Teknik - Programmering

Programmering med Arduino

Av Staffan Melin och Martin Blom
Bild & form-skolan, Masthugget, Göteborg
2015

Staffan Melin, staffan.melin@oscillator.se
Martin Blom, martinblomblom@hotmail.com

Teknik - Programmering

Om projektet

Genomförs i halvklass, 2 x 60 minuter x 5 veckor.

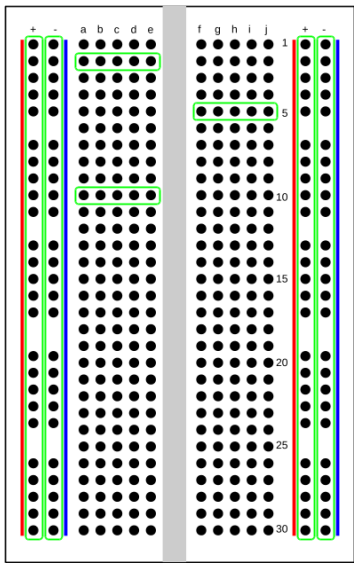
Programmering med Arduino med visst byggande med elektroniska komponenter på kopplingsbräda.

Eleverna arbetar två och två; viktigt att se till att de båda programmerar OCH bygger.

Vi kommer att arbeta med hur **program och hårdvara jobbar tillsammans**.

Komponenter

Kopplingsbräda. Används för att koppla ihop komponenter utan att behöva löda. Hålen är sammanlänkade enligt bilden nedan. Det innebär att alla sladdar som sticks ned i hål som är inom en grön cirkel blir sammankopplade.



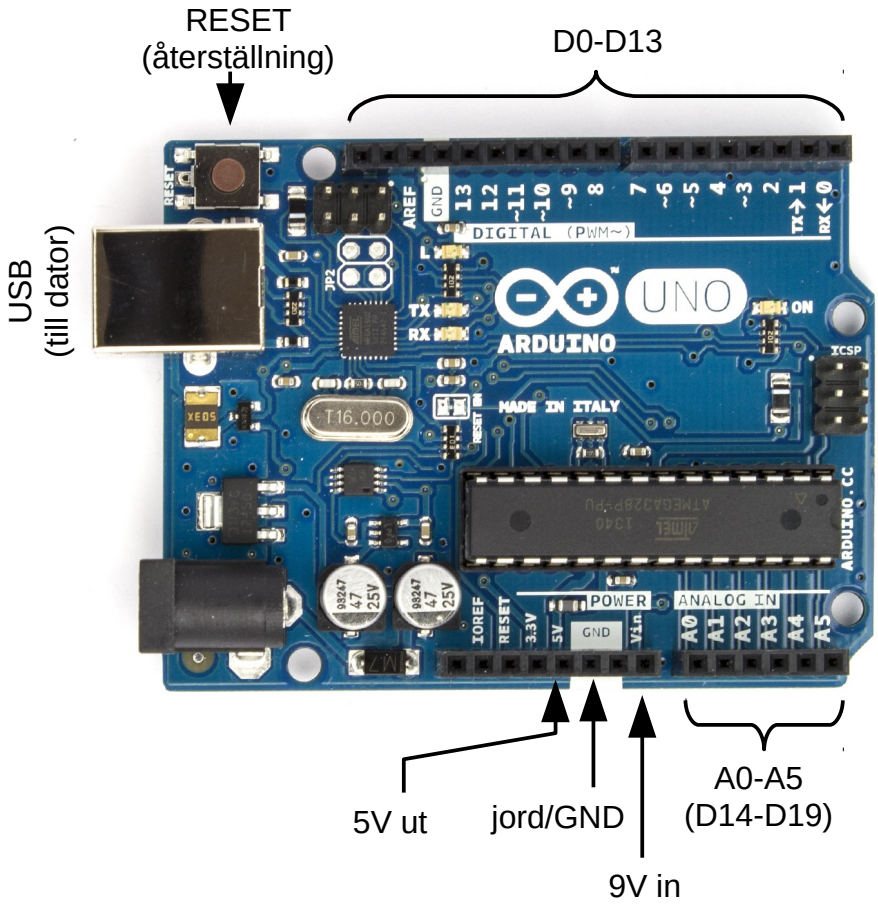
Arduino. En enkel programmerbar experimentdator som är bra på att kommunicera med omvärlden. Robotens ”hjärna”.

Det finns i huvudsak två olika sorters förbindelser hos varje komponent:

- de som skickar den **ström** som komponenterna behöver för att fungera (strömkrets)
- de som skickar **data** mellan komponenterna och Arduinon

Strömmen kommer från Arduinon, som i sin tur får den från datorn via USB-sladden.

För att kunna prata med omvärlden har Arduinon flera **stift eller portar**, dit du kan koppla komponenter (och sladdar):



- D0-D13: digitala stift/portar. De här kan ta emot och skicka två olika signaler: LOW (av, 0V) eller HIGH (på, +5V).
- A0-A5: analoga stift/portar. De kan ta emot värden från 0-1024 (normalt 0-5V). De kan också användas som digitala portar (och kallas då D14-D19).

