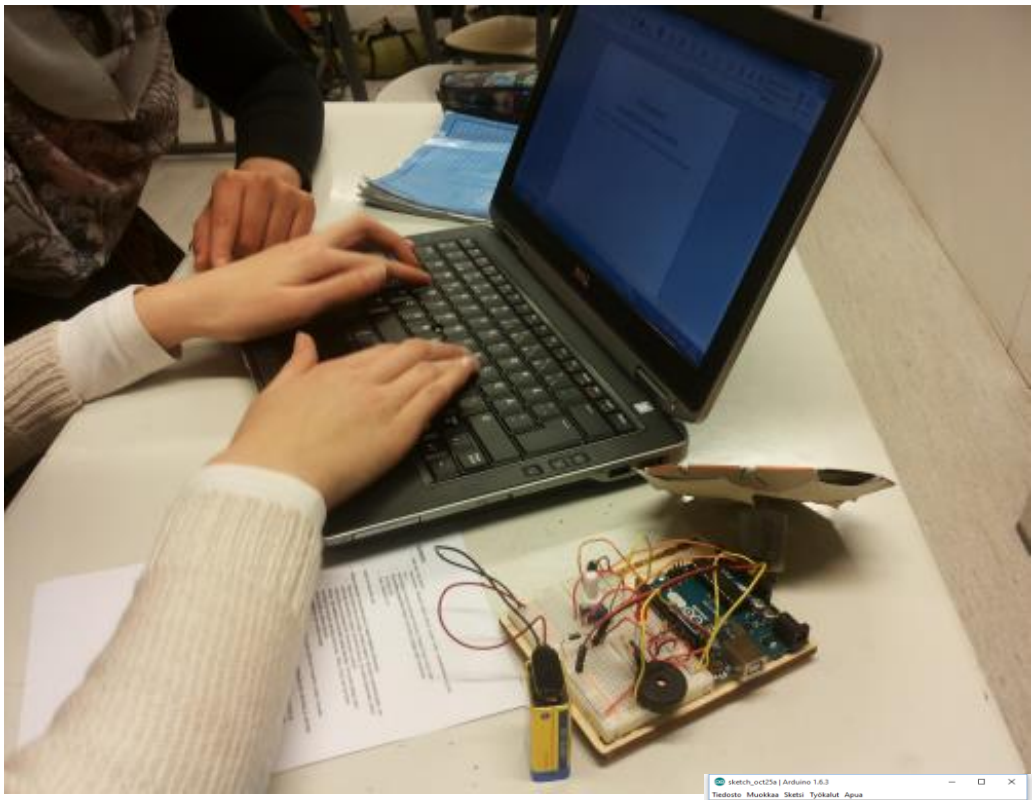
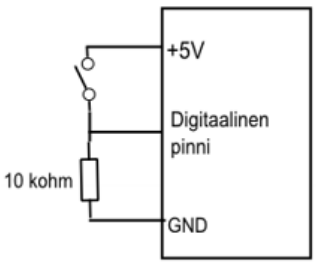
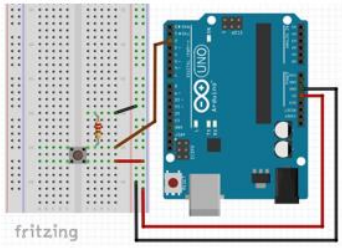
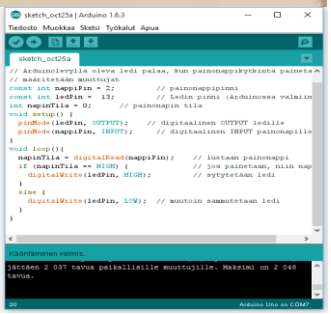
Bli bekant med Arduino

Kyösti Blinnikka









Bli bekant med Arduino

Efter att du har utfört arbeten i denna handbok har du grundläggande kunskaper och färdigheter i

- att skapa blinkande lysdiodsystem med Arduino
- att koppla tryckknappar till Arduino
- att mäta spänning med Arduino (→ mätning av spänning i ett batteri, mätning av temperatur, mätning av belysning)
- att producera ljud med Arduino
- att styra en servomotor
- att styra en elmotor
- att justera effekt (PWM ➐ dämpning av en lysdiod/justering av motorns varvtal)
- att mäta avstånd med en ultraljudssensor som är kopplad till Arduino
- att koppla en LCD-skärm till Arduino

Du kan använda dessa ”klossar” till att planlägga och pröva många olika apparater. Då du prövar olika kopplingar och program lär du dig mycket om elektroniska apparaters funktion.

Materialet passar i synnerhet för Arduino- eller programmeringskursernas material i högstadieskolor och gymnasier. Innan man börjar med uppgifterna lönar det sig att noggrant gå igenom forskningsuppgifter som finns i början av materialet (att blinka med en lysdiod, att läsa en tryckknapp, att mäta spänning). Annars kan man göra uppgifter i fri ordning.

* *

Bilder som hör till kopplingsdäcket har gjorts med programmet Fritzing http://fritzing.org/, kopplingsschemabilder med programmet Inkscape https://inkscape.org/. Foton Kyösti Blinnikka.

Tack till LUMA-centret Aalto och LUMARTS-laboratoriet för stöd med arbetet.

Opetus- ja kulttuuri- ministeriö

LUMA-KESKUS SUOMI

ISBN 978-952-93-6757-3

Kyösti Blinnikka 9/2016
kyosti.blinnikka@gmail.com

Innehåll

Arduino

”**Open Source Electronics Prototyping Platform**”

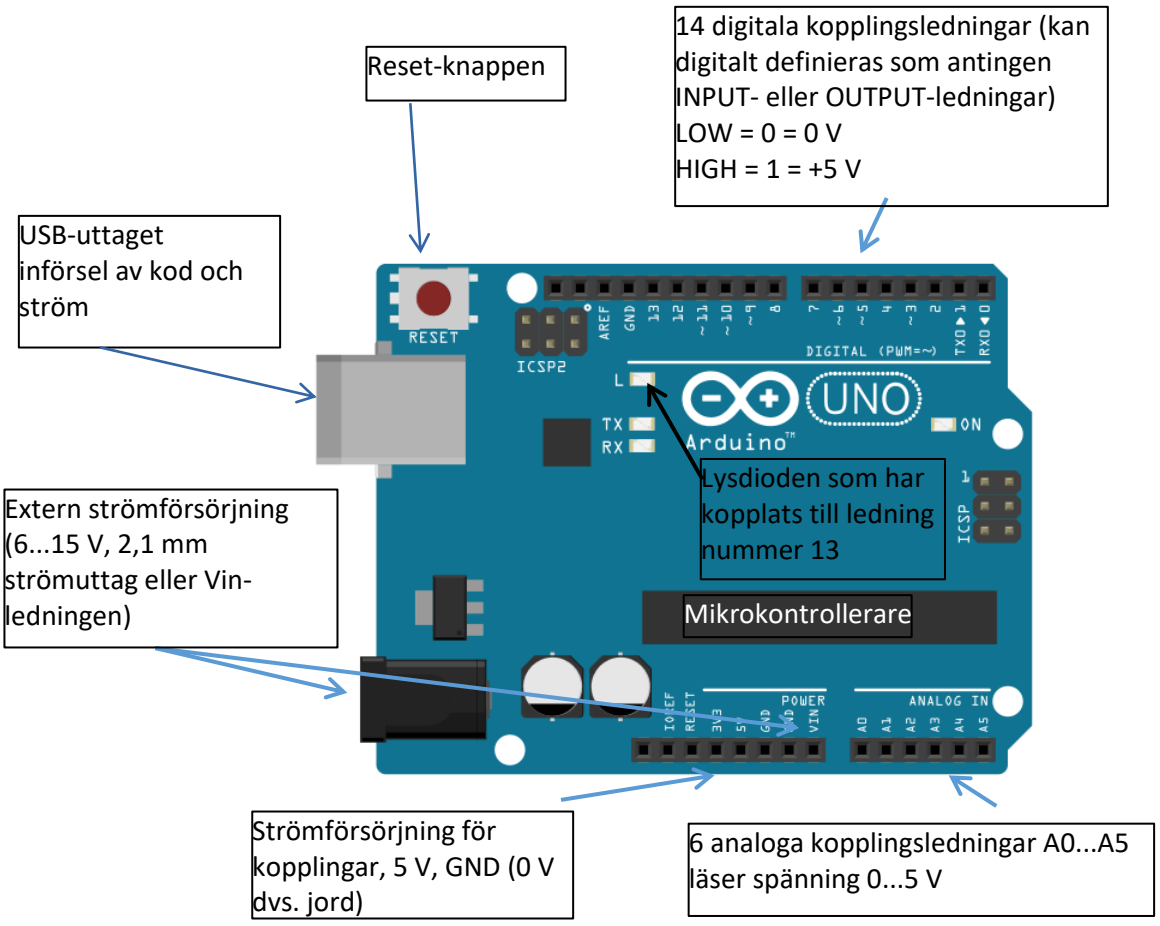
Arduino/Genuino är en mikrokontroller-elektronikplattform och programmeringsmiljö som baserar sig på öppen apparatur och programvara. En apparat som har genomförts med Arduino har två helhelter: **den externa kopplingen** som har kopplats till Arduino och **programmet** som har matats in till Arduino.

Till Arduino-kortets INPUT-poler kan man koppla olika sensorer, reglage eller kopplare, som styr lysdioder, reläer, servomotorer eller motorer som har kopplats till OUTPUT-poler. Apparatens funktioner definieras med programmet som matas in i apparaten. Arduino-miljön erbjuder ett lätt och billigt sätt att lära sig elektronik och programmering på ett modernt sätt.

Arduino är ett mycket öppet koncept och olika *duino-produkter säljs billigt i nätbutiker. I Europa säljs de ”äkta” Arduino-produkterna under namnet **Genuino**. Projektets hemsida finns på www.arduino.cc.

Arduinos delar

Arduino Uno är den populäraste Arduino-plattformen.

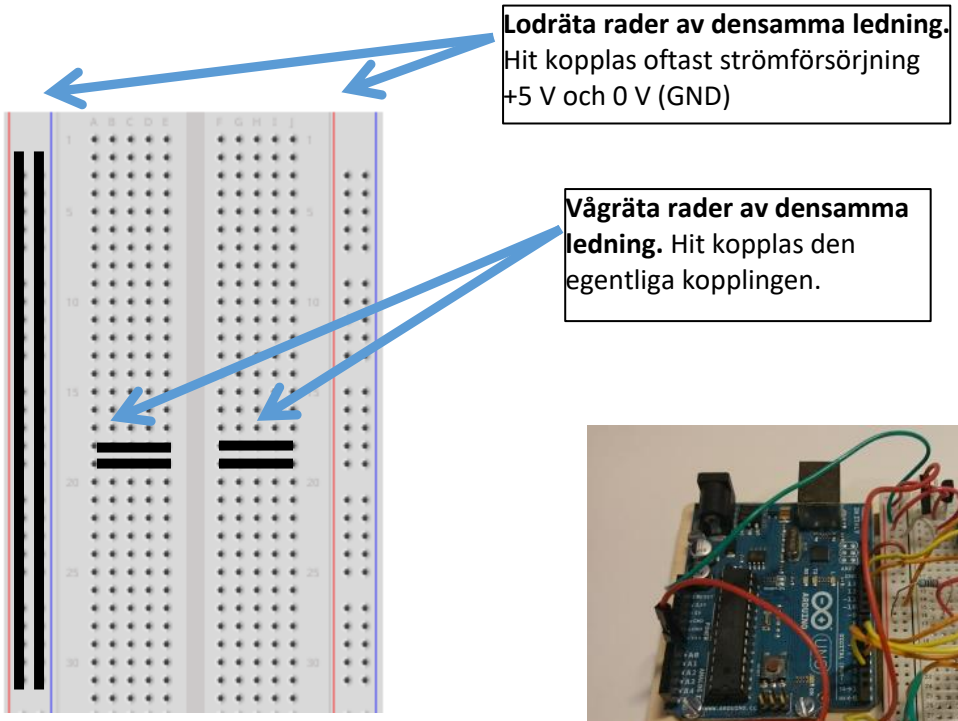


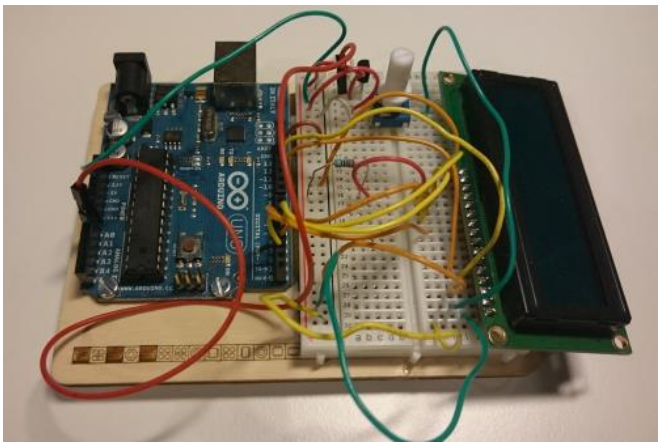
Provkopplingsdäcket

Det lönar sig att bygga provkopplingar på ett provkopplingsdäck. Provkopplingsdäcket (eng. Breadboard) är en plattform vart man kan samla en koppling utan att behöva löda komponenter fast. Apparater förverkligas genom att placera komponenterna på provkopplingsdäcket, där komponenternas ledningar som finns i samma rad är kopplade ihop. Externa kopplingar till Arduino och andra komponenter görs med förbindningstrådar.

Arduino/Genuino Starter Kit passar till börjande utövare och den är en serie där utöver Arduinokortet, komponenterna och en omfattande handbok får man en byggplattform med Arduino Uno och ett provkopplingsdäck installerade på densamma plattform. Det blir lite förmånligare att skaffa bara Arduinokortet och provkopplingsdäcket. Då behöver man dock skaffa andra komponenter separat. En lista över delar och komponenter som behövs i arbeten finns i bilaga 1.

Forskningsuppgifternas kopplingsbilder är ritade med tanke på att de skall genomföras med Starter Kit.





Komponenterna läggs mitt på kopplingsdäcket på området med kopplingspunkter i två block.

Hålrader för strömledningar på sidorna har markerats med ett rött och ett svart streck. Det lönar sig att koppla +5 V till den röda och 0 V (dvs. jord, GND) till den svarta raden så undviker man fel med apparaternas strömtillförsel.

Mellanrummen mellan hålen på provkopplingsdäcket är av standard storlek och således passar komponenterna som regel bra på kopplingsdäcket.

Programmeringsmiljö

Ladda ner Arduinos programmeringsmiljö på https://www.arduino.cc/en/Main/Software. Programmet är gratis och du kan välja knappen ”Just download”. Om du vill stöda programmets utvecklare så kan du välja en summa och knappen ”Contribute & Download”. På webbplatsen finns det versioner för Windows, Mac och Linux.

Då man installerar Windows-versionen ger installeringsprogrammet en möjlighet att automatiskt installera också en USB-drivrutin. Det lönar sig att installera drivrutinen för att utan den känner datorn inte igen Arduinokortet.

Användning av programmet

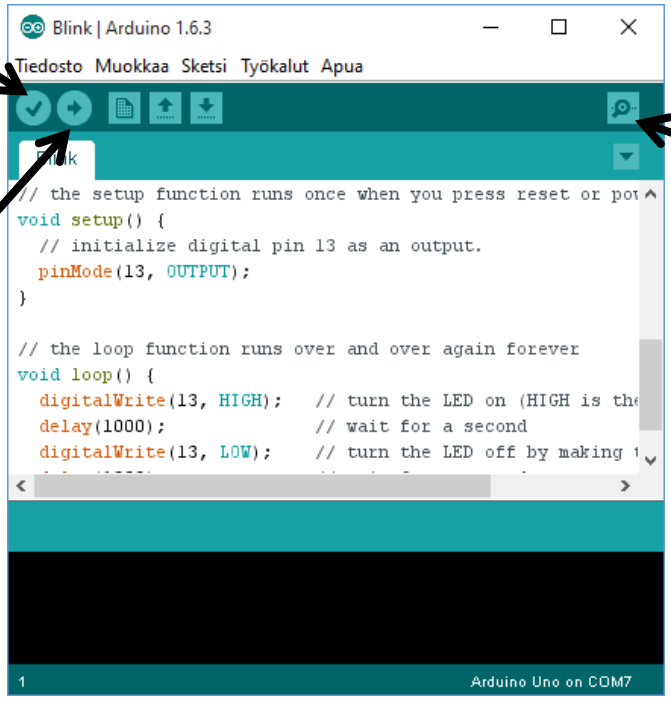
1. Öppna Arduino-programmet på datorn
2. Koppla Arduino-kortet med en USB-sladd till datorn
3. Granska i menyn Tools/Verktyg punkten Board/Kort (→ Arduino Uno) och Serial Port/Port (vanligtvis COM3 eller högre)
4. Programmet ”Sketch” laddas på Arduino från knappen med en pil som finns i övre vänstra hörnet

Detaljerade instruktioner finns på https://www.arduino.cc/en/Guide/Windows (instruktioner finns också i editorprogrammets Hjälp-funktion).

