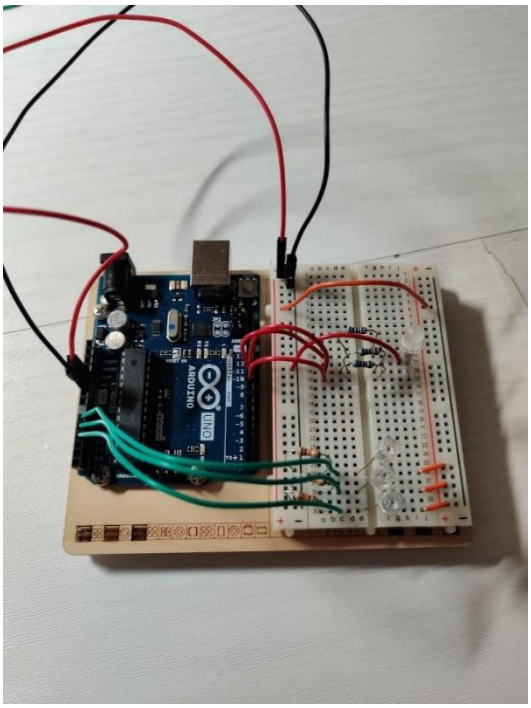
1)

Kretsen



Strekker først en ledning over slik at det er strøm på begge sider av brettet. Setter deretter opp tre foto-dioder ved siden av hverandre på brettet. Disse får strøm fra høyre side av brettet. Jord er koblet på $10K\Omega$ motstander. De er også koblet til hver sin analoge inngang på arduino-kortet. Jeg fant ikke noe utstyr i kitet for å feste den fargede filmen som følger med.

Over fotodiodene er LED- lampen som skal vise de forskjellige fargene koblet opp. Her er de tre inngangene som representerer de ulike fargene koblet på de digitale-pinene på arduino-brettet. Disse er koblet på med 220Ω mostander.

Kode

```
//definerer konstanter som sier hvi
const int greenLEDPin= 9;
const int redLEDPin= 11;
const int blueLEDPin= 10;

const int redSensorPin= A0;
const int greenSensorPin= A1;
const int blueSensorPin= A2;


//definerer variabler for sensor ve
int redValue=0;
int greenValue=0;
int blueValue=0;

int redSensorValue=0;
int greenSensorValue=0;
int blueSensorValue=0;



  void setup() {

     Serial.begin(9600);

     pinMode(greenLEDPin, OUTPUT);
     pinMode(redLEDPin, OUTPUT);
     pinMode(blueLEDPin, OUTPUT);

     }
```

```
void loop() {

  //kjører funksjonen
  readAnalog();

  //skriver ut verdiene fra sensorene
  Serial.print(" Raw Sensor Values \t Red: ");
  Serial.print(redSensorValue);
  Serial.print("\t Green: ");
  Serial.print(greenSensorValue);
  Serial.print("\t Blue: ");
  Serial.print(blueSensorValue);

  // setter outputen til å være sensor verdien/4
  redValue= redSensorValue/4;
  greenValue= greenSensorValue/4;
  blueValue= blueSensorValue/4;

  Serial.print(" Mapped Sensor Values \t Red: ");
  Serial.print(redValue);
  Serial.print("\t Green: ");
  Serial.print(greenValue);
  Serial.print("\t Blue: ");
  Serial.print(blueValue);


  writeAnalog();

}
```

```
void readAnalog(){

 redSensorValue= analogRead(redSensorPin);
 delay(5);
 greenSensorValue= analogRead(greenSensorPin);
 delay(5);
 blueSensorValue= analogRead(blueSensorPin);

}


//egen funksjon som sender verdiene som skal er
void writeAnalog(){

  analogWrite(redLEDPin,redValue);
  analogWrite(greenLEDPin, greenValue);
  analogWrite(blueLEDPin, blueValue);

}
```

