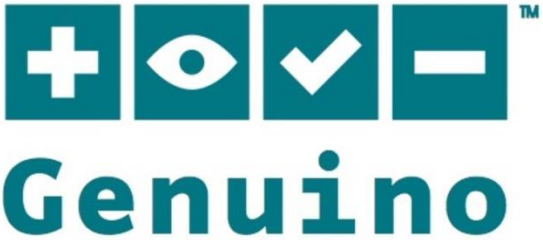
Programmering på Arduino/Genuino



Version 0.9b

Anders Berglund
@andersberglund_
anders.berglund@stockholm.se
https://about.me/andersberglund/

Mälarhöjdens skola
2016

Del 1

Varför programmera?

Ett kort svar skulle vara att vi gör det för att det är kul att lära sig nya saker men vi kan även motivera det i våra styrdokument som styr vad vi ska göra i skolan.
Till exempel i Läroplanens andra kapitel kan vi hitta det här:

Skolan ska ansvara för att varje elev efter genomgången grundskola

- *kan använda modern teknik som ett verktyg för kunskapssökande, kommunikation, skapande och lärande, och*
- *kan göra väl underbyggda val av fortsatt utbildning och yrkesinriktning.*

Och i kursplanen för teknik:

- *Undervisningen i ämnet teknik ska syfta till att eleverna utvecklar sitt tekniska kunnande och sin tekniska medvetenhet så att de kan orientera sig och agera i en teknikintensiv värld.*
- *Undervisningen ska bidra till att eleverna utvecklar intresse för teknik och förmåga att ta sig an tekniska utmaningar på ett medvetet och innovativt sätt.*

Vad är Arduino/Genuino?

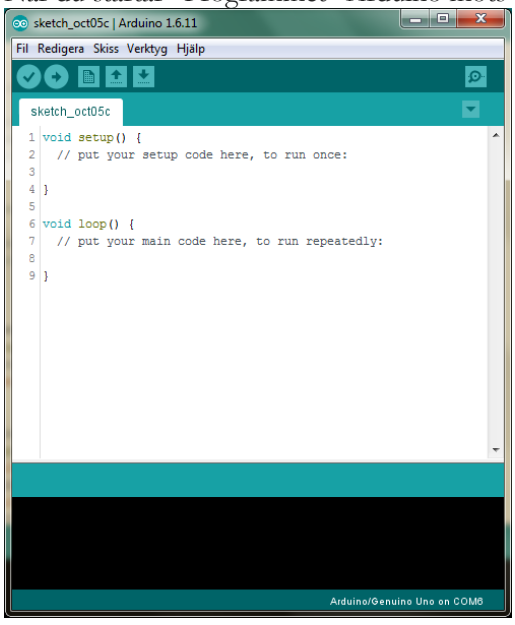
Enkelt förklarat är det en mycket enkel dator som styrs av en 8-bitars processor från tillverkaren Atmel. Detta är den stora avlånga komponenten på kretskortet. Det finns sedan ett antal hål längs två av sidorna där du kan koppla in saker som ska interagera med arduinon. Detta kan till exempel vara lysdioder (tänk små lampor så länge), knappar som ska styra vårt program, avståndsmätare, motorer, displayer, kretsar för att läsa och skriva kort liknande SL-kortet med mera.
Om jag förstått fakta på nätet rätt så har denna lilla dator ungefär 8 gånger så snabb processor och 8-16 gånger så mycket minne som den första månlandaren eller första Nintendo konsolen.
Det programmeringsspråk som används på Arduino är en lite förenklad version av C++ ett av världens mest använda programmeringsspråk. Egentligen ska vi i andra delar av världen än USA benämna Arduino som Genuino på grund av var de säljs men det är fortfarande vanligast att prata om Arduino. De är egentligen bara namnet som skiljer.

Installerat och klart?

Om allt fungerar som det ska kan du på skolans datorer bara kunna klicka längst ner till vänster på Windowsknappen och skriva arduino för att hitta programmet.
Vill du installera det hemma på din egen dator går du till websidan: www.arduino.cc och installerar det därifrån.
Du kan komma att behöva göra lite inställningar mer men det tar vi i nästa stycke.

IDE

(**I**ntegrated **D**evelopment **E**nvironment eller på Svenska Integrerad Utvecklingsmiljö)
När du startar ”Programmet” Arduino möts du av ett fönster som ser ut ungefär så här:



Det här i den här miljön som vi skapar våra program.
Längst upp har du en menyrad där du bland annat kan spara och göra inställningar.

Börja med att koppla in Arduinon med en USB sladen till datorn. Då ska det installeras lite drivrutiner så du behöver vänta ett tag.
För att kunna följa med i mina instruktioner ska du gå in under **Fil** -> **Inställningar** och kryssa i rutan ”Visa radnummer”.
Du kan också behöva gå in under **Verktyg** -> **Port** -> Och välja den **COM** som det står Arduino/Genuino Uno efter.
När du ska in på Verktyg kan det ta lite tid innan den blir aktiv så vänta bara innan den kommer fram.

