Projekt 11
Buzzer

Skolans namn: Thorildsplan Gymnasium
Kursens namn: EE19D

Innehåll

Inledning

Vi ska koppla in en analog piezosummer (buzzer) som kan spela olika toner och styrs därför INTE med på- och avslag av spänning. Istället används funktionen `tone()` som skickar ut en önskad ton (frekvens) under önskad tid.
Ja, `tone()` kan enbart generera fyrkantsvågor och saknar möjlighet att justera ljudstyrkan (volym). Det går därför inte att få någon hifi-upplevelse, men det är inga problem att spela enkla melodier.
För att spela melodier behövs två saker:

- en lista över vilka toner som motsvarar vilka frekvenser
- en melodi (en lista med toner).

Syfte med projektet

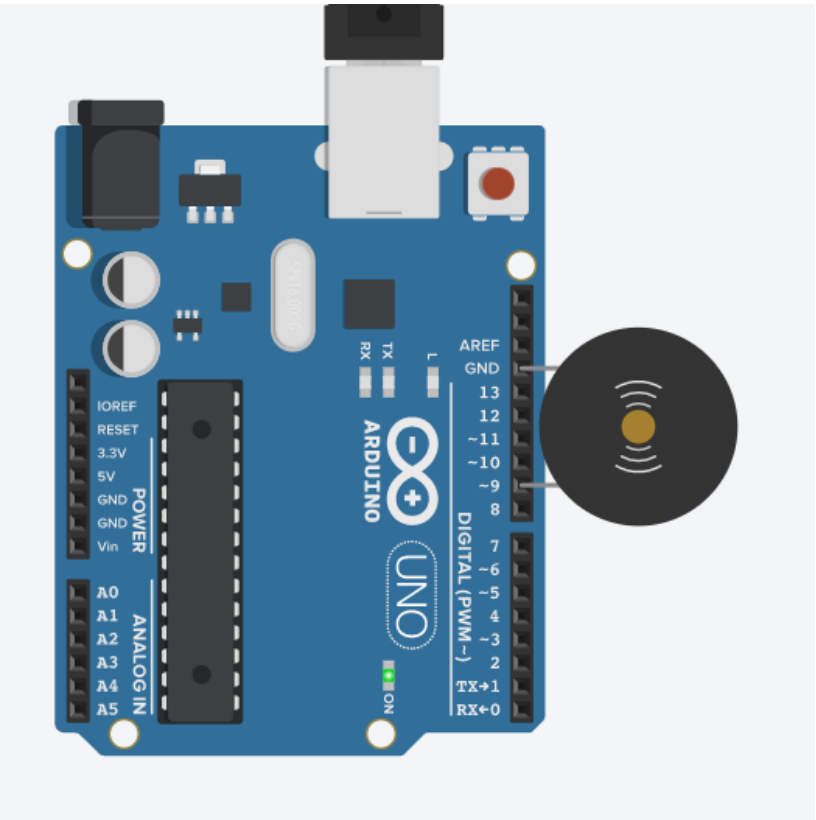
Vi kommer att lära oss om:

- hur man kopplar in en analog piezosummer i en krets.
- hur man kan styra dessa elektroniska komponenter genom att koda fram en algoritm på utvecklingsmiljön Arduino IDE.
- hur vanliga variabler deklareras, tilldelas och används.
- hur de huvudfunktionerna och funktioner som `tone()` och `noTone()` används.
- Hur INPUT_PULLUP används och ansluts till Arduino kretsen och vad det innebär denna koppling.

Elektroniska delar som behövs

- (1) Arduino Uno
- (1) USB A-to-B kabel
- (1) Kopplingsplatta
- piezosummer
- sladdar

Kretsritning



Programkod

```
/*****************************************
  Projekt 11 - Buzzer
  En buzzer/summer skapar enkla ljud.

  http://www.arduino.cc/en/Tutorial/Tone
*****************************************/

int buzzer = 9;

void setup() {
}

void loop() {
  tone(buzzer, 490);
  delay(200);
  noTone(buzzer);
  delay(100);
}
```

Övningar

Övning 1

Nu ska du spela en kort melodi med piezosummern.

