Designnotat

Tittel: FSK-demodulator

Forfatter: Karl Henrik Ejdfors

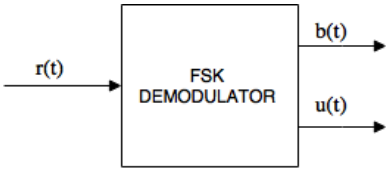
Versjon: 2.0	Dato: 1. desember 2017

Innhold

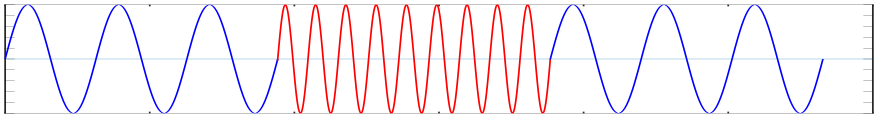
1 Problembeskrivelse

Innen digital kommunikasjon er behovet for å sende binære sekvenser fra en sender til en mottaker stort. Innen blant annet radiokommunikasjon er det nødvendig at det utsendte signalet holdes innen et bestemt frekvensbånd [1]. En måte å tilfredsstille dette kravet er å bruke frekvensskift-modulasjon (FSK). I FSK er amplituden til inngangssignalet konstant, men frekvensen varierer mellom to verdier f_0 og f_1, der $f_0 < f_1$.

I dette designet skal det implementeres en FSK-demodulator, som vist i figur 1.1. Demodulatoren sender ut bit sekvenser i form av et tidsvarienrende firkantsignal basert på hvilken frekvensen som er påtrykket inngangssignalet $r(t)$. Inngangssignalet er sinusformet og skifter mellom frekvensene f_0 og f_1, visualisert i figur 1.2. Når f_1 er påtrykket inngangssignalet, er $b[n]$ logisk høy, og logisk lav når $r(t) = f_0$. Utgang $b(t)$ sender ut firkantsignalet og $u(t)$ indikerer om det er påtrykket et signal på $r(t)$. Implementasjonen av demodulatoren skal være på under 4cm^2.



Figur 1.1: Oversiktsfigur over FSK-demodulator.



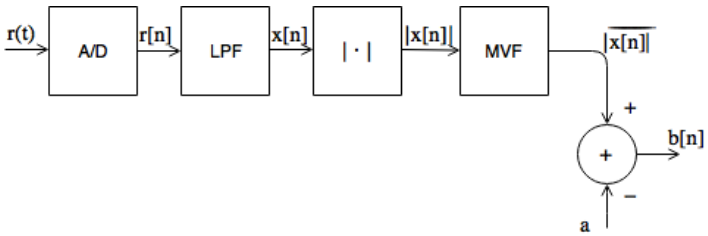
Figur 1.2: Inngangssignal $r(t)$.

2 Prinsipiell løsning

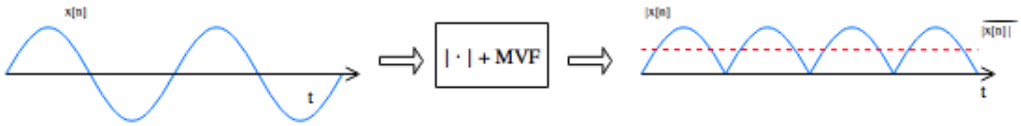
En FSK-demodulator kan designes på mange måter. Et mulig design er vist i figur 2.1, der a er den diskrete amplitudeverdien til f_0. Signalet $r(t)$ punktprøves til et tidsdiskret signal $r[n] = r(nT_s), n \in \mathbb{Z}$, der $T_s = \frac{1}{f_s}$ er punkprøvingsintervaller, og f_s er punktprøvingsfrekvensen.

En ATmega328P mikrokontroller [2] tilfredsstille kravet om at implementasjonen skal være mindre enn 4cm^2. En Arduino Uno er basert på denne mikrokontrolleren, og har en A/D omformer med 10bit oppløsning, som mapper $0 - 5$V til diskrete verdier $0 - 1023$.

