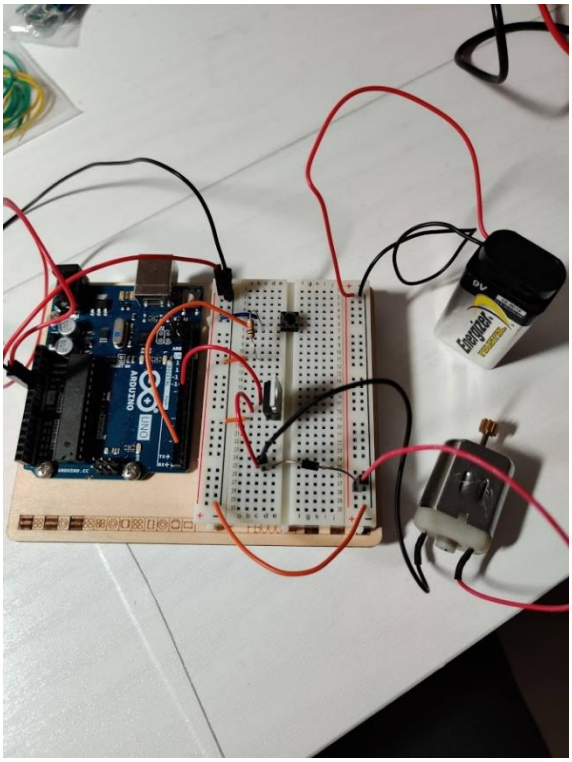
1)

Project 09

Krets



Kretsen består av en bryter som er koblet til jord gjennom en motstand på $10K\Omega$. Denne er også koblet til inngang 2 på brettet, og fungerer som inngangssignal.

Videre er en transistor koblet på. (Ytterligere forklart nedenfor). Denne bruker utgangs-signalet fra brettet som spenning på **gaten.** Hvis denne spenningen endres vil de to andre pin-ene bli sammenkoblet. Motorens jord er koblet til denne pinen, og det vil da gå strøm hvis det detekteres en endring på **gaten.**

9V- batteriet gir en spenning slik at motoren kan fungere, da arduinoens utganger ikke er tilstrekkelig. For å unngå at motoren gir strøm tilbake til kretsen, brukes en diode, som bare leder strøm en vei.

Kode

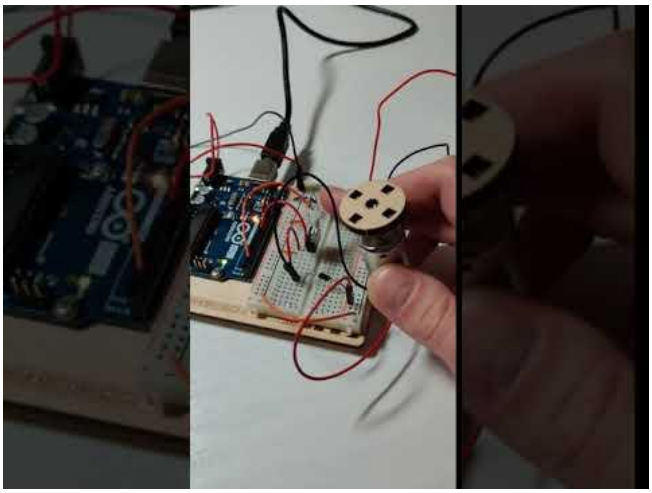
```
const int switchPin=2;
const int motorPin= 9;
int switchState=0;


  void setup() {

      pinMode(motorPin, OUTPUT);
      pinMode(switchPin,INPUT);

    }

  void loop() {

    switchState= digitalRead(switchPin);

     if(switchState==HIGH){
         digitalWrite(motorPin, HIGH);
     }

      else{
        digitalWrite (motorPin, LOW);
      }

}
```

Inngang og utgang-pins lagres som konstanter. I setup- funksjonen sies det at knappen skal fungere som en inngang, og motoren sin tilkobling skal fungere som en utgang.

I loop-kjøres en digitalRead av switchPin, noe som blir HIGH eller LOW avhengig av spenningen på inngangen. Deretter sjekkes det hvorvidt denne er av eller på, og motoren blir kjørt/ikke kjørt avhengig av dette.

Resultater

Koden kompilerte og kjørte som forventet. Kretsen fungerte også som forventet. En video av resultatet kan ses nedenfor.



