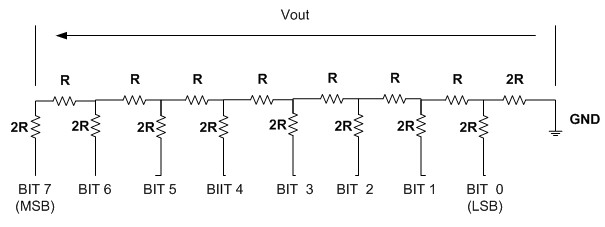
Wave-generator m/ timers og interrupt

Denne oppgaven går ut på å lage en generator som lager bølgeform. I denne rapporten har vi presentert vår forslag til løsning.

Kravspesifikasjonen for denne oppgaven er

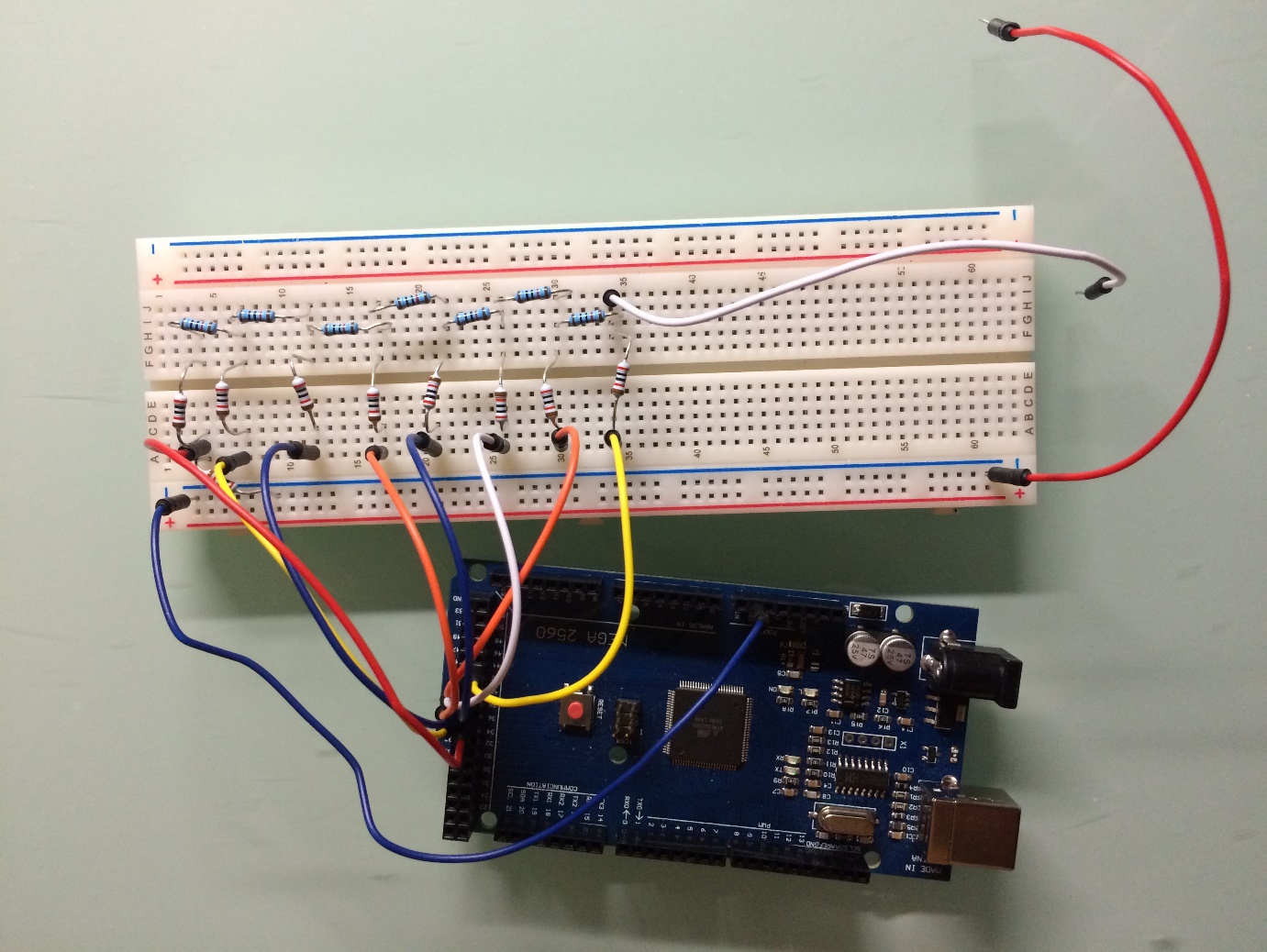
* To bølgeform
* En knapp som endrer mellom de to frekvensene.
* En knapp som endrer mellom to frekvenser.
* I oppgaven skal vi anvende timer og interrupt på en eller annen form.

