NTNU

elsys
ELEKTRONISK SYSTEMDESIGN & INNOVASJON

TTT4280 – Labrapport

Tittel: Lab 2: Mikrofonarray for å finne retning til en lydkilde

Skrevet av: Truls Østvedt, Arya Raeesi

Gruppe: 18

Dato: 6. mai 2026

Sammendrag

Denne rapporten dokumenterer utviklingen og testingen av et system for estimering av lydens innfallsvinkel ved hjelp av et mikrofonarray. Målet har vært å vurdere hvor nøyaktig man kan bestemme retningen til en lydkilde basert på tidsforskjeller mellom tre omnidireksjonelle mikrofoner plassert i en likesidet trekant. Systemet består av mikrofoner koblet til ADC-er, som igjen sender digitaliserte signaler til en Raspberry Pi for videre prosessering og analyse.

Estimeringene ble gjort ved hjelp av krysskorrelasjon, og vinklene ble beregnet basert på relative tidsforsinkelser mellom mikrofonene. Gjennom systematisk testing ved ulike innfallsvinkler og under varierende forhold ble nøyaktighet og robusthet vurdert. Resultatene viser at systemet presterer godt ved enkelte vinkler, med gjennomsnittlige avvik på under 2° og lavt standardavvik. Imidlertid oppstår det alvorlige avvik ved enkelte retninger, spesielt ved 270° og 300°, hvor både skjevhet og varians er markant høyere.

Mulige årsaker til avvikene inkluderer reflektert lyd i rommet, lav vinkeloppløsning grunnet begrenset samplingfrekvens og mikrofonavstand, samt feil eller variasjoner i en av mikrofonene. Rapporten diskuterer også forbedringstiltak, inkludert kalibrering, bytte av mikrofon, og alternative målemiljøer. Systemets begrensninger og forslag til videre arbeid presenteres i siste del av rapporten.

Innhold

Liste over forkortelser

Denne rapporten bruker forkortelser til flere ord. I Tabell 1 er disse forkortelsene satt sammen med deres fulle ord.

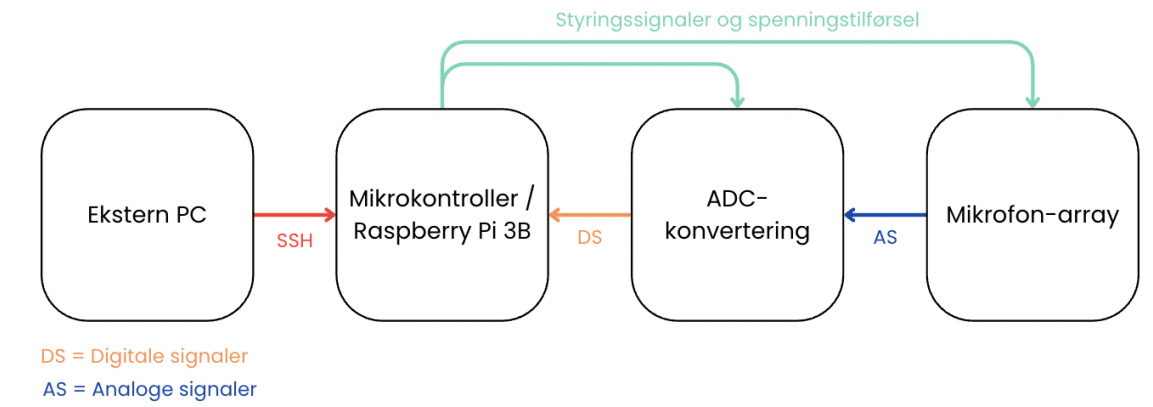
Tabell 1: Oversikt over forkortelser brukt i rapporten.

Forkortelse	Beskrivelse
ADC	Analog-til-digital konverter
Mic1	Mikrofon nummer 1
Mic2	Mikrofon nummer 2
Mic3	Mikrofon nummer 3
SSH	Secure Shell
SFTP	Secure File Transfer Protocol
DMA	Direct Memory Access
SNR	Signal-til-støy-forhold
CS	Chip Select
CLK	Klokkesignal
GND	Ground/Jord
SPI	Serial Peripheral Interface
MISO	Master In, Slave Out
VDD	Forsyningsspenning
VREF	Referansespenning
DC	Likestrøm

1 Innledning

Et mikrofonarray består av flere mikrofoner i bestemte posisjoner, og har en rekke bruksområder innen blant annet signalbehandling og akustikk. Ved å analysere forskjeller i ankomsttid mellom mikrofonene, kan retningen til lydkilden bestemmes. Dette resulterer i et verktøy som kan hente ut romlig informasjon av lydkilder og lydsignaler.

I dette prosjektet skal vi utvikle og teste et system som estimerer retningen til en lydkilde i planet, basert på signaler fra et array av tre små omnidireksjonelle mikrofoner. Systemet er implementert med blant annet en Raspberry Pi 3B, ADC-er og mikrofoner, der det legges høy vekt i presisjon i tid og synkronisering for nøyaktige målinger. Figur 1 viser et forenklet blokkskjema av systemet.



Figur 1: Forenklet blokkskjema av systemet. Ekstern PC kommuniserer med mikrokontrolleren (Raspberry Pi 3B) via SSH. Mikrokontrolleren styrer ADC-enhetene og mottar digitaliserte målesignaler. Mikrofon-arrayet mottar lydsignaler og sender analoge signaler til ADC-enhetene. Styringssignaler og spenningsforsyning distribueres fra mikrokontrolleren til ADC-modulen og mikrofon-arrayet.

2 Teoretisk grunnlag for målesystemet

Denne seksjonen gir en teoretisk oversikt over prinsippene som ligger til grunn for systemet beskrevet i Seksjon 1. Den dekker sentrale temaer som krysskorrelasjon, innfallsvinkel-beregning og sampling og oppløsning, samt analog-til-digital konvertering, signalbehandling og støykontroll, som sammen utgjør fundamentet for metodene som benyttes i Seksjon 3.

2.1 Mikrofoner og mikrofonarray

Et mikrofonarray består av flere mikrofoner plassert på kjente posisjoner, og brukes til å estimere retningen lydbølger kommer fra. I dette tilfellet benyttes omnidireksjonelle mikrofoner, som fanger opp lyd likt fra alle retninger [1]. Dette er hensiktsmessig fordi vi ønsker å måle forskjeller i ankomsttid.

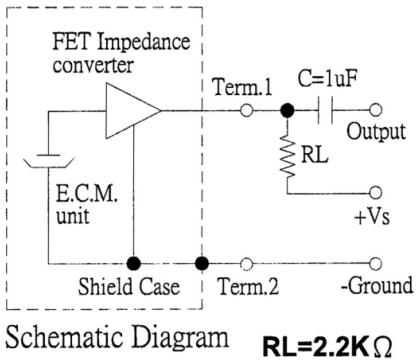
Ved å analysere tidsforsinkelsene mellom signalene som registreres i hver mikrofon, kan vi beregne innfallsvinkelen til lydkilden. Dette vil gjennomgås i resten av seksjonene i 'Teoretisk grunnlag for målesystemet'.

2.1.1 CMA-4544PF-W mikrofon-spesifikasjoner

I dette prosjektet benyttes CMA-4544PF-W, en elektret-kondensatormikrofon med omnidireksjonell karakteristikk. Mikrofonen har en sensitivitet på $-44 \pm 2\,\text{dB}$ ved 1 kHz og en inngang på 1 Pa, med et frekvensområde på 20 Hz til 20 kHz. Dette gjør den velegnet til generell lydinnsamling.

Driftsspenningen er nominelt 3 V (maksimalt 10 V), og utgangsimpedansen er 2,2 kΩ. Strømforbruket er maksimalt 0,5 mA. Signal-støy-forholdet (SNR) er oppgitt til 60 dBA, noe som gir tilstrekkelig klarhet i signalene for bruk i tidsbasert analyse som krysskorrelasjon.

Et skjematisk diagram av mikrofonen er i Figur 2.



Figur 2: Skjematisk diagram av CMA-4544PF-W mikrofon [2].

Til slutt finner du databladet til CMA-4544PF-W her.

2.2 Krysskorrelasjon og anvendelse for mikrofonarray

Krysskorrelasjon er en matematisk metode som kan brukes til å måle likheten mellom to signaler som funksjon av tidsforskyvning mellom dem. For to diskrete signaler $x[n]$ og $y[n]$, er krysskorrelasjonen deres definert i Likning 1.

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x[n] \cdot y[n+l], \quad l = 0, \pm 1, \pm 2, \pm 3, ... \tag{1}$$

Her representerer l antall samples signalet $y[n]$ forskyves med i tid. Et høyt korrelasjonsresultat ved en bestemt l betyr at $x[n]$ og $y[n]$ er svært like ved den tidsforskyvningen. Denne tidsforskyvningen kan beregnes med Likning 2, der f_s er samplingfrekvensen.

$$\Delta t = \frac{l}{f_s} \tag{2}$$

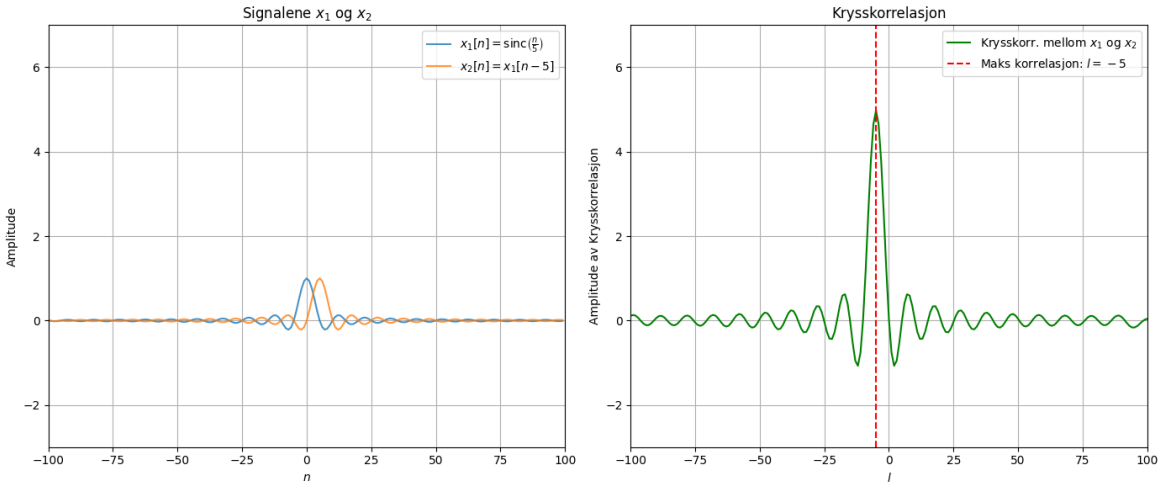
Dette gjør krysskorrelasjon til et effektivt verktøy for å måle tidsforskjellen mellom to signaler.

For et mikrofonarray kan krysskorrelasjon brukes for å finne tidsforsinkelsene mellom to og to mikrofoner. For et mikrofonpar j og i er tidsforsinkelsen definert med Likning 3.

$$\tau_{ji} = \frac{n_{ji}}{f_s} \tag{3}$$

Merk at Likning 3 bygger på samme prinsipp som Likning 2, men er spesifikt formulert for tidsforsinkelse mellom to mikrofoner.

Figur 3 viser en illustrasjon av krysskorrelasjonen mellom to sinc-funksjoner, der den andre funksjonen har en forskyvning.



Figur 3: Teoretisk krysskorrelasjon mellom $x_1[n] = \text{sinc}(\frac{n}{5})$ og $x_2[n] = x_1[n-5]$.

I figuren benyttes en diskret sinc-funksjon, definert som $x_1[n] = \text{sinc}(\frac{n}{5}) = \frac{\sin(\frac{n\pi}{5})}{\frac{n\pi}{5}}$, mens det forskjøvne signalet er gitt som $x_2[n] = x_1[n-5] = \frac{\sin(\frac{(n-5)\pi}{5})}{\frac{(n-5)\pi}{5}}$. Krysskorrelasjonen mellom de to signalene blir da som vist i Likning 4.

$$r_{x_1x_2}(l) = \sum_{n=-\infty}^{\infty} x_1[n] \cdot x_1[n-5+l], \quad l = 0, \pm1, \pm2, \pm3, ... \tag{4}$$

I Figur 3 kan en se at den maksimale korrelasjonen skjer ved $l = -5$, dermed vil tidsforsinkelsen bli $\tau_{x_1x_2} = \frac{-5}{f_s}$, som kan brukes videre for utregning av innfallsvinkel (se Seksjon 2.3).

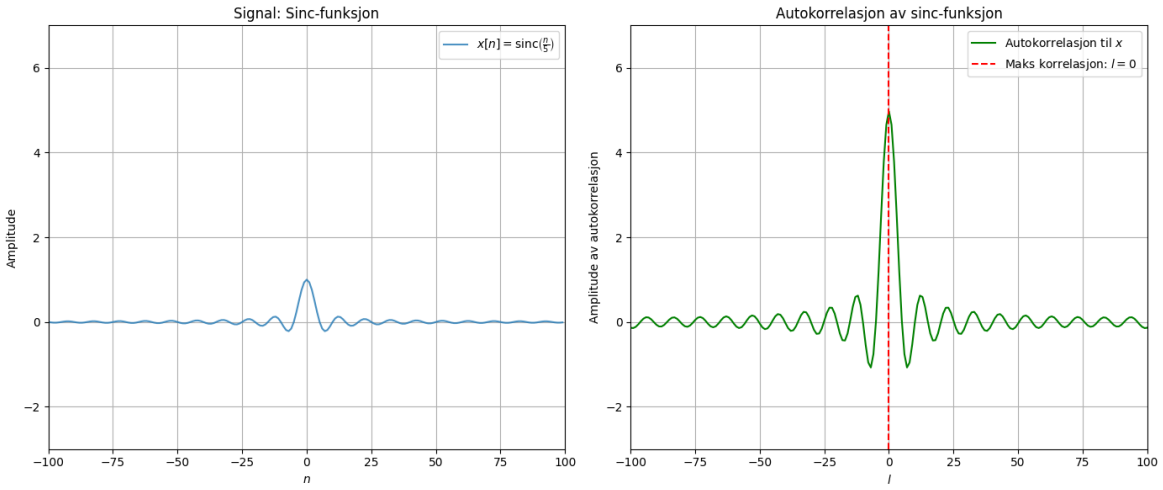
2.2.1 Spesialtilfelle av krysskorrelasjon

Autokorrelasjon er et spesialtilfelle av krysskorrelasjon der et signal sammenlignes med seg selv. Dette gir et nyttig verktøy for å undersøke signalets struktur og periode, og fungerer som en god sanity check i systemer som benytter krysskorrelasjon. Formelt kan autokorrelasjonen $r_{xx}(l)$ defineres som:

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x[n] \cdot x[n+l], \quad l = 0, \pm1, \pm2, \pm3, ... \tag{5}$$

Den maksimale verdien av autokorrelasjonen oppstår alltid ved $l = 0$, siden signalet da har perfekt overlapping med seg selv. Tilstedeværelsen av klare topper ved andre lagverdier kan indikere periodisitet i signalet.

Figur 4 viser et eksempel på autokorrelasjon av signalet $x[n] = \text{sinc}(\frac{n}{5})$.

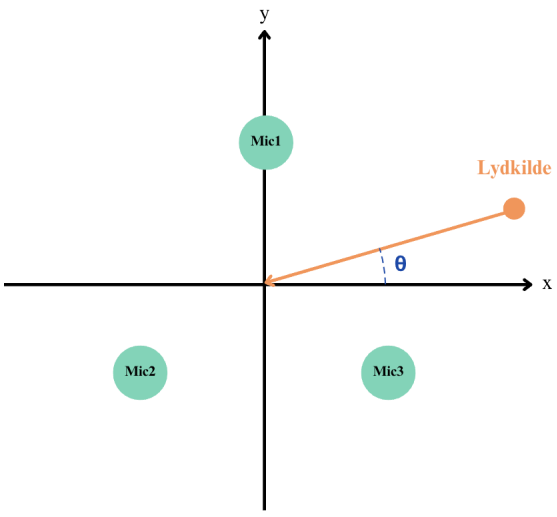


Figur 4: Teoretisk autokorrelasjon til $x[n] = \text{sinc}(\frac{n}{5})$.

Her observeres en tydelig topp ved $l = 0$, som forventet, og mindre symmetriske topper for øvrige lag, noe som reflekterer strukturen i det opprinnelige signalet.

2.3 Beregning av innfallsvinkel

Med Seksjon 2.2 som bakgrunn, kan innfallsvinkelen til en lydbølge beregnes. Med tre mikrofoner i en likesidet trekant, som i oppsettet i Figur 5, kan innfallsvinkelen til lydbølgen beregnes med Likning 6 [3].



Figur 5: θ i forhold til Mic1, Mic2 og Mic3 i xy-planet.

$$\theta = \arctan(\sqrt{3}\frac{\tau_{21} + \tau_{31}}{\tau_{21} - \tau_{31} - 2\tau_{32}}) \tag{6}$$

Der τ_{21}, τ_{31} og τ_{32} er tidsforsinkelsene til hvert mikrofonpar i en mikrofonarray som inneholder tre mikrofoner. Likning 6 forutsetter at lydkilden er langt unna (planbølge-antagelse) og at mikrofonene er plassert i en ideell likesidet trekant [3, 4].

Kort sagt, Likning 6 tillater oss å bestemme retningen til lydkilden.

2.4 Sampling og oppløsning

Når vi bruker krysskorrelasjon for å bestemme retningen til en lydkilde, er oppløsningen begrenset av både samplingfrekvensen (f_s) og avstanden mellom mikrofonene. Dette påvirker hvilke vinkler vi kan faktisk detektere.

I praksis måler vi tidsforsinkelser mellom mikrofonpar i antall samples. Siden samplingsfrekvensen er begrenset, vil vi kun kunne registrere tidsforsinkelser som er heltallige multiplum av $\frac{1}{f_s}$. Dette betyr at vi kan ikke skille mellom uendelig mange vinkler, men bare mellom et visst antall. Dette er avhengig av hvor mange ulike tidsforsinkelser som kan oppstå gitt mikrofonavstanden og samplingfrekvensen. Med andre ord, jo færre mulige tidsforsinkelser, jo grovere vinkeloppløsning.

Med tanke på dette, er best praksis å velge de 'beste' kombinasjonene av avstand mellom mikrofonene og samplingfrekvens, for flest mulige forsinkelser. Dette vil føre til høyest mulig vinkeloppløsning.

Til slutt, den maksimale forsinkelsen mellom to mikrofoner oppstår når lyden kommer rett mot én mikrofon og må bevege seg hele avstanden for å nå den andre. I antall samples vil det tilsi Likning 7 [5],

$$n_{maks} = \lfloor \frac{d \cdot f_s}{v_{lyd}} \rfloor \tag{7}$$

der d er avstanden mellom mikrofonene i meter, f_s er samplingfrekvensen, og $v_{lyd} = 343\text{m/s}$, altså lydens hastighet i luft. Et eksempel av dette er gitt i Likning 8.

$$n_{maks} = \lfloor \frac{d \cdot f_s}{v_{lyd}} \rfloor = \lfloor \frac{0.1 \cdot 16000}{343} \rfloor = \lfloor 4.7 \rfloor = 4 \Longrightarrow 4 \text{ samples} \tag{8}$$

Dette betyr at bare 9 ulike tidsforsinkelser (-4 til $+4$) er mulige, og dermed kun 9 distinkte vinkler kan estimeres.

2.5 Varierende forhold

I denne seksjonen gjennomgås teorien bak flere forhold under testing. Mer nøye gjennomgås varierende avstand mellom mikrofonene, varierende avstand på lydkilder og bakgrunnsstøy sin effekt på målinger.

2.5.1 Varierende avstand mellom mikrofoner

Som nevnt i Seksjon 2.4 er antall samples avhengig av avstanden mellom mikrofonene (se Likning 7). Fra Likning 7 og eksempelet i Likning 8 kan en se at jo større avstand det er mellom mikrofonene, desto flere samples kan måles. Siden flere samples fører til høyere vinkeloppløsning (se Seksjon 2.4), vil større avstand mellom mikrofonene føre til høyere vinkeloppløsning.

2.5.2 Varierende avstand mellom lydkilder

Når avstanden mellom lydkilden og mikrofonene er liten, brytes planbølge-antakelsen i Likning 6. Dette fører til ikke-lineære forskjeller i ankomsttid som gjør at estimeringen av innfallsvinkelen blir mindre nøyaktig. Ved en større avstand, er bølgefronten nær plan, og estimatene blir betraktelig mer pålitelige for Likning 6.

2.5.3 Bakgrunnstøy

Intuitivt skulle man forvente at tilstedeværelse av bakgrunnsstøy reduserer presisjonen i målingene, ved å gjøre signalene fra de ulike mikrofonene mindre like og dermed gi svakere topper i krysskorrelasjonen. Men dette gjelder ikke nødvendigvis for alle typer støy.

Krysskorrelasjon fungerer best når signalet inneholder mange ulike frekvenskomponenter, slik som ved impulslyder og bredbåndsstøy [3]. Dette fordi et bredt frekvensinnhold gir et tydelig korrelasjonsbilde med markerte topper og lav grunnstøy. Et eksempel på dette er såkalt *blue noise*, som kjennetegnes av jevn energifordeling og høyere intensitet i høye frekvenser. Slike signaler gir skarpe og presise toppverdier i krysskorrelasjonen, og kan derfor i noen tilfeller bidra til økt presisjon.

Prosjektbeskrivelsen anbefaler også bruk av signaler med mange frekvenskomponenter nettopp av denne grunn. I praksis kan det derfor hende at et visst nivå av jevn støy, med bredt spektrum, ikke forverrer, men faktisk forbedrer, systemets robusthet mot små variasjoner i signalet. Dette undersøkes videre i kapittel 5.

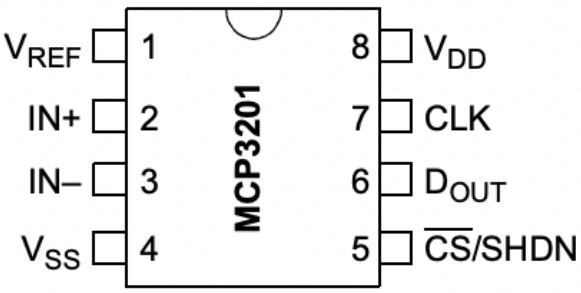
2.6 Analog-til-digital konvertering

Videre brukes ADC-er i dette prosjektet, siden signalene som mikrofon-arrayet tar opp blir sendt ut analogt. Dette må konverteres til et digitalt signal, som er det en ADC gjør [6]. Dette er nødvendig med tanke på videre analyse av opptak, som gjennomføres med digitale

signaler, ikke analoge. I denne sammenhengen brukes MCP3201. Dette er en 12-bits ADC med referansespenning på 3.3 V. Videre gir dette en oppløsning på omtrent 0.8 mV per diskret steg.

2.6.1 MCP3201 ADC-spesifikasjoner

MCP3201 er illustrert i Figur 6. Databladet er tilgjengelig her.



Figur 6: MCP3201 ADC. Bildet er hentet fra databladet til ADC-en [7].