Transistoren

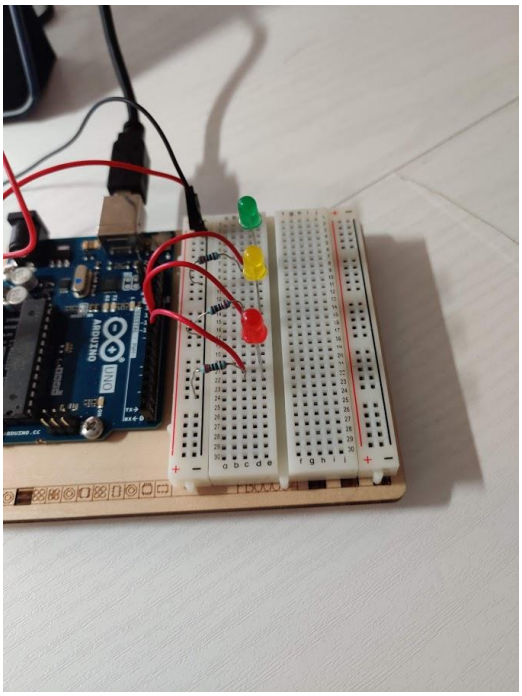
Transistorer dannes på grunnlag av dioder. Disse er elektroniske komponenter som bare kan lede strøm en vei. En diode består av to typer halvledere som igjen blir dannet gjennom doping av stoffer med spesielle egenskaper. (Antall valenselektroner). Elektrisk ledning, og feltet som oppstår i sammenslåingen gjør at dioden leder strøm en vei, men ikke andre veien.

En transistor består essensielt av to slike dioder, og er en elektrisk komponent som enten kan forsterke et elektrisk signal, eller fungere som en bryter. Når man gir en spenning på en av inngangene til en transistor, lukkes kretsen og det blir mulig å få høyere spenning innad i kretsen.

Transistorer er utrolig viktig i all moderne elektronikk. De kan som nevnt brukes som brytere, altså at de enten kan være av eller på, noe som er veldig nyttig med tanke på maskinkode. De er også viktige for å forsterke signaler i ulike kretser som brukes i elektroniske komponenter slik som datamaskiner og mobiltelefoner.

2 a)

Krets

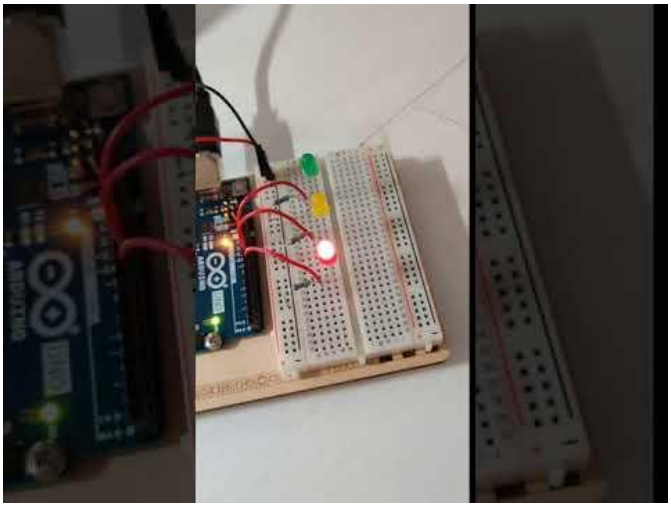


Kode

```
const int redPin=9;
const int yellowPin=10;
const int greenPin= 11;

int activePin= redPin;


  void setup() {

    pinMode(redPin, OUTPUT);
    pinMode(yellowPin, OUTPUT);
    pinMode(greenPin, OUTPUT);

  }

  void loop() {

    for(int i=0; i<=255; i++){
      analogWrite(activePin, i);
      delay(15);
    }

    for(int i=255; i>=0; i--){
      analogWrite(activePin, i);
      delay(15);
    }

  }
```

Resultater

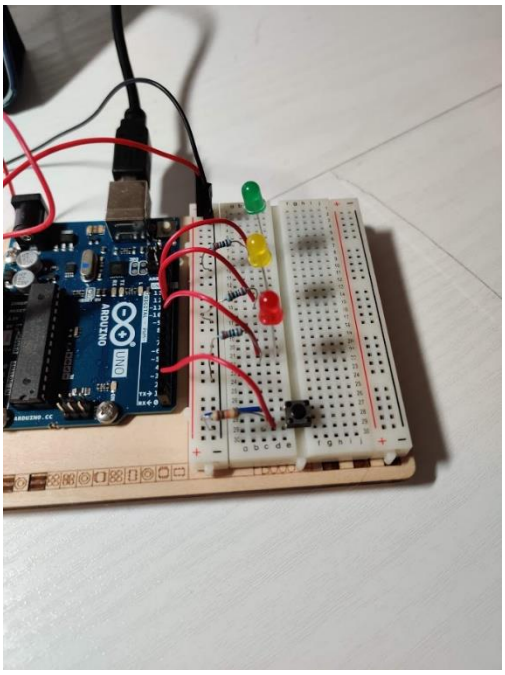
Video av resultatet kan ses her



Programmet fungerer som forventet. Har en egen variabel som holder på pin-verdien til LED-en man velger som aktiv. Loop-funksjonen og for-løkkene kjører dermed bare koden for den LED-en som er satt til å være aktiv. Ovenfor er det rød -led som er aktiv, men programmet fungerer på samme måte for en av de andre LED-ene.

b)

Krets



Kretsen ser lik ut som i oppgave a, men en bryter er koblet på i pull-down metode. Slik at den gir 5V når den blir trykket ned.