Det finnes mange muligheter for å løse denne oppgaven. Vi har valgt å lage en R2R eller resistor ladder.Vi har løst oppgaven ved å lage en R-2R/resistor ladder digital to analog konverter. Den består av 8 pins, som igjen utgjør 8 bits. Hver pin kan enten være high eller low, 0 eller 5. Og ved hjelp av hvilken pin som er av og hvilken som er på, blir output strømmen påvirket. Siden strømmen går igjennom motstandene blir strømmen mindre. Output strømmen kan være alt mellom 0-5v. Hvor mye strømmen av et pin påvirker den totale strømmen avhenger av hvor i rekken pin befinner seg. 8 bit gir oss 256, det kan vi bruke til å regne ut verdier for hvilken som helst waveform.



***Navnet R2R kommer fra motstandene fra pinene er dobbelt så stor som motstandene mellom motstandene.***

På bildet under kan man se hvordan min R2R ser ut. Jeg har valgt å bruke 10k og 20k motstand. De hvite er 20k og blåe er 10k. Den hvite ledningen er output og rødt er ground. Disse kan kobles til for eksempel et oscilloskop.



10k motstand

20k motstand

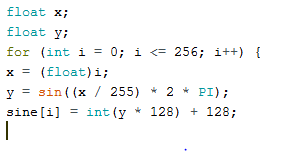
PORT C (pin fra 30-37)

Pin

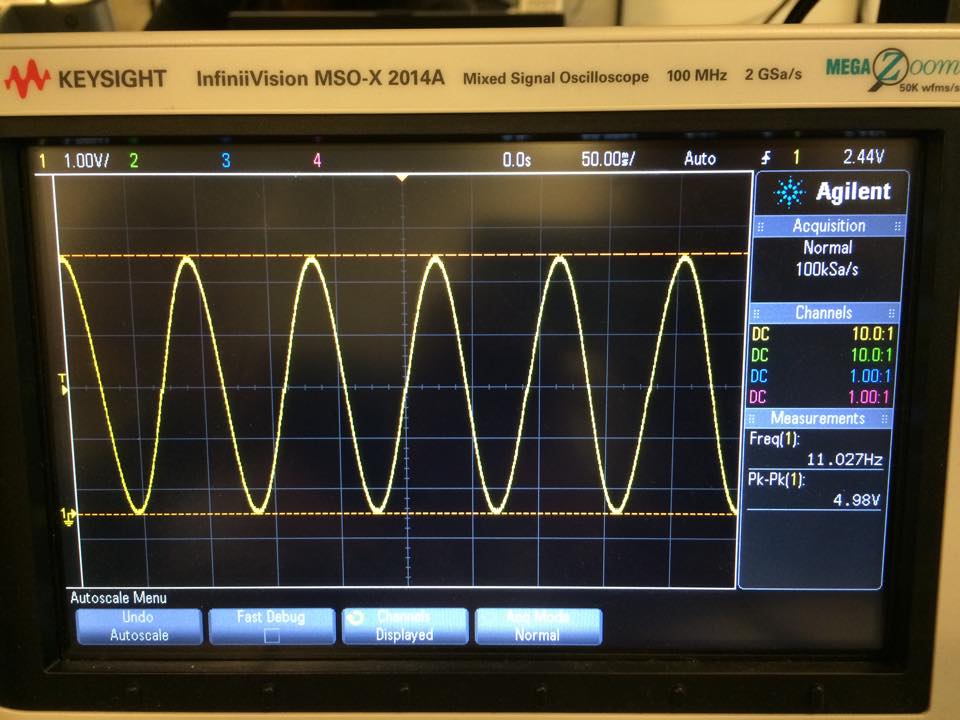
**Bølgeform:** Oppgavene krever at man skal kunne skifte mellom to bølgeform. Jeg har valgt å skifte mellom sinus og trekanter.

