ARDUINO PROGRAMMERING

Opgaver i programmering af Arduino

Teknologisk Institut

Indholdsfortegnelse

Kapitel 1

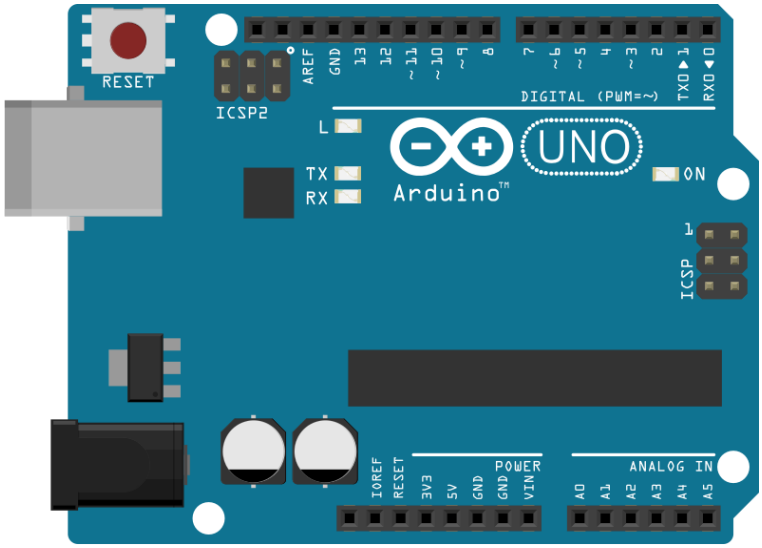
Arduino Uno mikrokontroller

En microcontroller (undertiden forkortet µC, UC eller MCU) er en lille computer i et enkelt integreret kredsløb. Den indeholder en processorkerne, hukommelse og programmerbare input-/output-enheder. Programhukommelsen forekommer i form af flash eller OTP-ROM findes også på chippen såvel som i en lille mængde RAM.

Mikrokontrollere er designet til indlejrede applikationer, i modsætning til mikroprocessorer som anvendes i personlige computere eller andre applikationer som anvendes til generelle formål. Det er altså en lille computer der kan bruges til at styre, måle eller kommunikere uden hjælp udefra.

Arduino er en open-source prototype platform, baseret på nemt tilgængelig hardware og software. Der er udviklet en del forskellige Arduino Microkontrollere. De findes i forskellige størrelser med forskellige antal input/output-porte og computerkraft. De er generelt indenfor en prisklasse der gør dem attraktive både for professionelle og hobbyister, og det har kastet uanede mængder erfaring og projekter med disse board af sig. Det betyder masser af viden der ofte er mulig at søge sig til online.

Nedenfor ses en tegning af en Arduino Uno mikrokontroller, som vil blive brugt i dette opgavesæt. Det er dog ingen hindring for at lave samme øvelser med en anden model.



Arduinoen kan enten få strøm igennem USB stikket, eller den kan forbindes til en anbefalet strømkilde på mellem 7V - 12V jævnstrøm. Her skal den negative pol (-) forbindes til en af 'GND' portene (GND = ground = jord), og den positive pol (+) forbindes til 'VIN'.

Man kan, når ardunioen er forsynet, få strøm fra arduinoen med stabilt signal på 5V og 3.3V relativt til GND, fra henholdsvis 5V og 3V3 porten. Dog kun 50mA fra 3.3V, hvor 5V porten begrænses igennem den eksterne forsyning.

Ud over det kan man få strøm fra alle 'Digital' porte, dog kun 20 mA per styk.

Til at interagere med arduinoen har man Input og Output porte. Til input er der 14 digitale porte, og 6 Analoge porte. Til Output kan alle 14 digitale porte bruges. 6 af dem kan endda bruges til at lave signaler kaldet PWM signaler, der i mange tilfælde kan bruges som et analogt output signal.

Selvom Arduinoen er begrænset i hvor meget strøm den kan levere, kan den oftest bruges til at styre større elektriske apparater ved at tænde for strømmen gennem andre elektriske komponenter.

Digitalt signal

Et digitalt signal har kun to tilstande, tændt eller slukket. De kaldes binære. Det bliver repræsenteret internt i Arduinoen med 1 for tændt og 0 for slukket. I den fysiske verden svarer det på arduinoen til 5V eller 0V enten som input eller output på de digitale porte. Med input vil ca. 0V-2V blive registreret som slukket (0), og ca. 3V-5V blive registreret som tændt (1).

