MALMÖ HÖGSKOLA

Inbyggda system och signaler
Digital signalbehandling

Labuppgift 2
1504b

Namn 1 Leonard Holgersson

Namn 2 Hadi Deknache

Tommy Andersson
januari 2017

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 Its learning.

Laborationen genomförs som vanligt i par dvs. ni jobbar två och två. Vid inlämningen på Its learning anges vem som jobbat ihop.

Utrustning

- Labplatta med uppkoppling från labuppgift 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

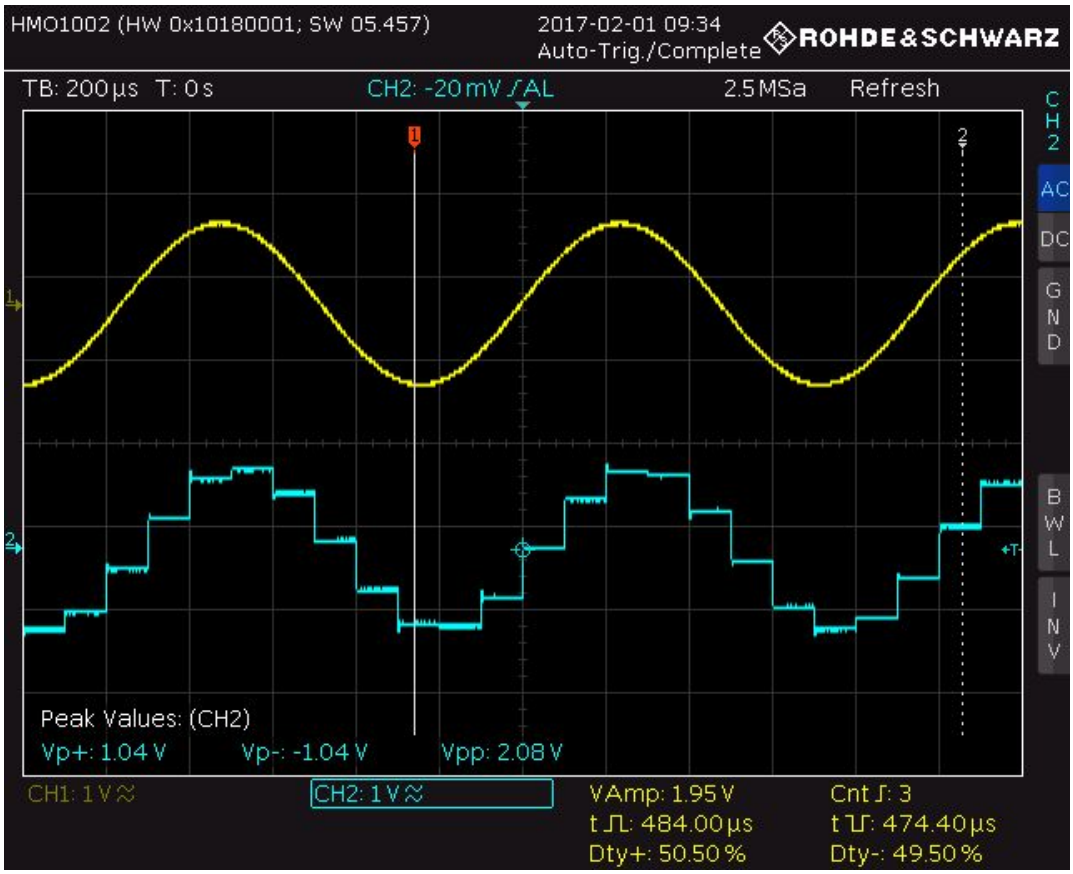
Mätningar i tidsplanet

Samma uppkoppling och program som i labuppgift 1 (1504a).
Använd Wavetek (samma som i labuppgift 1) som funktionsgenerator. Börja med sinus,1 kHz, amplitud 1,0 V (dvs 2,0 $V_{t\text{-}t}$).
Anslut och ställ in oscilloskopet så du kan se insignalen på kanal 1 överst och utsignalen på kanal 2 underst. Använd probar (x10).

