

Designnotat

Tittel: Designprosjekt 4 - Kunstig romklang	
Forfattere: Temmam Nabil Altaie	
Versjon: 2.0	Dato: 12.05.25

Innhold

1 Problembeskrivelse **2**

2 Prinsipiell løsning **3**
2.1 Matematisk modell . . . 3
2.2 Impulsrespons . . . 4
2.3 Relasjon mellom modell og etterklangstid . . . 5
2.4 Ringbuffer og forsinkelse . . . 5
2.5 Pseudokode for ringbuffer . . . 6

3 Realisering og test **7**
3.1 Testoppsett . . . 7
3.2 Implementasjon av DSP-algoritmen . . . 8
3.2.1 Kodeutdrag for ISR . . . 9
3.3 Gjennomføring og observasjoner . . . 9
3.4 Teoretisk måling gjennom python: . . . 10
3.5 Scope-målinger . . . 10
3.6 Respons ved enhetspuls . . . 12
3.7 Sammenligning mellom teori og måling . . . 13

4 Konklusjon **14**

5 Takk **15**

Vedlegg **16**

Referanseliste **17**

1 Problembeskrivelse



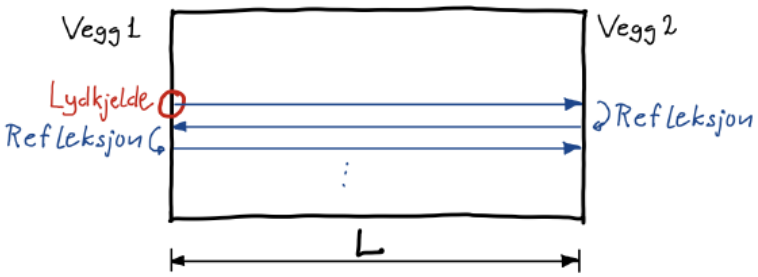
Figur 1: Illustrasjon av hvordan systemet implementeres

I dette designprosjektet skal vi utvikle et system som legger til kunstig romklang på et lydsignal. Målet er å undersøke om en enkel matematisk metode kan brukes til å gi etterklang på lydsignalet, og om dette kan gjøres i sanntid på en mikrokontroller. Vi ønsker også å utforske hvordan ulike modellparametere påvirker lydsignalet.

Systemet i figur 1 mottar et analogt lydsignal $x(t)$ som sendes inn til en ADC (Analog til Digital Converter), hvor det omgjøres til et digitalt signal $x[n]$. Dette signalet behandles av mikrokontrolleren, og det bearbeidede resultatet $y[n]$ sendes videre til en DAC (Digital til Analog Converter), som produserer et utgangssignal $y(t)$.

2 Prinsipiell løsning

Denne delen er i stor grad basert på modellen og metodikken presentert i teknisk notat for prosjektet [1].



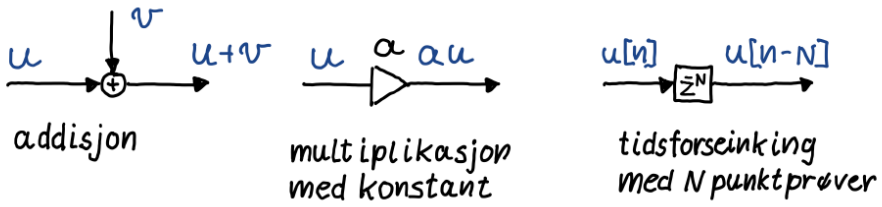
Figur 2: Lydrefleksjon mellom to vegger.

Kunstig romklang er en digital etterligning av naturlige lydrefleksjoner i et fysisk rom. Når lyd spilles i et rom, reflekteres lydbølgene mellom vegger og andre flater slik som figur 2 viser. Hver refleksjon blir gradvis svakere på grunn av absorpsjon i overflatene. Romklang oppfattes derfor som en serie med dempede og forsinkede kopier av det opprinnelige signalet. Figur 2 illustrerer en forenklet modell. Den tar for seg et rom av lengden L med en lydkilde i den ene enden. For enkelhetsskyld er det kun endeveggene som betraktes ved lydrefleksjon. I en virkelig situasjon ville sidevegger og andre flater også spille en rolle, men vi ser bort fra dette i illustrasjonen.