Alt der foregår inde i en computer foregår på laveste niveau med binære tal, da det er sådan hardwaren i sidste ende virker.

Analogt signal

Et analogt signal kan på Arduinoen derimod repræsentere en tilfældig spænding mellem 0V og 5V. Man kan altså vide om der på et signal er 1V, 2,7V eller 4,3V, osv..

Der er dog en begrænset opløsning på det analoge signal i Arduinoen. Da målingen skal repræsenteres digitalt inde i Arduinoen, bruger man et binært tal. Med det analoge signal er det et 10bit signal, hvilket betyder at det kan have 1024 forskellige værdier, som er alle heltal fra 0 til 1023.

2,5V vil dermed give en værdi på en analog læsning på 511, eller mere matematisk $analog\ værdi = \frac{V\_analog}{5V} * 1023.$

Man kan tilnærmelsesvist lave et analogt output signal via 6 af de digitale porte, markeret med '~', der kan lave et PWM signal. Mere om det i opgave 5.

Arduino Uno Specifikationer

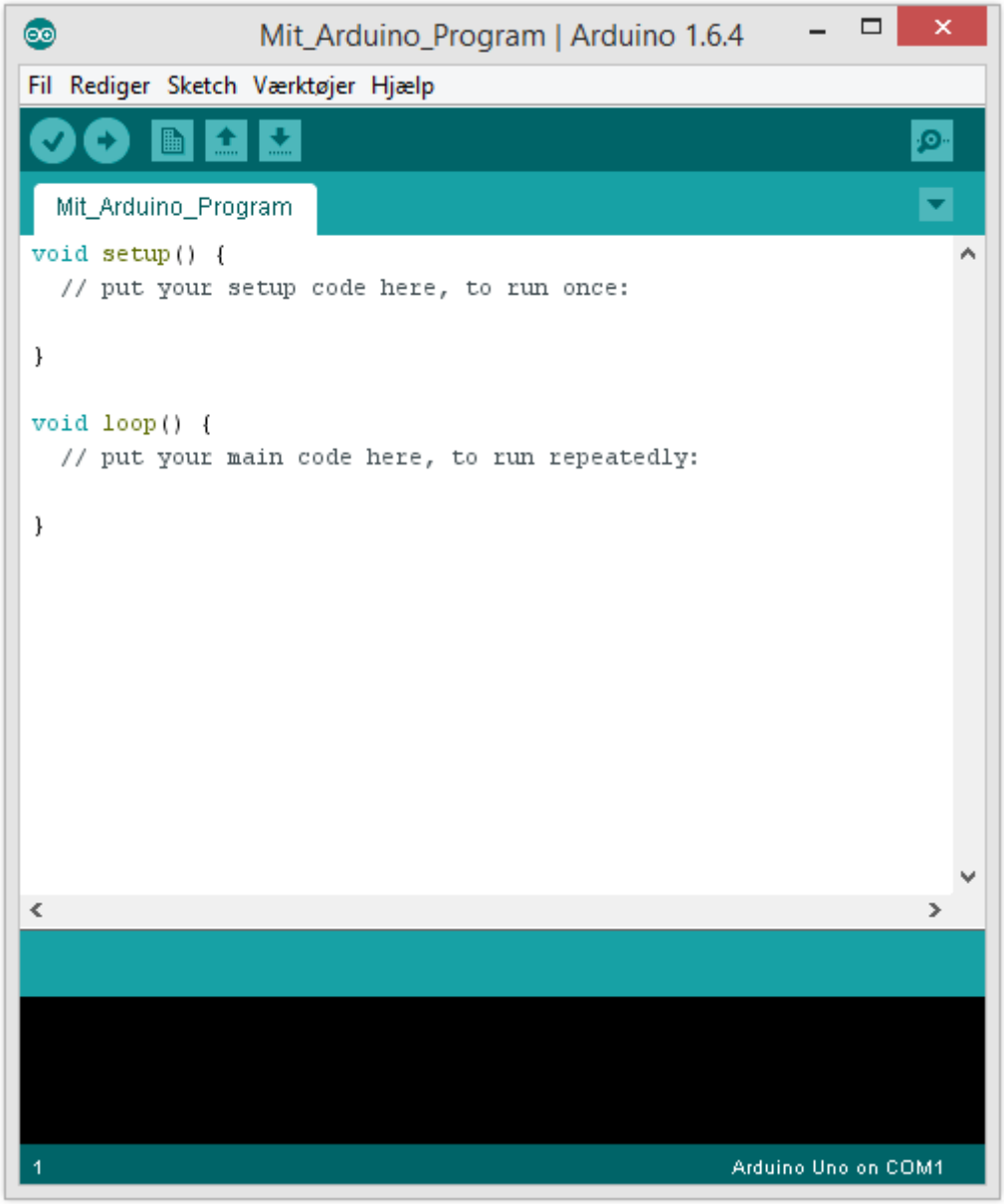
Nedenstående er de helt grundlæggende specifikationer for Arduino Uno. Det er en rimelig begrænset computer i forhold til en PC eller en smartphone, men det er muligt at programmere den præcis som man vil have, og den er nem at bruge med alverdens elektriske komponeneter! Hvorfor vente på at andre opfinder tingene når man selv kan gøre det?

