Signalbehandling

Laborationsuppgift 2

Namn 1 Maximilian Andersen

Namn 2 Elias Brännström

Genom att skicka in labbrapporten intygar du/ni att följande regler har följts:

1. Laborationsuppgifter skall lösas självständigt av varje laborationsgrupp. Det är tillåtet att diskutera lösningar, men INTE att kopiera lösningar! Det är alltså INTE tillåtet att ge laborationsresultat eller färdiga lösningar till en annan grupp.

2. Bägge gruppmedlemmarna förväntas ta aktiv del i genomförandet av laborationen och skrivandet av rapporten. Detta inkluderar att bygga, programmera, dokumentera, testa och felsöka. Bägge gruppmedlemmarna skall kunna svara på frågor om hur laborationen genomförts och vilka resultat som erhållits.

3. Examination baseras alltid på individuella resultat

Tommy Andersson
januari 2020

Aliasing och spektrum för samplad signal

Skriv inte ut detta dokument utan ha det öppet på datorn under laborationen och besvara frågorna direkt i dokumentet. En del foton skall tas och klistras in. Försök använda lämplig bildkvalitet så bilden syns tydligt men filstorleken inte blir för stor. (Bilden som redan finns i dokumentet har en storlek på ca 60 kB) Efter laborationen laddas dokumentet upp som pdf på Canvas.

Laborationen genomförs som vanligt i par dvs. ni jobbar två och två.

Utrustning

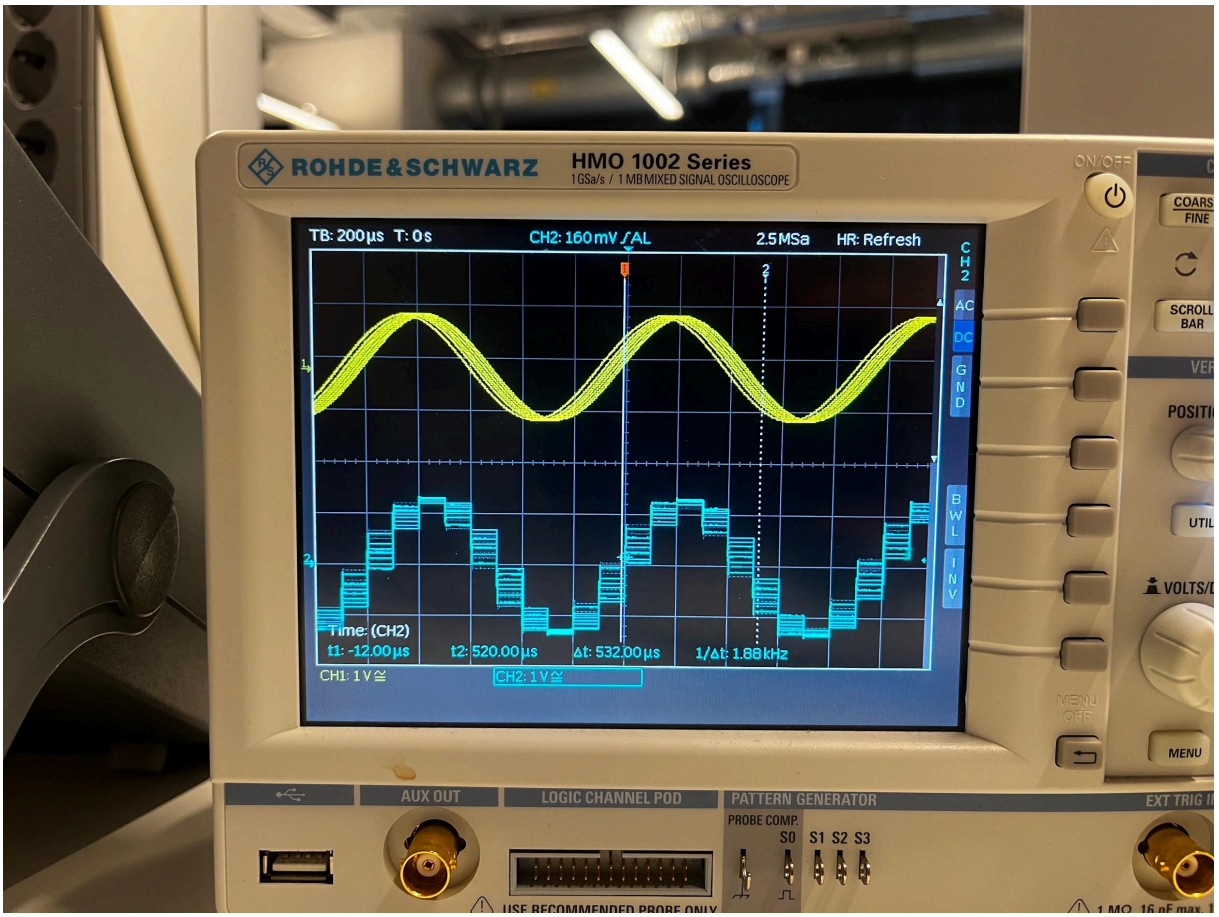
- Labbplatta med uppkoppling från labbuppgift 1
- Funktionsgenerator Wavetek
- Oscilloskop R&S HMO1002 med två probar
- 1 st koaxialkabel, 1 m med BNC-kontakter i ena änden och banankontakter i andra.
- (Funktionsgenerator HP3312A, eventuellt)
- Medhavd smartphone plus hörlurar med sladd och 3.5 mm anslutning