For å simulere dette fenomenet digitalt benyttes en enkel matematisk modell som kombinerer det direkte signalet med en forsinket og dempet versjon av tidligere utganger.

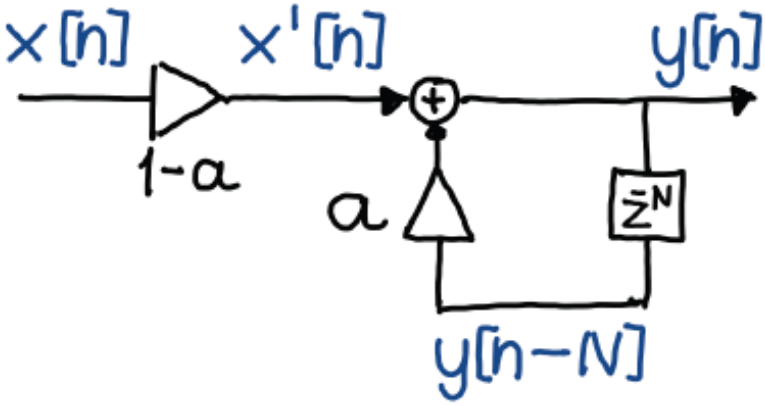
2.1 Matematisk modell

Figurene og teorien er hentet fra teknisk notat [1].



Figur 3: Standardiserte symbol for signalflytdiagram

Figur 3 viser symbolene som brukes i signalflytdiagrammet, slik som summeringspunkt, forsinkelsesledd og forsterkning. Disse symbolene brukes videre i figur 4, som illustrerer hvordan det digitale signalet behandles i modellen.



Figur 4: Signalflyt for romklang-modellen: kombinasjon av direkte og forsinket signal.

Romklangen simuleres digitalt gjennom følgende differenselikning:

$$y[n] = (1 - a)x[n] + a \cdot y[n - N] \tag{1}$$

hvor:

- $x[n]$ er det digitale inngangssignalet
- $y[n]$ er det prosesserte utgangssignalet
- a er refleksjonskoeffisienten, $0 < a < 1$
- N er forsinkelsen målt i samples

Ved sampling med konstant tidsintervall T_s representeres det kontinuerlige signalet $x(t)$ som en diskret sekvens $x[n] = x(nT_s)$. Modellen kombinerer det direkte inngangssignalet $x[n]$ med en tidligere utgangsverdi $y[n - N]$, som først skaleres med a og deretter legges sammen med $(1 - a) \cdot x[n]$. Forsinkelsen N implementeres med en *ringbuffer* (se seksjon 2.4), og vektingen med a og $(1 - a)$ gir graden av etterklang og hvor fort den dempes.

2.2 Impulsrespons

Formen på impulsresponsen er hentet fra teknisk notat [1].

Impulsresponsen gir innsikt i hvordan systemet reagerer på en enkelt lydhendelse:

$$h[n] = \begin{cases} a^k, & \text{for } n = kN, \quad k \in \mathbb{Z}_{\geq 0} \\ 0, & \text{ellers} \end{cases} \tag{2}$$

Dette beskriver en sekvens av dempede refleksjoner med mellomrom N samples, hvor amplituden avtar eksponentielt. Lav a gir tydelige ekko, mens høy a gir jevnere og mer vedvarende klang.

2.3 Relasjon mellom modell og etterklangstid

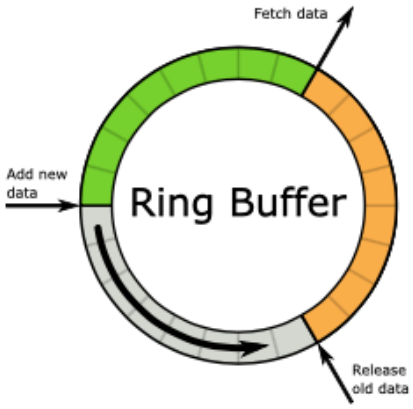
Uttrykket for å beregne N baserer seg på en forenklet analyse av impulsresponsen [1].

