet en større variasjon i invariantene en hva som på forhånd var forventet. Det viste seg at spesielt A_θ hadde en noe uforutsigbar utvikling ved forflytning. Fra [25] er det gitt at invariantene skal være uavhengige av fasestrømmen. Denne påstanden var ikke mulig å verifisere gjennom de innledende eksperimenter. Fra plottene til invariantene er det allikevel tydelig at de gjennspeiler en faktisk posisjonsendring og kan benyttes som forklart i [25].

På tross av at det ikke var mulig å verifisere at invariantene var uavhengig av fasestrømmen, kan det konkluderes med at de innledende eksperimentene var vellykkede og definerte mål ble oppnådd.

6.3 Eksperimenter

I dette kapittelet skal resultater fra målinger, utført på høyspentmaster presenteres og analyseres. Målingene vil her bære noe preg av at det er begrensede muligheter til å påvirke både magnetfeltet og avstanden opp til ledere.

De fleste høyspentmaster av en viss størrelse er plassert i utilgjengelige områder. Terrenget som målingene utføres i er dermed av en slik beskaffenhet at presisjon til avstander i det horisontale og vertikale plan må forventes lavere i forhold til hva som var mulig i de innledende eksperimenter.



Figur 6.24: Oversikt over produksjon og forbruk i Norge i 2016.

I figur 6.24 er data fra www.statnett.no plottet. Denne figuren viser tydelig variasjoner i forbruk og produksjon av strøm og denne variasjonen medfører også variasjon i magnetfeltet som skal måles.

Figur 6.25: Fotografi av måleutstyr benyttet til målinger i felt.

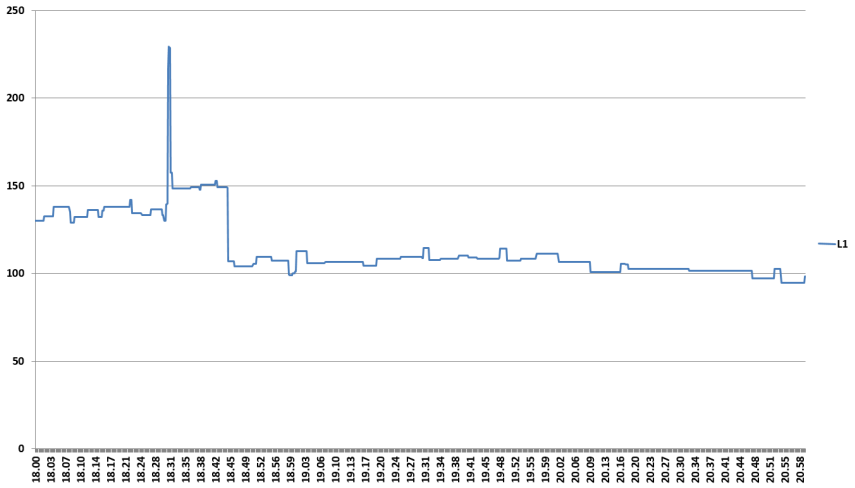
Figur 6.25 viser måleutstyret som benyttes til målinger i felt. Sensornodene er, som ved de innledende eksperimenter, montert på en rett plastkanal med 45 centimeter avstand mellom sensornodene. For å kunne endre høyde og tilpasse utstyret til det krevende terrenget, er sensornodene plassert på et stativ. Dette stativet er relativt fleksibelt da det har mulighet til individuell tilpassning av lengden til hvert ben. Med dette oppnås et målesystem som til en viss grad kan benyttes i krevende terreng.

6.3.1 Målinger på 300kV høyspentlinje fra Åna Sira til Kjelland

Figur 6.26: Fotografi av måleoppsett.

Fra fotografi i figur 6.26 vises måleoppsettet der eksperimentene ble utført. Som fotografiet viser, er terrenget i området krevende. På tross av krevende terreng, ble denne lokasjonen valgt på grunn av dens relativt lave avstand fra fasene. På øyemål, ble avstand fra måleutstyr til fasene estimert til 10 meter.

I høyspentlinjen mellom Åna Sira og Kjelland måles kun strømmen i fasen L1, figur 6.27. Dette innebærer at de andre fasestrømmene er ukjente størrelser. Siden invariantene antas uavhengige av fasestrømmen, anses ikke dette kritisk for eksperimentet.



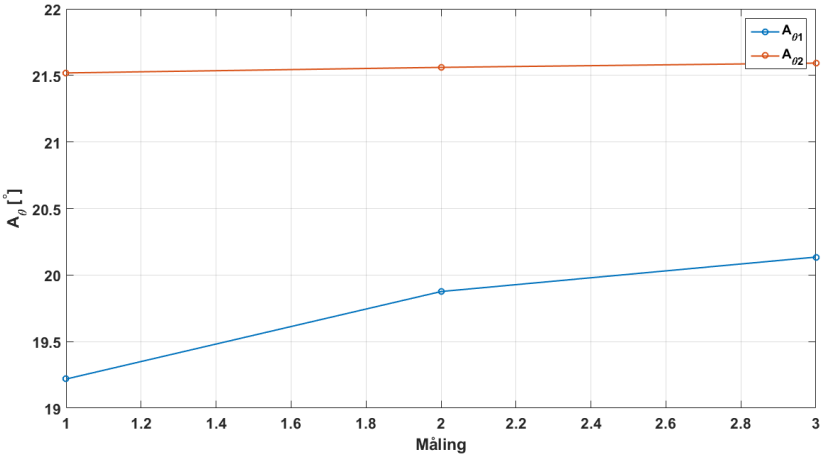
Figur 6.27: Fasestrøm i L1 under eksperimentet.

Under eksperimentene ble det registrert periodisk hørbar støy fra høyspentmasten. Støyen kommer trolig fra utladninger mellom en eller flere faser og til luft/konstruksjon. Støyen antas å også kunne påvirke magnetfeltet som måles.

Målinger i samme punkt

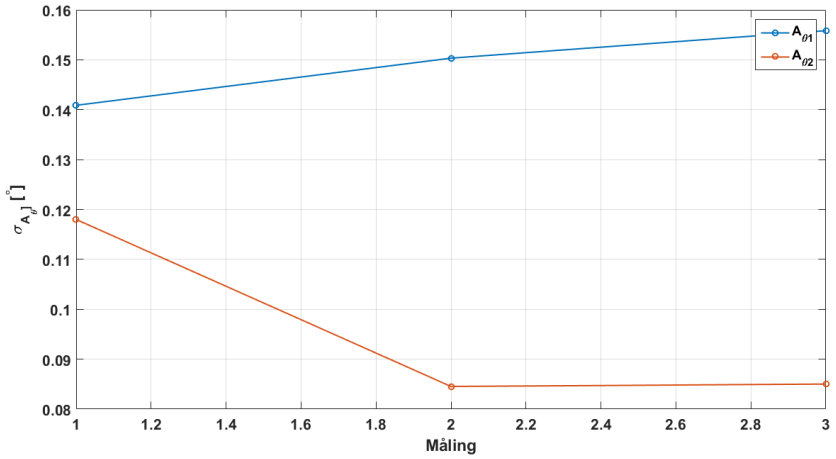
Det første settet med målinger som skal analyseres, er 3 etterfølgende målinger, målt i samme punkt.

Målingene som denne analysen baseres på er målt i intervallet fra 20:09 til 20:12. Ved å studere fasestrømmen til L1 i figur 6.27, kan det legges merke til at det var endring i fasestrømmen under målingene.



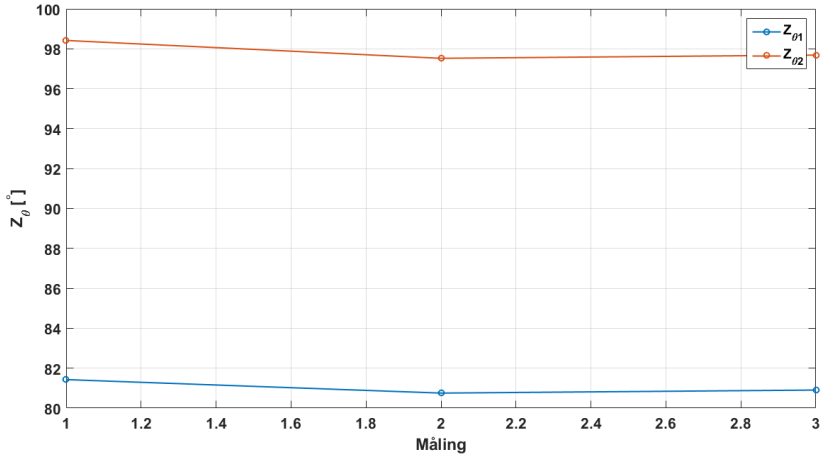
Figur 6.28: Beregnet A_θ.

Figur 6.28 viser den beregnede A_θ. Fra denne figuren synes invarianten $A_{\theta,2}$ å være helt stabil. Invarianten $A_{\theta,1}$ har derimot en endring på omtrent 1°. For å kunne si noe mer om variasjoner i invarianten skal standardavviket tilhørende målingen analyseres.



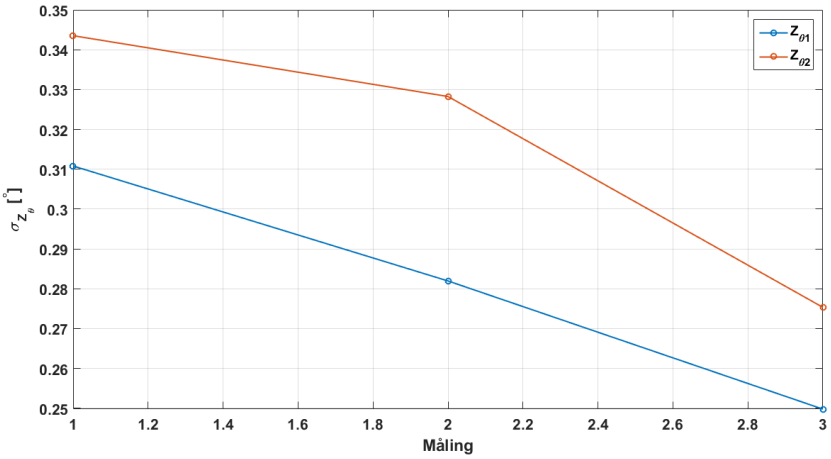
Figur 6.29: Beregnet σ_{A_θ}.

Figur 6.29 viser standardavviket til målingene i figur 6.28. Figuren viser tydelig at avviket ikke er konstant, men endres fra måling til måling. Endringen til invarianten $A_{\theta,1}$, i figur 6.28, kan her bekreftes ved økende standardavvik. I størrelsesorden synes standardavviket i dette eksperimentet å tilsvare standardavviket $\sigma_{A_{10A}}$, vist i figur 6.22.



Figur 6.30: Beregnet Z_θ.

Figur 6.30 viser den beregnede Z_θ. Fra denne figuren synes invariantene $Z_{\theta,1}$ og $Z_{\theta,2}$ å ha litt variajon fra måling til måling. Variasjonen i invariantene er allikevel så liten at de kan anses som stabile.



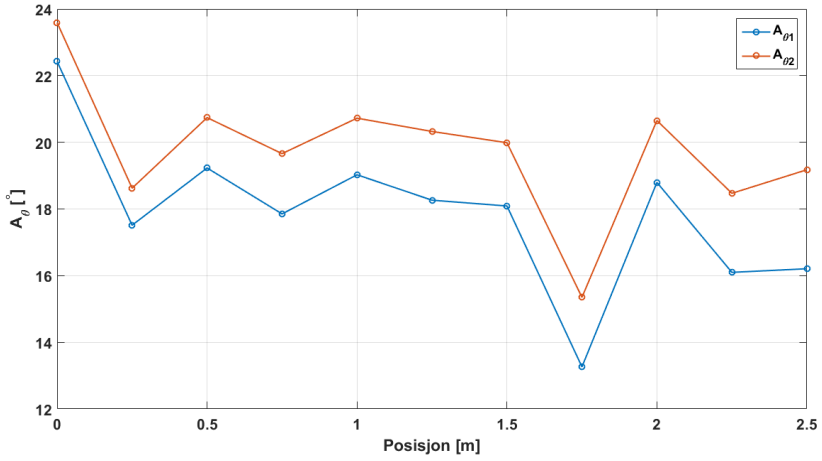
Figur 6.31: Beregnet σ_{Z_θ}.

Figur 6.31 viser standardavviket til målingene i figur 6.30. Figuren viser tydelig at avviket ikke er konstant, men endres fra måling til måling. Også denne invarianten kan synes å ha tilsvarende standardavvik som ble beregnet for σ_{Z_θ} i figur 6.23.

Horisontal posisjonsendring

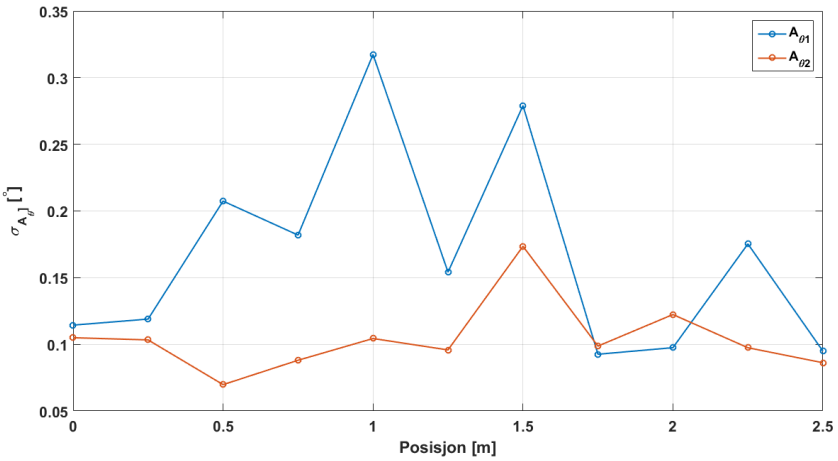
I dette eksperimentet er magnetfelt målt i 11 punkter, på tvers av høyspentmasten. Hver av punktene er forskjøvet med 25 centimeter i vertikal retning og det er forsøkt å opprettholde samme horisontale posisjon gjennom hele eksperimentet. Terrenget der målingene ble utført var av slik beskaffenhet at presisjon til målingene må antas noe redusert, se figur 6.26.

Målingene som denne analysen baseres på er målt i intervallet fra 19:47 til 20:06. Ved å studere fasestrømmen til L1 i figur 6.27, kan det legges merke til at det var en betydelig endring i fasestrømmen under målingene.



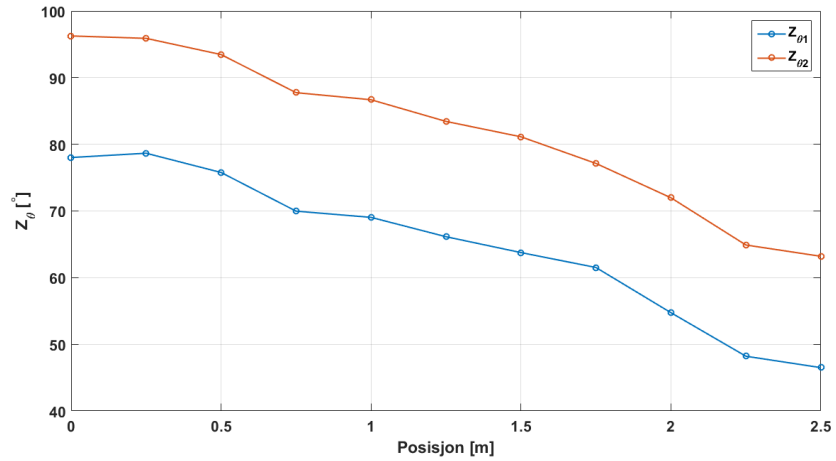
Figur 6.32: Beregnet A_θ.

Figur 6.32 viser den beregnede A_θ. Fra denne figuren synes invariantene A_θ å gjennspeile den horisontale forflytningen bra. Ved å studere figuren nærmere er det tydelig at invarianten $A_{\theta,2}$ er lik $A_{\theta,1} + 2°$.



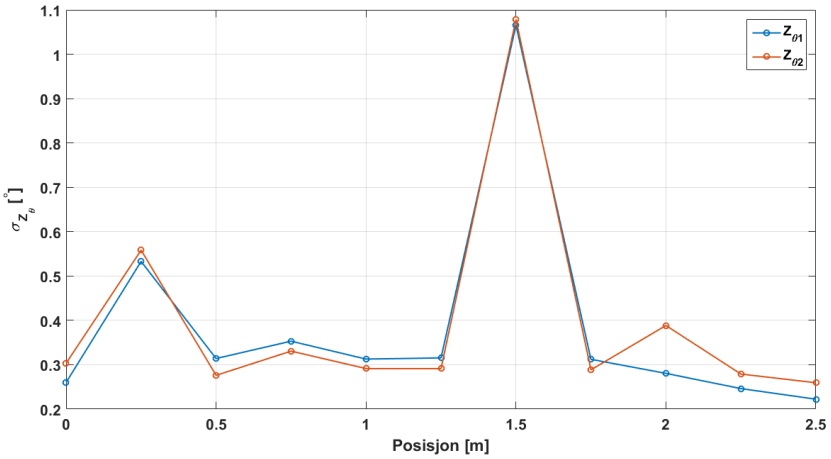
Figur 6.33: Beregnet σ_{A_θ}.

Figur 6.33 viser standardavviket til målingene i figur 6.32. Figuren viser tydelig at avviket ikke er konstant, men endres fra måling til måling. Standardavviket i dette eksperimentet synes å tilsvare standardavviket σ_{A_θ}, vist i figur 6.17.



Figur 6.34: Beregnet Z_θ.

Figur 6.34 viser den beregnede invarianten Z_θ. Fra denne figuren synes invariantene $Z_{\theta,1}$ og $Z_{\theta,2}$ å gjennspeile den horisontale forflytningen bra. Ved å sammenlikne invariantene synes de å ha samme form, men det kan observeres at $Z_{\theta,2}$ har en offsett på $\approx 20°$ i forhold til $Z_{\theta,1}$.



Figur 6.35: Beregnet σ_{Z_θ}.

Figur 6.35 viser standardavviket til målingene i figur 6.34. Fra figuren er det tydelig at avviket ikke er konstant, men endres fra måling til måling. Dersom ekstrempunktene ses bort fra, synes standardavviket i dette eksperimentet å tilsvare standardavviket σ_{Z_θ}, vist i figur 6.18.

6.3.2 Målinger på 60kV 15MW høyspentlinje fra Åna Sira til Titania AS

Figur 6.36: Fotografi av måleoppsett.

Fra fotografi i figur 6.36 vises måleoppsettet der eksperimentene ble utført. Lokasjonen til dette eksperimentet ble valgt på bakrunn av tilgjengelighet. En privat bilvei er anlagt nesten vinkelrett på fasene som skal måles. Veien er hovedsaklig plan og målingene vil dermed være målt på omtrent samme vertikale punkt. Dette tillater en enklere måling, med høyere presisjon.

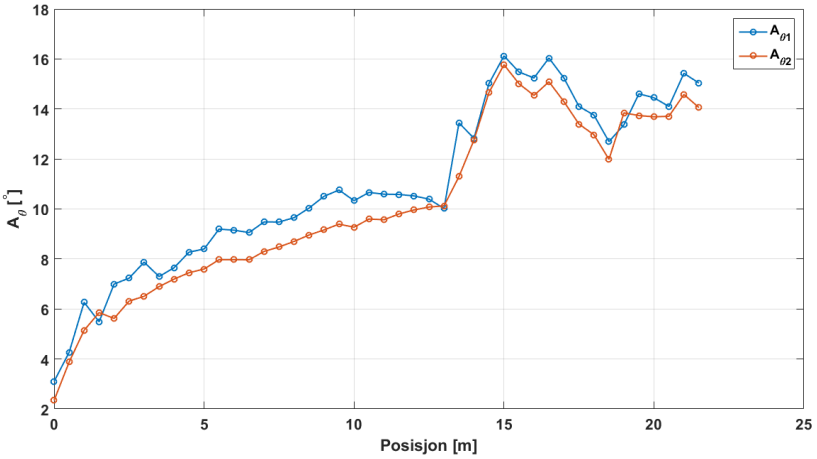
Fasestrømmene i masten blir ikke målt, men har et fast trekk på 150A under drift. Siden invariantene antas uavhengige av fasestrømmen, anses ikke dette kritisk for eksperimentet.

I eksperimentet er det valgt å måle magnetfeltet i 44 punkter, på tvers av høyspentmasten. Hver av punktene er forskjøvet med 50 centimeter i vertikal retning og det er forsøkt å opprettholde samme horisontale posisjon gjennom hele eksperimentet. Høyden fra sensor til faseledere ble estimert til omtrent 6 meter.

Målingene i dette eksperimentet kan deles inn i følgende 3 inndelinger:

1. **0.0m-12.5m** - Måleutstyret er på vei mot høyspentmast.
2. **13.0m-19.0m** - Måleutstyret er under høyspentmast.
3. **19.5m-21.5m** - Måleutstyret har passert høyspentmast.

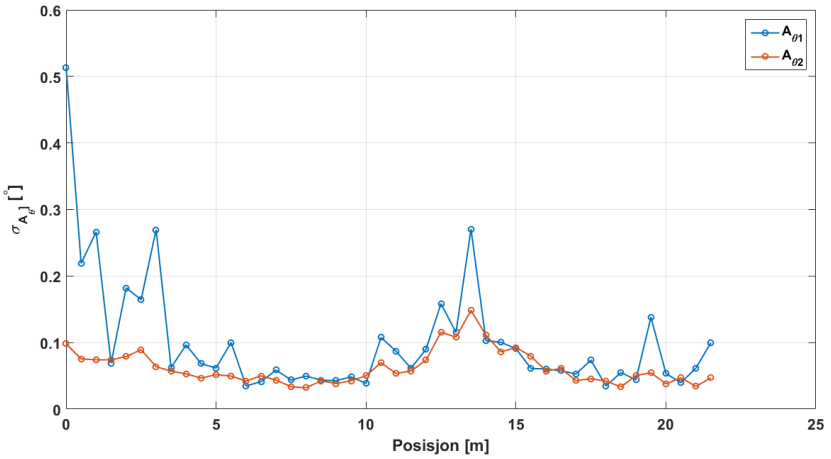
I analysene vil det fokuseres på de målinger som er målt i området der utstyret er direkte under høyspentmasten, det vil si området fra 13.0 meter til 19.0 meter.



Figur 6.37: Beregnet A_θ.

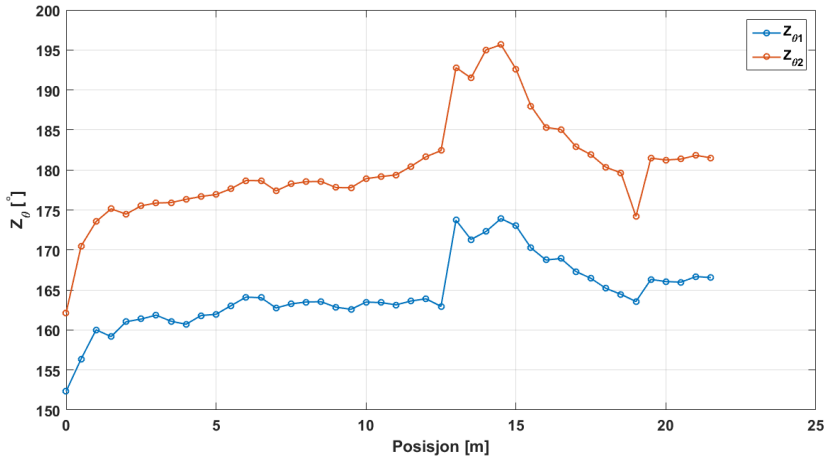
Figur 6.37 viser den beregnede A_θ. Fra denne figuren synes invariantene A_θ å gjennspeile den horisontale forflytningen bra. Selv magnetfelt med vertikal avstand på 13 meter og horisontal avstand på 6 meter synes å detekteres korrekt og gi stabile invarianter.

Ved å studere figuren nærmere er det tydelig at invarianten $A_{\theta,1}$ har en offsett i forhold til $A_{\theta,2}$. Størrelsen til offsetten synes å være av varierende karakter, men fra figuren kan det observeres at denne reduseres når målingene er nær høyspentmasten og kraftigere magnetfelt måles.



Figur 6.38: Beregnet σ_{A_θ}.

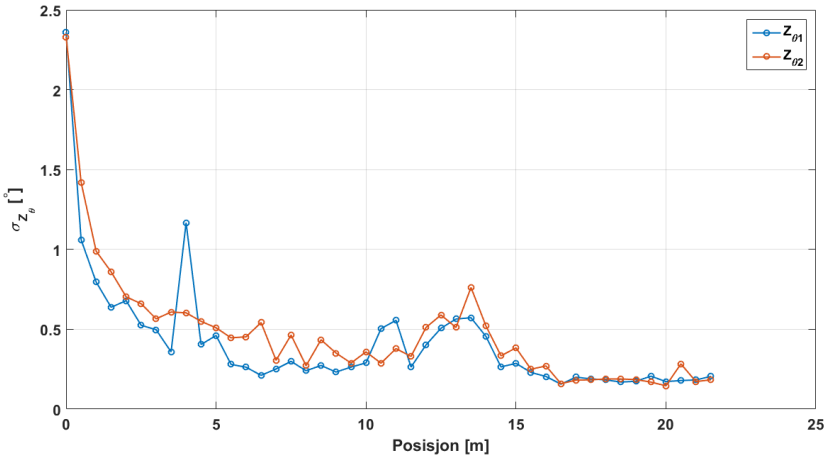
Figur 6.38 viser standardavviket til målingene i figur 6.37. Figuren viser tydelig at avviket ikke er konstant, men endres fra måling til måling. Standardavviket i dette eksperimentet synes å tilsvare standardavviket σ_{A_θ}, vist i figur 6.17.



Figur 6.39: Beregnet Z_θ.

Figur 6.39 viser den beregnede invarianten Z_θ. Fra denne figuren synes invariantene $Z_{\theta,1}$ og $Z_{\theta,2}$ å gjennspeile den horisontale forflytningen bra. Ved å sammenlikne invariantene synes de å ha samme form, men det kan observeres at $Z_{\theta,2}$ har en offsett på $\approx 20°$ i forhold til $Z_{\theta,1}$.

Ved å studere $Z_{\theta,1}$ i målepunktene 12.5 meter og 19 meter, er det tydelig at $Z_{\theta,1}$ har samme verdi i punktene. Ved å se på kategorien som disse punktene inngår i, finner vi at disse to tilhører overgangen fra å ikke være under høyspentmasten til å være under høyspentmasten. Dersom målepunktet i posisjonen 19.0m tilhørende invarianten $Z_{\theta,1}$, antas å være en målefeil, vil dette også være gyldig for $Z_{\theta,1}$.



Figur 6.40: Beregnet σ_{Z_θ}.

Figur 6.40 viser standardavviket til målingene i figur 6.39. Fra figuren er det tydelig at avviket ikke er konstant, men endres fra måling til måling. Dersom ekstrempunktene ses bort fra, synes standardavviket i dette eksperimentet å tilsvare standardavviket σ_{Z_θ}, vist i figur 6.18.

6.3.3 Konklusjon

Det første eksperimentet som ble utført beskrives i kapittel 6.3.1. Målet med dette eksperimentet var først å verifisere stabiliteten til målingnene, ved å måle gjenntatte ganger i samme punkt. Analysene viste noe variasjon i invariantene og tilhørende standardavvik. Men fra figur 6.29 og figur 6.31, er det allikevel tydelig at standardavviket tilsvarer det som ble målt i de innledende eksperimentene.

I det etterfølgende eksperimentet var målet å måle utslaget ved horisontal posisjonsendring. I analysen av dette eksperimentet ble det også her avdekket variasjoner i både invariantene og standardavviket. På tross av avvikene, synes invariantene å gjennspeile posisjonsendringen bra. Resultatene fra analysen synes å tilsvare det som ble observert i de innledende eksperimentene.

Det neste eksperimentet som ble utført, beskrives i kapittel 6.3.2. I dette eksperimentet var målet å måle magnetfeltet i flere punkter over en lengre horisontal strekning en hva som tidligere har vært mulig. Magnetfeltet kunne dermed logges med en betydelig avstand fra faseledere. Dette ble benyttet for å få større innsikt i utviklingen til invariantene og tilhørende standardavvik, ved en glidende horisontal bevegelse. Fra analysen synes invariantene å representere den horisontale forflytningen bra. Selv med en diagonal avstand på $D = \sqrt{13^2 + 6^2} \approx 14$ meter. Ved å studere standardavviket i figur 6.38 og figur 6.40, er det tydelig at standardavviket til invariantene reduseres når målingene utføres i et sterkere magnetfelt. Fra denne observasjonen kan det konkluderes med at standardaviket synes å ha sammenheng med målt magnetfeltstyrke. Standardavvikene i nærheten av høyspentmasten synes å tilsvare standardavvikene funnet i de innledende eksperimentene.

Kapittel 7

Diskusjon

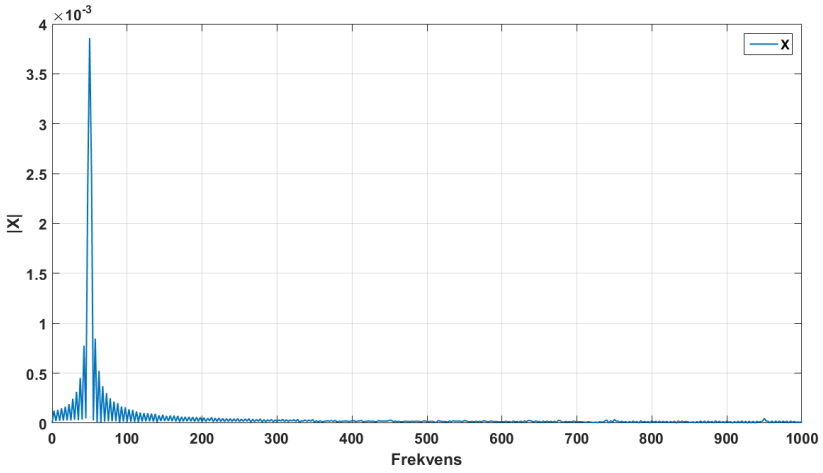
Dette kapittelet omhandler prosjektets forløp og gjennomføring. Resultater fra eksperimenter vil bli drøftet i kapittel 7.1. Videre vil det i kapittel 7.2 gis en trinnvis vurdering av arbeidet relatert til de ulike kapitlene i rapporten. Avslutningsvis vil planer for videre arbeid presenteres.

7.1 Resultater

Eksperimentene utført i dette prosjektet har alle bidratt til å underbygge kvaliteten til målesystemet. De viktigste resultater fra eksperimentene skal her oppsummeres.

Det ble valgt å utføre noen innledende eksperimenter før målinger på høyspentmast startet. Fra måling av statisk magnetfelt ble presisjonen til sensornoden i stor grad bekreftet. Det ble i dette eksperimentet funnet et avvik mellom beregnet og målt magnetfelt på $0.02\mu T$. Det ble argumentert for at dette avviket trolig var resultat av unøyaktigheter og presisjon i posisjonsmålinger og ikke selve magnetfeltmålingen.

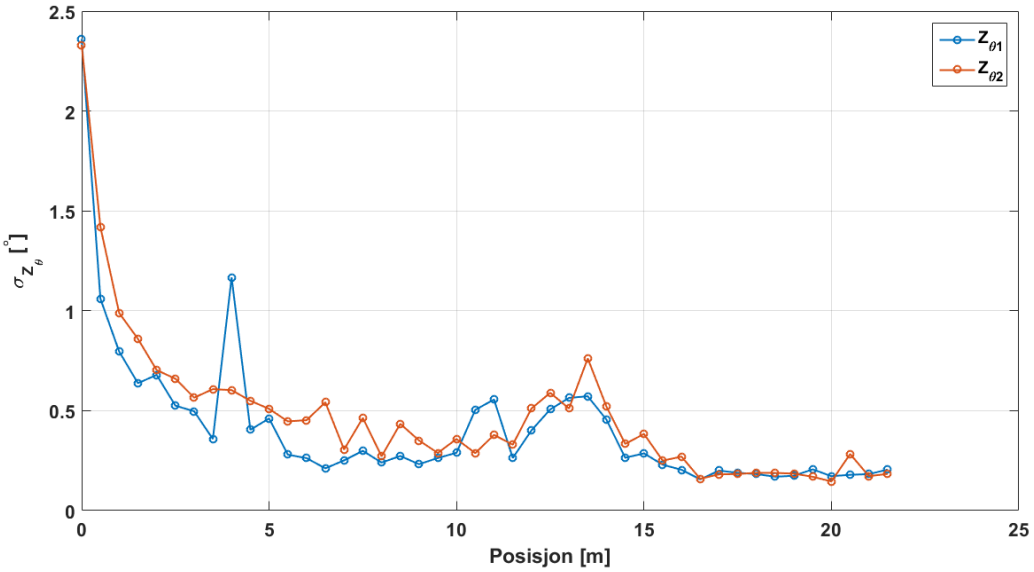
Videre fortsatte de innledende eksperimentene med å verifisere at designet sensornode er i stand til å måle et tidsvarierende magnetfelt. Også dette ble bekreftet og eksperimentet fortsatte med å utføre en frekvensanalyse av målte signaler. Figur 7.1 viser frekvensanalysen til x-aksen ved 8.6A last. Fra denne analysen, ble det konkludert med at det ikke er behov for ytterligere filtrering av de målte signalene.



Figur 7.1: Frekvensanalyse av målt signal ved 8.6A last.

De innledende eksperimentene ble avsluttet med å analysere utviklingen og standardavviket til invariantene A_θ og Z_θ. Fra eksperimentene ble det tydelig at invariantene ikke var så konstante, som det hevdes i [25]. Fra analysene ble det også litt uklart hvorvidt invariantene faktisk er uavhengig av fasestrøm.