Systemet i figur 2.1 tar inn et inngangssignal $r(t)$ som blir punktprøvet i en A/D omformer. De diskrete verdiene $r[n]$ blir filtrert i et digitalt lavpassfilter. Etter filtrering, $x[n]$, tas absoluttverdien av signalet, før det blir sendt til middelverdifilteret MVF. Denne verdien, $|\overline{x[n]}|$, blir sammenlignet med konstanten a, som blir presentert i neste avsnitt. Prosessen er fremstilt grafisk ved figur 2.2.



Figur 2.1: Mulig implementasjon logikk bak signal $b[n]$. Modifisert fra [1].



Figur 2.2: Visualisering av signalets forplantning gjennom systemet. Modifisert fra [1].

Ved å analysere frekvensspekteret til inngangssignalet $u(t)$ kommer frekvenskomponentene f_0 og f_1 tydelig frem. Dette kan brukes til å designe et filter som favoriserer den ene frekvensen. Ved å sende signalet gjennom dette filteret, vil det dempe amplituden til den andre frekvensen. Middelverdien av absoluttverdien over et antall punktprøver gir et estimat av amplituden av det filtrerte signalet. Det glidende vinduet av punktprøvinger må være stort nok til å ha med minst en periode av signalet. Ved å sammenligne amplituden til det filtrerte signalet med konstanten a, gitt ved (2.1), skilles frekvensene fra hverandre og sende ut et representativt firkantsignal på utgangen. A er amplituden til f_0.

$$a = \frac{2^{10}}{5\text{V}} \cdot \frac{A}{2} \cdot (A - |H(\omega_0)|), \quad \omega_0 = 2\pi \frac{f_0}{f_s} \tag{2.1}$$

Et digitalt middelverdifilter med punkprøver $x[n]$ har utgangssignal gitt ved (2.2), og kan brukes som digitalt lavpassfilter. N er lengden av filteret.

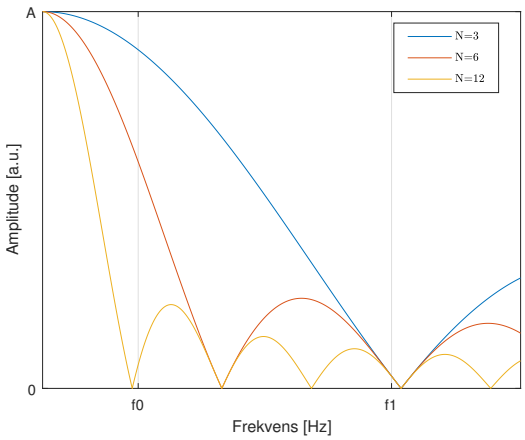
$$y[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n-k] \tag{2.2}$$

Amplituderesponsen til lavpassfilteret er gitt ved (2.3), og visualisert i figur 2.3.

$$|H(\omega)| = \frac{1}{N} \frac{|\sin(\frac{N}{2}\omega)|}{|\sin(\frac{\omega}{2})|}, \quad \omega = 2\pi \frac{f_1}{f_s}, \quad f_s \geq 2 \cdot f_1 \tag{2.3}$$

Det er ideellt at den høye frekvensen f_1 skal bli dempet mye i forhold til f_0. På denne måten er det lett å skille frekvensene fra hverandre, og å registrere om det er aktivt signal på inngangen.

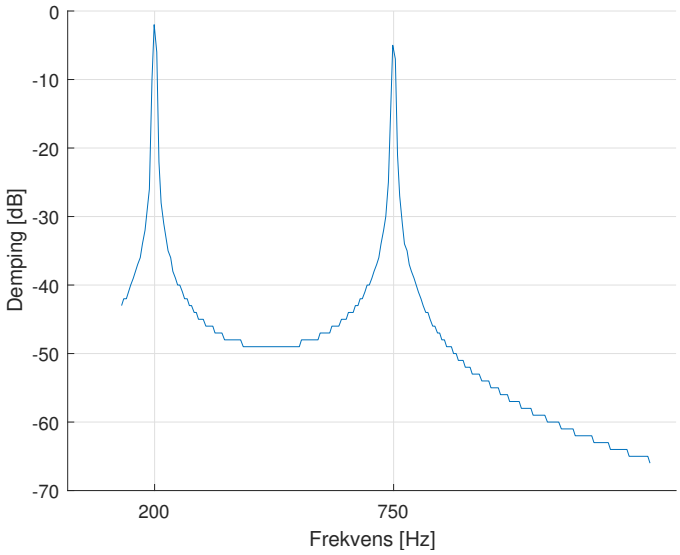
Fra figur 2.3 er det tydelig at stor verdi av N gir bratt dempingen. Siden målet er å skille f_0 og f_1 fra hverandre, kan $N = 3$ gi en respons som gir et vel definert passbånd.