Den viktigaste hjälpsidan med programmering är Arduinos egen handbok på https://www.arduino.cc/en/Reference/HomePage (länk Learning → Reference).

Verify/Granska
Programmet granskas men inte laddas på Arduino

Upload/Ladda
Programmet laddas på Arduino



Öppnar serieportens monitoreringsfönster

Det första programmet BLINK

Programmet blinkar lysdioden som redan finns på kortet kopplad till den digitala ledningen nummer 13. Inga kopplingar behöver göras på provkopplingsdäcket.

1. Öppna Arduino-programmet på datorn. Programmeringsmiljö öppnas.

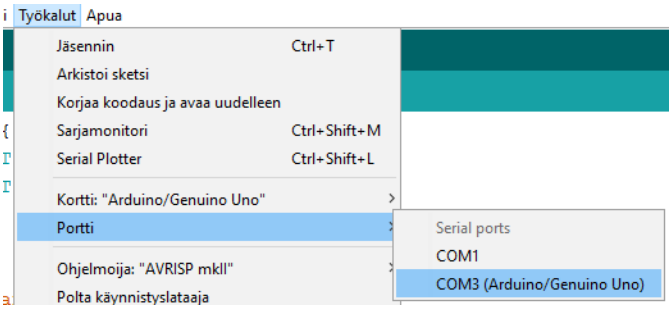
2. Öppna filen i editorprogrammet File/Fil → Examples/Exempel → Basics → Blink. Alternativt kan du skriva programmet i editor-fönstret (koden finns på nästa sida).

Ladda programmet på Arduino

3. Koppla Arduinokorttet till datorn med en USB-sladd.

Granska att datorn känner igen Arduino:

- I menyn Verktyg → Kort → Arduino/Genuino Uno,
- Verktyg → Port → (en av COM-portar finns vald, oftast COM3 eller högre).



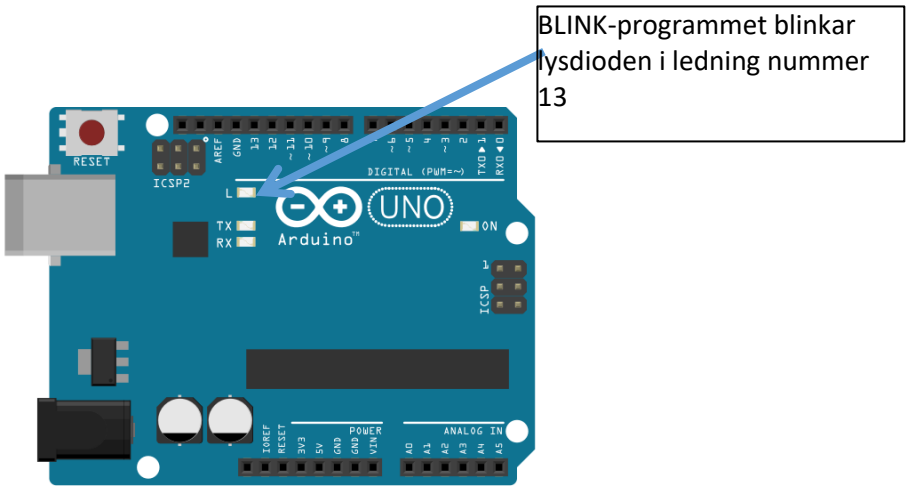
Ladda programmet på Arduino genom att klicka på verktygsfältets pilknapp.

Arduino kan på samma gång fråga om du vill spara filen.

Filen behöver inte sparas så du kan trycka på knappen ”Avbryt”.

Det laddade programmet blinkar lysdioden som har kopplats till ledningen nummer 13.

BLINK-programmet är också användbar senare, speciellt då man behöver snabbt pröva om kopplingen till Arduinokortet fungerar rätt och om kortet är i skick eller felaktigt.



```
/*  Blink
  Tänder och släcker lysdioden varje sekund
*/

// Lysdioden har kopplats till ledningen nummer 13 på Arduinokortet


// SETUP körs då programmet startas

void setup() {

  // ledningen 13 sätts som OUTPUT-polen.

  pinMode(13, OUTPUT);

}

// slingan LOOP upprepas fram tills strömmen stängs av från kortet:

void loop() {

  digitalWrite(13, HIGH);   // lysdioden på (HIGH → 5 volts spänning)

  delay(1000);              // vänta i en sekund

  digitalWrite(13, LOW);    // lysdioden av (LOW → 0 volts spänning)

  delay(1000);              // vänta i en sekund

}
```

Till bra programmeringssät hör att skriva förklarande kommentarer bland programtexten.

Programmets kommentarer har urskiljts med antingen tecken /*...*/ eller //. I kommentarerna förklaras funktion eller betydelse av den ifrågavarande satsen, kommandot eller programmets del.

Då programmet laddas på Arduino går maskinen förbi kommentarerna. Kommentarerna är anteckningar som har gjorts för programmeraren.

Programmets egentliga kommandon har markerats i det vidstående exemplet med fetstil text.

Arduino-programmet har två delar: *setup*-blocket och *loop*-blocket.

- I *Setup*-blocket definieras programmets engångsinställningar och programmets denna del körs endast en gång då programmet startas.
- *Loop*-blocket är den egentliga programslingan. Slingan upprepas så länge som kortet har strömmen på.

Studera koden och pröva

1. Studera programkoden. Pröva att konkludera vad var och ett kommando i programmet betyder. Programmets *delay*-kommando avbryter programmets körning för en sekund (1000 ms). Lysdioden blinkar alltså varje sekund.

2. Förändra tiden i *delay*-kommandot till längre eller kortare. Hur kort blink kan du ännu notera?

   Då du gör förändringar i programkoden, tryck laddningsknappen som finns i verktygsfältet. Det förändrade programmet överförs till Arduinokortet. Att bara förändra texten räcker inte!

Att blinka en lysdiod, trafikljus

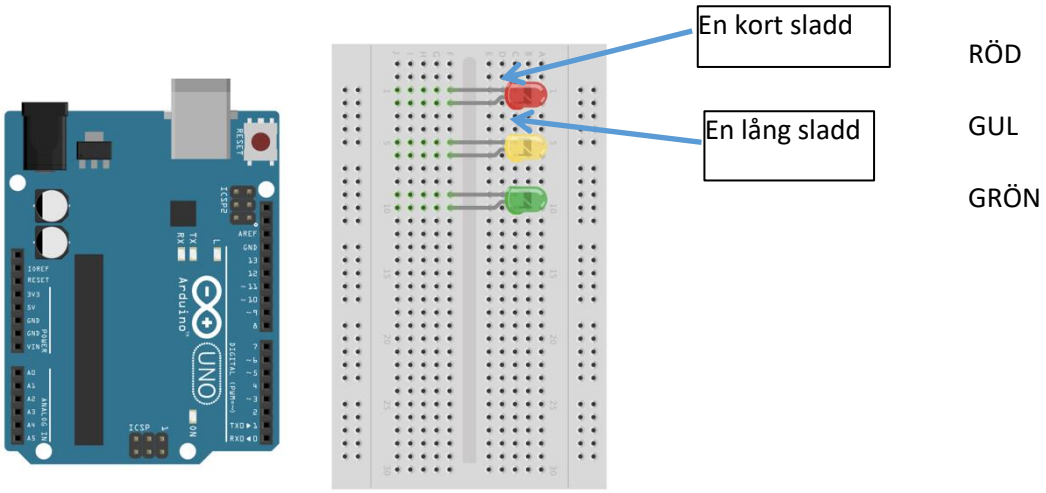
Utrustning som behövs:

- ett provkopplingsdäck och Arduino/Genuino Uno
- 3 lysdioder (röd, gul och grön)
- 3 st. 220 ohm resistorer (färgkod röd-röd-brun eller röd-röd-svart-svart. **Obs: det finns ett stycke om resistorernas färgkodsmarkeringar i slutet av handboken**)
- förbindningstrådar för kopplingar (4 st.)

Att samla ihop kopplingen

1.
Sätt Arduinokortet och provkopplingsdäcket som i bilden.

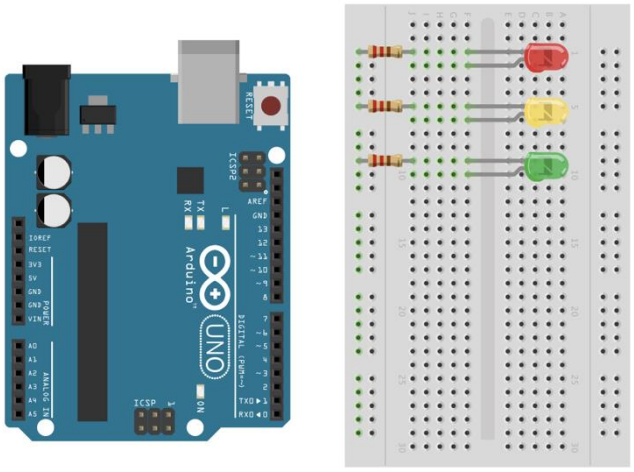
2.
Tryck en röd, en gul och en grön lysdiod på provkopplingsdäcket. Lysdioden har två ledningar. **Den längre ledningen sätts på ”undersidan”.**



3.
Lägg tre 220 ohms resistorer på provkopplingsdäcket såsom i bilden.

Runtom en 220 ohms resistor finns det färgringar röd-röd-brun (eller röd-röd-svart-svart). Resistorerna kan sättas åt vilket håll som helst, riktningen spelar ingen roll.

En av både resistorernas ledningar måste sättas till hålraden (en vertikal rad) som finns i kortets yttre kant. Ledningen som finns på lysdiodens sida måste fästas till samma rad med lysdiodens övre ledning (en horisontal rad).

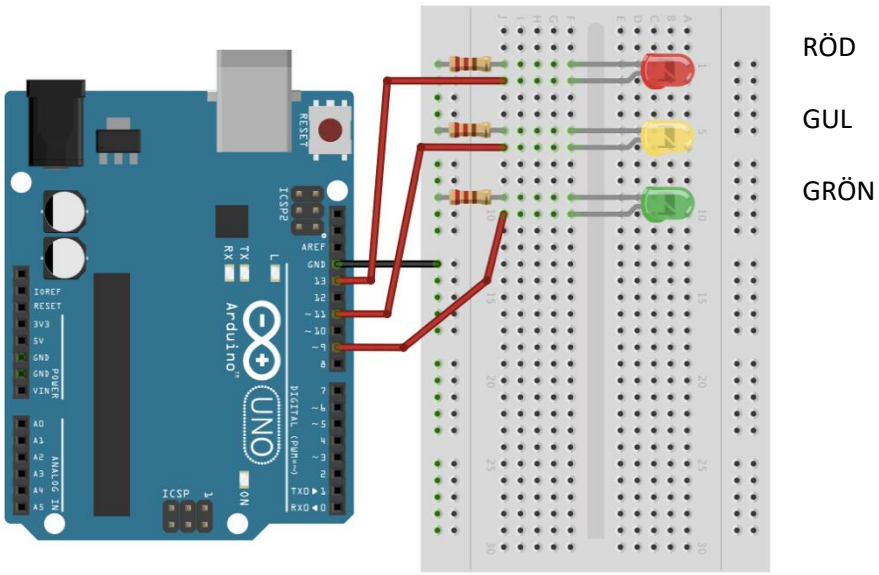


4.

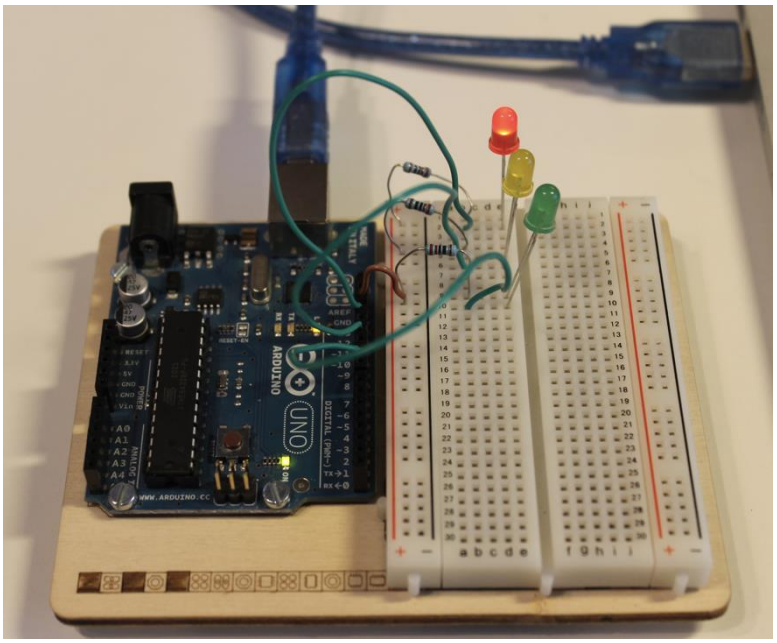
Koppla Arduinokortet till provkopplingsdäcket med tre förbindningstrådar. Trådarnas färg spelar ingen roll. Det viktigaste är att du själv kan urskilja mellan de olika ledningarna. Arduinos ledningar har numrerats på kretskortets kant.

Koppla förbindningstråden

- Från Arduinos ledning nummer 13 till den röda lysdiodens nedre kopplingsledning (på samma hålrad),
- Från Arduinos ledning nummer 11 till den gula lysdiodens nedre kopplingsledning (på samma hålrad),
- Från Arduinos ledning nummer 9 till den gröna lysdiodens nedre kopplingsledning (på samma hålrad)
- Från Arduinos ledning GND (Ground) till hålraden på provkopplingsdäckets kant.



Kopplingen är klar!



Testa BLINK-programmet:

Hitta från menyn Fil/File → Exemplar/Examples → Basics → Blink. Du får en färdig kod som ska bearbetas.

```
void setup() {
   pinMode(13, OUTPUT);
}

void loop() {
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
}
```

Komplettera programmet – att få alla lysdioder att blinkas

Tillägg de följande raderna till programmet

```
void setup() {
  pinMode(9, OUTPUT); // Ledningarna 9, 11 och 13 sätts som OUTPUT-ledningar
  pinMode(11, OUTPUT);
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // lysdioden i ledningen nummer 13 blinkas
  delay(1000);
  digitalWrite(13, LOW);

  digitalWrite(11, HIGH); // lysdioden i ledningen nummer 11 blinkas
  delay(1000);
  digitalWrite(11, LOW);

  digitalWrite(9, HIGH); // lysdioden i ledningen nummer 9 blinkas
  delay(1000);
  digitalWrite(9, LOW);
}
```

Ladda programmet på Arduino. Programmet blinkar lysdioderna men funktionen är inte än precis som i riktiga trafikljus.

Ytterligare undersökningar

1.
Förändra programmet på så sätt att du får trafikljuset att fungera ”rätt”:

- det röda lyser länge,
- då det röda byts till det gröna lyser det röda och det gula samtidigt för en stund,
- det gröna lyser länge,
- då det gröna byts till det röda lyser det gula för en stund.

Fundera på hur du får de här funktionerna till programmet?

2.
Studera programkoden och konkludera vad programmets kommandon

digitalWrite

delay

gör. Hur har kommandots ”mål” och ”effekt” markerats i koden?

Vilken betydelse måtte *pinMode*-kommandot som finns i början av programmet ha?

Varför tror du att programmet har två block, *setup* och *loop*?

3.

A. Gör ett program som blinkar alla lysdioderna samtidigt i samma takt. Ladda programmet på Arduino och testa hur det fungerar.

B. Bearbeta programmet på så sätt att lysdioderna tänds turvis och att du får ett ljus som glider från en lysdiod till den nästa.

C. Gör kopplingen till ”en metronom”. En ljusdiod (en färg) blinkar tre gånger (ett-två-tre-) och på det fjärde slaget (-fyra) lyser alla lysdioderna.

De viktiga funktionerna

pinMode(*ledning, funktion*);

Sätter ledningen som definieras i kommandot till antingen en OUTPUT- eller en INPUT-ledning.

ledning: ett av Arduinos ledningsnummer
funktion: INPUT eller OUTPUT

digitalWrite(*ledning*, *värde*);

Skriver värdet HIGH eller LOW till den digitala ledningen

ledning: ett av Arduinos ledningsnummer
värde: HIGH motsvarar 5 volts spänning i den digitala ledningen (logisk 1), LOW motsvarar 0 volts spänning i den digitala ledningen (logisk 0).

Parametern HIGH tänder och LOW släcker lysdioderna dvs. sätter ledningens spänning till antingen +5 V eller 0 V. Parametern betyder startvärden som ges till kommandot enligt vilka funktionen genomförs.

delay(*tid*);

Avbryter programmets funktion för den givna tiden.