Et viktig begrep i akustikk er etterklangstid T_e, som er definert som tiden det tar før et signal er redusert med 60 dB (en tusendel av opprinnelig styrke). I modellen er det mulig å koble T_e, a, og N via følgende uttrykk:

$$N = \frac{\log_{10}(1/a)}{3} \cdot \frac{T_e}{T_s} \tag{3}$$

hvor T_s er samplingstiden. Samme etterklangstid kan oppnås med ulike kombinasjoner av a og N, men lydkarakteristikken blir forskjellig. I vårt prosjekt er T_e og T_s oppgitt som kravspesifikasjoner.

2.4 Ringbuffer og forsinkelse



Figur 5: Illustrasjon av ringbuffer: sirkulær tilgang til tidligere verdier.

For å hente tidligere verdier $y[n - N]$ i sanntid må systemet huske gamle utganger. Dette gjøres ved hjelp av en *ringbuffer* — en sirkulær datastruktur som kontinuerlig lagrer de siste utgangsverdiene.

Når en ny verdi $y[n]$ er beregnet, lagres den i bufferens nåværende posisjon. Neste gang oppdateres posisjonen. Tidligere verdier hentes med en beregnet indeks:

$$\text{read_index} = (\text{write_index} - N) \mod \text{BUFFER_SIZE}$$

2.5 Pseudokode for ringbuffer

Under vises prinsippet bak beregningene i en forenklet pseudokode:

```
funksjon prosesser(x_n):
    # Juster input til symmetrisk område og skaler til fastpunkt
    x = (x_n - midtverdi) * skala

    # Hent tidligere utgangsverdi med forsinkelse N
    y_n_minus_N = buffer[ (index - N) % buff_størrelse ]

    # Beregn ny utgangsverdi med vekting
    y = (1 - a) * x + a * y_n_minus_N

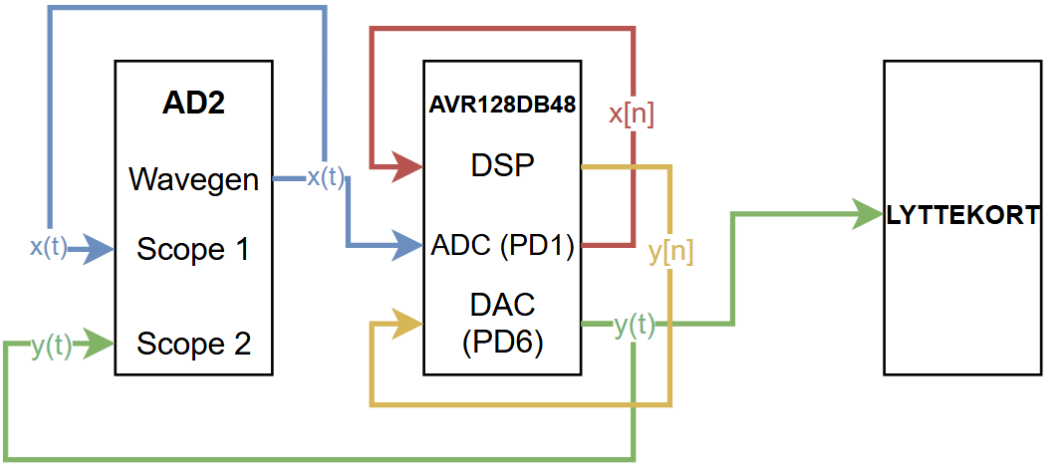
    # Lagre ny verdi i buffer
    buffer[index] = y

    # Flytt indeks videre (ringbuffer)
    index = (index + 1) % buff_størrelse

    # Konverter tilbake til originalt område og returner
    return y + midtverdi
```