**Sinus:**

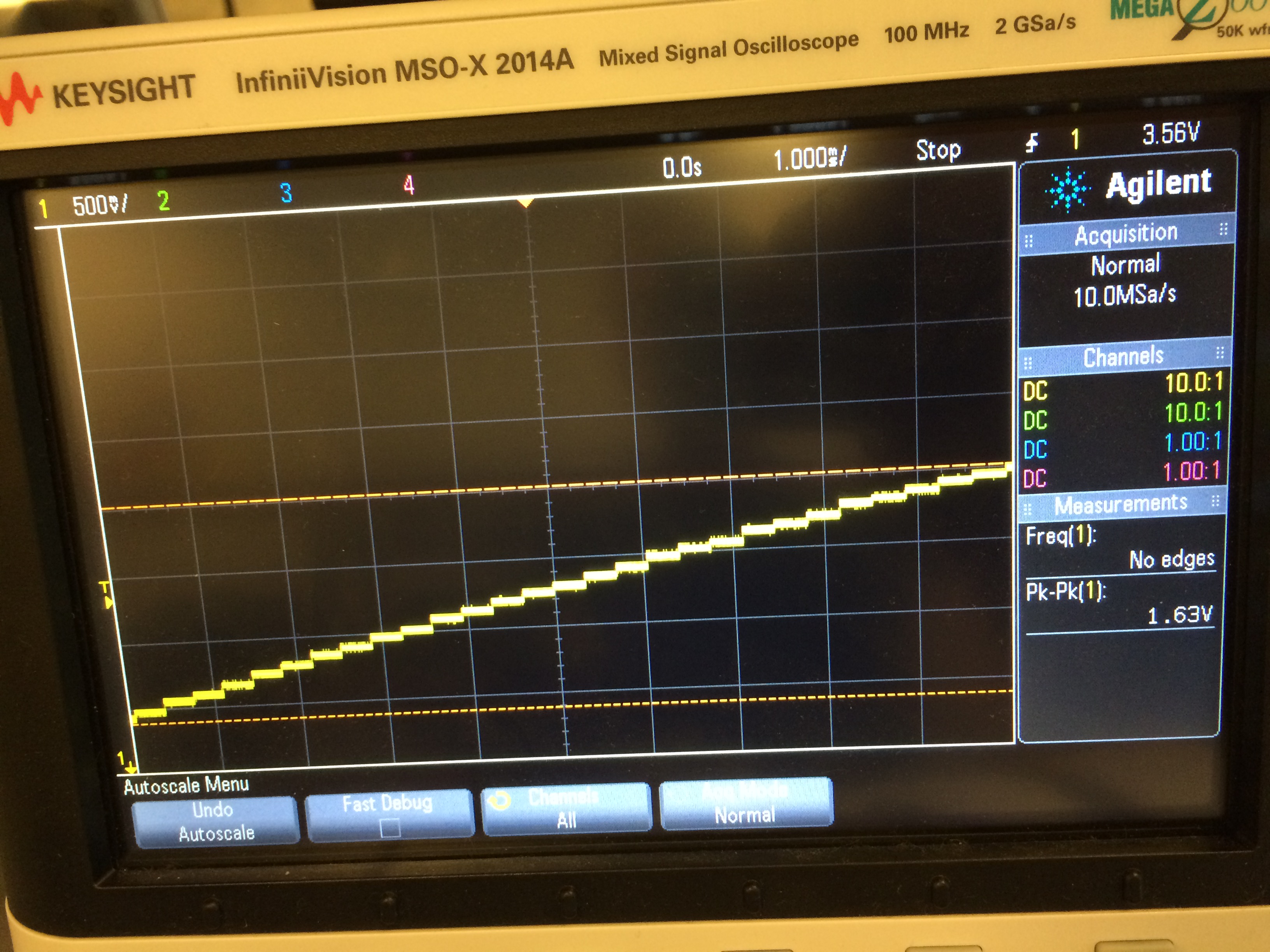
Fremgangsmåten av å regne ut sinus er som følgende. En full sinus periode er på 2pi. Vi har 256 verdier, så det vil si det er vår 2pi. Halv periode vil være 128.