Kode

```
const int redPin=9;
const int yellowPin=10;
const int greenPin= 11;
const int interruptPin= 2;

int activePin= 0;


  void setup() {

    pinMode(redPin, OUTPUT);
    pinMode(yellowPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(interruptPin, INPUT);

    activePin= redPin;

    attachInterrupt(digitalPinToInterrupt(interruptPin), shiftActive,RISING);

  }

  void loop() {

    for(int i=0; i<=255; i++){
      analogWrite(activePin, i);
      delay(15);
    }

    for(int i=255; i>=0; i--){
      analogWrite(activePin, i);
      delay(15);
    }

  }
```

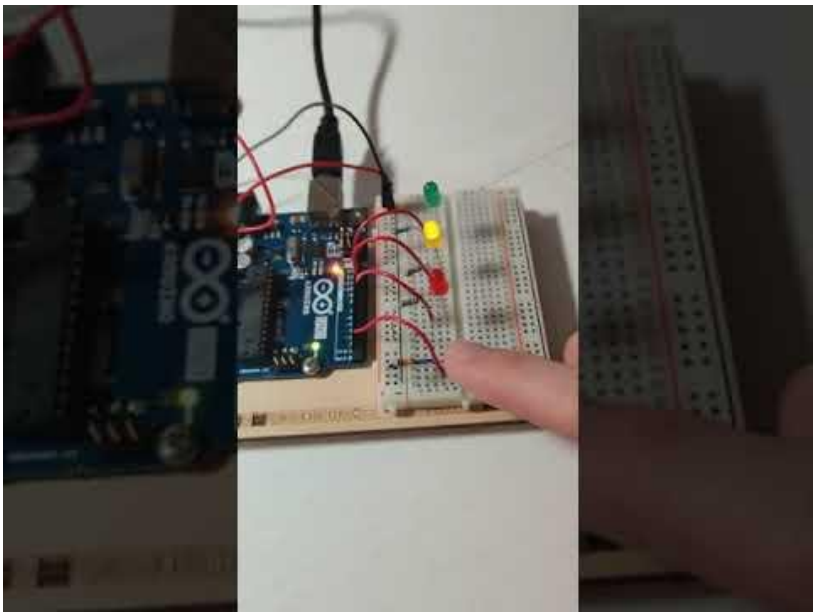
```
void resetPins(){
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin,LOW);
  }


void shiftActive(){

   static unsigned long lastInterruptedTime = 0;
   unsigned long interrupt_time = millis();

    if (interrupt_time - lastInterruptedTime > 200){

     if(activePin==redPin){
       activePin= greenPin;
     }
     else if(activePin==greenPin){
        activePin=yellowPin;
     }

     else if(activePin==yellowPin){
          activePin =redPin;
     }
  }

  resetPins();
  lastInterruptedTime  = interrupt_time;
  }
```

Koden bygger på kode i oppgave 2a. Vi har noen konstanter som holder på pin-ene til de ulike LED-lampene. Vi legger også til pin 2, hvor en interrupt kan bli signalisert. (Arduino har noen utvalgte pins hvor dette er mulig). activePin blir satt i setup-funksjonen, og en interrupt-handler blir lagt til ved interrupt på pin 2. Loop-funksjonen fungerer på samme måte som i 2a, hvor for-løkker benyttes for å skrive en analog-verdi til LED-ene som vil resultere i dimming opp og ned. Shift-active-funksjonen er funksjonen som blir kjørt ved et signalisert interrupt. Denne sjekker først om tiden som måles ved start (millis()) – tiden ved slutten av funksjon er større enn 200. Hvis den er mindre enn 200 kan man anta at interrupt-kallet skyldes bounce i knappen. Videre sjekkes det for verdi av activePin, og den aktive LED-en endres etter beskrivelser i oppgaven. Deretter oppdateres siste interrupt tid, slik at man på nytt kan sjekke om den tiden ved start- tiden ved slutt er større enn 200.

En funksjon resetPins() kalles også. Dette er for å sette alle til lav, altså ingen lys ved en signalisert interrupt. Har jeg ikke dette med vil alle lampene lyse hele tiden, etter hvert som de blir aktive gjennom interrupt-logikken.

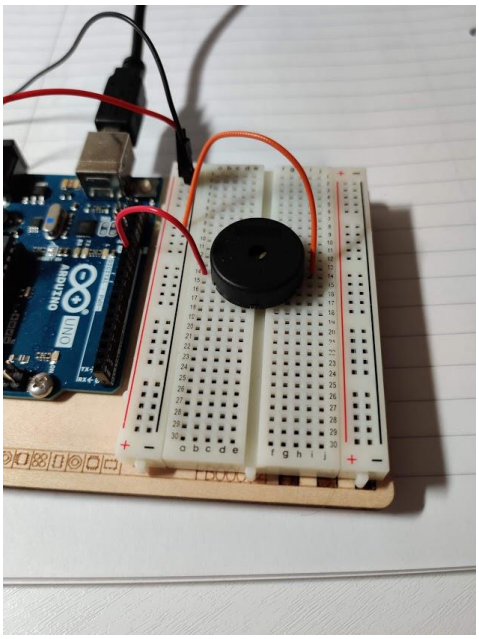
Resultater

Programmet fungerte som forventet, når en interrupt ble signalisert ble aktiv led- byttet og programmet fortsatte fra samme sted på den ny gjeldende LED-en.