Stiften kan ta emot eller skicka information - du talar om vilket i början av ditt program.

Det gäller med andra ord att dina kopplingar stämmer överens med hur du läser och skickar information i ditt program. Sladdar och komponenter samarbetar med programmet!

Datorer för programmering

Programmen för Arduinon skrivs på en ”vanlig” dator med hjälp av programmet **Arduino IDE**. Det kan laddas ner från Arduinons hemsida http://arduino.cc/en/Main/Software. Det finns till både Linux, Windows och Mac OSX.

Lektion 1. Introduktion till Arduino

I denna lektion lär vi oss vad en Arduino är, vad ett program är, och hur vi kan styra en lysdiod.

Utrustning

- Arduino
- USB-kabel
- dator (bärbar)

Vad är en Arduino?

Arduino är en enkel programmerbar experimentdator som är bra på att kommunicera med omvärlden.

För att kunna prata med omvärlden har Arduinon flera portar (pin), dit du kan koppla komponenter och sladdar. Portarna kan vara antingen utgångar (styr saker) eller ingångar (mäter saker). Portarna är antingen:

- **digitala**: De kan vara antingen HIGH (på) eller LOW (av).
- **analoga**: Dessa portar kan mäta spänningen och omvandla den till värden från 0 till 1023.

Arduinon måste programmeras från en annan dator via en USB-kabel.

Arduinon får sin ström från USB-kabeln. När programmet väl är överfört kan den dock drivas av ett 9V-batteri.

Läs mer om Arduinon: http://arduino.cc

Grunderna i programmering

Genom att skriva ett program talar du om för datorn vad den ska göra. Ett program skrivs ofta som en text i ett datorspråk. Det kan vara olika långt, från några rader till miljontals rader.

Obs! Det är viktigt att skriva rätt när du programmerar. Arduinon tolkar gemener och versaler som olika tecken. Du kan inte heller använda svenska tecken (åääöÅÄÖ) i namn på variabler och funktioner. Godkända tecken är a-z, 0-9 och understreck (_).

Obs! Alla rader ska avslutas med ett semikolon.

Program

Arduinon kör igång sitt program (den kan endast hålla ett program i minnet åt gången) när den får ström. När strömmen avbryts avslutas programmet (men finns kvar i Arduinons minne). Du kan starta om programmet från början genom att trycka på Arduinons RESET-knapp.

Arduino-program har två viktiga funktioner som alltid ska finnas med:

- **setup()**. Den här delen av programmet körs en gång i början.
- **loop()**. Den här delen körs gång på gång tills Arduinon inte längre får ström, eller ett nytt program laddas.

Variabler

En variabel är ett **värde** som kan ändras, och som används till exempel för att tala om hur länge en lampa ska lysa. Alla variabler har ett namn som den som skriver programmet bestämmer. Innan en variabel används i programmet måste den **deklareras**. Det innebär att du som skriver programmet talar om vilken typ av värden den innehåller och vilket namn den har. Vi använder oss i början endast av typen heltal, som på Arduinon kallas för **int** (efter engelskan integer, som betyder heltal).

Exempel på deklaration:

```
int ledPin;
```

För att stoppa in ett värde i en variabel använder du likhetstecknet, =. Det kallas för att du tilldelar en variabel ett värde.

Exempel på tilldelning av värde:

```
ledPin = 13;
```

Vi kan också “slå samman” de två raderna till en:

```
int ledPin = 13;
```

Kommandon

Rader i programmet som gör saker.

“Måsvingar” { }

Du grupperar flera rader i programmet med hjälp av “måsvingar” (även kallade “krullparenteser”). När och hur förklarar vi efter hand. Men i vårt första exempel används de för att tala om vilka kommandon som ingår i en funktion.

Kommentarer

Kommentarer är text i ditt program som inte gör något. Arduinon struntar helt enkelt i dem. De är ett sätt för programmeraren att förklara hur programmet fungerar.

Kommentarer kan se ut på två olika sätt:

```
// Detta är en kommentar på en rad

/*
      Detta är en kommentar som kan vara hur lång som helst.
*/
```

Arduino IDE

Programmet skrivs in på den bärbara datorn i ett särskilt program som kallas för Arduino IDE. Det ligger i mappen “Arduino” på skrivbordet. Då visas ett fönster som ser ut så här:

Felmeddelanden som visas kan vara lite svåra att förstå, men kan ofta bero på att du stavat fel eller glömt något tecken. Kontrollera först att du skrivit in programmet rätt och prova igen. Be annars om hjälp att tolka felmeddelandet!

Utförande

1. Koppla ihop datorn och Arduinon.

2. Kör igång Arduino IDE

3. Skriv in Blink-programmet nedanför.

4. Tryck på högerpilen i Arduino IDE så förs programmet över till Arduinon och körs igång. Om du har gjort allt rätt kommer en lysdiod på Arduinon att börja blinka.