Den benytter SPI-kommunikasjon for dataoverføring [7]. Figur 6 viser pinoppsettet. Denne består av V_{ref} (referansespenning), differensielle innganger ($IN+$ og $IN-$), forsyningsspenning (V_{DD}), potensialreferanse (V_{SS}), klokkesignal (CLK), chip select/shut-down ($CS/SHDN$) og utgangssignal ($DOUT$). ADC-en støtter både enkle og differensielle målinger, der differensielle målinger kan bidra til redusert støy [7].

2.7 Direct Memory Access (DMA) i Raspberry Pi

DMA (Direct Memory Access) gjør det mulig å overføre data direkte fra ADC-ene til minnet uten at CPU-en må involveres i hver enkelt overføring. Dette reduserer belastningen på prosessoren og bidrar til jevnere tidsintervaller mellom målingene, noe som kan gi høyere samplingsfrekvens.

2.7.1 Fordeler med DMA

DMA muliggjør at ADC-data overføres i et konstant tidsintervall, med tanke på at CPU-planlegging ikke forstyrrer selve overføringen. Dette medfører til mer pålitelige målinger med mindre tidsvariasjon. Når CPU-en ikke trenger å håndtere hver måling, blir den tilgjengelig for andre oppgaver, som for eksempel kommunikasjon med eksterne enheter eller databehandling.

2.8 Lavpassfilter for støydemping

I systemer som benytter ADC-er og tidsbasert analyse for retningbestemmelse, er det avgjørende at målesignalene er stabile og minst mulig påvirket av elektrisk støy. Siden både mikrofon-arrayet og ADC-ene drives fra samme forsyningskilde, kan høyfrekvent støy fra strømforsyningen overføres til målesignalene og føre til feil i tidsforsinkelser estimert via krysskorrelasjon. For å redusere dette, benyttes et 2. ordens lavpassfilter koblet til forsyningslinjen. Dette bidrar til å stabilisere driftsspenningen til mikrofonene og ADC-ene, og gir et renere signalgrunnlag for videre behandling. Likning 9 er systemfunksjonen for et 2. ordens lavpassfilter (se Figur 8 for kretsskjema).

$$H(\omega) = \frac{\frac{1}{j\omega C}}{\frac{1}{j\omega C} + j\omega L} \tag{9}$$

Den teoretiske knekkfrekvensen (f_c) til amplituderesponsen er vist i Likning 10. Den er beregnet til 36.07 Hz basert på tre kondensatorer på 100 µF, 470 µF og 100 nF, samt en spole på 100 mH (se Figur 8):

$$|H(\omega)| = \frac{1}{1-\omega^2 CL} \equiv \frac{\sqrt{2}}{2} \iff \omega^2 = \frac{1 \pm \sqrt{2}}{CL} = 4\pi^2 f_c^2 \implies f_c = \sqrt{\frac{1+\sqrt{2}}{4\pi^2 CL}} \tag{10}$$

Her brukes $\frac{\sqrt{2}}{2}$ som terskel for knekkfrekvensen, siden det tilsvarer -3 dB-punktet.

2.9 Teoretisk grunnlag bak impulssignaler

Impulssignaler, som for eksempel et håndklapp, har et bredt frekvensspekter og en tydelig avgrenset starttid. Dette gjør dem godt egnet til målinger som baserer seg på forskjeller i ankomsttid mellom mikrofoner. Siden impulsen er kortvarig og konsentrert i tid, er det lettere å identifisere toppene i krysskorrelasjonen, og dermed estimere tidsforsinkelsen nøyaktig.

Ved antakelse om planbølgefront, altså at bølgefronten treffer alle mikrofoner parallelt, vil en impuls produsere en nesten identisk signalform ved hver mikrofon, kun forskjøvet i tid. Dette gjør at krysskorrelasjonen får en tydelig topp, noe som er avgjørende for nøyaktig vinkelestimering i et mikrofonarray. Impulsens korte varighet minimerer også refleksjoner og romakustiske forstyrrelser i den initielle delen av signalet, som er den mest relevante for beregningene.

2.10 Mulige feilkilder

Det er flere praktiske faktorer som kan introdusere feil og redusere nøyaktigheten i de beregnede innfallsvinklene. Under nevnes noen av de relevante feilkildene for dette oppsettet.

2.10.1 Defekte mikrofoner

Dersom en eller flere mikrofoner er defekte, eller har avvik i følsomhet, impedans eller intern støy, vil de registrerte signalene ha ulik amplitude og kvalitet. Dette kan føre til at signalene fremstår mindre like, selv om de i teorien burde være det. Ved bruk av krysskorrelasjon vil dette gi lavere og potensielt forskjøvede korrelasjonstopper, som i verste fall kan føre til feilidentifikasjon av tidsforsinkelsen.

Eksempelvis vil en mikrofon med lavere følsomhet kunne ha lavere SNR, noe som gjør det vanskelig å identifisere riktig topp i krysskorrelasjonen [8]. Dersom en mikrofon har høyere intern forsterkning enn de andre, kan dette gi utilsiktet forsinkelse på grunn av forskjellig faseoppførsel i analogforsterkningen. Slike feil i tidsforsinkelsene forplanter seg direkte inn i utregningen av innfallsvinkelen, siden vinkelberegningen baserer seg på differanser mellom mikrofonsignaler (se Likning 6).

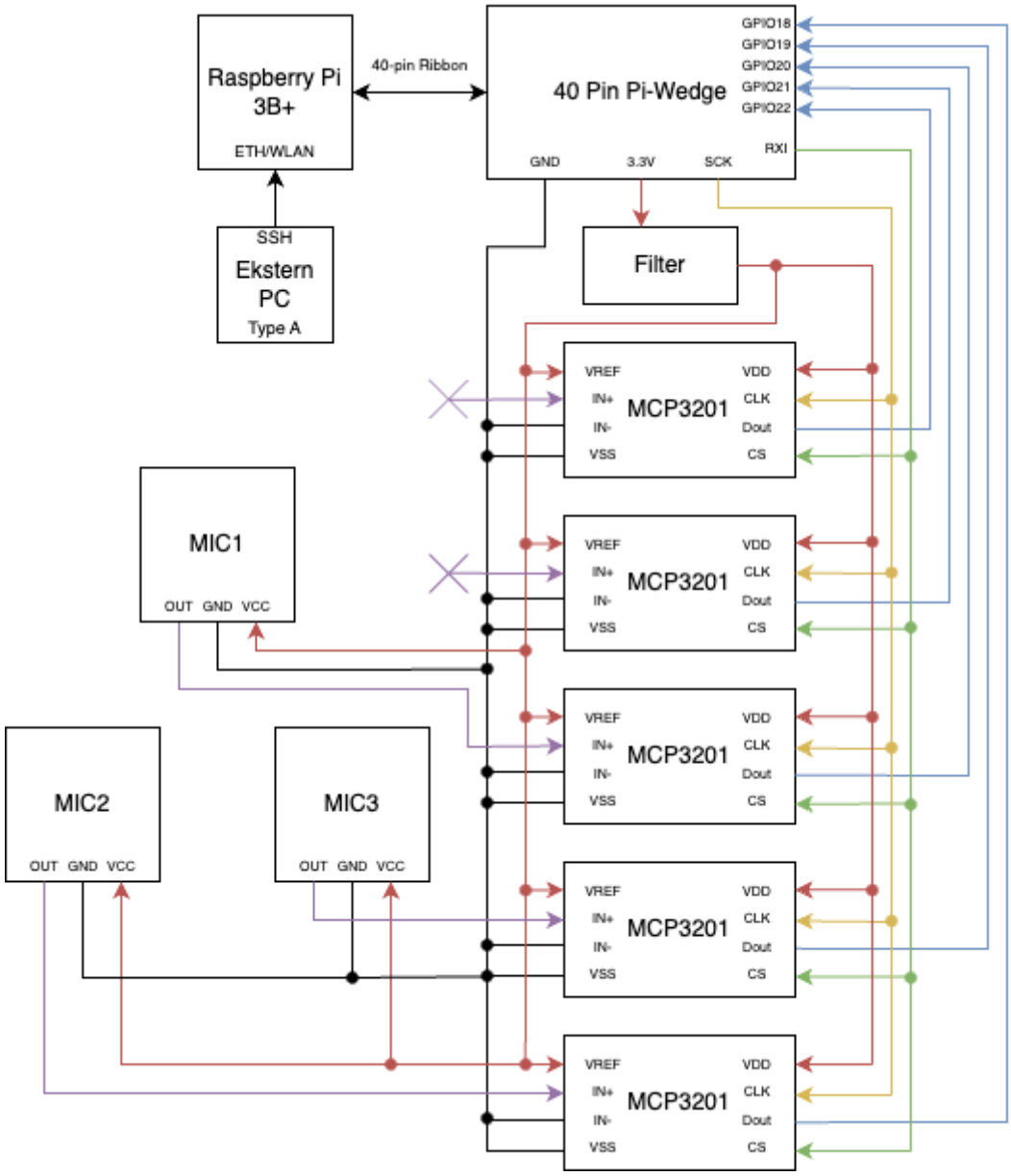
2.10.2 Refleksjoner eller akustiske asymmetrier

Romrefleksjoner og ekko er blant de vanligste feilkildene i akustiske målinger. Når lydbølgen fra impuls-signal (som et klapp) reflekteres fra vegger, tak eller objekter i rommet, vil den reflekterte bølgen nå mikrofonene etter den direkte bølgen, men ofte med høy nok amplitude til at den påvirker signalet. Dette kan føre til at krysskorrelasjonen får flere lokale topper, og gjør det vanskeligere å identifisere hvilken som faktisk representerer den direkte signalbanen.

Akustisk asymmetri i rommet forverrer dette ytterligere. Hvis én mikrofon er nærmere en vegg eller et objekt enn de andre, vil refleksjonen nå den tidligere eller med høyere amplitude, og forstyrre signalet lokalt. Dette bryter forutsetningen om at alle mikrofonene mottar en planbølge, og gjør modellantagelsen i Likning 6 mindre gyldig. I praksis vil dette bety at et impuls-signal fra en side av et rom, ikke vil oppføre seg likt som en fra en annen side, på grunn av asymmetrier i rommet.

3 Eksperimentelt oppsett av målesystem

Et blokkskjema av målesystemet er illustrert i Figur 7.



Figur 7: Blokkskjema av måleoppsettet. Her er komponentene plassert i blokker, med fargekoder på datalinjer og strømlinjer.

Målesystemet består av en Raspberry Pi som henter digitaliserte måleverdier fra tre ADC-er (MCP3201) via SPI-bussen. ADC-ene mottar analoge signaler fra mikrofonene. Videre er et lavpassfilter implementert for å stabilisere forsyningsspenningen (3.3V) til ADC-ene. Raspberry Pi-en fungerer som SPI-master og styrer kommunikasjonen med ADC-ene. Dette gjøres ved hjelp av et felles klokke- (SCK) og chip select-signal (CS). Mikrofonene tar opp lydsignalene og sender det analogt til ADC-ene. Deretter sender ADC-ene digitalisert mikrofon-måledata tilbake til Raspberry Pi via MISO-linjen. Pi Wedge[1] fungerer som en adapter mellom Raspberry Pi og breadboard.

3.1 Oppsett av Raspberry Pi

I dette målesystemet fungerer Raspberry Pi som bindeleddet mellom maskinvarekomponentene og programvaren. For å klargjøre enheten til eksperimentene, installerte vi et operativsystem, satte opp nettverkstilkobling og sørget for ekstern tilgang gjennom SSH.

3.1.1 Installasjon av operativsystem

Raspberry Pi Imager er den tryggeste og anbefalte programvaren for å installere operativsystemet på Raspberry Pi [9]. I denne sammenhengen benyttes 32-bit-versjonen av Raspberry Pi OS og Imagerens innstillinger for å forhåndskonfigurere SSH, endre enhetsnavn (hostname) og sette opp Wi-Fi-tilkoblingen. Dette førte til at vi kunne hoppe over manuelle konfigurasjonstrinn etter installasjonen. Det er viktig å være forsiktig ved valg av lagringsenhet for installasjonen, ettersom feil valg kan resultere i overskriving av harddisken.

3.1.2 SSH-tilkobling og filoverføring via SFTP

Med tanke på at vi forhåndsaktiverte SSH i Imager, kunne vi koble oss til Raspberry Pi umiddelbart etter oppstart uten ytterligere konfigurasjon. Tilkoblingen gjøres i terminal med kommandoen[2]:

```
1 ssh username@hostname.local
```

Kode 1: SSH-tilkobling til Raspberry Pi.

For å overføre filer mellom Raspberry Pi og den eksterne datamaskinen benyttes SFTP. Den anbefalte klienten for enkel filhåndtering varierer fra OS til OS[3].

[1] Pi Wedge er en adapter som gjør det enklere å koble Raspberry Pi sine GPIO-pinner til et breadboard. Pi Wedge sin datablad og oversikt over pinouts finnes her: Datablad/Hookup guide.

[2] Dette er kun et generelt eksempel, `username` og `hostname` må tilpasses til de faktiske `username` og `hostname` til installasjonen.

[3] For MacOS anbefales CyberDuck, for Windows anbefales enten WinSCP eller Cyberduck, mens for Linux anbefales Filezilla.

3.1.3 Verifikasjon av tilkobling

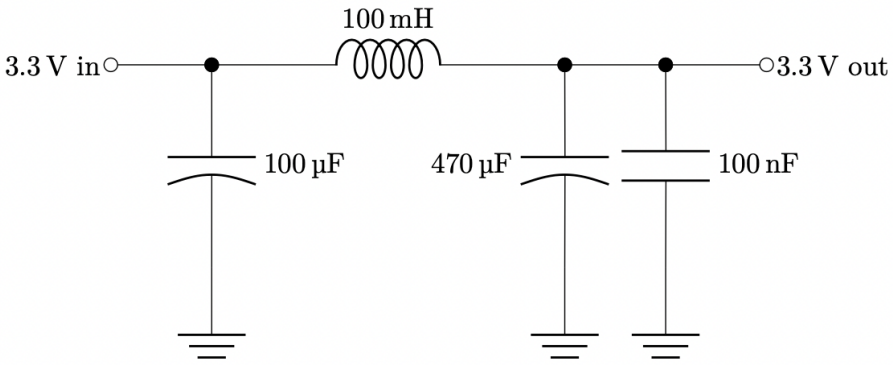
For å bekrefte at oppsettet fungerer som forventet, kan man teste det ved å lage, lagre og kjøre et enkelt Python-program direkte på Raspberry Pi. Dette verifiseres gjennom følgende kommandoer i terminalen.

```
nano test.py #Opprettelse av Python script
print("Hello, World!") #Innhold i test.py
python3 test.py #Kjoering av script
```

Kode 2: Test av Raspberry Pi.

3.2 Støyreduksjonsfilter

Det passive LC-lavpassfilteret mellom Pi Wedge og ADC-ene sikrer en stabil 3.3V-forsyning, og opererer som et støyreduksjonsfilter. Som beskrevet i Seksjon 2.8, er hensikten med filteret å dempe høyfrekvent støy fra strømforsyningen, da slik støy ellers kunne ha introdusert feil i målingene. Kretsskjemaet til lavpassfilteret er vist i Figur 8.



Figur 8: Skjematisk tegning av et passivt LC-lavpassfilter for en 3.3V strømforsyning. Hentet fra [9].

Filteret benytter en induktor på 100 mH i serie med forsyningslinjen for å motvirke raske variasjoner i strøm. I tillegg er kondensatorer på 100 µF, 470 µF og 100 nF koblet til jord for å avlede høyfrekvent støy. Sammen bidrar disse komponentene til å stabilisere utgangsspenningen på 3.3V, som benyttes til å forsyne VDD og VREF på samtlige tre ADC-er.

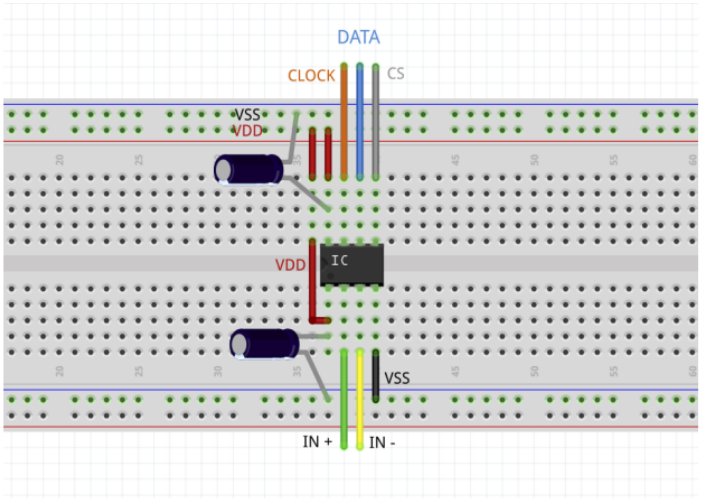
3.3 ADC

ADC-ene er tilkoblet Raspberry Pi i samsvar med Tabell 2. De benytter et felles SPI-klokkesignal (CLK) og chip select (CS), mens hver enkelt ADC har sin egen dedikerte datalinje (Dout) som kobles til separate GPIO-pinner på Raspberry Pi.

Tabell 2: Oversikt over tilkoblinger mellom MCP3201 og Raspberry Pi 3B+.

ADC-pinne	Raspberry Pi GPIO	Funksjon
VDD	3.3V	Forsyner ADC med driftsspenning
VREF	3.3V	Setter referansespenning for konvertering
GND	GND	Felles jordforbindelse
CLK	GPIO11	Klokkesignal generert av Raspberry Pi
DOUT (ADC U5)	GPIO18	Digitalt utgangssignal fra ADC 5
DOUT (ADC U2)	GPIO19	Digitalt utgangssignal fra ADC 2
DOUT (ADC U4)	GPIO20	Digitalt utgangssignal fra ADC 4
DOUT (ADC U1)	GPIO21	Digitalt utgangssignal fra ADC 1
DOUT (ADC U3)	GPIO22	Digitalt utgangssignal fra ADC 3
CS	GPIO15	Aktiverer alle ADC-ene samtidig
IN-	GND	Referanse for differensielle målinger (koblet til jord)
IN+	Ikke tilkoblet RPI	Mottar analogt utgangssignal fra mic1-mic3

For å redusere spenningssvingninger benyttes en 1 µF avkoblingskondensator mellom VREF og IN–, samt en tilsvarende kondensator mellom VDD og VSS for hver ADC. Oppkoblingen for hver enkelt ADC er vist i Figur 9.

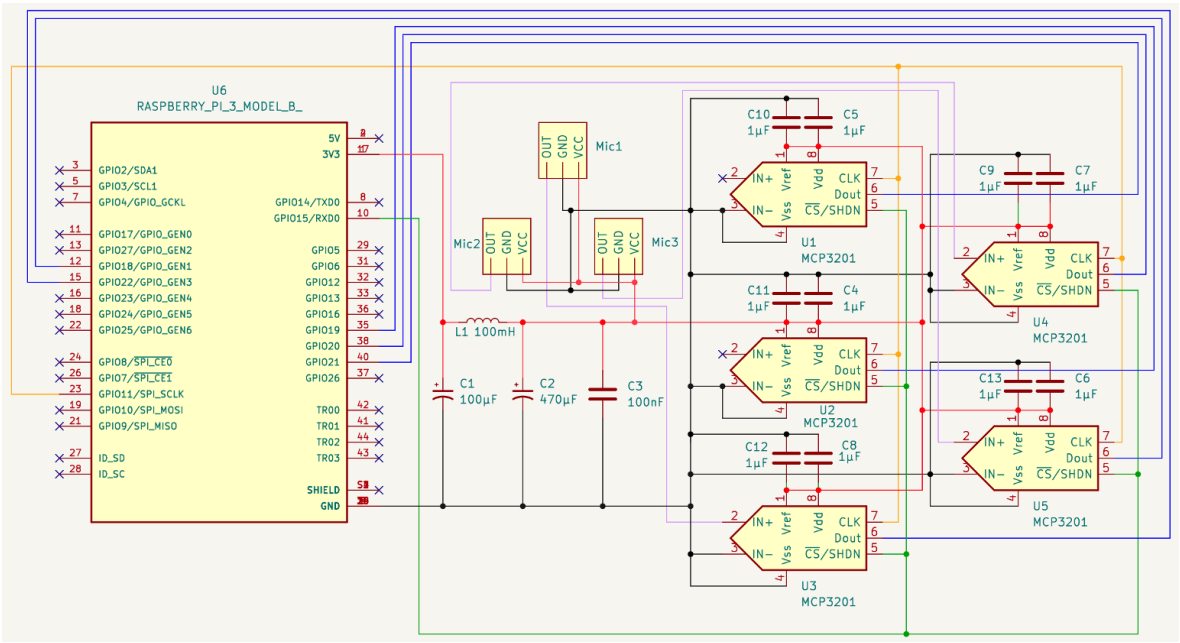


Figur 9: Modell av oppkobling av hver enkelt ADC. Hver ADC trenger sin egen avkoblingskondensator (bypass) for strømforsyningen. Figuren er hentet fra [9].

I motsetning til konvensjonelle SPI-oppsett, hvor hver enhet har sin egen chip select (CS)-linje, benyttes her én felles CS-linje som aktiverer alle ADC-ene samtidig. Data overføres parallelt gjennom separate datalinjer til hver ADC.

3.4 Fysisk implementasjon av målesystemet

Den praktiske oppbygningen av systemet er utført i tråd med blokkskjemaet vist i Figur 7, der de ulike komponentene er fysisk koblet sammen. En fullstendig oversikt over koblingene fremgår av kretstegningen i Figur 10.

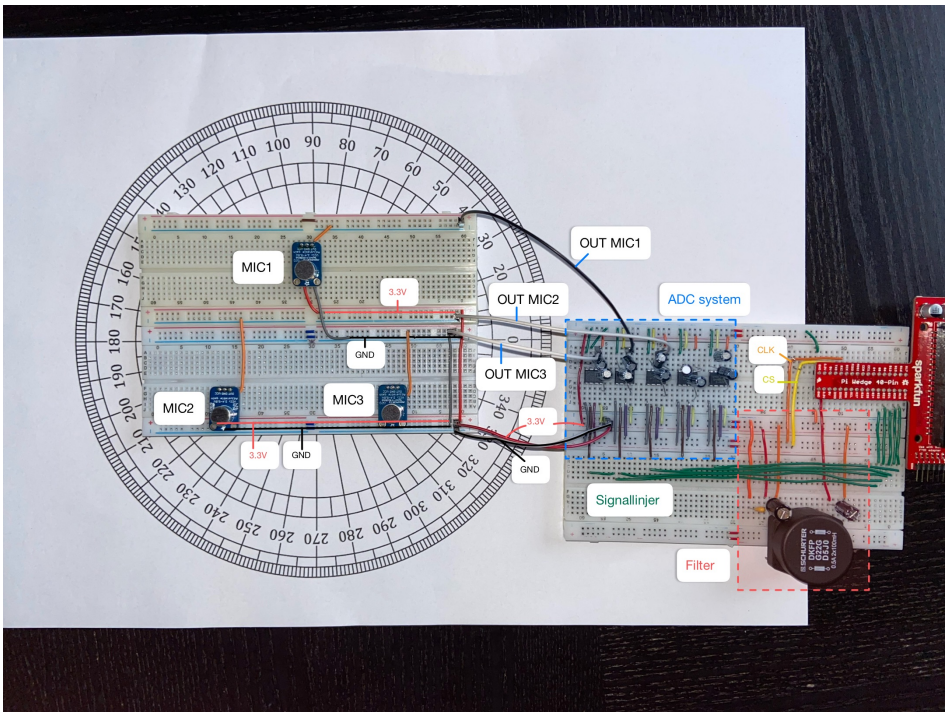


Figur 10: Kretstegning av realisert system tegnet i programmet KiCad. Med fargekoder for datalinjer og strømlinjer som tilsvarer samme farger som introdusert i Figur 7.

