



Kunnskap for en bedre verden

FAKTULTET FOR
INFORMASJONSTEKNOLOGI OG
ELEKTROTEKNIKK

INSTITUTT FOR ELEKTRONISKE SYSTEMER

TTT4280 - SENSORER OG INSTRUMENTERING

Lab 1 Systemoppsett

Forfattere:
Kristian Håland og Christian Heimvik

Dato 27. mars 2024

1 SAMMENDRAG

Denne rapporten tar for seg oppsett av et system som digitaliserer fem analoge signaler, som videre behandles ved bruk av digital signalbehandling. Digitaliseringen skjer ved å bruke analog-til-digital converteren (ADC) MCP3201 og enbrikkede datamaskinen Raspberry Pi 3B med en effektiv samplingfrekvens på 31.19(01) kHz med $k = 2$. For et digitalt signal med amplitude 0.3 V gir dette et teoretisk signal til støyforhold (SNR) lik $SNR'_{[dB]} = 59.2$ dB, mot et observert signal til støyforhold på $SNR_{[dB]} \approx 55$ dB. Respektive analoge og digitale signaler er tilnærmet identiske i amplitude og frekvens med en gjennomsnittlig likhetsgrad $\hat{S}_{[dB]} = -2.9389$ dB. I frekvensanalysen av de digitaliserte signalene observeres det forskjellige egenskaper avhengig av om zero-padding eller ulike vindusfunksjoner ble brukt, men ved høye frekvenser ble det observert et støygulv på omtrent -90 dB i alle tilfeller.

INNHOLD

1 Sammendrag	i
Figurer	iii
2 Innledning	1
3 Teori	2
3.1 Digitalisering i teorien	2
3.2 Digitalisering i praksis	3
3.3 Signalbehandling	4
3.3.1 Samplingsfrekvens	4
3.3.2 DFT og FFT-algoritme	5
3.3.3 Zero-padding	5
3.3.4 Vindusfunksjoner	5
3.4 Støy og filtrering	6
4 Eksperimentelt oppsett	9
4.1 Prosesseringsenhet	9
4.2 ADC	11
4.3 Filter	11
4.4 Systemsammenstilling	12
5 Resultater	15
5.1 Sampling av analoge signaler	15
5.2 Sammenligning med analogt signal	16
5.3 Frekvensspekter	17
5.4 Zero-padding	18
5.5 Windowing	19
5.6 Kombinering av zero-padding og windowing	20
5.7 Kvantisering	21
5.8 Filter og støy	22
6 Diskusjon	24
6.1 Datainnhenting	24
6.1.1 Samplingsfrekvens	24

6.2	Signalbehandling	24
6.3	Filter	25
6.4	Feilkilder	25
7	Konklusjon	27
	Kilder	28
	Appendix	29
A	Github	29
B	frekvensspektrum-generator.py	29

FIGURER

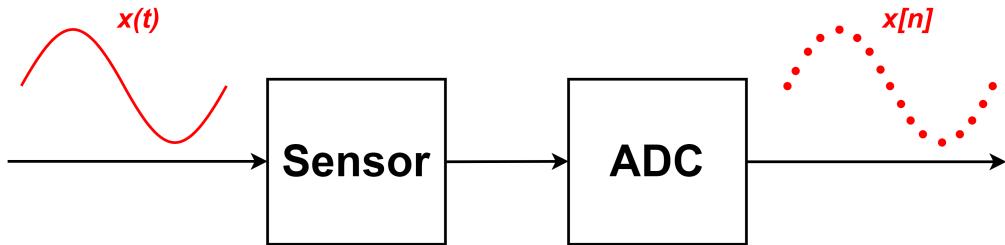
2.1	Diskretisering av signal	1
3.1	kvantisering av analogt signal	2
3.2	Timingdiagram for synkron seriell kommunikasjon	4
3.3	Windowing	6
3.4	Windowing periodogram	6
3.5	Skilling av analog og digital krets	7
3.6	Pi-filter	7
4.1	Blokkskjema av system for sampling	9
4.2	Raspberry Pi	10
4.3	Timingdiagram for MCP3201	11
4.4	Implementert Pi-filter	12
4.5	Blokkskjema systemoppsett	12
4.6	Kretsskjema av systemet	13
4.7	Realisert system	14
4.8	Testkrets for verifikasjon og testing av systemet	14
5.1	Analoge inngangssignaler	15
5.2	Samplede signaler	15
5.3	Sammenligning av $x_1[n]$ og $x_1(t)$	16
5.4	Nærbilde sammenligning av $x_1[n]$ og $x_1(t)$	17
5.5	Periodogram av $x_1[n]$	17
5.6	Periodogram av $x_1[n]$ opp til 15625 Hz	18
5.7	Periodogram zero-padding	18
5.8	Periodogram zero-padding nærbilde	19

5.9	Periodogram windowing	19
5.10	Periodogram windowing nærbilde	20
5.11	Periodogram med windowing og zero-padding	20
5.12	Periodogram av støy	21
5.13	Amplituderespons Pi-filter	22
5.14	Sammenligning mellom $VDDA(t)$ og $VDAA(t)$	23
6.1	Forhold mellom implementert DAC og ADC	25

2 INNLEDNING

I dagens samfunn blir vi stadig mer avhengige av teknologiske løsninger som hjelper oss å automatisere hverdagen vår. Sentralt innenfor denne overgangen er troverdige sensorer som hjelper oss å tolke omgivelsene rundt oss, for eksempel kamerasensorer i en bil. Det er dermed kritisk at signalene som lagres fra disse sensorene er troverdige, og behandles på riktig måte.

Ved bruk av en sensor ønsker vi å finne empirisk informasjon om omgivelsene rundt oss, noe som gjøres ved å innhente data om den størrelsen man velger å måle. For å kunne behandle signalet en sensor måler behøves naturligvis en datamaskin, noe som vil si at vi bare vil kunne hente ut diskrete verdier av signalet som måles. Dette er fordi en datamaskin ikke kan lagre et uendelig antall med datapunkter. Følgelig vil signalet som måles først diskretiseres, ofte av en analog-til-digital converter (ADC). Dette er illustrert i Figur 2.1.



Figur 2.1: Innkommende signal $x(t)$ måles først av et sensorelement, og diskretiseres så av en ADC. Det diskrete signalet er $x[n]$.

Her vil et innkommende kontinuerlig signal $x(t)$ først måles av et sensorelement, før det videre diskretiseres av en ADC. Det diskrete signalet $x[n]$ kan så lagres og behandles i en datamaskin.

Denne rapporten vil ta for seg oppsett av et system som sampler fem kontinuerlige signaler. Videre vil de samplede signalene behandles ved hjelp av digital signalbehandling. Formålet med rapporten er å verifisere at systemet virker som forventet. Det vil også undersøkes grad av støy i de digitaliserte signalene, samt å se på ulike egenskaper ved signalene i frekvensdomenett.

3 TEORI

3.1 Digitalisering i teorien

Sentralt i systemoppsettet er et sett med ADCer, som er delen av systemet som håndterer sampling av analoge signaler, for eksempel fra ultralydsensorer eller en radar, og konverterer dem til digitale verdier. På denne måten er det mulig å studere og manipulere fysiske signaler ved hjelp av digital signalbehandling.

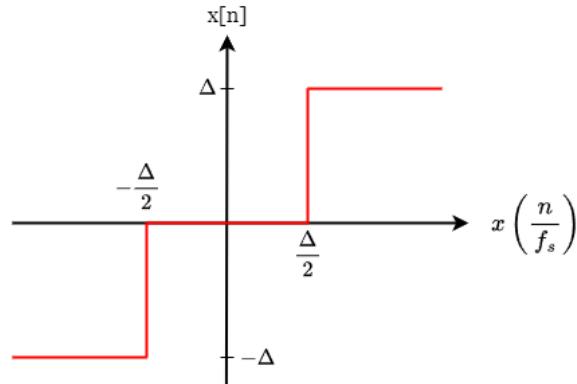
En konvertering fra en analog verdi til en digital verdi, krever sampling og kvantisering i amplitud, samt sampling i tid. Hvis vi betegner det analoge signalet som skal samples $x(t)$, det digitale signalet som $x[n]$ og feilen som introduseres i kvantiseringen som $\epsilon[n]$, vil

$$x[n] = x\left(\frac{n}{f_s}\right) + \epsilon[n] \quad (1)$$

beskrive diskretisering- og kvantiseringss prosessen ved et tidspunkt $t = \frac{n}{f_s}$. Her er f_s samplingsfrekvens og n sampelnummer til det digitale signalet $x[n]$. Feilfeilet $\epsilon[n]$ introduseres ettersom kvantisering vil runde av den analoge verdien $x\left(\frac{n}{f_s}\right)$. Dersom vi betegner de øvre og nedre avvikene fra den sanne verdien som henholdsvis $-\frac{\Delta}{2}$ og $\frac{\Delta}{2}$, vil feilfeilet $\epsilon[n]$ følgelig være distribuert i intervallet [1, s. 21]

$$-\frac{\Delta}{2} < \epsilon[n] < \frac{\Delta}{2}. \quad (2)$$

Figur 3.1 beskriver hvordan et analogt, kontinuerlig, signal $x\left(\frac{n}{f_s}\right)$ kvantifiseres og dermed avrundes ved et tidspunkt $t = \frac{n}{f_s}$.



Figur 3.1: Figuren viser hvordan forskjellige verdier av et analogt, kontinuerlig signal $x\left(\frac{n}{f_s}\right)$ blir kvantisert til et digitalt signal $x[n]$ ved tidspunktet $t = \frac{n}{f_s}$. Feilfeilet $\epsilon[n]$ vil alltid holde seg på ett ”trinn” og følgelig oppfylle ligning (2) som med ligning (1) gir at den kvantiserte verdien maksimalt vil avvike $\pm \frac{\Delta}{2}$ fra den kontinuerlige verdien.

Figur 3.1 viser de diskrete kvantiseringsnivåene som det digitale signalet $x[n]$ kan ta. Gitt at man ikke representerer negative verdier, vil hele bitdybden B til det digitale signalet $x[n]$ brukes for å representere en positiv tallverdi. Opplosningen, Δ , til det digitaliserte signalet som kan ta verdier mellom x_{min} og x_{max} med en bitdybde på B bits er gitt av [1, s. 33]

$$\Delta = \frac{x_{max} - x_{min}}{2^B - 1}. \quad (3)$$

Fra dette ser vi at en høyere bitdybde B vil minke feilledet i kvantiseringen. For kvantifisering av et vilkårlig valgt kontinuerlig signal, har vi ingen grunn til å tro at fordelingen til $\epsilon[n]$ ikke vil være uniformt fordelt i intervallet gitt av ligning (2) [1, s. 613–616]. For applikasjoner med lav bitdybde kan det derfor være nyttig å modellere feilledet som uniformt fordelt støy som legges til $x[n]$. Ved å bruke den modellerte, uniforme sannsynlighetsfordelingen til $\epsilon[n]$, kan det utledes at variansen til støyen blir [1, s. 613–620]

$$\sigma_\epsilon^2 = \frac{\Delta^2}{12}. \quad (4)$$

Ettersom $\epsilon[n]$ er et støysignal blir variansen, σ_ϵ^2 , et estimat for kvadratet av Root Mean Square (RMS) verdien til støyen, P_ϵ [2, s. 81]. Ved å betegne RMS effekten til signalet $x(t)$ som P_x får vi også det teoretiske signal-til-støyforholdet i desibel (dB), $SNR'_{[dB]}$, i desibel [1, s. 613–620]

$$SNR'_{[dB]} = 10 \log_{10} \left(\frac{P_x}{P_\epsilon} \right). \quad (5)$$

Dersom signalet $x(t)$ er en sinus med amplitude A , og signalet ligger over en motstand med verdien 1Ω , vil RMS effekten til $x(t)$, P_x , være gitt av [3, s. 37]

$$P_x = \frac{A^2}{2}. \quad (6)$$

For å gi et kvantitativt mål på graden signalene $x(t)$ og $x[n]$ ligner på hverandre, $S_{[dB]}$, kan vi bruke krysskorrelasjonen mellom $x(t)$ og $x[n]$, $r_{x_1(t)x_1[n]}[l]$, og autokorrelasjonen av $x_1(t)$, $r_{x_1(t)x_1(t)}[l]$, slik

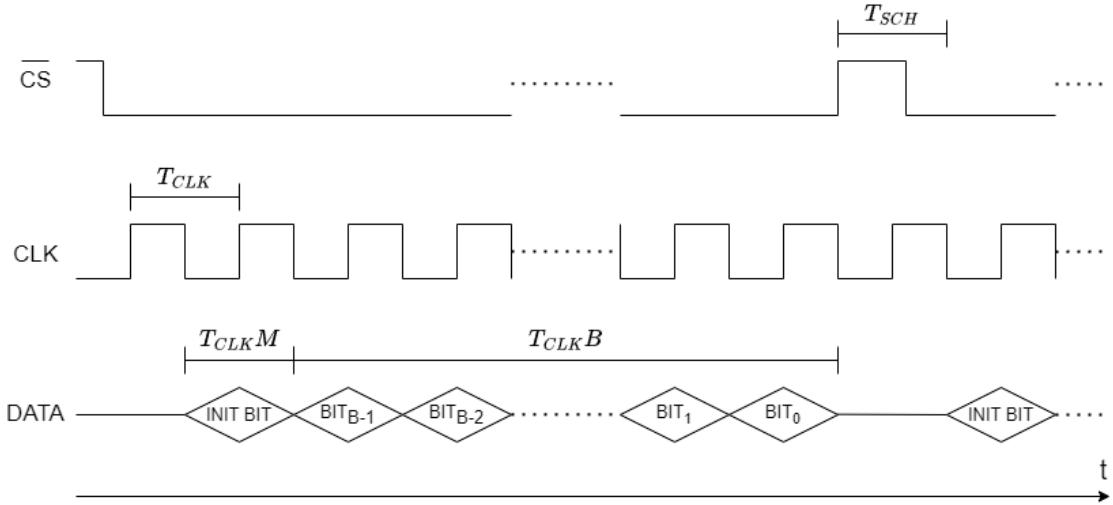
$$S_{[dB]} = 10 \log_{10} \left(\frac{\max\{r_{x_1(t)x_1[n]}[l]\}}{\max\{r_{x_1(t)x_1(t)}[l]\}} \right). \quad (7)$$

Verdien $S_{[dB]}$ blir dermed en ”likhetsgrad” og er 0 dersom signalene er identiske, og minker logaritmisk jo mer ulike signalene blir. I dette målet er det brukt den maksimale korrelasjonsverdien, og dette målet vil følgelig ikke fange ulikheter i signalene med tanke på faseforskyvninger. Dette er gjort ettersom $x(t)$ uansett ikke vil ha nøyaktig samme faseforskyvning som $x[n]$ på en felles tidsakse ¹. $S_{[dB]}$ fanger imidlertid godt opp forskjeller i frekvens og amplitude mellom signalene.