5. Funktionen delay(1000) gör att Arduinon pausar 1000 millisekunder (1 sekund). Prova med olika värden på delay(). Hur låga värden kan du ha innan blinkandet blir omärkbart? Kom ihåg att du måste för över programmet på nytt till Arduinon varje gång du vill prova en ändring.

6. Ändra nu i programmet så att lysdioden blinkar SOS i morsekod (morsekoderna kan du hitta

exempelvis här: http://morsealfabetet.se/)

7. Om du hinner: Ändra nu i programmet så att lysdioden blinkar ditt namn i morsekod

Exempelkod: Blink

```
/*
  Blink
  Sätter på och stänger av den lysdiod som sitter fast på Arduinon.
*/

// Den inbyggda lysdioden sitter fast på port 13.

int ledPin = 13;

// Denna funktion körs en gång i början av programmet

void setup() {

  // styrporten D13 är en UTGÅNG

  pinMode(ledPin, OUTPUT);

}

// Denna funktion körs om och om igen så länge Arduinon är på

void loop() {

  // slå på lysdioden

  digitalWrite(ledPin, HIGH);
  delay(1000); // vänta 1000 ms = 1 sekund

  // stäng av lysdioden

  digitalWrite(ledPin, LOW);
  delay(1000); // vänta 1000 ms = 1 sekund
}
```

Lektion 2 - Musik

Nu ska vi skapa egna ljud. Vi kommer att styra ljudens tonhöjd med hjälp av ett program, för att på det sättet skapa melodier.

Utrustning

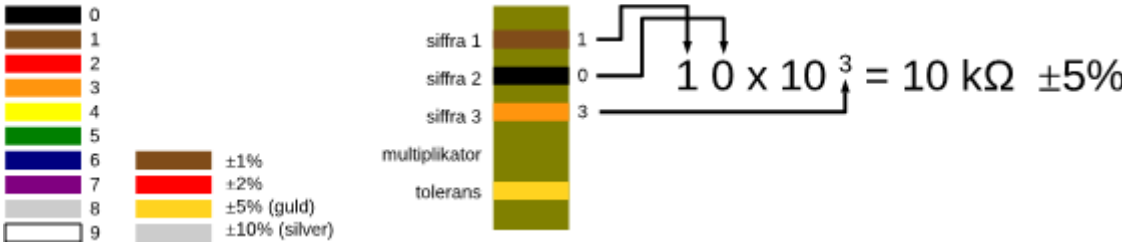
- Arduino + USB-kabel
- dator (bärbar)
- kopplingsbräda
- piezo-högtalare
- fotoresistor
- motstånd: 10kΩ
- sladdar

Komponenter

Motstånd (resistor)

Motstånd ändrar spänningen och strömmen i olika delar av kretsen. Du kan se dem som att de “gör sladden smalare” så att de släpper igenom mindre ström. Det har ingen betydelse åt vilket håll du vänder motståndet.

Motstånd finns i olika styrkor:



Fotoresistor

Ett motstånd som ändrar styrka beroende på hur mycket ljus som når det.

Piezo-högtalare

En enkel “högtalare” som kan låta i olika tonhöjder.

Funktioner

En funktion är ett antal rader med programkod som grupperats och givits ett namn.

Arduinon har en mängd inbyggda funktioner. Ett exempel är delay(). Du kan fler på: http://arduino.cc/en/Reference/HomePage

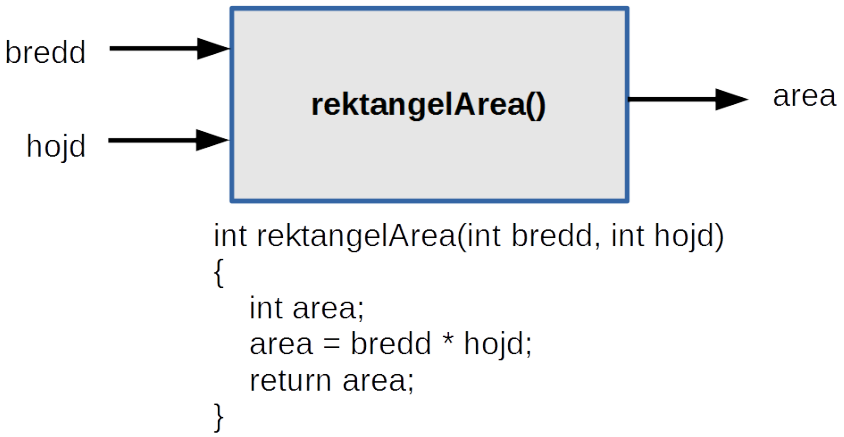
Du kan också skapa egna funktioner genom att gruppera några rader kod och ge dem ett namn (funktionens namn). Du kan sedan använda denna kod flera gånger. Du kan också låta en funktion “returnera” (skicka tillbaka) ett värde.