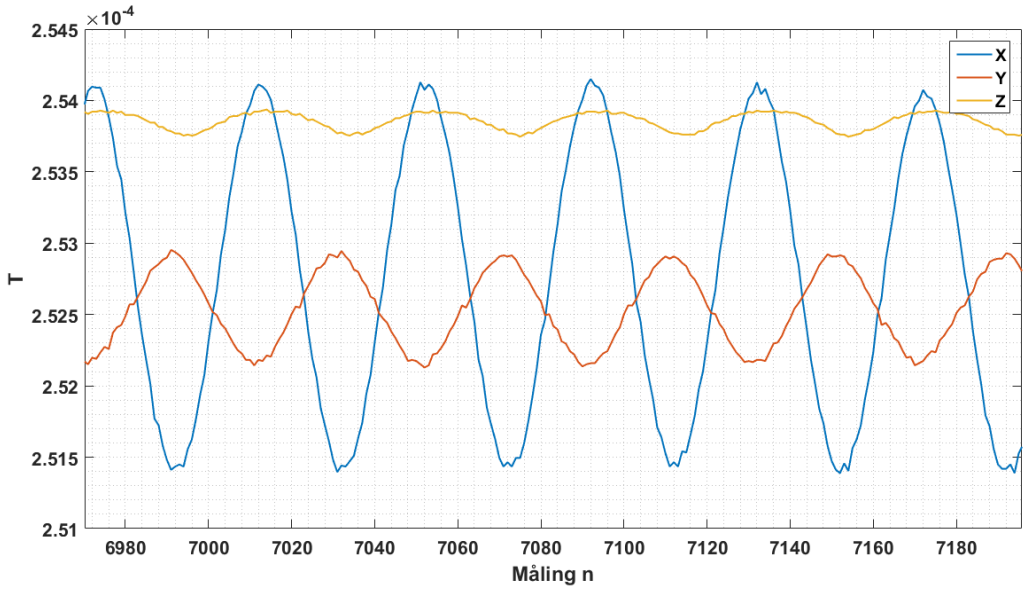
Ved videre eksperimenter ble det tydeligere at invariantene i stor grad kunne benyttes til å estimere posisjonen i et magnetfelt. Men det ble også klart at standardavviket til invariantene hadde stor knyttning til styrken av magnetfeltet og dermed fasestrømmen. Dette blir svært tydelig i figur 6.40 fra kapittel 6.3.2. Denne figuren anses som så viktig at den repeteres nedenfor.



Figur 7.2: Beregnet σ_{Z_θ}.

Fra figuren vises beregnet σ_{Z_θ} for eksperiment beskrevet i kapittel 6.3.2. Den viktigste konkusjonen som kan trekkes fra denne figuren, er at standardavviket er størst lengst bort fra

høyspentmasten og reduseres i nærhet av kraftigere magnetfelt. Denne observasjonen antyder at troverdigheten til de beregnede invariantene er proporsjonal med styrken til det målte magnetfelt.



Figur 7.3: Magnetfeltmålinger i ett målepunkt med uventet høyt avvik.

For å kunne si noe mer om de punktene som viste uventet høyt avvik, ble måledata analysert. Fra denne analysen og figur 7.3, er det tydelig at også z-aksen påtrykkes et magnetfelt. Dette antas å ha opphav i at sensornodens akser ikke er parallelle med høyspentmastens ledere. Magnetfeltet avbildes dermed ikke korrekt på sensorens akser og det antas at dette er årsaken til at noen målepunkter har uventet høyt avvik fra nabopunktene. For å rette dette kreves det at målingene kompenseres for orientering både i rommet og i forhold til faseledere.

7.2 Prosjektforløp

Det skal her gis en oppsummering over prosjektets forløp og hovedaktiviteter. Oppsummeringen vil være relativt overordnet og det anbefales å studere tilhørende kapittler for økt innsikt i detaljer.

7.2.1 Oppstart

I Januar startet prosjektet med å tilegne nødvendig kunnskap omkring magnetfeltteori. For å tilegne denne kunnskapen ble det startet med å detaljstudere masteroppgaven av Kristian S.Stangeland [25], med tilhørende referanser. Parallellt med dette arbeidet, ble det utviklet magnetfeltsimuleringer i MATLAB. Detaljer omkring simuleringer kan studeres i kapittel 2.3.

Resultatenne fra simuleringene dannet en oversikt over magnetfeltstyrkene som kan befinne seg rundt en høyspentmast og forventet endring ved forflytning. Dataene fra simuleringene var med på å danne et viktig grunnlag for valg av magnetfeltsensor.

I oppstartsfasen til prosjektet ble også en overordnet plan for design av målesystemet etablert.

7.2.2 Vurdering av sensorteknologi

For å komme frem til en magnetfeltsensor som er i stand til å måle tidsvarierende magnetfelt med sterkt varierende styrke, var det sentralt å definere konkrete krav til sensor. De mest sentrale krav til egenskaper var at magnetfeltsensorsensor måtte **måle vektorfelt**, inneha tilstrekkelig **båndbredde**, **måleområde** og **sensitivitet**. Det var også et viktig kriterie at magnetfeltsensoren var designet og produsert med hensyn på tilsvarende målesystem og krav. Sistnevnte krav, silte raskt ut magnetfeltsensorer som er hovedsaklig designet for underholdningsprodukter som mobiltelefon eller nettbrett. Detaljer omkring krav og vurderinger er plassert i kapittel 3.

Basert på vurdering og argumentasjon, ble magnetfeltsensoren HMC2003 [13] valgt. Som konsekvens av dette valget ble det nødvendig å revidere den opprinnelige overordnede planen over sensornoden. Målesystemet som nå skulle designes, ble basert på analoge signaler i stedet for en digital buss mellom mikrokontroller og magnetfeltsensor. For å opprettholde presisjon i magnetfeltsignaler fra sensor til mikrokontroller krevdes det et langt mer komplekst målesystem der alle valg må fokuseres på de analoge signalene.

7.2.3 Kravspesifikasjon for maskinvare

Innen valg av komponenter som skulle inngå i kretskortet kunne påbegynnes, var det nødvendig å revidere den overordnede planen for magnetfeltsensoren og målesystemet. Oppdatert plan for sensornoden kan studeres i figur 1.2.

Gjennom hele prosessen med å velge komponenter, var hovedfokuset å finne kvalitetskomponenter med egenskaper som lav støy og strømforbruk. Dette ble et omfattende arbeid og krevde betydelig mer tid en hva som først var tenkt. De ulike komponentene ble vurdert ut i fra datablad og applikasjonsnoter og deretter sammenliknet på bakgrunn av dette. Detaljer og argumentasjon vedrørende komponentvalg kan studeres i kapittel 4.1.1 til kapittel 4.1.7.

7.2.4 Utvikling av kretskort basert på kravspesifikasjon

Etter at komponentene som inngår i sensornoden var valgt, startet arbeidet med å tegne skjema og endelig utlegg. Til dette arbeidet ble det valgt å benytte det vederlagsfrie utleggsprogrammet DesignSpark PCB fra RS Components.

Siden det på forhånd var utarbeidet en detaljert oversikt over hvilke komponenter som skulle inngå i kretskortet til sensornoden, gikk arbeidet med å tegne skjema smidig. Den eneste reelle utfordringen som uventet dukket opp her, var at det i datablad [13] tilhørende sensoren

HMC2003, manglet detaljert beskrivelse av kretsens mekaniske dimensjoner. Det ble gjort utallige forsøk på å få oppgitt denne informasonen fra sensorprodusent Honeywell, uten at det førte frem. Til slutt fikk jeg oppgitt de mekaniske dimensjonene av en bruker på et internettforum. Detaljer rundt skjema finnes i kapittel 4.1.8.

Arbeidet med å legge ut kretskortet ble imidlertid en langt større utfordring en først antatt, basert på opprinnelig plan. Utlegget som vises i figur 4.54, består av 4 lag. Ytre lag er i hovedsak benyttet til signalbaner og de to indre lagene er designet med 3 positive forsyninger og 2 ulike jordpotensialer. I tillegg til dette krevde kretsen MAX11047 [21] et eget avkoblingslag for å kunne oppnå gitt presisjon. Igjennom hele prosessen med å designe kretskortet ble det fokusert på støy og hvordan den digitale delen av kretskortet i minst mulig grad skulle påvirke de analoge signalene. Detaljer rundt utlegg finnes i kapittel 4.1.9.

Ferdig designet kretskort ble levert til Elprint AS for produksjon.

7.2.5 Utvikling av programvare

For utvikling av programvare til mikrokontrolleren er det vederlagsfrie utviklingsverktøyet Coocox benyttet, sammen med den anbefalte kompilatoren *GCC-ARM-Embedded*.

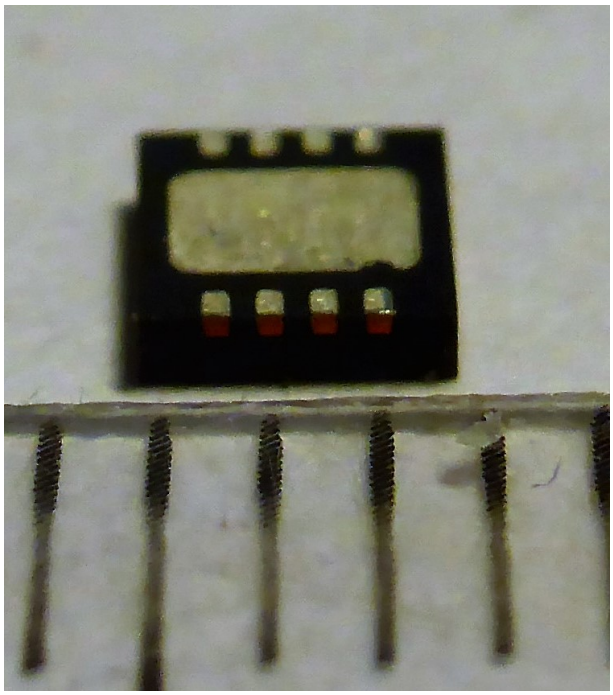
Parallellt med utviklingen av kretskortet, ble en overordnet skisse av programvaren utviklet. Skissen forkortet programmeringsprosessen betraktelig ved at det allerede forelå en god arbeidsplan når utviklingen av programvaren startet.

Programvaren ble i første omgang basert kun på den overordnede skissen. Dette på grunn av manglende kretskort å teste ut funksjoner på. Etter kretskortene var ferdig produsert og kretser montert, ble programvaren testet og feil rettet. Det viste seg raskt at spesielt overgangen fra ARM®Coretex®-M3 til ARM®Coretex®-M0 resulterte i uventede utfordringer. Spesielt vil jeg fremheve SPI-kontrolleren, som på tross av at en 1byte ble lastet inn i 1byte register for sending, ble byte verdien sendt som 16 bit. Problemet ble løst ved å benytte CMSIS metoden for å sende data og kommandoer over SPI-buss.

For detaljer angående programvaren, henvises det til kapittel 5.

7.2.6 Montering og testing

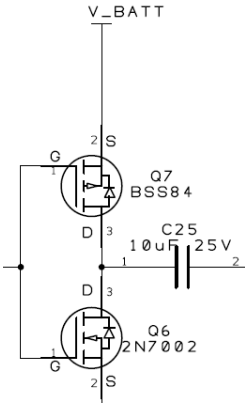
Montering av kretskort ble i hovedsak utført på lab tilhørende systemkonstruksjon ved UiS. På denne labben er det tilgjengelig det meste av utstyr som behøves for å bestykke opp kretskort med overflatekomponenter. Prosessen gikk hovedsaklig smidig, men det ble i løpet av prosessen oppdaget designfeil på kretskortet.



Figur 7.4: Lineærregulator ADM7150, linjal viser avstand i mm.

Figur 7.4 viser lineærregulatoren ADM7150 med feilvalgt pakke. Som figuren viser er valgt krets svært liten og komplisert å lodde på kretskortet. Selv om kretsen er på grensen av hva som lar seg montere uten spesialutstyr, gikk lodeprosessen overraskende bra.

Under test av det monterte kretskortet, ble de ulike funksjoner verifisert at de virket som tenkt. Det ble også i testprosessen avdekket en designfeil som resulterte i kortslutning i transistor Q7.



Figur 7.5: Utklipp fra skjema.

Designfeilen som ble avdekket, var at transistor Q7 manglet seriemotstand og følgelig er transistoren da i stand til å trekke en høyere strøm en hva den er oppgitt å tåle. Ved en uheldig feil i programvaren, var transistoren aktiv i for lang tid og dette medførte at overgangen *source-drain* kortsluttet. For å korrigere feilen ble det besluttet å fjerne transistor Q7 og erstatte den med en motstand mellom *source-drain*. Endringen resulterte i et fullt fungerende design.

7.2.7 Eksperimenter

For å verifisere at måleutstyret virket som tenkt, ble en rekke eksperimenter utført. De første innledende eksperimentene var hovedsaklig ment for å verifisere presisjon og at utstyret er i stand til å måle både statiske og tidsvarierende magnetfelt. Gjennom de ulike innledende eksperimentene, ble presisjonen i stor grad bekreftet. De etterfølgende innledende eksperimentene var ment på å verifisere hvorvidt invariantene oppfatter en posisjonsendring i både horisontal og vertikal retning. I analysene ble det lagt stor vekt på standardavviket til inavriantene.

Etter de innledende eksperimentene var gjennomført, startet målingene av magnetfelt fra ulike høyspentmaster. Noen av eksperimentene ble utført i relativt krevende terreng og derfor må det forventes reduksjon i presisjon til målinger. Felles for disse eksperimentene, var at det ikke var mulig å verifisere hvorvidt invariantene faktisk er uavhengig av fasestrømmen. Dette på grunn av at det på ingen måte lar seg gjøre å tilpasse fasestrømmen i en høyspentmast til eksperimentet som ønskes utført. I eksperimentene ble det fokusert på hvorvidt invariantene oppfatter en posisjonsendring i både horisontal og vertikal retning og det ble også her lagt stor vekt på standardavviket til inavriantene.

For detaljer angående utførte eksperimenter, henvises det til kapittel 6.

7.2.8 Veiledning

I dette prosjektet har det ikke blitt avholdt veiledningsmøter til fastsatte tider. I stedet er det på kort varsel avtalt møter, dersom det var ønskelig med en diskusjon eller statusoppdatering.

Dette har fremmet fleksibiliteten og har fungert bra for alle parter i prosjektet.

7.3 Forslag til videre arbeid

Gjennom prosjektets forløp er det avdekket flere forbedringer i forhold til valgte løsninger. Forbedringene kan i fremtiden bidra til å forbedre valgt design. Som følge av prosjektets omfang, ble det heller ikke anledning til å utvikle alle komponenter som det var planlagt at skulle inngå i målesystemet. Forslag til videre arbeid kan oppsummeres i følgende punkter:

Mikrokontroller - Det ble i kapittel 4.1.6 valgt å benytte en mikrokontroller basert på kjernen ARM®Cortex®-M0. Dette valget var basert på at mikrokontrolleren kun skulle hente målinger fra ADC og sende de målte verdier ubehandlet over CAN-buss. I ettertid har dette valget vist seg å medføre nesten uønsket høy trafikk på CAN-buss. Ved å velge en mikrokontroller som er i stand til å beregne frekvensanalyse og invarianter lokalt, åpnes muligheter for høyere sampling av magnetfeltsignaler og lavere last på buss.

Forslag til ny mikrokontroller er STM32F303 [29], med kjerne ARM®Cortex®-M4.

Prosesseringsnode - På grunn av arbeidsmengden i dette prosjektet var det ikke anledning til å utvikle prosesseringsnoden. I et komplett målesystem må denne noden utvikles.

Korrigere målinger for orientering - For å oppnå ønsket presisjon i målingene er det en forutsetning at aksene til sensor er parallelle med faselederenes akser. I [25] ble det vist at kryssproduktet mellom to vektorer kan benyttes for å beregne korrigeringen. På grunn av prosjektets omfang, har det ikke vært anledning til å se nærmere på dette.

Posisjonsestimering ved hjelp av Kalman-filter - Kalman-filter er en forholdsvis lett estimeringsalgoritme, som er godt egnet for posisjonsestimering i dette målesystemet. For å kunne benytte denne algoritmen, må imidlertid invariantligningene tilpasses og konverteres til tilstandsrommodell. Arbeidet med dette er påbegynt, men ikke fullført. I en videre utvikling av målesystemet synes det naturlig å fullføre dette arbeidet.

Brukergrensesnitt - I dette prosjektet er det utviklet et brukergrensesnitt som innehar de mest sentrale funksjoner som trengs for å kontollere og hente ut måledata fra målesystemet. Dette grensesnittet bør videreutvikles til et mer robust program, med flere funksjoner og mulighet for utvidede rapporteringer.

Kapittel 8

Konklusjon

Det er i denne oppgaven utviklet og konstruert et målesystem for å måle tidsvarierende magnetisk feltstyrke, generert av en høyspentmast. Basert på det målte magnetfeltet blir invariantene A_θ og Z_θ beregnet. Invariantene vil videre danne grunnlaget for posisjonsestimering i magnetfeltet.

I utviklingen av magnetfeltsensoren ble det valgt å benytte magnetfeltsensoren HMC2003. Dette valget innebar at grensesnittet mellom mikrokontrolleren og magnetfeltsensoren gikk fra å være basert på digital kommunikasjon, til et rent analogt grensesnitt. For å opprettholde ønsket presisjon i det analoge grensesnitte, medførte dette et betydelig mer kompekst design enn hva som innledningsvis var skissert. Det ble videre valgt en 4 kanal parallell 16bit ADC, basert på SAR teknologi. Fra magnetfeltsimuleringer fremkom det tydelig at 16bit var for grov oppløsning. Ved å inkludere en flerkanals DAC og instrumentforsterkere ble det valgt å trekke fra 90% av det forrige samplede signalet sin verdi, for så å forsterke opp det resterende signalet med inntil 11 ganger. Forsterkningen til instrumentforsterkerene ble tilpasset tilgjengelige komponentverdier og endte tilslutt på 10.89 ganger. Resultatet av dette var at laveste spenning som kunne måles gikk fra $76.3\mu V$ til $6.8\mu V$, med en effektiv oppløsning på 19bit. Laveste magnetfelt som sensornode er i stand til å måle, er $680pT$. Fra analysene ble dette verifisert til å være en tilstrekkelig oppløsning for å tydelig kunne skille mellom de laveste feltstyrker.

For å analysere presisjonen til målesystemet er det utført ulike eksperimenter. De viktigte resultater og observasjoner fra eksperimentene skal her belyses.

Presisjonen til målesystemet ble i de innledende eksperimentene bekreftet ved å måle et statisk magnetfelt. Avviket mellom målt og beregnet magnetfelt ble beregnet til $0.02\mu T$. Det antas at dette avviket er et resultat av presisjonen til den fysiske målingen og ikke fra sensor. Videre ble det utført målinger og frekvensanalyse av et tidsvarierende magnetfelt. Basert på frekvensanalysen ble det konkludert med at det ikke var behov for digital filtrering av signalene.

Det er gjennom eksperimentene utført med målesystemet, verifisert at invariantene i høy grad avbilder en posisjon i et magnetfelt. Fra analysene ble det litt uklart hvorvidt invariantene er uavhengige av fasestrømmen i høyspentmastene. Ved grundigere analyser ble det klart at standardavviket til invariantene har sammenheng med styrken på det målte magnetfeltet. Tro-

verdigheten til invariantene er proporsjonal med styrken på det målte magnetfeltet og dermed også den faktiske fasestrømmen. Forventningsverdi og tilhørende standardavvik til eksperimentet fra kapittel 6.3.2 oppsummeres i tabell 8.1. Tabellen viser at i det aktuelle eksperimentet kan det forventes et standardavvik tilhørende de beregnede invariantene på under 0.01°.

	μ_{A_θ}		σ_{A_θ}		μ_{Z_θ}		σ_{Z_θ}	
Måleposisjon	$A_{\theta,1}$	$A_{\theta,2}$	$A_{\theta,1}$	$A_{\theta,2}$	$Z_{\theta,1}$	$Z_{\theta,2}$	$Z_{\theta,1}$	$Z_{\theta,2}$
På vei inn	8.5140	7.5970	0.1128	0.0949	161.7992	176.6382	0.5309	0.6088
Under	13.8740	13.2280	0.0949	0.0787	169.1225	187.1646	0.3061	0.3478
På vei ut	14.3899	13.9105	0.0674	0.0449	165.7166	180.0214	0.1829	0.1911

Tabell 8.1: Forventningsverdi og standardavvik til invariantene ved måleposisjoner definert i kapittel 6.3.2. Verdier gitt i grader.

Basert på tabell 8.1, konkluderes det med at invariantene bør være velegnet for orientering og navigasjon i et magnetfelt. Algoritmer som Kalman-filter og CMA-ES vil med stor sikkerhet kunne benyttes for å estimere faktisk posisjon. I denne oppgaven har det derimot ikke vært tilgjengelige ressurser for å studere dette nærmere og det foreslås som et punkt til videre arbeid.

Fra analyser av målt magnetfelt i punkter med uventet avvik, er det avdekket at også z-aksen påtrykkes og måler et tidsvarierende magnetfelt. Dette antas å ha opphav i at sensornodens akser ikke er helt parallelle med høyspentmastens ledere under målingen. Magnetfeltet avbildes dermed ikke korrekt på sensorens akser og det antas at dette er årsaken til at noen målepunkter har uventet høyt avvik fra nabopunktene. For å korrigere dette kreves det at målingene kompenseres for orientering både i rommet og i forhold til faseledere. Arbeidet med å implementere dette i praksis er også foreslått som et punkt for videre arbeid.

Tillegg A

CAN-meldinger

Magnetfeltdata	
Melding ID [StdID]	0x000
Data lengde [DLC]	6byte
Tx frekvens	2kHz
$[D0 : D1]$	X-akse
$[D2 : D3]$	Y-akse
$[D4 : D5]$	Z-akse
Målt magnetfeltdata.	

Tabell A.1: CAN melding, magnetfeltmålinger

Kanalforsterkning	
Melding ID [StdID]	0x010 - x-akse
Melding ID [StdID]	0x020 - y-akse
Melding ID [StdID]	0x030 - z-akse
Data lengde [DLC]	6byte
Tx frekvens	0Hz
$[D0 : D3]$	Målt adcverdi
$[D4 : D5]$	Dac spenning
Meldingen inneholder 128 målinger av adc-kanalen og dac verdi. Danner grunnlaget for å beregne faktisk forsterkning. Faktisk forsterkning beregnes ved $$E[G(k)] = \frac{1}{N}\sum_{n=0}^{N} G(k,n)$$ Hvor G er gitt av: $$G(k,n) = \frac{1}{128 \cdot \Delta V_{DAC}} \cdot \big(V_{ADC}(k,n) - V_{ADC}(k,n-1)\big)$$	

Tabell A.2: CAN melding, kanalforsterkning

Målte spenninger	
Melding ID [StdID]	0x040
Data lengde [DLC]	6byte
Tx frekvens	0Hz
$[D0 : D1]$	Målt V_{batt}
$[D2 : D3]$	Målt V_{Ref}
$[D4 : D5]$	Målt $+V_{SR}$
Melding består av målte spenninger til sentrale deler av sensornoden. Basert på denne meldingen kan mottaker reagere dersom spenninger blir for lave.	

Tabell A.3: CAN melding, målte spenninger

Tildeling av CAN-adresse	
Melding ID [StdID]	0x060
Data lengde [DLC]	1byte
Tx frekvens	0Hz
$[D0]$	Ny CAN-adresse
Tildeler unik CAN-adresse til noden som mottar meldingen.	

Tabell A.4: CAN melding, tildele CAN-adresse

Oppdater kanalforsterkning	
Melding ID [StdID]	0x080
Data lengde [DLC]	5byte
Tx frekvens	0Hz
$[D0]$	Kanal
$[D1 : D2]$	Teller
$[D3 : D4]$	Nevner
Beregnet kanalforsterkning.	

Tabell A.5: CAN melding, oppdater kanalforsterkning

Start og stopp magnetfeltmålinger	
Melding ID [StdID]	0x050
Data lengde [DLC]	1byte
Tx frekvens	0Hz
$[D0]$	Flagg start/stopp
Start/stoppflagg som setter tilstanden i sensornoden. D0 = 1 - Start målinger. D0 = 0 - Stopp målinger	

Tabell A.6: CAN melding, start og stopp av magnetfeltmålinger.

Flagg, kalibrer kanalforsterkning	
Melding ID [StdID]	0x070
Data lengde [DLC]	1byte
Tx frekvens	0Hz
$[D0]$	Flagg start/stopp
Flagg som starter kalibrering av forsterkning. D0 = 1 - Start kalibrering	

Tabell A.7: CAN melding, flagg kalibrer kanalforsterkning

Flagg, oppdater offsetberegning	
Melding ID [StdID]	0x090
Data lengde [DLC]	1byte
Tx frekvens	0Hz
$[D0]$	Flagg
Flagg starter ny offestberegning. D0 = 1 - Start beregning	

Tabell A.8: CAN melding, flagg start offsetberegning

Tillegg B

Deleliste Sensornode

1		2.54mm		T1	2204298
1		5.08mm		T2	4258720
3	Polyfuse 0.1A		Littelfuse	FS1 ,FS2,FS3	7874211
5	60.4r 1%	0603	Vishay	R3,R4,R24,R31,R32	6790598
1	100r 1%	0603	Te Connectivity	R9	2132143
4	4.99k	0603	Panasonic	R5,R13,R14,R15	7087644
3	9.09k 0.1%	0603	Panasonic	R25,R26,R27	
9	10k 0.1%	0603	Panasonic	R8,R12,R16,R18,R19,R20,R21,R22,R23	566890
5	10k 1%	0603	Vishay	R1,R2,R6,R28,R33	6789667
3	25k 0.1%	0805	Te Connectivity	R10,R29,R30	8667355
1	34.8k 0.1%	0603	Panasonic	R7	7088886
2	86.6k	0603	Panasonic	R11,R17	7089007
2	1uF 50V X5R	2012	TDK	C17,C21	9039091
3	2n7002	SOT-23	Fairchild	Q5,Q6,Q8	6710312
2	22pF 50V COG	0603	Murata	C6,C7	6242351
2	68pF 50V COG	0603	Murata	C3,C5	6242373
1	4.7nF 50V X7R	0603	AVX	C4	6983273
3	6.2nF 50V COG	0805	Murata	C27 ,C29,C30	8202829
3	33nF 25V COG	1206	TDK	C32,C33,C34	7407455
36	100nF 50V X7R	0603	AVX		6983260
1	220nF 50V X7R	1206	AVX	C38	6983689
1	4.7uF 50V X7R	2220	TDK	C23	7416878
8	10uF 10V X7R	0805	Murata	C18,C19,C20,C22,C65,C66,C68,C76	8202780
1	10uF 16V X5R	2012	TDK	C16	9039088
2	10uF 16V Tantal		AVX	C1,C2	4070019
2	10uF 25V X5R	1206	TDK	C25,C26	9039151
6	22uF 16V X7R	1210	Kemet	C8,C9,C10,C11,C13,C15	6983829
1	47uF 6.3V X5R	1210	Kemet	C12	6911262
2	100uF 6.3V Tantal		Vishay	C14,C75	6845039
1	100uF 16V Tantal		AVX	C35	4648072
1	18uH 0.98A		Wurth	L2	8007961
1	22uH 3A		Wurth	L1	7896962
3	AD8226ARMZ		Analog Devices	IC6,IC7,IC8	7591288
6	BAT165E6327		Infineon	D1,D2,D3,D4,D5,D6	8270131
1	LM1117MP-5.0NOPB	4-Pin SOT-2231		IC13	5339583
3	MAX7480ESA+	SOIC-8	Maxim	IC9,IC10,IC11	7998137
1	SFH6156-2T	SO41		IC14	2841190
1	TSV321RILT	5-Pin SOT-231		IC16	6247952
1	XTAL 8MHz	HC49S		X1	5476452
1	STM32F072CBT6	LQFP-48	STMicroelectronics	IC5	8805392
1	TJA1052IT-2Y	SOIC-16	NXP	IC12	
2	MAX1837EUT33+T	SOT-23-6	Maxim	IC2,IC3	
1	MAX5714AUD+-ND	14-TSSOP	Maxim	IC17	
1	BSS84P		NXP	Q7	
1	IRF7343PBF	SOIC-8	International Rectifier	Q1	5411770

Tabell B.1: Deleliste.