3)

Krets



Kretsen består ganske enkelt av en høyttaler (piezo) som er koblet på den digitale inngangen (12) på brettet.

Kode

```
#include <TimerOne.h>
const int speakerPin= 12;

 const int C=260;
 const int D= 292;
 const int E= 328;
 const int F=348;
 const int G =392;
 const int A= 440;
 const int H=492;
 const int c=520;

 int onTone;
 int onName;

 //definerer en array med toner//
 int tones[]= {C,D,E,F,G,A,H,c};
 char names[]= {'C', 'D', 'E', 'F', 'G', 'A', 'H', 'c'};

   void setup() {
     Timer1.initialize(400000);
     Timer1.attachInterrupt(changeTone);
     int onTone=0;
     Serial.begin(9600);

   }
```

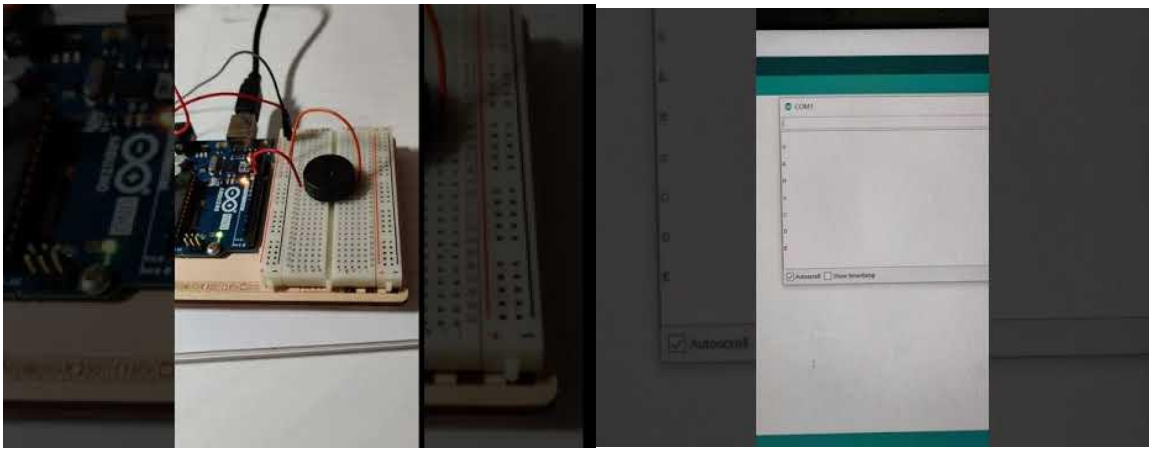
```
void loop() {
 onName=onTone-1;
 Serial.println(names[onName]);
 Serial.println();
 onName++;
 delay(400);

}

 void changeTone(){

 int elements= (sizeof(tones)/ sizeof(tones[0]))-1;

 if(onTone>elements){
  onTone=0;
 }

  tone(speakerPin, tones[onTone],390);
  onTone++;

}
```

Koden ser slik ut. Lastet først ned filene (TimerOne-biblioteket) og inkluderte dette i prosjektet. Lagde deretter en rekke konstanter, og instansierer onTone, og onName. Disse skal etter hvert holdet på hvilken tone som skal spilles/leses. Tar meg friheten til å endre det første tone-navnet til C (stor-c) og det siste til c (liten-c). Dette for at det skal være enklere å holde på navnene i en ren «char» array. Definerer deretter to arrayer. Den ene holder på frekvensene (int) og den andre holder

på navnene. (som skal skrives ut). I setup instansieres en timer. Denne skal kjøre funksjonen changeTone, og forekomme hvert 0.4 sekund. (40 0000 microsekunder).

I loop- skrives gjeldende tone ut. Jeg implementerte denne løsningen med en egen teller «onName» fordi det viste seg at onTone alltid lå en int-verdi bak i loop-funksjonen i forhold til changeTone, som jo gir mening da onTone inkrementeres der. I timer-funksjonen beregner jeg først hvor mange elementer det er i arrayen. (sizeof returnerer antall bit i hele arrayen), trekker fra 1, fordi jeg vil ha tak i indexene 0-7. Sjekker deretter om onTone er større enn antall elementer-1. Hvis den er det, er slutten av arrayen med toner nådd, og jeg vil starte på nytt. onTone settes derfor til 0 igjen. Til slutt kalles tone, som sender lyd til høyttaleren og onTone inkrementeres.