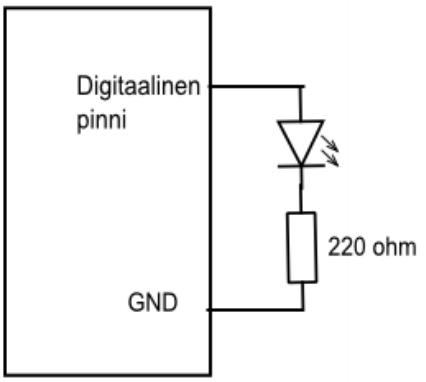
tid: tid ges i millisekunder (1000 *mS* = 1 *s*)

Uppgifter

1. Varför är det bra att kommentera programkoden?
2. Vilken betydelse har programmets setup-block?
3. I vilket block placeras det egentliga programmet?
4. Med vilket tecken skiljas programrader från varandra? Hur är det med programmets block?

Att koppla en lysdiod till Arduino

En diod är en halvledarkomponent som släpper elström till endast ett håll, till s.k. framriktning. Då dioden vänds åt andra hållet, till backriktningen, passerar elström inte genom den. LED (Light Emitting Diode) är en diod som emitterar ljus då den kopplas till framriktningen.



En diod som är i framriktning har låg resistans och då kan en stor elström passera genom komponenten även med låg spänning. Detta kan skada lysdioden. En serieresistor som begränsar elströmmen som passerar lysdioden till ca. 10...20 milliampere skall seriekopplas med lysdioden.

Mellan polerna av en glödande lysdiod verkar s.k. tröskelspänning, ca. 1,5 V. Arduinos digitala ledning har 5 volts spänning. Spänningen mellan resistorns poler är alltså *(5 – 1,5) V = 3,5 V*. Då man vet spänningen och elströmmen kan man räkna värdet på resistorns resistans:

R = U/I = 3,5 V / 0,02A = 175 Ω

Det närmaste standardvärdet på resistorer som finns i butikerna är 220 ohm som kan användas i kopplingar som lysdiodens serieresistor. Resistorer med värden 220...1000 ohm passar bra som lysdiodens serieresistor.

I handbokens kopplingar har lysdiodens serieresistor regelbundet kopplats i strömkretsen på lysdiodens ”jords” sida. Lika väl kan man koppla resistorn på andra sidan. Ordningen spelar ingen roll för elektrisk funktion.

RGB-lysdiod

Utrustning som behövs:

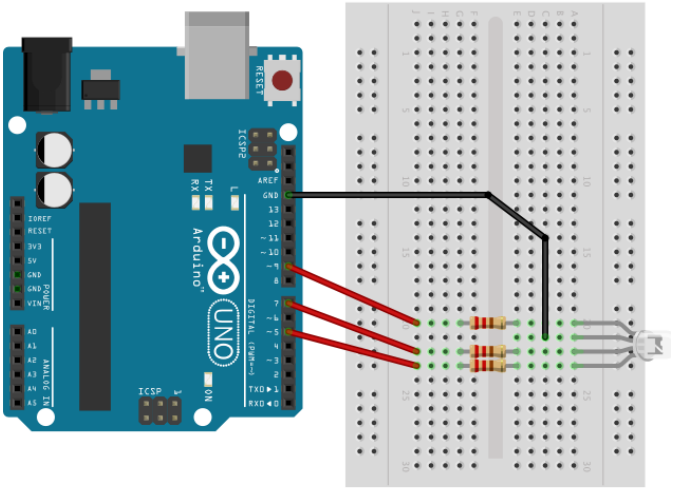
- ett provkopplingsdäck och en Arduino/Genuino Uno
- en RGB-lysdiod eller 3 lysdioder (röd, gul och grön)
- 3 st. 220 ohm resistorer (röd-röd-brun eller röd-röd-svart-svart)
- förbindningstrådar för ledningar (4 st)

En RGB-lysdiod innehåller en röd, en grön och en blå lysdiod i samma kapsel. Om du inte har en RGB-lysdiod kan du använda vanliga enfärgade lysdioder på samma sätt som i trafikljusuppgiften.

Lysdioder i en RGB-lysdiod har en gemensam katod. Det finns tre anodkopplingsledningar, en för varje färg. Kopplingsordningen är som i bilden, den längsta ledningen är katoden.

Var och en lysdiod kan styras genom att koppla Arduinos digitala OUTPUT-pol genom en 220 ohm resistor till lysdiodens anod.

Koppling



Koppla RGB-lysdioden och en 220 ohm resistor på provkopplingsdäcket. I bilden finns det en egen förbindningstråd och en resistor för varje lysdiod. Du kan också koppla resistorer direkt till Arduinos ledningar. Då är förbindningstrådarna onödiga.

Koppla RGB-lysdiodens röda lysdiod till Arduinos ledning nummer 3, gröna till ledning nummer 5 och blåa till ledning nummer 7. Det lönar sig att koppla lysdiodens katod genom provkopplingsdäckets kants kopplingsrad till Arduinos jord (GND).

Programmet

Programmet liknar det som fanns i tidigare trafikljus-programmet. Skillnaden är att man anger lysdiodernas definieringar som globala variabler. Detta bidrar till kodens läsbarhet, för att ledningarna ges ett namn som beskriver deras funktion i stället för bara ett nummer.

Pröva att tända lysdioder en lysdiod åt gången.

```
// Globala variabler som är synliga till hela programmet definieras (ledningar, dit RGB-lysdiodens olika färgerna har
// kopplats till)
int ledRöd = 3; // värdet på variabeln ledRöd sätts till 3
int ledBlå = 5; // värdet på variabeln ledBlå sätts till 5
int ledGrön = 7; // värdet på variabeln ledGrön sätts till 7

void setup() {
  pinMode(ledRöd, OUTPUT); // ledningarna sätts till OUTPUT-tillstånd
  pinMode(ledGrön, OUTPUT);
  pinMode(ledBlå, OUTPUT);
}

void loop() {
  digitalWrite(ledRöd, HIGH); // Här blinkas endast en lysdiod
  delay(1000);
  digitalWrite(ledRöd, LOW);
  delay(1000);
}
```

Först definieras lysdiodledningarna som *globala variabler.* Globala variablerna placeras till programmets början före setup-blocket. En global variabel är synlig till alla delar av programmet och man kan hänvisa till den från överallt i programmet. Tack vare denna definiering är det lätt att förändra lysdiodledningar. Man behöver inte byta ledningarnas numrering i många ställen, utan det räcker att göra förändringen till bara den ifrågavarande globala variabeln. Därutöver berättar variabelns namn instruktivt vad som är variabelns ”mål”. Koden blir lättare att läsa.

```
int ledRöd = 3;
int ledBlå = 5;
int ledGrön = 7;
```

De motsvarande förändringarna behöver också göras till programmets stam, i setup-blocket:

```
pinMode(ledRöd, OUTPUT);
pinMode(ledBlå, OUTPUT);
pinMode(ledGrön, OUTPUT);
```

Till den egentliga programslingan, loop-blocket, placeras sådana programkommandon som tänder lysdioderna:

```
digitalWrite(ledRöd, HIGH);
digitalWrite(ledBlå, HIGH);
digitalWrite(ledGrön, HIGH);
```

Man kan släcka lysdioderna med kommandon

```
digitalWrite(ledRöd, LOW);
digitalWrite(ledBlå, LOW);
digitalWrite(ledGrön, LOW);
```

Med en *delay*-sats får du ljusen att lysa under en viss period, t.ex.

delay(1000);

avbryter programmet för en sekund.

Bearbeta programmet
Försök att programmera desamma funktionerna som tidigare.

A. Bearbeta programmet så att lysdioderna tänds turvis och så att du får ett ljus som byts från en färg till den nästa. Ladda programmet på Arduino och testa att det fungerar.

B. Gör ett program som blinkar alla lysdioder samtidigt i samma takt. Vilken färg har RGB-lysdioden? Kan du urskilja de separata lysdioderna?

C. Gör ”en metronom” av kopplingen. En lysdiod (en färg) blinkar tre gånger (ett-två-tre-) och på det fjärde slaget (-fyra) lyser alla lysdioderna.

Uppgifter

1. Rita kopplingens kretsschema. Ta mall från kretsschemat med en koppling av en lysdiod. RGB-lysdioden har tre separata lysdioder.
2. Vad gör *pinMode*-satsen? Vilka *parametrar* behöver att ges till det ifrågavarande kommandot?
3. Hur får du *digitalWrite*-kommandot att hänvisa till en viss ledning? Vad betyder LOW och HIGH i *digitalWrite*-kommandot?
4. Du vill avbryta programmets funktion för 2,7 sekunder. Skriv en passande *delay*-sats.
5. Vad betyder *en global variabel?*

Programmets strukturer, FOR-slingan

Funktioner som upprepas i programmet kan utföras med till exempel en FOR-slinga.

Använd RGB-lysdiodens koppling i exemplet. Lysdioderna har kopplats till ledningarna 5, 7 och 9.

Strukturen av en FOR-slinga

```
for (int i=0; i <= 255; i++) {

// variabeln i är slingans räknare. Variabeln behöver eventuellt definieras (om den inte redan tidigare har
// definierats). Slingan kan namnges fritt (behöver inte vara i).

// Slingan utförs tills räknarens villkor uppfylls.

// Efter en runda i slingan förändras värdet på räknaren
// Markeringen i++ betyder substituering i = i + 1. Det vill säga att räknarens värde blir en högre

// Programkommandon inom klammerparenteser

 }
```

1.

```
// FOR-slingan

// Man blinkar lysdioden som har kopplats till ledningen nummer 5 på så sätt att lysdiodens lysningstid
// växer under varje for-slingans runda

int led = 5;  // globala variabler definieras och värden ges till dem;

void setup()
{
  pinMode(led, OUTPUT); // OUTPUT-ledningarna sätts
}

void loop()
{
  for (int i=1; i <= 10; i++){     // FOR-slingan, startvärdet på variabeln i är 1, värdet på i ökas med en varje
    digitalWrite(led, HIGH);       // runda tills i får värdet 10
    delay(200*i);                  // Längden av blinken ökas 0,2 s - - > 2 s
    digitalWrite(led, LOW);        // Den röda lysdioden blinkas (jfr. BLINK-programmet).
    delay(500);
  }
}
```

2.

I det följande exemplet placeras värden till *en tabell.* Tabellens värden kan hämtas till koden genom att hänvisa till tabellens element.

Lägg märke till att man också kan skriva vanlig programkod till programmets SETUP-block (alltså inte bara definieringar). I den egentliga programslingan har man använt två FOR-slingor innanför varandra.

```
// FOR-slingan

int Led[5, 7, 9];  // tabellen Led definieras och som tabellens element placeras värden 5, 7 och 9
// Tal symboliserar ledningar som har en lysdiod kopplad
// Man kan hänvisa till en viss ledning i koden genom att skriva till exempel Led[3]

void setup()
{
  for (int ledning = 1; ledning <=3; ledning ++) {
    pinMode(Led[ledning], OUTPUT); // OUTPUT-ledningar som hämtas från tabellen Led sätts
  }
}

void loop()
{
  for (int räknare=1; räknare <= 3; räknare ++) { // FOR-slingor innanför varandra

    for (int i=1; i <= 10; i++){          // en FOR-slinga, startvärdet på variabeln i är 1, värdet på i ökas
      digitalWrite(Led[markor], HIGH);    // med en varje runda
      delay(500);                         // tills i får värdet 10.
      digitalWrite(Led[markor], LOW);     // Lysdioden blinkas (jfr. BLINK-programmet).
      delay(500);
    }

  }

}
```

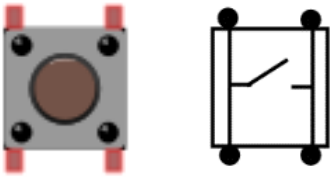
Att läsa en tryckknapp

Utrustning

- Arduino Uno, ett provkopplingsdäck och förbindningstrådar
- 2 st. tryckknappskopplare
- 2 st. resistorer 220 Ω (röd-röd-brun eller röd-röd-svart-svart)
- 2 st. resistorer kΩ, (brun-svart-orange eller brun-svart-svart-röd)
- 2 st. LED

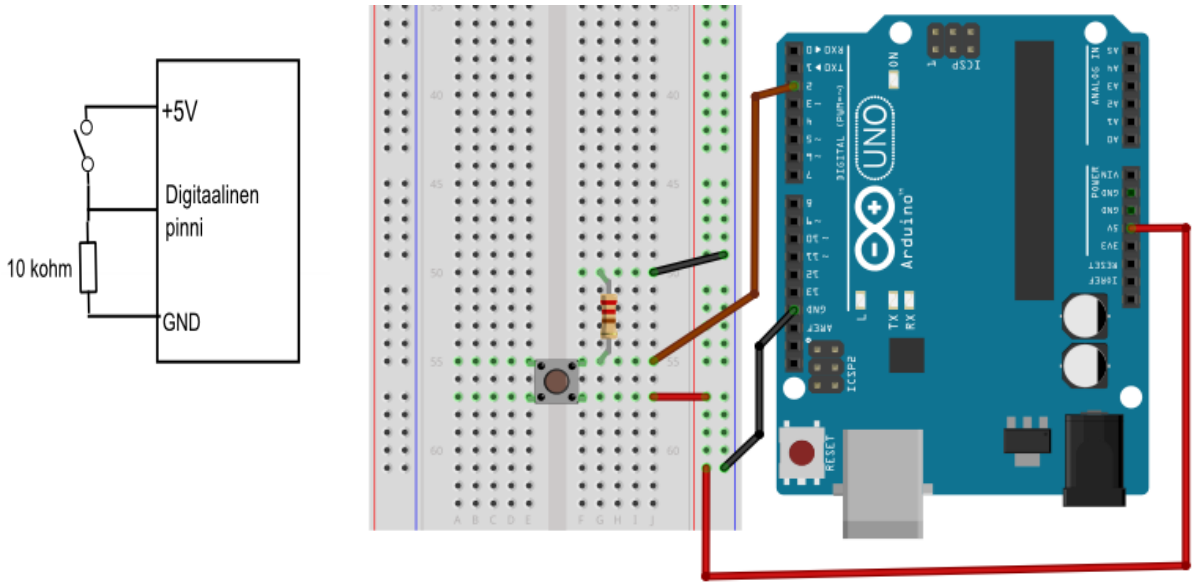
1.

Starter Kits tryckknapp har fyra kopplingsledningar. Två ledningar har kopplats ihop och kopplarfunktionens koppling finns i bilden till höger.



I samband med kopplaren används en 10 kΩ resistor som försäkrar att spänningsnivån är bestämt 0 V eller 5 V (logisk 0 eller 1). Då tryckknappen inte trycks är spänningen i den digitala ledningen 0 volt. Då tryckknappen trycks är spänningen i den digitala ledningen +5 volt.

En 10 kilo-ohm resistors färgkod är brun-svart-orange (eller brun-svart-svart-röd).



```
// Läsning av tryckknappen
// Lysdioden i ledning nummer 13 på däcket lyser,
// då tryckknappen trycks

// variabler definieras
const int knappPin = 2;          // tryckknappsledning
const int ledPin =  13;          // lysdiodens ledning
int knappensTillstånd = 0;       // tryckknappens tillstånd

void setup() {
  pinMode(ledPin, OUTPUT);  // OUTPUT för lysdioden
  pinMode(knappPin, INPUT); // INPUT
                            // för tryckknappen
}

void loop(){
  knappensTillstånd = digitalRead(knappPin);
        // tryckknappen läses

  if (knappensTillstånd == HIGH) {
// om den trycks så är knappensTillstånd HIGH:
    digitalWrite(ledPin, HIGH);  // lysdioden tänds
  }

  else {
    digitalWrite(ledPin, LOW);
// annars släcker man lysdioden
  }
}
```

2.

I den förra uppgiften användes en lysdiod som redan fanns på Arduinokortet.

A. Pröva nu att koppla en separat lysdiod. Tillägg till en av de digitala OUTPUT-ledningarna en lysdiod och en 220 ohm resistor.

   Lysdiodens kortare ledning (katoden) kopplas genom serieresistorn till ”jord” och den längre ledningen med hjälp av en förbindningstråd till den valda ledningen. Testa funktionen så att du får lysdioden att lysas genom att trycka på tryckknappen. Gör de förändringarna som behövs i programmet.

B. Pröva också att förändra programmet så att du får olika ljuseffekter från kopplingen. Till exempel så att lysdioden lyser kontinuerligt då knappen inte trycks och blinkar då knappen trycks.

3.

Tillägg en lysdiod och förändra programmet så att den första lysdioden lyser då tryckknappen trycks och den andra då knappen inte trycks.

Tryckknappsprogram använder en viktig styrningsstruktur:

if…else

Ifall villkoren som har skrivits efter ordet **if** så utförs funktion A. I annat fall utförs funktion B.