Tillegg C

Programvare

C.1 Sensornode

```
1   /**
2    * @file main.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Mainfila.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   * 1. Legge til beregning av offsett. Sjekk dokumentasjon
17   */
18
19  /*-------------------------------------------------------------------------------------------
    ------------
20                                      Inkluderinger
21  -------------------------------------------------------------------------------------------
    ----------*/
22  #include "stm32f0xx.h"
23  #include "system_stm32f0xx.h"
24  #include "stm32f0xx_rcc.h"
25  #include "stm32f0xx_gpio.h"
26  #include "stm32f0xx_adc.h"
27  #include "adc.h"
28  #include "dac.h"
29  #include "globDef.h"
30  #include "avbrudd.h"
31  #include "hmc2003.h"
32  #include "watchdog.h"
33  #include "flash.h"
34  #include "can.h"
35  #include "clkSync.h"
36  /*-------------------------------------------------------------------------------------------
    ------------
37                                      Variabler
38  -------------------------------------------------------------------------------------------
    ----------*/
39  volatile struct flagg statusFlagg;
40  volatile struct tid systemTid;
41  /*-------------------------------------------------------------------------------------------
    ------------
42                                      Funksjoner
43  -------------------------------------------------------------------------------------------
    ----------*/
44  /*****************************************************************************
45   * @brief Setter flagg og variabler til bestemte tilstander/verdier.
46   * @param  none.
47   * @return none
48   *****************************************************************************/
49  void initierSysVariabler( void ){
50      uint8_t teller = 0;
51      /* Hent inn data fra flash,
52       * dersom disse ikke er initiert kalles funksjon som setter standardverdier */
53      flashLagring.antallParametere = _ANT_PARAM_I_FLASH_;
54      flashLagring.baseAdresse = _FLASH_BASE_ADR_;
55      flashLagring.oppdatert = FALSE;
56      initFLASHVariabler( &flashLagring );
57
58      for( teller = 0; teller < (ADC_Kanaler - 1); teller++){
59          ekstADC[teller].offsett = 0;
60          ekstADC[teller].G_teller = 109;//flashLagring.param[ 2*teller ];
61          ekstADC[teller].G_nevner = 10; //flashLagring.param[ 2*teller + 1 ];
62      }
63
64      internADC.kanal[0] = adcVbatt;
65      internADC.kanal[1] = adcVsetReset;
66      statusFlagg.adcModus = FLAGG_ADCMODUS_Pause; /* Målinger startes via can kommando */
67  }
68
```

```
69
70  /*****************************************************************************
71   * @brief Hovedfunksjon i programmet. Variabler og moduler initieres før den uendelige
72   *          løkken startes.
73   * @param  none.
74   * @return none
75   *****************************************************************************/
76  int main(void){
77      /* Lokale variable */
78      uint16_t tull = 0;
79      uint16_t tull2 = 0;
80      uint8_t teller = 0;
81      uint8_t tabell[4];
82      uint32_t tidTilNesteSek = 0;
83      uint8_t tidTilLesInternADC = 0;
84      SystemInit ();
85
86      /* Initierer variabler */
87      initierSysVariabler( );
88      flashLagring.param[ _FLASH_PARAM_CAN_ID_ ] = CAN_DEFAULT_ADR;
89      /* Initierer moduler */
90      initEksternDAC();
91
92      initierEksternADC();
93      initSysTick( SYSTICK_250us );
94      initCAN( );
95      initCLKSync();
96
97      initierInternADC();
98
99      /* Initierer pinner og berergner offsettspenning */
100     initHmc2003();
101
102     statusFlagg.pulseVSR = FALSE;
103
104     statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
105     statusFlagg.adcModus = FLAGG_ADCMODUS_Run;
106     statusFlagg.hmc2003SR = FLAGG_HMC2003_Normal;
107
108     GPIOC->BSRR = GPIO_Pin_15;
109     initWatchDog( _PRESCALER_DIV_16_ );     /* Watchdog frekvens 2.5kHz, teller ned
        fra 10000
110                                                 Gir resett etter 4 sekund uten
                                                    kvittering    */
111
112     internADC.velgNesteKanal = FALSE;
113     statusFlagg.internAdc = FALSE;
114     internADC.kanalIdx = 0;
115     while(1){
116         _klappHunden_
117
118         /* Holder orden på systemklokka */
119         if( systemTid.sysTeller >= tidTilNesteSek ){ /* 1 sekund */
120
121             tidTilNesteSek += 3999; /* Legger til ett sekund, eventuell forsinkelse
                vil da ikke påvirke
122                                        systemtiden men kun en forsinkelse i
                                           oppdatering av sekund     */
123             systemTid.sekunder++;
124             systemTid.sensorReset++;
125             tidTilLesInternADC++;
126
127
128             //statusFlagg.pulseVSR = (statusFlagg.pulseVSR==TRUE? FALSE:TRUE);
129             if( flashLagring.param[ _FLASH_PARAM_GAIN_Kalibrert_ ] == FALSE){
130                 beregneGain();
131             }
132
133             /* Sender can melding dersom sensor ikke har unik canbus id */
134             if( flashLagring.param[ _FLASH_PARAM_CAN_ID_ ] == CAN_DEFAULT_ADR ){
135                 can1_TxData(CAN_DEFAULT_ADR, 0,0 );
136             }
137
138             /* Sjekker om det har gått ett minutt,
```

```
139                 * aktiverer så den interne adc for avlesing av signaler */
140                 if( tidTilLesInternADC >= 60){
141                     tidTilLesInternADC -= 60;
142                     statusFlagg.internAdc = FLAGG_ADCSEKVENS_Start;
143                     startInternADC( internADC.kanal[ 0 ] );
144                     internADC.velgNesteKanal = FALSE;
145                 }
146 
147                 /* Sjekker om HMC2003 skal kalibreres */
148                 if(systemTid.sensorReset >= HMC2003_SEKUND_TIL_SET_RESET ){
149                     systemTid.sensorReset -= HMC2003_SEKUND_TIL_SET_RESET;
150                     statusFlagg.hmc2003SR = FLAGG_HMC2003_SetReset;
151                 }
152             }
153 
154             /* Sjekker om alle målinger er utført før endelig adc verdi beregnes
155              * Setter deretter adc status til klar.  */
156             if( statusFlagg.adc == FLAGG_ADCSEKVENS_Ferdig ){
157                 korrigerSignal();
158                 statusFlagg.can = FLAGG_CAN_TxSensorData;
159 
160                 /* Sjekker om ADC skal starte ny sekvens */
161                 if( statusFlagg.adcModus != FLAGG_ADCMODUS_Pause && statusFlagg.hmc2003SR
                    != FLAGG_HMC2003_SetReset){ statusFlagg.adc = FLAGG_ADCSEKVENS_Start;}
162                 else{statusFlagg.adc = FLAGG_ADCSEKVENS_Pause;}
163             }
164 
165             /* Sjekker om sensor skal kalibreres og avlesninger av denne er stoppet */
166             if( (statusFlagg.hmc2003SR == FLAGG_HMC2003_SetReset) && (statusFlagg.adc ==
                FLAGG_ADCSEKVENS_Pause) ){
167                 hmc2003_SetReset();
168                 statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
169                 statusFlagg.adcModus = FLAGG_ADCMODUS_Run;
170             }
171 
172             /* Klar til å sende oppdaterte data ? */
173             if( statusFlagg.can == FLAGG_CAN_TxSensorData){
174                 can1_TxSensorData();
175                 statusFlagg.can = FALSE;
176             }
177 
178             /* Sjekker om tilstrekkelig sync pulser er motatt for å beregne hvorvidt den
                interne systick
179              *  er synkron med den eksterne klokken */
180             if( statusFlagg.clkSync == TRUE ){
181                 clkSync();
182                 statusFlagg.clkSync = FALSE;
183             }
184 
185             /* Leser målingene som utføres av den interne adc */
186             if( statusFlagg.internAdc == FLAGG_ADCSEKVENS_Start){
187 
188                 if( internADC.velgNesteKanal == TRUE ){
189                     can1_TxData(CAN_DEFAULT_ADR, 1,&internADC.kanalIdx );
190                     if( internADC.kanalIdx < 2 ){
191                         startInternADC();
192                         internADC.velgNesteKanal = FALSE;
193                     }
194                     else{
195 
196                         internADC.velgNesteKanal = FALSE;
197                         statusFlagg.internAdc = FLAGG_ADCSEKVENS_Ferdig;
198                     }
199                 }
200                 if( ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC ) == SET ){
201                     internADC.adcData[ internADC.kanalIdx ].verdi = ADC_GetConversionValue(
                        ADC1);
202                     internADC.velgNesteKanal = TRUE;
203                     internADC.kanalIdx++;
204                 }
205             }
206 
207         }
208     }
```

```
1   /**
2    * @file adc.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjoner som benyttes for å lese intern og ekstern ADC.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   *
17   */
18
19  /*-------------------------------------------------------------------------------------------
    ------------
20                                      Inkluderinger
21  -------------------------------------------------------------------------------------------
    ----------*/
22  #include "stm32f0xx.h"
23  #include "stm32f0xx_adc.h"
24  #include "adc.h"
25  #include "globDef.h"
26  /*-------------------------------------------------------------------------------------------
    ------------
27                                      Variabler
28  -------------------------------------------------------------------------------------------
    ----------*/
29  volatile struct param ekstADC[ ADC_Kanaler ];
30  volatile struct adcInt internADC;
31  /*-------------------------------------------------------------------------------------------
    ------------
32                                      Funksjoner
33  -------------------------------------------------------------------------------------------
    ----------*/
34
35  /*****************************************************************************
36   * @brief Filtrerer de oversamplede målinger. 10kHz -> 600Hz
37   * @param  none.
38   * @return none
39   *****************************************************************************/
40  void MAFilter_1( void ){
41
42  }
43
44  /*****************************************************************************
45   * @brief Leser data fra ekstern adc og oppdaterer tilhørende variabler.
46   * @param none.
47   * @return none
48   *****************************************************************************/
49  void lesEksternADC( void ){
50      uint8_t ch;
51      uint8_t kanal;
52      GPIOA->BRR = ADC_CS_Pin;
53      for (ch = 0; ch < ADC_Kanaler; ch++) {
54          GPIOA->BRR = ADC_RD_Pin;
55          kanal = kanalKorrigering( ch );
56          ekstADC[ kanal ].lesteADC_Data = GPIOB->IDR;
57
58          GPIOA->BSRR = ADC_RD_Pin;
59      }
60      GPIOA->BSRR = ADC_RD_Pin;
61      GPIOA->BSRR = ADC_CS_Pin;
62      statusFlagg.adc = FLAGG_ADCSEKVENS_Ferdig;
63  }
64
65  /*****************************************************************************
66   * @brief Tilpasser rekkefølgen til de eksterne adc kanalene til struktur i programmet
67   * @param uint8_t kanal, ønsket kanal der x=0,y=1,og z=2.
68   * @return uint8_t kanal, faktisk adc kanal.
```

```
69   ******************************************************************************/
70  uint8_t kanalKorrigering( uint8_t kanal ){
71      switch (kanal) {
72      case 0: kanal = 2;  break;
73      case 2: kanal = 0;  break;
74      default:
75          break;
76      }
77      return kanal;
78  }
79
80  /*******************************************************************************
81   * @brief Aktiverer en lesesyklus på en kanal
82   * @param uint8_t kanal, definert i adc.h.
83   * @return none
84   ******************************************************************************/
85  void startInternADC(){
86      ADC_StartOfConversion(ADC1);
87  }
88
89
90  /*******************************************************************************
91   * @brief Henter dekodet data
92   * @param none.
93   * @return uint16_t lest adc data.
94   ******************************************************************************/
95  uint16_t lesInternADC( void ){
96      return ADC_GetConversionValue(ADC1);
97  }
98
99
100 /*******************************************************************************
101  * @brief Oppdaterer strukturen som inneholder verdier fra den interne ADC
102  * @param none.
103  * @return none.
104  ******************************************************************************/
105 void lesInternADCSyklus( void ){
106     uint8_t idx = 0;
107
108     for( idx = 0; idx < 2; idx++ ){
109         startInternADC();
110         while( ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC ) == RESET );
111         internADC.adcData[ idx ].verdi = ADC_GetConversionValue(ADC1);
112     }
113 }
114
```

```
1   /**
2    * @file adcConf.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Konfigurasjon av intern og ekstern ADC.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   *
17   */
18
19  /*-------------------------------------------------------------------------------------------
    ------------
20                                      Inkluderinger
21  -------------------------------------------------------------------------------------------
    ----------*/
22  #include "stm32f0xx.h"
23  #include "stm32f0xx_rcc.h"
24  #include "stm32f0xx_gpio.h"
25  #include "stm32f0xx_adc.h"
26  #include "stm32f0xx_exti.h"
27  #include "stm32f0xx_misc.h"
28  #include "stm32f0xx_syscfg.h"
29  #include "adc.h"
30  #include "globDef.h"
31  /*-------------------------------------------------------------------------------------------
    ------------
32                                      Variabler
33  -------------------------------------------------------------------------------------------
    ----------*/
34
35  /*-------------------------------------------------------------------------------------------
    ------------
36                                      Funksjoner
37  -------------------------------------------------------------------------------------------
    ----------*/
38
39  /*****************************************************************************
40   * @brief Initierer ekstern adc MAX11044
41   * @param  none.
42   * @return none
43  *****************************************************************************/
44  void initierEksternADC( void ){
45      GPIO_InitTypeDef    GPIO_InitStructure;
46      EXTI_InitTypeDef    EXTI_InitStructure;
47      NVIC_InitTypeDef   NVIC_InitStructure;
48
49      RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);
50      RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE);
51      RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
52
53      /* ADC EOC */
54      GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_4;
55      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
56      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
57      GPIO_Init(GPIOA, &GPIO_InitStructure);
58
59
60  /*  SYSCFG_InitStruct.CFGR2 = SYSCFG_EXTICR2_EXTI4_PA;*/
61
62      SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource4);
63
64
65      EXTI_InitStructure.EXTI_Line = EXTI_Line4;
66      EXTI_InitStructure.EXTI_LineCmd = ENABLE;
67      EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
68      EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
```

```
69        EXTI_Init(&EXTI_InitStructure);
70
71        NVIC_InitStructure.NVIC_IRQChannel = EXTI4_15_IRQn;
72        NVIC_InitStructure.NVIC_IRQChannelPriority = 0x00;
73        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
74        NVIC_Init(&NVIC_InitStructure);
75
76        GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_2| GPIO_Pin_3 | GPIO_Pin_8 | GPIO_Pin_9 |
          GPIO_Pin_10;
77        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT ;
78        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
79        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
80        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
81        GPIO_Init(GPIOA, &GPIO_InitStructure);
82
83        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
84        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT ;
85        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
86        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
87        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
88        GPIO_Init(GPIOB, &GPIO_InitStructure);
89
90        /* Initierer ADC
91         * Intern referanse
92         * Offsett binary format
93         * CONVST kontrollerer når konvertering starter
94         */
95        GPIOA->BSRR = GPIO_Pin_9; /* CS */
96        GPIOA->BSRR = GPIO_Pin_10; /* WR */
97        GPIOB->ODR = 0x0000;
98        GPIOA->BRR = GPIO_Pin_9; /* CS */
99        GPIOA->BRR = GPIO_Pin_10; /* WR */
100       GPIOB->ODR = 0x0000;
101       GPIOB->ODR = 0x0000;
102       GPIOA->BSRR = GPIO_Pin_10; /* WR */
103       GPIOA->BSRR = GPIO_Pin_10; /* WR */
104       GPIOA->BSRR = GPIO_Pin_9; /* CS */
105       GPIOA->BRR = GPIO_Pin_9; /* CS */
106       GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
107       GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN ;
108       GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
109       GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
110       GPIO_InitStructure.GPIO_OType = GPIO_OType_OD;
111       GPIO_Init(GPIOB, &GPIO_InitStructure);
112   }
113
114   /*******************************************************************************
115    * @brief Initierer den interne adc.
116    * @param  none.
117    * @return none
118   *******************************************************************************/
119   void initierInternADC( void ){
120       /* Her må klokkedeling tas med!! */
121       GPIO_InitTypeDef   GPIO_InitStructure;
122       ADC_InitTypeDef    ADC_InitStructure;
123
124       RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);
125       RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
126       RCC_ADCCLKConfig(RCC_ADCCLK_HSI14);
127
128       GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;
129       GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
130       GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
131       GPIO_Init(GPIOA, &GPIO_InitStructure);
132
133       ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
134       ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
135       ADC_InitStructure.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None;
136       ADC_InitStructure.ADC_ExternalTrigConv =  DISABLE;
137       ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
138       ADC_InitStructure.ADC_ScanDirection = ADC_ScanDirection_Upward;
139       ADC_Init(ADC1, &ADC_InitStructure);
140
141       ADC_ChannelConfig(ADC1, ADC_Channel_0 , ADC_SampleTime_1_5Cycles );
```

```
142         ADC_ChannelConfig(ADC1, ADC_Channel_1 , ADC_SampleTime_1_5Cycles );
143         ADC_DiscModeCmd(ADC1,ENABLE);
144         ADC_GetCalibrationFactor(ADC1);
145         ADC_Cmd(ADC1, ENABLE);
146         ADC_StartOfConversion(ADC1);
147 
148         while( ADC_GetFlagStatus(ADC1, ADC_FLAG_ADEN) == RESET );
149 
150     }
151 
```

```
1   /**
2    * @file adc.h
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjonsprototyper og definisjoner for å konfigurere og lese intern og
     ekstern ADC.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  #ifndef _ADC_INC_
16  #define _ADC_INC_
17  /*-------------------------------------------------------------------------------------
    ------------
18                                      Inkluderinger
19  -------------------------------------------------------------------------------------
    ----------*/
20  #include "stm32f0xx_gpio.h"
21  #include "filter.h"
22
23  /*-------------------------------------------------------------------------------------
    ------------
24                                      Defineringer
25  -------------------------------------------------------------------------------------
    ----------*/
26  /* Definisjon av intern ADC kanaler */
27  #define adcVbatt        0b01        /* Intern ADC kanal 1, Batterispenning */
28  #define adcVsetReset    0b10        /* Intern ADC kanal 2, Set/Reset spenning */
29  #define INTERN_ADC_Vbatt    0       /* Referanse til tabell */
30  #define INTERN_ADC_VSR      1       /* Referanse til tabell */
31
32  #define ADC_CS_Pin          GPIO_Pin_9
33  #define ADC_RD_Pin          GPIO_Pin_8
34  #define ADC_WR_Pin          GPIO_Pin_10
35  #define ADC_EOC_Pin         GPIO_Pin_4
36  #define ADC_CONVST_Pin      GPIO_Pin_3
37  #define ADC_SHDN_Pin        GPIO_Pin_2
38
39  #define ADC_Kanaler         4
40  #define MSB                 1
41  #define LSB                 0
42
43  #define OFFSETT_FORTEGN_POS 1
44  #define OFFSETT_FORTEGN_NEG 0
45
46  /*
47  #define FILTER_LENGDE                   2
48  #define FILTER_DIVISJON                 1
49
50  #define oversamplinger_8    8
51  #define oversamplinger      oversamplinger_8
52
53  #ifdef oversamplinger_8
54  #define bredde              3
55  #endif
56  */
57  /*-------------------------------------------------------------------------------------
    ------------
58                                      Variabler
59  -------------------------------------------------------------------------------------
    ----------*/
60  struct param{
61      uint16_t lesteADC_Data;         // Leste data
62      uint16_t dac;
63      uint16_t offsett;
64      uint8_t fortegn;                /* Fortegn til offsett*/
65      uint16_t G_teller;
66      uint16_t G_nevner;
67      union{
```

```
68          uint8_t byte[2];                        // rs232/can [0]-LSB [1]-MSB
69          uint16_t verdi;                         // Korrigerte målinger
70      } korrigert;                                /* Faktisk magnetfelt */
71  };
72  extern volatile struct param ekstADC[ ADC_Kanaler ];
73
74
75  struct adcInt{
76      uint8_t velgNesteKanal:2;
77      uint8_t kanalIdx;
78      uint8_t kanal[2];
79      union{
80          uint8_t byte[2];                        // rs232/can [0]-LSB [1]-MSB
81          uint16_t verdi;                         // LesteADC_Data
82      }adcData[2];
83  };
84  extern volatile struct adcInt internADC;
85  /*-------------------------------------------------------------------------------------------
    ------------
86                                      Prototyper
87  ---------------------------------------------------------------------------------------------
    ----------*/
88  void lesEksternADC( void );
89  void kalibrerForsterkning ( void );
90  void MAFilter_1( void );
91  void initierEksternADC( void );
92  void initierInternADC( void );
93  uint8_t kanalKorrigering( uint8_t kanal );
94  void startInternADC( );
95  uint16_t lesInternADC( void );
96  void lesInternADCSyklus( void );
97  #endif
98
```

```
1   /**
2    * @file can.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjoner som benyttes for can kommunikasjon.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   *
17   */
18
19  /*-------------------------------------------------------------------------------------------
     ------------
20                                          Inkluderinger
21  -------------------------------------------------------------------------------------------
     ----------*/
22  #include "stm32f0xx.h"
23  #include "stm32f0xx_can.h"
24  #include "adc.h"
25  #include "globDef.h"
26  #include "can.h"
27  #include "flash.h"
28  #include "hmc2003.h"
29
30  /*-------------------------------------------------------------------------------------------
     ------------
31                                          Variabler
32  -------------------------------------------------------------------------------------------
     ----------*/
33  volatile CanTxMsg can1TxMsg;
34  volatile CanRxMsg can1RxMsg;
35  /*-------------------------------------------------------------------------------------------
     ------------
36                                          Funksjoner
37  -------------------------------------------------------------------------------------------
     ----------*/
38
39  /*****************************************************************************************
40   * @brief Sender en canbus pakke med maksimalt 8 byte data
41   * @param msgId  - Melding ID definert i can.h \n
42   * @param lengde - Pakkelengde [0..4] \n
43   * @param data - Data som skal sendes, uchar tabell med maksimalt 8 elementer \n
44   * @return none
45   *****************************************************************************************/
46  void can1_TxData( unsigned int msgId , unsigned char lengde, unsigned char *data ){
47      int i = 0;
48
49      can1TxMsg.StdId = msgId;
50      can1TxMsg.ExtId = 0x0;
51      can1TxMsg.IDE = CAN_ID_STD;
52      can1TxMsg.RTR = CAN_RTR_DATA;
53      can1TxMsg.DLC = lengde;
54
55      for( ; i < lengde ; i++ ){
56          can1TxMsg.Data[ i ] = *(i + data );
57      }
58      CAN_Transmit(CAN1, &can1TxMsg);
59  }
60
61  /*****************************************************************************************
62   * @brief Sender en canbus pakke med maksimalt 8 byte data
63   * @param can1Tx - Data som skal sendes. Strukturen er definert i stm32f0xx_can.h.
64   * @return none
65   *****************************************************************************************/
66  void can1_Tx( CanTxMsg *can1Tx ){
67      can1Tx->ExtId = 0x00;
68      can1Tx->IDE = CAN_ID_STD;
```

```
69      can1Tx->RTR = CAN_RTR_DATA;
70      CAN_Transmit(CAN1, can1Tx);
71  }
72
73  /*******************************************************************************
74   * @brief Sender en canbus pakke med sensordata
75   * @param none
76   * @return none
77   *******************************************************************************/
78  void can1_TxSensorData( void ){
79      uint8_t idx;
80
81      can1TxMsg.ExtId = 0x00;
82      can1RxMsg.StdId = CAN_MSG_SENSORDATA | 3; //flashLagring.param[
        _FLASH_PARAM_CAN_ID_ ];
83      can1TxMsg.IDE = CAN_ID_STD;
84      can1TxMsg.RTR = CAN_RTR_DATA;
85      can1TxMsg.DLC = 6;
86      for( idx = 0; idx < (ADC_Kanaler-1); idx++ ){
87          can1TxMsg.Data[ 2*idx ]   = ekstADC[ idx ].korrigert.byte[ MSB ];
88          can1TxMsg.Data[ 2*idx+1 ] = ekstADC[ idx ].korrigert.byte[ LSB ];
89      }
90      CAN_Transmit(CAN1, &can1TxMsg);
91  }
92
93  /*******************************************************************************
94   * @brief Canbus Rx Avbrudd
95   * Behandler mottatt canbus data
96   * @param none.
97   * @return none
98   *******************************************************************************/
99  void CEC_CAN_IRQHandler(void){
100     CAN_Receive(CAN1, CAN_FIFO0, &can1RxMsg);
101     can1_Tx(&can1RxMsg);
102     switch ( can1RxMsg.StdId & 0x7F0 ) {
103         case CAN_MSG_START_STOPP_MEAS:
104             if( statusFlagg.adc != FLAGG_ADCSEKVENS_Start ){statusFlagg.adc =
                FLAGG_ADCSEKVENS_Start;}
105             else{ statusFlagg.adc = FLAGG_ADCSEKVENS_Pause; }
106             break;
107
108         case CAN_MSG_CAN_ID_TILDELING:
109             /* Lagrer den oppdaterte canbus adressen */
110             flashLagring.param[ _FLASH_PARAM_CAN_ID_] = (uint16_t)can1RxMsg.Data[0];
111             flashLagring.param[ _FLASH_PARAM_CAN_Initiert_ ] = TRUE;
112             lagreTilFLASH( &flashLagring );
113             break;
114
115         case CAN_MSG_GAIN_KALIBRERING: /* Denne må kunne legge inn verdier dersom
            melding inneholder data */
116             /* Venter til adc sekvens er ferdig */
117             statusFlagg.adcModus = FLAGG_ADCMODUS_Pause;
118             while( statusFlagg.adc != FLAGG_ADCSEKVENS_Pause);
119             beregneGain();
120
121             statusFlagg.adcModus = FLAGG_ADCMODUS_Run;
122             break;
123
124         case CAN_MSG_START_SET_RESET:
125             statusFlagg.hmc2003SR = FLAGG_HMC2003_SetReset;
126             break;
127         case  CAN_MSG_SET_GAIN:
128             ekstADC[ can1RxMsg.Data[0] ].G_teller = can1RxMsg.Data[1]<<8;
129             ekstADC[ can1RxMsg.Data[0] ].G_teller |= can1RxMsg.Data[2];
130             ekstADC[ can1RxMsg.Data[0] ].G_nevner = can1RxMsg.Data[3]<<8;
131             ekstADC[ can1RxMsg.Data[0] ].G_nevner |= can1RxMsg.Data[4];
132             break;
133     }
134 }
135
136
```

```
1   /**
2    * @file canConf.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjoner som benyttes for can kommunikasjon.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   *
17   */
18
19  /*-------------------------------------------------------------------------------------------
     ------------
20                                      Inkluderinger
21  -------------------------------------------------------------------------------------------
     ----------*/
22  #include "stm32f0xx.h"
23  #include "stm32f0xx_can.h"
24  #include "stm32f0xx_rcc.h"
25  #include "stm32f0xx_gpio.h"
26  #include "stm32f0xx_misc.h"
27  #include "stm32f0xx_syscfg.h"
28  #include "globDef.h"
29  #include "can.h"
30  /*-------------------------------------------------------------------------------------------
     ------------
31                                      Variabler
32  -------------------------------------------------------------------------------------------
     ----------*/
33
34  /*-------------------------------------------------------------------------------------------
     ------------
35                                      Funksjoner
36  -------------------------------------------------------------------------------------------
     ----------*/
37  /*****************************************************************************
38   * @brief Initierer intern can kontroller og tilhørende filter.
39   * @param  none.
40   * @return none
41  *****************************************************************************/
42  void initCAN( void ){
43      GPIO_InitTypeDef    GPIO_InitStructure;
44      NVIC_InitTypeDef   NVIC_InitStructure;
45      CAN_InitTypeDef CAN_InitStructure;
46      CAN_FilterInitTypeDef CAN_FilterInitStructure;
47
48
49      RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN1, ENABLE);
50      RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);
51
52      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 | GPIO_Pin_12  ;
53      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
54      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
55      GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
56
57      GPIO_Init(GPIOA, &GPIO_InitStructure);
58      GPIO_PinAFConfig(GPIOA, GPIO_PinSource11, GPIO_AF_4 );
59      GPIO_PinAFConfig(GPIOA, GPIO_PinSource12, GPIO_AF_4 );
60
61      /* CAN Kommunikasjon*/
62      CAN_DeInit( CAN1);
63      CAN_StructInit(&CAN_InitStructure);
64
65      CAN_InitStructure.CAN_TTCM = DISABLE;
66      CAN_InitStructure.CAN_ABOM = DISABLE;
67      CAN_InitStructure.CAN_AWUM = DISABLE;
68      CAN_InitStructure.CAN_NART = DISABLE;
```

```
69        CAN_InitStructure.CAN_RFLM = DISABLE;
70        CAN_InitStructure.CAN_TXFP = DISABLE;
71        CAN_InitStructure.CAN_Mode = CAN_Mode_Normal;
72        CAN_InitStructure.CAN_SJW = CAN_SJW_1tq;
73        CAN_InitStructure.CAN_BS1 = CAN_BS1_4tq;
74        CAN_InitStructure.CAN_BS2 = CAN_BS2_3tq;
75        CAN_InitStructure.CAN_Prescaler = 6;
76        CAN_Init(CAN1, &CAN_InitStructure);
77
78        /* CAN filter init
79         * Ønsker bare å motta meldinger fra masternode og ikke andre sensorer på bus */
80        CAN_FilterInitStructure.CAN_FilterNumber = 0;
81        CAN_FilterInitStructure.CAN_FilterMode = CAN_FilterMode_IdMask;
82        CAN_FilterInitStructure.CAN_FilterScale = CAN_FilterScale_16bit;
83        CAN_FilterInitStructure.CAN_FilterIdHigh = (uint16_t)CAN_MASTER_ID<<5;
84        CAN_FilterInitStructure.CAN_FilterIdLow = (uint16_t)CAN_MASTER_ID<<5;
85        CAN_FilterInitStructure.CAN_FilterMaskIdHigh = (uint16_t)CAN_FILTER_MASKE<<5;
86        CAN_FilterInitStructure.CAN_FilterMaskIdLow = (uint16_t)CAN_FILTER_MASKE<<5;
87        CAN_FilterInitStructure.CAN_FilterFIFOAssignment = CAN_FIFO0;
88        CAN_FilterInitStructure.CAN_FilterActivation = ENABLE;
89
90        CAN_FilterInit(&CAN_FilterInitStructure);
91
92        CAN_ITConfig(CAN1, CAN_IT_FMP0, ENABLE);
93        NVIC_InitStructure.NVIC_IRQChannel = CEC_CAN_IRQn;
94        NVIC_InitStructure.NVIC_IRQChannelPriority = 0x00;
95        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
96        NVIC_Init(&NVIC_InitStructure);
97    }
98
```

```
1   /**
2    * @file can.h
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjonsprototyper og definisjoner for å konfigurere og lese intern og
     ekstern ADC.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  #ifndef _CAN_INC_
16  #define _CAN_INC_
17  /*-------------------------------------------------------------------------------------
    ------------
18                                      Inkluderinger
19  -------------------------------------------------------------------------------------
    ----------*/
20  #include "stm32f0xx_can.h"
21
22  /*-------------------------------------------------------------------------------------
    ------------
23                                      Defineringer
24  -------------------------------------------------------------------------------------
    ----------*/
25
26  /*****************************************************
27   * Oversikt over struktur til canbus melding
28      MSB                          LSB
29      ------------------------------
30      |Type melding | SenderID 4bit|
31      ------------------------------
32     11            3               0
33
34  Sender ID
35  0001    -   Master
36  0010    -   Sensor1
37  0100    -   Sensor2
38  ..
39  ..
40  1110    -   Sensor7
41  1111    -   Mangler adresse
42
43  Datafelt: Her sendes selve informasjonen i meldingen. I henhold til canbus standard
    sendes denne
44              informasjonen med 8bit databredde. Dersom variabler er større enn 8bit
                deles disse opp
45              i 8bit og sendes som big endian.
46              Eksempel: 0x1234 -> data[0] = 12 , data[1]= 34
47
48  *****************************************************/
49
50  #define CAN_MASTER_ID                   0x1 /* CAN bus ID tilhørende masternode
51                                               Dette er den eneste som skal kommunisere
                                                 med sensornodene */
52  #define CAN_FILTER_MASKE                0xF /* Filtermaske som krever at de første 4
    bit stemmer overens
53                                                  med filterID */
54  #define CAN_DEFAULT_ADR                 0xF /* Indikerer at node ikke har fått tildelt
    adresse,
55                                                 dette tildeles fra master. */
56  /* Meldinger som sendes */
57  #define CAN_MSG_SENSORDATA              0x000 /* Sensordata for x,y og z akse.
58                                                   data[0 ..20] - Sensor X
59                                                   data[21..41] - Sensor Y
60                                                   data[42..64] - Sensor Z*/
61  #define CAN_MSG_GAIN_x                  0x010 /* Forsterkning til X akse
62                                                   data[0..4] - 128 målinger av Vref */
63  #define CAN_MSG_GAIN_y                  0x020 /* Forsterkning til Y akse
64                                                   data[0..4] - 128 målinger av Vref */
```

```
65  #define CAN_MSG_GAIN_z                  0x030 /* Forsterkning til Z akse
66                                                  data[0..4] - 128 målinger av Vref */
67  #define CAN_MSG_SENSOR_SPENNINGER       0x040 /* Forsterkning til interne spenninger
    på sensorkort
68                                                  data[0..1] - Vbatt
69                                                  data[2..3] - Sensor Vref
70                                                  data[4..5] - VSR*/
71  /* Meldinger som mottas */
72  #define CAN_MSG_START_STOPP_MEAS        0x050 /* Starter og stopper målinger av
    magnetfelt
73                                                  data[0] 1:Start,0:Stopp*/
74  #define CAN_MSG_CAN_ID_TILDELING        0x060 /* Tildeler canbus id til node
75                                                  data[0] - Ny canbus id*/
76  #define CAN_MSG_GAIN_KALIBRERING        0x070 /* Starter kalibrering av forsterkning
77                                                  data[0] 1:start */
78  #define CAN_MSG_SET_GAIN                0x080 /* Oppdaterer forsterkning
79                                                  data[0] - Kanal
80                                                  data[1] - Teller MSB
81                                                  data[2] - Teller LSB
82                                                  data[3] - Nevner MSB
83                                                  data[4] - Nevner LSB */
84  #define CAN_MSG_START_SET_RESET         0x090 /* Starter set/reset av hmc2003
85                                                  data[0] 1:start */
86
87  /*-------------------------------------------------------------------------------------
    ------------
88                                      Variabler
89  -------------------------------------------------------------------------------------
    ----------*/
90  extern volatile CanTxMsg can1TxMsg;
91  extern volatile CanRxMsg can1RxMsg;
92  /*-------------------------------------------------------------------------------------
    ------------
93                                      Prototyper
94  -------------------------------------------------------------------------------------
    ----------*/
95  void initCAN( void );
96  void can1_TxData( unsigned int msgId , unsigned char lengde, unsigned char *data );
97  void can1_Tx( CanTxMsg *can1Tx );
98  void CEC_CAN_IRQHandler(void);
99  void can1_TxSensorData( void );
100 #endif
101
```

```
1   /**
2    * @file avbrudd.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    12.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Avbrudd.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   *
17   */
18
19  /*-------------------------------------------------------------------------------------------
     ------------
20                                          Inkluderinger
21  -------------------------------------------------------------------------------------------
     ----------*/
22  #include "stm32f0xx.h"
23  #include "stm32f0xx_rcc.h"
24  #include "stm32f0xx_gpio.h"
25  #include "stm32f0xx_adc.h"
26  #include "stm32f0xx_exti.h"
27  #include "stm32f0xx_misc.h"
28  #include "stm32f0xx_syscfg.h"
29  #include "adc.h"
30  #include "globDef.h"
31  #include "avbrudd.h"
32  #include "clkSync.h"
33
34
35  /*-------------------------------------------------------------------------------------------
     ------------
36                                          Variabler
37  -------------------------------------------------------------------------------------------
     ----------*/
38
39
40  /*-------------------------------------------------------------------------------------------
     ------------
41                                          Funksjoner
42  -------------------------------------------------------------------------------------------
     ----------*/
43
44  /*****************************************************************************
45   * @brief Eksternt pinne programavbrudd for ADC EOC og CLK_Sync.
46   * @param  none.
47   * @return none
48   *****************************************************************************/
49  void EXTI4_15_IRQHandler(void){
50      if(EXTI_GetITStatus(EXTI_Line4) != RESET){
51          /* ADC EOC Avbrudd */
52          lesEksternADC();
53
54          EXTI_ClearITPendingBit(EXTI_Line4);
55      }
56      else if(EXTI_GetITStatus(EXTI_Line15) != RESET){
57          /* CLK_Sync */
58
59          clkSyncTid.tid[ clkSyncTid.teller ] = systemTid.sysTeller;
60          clkSyncTid.teller++;
61          statusFlagg.clkSync = TRUE; //( clkSyncTid.teller == 0 ? TRUE : FALSE );
62          initSysTick( SYSTICK_250us );
63
64          EXTI_ClearITPendingBit(EXTI_Line15);
65      }
66
67  }
68
```

```
69  /*****************************************************************************
70   * @brief SysTick avbrudd. Kalles hver 100us.
71   * @param  none.
72   * @return none
73   *****************************************************************************/
74  static teller5 = 0;
75  void SysTick_Handler(void) {
76      /* Starter og stopper ekstern ADC */
77      if( statusFlagg.adc == FLAGG_ADCSEKVENS_Start){
78          GPIOA->ODR ^= ADC_CONVST_Pin;
79      }
80
81      /* Pulser spenningsdobbleren som generere set reset spenning */
82      if( statusFlagg.pulseVSR == TRUE ){
83          GPIOC->ODR ^= GPIO_Pin_15;
84      }
85      else{
86          GPIOC->BSRR = GPIO_Pin_15;
87      }
88      /* 100us teller */
89      systemTid.sysTeller++;
90
91  }
92
```

```
1   /**
2    * @file avbruddConf.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    12.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Konfigurer avbrudd.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /*-------------------------------------------------------------------------------------------
     ------------
16                                      Inkluderinger
17  -------------------------------------------------------------------------------------------
     ----------*/
18  #include "stm32f0xx.h"
19  #include "stm32f0xx_rcc.h"
20  #include "stm32f0xx_gpio.h"
21  #include "stm32f0xx_adc.h"
22  #include "stm32f0xx_exti.h"
23  #include "stm32f0xx_misc.h"
24  #include "stm32f0xx_syscfg.h"
25  #include "globDef.h"
26
27  /*-------------------------------------------------------------------------------------------
     ------------
28                                      Variabler
29  -------------------------------------------------------------------------------------------
     ----------*/
30
31
32  /*-------------------------------------------------------------------------------------------
     ------------
33                                      Funksjoner
34  -------------------------------------------------------------------------------------------
     ----------*/
35  //__enable_irq(); __disable_irq();
36
37  /*****************************************************************************************
38   * @brief Initierer Systick til å gi avbrudd hver 100us.
39   * @param  none.
40   * @return none
41  *****************************************************************************************/
42  void initSysTick( uint16_t load ) {
43      SysTick->CTRL = 0;
44      SysTick->LOAD = load;
45      SysTick->VAL = 0;
46      SysTick->CTRL = 0x07 ;
47      NVIC_SetPriority(SysTick_IRQn, 1);
48  }
49
50  /*****************************************************************************************
51   * @brief Initierer pinnen CLK_Synk og kobler den til ekstern pinne avbrudd .
52   * @param  none.
53   * @return none
54  *****************************************************************************************/
55  void initCLKSync ( void ){
56      GPIO_InitTypeDef    GPIO_InitStructure;
57      EXTI_InitTypeDef    EXTI_InitStructure;
58      NVIC_InitTypeDef   NVIC_InitStructure;
59
60      RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);
61      RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
62
63      /* ADC EOC */
64      GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_15;
65      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN; /*TODO: Er denne AF??*/
66      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
67      GPIO_Init(GPIOA, &GPIO_InitStructure);
68
```

```
69        SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource15);
70
71        EXTI_InitStructure.EXTI_Line = EXTI_Line15;
72        EXTI_InitStructure.EXTI_LineCmd = ENABLE;
73        EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
74        EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
75        EXTI_Init(&EXTI_InitStructure);
76
77        NVIC_InitStructure.NVIC_IRQChannel = EXTI4_15_IRQn;
78        NVIC_InitStructure.NVIC_IRQChannelPriority = 0x00;
79        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
80        NVIC_Init(&NVIC_InitStructure);
81    }
82
```

```
1   /**
2    * @file avbrudd.h
3    *
4    * @author Jørn Tore Ørsland
