NTNU

elsys
ELEKTRONISK SYSTEMDESIGN & INNOVASJON

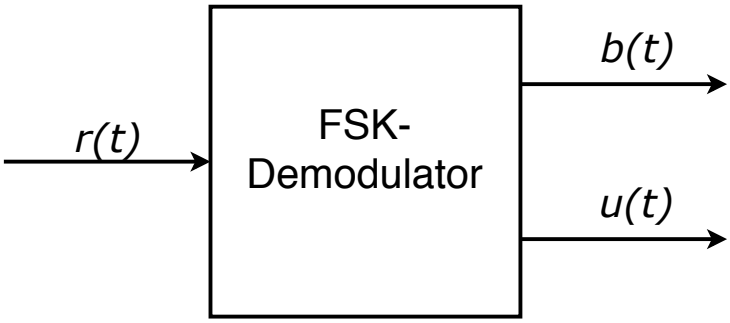
Designnotat

Tittel: FSK-Demodulator	
Forfattere: Øyvind Skaaden	
Versjon: 2.0	Dato: 1. desember 2019

Innhold

---

1 Problembeskrivelse



Figur 1: En prinsipiell FSK-demodulator. Har inngangen $r(t)$ og utgangene $b(t)$ og $u(t)$.

Det å overføre data er en viktig oppgave innenfor elektronikk. Det kan gjøres på veldig mange måter, som for eksempel å gjøre det direkte ved å sende digitale pulser. Men i andre tilfeller ønsker vi at signalet skal være så simpelt som overhodet mulig.

Et sinus-signal har den egenskapen at den er veldig enkel og har en veldig definert oppførsel gjennom veldig mange systemer og medier. Si hvis du skal sende et radio-signal er et sinus-signal ofte det beste signalet. Men hvordan skal vi overføre informasjon gjennom et sinus-signal? Igjen er det mange måter å gjøre det på men en av de er å endre litt på frekvensen til signalet, såkalt FSK (Frekvensskift-modulasjon [1]).

For å lese av informasjonen som er modulert av FSK, må vi ha en FSK-demodulator som i figur 1.

Her vil systemet ta inn et FSK-signal på inngangen $r(t)$, og utgangen $b(t)$ vil være det demodulerte signalet. Utgangen $u(t)$ vil fortelle status på om det kommer inn et FSK-signal som demoduleres til ugangen $b(t)$.

Inngangssignalet vil inneholde to frekvensen f_0 og f_1. Vi ønsker at $b(t) =$ HØY når f_1 er på inngangen $r(t)$ og $b(t) =$ LAV når f_0 er på inngangen $r(t)$.

Den ferdige demodulatoren må også ha et areal mindre enn 4cm^2.

2 Prinsipiell løsning

Det å lage en enkel FSK-demodulator, kan gjøres på mange måter. Det går ann å bruke digital signalprossesering og digitale filtere for å oppnå ønsket oppførsel. Men her baseres vi oss på å måle perioden på signalet som kommer.

Det er ønskelig å lage et firkantpuls-tog med samme frekvens som inngangssignalet, fordi det er mye letter å måle perioden, eller bredden, på signalet med et signal med en brå kant når det skal leses av med en mikrokontroller.

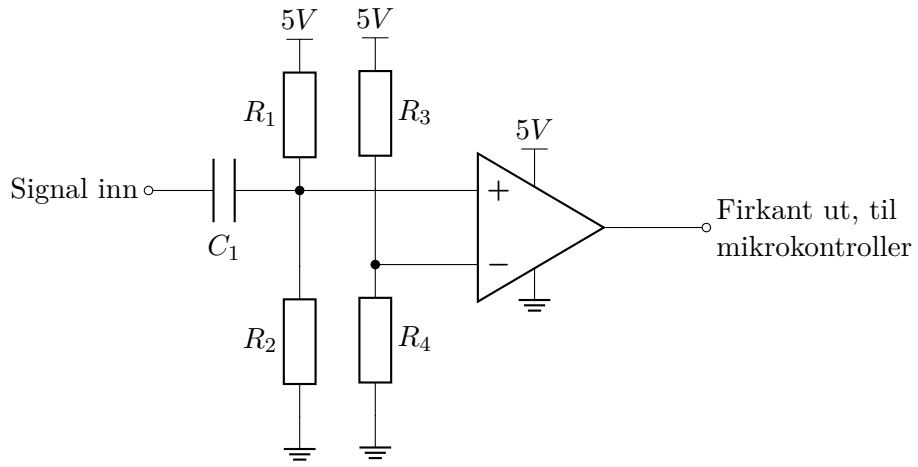
Vi trenger en komparator som kan gjøre om et sinussignal til et firkantpuls-tog med samme frekvens. En enkel komparatorkrets er som i figur 2.