Figur 2.3: Amplituderespons til lavpassfilter med ulik verdi N.

3 Realisering

Ved analyse av frekvensspekteret til inngangssignalet $r(t)$, vist i figur 3.1, er det tydelig at signalet inneholder frekvenskomponentene $f_0 = 200$Hz og $f_1 = 750$Hz.



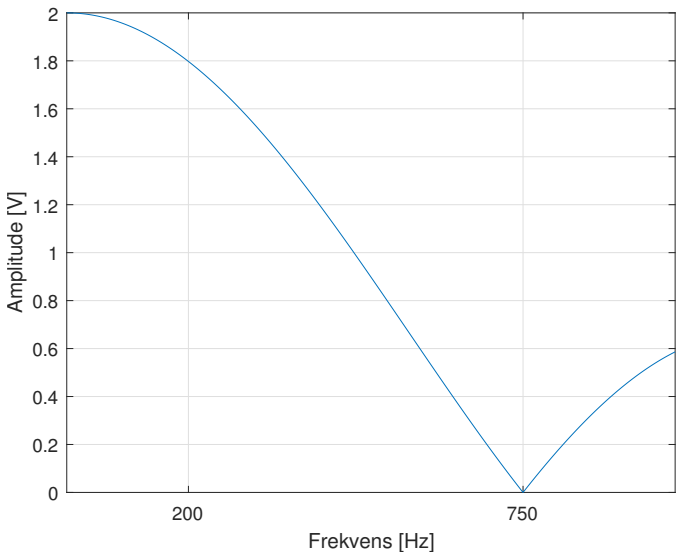
Figur 3.1: Frekvensspekter til $r(t)$.

Av amplituderesponsen gitt ved (2.3) med $N = 3$, kommer uttrykket for punktprøvingsfrekvensen f_s frem:

$$|H(\omega)| \approx 0 \Longrightarrow \sin(\frac{3}{2}\omega) \approx 0$$
$$\omega = 2\pi\frac{f_1}{f_s}k \approx \frac{2}{3}\pi k, \quad k \in \mathbb{Z}$$
$$f_s \approx 3f_1 = 2250\text{Hz}$$

Det er ikke ønskelig at f_1 skal dempes maksimalt, så dermed settes $f_s \approx 2250$Hz.

Med amplitudeverdi 2V på inngangssignalet, blir amplituderesponsen til f_0 og f_1 henholdsvis $|H_0(\omega_0)| = 1.8$V og $|H_1(\omega_1)| \approx 0$, som vist i figur 3.2.



Figur 3.2: Amplituderespons til system.

Konstanten a i figur 2.1 blir den diskrete amplituden til f_0 ved (2.1),

$$a = \frac{1024}{5\text{V}} \cdot \frac{2\text{V}}{2\text{V}} \cdot (2\text{V} - 1.8\text{V}) = 41.$$

Ved å bruke et glidende vindu på 20 punktprøvinger, vil minst en periode av signalet bli punktprøvet, og absoluttverdien til middelverdien av disse være presentabel for den diskrete amplituden til signalet. Ved implementasjon av koden i vedlegg A får man dermed firkantsignalet på utgang b vist i figur 3.3, som følger logikken gitt i listing 1.

Listing 1: .]Logikk for utgangssignal $b[n]$.