All kod som finns i den vita rutan kommer sedan att göras om (kompileras)till körbar kod och föras över till arduinon när du trycker på **Pilen**. Om du bara vill kontrollera om programmet är skrivet på ett korrekt sätt trycker du på **Bocken.**

Hittills finns två delar i vårt program:

- void setup() – Där vi kommer att ställa in olika saker
- void loop() – Där vi kommer att skriva det som ska köras om och om och om igen.

Vi hoppar direkt in i ett exempel som vi använder för att förstå hur våra Arduinos programmeras.
Gå in under Fil -> Exempel -> 01.Basics -> Blink (som får en liten gul lysdiod märkt L att blinka)

För att hela programmet ska få plats på så har jag kopierat koden in det i ett annat program(notepad++) men du kan följa med på din dator.

```
1  /*
2    Blink
3    Turns on an LED on for one second, then off for one second, repeatedly.
4
5    Most Arduinos have an on-board LED you can control. On the Uno and
6    Leonardo, it is attached to digital pin 13. If you're unsure what
7    pin the on-board LED is connected to on your Arduino model, check
8    the documentation at http://www.arduino.cc
9
10   This example code is in the public domain.
11
12   modified 8 May 2014
13   by Scott Fitzgerald
14  */
15
16
17 // the setup function runs once when you press reset or power the board
18 void setup() {
19   // initialize digital pin 13 as an output.
20   pinMode(13, OUTPUT);
21 }
22
23 // the loop function runs over and over again forever
24 void loop() {
25   digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
26   delay(1000);              // wait for a second
27   digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
28   delay(1000);              // wait for a second
29 }
30
```

På nästa sida går jag igenom rad för rad vad programmet gör.

```
Rad 1 :    /*
```
Detta betyder att det kommer en kommentar, alltså något som inte ska köras eller förstås av datorn utan som är en hjälp för oss människor att förstå programmet.

```
Rad 2 – 13
```
Fortsättning på kommentaren om vad programmet gör och vem som gjort det.

```
Rad 14     */
```
Här slutar kommentaren

```
Rad 17     //
```
Även här en kommentar, denna fortsätter raden ut men inte på nästa rad

```
Rad 18     void setup() {
```
Här startar rutinen setup, den slutar med ”måsvingen” = } på rad 21

```
Rad 19     // initialize digital pin 13 as an output.
```
Kommentar till raden under (alltså rad 20)

```
Rad 20     pinMode(13, OUTPUT);
```
Ställer in en utgång (nummer 13) att användas som en utgång. Denna utgång är alltid den lilla gula lysdioden (tänk lampa så länge) på Arduinon.

```
Rad 21     }
```
Här slutar rutinen setup

```
Rad 23     // the loop function runs over and over again forever
```
Kommentar till raden under

```
Rad 24     void loop() {
```
Här börjar rutinen loop som kommer att köras om och om igen.

```
Rad 25     digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
```
Här slås strömmen på till utgång 13, det betyder att lysdioden tänds.
Som du ser avslutas också raden med en kommentar med att detta sker på engelska!

```
Rad 26     delay(1000);              // wait for a second
```
Datorn får instruktionen att vänta 1000 millisekunder = 1 sekund

```
Rad 27     digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
```
Här slås strömmen av till utgång 13, det betyder att lysdioden släcks.

```
Rad 28     delay(1000);              // wait for a second
```
Datorn får instruktionen att vänta 1000 millisekunder = 1 sekund

```
Rad 29     }
```
Här är rutinen loop slut och programmet hoppar tillbaka till rad 24 och kör igenom instruktionerna om och om igen.

Del 2 - Ändra i koden till blink

Nu är det dags för dig att ändra lite i koden som vi nyss gick igenom.
Ta för vana att spara dina program med nya namn som du sedan kan gå tillbaka och titta på, tänk också på att du bör kommentera din kod så att du eller någon annan kan förstå vad koden gör.

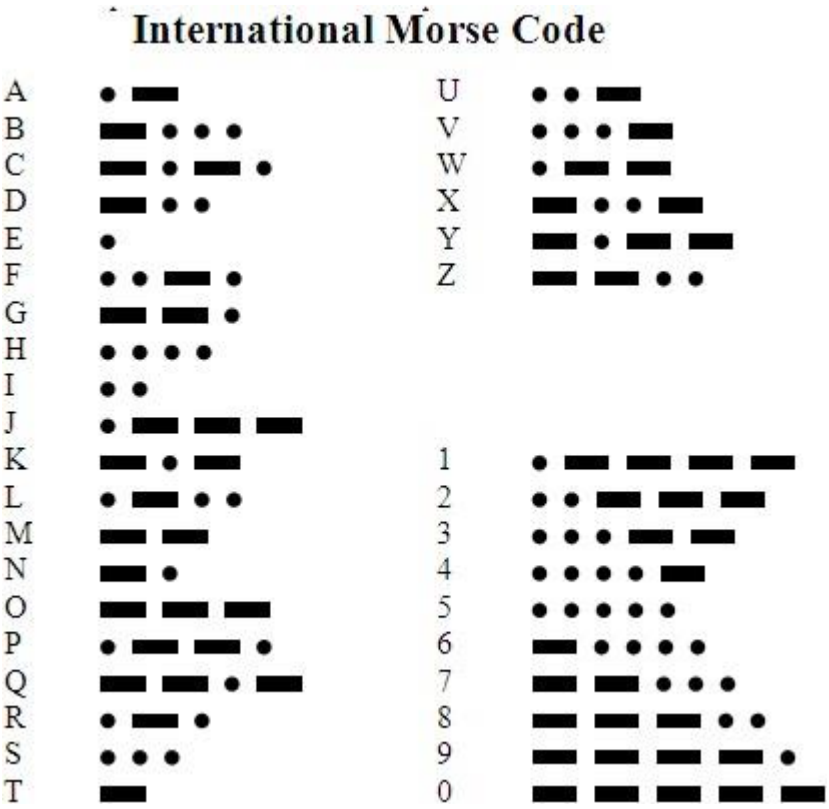
Blink1.1
Försök ändra så att lysdionden är tänd två sekunder och släckt en halv sekund.