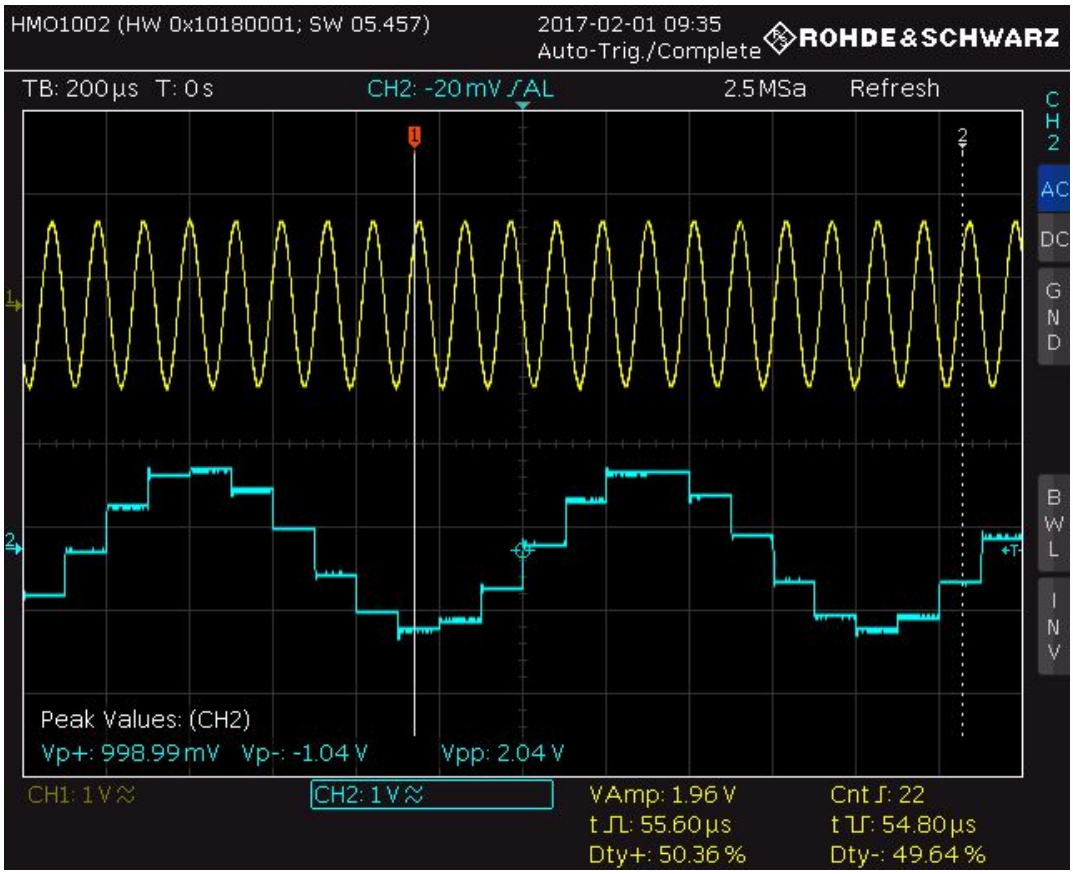
Ställ in tidbasen så du ser ca 2 perioder på skärmen. Trigga på kanal 2 (dvs. utsignalen). Öka frekvensen och se vad som händer. Behåll tidbasinställningen och fortsätt trigga på utsignalen.
Kolla speciellt vad som händer vid 9 kHz, 11 kHz, 19 kHz och 21 kHz!