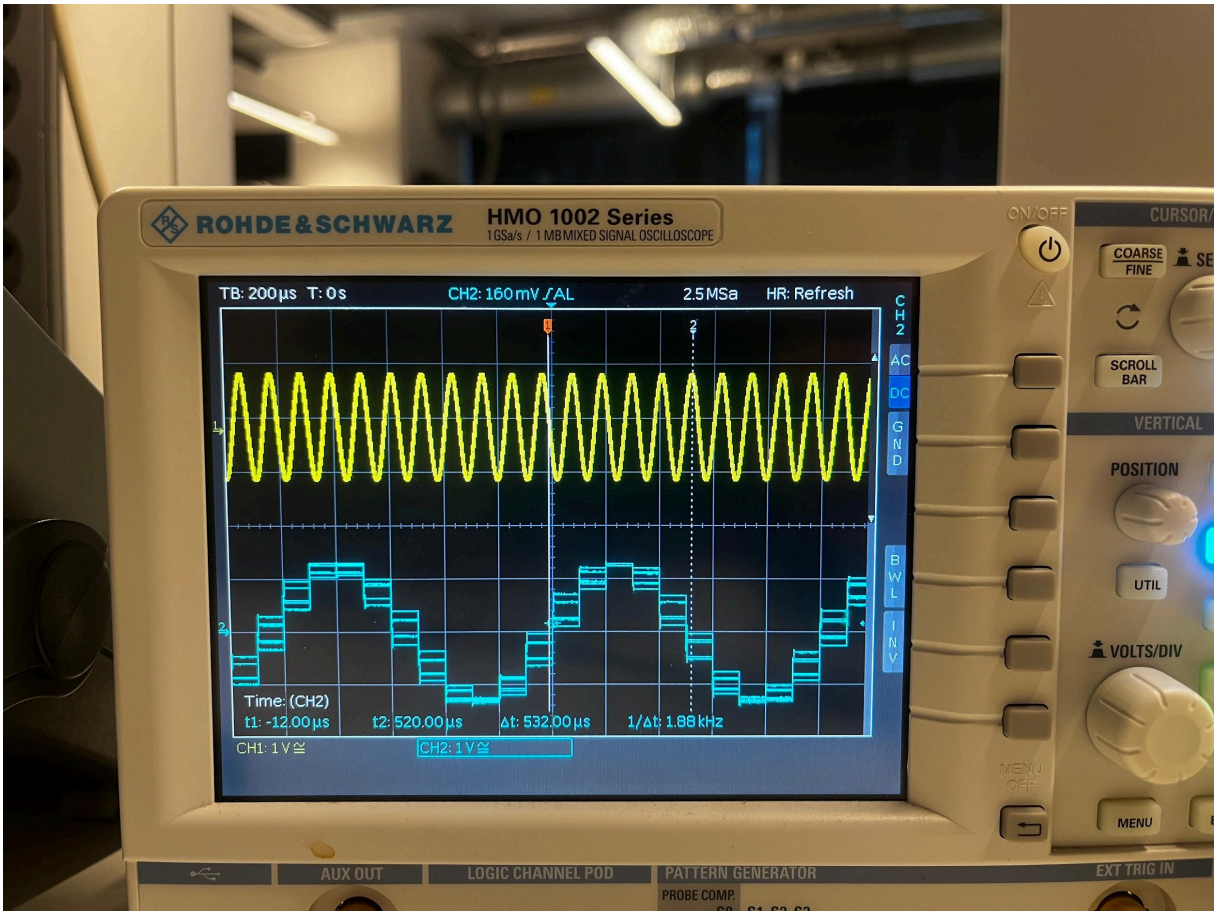
1. Mätningar i tidsplanet

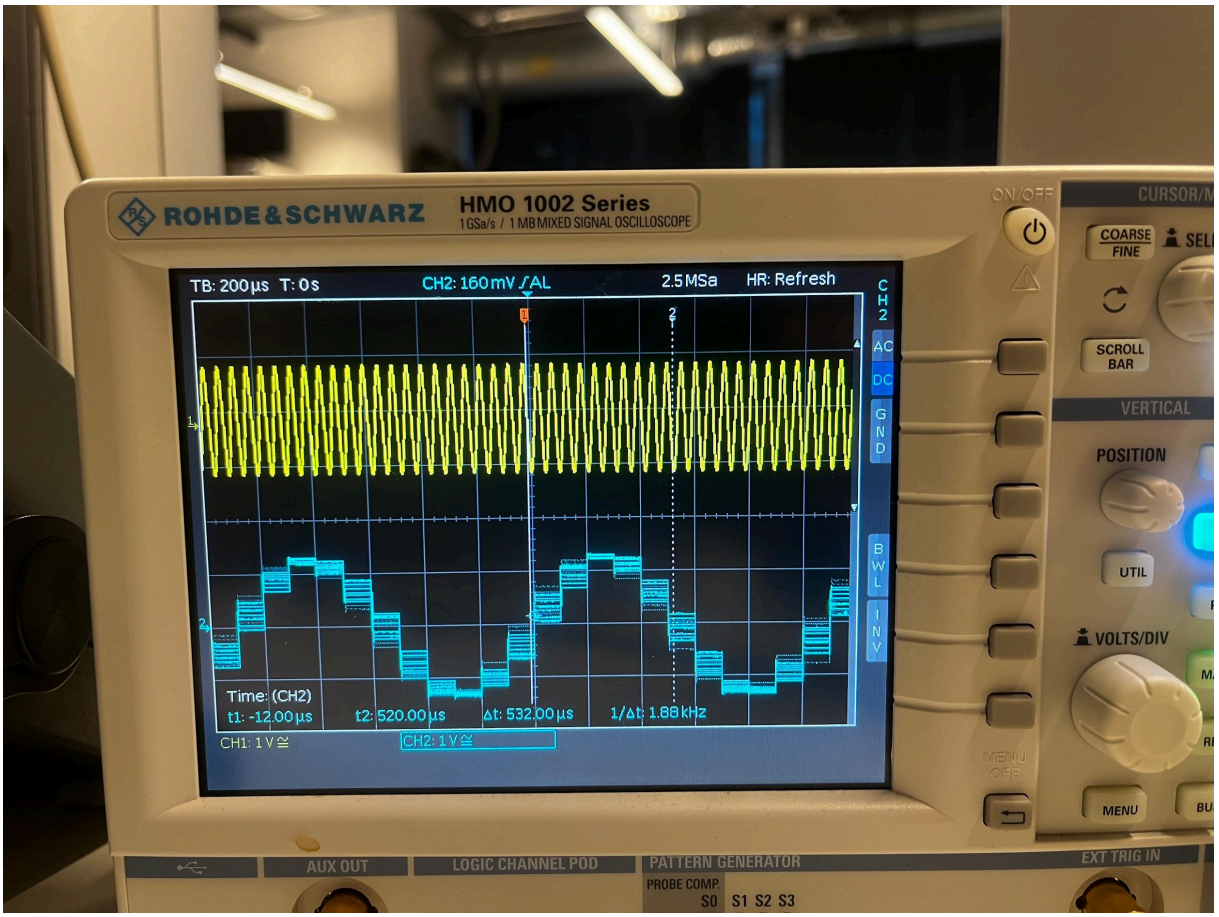
Samma uppkoppling och program som i labbuppgift 1.
Använd Wavetek (samma som i labbuppgift 1) som funktionsgenerator. Börja med sinus,1 kHz, amplitud 1,0 V (dvs 2,0 V_{t-t}).
Anslut och ställ in oscilloskopet så du kan se insignalen på kanal 1 överst och utsignalen på kanal 2 underst. Använd probar (x10).

Ställ in tidbasen så du ser ca 2 perioder på skärmen. Trigga på kanal 2 (dvs. utsignalen). Öka frekvensen och se vad som händer. Behåll tidbasinställningen och fortsätt trigga på utsignalen.
Kolla speciellt vad som händer vid 9 kHz, 11 kHz, 19 kHz och 21 kHz!

1.1 Beskriv! Lägg in foto på 1 kHz, 9 kHz och ytterligare något fall.


ROHDE&SCHWARZ
HMO 1002 Series
1 GSa/s / 1 MB MIXED SIGNAL OSCILLOSCOPE
TB: 200 µs T: 0 s
CH2: 160 mV ∫AL
2.5 MSa
HR: Refresh
Time: (CH2)
t1: -12.00 µs
t2: 520.00 µs
Δt: 532.00 µs
1/Δt: 1.88 kHz
CH1: 1 V
CH2: 1 V
AUX OUT
LOGIC CHANNEL POD
PATTERN GENERATOR
PROBE COMP.