Blink 1.2
Ändra så att lysdioden först är på två sekunder och av två sekunder detta ska den göra två gånger. Sedan ska den vara på en sekund och av en sekund detta ska den också göra två gånger

Blink 1.3
Morsealfabetetet är ett sätt att sända bokstäver med korta och långa ljud eller ljussignaler du har kanske hört talas om att signalen SOS betyder att någon behöver hjälp. På ”morsespråk” betyder det att tre korta signaler skickas först, sedan tre långa och sedan tre korta igen.
Försök nu skriva ett program som skickar detta med lysdioden. Du kan ha de långa delarna som tre sekunder, de korta som en sekund och mellanrummet mellan bokstäverna som två sekunder.

Om du vill kan du också få programmet att skicka ditt namn. Om du har å, ä eller ö i ditt namn så är koden för de bokstäverna Å = kort, lång, lång, kort, lång
Ä = kort, lång, kort, lång
Ö = lång, lång, lång, kort

Annars har du koderna här nedanför.



Del 3 – Text på datorn från programmet.

Ibland kan man vilja få något annat ut än en lysdiod som blinkar till exempel när man vill se var ett program går fel.
Börja med att ladda in exempelprogrammet blink igen.
Lägg sedan till de rader som jag markerat med gult här nedanför.

```
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  Serial.println("On");
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  Serial.println("Off");
  delay(1000);              // wait for a second
}
```

Vad betyder nu de nya radera? Jo
`Serial.begin(9600);`
Denna rad betyder att vi ska starta en kanal ut som skickas seriellt (det skickas ett tecken i taget) med hastigheten 9600 tecken per sekund.
Detta kommer sedan att skickas till något som heter ”**seriell monitor**”, den kan du hitta under verktyg, eller trykca på <ctrl>+<SHIFT>+M.

`Serial.println("On");`
Skicka texten ”On” till den seriella monitorn.

`Serial.println("Off");`
Skicka texten ”Off” till den seriella monitorn.

Detta kan du till exempel använda när du ska felsöka ditt program, debugga, som det också kallas.
Du kommer också att kunna använda serial när du vill kommunicera med en annan komponent så som en display eller när du vill läsa av ett kort liknande SLs.

I nästa exempel ska du också få se ett annat exempel på hur vi kan använda serial för att se vad som händer inne i ett program medan det körs.

Men först lite fakta om hur datorn håller reda på saker i sitt minne. Även om de flesta program som vi ska skriva inte kommer att bli så långa så kan det ändå vara bra att de inte tar mer plats i datorns minne än nödvändigt och för att vi ska kunna lagra något där så talar vi om hur mycket minne som det ska få plats. Vi kan till exempel tala om att vi ska lagra antingen bara om något är sant eller falskt det kan till exempel vara om vi vill lagra om vi är man eller kvinna. Vi väljer då att lagra det i en variabel av typen boolean.
Om vi vet att vi bara ska lagra små positiva siffror kan vi använda typen byte. Detta kan vi använda om vi vill kunna räkna upp till 255.
Om vi vill använda även de negativa siffrorna men bara kommer att räkna mellan -128 och 127 så kan du använda typen Char.
Om vi vill kunna räkna med större heltal så använder vi typen integer (som förkortas int när vi kodar) då kan vi lagra tal mellan -32768 och 32767.
Om du vill räkna med decimaltal så får du använda typen float, du kan då också lagra enormt små eller stora tal.
Å andra sidan tar denna typ upp fyra gånger så mycket plats i minnet som typen byte.
Denna tabell sammanfattar detta. (Det finns andra typer också)

Variabeltyp	**Storlek**	**Min - Max**
Boolean	8 bitar	sant/falskt eller 0/1
Byte	8 bitar	0-255
Char	8 bitar	-128 – 127
int	16 bitar	-32768 – 32767
float	32 bitar	mycket litet eller mycket stort

Del 4 – Blinka ett visst antal gånger (och lite räknande).