5    * @date    12.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Globale defineringer.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  #ifndef _AVBRUDD_INC_
16  #define _AVBRUDD_INC_
17  /*-------------------------------------------------------------------------------------------
    ------------
18                                      Inkluderinger
19  -------------------------------------------------------------------------------------------
    ----------*/
20
21  /*-------------------------------------------------------------------------------------------
    ------------
22                                      Defineringer
23  -------------------------------------------------------------------------------------------
    ----------*/
24  #define SYSTICK_100us               4800    /* Systick avbrudd hver 100us 10kHz*/
25  #define SYSTICK_250us               12000   /* Systick avbrudd hver 250us 4kHz*/
26  #define SYSTICK_500us               24000   /* Systick avbrudd hver 500us 2kHz*/
27  /*-------------------------------------------------------------------------------------------
    ------------
28                                      Variabler
29  -------------------------------------------------------------------------------------------
    ----------*/
30
31  /*-------------------------------------------------------------------------------------------
    ------------
32                                      Prototyper
33  -------------------------------------------------------------------------------------------
    ----------*/
34
35  void EXTI4_15_IRQHandler(void);
36  void SysTick_Handler(void);
37  void initSysTick( uint16_t load );
38  void initCLKSync ( void );
39  #endif
40
```

```
1   /**
2    * @file clkSync.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    12.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjoner for synkronisering av systick klokke.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   *
17   */
18
19  /*-------------------------------------------------------------------------------------------
    ------------
20                                          Inkluderinger
21  -------------------------------------------------------------------------------------------
    ----------*/
22  #include "stm32f0xx.h"
23  #include "stm32f0xx_rcc.h"
24  #include "stm32f0xx_gpio.h"
25  #include "stm32f0xx_adc.h"
26  #include "stm32f0xx_exti.h"
27  #include "stm32f0xx_misc.h"
28  #include "stm32f0xx_syscfg.h"
29  #include "adc.h"
30  #include "globDef.h"
31  #include "avbrudd.h"
32  #include "clkSync.h"
33
34  /*-------------------------------------------------------------------------------------------
    ------------
35                                          Variabler
36  -------------------------------------------------------------------------------------------
    ----------*/
37  volatile struct clkTid clkSyncTid;
38  /*-------------------------------------------------------------------------------------------
    ------------
39                                          Funksjoner
40  -------------------------------------------------------------------------------------------
    ----------*/
41  /*****************************************************************************
42   * @brief Synkroniserer SysTick med ekstern synkroniseringspuls.
43   * @param  none.
44   * @return none
45  *****************************************************************************/
46  void clkSync( void ){
47      uint8_t teller = 0;
48      int32_t avvik = 0;
49
50      /* Bergner avviket mellom målte tider og ønsket tid
51       * avvik = SUM[ (Tid(n+1) - Tid(n)) - forventetTid ]
52       * */
53      for( teller = 0; teller < ELEMENTER_I_CLKSYNC - 1 ; teller++ ){
54          avvik += ( clkSyncTid.tid[teller + 1] - clkSyncTid.tid[teller] ) -
            CLKSYNC_FORVENTET_TID;
55      }
56
57      /* Korrigerer systick */
58      if( avvik < (- CLKSYNC_AVVIK) ){
59          clkSyncTid.systickLoad++;
60          SysTick->LOAD = clkSyncTid.systickLoad;
61      }
62      else if( avvik > CLKSYNC_AVVIK ){
63          clkSyncTid.systickLoad--;
64          SysTick->LOAD = clkSyncTid.systickLoad;
65      }
66  }
67
```

```
1   /**
2    * @file clkSync.h
3    *
4    * @author Jørn Tore Ørsland
5    * @date    12.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note .
10   * Oppdateringer:
11   *
12   *
13   */
14
15  #ifndef _CLKSYNC_INC_
16  #define _CLKSYNC_INC_
17  /*-------------------------------------------------------------------------------------------
    ------------
18                                      Inkluderinger
19  -------------------------------------------------------------------------------------------
    ----------*/
20
21  /*-------------------------------------------------------------------------------------------
    ------------
22                                      Defineringer
23  -------------------------------------------------------------------------------------------
    ----------*/
24  #define ELEMENTER_I_CLKSYNC             8
25  #define CLKSYNC_FORVENTET_TID           (9999)
26  #define CLKSYNC_AVVIK                   2           /* Tillatt avvik fra clksync */
27  /*-------------------------------------------------------------------------------------------
    ------------
28                                      Variabler
29  -------------------------------------------------------------------------------------------
    ----------*/
30  struct clkTid{
31      uint32_t tid[ ELEMENTER_I_CLKSYNC ];
32      uint8_t teller:3;   /* Ringteller med område [0..7] */
33      uint32_t systickLoad;
34  };
35  extern volatile struct clkTid clkSyncTid;
36
37  /*-------------------------------------------------------------------------------------------
    ------------
38                                      Prototyper
39  -------------------------------------------------------------------------------------------
    ----------*/
40  void clkSync( void );
41  #endif
42
```

```
1   /**
2    * @file dac.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjonsprototyper og definisjoner for skrive til DAC.
10   * Oppdateringer:
11   *
12   *
13   */
14  /*-------------------------------------------------------------------------------------------
     ------------
15                                          Inkluderinger
16  -------------------------------------------------------------------------------------------
     ----------*/
17  #include "stm32f0xx.h"
18  #include "stm32f0xx_spi.h"
19  #include "stm32f0xx_gpio.h"
20  #include "dac.h"
21  #include "globDef/globDef.h"
22  /*-------------------------------------------------------------------------------------------
     ------------
23                                          Variabler
24  -------------------------------------------------------------------------------------------
     ----------*/
25
26  /*-------------------------------------------------------------------------------------------
     ------------
27                                          Funksjoner
28  -------------------------------------------------------------------------------------------
     ----------*/
29
30  /*****************************************************************************
31   * @brief Skriver data til ekstern DAC.
32   * @param  uint8_t kommando, spesifiserer type melding.
33   * @param uint16_t data, dataene som skal sendes til dac.
34   * @return none
35   *****************************************************************************/
36  void skrivEksternDAC( uint8_t kommando, uint16_t data ){
37      GPIOA->BRR = GPIO_Pin_14; /* Setter pinnen [A14] lav (LDAC inaktiv) */
38      /* CS */
39      GPIOA->BSRR = GPIO_Pin_13;
40      GPIOA->BRR = GPIO_Pin_13;
41
42
43      while((SPI1->SR&SPI_I2S_FLAG_TXE)==RESET);
44      SPI_SendData8(SPI1,(uint8_t)kommando);
45      SPI_SendData8(SPI1,(uint8_t)(data>>2));
46      SPI_SendData8(SPI1,(uint8_t)(data<<6));
47      while((SPI1->SR&SPI_I2S_FLAG_TXE)==RESET);
48      GPIOA->BSRR = GPIO_Pin_13;
49  }
50  void oppdaterVerdier( void){
51      GPIOA->BRR = GPIO_Pin_14; /* Setter pinnen [A14] lav (LDAC inaktiv) */
52      /* CS */
53      GPIOA->BSRR = GPIO_Pin_13;
54      GPIOA->BRR = GPIO_Pin_13;
55
56      while((SPI1->SR&SPI_I2S_FLAG_TXE)==RESET);
57      SPI_SendData8(SPI1,0x83);
58      while((SPI1->SR&SPI_I2S_FLAG_TXE)==RESET);
59      SPI_SendData8(SPI1,0xFF);
60      while((SPI1->SR&SPI_I2S_FLAG_TXE)==RESET);
61      SPI_SendData8(SPI1,0xFF);
62
63      GPIOA->BRR = GPIO_Pin_13;
64      GPIOA->BRR = GPIO_Pin_13;
65      GPIOA->BSRR = GPIO_Pin_13;
66
67  }
68  /*****************************************************************************
```

```
69    * @brief Reduserer bit bredden fra 16 bit til 10bit.
70    * @param  none.
71    * @return none
72    ******************************************************************************/
73   uint16_t adc2Dac( uint16_t data ){
74       return ( data>>6)&0x03FF ;
75   }
76
```

```
1   /**
2    * @file dacConf.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjonsprototyper og definisjoner for å konfigurere DAC.
10   * Oppdateringer:
11   *
12   *
13   */
14  /*-------------------------------------------------------------------------------------------
     ------------
15                                      Inkluderinger
16  -------------------------------------------------------------------------------------------
     ----------*/
17  #include "stm32f0xx.h"
18  #include "stm32f0xx_spi.h"
19  #include "stm32f0xx_rcc.h"
20  #include "stm32f0xx_gpio.h"
21  #include "dac.h"
22  #include "adc.h"
23  /*-------------------------------------------------------------------------------------------
     ------------
24                                      Variabler
25  -------------------------------------------------------------------------------------------
     ----------*/
26
27  /*-------------------------------------------------------------------------------------------
     ------------
28                                      Funksjoner
29  -------------------------------------------------------------------------------------------
     ----------*/
30  /*****************************************************************************
31   * @brief Initierer den eksterne dac.
32   * @param  none.
33   * @return none
34   *****************************************************************************/
35  void initEksternDAC( void ){
36      uint8_t ch = 0;
37
38
39      GPIO_InitTypeDef GPIO_SPI_CONF;
40      GPIO_InitTypeDef GPIO_InitStructure;
41      SPI_InitTypeDef SPI_CONF;
42
43      RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);
44      RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1, ENABLE);
45
46      GPIO_PinAFConfig(GPIOA,GPIO_PinSource5, GPIO_AF_0);
47      GPIO_PinAFConfig(GPIOA,GPIO_PinSource6, GPIO_AF_0);
48      GPIO_PinAFConfig(GPIOA,GPIO_PinSource7, GPIO_AF_0);
49
50      GPIO_StructInit(&GPIO_SPI_CONF);
51      GPIO_SPI_CONF.GPIO_Mode = GPIO_Mode_AF;
52      GPIO_SPI_CONF.GPIO_OType = GPIO_OType_PP;
53      GPIO_SPI_CONF.GPIO_PuPd = GPIO_PuPd_UP;
54      GPIO_SPI_CONF.GPIO_Speed = GPIO_Speed_50MHz;
55      GPIO_SPI_CONF.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_6 |GPIO_Pin_7;
56      GPIO_Init(GPIOA, &GPIO_SPI_CONF);
57
58      SPI_CONF.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_2;
59      SPI_CONF.SPI_CPHA = SPI_CPHA_1Edge;
60      SPI_CONF.SPI_CPOL = SPI_CPOL_High;
61      SPI_CONF.SPI_DataSize = SPI_DataSize_8b;
62      SPI_CONF.SPI_Direction = SPI_Direction_1Line_Tx;
63      SPI_CONF.SPI_FirstBit = SPI_FirstBit_MSB;
64      SPI_CONF.SPI_Mode = SPI_Mode_Master;
65      SPI_CONF.SPI_NSS = SPI_NSS_Soft | SPI_NSSInternalSoft_Set;
66      SPI_Init(SPI1,&SPI_CONF);
67      SPI_Cmd(SPI1,ENABLE);
68
```

```
69        /* CS og LDAC*/
70        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13 |GPIO_Pin_14 ;
71        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
72        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
73        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
74        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
75
76        GPIO_Init(GPIOA, &GPIO_InitStructure);
77
78        /* Konfigurer DAC */
79        skrivEksternDAC(0x50,0xFFFF); /* SW Clear */
80        skrivEksternDAC(0x51,0xFFFF); /* SW Reset */
81        /* Power */
82        skrivEksternDAC(DAC_CMD_POWER|DAC_POW_Mode_Normal, (uint16_t)(DAC_CONF_CH_ABC_<<8)
          );
83
84        /* Config */
85        skrivEksternDAC(DAC_CMD_CONFIG | DAC_CH_ALLE | DAC_CONF_LATCH_Transperent, (
          uint16_t)(DAC_CONF_CH_ABC_<<8) );
86        /* Ref */
87        skrivEksternDAC(DAC_CMD_REF|DAC_REF_Pow_On|DAC_REF_Ref_4_1V,0xFFF);
88
89        /* Setter alle kanaler til V/2 */
90        skrivEksternDAC(DAC_CMD_CODEn_LOADn|DAC_CH_ALLE, 100 );
91        for( ch = 0; ch < (ADC_Kanaler-1); ch++){
92            ekstADC[ ch ].dac = 512;
93        }
94    }
95
```

```
1   /**
2    * @file dac.h
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjonsprototyper og definisjoner for å konfigurere og skrive til DAC.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  #ifndef _DAC_INC_
16  #define _DAC_INC_
17  /*-------------------------------------------------------------------------------------------
    ------------
18                                      Inkluderinger
19  -------------------------------------------------------------------------------------------
    ----------*/
20
21  /*-------------------------------------------------------------------------------------------
    ------------
22                                      Defineringer
23  -------------------------------------------------------------------------------------------
    ----------*/
24  #define DAC_CH0_x                   0x00
25  #define DAC_CH1_y                   0x01
26  #define DAC_CH2_z                   0x02
27  #define DAC_CH_ALLE                 0x08
28  #define DAC_CONF_CH_ABC_            0x07
29
30  /* Kommandoer */
31  #define DAC_CMD_CODEn_LOADn         0x30
32  #define DAC_CMD_POWER               0x40
33  #define DAC_CMD_CONFIG              0x60
34  #define DAC_CMD_REF                 0x70
35  #define DAC_CMD_CODE_ALL_LOAD_ALL   0x83
36
37  /* REF parametere */
38  #define DAC_REF_Pow_Min_1_DAC       0x00
39  #define DAC_REF_Pow_On              0x04
40  #define DAC_REF_Ref_2_5V            0x01
41  #define DAC_REF_Ref_2_0V            0x02
42  #define DAC_REF_Ref_4_1V            0x03
43
44  /* CONFIG parametere */
45  #define DAC_CONF_LATCH_On           0x00
46  #define DAC_CONF_LATCH_Transperent  0x01
47
48  /* POWER parametere */
49  #define DAC_POW_Mode_Normal         0x00
50  #define DAC_POW_Mode_PowDwn_1k      0x01
51  #define DAC_POW_Mode_PowDwn_100k    0x02
52  #define DAC_POW_Mode_PowDwn_HiZ     0x03
53
54
55  /*-------------------------------------------------------------------------------------------
    ------------
56                                      Variabler
57  -------------------------------------------------------------------------------------------
    ----------*/
58
59  /*-------------------------------------------------------------------------------------------
    ------------
60                                      Macroer
61  -------------------------------------------------------------------------------------------
    ----------*/
62
63
64  /*-------------------------------------------------------------------------------------------
    ------------
65                                      Prototyper
```

```
66  ----------------------------------------------------------------------------------------
    ----------*/
67  void initEksternDAC( void );
68  void skrivEksternDAC( uint8_t kommando, uint16_t data );
69  uint16_t adc2Dac( uint16_t data );
70  void oppdaterVerdier( void);
71  #endif
72
```

```
1   /**
2    * @file flash.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    16.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjoner som benyttes for å lagre variabler i flash.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   *
17   */
18
19  /*-------------------------------------------------------------------------------------------
    ------------
20                                      Inkluderinger
21  -------------------------------------------------------------------------------------------
    ----------*/
22  #include "stm32f0xx.h"
23  #include "stm32f0xx_flash.h"
24  #include "globDef.h"
25  #include "flash.h"
26  #include "can.h"
27
28  /*-------------------------------------------------------------------------------------------
    ------------
29                                      Variabler
30  -------------------------------------------------------------------------------------------
    ----------*/
31  volatile struct flash flashLagring;
32
33  /*-------------------------------------------------------------------------------------------
    ------------
34                                      Funksjoner
35  -------------------------------------------------------------------------------------------
    ----------*/
36
37  /*****************************************************************************************
38   * @brief Lagrer flash struktur i flashminnet.
39   * @param  none.
40   * @return none
41  *****************************************************************************************/
42  void lagreTilFLASH( struct flash *data ){
43      uint8_t teller = 0;
44      FLASH_Unlock();
45      FLASH_ClearFlag(FLASH_FLAG_BSY | FLASH_FLAG_EOP | FLASH_FLAG_PGERR );
46      FLASH_ErasePage( data->baseAdresse );
47      FLASH_Unlock();
48      for (teller = 0; teller < data->antallParametere; teller++) {
49          FLASH_ProgramWord( (data->baseAdresse + teller*4) , data->param[ teller ] );
50      }
51      FLASH_Lock();
52  }
53
54  /*****************************************************************************************
55   * @brief Oppdaterer flash strukturen med data fra flash.
56   * @param  flash *data, strukturen som skal oppdateres.
57   * @return none
58  *****************************************************************************************/
59  void leseFraFLASH(  struct flash *data ){
60      uint8_t teller;
61
62      for( teller = 0; teller < data->antallParametere-1; teller++){
63          data->param[ teller ] = (*(uint32_t *)( data->baseAdresse + 4*teller));
64      }
65  }
66
67  /*****************************************************************************************
68   * @brief Initierer flash variabler til default verdier dersom ikke disse er initierte.
```

```
69    * @param  flash *data, strukturen som skal oppdateres.
70    * @return none
71   ********************************************************************************/
72   void initFLASHVariabler( struct flash *data ){
73       /* Legger standard eller kalibrerte verdier inn i struktur */
74       leseFraFLASH( data );
75
76       if ( data->param[ _FLASH_PARAM_CAN_Initiert_ ] != TRUE ) {
77           data->param[ _FLASH_PARAM_CAN_Initiert_ ] = CAN_DEFAULT_ADR;
78       }
79
80       if ( data->param[ _FLASH_PARAM_GAIN_Kalibrert_ ] != TRUE ) {
81           /* Legger inn estimert forsterkning */
82           data->param[ _FLASH_PARAM_GAIN_X_t_ ] = 109;
83           data->param[ _FLASH_PARAM_GAIN_X_n_ ] = 10;
84
85           data->param[ _FLASH_PARAM_GAIN_Y_t_ ] = 109;
86           data->param[ _FLASH_PARAM_GAIN_Y_n_ ] = 10;
87
88           data->param[ _FLASH_PARAM_GAIN_Z_t_ ] = 109;
89           data->param[ _FLASH_PARAM_GAIN_Z_n_ ] = 10;
90       }
91   }
92
```

```
1  /**
2   * @file dac.h
3   *
4   * @author Jørn Tore Ørsland
5   * @date    11.04.16
6   * @version V1.00
7   * @brief
8   *
9   * @note Funksjonsprototyper og definisjoner for å konfigurere og skrive til DAC.
10  * Oppdateringer:
11  *
12  *
13  */
14
15 #ifndef _FLASH_INC_
16 #define _FLASH_INC_
17 /*-------------------------------------------------------------------------------------------
    ------------
18                                      Inkluderinger
19 -------------------------------------------------------------------------------------------
    ----------*/
20
21 /*-------------------------------------------------------------------------------------------
    ------------
22                                      Defineringer
23 -------------------------------------------------------------------------------------------
    ----------*/
24 #define _MAX_ANT_PARAM_I_FLASH_          28
25 #define _ANT_PARAM_I_FLASH_              9
26 #define _FLASH_BASE_ADR_                 ((uint32_t)0x08004200)
27
28 #define _FLASH_PARAM_GAIN_X_t_           0
29 #define _FLASH_PARAM_GAIN_X_n_           1
30 #define _FLASH_PARAM_GAIN_Y_t_           2
31 #define _FLASH_PARAM_GAIN_Y_n_           3
32 #define _FLASH_PARAM_GAIN_Z_t_           4
33 #define _FLASH_PARAM_GAIN_Z_n_           5
34 #define _FLASH_PARAM_GAIN_Kalibrert_     6
35 #define _FLASH_PARAM_CAN_ID_             7
36 #define _FLASH_PARAM_CAN_Initiert_       8
37
38 /*-------------------------------------------------------------------------------------------
    ------------
39                                      Variabler
40 -------------------------------------------------------------------------------------------
    ----------*/
41 struct flash{
42     uint32_t param[ _ANT_PARAM_I_FLASH_ ];
43     uint32_t baseAdresse;
44     uint8_t oppdatert;
45     uint8_t antallParametere;
46 };
47 extern volatile struct flash flashLagring;
48
49 /*-------------------------------------------------------------------------------------------
    ------------
50                                      Macroer
51 -------------------------------------------------------------------------------------------
    ----------*/
52
53
54 /*-------------------------------------------------------------------------------------------
    ------------
55                                      Prototyper
56 -------------------------------------------------------------------------------------------
    ----------*/
57 void lagreTilFLASH( struct flash *data );
58 void leseFraFLASH( struct flash *data );
59 void initFLASHVariabler( struct flash *data );
60
61 #endif
62
```

```
1   /**
2    * @file hmc2003.c
3    *
4    * @author Jørn Tore Ørsland
5    * @date    12.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Funksjoner tilhørende sensor HMC2003.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  /* TODO LIST
16   */
17
18  /*-------------------------------------------------------------------------------------------
    ------------
19                                      Inkluderinger
20  -------------------------------------------------------------------------------------------
    ----------*/
21  #include "stm32f0xx.h"
22  #include "stm32f0xx_rcc.h"
23  #include "stm32f0xx_gpio.h"
24  #include "stm32f0xx_adc.h"
25  #include "stm32f0xx_exti.h"
26  #include "stm32f0xx_misc.h"
27  #include "stm32f0xx_syscfg.h"
28  #include "hmc2003.h"
29  #include "dac.h"
30  #include "adc.h"
31  #include "globDef.h"
32  #include "flash.h"
33  #include "can.h"
34  /*-------------------------------------------------------------------------------------------
    ------------
35                                      Variabler
36  -------------------------------------------------------------------------------------------
    ----------*/
37
38  volatile struct v VSR;
39  static uint32_t teller2 = 0;
40  static uint32_t teller3 = 0;
41  /*-------------------------------------------------------------------------------------------
    ------------
42                                      Funksjoner
43  -------------------------------------------------------------------------------------------
    ----------*/
44
45  /*****************************************************************************
46   * @brief Initierer sensoren HMC2003.
47   * @param  none.
48   * @return none
49   *****************************************************************************/
50  void initHmc2003( void ){
51      /* Reset - PC13,
52       * Set   - PC14
53       * VSR   - PC15 */
54
55      RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
56      GPIO_InitTypeDef    GPIO_InitStructure;
57      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13 | GPIO_Pin_14 | GPIO_Pin_15;
58      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT ;
59      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
60      GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;
61      GPIO_Init(GPIOC, &GPIO_InitStructure);
62
63      //hmc2003_SetReset();
64  }
65
66  /*****************************************************************************
67   * @brief Utfører SET/RESET og beregner sensor offsett.
68   * @param  none.
```

```
69   * @return none
70   ********************************************************************************************/
71  void hmc2003_SetReset( void ){
72      uint32_t tmpTid = 0;
73      uint8_t teller;
74      uint8_t idx = 0;
75      uint32_t Vset[ ADC_Kanaler ];
76      uint32_t Vreset[ ADC_Kanaler ];
77
78      for(teller = 0; teller < (ADC_Kanaler -1); teller++){
79          Vset[teller] = 0;
80          Vreset[teller] = 0;
81          ekstADC[ teller ].offsett = 0;
82      }
83
84      /* Transformerer opp spenning */
85      statusFlagg.pulseVSR = TRUE;
86      GPIOC->BRR = GPIO_Pin_13;
87      tmpTid = systemTid.sysTeller + 2000;
88      while( systemTid.sysTeller < tmpTid );  /* Etter anbefalig fra datablad, venter 2
        sek før måling startes*/
89      /* Set */
90      GPIOC->BSRR = GPIO_Pin_14;
91
92      tmpTid = systemTid.sysTeller + 2000;
93      while( systemTid.sysTeller < tmpTid );  /* Etter anbefalig fra datablad, venter 2
        sek før måling startes*/
94
95      statusFlagg.pulseVSR = FALSE;
96
97      tmpTid = systemTid.sysTeller + TIMEOUT_2s;
98      while( systemTid.sysTeller < tmpTid );  /* Etter anbefalig fra datablad, venter 2
        sek før måling startes*/
99
100     statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
101     statusFlagg.adcModus = FLAGG_ADCMODUS_Pause;
102
103     while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på oppdatert måling*/
104     korrigerSignal();
105
106     for(teller = 0; teller < 16; teller++){
107         statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
108         while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på oppdatert
            måling*/
109         korrigerSignal();
110     }
111     tmpTid = systemTid.sysTeller + 200;
112     while( systemTid.sysTeller < tmpTid );  /* Etter anbefalig fra datablad, venter 2
        sek før måling startes*/
113     for(teller = 0; teller < 64; teller++){
114         statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
115         while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på oppdatert
            måling*/
116         korrigerSignal();
117         for (idx = 0; idx < (ADC_Kanaler-1) ; idx++) {
118             Vset[ idx ] += ekstADC[ idx ].korrigert.verdi;
119         }
120     }
121     GPIOC->BRR = GPIO_Pin_14;
122     tmpTid = systemTid.sysTeller + 200;
123     while( systemTid.sysTeller < tmpTid );
124
125     /* Reset */
126     GPIOC->BSRR = GPIO_Pin_13;
127     tmpTid = systemTid.sysTeller + TIMEOUT_2s;/*TODO: OMSKRIVES*/
128     while( systemTid.sysTeller < tmpTid );  /* Etter anbefalig fra datablad, venter 2
        sek før måling startes*/
129     statusFlagg.adcModus = FLAGG_ADCMODUS_Pause;
130     statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
131     while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på oppdatert måling*/
132     korrigerSignal();
133     for(teller = 0; teller < 16; teller++){
134         statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
135         while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på oppdatert
```

```
            måling*/
136             korrigerSignal();
137         }
138         tmpTid = systemTid.sysTeller + 200;
139         while( systemTid.sysTeller < tmpTid );  /* Etter anbefalig fra datablad, venter 2
            sek før måling startes*/
140         for(teller = 0; teller < 64; teller++){
141             statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
142             while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på oppdatert
                måling*/
143             korrigerSignal();
144             for (idx = 0; idx < (ADC_Kanaler-1) ; idx++) {
145                 Vreset[ idx ] += ekstADC[ idx ].korrigert.verdi;
146             }
147         }
148 
149         /*Beregner offsett til kanal */
150         for (teller = 0; teller < (ADC_Kanaler-1) ; teller++) {
151 
152             if( Vset[teller] > Vreset[teller]){
153                 ekstADC[ teller ].offsett = (Vset[teller] - Vreset[teller])>>7;
154                 ekstADC[ teller ].fortegn = OFFSETT_FORTEGN_POS;
155             }
156             else{
157                 ekstADC[ teller ].offsett = (Vreset[teller] - Vset[teller] )>>7;
158                 ekstADC[ teller ].fortegn = OFFSETT_FORTEGN_NEG;
159             }
160             //can1_TxData(CAN_DEFAULT_ADR, 1,&ekstADC[ teller ].fortegn );
161             can1TxMsg.Data[ 2*teller ]   = ekstADC[ teller ].offsett>>8;
162             can1TxMsg.Data[ 2*teller+1 ] = ekstADC[ teller ].offsett;
163         }
164         GPIOC->BSRR = GPIO_Pin_13;
165         can1TxMsg.DLC = 6;
166         can1TxMsg.StdId = 0;
167         can1_Tx( &can1TxMsg );
168     }
169 
170 
171     /*****************************************************************************
172      * @brief Transformerer opp spenningen VSR til ønsket nivå (ca20V).
173      * @param  none.
174      * @return none
175      *****************************************************************************/
176     void transformer_VSR ( void ){
177         uint16_t tmpTid = 0;
178         tmpTid = systemTid.sysTeller + MAX_LADETID;
179 
180         /* Aktiver pulsing av V_SR - PC15 */
181         statusFlagg.pulseVSR = TRUE;
182 
183         /* Mål spenning og vent til denne er ca 20V */
184         lesInternADCSyklus();
185         while ( (VSR_MIN > internADC.adcData[INTERN_ADC_VSR].verdi) && (systemTid.sysTeller
             < tmpTid) ){
186             lesInternADCSyklus();
187         }
188         /* Dektiver pulsing av V_SR - PC15 */
189         statusFlagg.pulseVSR = FALSE;
190 
191         lesInternADCSyklus();
192         VSR.VSRoppladet = internADC.adcData[INTERN_ADC_VSR].verdi;
193         VSR.oppladet = TRUE;
194     }
195 
196     /*****************************************************************************
197      * @brief Beregner den faktiske forsterkningen tilhørende hver instrumentforsterker.
198      * @param  none.
199      * @return none
200      *****************************************************************************/
201     /* Leverer ikke forventet forsterkning, antar dette skyldes manglende presisjon */
202     void beregneGain( void ){
203         uint32_t tmpTid = 0;
204         uint8_t chTeller = 0;
205         uint8_t teller = 0;
```

```
206         uint16_t deltaDAC_16b=0;
207         uint32_t lestVref = 0;
208         uint16_t dacVerdi;
209         uint32_t adcVo;
210         uint16_t vRef_10b = 0;
211         uint32_t sumDeltaADC=0;
212         uint16_t deltaDAC_10b = 0;
213         uint16_t sampler = 0;
214
215
216         tmpTid = systemTid.sysTeller + TIMEOUT_2s;
217
218         /* 1) Start med å måle Vref slik at denne er kjent. */
219         statusFlagg.adcModus = FLAGG_ADCMODUS_Pause;
220         for(teller = 0; teller < 50; teller++){
221             statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
222             while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på oppdatert
                måling*/
223         }
224         for(teller = 0; teller < 128; teller++){
225             statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
226             while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på oppdatert
                måling*/
227             lestVref += ekstADC[ 3 ].lesteADC_Data;
228         }
229         lestVref = lestVref>>7;
230
231         /* Definerer delta DAC verdier */
232         deltaDAC_10b = 1;
233         /* Konverterer til 16bit representasjon */
234         deltaDAC_16b = ((uint32_t)deltaDAC_10b<<15)/625;
235         vRef_10b = (625*(uint32_t)lestVref)>>15;
236
237         skrivEksternDAC(DAC_CMD_CODEn_LOADn|DAC_CH0_x|DAC_CH1_y|DAC_CH2_z, vRef_10b );
238
239         /* Itererer gjennom aksene, men ikke Vref */
240         for (chTeller = 0; chTeller < ( ADC_Kanaler - 1 ); chTeller++) {
241             switch (chTeller) {
242             case 0:
243                 can1TxMsg.StdId = CAN_MSG_GAIN_x ;//| flashLagring.param[
                    _FLASH_PARAM_CAN_ID_ ];
244                 break;
245             case 1:
246                 can1TxMsg.StdId = CAN_MSG_GAIN_y ;//| flashLagring.param[
                    _FLASH_PARAM_CAN_ID_ ];
247                 break;
248             case 2:
249                 can1TxMsg.StdId = CAN_MSG_GAIN_z ;//| flashLagring.param[
                    _FLASH_PARAM_CAN_ID_ ];
250                 break;
251             default:
252                 break;
253             }
254
255             for(teller = 0 ; teller <= (nMaalinger+1) ; teller++ ){
256                 //adcVo[] = {0,0,0,0,0,0,0,0,0,0};
257                 /* Beregner ny dac verdi som skal settes ut */
258                 dacVerdi = (vRef_10b - (teller+1)*deltaDAC_10b);
259                 /* Skriver ny verdi til DAC */
260                 skrivEksternDAC(DAC_CMD_CODEn_LOADn|chTeller, dacVerdi );
261                 /* Venter 2s slik at DAC kan stabiliseres
262                  * Benytter SysTick klokken */
263                 tmpTid = systemTid.sysTeller + TIMEOUT_2s;
264                 while( systemTid.sysTeller < tmpTid);
265                 /* Måler Vo til instrumentforsterkere */
266                 adcVo = 0;
267                 for(sampler = 0; sampler < 16; sampler++){
268                     statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
269                     while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på
                        oppdatert måling*/
270                 }
271                 for(sampler = 0; sampler < 128; sampler++){
272                     statusFlagg.adc = FLAGG_ADCSEKVENS_Start;
273                     while( statusFlagg.adc != FLAGG_ADCSEKVENS_Ferdig ); /* Venter på
```

```
                oppdatert måling*/
274             adcVo += ekstADC[ chTeller ].lesteADC_Data;
275         }
276         can1TxMsg.DLC = 6;
277         can1TxMsg.Data[ 0 ] = 0xFF&(adcVo>>24);
278         can1TxMsg.Data[ 1 ] = 0xFF&(adcVo>>16);
279         can1TxMsg.Data[ 2 ] = 0xFF&(adcVo>>8);
280         can1TxMsg.Data[ 3 ] = 0xFF&adcVo;
281         can1TxMsg.Data[ 4 ] = 0xFF&(dacVerdi>>8);
282         can1TxMsg.Data[ 5 ] = 0xFF&(dacVerdi);
283         can1_Tx( &can1TxMsg );
284       }
285     }
286     statusFlagg.adcModus = FLAGG_ADCMODUS_Run;
287 }
288
289 /*****************************************************************************
290  * @brief Beregner faktisk signal basert på forsterkning, Vdac og målt verdi.
291  * @param  none.
292  * @return none
293  *****************************************************************************/
294 void korrigerSignal ( void ){
295
296     uint8_t teller = 0;
297     uint32_t dacVerdi = 0;
298     for( teller = 0 ; teller < ( ADC_Kanaler - 1 ) ; teller++){
299         dacVerdi = ((uint32_t)ekstADC[ teller ].dac<<15)/625;
300
301         /* Beregner faktisk signal
302          *
303          *          [ Vadc(n)*G_nevner ]
304          * Vs(n) =  ------------------------------   +  Vdac(n-1)   +/- offsett
305          *                  G_teller
306          * */
307         ekstADC[ teller ].korrigert.verdi = (uint16_t)((( (uint32_t)ekstADC[ teller ].
            lesteADC_Data )* (uint32_t)ekstADC[ teller ].G_nevner )
308                 / (uint32_t)ekstADC[ teller ].G_teller) + (uint32_t)dacVerdi;
309
310         dacVerdi =  ((5625*(uint32_t)ekstADC[ teller ].korrigert.verdi)/10)>>15;
311         ekstADC[ teller ].dac = (uint16_t)(dacVerdi);
312         skrivEksternDAC(DAC_CMD_CODEn_LOADn|teller, dacVerdi );
313
314         if(ekstADC[teller].fortegn == OFFSETT_FORTEGN_POS){
315             ekstADC[ teller ].korrigert.verdi +=ekstADC[ teller ].offsett;
316         }
317         else{
318             ekstADC[ teller ].korrigert.verdi -=ekstADC[ teller ].offsett;
319         }
320     }
321 }
322
```

```
1   /**
2    * @file hmc2003.h
3    *
4    * @author Jørn Tore Ørsland
5    * @date    12.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Defineringer tilhørende HMC2003 sensor.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  #ifndef _HMC2003_INC_
16  #define _HMC2003_INC_
17  /*-------------------------------------------------------------------------------------------
    ------------
18                                      Inkluderinger
19  -------------------------------------------------------------------------------------------
    ----------*/
20  #include "globDef.h"
21  /*-------------------------------------------------------------------------------------------
    ------------
22                                      Defineringer
23  -------------------------------------------------------------------------------------------
    ----------*/
24  #define VSR_MIN                     2569    /* 2569 tilsvarer 20V VSR */
25  #define MAX_LADETID                 TIMEOUT_2s      /* Lengste ladetid som tillates */
26
27  #define DELTA_DAC                   5       /* Delta endring i DAC ved gain beregning */
28
29  #define nMaalinger_8                8       /* Antal delta målinger som skal utføres
    for å bestemme G */
30  #define nMaalinger                  32
31  #ifdef nMaalinger_8
32  #define nMaalingerDiv               3
33  #endif
34  /*-------------------------------------------------------------------------------------------
    ------------
35                                      Variabler
36  -------------------------------------------------------------------------------------------
    ----------*/
37  struct v{
38      uint8_t oppladet:1;
39      uint16_t VSRoppladet;
40      uint16_t VSRutladet;
41  };
42  extern volatile struct v VSR;
43  /*-------------------------------------------------------------------------------------------
    ------------
44                                      Prototyper
45  -------------------------------------------------------------------------------------------
    ----------*/
46  void initHmc2003( void );
47  void hmc2003_SetReset( void );
48  void transformer_VSR ( void );
49  void beregneGain( void );
50  void korrigerSignal ( void );
51  #endif
52
```

```
1  /**
2   * @file watchdog.c
3   *
4   * @author Jørn Tore Ørsland
5   * @date    05.06.15
6   * @version V1.00
7   * @brief watchdog funksjoner (IWDG kap 23)
8   *
9   * @note
10  * Oppdateringer:
11  * 14.04.16 - Opprettet
12  */
13
14
15 #include "stm32f0xx.h"
16
17
18 /*******************************************************************************
19  * @brief Initierer og starter Independent Watch Dog
20  * @param Prescaler verdi.
21  * @return none
22 *******************************************************************************/
23 void initWatchDog( unsigned char prescaler ){
24
25      IWDG->KR = 0x5555;
26      IWDG->RLR = 0x3E8; /* Teller ned fra denne verdien */
27      IWDG->PR |= prescaler;
28      IWDG->KR = 0xCCCC;
29      IWDG->KR = 0xAAAA;
30 }
31
32
33
34
```

```
1   /**
2    * @file watchdog.h
3    *
4    * @author Jørn Tore Ørsland
5    * @date    14.04.16
6    * @version V1.00
7    * @brief watchdog defineringer og prototyper til watchdog.c (IWDG kap 23)
8    *
9    * @note
10   * Oppdateringer:
11   * 14.04.16 - Opprettet
12   *
13   */
14
15  /** @defgroup watchDog
16   * @brief watchdog funksjoner (IWDG kap 18)
17   */
18
19  /** @addtogroup watchDog */
20  /*@{*/
21
22  #ifndef _WATCHDOG_DEFINES_
23      #define _WATCHDOG_DEFINES_
24
25
        /*-------------------------------------------------------------------------------------
        ----------------
26                                      Prescaler verdier
27
        -------------------------------------------------------------------------------------
        --------------*/
28      /* Prescaler defines */
29      #define _PRESCALER_DIV_4_       0x0     /**< WatchDog LSI Clk Prescaler DIV 4 \n
        Timeout(s): 0.4095*/
30      #define _PRESCALER_DIV_8_       0x1     /**< WatchDog LSI Clk Prescaler DIV 8 \n
        Timeout(s): 0.8190 */
31      #define _PRESCALER_DIV_16_      0x2     /**< WatchDog LSI Clk Prescaler DIV 16 \n
        Timeout(s): 1.6380 */
32      #define _PRESCALER_DIV_32_      0x3     /**< WatchDog LSI Clk Prescaler DIV 32 \n
        Timeout(s): 3.2760 */
33      #define _PRESCALER_DIV_64_      0x4     /**< WatchDog LSI Clk Prescaler DIV 64 \n
        Timeout(s): 6.5520 */
34      #define _PRESCALER_DIV_128_     0x5     /**< WatchDog LSI Clk Prescaler DIV 128 \n
        Timeout(s): 13.1040 */
35      #define _PRESCALER_DIV_256_     0x6     /**< WatchDog LSI Clk Prescaler DIV 256 \n
        Timeout(s): 26.2080 */
36
37
        /*-------------------------------------------------------------------------------------
        ----------------
38                                      Macroer
39
        -------------------------------------------------------------------------------------
        --------------*/
40      #define _klappHunden_           IWDG->KR = 0xAAAA;  /**< WatchDog teller resett */
41
42
        /*-------------------------------------------------------------------------------------
        ----------------
43                                      Prototyper
44
        -------------------------------------------------------------------------------------
        --------------*/
45      void initWatchDog( unsigned char prescaler );
46
47  #endif
48
49  /*@}*/
50
```

```
1   /**
2    * @file globDef.h
3    *
4    * @author Jørn Tore Ørsland
5    * @date    11.04.16
6    * @version V1.00
7    * @brief
8    *
9    * @note Globale defineringer.
10   * Oppdateringer:
11   *
12   *
13   */
14
15  #ifndef _GLOB_INC_
16  #define _GLOB_INC_
17  /*-------------------------------------------------------------------------------------------
    ------------
18                                      Inkluderinger
19  -------------------------------------------------------------------------------------------
    ----------*/
20  #include "stm32f0xx.h"
21  /*-------------------------------------------------------------------------------------------
    ------------
22                                      Defineringer
23  -------------------------------------------------------------------------------------------
    ----------*/
24  #define TRUE                        1
25  #define FALSE                       0
26
27  /* ADC MAX11044 */
28  #define FLAGG_ADCSEKVENS_Ferdig     1   /* Samplet ferdig */
29  #define FLAGG_ADCSEKVENS_Start      2   /* Klar til ny sekvens */
30  #define FLAGG_ADCSEKVENS_Pause      3   /* Venter med å starte ny sekvens*/
31
32  #define FLAGG_ADCMODUS_Pause        0   /* Venter med å starte ny sekvens */
33  #define FLAGG_ADCMODUS_Run          1   /* Starter ny sekvens når forrige er ferdig */
34
35  /* CANBUS */
36  #define FLAGG_CAN_TxSensorData      1   /* Nye målinger er klare for sending */
37
38  /* HMC2003 */
39  #define HMC2003_SEKUND_TIL_SET_RESET        600 /* Tid i sekund mellom hver set/reset */
40  #define FLAGG_HMC2003_Normal        0
41  #define FLAGG_HMC2003_SetReset      1
42  #define TIMEOUT_2s                  8000
43
44  /*-------------------------------------------------------------------------------------------
    ------------
45                                      Variabler
46  -------------------------------------------------------------------------------------------
    ----------*/
47
48  /* Statusflagg som benyttes for å kontrollere utførelsen av programmet */
49  struct flagg{
50      uint16_t adc : 4;
51      uint16_t adcModus : 1;
52      uint16_t internAdc:4;
53      uint16_t can:1;
54      uint16_t hmc2003SR:1;
55      uint16_t pulseVSR:1;
56      uint16_t clkSync:1;
57  };
58  extern volatile struct flagg statusFlagg;
59
60  /* Holder orden på tiden slik at HMC2003 kan utføre set/reset ved bestemte tider */
61  struct tid{
62      uint32_t sekunder;
63      uint16_t sensorReset;
64      uint16_t internAdc;
65      uint32_t sysTeller;
66  };
67
68  extern volatile struct tid systemTid;
```

```
69  /*-------------------------------------------------------------------------------------
    ------------
70                                          Prototyper
71  -------------------------------------------------------------------------------------
    ----------*/
72  #endif
73
```