Komparatoren har en spennings-bias på inngangene. Dette er for at den skal kunne fungere med en enkel spenningskilde. Da lager vi en virtuel jord med mostandene R_3 og R_4 og flytter nullpunktet til inngangen like mye. For enkelhetens skyld, pleier alle motstandene å være like store, i størrelsesorden $1k\Omega$ til $100k\Omega$ grunnet komparatoren. Kondensatoren C_1 må kun være tilstrekkelig stor for å ikke endre på det orginale signalet.

Ved å ha signalet som er firkantpuls med samme periode eller frekvens som det orginale signalet kan vi bruke signalet til å trigge en interupt på en mikrokontroller og måle perioden mellom interuptsene. Vi kan da enkelt regne ut frekvensen med (1), der perioden er T og frekvensen f.

$$f = \frac{1}{T} \tag{1}$$

Etter å ha regnet ut frekvensen er det så enkelt som å sjekke om frekvensen som leses er enten f_0 eller f_1 for å så sette utgangene $b(t)$ og $u(t)$ etter kravene i avsnitt 1.



Figur 2: Enkel komparator-krets for enkel strømforsyning. Tar inn et periodisk signal, og på utgangen er det et firkanpuls-tog med samme periode.

3 Realisering og test

3.1 Realisering

For å realisere kretsen vil vi bruke en Arduino Uno, med mikrokontrolleren ATmega328P [2]. Denne finnes i to størrelser, der den ene er under 1cm^2. Vi kommer også til å bruke en LF353-P [3] operasjons-forsterker som komparator.

Som motstander bruker vi $20k\Omega$ motstander. Dette vil sette spenningsbiasen inn på komparatoren til ca 2.5V. Kondensatoren ble valgt til 1μF.

Vi velger utganene D6 og D7 på mikrokontrolleren som utganger til henholdsvis $u(t)$ og $b(t)$. Biblioteket som blir brukt til å måle frekvensen bruker pinne D8. Så inngangen $r(t)$ skal inn på D8.

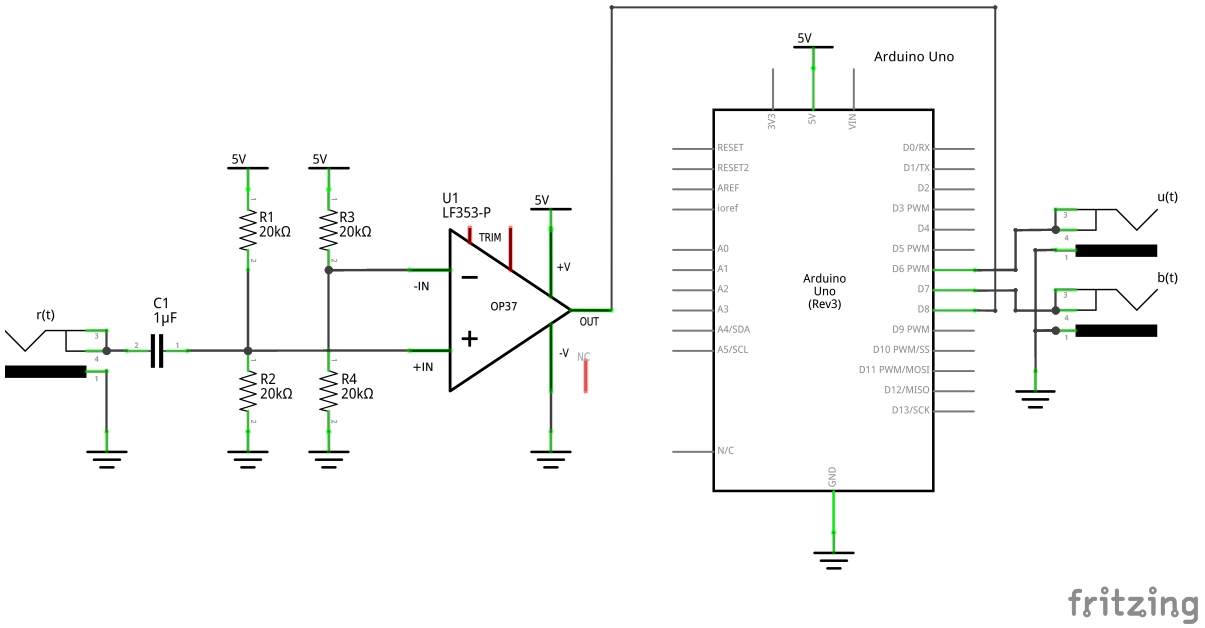
Biblioteket som blir brukt heter *FreqMeassure* [4]. Det måler frekvensen på pinne D8 på en Arduino. Frekvensen kan da taes gjennomsnitt av og deretter brukes til å bestemme hvordan $u(t)$ og $b(t)$ skal oppføre seg. Biblioteket krever også at signalet er enten logisk høy eller lav, altså et firkantpuls-tog.