Utifrån programmet som du skapade i del 3 kan vi nu använda våra kunskaper om variabler och olika sorters sådana. Vi kommer att skapa en enkel räknare, hur man kontrollerar om något inträffat och få vårt program att stoppa. Om du tittar på programmet här nedanför så är de nya delarna markerat med gult.

```
/*
  Blink
  With text and counting breaking after ten blinks
 */

// Declarations
byte antal = 0;

void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

// the loop function runs over and over again forever
void loop() {
  antal = antal+1;
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
     if (antal == 10){
        Serial.println("TIO, nu orkar jag inte mer!");
        delay(30);
        exit(0);}
  Serial.print (antal);
  Serial.println(" times on");
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  Serial.println("Off");
  delay(1000);              // wait for a second
}
```

Vad betyder då dessa nya rader?

```
// Declarations
byte antal = 0;
```

Första raden har jag bara lagt till en kommentar för att tala om att jag ska deklarera mina variabler här. Du kan tänka dig en deklaration som en lapp som du skriver ner något på. Variabeltypen kommer först, sedan kommer vad det står på lappens ena sida med en bläckpenna och sist kommer vad som står på andra sidan av lappen skrivet med blyertspenna. Variabeltypen kan du tänka dig som hur stor lappen är, alltså hur stor plats den tar i minnet.
Här har vi alltså skapat en lapp som det står antal på ena sidan och en 0:a på andra sidan.

```
antal = antal+1;
```

Detta betyder att vi skriver om lappen med antal + 1. Det stod ju 0 på lappen från början så första gången vi kommer till den här raden så kommer det att stå 0+1 = 1 på lappen.

```
if (antal == 10){
```

Denna rad kontrollerar om det står 10 på lappen som det står antal på. Om det gör det så ska det som står inom måsvingarna göras. Om det inte står 10 på lappen så hoppas det över.

```
Serial.println("TIO, nu orkar jag inte mer!");
```

Borde inte vara så svårt att förstå.

```
delay(30);
```

Inte heller detta borde vara så svårt att förstå, men det verkar behövas en liten paus för att texten på raden ovanför att hinna skrivas ut innan nästa rad körs.

```
exit(0);
```

Stoppar programmet

```
Serial.print (antal);
```

Skriver ut det som står på lappen med namnet antal till serial-monitor.

När du kör programmet kommer det att sluta med att lysdioden är tänd.
Försök ändra programmet så att den är släckt när det slutar.

Om du lyckas med detta försök få lysdioden att blinka snabbare de fem första gångerna och långsammare de sista fem gångerna.

Du kan också prov att skriva ut andra saker i den seriella monitorn.
Skapa ett par andra variabler tal1, tal2, tal3 och tal4 med olika värden.
Låt sedan arduinon hjälpa dig med matematiken, till exempel genom att skriva koden:

```
Serial.print (“tal1 + tal2 = )”; // Denna rad skriver ut texten
Serial.println tal1 + tal2; // Denna rad skriver ut resultatet
```

Prova alla fyra räknesätten och håll speciellt koll på division.

Märkte du något konstigt? Om inte prova med några andra siffror vid divisionen.
Då borde du märka att divisionen bara fungerar om svaret blir ett heltal, till exempel verkar arduinon tro att tre delat med fyra är noll. Detta beror på att du (troligen) har valt en variabeltyp som är ett heltal (byte = små positiva heltal).
För att du ska kunna få svaret som ett decimaltal så måste de tal som du räknar med också vara det.
Du behöver alltså ändra så att du deklarerar dina variabler som float, till exempel:

```
float tal3 = 3;
```

Del 5 – Arrayer & Loopar

En variabel kan bara innehålla ett värde, men ibland kan det vara bra att kunna spara flera saker på något sätt.
Då kan vi använda något som heter **array**.
Du kan tänka dig en array som en tabell med en kolumn och flera rader där variabeln står som rubrik.
Till exempel.

Till exempel	Frukter	Exempel 2	Antal
	Plats 0		Plats 0
	Plats 1		Plats 1
	Plats 2		Plats 2
	Plats 3		Plats 3
	o.s.v		osv.

Ett exempel på detta skulle i koden skrivas:
```
char* frukt[] = {”Apelsin”, ”Banan”, ”Citron”, ”Druva”}; //Typ char = bokstäver
```

Och det andra exemplet skulle kunna se ut så här:
```
Int antal[] = {2,5,6,78}; // Typ int = heltal mellan -32768  &  32767
```

Vi kan skriva ut innehållet i en array genom att skriva koden:
```
Serial.println(frukt[0]);
Serial.println(frukt[1]);
Serial.println(frukt[2]);
Serial.println(frukt[3]);
```

Vi skulle då få ut resultatet
```
Apelsin
Banan
Citron
Druva
```

Men vi kan göra detta mycket snyggare! Vi använder då något som kallas en loop (i detta fall en FOR-loop).
```
/*
  Skriva ut en Array med räknare – FOR-loop
 */

// Deklarationer
char* frukter[]={"Apelsin", "Banan", "Citron", "Druva"}; // Typ är char
raknare = 0;

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

void loop() {
 Serial.println("Lets go!");
  for (raknare = 0; raknare < 4; raknare++) //Nyheter på denna rad!
  {
    Serial.println(frukter[raknare]); //nyheter här också
          delay (200);
  }
delay(10000); //Bara tillagd så att programmet ska stå och vila.
}
```

Vad betyder då de nya raderna?
Jo först talar vi om att variabeln raknare ska börja på noll, sedan kommer villkoret som talar om vad vi vill ska gälla så länge (i detta fall att raknare ska vara mindre än 3 och sist på den raden (raknare++) betyder att vi ska öka värdet på raknare med 1.
Så länge raknare är mindre än 4 så körs de som står inom de rödmarkerade måsvingarna.