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Programmering af Arduino

Man programmere Arduino mikrokontrollere i programmet ”Arduino”, som kan downloades gratis fra hjmmesiden: http://www.arduino.cc/en/Main/Software.

Programmet ser ud, som på nedenstående billede.



Når man tilslutter sin Arduino til sin PC skal man under ’\Værktøjer\port’ vælge en COM port hvor arduinoen er tilsluttet. Programmet foreslår oftest selv hvilken port der skal bruges. Alternativt kan man når Arduinoen er tilsluttet, i windows gå ind i ”enhedshåndtering”, og under punktet ”porte” se hvilken COM port arduinoen har sluttet sig til.

Et Arduino program indeholder altid en **setup( )**-funktion og en **loop( )**-funktion. Setup-funktionen kører kun én gang når Arduinoen tændes, hvorimod alt inden for loop-funktionens tuborg-paranteser kører igen og igen, så længe Arduinoen er tændt.

Man kan godt definere variabler udenfor både setup og loop, men de vil kun blive defineret når Arduinoen startes.

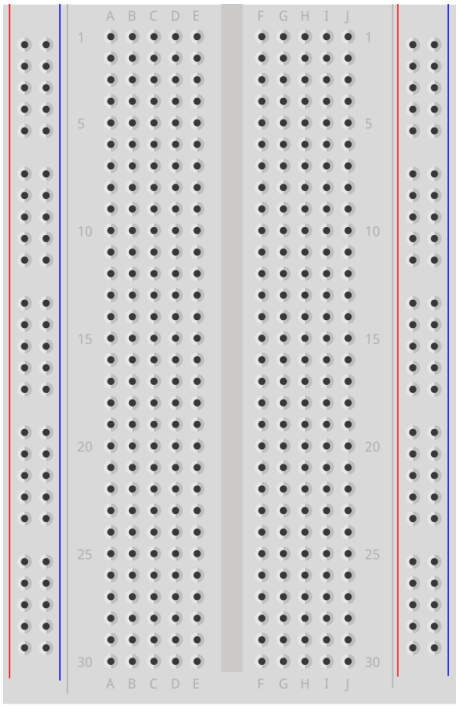
Skriver man 2 skråstreger '/ /' udkommenterer man resten af linjen. På den måde kan man enten skrive kommentarer ind i programmet som ikke vil have nogen indflydelse på kørslen af programmet, men vil hjælpe på læseligheden af ens kode, eller man kan udkommentere linjer der midlertidigt ikke skal med i programmet.