S0 S1 S2 S3
EXT TRIG IN
MENU


ROHDE&SCHWARZ
HMO 1002 Series
1 GSa/s / 1 MB MIXED SIGNAL OSCILLOSCOPE
TB: 200 µs T: 0 s
CH2: 160 mV ∫AL
2.5 MSa
HR: Refresh
Time: (CH2)
t1: -12.00 µs
t2: 520.00 µs
Δt: 532.00 µs
1/Δt: 1.88 kHz
CH1: 1 V
CH2: 1 V
AUX OUT
LOGIC CHANNEL POD
PATTERN GENERATOR
PROBE COMP.
EXT TRIG IN
COARSE
FINE
SCROLL BAR
VERTICAL
POSITION
UTIL
VOLTS/DIV
MENU



Utsignalen bildar en kurva med en lägre frekvens pga aliasing.

> 2. Mätningar i frekvensplanet

Fortsätt med samma uppkoppling som tidigare. Använd frekvensen 1 kHz, sinusform och amplituden 1,0 V , dvs. $V_{t\text{-}t}$ = 2,0 V. Se till oscilloskopet står AC-läge. Klicka ”AUTO SET” (kort klick annars ändras probinställningen till x1). Se till att knappen ”CH1” lyser (gul).

Nu vill vi se insignalens spektrum. Eftersom det bara är en ren sinussignal hoppas vi kunna se en spik vid frekvensen 1 kHz…

Börja med att klicka på ACQUIRE längst ner till höger. Se till HIGH RESOLUTION står på Auto (verkar vara en fördel vid mätningar av spektrum).