I kretstegningen er de fem analog-til-digital-konverterne (ADC) merket U1–U5, hvorav kun de tre siste er i bruk i dette oppsettet (tilknyttet signalet IN+). Raspberry Pi (RPI) er merket som U6. Kondensatorene er nummerert C1–C13, og spolen er merket L1. Verdiene for hver kondensator og spolen er angitt under komponentbetegnelsene. Mikrofonene Mic1, Mic2 og Mic3 kobles til en felles spenningsforsyning (VCC) og jord (GND), mens deres utgangssignaler sendes til hver sin ADC via separate signalbaner markert i lilla. Hver mikrofon kobles dermed til én av de tre aktive ADC-inngangene.

Kretstegningen benytter en konsekvent fargekoding: oransje angir klokkesignalet (CLK), grønn brukes for chip select (CS), rød representerer forsyningsspenningen på 3.3 V, svart markerer jord (GND), og blå viser datalinjene fra ADC-ene til Raspberry Pi.

Figur 11 viser den realiserte kretsen på breadboard, hvor mikrofonene er koblet til ADC-ene, og ADC-ene er tilkoblet Raspberry Pi. Merk her at mikrofonene er fordelt i en likesidet trekant, på grunn av vår utgangspunkt i Likning 6 (se Seksjon 2.3).



Figur 11: Oppkoblet krets på breadboard for måling av impulssignaler via mikrofoner. Hver del av kretsen er markert med hver sin tekstboks.

3.5 Metodikk for opptak og måling av lydsignaler

Til slutt må metoden bak opptak og måling til lydsignaler avklares. Lyd genereres med fysiske klapp i forskjellige posisjoner i et rom. Med utgangspunkt i Likning 7, krever dette at avstanden mellom mikrofon-arrayet og lydkilden (person som gjør et klapp), er lang nok for at lydbølgen som mikrofonene mottar oppfører seg som en planbølge (se Seksjon 2.3). Dermed skal det utføres klapp under måling ved relativt lange avstander, samt korte avstander for å se om endringer i lydbølge-form vil endre resultatene drastisk. I tillegg skal målingene utføres i forskjellige vinkler, for å se hvor nøyaktig systemet er. Herfra kan ordinær gjennomsnitt, standardavvik og varians måles.

4 Tester og resultater

Dette kapittelet presenterer resultatene fra testene som ble utført for å evaluere mikrofonarrayets evne til å bestemme vinkelen til en lydkilde. Til disse testene ble ADC-målesystemet utviklet i Prosjekt 1 brukt for innsamling og behandling av signaler fra mikrofonene. Mikrofonene er plassert i et trekantformet array med en innbyrdes avstand på ca. 8 cm under standard måleforhold, definert som et stille rom uten bakgrunnsstøy, og med lydkilden (et håndklapp) plassert omtrent 1,5 meter fra arrayet. Resultatene under disse standardiserte forholdene fungerer som et referansegrunnlag for evaluering av målesystemets ytelse under varierende eksperimentelle betingelser, slik som kortere avstand mellom mikrofonene, kortere avstand til lydkilden, og med tilstedeværelse av bakgrunnsstøy.

4.1 Relevante oppdagelser gjort i Prosjekt 1

Resultatene fra Prosjekt 1 viser at ADC-ene registrerer signalene med korrekt bølgeform og amplitude, men det observeres en systematisk liten DC-offset i signalene [10]. Denne offseten fremtrer som en tydelig komponent ved 0 Hz i frekvensspekteret. Signal-til-støy-forholdet (SNR) er beregnet for signaler med frekvenser på 1 kHz, 2 kHz og 10 kHz, og disse viser verdier rundt 55 dB, som er betydelig lavere enn den teoretiske verdien på 74 dB for ADC-en [10]. Dette avviket skyldes sannsynligvis kvantiseringsstøy og interferens fra strømforsyningen. I tillegg ble det målt en lavere knekkfrekvens i lavpassfilteret enn forventet, noe som kan være et resultat av målinger utført med resten av systemet tilkoblet [10]. Lavere knekkfrekvens vurderes imidlertid ikke å ha en negativ effekt på SNR, og kan tvert imot være gunstig for støydemping. Disse observasjonene er relevante for tolkningen av resultatene i de videre testene.

4.2 Måleresultater

Tabell 3 gir en oppsummering av resultatene fra vinkelmålinger under standard måleforhold. Det observeres at den gjennomsnittlig målte vinkelen er lavere enn den forventede vinkelen for alle målinger unntatt ved 160°. Spesielt høy usikkerhet (standardavvik og varians) ble observert ved vinklene 270° og 300°. Årsakene til disse avvikene vil bli diskutert nærmere i kapittel 5.

Tabell 3: Oppsummering av måleresultater for alle testede vinkler (vinklene listet under `Forventet vinkel`). Verdiene er beregnet som gjennomsnitt, standardavvik og varians over 9 målinger per vinkel.

Forventet vinkel (°)	Gjennomsnittlig estimert vinkel (°)	Standardavvik (°)	Varians (°²)
0	356.99	4.696	22.055
30	28.90	3.122	9.753
55	54.19	2.252	5.073
90	87.80	3.819	14.582
110	106.97	2.633	6.933
130	128.93	5.973	35.677
160	160.16	7.596	57.692
210	195.50	7.957	63.321
240	237.43	3.676	13.515
270	229.39	13.087	171.264
300	296.11	10.737	115.278
330	326.07	3.830	14.673

Detaljerte visualiseringer av samtlige 9 målinger for hver av de 12 vinklene ligger i vedlegg A, mens fire av disse vinklene er presentert nedenfor.

4.2.1 Måleresultater ved 0°

Tabell 4 viser resultatene fra ni individuelle målinger der lydkilden ble plassert ved 0°, ca. 1,5 meter foran mikrofonarrayet.

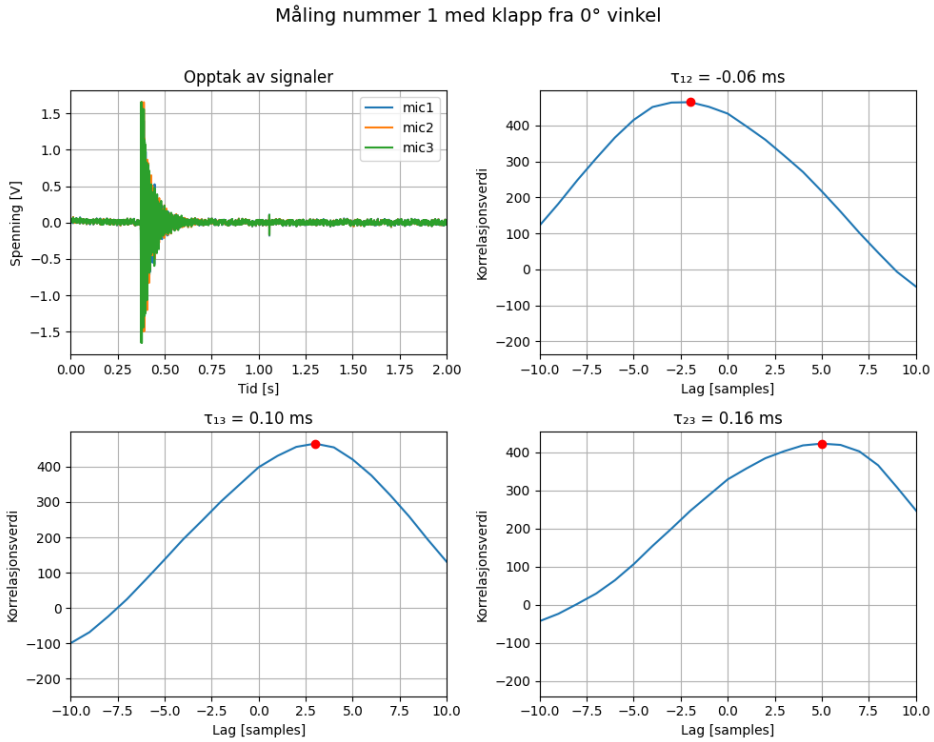
Tabell 4: Vinkel beregnet ved 9 forskjellige målinger av 0°, og tilhørende tidsforsinkelser τ_{ij} mellom mikrofon $i = 1, 2, 3$ og mikrofon $j = 1, 2, 3$.

Måling #	Vinkel i grader (°)	τ_{12} (ms)	τ_{13} (ms)	τ_{23} (ms)
1	353.413	-0.064	0.096	0.160
2	0	-0.064	0.064	0.096
3	0	-0.064	0.064	0.128
4	353.413	-0.064	0.096	0.160
5	0	-0.064	0.064	0.064
6	346.102	-0.032	0.064	0.064
7	0	-0.064	0.064	0.096
8	0	-0.064	0.064	0.128
9	0	-0.064	0.064	0.128

Ved 0° er lydkilden plassert ca. 1,5 meter foran mikrofonarrayet, nærmest mikrofon 3, og litt nærmere mikrofon 1 enn mikrofon 2. Seks av de ni målingene treffer svært godt med den forventede vinkelen. Tidsforsinkelsene τ_{12} er negative, mens τ_{13} og τ_{23} er positive, noe som er i tråd med mikrofonenes plassering og teorien beskrevet i Seksjon 2.2. Vinklene er beregnet ved hjelp av Likning 6.

Ved forventet vinkel på 0° betyr en gjennomsnittsmåling på 356.99° at målingene ligger nær 0°, da systemet måler vinkler mellom 0° og 360°.

Figur 12 viser signalene fra de tre mikrofonene under måling 1 ved 0° vinkel, samt de beregnede krysskorrelasjonene mellom mikrofonparene. Toppen i hver krysskorrelasjonskurve markerer det lag (antall samples forskyvning) som gir høyest korrelasjon, og denne verdien er deretter konvertert til tidsforsinkelse i millisekunder. De tre tidsforsinkelsene ble brukt som grunnlag for beregning av innfallsvinkelen ved hjelp av Likning 6.



Figur 12: Målinger i stille rom 0° på oppsett

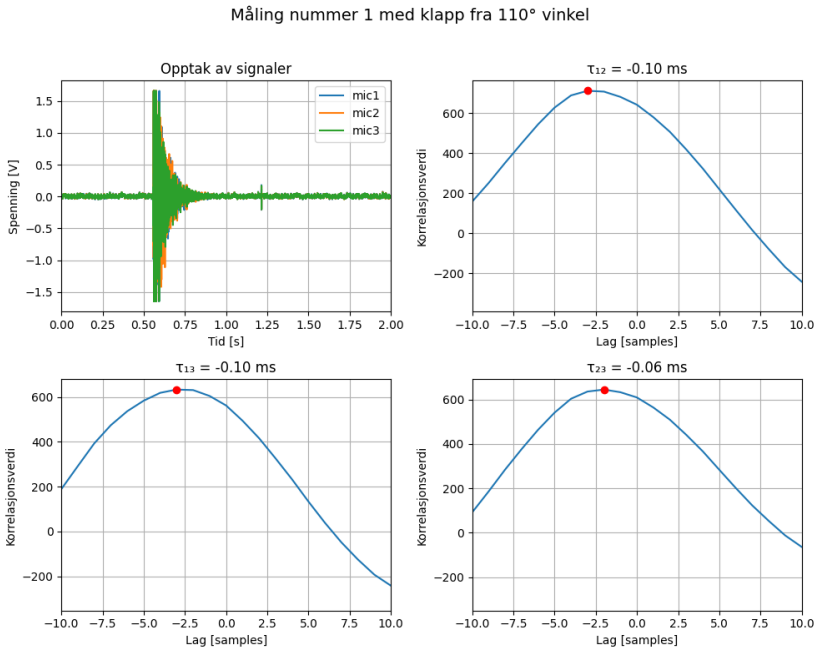
4.2.2 Måleresultater ved 110°

Resultatene ved 110° ble inkludert for å gi et representativt bilde av generelle resultater uten spesifikke forventninger eller problemer knyttet til denne vinkelen. Målingene viser stabilitet med relativt lav standardavvik og er presentert på samme måte som for 0° i Tabell 5 og Figur 13.

Tabell 5: Vinkel beregnet ved 9 forskjellige målinger av 110°, og tilhørende tidsforsinkelser τ_{ij} mellom mikrofon $i = 1, 2, 3$ og mikrofon $j = 1, 2, 3$.

Måling #	Vinkel i grader (°)	τ_{12} (ms)	τ_{13} (ms)	τ_{23} (ms)
1	111.052	-0.096	-0.096	-0.064
2	107.784	-0.128	-0.160	-0.064
3	106.102	-0.128	-0.128	-0.064
4	107.784	-0.128	-0.160	-0.064
5	100.893	-0.128	-0.160	-0.032
6	106.102	-0.128	-0.128	-0.064
7	106.102	-0.128	-0.128	-0.064
8	109.107	-0.128	-0.192	-0.064
9	107.784	-0.128	-0.160	-0.064

Det er verdt å merke seg at alle tidsforsinkelsene τ_{12}, τ_{13} og τ_{23} er negative i disse målingene. Dette indikerer at mikrofon 1 mottar signalet før både mikrofon 2 og mikrofon 3, noe som er konsistent med en lydkilde som kommer inn fra en vinkel foran og til venstre for arrayet, slik som ved 110°.



Figur 13: Målinger i stille rom 110° på oppsett

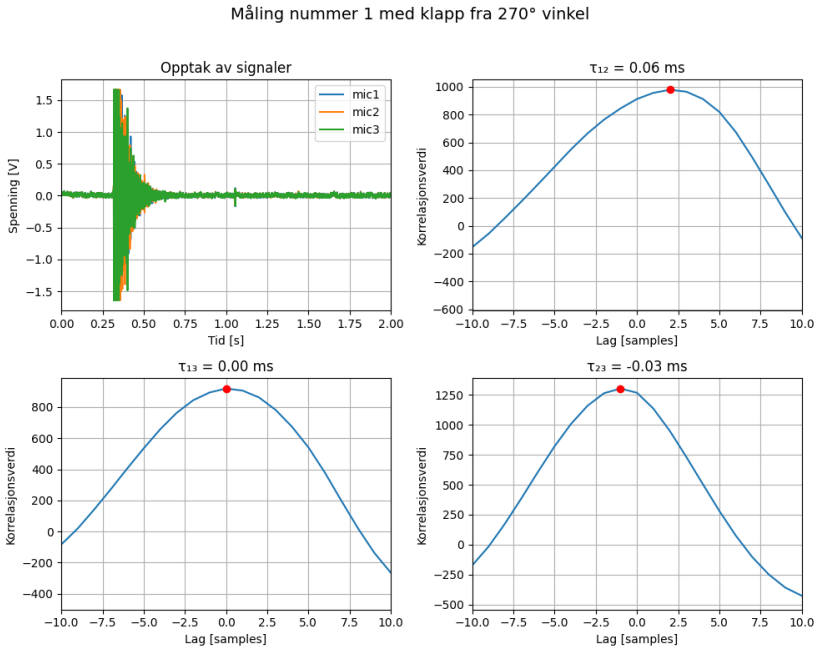
4.2.3 Måleresultater ved 270°

Vinkelen 270° ga høyest usikkerhet i målingene med et betydelig standardavvik på 13.087° og varians på 171.264. Dette antyder betydelige utfordringer med måling fra denne retningen, noe som kan være relatert til refleksjoner eller geometri i oppsettet. Dette analyseres videre i kapittel 5. De ulike målte vinklene fra de 9 målingene gjort ved 270° og deres tidsforsinkelser er presentert i Tabell 6.

Tabell 6: Vinkel beregnet ved 9 forskjellige målinger av 270°, og tilhørende tidsforsinkelser τ_{ij} mellom mikrofon $i = 1, 2, 3$ og mikrofon $j = 1, 2, 3$.

Måling #	Vinkel i grader (°)	τ_{12} (ms)	τ_{13} (ms)	τ_{23} (ms)
1	220.893	0.064	0.000	-0.032
2	259.107	0.128	0.064	0.000
3	220.893	0.064	0.000	-0.032
4	240.000	0.064	0.032	-0.032
5	220.893	0.064	0.000	-0.032
6	220.893	0.064	0.000	-0.032
7	240.000	0.064	0.032	-0.032
8	220.893	0.064	0.000	-0.032
9	220.893	0.064	0.000	-0.032

Figur 14 viser resultatene av måling 1 med klapp fra 270°.



Figur 14: Målinger i stille rom 270° på oppsett

I Tabell 4, 5, og 6 ser vi at flere av vinklene går igjen med ganske presise desimaler. Dette er

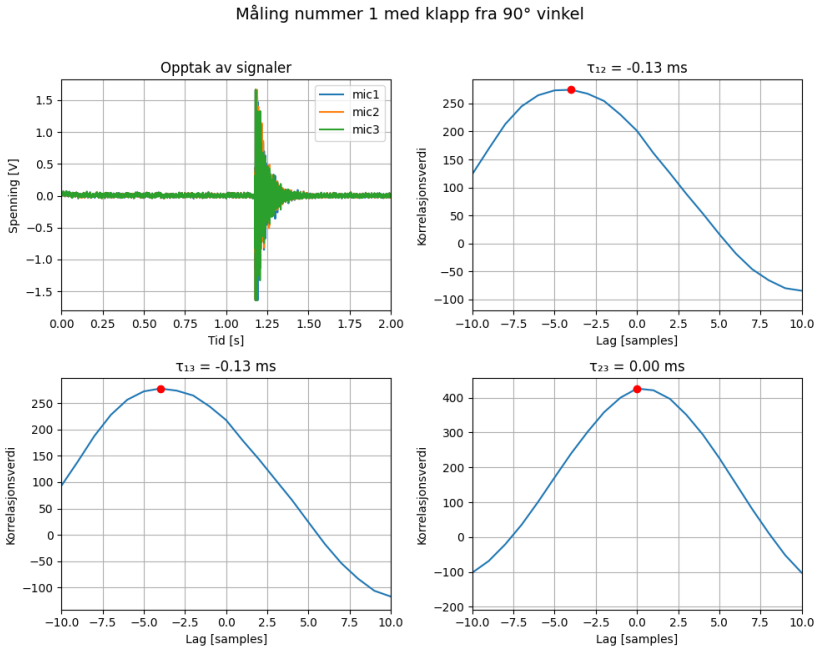
ikke tilfeldig og er en konsekvens av samplingfrekvensen og avstanden mellom mikrofonene. Systemet klarer altså ikke å måle alle vinkler mellom 0° og 360°, som beskrevet i teorien i Seksjon 2.4.

4.2.4 Måleresultater ved 90°

Målingene ved 90° er gjennomført under referanseoppsettet og benyttes som utgangspunkt for sammenligning med testene under varierende forhold. Tabell 7 viser de beregnede vinklene og tilhørende tidsforsinkelser for ni målinger, mens Figur 15 illustrerer signalforløp og krysskorrelasjon for én av målingene.

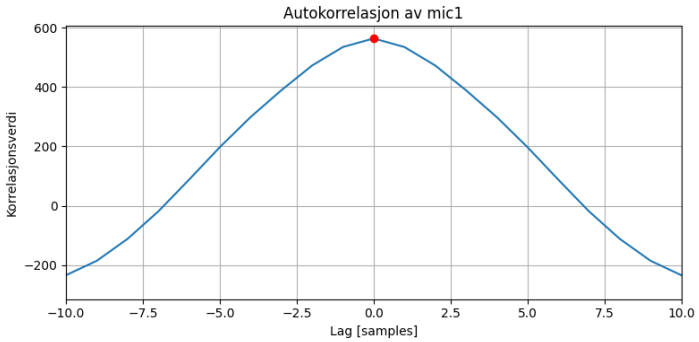
Tabell 7: Vinkel beregnet ved 9 forskjellige målinger av 90°, og tilhørende tidsforsinkelser τ_{ij} mellom mikrofon $i = 1, 2, 3$ og mikrofon $j = 1, 2, 3$.

Måling #	Vinkel i grader (°)	τ_{12} (ms)	τ_{13} (ms)	τ_{23} (ms)
1	90.000	-0.128	-0.128	0.000
2	85.285	-0.128	-0.096	0.000
3	90.000	-0.128	-0.128	0.000
4	90.000	-0.096	-0.096	0.000
5	83.413	-0.096	-0.064	0.000
6	90.000	-0.160	-0.160	0.000
7	83.413	-0.096	-0.064	0.000
8	94.715	-0.096	-0.128	0.000
9	83.413	-0.096	-0.064	0.000



Figur 15: Målinger i stille rom 90° på oppsett

Figur 16 viser autokorrelasjonen til signalet fra mikrofon 1 under måling 1 med lydkilden plassert ved 90°. Som nevnt i Seksjon 2.2.1 er autokorrelasjon et spesialtilfelle av krysskorrelasjon hvor signalet korreleres med seg selv. Som forventet gir dette en tydelig toppverdi ved lag 0, siden et signal alltid vil være maksimalt korrelert med seg selv uten tidsforskyvning. Den symmetriske formen rundt senter skyldes at signalet korreleres med både frem- og bakoverforskyvninger.



Figur 16: Autokorrelasjon av mikrofon 1 for måling 1 ved 90°.

Denne analysen er ikke direkte brukt i vinkelberegningen, men den gir en verifikasjon på at signalet inneholder en klar, temporært konsistent impuls, som er essensiell for at krysskorrelasjonen mellom ulike mikrofoner skal fungere presist (jf. Seksjon 2.2.1).

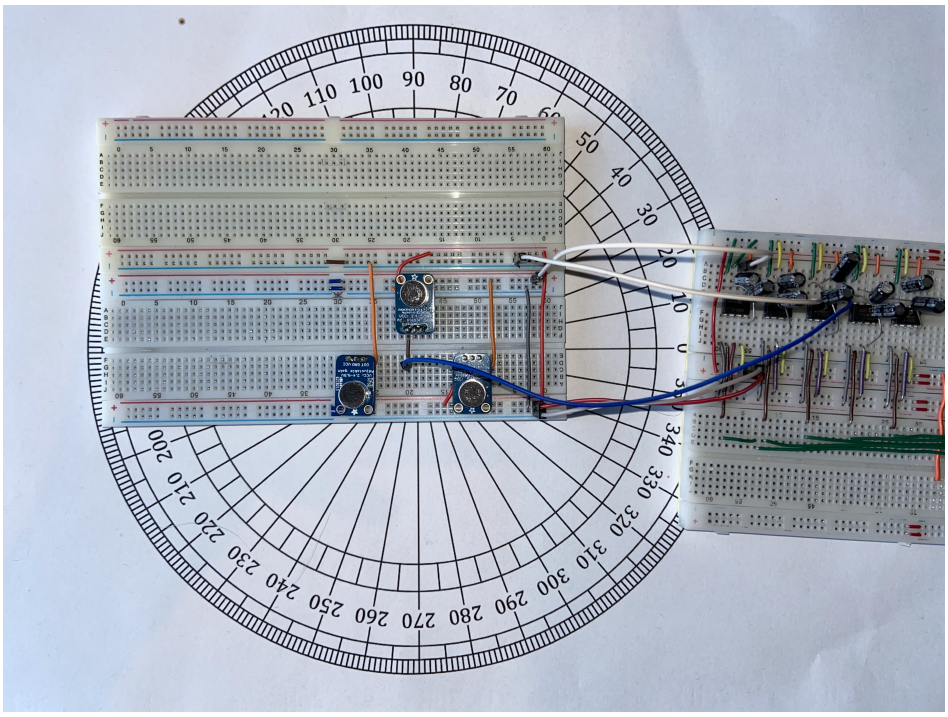
4.3 Tester under varierende forhold

Resultatene som er presentert til nå stammer fra tester gjennomført under det vi har definert som referanseforhold. Disse testene ble utført i et stille rom, med 8 cm avstand mellom hver mikrofon og lydkilden plassert omtrent 1,5 meter foran mikrofonarrayet. Målingene ved 90° ovenfor, presentert i Tabell 7, fungerer som et referansegrunnlag for sammenligningen med de testene som ble utført under varierende eksperimentelle forhold presentert nedenfor.