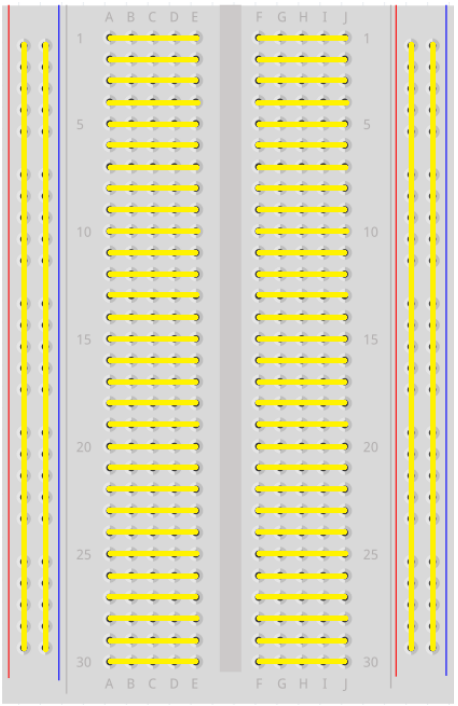
Man kan kontrollere at ens program ikke har nogen syntax-fejl ved at trykke på fluebenet i øverste venstre hjørne. En syntax-fejl er en fejl i hvordan kommandoer eller navne på ting er skrevet i programmet. Det er ikke ligegyldigt om man har brugt store eller små bogstaver, og det er ikke ligegyldigt om man bruger punktum eller komma, eller andre tegn. Vær opmærksom på hvordan funktionerne skrives, og kopier gerne fra program eksempler der allerede virker! Paranteser skal altid havde både start og slut parantes!

Hvis der ikke er nogen syntax-fejl kan man så prøve at uploade programmet til sin Arduino ved at trykke på den højrepegende pil ved siden af fluebenet. Husk at USB stikket skal være i både computer og Arduino ☺

Breadboard

Et breadboard er en plade hvorpå man kan lave elektroniske kredsløb. Et breadboard, som også kaldes et fumlebræt, ser typisk således ud:

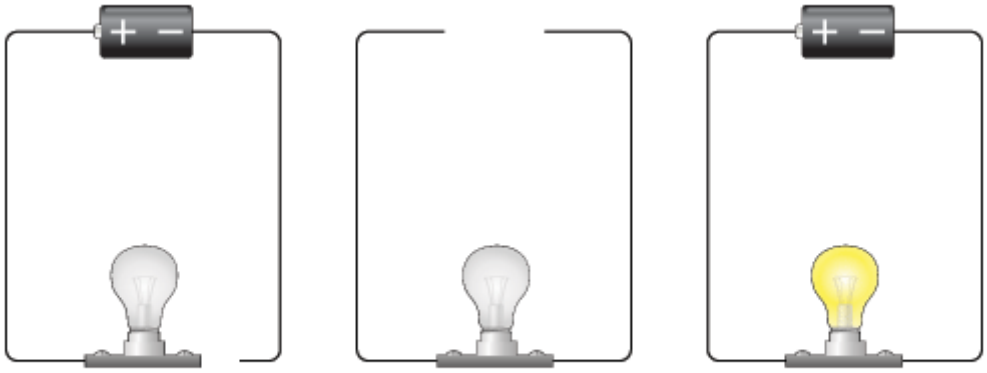




Breadboardet består af en plade med nogle huller, hvor nogle huller er forbundet så man let kan stikke ledninger i fra flere forskellige komponenter og derved forbinde dem.

Hver række, bestående af 5 huller, som er indbyrdes forbundet bruges oftest til komponenter. De to rækker i hver side, har på hver side en (+) og en (-) række. På den måde kan man altid nemt få 5V eller 0V tilsluttet ind til sine komponenter.

Man skal huske at strømmen altid løber fra (+) til (-), eller fra høj spænding til lav. Der skal altid være et lukket kredsløb før der løber en strøm.



Kapitel 2

I dette kapitel gennemgås hvordan man helt grundlæggende får arduinoen til at styre eksterne ting, via output, og hvordan man kan kommunikere til arduinoen, via input.

Samtidig vil if-, else if-statements, og for-løkker blive præsenteret.

Komponenter til opgaverne i Kapitel 2

I dette kapitel skal følgende komponenter bruges for, at opgaverne kan løses:

- Arduino Uno Mikrokontroller
- A-B USB kabel (medfølger som regel med Arduino'en)
- Breadboard
- 1 stk. Lysdiode
- 1 stk. 150 Ohms modstand
- Hårde ledninger
- 1 stk. trykknap

Opgave 1: Lys i pæren

I denne opgave skal vi få en lysdiode til at lyse ved at bruge Arduinoen som strømkilde. Da en Arduino Uno leverer 5 volt og en typisk lysdiode kun skal have 2 volt, skal vi bruge en modstand for, at begrænse spændingen fra 5 volt til 2 volt.