Klicka ”FFT” (upptill på panelen) . Skärmen delas nu, överst visas signalen i tidsplanet, under i en större ruta visas signalens spektrum.
Men vi behöver göra lite inställningar.
Till höger på skärmen har det dykt upp en speciell meny för FFT.
Ställ in så att
MODE: Refresh
POINTS: 2048
WINDOW: Hanning (mer om vad detta är senare i kursen)
Y-SCALING: Veff
(klicka på motsvarande funktionsknapp höger om menyn och välj med ratten SELECT alternativt flera tryck på funktionsknappen)
Längst ner på skärmen har vi info om Span och Center. Span anger hur stort frekvensområde som visas, Center anger vilken frekvens som ligger i mitten på skärmen.
Vi kan ändra Span genom att vrida ratten ”TIME/DIV” och Center med hjälp av ratten ”POSITION” som sitter ovanför TIME/DIV. Center kan även ändras med piltangenterna till vänster om ”POSITION”

Vi skulle nu vilja ha Span = 20 kHz och Center = 10 kHz (Då blir 0 Hz i vänster kanten)
Men det går nog inte i detta läge…
För att åstadkomma detta måste beräkningen ske över fler perioder av signalen.
Klicka på ratten ”TIME/DIV”. Den kommer nu att påverka tidbasen (man ser att ramen kring det övre fönstret lyser lite mer). Vrid så att många fler perioder visas i fönstret upptill. Vrid tills TB: 5 ms visas längst upp till vänster.
Klicka en gång till på ”TIME/DIV”. Två vertikala markörer lyser nu lite tydligare i övre fönstret.
De anger vilken del av signalen som används för beräkningen. Vrid på ”TIME/DIV” så att så mycket som möjligt av signalen kommer med.
Klicka ytterligare en gång ”TIME/DIV” (nu lyser ramen kring kring det undre fönstret lite starkare). Nu kan Span ändras till 20 kHz. Ändra Center till 10 kHz (enklast med piltangenterna).

Nu bör du se en spik, en halv ruta in från vänsterkanten. Med ”VOLT/DIV” kan du ändra höjden. Justera så att du ser hela spiken men så stor som möjligt. Om du ser ytterligare spikar till höger har du råkat ut för en funktionsgenerator som är dålig på att generera rena sinussignaler. Byt i så fall till en annan funktionsgenerator, ev. till en HP3312A (nackdelen med den är den inte har en inbyggd frekvensräknare).

För att förenkla avläsningen kan vi använda markörer. Klicka på ”CURSOR MEASURE” och flytta markörerna med SELECT-ratten. Frekvenser och nivåer kan nu avläsas på skärmen. Flytta en markör så den hamnar på spiken.

2.1 Avläs på skärmen
Frekvens: 996Hz
Spänning: 680mv

Spänningen som visas är effektivvärdet. För en ren sinussignal är toppvärdet $\sqrt{2}$ x effektivvärdet.

2.2 Beräkna toppvärdet: $\sqrt{2} \times 680mV = 961mV$

2.3 Stämmer värdena med utsignalen från funktionsgeneratorn?

Värdena stämmer överens då Vpp = 2V vilket ger ett toppvärde på 1V

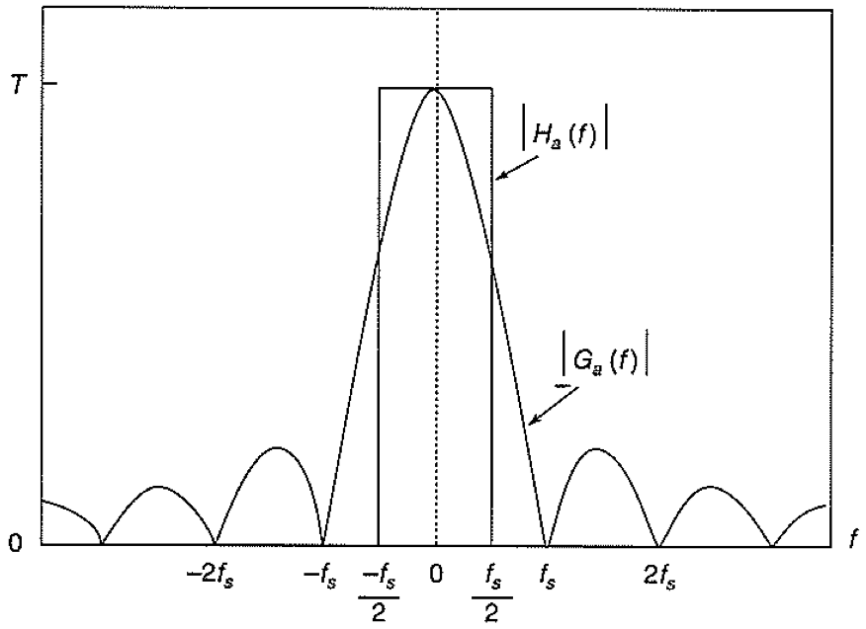
Spektrum för utsignalen
Vi skall nu studera spektrum för utsignalen som passerat D/A-omvandlaren på Arduinokortet.

Men vi startar med teorin. Enligt teorin fås spektrum för utsignalen genom att man utgår från spektrat från den tidsdiskreta signalen (som ju är periodiskt) och ser D/A-omvandlaren som ett filter med frekvensfunktionen

$$G(f) = e^{-j\pi\frac{f}{f_s}} \cdot \frac{\sin(\pi\frac{f}{f_s})}{\pi\frac{f}{f_s}}$$

Figur 1 visar absolutbeloppet av frekvensfunktionen $G_a(f)$[1] som är definierad som $T_s \cdot G(f)$
.