3.2 Digitalisering i praksis

I praksis gjøres digitaliseringen ved hjelp av en ADC som tar punktprøver av et analogt signal og digitaliserer denne punktprøven til en digital verdi med endelig oppløsning. For at denne endelige oppløsningen skal ha minst mulig innvirkning på det ønskede digitale signalet $x[n]$ (les: høy SNR) ser vi av ligning (2) og (3) at høy bitdybde B og et signal $x(t)$ som varierer så nært x_{min} og x_{max} som mulig er ønskelig. Arbeidsområdet til $x(t)$ kan justeres ved for eksempel en spenningsdeler, men bitdybden B avhenger av hvor stor samplingsfrekvens systemet krever (dersom ADCene bruker seriell kommunikasjon). Ta eksempelet på et timingdiagram for seriell overføring av data i Figur 3.2.

¹Mer om dette i Seksjon 5.



Figur 3.2: Eksempel på hvordan et timingdiagram kan se ut for seriell overføring av data. Det digitale signalet \overline{CS} angir når seriell data skal legges ut på $DATA$, og en ny bit legges på linjen ved hver synkende flanke av CLK . Ettersom klokkeperioden er T_{CLK} blir klokkefrekvensen i figuren $f_{CLK} = T_{CLK}^{-1}$. Perioden T_{SCH} angir tiden fra siste bit i en transaksjon til første bit i neste transaksjon. Variabelen M angir antall klokkesykler i transaksjonen som ikke brukes på de B informasjonsbærende bitene. Tidsaksen er indikert i bunnen av timingdiagrammet.

Fra Figur 3.2 kan det sees at antall klokkesykler som brukes når en sampel overføres serielt kan betegnes som $B+M$ bits, der alle B bitene utgjør et element i signalet $x[n]$ og M er andre klokkesykler som ikke direkte brukes til å bære de B bitene serielt. Perioden mellom hver sampel som overføres, T_s , vil med en klokkefrekvens på f_{CLK} , og en tid T_{SCH} mellom starten og slutten på to etterfølgende $B+M$ bits være gitt av

$$T_s = \frac{B + M}{f_{CLK}} + T_{SCH}. \quad (8)$$

Den teoretiske, maksimale samplingsfrekvensen, f'_s , blir derfor

$$f'_s = \frac{1}{T_s} = \left(\frac{B + M}{f_{CLK}} + T_{SCH} \right)^{-1}. \quad (9)$$

Denne utledningen forutsetter at klokkeperioden T_{CLK} og T_{SCH} er like hver gang, noe som også bør være tilfelle dersom ligning (9) skal kunne brukes.

3.3 Signalbehandling

3.3.1 Samplingsfrekvens

Viktig for sampling av analoge signaler er samplingsfrekvensen f_s . Nyquist samplingsteorem sier at samplingsfrekvensen f_s må minst være dobbelt så stor som båndbredden til signalet man sampler, dette for å unngå aliasing [4]. Dette kan uttrykkes som

$$f_s \geq 2 \cdot f_{max}, \quad (10)$$

hvor f_{max} er høyeste frekvens i signalet man sampler. Dersom ligning (10) er oppfylt, sørger man for at videre signalbehandling med samplet signal blir riktig.

3.3.2 DFT og FFT-algoritme

For å studere egenskaper til det samplede signalet $x[n]$ er det nyttig å studere signalet i frekvensdomenet, og dermed benytte en FFT-algoritme. En FFT-algoritme er en algoritme som først bryter signalet $x[n]$ inn i mindre biter. Videre vil den diskrete fourier transformen (DFT), gitt i ligning (11), til hver oppdelte bit av $x[n]$ bli beregnet. Resultatet av hver DFT vil så kombineres sammen i flere steg for å beregne frekvensspekteret til $x[n]$. Alternativet ville vært å beregne DFT til hele $x[n]$ direkte. Fordelen ved å bruke FFT istedenfor DFT på $x[n]$ er at beregning av frekvensspekteret kan utregnes mye raskere [5].

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \quad n = 0, 1, \dots, N-1 \quad (11)$$

Her er k/N normaliserte frekvenser, og har sammenhengen $f = \frac{f_s k}{N}$ med de analoge frekvensene i signalet $x(t)$. f er frekvens, f_s samplingsfrekvens og N totalt antall diskrete punkter i $x[n]$.

3.3.3 Zero-padding

Det finnes flere eksisterende FFT-algoritmer, og felles for mange av dem er at de blir mer effektive når signallengden N er på formen [6]

$$N = 2^\gamma \quad \gamma \in \mathbb{N}. \quad (12)$$

Dersom $x[n]$ har en signallengde N innenfor intervallet $[2^{\gamma-1}, 2^\gamma]$, vil FFT-utregningen uansett omtrent ta like lang tid. Å ha en signallengde på formen i ligning (12) er dermed ønskelig fordi for en gitt tid FFT-utregningen bruker innenfor overnevnte intervall, vil oppløsningen på frekvensspekteret maksimeres. Av ligning (11) ser man at økt N gir bedre oppløsning i frekvensdomenet. Zero-padding kan også tydeliggjøre effekten windowing (beskrevet i Seksjon 3.3.4) har på $x[n]$ sitt frekvensspekter.

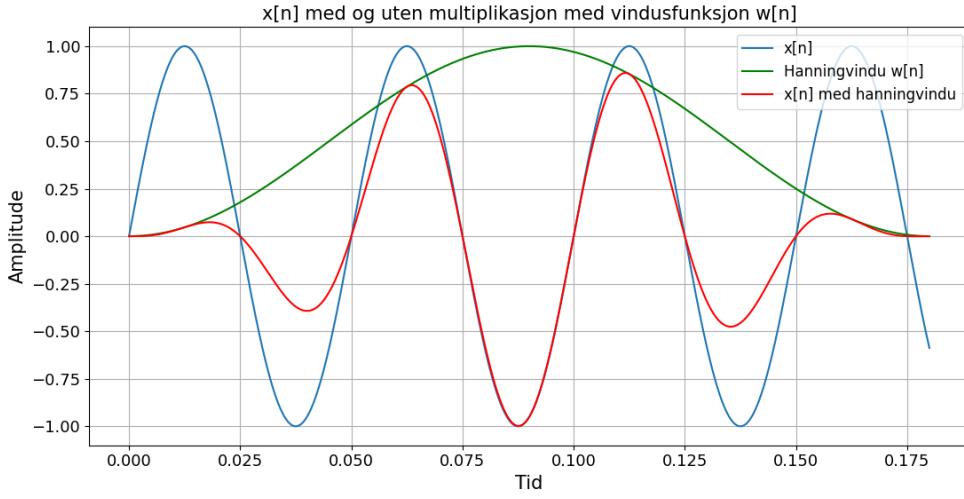
Dersom signalet $x[n]$ ikke har en lengde som kan uttrykkes på formen i ligning (12), så er det mulig å øke lengden på signalet slik at ligning (12) er oppfylt. Da legger man til signalverdier 0 på slutten av $x[n]$, kalt zero-padding, slik at man sørger for økt relativ hastighet på FFT-utregningen. I tillegg vil oppløsningen til frekvensspekteret forbedres [7]. Merk imidlertid at vi ikke får mer informasjon om frekvensinnholdet til $x[n]$, frekvensspekteret ser bare jevnere ut.

3.3.4 Vindusfunksjoner

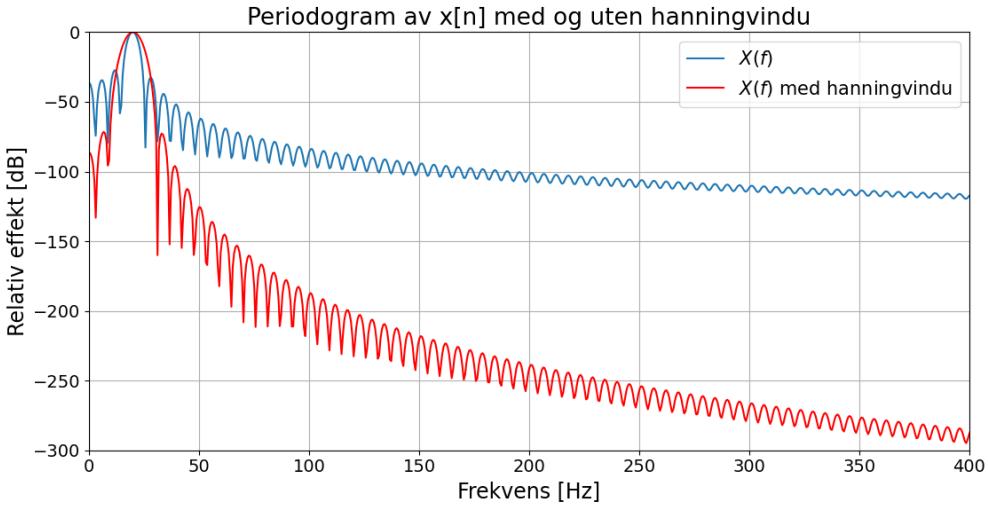
Ved sampling av et analogt signal $x(t)$ vil det samplede signalet $x[n]$ i praksis alltid være av endelig lengde. Dette kan tenkes på som et signal med samme signalverdier innenfor tiden signalet samples, og signalverdi 0 ellers. Konsekvensen av å benytte et endelig signal er at frekvensspekteret til $x[n]$ vil bli deformert i form av spektral lekasje, det vil si sidelober rundt hovedlober der vi har de faktiske frekvensene til $x[n]$.

En måte å redusere effekten av spektral lekasje er å multiplisere $x[n]$ med en vindusfunksjon $w[n]$. En vindusfunksjon er en funksjon som vekter hvert sampel i $x[n]$. Det viser seg at dette kan bevare de faktiske frekvensene i $x[n]$, samtidig som den relative effekten til frekvenser inneholdt i sidelobene reduseres i forhold til hovedlobene [8].

Figur 3.3 viser et eksempel på et signal $x[n]$ med frekvens $f = 20$ Hz før og etter multiplikasjon med et hanningvindu $w[n]$. Figur 3.4 viser tilhørende periodogram før og etter windowing.



Figur 3.3: $w[n]$ vekter $x[n]$ og dermed deformerer det originale signalet $x[n]$.

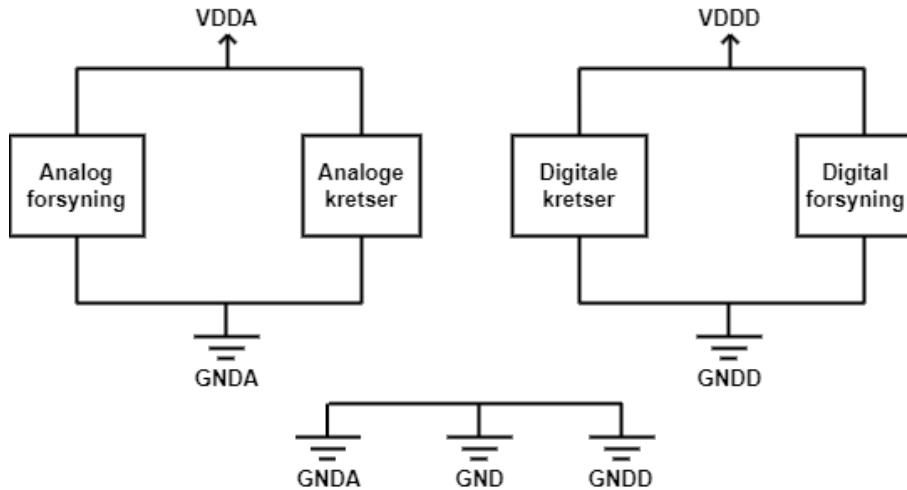


Figur 3.4: Hovedloben rundt $f = 20$ Hz bevares, mens sidelobene sin relative effekt i forhold til hovedloben reduseres.

Her ser man at resultatet av windowing er at den relative effekten til sidelobene i $X(f)$ reduseres i forhold til hovedloben som er sentrert rundt $f = 20$ Hz. En konsekvens er at båndbredden til hovedloben blir større. Det finnes flere typer vindusfunksjoner, alle med sine fordeler og ulemper [8].

3.4 Støy og filtrering

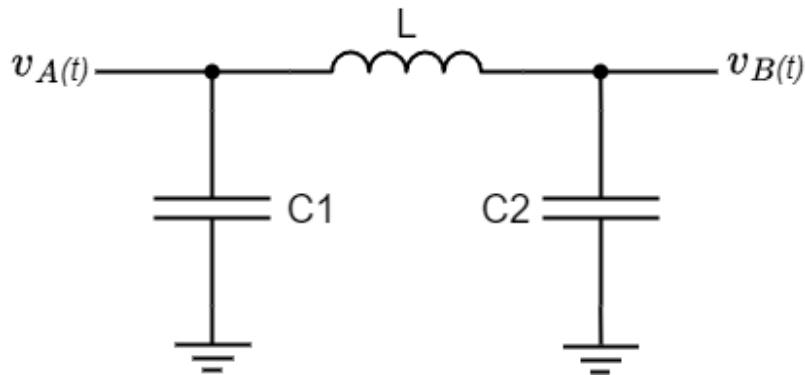
En ”mixed-design” krets er en krets hvor man både behandler analoge og digitale signaler på samme koblingsbrett/kretskort. En slik implementasjon kan føre til at høyfrekvente digitale signaler induserer støy i nærliggende analoge linjer og forsyningssignaler [2, s. 58]. For å minimere denne støyen burde domenene skilles så godt som fysisk mulig. Videre kan man også implementere separate analoge og digitale forsyninger, slik at eventuelle forstyrrelser i den digitale delen av kretsen ikke propagerer over til den analoge delen [2, s. 71]. Et blokkdiagram som viser hvordan dette kan gjøres er vist i Figur 3.5.



Figur 3.5: Figuren viser et blokkdiagram over hvordan man kan skille analoge og digitale forsyninger. Terminalene for analog forsyning og analog jord er benevnt VDDA og GNDA. Terminalene for digital forsyning og digital jord er benevnt VDDD og GNDD. Stjernejord er benevnt GND. Merk at det er ønskelig å legge stjernejord på en liten del av koblingsbrettet/kretskortet for å hindre uønskede strømsløyfer som induserer støy og interferens [9] [2, s. 71].