Ferdig krets som i figur 3.

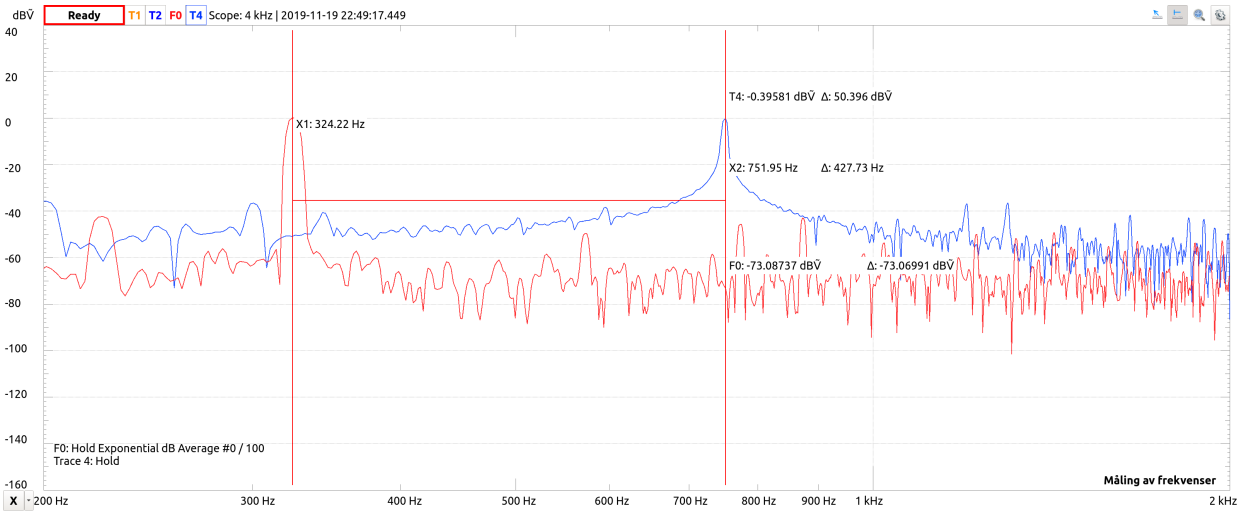
For å finne frekvensene FSK-demodulatoren skal fungere på, så sjekker vi lydsignalet med en spektrumanalysator. Ut i fra målinger gjort i figur 4, så ser vi at frekvensene er $f_0 = 325$Hz og $f_1 = 750$Hz. Bruker dette i koden som kan leses i vedlegg A.

3.2 Test

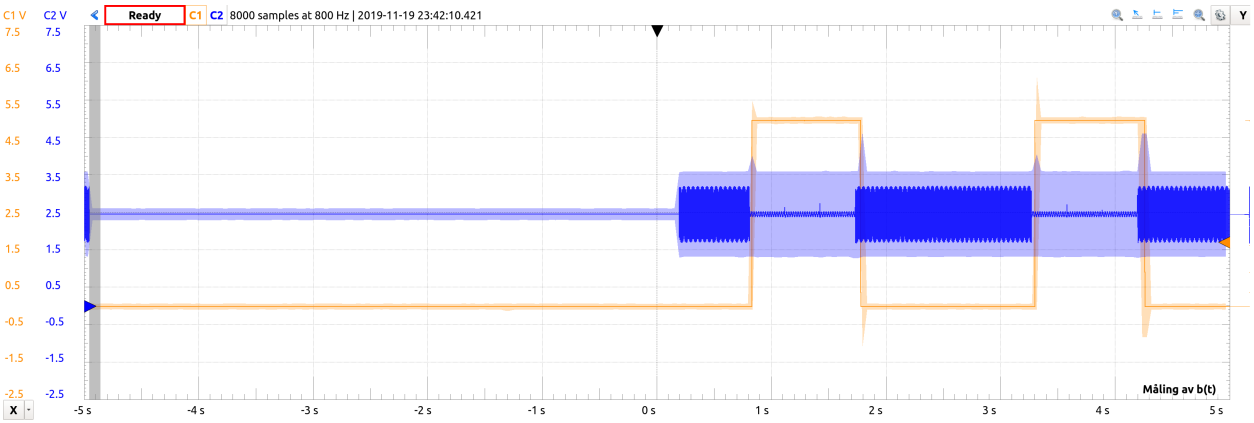
Etter å ha skrevet inn frekvensene i koden, så klarer Arduinoen å demodulere signalet i signalet som skal testes. Se figur 5, figur 6 og figur 7.