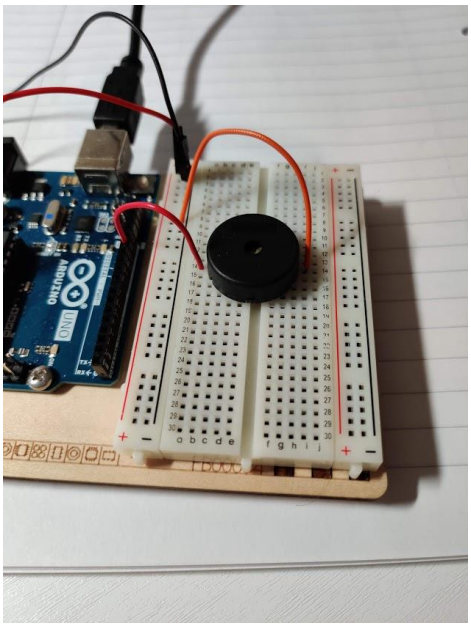
Resultater



Videoen til venstre vises de forskjellige tonene som spilles av etter en timer-rutine. Videoen til høyre viser monitoren som skriver ut gjeldende tone. (Tonen som spilles).

4)

Krets



Benytter samme krets som i oppgave 3.

Kode

```
const int C=260;
const int D= 292;
const int E= 328;
const int F=348;
const int G =392;
const int A= 440;
const int H=492;
const int c=520;

int speakerPin=12;
int currentDuration=0;
int sizeOfArray=0;

int melodi[22] = {C,D,E,F,G,G,A,A,A,A,G,F,F,F,F,E,E,D,D,D,D,C};
int duration[22] = {1,1,1,1,2,2,1,1,1,1,4,1,1,1,1,2,2,1,1,1,1,4};

void setup() {

   Serial.begin(9600);

}

void loop() {

  sizeOfArray= (sizeof(melodi)/ sizeof(melodi[0]));

  for(int i=0; i<sizeOfArray; i++){
    currentDuration= duration[i]*500;
    Serial.println(melodi[i]);
    tone(speakerPin, melodi[i], currentDuration);
    delay(currentDuration+10);

  }

}
```

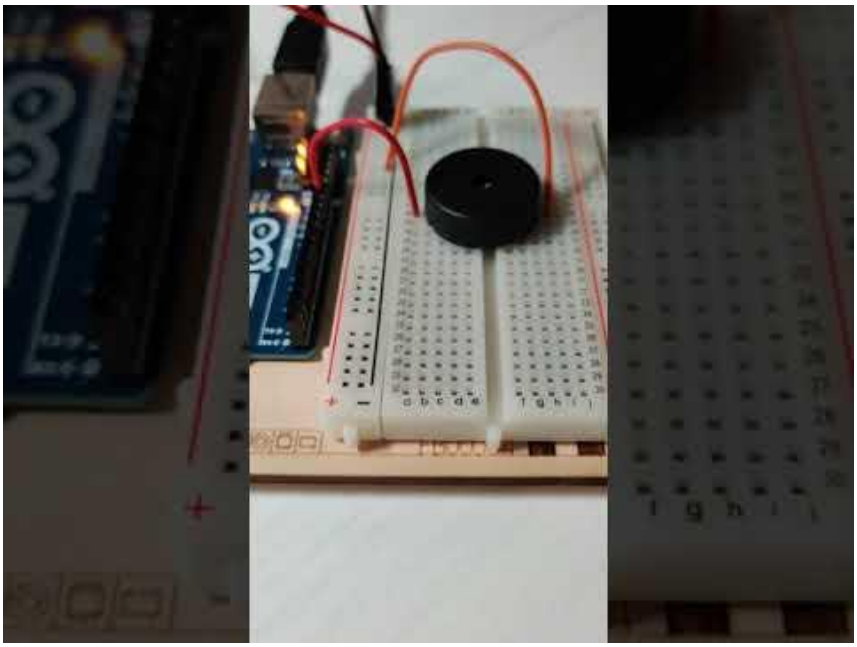
Koden tar utgangspunkt i samme toner som i oppgave 3. Jeg definerer først to arrayer med heltall. En representerer de ulike tonene (melodien) og varigheten til hver av de. I setup initialiserer jeg bare en kobling til monitoren som jeg brukte til feilsøking.

I loop regner jeg først ut hvor mange elementer det er i arrayen, deretter kjøres en for-løkke som teller gjennom «antall» elementer (toner) i arrayen. Jeg lagrer varigheten i en variabel, bruker samme teller fordi arrayene vil være like store. Denne verdien ganger jeg med 500 med bakgrunn i forslag fra oppgaven. Videre sendes den gjeldende melodien til høyttaleren, med riktig varighet. Legger inn et delay slik at melodien er cirka 10ms kortere enn pausen imellom hver.