Exempel på hur du skapar och använder en funktion:

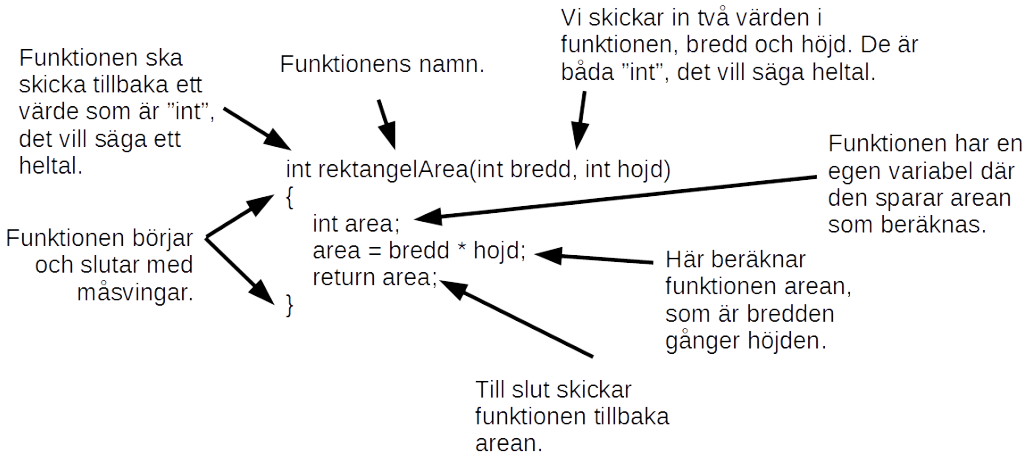
```
// Den här funktionen beräknar arean på en rektangel
int rektangelArea(int bredd, int hojd) {

      int area; // en funktion kan ha egna variabler

      area = bredd * hojd; // räkna ut arean

      return area; // “skicka tillbaka” den uträknade arean
}

// Så här använder du den i programmet

int minArea;

minArea = rektangelArea(10, 20);

// nu innehåller variabeln minArea värdet 200
```

En funktion är som en låda som gör jobbet. För att använda den behöver vi inte veta hur den ser ut ”inuti”.

Vi stoppar in bredd och höjd och får ut arean:



Så här skapar du en egen funktion:

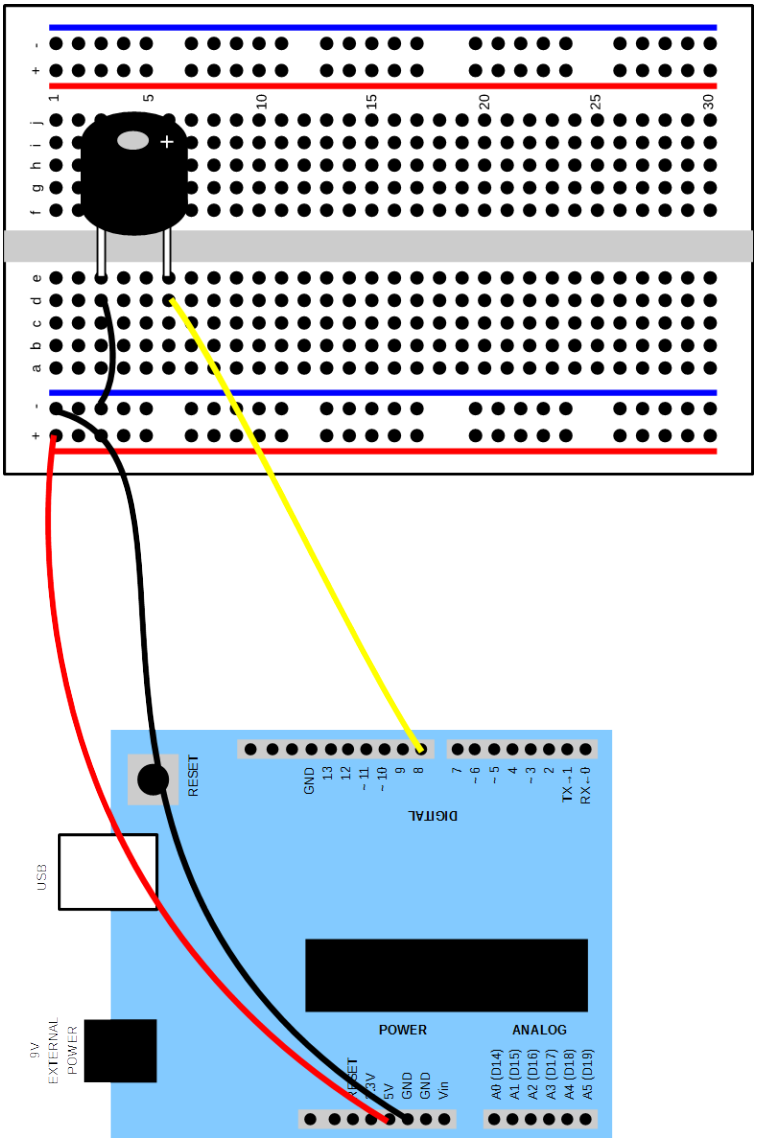


Läs mer om egna funktioner: http://arduino.cc/en/Reference/FunctionDeclaration

Teknik - Programmering

Lektion 2A - Sekvenser av toner

1. Koppla komponenterna:



2. Koppla ihop datorn och Arduinon. Kör igång Arduino IDE.

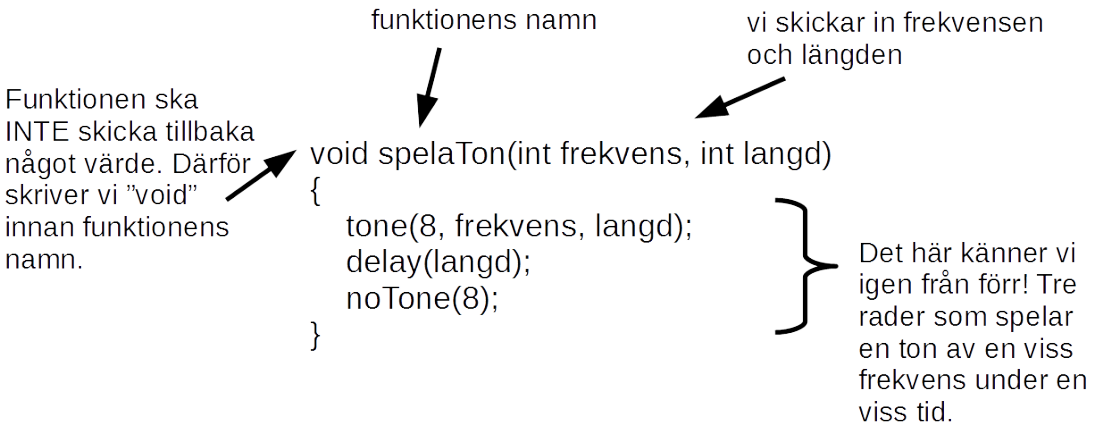
3. Skriv in och prova följande program:

```
int tonePin = 8;

void setup() {
}

void loop() {
  // Spela första noten
```

```
  tone(tonePin, 440, 125); // spela C4 i 125 ms
  delay(125); // vänta så att tonen hinner spela färdigt
  noTone(tonePin); // stäng av tonen

  // Spela andra noten
  tone(tonePin, 880, 125);
  delay(125);
  noTone(tonePin);
}
```

4. Försök få programmet att spela början på “Blinka lilla stjärna” eller någon annan enkel melodi. Om du vill använda fler tonhöjder så hittar du fler frekvenser på http://arduino.cc/en/Tutorial/tone.

5. Det går att göra programmet både kortare och snyggare genom att använda funktioner. Lägg in de tre rader som spelar en ton i en funktion. Kalla den för spelaTon, så här:



Skriv om programmet så att den använder den här funktionen:

```
spelaTon(440, 1000);
```

6. Du kan skapa egna konstanter för längden på tonerna. Om du exempelvis spelar musiken i 120 bpm och vill skapa en konstant för längden ¼ (fjärdedelar), så ska tonen ljuda i 500 ms (en halv sekund).

Du skapar en sådan konstant genom att lägga in följande rad

```
#define t4 500
```

innan raden med void setup().

Skapa nu även konstanter även för helnoter (t1), halvnoter (t2), åttondelar (t8) och sextondelar (t16).

Du använder dina konstanter genom att exempelvis skriva

```
      tone(tonePin, 440, t4);
```

för att spela en ton med längden ¼.

7. Hur gör du om du även vill ha pauser?

Inspirationskod: Sekvens

```
// Frekvens för olika toner. Mitten-A brukar ligga på 440 Hz.
#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

// Högtalaren är kopplad till port 8
int tonePin = 8;

void setup() {
      // Inget särskilt behöver göras här
}

void loop() {

      // Spela första noten
      tone(tonePin, NOTE_C4, 125); // spela C4 i 125 ms
      delay(125); // vänta så att tonen hinner spela färdigt
      noTone(tonePin); // stäng av tonen

      // Spela andra noten
      tone(tonePin, NOTE_E4, 125);
      delay(125);
      noTone(tonePin);

      // Spela tredje noten
      tone(tonePin, NOTE_F4, 125);
      delay(125);
      noTone(tonePin);

      // Spela fjärde noten
      tone(tonePin, NOTE_C5, 125);
      delay(125);
      noTone(tonePin);
```

```
      delay(2000); // vänta 2 sekunder innan melodin upprepas
}
```