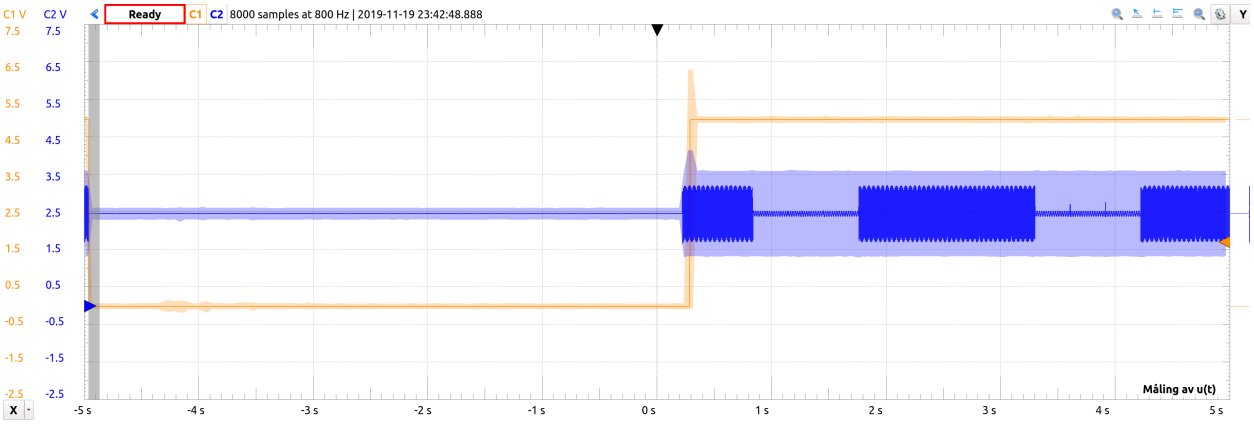
Figur 3: Den ferdige kretsen med ferdig oppkoblede pinner på Arduinoen.



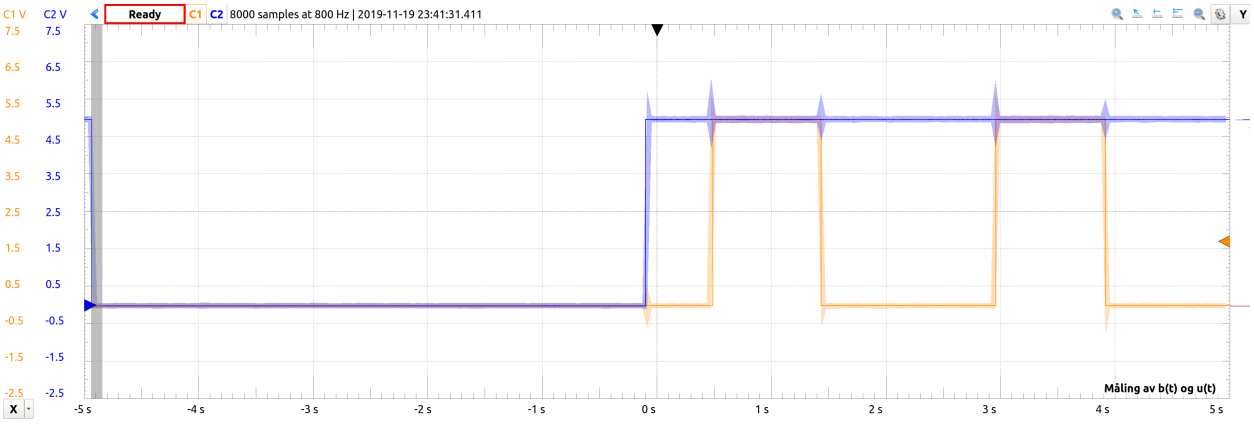
Figur 4: Måling av frekvenser i lydsignal. Den røde linjen er f_0 og den blå linjen for f_1.