Andre tiltak for å hindre støy på likespenningskilder, inkluderer bruk av avkoblingskondensatorer. En avkoblingskondensator fungerer effektivt sett som en kortslutning for høyfrekvente signaler, og eliminerer høyfrekvent støy på forsyningslinjer.

Komplette lavpassfiltere kan også brukes for å filtrere bort støy på analoge signaler. Et spesielt egnet filter som ofte brukes til dette er et såkalt Pi-filter. Et kretsskjema av hvordan et lavpass Pi-filter kan se ut, med en spole med induktans L og kondensatorer med kapasitans C_1 og C_2 , er vist i Figur 3.6.



Figur 3.6: Figuren viser hvordan et Pi-filter kan se ut. Spenningen $v_B(t)$ vil inneholde flesteparten av de lave frekvensene til $v_A(t)$.

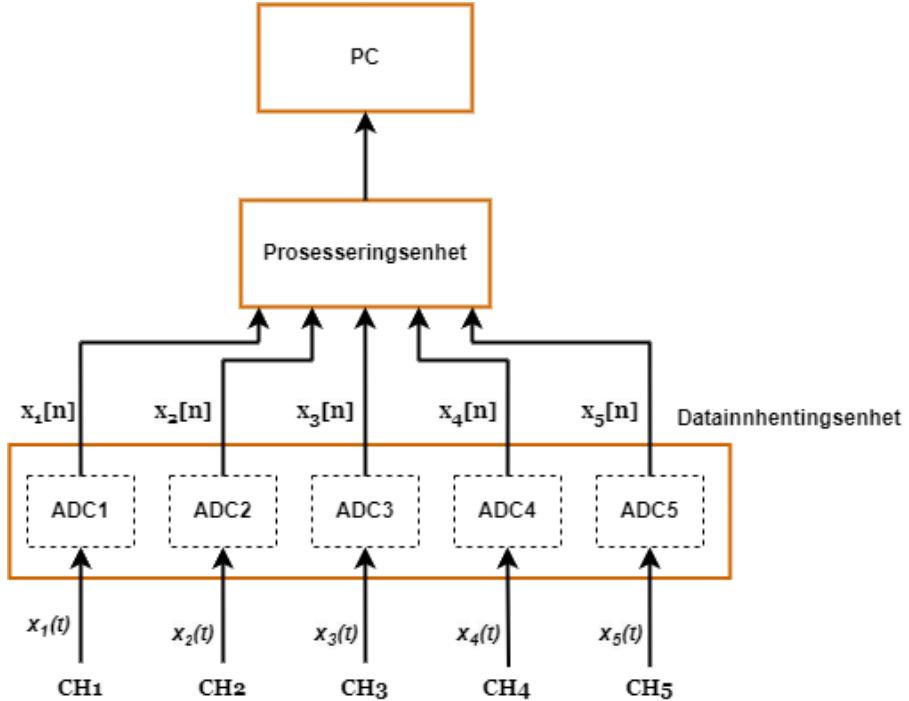
Dersom spenningen $v_A(t)$ inneholder høye frekvenskomponenter vil kondensatoren til venstre dempe disse, og spolen vil slippe gjennom de laveste frekvensene. Kondensatoren til høyre vil dempe gjenværende høye frekvenskomponenter. Knekkfrekvensen til filteret kan finnes ved å behandle kondensatoren C_1 som en avkoblingskondensator med konstant spenning for lave frekvenser. En tilnærming til frekvensresponsen finnes ved å se på spolen med verdi L og kondensatoren med verdi C_2 . Ettersom C_1 ikke er med i beregningen av frekvensresponsen, kan et anslag for den teoretiske knekkfrekvensen, f'_c , finnes ved å sette absoluttverdien av frekvensresponsen lik $\frac{1}{\sqrt{2}}$. Resultatet blir [10]

$$f'_c \approx \frac{1}{2\pi\sqrt{LC}}. \quad (13)$$

Den komplette frekvensresponsen når kondensatoren $C1$ også tas med kan finnes analytisk, men er ikke like triviell som med et ordinært RC-lavpassfilter.

4 EKSPERIMENTELT OPPSETT

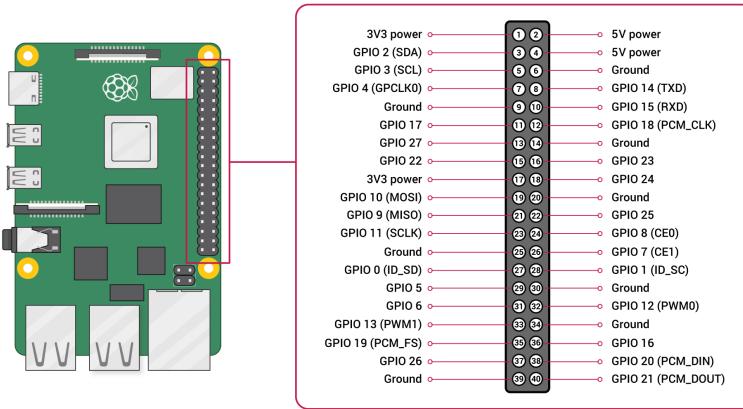
Formålet med systemet er å gjøre målinger av 5 analoge signaler $x_1(t)$ - $x_5(t)$ for så å utføre digital signalbehandling på den digitaliserte versjonen $x_1[n]$ - $x_5[n]$ av dem. De analoge signalene må derfor først samples og kvantifiseres i ADCer, for derefter å prosesseres i en prosesseringsenhet. I tillegg må enheten som innhenter de digitaliserte signalene styres, noe som kan gjøres med en ekstern PC med en nettverksprotokoll. Denne prosessen kan oppsummeres i blokkskjemaet vist i Figur 4.1 hvor signal nummer s , $x_s(t)$, sendes inn på sin respektive kanal CHs hvor så ADCen gir ut det digitaliserte signalet $x_s[n]$.



Figur 4.1: Blokkskjema av et system som sampler og prosesserer 5 analoge signaler $x_1(t)$ - $x_5(t)$ på kanalene CH1-CH5. En PC blir brukt for å motta dataen fra prosesseringsenheten.

4.1 Prosesseringsenhet

Som prosesseringsenhet brukes en Raspberry Pi 3B med et fordelerkort, Pi Wedge, som letter implementasjonen på et koblingsbrett [11][12]. Dataoverføringen mellom ADCene og prosesseringsenheten skjer ved hjelp av Serial Peripheral Interface (SPI) [13]. Denne synkrone kommunikasjonsprotokollen benytter seg av en master-slave struktur hvor masteren igangsetter og styrer all kommunikasjon. Protokollen bruker 4 signaler for kommunikasjonen: Serial Clock (*SCLK*), Master Out Slave In (*MOSI*), Master In Slave Out (*MISO*) og Chip Select (*CS*) [13]. En av oppgavene til det gitte programmet `adc_sampler.c` er å styre all kommunikasjonen over SPI, se Appendix A. I programmet defineres derfor General Purpose Input Output (GPIO) terminaler opp mot hvilke SPI signaler de skal behandle. De ulike GPIO terminalene på Raspberry Pi 3B, er vist i Figur 4.2.



Figur 4.2: De ulike GPIO terminalene til Raspberry Pi 3B. Figuren er hentet fra [14].

Tabell 4.1 viser en oversikt over tilordningen mellom GPIO terminaler og hvilke SPI signaler de tilknyttes.

Tabell 4.1: Tilordningen mellom SPI signaler og GPIO terminaler.

SPI signal	GPIO terminal
<i>MISO</i> ₁	17
<i>MISO</i> ₂	16
<i>MISO</i> ₃	13
<i>MISO</i> ₄	12
<i>MISO</i> ₅	6
<i>MOSI</i>	18
<i>CS</i>	8
<i>SCLK</i>	11

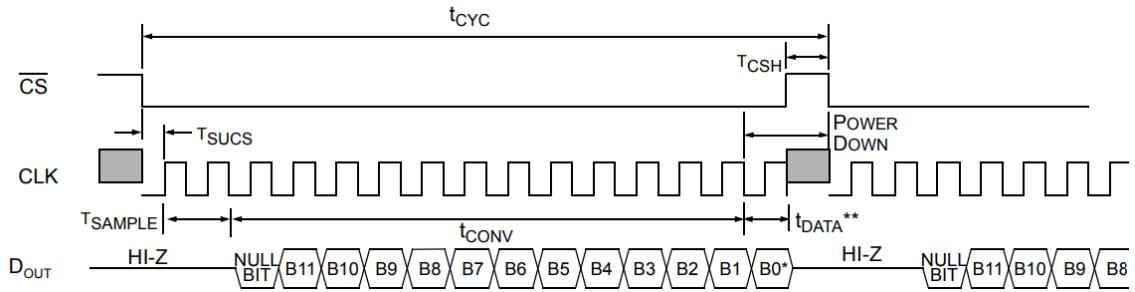
En annen oppgave til programmet `adc_sampler.c` er å sette opp Direct Memory Access (DMA) for brukte GPIO terminaler [15]. Bruken av DMA er for å sikre at prosessoren ikke er opptatt med selve innhenting av det digitaliserte signalet $x_s[n]$, men blir heller varslet av DMA når dataen ligger klar i minnet [15]. Programmet bruker biblioteket `pigpio` for å sette opp denne håndteringen [16]. Gitt at kompilatoren `gcc` og `pigpio` er installert, kan programmet kompileres med kommandoen

```
gcc -Wall -lpthread -o adc_sampler adc_sampler.c -lpigpio -lm
```

og kjøres med kommanoden

```
sudo ./adc_sampler 31250
```

. Samplingsfrekvensen f_s er satt til $f_s = 31250$ Hz. Programmet begynner da å faselitere kommunikasjonen med ADCene. Den maksimale, teoretiske samplingsfrekvensen f'_s som kan oppnås kan finnes ved å studere Figur 3.2 opp mot timingdiagrammet i databladet til ADCen MCP3201 [17]. Dette er vist i Figur 4.3.



Figur 4.3: Timingdiagrammet til ADCen MCP3201. Figuren er hentet fra databladet [17].

Med konvensjonen etablert i Seksjon 3.2 får vi fra Figur 4.3 at $M = 3$ og $B = 12$. Perioden mellom etterfølgende samples, T_{SCH} , blir $T_{SCH} = T_{CSH} + T_{SUCS} + T_{SAMPLE}$. Klokkefrekvensen, f_{CLK} , finnes i programmet `adc_sampler.c` til å være $f_{CLK} = 500$ kHz, se Appendiks A. Ved å sette inn verdier for de ulike periodene, gir ligning (9) den teoretiske samplingsfrekvensen f'_s lik

$$f'_s = \left(\frac{12 + 3}{0.5} \mu s + 0.625 \mu s + 0.1 \mu s + 3 \mu s \right)^{-1} = 33.6 \text{ kHz.} \quad (14)$$

Merk imidlertid at det i dette tilfellet er brukt minimumsverdier for periodene fra databladet, noe som dermed gir den maksimale samplingsfrekvensen man teoretisk kan oppnå. Den effektive samplingsraten f_s kan derfor avvike noe fra den teoretiske samplingsfrekvensen f'_s over. Etter å ha samplet med den effektive samplingsfrekvensen f_s i ett sekund skriver programmet den samplede daten til en binær fil. For kommunisering mellom PC og prosesseringenheten brukes nettverksprotokollen Secure Shell (SSH) samt programmet WinSCP for filoverføring [18].

Binærfilen med den samplede dataen leses og prosesserer med programmet `frekvensspektrum_generator.py`, se Appendiks B.

4.2 ADC

For å digitalisere de analoge signalene er delsystemet datainnhentingsenhet vist i Figur 4.1 implementert som 5 ADCer av typen MCP3201 [17]. Dette er en enkanals, 12-bits ADC som kommuniserer med en tilsvarende seriell SPI protokoll beskrevet i Seksjon 4.1. Ettersom den er en enkanals ADC, skal det ikke gå annen informasjon enn *SCLK* og *CS* fra Raspberry Pien til ADCene. Dette resulterer i at det bare er SPI datasignalet *MISO* som er i bruk, som følgelig blir det digitale signalet på linja *Dout* vist i Figur 4.6. Videre har ADCene en analog inngang, *Vref*, hvor spenningen på denne terminalen er den største analoge spenningen som kan representeres av ADCene. Det digitaliserte signalet fra ADCene blir etter noen bits for initialisering av kommunikasjon lagt serielt på linjen *Dout* som en digital verdi mellom 0 og 4095. Den digitale verdien for et sampel $x_s[n]$ fra ADCs er gitt av

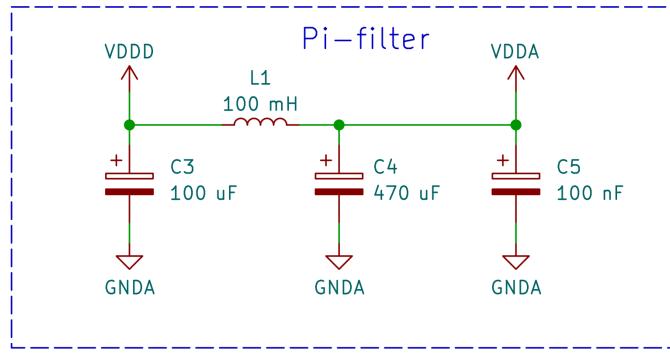
$$x_s[n] = \frac{V_{ref}}{4095} \text{round} \left\{ \frac{4095}{V_{ref}} x_s \left(\frac{n}{f_s} \right) \right\}. \quad (15)$$

De resterende terminalene til ADCene er tilførsel (*Vdd* og *Vss*), og en differensiell inngang for det analoge inngangssignalet $x_s(t)$, *IN-* og *IN+* [17].

4.3 Filter

Separate analoge og digitale forsyninger er som beskrevet i Seksjon 3.4 ønskelig for å hindre høyfrekvent støy på de analoge delene av kretsen. I denne implementasjonen brukes også separate forsyninger der den analoge forsyningen, *VDDA*, er koblet til den digitale forsyningen, *VDDD*,

ved hjelp av et Pi-filter. Analog og digital jord, GNDA og GNDD, er koblet rett i stjernejord GND. Kretsskjema til Pi-filteret er vist i Figur 4.4.