For å undersøke målesystemets presisjon og robusthet ble det gjennomført tilleggsmålinger under tre andre forhold: redusert avstand mellom mikrofonene, kortere avstand til lydkilden, og tilstedeværelse av bakgrunnsstøy. Hensikten med disse testene var å belyse hvordan slike variasjoner påvirker nøyaktigheten i vinkelestimatene. For å beholde et sammenlignbart grunnlag ble alle testene under varierende forhold gjennomført med lydkilden plassert ved 90° på mikrofonarrayet, og i samme rom. Dette reduserer antall feilkilder og gjør analysen mer konsistent og relevant.

4.3.1 Kortere avstand mellom mikrofonene (4cm)

Figur 17 viser oppsettet av mikrofon-arrayet med 4 cm avstand mellom mikrofonene.



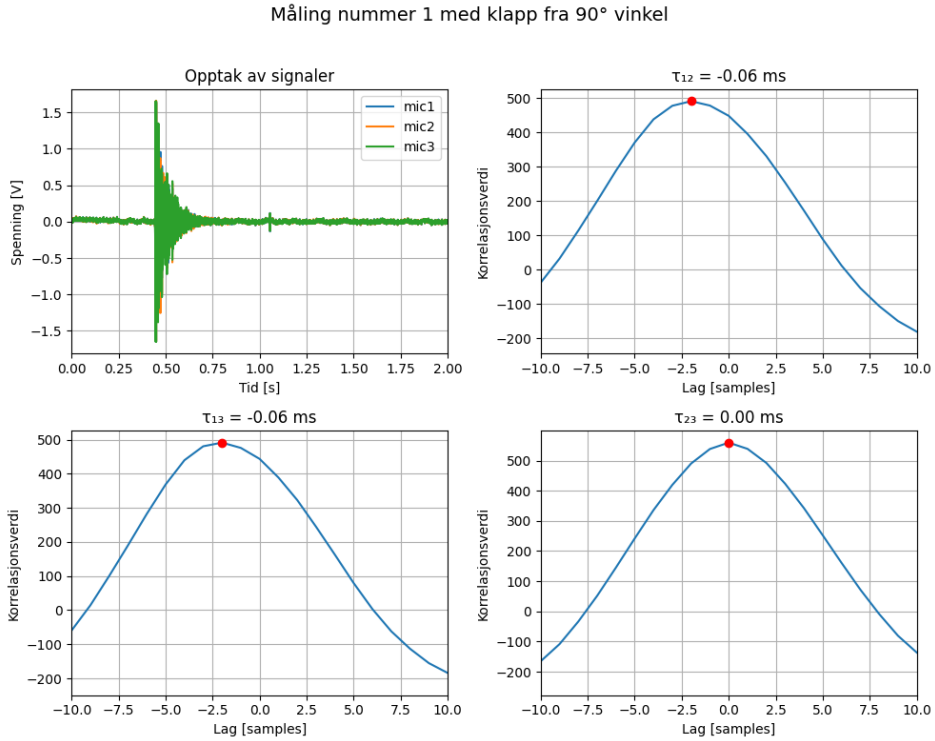
Figur 17: Fysisk oppsett av mikrofon-arrayet med 4 cm avtand mellom hver mikrofon.

Det nye systemet er testet på lik linje som det forrige systemet, og resultatene av 9 individuelle målinger er oppsummert i Tabell 8.

Tabell 8: Vinkel beregnet ved 9 forskjellige målinger av 90°, og tilhørende tidsforsinkelser τ_{ij} mellom mikrofon $i = 1, 2, 3$ og mikrofon $j = 1, 2, 3$.

Måling #	**Vinkel i grader (°)**	τ_{12} **(ms)**	τ_{13} **(ms)**	τ_{23} **(ms)**
1	90.000	-0.064	-0.064	0.000
2	90.000	-0.032	-0.032	0.000
3	90.000	-0.032	-0.032	0.000
4	90.000	-0.064	-0.064	0.000
5	90.000	-0.064	-0.064	0.000
6	90.000	-0.032	-0.032	0.000
7	79.107	-0.064	-0.032	0.000
8	90.000	-0.032	-0.032	0.000
9	79.107	-0.064	-0.032	0.000

Som man kan legge merke til i Tabell 8, er variasjonen i tidsforsinkelsene mellom målingene relativt begrenset. Dette gjenspeiler det reduserte antallet mulige tidsforsinkelser som følge av den kortere mikrofonavstanden. En nærmere analyse av dette presenteres i kapittel 5. Figur 18 illustrerer signal og krysskorrelasjonene fra den første målingen.



Figur 18: Målinger i stille rom 90° på oppsett med kun 4cm mellom mikrofonene

4.3.2 Kortere avstand fra lydkilde (10cm)

Resultatene fra målinger med kortere avstand mellom lydkilden og mikrofonarrayet (omtrent 10 cm) er presentert i Tabell 9. Oppsettet var ellers identisk med testen med 4 cm mellom mikrofonene (se Figur 17), og lydkilden ble som tidligere plassert ved 90°.

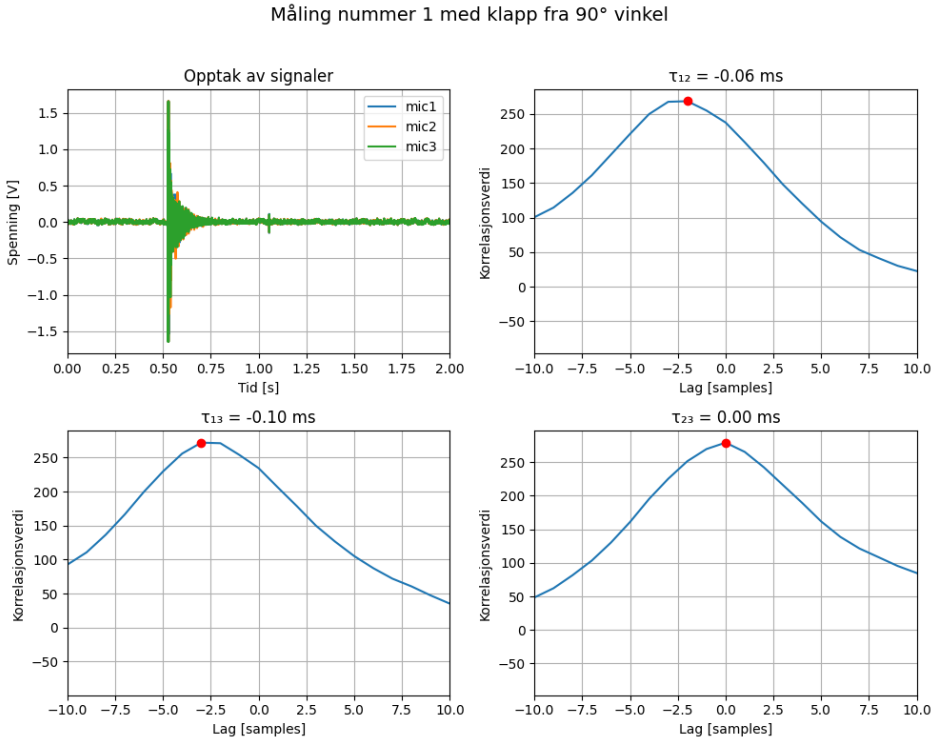
Tabell 9: Vinkel beregnet ved 9 forskjellige målinger av 90°, og tilhørende tidsforsinkelser τ_{ij} mellom mikrofon $i = 1, 2, 3$ og mikrofon $j = 1, 2, 3$.

Måling #	**Vinkel i grader (°)**	τ_{12} **(ms)**	τ_{13} **(ms)**	τ_{23} **(ms)**
1	96.587	-0.064	-0.096	0.000
2	90.000	-0.064	-0.064	0.000
3	90.000	-0.096	-0.096	0.000
4	90.000	-0.032	-0.032	0.000
5	90.000	-0.064	-0.064	0.000
6	90.000	-0.064	-0.064	0.000
7	90.000	-0.096	-0.096	0.000
8	70.893	-0.096	-0.064	0.032
9	90.000	-0.064	-0.064	0.000

Tabellen viser at flere av målingene gir vinkler tett opp mot 90°, men det er også tydelige variasjoner i både vinkel og tidsforsinkelser. Særlig utpeker måling 8 seg med et markant avvik både i estimert vinkel og i τ_{23}, som er 0.032 ms, en verdi som ikke opptrer i referansemålingene.

Disse avvikene indikerer at redusert avstand mellom lydkilde og mikrofonarray kan påvirke presisjonen i vinkelestimatene. Figur 19 viser de rå måledataene og korresponderende kryss-korrelasjoner for én av målingene.

En mulig forklaring på variasjonene knyttes til brudd på planbølgeantakelsen, som forutsettes i Likning 6. Dette vil diskuteres nærmere i kapittel 5.



Figur 19: Målinger i stille rom 90° på oppsett med kun 4cm mellom mikrofonene, og kort avstand (10 cm) fra lydkilde.

4.3.3 Bakgrunnstøy under måling

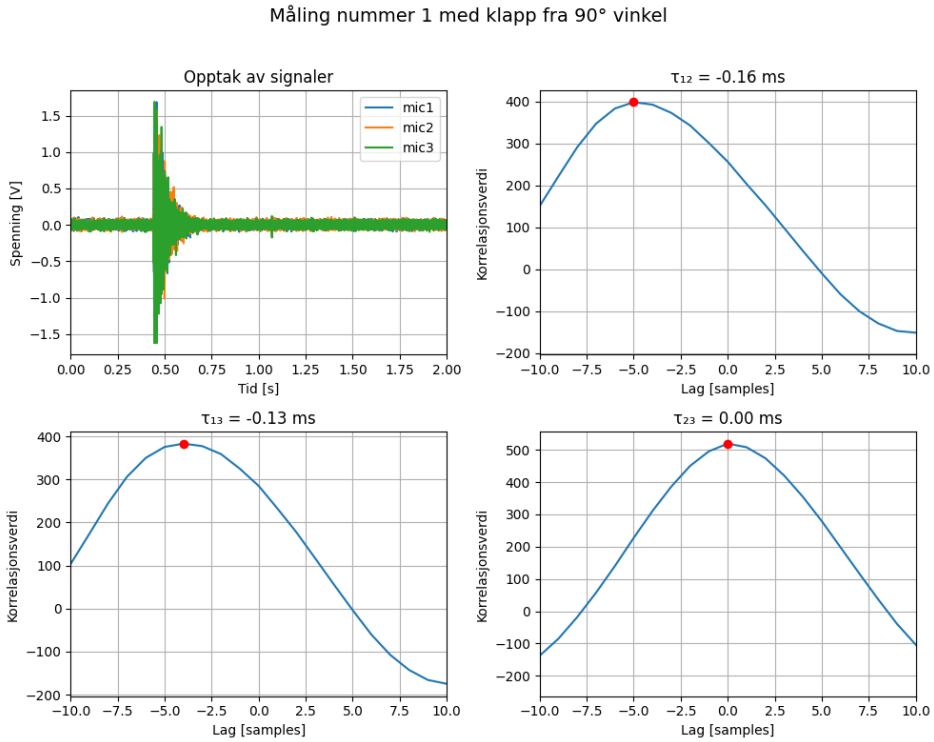
Resultatene fra målingene med bakgrunnsstøy i rommet er oppsummert i Tabell 10. Testen ble gjennomført under samme oppsett som referansemålingene, med 8 cm mellom mikrofonene og lydkilden plassert ved 90°. I denne testen ble det spilt av jevn bakgrunnsstøy (såkalt blue noise) under målingene.

Tabell 10: Vinkel beregnet ved 9 forskjellige målinger av 90°, og tilhørende tidsforsinkelser τ_{ij} mellom mikrofon $i = 1, 2, 3$ og mikrofon $j = 1, 2, 3$.

Måling #	**Vinkel i grader (°)**	τ_{12} **(ms)**	τ_{13} **(ms)**	τ_{23} **(ms)**
1	86.330	-0.160	-0.128	0.000
2	90.000	-0.096	-0.096	0.000
3	90.000	-0.064	-0.064	0.000
4	90.000	-0.096	-0.096	0.000
5	85.285	-0.128	-0.096	0.000
6	85.285	-0.128	-0.096	0.000
7	90.000	-0.096	-0.096	0.000
8	90.000	-0.064	-0.064	0.000
9	85.285	-0.128	-0.096	0.000

Som vist i tabellen varierer de estimerte vinklene noe mellom målingene, men generelt er resultatene stabile og ligger nær 90°. Tidsforsinkelsene følger et mønster tilsvarende det som ble observert i referansemålingene. Figur 20 viser signalene og krysskorrelasjonene fra én av målingene under støybetingelser.

Det bemerkes at denne testen gav lavest varians og standardavvik av alle eksperimentene, til tross for at den ble gjennomført med støy i rommet. Én mulig forklaring er at støyen, som inneholder et bredt spekter av frekvenser, faktisk forbedrer krysskorrelasjonens evne til å identifisere tidsforskyvninger. En mer detaljert drøfting av dette følger i kapittel 5.



Figur 20: Målinger i støyfylt rom 90° på oppsett

4.4 Analyse og oppsummering av resultater

For å gi en oversiktlig sammenligning mellom de ulike eksperimentelle forholdene, er nøkkelstatistikk samlet i Tabell 11.

Tabell 11: Sammenligning av statistiske mål for vinkelmålinger ved fire ulike eksperimentelle forhold. Verdiene er beregnet ut fra 9 målinger for hvert tilfelle.

Forhold	**Gjennomsnittlig estimert vinkel (°)**	**Standardavvik (°)**	**Varians ($^\circ{}^2$)**
Referanseoppsett	87.80	3.82	14.58
Kortere avstand mellom mikrofonene	87.58	4.53	20.51
Kortere avstand til lydkilde	88.61	6.59	43.45
Bakgrunnstøy	88.02	2.23	4.99

De observerte forskjellene er delvis i tråd med forventningene basert på teorien i Seksjon 2, og vil bli analysert mer i det påfølgende kapittelet.

5 Diskusjon

5.1 Oppløsning og mikrofonavstand

Et viktig funn fra testene er hvordan den fysiske avstanden mellom mikrofonene påvirker systemets oppløsning og presisjon i estimering av innfallsvinkel. Teorien i Seksjon 2.4 forklarer at maksimal tidsforsinkelse mellom mikrofoner avhenger av både mikrofonavstand og samplingfrekvens, og at dette igjen bestemmer hvor mange forskjellige forsinkelser, og dermed vinkler, som systemet faktisk kan skille mellom. Ved å bruke samplingfrekvensen $f_s = 31250$ Hz og mikrofonavstander på henholdsvis 8 cm og 4 cm, får vi følgende teoretiske maksforsinkelser fra Likning 7;

$$n_{\text{maks, 8cm}} = \left\lfloor \frac{0.08 \cdot 31250}{343} \right\rfloor = \lfloor 7.29 \rfloor = 7,$$

$$n_{\text{maks, 4cm}} = \left\lfloor \frac{0.04 \cdot 31250}{343} \right\rfloor = \lfloor 3.65 \rfloor = 3.$$

Systemet kan altså skille mellom 15 forskjellige forsinkelser (fra -7 til $+7$) ved 8 cm, men bare 7 forsinkelser (fra -3 til $+3$) ved 4 cm. Dette reduserer antall mulige vinkler systemet kan estimere, og gir grovere vinkeloppløsning. Dette stemmer overens med observasjonene fra Tabell 8, hvor alle målte tidsforsinkelser ligger i området 0, ± 0.032 ms og ± 0.064 ms. I kontrast gir referanseoppsettet flere ulike tidsforsinkelsesverdier, som gjør det mulig å estimere vinkler med høyere presisjon. Det ble også observert noe høyere varians ved kort mikrofonavstand, noe som kan forklares med at små endringer i målt tidsforsinkelse får større utslag når oppløsningen er lav.

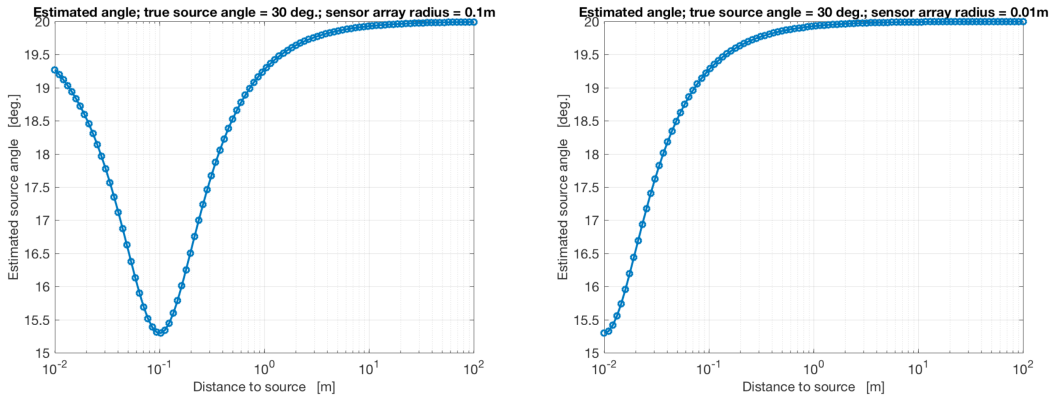
Dette illustrerer hvordan mikrofonavstand er en sentral parameter i systemets oppløsning, og må tilpasses i balanse med krav til plass og praktiske hensyn.

5.2 Planbølgeantakelse og avstand til lydkilde

Resultatene fra testen med kort avstand mellom lydkilden og mikrofonarrayet (ca. 10 cm) viste størst spredning i vinkelestimatene, målt både i standardavvik og varians. Dette kan forklares med at planbølgeantakelsen, som ligger til grunn for vinkelberegningen i Likning 6, ikke lenger er gyldig når lydkilden plasseres med så kort avstand til oppsettet [3].

Ved korte avstander treffer lydbølgene mikrofonene med sfæriske bølgefronter i stedet for plane [3]. Dette gir ikke-lineære tidsforsinkelser mellom mikrofonene som ikke stemmer overens med forutsetningene i modellen, og det gir dermed større feil i beregnet vinkel. I én av målingene ble for eksempel en vinkel helt ned mot 70.893° estimert, til tross for at lydkilden fysisk var plassert i 90°-retning. Slike avvik oppsto ikke i de øvrige 90°-målingene, og illustrerer hvordan små avvik i bølgefrontens form kan føre til store feil i estimering.

Denne effekten er også illustrert i prosjektoppgaven [3], der man simulerer hvordan vinkelestimatet påvirkes av avstanden til lydkilden. Figur 21 viser at feilen i estimert vinkel øker kraftig når avstanden reduseres under 0.5 meter.



Figur 21: Estimert vinkel som funksjon av avstand til lydkilden ved fast innfallsvinkel på 20° (tittelen på grafen er feil: 30° skal altså være 20°). Figuren er hentet og gjengitt direkte fra prosjektbeskrivelsen [3] med tillatelse fra emneansvarlig.

For å sikre pålitelige vinkelestimater i praksis, bør derfor lydkilden plasseres minst 1 meter fra mikrofonarrayet. Da vil bølgefrontene i større grad være plane, og modellen som ligger til grunn for beregningen vil være gyldig.

5.3 Støy og robusthet i krysskorrelasjon

En interessant observasjon fra eksperimentene er at testene med bakgrunnsstøy (blue noise) ga den laveste variansen og standardavviket i vinkelestimatene. Dette kan virke kontraintuitivt, da støy vanligvis forbindes med redusert målenøyaktighet. Likevel stemmer resultatene godt med teorien for krysskorrelasjonsbasert måling. Krysskorrelasjon gir best resultat når signalet inneholder mange frekvenskomponenter, noe som er typisk for impulslyder og bredbåndsstøy som forklart i Seksjon 2.5.3. Slike signaler gir et korrelasjonsbilde med en markert topp og lav grunnstøy, noe som gjør det enklere å identifisere korrekt tidsforsinkelse mellom mikrofonene.

Det er spesielt relevant at signalet i støytesten hadde karakter av såkalt blue noise, som er en type støy med høyere energitetthet i de høye frekvensene og lite lavfrekvent innhold. Slike signaler er kjent for å gi tydelige og presise korrelasjonstopper [11, 12]. Dette kan forklare hvorfor systemet oppnådde høyest presisjon i målingene nettopp under bakgrunnsstøy.

I tillegg vil tilfeldig støy, så lenge den er jevnt fordelt i frekvens og ikke domineres av enkelte komponenter, ha lav sannsynlighet for å gi sterke krysskorrelasjoner over tid. Dette gir som resultat en markant topp i korrelasjonsfunksjonen rundt selve hendelsen (f.eks. klappet), og lav korrelasjonsverdi ellers. Det fører til et bedre signal-til-støy-forhold i krysskorrelasjonsbildet, og dermed mer stabile tidsforsinkelser.

Videre ble det ikke observert systematiske feil verken i tidsforsinkelser eller i beregnet vinkel i støytesten. Dette tyder på at målesystemet er robust mot jevn, bredbåndet bakgrunnsstøy, i hvert fall innenfor det testede frekvensområdet og det aktuelle signalnivået. Disse funnene støtter anbefalingene i prosjektbeskrivelsen [3], som påpeker at signaler med bredt frekvensinnhold, slik som støy og impulser, gir de beste resultatene ved bruk av krysskorrelasjon.

5.4 Vurdering av målenøyaktighet

Systemets evne til å estimere innfallsvinkel med høy presisjon og nøyaktighet varierer betydelig mellom de ulike testede retningene. Målingene under referanseforhold gir et godt grunnlag for å vurdere systemets ytelse, og resultatene presentert i Tabell 3 viser både styrker og svakheter ved tilnærmingen.

Ved vinklene 0°, 30°, 55°, 90°, 110°, 240° og 330° observeres lav standardavvik og varians, samt et gjennomsnittlig vinkelestimat som ligger nært opptil den forventede verdien. Dette indikerer at systemet i disse retningene fungerer relativt bra. Estimatene er konsistente, og tidsforsinkelsene målt med krysskorrelasjon gir stabile resultater som reflekterer de fysiske forholdene. Spesielt ved 55° og 110° er presisjonen svært god, med standardavvik under 3°, noe som demonstrerer at systemet har potensial til å oppnå høy målenøyaktighet under stabile forhold og enkelte innfallsvinkler.