[1] Figuren är hämtad från J. Cartinhour, *Digital Signal Processing*, Prentice Hall, 2000

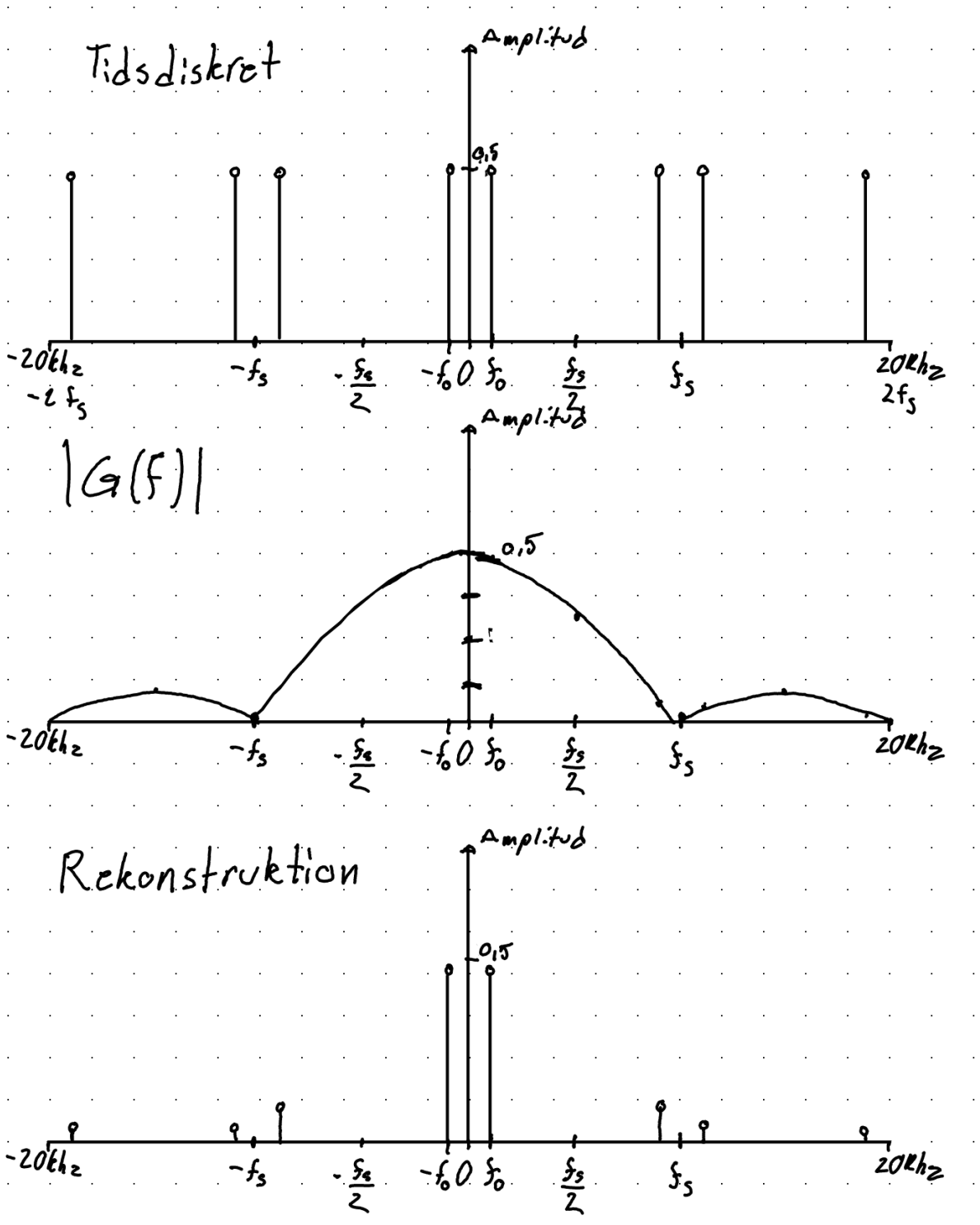


Figur 1. Absolutbeloppet av frekvensfunktionen för en ideal D/A-omvandlare, $H_a(f)$, respektive en ”vanlig” (0-ordningens) D/A-omvandlare, $G_a(f)$.

2.4 Rita i en figur på rutat papper dubbelsidigt spektrum upp ca ±20 kHz (jämför med övningsuppgift 13 och dess lösning)

- för den tidsdiskreta signalen när insignalen har frekvensen 1 kHz
- för $|G(f)|$
- för den rekonstruerade signalen

Fota och klistra in



Dags att även mäta på utsignalen! Insignal sinus 1 kHz, amplitud 1,0 V som tidigare. Klicka på knappen ”CH2”. Du bör nu spekrat för utsignalen och enligt teorin bör du se spikar på 1 kHz, 9 kHz, 11 kHz och 19 kHz.

Mät nivån på 9 kHz och 11 kHz spikarna och kolla om den stämmer med teorin, dvs. sätt in aktuella värden i formeln för $G(f)$ och jämför med mätningen.