Som jämförelseoperator används tecknet == (likhet).

```
if (variabel == 500)
{
  // funktion A
}

else
{
  // funktion B
}
```

Variabler

Data som hanteras i programmet sparas till variabler. Variablerna skall **definieras** i början av programmet så att datorn kan reservera plats i minnet för dem. Definieringen förklarar vilka slags data som sparas till variabeln. Viktiga typdefinitioner är till exempel **int** och **float**.

int är en heltalsvariabel (*int = integer = heltal*),
float är en decimaltalsvariabel (*float = floating number = flyttal*).

I samband med variabelns definiering kan den **formateras**, dvs. man kan ge ett startvärde till den.

Nyckelordet **const** betyder att variabeln är en konstant dvs. dess värde inte förändras i programmet (*const = constant = konstant*).

Viktiga funktioner

int digitalRead(*ledning*);

Läser den digitala ledningens värde, antingen HIGH (logisk 1, +5 V) eller LOW (logisk 0, 0 V). Funktionen returnerar antingen värdet 1 (HIGH) eller 0 (LOW) till variabeln som har definierats i programmet.

ledning: ett av Arduinos ledningsnummer

Variabel, konstant, funktion?

I samband med programmering används många matematiska begrepp. I programmering och matematik är det viktigt att framställa saker precist och logiskt. Nyttan är dubbelsidig: matematisk kunskap kan direkt tillämpas till programmering och å andra sidan piggar upp programmering matematiska begrepp.

I programmering är en funktion (eng. function) ett kommando eller kommandosekvens som är avsedd för att utföra en viss funktion. Funktionen returnerar ofta något värde. Till exempel returnerar funktionen *digitalRead* antingen värdet HIGH (1) eller LOW (0). **Int** i funktionsdefinieringen förklarar att funktionen returnerar ett heltal (i detta fall 1 eller 0). Ifall definieringen har **void** före funktionsnamnet så returnerar funktionen inget värde (till exempel kommandot *digitalWrite*).

Variabler (eng. variable) är dataområden som har reserverats i datorns arbetsminne var data kan sparas. Information som har sparats till variabeln kan bearbetas och sökas med hjälp av programmet. Då variabeln första gången introduceras måste man förklara för datorn vilka slags data de är som ska sparas till variabeln. Till exempel variabeltypen **int** (heltal) sparar två byte lagringsplats. En byte har åtta bitar (en bit är en etta eller en nolla). **Float** eller decimaltal tar fyra byte lagringsplats.

En konstant (eng. constant) är ett tal dess värde inte förändras. Om man lägger nyckelordet **const** före variabeldefinieringen så sparas variabeln så att den kan endast läsas (”read-only”).

Uppgifter

1. Vad betyder termer a) funktion, b) variabel, c) konstant i programmeringen?

2. Hur kan man definiera en variabel i programmet? Hur kan du formatera variabeln? Ge ett exempel.

3. Vad betyder ”funktionen returnerar ett värde”?

4. Hurdana värden returnerar funktionen *digitalRead*?

5. Hitta från Arduinos egen dokumentation information om hurdant talområdes tal kan sparas till en **int**-variabel.

6. Hitta från Arduinos dokumentation hur **if...else**-strukturens jämförelsevillkor kan formuleras.

Att mäta spänning – den analoga inputen

Utrustning

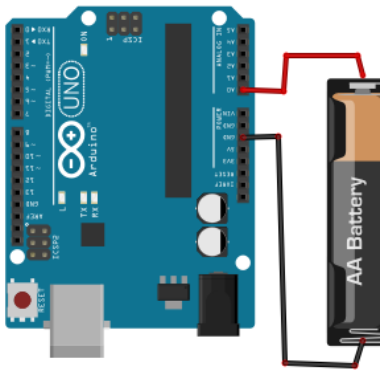
- Arduino Uno, ett provkopplingsdäck och förbindningstrådar
- 3 st. 1,5 V batterier
- En lysdiod, en 220 Ω resistor

En AD-omvandlare omvandlar analog spänning, som kan få vilka värden som helst, till ett exakt digitalt tal. Arduinokortet innehåller en AD-omvandlare som läser spänningen (0...5 V) från ett batteri eller en annan komponent som har kopplats till Analog input-polen och omvandlar den till ett heltal 0… 1023. Arduinos AD-omvandlares exakthet vid mätning är alltså 5,0 V / 1024 = 4,9 mV.

Med 5 volts spänning ger AD-omvandlaren talet 1023, med 0 volts spänning ger den talet 0. Värden som ligger däremellan kan räknas, till exempel 1,0 V spänning motsvarar heltal (1 V/5 V)*1023 = 205.

1.
Skriv den vidstående koden och ladda den till Arduino.

```
// Läsning av spänning från den analoga ledningen

void setup() {
  Serial.begin(9600); // serieportens hastighet sätts till 9600 bitar per sekund
}

void loop() {
 // spänning i den analoga ledningen A0 läses
 int lukema = analogRead(A0);
 // värdet skrivs ut till serieporten
 Serial.println(lukema);
 delay(1000);    // dröjsmålet för att läsa värden varje sekund
}
```

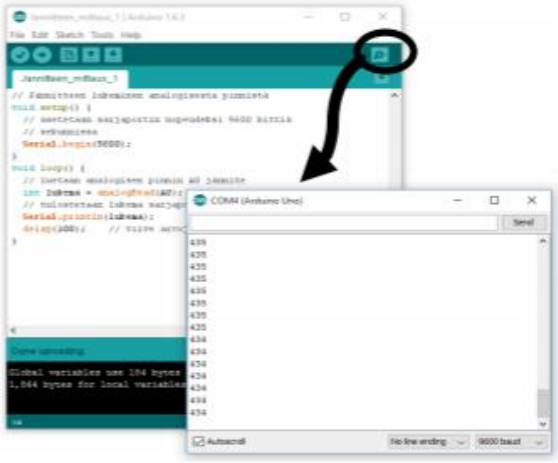


Koppla pluspolen av en 1,5 batteriets till Arduinos analoga ledning A0. Koppla batteriets minuspol till GND-ledningen.

På samma sätt kan du studera två och tre seriekopplade batterier. Batterier kan du seriekoppla enkelt så att du lägger batterier efter varandra. Som hjälp kan du använda en öppen batterikapsel om du har en sådan tillhanda.

Öppna monitoreringsfönster genom att klicka på editorprogrammets seriemonitorknapp. Granska att dataöverföringshastighet som finns i monitor-fönstrets högra nedre kant är densamma som har definierats i programmet (9600 bit/s).

Att öppna seriemonitor-fönstret



Pröva vilka värden du får med ett batteri (nominala spänningar 3,0 V och 4,5 V). **Seriekoppla inte fyra batterier för att den analoga inputen är avsedd för spänningar under 5 volt.** Anteckna värden från varje fall.

Koppla förbindningstråden som är kopplat till ledningen A0 direkt till Arduinos egna spänningsledningar GND, 3.3 V och 5 V. Anteckna värden också från dessa mätningar.

Räkna spänningar i batterikopplingarna baserad på mätvärden. Använd formeln

$$spänning = \frac{värde}{1023} * 5V$$

Samla mätvärden till en tabell. Mät spänningar också med en digital universalmätare.

Batteri/ledning	**AD-omvandlarens värde**	**Spänning (V) som har räknats från värdet**	**Spänning (V) som har mätts med universalmätaren**
1 batteri			
2 batterier			
3 batterier			
5 V ledning			
3,3 V ledning			
GND ledning			

Motsvarar spänningsvärden förmodade batterispänningar (nominala spänningar 1,5 V, 3,0 V, 4,5 V)? Vad med Arduinokortets egna spänningar?

Spänningen av ett nytt 1,5 volts batteri kan ligga omkring 1,6 volt och spänningen av ett batteri som har använts länge kan ligga rejält under den nominala spänningen.

Pröva också så att du håller endast en ledning i den analoga ledningen. Ta din hand nära ledningen och rör i den. Iaktta värden i monitor-fönstret. Du kommer att märka att den analoga input-polen är mycket känslig för störningar. Värdet kan variera mycket.

2.
Bearbeta det förra programmet så att du får en spänningsmätare. Resultat får du direkt i volt.

Skriv den vidstående koden och ladda den till Arduino.

```
// Spänningsmätaren. Läsning av spänning från den analoga ledningen och utskrivning i volt

void setup() {
  Serial.begin(9600); // serieportens hastighet sätts till 9600 bitar per sekund
}

void loop() {
 // spänning i den analoga ledningen A0 läses
 int lukema = analogRead(A0);
 // värdet omvandlas till volt och datatypen förändras med detsamma
 float jannite = (lukema/1023.0)*5.0;
 // värdet skrivs ut till serieporten
 Serial.print("Spänning i volt: "); // rapporttext skrivs ut
 Serial.println(jannite); // spänningsvärdet skrivs ut
 delay(1000);    // dröjsmålet för att läsa värden
}
```

Pröva kretsens och programmets funktion med ett, två eller tre seriekopplade batterier. Är resultaten förväntade?

3.
Gör en testapparat för batterier.

Lägg en röd och en grön lysdiod och 220 ohms resistorer på provkopplingsdäcket. Koppla lysdioder till två av de digitala output-ledningarna. Vid behov ta mall från lysdiodens blinkningsuppgifter. Rita kopplingens kretsschema till häftet.

Bearbeta programmet i den förra uppgiften så att du får en spänningslarmapparat: lysdioden tänds då spänning underskrider värdet som har definierats i programmet i förväg. 1,5 volts batteri som används kan ha en spänning på under 1,4 volt. Spänningen i ett nytt batteri är 1,5–1,6 V.

Du kan använda en if – else -styrningsstruktur i programmet så här:

```
if (värde < 500) {
  // lysdioden 1 tänds
  // lysdioden 2 släcks
}

else {
  // lysdioden 2 tänds
  // lysdioden 1 släcks
}
```

För lysdioden måste man också tillägga *pinMode*- och *digitalWrite*-kommandon till rätta ställen. Pröva!

Utred vilket skulle vara ett passande värde för ett jämförelsevärde. Använd serieportens monitor och spänningsvärden och värden i AD-omvandlaren som mätts i förra uppgiften som hjälp. Då programmet är färdigt kan du ”kommentera” bort programrader som hör till serieporten (tillägg tecknen // i början av programraden så blir raderna överhoppade när programmet kompileras).

Du kan också tillägga en gul lysdiod till kopplingen. Programmera funktionen så att den gröna lysdioden lyser om batteriets spänning är över 1,5 V, den gula lysdioden lyser då batteriets spänning är inom 1,4–1,5 V och den röda lysdioden lyser då spänningen är under 1,4 volt. I detta fall se instruktioner för if – else-strukturen i Arduinos egen dokumentation.

Du kan testa programmets och kretsens funktion med en justerbar spänningsskälla. **Kom ihåg att Arduino är avsett för högst 5 volts spänning. Använd aldrig större spänning eftersom Arduinokortet kan gå sönder!**

De viktiga funktionerna

int analoglRead(*ledning*);
Läser spänningen (0...5 V) i den analoga ledningen och omvandlar spänningsvärdet till ett heltal mellan 0...1023. Funktionen returnerar det ifrågavarande heltalet till variabeln som har definierats i programmet.
ledning: ett av Arduinos ledningsnummer

Serial.begin
Sätter dataöverföringshastighet, t. ex. **Serial.begin(9600);** sätter hastigheten till värdet 9600 bit/s.

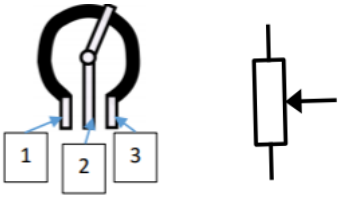
Serial.println(värde)
Skriver ut text till serieporten på så sätt att raden slutar till ett radbyte. Raden kan läsas i serieportens monitorfönster. **Serial.println(värde);** skriver ut värdet på variabeln *värde*. Om du vill skriva ut text, lägg texten inom citattecken, till exempel **Serial.println(”spänningen är: ”);**.

Uppgifter

1. Gör en tabell och anteckna till den heltal som motsvarar Arduinos AD-omvandlarens spänningar 0 V, 1 V, 2 V, 2,5 V, 3 V, 4 V och 5 V.
2. Skriv kretsscheman för de olika kopplingarna.
3. Berätta med egna ord hur man får spänningens beräkningsformel.

Spänningsdelaren – att läsa värdet i en potentiometer

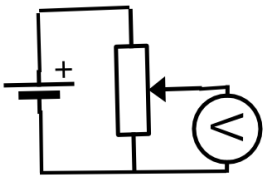
En potentiometer är en justerbar resistor som har tre anslutningar så som i bilden. Potentiometern har en hästskoformad resistorremsa som har framställts av till exempel kol. Anslutningarna har fästats till remsans ändar. I mitten, kopplad till en axel, finns en glidkontakt som kan steglöst flyttas på resistorremsan.



Potentiometerns struktur har också avbildats i symbolen. Pilen avbildar glidkontakten.

En 10 kilo-ohms potentiometer har 10 kΩ resistans mellan polerna 1 och 3. Resistansen mellan polerna 1–2 eller 2–3 kan fritt justeras med glidkontakten mellan 0–10 kΩ.

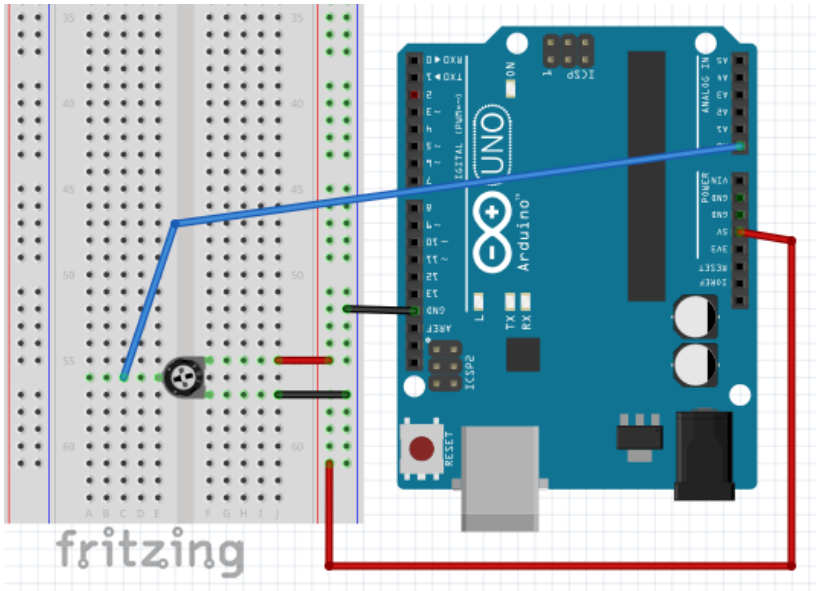
Potentiometern kan kopplas som en **spänningsdelare**. Vanligtvis har potentiometerns ena kantanslutning kopplats till spänningskällans ena pol (t.ex. Arduinos +5 V) och den andra anslutningen till jord (Arduinos GND-ledning, alltså 0 V). Då kan man reglera vilken som helst spänning mellan noll och fem volt till den mellersta anslutningen.

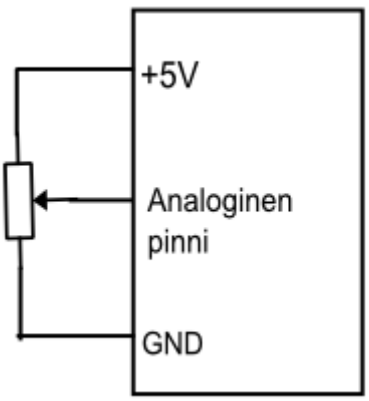


Utrustning

- Arduino Uno, ett provkopplingsdäck och förbindningstrådar
- en 10 kΩ potentiometer
- en 220 Ω resistor
- en lysdiod

1.
Gör kopplingen.





Pröva förra arbetets (mätning av batteriets spänning) program. Bearbeta programmet så att du kan läsa mätresultatet både som ett heltal och i volt.

Koden läser spänningen i den mellersta anslutningen i potentiometern och skriver ut mätresultatet till datorns monitor-fönster.

2. Tillägg en resistor och en lysdiod på provkopplingsdäcket. Koppla dem till en av de digitala output-ledningarna. Skriv ett program som blinkar lysdioden. Blinkningsfrekvensen kan justeras med potentiometern.