När programmet kommer till ”serial.println-raden” första gången är alltså raknare 0 och därför skrivs det 0:te värdet på frukter ut, precis som när vi tidigare skrev `Serial.println(frukt[0]);`
Programmet hoppar sedan upp till for raden och lägger till 1 till raknare så att den nu är 1, skriver ut Serial.println(frukt[1]); och så vidare tills raknare blir 4 och alla våra frukter är utskrivna.

På nästa sida får du se att annat sätt att göra en loop.

Om du tittar på programmet här nedanför så ser du ett annat sätt att göra samma sak fast vi använder en annan typ av loop, en While-loop. Det går lika bra i detta fall och är mest en smaksak vilket sätt du vill använda.

```
/*
  Skriva ut en Array med ränkare och While-loop
*/

// Deklarationer
char* frukter[] = {"Apelsin", "Banan", "Citron", "Druva"}; // Typen är char alltså bokstäver
int raknare = 0;

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

void loop() {
  while (raknare > 3); {       //Kör det mellan måsvingarna så länge räknaren är mindre än 3
    Serial.println(frukter[raknare]);
    delay (300);
    raknare = raknare + 1;     // öka värdet på raknare med 1 varje gång (raknare++
  }
}
```

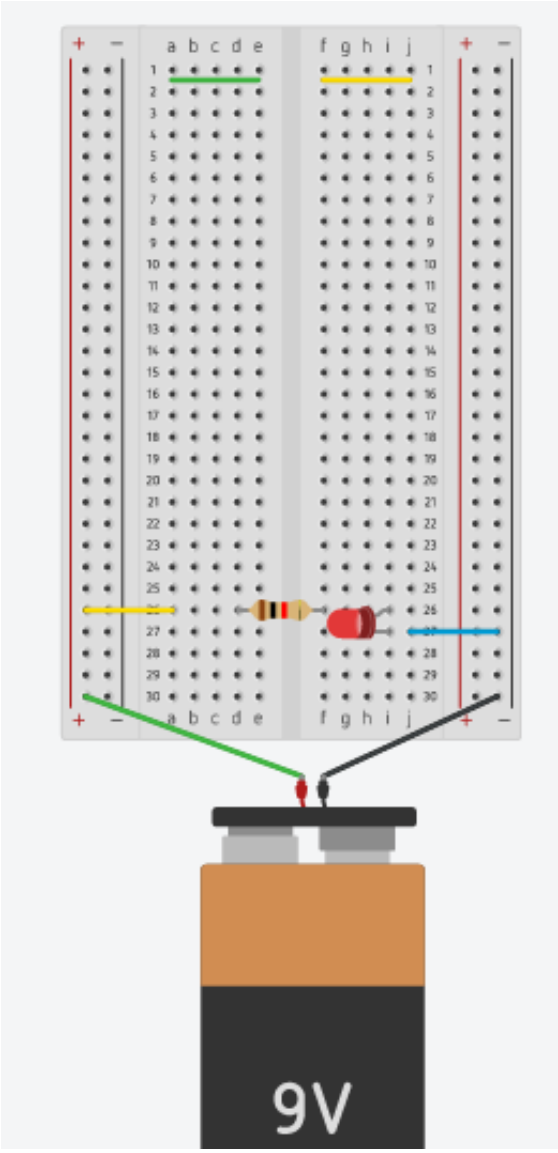
Försök nu skriva ut listan från Druva ner till Apelsin! Gärna både med **for** och **while loopar**.

Prova också att skapa två olika arrayer och skriv ut de på olika sätt.

Del 6 – Kopplingsdäck och lysdioder

Hittills har vi bara använt den inbyggda lysdioden och den seriella monitorn, men i början av kursen lovade jag ju att du skulle få arduinon att kunna interagera med verkligheten. Det är därför dags att titta på kopplingsdäcket och några av komponenterna som finns.

Vi använder ett så kallat kopplingsdäck för att koppla upp våra elektriska kretsar. Det är kopplat så att alla hål i kolumnenlängst till vänster(märkt +) sitter ihop under plasten, på samma sätt är alla i den näst längst till vänster (märkt -) ihopkopplade. Sedan är hålen 1a, 1b, 1c, 1d & 1e ihopkopplade som jag visat med en grön linje. Och 1f, 1g, 1h, 1i & 1j ihopkopplade. Sedan kommer två kollumner till där alla hål är kopplade som de två längst till vänster.



Som du förhoppningsvis kommer ihåg från någon fysikkurs så måste det vara en sluten krets för att något ska fungera. Och om man kopplar som jag visar i nedre delen av kopplingsdäcket så skulle lysdioden lysa.

Då skulle strömmen gå från den röda sidan på batteriet, genom den gröna sladden upp i kolumnen längst till vänster,
Genom den gula sladden, genom det som heter motstånd (eller resistor), genom den röda lysdioden, genom den blå sladden, ner i kolumnen längst till höger och sist tillbaka till batteriet i den svarta sladden.