C.2 MATLAB

```
function [ A_1,sdA1, Z_1,sdZ1, A_2,sdA2, Z_2,sdZ2, X1, Y1, Z1, X2, Y2, Z2 ] =
beregnInvarianter( filnavn )
V = textread(filnavn);

V_s1 = [0 0 0];
V_s2 = [0 0 0];
for idx=1:length(V)
    if(V(idx,1) == 3)
        V_s1 = [V_s1; V(idx,2:4)];
    elseif(V(idx,1) == 15)
        V_s2 = [V_s2; V(idx,2:4)];
    end
end

% Sensor 1
Xs1 = V_s1(200:end-10,1);
Ys1 = V_s1(200:end-10,2);
Zs1 = V_s1(200:end-10,3);

%Konverterer til spenning
x_s1 = Xs1*(5/2^16);
y_s1 = Ys1*(5/2^16);
z_s1 = Zs1*(5/2^16);


%figure();plot(V_xs1,'LineWidth',1.5);hold on;
plot(V_ys1,'LineWidth',1.5);plot(V_zs1,'LineWidth',1.5);legend('X','Y','Z')
%xlabel('Tid');ylabel('V');grid on;
%x_s1 = 0.0001*x_s1;
%y_s1 = 0.0001*y_s1;
%z_s1 = 0.0001*z_s1;

%x_s1 = x_s1- mean(x_s1);
%y_s1 = y_s1-mean(y_s1);
%z_s1 = 0.0001*z_s1;

% figure();plot(x_s1,'LineWidth',1.5);hold on;
plot(y_s1,'LineWidth',1.5);plot(z_s1,'LineWidth',1.5);legend('X','Y','Z');xlabel('Måling
 n');ylabel('T');grid minor;
%  set(findall(gcf,'type','axes'),'fontsize',16,'fontWeight', 'bold')
%  set(findall(gcf,'type','text'),'fontSize',16,'fontWeight', 'bold')
% Sensor 2
Xs2 = V_s2(200:end-10,1);
Ys2 = V_s2(200:end-10,2);
Zs2 = V_s2(200:end-10,3);

% Konverterer til spenning
x_s2 = Xs2*(5/2^16);
y_s2 = Ys2*(5/2^16);
z_s2 = Zs2*(5/2^16);


% figure();plot(x_s2,'LineWidth',1.5);hold on;
plot(y_s2,'LineWidth',1.5);plot(z_s2,'LineWidth',1.5);legend('X','Y','Z');xlabel('Måling
 n');ylabel('V');grid minor;


%figure();plot(x_s2);hold on; plot(y_s2);plot(z_s2);legend('X','Y','Z')
%figure();plot(x_s1);hold on;
plot(y_s1);plot(z_s1);plot(x_s2);plot(y_s2);plot(z_s2);legend('X_1','Y_1','Z_1','X_2','Y
_2','Z_2')
%% FFT av målinger

Fs = 2000;
T = 1/Fs;
L = 400;
f = Fs*(0:0.5:L/2)/L;

% Dele opp i blokker av lengde L
AntallBlokkerS1 = floor( length(x_s1)/L );
AntallBlokkerS2 = floor( length(x_s2)/L );
AntallBlokker = min(AntallBlokkerS1,AntallBlokkerS2)-1;

% Utfører FFT beregning
```

```
67  for idx=1:(AntallBlokker)
68      start= idx + (idx-1)*L;
69      stopp = start+L;
70      X1(idx,:) = fft((x_s1(start:stopp)-mean(x_s1(start:stopp))),2*L)/(2*L);
71      X2(idx,:) = fft((x_s2(start:stopp)-mean(x_s2(start:stopp))),2*L)/(2*L);
72
73      Y1(idx,:) = fft((y_s1(start:stopp)-mean(y_s1(start:stopp))),2*L)/(2*L);
74      Y2(idx,:) = fft((y_s2(start:stopp)- mean(y_s2(start:stopp))),2*L)/(2*L);
75
76      Z1(idx,:) = fft((z_s1(start:stopp)-mean(z_s1(start:stopp))),2*L)/(2*L);
77      Z2(idx,:) = fft((z_s2(start:stopp)-mean(z_s2(start:stopp))),2*L)/(2*L);
78  end
79
80  % Plotter alle FFT
81
82  %  figure();plot(
    f(1:L+1),abs(X1(1,1:L+1)/(2*L)),'LineWidth',1.5);legend('X');xlabel('Frekvens');ylabel('
    |X|');grid on;
83  %  set(findall(gcf,'type','axes'),'fontsize',16,'fontWeight', 'bold')
84  %  set(findall(gcf,'type','text'),'fontSize',16,'fontWeight', 'bold')
85  %   figure();plot(
    f(1:L+1),abs(Y1(1,1:L+1)/(2*L)),'LineWidth',1.5);legend('Y');xlabel('Frekvens');ylabel('
    |Y|');grid on;
86  %  set(findall(gcf,'type','axes'),'fontsize',16,'fontWeight', 'bold')
87  %  set(findall(gcf,'type','text'),'fontSize',16,'fontWeight', 'bold')
88  %   figure();plot(
    f(1:L+1),abs(Z1(1,1:L+1)/(2*L)),'LineWidth',1.5);legend('Z');xlabel('Frekvens');ylabel('
    |Z|');grid on;
89
90  % figure();plot(
    f(1:L+1),abs(X2(1,1:L+1)/(2*L)),'LineWidth',1.5);legend('X');xlabel('Frekvens');ylabel('
    |X|');grid on;
91  % figure();plot(
    f(1:L+1),abs(Y2(1,1:L+1)/(2*L)),'LineWidth',1.5);legend('Y');xlabel('Frekvens');ylabel('
    |Y|');grid on;
92  % figure();plot(
    f(1:L+1),abs(Z2(1,1:L+1)/(2*L)),'LineWidth',1.5);legend('Z');xlabel('Frekvens');ylabel('
    |Z|');grid on;
93
94  %axes('FontSize',14);
95  % Invariantberegninger
96  % 1 - A_theta = atan2( Ay, Ax )
97  %       Ay = abs(FFTy)
98  % 2 - Z_theta = Anglediff( dirac_y , dirac_x )
99  %        dirac_y = atan2( Im(FFTy), Re(FFT_y) )
100
101 Ay1 = abs(Y1(:,41));
102 Ax1 = abs(X1(:,41));
103 Az1 = abs(Z1(:,41));
104 A_theta1 = atan2( Ay1, Ax1);
105
106 dy1 = atan2( imag(Y1(1:AntallBlokker,41)), real(Y1(1:AntallBlokker,41)) );
107 dx1 = atan2( imag(X1(1:AntallBlokker,41)), real(X1(1:AntallBlokker,41)) );
108 for idx=1:length(dx1)
109     Z_theta1(idx) = AngleDiff(dy1(idx),dx1(idx));
110 end
111
112 Ay2 = abs(Y2(1:AntallBlokker,41));
113 Ax2 = abs(X2(1:AntallBlokker,41));
114 A_theta2 = atan2( Ay2, Ax2);
115
116 dy2= atan2( imag(Y2(1:AntallBlokker,41)), real(Y2(1:AntallBlokker,41)) );
117 dx2 = atan2( imag(X2(1:AntallBlokker,41)), real(X2(1:AntallBlokker,41)) );
118
119 for idx=1:length(dx2)
120     Z_theta2(idx) = AngleDiff(dy2(idx),dx2(idx));
121 end
122 A_1 = mean(A_theta1);
123 A_2 = mean(A_theta2);
124 Z_1 = mean(Z_theta1);
125 Z_2 = mean(Z_theta2);
126 sdA1 = std(A_theta1);
127 sdA2 = std(A_theta2);
128 sdZ1 = std(Z_theta1);
```

```
129 sdZ2 = std(Z_theta2);
130
131
132 %  figure();plot(A_theta1(1:end-1),'LineWidth',1.5);legend('A_{\theta,1}');xlabel('FFT
    vindu');ylabel('A_y / A_x');grid on;
133 %  set(findall(gcf,'type','axes'),'fontsize',16,'fontWeight', 'bold')
134 %  set(findall(gcf,'type','text'),'fontSize',16,'fontWeight', 'bold')
135 % %
136 %  figure();histfit( A_theta1(1:end-1));ylabel('Frekvens');xlabel('A_{\theta,1}');grid
    on;
137 %  set(findall(gcf,'type','axes'),'fontsize',16,'fontWeight', 'bold')
138 %  set(findall(gcf,'type','text'),'fontSize',16,'fontWeight', 'bold')
139 % %
140 %  figure();plot(Z_theta1(1:end-1),'LineWidth',1.5);legend('Z_{\theta,1}');xlabel('FFT
    vindu');ylabel('atan2(\delta_y / \delta_x)');grid on;
141 %  set(findall(gcf,'type','axes'),'fontsize',16,'fontWeight', 'bold')
142 %  set(findall(gcf,'type','text'),'fontSize',16,'fontWeight', 'bold')
143 % %
144 %  figure();histfit( Z_theta1(1:end-1));ylabel('Frekvens');xlabel('Z_{\theta,1}');grid
    on;
145 %  set(findall(gcf,'type','axes'),'fontsize',16,'fontWeight', 'bold')
146 %  set(findall(gcf,'type','text'),'fontSize',16,'fontWeight', 'bold')
147 % %
148 % % %end
149 %  figure();plot(Ay1,'LineWidth',1.5);hold
    on;plot(Ax1,'LineWidth',1.5);legend('A_{y,1}','A_{x,1}')
150 % figure();plot(Ay2,'LineWidth',1.5);hold
    on;plot(Ax2,'LineWidth',1.5);legend('A_{y,1}','A_{x,1}')
151 % figure();plot(dy1,'LineWidth',1.5);hold
    on;plot(dx1,'LineWidth',1.5);plot(Z_theta1,'LineWidth',1.5);legend('\delta_{y,1}','\delt
    a_{x,1}','Z_{\theta1}')
152 %  figure();plot(dy1,'LineWidth',1.5);hold
    on;plot(dx1,'LineWidth',1.5);plot(Z_theta1,'LineWidth',1.5);legend('\delta_{y,1}','\delt
    a_{x,1}','Z_{\theta1}')
153 %  xlabel('FFT vindu');ylabel('Z_\theta');grid on;
154 %  set(findall(gcf,'type','axes'),'fontsize',16,'fontWeight', 'bold')
155 %  set(findall(gcf,'type','text'),'fontSize',16,'fontWeight', 'bold')
156 % figure();plot(dy2,'LineWidth',1.5);hold
    on;plot(dx2,'LineWidth',1.5);legend('d_{y,2}','d_{x,2}')
157 % figure();plot(A_theta1,'LineWidth',1.5);hold
    on;plot(A_theta2,'LineWidth',1.5);legend('A_{\theta,1}','A_{\theta,2}')
158 % figure();plot(Z_theta1,'LineWidth',1.5);hold
    on;plot(Z_theta2,'LineWidth',1.5);legend('Z_{\theta,1}','Z_{\theta,2}')
159 end
160
```

```
function [] = simulerMagnetfelt( steg,deg );
% Konstanter
I=[100:100:2400];
Xc      = 15;  % Senterposisjon x-akse
Yc      = 15;  % Senterposisjon y-akse
lengde  = 30;
%steg    = 0.1; % Oppløsning i meter
dx      = 5.0;  %Høyde over/under linja som ikke tas med
dy      = 11.5;  % Bredde fra senter som ikke tas med
lederDiff = 9;    % Avstand mellom ledere
%deg     = [0];   % Grader
deltax = zeros(length(I),length(deg),4);
deltay = zeros(length(I),length(deg),4);
minimaltFeltx= zeros(length(I),length(deg));
maksimalFeltx= zeros(length(I),length(deg));
minimaltFelty= zeros(length(I),length(deg));
maksimalFelty= zeros(length(I),length(deg));

% Simulering
for i=1:length(I)
    for d=1:length(deg)
        [Bx, By] = biot2D( I(i),steg, lederDiff, Xc,Yc, deg(d)  );
        [deltax(i,d,:), minimaltFeltx(i,d) ,maksimalFeltx(i,d),snittXx(i,d),snittXy(i,d
        ),sigmaXx(i,d),sigmaXy(i,d),feltStyrkeMuX(i,d),feltStyrkeSigmaX(i,d)] =
        finnFelt( Bx, Xc, Yc, dx,dy,lengde,steg );
        [deltay(i,d,:), minimaltFelty(i,d) ,maksimalFelty(i,d),snittYx(i,d),snittYy(i,d
        ),sigmaYx(i,d),sigmaYy(i,d),feltStyrkeMuY(i,d),feltStyrkeSigmaY(i,d)] =
        finnFelt( By, Xc, Yc, dx,dy,lengde,steg );
    end
end

filnavn = sprintf('%s_%.1fmAvstMellomLedere_%.4f_MedDeg.tex',date,lederDiff,steg);
FID = fopen(filnavn, 'w');
fprintf(FID, '\\begin{tabular}{ccccccccc}\\hline \n');
fprintf(FID, 'Strøm & Grader & Maksimal Feltx & Minmalt Feltx&Maksimal Felty & Minmalt
Felty & Vmaks&Vmin&Hmaks&Hmin\\\\ \\hline \n');
for i = 1:length(I)
    for k=1:length(deg)
        fprintf(FID, '%8.2d & %8.2d & %8.2e & %8.2e  & %8.2e  & %8.2e  & %8.2e  &
        %8.2e  & %8.2e  & %8.2e & %8.2e \\\\ ', I(i), deg(k), maksimalFeltx(i,d),
        minimaltFeltx(i,d),maksimalFelty(i,d),  minimaltFelty(i,d),deltax(i,d,1),deltax
        (i,d,2),deltax(i,d,3),deltax(i,d,4));
        fprintf(FID, '\n');
    end

end
fprintf(FID, '\\end{tabular}\n');
fclose(FID);
filnavn = sprintf('%s_%.1fmAvstMellomLedere_%.4f.tex',date,lederDiff,steg);
FID = fopen(filnavn, 'w');
fprintf(FID, '\\begin{tabular}{ccccccccc}\\hline \n');
fprintf(FID, 'Strøm & Maksimal Feltx & Minmalt Feltx&Maksimal Felty & Minmalt Felty &
Vmaks&Vmin&Hmaks&Hmin\\\\ \\hline \n');
for i = 1:length(I)
    for k=1:length(deg)
        fprintf(FID, '%8.2d & %8.2e & %8.2e  & %8.2e  & %8.2e  & %8.2e  & %8.2e  &
        %8.2e  & %8.2e \\\\ ', I(i), maksimalFeltx(i,d),  minimaltFeltx(i,d),
        maksimalFelty(i,d),  minimaltFelty(i,d),deltax(i,d,1),deltax(i,d,2),deltax(i,d,
        3),deltax(i,d,4));
        fprintf(FID, '\n');
    end

end
fprintf(FID, '\\end{tabular}\n');
fclose(FID);

%Latex tabell: Fordeling av x og y felt
filnavn = sprintf('%s_%.1fmFeltFordeling_%.4f.tex',date,lederDiff,steg);
FID = fopen(filnavn, 'w');
fprintf(FID, '\\begin{tabular}{ccccccc}\\hline \n');
fprintf(FID, 'Strøm & Maksimal Feltx & $\mu_x$& $\sigma_x$ & maxy & $\mu_y$ &
$\sigma_y$\\\\ \\hline \n');
for i = 1:length(I)
    for k=1:length(deg)
```

```
62          fprintf(FID, '%8.2d & %8.2e & %8.2e  & %8.2e  & %8.2e  & %8.2e  & %8.2e \\\\ ',
             I(i), maksimalFeltx(i,d),  feltStyrkeMuX(i,d),feltStyrkeSigmaX(i,d),
            maksimalFelty(i,d),  feltStyrkeMuY(i,d),feltStyrkeSigmaY(i,d));
63          fprintf(FID, '\n');
64      end
65
66  end
67  fprintf(FID, '\\end{tabular}\n');
68  fclose(FID);
69
70  %Latex tabell: Fordeling av deltaverdier av x og y felt
71  filnavn = sprintf('%s_%.1fmDeltaTabell_%.4f.tex',date,lederDiff,steg);
72  FID = fopen(filnavn, 'w');
73  fprintf(FID, '\\begin{tabular}{ccccccccc}\\hline \n');
74  fprintf(FID, 'Strøm & Grader & Maksimal Feltx & Minmalt Feltx&Maksimal Felty & Minmalt
    Felty & Vmaks&Vmin&Hmaks\\\\ \\hline \n');
75  for i = 1:length(I)
76      fprintf(FID, '%8.2d & %8.2d & %8.2e & %8.2e  & %8.2e  & %8.2e  & %8.2e  & %8.2e  &
        %8.2e  \\\\ ', I(i), snittXx(i,d), sigmaXx(i,d),snittXy(i,d),sigmaXy(i,d),snittYx(i
        ,d), sigmaYx(i,d),snittYy(i,d),sigmaYy(i,d));
77      fprintf(FID, '\n');
78  end
79
80
81
82
83
84
85
86
87  % Lagrer workspace variabler
88  filnavn = sprintf('%s_maksimalFelt_%.4f_x.mat',date,steg);
89  save( filnavn,'maksimalFeltx');
90  filnavn = sprintf('%s_maksimalFelt_%.4f_y.mat',date,steg);
91  save( filnavn ,'maksimalFelty');
92  filnavn = sprintf('%s_wrkSpace_%.4f_y.mat',date,steg);
93  save(filnavn);
94
```

```
function [delta, minimaltFelt ,maksimalFelt,snittx,snitty,sigmax,sigmay, feltStyrkeMu,
feltStyrkeSigma] = finnFelt( B, Xc, Yc, dx,dy,lengde,steg )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Funksjonen finner forskjell på minste og største feltstyrke som er i
%  variabel B
%
%  delta - Inneholder største og minste endring mellom to naboceller
%  min   - Minste verdi som er funnet (større enn 0)
%  max   - Største verdi som er funnet.

% Oppretter variabler
deltaXmax = 0.0;    % Største deltaverdi i x-akse
deltaXmin = 100.0;  % Minste deltaverdi i x-akse
deltaYmax = 0.0;    % Største deltaverdi i y-akse
deltaYmin = 100.0;    % Minste deltaverdi i y-akse
%maks = 0.0;          % Største felt funnet
%minste = 100.0;        % Minste felt funnet

%LengdeX = 30;  % Lengde i meter i x-akse
%LengdeY = 30;  % Lengde i meter i y-akse
%   Xc      = 15;  % Senterposisjon x-akse
%   Yc      = 15;  % Senterposisjon y-akse
%
%   lengde = 30;
%   steg    = 0.1; % Oppløsning i meter
%  dx = 2.5;
%  dy=7.5;
%antElementerX = LengdeX/steg;    % Antall elementer i x-akse
%antElementerY = LengdeY/steg;    % Antall elementer i y-akse
akse = (0:steg:lengde);


% Søker igjennom og fjerner 0.0 felt.
for i = 1:(length(B)-1)
    for k = 1:(length(B(i,:))-1)
        if B(i,k)==0.0
            if (( i-1 > 0 ) && (i+1<length(B(i,:))))
                B(i,k) = (B(i-1,k) + B(i+1,k))/2;
            else
                B(i,k) = B(i+1,k);
            end
        end
        if k == (length(B(i,:))-1)
            B(i,k+1) = B(i,k);
        end
    end
    if i == (length(B)-1)
        B(i+1,:)=B(i,:);
    end
end

% Finner koordinater til punktene rundt lederne.
[senterX] = find(akse==Xc);
[senterY] = find(akse==Yc);
[Xstart]  = find(akse==(Xc-dx));
[Xstopp]  = find(akse==(Xc+dx));
[Ystart]  = find(akse==(Yc-dy));
[Ystopp]  = find(akse==(Yc+dy));

% Starter med å finne minste verdi
for i = 1:(length(B)-1)
    minste(i) = min(abs(B(i,1:end-1)));
end

% Fjerner området rundt lederne
B(Xstart:Xstopp,Ystart:Ystopp)=0;
%figure;mesh(B);
% Søker gjennom B og leter etter største differanse mellom cellene samt
%  største og minste felt.
for i = 1:(length(B)-1) % Søker igjennom radene
    maks(i) = max(abs(B(i,:)));
    [fltSt, n] =sumabs(abs(B(i,:)));  %Samler den totale feltstyrken
    feltstyrkeTmp(i) = fltSt/n;
    SD(i) = std(B(i,:));
```

```
74        for k=1:(length(B(i,:))-1) % Søker gjennom kolonnene
75            if B(i,k)~=0
76                if B(i,k+1)~=0
77                    diffX(i,k) = abs(B(i,k)-B(i,k+1));
78                else
79                    diffX(i,k) = 0;
80                end
81                if B(i+1,k)~=0
82                    diffY(i,k) = abs(B(i,k)-B(i+1,k));
83                else
84                    diffY(i,k) = 0;
85                end
86            else
87                diffX(i,k) = 0;
88                diffY(i,k) = 0;
89            end
90        end
91    end
92
93    % Beregner feltstyrke mu
94    [feltStyrkeMu] = sum(feltstyrkeTmp)/length(feltstyrkeTmp);
95    for idx=1:length(feltStyrkeMu)
96        feltStyrkeSigma = sqrt((1/n)*sum((feltstyrkeTmp(idx)-feltStyrkeMu)^2));
97    end
98
99
100
101   diffxRed = diffX(diffX~=0);
102   [s,n] = sumabs(diffxRed);
103   snittx = s/n;
104   for idx=1:length(diffxRed)
105       sigmax = sqrt((1/n)*sum((diffxRed(idx)-snittx)^2));
106   end
107   diffyRed = diffY(diffY~=0);
108   [s,n] = sumabs(diffyRed);
109   snitty = s/n;
110   for idx=1:length(diffyRed)
111       sigmay = sqrt((1/n)*sum((diffyRed(idx)-snitty)^2));
112   end
113
114   maksimalFelt = max(maks);
115
116   deltaXmax = max(max(diffX));
117   deltaYmax = max(max(diffY));
118
119   diffY(diffY==0)=inf;
120   diffX(diffX==0)=inf;
121   minste(minste==0)=inf;
122   minimaltFelt = min(minste);
123   deltaXmin = min(min(diffX));
124   deltaYmin = min(min(diffY));
125
126   delta=[deltaXmax deltaXmin deltaYmax deltaYmin];
127   end
128
```

```
function [B] = biotSavart( I,Xc,Yc,x,y )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% biotSavart.m
%
% Funksjonen beregner feltstyrke i et gitt punkt (x,y),
%  der leder er plassert med senter i (Xc,Yc)
%
%   I  - Strøm i leder
%   Xc - Senterposisjon x-akse
%   Yc - Senterposisjon y-akse
%   x  - Punktets posisjon i x-akse
%   y  - Punktets posisjon i x-akse
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initiering av variabler og konstanter
  u0 = 4*pi*10^-7;
  dx = x-Xc;
  dy = y-Yc;

% Biot Savart
  Bx = (u0*I*dy)/(2*pi*(dx^2+dy^2));
  By = -(u0*I*dx)/(2*pi*(dx^2+dy^2));
  B = [Bx By];
end