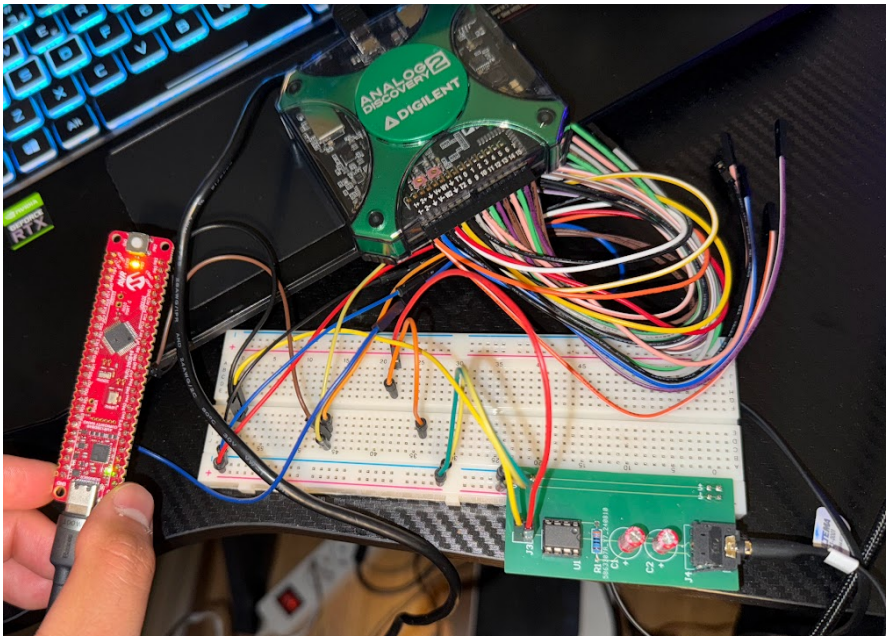
3 Realisering og test

3.1 Testoppsett



Figur 6: Blokkskjema som viser signalflyten i systemet.

Figur 6 viser hele signalflyten i systemet, fra generering av lydsignal til måling og lytting. Både analoge og digitale signalbaner er illustrert, samt hvilke pinner som benyttes på mikrokontrolleren.



Figur 7: Fysisk oppkobling mellom AD2, AVR og lyttekort.

Systemet ble realisert ved bruk av en AVR128DB48 mikrokontroller. Inngangssignalet $x(t)$ ble generert via AD2s (Analog Discovery 2) Wavegen-funksjon med et .wav-lydsignal som ble sendt inn til mikrokontrollerens ADC-inngang (PD1). Signalets offset ble satt til 1.65 V for korrekt midtstilling i 10-bit ADC. Etter digital signalbehandling ble $y[n]$ sendt til DAC-utgangen (PD6) og utgangssignalet $y(t)$ ble analysert ved lyttekort og oscilloskop for visuell analyse. Se figur 6 for visuell representasjon.

Wavegen ble satt opp med følgende parametre under test:

- Amplitude: 1 V
- Offset: 1.65 V
- Sample rate: 16 kHz
- Repeat: 1 (enkeltutløsning for analyse av etterklangstiden)

3.2 Implementasjon av DSP-algoritmen

Etterklangstiden $T_e = 0.8$ s og samplingstiden $T_s = 30$ μs er individuelle kravspesifikasjoner. Signalbehandlingen er implementert direkte i en interrupt service routine (ISR), som trigges med jevne mellomrom basert på en hardware-timer konfigurert med ønsket samplingstid ($T_s = 30\,\mu s$). I ISR-en leses et nytt samplesignal fra ADC, behandles i henhold til en rekursiv modell, og sendes til DAC for analog utgang.

Modellen som er implementert følger differenselikning 1:

For å muliggjøre tilbakematingen av $y[n-N]$, er det benyttet en *ringbuffer* (seksjon 2.4). Hele signalbehandlingen skjer i fastpunktaritmetikk ved bruk av skalerte heltallsverdier (int16 og int32), for å unngå unødvendig ressursbruk og sikre sanntidsegenskaper på mikrokontrolleren.