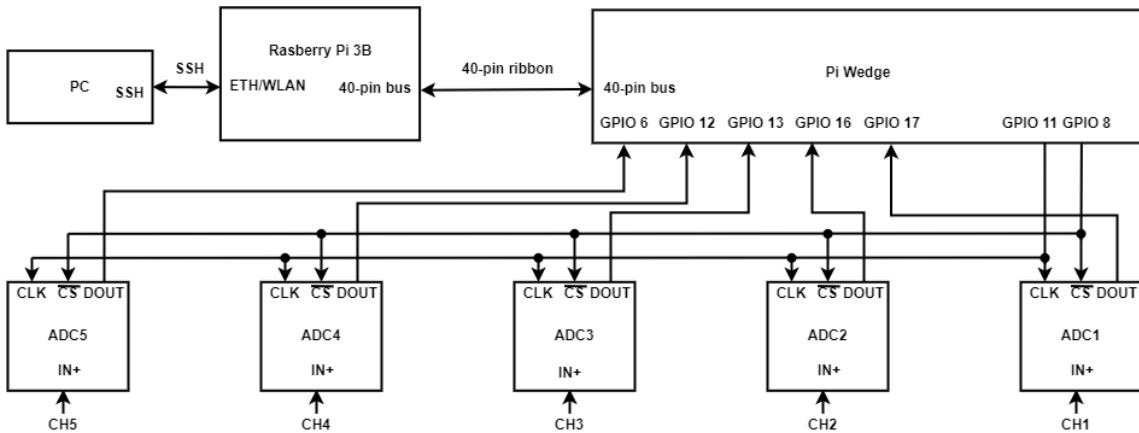
Figur 4.4: Kretsskjema av det implementerte Pi-filteret.

Komponentverdiene er funnet på bakgrunn av beskrivelsen i Seksjon 3.4. Verdien til kondensatorene C_3 og C_4 må være stor nok (les: i størrelsesordenen μF) slik at høyfrekvent støy blir dempet tilstrekkelig mye. Verdien til spolen L_1 velges også stor slik at mer av den høyfrekvente støyen dempes. Estimatet for den teoretiske knekkfrekvensen f'_c finnes av ligning (13)

$$f'_c \approx \frac{1}{2\pi\sqrt{L_1 C_4}} = 23.2 \text{ Hz.} \quad (16)$$

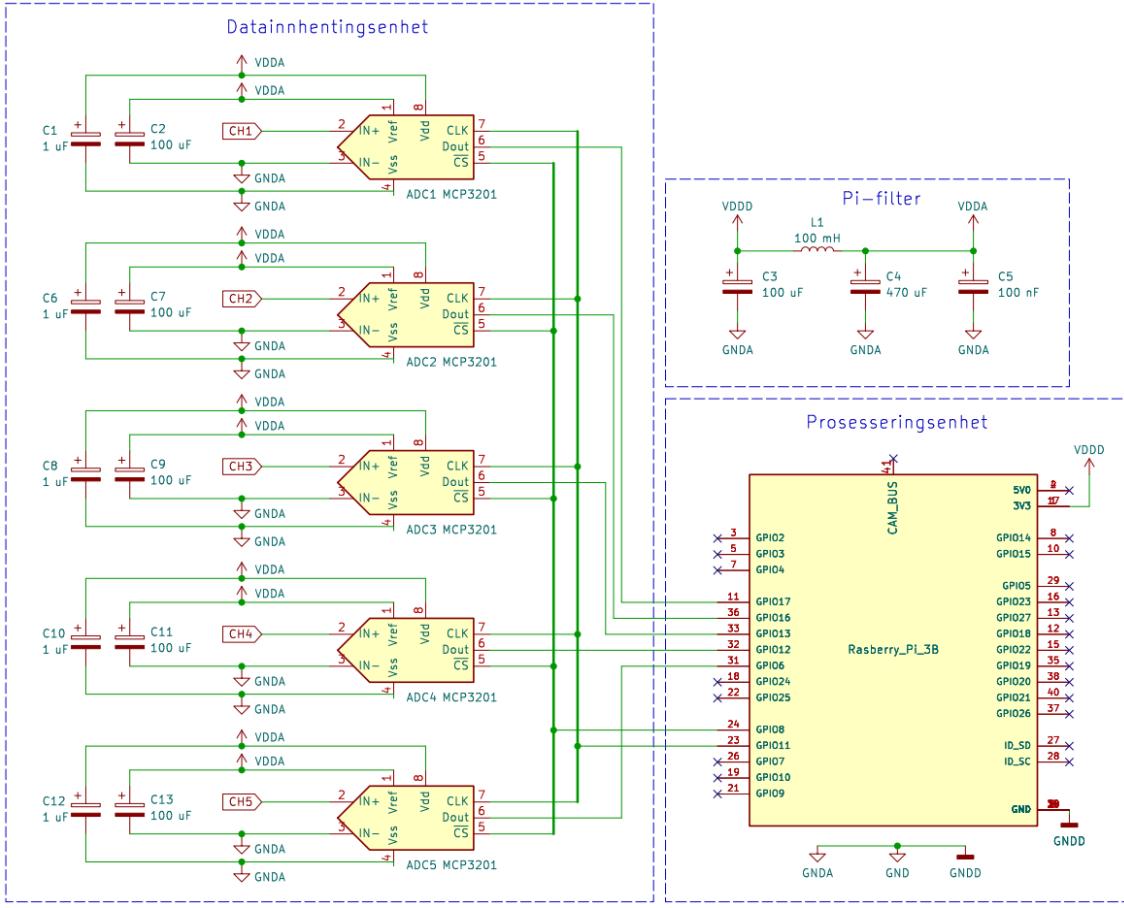
4.4 Systemsammenstilling

Et blokkskjema for alle signalene som ikke brukes som forsyning er utarbeidet for å synliggjøre hvordan signalene skal gå mellom ADCene og Raspberry Pien. Blokkskjemaet er vist i Figur 4.5.



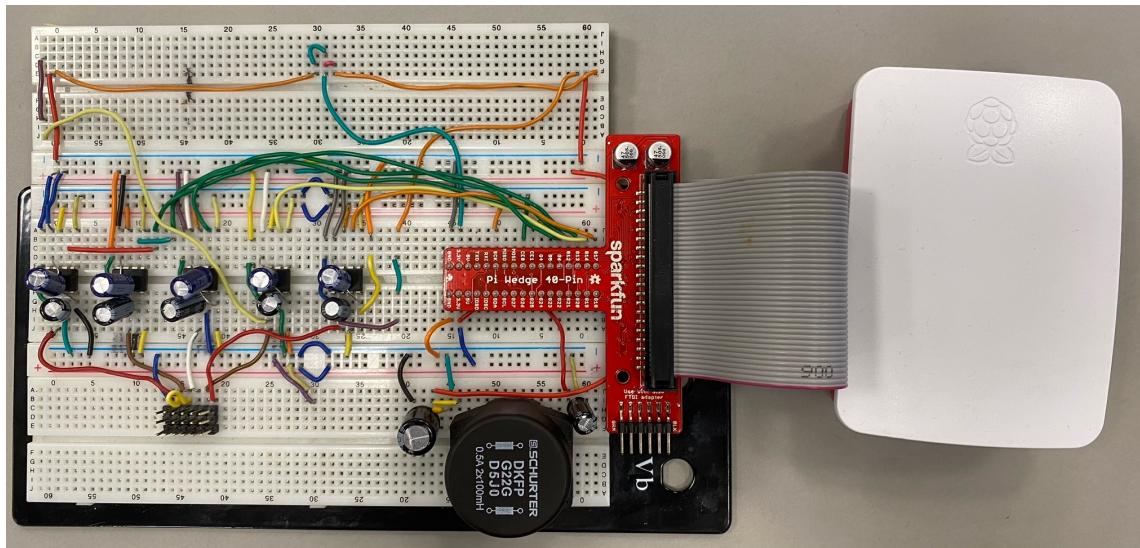
Figur 4.5: Blokkskjemaet viser hvordan alle signalene som ikke er koblet til forsyning skal gå innad i systemet.

Støyreduksjon i den analoge delen av systemet gjøres ved å bruke filteret i Figur 4.4. Videre brukes avkoblingskondensatorer mellom noder som er spesielt utsatt for høyfrekvent støy. Da spenningen mellom terminalene V_{ref} og IN^- vil ha direkte innvirkning på det digitaliserte signalet $x_s[n]$ dersom støy introduseres her, brukes en polarisert avkoblingskondensator på 100 μF mellom disse nodene. I forsyningen $VDDA$ til ADCene brukes polariserte avkoblingskondensatorer på 1 μF , ettersom disse linjene ikke er like utsatt. Verdiene begrunnes med at verdier i samme størrelsesorden er i finne i databladet til MCP3201 [17, s. 22]. Figur 4.6 viser komplett kretstegning av det realiserte systemet.



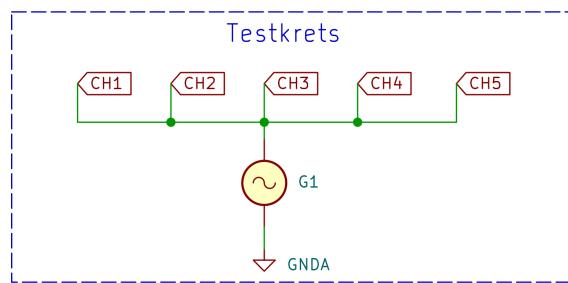
Figur 4.6: Komplett kretsskjemaet av systemet. Systemet er delt inn i tre hovedenheter: datainnhentingsenhet, prosesseringsenhet og filter. Datainnhentingsenheten består av 5 ADCer, og Raspberry Pi 3B utgjør prosesseringsenheten. Merk de adskilte jordingene GNDA, GNDD som forenes til stjernejord GND i bunnen av prosesseringsenheten. Merk også at avkoblingskondensatorene beskrevet over ligger parallelt over samme noder, hvor vi effektivt sett oppnår en kapasitans på 101 uF over begge nodene hvis vi ser bort fra at plasseringen av kondensatorene i forhold til tilkoblingspunktene kan ha noe å si.

Figur 4.7 viser det realiserte systemet.



Figur 4.7: Bilde av det realiserte systemet. Raspberry Pi som prosesseringsenhet kan ses til høyre på bildet, PiWedge og Pi-filter i midten, og systemet for datainntaking til venstre.

Testing og verifisering av systemet gjøres ved hjelp av en testkrets og en signalgenerator. Testkretsen med en signalgenerator merket $G1$ er vist i Figur 4.8.

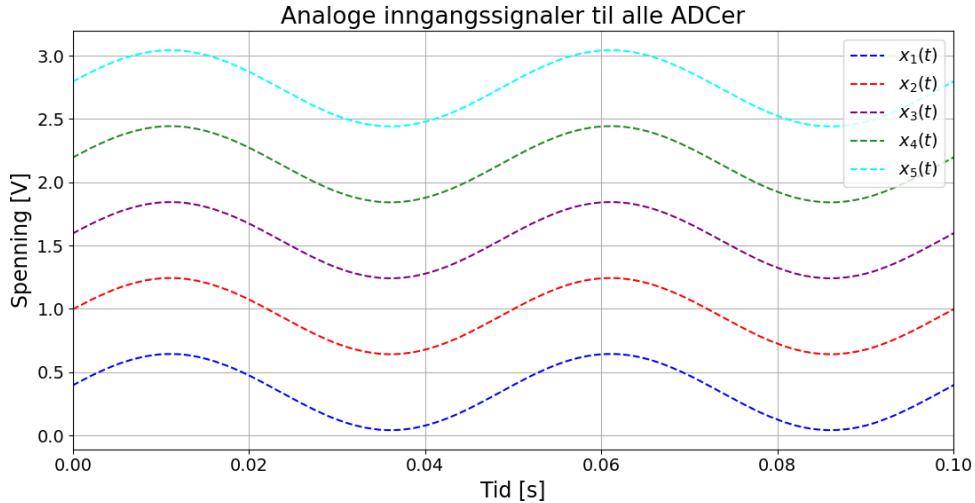


Figur 4.8: Kretstegning over hvordan systemet kobles for testing og verifikasjon.

5 RESULTATER

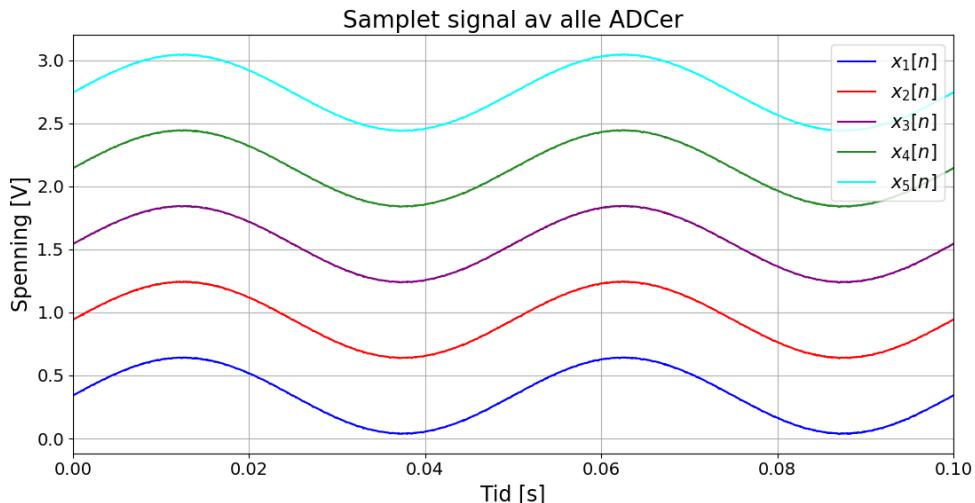
5.1 Sampling av analoge signaler

Analoge inngangssignal til hver kanal er vist i Figur 5.1. Inngangssignalene er en sinus med frekvens $f_1 = 20$ Hz og amplitude $A = 0.3$ V. Signaler vist i Figur 5.1 er samplet med oscilloskopet Analog Discovery 2 (AD2), noe som vil si at det også er diskrete samplede signaler [19]. I og med at samplingsfrekvensen til en AD2 er 0.1 MHz, noe som er mye høyere enn den oppnåde effektive samplingsfrekvensen f_s til ADCene (31.19(01) kHz med $k = 2$), kan signalene i Figur 5.1 tilnærmes identisk de analoge inngangssignalene [19].



Figur 5.1: Analoge inngangssignaler $x_1(t)$ - $x_5(t)$ på hver kanal.

De 5 digitaliserte signalene på hver kanal er vist i Figur 5.2. Kode for plotting er gitt i Appendiks B.



Figur 5.2: Samplede signaler $x_1[n]$ - $x_5[n]$ på hver kanal.