I kontrast viser systemet betydelige svakheter ved enkelte vinkler, særlig 270° og 300°. Ved 270° er det estimerte gjennomsnittet hele 229.39°, altså over 40° feil, og standardavviket er så høyt som 13.087°. Dette representerer en alvorlig feil, kombinert med høy usikkerhet. Tilsvarende er målingene ved 300° preget av et gjennomsnittlig avvik på nesten 4°, med et standardavvik over 10°. Siden disse avvikene er konsistente over ni målinger under identiske forhold, tyder det på en systematisk feil snarere enn tilfeldig variasjon. Én mulig forklaring er at systemet blir forstyrret av refleksjoner eller akustiske asymmetrier i rommet, spesielt bakover mot 270°, hvor lyd kanskje treffer flater nær arrayet (se Seksjon 2.10.2). En annen mulighet er svakheter i systemets geometri eller i antagelsen om planbølgefront ved disse innfallsvinklene. En mer alvorlig, men plausibel forklaring er at én av mikrofonene har nedsatt ytelse eller svak følsomhet, noe som påvirker signalene når lydkilden treffer fra spesifikke retninger (se Seksjon 2.10.1). Dette kan forklare hvorfor avvikene er konsentrert i et bestemt vinkelområde, mens resten av målingene viser akseptabel presisjon. Det ble ikke gjennomført individuelle tester av mikrofonene, og mikrofonene ble heller ikke byttet ut i løpet av prosjektet, grunnet lav tilgjengelighet på reservekomponenter. Dermed kan en maskinvaresvakhet ikke utelukkes, og bør undersøkes videre.

Flere av vinkelestimatene går igjen med identiske desimalverdier, særlig ved 0°, 90° og 110°. Dette støttes av teorien i Seksjon 2.4, og skyldes at systemet er begrenset til et sett med diskrete tidsforsinkelser bestemt av samplingfrekvensen og mikrofonavstanden. Med $f_s = 31250$ Hz og mikrofonavstand på 8 cm er det kun 15 mulige verdier for forsinkelse (-7 til +7 samples), noe som igjen gir et tilsvarende begrenset sett med mulige vinkler.

Det er også verdt å merke seg at avvikene i flere tilfeller er systematiske. Ved 210°, for eksempel, estimeres gjennomsnittsvinkelen til 195.50°, altså en systematisk forskyvning på -14.5°, med samtidig høy varians. Dette antyder at feilkildene ikke kun skyldes de nevnte over, men at det også finnes underliggende strukturelle skjevheter i oppsettet eller analysen, for eksempel asymmetri i mikrofonplasseringen og refleksjoner i rommet.

5.5 Systemets begrensninger og mulige forbedringer

En potensiell årsak til de systematiske feilene observert i enkelte vinkelmålinger, spesielt ved 270° og 300°, kan være at én av mikrofonene i arrayet har svakere ytelse enn de andre. Dette kan skyldes redusert følsomhet, intern skade, dårlig loddepunkt, eller variasjoner i frekvensrespons. I et system som baserer seg på presise relative tidsforsinkelser, kan en slik variasjon gi opphav til feilaktige og ustabile vinkelestimater, spesielt dersom den aktuelle mikrofonen spiller en avgjørende rolle i estimeringen ved bestemte innfallsvinkler. Ved vinkler som 0° og 90° kan effekten av en svak mikrofon være mindre, avhengig av dens posisjon i arrayet. Men ved 270° og 300° er det tydelig at målingene er preget av både systematiske skjevheter og høy spredning, noe som kan indikere at signalet fra én mikrofon ikke oppfører seg som forventet i krysskorrelasjonen.

Det ble ikke gjennomført tester av mikrofonene enkeltvis for å vurdere deres individuelle ytelse, og det ble heller ikke forsøkt å bytte ut mikrofonene mellom målingene. Dette skyldes begrenset tilgang til ekstra komponenter og lavt antall tilgjengelige mikrofoner på laboratoriet. For å etterprøve denne hypotesen bør mikrofonene testes individuelt, for eksempel ved å sammenligne deres responser på identiske lydimpulser i kontrollert miljø. Et annet tiltak er å bytte plass på mikrofonene og gjenta testene. Dersom feilene følger mikrofonen og ikke posisjonen, er det et klart tegn på komponentfeil. En langsiktig forbedring for systemet vil være å gjennomføre kalibrering av mikrofonene for både amplitude og fase, eventuelt finne og bytte ut komponenter med dårlig ytelse. Dette kan bidra til å redusere systematiske avvik og forbedre robustheten til vinkelestimatene.

I tillegg bør det tas hensyn til refleksjoner fra omgivelsene. Målingene ble gjort i et vanlig rom uten akustisk demping, og refleksjoner fra vegger eller inventar kan ha påvirket signalene. For å minimere slike feil bør fremtidige målinger enten gjennomføres i store åpne rom med lite refleksjoner, eller i kontrollerte rom med akustikkplater som demper etterklang. Dette kan gjøre systemet mer robust mot feil som skyldes rommets geometri.

6 Konklusjon

Prosjektet har demonstrert hvordan et enkelt mikrofonarray kan benyttes for å estimere innfallsvinkelen til en lydkilde basert på krysskorrelasjon av signaler. Under gunstige forhold gir systemet relativt presise vinkelestimater, med avvik på under 2° og lav variasjon. Dette viser at metoden fungerer i praksis og kan gi pålitelige resultater.

Samtidig har prosjektet avdekket klare svakheter ved enkelte innfallsvinkler, spesielt rundt 270° og 300°, der både systematiske skjevheter og høy varians oppstår. Slike avvik tyder på begrensninger i systemets vinkeloppløsning, mulige reflekser i målemiljøet, og sannsynligvis også ytelsesforskjeller mellom mikrofonene. Dette fremhever behovet for komponenttesting og kalibrering.

Til tross for disse utfordringene gir prosjektet et godt grunnlag for videre arbeid. Med forbedret oppløsning, kalibrering og kontrollert målemiljø har systemet potensial til å bli mer robust. Prosjektet viser viktigheten av både teoretisk modellforståelse og praktisk feilsøking i utvikling av målesystemer.

Referanser

[1] Merritt, David: *Omnidirectional Microphone vs. Unidirectional Mic*, 2024. https://headsetadvisor.com/blogs/headset/omnidirectional-vs-unidirectional-microphones-a-comprehensive-guide?srsltid=AfmBOorsEj-BPu6IUsmW3aOak-pVzdBoSNxKMIBCIlPu-mqArFBgUryn.

[2] CUI Devices: *CMA-4544PF-W Electret Condenser Microphone Datasheet*. https://cdn-shop.adafruit.com/datasheets/Adafruit_1063_eng_tds.pdf, 2008. Accessed: 2025-04-24.

[3] *TTT4280 Sensors and Instrumentation Lab 2*, 2025. NTNU – Institutt for teknisk kybernetikk.

[4] Hioka, Yusuke og Yasunobu Hamada: *Direction of Arrival Estimation of Speech Signals Using Microphones Located at the Vertices of an Equilateral Triangle*. I *Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH)*, Geneva, Switzerland, 2003. https://www.isca-archive.org/eurospeech_2003/hioka03_eurospeech.html, Accessed: 2025-04-24.

[5] Scola, Carlos Fernández og María Dolores Bolaños Ortega: *Direction of arrival estimation – A two microphones approach*. Blekinge, Sweden, 2010. https://www.diva-portal.org/smash/get/diva2%3A830430/FULLTEXT01.pdf, Accessed: 2025-04-24.

[6] Malinverno, Matteo: *Analog-to-digital conversion (ADC): What it is and how it works*, 2020. https://splice.com/blog/analog-to-digital-conversion/.

[7] Inc., Microchip Technology: *MCP3201 2.7V 12-Bit A/D Converter with SPI Serial Interface*, November 2002. https://ww1.microchip.com/downloads/en/devicedoc/21290f.pdf, Document No. DS21290F.

[8] Lewis, Jerad: *Understanding Microphone Sensitivity*, 2025. https://www.analog.com/en/resources/analog-dialogue/articles/understanding-microphone-sensitivity.html.

[9] NTNU: *TTT4280 - Sensors and Instrumentation Lab Manual*, 2025. Lastet ned fra Blackboard: January 9, 2025.

[10] Truls Østvedt, Arya Raeesi: *Lab 1: Måleoppsett*, 2025.

[11] contributors, Wikipedia: *Blue noise — Wikipedia, The Free Encyclopedia*, 2024. https://en.wikipedia.org/wiki/Blue_noise, Accessed: 2025-04-23.

[12] Huijben, C. og A. O. Hero: *Blue-Noise Sampling of Signals on Graphs*. Proceedings of the IEEE SAMPTA, 2019. https://www.researchgate.net/publication/339903944_Blue-Noise_Sampling_of_Signals_on_Graphs, Accessed: 2025-04-23.

A Utfyllende plott

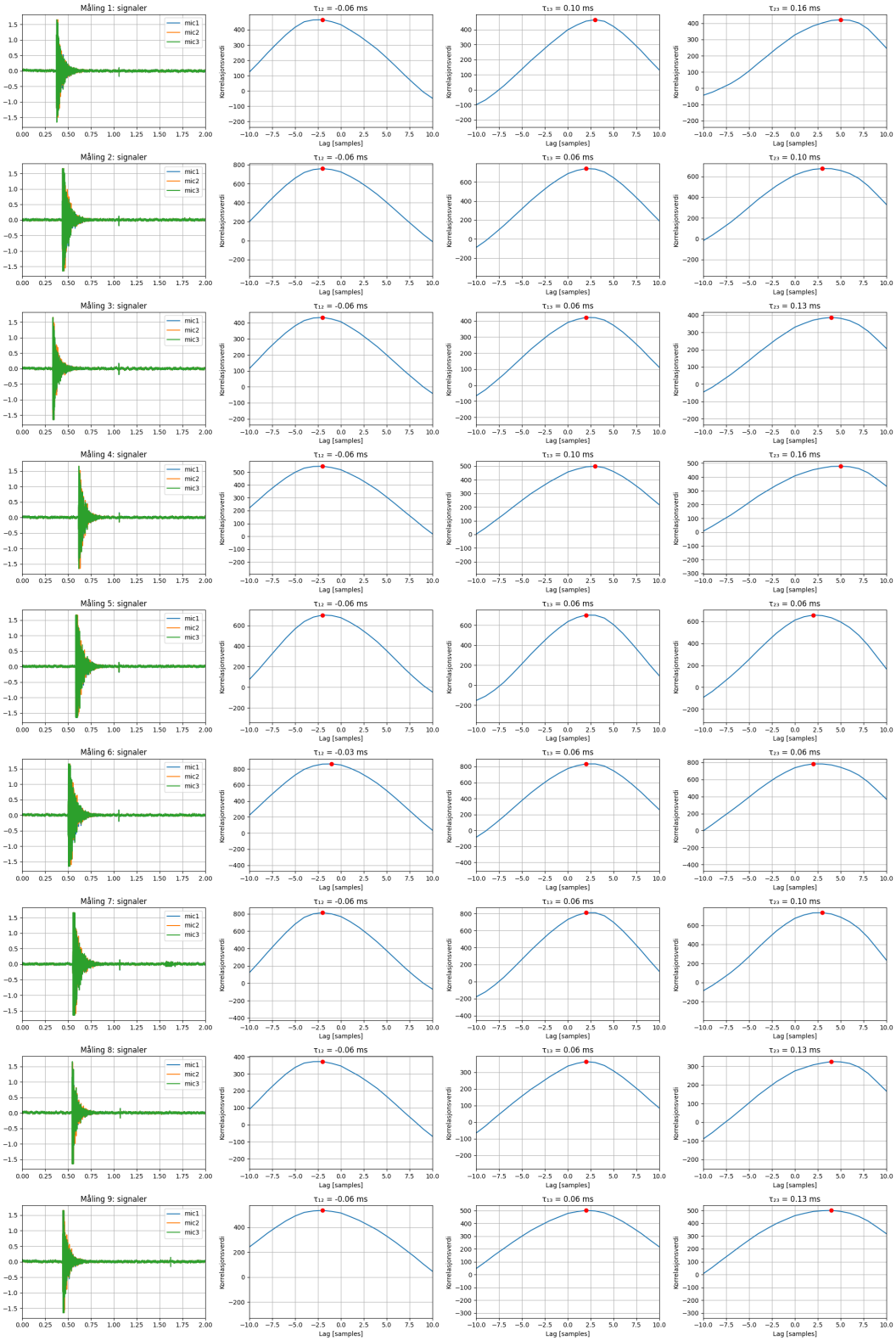
A.1 Normale forhold

Plottene under viser et mer detaljert datagrunnlag enn det som er inkludert i hoveddelen av rapporten. De er utelatt fra hovedteksten på grunn av plassbegrensninger og fordi de går dypere enn nødvendig for å støtte rapportens hovedkonklusjoner. I hovedteksten presenterer vi én representativ måling per eksperiment, mens figurene under viser samtlige målinger for hver vinkel. Dette gir interesserte lesere mulighet til å studere hele datasettet og se variasjoner mellom målingene.

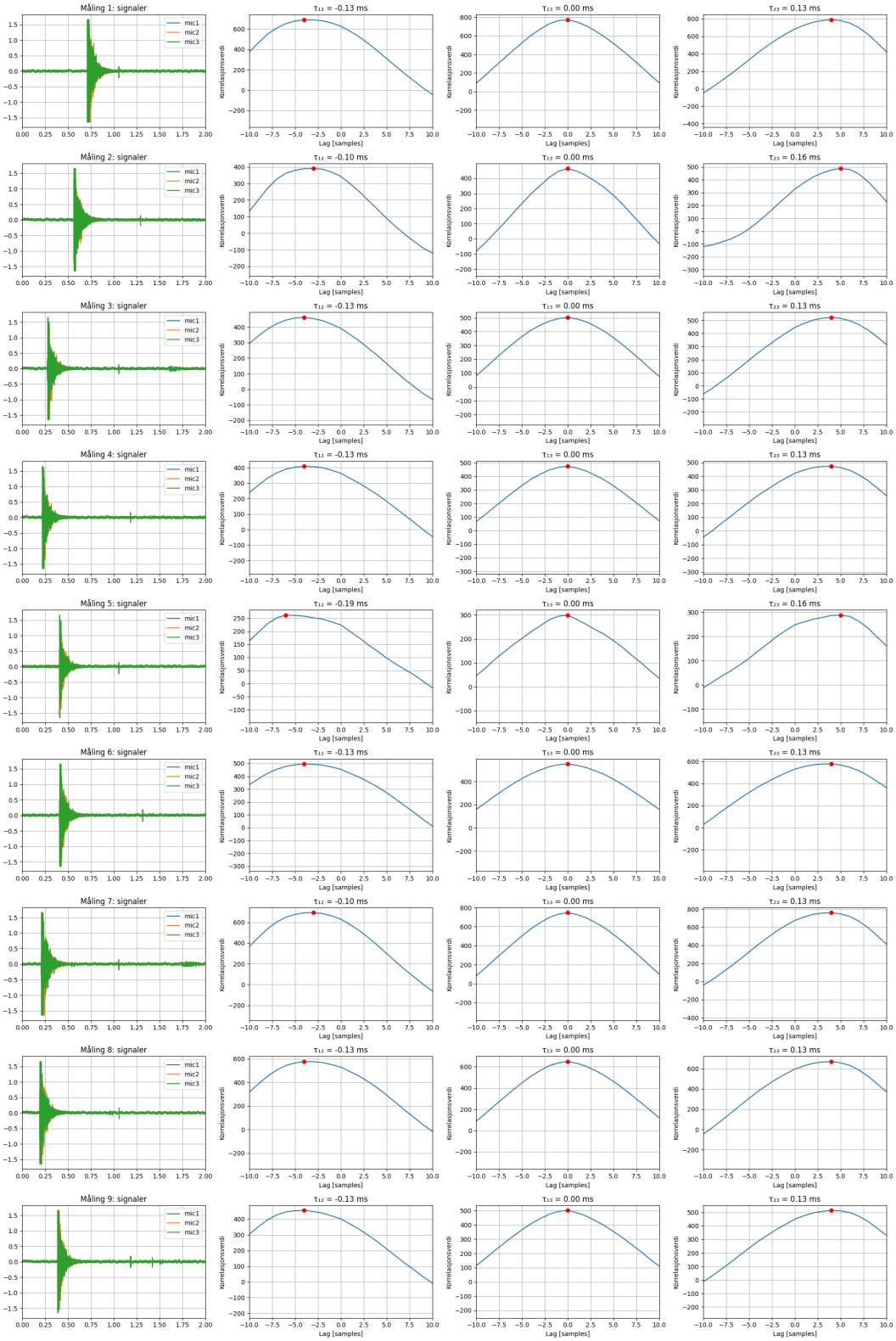
Hver figur representerer én vinkel: 0°, 30°, 55°, 90°, 110°, 130°, 160°, 210°, 240°, 270°, 300° og 330°. Hver figur består av 9 rader (én per måling) og 4 kolonner:

- Kolonne 1: Råsignaler fra de tre mikrofonene
- Kolonne 2–4: Krysskorrelasjoner mellom mikrofonparene (1–2, 1–3, 2–3)

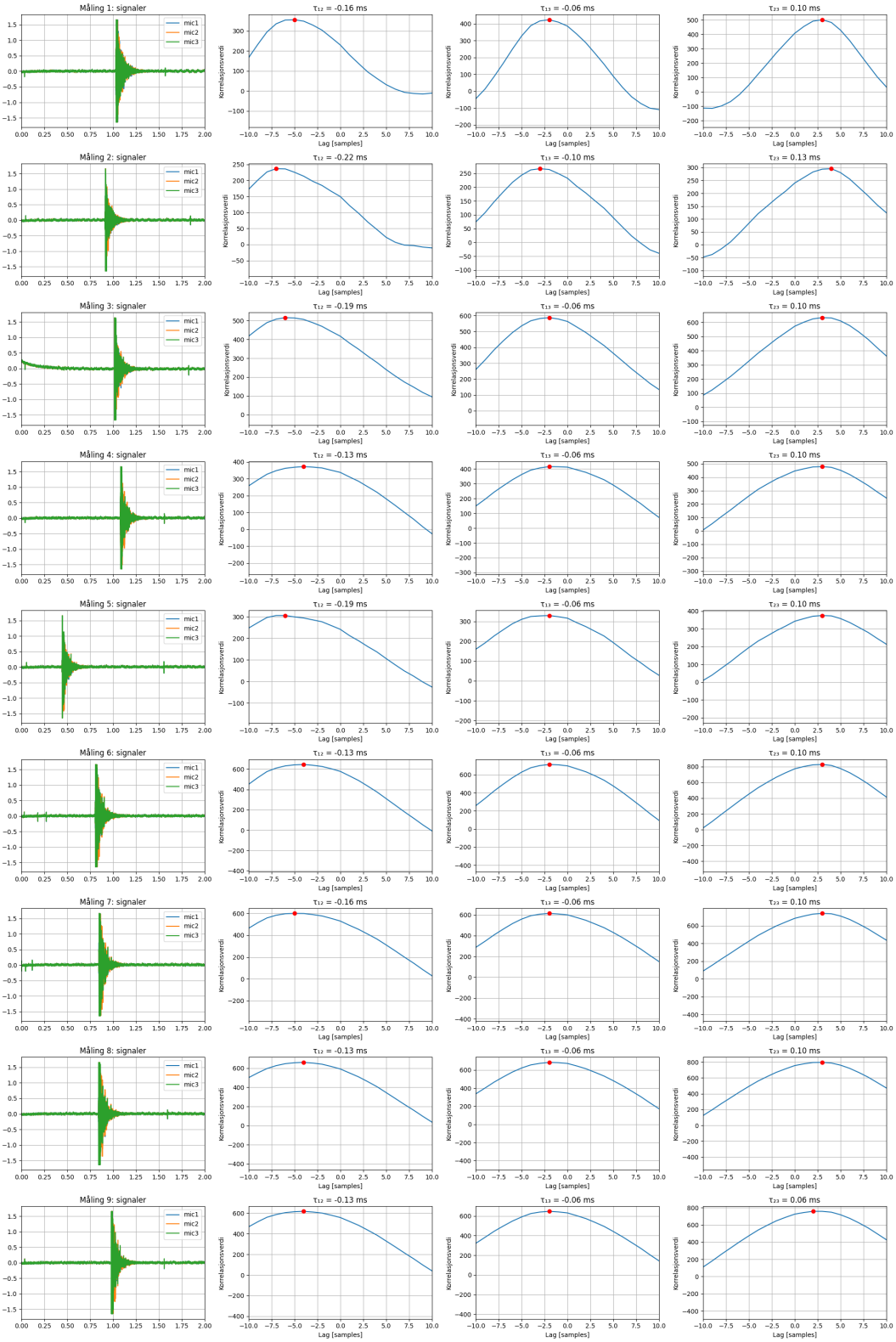
Toppen i krysskorrelasjonen er markert i rødt og benyttes til å finne τ_{ij}, tidsforsinkelsen mellom mikrofon i og j. Alle målingene her ble gjort med 8 cm avstand mellom mikrofonene og 1 meter avstand til lydkilden.



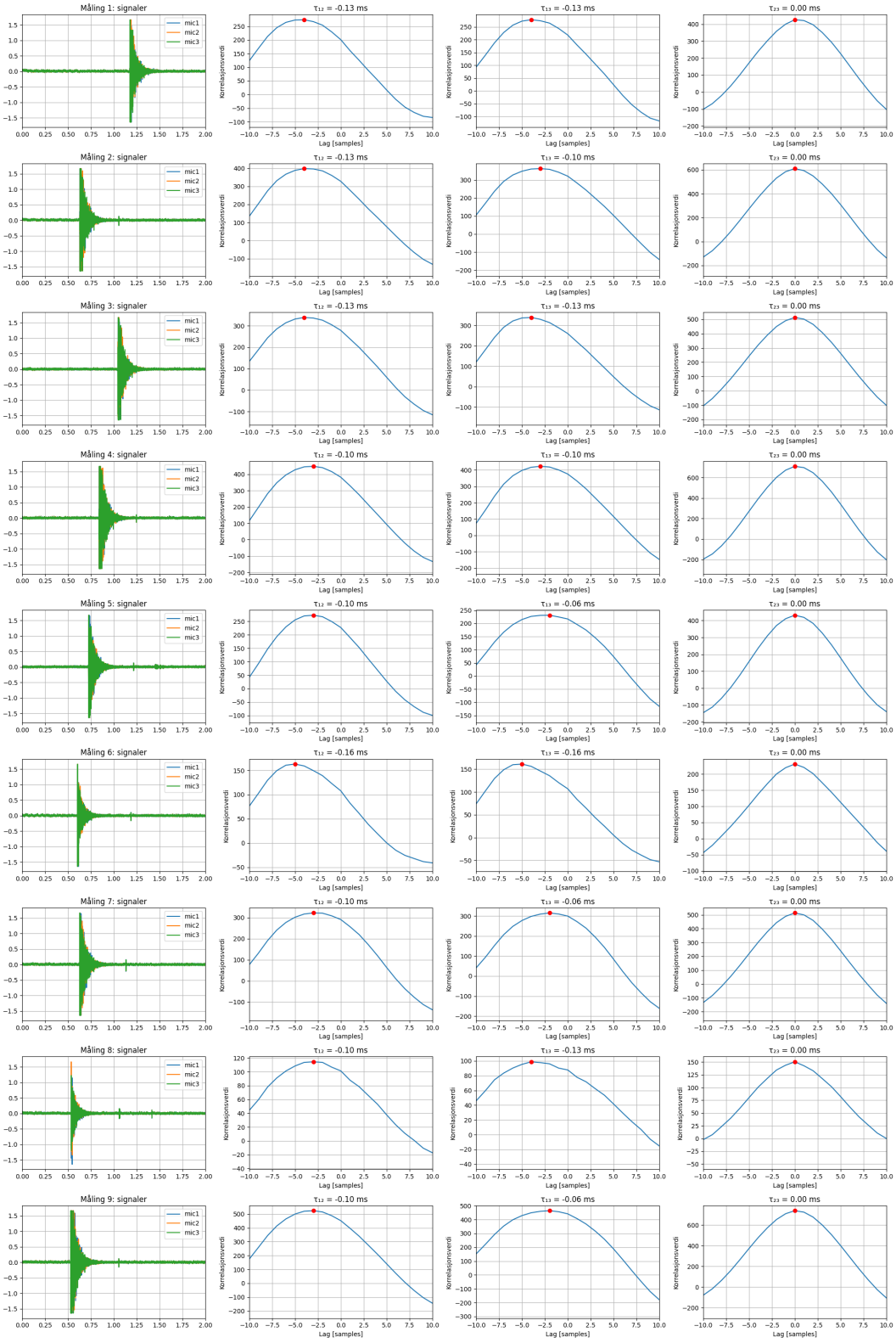
Figur 22: Alle 9 målinger ved 0° vinkel under normale forhold.