3.2.1 Kodeutdrag for ISR

```
107  ISR(TCA0_OVF_vect) {
108      static uint16_t inputSample = 0;
109      static uint16_t outputSample = 0;
110      static int16_t x = 0;
111      static int16_t y = 0;
112      static int32_t temp1 = 0;
113      static int32_t temp2 = 0;
114
115      // 1. Les ny sample fra ADC
116      inputSample = ADC0_read();
117
118      // 2. Konverter til signed og fastpunkt (fra [0, 1023] → [-512, 511] → <<6)
119      x = ((int16_t)inputSample - 512) << 6;
120
121      // 3. Hent tidligere y[n - N] fra ringbuffer
122      uint16_t read_index = (write_index + BUFFER_SIZE - N) % BUFFER_SIZE;
123      int16_t y_n_minus_N = y_buffer[read_index];
124
125      // 4. Beregn rekursiv modell: y[n] = (1 - a)x[n] + a*y[n-N]
126      temp1 = ((int32_t)(64 - A)) * x;
127      temp2 = ((int32_t)A) * y_n_minus_N;
128      y = (int16_t)((temp1 + temp2) >> 6);
129
130      // 5. Lagre ny y[n] i ringbuffer
131      y_buffer[write_index] = y;
132      write_index = (write_index + 1) % BUFFER_SIZE;
133
134      // 6. Konverter tilbake til unsigned for DAC
135      outputSample = y + 512;
136      DAC0_setVal(outputSample);
137
138      // 7. Nullstill interrupt-flagget
139      TCA0.SINGLE.INTFLAGS = TCA_SINGLE_OVF_bm;
140  }
141
```

Figur 8: Kodeutdraget til ISR. Inneholder kommentarer.

3.3 Gjennomføring og observasjoner

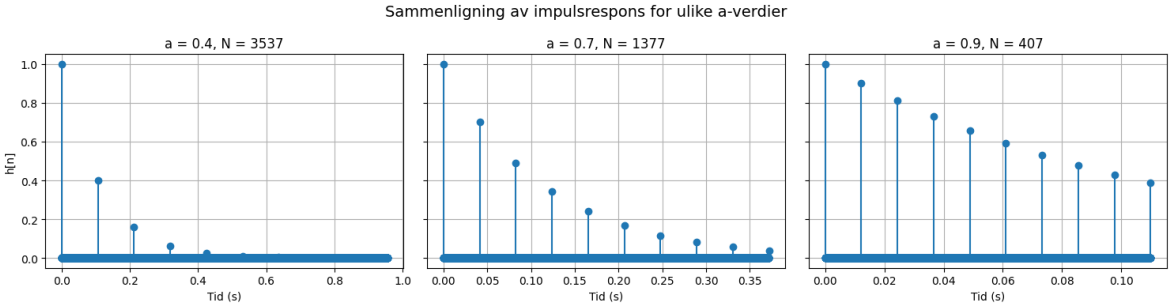
Verdien for N er beregnet ved likning 3.

Tabellen under viser de testede verdiene for a, tilhørende N, og estimert etterklangslengde ut fra scope-måling:

Tabell 1: Verdier for refleksjonskoeffisient a og tilhørende forsinkelse N for $T_e = 0.8\,s$ og $T_s = 30\,\mu s$

a-verdi	N-verdi (samples)
0.1	8889
0.2	6213
0.3	4648
0.4	3537
0.5	2676
0.6	1972
0.7	1377
0.8	861
0.9	407