Lektion 2B - En synt

Utrustning

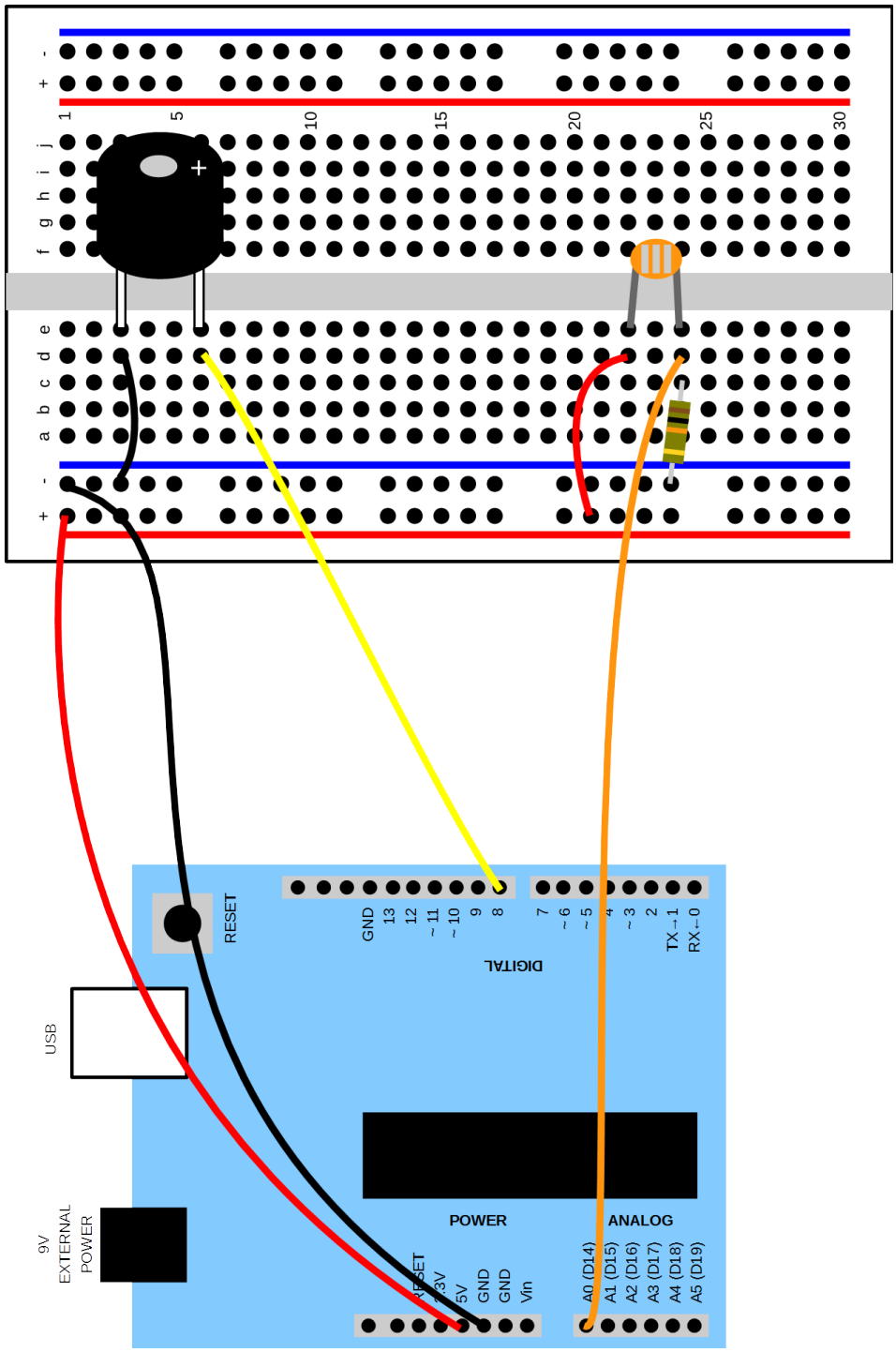
- Arduino + USB-kabel
- dator (bärbar)
- kopplingsbräda
- piezo-högtalare
- fotoresistor
- motstånd: 10kΩ
- sladdar

Utförande

Nu ska vi styra tonhöjden med handen. Det gör vi med hjälp av en fotoresistor, som vi kan skugga i olika grad med hjälp av handen. Fotoresistorn ändrar då hur mycket ström den släpper igenom, och därmed ändras spänningen i kretsen. Detta kan vi mäta på följande sätt:

Fotoresistorn är kopplad till en analog port som kan mäta och skicka tillbaka värden mellan 0 och 1023 till vårt program. (Digitala portar kan bara registrera HIGH eller LOW, alltså två olika värden.) Funktionen som vi använder för att läsa in värdet heter analogRead().

1. Koppla komponenterna:



2. Skriv in följande program.

```
int sensorPin = A0;
int buzzerPin = 8;
```

```
int sensor;

void setup() {
}

void loop() {

  sensor = analogRead(sensorPin);
  tone(buzzerPin, sensor, 100);
  delay(100);
  noTone(buzzerPin);

}
```

3. För över programmet till Arduinon. Högtalaren börjar låta. Flytta handen nära fotoresistorn. Kupa handen över den. Frekvensen ändras.

Fundera över: Vilka frekvenser kan du nu spela?