Anledningen till att det behövs ett motstånd är att en lysdiod bara tål en viss ström igenom sig, annars går den sönder.
I vårt fall kan vi använda de motstånd som har färgerna RÖD, RÖD, BRUN, GULD som har motståndet 220 Ohm.
Motstånden med färgerna BRUN, SVART; RÖD, GULD har motståndet 1000 Ohm och motståndet med färgerna BRUN, SVART, ORANGE, GULD har motståndet 10000 Ohm.

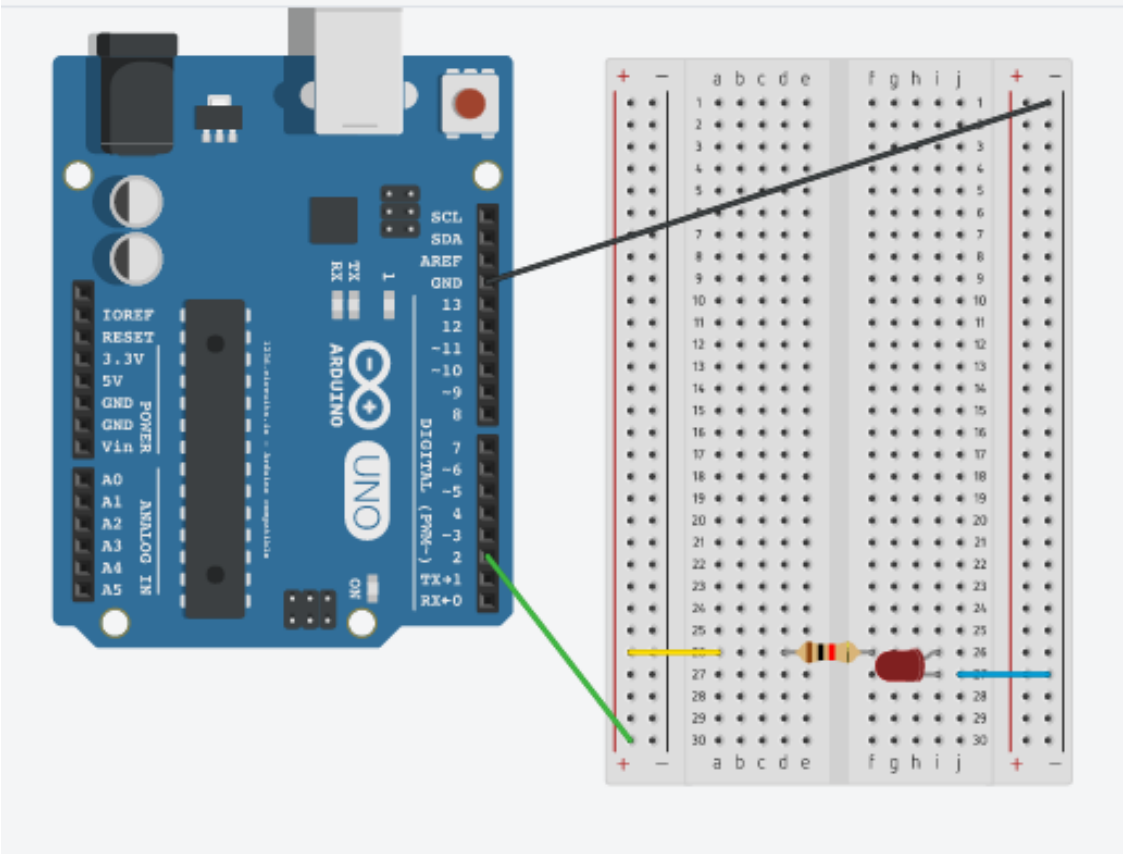
Anledningen till att jag valt ett annat på min bild här är att batteriet har spänningen 9 Volt, medan Arduinon normalt ger 5 Volt. Och jag behöver alltså använda ett motstånd med högre resistans för att inte för stor ström ska gå genom min lysdiod.

Lysdioden har ett ben som är lite längre än det andra, på min bild här ovanför så är det markerat med att det är lite böjt. Det är alltså det övre av benen. En lysdiod måste alltid kopplas så att strömmen går in i det längre benet (alltså plussidan) annars lyser den inte.

Varje ben som är numrerat 2 till 13 kan användas som antingen in eller utgångar från våra arduinos, de som har en liten ~ markering kan dessutom göra lite mer men det tar vi senare.
Vi kan alltså byta ut plussidan på batteriet med ett av de hålen på Arduinon.
Minussidan på batteriet byter vi sedan ut mot en av de hål där det står GND.

På nästa sida ser du hur den här kopplingen skulle kunna se ut.

Ladda nu in exempelprogrammet blink igen. Lägg sedan till tre rader kod så att inte bara den inbyggda lysdioden (13) blinkar utan även den som du satt din lysdiod på. På min bild är det nummer 2.



Om du lyckats koppla rätt så ska både den inbyggda lysdioden och den du satte på kopplingsdäcket blinka samtidigt.

Om det inte händer har du antingen skrivit fel i din kod, eller kopplat fel.
Vanliga fel är:

- Glömt att lägga till raden i setup (det måste finnas en rad för varje lysdiod).
- Lysdioden är vänd åt fel håll. Långa benet mot plus alltså mot där din siffra på arduinon står.