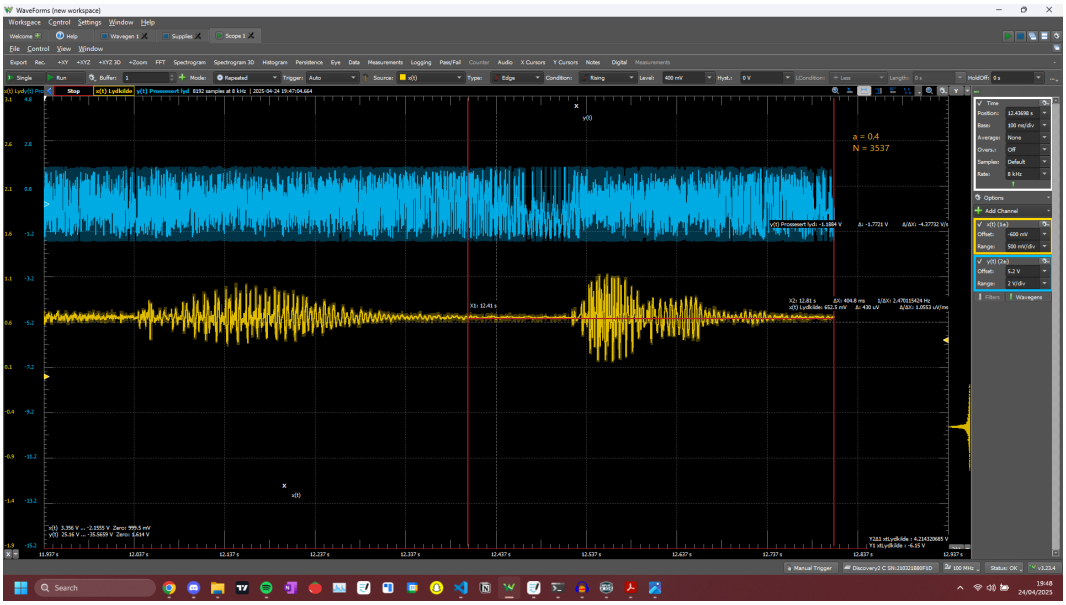
3.4 Teoretisk måling gjennom python:



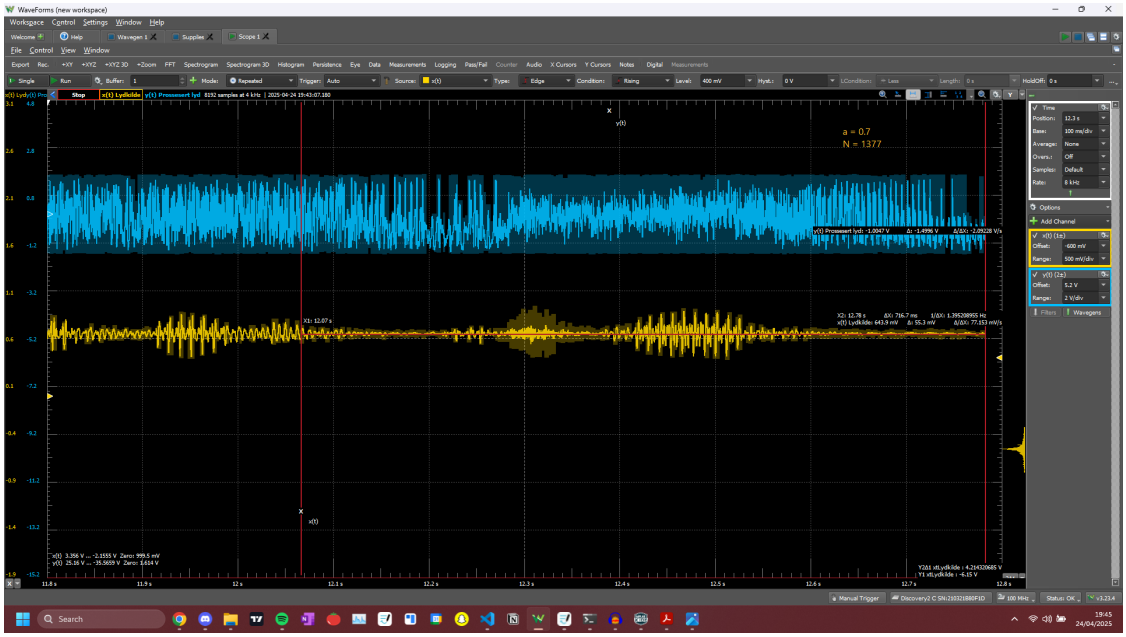
Figur 9: Enhetspulsrespons ved gitte verdier.

Vi tar utgangspunkt i likning 2 for å danne enhetspulsresponsene. Her kan vi se de teoretiske forventningene til systemet ved bruk av programmering. I neste del vil det være mulig å sammenligne teori og praksis. Se vedlegg 13 for kodeutdrag.