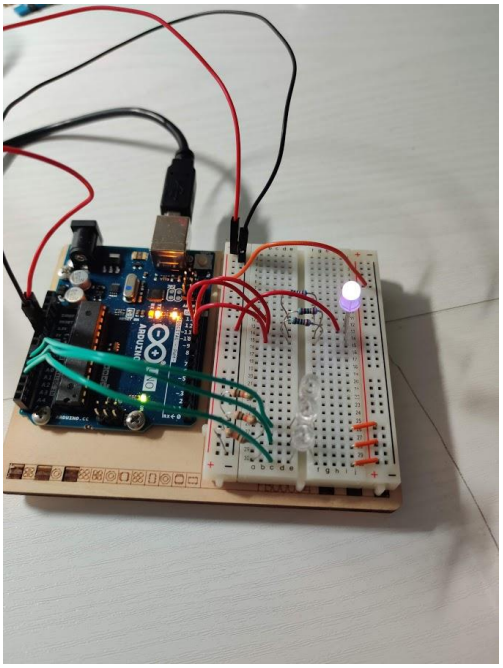
Definerer først en rekke konstanter og variabler som holder på hvilke pins som blir brukt til hva, og hva verdien på disse er/skal være. I setup-funksjonen defineres det hvilke pins utgangene er på. Videre i loop-funksjonen så kjøres en engen funksjon som leser verdiene fra sensorene (de analoge inngangene). Deretter skrives disse ut til serial-monitoren. Deretter settes verdiene på utgangene til å være inngangsverdiene delt på fire. Det fordi tallet skal passe med analogWrite sin parameter. De oppdaterte utgangene skrives til monitoren, og en egen funksjon som legger disse verdiene på utgangene som LED-lampen bruker.

Testing

Åpner opp serialmonitor og ser at raw-verdiene ligger rundt 50-60 på alle sensorene, mapped ligger da på 10-15.

```
Mapped Sensor Values   Red: 13        Green: 14       Blue: 13

Raw Sensor Values      Red: 56        Green: 56       Blue: 59
```

Lyser ser slik ut



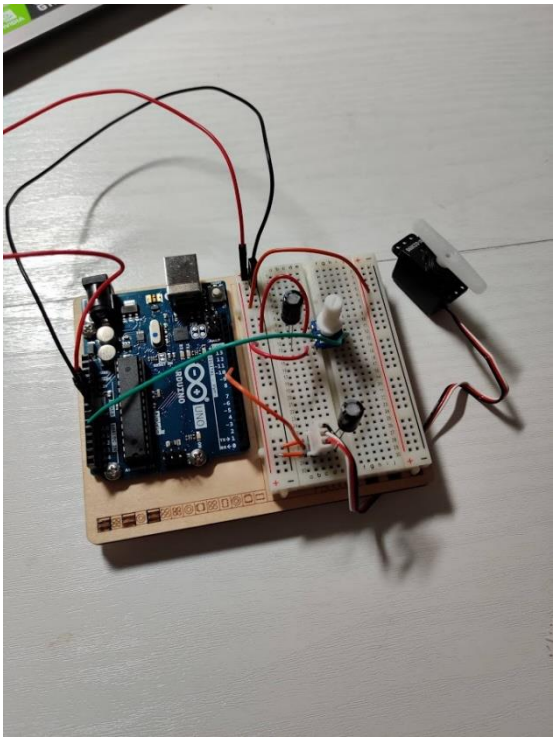
Skrur av lyset og alle verdiene på sensoren går ned til null

```
Raw Sensor Values       Red: 0  Green: 0        Blue: 0 M
```

Det er noe problematisk å få tatt bilde av/filmet de ulike fargene på LED-lampen, men i et mørkt rom brukte jeg mobilen som lommelykt og vinklet den slik at den lyste mest mulig på en av fotodiodene om gangen, og så klart og tydelig at den skiftet mellom de tre fargene respektivt.