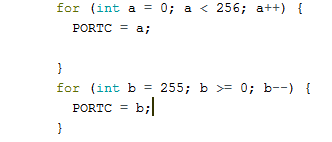
Jeg har brukt funksjonen sine, som ut i fra definisjonen på arduino.cc gjør følgende. «Beregner sinus av en vinkel (i radianer). Resultatet vil være mellom -1 og 1.» Resultatet av dette blir følgende sinusbølge.

***Her er kode som regner ut sinus-verdiene.***

Resultatet av kode og sinus-verdiene, er denne sinusbølge. Med tanke på at dette er et digitalsignal som har blitt konvertert, vil jeg si den er nydelig og absolutt godkjent.

Men selvsagt dersom vi forstørrer bilde vil vi kunne se at dette er et digitalsignal. Dette kan vi se på bildet under sinus bølgen. Det er samme bølge, bare forstørret.



**Trekanter:**

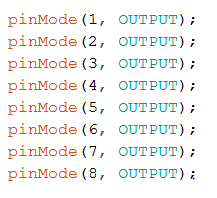
For å kunne fremvise trekanter på et oscilloskop kan vi bare telle fra 0 til 256, og ned igjen fra 256 til 0.

Dette har jeg oppnådd ved hjelp av to for-loops. Den ene teller opp, og den andre teller ned. I tillegg setter den PORTC, som jeg kommer tilbake til senere, lik verdien fra for-loopen.

**Port manipulasjon:**

Dette gjør det mulig å manipulere flere I/O fra en kodelinje. For eksempel i mitt tilfelle skulle jeg bruke 8 pins, og istedenfor gå i setup å sette alle de 8 pins til output, kan man bare gjøre det i starten av programmet. Det er vanskelig å forklare, men ved å se bildene under, kan vi godt se forskjellen.

## Til venstre har vi den vanlig metoden av å sette en pin til output/input, mens til høyre har vi port registers. Det er tydelig at port registers gjør det lettere og koden blir kort og simpelt. Men begge gjør det samme, så man kan fritt velge hvilken metode man vil bruke. Jeg har brukt den til høyre.

****

**Interrupt:**

Jeg har brukt interrupt i forbindelse en knapp som endrer bølgeform. Interrupt virker ved at når man feks trykker på en knapp, som så sender en signal til pinen, så brytes programmet ut av kodens flow. Programmet har blitt ”interrupted”.

Hvis normal program flow er det blå, så er interrupt funksjonen det lilla. Så hver gang man sender et signal til en bestemt pin, vil programmet stoppe og kjøre den lilla koden.

Kode:

#include "TimerOne.h"

const byte buttonPinA = 2; // the number of the pushbutton pin, changes wave

const byte buttonPinB = 3; // the number of the pushbutton pin, changes frequency.

const byte ledPin = 13; // for testing.

volatile bool state;// Variable used for to wave form.

volatile byte c = 0;//variable used for

volatile unsigned int frequency = 100000;//frequency, 0.1 secounds.

long debouncing\_time = 15; //Debouncing Time in milliseconds

volatile unsigned long last\_micros;

byte sine[256] = {128, 131, 134, 137, 140, 143, 146, 149,//array with all the sine values.