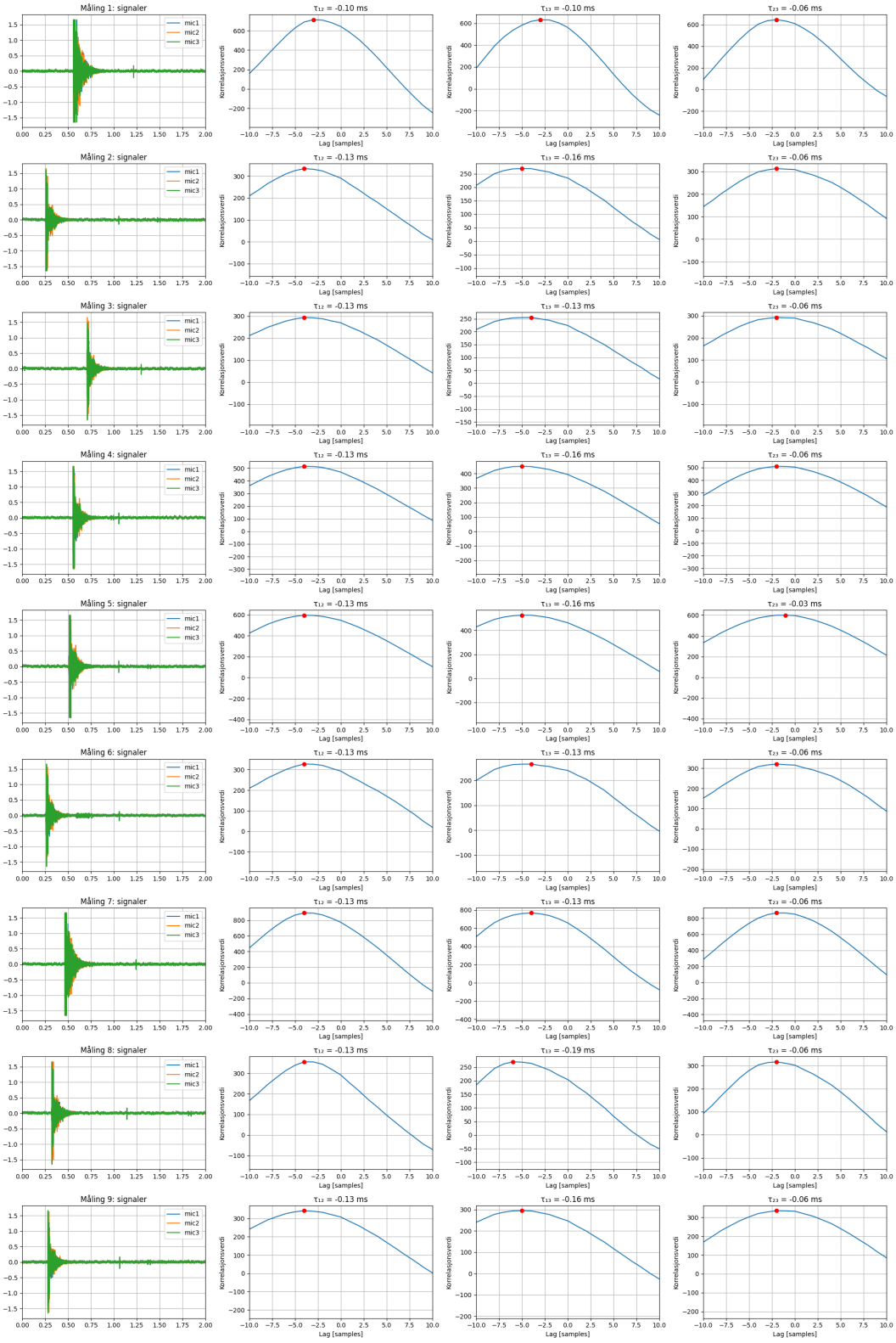
Figur 23: Alle 9 målinger ved 30° vinkel under normale forhold.



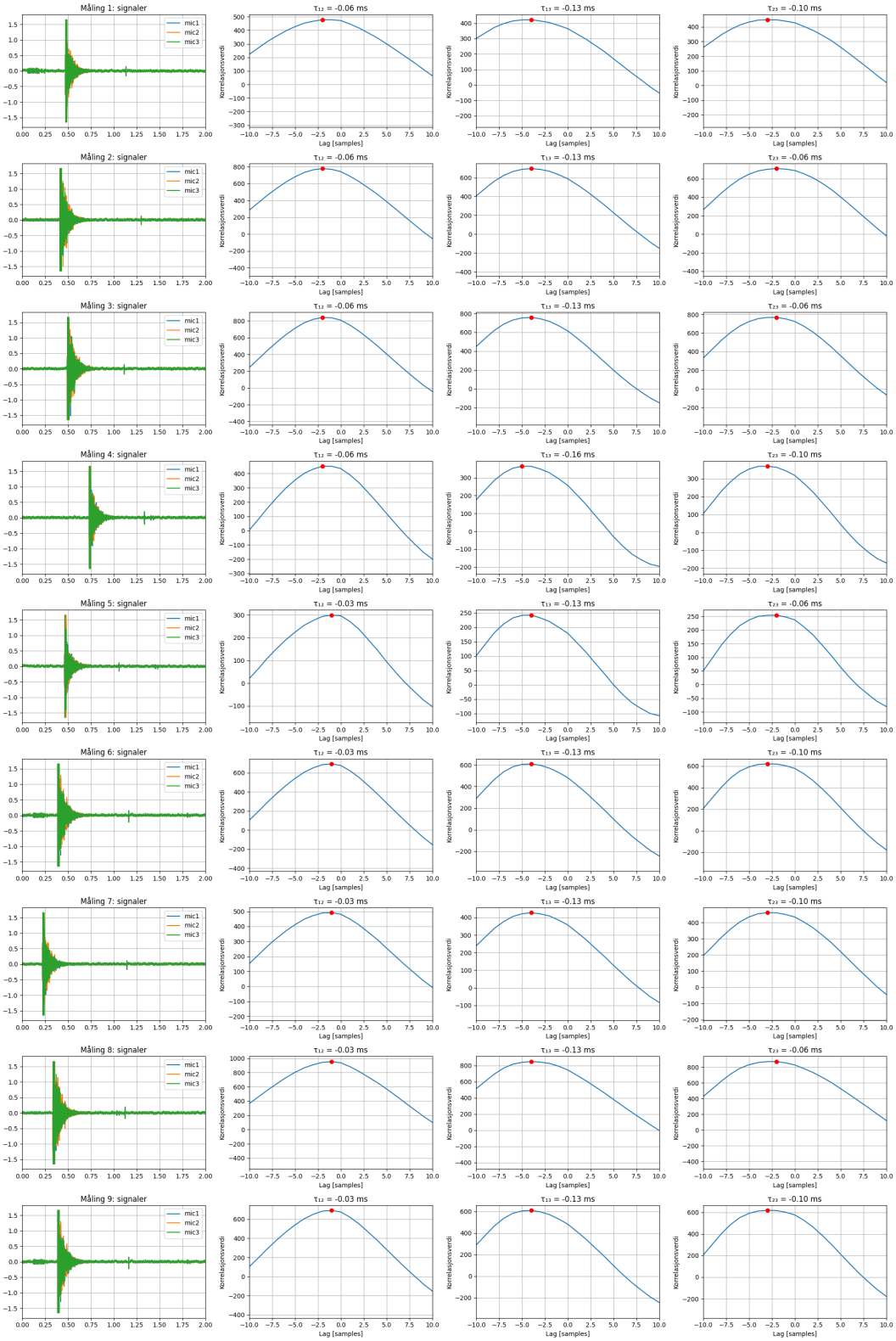
Figur 24: Alle 9 målinger ved 55° vinkel under normale forhold.



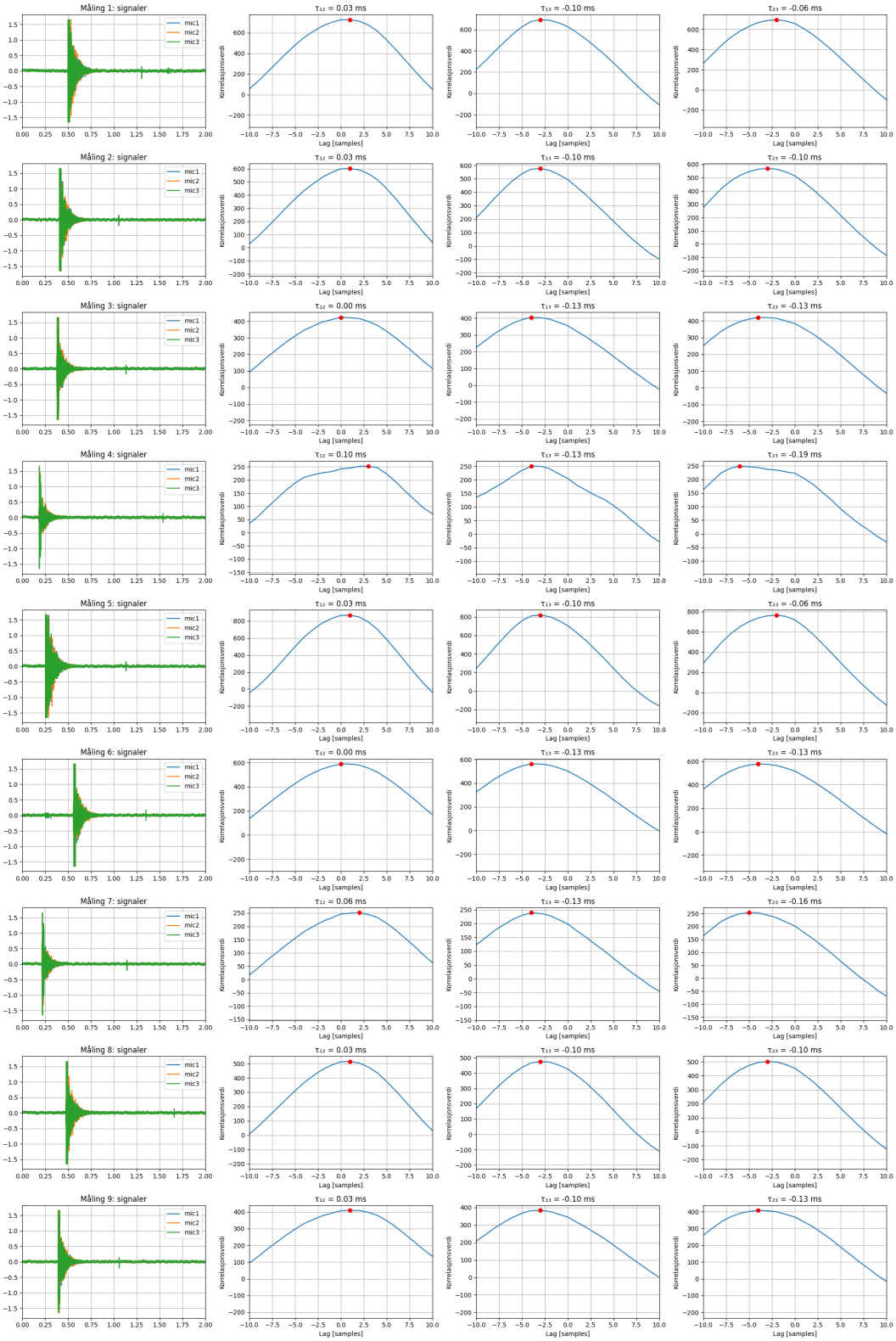
Figur 25: Alle 9 målinger ved 90° vinkel under normale forhold.



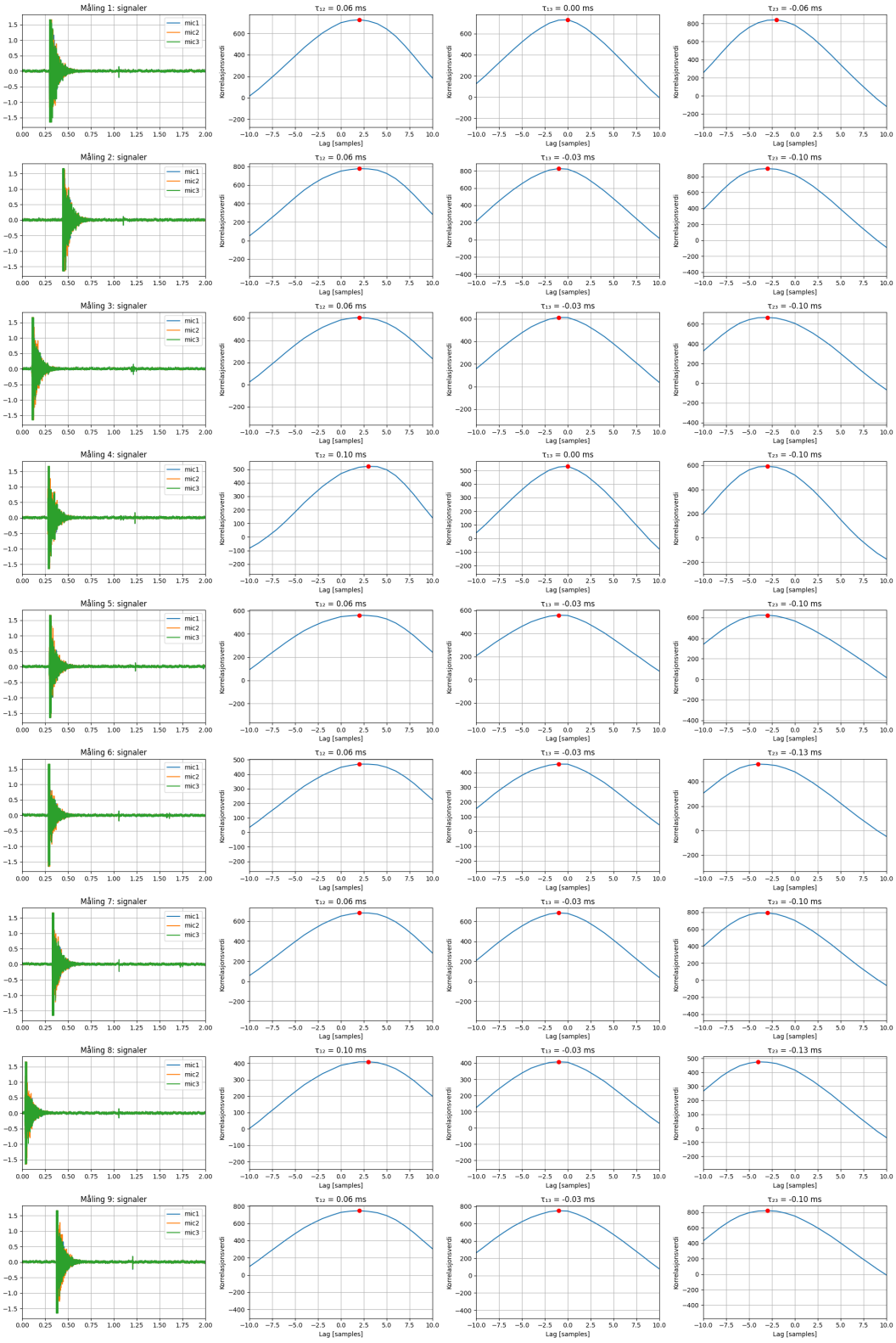
Figur 26: Alle 9 målinger ved 110° vinkel under normale forhold.



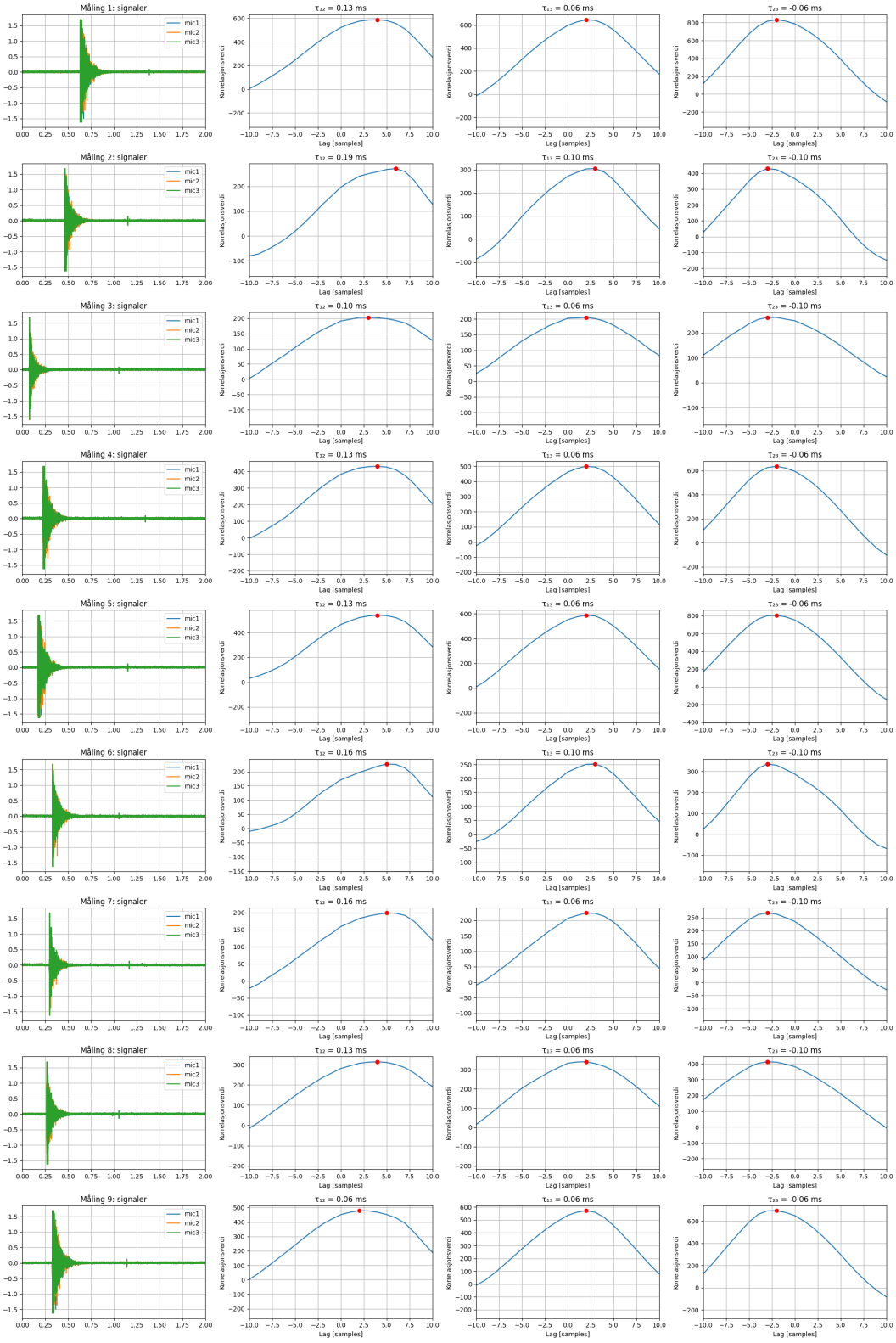
Figur 27: Alle 9 målinger ved 130° vinkel under normale forhold.



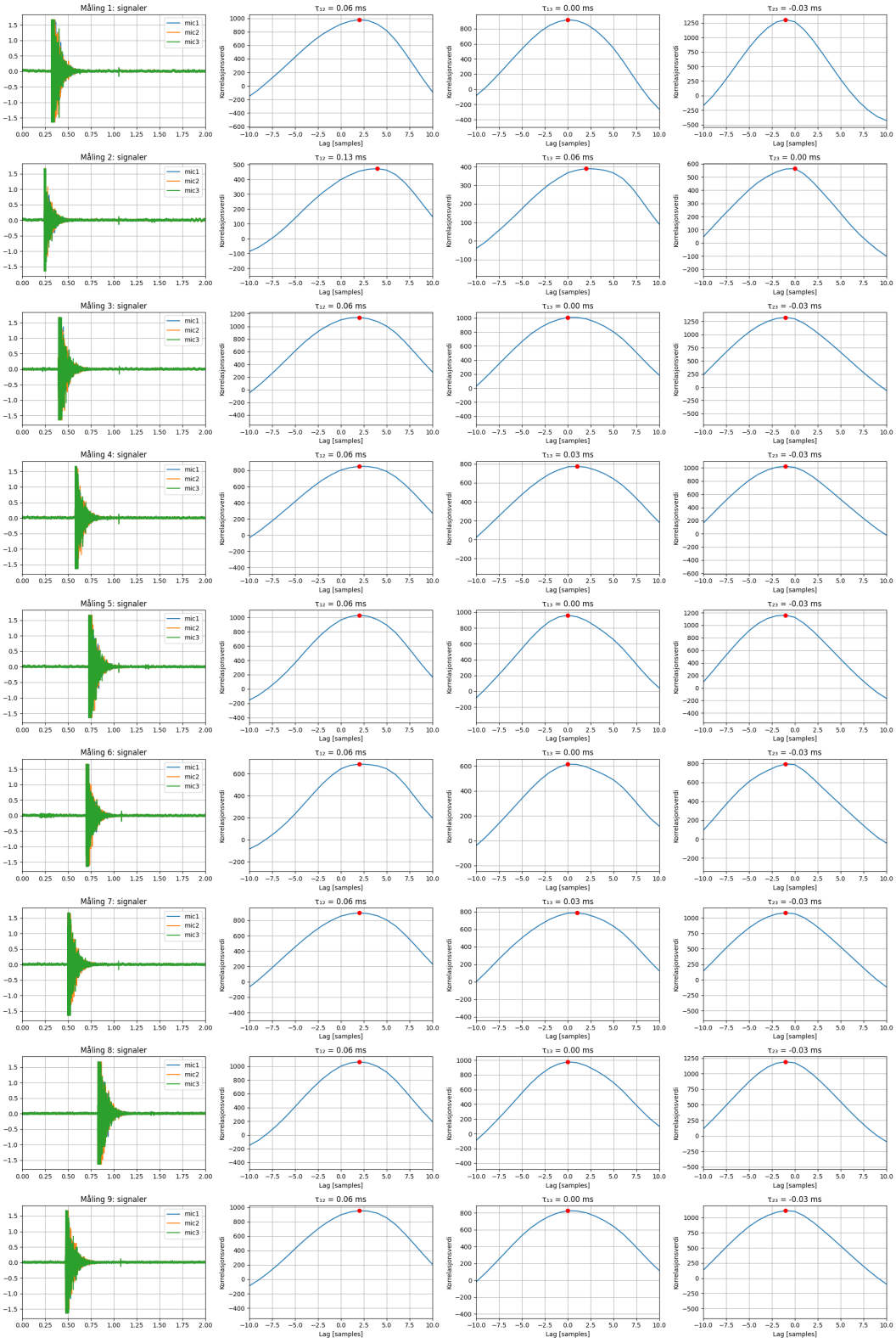
Figur 28: Alle 9 målinger ved 160° vinkel under normale forhold.