2)

Kretsen



Plasserer et potensiometer som er en elektrisk komponent som fungerer som en spenningsdeler. Når man vrir på denne endres spenningsforholdet mellom den miderste pinen og +(strøm). Denne endringen kan leses som en analog input, og kobles på A0 på brettet.

Kobler deretter opp servoen ved hjelp av hann-plugger. Denne tar strøm, jord og en input. Denne inputen kommer fra en digital pin på brettet.

Servomotoren drar mer strøm når den begynner å bevege seg enn når den står stille, det oppstår derfor et spenningsfall over brettet. Dette kan unngås ved å bruke en kondensator ved tilkoblingen til servoen. Det samme gjøres ved potensiometeret, dette er for å «glatte» ut mulige endringer i spenning på brettet.

Kode

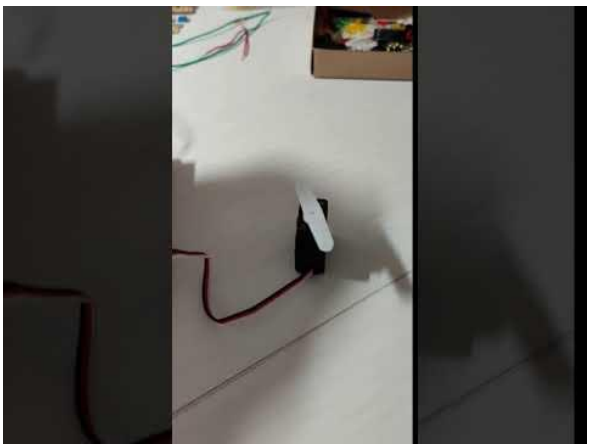
```
#include <Servo.h>
Servo myServo;
int const potPin= A0;
long potVal;
int angle;

  void setup() {

   myServo.attach(9);
   Serial.begin(9600);

  }

  void loop() {

    potVal=analogRead(potPin);
    Serial.print("potVal: ");
    Serial.print(potVal);
    angle= scale(potVal, 0,1023, 0, 179);
    Serial.print(" , angle: ");
    Serial.println(angle);

    myServo.write(angle);
    delay(15);

  }


long scale(long potVal, long scaleLow, long scaleHigh,
              long angleLow, long angleHigh){

  int value= (potVal-scaleLow) *(angleHigh- angleLow) /
                (scaleHigh-scaleLow) + angleLow;

  return value;

}
```

Begynner med å inkludere servo-bibloteket til arduino. Dette tilbyr funksjonalitet som gjør det greit å interraktere med servo-motorer. Lager deretter en instanse av servo-bibloteket gjennom et objekt. Dette gjør at vi kan nå all funskonalitet gjennom dette objektet.

Definerer variabler som holder på inputverdien, og vinkelen man vil sensoren skal flyttes til. I setup-defineres hvilken pin servoen er koblet til. Videre lagres verdien på den analoge inputten i variabelen som er definert. Deretter kjøres «scale» funksjonen, som returnerer en verdi (grader servoen skal flytte seg) basert på verdien på potensiometeret. Dette tilsvarer logikken i den innebygde map-funksjonen. Jeg måtte for øvrig endre return-verdien og parameterene til å være av typen long, og ikke int som var definert i oppgaveteksten. Tislutt skrives den gjeldende vinkelen til servoen, som flytter seg deretter.