(det är $|G(f)|$ som skall in i beräkningarna…)

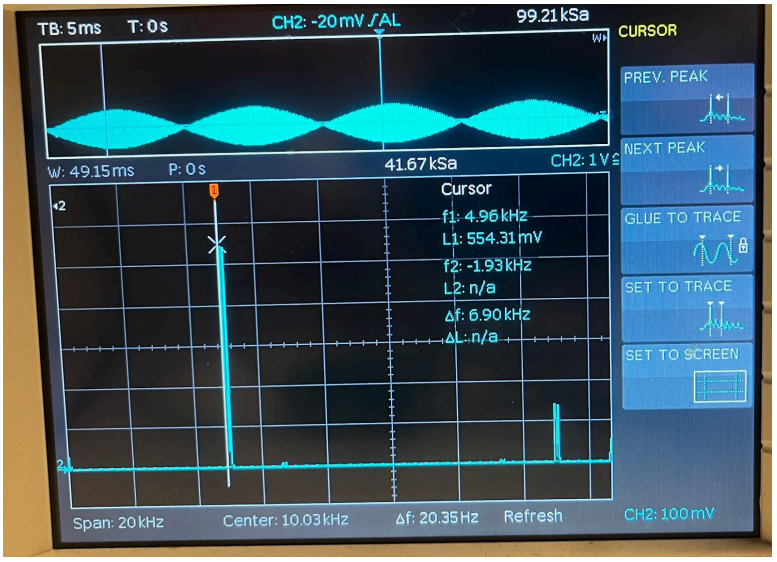
2.5 Redovisa både mätningar och beräkningar (gör gärna beräkningarna på papper, fota och klistra in)

Frekvens (kHz)	Uppmätt amplitud (mV)	|G(f)|	Beräknad amplitud (mV)
1	870	0,93	NA
9	94	0,11	95,7
11	74	0,085	74,0
19	30	0,048	41.8

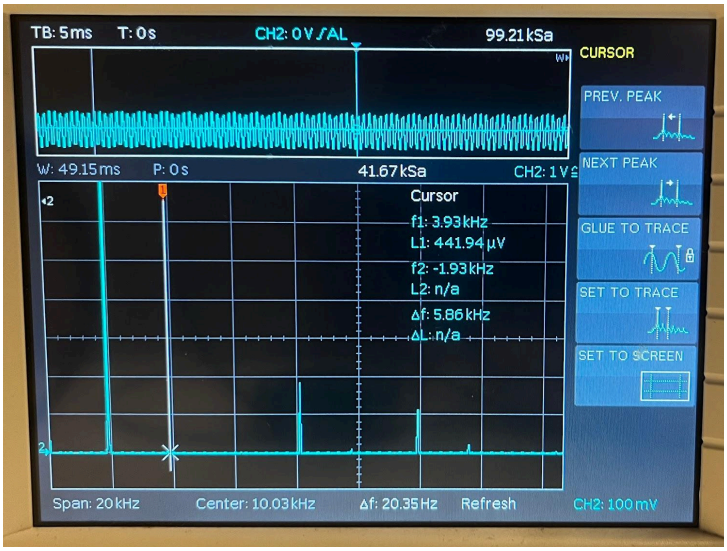
Öka långsamt frekvensen för insignalen inom området från 1 till 9 kHz. Lägg märke till hur spikarna på utsignalen rör sig.

2.6 Beskriv, kommentera och lägg in ett par foto med exempel, ange för varje foto insignalens frekvens. Tycks mätningarna stämma med teorin? (du behöver inte beräkna, bara uppskatta)

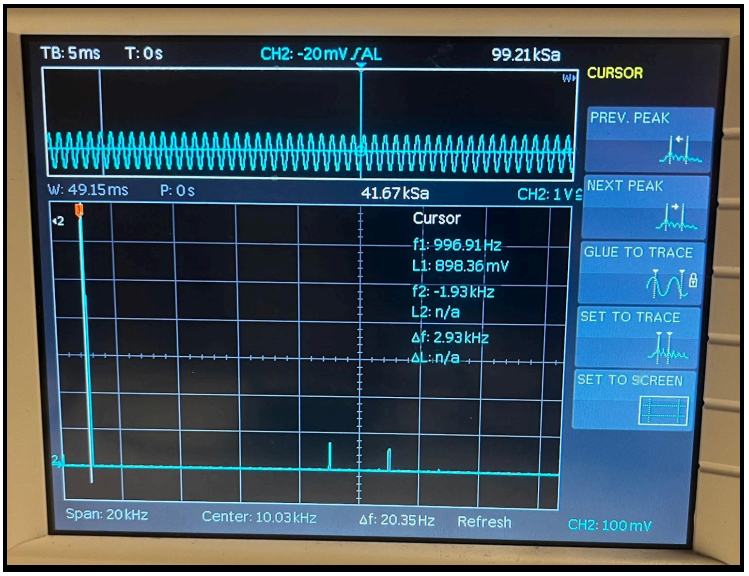
5kHz



8kHz



9kHz



Enligt våra uppskattningar stämmer våra mätningar mycket bra överens med teorin.
Amplituden på 5kHz skulle vara enligt teorin vara: $\frac{sin(0.5\pi)}{0.5\pi} * 870 = 553.85920mV$

2.7 Beskriv speciellt vad som händer med den spiken som först fanns vid 9 kHz när du ökar insignalens frekvensen till 9 kHz.
Spik vid 9kHz och 1 kHz byter plats när frekvensen ökas från 1 kHz till 9kHz.

Gå tillbaka och titta på fotot du tog vid mätningarna i tidsplanet då insignalens frekvens var 9 kHz.
2.8 Kommentar?
Utsignalen ser lika ut för 1 kHz och 9 kHz. Vikning…

Öka långsamt frekvensen för insignalen ytterligare upp till 19 kHz. Lägg märke till hur spikarna rör sig. Det tycks dyka upp spikar från vänster! Jämför med figuren du ritade med det dubbelsidiga spektrat.