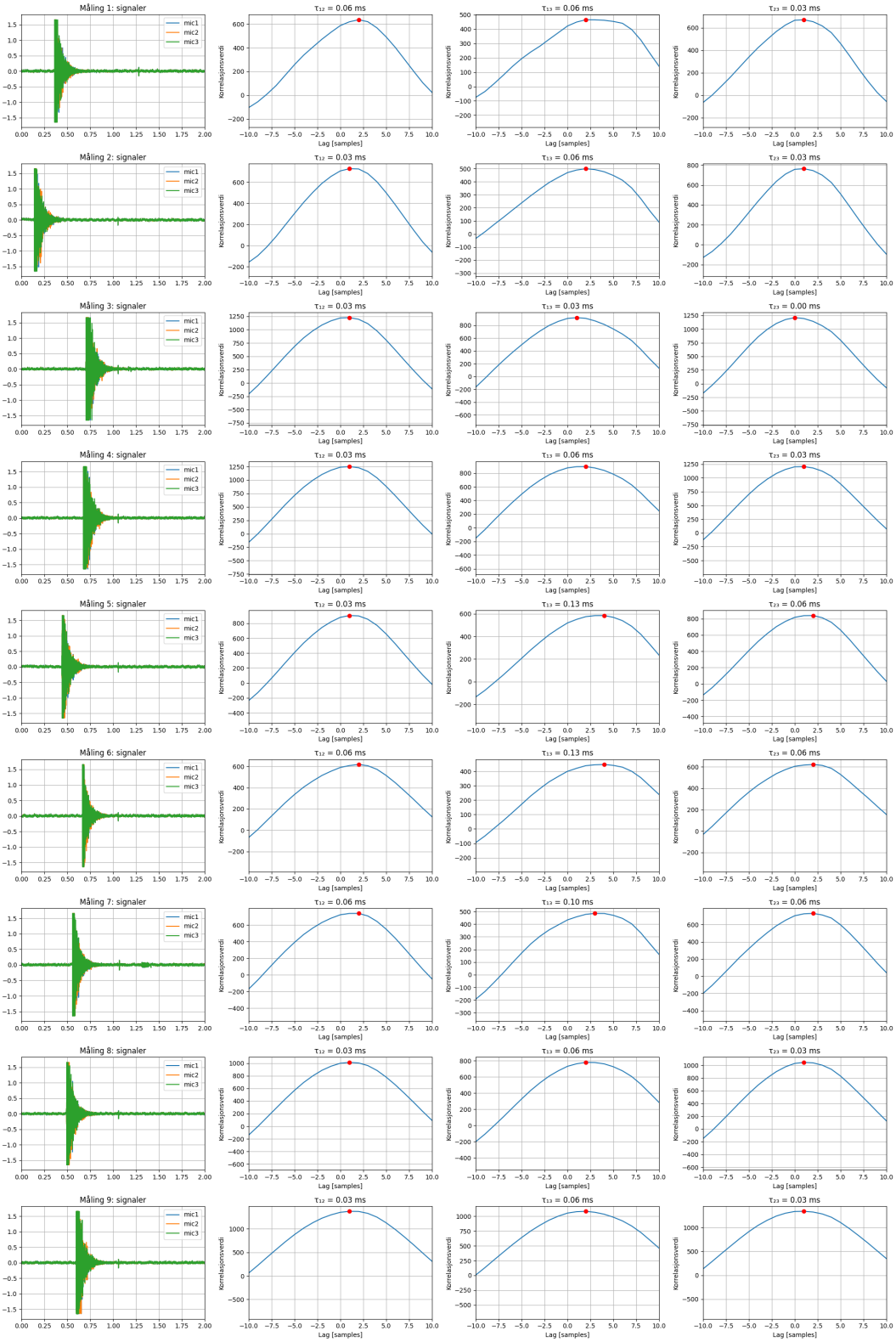
Figur 29: Alle 9 målinger ved 210° vinkel under normale forhold.



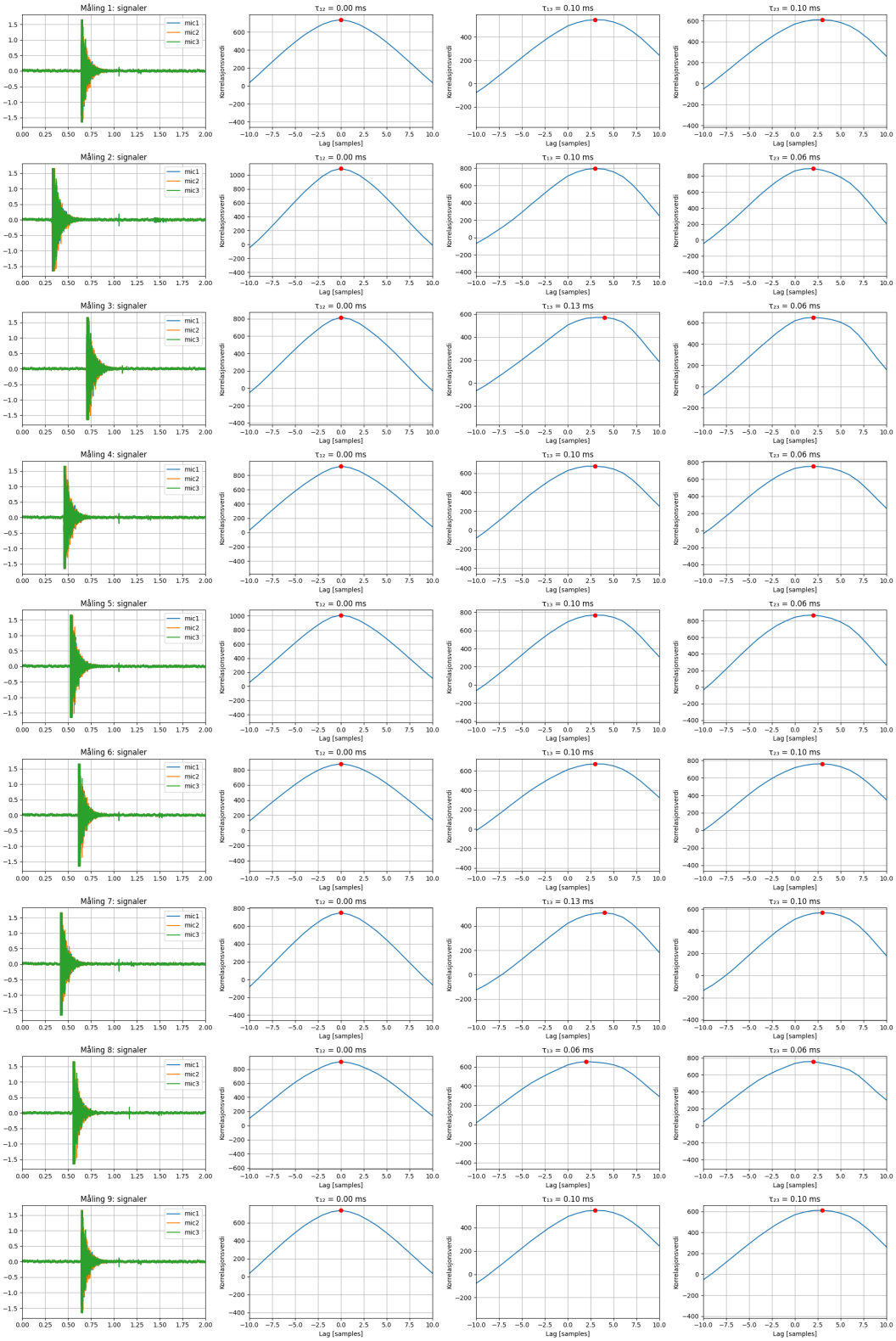
Figur 30: Alle 9 målinger ved 240° vinkel under normale forhold.



Figur 31: Alle 9 målinger ved 270° vinkel under normale forhold.



Figur 32: Alle 9 målinger ved 300° vinkel under normale forhold.

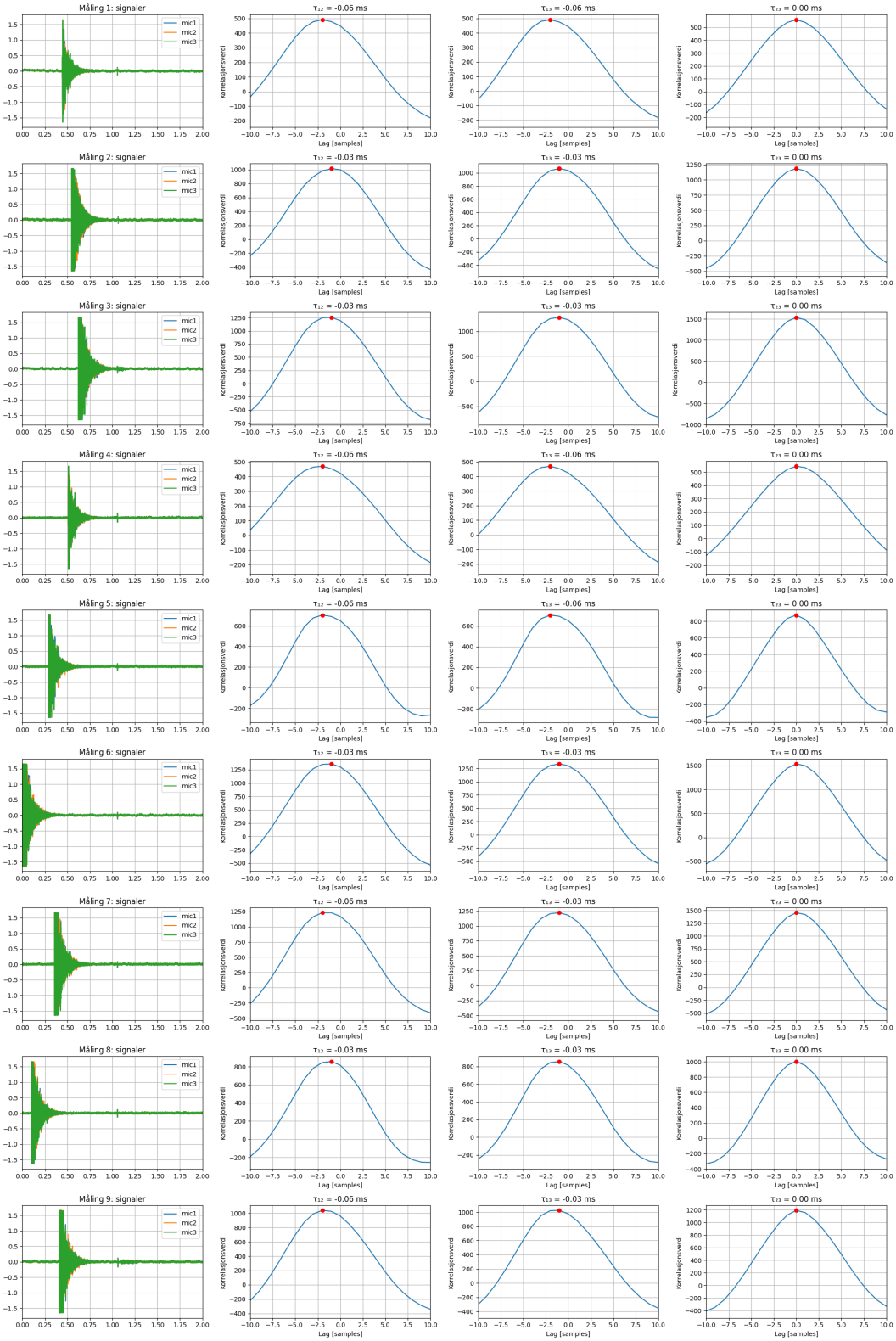


Figur 33: Alle 9 målinger ved 330° vinkel under normale forhold.

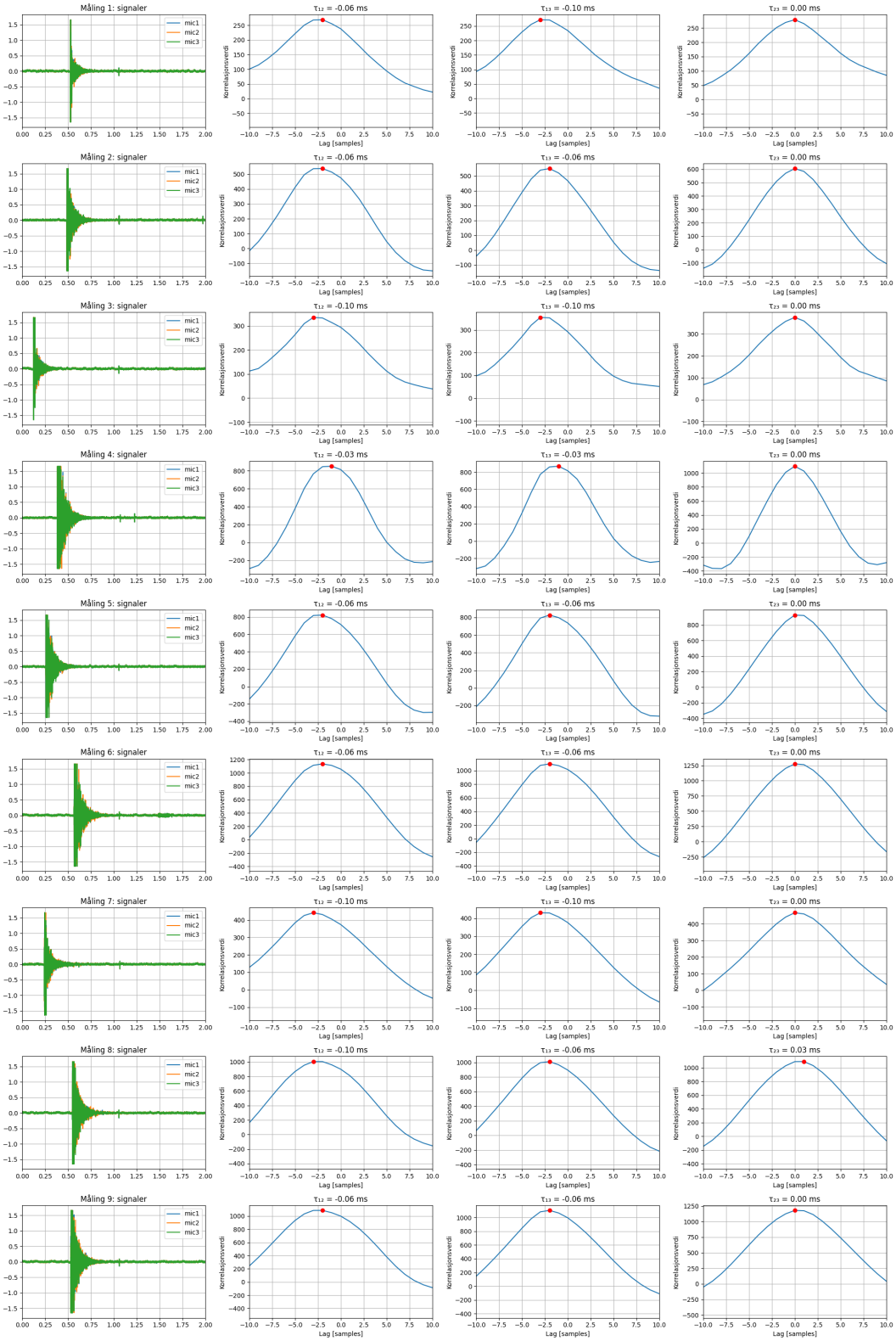
A.2 Varierende forhold

Her følger plott fra målinger gjort under endrede forhold. I alle tilfellene var lydkilden plassert ved 90°.

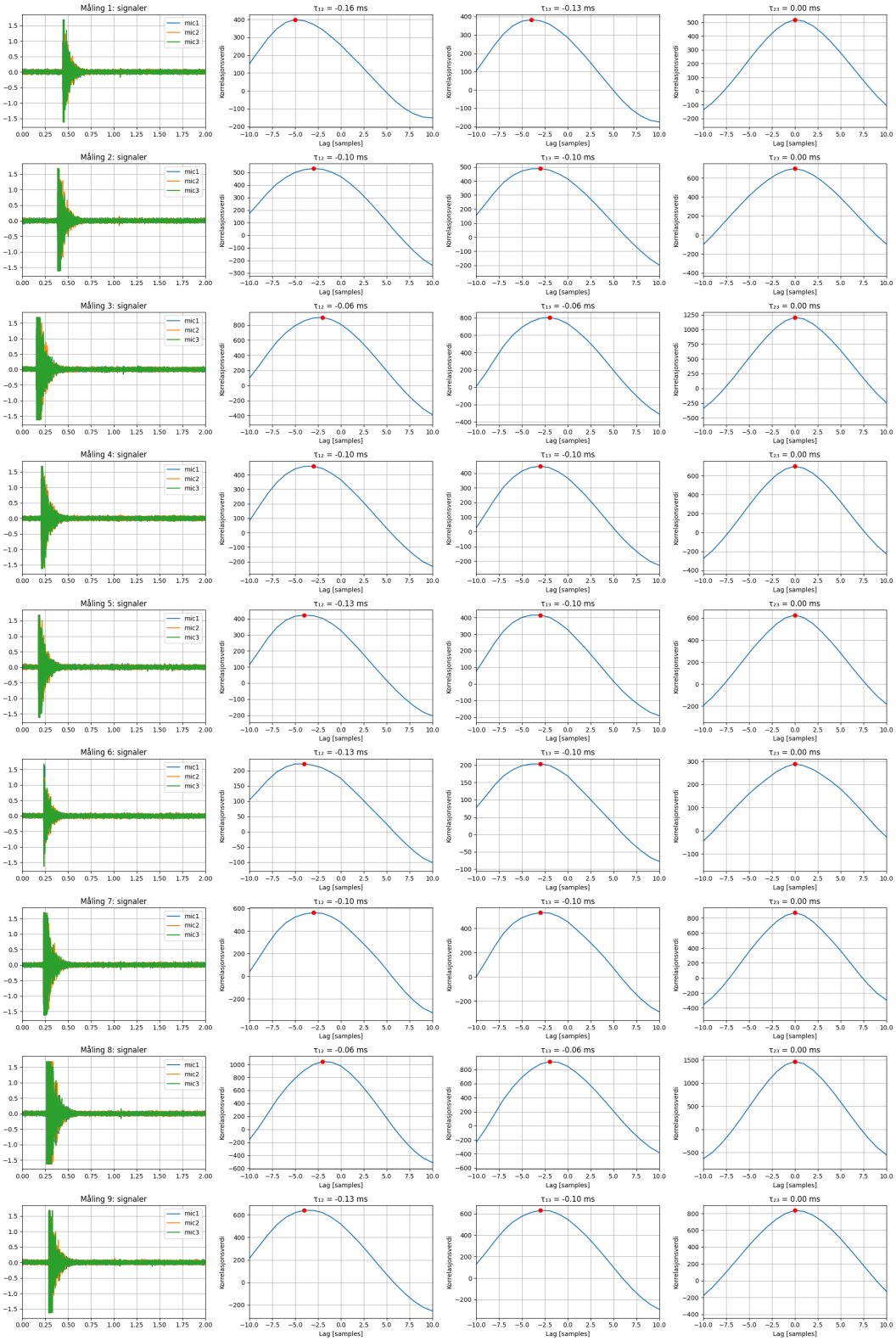
- Første måling: mikrofonavstand redusert til 4 cm (fra 8 cm).
- Andre måling: mikrofonavstand 4 cm og lydkilden flyttet nærmere (10 cm).
- Tredje måling: konstant bakgrunnsstøy i form av blue noise"under hele målingen.



Figur 34: Måling ved 90° med 4 cm avstand mellom mikrofonene.



Figur 35: Måling ved 90° med 4 cm mikrofonavstand og kortere avstand til lydkilde (10 cm).



Figur 36: Måling ved 90° med konstant bakgrunnsstøy (blue noise).

B Kode

Her følger fullstendig kode brukt i prosjektet. Enkelte kodesnutter er hentet og justert fra github, Blackboard og labmanualen [9].