Ifølge Ohms lov kan størrelsen på denne modstand beregnes med formlen:

$$R = \frac{V}{I}$$

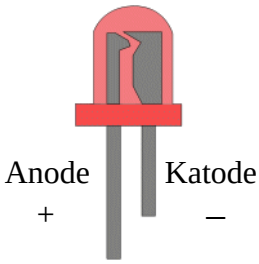
Hvor $V = V_{arduino} - V_{lysdiode}$

En lysdiode trækker en strømstyrke på ca. 20mA = 0,02A. Vi kan nu beregne modstanden til:

$$R = \frac{V_{arduino} - V_{lysdiode}}{I} = \frac{5V - 2V}{0{,}02A} = 150\Omega$$

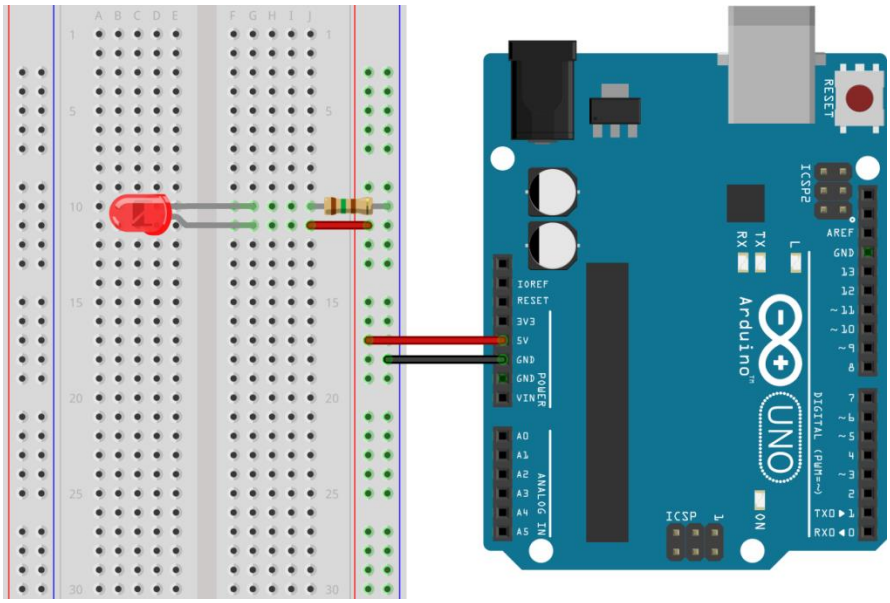
Vi skal altså bruge en 150Ω modstand i kredsløbet for, at 'bremse' 3 af de 5 volt, så lysdioden ikke går i stykker.

En lysdiode har to ben, som kaldes en Anode og en Katode. Anoden er længere end Katoden og strømmen skal løbe fra Anoden til Katoden. Det vil sige, at det lange ben på lysdioden skal tilsluttet + og det korte ben skal tilsluttes –.



Hvis man tilslutter strømmern til lysdioden omvendt, vil den ikke tænde fordi strømmen kun kan løbe én vej gennem den. Derfor er det vigtigt, at vende lysdioden rigtigt. Hvis den ikke lyser når man sætter strøm på kredsløbet, kan det derfor også være en ide at tjekke om dioden skal vendes.

På billedet nedenfor er det vist hvordan lysdioden skal forbindes til fumlebrættet, og videre til Arduinoen for at den lyser.