```

```
function [Bx By] = biot2D( I,steg, lederDiff, Xc,Yc, t  )
% Konstanter
  %I = 2400;       % Strøm i ampere
  LengdeX = 30;  % Lengde i meter i x-akse
  LengdeY = 30;  % Lengde i meter i y-akse
  %Xc      = 15;  % Senterposisjon x-akse
  %Yc      = 15;  % Senterposisjon y-akse
  %steg    = 0.1; % Oppløsning i meter
  antElementerX = LengdeX/steg;    % Antall elementer i x-akse
  antElementerY = LengdeY/steg;    % Antall elementer i y-akse
  Xakse = (0:steg:LengdeX);
  Yakse = (0:steg:LengdeY);

% Oppretter areal
  Bx = zeros(antElementerX,antElementerY);
  By = zeros(antElementerX,antElementerY);

% Beregner feltstyrke
  for a=1:(antElementerX-1)  % x-akse først
      for b=1:(antElementerY-1)  % deretter y-akse
          B = biotSavart(I*sind(t),Xc,Yc, Xakse(a), Yakse(b));
          B2 = biotSavart(I*sind(t+120),Xc,Yc-lederDiff, Xakse(a), Yakse(b));
          B3 = biotSavart(I*sind(t+240),Xc,Yc+lederDiff, Xakse(a), Yakse(b));
         Bx(a,b) = B(1)+B2(1)+B3(1);
         By(a,b) = B(2)+B2(2)+B3(2);
         V(a,b)=atan2(By(a,b),Bx(a,b));
      end
  end

 % B=[Bx By];

  if 0
    l = sqrt(Bx.^2);
    m = sqrt(By.^2);
    figure(1);mesh(l,m)
    figure(2);mesh(Bx,By)
    figure(3); quiver(By,Bx)
    figure(4);mesh(V)
    %figure; bar3(Bx,By)
  end
end
```

C.3 Grafisk brukergrensesnitt

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using canlibCLSNET;
using System.ComponentModel;

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    ///
    public partial class MainWindow : Window
    {
        private int handle = -1;
        private int channel = -1;
        private int readHandle = -1;
        private bool onBus = false;
        private BackgroundWorker dumper;
        private bool fileOpen = false;
        List<String> _list = new List<String>();
        private int fillIndex = 0;
        private int melding = 0;
        private string mottakerId = "";
        private int dlcReg;

        public MainWindow()
        {
            InitializeComponent();
            Loaded += OnLoaded;

        }

        private void OnLoaded(object sender, RoutedEventArgs routedEventArgs)
        {
            Initialize();


            //Sets up a BackgroundWorker and adds delegates to
            //the DumpMessageLoop and ProcessMessage methods
            dumper = new BackgroundWorker();
            dumper.DoWork += DumpMessageLoop;
            dumper.WorkerReportsProgress = true;
            dumper.ProgressChanged += new ProgressChangedEventHandler(ProcessMessage);


            //string lines = "First line.\r\nSecond line.\r\nThird line.";
            //file.WriteLine(lines);
            //file.Close();
        }

        private void Start_OnClickButtonClick(object sender, RoutedEventArgs e)
        {
            Initialize();
            Loaded += OnLoaded;

            channel = Convert.ToInt32(channelBox.Text);
            int hnd = Canlib.canOpenChannel(channel, Canlib.canOPEN_ACCEPT_VIRTUAL);

            CheckStatus("Open channel", (Canlib.canStatus)hnd);
            if (hnd >= 0)
            {
                handle = hnd;
```

```
75              }
76
77
78              const int bitrate = Canlib.BAUD_1M;
79
80              Canlib.canStatus nyStatus = Canlib.canSetBusParams(handle, bitrate, 0, 0, 0
                , 0, 0);
81
82              //CheckStatus("Setting bitrate to " +
                ((ComboBoxItem)bitrateBox.SelectedValue).Content, nyStatus);
83
84
85              nyStatus = Canlib.canBusOn(handle);
86              CheckStatus("Bus on", nyStatus);
87              if (nyStatus == Canlib.canStatus.canOK)
88              {
89                  onBus = true;
90
91                  //This starts the DumpMessageLoop method
92                  if (!dumper.IsBusy)
93                  {
94                      dumper.RunWorkerAsync();
95                  }
96              }
97
98
99          }
100
101         //Initializes Canlib
102         private void initButtonClick(object sender, RoutedEventArgs e)
103         {
104            Initialize();
105         }
106
107         private void Initialize()
108         {
109             Canlib.canInitializeLibrary();
110             statusText.Text = "Canlib initialized";
111         }
112
113
114         //Opens the channel selected in the "Channel" input box
115         private void openChannelButton_Click(object sender, RoutedEventArgs e)
116         {
117             channel = Convert.ToInt32(channelBox.Text);
118             int hnd = Canlib.canOpenChannel(channel, Canlib.canOPEN_ACCEPT_VIRTUAL);
119
120             CheckStatus("Open channel", (Canlib.canStatus)hnd);
121             if (hnd >= 0)
122             {
123                 handle = hnd;
124             }
125         }
126
127         //Sets the bitrate
128         private void setBitrateButton_Click(object sender, RoutedEventArgs e)
129         {
130
131
132             int bitrate = Canlib.BAUD_1M;
133
134
135             Canlib.canStatus status = Canlib.canSetBusParams(handle, bitrate, 0, 0, 0,
                0, 0);
136
137
138         }
139
140         //Goes on bus
141         private void busOnButton_Click(object sender, RoutedEventArgs e)
142         {
143             Canlib.canStatus status = Canlib.canBusOn(handle);
144             CheckStatus("Bus on", status);
145             if (status == Canlib.canStatus.canOK)
```

```
146             {
147                 onBus = true;
148 
149                 //This starts the DumpMessageLoop method
150                 if (!dumper.IsBusy)
151                 {
152                     dumper.RunWorkerAsync();
153                 }
154             }
155         }
156 
157         //Reads message data from user input and writes a message to the channel
158         private void sendButton_Click(object sender, RoutedEventArgs e)
159         {
160             const int grense = 65535;
161 
162             if (melding > 0 && mottakerId.Length > 0 && Convert.ToInt32(dataBox0_1.Text
                ) < grense && Convert.ToInt32(dataBox2_3.Text) < grense && Convert.ToInt32(
                dataBox4_5.Text) < grense && Convert.ToInt32(dataBox6_7.Text) < grense)
163             {
164 
165                 int id = Convert.ToInt32(mottakerId);
166 
167                 TextBox[] textBoxes = {dataBox0_1, dataBox2_3, dataBox4_5, dataBox6_7};
168 
169                 byte[] data = new byte[8];
170                 int k = 0;
171 
172                 for (int j = 0; j < 4; j++)
173                 {
174                     data[k] = (byte)(Convert.ToInt16(textBoxes[j].Text) & 0xff);
175                     data[k + 1] = (byte)((Convert.ToInt16(textBoxes[j].Text) >> 8) &
                        0xff);
176 
177                     k = k + 2;
178                 }
179 
180 
181 
182                 CheckBox[] boxes = {RTRBox, STDBox, EXTBox, WakeUpBox, NERRBox,
                    errorBox,
183                                     TXACKBox, TXRQBox, delayBox, BRSBox, ESIBox};
184                 int flags = 0;
185                 foreach (CheckBox box in boxes)
186                 {
187                     if (box.IsChecked.Value)
188                     {
189                         flags += Convert.ToInt32(box.Tag);
190                     }
191                 }
192 
193                 int dlc = dlcReg;
194 
195                 string msg = String.Format("{0}  {1}  {2:x2} {3:x2} {4:x2} {5:x2}
                    {6:x2} {7:x2} {8:x2} {9:x2}   to handle {10}",
196                                            id, dlc, data[0], data[1], data[2],
                                               data[3], data[4],
197                                            data[5], data[6], data[7], handle);
198                 Canlib.canStatus status = Canlib.canWrite(handle, id, data, dlc, flags);
199                 CheckStatus("Writing message " + msg, status);
200 
201                 sendeStatus.Text = "Melding sendt";
202 
203             }
204             else
205             {
206                 sendeStatus.Text = "Feil med melding";
207             }
208 
209         }
210 
211 
212         private void busOffButton_Click(object sender, RoutedEventArgs e)
213         {
```