Som man kan se tilsvarer samplede signal $x_1[n]$ - $x_5[n]$ sin amplitude og frekvens med de påtrykte

signalene fra Figur 5.1. En forskjell er derimot faseforskyvningen, noe som skyldes at de analoge og digitale signalene er samplet på ulike tidspunkt. Det viktigste er at ADCene klarer å sample riktig frekvens og amplitude, og at alle samplede signal ikke er faseforskjøvet i forhold til hverandre, noe som er oppfylt. For å kunne si noe om hvor likt signalet $x_s[n]$ er i forhold til det påtrykte signalet $x_s(t)$, brukes det relative målet for likhet etablert i ligning (7). Den relative likheten i frekvens og amplitude mellom signalet $x_s(t)$ og $x_s[n]$, $S_{[dB],s}$, for alle signalene er vist Tabell 5.1.

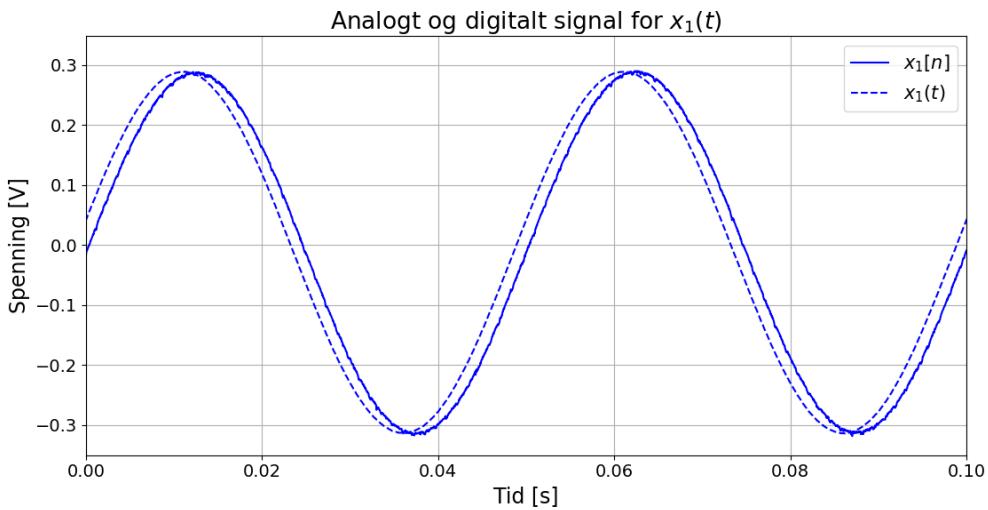
Tabell 5.1: Tabellen viser ulike verdier for likhetsgraden, $S_{[dB]}$, mellom alle korresponderende analoge og digitale signaler. Den gjennomsnittlige likhetsgraden er for alle kanalene er notert $\hat{S}_{[dB]}$. Fire desimaler er inkludert for å se forskjell på likhetsgradene.

Likhetsgrad	Verdi
$S_{[dB],1}$	-2.9393 dB
$S_{[dB],2}$	-2.9387 dB
$S_{[dB],3}$	-2.9388 dB
$S_{[dB],4}$	-2.9389 dB
$S_{[dB],5}$	-2.9387 dB
$\hat{S}_{[dB]}$	-2.9389 dB

Videre analyse vil bare ta for seg signal $x_1[n]$ fra ADC1, der dens DC-komponent er fjernet, dette for å fokusere på påtrykt frekvens samt støy i signalet.

5.2 Sammenligning med analogt signal

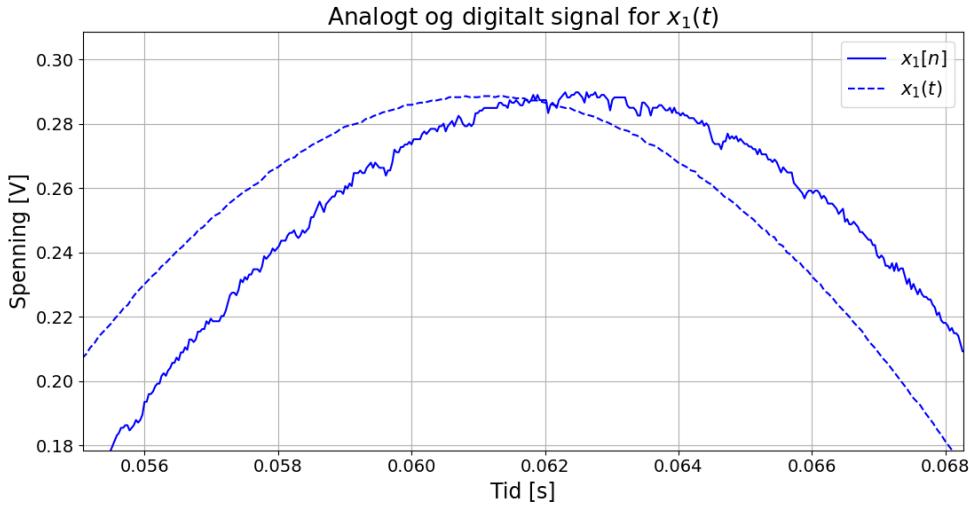
Figur 5.3 viser $x_1[n]$ sammenlignet med det analoge inngangssignalet $x_1(t)$.



Figur 5.3: Sammenligning av samplet signal $x_1[n]$ med det analoge inngangssignalet $x_1(t)$.

Ved observasjon ser signalene relativt like ut. Det relative likhetsmålet vist i Tabell 5.1 bekrefter også dette, hvor likheten mellom $x_1(t)$ og $x_1[n]$ er $S_{[dB],1} = -2.9393$ dB.

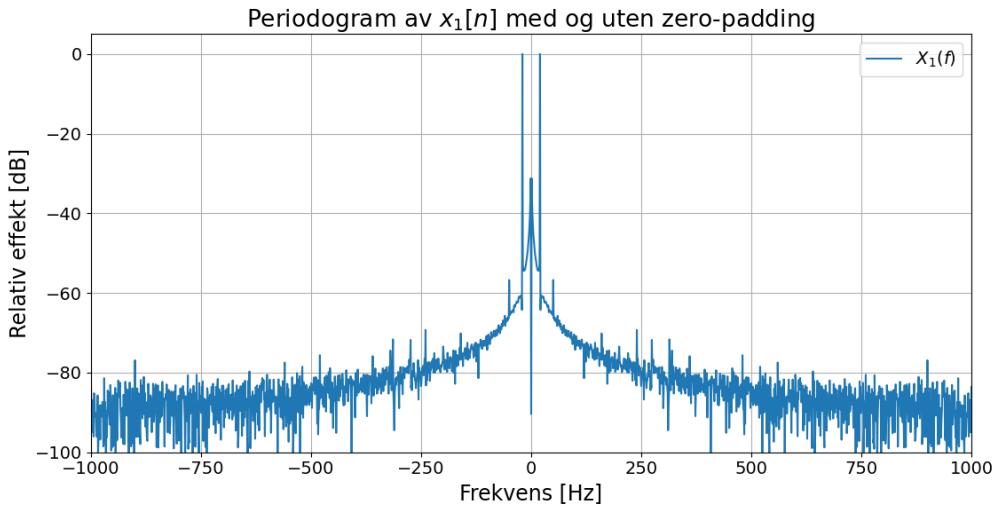
En merkbar observasjon er at både analogt og digitalt signal svinger mellom -0.31 V og 0.29V. For å fjerne likespenningskomponenten er funksjonen ”*detrend*” fra python-biblioteket ”*scipy.signal*” benyttet [20]. Grunnen til at analogt og digitalt signal ikke svinger fra -0.3 V til 0.3 V etter funksjonen utføres er ukjent. Videre kan man se i Figur 5.4 at $x_1[n]$ er mer støyete en $x_1(t)$, dette kan være på grunn av kantifiseringsstøy og annen støy og interferens i systemet. Signalet $x_1(t)$ påvirkes også av støy i og med at det er samplet med en AD2, men ikke i like stor grad som $x_1[n]$.



Figur 5.4: $x_1[n]$ inneholder mer støy en det analoge signalet $x_1(t)$.

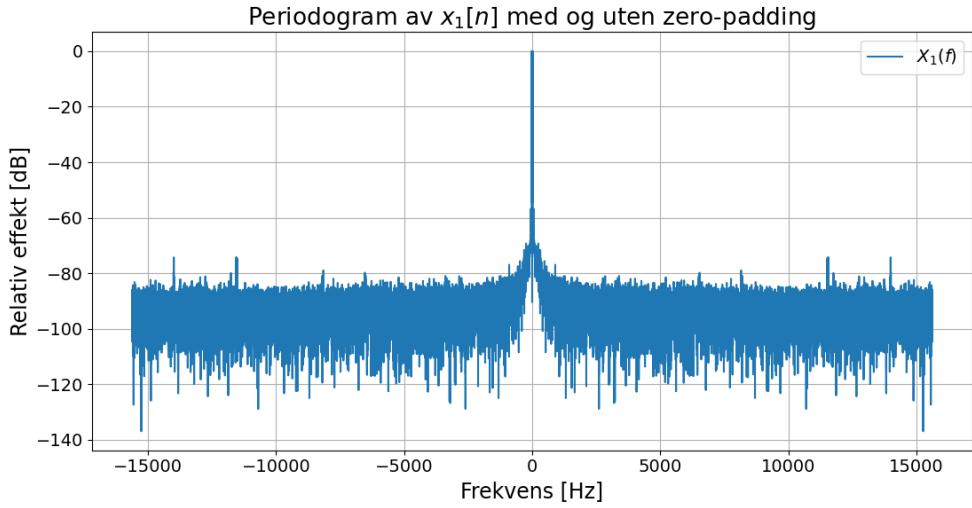
5.3 Frekvensspekter

FFT av $x_1[n]$ er beregnet ved hjelp av funksjonen ”*fft*” fra python-biblioteket ”*scipy.fft*” [21]. Periodogrammet til $x_1[n]$ er vist i Figur 5.5. Kode for plotting av periodogrammet er gitt i Appendiks B.



Figur 5.5: Periodogram av $x_1[n]$, der periodogrammet domineres av frekvensen $f_1 = 20$ Hz.

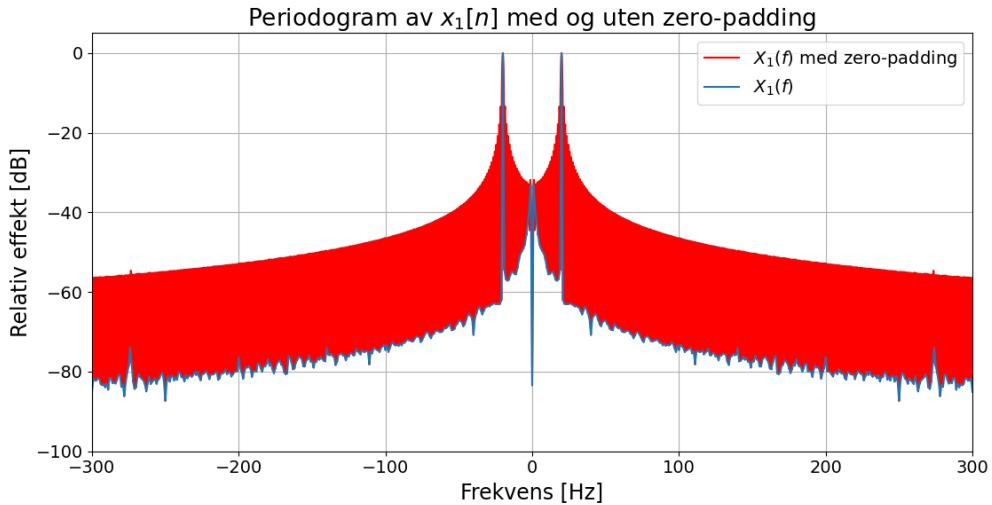
Ut fra periodogrammet vil $x_1[n]$ domineres av en frekvens $f_1 = 20$ Hz, noe som er i samsvar med Figur 5.2. Periodogrammet har et støygulv med relativ effekt omrent lik -90 dB, vist i Figur 5.6 [22].



Figur 5.6: Periodogram av $x_1[n]$ opp til 15625 Hz.

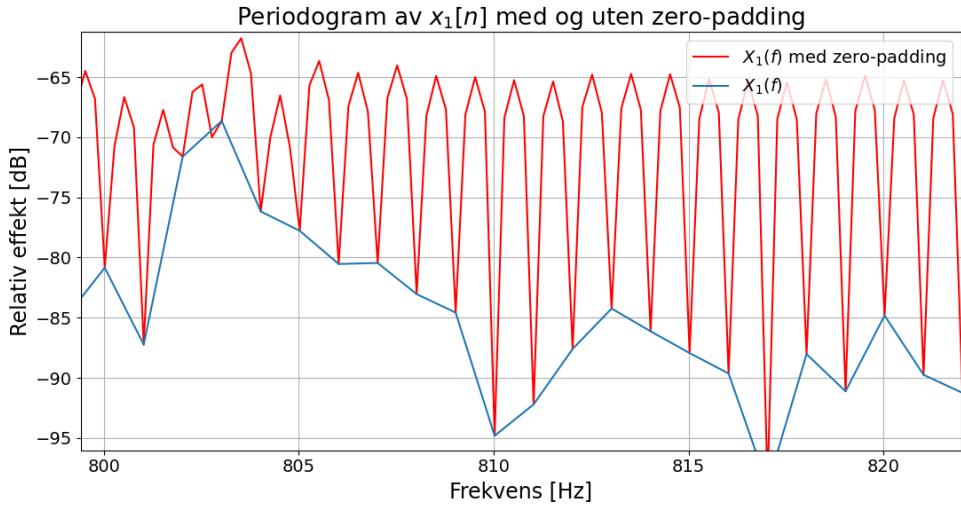
5.4 Zero-padding

Figur 5.7 viser periodogrammet til $x_1[n]$ med og uten zero-padding.



Figur 5.7: Periodogram av $x_1[n]$ med og uten zero-padding. Signalet er zero-paddet slik at $X_1(f)$ med zero-padding har lengde fire ganger så lang som uten zero-padding.

Med zero-padding ser man tydelig fra periodogrammet effekten av spektral lekasje. For frekvenser rundt f_1 heves støygulvet med omtrent 20-30 dB, mens for høyere frekvenser ligger støygulvet fortsatt på -90 dB. Signalet er zero-paddet slik at signalet blir fire ganger så langt, noe som gir tre flere sampler mellom samplene uten zero-padding i frekvensdomenet. Dette er vist i Figur 5.8.