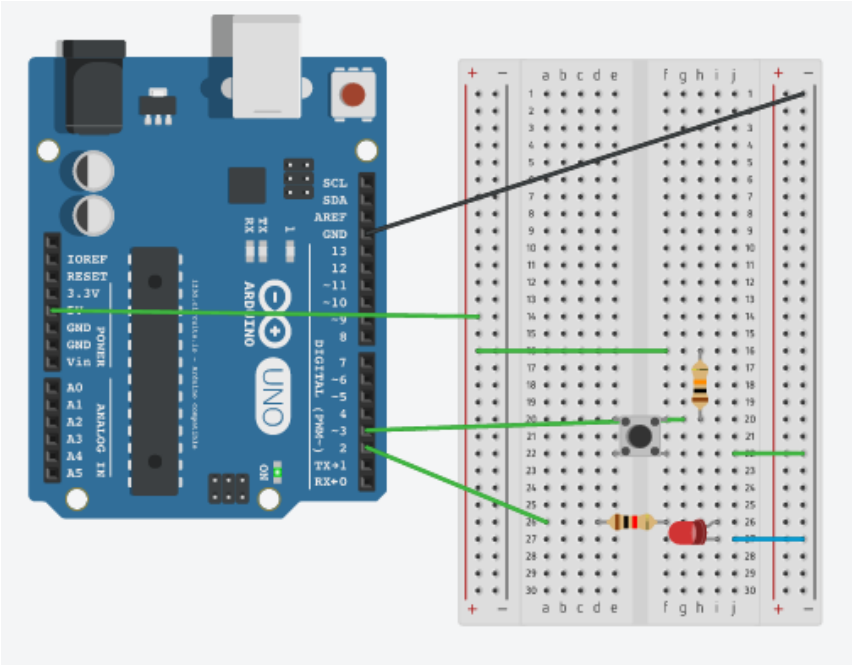
Prova nu att få den inbyggda lysdioden att vara tänd samtidigt som den du kopplade dit är släckt.
Du kan också prova att koppla upp ett par lysdiodet till. Tänk på att varje lysdiod måste ha var sitt motstånd. Du kommer också att behöva tala om vilka utgångar du vill använda i void setup().

Nästa utmaning är att få till så att tre lysdioder tänds efter varandra.
Alltså först är en Röd lysdiod tänd och en Orange och en grön släckt.
Sedan tänds den Orangea lysdioden, och de två andra är släckta.
Slutligen är den gröna tänd och de två andra släckta.

En sista rätt svår utmaning är att prova att göra en modell av en korsning med trafikljus. Alltså när det är rött åt ett håll så är det grönt åt det andra hållet. Sedan ska detta bytas så att det är rött åt andra hållet och grönt åt andra hållet. Det ska då vara en kort period av gult innan det blir rött. Till detta kommer du alltså att behöva sex lysdioder och sex motstånd.

Del 7 – Kopplingsdäck och tryckkontakt

Nu är det dags att prova att koppla in en strömbrytare på kopplingsdäcket. Den typ av strömbrytare vi har är sådana som är öppna när man inte trycker på dem och stänger/släpper fram ström när man trycker på dem. Dessutom har de hänger benen ihop parvis så att de som sticker ut på samma sida kopplas ihop.
Om du bygger upp detta på kopplingsdäcket och skriver in programmet här nedanför så ska lysdioden lysa när du trycker på knappen. OBS att den nya resistorn ska ha färgerna BRUN, SVART, ORANGE, GULD (alltså 10000 Ohm = 10kOhm) Det här används så att det alltid ska vara antingen noll eller ett och inte något mitt emellan.



```
/*
  Tryckknapp
* Lysdiod kopplad till pin 2
* Tryckknapp kopplad till pin 3 med 10Kohm resistor som Pullup
*/

// const = Något som bestäms här kommer inte att ändras i programmet:
const int buttonPin = 3;       // till vilket ben knappen är kopplad
const int ledPin = 2;          // till vilket ben lysdioden är kopplad

// variabler som kommer att ändras:
int buttonState = 0;           // variabel för att ha koll på tryckknappen

void setup() {
    pinMode(ledPin, OUTPUT);   // ställer in lysdioden som en utgång:
    pinMode(buttonPin, INPUT); // ställer in tryckknappen som ingång:
}

void loop() {
  buttonState = digitalRead(buttonPin);// Läs in knappen till variabeln buttonstate

  if (buttonState == LOW) {        //kolla om tryckknappen är tyckt, i så fall är den LOW
    digitalWrite(ledPin, HIGH);    // Slå på lysdioden.

  } else {                         // om knappen inte är nedtryckt (den är HIGH)
    digitalWrite(ledPin, LOW);     // Slå av lysdioden.
  }
}
```

Som du ser här ovanför så har jag här valt att skriva kommentarerna efter varje rad som en god programmerare bör göra :)

Nu har vi alltså ett sätt att känna av något i omvärlden som en knapp och skicka en signal ut till omvärlden som en lysdiod

Del 8 – Stegvis tändning av lysdioden (analog)