Programkod

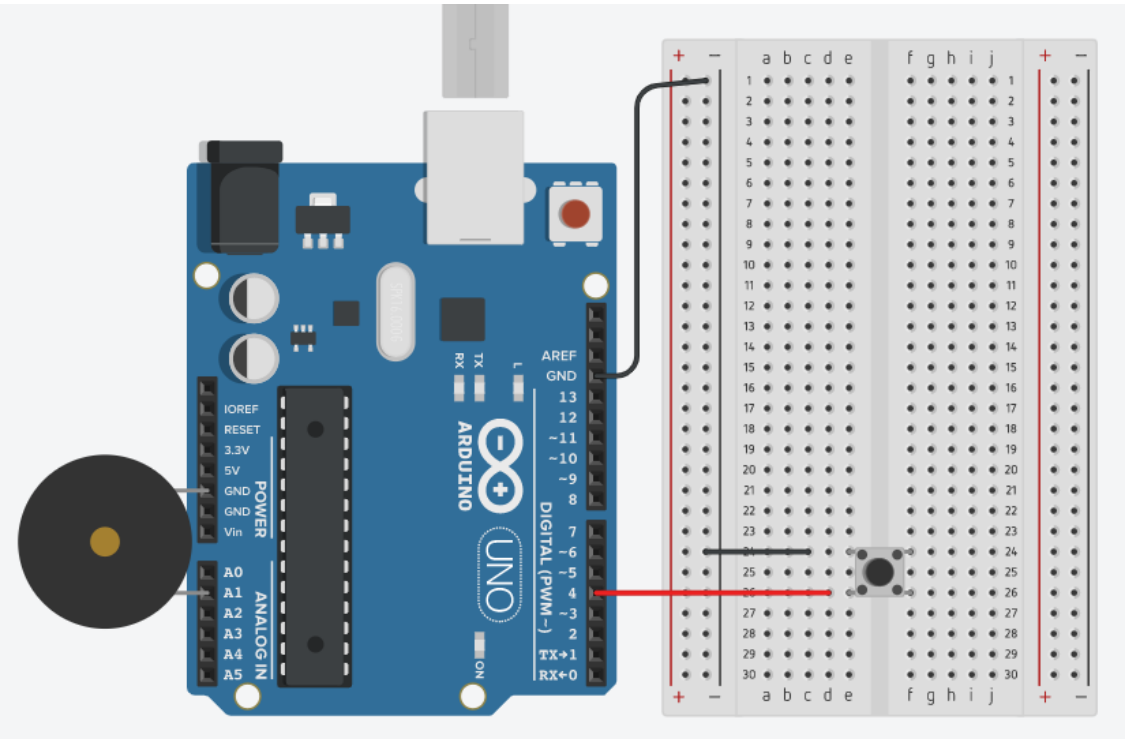
```
/*****************************************
  Projekt 11 - Buzzer övn 1
  En buzzer/summer spelar en melodi

  http://www.arduino.cc/en/Tutorial/Tone
*****************************************/
// tonerna i melodin:
#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247
int melody[] = {262, 196, 196, 220, 196, 0, 247, 262};

// tonens längd: 4 = kvarton, 8 = åttonde ton, etc.:
int noteDurations[] = {4, 8, 8, 4, 4, 4, 4, 4};
int buzzer = A1;

void setup() {
}

void loop() {
  // loopar igenom över meloditoner:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // ta en sekund för att beräkna tonens längd
    // dividerat med tontypen.
    // t.ex. kvartton = 1000/4, åttonde ton = 1000/8, etc.
    // totala längd=2500 ms
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(buzzer, melody[thisNote], noteDuration);

    // för att skilja tonerna, ställ in en minimitid mellan dem.
    // tonens längd + 30% verkar fungera bra. Totala längd=3250ms
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // sluta spela ljud
    noTone(buzzer);
    // summan av alla totala längder= 5750 ms
  }
}
```

Övning 2

Nu ska vi en tryckknappbrytare för att sätta på och av piezosummern. Vi har använt knappar på föregående projekt men med Arduino Uno kan vi ansluta tryckknappar på ett annat sätt. Denna övning nedan visar användningen av variabeln INPUT_PULLUP med **`pinMode(pin, INPUT_PULLUP)`**.
Den ena sladden (t. ex röda) går från Arduino Unos stift 4, du kan också ansluta det till ett annat digitalt stift, till tryckknappens ena ben, och tryckknappens andra ben ansluts till jorden (GND) med en svart sladd.
Ja, när tryckknappen är öppen (INTE INTRYCKT) finns det ingen förbindelse mellan de två benen på tryckknappen. Eftersom den interna uppdragningen på stift 8 och 9 är aktiva och anslutna till 5V läser vi HÖG (1). När knappen TRYCKS IN läser Arduino Uno LÅG (0) eftersom anslutningen går till jorden (GND).
Ja, det blir tvärtemot som vi har haft det förr
EJ INTRYCK KNAPP = LOW = 0
INTRYCK KNAPP = HIGH = 1

NU med variabel **INPUT_PULLUP** blir det på följande sätt:
EJ INTRYCK KNAPP = HIGH = 1
INTRYCK KNAPP = LOW = 0

Kretsritning



Programkod

```
/*****************************************
  Projekt 11 - Buzzer- ovn2
  När Tryckknappen trycks ner går igång
  piezosummer/buzzer och spelar en melodi.
*****************************************/
// tonerna i melodin:
#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247
int melody[] = {262, 196, 196, 220, 196, 0, 247, 262};

// tonens längd: 4 = kvarton, 8 = åttonde ton, etc.:
int noteDurations[] = {4, 8, 8, 4, 4, 4, 4, 4};
int buzzer = A1;
int buttonPin = 4;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  int pushValue = digitalRead(buttonPin);
  Serial.print("pushValue = ");
  Serial.println(pushValue);

  if(pushValue == LOW){
    myMelodi();
  }
  else {
    noTone(buzzer);
  }
}

void myMelodi(){
  // loopar igenom över meloditoner:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // ta en sekund för att beräkna tonens längd
    // dividerat med tontypen.
    // t.ex. kvartton = 1000/4, åttonde ton = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(buzzer, melody[thisNote], noteDuration);

    // för att skilja tonerna, ställ in en minimitid mellan dem.
    // tonens längd + 30% verkar fungera bra:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // sluta spela tonen:
    noTone(buzzer);
  }
}
```

Lista med toner

```
/*************************************************
* Public Constants
*************************************************/
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
```

```
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```