Figur 5: Demodulering av FSK signalet, den gule linjen er det detmodulerte signalet $b(t)$, det blå er inngangssignalet $r(t)$. De partinene med liten amplitude er 750Hz og de med stor er 325Hz.

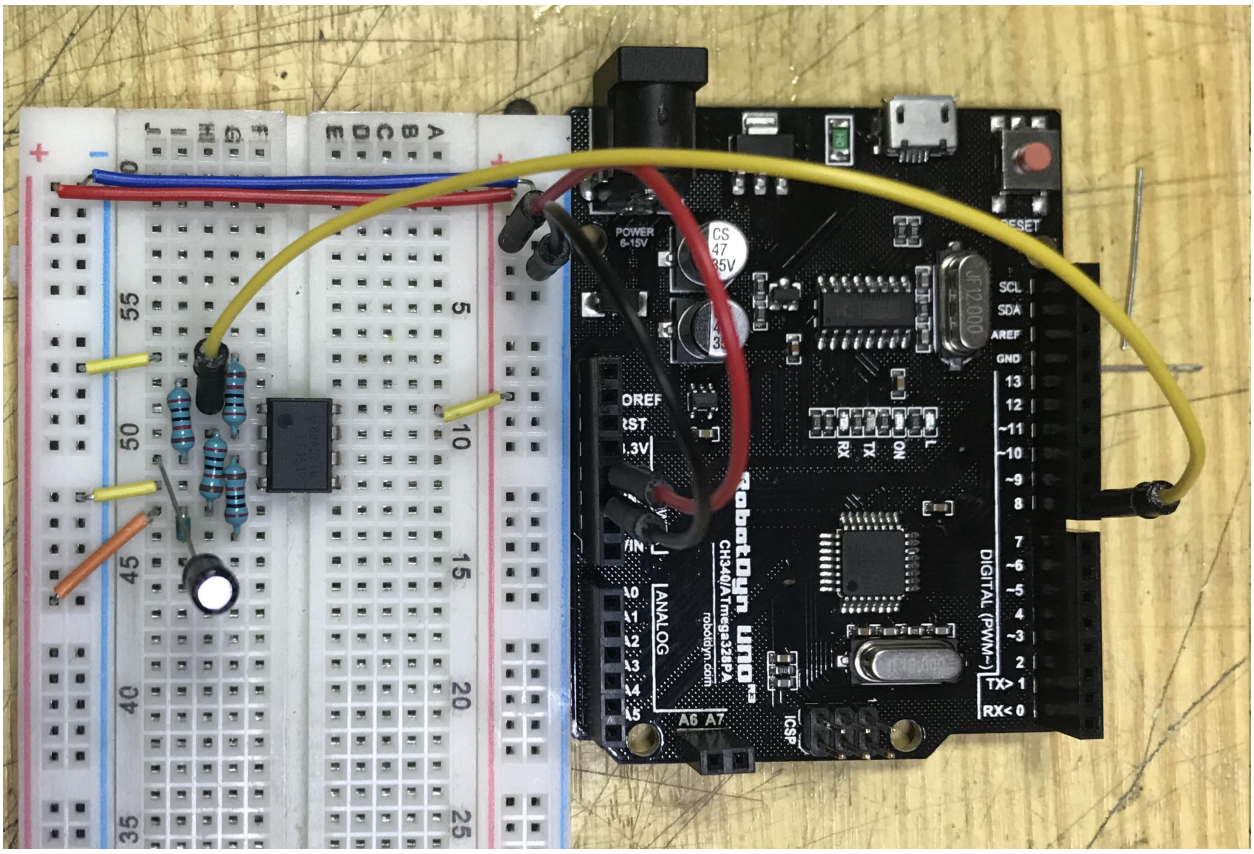


Figur 6: Demodulering av FSK signalet, den gule linjen er signalet $u(t)$ som sier om det er et demodulert signal, og det blå er inngangssignalet $r(t)$. De partinene med liten amplitude er 750Hz og de med stor er 325Hz.



Figur 7: Demodulering av FSK signalet, den gule linjen er det detmodulerte signalet $b(t)$, det blå er $u(t)$.