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

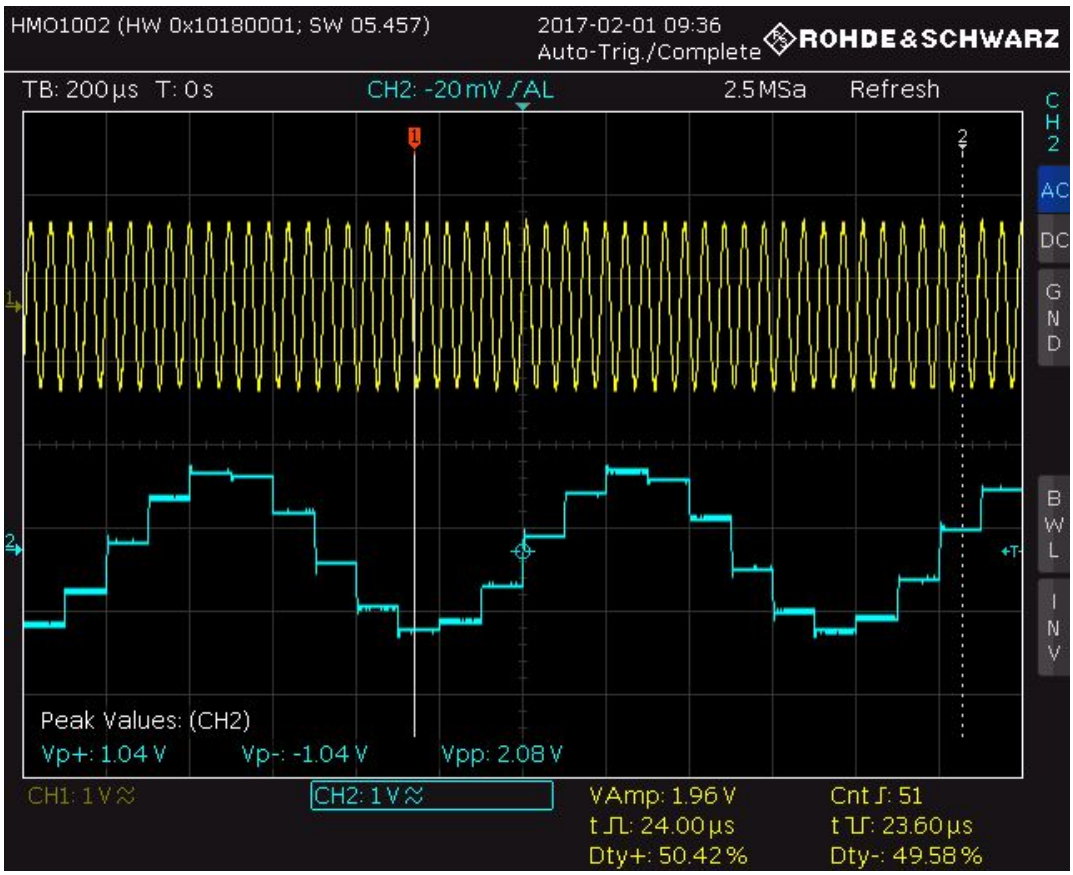
1 kHz:


HMO1002 (HW 0x10180001; SW 05.457)
2017-02-01 09:34
Auto-Trig./Complete
ROHDE&SCHWARZ
TB: 200 µs T: 0 s
CH2: -20 mV /AL
2.5 MSa
Refresh
CH 2
AC
DC
GND
BWL
INV
Peak Values: (CH2)
Vp+: 1.04 V
Vp-: -1.04 V
Vpp: 2.08 V
CH1: 1 V
CH2: 1 V
VAmp: 1.95 V
t ⎍: 484.00 µs
Dty+: 50.50 %
Cnt ∫: 3
t ⎏: 474.40 µs
Dty-: 49.50 %

9 kHz:


HMO1002 (HW 0x10180001; SW 05.457)
2017-02-01 09:35
Auto-Trig./Complete
ROHDE&SCHWARZ
TB: 200 µs T: 0 s
CH2: -20 mV ƒAL
2.5 MSa
Refresh
CH 2
AC
DC
GND
BWL
INV
Peak Values: (CH2)
Vp+: 998.99 mV Vp-: -1.04 V Vpp: 2.04 V
CH1: 1 V≈
CH2: 1 V≈
VAmp: 1.96 V
t ⎍: 55.60 µs
Dty+: 50.36 %
Cnt ∫: 22
t ⎏: 54.80 µs
Dty-: 49.64 %

21 kHz:


HMO1002 (HW 0x10180001; SW 05.457)
2017-02-01 09:36
Auto-Trig./Complete
ROHDE&SCHWARZ
TB: 200 µs T: 0 s
CH2: -20 mV ƒAL
2.5 MSa
Refresh
CH 2
AC
DC
GND
BWL
INV
Peak Values: (CH2)
Vp+: 1.04 V Vp-: -1.04 V Vpp: 2.08 V
CH1: 1 V≈
CH2: 1 V≈
VAmp: 1.96 V
t ⎍: 24.00 µs
Dty+: 50.42 %
Cnt ∫: 51
t ⎏: 23.60 µs
Dty-: 49.58 %

Vad som händer när vi satte den till 1kHz, 9kHz, 11kHz, 19kHz samt 21 kHz får vi samma kurva, eftersom samplingspunkterna sammanfaller på samma ställe som för 1,9,11,19 och 21 kHz. Insignalen förändras till repsektive frekvens men utsignalen kan inte tyda på vilken frekvens den samplats på för att återställa till samma frekvens. Detta eftersom fs återupprepas efter ett 10kHz, där en peak hamnar på +-1kHz .

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”. 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 ramVen 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.

Avläs på skärmen
Frekvens: 996.92kHz
Spänning: 650mV

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

Beräkna toppvärdet: 919 mV

Stämmer värdena med utsignalen från funktionsgeneratorn?

Enligt funktionsgeneratorn ska frekvensen vara 1 kHz, och toppvärdet 1 V. Skillnad i amplitud är ~8%, och skillnad frekvens är <0.4%.

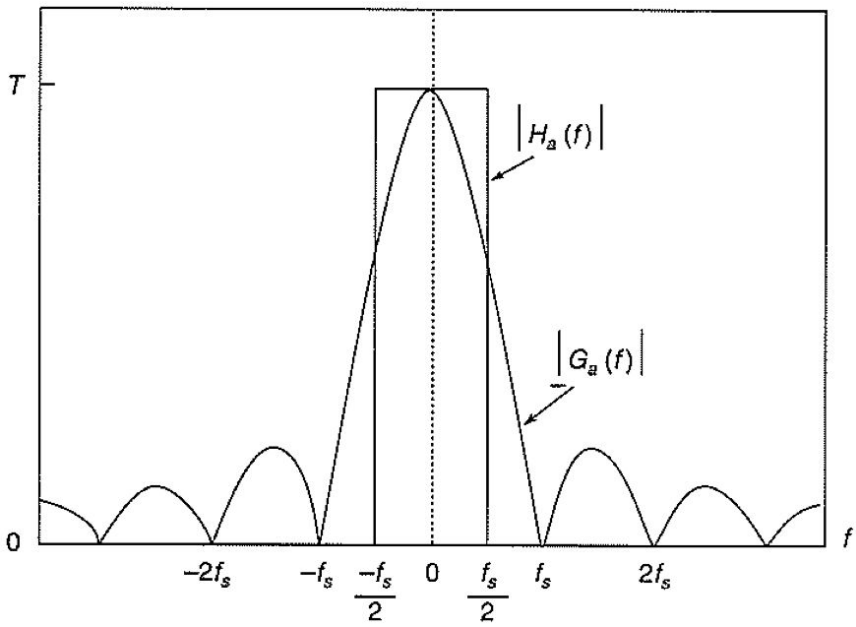
Spektrum för utsignalen
Vi skall nu studera spektrum för utsignalen som passerat D/A-omvandlaren på Arduinokortet.