3.5 Scope-målinger



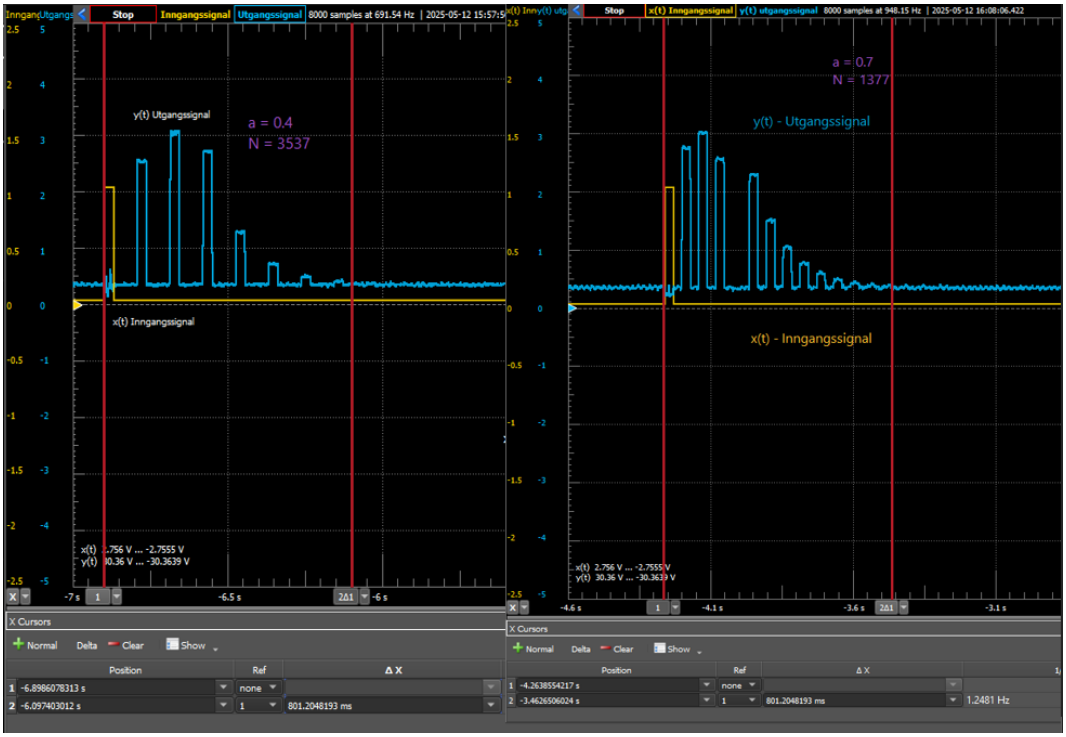
Figur 10: Etterklangsrespons med $a = 0.4$, $N = 3537$. Tydelig ekko-effekt med rask demping.



Figur 11: Etterklangsrespons med $a = 0.7$, $N = 1377$. Mykere og lengre etterklangseffekt.

Figur 10 og figur 11 viser henholdsvis resultatet for $a = 0.4$ og $a = 0.7$, da disse verdiene ga best resultat på waveforms. Disse verdiene har tilhørende etterklang som er synlige i det prosesserte signalet. $x(t)$ (gul) er kort og tydelig, mens $y(t)$ (blå) viser en karakteristisk demping.

3.6 Respons ved enhetspuls



Figur 12: Eksperimentell enhetspulsrespons med $a = 0.4$, $N = 3537$.

For ytterligere å teste systemets impulsrespons ble enheten $x[n]$ simulert ved hjelp av AD2s Wavegen-funksjon, konfigurert til å sende en kort puls. Dette gjør det mulig å observere systemets grunnrespons og sammenligne med teoretisk modell vist i figur 12.

Som figurene viser, følger systemets utgang $y(t)$ en karakteristisk dempet pulssekvens, med eksponentielt avtagende amplitude. Dette samsvarer med forventet impulsrespons som beskrevet i likning 2, og illustrert teoretisk i figur 9. Høyere verdier av a gir lengre og jevnere klang, mens lavere verdier gir raskere demping.