Ibland vill man inte bara ha antingen på eller av som lysdioden tidigare.
Vi använder då någon av utgångarna som har en ~ markering. I detta fall har jag valt pin3.
Byt alltså plats på de sladdar som går till pin2 och pin3 i kopplingen i del6.
Då kan vi nämligen använda något som heter PWM.
Du använder då funktionen analogWrite(ett tal mellan 0 och 255).
Om du skriver analogWrite(128) skickas pulser ut 128 av 255 alltså hälften av tiden. Detta betyder att till exempel en lysdiod i alla fall i teorin ska lysa hälften så starkt som om den får ström hela tiden.

```
/*
 PWM - Styra de analoga utgångarna
 */

int ledPin = 3;    // Lysdiod kopplad pin 3 med motstånd

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

void loop() {
  // Ändra från 0 till 255 i steg om 5 med FOR-loop:
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {
    analogWrite(ledPin, fadeValue); // skriver värdet till ledPin:
    Serial.println(fadeValue);
    delay(50); // vänta 50ms
  }

  // Ändra från 255 till 0 i steg om 5 med FOR-loop:
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {
    analogWrite(ledPin, fadeValue); // skriver värdet till ledPin:
    Serial.println(fadeValue);
    delay(50); // vänta 50ms
  }
}
```

Du kan nu prova att ändra värdet på ”delay:erna” och ändra så att det hoppar mer eller mindre än 5 steg åt gången.

Du kan också prova att ha en lysdiod kopplad till pin 3 och en annan kopplad till pin 5 och få en lysdiod att börja som tänd och en annan som släkt och få de att gå ”mot varandra” i tändning.

Del 9 – Stegvis tändning av lysdioden med potentiometer

Vi kan också känna av analoga signaler om vi använder någon av ingångarna A0 till A5.
Vi behöver då ha något som kan ge oss denna analoga signal. Vi tar därför fram något som kallas potentiometer. Det är ett motstånd som ändrar resistansen när vi vrider på en ratt. Det är i detta fall lämpligt att använda en potentiometer som har maxvärdet 1000 Ohm (1kOhm).
Lysdioden kan sitta kvar som i del 7 och i kommentarera nedan kan du läsa hur potentiometern ska kopplas.

Här nedanför ser du ett program som visar detta:

```
/*
  Analog Input som styr en lysdiods blinkande
 Koppling:
 * Koppla mittenbenet på potentiometern till A0
 * Ett av ytterbenen till GND
 * Det andra ytterbenet till +5V
 * Lysdioden till någon digital utgång med motstånd
*/

int sensorPin = A0;  // mittenbenet på potentiometern till A0
int ledPin = 3;      // där du kopplat lysdioden
int sensorValue = 0; // variabel för att lagra värdet från potentiometern

void setup() {
  pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT
  Serial.begin(9600); // starta seriella monitorn
}

void loop() {
  sensorValue = analogRead(sensorPin); // läs värdet på sensorn
  digitalWrite(ledPin, HIGH); // slå på lysdioden
  Serial.println(sensorValue); //skriv ut värdet på sensorn
  delay(sensorValue); // vänta så många millisekunder som <sensorValue> är
  digitalWrite(ledPin, LOW);  // slå av lysdioden
  delay(sensorValue); // vänta så många millisekunder som <sensorValue> är
}
```

Del 10 – Upp och ner av lysdioden med brytare

Här nedanför hittar du ett exempel på hur du kan få en lysdiod att stegvis tändas och släckas efter det att du tryckt på en brytare.

```
/*
    Ramp Up and down with Switch
*/
int ledPin = 3;      // Led kopplad till pin 3
int switchPin = 4;   // Brytare kopplad till pin 4
int val;             // variabel för att läsa status på brytaren
int buttonState;     // variabel för att spara sista läget på brytaren
int raknare;         // variabel för att räkna upp/ner

void setup() {
  pinMode(switchPin, INPUT); // Ställer in brytarens pin som ingång
  Serial.begin(9600);        // Startar serial monitor 9600bps
  buttonState = digitalRead(switchPin); // läser brytarens startläge
}


void loop() {
  val = digitalRead(switchPin);  // läser brytaren och lagrar i variabeln val

  if (val != buttonState) {      // brytaren har ändrats!
    if (val == LOW) {            // kolla in brytaren är nertryckt.
      Serial.println("Brytaren nertryckt");
      for (raknare = 0; raknare < 255; raknare++) // börja på 0, gå till 255 1 steg i taget
      {
        Serial.println(raknare);  //Skriver ut värdet på raknare till seriella monitorn
        delay(10);
        analogWrite(ledPin, raknare); // Skickar den bråkdel av 255 som räknaren visar
      }
    }
    else {                          // brytaren är inte nedtryckt
      Serial.println("Brytaren just släppt");
      for (raknare = 255; raknare >= 0; raknare--) // börja på 255, gå till 0 1 steg i taget
      {
        Serial.println(raknare);  //Skriver ut värdet på raknare till seriella monitorn
        delay(10); // Kort paus
        analogWrite(ledPin, raknare); // Skickar den bråkdel av 255 som räknaren visar
      }
    }

    buttonState = val;   // sparar nya läget i variabeln val
  }
}
```