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

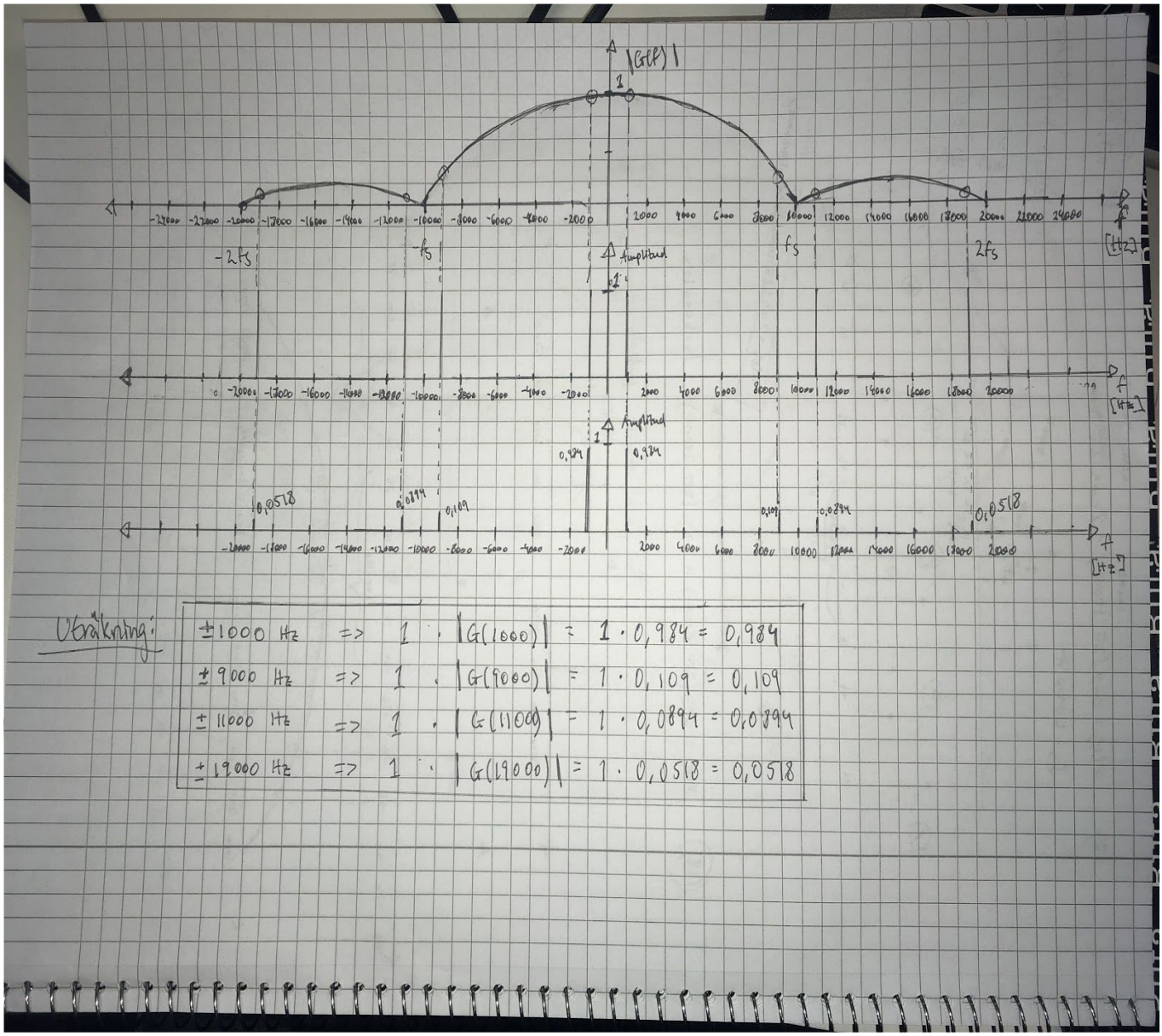


Figur 1. Absolutbeloppet av frekvensfunktionen för en ideal D/A-omvandlare, $H_a(f)$, respektive en ”vanlig” (0-ordningens) D/A-omvandlare, $G_a(f)$.

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.