Resultater

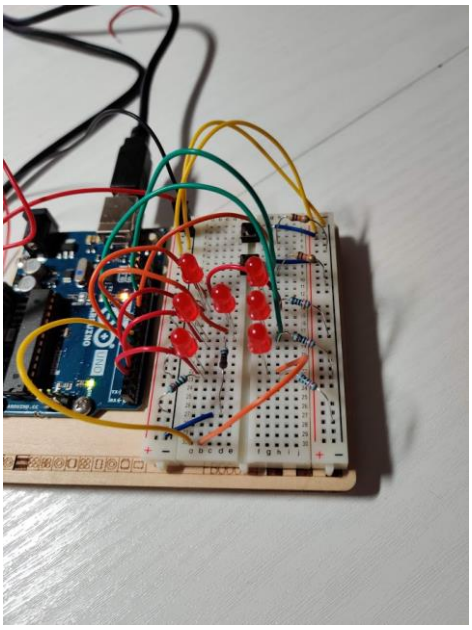
Programmet fungerte som forventet. *Melodien som spilles er Lisa gikk til skolen*

Video:



5)

Krets



Kretsen består av 7 lysdioder som er koblet gjennom 220Ω motstander, og til inngangene 2-8, slik som beskrevet i oppgaven. Videre er to brytere koblet på i pull-down metode gjennom $10K\Omega$ mostander. De er koblet til brettet på inngang 10 og 11.

Kode

```
const int ledOnePin= 2;
const int ledTwoPin= 3;
const int ledThreePin= 4;
const int ledFourPin= 5;
const int ledFivePin= 6;
const int ledSixPin=7;
const int ledSevenPin= 8;

const int resetButtonPin= 10;
const int randomButtonPin=11;

int resetState=0;
int randomState=0;
int numberToPrint=0;

void setup() {
  pinMode(ledOnePin, OUTPUT);
  pinMode(ledTwoPin, OUTPUT);
  pinMode(ledThreePin, OUTPUT);
  pinMode(ledFourPin, OUTPUT);
  pinMode(ledFivePin, OUTPUT);
  pinMode(ledSixPin, OUTPUT);
  pinMode(ledSevenPin, OUTPUT);

  pinMode(resetButtonPin, INPUT);
  pinMode(randomButtonPin, INPUT);

  Serial.begin(9600);

}
```

```
void loop() {

  randomState= digitalRead(randomButtonPin);
  resetState= digitalRead(resetButtonPin);

  if(randomState==HIGH){
    resetLeds();
    printRandom();
  }

  if(resetState==HIGH){
      resetLeds();
  }


}

void printRandom(){

  numberToPrint= random(1,7);

  if(numberToPrint==1){
    digitalWrite(ledOnePin, HIGH);
  }
  else if(numberToPrint==2){
    digitalWrite(ledOnePin, HIGH);
    digitalWrite(ledFourPin, HIGH);
  }
  else if(numberToPrint==3){
    digitalWrite(ledOnePin, HIGH);
    digitalWrite(ledFourPin, HIGH);
    digitalWrite(ledSevenPin, HIGH);
    }
```

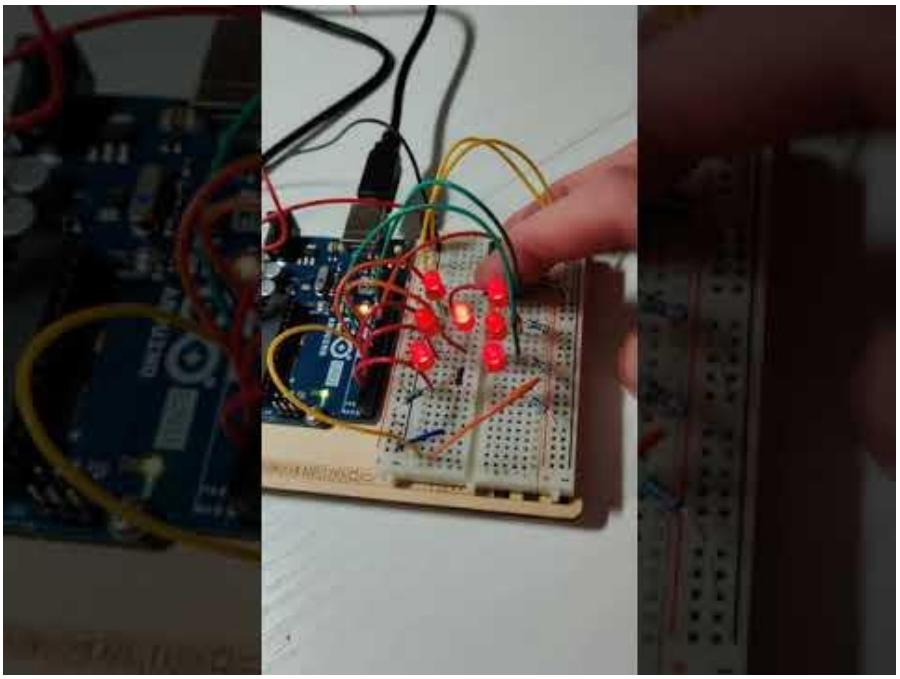
```
  else if(numberToPrint==4){
    digitalWrite(ledOnePin, HIGH);
    digitalWrite(ledFourPin, HIGH);
    digitalWrite(ledThreePin, HIGH);
    digitalWrite(ledSixPin, HIGH);

  }
  else if(numberToPrint==5){
    digitalWrite(ledOnePin, HIGH);
    digitalWrite(ledFourPin, HIGH);
    digitalWrite(ledThreePin, HIGH);
    digitalWrite(ledSixPin, HIGH);
    digitalWrite(ledSevenPin, HIGH);
  }

  else if(numberToPrint==6){
    digitalWrite(ledOnePin, HIGH);
    digitalWrite(ledTwoPin, HIGH);
    digitalWrite(ledThreePin, HIGH);
    digitalWrite(ledFourPin, HIGH);
    digitalWrite(ledFivePin, HIGH);
    digitalWrite(ledSixPin, HIGH);
  }

}


void resetLeds(){
  digitalWrite(ledOnePin, LOW);
  digitalWrite(ledTwoPin, LOW);
  digitalWrite(ledThreePin, LOW);
  digitalWrite(ledFourPin, LOW);
  digitalWrite(ledFivePin, LOW);
  digitalWrite(ledSixPin, LOW);
  digitalWrite(ledSevenPin, LOW);
}
```