Det observeres imidlertid at den eksperimentelle responsen ikke er helt ideell. Den blå kurven ($y(t)$) har variasjoner i amplituden mellom pulsene som ikke er fullt symmetriske eller perfekte. Dette kan skyldes flere praktiske forhold, som:

- Svingninger i ADC-innlesning og DAC-utgang
- Støy i målingene eller på inngangssignalet
- Runde- og kvantiseringsfeil i fastpunktaritmetikken
- Ikke-initialisert ringbuffer ved første kall

Til tross for dette viser målingene tydelig den teoretiske strukturen, og bekrefter at modellen fungerer i praksis.

3.7 Sammenligning mellom teori og måling

For å undersøke om implementasjonen faktisk gir ønsket etterklangstid, ble målinger gjort i AD2 Scope for de mest representative testverdiene. Målt etterklangstid ble estimert ved å plassere markører i grafen fra tidspunktet utgangen starter til den visuelt har avtatt til et tilnærmet konstant nivå.

Tabell 2 viser en sammenligning mellom teoretisk beregnet etterklangstid (basert på valgte verdier for a og N) og det som faktisk ble målt i scope.

Tabell 2: Sammenligning av teoretisk og målt etterklangstid for utvalgte a-verdier

a	N **(samples)**	**Forventet** T_e **[s]**	**Målt** T_e **[s]**
0.4	3537	0.80	0.78
0.7	1377	0.80	0.81

Som det fremgår av tabellen, gir lavere a-verdier en raskere dempet respons, noe som reduserer opplevd etterklangstid til under den teoretiske verdien. For $a = 0.7$ stemmer målingen derimot svært godt med forventet $T_e = 0.8\,s$, og viser at modellen fungerer i praksis. Uoverensstemmelser ved lave verdier kan forklares med at refleksjonene dempes under det hørbare/stabile nivået før 60 dB-nedgangen fullføres.

4 Konklusjon

Resultatene fra målingene viste at systemet oppførte seg i tråd med teoretiske forventninger. Etterklangen kunne tydelig observeres som en gradvis avtagende komponent i utgangssignalet, med varighet kontrollert gjennom parameteren a og den tilhørende forsinkelsen N. Visuelle målinger med AD2 Scope bekreftet at etterklangstiden lå tett opptil spesifiserte $T_e = 0.8$ s, og forskjellen mellom ulike parameterverdier ga klare endringer i lydkarakter.

Videreutvikling av prosjektet kan innebære støyreduksjon, videreutvikling av programmering og økt kompetanse innenfor digital signalbehandling.

5 Takk

Takk til klassekameratene som har bidratt til et godt samarbeid for å løse dette designprosjektet. I tillegg har jeg brukt ChatGPT for forklaringer av begreper, tilbakemeldinger og forbedringspotensialer i teksten min.

Vedlegg

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

# Funksjon for å generere impulsrespons
def generate_impulse_response(a, N, T_s, k_max=10):
    n_total = N * (k_max - 1) + 1
    h = np.zeros(n_total)
    for k in range(k_max):
        h[k * N] = a**k
    t = np.arange(0, len(h)) * T_s
    return t, h

# Parametre
T_s = 30e-6
a_values = [0.4, 0.7, 0.9]
N_values = [3537, 1377, 407]  # For Te=0.8s og Ts=30us
k_max = 10

# Plotting side om side
fig, axes = plt.subplots(1, 3, figsize=(15, 4), sharey=True)

for i, (a, N) in enumerate(zip(a_values, N_values)):
    t, h = generate_impulse_response(a, N, T_s, k_max)
    axes[i].stem(t, h, basefmt=" ")
    axes[i].set_title(f"a = {a}, N = {N}")
    axes[i].set_xlabel("Tid (s)")
    axes[i].grid(True)

axes[0].set_ylabel("h[n]")
fig.suptitle("Sammenligning av impulsrespons for ulike a-verdier", fontsize=14)
plt.tight_layout()
plt.show()
```

Figur 13: Kodeutdrag for teoretisk måling av enhetspulsrespons

Referanseliste

[1] Lundheim, L. (2025). *Ei enkel romklangs-eining* (Teknisk notat for DP4, NTNU – Institutt for elektroniske system).