Tillägg en till lysdiod till kopplingen som blinkar i motsatta takten.

```
// justering av lysdiodens blinkningsfrekvens med potentiometern

int led = 3;  // lysdioden till ledningen nummer 3

void setup() {
   pinMode(led, OUTPUT); // ledningen 3 sätts som OUTPUT-ledning
}

void loop() {
  int aika = analogRead(A0);  // spänning i den analoga ledning A0 läses
  digitalWrite(led, HIGH); // lysdioden tänds
  delay(tid);    // dröjsmålet, tid är spänning som lästs från potentiometern
  digitalWrite(led, LOW); // lysdioden släcks
  delay(tid);    // dröjsmålet, tid är spänning som lästs från potentiometern
}
```

Att producera ljud

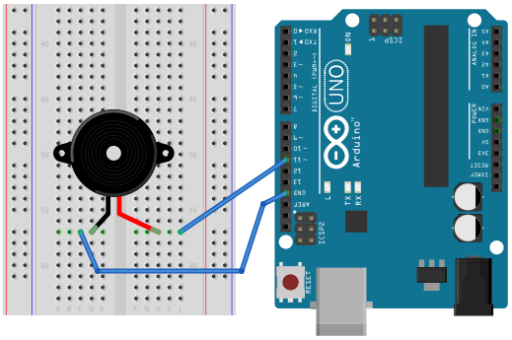
Du kan producera ljud med Arduino.

tone(ledning, frekvens, (varaktighet))
producerar ett ljud till output-ledningen, ljudets varaktighet ges i millisekunder.

noTone(ledning)
ljudet i ledningen dämpas

tone-kommandot producerar fyrkantsvåg med den givna frekvensen till den digitala output-polen. Man kan ge endast ett ljud per gång.

1.
Koppla en piezoelektrisk högtalare till Arduinos ledningar 11 och GND. Testa koden. För kommandot *tone* behövs ingen ledning sättas till OUTPUT-moden i setup-blocket (dvs. man behöver inte skriva ett *pinMode*-kommando). Den vidstående kopplingen och programmet producerar tonerna i en C-durackord.



Pröva hur ljudet låter utan kommandot **delay**. Vad kan detta bero på? Du kan enkelt ”kommentera” delay-raderna bort genom att tillägga två snedstreck (//) i början av raden. Då hoppar kompilatorn över raderna när programmet laddas.

```
void setup() {
}

void loop() {
  tone(11, 262, 500);  // ett ljud på 262 Hz produceras (tonen C)
  delay(1000); // ett dröjsmål på 1 sekund
  tone(11, 330, 500);  // ett ljud på 330 Hz produceras (tonen E)
  delay(1000); // ett dröjsmål på 1 sekund
  tone(11, 392, 500);  // ett ljud på 392 Hz produceras (tonen G)
  delay(1000); // ett dröjsmål på 1 sekund
}
```

2.
”Spela” någon enkel melodi med Arduino.

Tips:
Frekvenser för tonerna i C-durackorden:
C 262 Hz, D 294 Hz, E 330 Hz, F 350 Hz, G 392 Hz, A 440 Hz, H 494 Hz; C 523 Hz.

(http://www2.siba.fi/akustiikka/index.php?id=18)

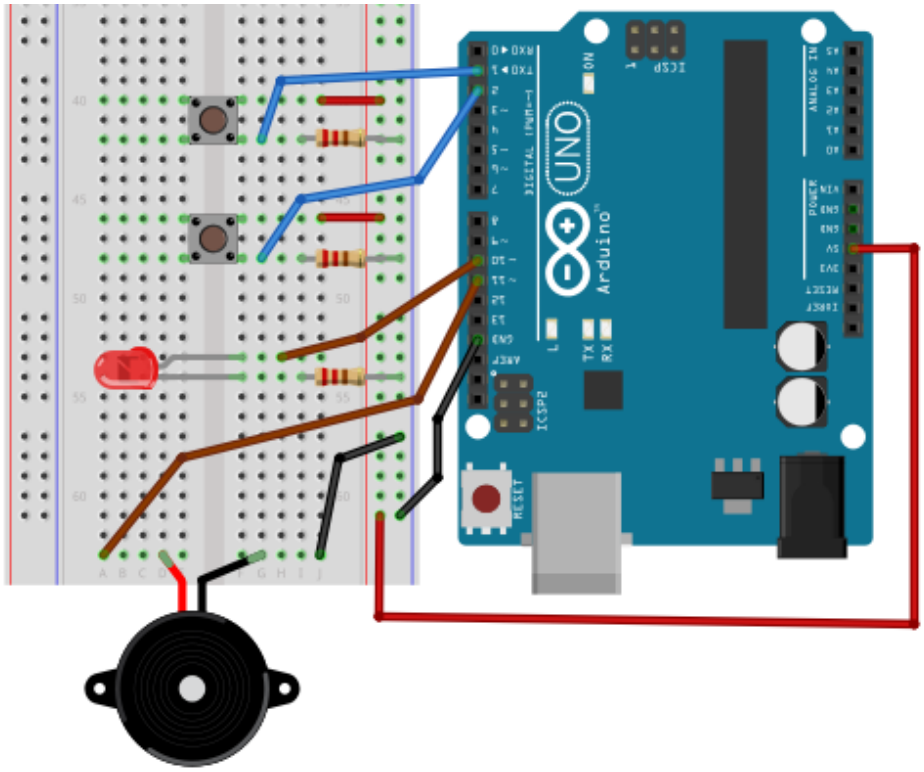
3.
Tillägg en 10 kilo-ohms resistor till kopplingen och en tryckknapp till ledningen nummer 2 (se exempelkopplingen i de föregående uppgifterna). Med det vidstående programmet producerar kopplingen ett ljud då tryckknappen trycks.

```
// TONE ljudproducering 2
// konstanter definieras

const int högtalare=  11;      // ledningen som högtalaren har kopplats i
const int knapp = 2;     // ledningen som tryckknappen och resistorn har kopplats i
int knappensTillstånd = 0;        // variabeln som knappens tillstånd läses till (on/off)

void setup() {
  pinMode(knapp, INPUT); // INPUT-ledningen sätts
}

void loop() {
  // knappens tillstånd läses:
  knappensTillstånd = digitalRead(knapp);
// om knappen trycks,
// är variabelns tillstånd HIGH:

  if (knappensTillstånd == HIGH) {
    // ett ljud på 300 Hz produceras
    tone(högtalare, 300);
    }
   else
  {
    // Om knappen inte trycks är dess tillstånd LOW
    // högtalaren dämpas
    noTone(högtalare);
  }
}
```

4.

Gör den vidstående kopplingen på provkopplingsdäcket. Värden på resistorerna vid tryckknapparna är 10 kΩ och vid lysdioden 220 Ω.

Rita apparatens kretsschema i ditt häfte.



Studera koden nedan. Vad gör den?

Uppgift 1: Skriv och mata in koden till Arduino.

```
// TONE ljudproducering 2

// konstanter definieras

const int knapp1 = 1;     // knapp 1
const int knapp2 = 2;     // knapp 2
const int högtalare =  11;      // högtalaren
const int led=  10;   // lysdioden

// variabler definieras för både knapparna
// och startvärden (0 = LOW) substitueras

int knappensTillstånd1 = 0;        // en variabel i stället för knapp 1
int knappensTillstånd2 = 0;        // en variabel i stället för knapp 2



void setup() {

  // Output-ledningarna sätts
```

```
  pinMode(högtalare, OUTPUT);
  pinMode(led, OUTPUT);

  // output-ledningarna sätts:

  pinMode(knapp1, INPUT);
  pinMode(knapp2, INPUT);
}

void loop(){
  // knapparnas tillstånd läses:
  knappensTillstånd1 = digitalRead(knapp1);
  knappensTillstånd2 = digitalRead(knapp2);

// ifall en av knapparna trycks, blir
  // den motsvarande variabelns tillstånd HIGH:

  if (knappensTillstånd1 == HIGH) {

    // ett ljud på 300 Hz produceras och lysdioden släcks:

    digitalWrite(led, LOW);
    tone(högtalare tin, 300);
  }

  else if (knappensTillstånd2 == HIGH)
  {
    // ett ljud på 450 Hz produceras och lysdioden släcks:

    digitalWrite(led, LOW);
    tone(högtalare, 450);
  }

  else
  {
    //Om knapparna inte trycks är deras tillstånd LOW
    // i detta fall tänds lysdioden och högtalaren dämpas

    noTone(högtalare);
    digitalWrite(led, HIGH);
  }
}
```

Uppgift: Förändra programmet på så sätt att ett annorlunda ljud produceras när båda knapparna trycks samtidigt. Tips: när du trycker knapparna ner samtidigt tillägg en jämförelse av båda knapparnas tillstånd till if-satsen.

Mätning av temperatur

TMP36 är en mikrokrets för temperaturmätning med tre anslutningar. Mikrokretsens yttre anslutningar kopplas till Arduinos +5 V- och GND-ledningar (i bilden polerna 1 och 3). Output-polen i mitten (polen nummer 2) kopplas till Arduinos analoga input-ledning. Spänningen i mikrokretsens output-pol beror på temperaturen linjärt 10 mV/°C. Temperaturen 0°C orsakar 500mV (0,5 V) spänning. Dvs. 20°C orsakar 700 mV (0,7 V) spänning. Tillverkaren garanterar sensorns exakthet vid mätning för temperaturintervallen -40…+125 °C.

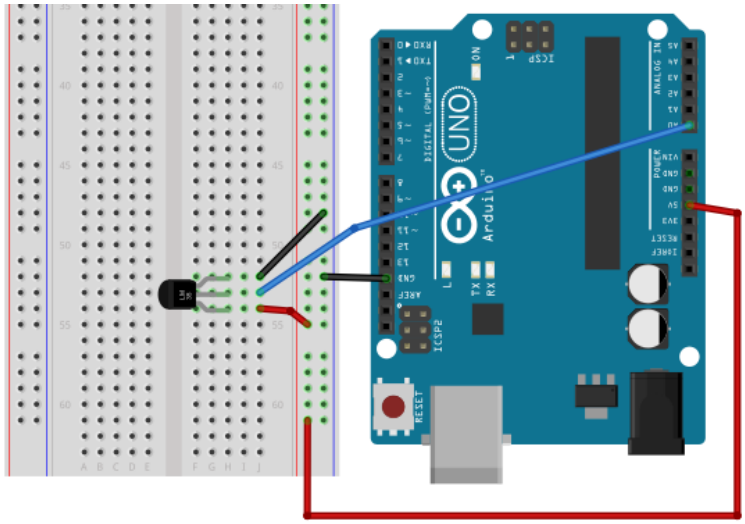
Man kan lätt koppla temperatursensorn till Arduino. Kom ihåg, att den analoga inputen fungerar på så sätt att spänning 0…5 volt omvandlas till ett heltal 0…1023.

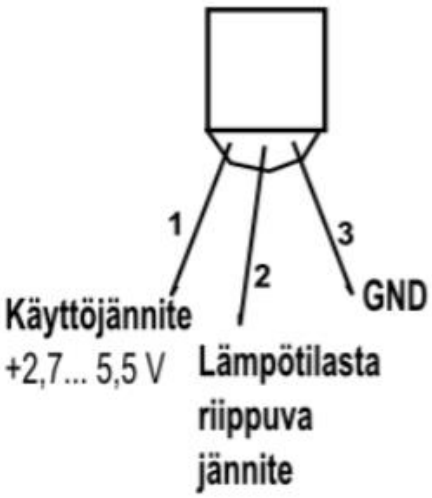
Du kan räkna spänningen från värdet i den analoga ledningen:

$$Spänningen = \frac{Värdet}{1023} * 5V$$

Från spänningen fås sen temperaturen

$$Temperaturen = (Spänningen - 0{,}5) * 100$$





Uppgifter

1. Samla ihop den vidstående kopplingen på provkopplingsdäcket. **Se absolut till att temperatursensorn är rättvänt (+5 V och GND rätt).** Annars kan sensorkretsen överhettas och gå sönder.
2. Rita kopplingsscheman i ditt häfte.
3. Pröva att mäta temperaturen med hjälp av exempelprogrammet. Värdet syns på skärmen i Arduinos seriemonitor.

```
//TMP36 prövning av temperatursensorn

int sensornLedning = 0;  // Sensorn är kopplad till den analoga ledningen nummer 0

void setup()
{
   Serial.begin(9600);  // en förbindelse till serieporten öppnas så att vi kan se värdet på skärmen
}

void loop()
{
    int lukema = analogRead(sensornLedning);              // värdet i den analoga ledningen läses
       float jannite = lukema * 5.0;
       jannite /= 1024.0;                          // värdet omvandlas till spänning

    Serial.print(jannite); Serial.println("volt");              // spänningen utskrivs

    float lampotila = (jannite - 0.5) * 100 ;  // spänningen omvandlas till temperatur

   Serial.print(lampotila); Serial.println("grader");          // temperaturen utskrivs

   delay(1000);                          // vänta i en sekund
}
```

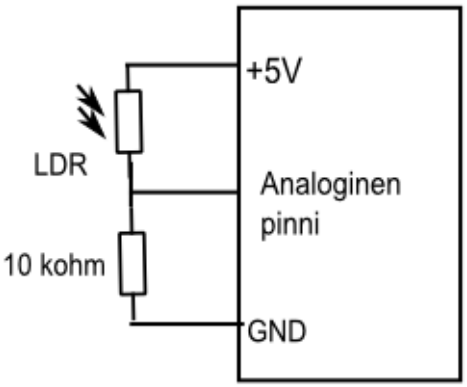
Extrauppgifter:

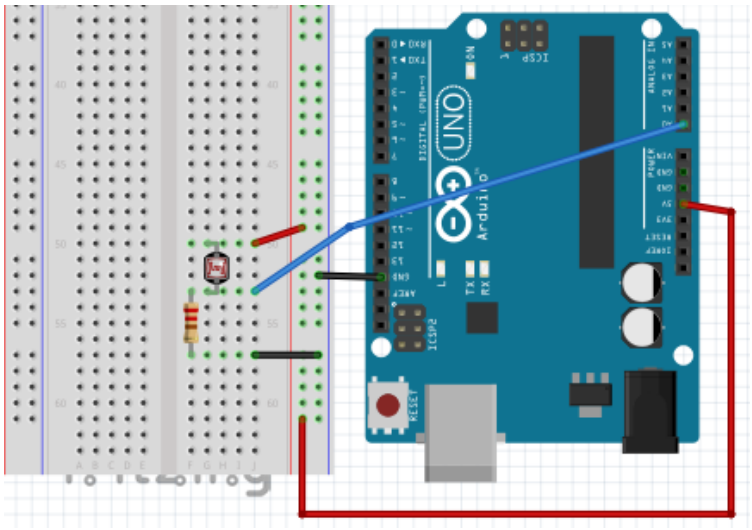
4. Tillägg en piezoelektrisk högtalare eller en lysdiod med en 220 ohm resistor på kopplingsdäcket. Bearbeta programmet så att du får en larmapparat för temperaturen. Larmapparaten meddelar att en viss temperatur är nådd genom att till exempel producera ett larmljud eller genom att blinka lysdioden.
5. Förändra kopplingen och programmet så att du kan använda potentiometern för att sätta den önskade larmtemperaturen.
   Tillägg potentiometern mellan polerna +5 V och GND så att du läser spänningen i potentiometerns glidkontakt med en av Arduinos digitala ledningar. Anteckna de sakerna som du behöver för programmet i häftet och planera i förväg.
6. Hitta TMP36-mikrokretsens uppgifter (TMP36 datasheet) på Internet och bekanta dig med mikrokretsens egenskaper.

Mätning av belysning

1.
En koppling till en spänningsdelare kan användas till exempel med en LDR-resistor (LDR = Light Dependent Resistor).

Vad gör det vidstående programmet? Jämför med den förra potentiometerns läsningsprogram.





```
// Belysningsmätning med en LDR-resistor.

int led = 13;  // Lysdioden (ledning nummer 13) som finns färdigt på Arduinokortet

void setup() {
  pinMode(led, OUTPUT); // ledning nummer 13 sätts som en OUTPUT-ledning
}

void loop() {
 int aika = analogRead(A0);  // spänning i den analoga ledningen A0 läses
 digitalWrite(led, HIGH); // lysdioden tänds
 delay(aika);    // dröjsmålet, tiden är spänning som står på spänningsdelaren
 digitalWrite(led, LOW); // lysdioden släcks
 delay(aika);    // dröjsmålet, tiden är spänning som står på spänningsdelaren
}
```

2.
Skriv den vidstående koden och ladda den till Arduino. Med programmet kan du skriva ut värdet i serieporten (ett tal som är jämförbar med belysningen) till datorskärmen.

Om du har en belysningsmätare till reds kan du kalibrera Arduinokopplingen till en riktig belysningsmätare. Ta mall från temperaturens mätningsprogram.

```
// Spänningsläsning från den analoga ledningen

void setup() {
  Serial.begin(9600); // serieportens hastighet sätts till 9600 bitar per sekund
}

void loop() {
 // spänning i den analoga ledningen A0 läses
 int lukema = analogRead(A0);
 // värdet skrivs ut till serieporten
 Serial.println(lukema);
 delay(100);    // dröjsmålet för att läsa värden
}
```

3.

Tillägg en lysdiod och en 220 ohms resistor till en av de digitala output-ledningarna. Ta mall från lysdiodens blinkningsarbeten. Utvidga programmet så att du får en nattlampa. Då belysningen minskar tillräckligt tänds lysdioden.

Rita kopplingens kretsschema i häftet.

Använd programmet för belysningsmätaren som fanns i förra uppgiften som programbotten. Bearbeta programmet så att du kan tända och släcka lysdioden enligt belysningen. Vid behov ta mall från arbetet *Mätning av spänning* (punkt 3 i undersökningar).