Lager først konstanter som holder på pin-ene til de ulike lampene, samt det samme for bryterne. Instansierer noen variabler som skal holde på tilstanden til knappene, samt nummeret som terningen skal vise. Definerer inn/ut-ganger i setup-funksjonen. I loop hentes tilstanden til knappen, er reset-knappen 1(HIGH) skal reset funksjonen kjøres. Den setter alle Led-pins til LOW. Som gjør at Led-ene ikke lyser. Videre sjekkes det om random-knappen trykkes, hvis dette er tilfelle slukkes først foregående kombinasjon (ved å slukke alle) og deretter kjøres random-funksonen. Denne lagrer den tilfeldige verdien i en variabel. Deretter sjekkes det for ulike verdier av denne og samsvarende Led-er tennes.

Resultater

Programmet fungerte som forventet, tilfeldige terning-verdier blir vist ved trykk på den ene knappen, og alle slukkes når den andre knappen trykkes.

Video



6)

```
void setup()
{
  Serial.begin(9600);
}


void loop()
{
  int test = 1;
  int verdi = 3, total = 4;
  float resultat = 1.0;

    resultat = verdi/total;
    Serial.print("resultat  = ");
    Serial.println(resultat);

    resultat = (float) verdi/total;
    Serial.print("resultat  = ");
    Serial.println(resultat);

    test =  verdi%total;
    Serial.print("test = ");
    Serial.println(test);

    test  =  total >> 1;
    Serial.print("test = ");
    Serial.println(test);

    test  =  total & 1;
    Serial.print("test = ");
    Serial.println(test);

    test  =  total & 4;
    Serial.print("test = ");
    Serial.println(test);

    test  =  total | 0;
    Serial.print("test = ");
    Serial.println(test);

    test  =  total ^ 4;
    Serial.print("test = ");
    Serial.println(test);

  for(;;)
  { }
}
```

Det defineres først 4 variabler: **test, verdi,total og resultat.** I den første beregningen deles 3/4 (verdi/total). Fordi verdi og total er heltall vil monitoren skrive ut resultat= 0.0. I andre beregning gjøres det samme, men det sies først at svaret skal være en float verdi. Det blir da skrevet ut 0.75, som er det «riktige svaret».

Videre sies det at test skal være lik 3%4. 3%4 er lik 3, og dette blir lagret riktig i test, fordi den kan holde på heltall. Det blir skrevet ut: test = 3.

I neste beregning skal total (4) bitene som total består av, bli «flyttet» 1 bit mot høyre. $4 = 0100$

Disse flyttes 1 bit mot høyre, og vi får $0010 = 2$. Det skrives ut test = 2

I neste skal test være lik $total \ \& \ 1$. Det vil si 4 og 1. Denne operatoren sammenlikner hvert bit, og resultatet blir bare 1, så lenge begge bit-ene som blir sammenliknet er 1. $4 = 0100$ og $1 = 0000$. Dette resulterer i 0000 som er 0. Det blir skrevet ut test =0.

I neste beregning gjøres det samme som ovenfor, bare at her er tallene like. Det vil da bli skrevet ut det samme tallet som de som sammenliknes. Altså test = 4.

Videre sammenliknes total (4) | 0. Som er 4 eller 0. Dette blir også da 4, og test= 4 skrives ut.

I siste beregning utføres en XOR- logikk. Det er 4 XOR 4. Som kjent gir XOR-logikken ut 0 hvis input-bitene er like, og det skrives følgelig ut test = 0.

Til slutt kjøres en uendelig for-løkke slik at det overnevnte bare skal skrives ut en gang.