```
214             Canlib.canStatus status = Canlib.canBusOff(handle);
215             CheckStatus("Bus off", status);
216             onBus = false;
217         }
218
219         private void closeButton_Click(object sender, RoutedEventArgs e)
220         {
221             Canlib.canStatus status = Canlib.canClose(handle);
222             CheckStatus("Closing channel", status);
223             handle = -1;
224         }
225
226         /*
227          * Looks for messages and sends them to the output box.
228          */
229         private void DumpMessageLoop(object sender, DoWorkEventArgs e)
230         {
231             BackgroundWorker worker = sender as BackgroundWorker;
232             Canlib.canStatus status;
233             int id;
234             byte[] data = new byte[8];
235             int dlc;
236             int flags;
237             long time;
238             bool noError = true;
239             string msg;
240             int xakse;
241             int yakse;
242             int zakse;
243             string Linje;
244
245             //Open up a new handle for reading
246             readHandle = Canlib.canOpenChannel(channel, Canlib.canOPEN_ACCEPT_VIRTUAL);
247
248             status = Canlib.canBusOn(readHandle);
249
250             while (noError && onBus && readHandle >= 0)
251             {
252                 status = Canlib.canReadWait(readHandle, out id, data, out dlc, out
                    flags, out time, 50);
253
254                 if (status == Canlib.canStatus.canOK)
255                 {
256                     if ((flags & Canlib.canMSG_ERROR_FRAME) == Canlib.
                        canMSG_ERROR_FRAME)
257                     {
258                         msg = "***ERROR FRAME RECEIVED***";
259                     }
260                     else
261                     {
262                         msg = String.Format("{0}  {1}  {2:x2} {3:x2} {4:x2} {5:x2}
                            {6:x2} {7:x2} {8:x2} {9:x2}   {10}\r",
263                                             id, dlc, data[0], data[1], data[2],
                                                data[3], data[4],
264                                             data[5], data[6], data[7], time);
265
266                         xakse = data[0] << 8 | data[1];
267                         yakse = data[2] << 8 | data[3];
268                         zakse = data[4] << 8 | data[5];
269
270                        Linje = id + " " + xakse + " " + yakse + " " + zakse;
271
272
273                         if (fileOpen)
274                         {
275                             string Sti = System.IO.Directory.GetCurrentDirectory();
276
277                             Sti = Sti + "\\OutputFile\\";
278
279
280                             _list.Add(Linje);
281                         }
282                     }
283
```

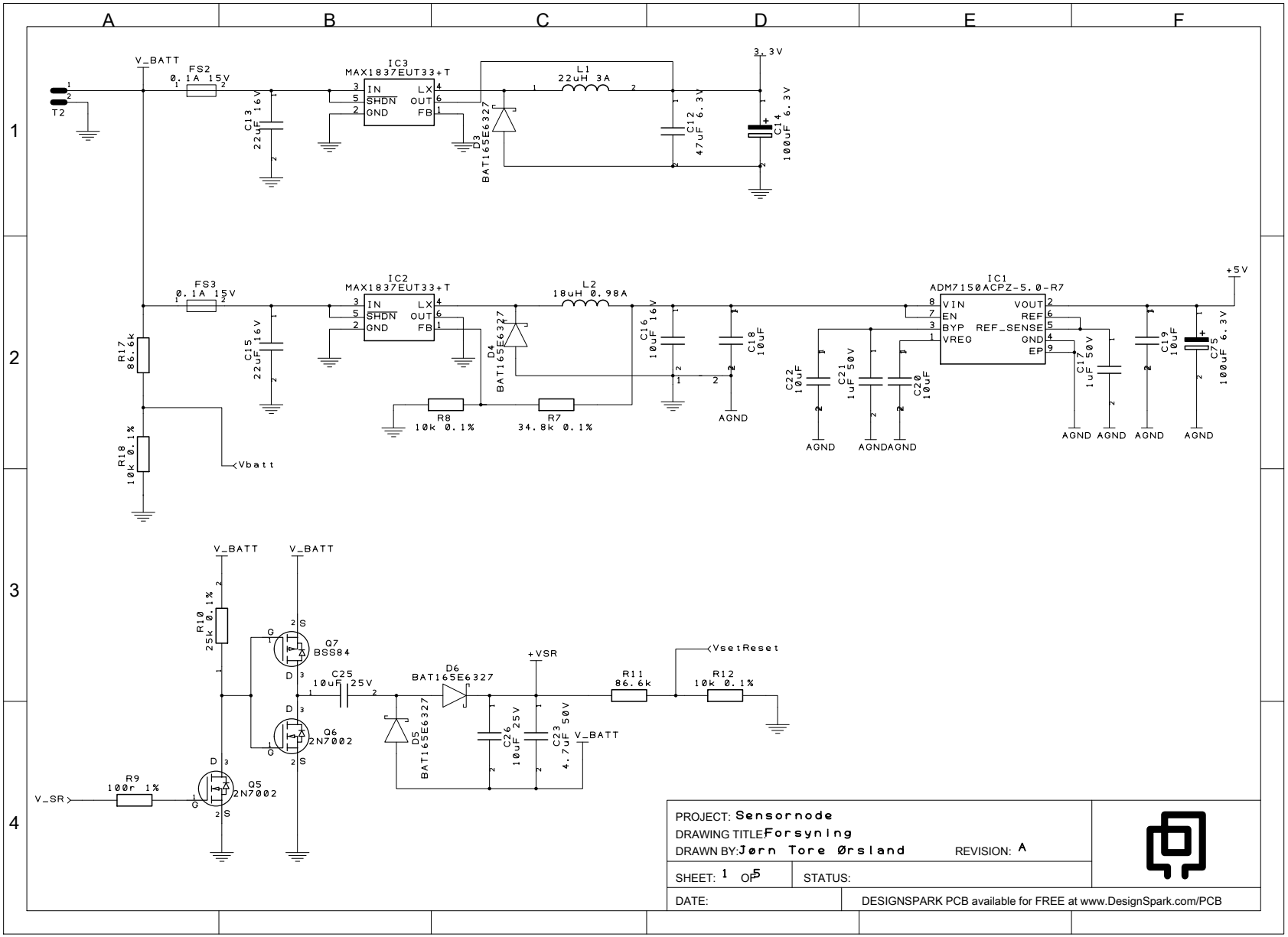
```
284
285                 }
286
287                 else if (status != Canlib.canStatus.canERR_NOMSG)
288                 {
289                     worker.ReportProgress(100, status);
290                 }
291             }
292             Canlib.canBusOff(readHandle);
293         }
294
295         /*
296          * Adds the messages to the output box
297          */
298         private void ProcessMessage(object sender, ProgressChangedEventArgs e)
299         {
300             if (e.ProgressPercentage == 0)
301             {
302                 string output = (string)e.UserState;
303                 outputBox.AppendText(output);
304                 outputBox.ScrollToEnd();
305             }
306             else
307             {
308                 CheckStatus("Reading", (Canlib.canStatus)e.UserState);
309             }
310         }
311
312         /*
313          * Makes sure that only positive integers can be used as input
314          */
315         private void CheckTextBox(object sender, TextCompositionEventArgs e)
316         {
317             e.Handled = !isInt(e.Text);
318         }
319
320         private bool isInt(String s)
321         {
322             foreach (char c in s.ToCharArray())
323             {
324                 if (!char.IsDigit(c))
325                 {
326                     return false;
327                 }
328             }
329             return true;
330         }
331
332         /*
333          * Updates the status bar, prints error message if something goes wrong
334          */
335         private void CheckStatus(String action, Canlib.canStatus status)
336         {
337             if (status != Canlib.canStatus.canOK)
338             {
339                 String errorText = "";
340                 Canlib.canGetErrorText(status, out errorText);
341                 statusText.Text = action + " failed: " + errorText;
342             }
343             else
344             {
345                 statusText.Text = action + " succeeded";
346             }
347         }
348
349         private void ApneFil_OnClickButtonClick(object sender, RoutedEventArgs e)
350         {
351
352
353             fileOpen = true;
354
355
356             Filstatus.Text = "Logges til fil";
357         }
```

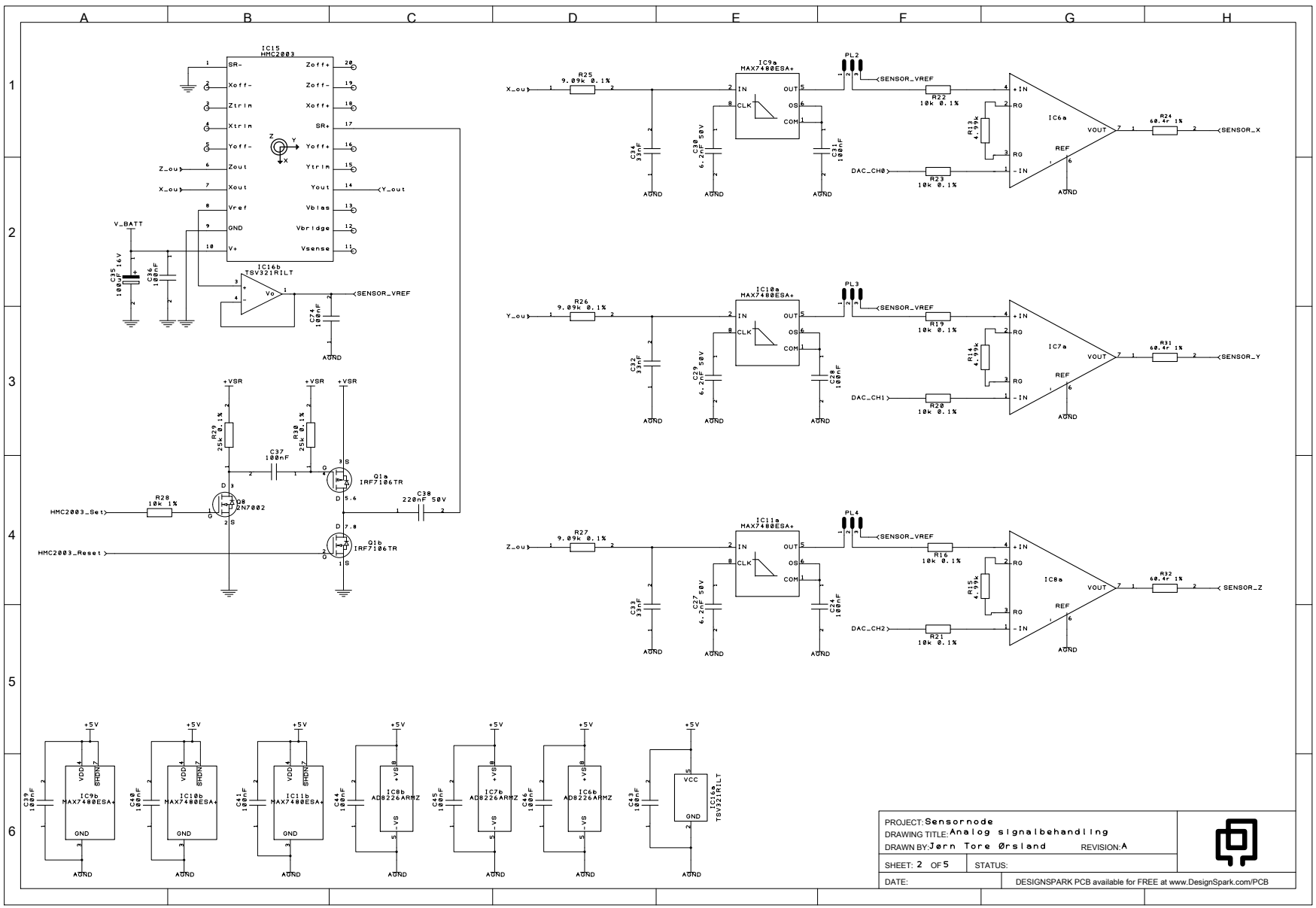
```
358
359
360             private void LukkFil_OnClickFil_OnClickButtonClick(object sender,
                RoutedEventArgs e)
361             {
362                 fileOpen = false;
363                 string Sti = System.IO.Directory.GetCurrentDirectory();
364
365                 Sti = Sti + "\\OutputFile\\";
366
367                 string Tillegg = filtillegg.Text;
368
369
370
371                 File.WriteAllLines(Sti + "\\Logg" + Tillegg + filIndex + ".txt", _list);
372
373
374                 _list.Clear();
375
376                 filIndex++;
377
378                 Filstatus.Text = "Fillogging stoppet";
379
380
381             }
382
383             private void MeldingType_SelectionChanged(object sender,
                SelectionChangedEventArgs e)
384             {
385                 int[] meldinger = new int[10] { 0, 50, 10, 20, 30, 40, 60, 80, 70, 90};
386                 melding = meldinger[MeldingType.SelectedIndex];
387
388                 int[] dlcRegs = new int[10] { 6, 1, 6, 6, 6, 6, 1, 5, 1, 1 };
389                 dlcReg = dlcRegs[MeldingType.SelectedIndex];
390
391             }
392
393             private void SensorID_SelectionChanged(object sender, SelectionChangedEventArgs
                 e)
394             {
395                 string[] mottakerIder = new string[10] { "0", "0001", "0010", "0100",
                    "0110", "1000", "1010", "1100", "1110", "1111" };
396                 mottakerId = mottakerIder[SensorID.SelectedIndex];
397
398             }
399
400         }
401     }
402
```

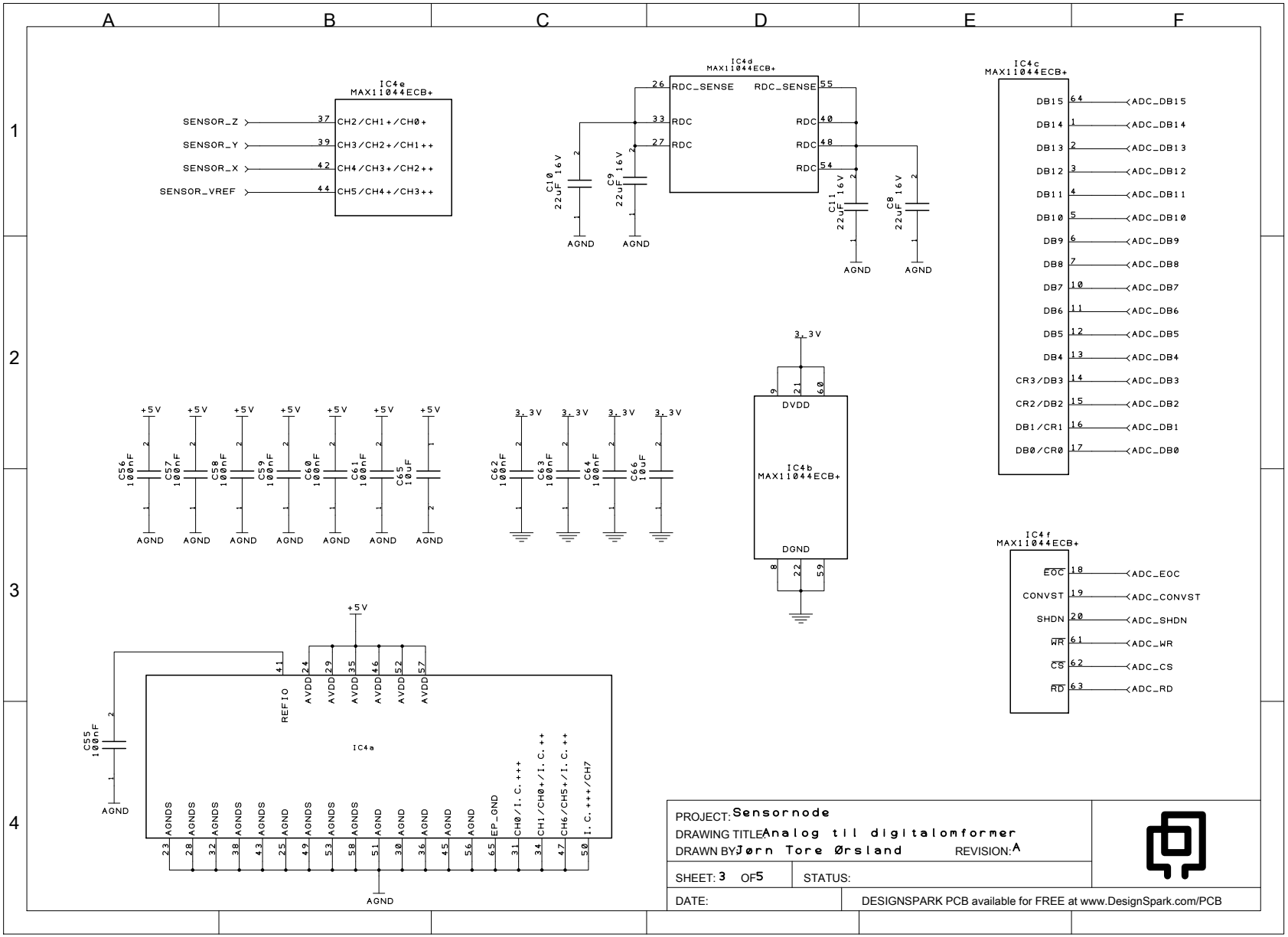
Tillegg D

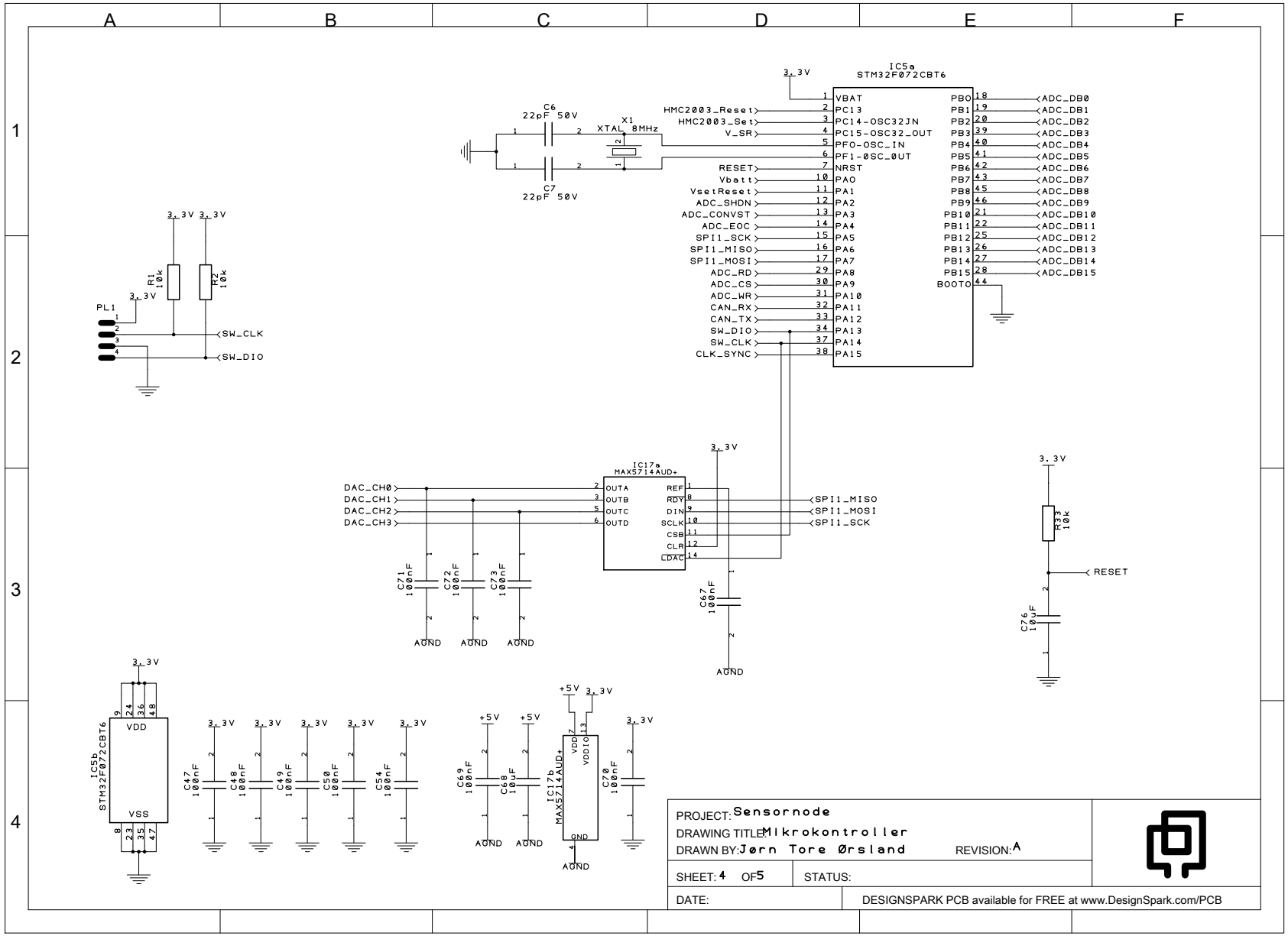
Kretskort

D.1 Sensornode


A
B
C
D
E
F
1
2
3
4
V_BATT
FS2
0.1A 15V
T2
C13
22uF 16V
IC3
MAX1837EUT33+T
IN
SHDN
GND
LX
OUT
FB
L1
22uH 3A
D3
BAT165E6327
C12
47uF 6.3V
3.3V
C14
100uF 6.3V
FS3
0.1A 15V
C15
22uF 16V
IC2
MAX1837EUT33+T
L2
18uH 0.98A
D4
BAT165E6327
C16
10uF 16V
C18
10uF
AGND
R17
86.6k
R18
10k 0.1%
Vbatt
R8
10k 0.1%
R7
34.8k 0.1%
IC1
ADM7150ACPZ-5.0-R7
VIN
EN
BYP
VREG
VOUT
REF
REF_SENSE
GND
EP
+5V
C22
10uF
C21
1uF 50V
C20
10uF
C17
1uF 50V
C19
10uF
C75
100uF 6.3V
AGND
R10
25k 0.1%
Q7
BSS84
Q6
2N7002
Q5
2N7002
R9
100r 1%
V_SR
C25
10uF 25V
D6
BAT165E6327
D5
BAT165E6327
+VSR
C26
10uF 25V
C23
4.7uF 50V
R11
86.6k
R12
10k 0.1%
VsetReset
PROJECT: Sensornode
DRAWING TITLE: Forsyning
DRAWN BY: Jørn Tore Ørsland
REVISION: A
SHEET: 1 OF 5
STATUS:
DATE:
DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB


IC15
HMC2003
SR-
Xoff-
Ztrim
Xtrim
Yoff-
Zout
Xout
Vref
GND
V+
Zoff+
Zoff-
Xoff+
SR+
Yoff+
Ytrim
Yout
Vbias
Vbridge
Vsense
Z_out
X_out
Y_out
V_BATT
C35 100nF 16V
C36 100nF
IC16b
TSV321RILT
SENSOR_VREF
C74 100nF
AGND
+VSR
R29 25k 0.1%
R30 25k 0.1%
C37 100nF
Q1a
IRF7106TR
Q1b
IRF7106TR
D8
DN7002
R28 10k 1%
HMC2003_Set
HMC2003_Reset
C38 220nF 50V
IC9a
MAX7480ESA+
IC10a
MAX7480ESA+
IC11a
MAX7480ESA+
R25 9.09k 0.1%
R26 9.09k 0.1%
R27 9.09k 0.1%
X_out
Y_out
Z_out
C34 33nF
C30 6.2nF 50V
C31 100nF
C32 33nF
C29 6.2nF 50V
C28 100nF
C33 33nF
C27 6.2nF 50V
C24 100nF
PL2
PL3
PL4
R22 10k 0.1%
R23 10k 0.1%
R19 10k 0.1%
R20 10k 0.1%
R16 10k 0.1%
R21 10k 0.1%
R13 4.99k
R14 4.99k
R15 4.99k
IC6a
IC7a
IC8a
VOUT
REF
+IN
-IN
RG
DAC_CH0
DAC_CH1
DAC_CH2
R24 60.4r 1%
R31 60.4r 1%
R32 60.4r 1%
SENSOR_X
SENSOR_Y
SENSOR_Z
+5V
C39 100nF
C40 100nF
C41 100nF
C44 100nF
C45 100nF
C46 100nF
C43 100nF
IC9b
MAX7480ESA+
IC10b
MAX7480ESA+
IC11b
MAX7480ESA+
IC8b
AD8226ARMZ
IC7b
AD8226ARMZ
IC6b
AD8226ARMZ
IC16a
TSV321RILT
VDD
SHDN
GND
+VS
-VS
VCC
PROJECT: Sensornode
DRAWING TITLE: Analog signalbehandling
DRAWN BY: Jørn Tore Ørsland
REVISION: A
SHEET: 2 OF 5
STATUS:
DATE:
DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB


A
B
C
D
E
F
1
2
3
4
IC4e
MAX11044ECB+
SENSOR_Z
SENSOR_Y
SENSOR_X
SENSOR_VREF
37 CH2/CH1+/CH0+
39 CH3/CH2+/CH1++
42 CH4/CH3+/CH2++
44 CH5/CH4+/CH3++
IC4d
MAX11044ECB+
26 RDC_SENSE
33 RDC
27 RDC
55 RDC_SENSE
40 RDC
48 RDC
54 RDC
C10 22uF 16V
C9 22uF 16V
C11 22uF 16V
C8 22uF 16V
AGND
IC4c
MAX11044ECB+
DB15 64 ADC_DB15
DB14 1 ADC_DB14
DB13 2 ADC_DB13
DB12 3 ADC_DB12
DB11 4 ADC_DB11
DB10 5 ADC_DB10
DB9 6 ADC_DB9
DB8 7 ADC_DB8
DB7 10 ADC_DB7
DB6 11 ADC_DB6
DB5 12 ADC_DB5
DB4 13 ADC_DB4
CR3/DB3 14 ADC_DB3
CR2/DB2 15 ADC_DB2
DB1/CR1 16 ADC_DB1
DB0/CR0 17 ADC_DB0
+5V
C56 100nF
C57 100nF
C58 100nF
C59 100nF
C60 100nF
C61 100nF
C65 10uF
3.3V
C62 100nF
C63 100nF
C64 100nF
C66 10uF
IC4b
MAX11044ECB+
DVDD
DGND
9 21 60
8 22 59
IC4f
MAX11044ECB+
EOC 18 ADC_EOC
CONVST 19 ADC_CONVST
SHDN 20 ADC_SHDN
WR 61 ADC_WR
CS 62 ADC_CS
RD 63 ADC_RD
C55 100nF
IC4a
REFIO 41
AVDD 24 29 35 46 52 57
AGNDS 23 28 32 38 43 25 49 53 58
AGND 51 30 36 45 56
EP_GND 65
CH0/I.C.+++
CH1/CH0+/I.C.++
CH6/CH5+/I.C.++
I.C.+++/CH7
PROJECT: Sensornode
DRAWING TITLE: Analog til digitalomformer
DRAWN BY: Jørn Tore Ørsland
REVISION: A
SHEET: 3 OF 5
STATUS:
DATE:
DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB


A
B
C
D
E
F
1
2
3
4
IC5a
STM32F072CBT6
3.3V
VBAT
PC13
PC14-OSC32JN
PC15-OSC32_OUT
PF0-OSC_IN
PF1-0SC_0UT
NRST
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB11
PB12
PB13
PB14
PB15
BOOT0
HMC2003_Reset
HMC2003_Set
V_SR
RESET
Vbatt
VsetReset
ADC_SHDN
ADC_CONVST
ADC_EOC
SPI1_SCK
SPI1_MISO
SPI1_MOSI
ADC_RD
ADC_CS
ADC_WR
CAN_RX
CAN_TX
SW_DIO
SW_CLK
CLK_SYNC
ADC_DB0
ADC_DB1
ADC_DB2
ADC_DB3
ADC_DB4
ADC_DB5
ADC_DB6
ADC_DB7
ADC_DB8
ADC_DB9
ADC_DB10
ADC_DB11
ADC_DB12
ADC_DB13
ADC_DB14
ADC_DB15
C6
22pF 50V
C7
22pF 50V
X1
XTAL 8MHz
R1
10k
R2
10k
PL1
SW_CLK
SW_DIO
IC17a
MAX5714AUD+
DAC_CH0
DAC_CH1
DAC_CH2
DAC_CH3
OUTA
OUTB
OUTC
OUTD
REF
RDY
DIN
SCLK
CSB
CLR
LDAC
SPI1_MISO
SPI1_MOSI
SPI1_SCK
C71
100nF
C72
100nF
C73
100nF
C67
100nF
AGND
R33
10k
RESET
C76
10uF
IC5b
STM32F072CBT6
VDD
VSS
C47
100nF
C48
100nF
C49
100nF
C50
100nF
C54
100nF
C69
100nF
C68
100nF
IC17b
MAX5714AUD+
VDD
VDDIO
GND
C70
100nF
+5V
PROJECT: Sensornode
DRAWING TITLE: Mikrokontroller
DRAWN BY: Jørn Tore Ørsland
REVISION: A
SHEET: 4 OF 5
STATUS:
DATE:
DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB

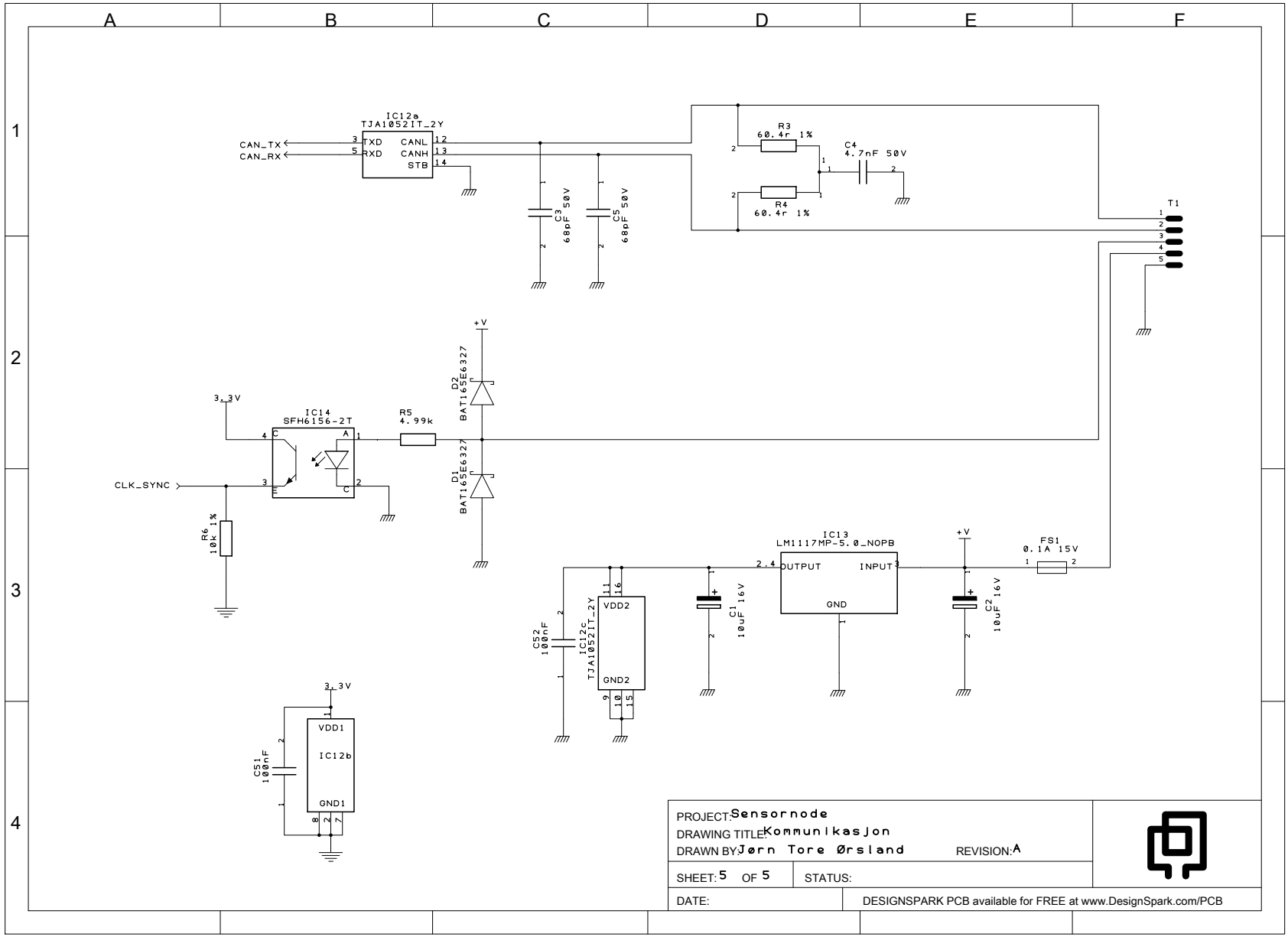

A
B
C
D
E
F
1
2
3
4
IC12a
TJA1052IT_2Y
CAN_TX
CAN_RX
TXD
RXD
CANL
CANH
STB
C3
68pF 50V
C5
68pF 50V
R3
60.4r 1%
R4
60.4r 1%
C4
4.7nF 50V
T1
+V
D2
BAT165E6327
D1
BAT165E6327
3.3V
IC14
SFH6156-2T
R5
4.99k
CLK_SYNC
R6
10k 1%
IC13
LM1117MP-5.0_NOPB
OUTPUT
INPUT
GND
FS1
0.1A 15V
C1
10uF 16V
C2
10uF 16V
CE2
100nF
IC12c
TJA1052IT_2Y
VDD2
GND2
CE1
100nF
IC12b
VDD1
GND1
PROJECT: Sensornode
DRAWING TITLE: Kommunikasjon
DRAWN BY: Jørn Tore Ørsland
REVISION: A
SHEET: 5 OF 5
STATUS:
DATE:
DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB

Table of Contents

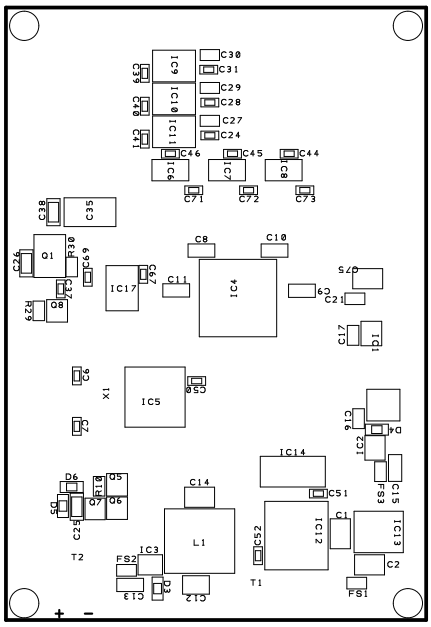
Product Info

Product number: 208779
Product name: magnetfeltSensor -
Board size: 73.03 x 106.43 mm
Board area: 0.78 dm²
Conductive layers: 4
Plated thru holes: 267 tools: 8
Unplated thru holes: 1 tools: 1
Minimum through hole: 0.4 mm
Minimum track width: 0.13 mm
Minimum clearance: 0.13 mm
Minimum annular ring: 0.3 mm
Minimum hole-to-Cu clearance: 0.74 mm