4. En ung människa kan uppfatta frekvenser mellan 20 och 20 000 Hz (hertz, svängningar per sekund).

Eftersom fotoresistorn i bästa fall skickar tillbaka värden mellan 0 och 1023, så är det enbart dessa frekvenser som kommer att spelas.

Vi kan börja med att multiplicera det vi får in från potentiometern med 4. Då kan vi spela frekvenser mellan 0 och 4000.

Gör så här:

Skapa en variabel kallad “frekvens” (på samma sätt som “sensor” som redan finns):

```
int frekvens;
```

I loop() sätter du frekvens till 4 gånger sensor, så att loop() ser ut så här:

```
void loop() {
  sensor = analogRead(sensorPin);
  frekvens = sensor * 4;
  tone(buzzerPin, frekvens, 100);
  delay(100);
  noTone(buzzerPin);
}
```

Testa programmet. Prova med andra värden än 4.

5. Det är svårt att spela en melodi eftersom tonerna ligger så tätt när du flyttar din hand.

Det vore därför bra om vi kunde “dela upp” det utrymme vi spelar på i lite större områden som spelar var sin ton.

För att kunna göra det behöver vi känna till lite mer om Arduinons språk. För att jämföra värden använder vi if (engelska för ”om”).

if (uttryck) { } else { }: Kör de programrader som finns mellan “måsvingarna” om uttrycket i parentesen är sant. Kör annars det som ligger efter “else”. Exempel:

```
if (x == 4) { // om variabeln x är lika med 4
  digitalWrite(ledPin, HIGH); // tänd lampan
} else {
  digitalWrite(ledPin, LOW); // släck lampan
}
```

Förutom jämförelsen lika med (==, obs två likhetstecken) kan vi också använda >, <, >= och <=.

Så här skulle då vår loop()-funktion kunna se ut:

```
void loop() {
      sensor = analogRead(sensorPin);

      if (sensor > 750) {
        frekvens = 1720;
      } else if (sensor > 600) {
        frekvens = 880;
      } else {
        frekvens = 440;
      }

      tone(buzzerPin, frekvens, 100);
      delay(100);
      noTone(buzzerPin);
}
```

För in ändringen och prova programmet. För handen upp och ner. Vi har nu delat in sensorns avkänning i tre områden som ger tre olika toner.

Är de värden som ni jämför sensorn med lämpliga? Ändra annars!

Kan du införa fler else-satser så att du kan spela fler toner? Kanske en hel oktav? Frekvenser hittar du i del A.

Lektion 3 - Ljus

Utrustning

- Arduino + USB-kabel
- dator (bärbar)
- kopplingsbräda
- motstånd: 220Ω
- lysdioder i olika färger
- sladdar