7)

```
void setup()
{
  Serial.begin(9600);
}


void loop()
{
  char bokstav = 'A';
  char *pkrBokstav = &bokstav;
  char fruktTyper[5][20] = {"Eple", "Banan", "Drue", "Mango", "Sitron"};
  char *pkrFrukt =  "Ananas";

  Serial.println(*pkrBokstav);
  *pkrBokstav = *pkrBokstav + 1;
  Serial.println(*pkrBokstav);

  Serial.println(fruktTyper[3]);
  Serial.println(strlen(fruktTyper[3]));

  strcpy(fruktTyper[4], pkrFrukt);
  Serial.println(fruktTyper[4]);


  strcat(fruktTyper[1]," og ");
  strcat(fruktTyper[1],fruktTyper[4]);
  Serial.println(fruktTyper[1]);

  fruktFunksjon ( fruktTyper );
  Serial.println(fruktTyper[2]);

  for (;;)
  {}
}


void fruktFunksjon ( char frukter[5][20] )
{
   strcpy(frukter[2],  "Plomme");
   Serial.println(frukter[2]);
}
```

I første utskrift skrives det ut en variabel som peker på bokstav-variabelen, og denne inneholder A. Deretter inkrementeres denne, og den vil nå representere en ACII-verdi og den skriver nå ut B.

Den 3-indexen i fruktyper—arrayen hentes og skrives ut. Denne er mango. Strlen() henter ut lengden på stringen, og det skrives da ut 5.

strcpy() kopierer den andre stringen som sendes med over den første stringen. Det står da Ananas på den fjerde indexen i arrayen istedenfor Sitron. Det skrives dermed ut Ananas.

strcat() legger på stringen som sendes med som andre argrumentet til den første stringen blant argumentene. Det legges derfor på «og» til Banan og deretter Ananas til «Banan og». Resultatet av dette blir da «Banan og Ananas» som skrives ut.

Fruktfunksjon tar en array og erstatter index 2 med ordet «Plomme». Dette skrives ut både i funksjonen og deretter i loop-funksjonen. Det vil derfor bli skrevet ut Plomme to ganger som siste del av utskriften.

8)

Kode

```
  int sensorOneValue;
  int sensorTwoValue;
  int sensorThreeValue;

  float temperatureOne;
  float temperatureTwo;
  float temperatureThree;

  float averageTemp;
  float voltagePerDegree;


void setup() {
  voltagePerDegree= 5.0/300.0;
  Serial.begin(9600);
}

void loop() {

  sensorOneValue=analogRead(simulateTemp(50));
  sensorTwoValue= analogRead(simulateTemp(50));
  sensorThreeValue= analogRead(simulateTemp(50));

  temperatureOne= checkTemp(sensorOneValue);
  temperatureTwo= checkTemp(sensorTwoValue);
  temperatureThree= checkTemp(sensorThreeValue);

  compareTemp(temperatureOne, temperatureTwo, temperatureThree);

  for(;;){}
  delay(2000);
}


float simulateTemp(float temp){
  //finner økningen i antall grader fra 0
  float degreesFromZero= temp + 100;
  //finner antall volt per grad//

  return degreesFromZero * voltagePerDegree;
}

float checkTemp(int sensorValue){
    float voltagePerBit= 5.0/1023;
    float currentVoltage= voltagePerBit* sensorValue;
    float degreesFromZero= currentVoltage/voltagePerDegree;

    float finalTemp;

    if(degreesFromZero>0){
       finalTemp=degreesFromZero-100;
    }
    if(degreesFromZero<0){
      finalTemp= 0 -(degreesFromZero-100);
    }

    return finalTemp;
}

void compareTemp(float temperatureOne, float temperatureTwo,float temperatureThree) {
   averageTemp= (temperatureOne+ temperatureTwo+ temperatureThree)/3;

  if((temperatureOne-temperatureTwo)> 0.6 || (temperatureOne-temperatureThree)>0.6 || (temperatureTwo- temperatureThree)>0.6){
    Serial.println("En eller flere av sensorene er utenfor spesifikasjon. Må undersøkes");
  }
  else{
    Serial.println("Sensorene er OK");
```

Velger etter avklaring med Robert å simulere alle de tre sensorene, og kobler derfor ikke opp noen krets.

Programmet initialiserer noen float-verdier som skal holde på verdier fra sensorene, og temperaturene. Sender først noen verdier inn i en funksjon som returnerer en float, denne simulerer en spenning. Denne leses i loop, og sendes deretter tilbake til en andre funksjon som sjekker denne verdien og gir tilbake en faktisk temperatur, det basert på forholdet mellom spenning og temperatur som er gitt i oppgaven. Har også tatt utgangspunkt i at sensoren leser 0V ved -100 grader.

Deretter lagres disse tre temperaturene som kjøres inn i en compare -funksjon. Denne sjekker for de gitte spesifikasjonene, og skriver ut til monitoren deretter. Regner også ut gjennomsnittet.

Resultater

Kjører samme verdi inn i funksjonen som simulerer en spenning, og får da forventet resultat:

```
Sensorene er OK
Gjennomsnittstemperatur:42.33
```

Endrer en eller flere av verdiene inn til funksjonen som simulerer en spenning, og får dette:

```
En eller flere av sensorene er utenfor spesifikasjon. Må undersøkes
```

Hadde en liten Bug, i henhold til at temperaturen ut av checkTemp -funksjonen ikke stemte 100% overens med temperaturen jeg prøvde å simulere, det hendte det var et merkbart avvik. Dette er trolig pågrunn av de mange beregningene, og at disse ikke blir helt nøyaktig i forhold til desimaler osv.

Eller fungerte programmet som forventet.