Surface finish: Lead-free HASL
Legend color, top: White bottom: White
Scoring: NO
Peel-off mask: NO
Stackup: 4001: Standard 1.6mm 1oz
Stackup material: FR-4 -- IPC-4101 /21
Number of routs: 1 total length: 0.359 m
Estimated board weight: 34 g
Total thickness: 1.6 mm Copper thickness: 1 oz (38 µm) outer, 1 oz (35 µm) inner
Solder mask color, top: Black bottom: Black

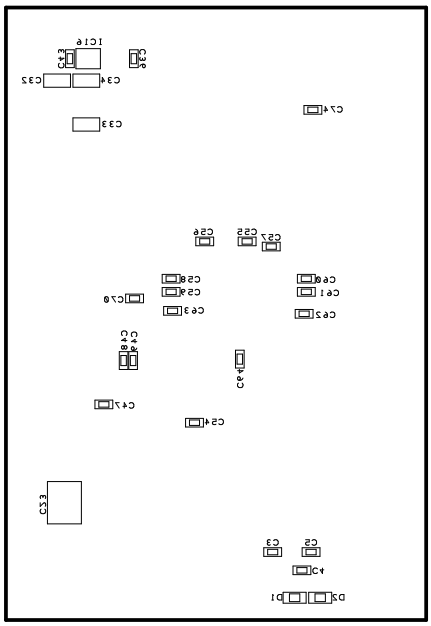
Additional Documents in Product

StackupDrawing.pdf

Responsible dept.	Technical reference	Document type	Document status			
		Layout documentation				
Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Table of Contents	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: A4	Sheet: 1/13

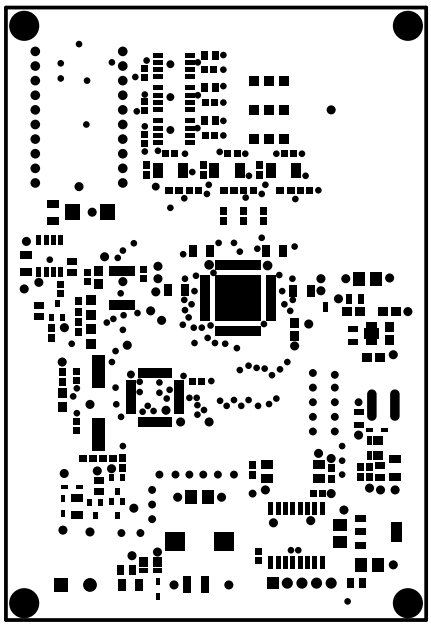


Responsible dept.	Technical reference	Document type	Document status	
		Layout documentation		
Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Notation top)	**208779**	
	Approved by:		Rev: A	Date of issue: 08.04.2016
			Scale: 1:1 A4	Sheet: 2/13



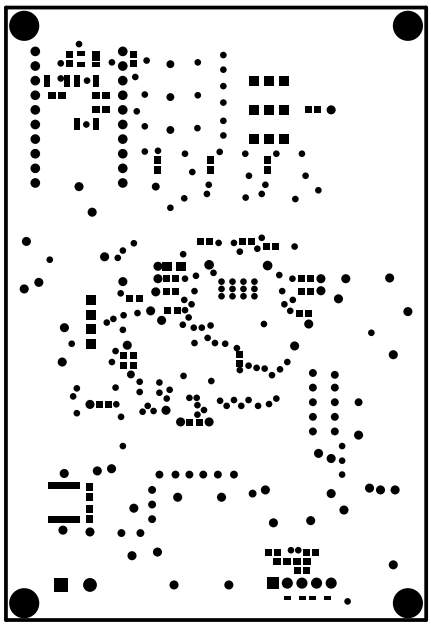
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Notation bottom)	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: 1:1 A4	Sheet: 3/13



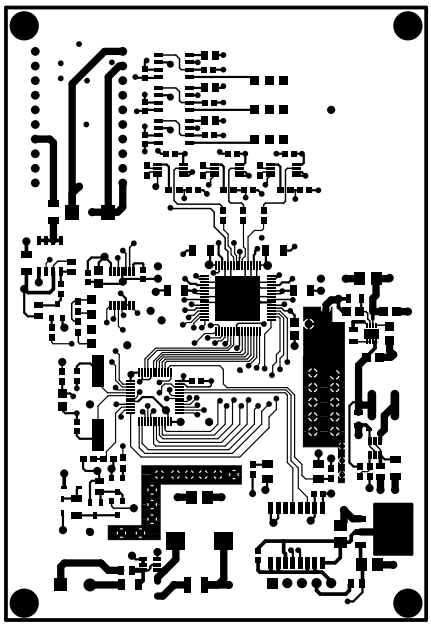
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Solder mask top)	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: 1:1 A4	Sheet: 4/13



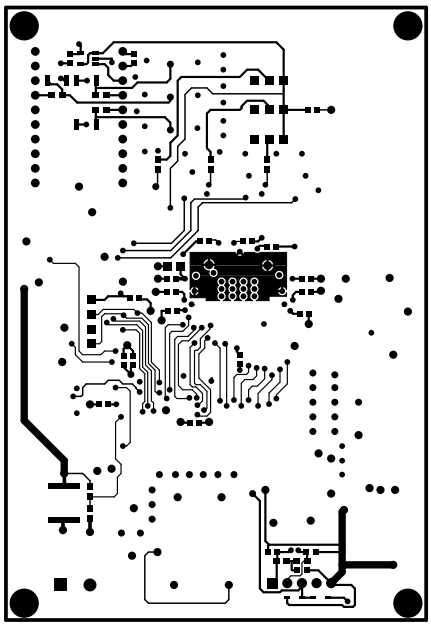
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

	Created by	Title	208779			
Macaos	Jørn Tore Ørsland	208779 magnetfeltSensor - Layout drawing (Solder mask bottom)	Rev	Date of issue	Scale	Sheet
	Approved by		A	08.04.2016	1:1 A4	5/13



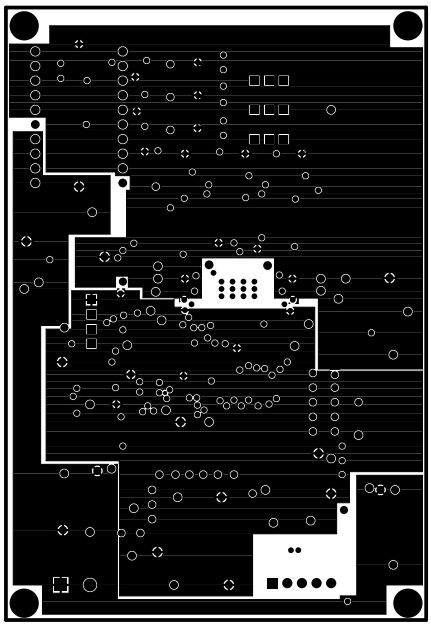
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

	Created by	Title	208779			
Macaos	Jørn Tore Ørsland	208779 magnetfeltSensor - Layout drawing (Copper top)	Rev	Date of issue	Scale	Sheet
	Approved by		A	08.04.2016	1:1 A4	6/13



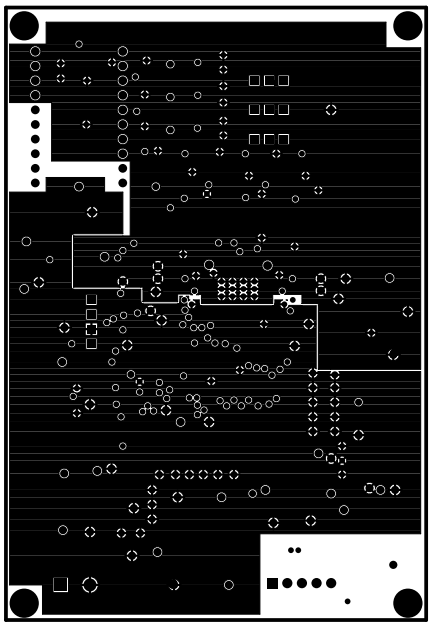
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Copper bottom)	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: 1:1 A4	Sheet: 7/13



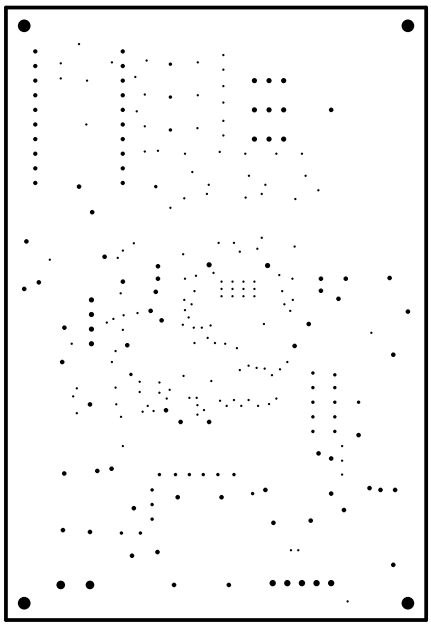
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Inner 1)	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: 1:1 A4	Sheet: 8/13



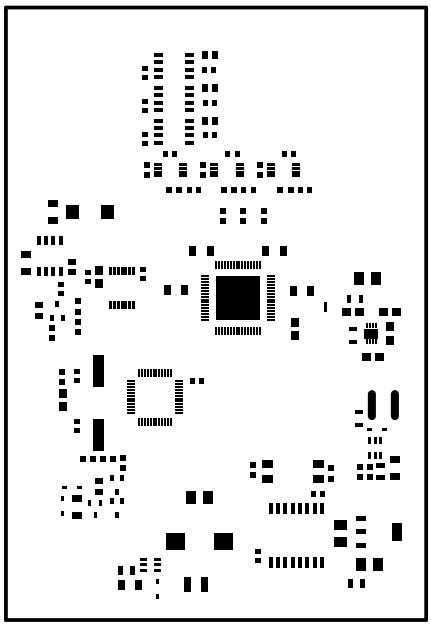
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Inner 2)	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: 1:1 A4	Sheet: 9/13



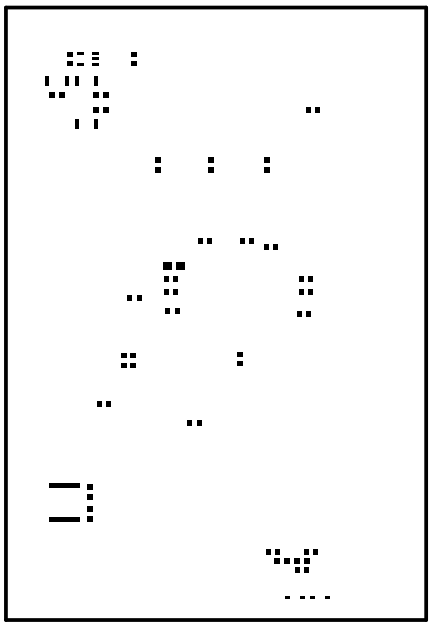
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Drill through)	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: 1:1 A4	Sheet: 10/13



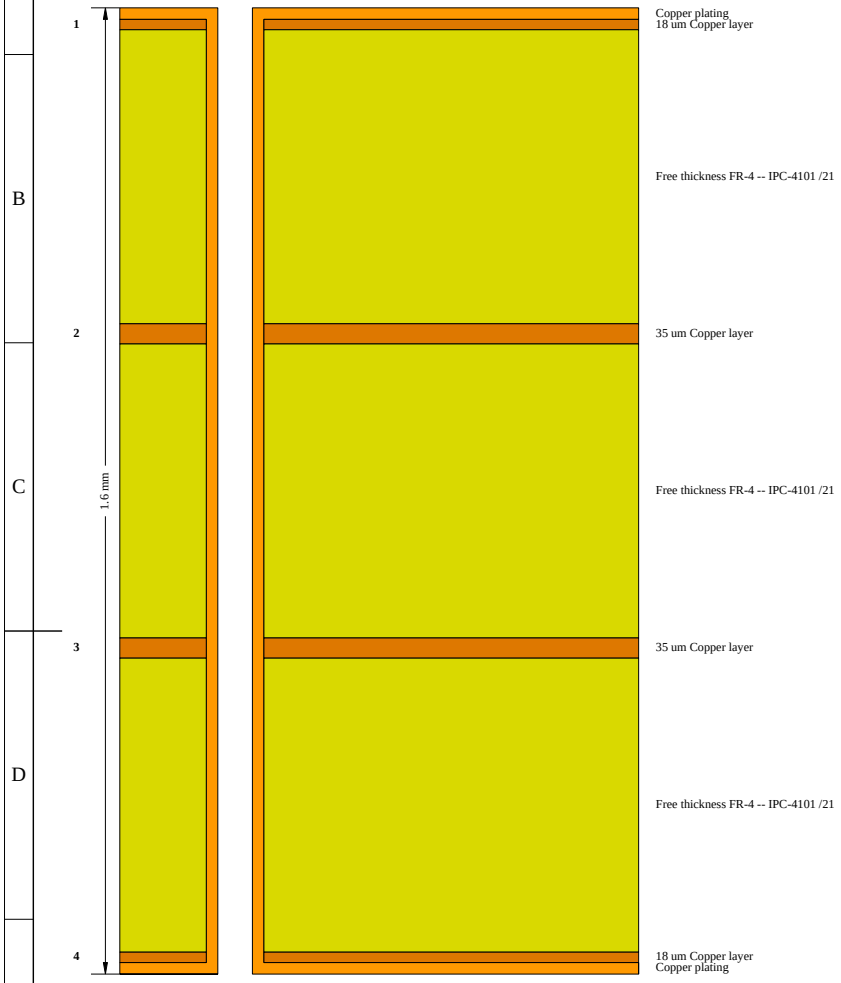
Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Paste mask top)	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: 1:1 A4	Sheet: 11/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

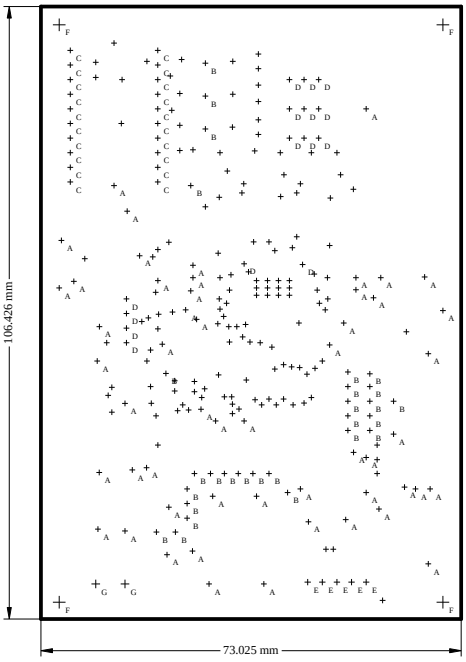
Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Layout drawing (Paste mask bottom)	**208779**			
	Approved by:		Rev: A	Date of issue: 08.04.2016	Scale: 1:1 A4	Sheet: 12/13



Total thickness: 1.6 mm ± 10%
Copper plating in accordance with IPC-601x Hole Cu plating tables, as applicable

Responsible dept.	Technical reference	Document type	Document status	
		Layout documentation		
Macaos	Created by: Jørn Tore Ørsland	Title: 208779 magnetfeltSensor - Stackup drawing	**208779**	
	Approved by:		Rev: A	Date of issue: 08.04.2016
			Scale: 10:1 A4	Sheet: 13/13

Symbol	Diameter	Qty	Plated
	0.4	140	Yes
B	0.6	28	Yes
C	0.75	20	Yes
A	0.8	53	Yes
D	0.9	15	Yes
E	1.1	5	Yes
G	1.5	2	Yes
F	2.2	4	Yes



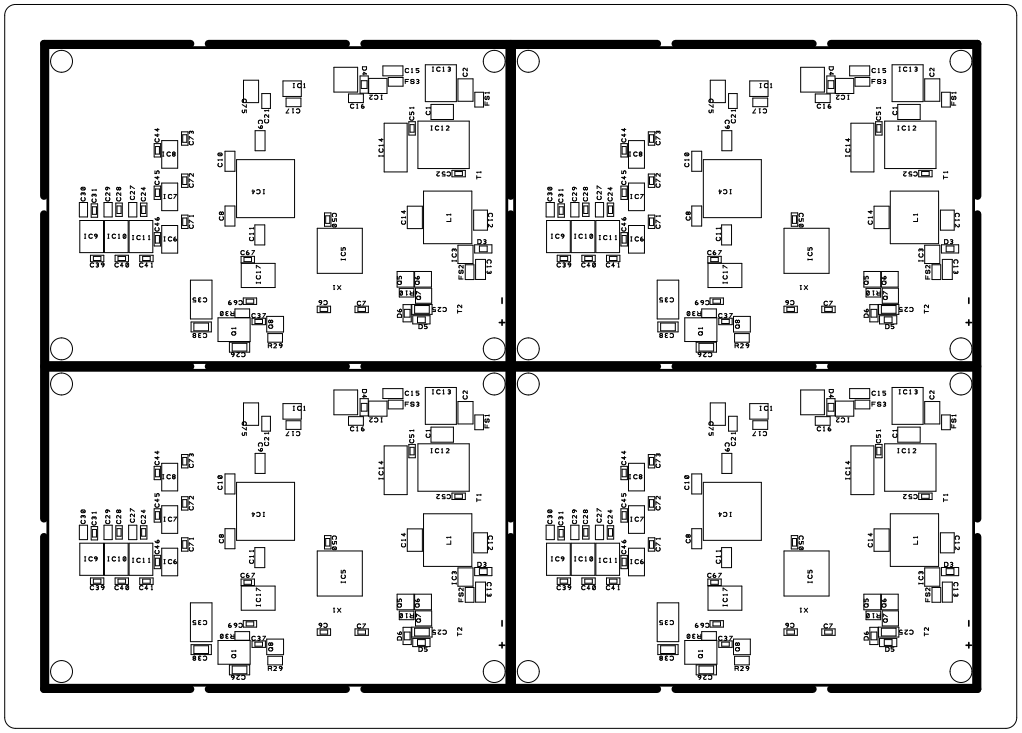
Responsible dept.	Technical reference	Document type: Layout documentation	Document status
Macaos	Created by: Jørn Tore Ørsland Approved by:	Title: 208779 magnetfeltSensor - Drill template (Drill through)	**208779** Rev: A Date of issue: 08.04.2016 Scale: 1:1 A4 Sheet: 14/13

Product Info

Product number: 208780
Product name: magnetfeltSensor -
Board size: 73.03 x 106.43 mm
Board area: 0.78 dm²
Conductive layers: 4
Plated thru holes: 267 tools: 8
Unplated thru holes: 1 tools: 1
Minimum through hole: 0.4 mm
Minimum track width: 0.13 mm
Minimum clearance: 0.13 mm
Minimum annular ring: 0.3 mm
Minimum hole-to-Cu clearance: 0.74 mm
Surface finish: Lead-free HASL
Legend color, top: White bottom: White
Scoring: NO
Peel-off mask: NO
Stackup: 4001: Standard 1.6mm 1oz
Stackup material: FR-4 -- IPC-4101 /21
Number of routs: 1 total length: 0.359 m
Estimated board weight: 34 g
Total thickness: 1.6 mm Copper thickness: 1 oz (38 µm) outer, 1 oz (35 µm) inner
Solder mask color, top: Black bottom: Black

Additional Documents in Product

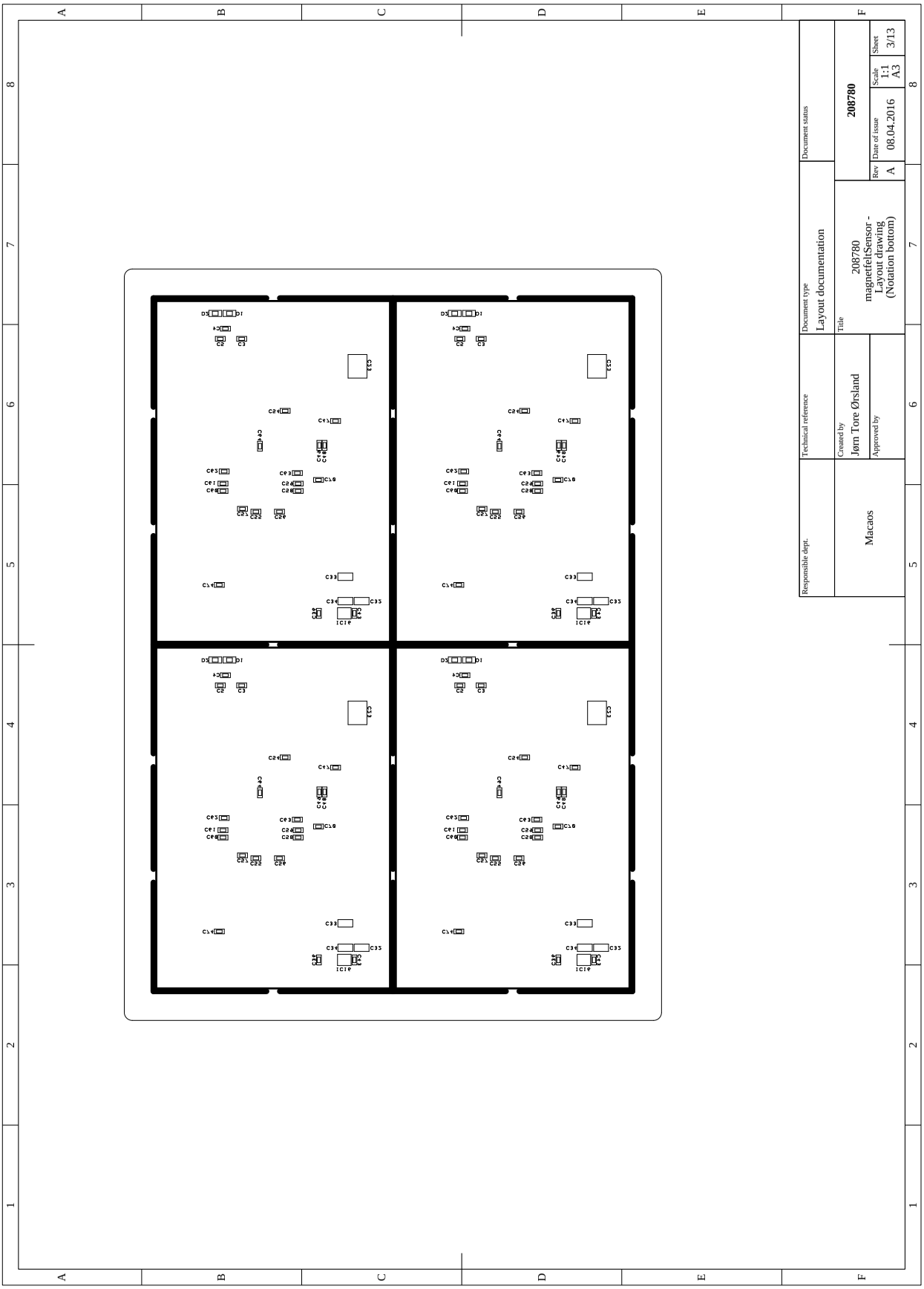
PanelReport.txt
PanelDrawing.pdf
StackupDrawing.pdf

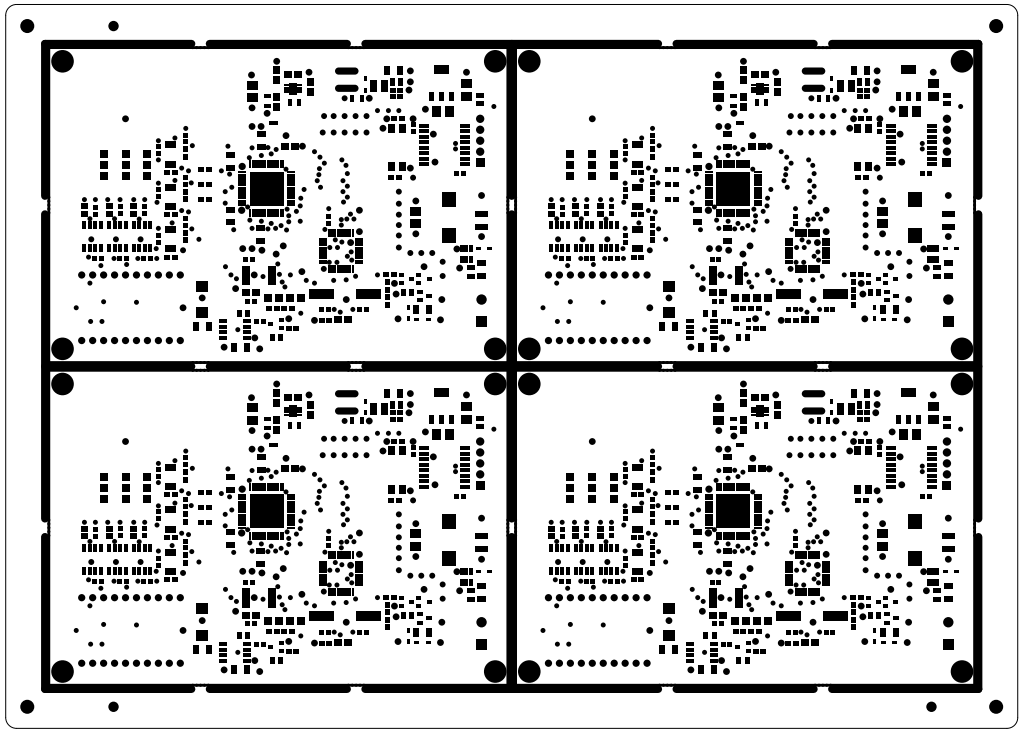


Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

	Created by	Title	208780
Macaos	Jørn Tore Ørsland	208780 magnetfeltSensor - Layout drawing (Notation top)	
	Approved by		

Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	2/13

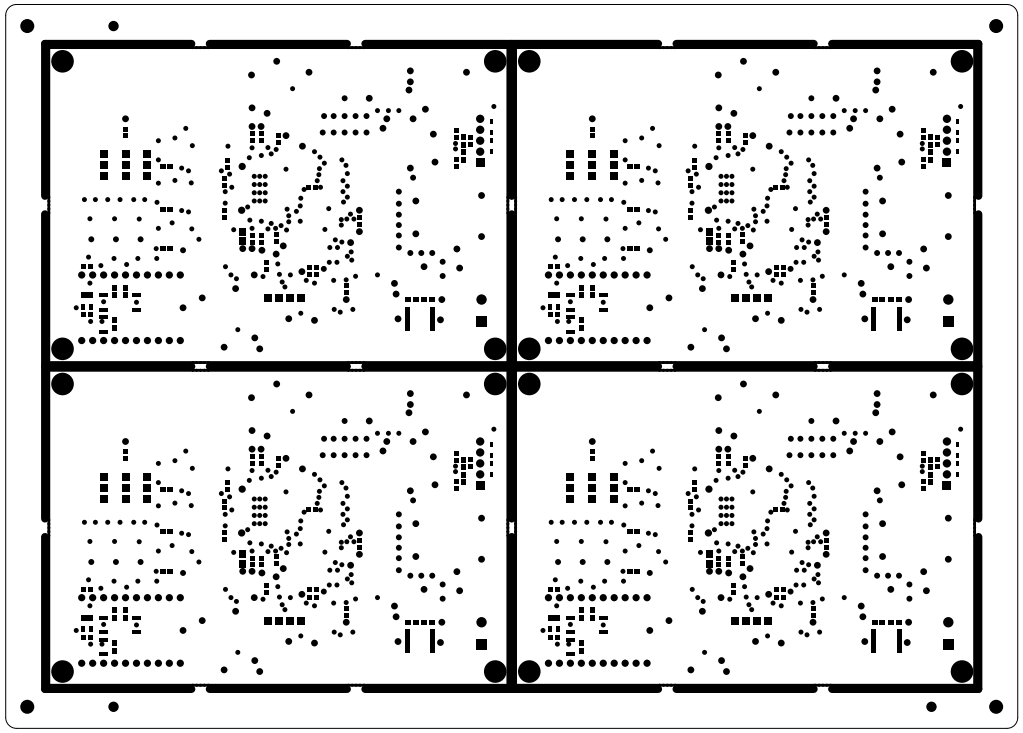

Responsible dept.
Macaos
Technical reference
Created by
Jørn Tore Ørsland
Approved by
Document type
Layout documentation
Title
208780
magnetfeltSensor -
Layout drawing
(Notation bottom)
Document status
208780
Rev
A
Date of issue
08.04.2016
Scale
1:1
A3
Sheet
3/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

	Created by	Title	208780
Macaos	Jørn Tore Ørsland	208780 magnetfeltSensor - Layout drawing (Solder mask top)	
	Approved by		

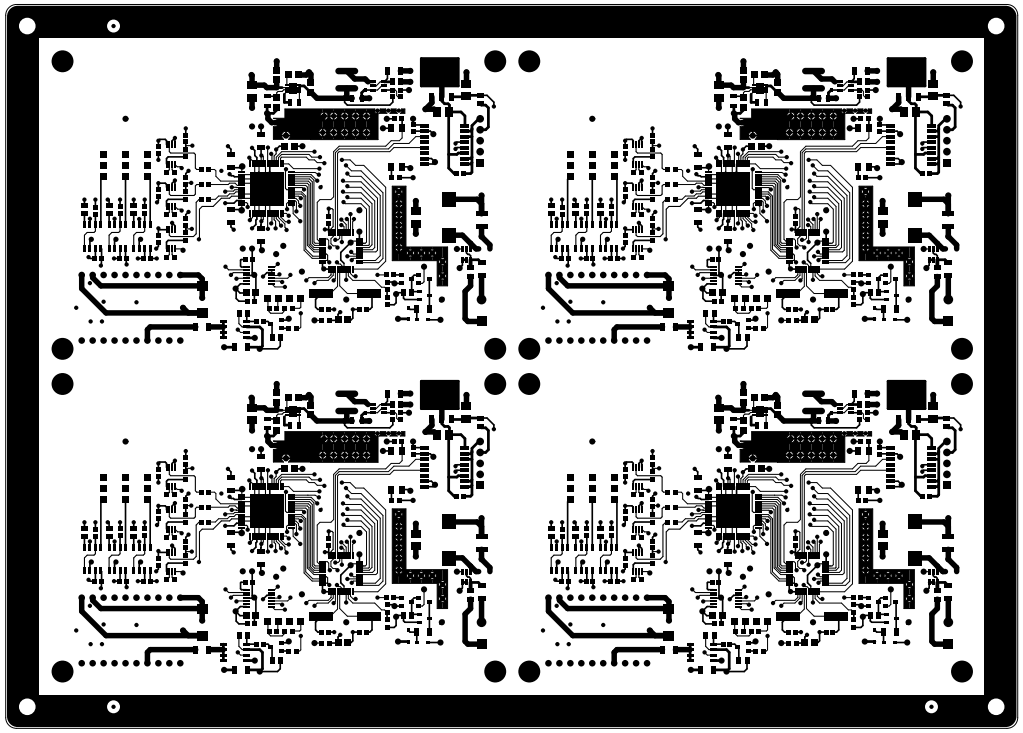
Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	4/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

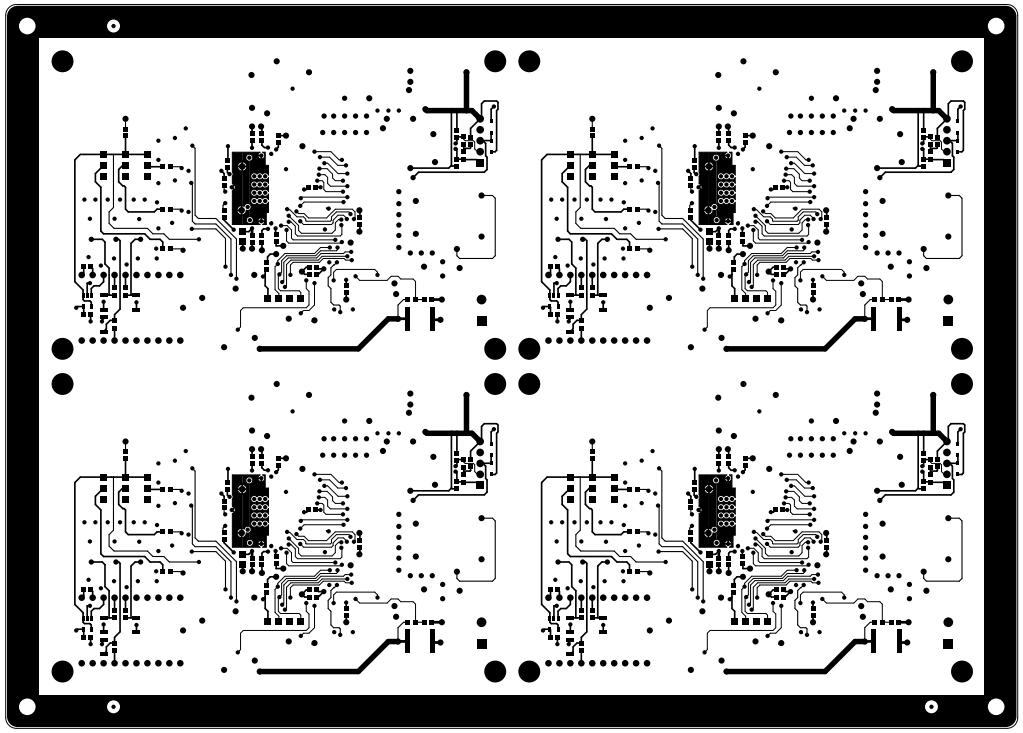
	Created by	Title	208780
Macaos	Jørn Tore Ørsland	208780 magnetfeltSensor - Layout drawing (Solder mask bottom)	
	Approved by		

Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	5/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

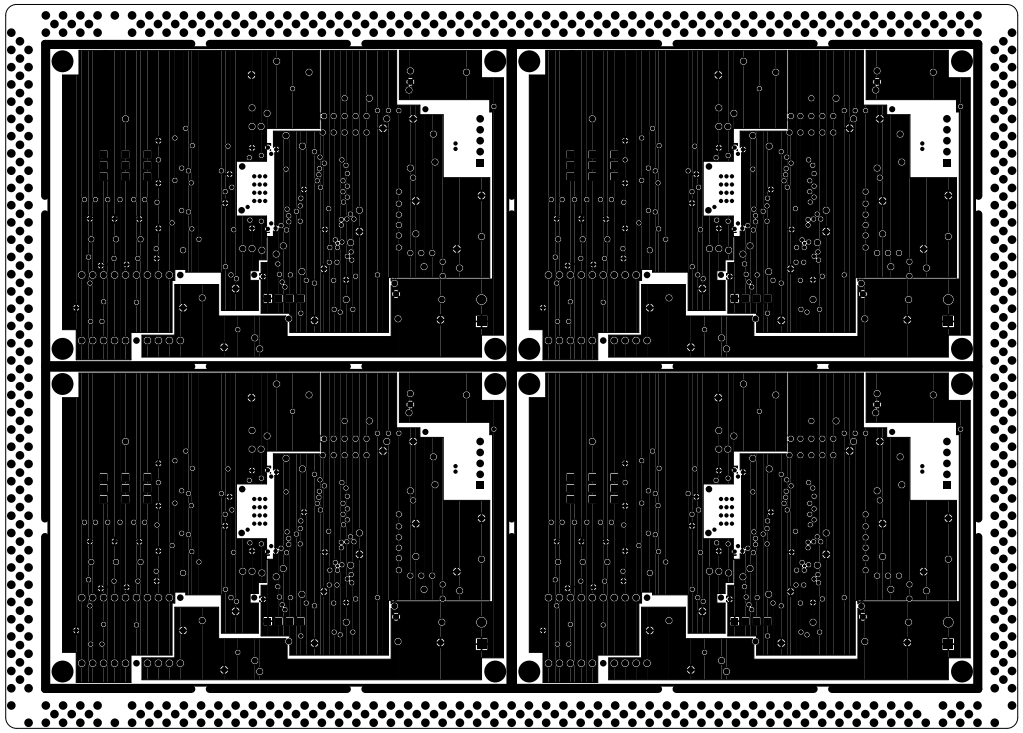
Macaos	Created by: Jørn Tore Ørsland	Title: 208780 magnetfeltSensor - Layout drawing (Copper top)	**208780**
	Approved by:		Rev: A; Date of issue: 08.04.2016; Scale: 1:1 A3; Sheet: 6/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

	Created by	Title	208780
Macaos	Jørn Tore Ørsland	208780 magnetfeltSensor - Layout drawing (Copper bottom)	
	Approved by		

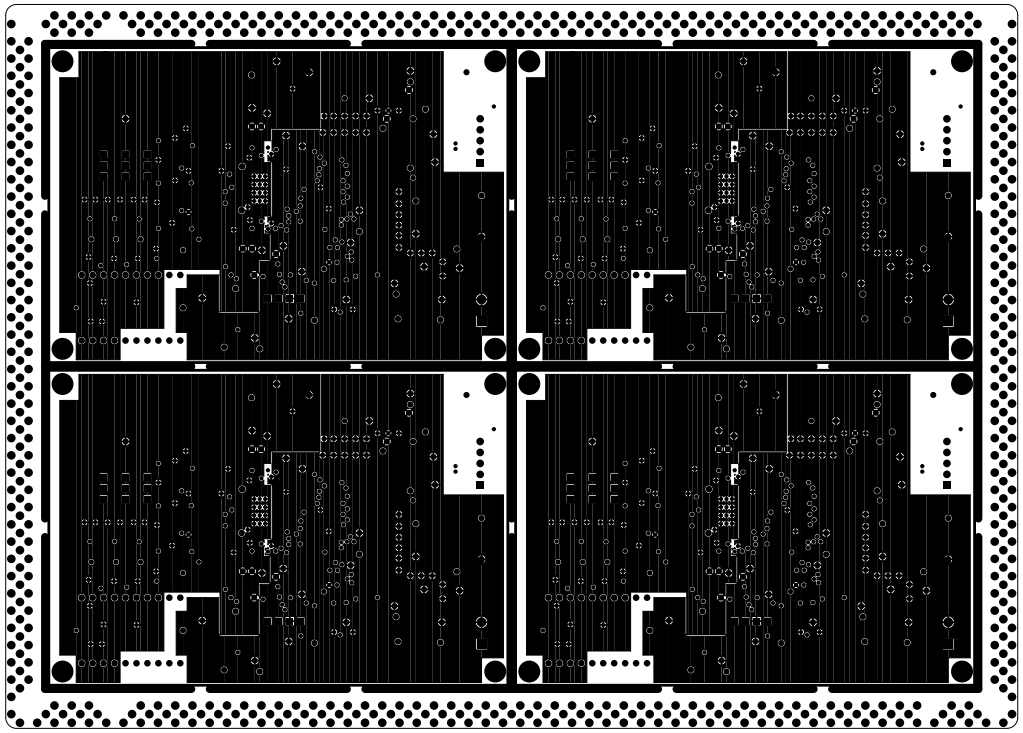
Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	7/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland Approved by:	Title: 208780 magnetfeltSensor - Layout drawing (Inner 1)	**208780**

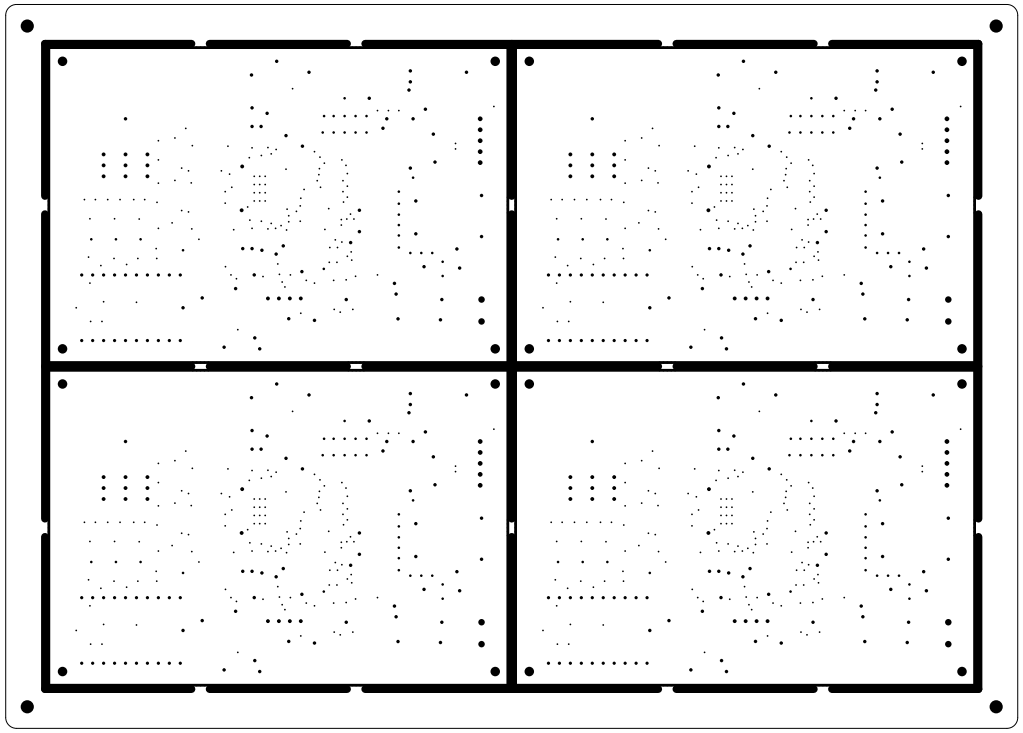
Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	8/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

Macaos	Created by: Jørn Tore Ørsland Approved by:	Title: 208780 magnetfeltSensor - Layout drawing (Inner 2)	**208780**

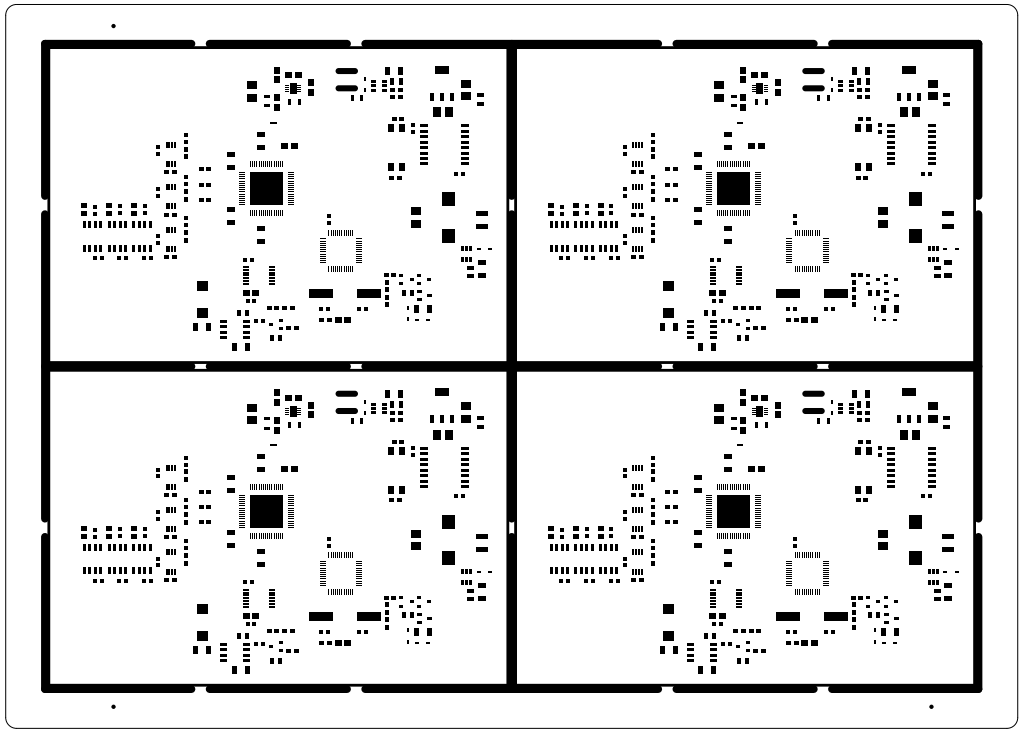
Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	9/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

	Created by	Title	208780
Macaos	Jørn Tore Ørsland	208780 magnetfeltSensor - Layout drawing (Drill through)	
	Approved by		

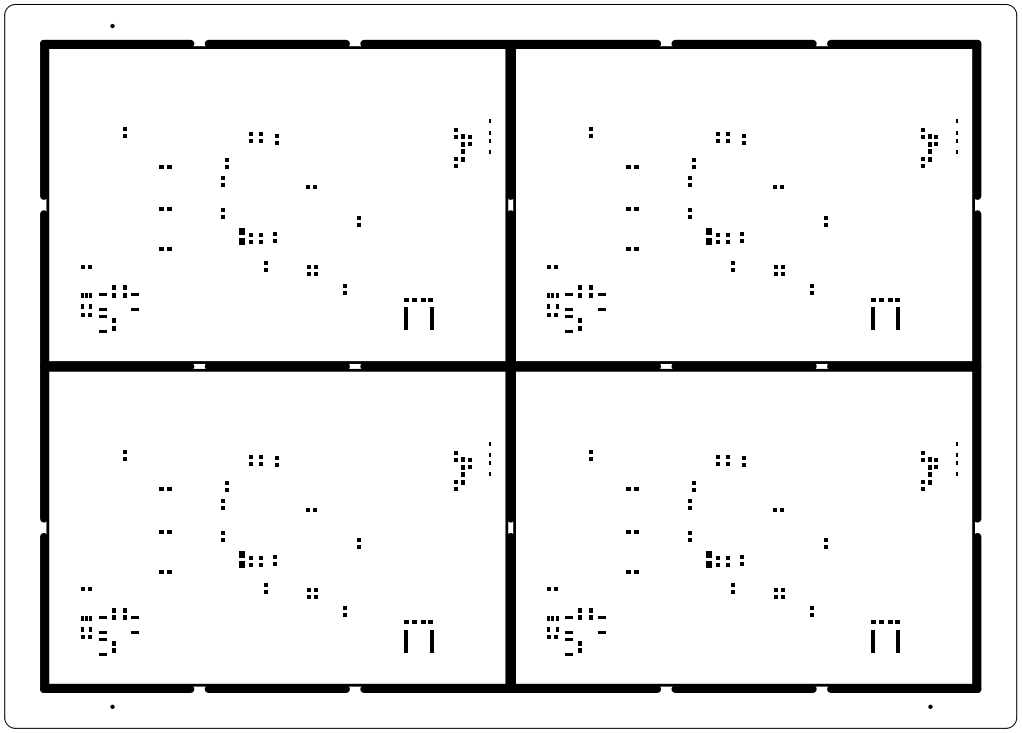
Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	10/13



Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

	Created by	Title	208780
Macaos	Jørn Tore Ørsland	208780 magnetfeltSensor - Layout drawing (Paste mask top)	
	Approved by		

Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	11/13

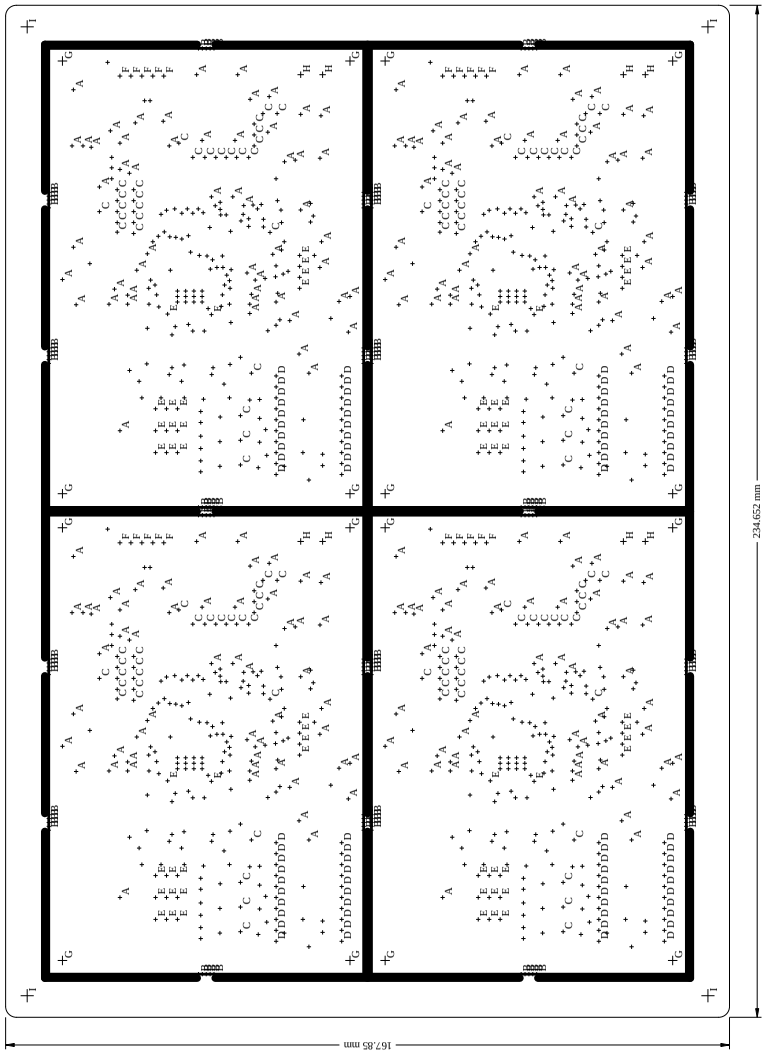


Responsible dept.	Technical reference	Document type	Document status
		Layout documentation	

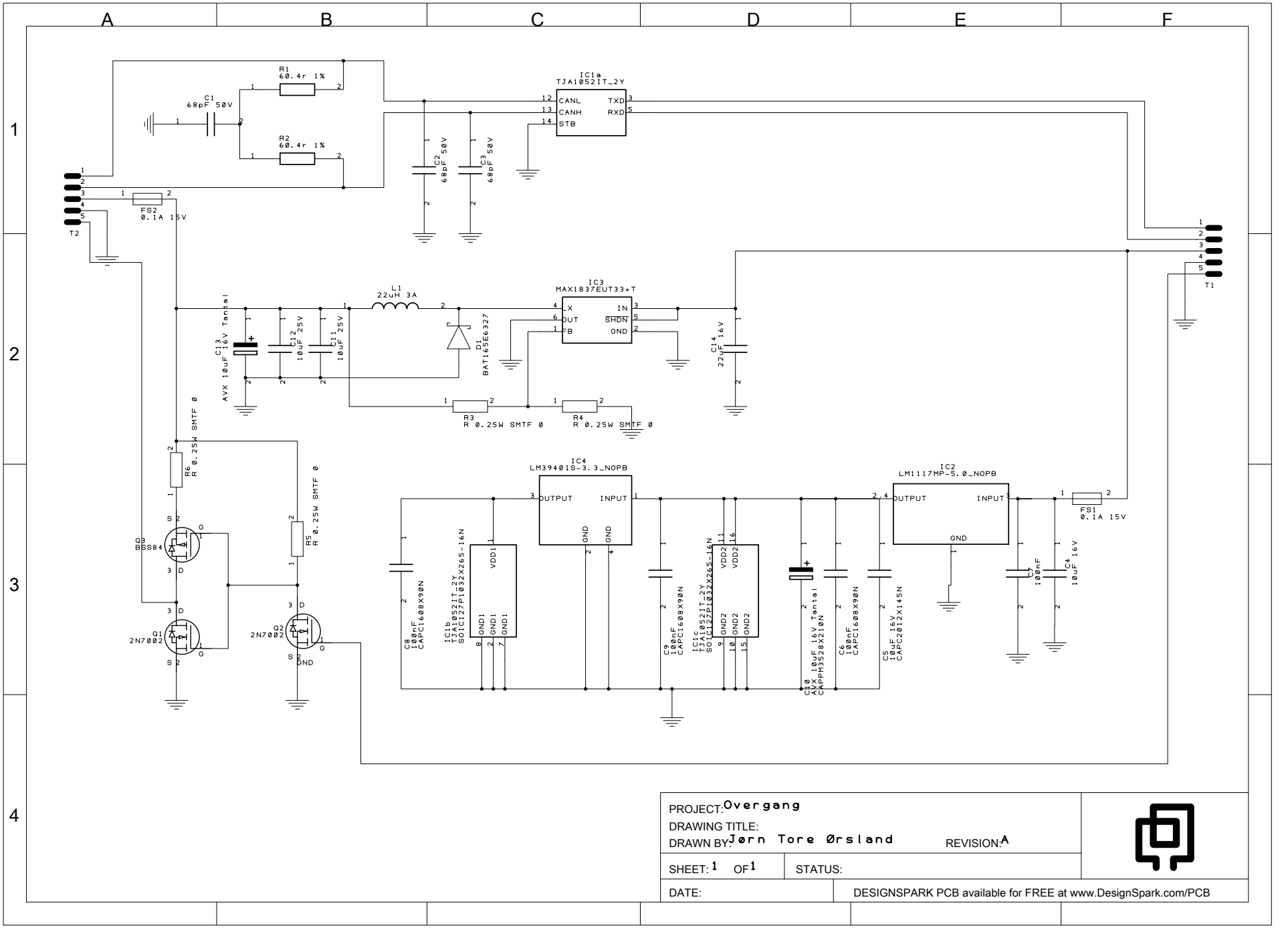
	Created by	Title	208780
Macaos	Jørn Tore Ørsland	208780 magnetfeltSensor - Layout drawing (Paste mask bottom)	
	Approved by		

Rev	Date of issue	Scale	Sheet
A	08.04.2016	1:1 A3	12/13

Total thickness: 1.6 mm ± 10%
Copper plating in accordance with IPC-601x Hole Cu plating tables, as applicable


234.652 mm
167.85 mm

D.2 Overgang


A
B
C
D
E
F
1
2
3
4
R1
60.4r 1%
R2
60.4r 1%
C1
68pF 50V
C2
68pF 50V
C3
68pF 50V
IC1a
TJA1052IT_2Y
CANL
CANH
STB
TXD
RXD
FS2
0.1A 15V
T2
T1
L1
22uH 3A
IC3
MAX1837EUT33+T
C13
AVX 10uF 16V Tantal
C12
10uF 25V
C11
10uF 25V
D1
BAT165E6327
C14
22uF 16V
R3
R 0.25W SMTF 0
R4
R 0.25W SMTF 0
R6
R 0.25W SMTF 0
R5
R 0.25W SMTF 0
Q3
BSS84
Q1
2N7002
Q2
2N7002
IC4
LM3940IS-3.3_NOPB
IC2
LM1117MP-5.0_NOPB
FS1
0.1A 15V
C8
100nF
CAPC1608X90N
IC1b
TJA1052IT_2Y
SOIC127P1032X265-16N
C9
100nF
CAPC1608X90N
IC1c
TJA1052IT_2Y
SOIC127P1032X265-16N
C10
AVX 10uF 16V Tantal
CAPPM3528X210N
C6
100nF
CAPC1608X90N
C5
10uF 16V
CAPC2012X145N
C7
100nF
C4
10uF 16V
PROJECT: Overgang
DRAWING TITLE:
DRAWN BY: Jørn Tore Ørsland
REVISION: A
SHEET: 1 OF 1
STATUS:
DATE:
DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB

Referanser

[1] Ee-sim design generation and simulation tool. https://www.maximintegrated.com/en/ee_sim/index.mvp?ReturnUrl=http%2525253a%2525252f%2525252fmaxim.transim.com%2525252fLoader%2525252fDownloads.aspx.

[2] Kvaser canlib sdk. https://www.kvaser.com/developer/canlib-sdk/.

[3] *The design of small-scale embedded systems.* Palgrave, 2001.

[4] *Linear Systems And Signals.* Oxford University Press, 2010.

[5] *Digital Signal Processing.* Pearson, 2014.

[6] Analog Devices. *Instrumentation Amplifier AD8226*, 2012. Rev C.

[7] Analog Devices. *ADM7150 800 mA Ultralow Noise,High PSRR,RF Linear Regulator*, 2013. Rev 0.

[8] IEEE Standards Board. Ieee standard procedures for measurement of power frequency electric and magnetic fields form ac power lines. Technical Report Std 644-1994, IEEE, 2008.

[9] BOSCH. *CAN Spesification 2.0*, 9 1991.

[10] Steve Corrigan. *Application Report SLLA270. Controller Area Network Physical Layer Requirements.* Texas Instruments, 2008.

[11] Honeywell. *1-and 2-Axis Magnetic Sensors.* Rev B.

[12] Honeywell. *Application Note Magnetoresistive Sensors*, 2003.

[13] Honeywell. *3-Axis Magnetic Sensor Hybrid HMC2003*, 2011. Rev G.

[14] Jørn Tore Ørsland Jarle Urdal. Ion racing 2014, elektrisk styresystem, 2014.

[15] Gunnar G. Løvås. *Statistikk for Universiteter og Høgskoler.* Universitetsforlaget, 2011.

[16] Steven A. Macintyre. Magnetic field measurement. *CRC Press LCC*, 1999.

[17] Maxim Integrated. *MAX5713/MAX5714/MAX5715 Ultra-Small, Quad-Channel, 8-/10-/12-Bit Buffered Output DACs with Internal Reference and SPI Interface.* Rev 3.

[18] Maxim Integrated. *8th-Order, Lowpass, Butterworth,Switched-Capacitor Filter.MAX7480*, 1999. Rev 0.

[19] Maxim Integrated. *TUTORIAL 1080, Understanding SAR ADCs: Their Architecture and Comparison with Other ADCs*, 10 2001.

[20] Maxim Integrated. *MAX1836/MAX1837 24V Internal Switch, 100Converters*, 2006. Rev 3.

[21] Maxim Integrated. *4-/6-/8-Channel, 16-/14-Bit,Simultaneous-Sampling ADCs. MAX11047-MAX11049/MAX11057-MAX11059*, 2011. Rev 2.

[22] Maxim Integrated. *MAX11044/MAX11044B/MAX11045/MAX11045B/MAX11046/MAX11046B/MAX1 4-/6-/8-Channel, 16-/14-Bit,Simultaneous-Sampling ADCs*, 2013. Rev 6.

[23] NXP. *AH1301 TJA1052i Galvanically isolated high-speed CAN transceiver*, 2014. Rev 1.2.

[24] NXP. *TJA1052i Galvanically isolated high-speed CAN transceiver*, 2016. Rev 3.

[25] Kristian S. Stangeland. Positioning in electromagnetic fields. Master's thesis, University of Stavanger, 2015.

[26] STMicroelectronics. *PM0215 STM32F0xxx Cortex-M0 programming manual*, 4 2012. Rev 1.

[27] STMicroelectronics. *RM0091 Reference manual,STM32F0x1/STM32F0x2/STM32F0x8*, 7 2015. Rev 8.

[28] STMicroelectronics. *STM32F072x8 STM32F072xB*, 12 2015. Rev 4.

[29] STMicroelectronics. *STM32F303xB STM32F303xC*, 5 2016. Rev 13.

[30] Texas Instruments. *LM1117/LM1117I 800mA Low-Dropout Linear Regulator*, 7 2012.

[31] Vishay Semiconductors. *SFH6156 Optocoupler, Phototransistor Output, High Reliability, 5300 VRMS*, 10 2015.

[32] Dr. Conal Watterson. *Application Note 1123. Controller Area Network (CAN) Implementation Guide.* Analog Devices, 2012. Rev 0.

[33] Peter Wilson. *The Circuit Designer?s Companion.* 3 edition, 2012.

Figurer

Tabeller