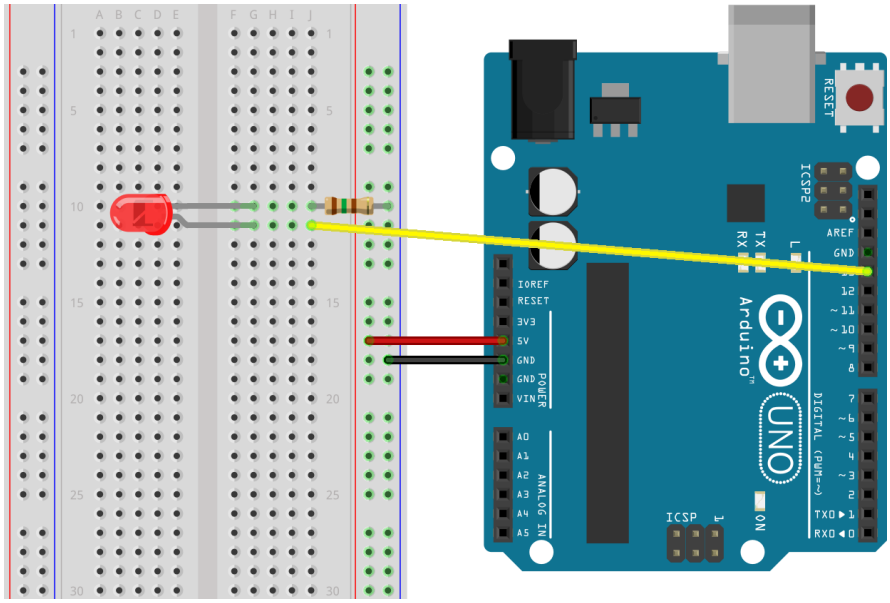


1. Placer lysdioden på Breadboardet, så hvert ben går i et hul.
2. Placer en 150Ω modstand med forbindelse til lysdiodens (-) med det ene ben, og til GND på arduninoen
3. Dioden skulle nu gerne lyse så snart arduinoen får strøm, via USB eller VIn.

Opgave 2: Blinke

Åben eksemplet "Blink"

Fil \ Eksempler \ 01.Basics \ Blink



1. Placer lysdioden på Breadboardet, så hvert ben går i et hul.
2. Placer en 150Ω modstand med forbindelse til lysdiodens (-) med det ene ben, og til GND på arduninoen med modstandens andet ben.
3. Forbind (+) benet på dioden til hul 13 på arduinoen med en ledning.
4. Når arduinoen får strøm og programmet er uploadet, vil dioden blinke.

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

void loop() {
  digitalWrite(13, HIGH);   //Tænder for dioden
  delay(1000);              //1 sekund pause
  digitalWrite(13, LOW);    //Slukker for dioden
  delay(1000);
}
```

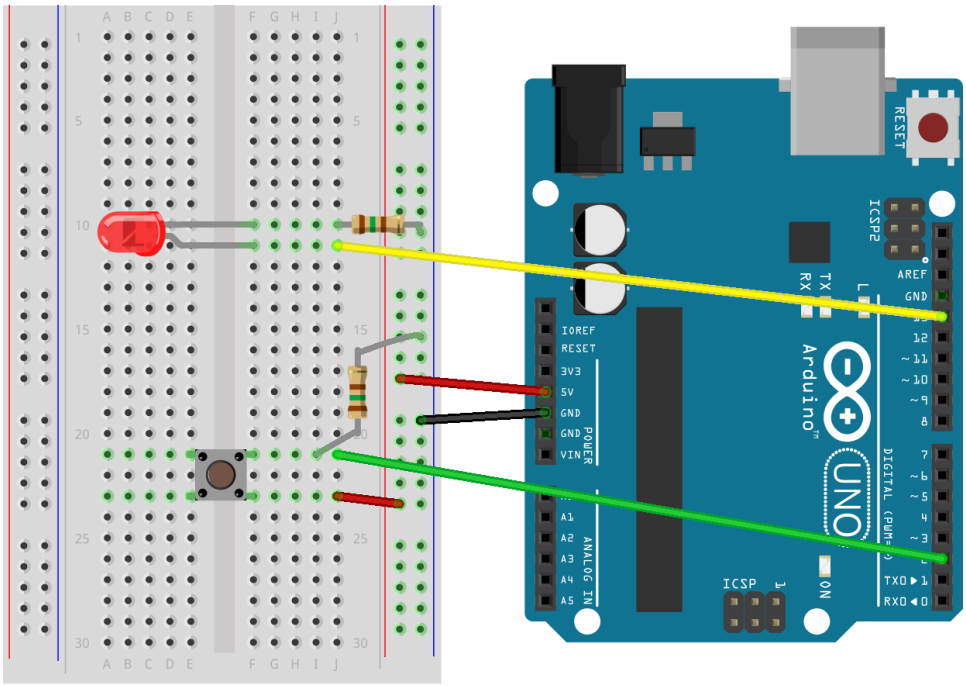
Linien `pinMode(13, OUTPUT);` i setup-funktionen gør pin nummer 13 på Arduino'en til en output-pin. Dvs. at pin nr. 13 i dette tilfælde skal bruges til, at tænde og slukke strømmen fra.

Opgave 3: Tænd LED via. trykknap

Åben eksemplet ”Button”

Fil \ Eksempler \ 02.Digital \ Button

Der bruges i denne opgave en 150 ohm modstand mellem GND og signal pin (pin 2) til knappen. Denne kaldes en pull-down resistor. Den bruges til holde spændingen på input signalet på 0 v, indtil knappen trykkes ned, og 5 volt signalet får lov at løbe igennem. Forsøg også at se hvad der sker hvis den ikke er på.



1. Brug opsætningen fra opgave 2.
2. Tilføj en trykknap ind over midten af breadboardet, med et ben forbundet til 5v, og det andet ben forbundet til pin 2. Forbind også pin 2 til ground med en 150 ohm modstand.
   Når man trykker på knappen er der er forbindelse imellem to eller flere ben på knappen. Den virker derved bare som en lille kontakt der kun lader strøm løbe når den er trykket ind.
3. Sæt strøm til arduinoen og upload programmet ”Button”.
4. LED’en lyser når knappen trykkes ned.

Systemet virker ikke som en normal stikkontakt, hvor kontakten er en reel kontakt, der ”åbner” for strømmen til en pære når kontakten er tændt. (Eller lukker kredsløbet)
Systemet virker derimod ved at arduinoen kigger på input fra det ben som knappen er forbundet til, og hvis den er trykket ned, ”tænder” arduinoen for strømmen på output på det ben LED’en er forbundet til. På den måde kan man med input kommunikere til microcontrolleren og få den til at udføre handlinger ved hjælp af output til forskellige genstande.