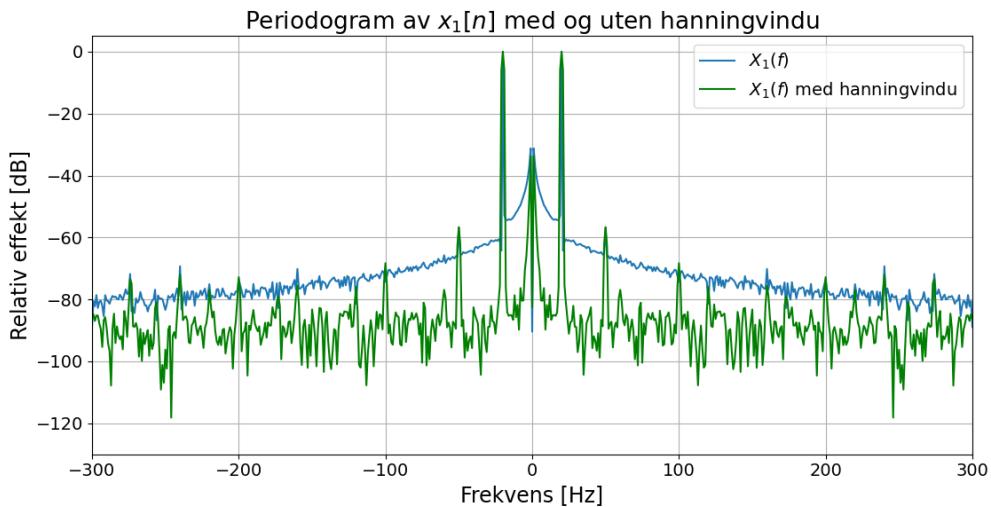


Figur 5.8: Periodogram av $x_1[n]$ med og uten zero-padding.

Merk at hvert datapunkt for $X_1(f)$ tilsvaret et punkt på grafen til $X_1(f)$ med zero-padding. Med zero-padding økes altså oppløsningen i frekvensdomenet, slik at man i større grad ser effekten av spektral lekasje.

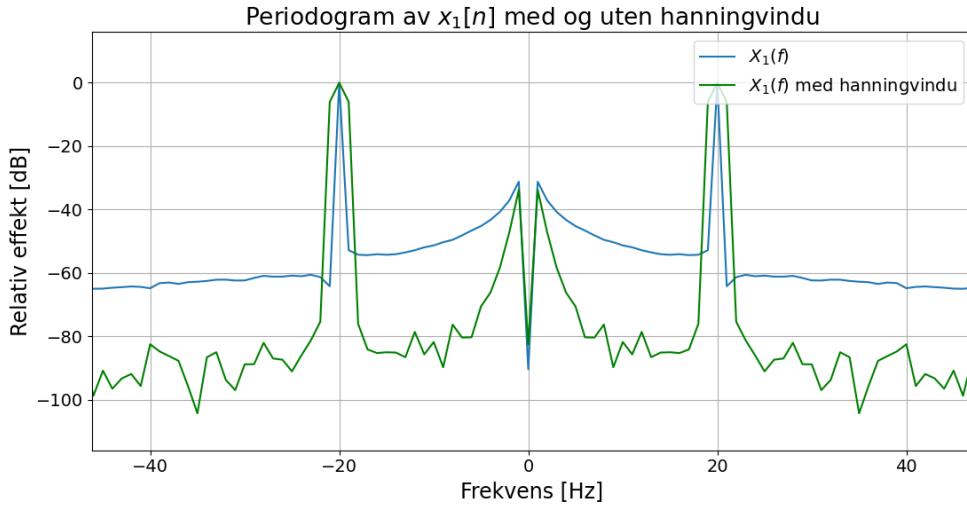
5.5 Windowing

Figur 5.9 viser $x_1[n]$ med og uten multiplikasjon med et hanningvindu (se Figur 3.3).



Figur 5.9: Periodogram av $x_1[n]$ med og uten et hanningvindu.

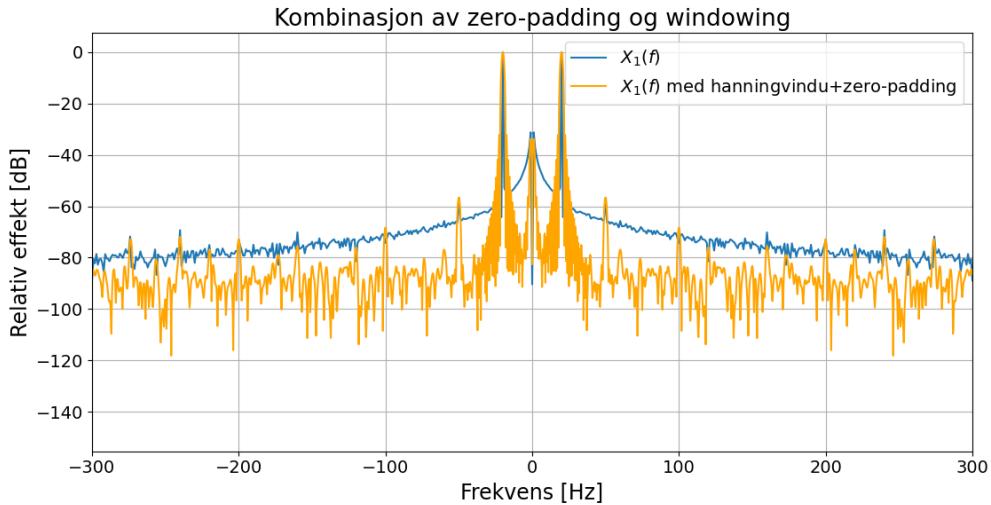
Her ser man at den relative effekten til støygulvet senkes ned til -90 dB også for frekvenser rundt f_1 , slik at flere frekvenser i $x_1[n]$ holdes rundt -90 dB relativt til f_1 . Også merkbart er at hovedlobene rundt f_1 blir større, vist i Figur 5.10. Dette er på grunn av at hanningvinduet har effekten av å redusere spektral lekasje, mens hovedlober sin båndbredde blir større [8].



Figur 5.10: Periodogram av $x_1[n]$ med og uten et hanningvindu rundt frekvensen $f_1 = 20$ Hz. Relativ effekt rundt f_1 reduseres, mens båndbredden til hovedlobene økes.

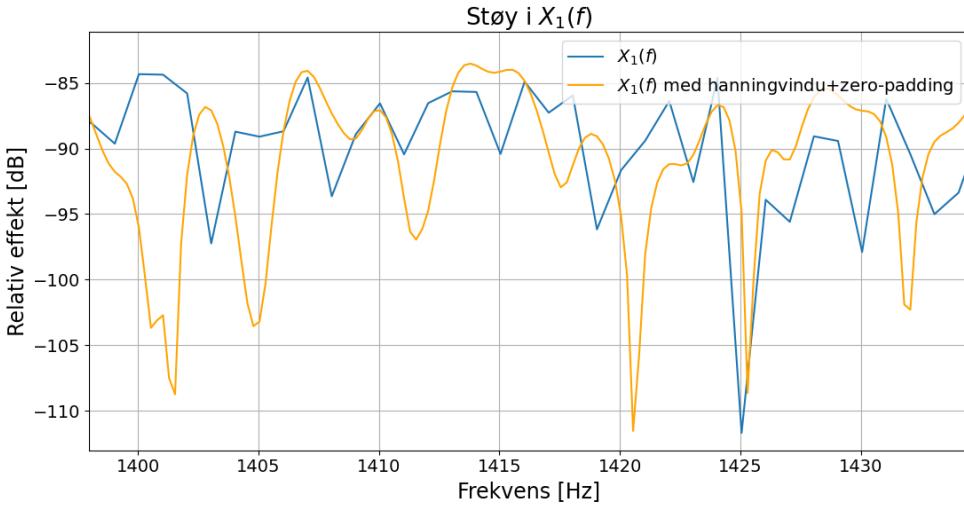
5.6 Kombinering av zero-padding og windowing

Vist i Seksjon 5.4 og Seksjon 5.5 viser zero-padding i større grad spektral lekasje, mens windowing reduserer den relative effekten til disse uønskede frekvenskomponentene. Figur 5.11 viser kombinering av zero-padding og windowing.



Figur 5.11: Periodogram der zero-padding og windowing kombineres. Den oransje grafen er $x_1[n]$ først multiplisert med et hanningvindu, så zero-paddet.

Den oransje grafen er $x_1[n]$ først multiplisert med et hanningvindu, så zero-paddet med samme lengde som i Seksjon 5.4. Igjen dempes frekvenser rundt f_1 ned til -90 dB, samt at resten av støyen også forblir på -90 dB. Merkbart er at med zero-padding får støyen en jevnere form. Dette kan ses ved å studere støyen i signalet, vist i Figur 5.12.



Figur 5.12: Støy fra $X_1(f)$ med hanningvindu og zero-padding. Som man kan se har støyen en jevnere form.

Her ser man at selv om støygulvet holdes rundt samme dB-verdier, vil støyen i $X_1(f)$ med hanningvindu og zero-padding få en jevnere form relativt til $X_1(f)$.

5.7 Kvantisering

Da ADCene som er brukt har en endelig bitdybde på $B = 12$ bits, vil de overnevnte resultatene være påvirket av kvantiseringstøy. Ved å bruke at ADCene har en referansespenning $V_{ref} = 3.3$ V, og jordingsspenning på $V_{ss} = 0$ V, vil vi kunne representere spenninger som ligger $x_{max} - x_{min} = 3.3$ V fra hverandre. De overnevnte verdiene innsatt i ligning (3) og (4) gir henholdsvis oppløsning Δ og varians til feilreddet σ_ϵ^2 . Ved å bruke at det ønskede signalet i $x_1(t)$ er en ren sinus som varierer om 0 V, kan RMS effekten til dette signalet finnes ved å bruke ligning (6). Tallverdier og den resulterende $SNR'_{[dB]}$ -verdien fra ligning (5) er vist i Tabell 5.2.

Tabell 5.2: Tabellen viser numeriske verdier for signalet $x(t)$ sin oppløsning, Δ , effekten til kvantiseringseffelen, P_ϵ , effekten til $x_1(t)$, P_x og den resulterende $SNR'_{[dB]}$ -verdien.

Variabel	Verdi
Δ	$806 \mu\text{V}$
P_ϵ	649nW
P_x	45mW
$SNR'_{[dB]}$	59.2dB

En omtrentlig verdi for $SNR_{[dB]}$ til det realiserte systemet er funnet ved å ta differansen mellom den relative effekten til det ønskede signalet med frekvens $f_1 = 20$ Hz og den relative effekten til støygulvet med zero-padding vist i Figur 5.7. Dette frekvensområdet er valgt ettersom støyen vi er ute etter antas å ligge i dette området. Med signalet $x_1(t)$ gir dette en $SNR_{[dB]} \approx 55$ dB, som er noe dårligere enn det teoretiske SNR en.

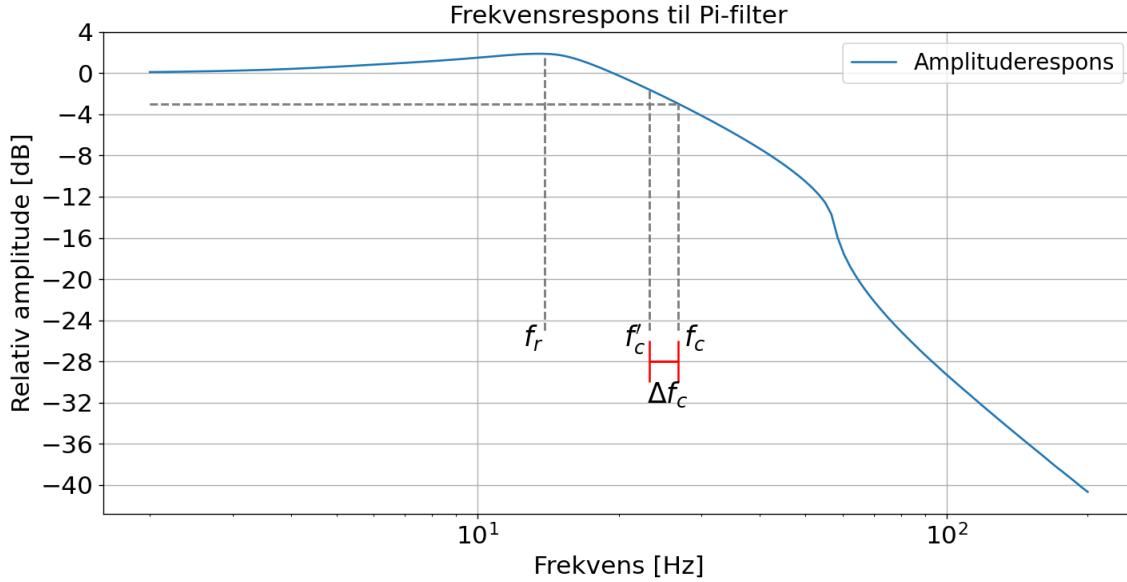
For å også kunne sammenligne med en referanseverdi, er det samme analoge signalet $x_1(t)$ og analysert med oscilloskopet AD2. Ettersom dette oscilloskopet har en bitdybde på $B = 14$, er $SNR_{[dB]}$ forventet høyere i dette tilfellet. Signalet $x_1(t)$ samplet med dette oscilloskopet gir $SNR_{[dB]} = 68.34(24)$ dB med $k = 2$. Dette er vesentlig bedre enn samlingen med en 12 bit bitdybde.

Et viktig poeng er imidlertid at signalet $x_1(t)$ frem til nå bare har tatt verdier i et lite intervall av x_{min} og x_{max} , noe som påvirker $SNR_{[dB]}$ betydelig. Ved å bruke samme ligninger som tidligere,

derimot med maksimal amplitude for $x_1(t)$ vil teoretisk signal til støyforhold bli $SNR'_{[dB]} = 74.1$ dB. Dette er et resultat som underbygger viktigheten av å bruke hele utsvinget vi har tilgjengelig i ADCene.

5.8 Filter og støy

Pi-filteret beskrevet i Seksjon 4.3 har en amplituderespons vist i Figur 5.13.



Figur 5.13: Amplituderesponsen til Pi-filteret. Merk at frekvensaksen er plottet logaritmisk.