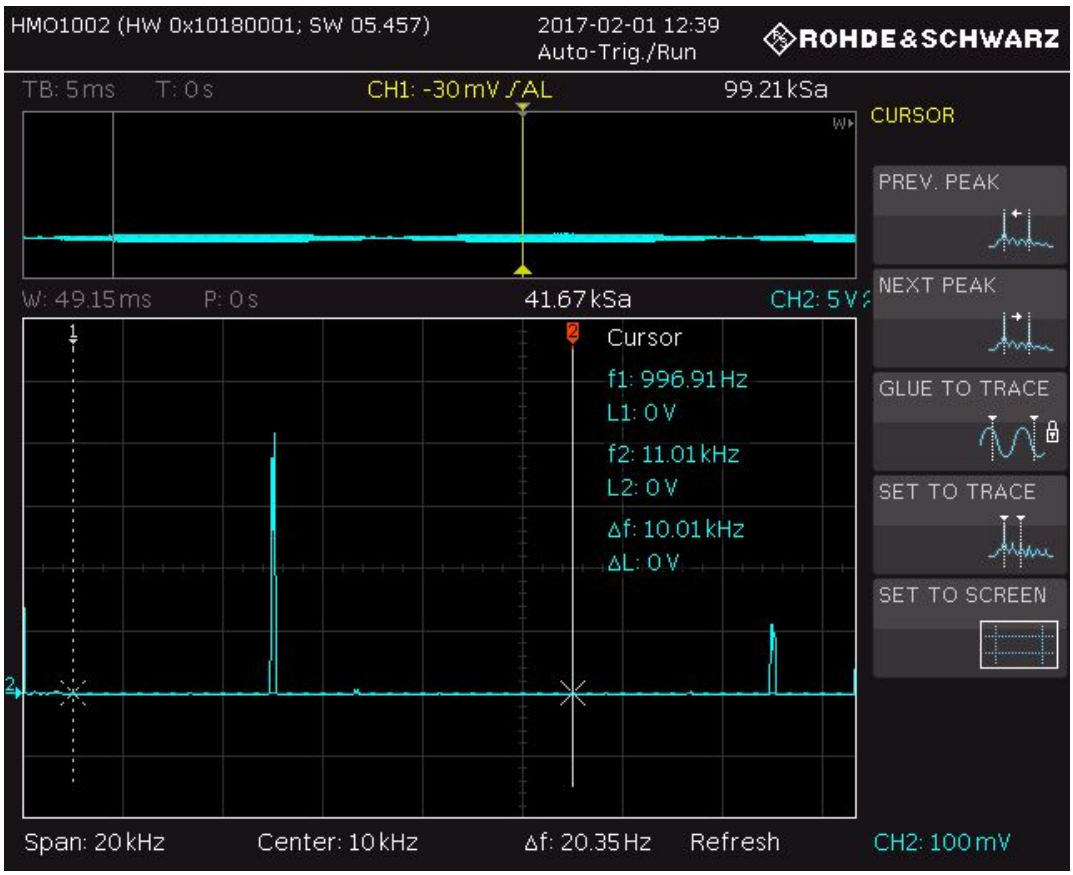
(det är $|G(f)|$ som skall in i beräkningarna…)

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

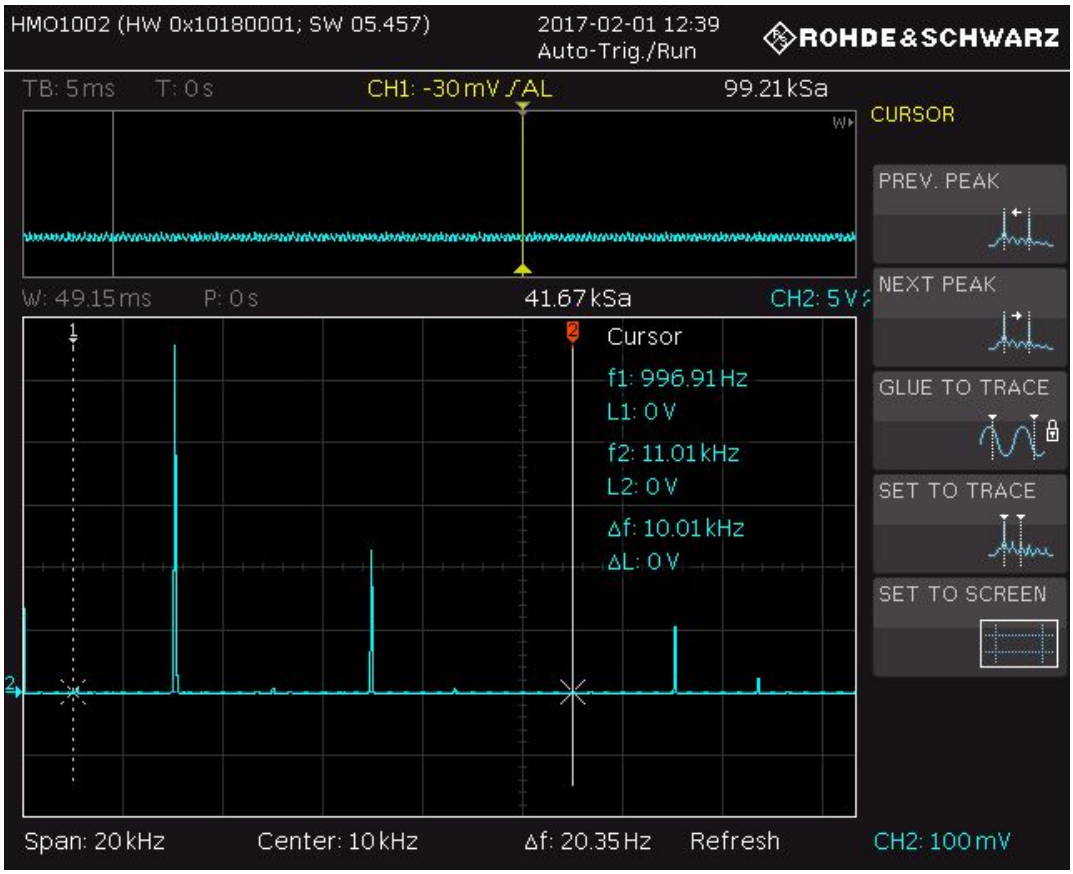
Spik [Hz]	Teoretisk spänning [V]	Uppmätt spänning [V]	Skillnad
9 000	$\frac{0,109}{\sqrt{2}} \approx 0,077$	0,067	0,01 V (13 %)
11 000	$\frac{0,0894}{\sqrt{2}} \approx 0,063$	0,053	0,01 V (16 %)