2.9 Förklara!

Spikarna som dyker upp från vänster är de negativa frekvenskomponenterna som flyttas upp i det synliga området. När frekvensen går över 10kHz då dyker spiken vid $(-f_s + f_0)$upp.

När frekvensen närmar sig 19kHz kommer spiken att hamna på:

$f_s = 10kHz,\ f_0 = 19kHz \rightarrow f_{ut} = -10 + 19 = 9kHz.$

3. Exempel på signalbehandling

Nu när vi vet allt om sampling, tidsdiskret spektrum och rekonstruktion är det dags för ett första signalbehandlingsexempel!

Fördröjning

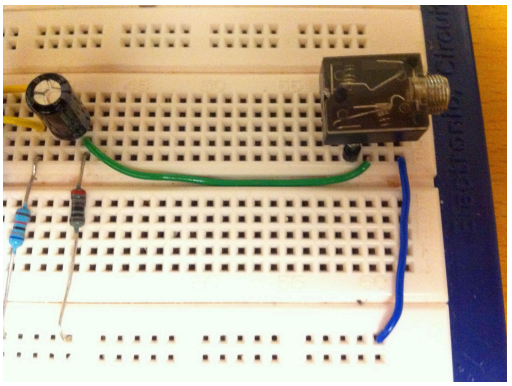
Vi vill testa att göra en fördröjning. Då behöver vi en signal som inte är periodisk

Det kan vi få med hjälp av en smartphone och ljudinspelning. **Tänk på att koppla bort signalgeneratorn!**

Vi använder en specialgjord kabel med 3,5 mm teleplugg i ena änden och kopplingstrådar den andra.

Den svarta (blå i någon kabel) kopplingstråden ansluts till jord och en av de andra (vi använder bara en kanal) ansluts till ingångssteget.

För att kunna lyssna på meddelandet vill vi kunna ansluta hörlurar till utgångssteget. Montera och anslut det speciella hörlursjacket enligt bilden nedan.



Figur 2. Hörlursjacket, anslutning längst till höger i bilden ansluts till jord

Låt gärna handledaren kontrollera kopplingen innan du ansluter din mobiltelefon och dina hörlurar.

Läs in ett kort meddelande, t.ex. ”*Hallå*” på smartphonen.

Anslut nu din smartphone och testa att signalen passerar igenom systemet som det skall och hörs i hörlurarna.

När detta fungerar är det dags att fundera hur ett program som skall ge en fördröjning kan se ut.

Om vi vill fördröja signalen 1 sekund skall varje sampel som A/D-omvandlas fördröjas 1 sekund innan det läggs ut på D/A-omvandlaren. Då måste vi spara 10000 värden i minnet eftersom vi samplar med 10 kHz. Om vi använder 32-bitars värden går det åt 4 x 10 kB. Eftersom det 12-bitars A/D- och D/A- omvandlare verkar ju detta som slöseri. Genom att använda 16-bitars variabler sparar vi utrymme. Processorn har 64 + 32 kbytes så det skall gå bra.

Koden för detta skulle kunna se ut så här:

```
static uint16_t buffer[10000]={0};

for (uint32_t k=9999;k>0;k--)
    buffer[k]=buffer[k-1];
buffer[0]=invalue;

outvalue = buffer[9999];
```

där invalue är värdet från A/D-ovandlaren och outvalue är värdet som skall läggas ut till D/A-omvandlaren.

3.1 Beskriv i ord hur koden är tänkt att ge en fördröjning:
Koden lägger in det senast lästa värdet i en buffer med storlek 10k. For loopen flyttar alla värden i buffern ett steg åt vänster. Efter 10k avläsningar kommer första värdet ligga sist i buffern och sättas som outvalue. Eftersom vi samplar 10k/s kommer det ta en sekund för första värdet att skickas till DAC.