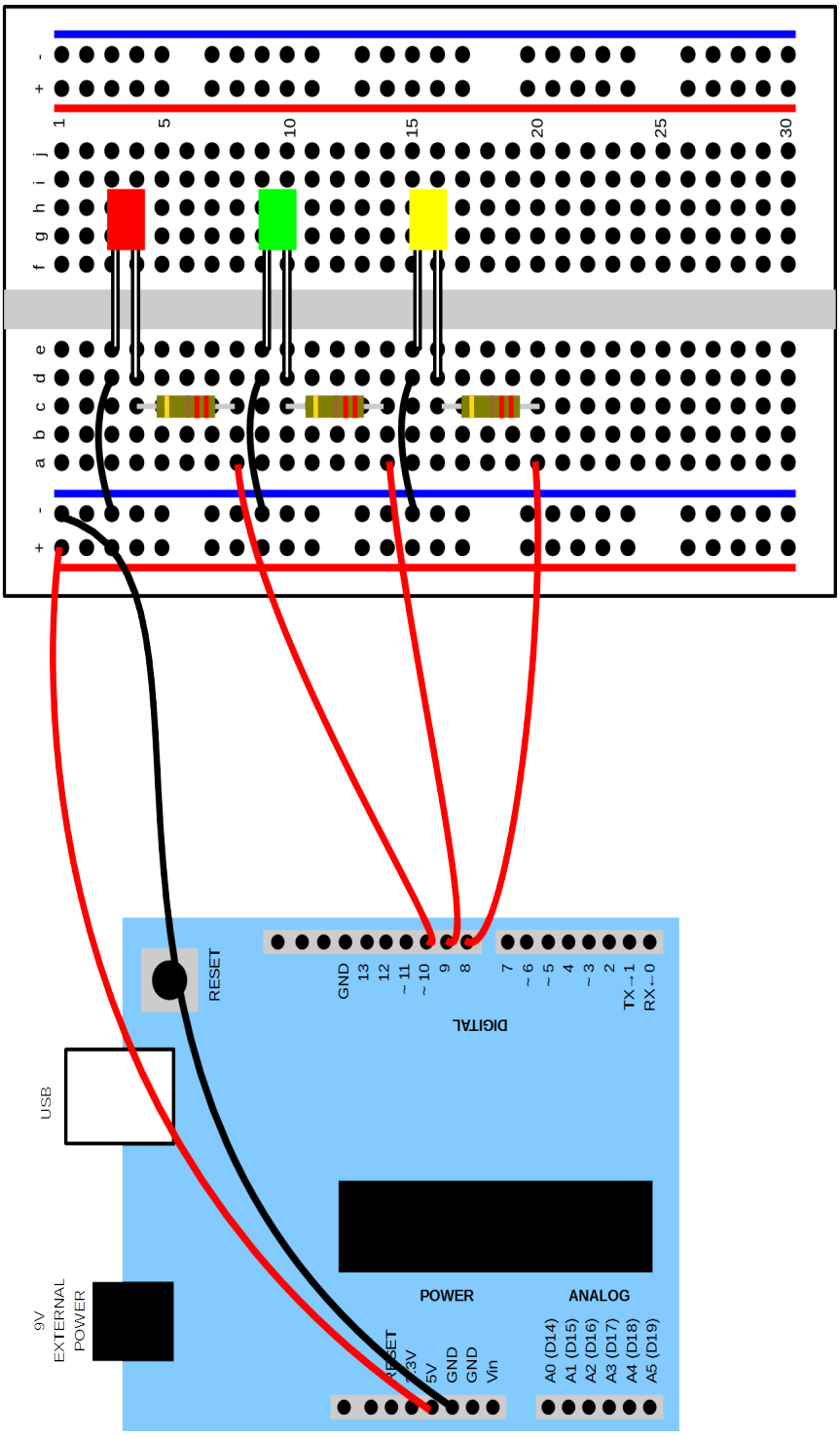
Utförande

En lysdiod (LED) avger ljus inom ett smalt spektrum (det vill säga med en viss färg) om ström leds i rätt riktning igenom den.

Ett ben på lysdioden är kortare, den är - och kallas katod. Det längre benet är + och kallas anod.

Lysdioder är svala = energieffektiva, tål stötar bra och har lång livslängd.

1. Koppla komponenterna:



2. Testa kretsen med följande program, som blinkar med den lysdiod som är ansluten till D10.

```
int pinLed1 = 10;
```

```
void setup() {
  pinMode(pinLed1, OUTPUT);
}

void loop() {
  digitalWrite(pinLed1, HIGH);
  delay(1000);
  digitalWrite(pinLed1, LOW);
  delay(1000);
}
```

3. Hur gör du om du vill blinka med en annan lysdiod?

4. Hur gör du om du vill blinka med flera lysdioder samtidigt?

5. Kan du skriva ett program som blinkar de tre lysdioder efter varandra?

Försök först själv. Ta annars hjälp av nedanstående exempel:

```
int pinLed1 = 10;
int pinLed2 = 9;
int pinLed3 = 8;

void setup() {
  pinMode(pinLed1, OUTPUT);
  pinMode(pinLed2, OUTPUT);
  pinMode(pinLed3, OUTPUT);
}

void loop() {
  digitalWrite(pinLed1, HIGH);
  delay(1000);
  digitalWrite(pinLed1, LOW);

  digitalWrite(pinLed2, HIGH);
  delay(1000);
  digitalWrite(pinLed2, LOW);

  digitalWrite(pinLed3, HIGH);
  delay(1000);
  digitalWrite(pinLed3, LOW);
}
```

6. Försök nu att kombinera lektion 2A och 3 så att du spelar toner samtidigt som lysdioderna blinkar!

Bilaga - kunskapskrav

E	C	A
Använder några tekniska begrepp när du beskriver hur en teknisk lösning är konstruerad och fungerar. (1. UNDERSÖKA OCH BESKRIVA TEKNISKA LÖSNINGAR OCH PROCESSER)	Använder alltid tekniska begrepp när du beskriver hur en teknisk lösning är konstruerad och hur dess delar fungerar och samverkar	Använder alltid, och på ett korrekt sätt, tekniska begrepp när du beskriver hur en teknisk lösning är konstruerad och hur dess delar fungerar och samverkar, samt visar på liknande lösningar
Kan bygga enkla konstruktioner och pröva lösningar för detta (3. BYGGA OCH TESTA OLIKA LÖSNINGAR)	Kan bygga enkla konstruktioner och pröva lösningar för detta samt förändra dessa lösningar för att nå ett bättre resultat	Kan bygga enkla konstruktioner och systematiskt pröva dig fram till lösningar som ger så bra resultat som möjligt
Kan med lärares hjälp ta dig framåt i arbetet (4. ARBETA SJÄLVSTÄNDIGT)	Kan med viss hjälp från lärare ta dig framåt i arbetet	Kan med självständigt ta dig framåt i arbetet
Gör enkla dokumentationer med bilder, texter och skisser (5. DOKUMENTERA DITT ARBETE)	Gör tydliga dokumentationer med bilder, texter och skisser	Gör detaljerade dokumentationer med bilder, texter och skisser