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

5 kHz:



7 kHz:



Samplingsfrekvensen är 10 kHz, så avståndet mellan samplingarna kommer alltid vara samma. När vi ökar frekvensen på signalgeneratorn ökar avståndet mellan de två spikar som genereras vid varje steg med den frekvens vi väljer på signalgeneratorn. Väljer vi exempelvis 5 kHz, kommer spikar från olika samplingar att överlappa varandra. Detta eftersom samplingsfrekvensen är 10 kHz och inmatad frekvens är 5 kHz, och 0+5 = 10-5, detta stämmer då enligt teorin då detta är halva samplingsfrekvensen.

Beskriv speciellt vad som händer med den spiken som först fanns vid 9 kHz när du ökar insignalens frekvensen till 9 kHz.

Spiken som tidigare befann sig vid 9 kHz befinner sig nu vid 1 kHz, detta eftersom när vi ökade frekvensen från funktionsgeneratorn förskjuter den dessa spikar som ligger i dess fs område.

Gå tillbaka och titta på fotot du tog vid mätningarna i tidsplanet då insignalens frekvens var 9 kHz.
Kommentar?

Spiken vid 1 kHz som var hög från början blir mindre och mindre samt förskjuts höger. Det som händer är att eftersom fs=10kHz så finns det signaler från vänster som förskjut och därför ser ut som att nya signaler kommer in. Det som händer är att samma samplingspunkt hamnar på samma position för 1kHz, 9kHz osv.

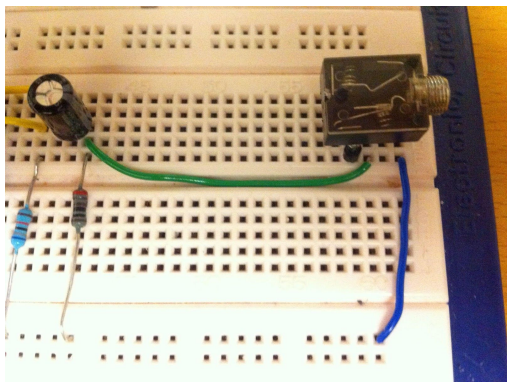
Öka långsamt frekvensen för insignalen ytterligare upp till 19 kHz. Lägg märke till hur spikarna rör sig. Det tycks dyka upp spikar från vänster! Jämför med figuren du ritade med det dubbelsidiga spektrat.
Förklara!
Eftersom vi har ett fs på 10 kHz så återupprepas signalen, vilket ger oss dessa 2 spikarna +-1000Hz och därför ser ut som att spikar kommer från vänster eftersom dessa förskjuts åt vänster och återupprepas. Spikarna kommer då från dess fs då man ökar frekvensen så förskjuts dem åt sidorna.