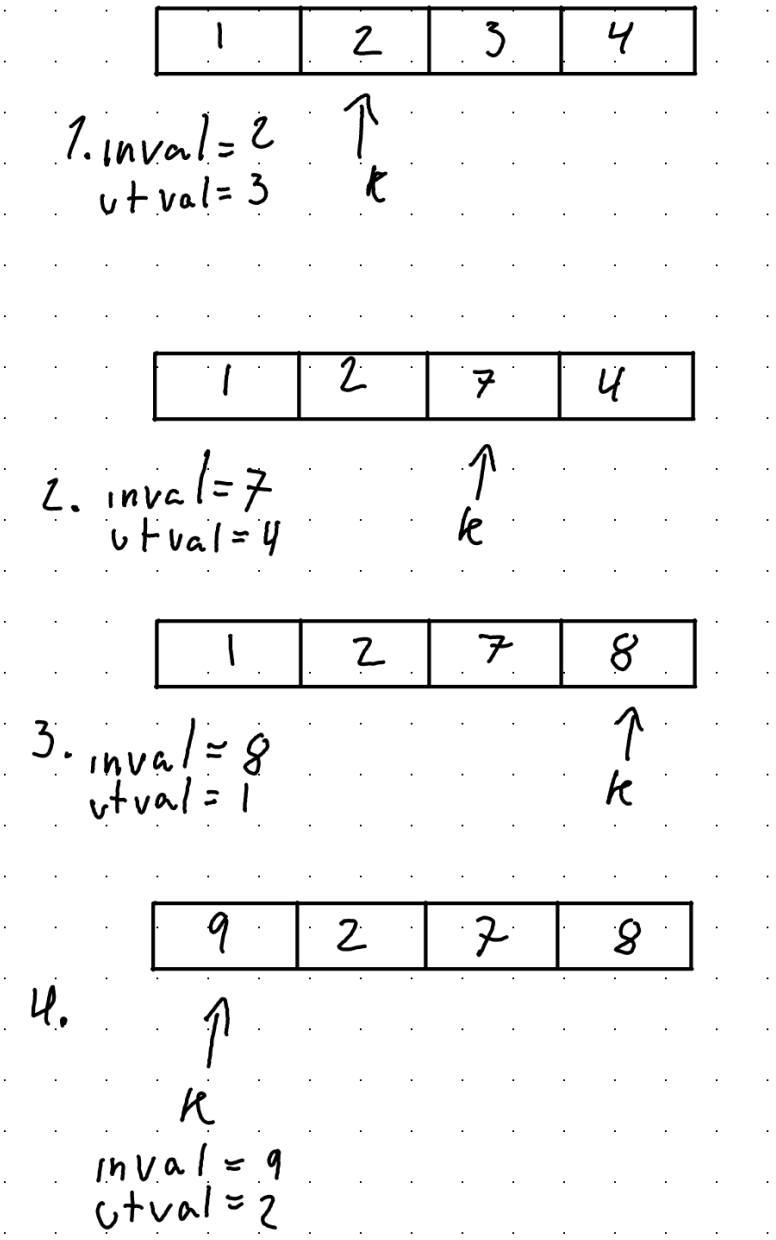
Det finns dock ett problem med ovanstående kod som gör att en mycket bättre lösning är:

```
static uint16_t buffer[10000]={0};
static uint32_t k=0;

buffer[k]=invalue;
k++;
if (k==10000) k=0;
outvalue = buffer[k];
```

Andra lösningen kan vara lite knepigare att förstå.
Tänk efter hur värdena läggs in buffer[]. Var ligger det senast inlagda värdet, var ligger det äldsta värdet när t.ex. k=1000?
Om du fortfarande tycker det är svårt, tänk en kortare buffert och att k maximalt är t.ex. 4. Då kan du rita upp hela arrayen och se vad som händer när du ”stegar” igenom kodsnutten)

3.2 Beskriv funktionen (rita gärna och fota din figur):
Funktionen ökar k-värdet efter k position i arrayen har satts till det avlästa värdet. Efter det skickar den ut värdet som ligger efter i arrayen. När k värdet når längden av arrayen wrappar den runt till 0. Vilket gör att ett värde som lagts till i arrayen inte skickas ut förrän det gjorts lika samplingar som arrayens längd.



3.3 Vad är fördelen med den sista koden?

Fördelen med sista koden är att du inte behöver loopa igenom arrayen varje sampling. Detta sparar tid.

3.4 Varför är buffert[] och k deklarerade som static i den sista koden? När nollställs de?

För att annars skulle arrayen bli nollad efter varje sampling. Att deklarera den som static gör att variabelns scope flyttas till programmets livslängd. vilket betyder att den behåller värdet efter att metoden returnerat.

För att kunna höra att det blir en fördröjning behöver vi ju höra båda den ursprungliga signalen och den fördröjda signalen men när kabeln med 3,5 mm teleplugg kopplas in på mobilen kopplas högtalaren bort.

Vi löser detta genom att lägga till det sista inkomna sampelvärdet till utvärdet.
Den slutliga koden blir:

```
static uint16_t buffer[10000]={0};
static uint32_t k=0;

buffer[k]=invalue;
k++;
if (k==10000) k=0;
outvalue = buffer[k]+invalue;
```

Lägg in koden i avbrottsrutinen, avsnittet för filterkod och ta bort raden `outvalue = invalue.`

Testa!
Spela upp ditt meddelande och lyssna med hörlurarna!
3.5 Kommentar?
Ljudet upprepas efter 1 sekund.

Mät upp beräkningstiden på samma sätt som i labbuppgift .

3.6
Hur lång tid tog avbrottskoden i labbuppgift 1? 52,02us
Hur lång tid tar hela avbrottskoden nu? 52,25us
Hur lång tid tar således den nya tillagda koden? 0,23us

Vi samplar ju nu med 10 kHz. Det är lågt med tanke på att vi kan höra ljud med frekvenser uppemot 20 kHz.
3.7 Hur stor borde sampelfrekvensen minst vara om vi skall undvika aliaseffekter och kunna rekonstruera ljudet hyggligt?
över 40kHz då människan kan höra ljud mellan 20hz - 20kHz. Ca 44100 samples/s brukar vara standard för ljud.

3.8 Skulle processorkapaciteten räcka till? Dvs. a) Räcker minnet? b) Räcker tillgänglig beräkningstid? Motivera svaren!
a) Ramminnet som kan alloceras under runtime är 160kb vilket skulle räcka. Vår array skulle ha storleken 40kb om vi använder oss utav uint16.

b) Då en sample tar 52,02us skulle den teoretiska max sampling frekvensen vara
$\frac{1}{52{,}02 * 10^{-6}} = 19223 \approx 19kHz\ sampling\ frekvens$
För att uppnå en samplingfrekvens på 44kHz skulle det behövas en samplingperiod på:
$\frac{1}{44*10^{3}} = 22,7\ \mu s$

Lämna in rapporten som pdf på Canvas. Var beredd på att kunna demonstrera programmet och förklara hur det fungerar.