Testing

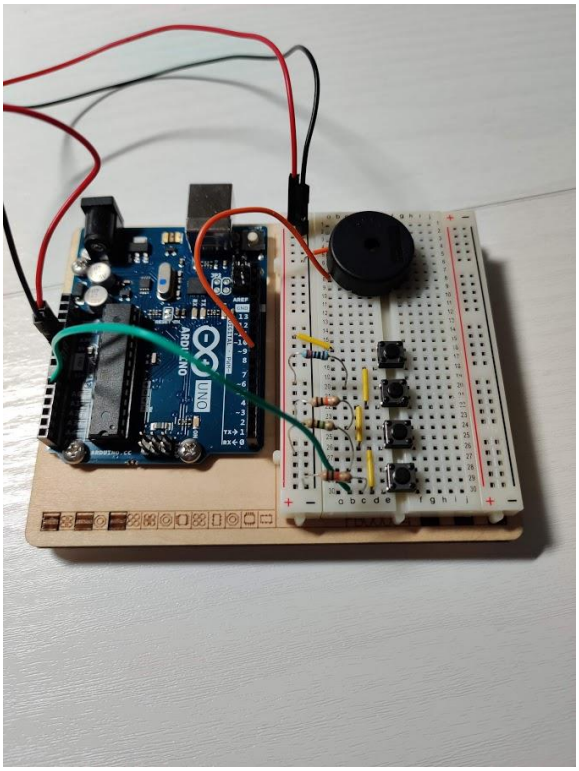
Video av servoen som snur seg 180 grader:



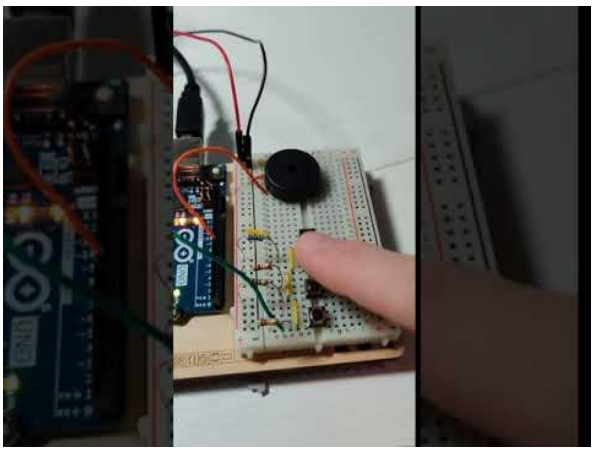
Jeg testet først programmet slik det står beskrevet i prosjekt-boken (med bruk av map-funksjonen), for å finne ut hvordan det var ment til å fungere. Deretter implementerte jeg en egen map-funksjon, og denne fungerte ikke på første forsøk. Etter litt søking fant jeg ut at variabel-typene burde være long, og endret dette. Da kjørte programmet og servoen flyttet seg slik den skulle som vist på videoen ovenfor.

3)

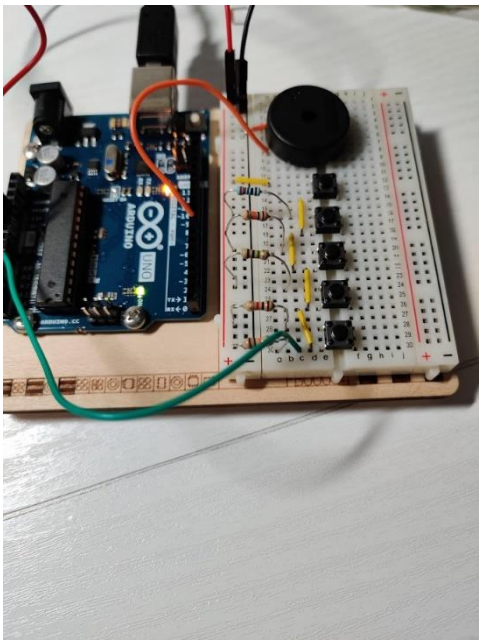
For første del av oppgaven koblet jeg opp kretsen slik. (Denne blir nærmere beskrevet i andre del av oppgaven)



Det måtte litt kalibrering til, men fikk etter hvert frem klare lyder som vist på videoen nedenfor. Koden blir vist/nærmere beskrevet i andre del.



Kretsen



Kretsen ser slik ut. Piezo (høytaleren) er koblet til ground og en digital inngang på brettet. I denne kretsen er den ekstra bryteren koblet opp, slik at det til sammen en 5. Den første bryteren er koblet til strøm, og videre er de koblet i parallell hvor den siste utgangen er jordet med en motstand på $10K\Omega$.