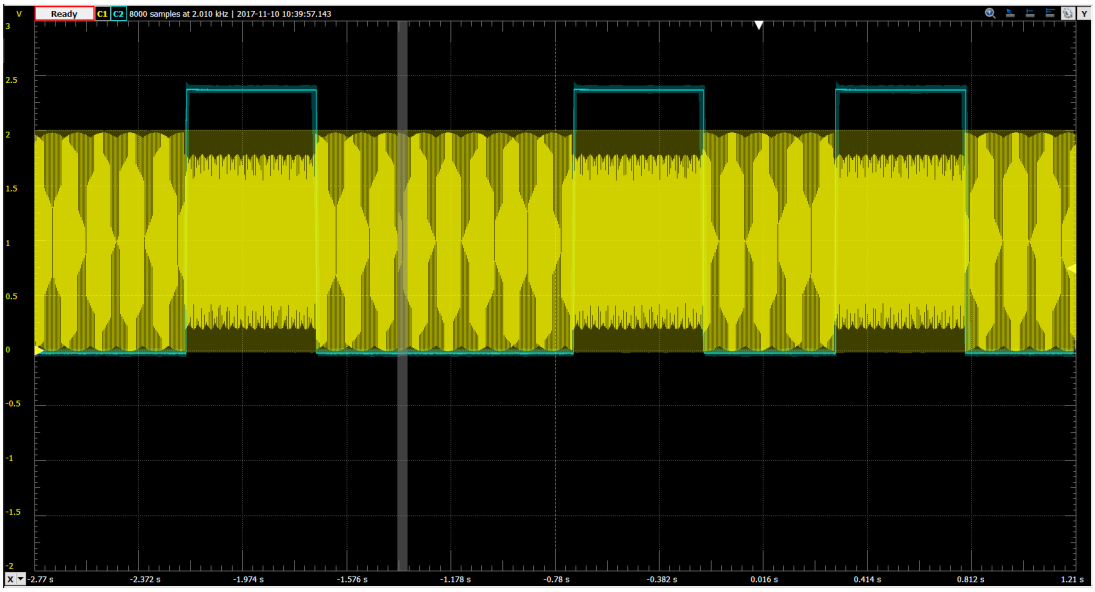
```
1    if(absAvg<41){ // Gjennomsnitt mindre enn 41 gir f_0
2       digitalWrite(8,HIGH);
3     } else {
4       digitalWrite(8,LOW);
5     }
```

Utgangssignalet u følger logikken gitt i listing 2.

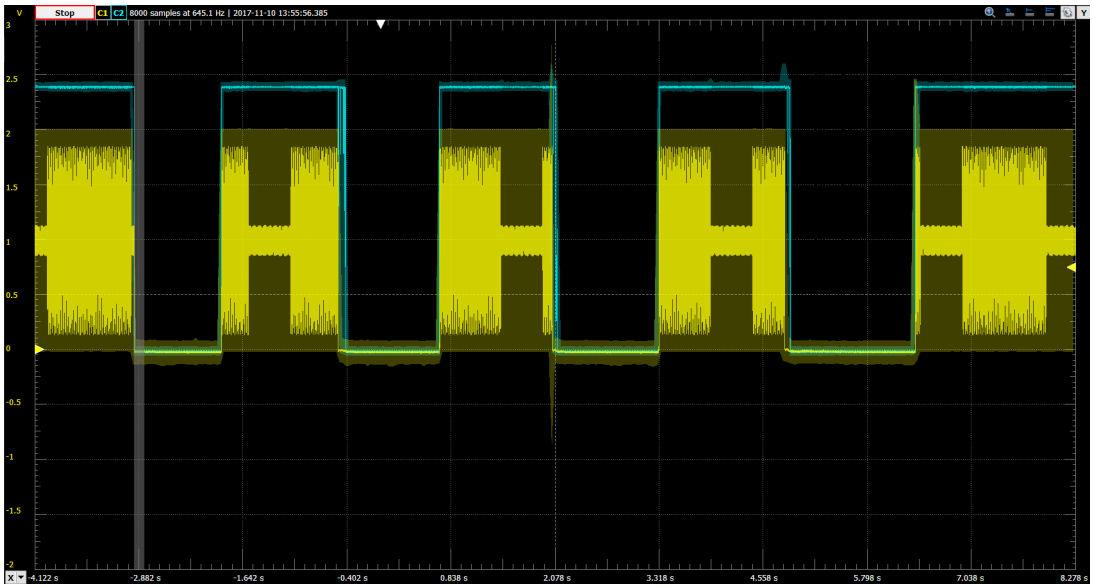
Listing 2: .]Logikk for utgangssignal $u[n]$.

```
1     if(!sample){   // Ikke noe signal
2       digitalWrite(7, LOW);
3     } else {
4       digitalWrite(7, HIGH);
5     }
```

Dette gir henholdsvis utgangssignalene $b(t)$ og $u(t)$ vist i figur 3.3 og 3.4.