Studera vilket skulle vara ett passande jämförelsevärde. Använd serieportens monitor som hjälp. Då programmet är färdigt kan du ”kommentera” bort programrader som hör till serieporten (tillägg tecknen // i början av programraden så hoppar raderna över då programmet kompileras).

Styrning av en servo med Arduino

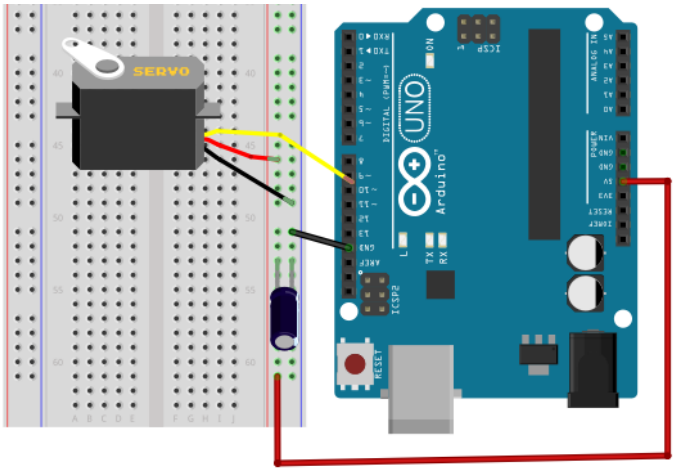
En elektrisk servo är en styrningsapparat som har utförts med en elmotor. Servomotorns axel kan roteras precist till en viss vinkel. Servos axel kan roteras till exempel 0...180 grader. Servomotorer används till exempel i fjärrkontrollerade apparater eller robotar för att styra rörelser.

Servomotorn har typiskt tre ledningar: ström, jord och signal. Ledningarnas ändar har vanligtvis ett uttag som man kan i provkopplingar använda till att koppla förbindningstrådar till Arduino.

Uppgifter:

1.

Koppla servos elledning (röd) till Arduinos +5 V-pol, jord (svart) till GND-pol och signalledning (gul) till en digital output-ledning som du kan själv välja (i exemplen ledningen nummer 9).



Koppla en 100 µF elektrolytkondensator mellan servos strömledningar (+5 V och GND). En kondensator är en komponent som lagrar el. Då servo används kan den ta en stor strömspik från Arduinos ledningar. Då kan ledningens spänning ”svikta” ner och orsaka en störning. El som har lagrats till kondensatorn hindrar detta. **Kom ihåg att koppla elektrolytkondensatorn rättvänt (+ och - poler rätt!).**

Ladda det vidstående programmet på Arduino. Programmet roterar servos axel först till utgångsställningen 0 graders vinkel och därefter till 90 och 120 graders vinklar.

Programmet utnyttjar ett färdigt servobibliotek som har färdiga kommandon och funktioner för att styra servon. Biblioteket bifogas till programmet med satsen *#include <servo.h>* i början av programmet.

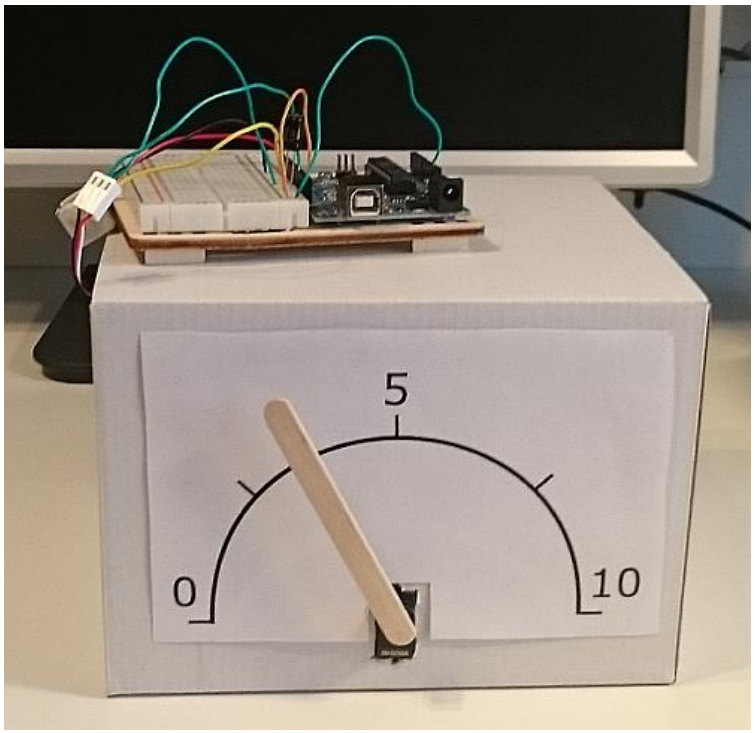
```
// Styrning av en servo

#include <Servo.h>          // Servo-biblioteket används

Servo minServo;             // Objektet minServo definieras

void setup() {
  minServo.attach(9);       // minServo kopplas till ledningen nummer 9
}

void loop() {
  minServo.write(0);        // Servon styrs till 0 graders vinkel
  delay(500);               // Vänta i 0,5 sekunder
  minServo.write(90);       // Servon styrs till 90 graders vinkel
  delay(500);               // Vänta i 0,5 sekunder
  minServo.write(120);      // Servon styrs till 120 graders vinkel
  delay(1000);              // Vänta i en sekund
}
```

2.
http://arduino.cc/en/Tutorial/Sweep (finns i programeditorns meny File → Examples → Servo)

Pröva det färdiga exemplet från Arduino-webbplatsen, *Sweep*, där servos axel sveper en 180 graders sektors område fram och tillbaka. I exemplet används en for-slinga. Studera slingans funktion på Arduinos instruktionssidor.

3.
http://arduino.cc/en/Tutorial/Knob (finns i programeditorns meny File → Examples → Servo)

Pröva exempelprogrammet *Knob*. I detta program justeras servos ställning med en potentiometer. Det lönar sig att fästa potentiometern till provkopplingsdäcket. I exemplet används map-funktion för omvandlingen 0…1023 (värdet som funktionen analogRead ger) → 0…180 (servos vridvinkel). Bekanta dig med map-funktionen på Arduinos webbplats.



Man kan göra till exempel en analog skärmpanel med hjälp av en servo. Här har man sysslat ihop en visarskärm av en servo, en papplåda och en glasspinne. Skalan kan man göra genom att rita direkt till panelen eller med ett grafikprogram på datorn.

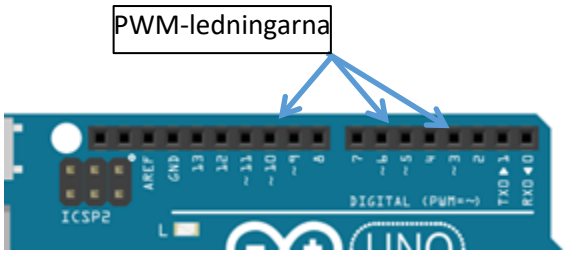
PWM Pulse Width Modulation

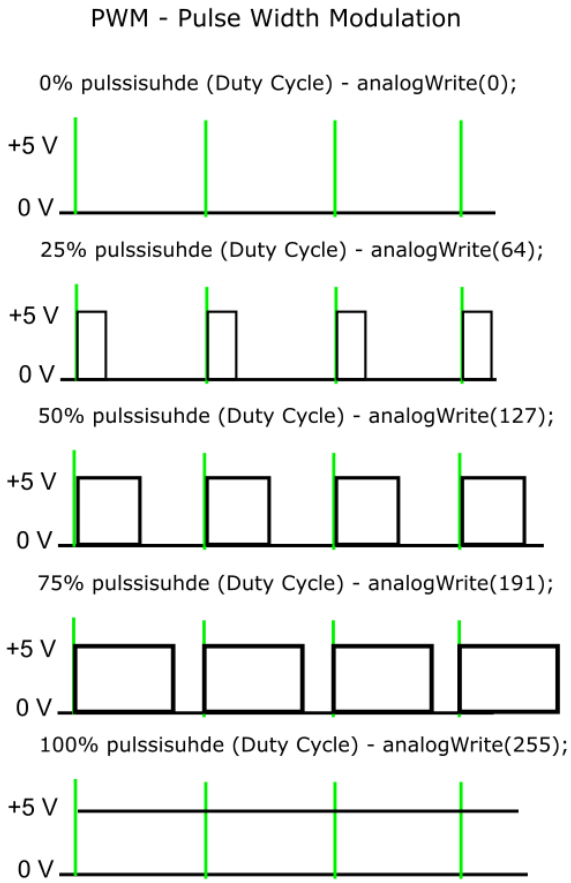
Spänning i den digitala output-ledningen är alltid 0 eller 5 volt. Ibland skulle det vara nyttigt att kunna justera till exempel lysdiodens ljusstyrka eller elmotorns rotationshastighet. Då kan man använda en PWM-signal (PWM = Pulse Width Modulation)

Kommandot *analogWrite* matar fyrkantsvåg till den digitala utpolen så att signalens pulskvot (Duty Cycle) kan förändras. Då pulskvoten är 50 % så är elektrisk effekt likaledes 50 % ”svagare” än kontinuerlig elström.

Pulskvoten 0% motsvarar 0 V spänning, 100% motsvarar +5 V spänning.

PWM-ledningarna har märkts på Arduinos kretskort med en vågsymbol ~. Arduino Unos digitala ledningar 3, 5, 6, 9, 10, 11 kan producera PWM-signal. PWM-signals frekvens är 490 Hz (i vissa ledningar 980 Hz)





Syntaxen: ***analogWrite(ledning, värde)***

Parametern ***värde*** är en heltalsvariabel och dess värde kan variera mellan 0…255. 0 motsvarar pulskvoten 0%, 255 motsvarar pulskvoten 100%.

T.ex.
analogWrite(10, 64) matar in signal med pulskvot 25 % till ledningen nummer 10.

Uppgifter:

Utrustning

- Arduino Uno, ett provkopplingsdäck och förbindningstrådar
- en 220 Ω, en lysdiod (flera vid behov)
- en 10 kΩ potentiometer
- en 10 kΩ resistor, en tryckknappskopplare (flera vid behov)

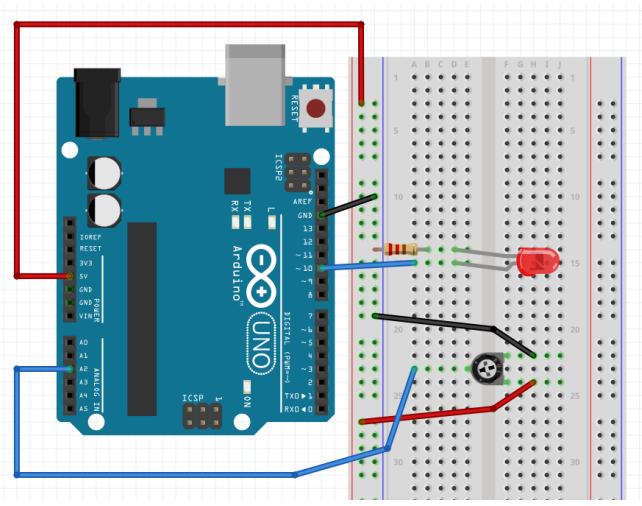
1.
Koppla en lysdiod och en 220 ohm resistor till Arduinos ledning nummer 10 (se till att ledningen har PWM-markeringen ~).

Ladda programmet till Arduino

```
int led = 10;

void setup() {
  // analogWrite kräver inte definiering pinMode
}

void loop() {

  analogWrite(led, 255);   // lysdioden lyser starkt
  delay(1000);             // vänta i en sekund
  analogWrite(led, 130);   // lysdioden blir mattare
  delay(1000);             // vänta i en sekund
  analogWrite(led, 20);    // lysdioden mattast
  delay(1000);             // vänta i en sekund

}
```

2.

Koppla potentiometerns kantpoler till +5 V- och GND-ledningar. Potentiometerns glid kopplas till den analoga ledningen A2.



Skriv ett program som du kan använda till att justera lysdiodens ljusstyrka steglöst med potentiometern. Öppna till exempel *AnalogReadSerial*-programexemplet och börja bearbeta därifrån (byt A0 → A2). Du kan också öppna serieportens monitor för att kunna granska värden på variabler.

Fundera:
Då man läser potentiometern funktionen *analogRead* ger värdet 0...1023.

PWM-ledningen kräver dock värdet 0...255.
Hur kan man omvandla 0…1023 → 0…255?

Ett av de möjliga programexemplen är det följande:

```
int Led = 10;      // Lysdioden kopplas till ledningen nummer 10
int Potikka = 2;   // Potentiometerns spänning läses från den analoga ledningen nummer 2 (0…5 V)
int arvo = 0;      // Variabeln, som sparar potentiometerns värde (ett heltal 0… 1023)

void setup()
{
}

void loop()
{
  arvo = analogRead(Potikka);   // Potentiometerns värde läses
  analogWrite(Led, arvo/4);  // Potentiometerns värde 0… 1023 omvandlas till att passa kommanddot
analogWrite 0…255
}
```

3.
Tillägg en lysdiod och en 220 ohm resistor till en av PWM-ledningarna (till exempel ledningen nummer 11). Bearbeta programmet från punkt 2 på så sätt att lysdiodernas ljusstyrka förändras omvänt, dvs. då den ena lyser klart lyser den andra inte. Genom att justera potentiometern blir den ena mattare och den andra ljusare. Fundera: Hur får du 0…255 → 255…0?

4.
For-slingan
Dämpning av en lysdiod med hjälp av en for-slinga

Bevara lysdiodkopplingen från det förra exemplet och pröva programmet:

```
// Dämpning av en lysdiod med hjälp av en FOR-slinga
int Led = 10; // värdet på variabeln Led sätts till 10

void setup()
{
}

void loop()
{
  for (int i=0; i <= 255; i++) {
    analogWrite(Led, i);
    delay(10);
  }
}
```

Ta reda på hur programmet fungerar. Pröva vad som händer då du byter kommandot till **analogWrite(Led, 255-i);**

5.

Tillägg en tryckknapp och en resistor till kopplingen. Använd de tidigare forskningsuppgifterna som modell.

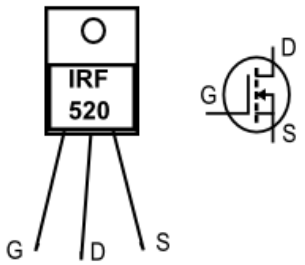
Bearbeta programmet så att du får lysdioden att bli mattare/ljusare genom att trycka på på tryckknappen..

Du kan också tillägga flera kopplare och lysdioder till Arduinos poler och utarbeta funktioner för dem. Koppla till exempel två tryckknappar så att lysdioden blir ljusare då man trycker på den första och mattare då man trycker på den andra.

Motorstyrning med Arduino

Utrustning

- Arduino Uno, ett provkopplingsdäck och förbindningstrådar
- MOSFET IRF520
- en 10 kΩ potentiometer
- en diod (t.ex. 1N4005)
- 9 V batteri och en batterikontakt (eller en motsvarande strömkälla)



Ledningsordning för transistorn IRF520

1. Gör den vidstående kopplingen, skriv programmet och testa hur det fungerar.

En diod har kopplats bredvid motorn. Diodens uppgift är att kortsluta eventuella strömspikar som motorn orsakar och som kunde skada Arduino.

2. Om du använder Arduino Starter kit, lägg dess CD-rullare av plywood på motorns axel. Starter Kit -serien har också en färgad papplatta som kan rullas med hjälp av motorn. Om du inte har en Starter kit lägg till exempel en maskeringstejpsbit på motorns axel så ser du hur motorn rullar.

3. Pröva att rulla motorn först endast programstyrt, utan att justera hastighet med potentiometern. Upprepa principen av PWM-justering från sida 40. Skriv koden och mata in den till Arduino.

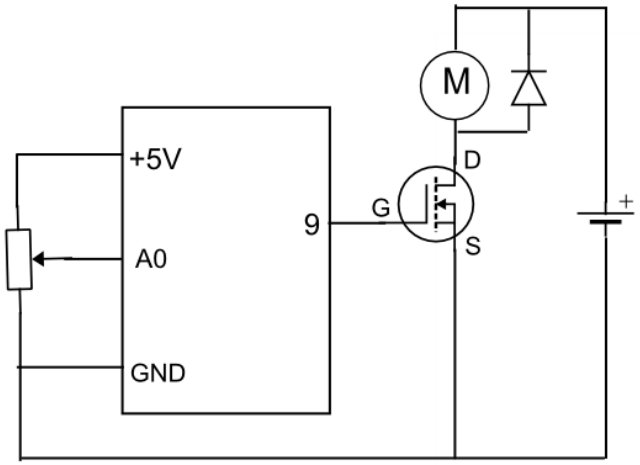
```
// Programstyrt motorhastighetsjustering

 const int transistoriPin = 9;  // MOSFETs styrningsledning
 int nopeus = 64;               // hastigheten justeras

 void setup() {
  pinMode(transistoriPin, OUTPUT); // en OUTPUT-ledning sätts
}

 void loop() {
  analogWrite(transistoriPin, nopeus);
    // PWM-styrning skrivs till transistorn
}
```

Pröva värden 0...255 för variabeln **nopeus**.