Den realiserte kretsen ser ut som i figur 8. Det totale arealet overstiger ikke 4cm^2. Det ser kanskje ikke sånn ut, men breadboard tar veldig mye plass.



Figur 8: Realisert krets, signalet $r(t)$ går inn ved kondensatoren, $u(t)$ kommer ut på D6, $b(t)$ kommer ut på D7.

4 Konklusjon

Kretsen gjorde det den skal gjøre, ved å måle frekvensen ved hjelp av et bibliotek til Arduino. Siden det kun er to variabler som styrer hvilke frekvenser som skal brukes i demoduleringen, er det også en veldig enkel demodulator å bruke. Den kunne vært gjort mindre ved å ikke bruke et breadboard, men klarer å fremdeles ha et totalareal på under 4cm^2.

Referanser

[1] Wikipedia contributors. (2019, November 10). *Frequency-shift keying*. In Wikipedia, The Free Encyclopedia. Retrieved 18:13, November 19, 2019, from https://en.wikipedia.org/w/index.php?title=Frequency-shift_keying&oldid=925429929

[2] ATMEL. (2009). *ATmega328P, Rev. 8025I–AVR–02/09*. https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf

[3] Texas Instruments. (2009). *LF353 Wide-Bandwidth JFET-Input Dual Operational Amplifier*. SLOS012C –MARCH 1987–REVISED MARCH 2016. http://www.ti.com/lit/ds/symlink/lf353.pdf

[4] PJRC, (Hentet 19. november 2019). *FreqMeasure Library*. https://www.pjrc.com/teensy/td_libs_FreqMeasure.html

[5] L. Lundheim. (05.11.2019). *Teknisk Notat: Digital kommunikasjon med FSK, v.3*. NTNU, Elsys-2017-LL-1.2.

A Kode til arduino

```
/*  Program to measure the frequenncy of a input, on digital pin 8
 *  Made by Oyvind Skaaden
 */
#include <FreqMeasure.h>  // Library for measuring the frequency
#define r_in 8 // Pinnen som brukes til å lese frekvensen

// Pinner som skal skrives til
#define u_out 6
#define b_out 7

// Frekvensene som blir brukt til demodulering
const unsigned int f0 = 325;
const unsigned int f1 = 750;

// Diverse kalkulasjoner for å kompensere for at frekvensen kan leses
// eller være feil på
const float offs = 0.05;
const float lowB = 1 - offs;
const float highB = 1 + offs;

// Hvor mange målinger som skal snittes
const unsigned int maxCount = 10;

// #### Globale verdier ####
double sum = 0;
unsigned int count = 0;

// Variabel for en timeoutfunksjon
unsigned long mesTime = 0;

void setup() {
  // Starter en seriel kommunikasjonsport for å kunne bruke det
  // demodulerte signalet gjennom feks en data
  Serial.begin(9600);
  // Start opp måling av frekvenser på pinne 8
  FreqMeasure.begin();

  // Setter pinmode til utgangspinnene
  pinMode(u_out, OUTPUT);
  pinMode(b_out, OUTPUT);
}

void loop() {
```

```
  // Dersom biblioteket kan lese frekvenser...
  if (FreqMeasure.available()) {
    // Time-out funksjonen
    mesTime = millis();

    // Summer sammen målinger for å snitte dem
    sum = sum + FreqMeasure.read();
    count = count + 1;
    if (count > maxCount) {
      // Snitt målingene for å gi en bedre måling
      float freq = FreqMeasure.countToFrequency(sum / count);

      // Her er grensene for hva som er lav og høy bit på FSK signalet.
      if (freq > f0 * lowB && freq < f0 * highB) {
        digitalWrite(b_out, LOW);
        digitalWrite(u_out, HIGH);
        Serial.print("LOW  : ");
      }
      else if (freq > f1 * lowB && freq < f1 * highB) {
        digitalWrite(b_out, HIGH);
        digitalWrite(u_out, HIGH);
        Serial.print("HIGH : ");
      }
      else {
        digitalWrite(b_out, LOW);
        digitalWrite(u_out, LOW);
        Serial.print("OFF  : ");
      }
      Serial.println(freq);
      sum = 0;
      count = 0;
    }
  }
  else{
    // Time-out funksjonen, dersom du ikke får noe signal på 10 ms,
    // sett alle utganger til LOW
    if (millis() - mesTime > 10){
      digitalWrite(b_out, LOW);
      digitalWrite(u_out, LOW);
      Serial.println("OFF");
      delay(10);
    }
  }
}
```