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.
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.

Beskriv i ord hur koden är tänkt att ge en fördröjning:

Den är tänkt att ge fördröjning genom att outvalue inte körs förrän for-loopen kört igenom alla 10 000 värden för buffert, samt att den måste skifta igenom alla värden innan ljudet som ska fördröjas spelas upp.

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)

Beskriv funktionen (rita gärna och fota din figur):

När element n spelas in, spelas element n+1 upp, utom när element 10 000 spelas in. Då spelas element 0 upp. Det sker således en fördröjning på 10 000 värden. Så att när man skapar en sorts variable vilket läser in på det värdet som spelades upp tidigare samtidigt så läser den ut ett värde efteråt vilket sparar på mycket tid genom att slippa forloopen.

Vad är fördelen med den sista koden?

Den kan spela in och spela upp samtidigt, till skillnad från första koden som spelar in alla värden innan den spelar upp någonting. Samtidigt slipper man köra en hel forloop för alla värden och bespara processortid och förhindra att interrupt triggas på ett annat interrupt.

Varför är buffert[] och k deklarerade som static i den sista koden?
När nollställs de?
Buffert och k är static eftersom dessa ligger i interrupt vilket kallas hela tiden och därför bör bevara dess värde mellan anropen. K nollställs när man har nått 10000 vilket börjar om på 0.

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!
Kommentar?
Ljudet upprepas när vi spelade upp ett ljud, den blir lite filtrerad men fortfarande brusig.

Mät upp beräkningstiden på samma sätt som i labuppgift 1(1504a).

Hur lång tid tog avbrottskoden i labuppgift 1? 3,28 µs
Hur lång tid tar hela avbrottskoden nu? 3,62 µs
Hur lång tid tar således den nya tillagda koden? 0,34 µs

Vi samplar ju nu med 10 kHz. Det är lågt med tanke på att vi kan höra ljud med frekvenser uppemot 20 kHz.
Hur stor borde sampelfrekvensen minst vara om vi skall undvika aliaseffekter och kunna

rekonstruera ljudet hyggligt?

40 kHz, eftersom man då får två punkter per våg vid 20 kHz, och bättre kan rekonstruera signalen.

Skulle processorkapaciteten räcka till? Dvs. a) Räcker minnet? b) Räcker tillgänglig beräkningstid? Motivera svaren!

a) om en sekunds fördröjning önskas kommer buffert-innehållet ta 80 kilobyte, då varje värde sparas i 16-bitars heltal, och vi behöver 40 000 av dem. (Vid 10 kHz behövdes 10 000, så vid 40 kHz kommer vi behöva 4 gånger så många, eftersom koden kommer jobbas igenom 4 gånger snabbare). Resten av koden tar inte så mycket minne, så de 96 kilobyte som finns tillgängligt ska räcka!

b) avbrottskoden tar 3,62 µs. för att hinna med att göra beräkningarna krävs det att koden tar mindre än 1/40000 sekunder, alltså 25 µs. Detta ska alltså inte heller vara några problem! Pulsavstånden mellan pulserna ligger runt 100µs/4 för 40kHz vilket då betyder att vi ligger bra till med tiden och därför räcker tiden gott.

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

Frivillig extra uppgift

Genom att lägga ett analogt lågpassfilter efter D/A-omvandlaren kan utsignalen snyggas till något.
Fundera över lämplig brytfrekvens med tanke på aliasfrekvenserna och designa ett enkelt LP-filter (se t.ex. övningsuppgift 1). Dimensionera *R* så du kan använd en överbliven kondensator från din komponentlåda.
Koppla in på lämpligt ställe i din koppling.
Mät både i tidsplanet och frekvensplanet, fota representativa exempel.
Kör även fördröjningskoden. Blir det någon förbättring av ljudkvaliteten?
Redogör för vad du gjort och resultatet (med foto).