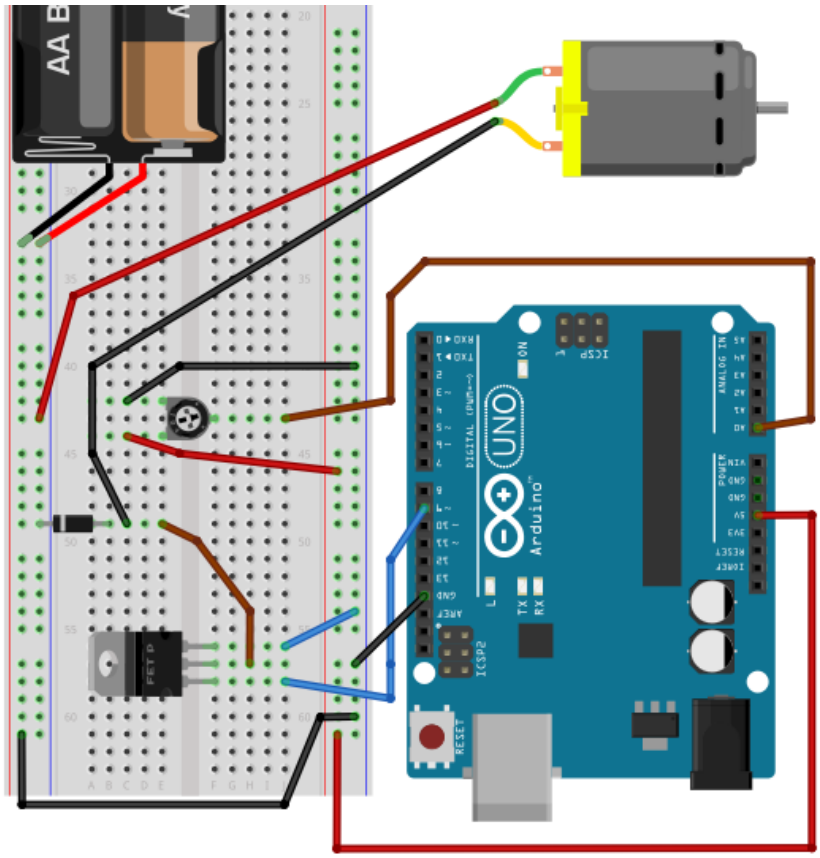
4. Studera motorjustering med potentiometern. Förändra koden så att Arduinos analoga ledning läser spänning i potentiometerns glidkontakt. Spänningen kan justeras mellan 0… 5 volt.

```
// Motorhastighetsjustering

const int potikkaPin = 0; // Potentiometerns glid till den analoga ledningen nummer 0
const int transistoriPin = 9;  // MOSFETs styrningsledning
int potikka;   // Värdet som returnerades från potentiometern

 void setup() {
  pinMode(transistoriPin, OUTPUT); // En OUTPUT-ledning sätts
}

void loop() {
  potikka = analogRead(potikkaPin)/4;
    // Potentiometern läses och värdet omvandlas samtidigt för att passa för PWM-
    // funktionen
  analogWrite(transistoriPin, potikka);
    // PWM-styrning skrivs för transistorn
}
```

Du kan pröva kopplingen också för att styra en lampa. En 6 volts lampa som används till att studera optikens fenomen passar bra till kopplingen. Lampan kopplas i stället för motorn. Med lampan kan man lämna skyddsdioden bort.



En extrern strömkälla har kopplats till motorn, till exempel en 9 V transistorbatteri. Motorn fungerar alltså inte med ”Arduinos elström”. Om du styr en lampa med hög ström, som man använder i optikens undersökningar, använd en elevströmkälla som lampans strömkälla. Se då till att transistorn inte överhettas.



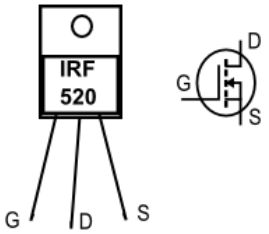
Uppgifter:

1. Varför divideras värdet som lästes från potentiometern med fyra?

2. Transistorn styrs med en PWM-signal. Vad betyder detta?

Den maximala elströmmen som Arduinos ledning kan ge är ca. 40 mA. Denna ström räcker inte för högeffektiva apparater som en elmotor eller högeffektiva lampor.

Arduino kan ändå kopplas med en extern komponent som för sin del justerar strömmen i motorn. Man kan använda till exempel en transistor som en styrningskomponent. I exempelkopplingen används en FET-transistor som inte behöver andra komponenter omkring sig.



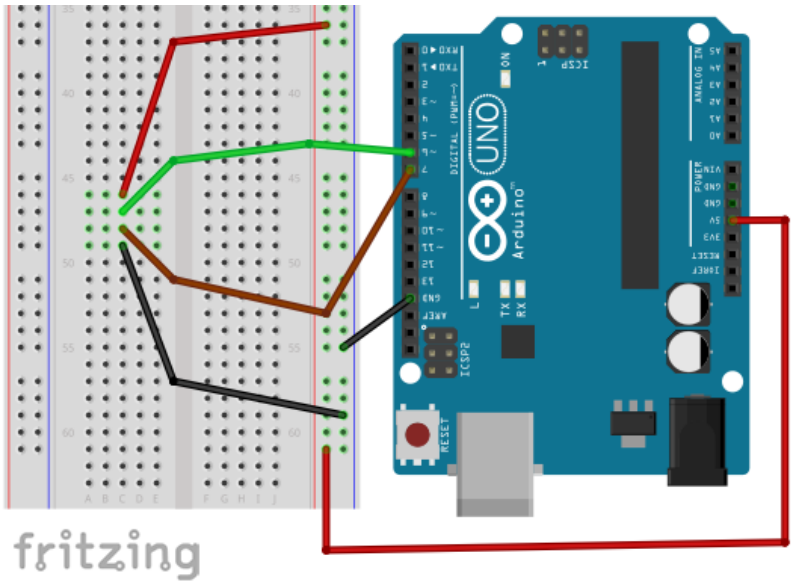
IRF520 är en MOSFET som har tillverkats för kopplarändamål. Dess ledningsordning finns i den vidstående bilden (G = Gate, gitter, D = Drain, S = Source).

Man kan jämföra en transistor med en extremt snabb kopplare som har inga rörliga delar. Med gitters (G) spänning kan man påverka elströmmen som går mellan polerna D och S. Då spänning inte finns, går där ingen ström, och på motsvarande sätt då gittren har spänning så går strömmen.

Att koppla en ultraljudssensor till Arduino

Utrustning

- Arduino Uno, ett provkopplingsdäck och förbindningstrådar
- En ultraljudssensor HC-SR04 (sensorn mäter avstånd mellan 2…400 cm)
- Komponenter för larmapparaten (t.ex. en piezoelektrisk högtalare, en lysdiod + 220 Ω resistor etc.)



Uppgifter:

1.

Gör den vidstående kopplingen på provkopplingsdäcket.

Sensorn har fyra kopplingsledningar: *+5 V, GND, Trig* och *Echo*. Driftsspänningen kopplas till ledningarna +5 V och GND. Kopplingsavtryckare kopplas till ledningen *Trig* (*Trigger = avtryckare*). Från ledningen *Echo* (*Echo = eko*) fås en puls som är jämförbar med avståndet till ett föremål.

Trigger kopplas till den digitala ledningen nummer 6, *Echo* till ledningen nummer 7.

2.

Programmet använder ett bibliotek med färdiga kommandon och funktioner för att styra en ultraljudssensor. Detta bibliotek finns inte färdiginstallerat i Arduinomiljön.

Biblioteket behövs alltså installera separat.

Hämta biblioteket från http://playground.arduino.cc/Code/NewPing --> Download.

Du kan installera biblioteket manuellt genom att packa upp mappen till Arduinomiljöns mapp Libraries. Stäng editorprogrammet och starta om. Programmet hittar det nya biblioteket och det finns till reds för Arduino-programmen.

3.

Skriv det vidstående programmet till editorn.

Biblioteket bifogas till programmet med satsen *#include <NewPing.h>* i början av programmet. Mätresultaten läses från serieportens monitor på datorn.

I exempelprogrammet har variabler definierats med #define-satser.

```
// Ultraljussensor

#include <NewPing.h>          // biblioteket NewPing.h tas i bruk
NewPing sonar(6, 7, 200); // NewPing formateras
// ledningen 6 trigger output, ledningen 7 echo input, maximalavstånd 200 cm

void setup() {
  Serial.begin(9600);                    // Serieporten öppnas
}

void loop() {
  delay(1000);                           // Avståndsmätning varje sekund
  int etaisyys = sonar.ping_cm();        // Avstånd mäts i centimeter
  Serial.print("Etäisyys: ");            // Skrivs ut på skärmen
  Serial.print(etaisyys);
  Serial.println(" cm");
}
```

4.

Granska sensorns exakthet vid mätning med hjälp av ett metermått. Lägg måttet så att nollpunkten är vid sensorn och att reflektionerna sker från någon hård yta, till exempel en bok. Är mätresultaten korrekta?

Bekanta dig med bibliotekets dokumentation på http://playground.arduino.cc/Code/NewPing. Där används uttrycket *metod* för bibliotekets kommandon och funktioner.

5.

Utvidga kopplingen så att du får någon fungerande apparat, till exempel en larmapparat med hjälp av en lysdiod eller en summer. Du kan göra en backningsradar för en moped!

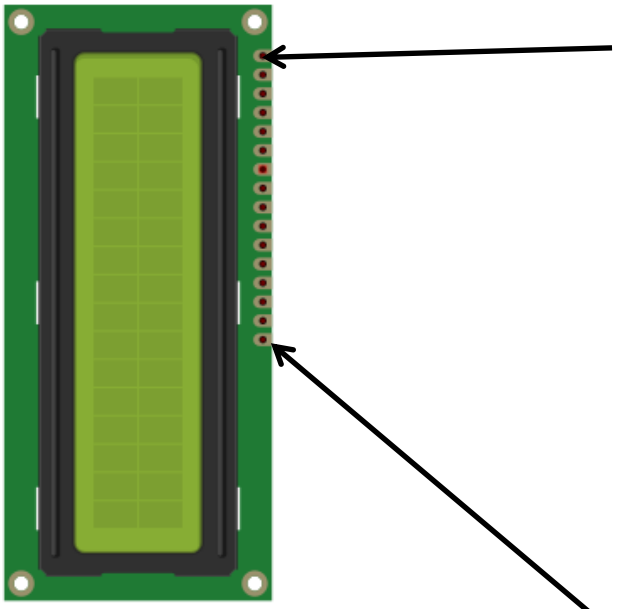
Obs.: Arduinos kommando *tone* fungerar inte med NewPing-biblioteket. Hitta biblioteket NewTone med hjälp av sökningstjänsten för att få ljussignalen att fungera.

Koppling av en LCD-skärm till Arduino

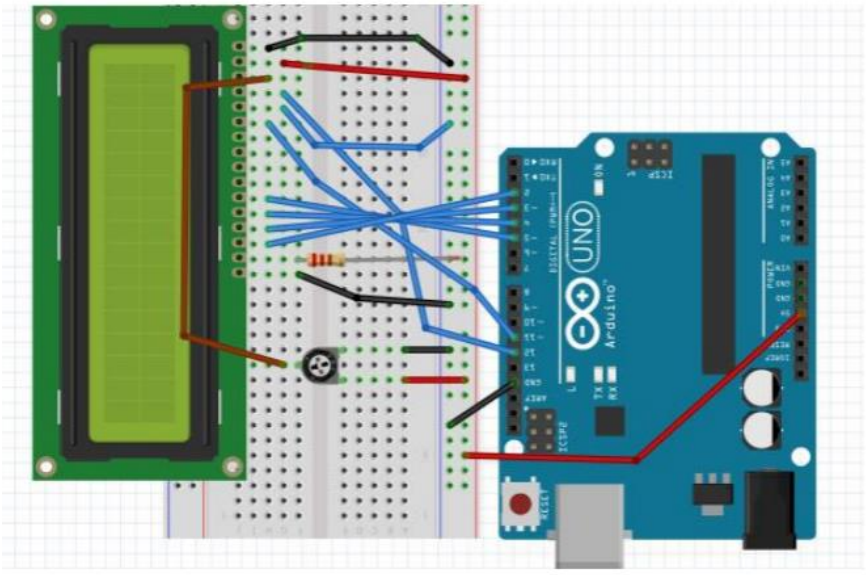
Utrustning

- Arduino Uno, ett provkopplingsdäck och förbindningstrådar
- en LCD-skärm (LCD står för *Liquid Crystal Display* dvs. flytkristallskärm)
- en 10 k Ω potentiometer, en 220 Ω resistor, en temperaturssensor TMP36

Man kan koppla en LCD-skärm till Arduino. Skärmen har 16 anslutningsledningar skärmen kan styras med.



LCD-ledning	kopplas till
1 VSS	Arduino GND
2 VDD	Arduino +5 V
3 kontrast	Potentiometerns glid
4 Register Select (RS)	Arduinos ledning nummer 12
5 Read/Write (RW)	GND
6 Enable	Arduinos ledning nummer 11
7 Data 0	Kopplas inte
8 Data 1	Kopplas inte
9 Data 2	Kopplas inte
10 Data 3	Kopplas inte
11 Data 4	Arduinos ledning nummer 5
12 Data 5	Arduinos ledning nummer 4
13 Data 6	Arduinos ledning nummer 3
14 Data 7	Arduinos ledning nummer 2
15 Bakgrundsljus +	220 ohm resistor +5 V
16 Bakgrundsljus GND	Arduino GND



1.
Koppla LCS-skärmen noggrant. Börja från en av kopplingsradens ändor och fortsätt en koppling åt gången. Då USB-sladden kopplas till Arduino skulle bakgrundsljuset tändas till skärmen. Man justerar skärmens kontrast med potentiometern. Om ingenting syns i skärmen, granska potentiometerns kopplingsledningar.

För LCD-skärmen ska biblioteket **LiquidCrystal** laddas till programmet. Biblioteket finns redan installerat som standard med Arduinos editorprogram.

Pröva programstycket:

```
// Prövning av LCD-skärmen

// LCD-biblioteket tas i bruk
#include <LiquidCrystal.h>

// Arduinos ledningar som programmet kan använda definieras
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // Antalet av LCD-skärmens rader och kolumner sätts
  lcd.begin(16, 2);
}

void loop() {
  lcd.print("hello, world!"); // skriv ut text på skärmen

// tal från noll till tio skrivs ut på skärmens andra rad
for (int i=0; i<=10;i++) {
   lcd.setCursor(0, 1); // markören sätts till början av andra raden (numrering börjar från noll)
  lcd.print(i); // värdet på variabeln i skrivs ut
  delay(500); // vänta i 0,5 sekund
   }

lcd.clear(); // skärmen töms

}
```

2.
Bekanta dig med *LiquidCrystal*-bibliotekets dokumentation (*https://www.arduino.cc/en/Reference/LiquidCrystal*) och pröva de färdiga övningsprogrammen.

3.
Studera *metoder* i biblioteket. Ta reda på vad de är.

4.
Pröva att tillägga någon sensor (till exempel en temperaturssensor) till kopplingen och studera hur du kan få sensorns värde på LCD-skärmen.

Att skriva en egen funktion

Funktioner som upprepas kan man samla ihop behändigt till underprogram (funktioner). Funktionerna programmeras utanför det egentliga huvudprogrammet (loop-slingan), så att man undviker att skriva densamma programkoden om och om igen. Vart och ett block, underprogram, utför någon funktion, i exemplet vrider servon till en önskad ställning, ger en önskad ljudsignal eller går och läser sensorns mätvärde. Underprogram kallas också för procedurer, funktioner, metoder eller rutiner. I språket C, som Arduinos kod baserar sig på, används generellt benämningen funktion.

Studera de vidstående programmen och ladda dem på Arduino.

1.
Styrning av en servomotor:
Programmet används för att styra servon och programmets funktioner har placerats till underprogram. Ordet *void* i funktionsdefinieringen betyder att funktionen inte returnerar något värde till huvudprogrammet. Heltalsvariabler *kulma* och *aika* matas in till funktionen som variabler.

Kopplingen är densamma som i den tidigare servoundersökningen.