Figur 3.3: Gul er inngangssignal $r(t)$ og blå er utgangssignal $b(t)$.



Figur 3.4: Gul er inngangssignal $r(t)$ og blå er utgangssignal $u(t)$.

4 Konklusjon

FSK-demodulatoren er implementert på en ATmega328P på et Arduino Uno utviklingskort. Inngangssignalet $r(t)$ inneholder frekvenskomponentene $f_0 = 200$Hz og $f_1 = 750$Hz. Samplingsfrekvensen satt til $f_s \approx 2250$Hz, og det digitale lavpassfilteret har $N = 3$. Ved implementasjon av koden i vedlegg A blir utgangssignalet $b(t)$ loisk høy når $r(t) = f_1$ og lav når $r(t) = f_0$. Utgangssignalet $u(t)$ er logisk høy når det er påtrykt et signal på inngangen.

5 Takk

Takk til medstudenter Ole Bjørn Eithun Pedersen og Torstein Langan for fruktbare diskusjoner rundt mulige implementeringer av FSK-demodulatoren.

Referanser

[1] Lars Lundheim. *Digital kommunikasjon med FSK*. NTNU, 2017.

[2] Atmel. *ATmega328P Datasheet*, 2015.

A Arduinokode for FSK-demodulator

```
// Eksempel kode for oppsett av timer-interrupt for en konstant samplingsfrekvens

#include <TimerOne.h>
/* Om du ikke har innstallert dette biblioteket
g  til Sketch -> Include library -> Manage Libraries
S k opp TimerOne og innstaller*/

// Globale variaber
volatile int sample; // Holder siste sample
bool newSample; // St tte varibel for  sjekke om ny sample er tatt

const int SAMPLING_PERIOD = 444; // f_1 = 750 Hz, F_s = N*f_1*1/n, T = 1/F_s
const int N = 3;
const int M = 20; // Glidende vindu

const int AMPLITUDE = 2; //V

int samplingList[N] = {0};
int avgList[M] = {0};

long sum = 0;

int n = 0;
int m = 0;

double absAvg = 0;

void setup() {
  // Oppsett av timer interrupt
  Timer1.initialize(SAMPLING_PERIOD); // 500 mikrosekund mellom hver sample -> gir F_s = 2kHz
  // Argumentet i "attachInterrupt" bestemmer hvilken funskjon som er interrupt handler
  Timer1.attachInterrupt(takeSample);
  Serial.begin(9600);
}

void loop() {
  if(newSample){
    samplingList[n] = sample-(1024*AMPLITUDE/(5*2)); // 10 bit oppl sning p  ADC, sampler verdier mellom 0-5V
    sum = 0;
```

```
40          for (int i = 0; i < N; i++) {
41            sum += samplingList[i];
42          }
43
44          avgList[m] = sum/N;
45
46          sum = 0;
47          for (int i = 0; i < M; i++) {
48            sum += abs(avgList[i]);
49          }
50
51          absAvg = sum/M;
52
53          //Serial.println(sample);
54
55          if(absAvg<41){ // Gjennomsnitt mindre enn 41 gir f_0
56            digitalWrite(8,HIGH);
57          } else {
58            digitalWrite(8,LOW);
59          }
60
61          if(!sample){  // Ikke noe signal
62            digitalWrite(7, LOW);
63          } else {
64            digitalWrite(7, HIGH);
65          }
66
67          newSample = false;
68          m++;
69          n++;
70          if (n == N){
71            n = 0;
72          }
73          if (m == M){
74            m = 0;
75          }
76        }
77      }
78
79      // Interrupt-handler (denne kalles ved hvert interrupt)
80      void takeSample(void){
81        sample = analogRead(0); // Sampler p  A0
82        newSample = true;
83      }
```