Jeg valgte å koble den femte (ekstra bryteren) på nederst på brettet slik at det kunne bli en stigende frekvens på lyden. (Denne skulle outputte 440 Hz som da blir den høyeste frekvensen. Denne ble koblet til strøm gjennom en mostand på $1K\Omega$. De andre (fra toppen og ned) var koblet på respektive $220\Omega, 10K\Omega \; og \; 1M\Omega$ mostander. Den siste utgangen på bryter 5 er også koblet til det analog inputtet A0 på brettet.

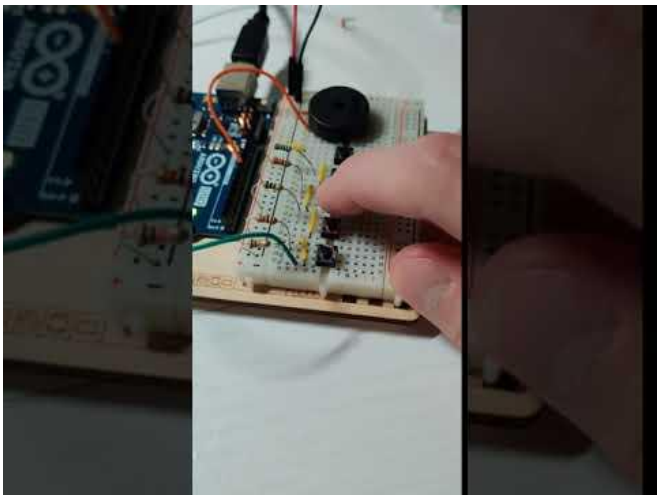
Kode

```
int notes[]= {200, 294, 330, 349, 440};

void setup() {

   Serial.begin(9600);

    }

    void loop() {

       delay(400);

        int keyVal= analogRead(A0);
        Serial.println(keyVal);

        if(keyVal>990){
          tone(8,notes[0]);
        }

        else if(keyVal >=970 && keyVal <=990){
          tone(8,notes[1]);
        }

        else if(keyVal >=900 &&keyVal <=960){
          tone(8, notes[4]);
        }

        else if(keyVal >=505 && keyVal <=515){
          tone(8,notes[2]);
        }

        else if(keyVal >=5 && keyVal <=50){
          tone(8,notes[3]);
        }

        else{
          noTone(8);
        }
```

Programmet består av en array som skal holde på de forskjellige lydene (frekvenser) som skal høres på høytaleren. Jeg legger her til verdien for den femte knappen til i slutten av arrayen. Videre lages den en kobling til serial-monitoren i setup-funksjonen. Har lagt til en liten delay i starten av loop, slik at det skal være lettere å lese av verdiene i monitoren.

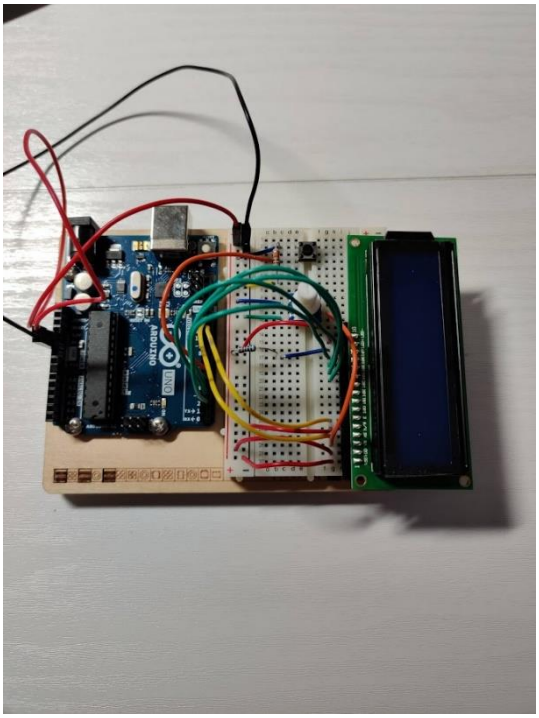
Videre leses spenningen på inngangen A0 og lagrer som en digital verdi (0-1023) i en variabel. Det sjekkes deretter for ulike tilfeller (områder) denne verdien kan være og det legges en lyd på utgangen deretter. Jeg brukte stort sett verdiene fra boka, men måtte kalibrere litt på de lavere frekvensene, da disse var litt skurrete. Når det gjelder kalibrering av den femte knappen, koblet jeg den først opp. Deretter sjekket jeg verdiene som kom fra den i monitoren, og lagde en passende betingelse deretter.