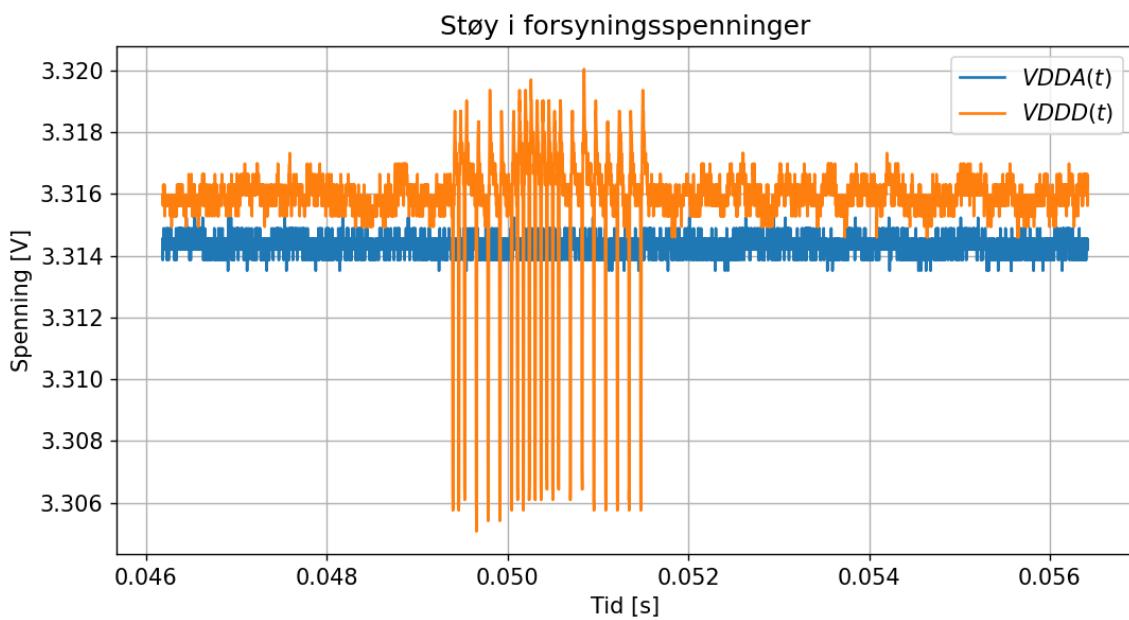
Fra Figur 5.13 kan det observeres at den observerte knekkfrekvensen er $f_c = 26.7$ Hz. Den teoretiske knekkfrekvensen er gitt i Seksjon 4.3 og er $f'_c = 23.2$ Hz. Dette utgjør en forskjell mellom observert og teoretisk knekkfrekvens, Δf_c , lik

$$\Delta f_c = |f_c - f'_c| = 3.5 \text{ Hz}. \quad (17)$$

Noe annet som er verd å merke seg fra Figur 5.13 er resonansen som oppstår mellom spolen og kondensatorene. Resonansfrekvensen har en verdi på $f_r = 13.9$ Hz, hvor frekvenskomponentene her blir forsterket 1.85 dB. Dette er en oppførsel som i utgangspunktet ikke er ønskelig, men er ikke tatt hensyn til i implementert system.

Ettersom formålet med Pi-filteret er å filtrere likespenninger, hvor eventuelle vekselspenninger er uønsket støy, er det ikke relevant å se på faseresponsen.

I tillegg til at Pi-filteret demper støy, vil også avkoblingskondensatorene bidra til demping av støyen. Den totale innvirkningen både Pi-filteret og avkoblingskondensatorene har på den analoge forsyningsspenningen $VDDA(t)$, i forhold til den digitale forsyningsspenningen $VDDD(t)$ er vist i Figur 5.14.



Figur 5.14: De to spenningene $VDDA(t)$ og $VDDD(t)$ under sampling.

Fra Figur 5.14 ser vi forskjellene i forsyningsspenningene under sampling. I tillegg til at den analoge forsyningsspenningen $VDDA(t)$ har mindre lavfrekvente variasjoner, mangler den analoge forsyningen også de store forstyrrelsene i den digitale forsyningen på over 12 mV. Dette indikerer at avkoblingskondensatorene og Pi-filteret fungerer etter sin hensikt.

6 DISKUSJON

6.1 Datainnhenting

Resultatet illustrert i Figur 5.3 viser hvordan det påtrykte signalet $x_1(t)$ gir endinger i det avleste signalet $x_1[n]$. Resultatet viser at signalet $x_1[n]$ i stor grad ligner på det påtrykte signalet $x_1(t)$. Dette gjelder både signaleta amplitude og frekvens. Samsvar mellom det påtrykte signalet og avlest verdi ble også observert på de resterende 4 kanalene som vist i Figur 5.1 og 5.2. At alle disse verdiene er tilnærmet identiske er avgjørende for at systemet skal fungere som det skal, og de diskuterte resultatene tilsier at datainnhentingsenheten sampler sine respektive kanaler som forventet. Videre kan det sees av Figur 5.4 at signalet $x_1[n]$ inneholder noe støy som kan skyldes både kvantifiseringen og/eller annen indusert støy og interferens. SNR på dette signalet er funnet til å være $SNR_{[dB]} \approx 55.0$ dB. Dette stemmer godt overens med den teoretiske verdien $SNR'_{[dB]}$ som er funnet til å være 59.2 dB. Grunnen til at den observerte verdien forventes noe lavere enn den teoretiske, er at den teoretiske verdien bare gir den relative styrken av kvantifiseringsstøyen, mens den observerte verdien også tar med annen støy og interferens.

6.1.1 Samplingsfrekvens

Som nevnt i Seksjon 4.1 ble den teoretiske sampelfrekvensen f'_s funnet til å være $f'_s = 33.6$ kHz. I Seksjon 5.1 ble det derimot funnet en lavere effektiv sampelfrekvens $f_s = 31.19(13)$ Hz med $k = 2$. Hovedgrunnen til dette avviket ligger i at ved start av programmet `adc_sampler.c` legges sampelfrekvensen 31.25 kHz inn som parameter, og programmet vil derfor sikte seg inn på denne sampelfrekvensen, heller enn den teoretisk maksimale samplingsfrekvensen. Ytterligere utvikling av systemet kunne basert seg på å prøve å øke den effektive samplingsfrekvensen f_s ytterligere opp mot den teoretisk maksimale sampelfrekvensen f'_s for den gitte klokkefrekvensen f_{CLK} . I et slikt scenario kan det argumenteres for at hvor nærmere den effektive sampelfrekvensen f_s kommer f'_s , vil blant annet være gitt av hvor raskt prosesseringsenheten klarer å reagere etter kommunikasjon av en sampel. Dette innebærer for eksempel at signalet CS må dras til logisk lavt nivå igjen etter akkurat minimumsverdien til T_{CSH} nevnt i [17]. Dersom det ikke kreves en klokkefrekvens $f_{CLK} = 500$ kHz, vil det være betydelig enklere å heller øke klokkefrekvensen f_{CLK} for å øke sampelfrekvensen f_s . ADCen MCP3201 har for eksempel en maksimal klokkefrekvens på 1.6 MHz ved en forsyningsspenning på 5V [17].

6.2 Signalbehandling

Generelle resultater for $x_1[n]$ i frekvensdomenet er som forventet, der zero-padding viser effekten av spektral lekasje og windowing reduserer den relative effekten til uønskede frekvenskomponenter, noe som er i samsvar med teoretisk oppførsel fra Seksjon 3.3.3 og Seksjon 3.3.4. Beskrevet i Seksjon 3.3.3 vil zero-padding gjøre oppløsningen til frekvensspekteret best mulig for en viss tid FFT brukes på å utregnes. Signalet $x_1[n]$ i Seksjon 5.4 er zero-padded med betydelig flere sampler en ligning (12) tilsier, og er valgt for å tydelig vise spektral lekasje. Tiden FFT brukes på å utregnes er ikke tatt hensyn til.

Kombinasjon av zero-padding og windowing viser både spektral lekasje, samtidig som sideløber sin relative effekt reduseres til -90 dB, også for frekvenser rundt f_1 . En negativ konsekvens er at båndbredden til hovedloben rundt f_1 blir større, og er på grunn av at et hanningvindu er benyttet. Dersom dette ikke er ønskelig for dataen som samples, burde videre arbeid undersøke hvordan andre vindusfunksjoner påvirker frekvensspekteret til $x_1[n]$. Som nevnt i Seksjon 3.3.4 har alle vindusfunksjoner sine fordeler og ulemper.

6.3 Filter

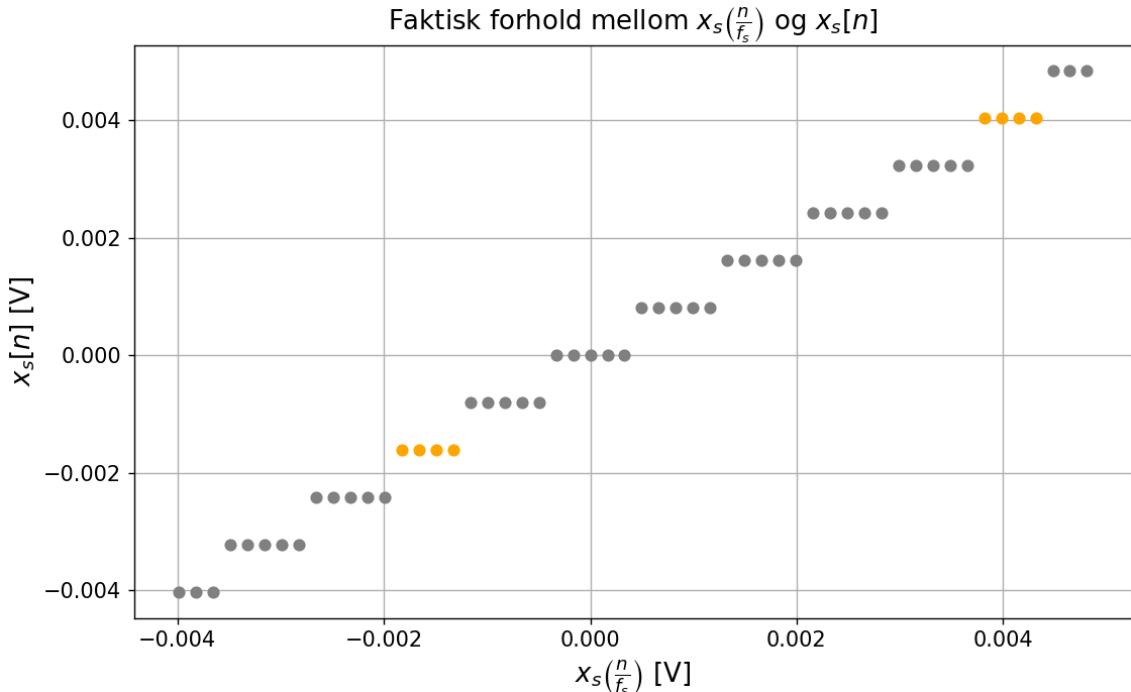
Det implementerte Pi-filteret har et avvik på $\Delta f_c = 3.5$ Hz mellom den teoretiske og observerte knekkfrekvensen. Et avvik i denne størrelsesordenen er imidlertid forventet da ligning (13) er en tilnærming. Dersom en mer nøyaktiv verdi er nødvendig, vil simulering være et bedre alternativ. Ytterligere demping kan også oppnås om ønskelig, for eksempel ved å implementere flere identiske filtrere i kaskade.

6.4 Feilkilder

Frem til nå er det blitt antatt at det analoge signalet på kanal s , $x_s(t)$, er et kontinuerlig signal. Dette er imidlertid ikke tilfellet og kommer av at signalet er generert av signalgeneratoren på AD2, med en Digital til Analog Converter (DAC) [19]. Signalgeneratoren justerer imidlertid oppløsningen til signalet den skal generere, avhengig av amplituden til signalet. I tilfellet med et signal $x_s(t)$ med amplitude $A = 0.3$ V, gir AD2 en oppløsning på $\Delta_{AD2} = 166 \mu\text{V}$, mens oppløsningen til ADCene er $\Delta = 806 \mu\text{V}$ [19]. Dette gir forholdet mellom oppløsningene

$$\frac{\Delta}{\Delta_{AD2}} = 4.855 \notin \mathbb{N}. \quad (18)$$

Ideelt sett burde dette forholdet vært uendelig stort, slik at det påtrykte signalet hadde vært kontinuerlig. Et annet aspekt verd å merke seg, er at forholdet $\frac{\Delta}{\Delta_{AD2}}$ ikke er et heltall. Dette indikerer at det ikke er en direkte korrespondanse mellom alle spenningsnivåer fra DAC fra oscilloskopet til ADCene i det implementerte systemet, som er tilfellet når $x_s(t)$ er kontinuerlig. Se for eksempel Figur 3.1 i Seksjon 3.1. En simulering av effekten dette har er vist i Figur 6.1.



Figur 6.1: Figuren viser en simulering av forholdet mellom signalene når det benyttes en DAC i stedet for et faktisk kontinuerlig signal. Kvantifiseringsnivåer med lengde som avviker fra de andre er markert med oransje. Merk at den tilsvarende figuren for kontinuerlige verdier er Figur 3.1.

Ettersom oppløsningen til oscilloskopet, Δ_{AD2} , er såpass mye høyere enn oppløsningen til ADCene, Δ , vil sammenhengen mellom signalene uansett bli tilnærmet lineær, så denne effekten er antatt å

ha liten innvirkning på resultatene. Det er imidlertid et viktig fenomen å være klar over. En kuriositet er parallelten til Nyquist samplingsteorem, hvor ”kvantiserinsaliasing” kan oppstå ettersom ADCen ikke vil være i stand til å rekonstruere alle unike signalnivåer DACen produserer.

En annen viktig feilkilde er at all støyen i kretsen ikke stammer fra kvantifiseringen av signalet, som er antatt i utregning av $SNR_{[dB]}$. Annen støy og interferens, for eksempel høyfrekvent støy indusert fra den digitale delen av kretsen, er forventet å bidra minst like mye til støyen som det kvantifiseringsfeilen gjør.

7 KONKLUSJON

Det realiserte systemet bruker 5 ADCer for å sample og kvantifisere 5 analoge signaler $x_1(t)$ - $x_5(t)$, til digitale signaler $x_1[n]$ - $x_5[n]$. Ved observasjon har de digitaliserte signalene $x_1[n]$ - $x_5[n]$ tilnærmet samme amplitude og frekvens som de analoge signalene $x_1(t)$ - $x_5(t)$. Likhetsgraden mellom de ulike analoge og digitale signalene, $S_{[dB]}$, bekrefter også dette. Den gjennomsnittlige likhetsgraden er funnet til å være -2.9389 dB. Ved nøyere analyse på signalet $x_1(t)$ blir mer støy observert på det digitaliserte signalet $x_1[n]$, enn på det analoge signalet $x_1(t)$. Den teoretiske verdien for signal til støyforholdet på grunn av kvantifisering er $SNR'_{[dB]} = 59.2$ dB. Det observerte signal til støyforholdet er $SNR_{[dB]} \approx 55$ dB, noe som var forventet.