152, 156, 159, 162, 165, 168, 171, 174,

176, 179, 182, 185, 188, 191, 193, 196,

199, 201, 204, 206, 209, 211, 213, 216,

218, 220, 222, 224, 226, 228, 230, 232,

234, 235, 237, 239, 240, 242, 243, 244,

246, 247, 248, 249, 250, 251, 251, 252,

253, 253, 254, 254, 254, 255, 255, 255,

255, 255, 255, 255, 254, 254, 253, 253,

252, 252, 251, 250, 249, 248, 247, 246,

245, 244, 242, 241, 239, 238, 236, 235,

233, 231, 229, 227, 225, 223, 221, 219,

217, 215, 212, 210, 207, 205, 202, 200,

197, 195, 192, 189, 186, 184, 181, 178,

175, 172, 169, 166, 163, 160, 157, 154,

151, 148, 145, 142, 138, 135, 132, 129,

126, 123, 120, 117, 113, 110, 107, 104,

101, 98, 95, 92, 89, 86, 83, 80,

77, 74, 71, 69, 66, 63, 60, 58,

55, 53, 50, 48, 45, 43, 40, 38,

36, 34, 32, 30, 28, 26, 24, 22,

20, 19, 17, 16, 14, 13, 11, 10,

9, 8, 7, 6, 5, 4, 3, 3,

2, 2, 1, 1, 0, 0, 0, 0,

0, 0, 0, 1, 1, 1, 2, 2,

3, 4, 4, 5, 6, 7, 8, 9,

11, 12, 13, 15, 16, 18, 20, 21,

23, 25, 27, 29, 31, 33, 35, 37,

39, 42, 44, 46, 49, 51, 54, 56,

59, 62, 64, 67, 70, 73, 76, 79,

81, 84, 87, 90, 93, 96, 99, 103,

106, 109, 112, 115, 118, 121, 124, 128

};

void setup()

{

pinMode(buttonPinA, INPUT);// initialize the pushbutton pin as an input:

pinMode(buttonPinB, INPUT);// initialize the pushbutton pin as an input:

pinMode (ledPin, OUTPUT);//testing..

Serial.begin(9600);//testing..

Timer1.initialize(frequency); // initialize timer1, and set a 0,1 second period

Timer1.attachInterrupt(waveGen); // attaches callback() as a timer overflow interrupt

attachInterrupt(0, waveChange, RISING); //Interrupt 5 is Digital Pin 2

attachInterrupt(1, freqChange, CHANGE); //Interrupt 5 is Digital Pin 3

// for (int i = 30; i < 38; i++) {//testing..

// pinMode(i, OUTPUT);

// }

}

void loop() {

}

void waveChange() {//function used for deboucing.

if ((long)(micros() - last\_micros) >= debouncing\_time \* 1000) {//wait for aleat "debouncing\_time" of time.

//if its has been that amount of time, then we know for sure the button is pushed, so we change the state.

Interrupt();

last\_micros = micros();

}

}

void Interrupt() {//this function changes the state, which again chagnes the wave form.

state = !state;//change state to the oppisite.

}

void freqChange() {

frequency = frequency - 10000;

Serial.println("frequency");

Serial.println(frequency);

if (frequency < 10000) {

frequency = 100000;

}

}

void waveGen() {

switch (state) { //this switch case runs eiher sinus or triangels code.

case 0:

Serial.println("Sinus!");

PORTC = sine[c];

Serial.println(sine[c]);

if (c == 255) {

c = 0;

}

c++;

break;

case 1:

Serial.println("triangles!");

for (int a = 0; a < 255; a++) {

PORTC = a;

}

for (int b = 255; b >= 0; b--) {

PORTC = b;

}

break;

}

}

Dette er nå en ganske simple design, så det er ikke mye som kan gå galt. Men man må selvsagt være oppmerksom på en del ting. For selv om koden er enkel, så må man være sikker på at hardwaren er riktig tilsluttet. Og alt fungerer. For eksempel så hadde vi problemer med hvilke pins vi brukte, kunne ikke får den til å virke. Dette medførte at vi brukte unødvendig lang tid til å få den til å fungere.

Dette er jo en digital signal som har blitt konvertert til analog. Så man vil alltid se spor av at det er et digital signal. Men den virker og gjør jobben ganske godt. Man må virkelig se godt etter eller zoome riktig godt inn for å kunne se at dette er en digital signal. Men likevel så fungerer den utmerket, og man må virkelig se nøye på om man skal se at det en digital signal.

Jeg er ganske fornøyd med hvordan vi har løst oppgaven på og resultat vi har fått. Dessverre ble ikke vi helt ferdig. Det meste av programmet og hardwaren virker, men vi mangler å endre frekvensen. Hovedproblem i dette designet er kanskje frekvens, siden vi ikke har fått det til å virke enda. Jeg har prøvd å fikse problemet, men har ikke hatt tilgang til oscilloskop for å kunne teste koden.

Kilder:

<http://www.ermicro.com/blog/wp-content/uploads/2008/11/basic_r8.jpg>

https://www.arduino.cc/en/Reference/Sin