B.1 C kode

```
/*
adc_sampler.c
Public Domain
January 2018, Kristoffer KjÃfÂ¦rnes & Asgeir BjÃfÂ¸rgan
Based on example code from the pigpio library by Joan @ raspi forum and github
https://github.com/joan2937 | http://abyz.me.uk/rpi/pigpio/

Compile with:
gcc -Wall -lpthread -o adc_sampler adc_sampler.c -lpigpio -lm

Run with:
sudo ./adc_sampler

This code bit bangs SPI on several devices using DMA.

Using DMA to bit bang allows for two advantages
1) the time of the SPI transaction can be guaranteed to within a
   microsecond or so.

2) multiple devices of the same type can be read or written
   simultaneously.

This code reads several MCP3201 ADCs in parallel, and writes the data to a binary file.
Each MCP3201 shares the SPI clock and slave select lines but has
a unique MISO line. The MOSI line is not in use, since the MCP3201 is single
channel ADC without need for any input to initiate sampling.
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#include <pigpio.h>
#include <math.h>
#include <time.h>
```

```
36 /////// USER SHOULD MAKE SURE THESE DEFINES CORRESPOND TO THEIR SETUP ///////
37 #define ADCS 5 // Number of connected MCP3201.
38
39 #define OUTPUT_DATA argv[2] // path and filename to dump buffered ADC data
40
41 /* RPi PIN ASSIGNMENTS */
42 #define MISO1 18 // ADC 1 MISO (GPIO pin number)
43 #define MISO2 19 // ADC 2 MISO (GPIO pin number)
44 #define MISO3 20 // ADC 3 MISO (GPIO pin number)
45 #define MISO4 21 // ADC 4 MISO (GPIO pin number)
46 #define MISO5 22 // ADC 5 MISO (GPIO pin number)
47
48 #define MOSI 10   // GPIO for SPI MOSI (GPIO 10 aka SPI_MOSI). MOSI not in use here due to single ch. ADCs, but must be defined anyway.
49 #define SPI_SS 15 // GPIO for slave select (GPIO 15).
50 #define CLK 11    // GPIO for SPI clock (GPIO 16).
51 /* END RPi PIN ASSIGNMENTS */
52
53 #define BITS 12             // Bits per sample.
54 #define BX 4                // Bit position of data bit B11. (3 first are t_sample + null bit)
55 #define B0 (BX + BITS - 1) // Bit position of data bit B0.
56
57 #define NUM_SAMPLES_IN_BUFFER 300 // Generally make this buffer as large as possible in order to cope with reschedule.
58
59 #define REPEAT_MICROS 32 // Reading every x microseconds. Must be no less than 2xB0 defined above
60
61 #define DEFAULT_NUM_SAMPLES 31250 // Default number of samples for printing in the example. Should give 1sec of data at Tp=32us.
62
63 int MISO[ADCS] = {MISO1, MISO2, MISO3, MISO4, MISO5}; // Must be updated if you change number of ADCs/MISOs above
64 /////// END USER SHOULD MAKE SURE THESE DEFINES CORRESPOND TO THEIR SETUP ///////
65
66 /**
67  * This function extracts the MISO bits for each ADC and
68  * collates them into a reading per ADC.
69  *
70  * \param adcs Number of attached ADCs
71  * \param MISO The GPIO connected to the ADCs data out
72  * \param bytes Bytes between readings
73  * \param bits Bits per reading
74  * \param buf Output buffer
```

```
75   */
76  void getReading(int adcs, int *MISO, int OOL, int bytes, int bits, char
    ↪  *buf)
77  {
78      int p = OOL;
79      int i, a;
80
81      for (i = 0; i < bits; i++)
82      {
83          uint32_t level = rawWaveGetOut(p);
84          for (a = 0; a < adcs; a++)
85          {
86              putBitInBytes(i, buf + (bytes * a), level & (1 << MISO[a]));
87          }
88          p--;
89      }
90  }
91
92  int main(int argc, char *argv[])
93  {
94      // Parse command line arguments
95      long num_samples = 0;
96      if (argc <= 1)
97      {
98          fprintf(stderr, "Usage: %s NUM_SAMPLES\n\n"
99                          "Example: %s %d\n",
100                 argv[0], argv[0], DEFAULT_NUM_SAMPLES);
101         exit(1);
102     }
103     sscanf(argv[1], "%ld", &num_samples);
104
105     // Array over sampled values, into which data will be saved
106     uint16_t *val = (uint16_t *)malloc(sizeof(uint16_t) * num_samples *
    ↪  ADCS);
107
108     // SPI transfer settings, time resolution 1us (1MHz system clock is
    ↪  used)
109     rawSPI_t rawSPI =
110         {
111             .clk = CLK,   // Defined before
112             .mosi = MOSI, // Defined before
113             .ss_pol = 1,  // Slave select resting level.
114             .ss_us = 1,   // Wait 1 micro after asserting slave select.
115             .clk_pol = 0, // Clock resting level.
116             .clk_pha = 0, // 0 sample on first edge, 1 sample on second
                ↪  edge.
117             .clk_us = 1,  // 2 clocks needed per bit so 500 kbps.
```

```
118         };
119
120     // Change timer to use PWM clock instead of PCM clock. Default is PCM
121     // clock, but playing sound on the system (e.g. espeak at boot) will
        ↪  start
122     // sound systems that will take over the PCM timer and make
        ↪  adc_sampler.c
123     // sample at far lower samplerates than what we desire.
124     // Changing to PWM should fix this problem.
125     gpioCfgClock(5, 0, 0);
126
127     // Initialize the pigpio library
128     if (gpioInitialise() < 0)
129     {
130         return 1;
131     }
132
133     // Set the selected CLK, MOSI and SPI_SS pins as output pins
134     gpioSetMode(rawSPI.clk, PI_OUTPUT);
135     gpioSetMode(rawSPI.mosi, PI_OUTPUT);
136     gpioSetMode(SPI_SS, PI_OUTPUT);
137
138     // Flush any old unused wave data.
139     gpioWaveAddNew();
140
141     // Construct bit-banged SPI reads. Each ADC reading is stored
        ↪  separatedly
142     // along a buffer of DMA commands (control blocks). When the DMA engine
143     // reaches the end of the buffer, it restarts on the start of the
        ↪  buffer
144     int offset = 0;
145     int i;
146     char buf[2];
147     for (i = 0; i < NUM_SAMPLES_IN_BUFFER; i++)
148     {
149         buf[0] = 0xC0; // Start bit, single ended, channel 0.
150
151         rawWaveAddSPI(&rawSPI, offset, SPI_SS, buf, 2, BX, B0, B0);
152         offset += REPEAT_MICROS;
153     }
154
155     // Force the same delay after the last command in the buffer
156     gpioPulse_t final[2];
157     final[0].gpioOn = 0;
158     final[0].gpioOff = 0;
159     final[0].usDelay = offset;
160
```

```
161     final[1].gpioOn = 0; // Need a dummy to force the final delay.
162     final[1].gpioOff = 0;
163     final[1].usDelay = 0;
164 
165     gpioWaveAddGeneric(2, final);
166 
167     // Construct the wave from added data.
168     int wid = gpioWaveCreate();
169     if (wid < 0)
170     {
171         fprintf(stderr, "Can't create wave, buffer size %d too large?\n",
            ↪  NUM_SAMPLES_IN_BUFFER);
172         return 1;
173     }
174 
175     // Obtain addresses for the top and bottom control blocks (CB) in the
        ↪  DMA
176     // output buffer. As the wave is being transmitted, the current CB will
        ↪  be
177     // between botCB and topCB inclusive.
178     rawWaveInfo_t rwi = rawWaveInfo(wid);
179     int botCB = rwi.botCB;
180     int topOOL = rwi.topOOL;
181     float cbs_per_reading = (float)rwi.numCB /
        ↪  (float)NUM_SAMPLES_IN_BUFFER;
182 
183     float expected_sample_freq_khz = 1000.0 / (1.0 * REPEAT_MICROS);
184 
185     printf("# Starting sampling: %ld samples (expected Tp = %d us, expected
        ↪  Fs = %.3f kHz).\n",
186            num_samples, REPEAT_MICROS, expected_sample_freq_khz);
187 
188     // Start DMA engine and start sending ADC reading commands
189     gpioWaveTxSend(wid, PI_WAVE_MODE_REPEAT);
190 
191     // Read back the samples
192     double start_time = time_time();
193     int reading = 0;
194     int sample = 0;
195 
196     while (sample < num_samples)
197     {
198         // Get position along DMA control block buffer corresponding to the
            ↪  current output command.
199         int cb = rawWaveCB() - botCB;
200         int now_reading = (float)cb / cbs_per_reading;
201 
```

```
202         while ((now_reading != reading) && (sample < num_samples))
203         {
204             // Read samples from DMA input buffer up until the current
                ↪  output command
205 
206             // OOL are allocated from the top down. There are BITS bits for
                ↪  each ADC
207             // reading and NUM_SAMPLES_IN_BUFFER ADC readings. The readings
                ↪  will be
208             // stored in topOOL - 1 to topOOL - (BITS *
                ↪  NUM_SAMPLES_IN_BUFFER).
209             // Position of each reading's OOL are calculated relative to
                ↪  the wave's top
210             // OOL.
211             int reading_address = topOOL - ((reading %
                ↪  NUM_SAMPLES_IN_BUFFER) * BITS) - 1;
212 
213             char rx[8];
214             getReading(ADCS, MISO, reading_address, 2, BITS, rx);
215 
216             // Convert and save to output array
217             for (i = 0; i < ADCS; i++)
218             {
219                 val[sample * ADCS + i] = (rx[i * 2] << 4) + (rx[(i * 2) +
                    ↪  1] >> 4);
220             }
221 
222             ++sample;
223 
224             if (++reading >= NUM_SAMPLES_IN_BUFFER)
225             {
226                 reading = 0;
227             }
228         }
229         usleep(1000);
230     }
231 
232     double end_time = time_time();
233 
234     double nominal_period_us = 1.0 * (end_time - start_time) / (1.0 *
    ↪  num_samples) * 1.0e06;
235     double nominal_sample_freq_khz = 1000.0 / nominal_period_us;
236 
237     printf("# %ld samples in %.6f seconds (actual T_p = %f us, nominal Fs =
    ↪  %.2f kHz).\n",
238            num_samples, end_time - start_time, nominal_period_us,
               ↪  nominal_sample_freq_khz);
```

```
239
240     double output_nominal_period_us = floor(nominal_period_us); // the
        ↪  clock is accurate only to us resolution
241
242     // Path to your data directory/file from previous define
243     const char *output_filename;
244     char hold_fname[32];
245     if (argc < 3)
246     { // No filename supplied, get default
247         time_t t = time(NULL);
248         struct tm tm = *localtime(&t);
249         snprintf(hold_fname, 32, "./out-%4d-%02d-%02d-%02d.%02d.%02d.bin",
250                  tm.tm_year + 1900, tm.tm_mon + 1, tm.tm_mday,
251                  tm.tm_hour, tm.tm_min, tm.tm_sec);
252         output_filename = hold_fname;
253     }
254     else
255     {
256         output_filename = OUTPUT_DATA;
257     }
258
259     // Write sample period and data to file
260     FILE *adc_data_file = fopen(output_filename, "wb+");
261     if (adc_data_file == NULL)
262     {
263         fprintf(stderr, "# Couldn't open file for writing: %s (did you
            ↪  remember to change OUTPUT_DATA?)\n", output_filename);
264         return 1;
265     }
266
267     fwrite(&output_nominal_period_us, sizeof(double), 1, adc_data_file);
268     fwrite(val, sizeof(uint16_t), ADCS * num_samples, adc_data_file);
269     fclose(adc_data_file);
270     printf("# Data written to file. Program ended successfully.\n\n");
271
272     gpioTerminate();
273     free(val);
274
275     return 0;
276 }
```

B.2 Python kode for plotting av teoretisk krysskorrelasjon

```python
import numpy as np
import matplotlib.pyplot as plt

# Parametre
N = 100
lag_shift = 5
n = np.arange(-N, N)

x1 = np.sinc(n / 5)
x2 = np.roll(x1, lag_shift)

corr = np.correlate(x1, x2, mode='full')
lags = np.arange(-len(x1) + 1, len(x1))
max_index = np.argmax(corr)

fig, axs = plt.subplots(1, 2, figsize=(14, 6))

axs[0].plot(n, x1, label=r'$x_1[n] =
↪  \mathrm{sinc}\left(\frac{n}{5}\right)$', alpha=0.8)
axs[0].plot(n, x2, label=r'$x_2[n] = x_1[n - 5]$', alpha=0.8)
axs[0].set_title(r'Signalene $x_1$ og $x_2$')
axs[0].set_xlabel(r'$n$')
axs[0].set_ylabel('Amplitude')
axs[0].set_xlim(-N, N)
axs[0].set_ylim(-3, 7)
axs[0].legend()
axs[0].grid(True)

axs[1].plot(lags, corr, label=r'Krysskorr. mellom $x_1$ og $x_2$',
↪  color='green')
axs[1].axvline(x=lags[max_index], color='red', linestyle='--',
↪  label=fr'Maks korrelasjon: $l = {lags[max_index]}$')
axs[1].set_title(r'Krysskorrelasjon')
axs[1].set_xlabel(r'$l$')
axs[1].set_ylabel('Amplitude av Krysskorrelasjon')
axs[1].set_xlim(-N, N)
axs[1].set_ylim(-3, 7)
axs[1].legend()
axs[1].grid(True)

plt.tight_layout()
plt.savefig("krysskorrelasjon_teoretisk.png")
plt.show()
```

B.3 Python kode for plotting av teoretisk autokorrelasjon

```python
import numpy as np
import matplotlib.pyplot as plt

N = 100
n = np.arange(-N, N)

x = np.sinc(n / 5)

corr = np.correlate(x, x, mode='full')
lags = np.arange(-len(x) + 1, len(x))
max_index = np.argmax(corr)

fig, axs = plt.subplots(1, 2, figsize=(14, 6))

axs[1].plot(lags, corr, label=r'Autokorrelasjon til $x$', color='green')
axs[1].axvline(x=lags[max_index], color='red', linestyle='--',
 ↪  label=fr'Maks korrelasjon: $l = {lags[max_index]}$')
axs[1].set_title(r'Autokorrelasjon av sinc-funksjon')
axs[1].set_xlabel(r'$l$')
axs[1].set_ylabel('Amplitude av autokorrelasjon')
axs[1].set_xlim(-N, N)
axs[1].set_ylim(-3, 7)
axs[1].legend()
axs[1].grid(True)

axs[0].plot(n, x, label=r'$x[n] = \mathrm{sinc}\left(\frac{n}{5}\right)$',
 ↪  alpha=0.8)
axs[0].set_title(r'Signal: Sinc-funksjon')
axs[0].set_xlabel(r'$n$')
axs[0].set_ylabel('Amplitude')
axs[0].legend()
axs[0].set_xlim(-N, N)
axs[0].set_ylim(-3, 7)
axs[0].grid(True)

plt.tight_layout()
plt.savefig("autokorrelasjon_og_signal.png")
plt.show()
```

B.4 Python kode for plott av enkeltmålinger

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as signal
import os

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")
        data = data.reshape((-1, channels))
    sample_period *= 1e-6
    return sample_period, data

def find_delay(signal1, signal2, fs):
    corr = np.correlate(signal1 - np.mean(signal1), signal2 -
    ↪  np.mean(signal2), mode="full")
    lags = np.arange(-len(signal1) + 1, len(signal1))
    max_index = np.argmax(np.abs(corr))
    max_lag = lags[max_index]
    delay = max_lag / fs
    return delay, lags, corr

def find_samples(fs, delay):
    return int(fs * delay)

def find_angle(n31, n21, n32):
    nevner = n31 - n21 + (2 * n32)
    if nevner < 0:
        theta = -np.arctan(np.sqrt(3) * (n31 + n21) / nevner) + np.pi
    else:
        theta = -np.arctan(np.sqrt(3) * (n31 + n21) / nevner)

    return theta + 2 * np.pi if theta < 0 else theta

# --- KONFIGURER FILNAVN ---
base_path = "/Users/trulsostvedt/Library/Mobile
↪  Documents/com~apple~CloudDocs/semester6/Sensorer/Lab2/ny/8cm_stoy/"
file_path = os.path.join(base_path, "klapp90_4cmshortdistance_1")

sample_period, data = raspi_import(file_path)
data = data[int(1 / sample_period):]  # fjern første sekund

for i in range(5):
    data[:, i] *= 3.3 / 4095
```

```
43      data[:, i] -= np.mean(data[:, i])
44
45  mic1 = data[1:, 2]
46  mic2 = data[1:, 3]
47  mic3 = data[1:, 4]
48  fs = 1 / sample_period
49  t = np.arange(len(mic1)) * sample_period
50
51  # Beregn krysskorrelasjoner og forsinkelser
52  delay12, lags12, corr12 = find_delay(mic1, mic2, fs)
53  delay13, lags13, corr13 = find_delay(mic1, mic3, fs)
54  delay23, lags23, corr23 = find_delay(mic2, mic3, fs)
55
56  # Beregn vinkel
57  n21 = find_samples(fs, delay12)
58  n31 = find_samples(fs, delay13)
59  n32 = find_samples(fs, delay23)
60  angle = find_angle(n31, n21, n32) * 180 / np.pi
61
62  # ---- PLOTTING ----
63  fig, axs = plt.subplots(2, 2, figsize=(10, 8))
64  fig.suptitle("Måling nummer 1 med klapp fra 90° vinkel", fontsize=14)
65
66  # Signalplot
67  axs[0, 0].plot(t, mic1, label="mic1")
68  axs[0, 0].plot(t, mic2, label="mic2")
69  axs[0, 0].plot(t, mic3, label="mic3")
70  axs[0, 0].set_title("Opptak av signaler")
71  axs[0, 0].set_xlim(0, 2)
72  axs[0, 0].set_xlabel("Tid [s]")
73  axs[0, 0].set_ylabel("Spenning [V]")
74  axs[0, 0].legend()
75  axs[0, 0].grid(True)
76
77  # mic1 vs mic2 →
78  max_idx12 = np.argmax(np.abs(corr12))
79  max_lag12 = lags12[max_idx12]
80  axs[0, 1].plot(lags12, corr12)
81  axs[0, 1].plot(max_lag12, corr12[max_idx12], "ro")
82  axs[0, 1].set_xlim(-10, 10)
83  axs[0, 1].set_title(f" = {delay12*1e3:.2f} ms")
84  axs[0, 1].set_xlabel("Lag [samples]")
85  axs[0, 1].set_ylabel("Korrelasjonsverdi")
86  axs[0, 1].grid(True)
87
88  # mic1 vs mic3 →
89  max_idx13 = np.argmax(np.abs(corr13))
```

```
max_lag13 = lags13[max_idx13]
axs[1, 0].plot(lags13, corr13)
axs[1, 0].plot(max_lag13, corr13[max_idx13], "ro")
axs[1, 0].set_xlim(-10, 10)
axs[1, 0].set_title(f" = {delay13*1e3:.2f} ms")
axs[1, 0].set_xlabel("Lag [samples]")
axs[1, 0].set_ylabel("Korrelasjonsverdi")
axs[1, 0].grid(True)

# mic2 vs mic3 →
max_idx23 = np.argmax(np.abs(corr23))
max_lag23 = lags23[max_idx23]
axs[1, 1].plot(lags23, corr23)
axs[1, 1].plot(max_lag23, corr23[max_idx23], "ro")
axs[1, 1].set_xlim(-10, 10)
axs[1, 1].set_title(f" = {delay23*1e3:.2f} ms")
axs[1, 1].set_xlabel("Lag [samples]")
axs[1, 1].set_ylabel("Korrelasjonsverdi")
axs[1, 1].grid(True)

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

print(f"Beregnet vinkel: {angle:.2f}°")


```

B.5 Python kode for plott av flere målinger og tabell med usikkerhet

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as signal
import os
import pandas as pd

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")
        data = data.reshape((-1, channels))
    sample_period *= 1e-6
    return sample_period, data

def find_delay(signal1, signal2, fs):
    corr = np.correlate(signal1 - np.mean(signal1), signal2 -
    ↪  np.mean(signal2), mode="full")
    lags = np.arange(-len(signal1) + 1, len(signal1))
    max_index = np.argmax(np.abs(corr))
    max_lag = lags[max_index]
    delay = max_lag / fs
    return delay, lags, corr

def find_samples(fs, delay):
    return int(fs * delay)

def find_angle(n31, n21, n32):
    nevner = n31 - n21 + (2 * n32)
    if nevner < 0:
        theta = -np.arctan(np.sqrt(3) * (n31 + n21) / nevner) + np.pi
    else:
        theta = -np.arctan(np.sqrt(3) * (n31 + n21) / nevner)

    return theta + 2 * np.pi if theta < 0 else theta

# --- KONFIGURER DENNE STIEN ---
base_path = "/Users/trulsostvedt/Library/Mobile
↪  Documents/com~apple~CloudDocs/semester6/Sensorer/Lab2/ny/8cm_stoy/"
file_names = [f"klapp90_4cmshortdistance_{i}" for i in range(1, 10)]

results = []
angles = []

fig, axs = plt.subplots(9, 4, figsize=(20, 30))  # 9 målinger, 4 plott hver
```

```
43
44  for idx, fname in enumerate(file_names):
45      full_path = os.path.join(base_path, fname)
46      try:
47          sample_period, data = raspi_import(full_path)
48      except FileNotFoundError:
49          print(f"Fant ikke: {fname}")
50          continue
51
52      data = data[int(1 / sample_period):]  # Fjern første sekund
53
54      for i in range(5):
55          data[:, i] *= 3.3 / 4095
56          data[:, i] -= np.mean(data[:, i])
57
58      mic1 = data[1:, 2]
59      mic2 = data[1:, 3]
60      mic3 = data[1:, 4]
61      fs = 1 / sample_period
62      t = np.arange(len(mic1)) * sample_period
63
64      # Beregn forsinkelser og korrelasjoner
65      delay12, lags12, corr12 = find_delay(mic1, mic2, fs)
66      delay13, lags13, corr13 = find_delay(mic1, mic3, fs)
67      delay23, lags23, corr23 = find_delay(mic2, mic3, fs)
68
69      n21 = find_samples(fs, delay12)
70      n31 = find_samples(fs, delay13)
71      n32 = find_samples(fs, delay23)
72
73      angle = find_angle(n31, n21, n32) * 180 / np.pi
74      angles.append(angle)
75      results.append([idx + 1, angle, delay12, delay13, delay23])
76
77      # ---- PLOTTING I SUBPLOTS ----
78      axs[idx, 0].plot(t, mic1, label="mic1")
79      axs[idx, 0].plot(t, mic2, label="mic2")
80      axs[idx, 0].plot(t, mic3, label="mic3")
81      axs[idx, 0].set_title(f"Måling {idx + 1}: signaler")
82      axs[idx, 0].set_xlim(0, 2)
83      axs[idx, 0].legend()
84      axs[idx, 0].grid(True)
85      # mic1 vs mic2 →
86      max_idx12 = np.argmax(np.abs(corr12))
87      max_lag12 = lags12[max_idx12]
88      axs[idx, 1].plot(lags12, corr12)
89      axs[idx, 1].plot(max_lag12, corr12[max_idx12], "ro")
```

```
    axs[idx, 1].set_xlim(-10, 10)
    axs[idx, 1].set_title(f" = {delay12*1e3:.2f} ms")
    axs[idx, 1].set_xlabel("Lag [samples]")
    axs[idx, 1].set_ylabel("Korrelasjonsverdi")
    axs[idx, 1].grid(True)
    # mic1 vs mic3 →
    max_idx13 = np.argmax(np.abs(corr13))
    max_lag13 = lags13[max_idx13]
    axs[idx, 2].plot(lags13, corr13)
    axs[idx, 2].plot(max_lag13, corr13[max_idx13], "ro")
    axs[idx, 2].set_xlim(-10, 10)
    axs[idx, 2].set_title(f" = {delay13*1e3:.2f} ms")
    axs[idx, 2].set_xlabel("Lag [samples]")
    axs[idx, 2].set_ylabel("Korrelasjonsverdi")
    axs[idx, 2].grid(True)
    # mic2 vs mic3 →
    max_idx23 = np.argmax(np.abs(corr23))
    max_lag23 = lags23[max_idx23]
    axs[idx, 3].plot(lags23, corr23)
    axs[idx, 3].plot(max_lag23, corr23[max_idx23], "ro")
    axs[idx, 3].set_xlim(-10, 10)
    axs[idx, 3].set_title(f" = {delay23*1e3:.2f} ms")
    axs[idx, 3].set_xlabel("Lag [samples]")
    axs[idx, 3].set_ylabel("Korrelasjonsverdi")
    axs[idx, 3].grid(True)
plt.tight_layout()
plt.show()

# ---- TABELL OG OPPSUMMERING ----
df = pd.DataFrame(results, columns=["Måling", "Vinkel (grader)", "Tau12
↪  (s)", "Tau13 (s)", "Tau23 (s)"])
print("\n--- Resultattabell ---\n")
print(df.to_string(index=False))

summary = {
    "Forventet vinkel": 90,
    "Gjennomsnittlig målt vinkel": np.mean(angles),
    "Standardavvik": np.std(angles),
    "Varians": np.var(angles)
}
summary_df = pd.DataFrame([summary])

print("\n--- Oppsummering ---\n")
print(summary_df.to_string(index=False))
```