Signalbehandling utført på $x_1[n]$ gir oppførsel i samsvar med teori fra Seksjon 3.3, der zero-padding viser spektral lekasje og windowing reduserer den relative effekten til uønskede frekvenskomponenter. Generelt sett påvirker windowing og zero-padding den relative effekten til frekvenser rundt f_1 , mens støygulvet holdes på -90 dB for høyere frekvenser. Kombinering av zero-padding og windowing senker den relative effekten til frekvenser rundt f_1 ned til -90 dB, samtidig som støygulvet får en jevnere form.

Det implementerte Pi-filteret har en knekkfrekvens $f_c = 26.7$ Hz, som gir et avvik på $\Delta f_c = 3.5$ Hz fra den teoretiske f'_c . Filteret bidrar til at systemet fungerer som forventet ved å filtrere bort høye frekvenskomponenter. Figur 5.13 bekrefter dette.

KILDER

- [1] Dimitri G. Mankolis John G. Proakis. *Digital Signal Processing*. Pearson, 2013.
- [2] Lise Lyngnes Randaberg Peter Svensson Egil Eide. *Kompendie TTT4280 Sensorer og instrumentering*. NTNU, 2024.
- [3] Cleaning Technologies Group. *A Little About RMS (Root Mean Squared)*. 2024. URL: <https://techblog.ctclean.com/2015/07/little-rms-root-mean-square/> (sjekket 15.02.2024).
- [4] G. Stette. *Aliasing*. URL: <https://snl.no/aliasing> (sjekket 05.02.2024).
- [5] S.W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. URL: <https://www.dspproject.com/ch12/4.htm> (sjekket 05.02.2024).
- [6] VRU. *The FFT and Digital Sampling*. URL: <https://vru.vibrationresearch.com/lesson/the-fft-and-digital-sampling/> (sjekket 16.02.2024).
- [7] S. Hilbert. *FFT Zero Padding*. URL: <https://www.bitweenie.com/listings/fft-zero-padding/> (sjekket 13.02.2024).
- [8] Wikipedia. *Window function*. URL: https://en.wikipedia.org/wiki/Window_function (sjekket 05.02.2024).
- [9] Arrow. *Grounding Principles for Mixed-Signal Designs*. URL: <https://www.arrow.com/en/research-and-events/articles/principles-of-grounding-for-mixed-signal-designs> (sjekket 07.02.2024).
- [10] Cadence PCB solutions. *Passive Pi Filter Design and Simulation*. URL: <https://resourcespcb.cadence.com/blog/2020-passive-pi-filter-design-and-simulation> (sjekket 06.02.2024).
- [11] Raspberry Pi. *Raspberry Pi 3 Model B*. URL: <https://www.raspberrypi.com/products/raspberry-pi-3-model-b/> (sjekket 13.02.2024).
- [12] Sparkfun. *SparkFun Pi Wedge*. URL: <https://www.sparkfun.com/products/13717> (sjekket 13.02.2024).
- [13] Analog Devices. *Introduction to SPI Interface*. URL: <https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html> (sjekket 04.02.2024).
- [14] Raspberry Pi. *Raspberry Pi documentation*. URL: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html> (sjekket 07.02.2024).
- [15] DAV university. *DMA (Direct Memory Access)*. URL: <https://www.davuniversity.org/images/files/study-material/DMA%20Controller.pdf> (sjekket 04.02.2024).
- [16] Joan Abyz. *pigpio*. 2024. URL: <https://github.com/joan2937/pigpio>.
- [17] Microchip. *2.7V 12-Bit A/D Converter with SPI Serial Interface*. URL: <https://ww1.microchip.com/downloads/en/devicedoc/21290f.pdf>.
- [18] WinSCP. *Introducing WinSCP*. 2024. URL: <https://winscp.net/eng/docs/introduction> (sjekket 24.01.2024).
- [19] Digilent. *Analog Discovery 2 Specifications*. URL: <https://digilent.com/reference/test-and-measurement/analog-discovery-2/specifications> (sjekket 06.02.2024).
- [20] SciPy. *Signal processing (scipy.signal)*. 2024. URL: <https://docs.scipy.org/doc/scipy/reference/signal.html> (sjekket 14.01.2024).
- [21] Scipy. *Fourier Transforms (scipy.fft)*. URL: <https://docs.scipy.org/doc/scipy/tutorial/fft.html> (sjekket 13.02.2024).
- [22] Wikipedia. *Noise floor*. URL: https://en.wikipedia.org/wiki/Noise_floor (sjekket 06.02.2024).

APPENDIX

A Github

URL: <https://github.com/heimsnik/Sensorer-og-instrumentering---Lab/tree/master/Lab%201>

B frekvensspektrum_generator.py

```
# Importerer pakker
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, ifft
import sys
import os
import csv
import scipy.signal as ss
import time

def raspi_import(path, channels=5):

    with open(path, 'r') as fid:
        sample_period = np.fromfile(fid, count=1, dtype=float)[0]
        data = np.fromfile(fid, dtype='uint16').astype('float64')
        # The "dangling" `astype('float64')` casts data to double precision
        # Stops noisy autocorrelation due to overflow
        data = data.reshape((-1, channels))

    # sample period is given in microseconds, so this changes units to seconds
    sample_period *= 1e-6
    return sample_period, data #The data-array generates a (31250, 5) array, meaning 5
                               channels

# Import data from bin file
if __name__ == "__main__":
    sample_period, data = raspi_import('C:/Users/bruker/OneDrive - NTNU/6.
                                         semester/TTT4280 Sensorer og
                                         instrumentering/Lab/Sensorer-og-instrumentering---Lab/Lab
                                         1/Data/sampledData_163208.bin')
    dt = sample_period
    data = (data*3.308)/(2**12)

    data = ss.detrend(data, axis=0)

def analog_signal():

    analog_data = []

    filepath1 = r'C:/Users/bruker/OneDrive - NTNU/6. semester/TTT4280 Sensorer og
                instrumentering/Lab/Sensorer-og-instrumentering---Lab/Lab
                1/Data/scope_CH1_new/CH1_new.csv'

    # Read data from the specified filepath
    with open(filepath1) as csvfile:
        csvreader = csv.reader(csvfile)
        header = next(csvreader)
        for datapoint in csvreader:
            values = [float(value) for value in datapoint]
```

```

analog_data.append(values)

time1 = [p[0] for p in analog_data]
ch1 = [p[1] for p in analog_data]
#ch1 = ss.detrend(ch1, axis=0)
ch2 = [p[1]+0.6*1 for p in analog_data]
ch3 = [p[1]+0.6*2 for p in analog_data]
ch4 = [p[1]+0.6*3 for p in analog_data]
ch5 = [p[1]+0.6*4 for p in analog_data]

plt.plot(time1,ch1, label=f'$x_{1}(t)$', color = 'blue', linestyle='--')
plt.plot(time1,ch2, label = f'$x_{2}(t)$', color = 'red', linestyle='--')
plt.plot(time1,ch3, label=f'$x_{3}(t)$', color = 'purple', linestyle='--')
plt.plot(time1,ch4, label =f'$x_{4}(t)$', color = 'forestgreen', linestyle='--')
plt.plot(time1,ch5, label =f'$x_{5}(t)$', color = 'cyan', linestyle='--')

plt.grid()
plt.xlim(0, 0.1)
plt.title(f'Analoge inngangssignaler til alle ADCer', fontsize = 19)
plt.xlabel('Tid [s]', fontsize=17)
plt.ylabel('Spennin [V]', fontsize=17)
plt.legend(loc='upper right', fontsize=14)

plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

plt.show()

return time1, ch1

def plot_data(data_plot):
    #Tidsakse for samplede signaler
    t = np.arange(0, dt*(len(data_plot[:, 0])-1), dt)

    colours = ['blue', 'red', 'purple', 'forestgreen', 'cyan']

    for i in range(len(data_plot[0, :])):
        if i == 0:
            data_plot[:, i:(i+1)] += 0
        else:
            data_plot[:, i:(i+1)] += 0.6*i

    plt.plot(t, data_plot[:, 0], label=f'$x_{1}[n]$', color = colours[0])
    plt.plot(t, data_plot[:, 1], label=f'$x_{2}[n]$', color = colours[1])
    plt.plot(t, data_plot[:, 2], label=f'$x_{3}[n]$', color = colours[2])
    plt.plot(t, data_plot[:, 3], label=f'$x_{4}[n]$', color = colours[3])
    plt.plot(t, data_plot[:, 4], label=f'$x_{5}[n]$', color = colours[4])

    # time_analog, ch1 = analog_signal()
    # plt.plot(time_analog,ch1, label=f'$x_{1}(t)$', color = 'blue', linestyle='--')

    plt.xlabel("Tid [s]", fontsize=17)
    plt.ylabel("Spennin [V]", fontsize=17)
    plt.title(f'Analogn og digitalt signal for $x_{1}(t)$', fontsize = 19)
    plt.legend(loc="upper right", fontsize=15)
    plt.xlim(0,0.1)
    plt.xticks(fontsize=14)
    plt.yticks(fontsize=14)
    plt.grid()
    plt.show()

def zero_pad(data_pad):

```

```

n = 3*(len(data_pad)-1)

data_pad_copy = data_pad.copy()

data_transposed = np.transpose(data_pad_copy)

# Pad zeros to each row
padded_rows = [np.concatenate((row, np.zeros(n))) for row in data_transposed]

# Transpose the padded rows back to the original shape
padded_data = np.transpose(padded_rows)

return padded_data


def window(data_window):
    #Bartlett, blackman, hamming, kaiser
    hanning = np.hanning(len(data_window[:, 0]))

    data_window_copy = data_window.copy()

    for i in range(len(data_window_copy[:, 0])):
        data_window_copy[i:i+1, :] *= hanning[i]

    return data_window_copy


#FFT for alle signaler
def FFT(data_FFT):

    data_FFT_copy = data_FFT.copy()

    FFT_ADC1 = np.fft.fftshift(np.fft.fft(data_FFT_copy[1:, 0], len(data_FFT_copy[1:, 0])))
    FFT_ADC2 = np.fft.fftshift(np.fft.fft(data_FFT_copy[1:, 1], len(data_FFT_copy[1:, 0])))
    FFT_ADC3 = np.fft.fftshift(np.fft.fft(data_FFT_copy[1:, 2], len(data_FFT_copy[1:, 0])))
    FFT_ADC4 = np.fft.fftshift(np.fft.fft(data_FFT_copy[1:, 3], len(data_FFT_copy[1:, 0])))
    FFT_ADC5 = np.fft.fftshift(np.fft.fft(data_FFT_copy[1:, 4], len(data_FFT_copy[1:, 0])))

    #Frekvensakse
    freq = np.fft.fftshift(np.fft.fftfreq(n=len(FFT_ADC1), d=sample_period))

    return FFT_ADC1, FFT_ADC2, FFT_ADC3, FFT_ADC4, FFT_ADC5, freq


def plot_FFT(data, data_window, data_padded):
    FFT_ADC1, FFT_ADC2, FFT_ADC3, FFT_ADC4, FFT_ADC5, freq = FFT(data)
    FFT_ADC1_window, FFT_ADC2_window, FFT_ADC3_window, FFT_ADC4_window, FFT_ADC5_window,
    freq_window = FFT(data_window)
    FFT_ADC1_padded, FFT_ADC2_padded, FFT_ADC3_padded, FFT_ADC4_padded, FFT_ADC5_padded,
    freq_padded = FFT(data_padded)

    plt.xlabel("Frekvens [Hz]", fontsize=15)
    plt.ylabel("Relativ effekt [dB]", fontsize=15)
    plt.title("Periodogram av $x_{1:n}$", fontsize=17)
    plt.plot(freq, abs(FFT_ADC1)/max(abs(FFT_ADC1)))
    #plt.plot(freq_window, abs(FFT_ADC1_window)//max(abs(FFT_ADC1_window)), color = 'g')
    #plt.plot(freq_padded, abs(FFT_ADC1_padded)/max(abs(FFT_ADC1_padded)), color = 'r')

```

```

plt.xlim(-100,100)
#plt.ylim(-140, 5)
plt.legend([f'$X_{i}(f)$'], loc="upper right", fontsize=15)
plt.grid()
plt.show()

def plot_periodogram(data_original, data_window, data_padded):
    FFT_ADC1, FFT_ADC2, FFT_ADC3, FFT_ADC4, FFT_ADC5, freq = FFT(data_original)
    FFT_ADC1_window, FFT_ADC2_window, FFT_ADC3_window, FFT_ADC4_window, FFT_ADC5_window,
    freq_window = FFT(data_window)
    FFT_ADC1_padded, FFT_ADC2_padded, FFT_ADC3_padded, FFT_ADC4_padded, FFT_ADC5_padded,
    freq_padded = FFT(data_padded)

    plt.xlabel("Frekvens [Hz]", fontsize=17)
    plt.ylabel("Relativ effekt [dB]", fontsize=17)
    plt.title(f"Sty i $X_{i}(f)$", fontsize=19)
    plt.plot(freq, 20*np.log10((np.abs(FFT_ADC1)/max(abs(FFT_ADC1)))), label =
              f'$X_{i}(f)$')
    #plt.plot(freq_window,
    #          20*np.log10((np.abs(FFT_ADC1_window)/max(abs(FFT_ADC1_window)))), label =
    #          f'$X_{i}(f)$ med hanningvindu', color = 'g')
    plt.plot(freq_padded,
              20*np.log10((np.abs(FFT_ADC1_padded)/max(abs(FFT_ADC1_padded)))), color =
              'orange', label = f'$X_{i}(f)$ med hanningvindu+zero-padding')
    #plt.xlim(-300,300)
    #plt.ylim(-130, 5)
    plt.xticks(size = 14)
    plt.yticks(size = 14)
    plt.legend(loc="upper right", fontsize=14)
    plt.grid()
    plt.show()

#The original data zero-padded
data_padded = zero_pad(data)

#The original data windowed
data_window = window(data)

#The windowed data zero-padded
data_window_padded = zero_pad(data_window)

#The padded data windowed
data_padded_window = window(data_padded)

#plot_FFT(data, data_window, data_padded)
#plot_periodogram(data, data_window, data_window_padded)
#plot_data(data)

```