Reultater



4)

Kretsen



Koblet en vanlig knapp istedenfor tilt-sensoren, da denne var noe lettere å forholde seg til. Valgte å bruke et potentiometer, og koble dette til inngangen på displayet for å kunne styre kontrasten på displayet. (se hvordan dette fungerte)

Videre er de ulike inngangene på displayet koblet til brettet slik som beskrevet i boka

Første del av oppgaven

Skrev koden slik som beskrevet i prosjekt-boken og fikk et forventer resultat. En video av dette kan ses nedenfor

Del 2 av oppgaven- funksjon/logikk som skriver ut et symbol gjennom alle kolonner og rader

Kode

```
#include <LiquidCrystal.h>
LiquidCrystal lcd (12,11,5,4,3,2);

const int switchPin = 6;
int switchState = 0;

  void setup() {

      Serial.begin(9600);
      lcd.begin(16,2);
      pinMode(switchPin, INPUT);

    }

        void loop() {

        switchState= digitalRead(switchPin);
        Serial.println(switchState);

        if(switchState==HIGH){
           printSymbol();
         }
    }
```

```
void printSymbol(){

      //knappen må trykkes på
      //ytterste loop styrer rad, mens innserste
        for(int j=0; j<2; j++){
            for(int i=0; i<=15; i++){
               if(digitalRead(switchPin)==LOW){
                 break;
               }
               lcd.setCursor(i,j);
               lcd.print("#");
               delay(250);
               lcd.clear();
            }

        }
}
```

Beskrivelse av koden

Starter med å importere biblioteket som tar seg av liquid-crystal displayet. Initialiserer deretter dette og forteller hvilke pins- som skal bli brukt til å komunisere. I setup-funksjonen startes en conncetion til monitoren, dette for å kunne se når inputen er høy og lav. LCD.begin- forteller at skjermen er 16 kolonner ganger 2 rader stor. Deretter settes det at pin nummer 6 er en input.

I loop- funksjonen sies det først at switchstate variabelen skal være en digitalRead av inputen. Denne blir da HIGH eller LOW (0/1) basert på spenningen på inputten. Hvis denne er høy, altså at knappen blir trykket på skal print – symbol funksjonen kjøres.

Print-symbol funksjonen består av nestede for-løkker. Den ytterste løkken kjører to ganger, en for hver av de to radenene. Den innerste løkken kjører 16 ganger, en for hver kolonne i hver rad. Her sjekkes det om inputten blir LOW, hvis det er tilfelle skal løkken stoppe. (Slik at knappen må holdes inne for at funksjonen skal bli kjørt). Cellen symbolet skal tegnes i blir dermed satt til (i,j) hvor «i» styrer hvilken kolonne som skal fylles, og «j» styrer hvilken rad som behandles. Symbolet «#» blir skrevet til cellen. Legger til en liten delay på 250ms slik at det ikke skal gå fort fort. Deretter kjøres lcd.clear() slik at cellen tømmes før den går videre til neste.

Resultater

Et forventet resultat kom frem. Hadde ved første test forvekslet rader og kolonner i lcd.setCursor() funksjonen slik at symbolet bare ble skrevet ut i første celle av hver rad. Byttet om på dette og programmet fungerte som forventet. Nedenfor er en video av implementasjonen går igjen og igjen, og en video hvor løsningen med knapp er implementert.