Pröva att förändra värden på variabler som ges till funktionen *kaannaServo*. (Obs.: vinkels värde måste vara mellan 0…180 grader. Tiden kan inte vara mycket snabb på grund av servos långsamhet.)

```
// Styrning av en servo med en egen funktion

#include <Servo.h>          // Servo-biblioteket används

Servo minServo;             // Objektet minServo definieras


void setup() {
  minServo.attach(9);       // minServo anslutas till ledningen nummer 9
}

// Funktionen kaannaServo definieras. Variabler kulma och aika ges till den.

void kaannaServo(int kulma, int aika) {
  minServo.write(kulma);    // Servo styrs till vinkeln som definierades i variabeln kulma
  delay(aika);              // Man väntar i tiden som definierades i variabeln aika
}

// Huvudprogrammet börjar

void loop() {
   kaannaServo(0, 1000);    // Servon vrids till den önskade vinkeln
   kaannaServo(45, 1000);   // och man väntar i den givna tiden
   kaannaServo(90, 2000);
   kaannaServo(135, 500);
   kaannaServo(180, 400);
}
```

1.

Att producera ljud

Det följande programmet producerar ett stigande ljud till Arduinos ledning nummer 11. Underprogrammet **nousu** har utarbetats så att ljudets begynnelse- och slutfrekvenser eller s.k. parametrar ges då programmet påkallas. En parameter betyder begynnelsevärden som ges till kommandot och enligt vilka funktionen utförs.

Koppla piezohögtalarens poler till Arduinos ledning nummer 9 och GND-ledningen och ladda det vidstående programmet.

```
void setup() { // inställningar behövs inte
}

void nousu(int alku, int loppu)          // underprogrammet, begynnelse- och slutfrekvens som
                                         // parametrar
{
  for (int taajuus=alku; taajuus < loppu; taajuus++) {
    tone(11, taajuus, 50);               // slingan rullas tills slutfrekvensen nås
    delay(50);
  }
}


void loop() {                            // huvudprogrammet
  nousu(100, 600);                       // funktionen nousu kallas, 100 Hz - - > 600 Hz
  nousu(300, 800);                       // funktionen nousu kallas
  nousu(500, 1000);                      // funktionen nousu kallas
}
```

2.

Tillägg en funktion med namnet **lasku**, som producerar ett fallande ljud, till programstammen. (ett tips för for-slingan: i-- betyder i = i - 1).

Pröva funktioner **nousu** och **lasku** efter varandra så att du kallar dem från huvudprogrammet. Fungerar det?

Bearbeta underprogrammen så att du tillägger en ledning dit högtalaren har kopplats i som en parameter som ges till funktionen. Fråga om råd vid behov.

3.

Gör en koppling och ett program där du ”simulerar” ett utryckningsfordon (”piipaa-bil”). Tillägg en lysdiod och en 220 ohm resistor (eller flera) till kopplingen så får du blinkljus.

Du kan arbeta programmet så att

a) hela programmet finns i huvudprogrammet (loop-slingan),

b) programmet har delats till underprogram som kallas från huvudprogrammet.

Pröva både sätt!

Att göra en egen apparat – projektarbete

Du kan lätt kombinera utkast som har presenterats i detta häfte. Som hjälp med planeringen kan du använda den nedanstående listan där funktioner som programmeras för Arduino beskrivs.

- att blinka en ljusdiod
- att koppla tryckknapp
- att producera ljud
- att mäta spänning (att mäta temperatur, ljud)
- styrning av servo
- styrning av motorn
- PWM-motorstyrning
- att mäta avstånd med ultraljudssensor
- att koppla en LCD-skärm till Arduino

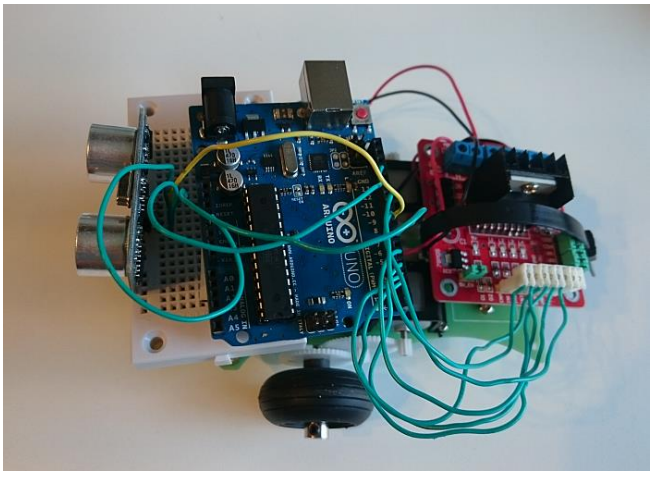
PROJEKTARBETE

- spel?
- mätanordning?
- alarm?
- markor? / räknare?
- timer?
- instrument? / spelare?
- ljuseffektanordning?
- något annat?

Du kan till exempel göra en termometer som har en LCD-skärm. Du kan tillägga en lysdiod eller en summer till termometern som alarm. Du kan också göra en termometer som har förslagsvis en visare (en servo + en visare) eller utföra skärmen i form av en lysdiodstaty. Termometern kan telegrafera värdet med morsealfabetet. Det finns många möjligheter!

Det lönar sig att leta efter idéer på Internet. I synnerhet från t. ex.

- Arduinos egna sidor www.arduino.cc
- Webbplatsen www.instructables.com (webbplatsen har mycket av apparater som har med Arduino att göra, sökord Arduino)
- Youtube (Arduino + saken som söks som sökord, t.ex. arduino led cube)

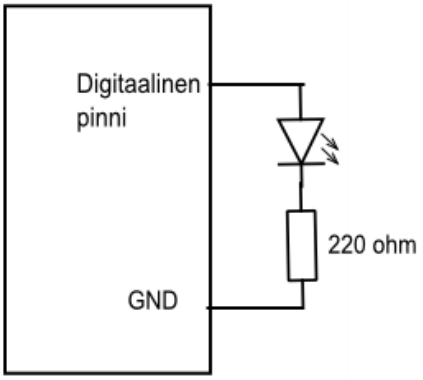


Det mekaniska utförandet av en apparat är ofta den mest utmanande delen av projektet. Det finns till exempel färdiga ramar för en rörlig robotbil på internets handelsplatser. En färdig ram avsevärt underlättar det mekaniska utförandet av en apparat. I slutet av häftet finns det instruktioner för utförandet av projektarbetet.

Repetition: baskopplingar

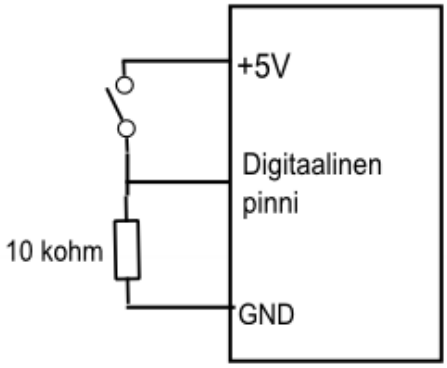
En lysdiod till en OUTPUT-ledning
T.ex.:

```
…
loop() {
digitalWrite(led, HIGH);
}
```



En tryckknapp till en INPUT-ledning
T.ex.:

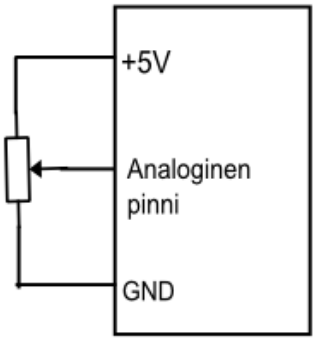
```
…
void loop(){
  knapp = digitalRead(knappPin);
  if (knapp == HIGH) {
     digitalWrite(ledPin, HIGH);
      }
  else {
   digitalWrite(ledPin, LOW);
    }
}
```

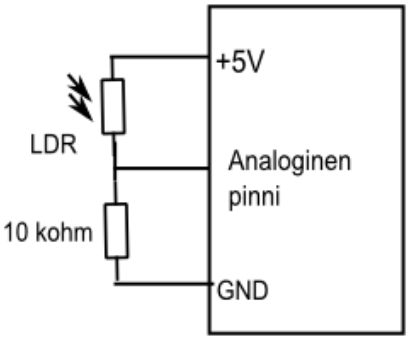


En LDR eller en potentiometer till den analoga input-ledningen

T.ex.:

```
…
void loop()
{
     int lukema = analogRead(sensornLedning);
     Serial.println(lukema);
    delay(1000);
}
```





Programmets styrningsstrukturer

Styrningsstrukturen	**Exemplet**

if
Ifall villkoret är sant utförs satsen.
 	
Jämförelseoperatorer
x == y (x är lika med y)
x != y (x är inte lika med y)
x < y (x är mindre än y)
x > y (x är större än y)
x <= y (x är mindre än eller lika med y)
x >= y (x är större än eller lika med y)
	if (variabel > 50)
	{
	// gör någonting
	}

if – else
 	
Ifall villkoret är sant så utförs funktionen A. I annat fall utförs funktionen B.
	if (variabel < 500)
	{
	// funktionen A
	}
	else
	{
	// funktionen B
	}

if – else if – else
 	
Man kan tillägga flera villkor till en if-sats.
 	
Det kan finnas flera else – if-block. Den sista else-satsen kan också lämnas bort.
	if (variabel < 500)
	{
	// funktionen A
	}
	else if (variabel >= 1000)
	{
	// funktionen B
	}
	else
	{
	// funktionen C
	}

switch / case
 	
Då variabelns värde stämmer med värdet på case-villkoret utförs alternativet.
En motsvarande struktur kan också göras med en if-sats, men en case-struktur är i många fall mer praktisk.
 	
Man avgår från case-funktionen med nyckelordet **break.**
	switch (variabel) {
	case 1:
	// funktionen A, då variabeln är 1
	break;
	case 2:
	// funktionen B, då variabeln är 2
	break;
	default:
	// ifall inget alternativ blir av
	// ”default” är valfri (kan lämnas bort)
	}

Upprepningssatser

Upprepningssatsen	**Exemplet**

for
 	
Upprepar funktionen det önskade antalet gånger
 	
for (formatering av variabeln; villkoret; variabelns ökning) {
// funktioner
}
 	
Obs:
Markeringen i++ betyder i = i +, dvs. att man adderar 1 till variabelns värde.
	int PWMPin = 9;
	
	for (int i=0; i <= 255; i++){
	  analogWrite(PWMpin, i);
	  // det skrivs PWM-signal
	  // till ledningen nummer 9
	  delay(10);
	}

while
 	
Upprepar funktionen tills det givna villkoret är osant.
 	
while(villkoret){
  // funktionerna
}
	variabel = 0;
	while(variabel < 200){
	  // funktion
	  variabel++;
	}
	// slingan avslutas då funktionen har
	// utförts 200 gånger

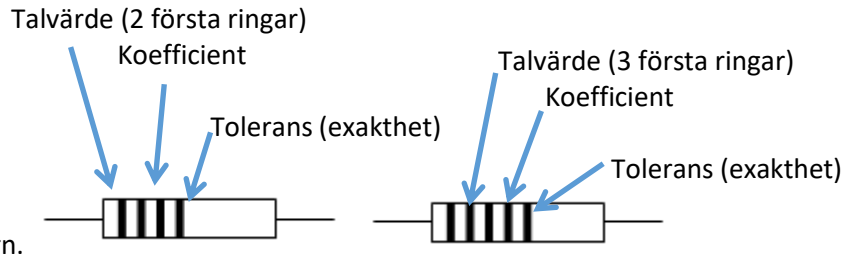
do - while
 	
Upprepar funktionen tills det givna villkoret är osant (på det samma sätt som **while**). Testning av villkoret finns i slutet av satsen så funktionen utförs åtminstone en gång.
 	
do
{
  // funktionerna
} while (villkoret);
	do
	{
	  delay(50);
	  x = analogRead(0); // den analoga
	  // ledningen läses till
	  // variabeln x
	} while (x < 100);
	  // slingan avslutas då det lästa
	  // värdet är under 100

Resistorernas färgkod

0 = svart
1 = brun
2 = röd
3 = orange
4 = gul
5 = grön
6 = blå
7 = lila
8 = grå
9 = vit

Tolerans (exakthet):

1 % = brun
2 % = röd
5 % = guld
10 % = silver



Det finns fyra eller fem färgringar runt resistorn.

1. och 2. ring (eller 1., 2. och 3.) betyder två första siffror av resistansens talvärde. 3. ring (eller 4.) är talvärdets koefficient (potens av 10, mängden av nollor) 4. ring (eller 5.) är resistorns tolerans eller exakthet.

T.ex.

brun - 1	svart - 0	röd – gul 2	R = 1000 Ω = 1 kΩ , ± 5 %
gul - 4	lila - 7	gul – guld 4	R = 470 000 Ω = 470 kΩ, ± 5 %
röd - 2	röd - 2	svart - silver 0	R = 22 Ω, ± 10 %

Resistorernas resistans

Läsning börjar från färgringen som ligger närmare resistorns ända. Resistorns resistans anges antingen med tre eller fyra ringar, den sista ringen uttrycker resistorns exakthet (tolerans). 5 % exakthet betyder att den egentliga resistansen i en 1000 Ω resistor kan ligga mellan 950 Ω – 1050 Ω.

Uppgifter

A.
Vad är de följande resistorernas resistans? Vad är deras exakthet?

1. gul - lila - röd - guld

2. brun - svart – blå - silver

3. orange - vit - orange - guld

4. röd - blå - gul - orange - brun

5. orange - orange - svart - silver

B.
Undersöka några resistorers färgringar och utreda vad som är resistorernas resistans. Mäta då resistansens värde med en digital universalmätare. Jämför resultaten: är resistorernas resistanser exakta (motsvarar den resistansen som mätts värdet som uttrycks med färgkoden)?

Projektarbetet

- Vad kunde apparaten göra?
    - Är det ett spel, ett mätinstrument, en larmapparat, en räknare, en timer, en ljuseffektsapparat eller någonting annat?
    - **Planera** apparatens funktion **på papper** och **anteckna** egenskaper som du önskar att den ska ha
    - Kom på ett namn för apparaten

- Gör upp en lista över delar som du behöver

- **Samla kopplingen** på kopplingsdäcket och **utarbeta ett program** för apparaturen.
    - Det lönar sig att börja utarbeta programmet från något färdigt exempelprogram. Om programmet till exempel läser temperatursensorn, börja med att ladda ner ett färdigt mallprogram och bearbeta det till den önskade formen.
    - Kom ihåg att du **inte behöver göra allt på en gång**. Du kan sätta ihop apparaten ett block åt gången och med detsamma granska funktionen. Så lär du dig också programmering och elektronik på det bästa sättet!
    - Ta **bilder** av arbetets framåtskridande.

- Utarbeta **en slutrapport** som en pdf-fil som du lämnar in enligt överenskommelse. Rapporten bör innehålla:
    - En skriftlig **beskrivning** av apparatens **funktion**, **bilder** av arbetets faser och den färdiga apparaten
    - **Ett kopplingsschema** (kan göras till exempel med något grafikprogram eller helt enkelt genom att skriva på pappret och fotografera/skanna)
    - **Programlistning**
    - **Reflexioner** om arbetet var lyckat, problem som upptäckts under arbetet, utvecklingsförslag för apparatens funktion

Övningsarbeten

Varje forskningsuppgift har grundläggande faktorer i början och därefter finns det fördjupade uppgifter och programmeringsövningar. Du kan anteckna uppgifter som du gjort i tabellen. Du kan också bedöma ditt arbete och anteckna din egen bedömning om hur du lyckades med ditt arbete.

Listans fetstilta forskningsuppgifter (att blinka en lysdiod, att läsa en tryckknapp, att mäta spänning) borde gås igenom noggrant. I dessa arbeten används Arduinos mest centrala egenskaper, den digitala ingången och utgången, läsning av analog spänning. Annars är uppgiftsordningen fri.

Forskningsarbete	**Grundläggande faktorer**	**Fördjupade arbeten**
Det första programmet BLINK		
Att blinka en ljusdiod, trafikljus		
RGB-lysdioden		
Programmets strukturer, FOR-slingan		
Att läsa en tryckknapp		
Att mäta spänning – den analoga inputen		
Spänningsdelaren – att läsa värdet i potentiometern		
Att producera ljud		
Mätning av temperatur		
Mätning av belysning		
Styrning av en servo med Arduino		
PWM Pulse Width Modulation: att dämpa en diod		
PWM: Motorstyrning med Arduino		
Att koppla en ultraljudssensor till Arduino		
Att koppla en LCD-skärm till Arduino		
Att skriva en egen funktion		
Resistorernas färgkod		

Utrustning och komponenter som behövs i undersökningar

Arduino Uno
Provkopplingsdäcket
Ett sortiment av förbindningstrådar (eller kopplingstråd som man kan klippa bitar av)
220 ohms resistor, 0,5 W
10 kohms resistor, 0,5 W
Tryckknapp, anslutande folieströmbrytare
10 kohm potentiometer (vid behov en justeringsaxel för potentiometern)
Miniatyrpiezohögtalare
LDR
100 uF, 16 V elko
LED röd
LED gul
LED grön
RGB LED gemensam katod
Temperaturssensor TMP36
MOSFET IRF530
Diod 1N4005
LCD-skärm
Liten elmotor (driftsspänning ca. 9 V)
Ultraljudssensormodul HC-SR04
Liten servo