```
const int buttonPin = 2;     // Tal på knap pin
const int ledPin =  13;      // tal på led pin
// variable tal:
int buttonState = 0;         // variabel til at læse knap
status
void setup() {
  // Sæt LED pin til output:
  pinMode(ledPin, OUTPUT);
  // Sæt knap pin til input:
  pinMode(buttonPin, INPUT);
}
void loop() {
  // Læs tilstanden på knappen (0 (intet tryk) eller 1 (tryk)):
  buttonState = digitalRead(buttonPin);

  // Tjek om knappen er trykket ned.
  // Hvis den er, er buttonState HIGH (høj):
  if (buttonState == HIGH) {
    // tænd LED:
    digitalWrite(ledPin, HIGH);
  }
  else {     //ellers
    // sluk LED:
    digitalWrite(ledPin, LOW);
  }
}
```

If...else funktion

Der bliver i programmet brugt en if...else-funktion:

```
  if (buttonState == HIGH) {
    // tænd LED:
    digitalWrite(ledPin, HIGH);
  }
  else {      //ellers
    // sluk LED:
    digitalWrite(ledPin, LOW);
  }
```

Som grundlæggende virker ved:

```
  if (<betingelse>) {
    //Udfør kommandoer I denne parantes hvis <betingelse> er
sand (1)
  }
  else {      //ellers
    //Hvis <betingelse> var falsk (0) udføres kommandoerne i
denne parantes
  }
```

I eksemplet er betingelsen: ”`buttonState == HIGH`”. buttonState er 1 hvis knappen er trykket ned, og 0 hvis knappen ikke er trykket ned. Betingelsen spørger derfor ”er buttonState lig 1?” Hvis den er 1, er knappen trykket, og ”`buttonState == HIGH`” = 1, hvis ikke, er den lig 0.

Med den viden kunne man egentlig droppe ”== HIGH”, da buttonState lig 1, evaluerer til 1, og buttonState lig 0, evaluerer til 0. Første linje ville dermed hedde ” `if (buttonState) {`”, men det ville jo ikke være lige så forklarende i et eksempel ☺

Der findes forskellige betingelses-operatorer. De forholdsmæssige operatorer er:

```
>     Større end                  5 > 4   er TRUE
<     Mindre end                  4 < 5   er TRUE
>=    Større end eller lig med    4 >= 4 er TRUE
<=    Mindre end eller lig med    3 <= 4 er TRUE
==    lig med                     5 == 5 er TRUE
!=    ikke lig med                5 != 4 er TRUE
```

Med disse kan man sætte forskellige betingelser op med to eller flere variable og afhængig af udfaldet få microcontrolleren til at reagere på forskellige måder.

Eksempel fra det virkelige liv (undlad brug af æ, ø og å i programmering):

```
If (Opsparing >= Uundværligt_grej) {

Tag spenderbukserne på

}

Else {

Få et job

}
```

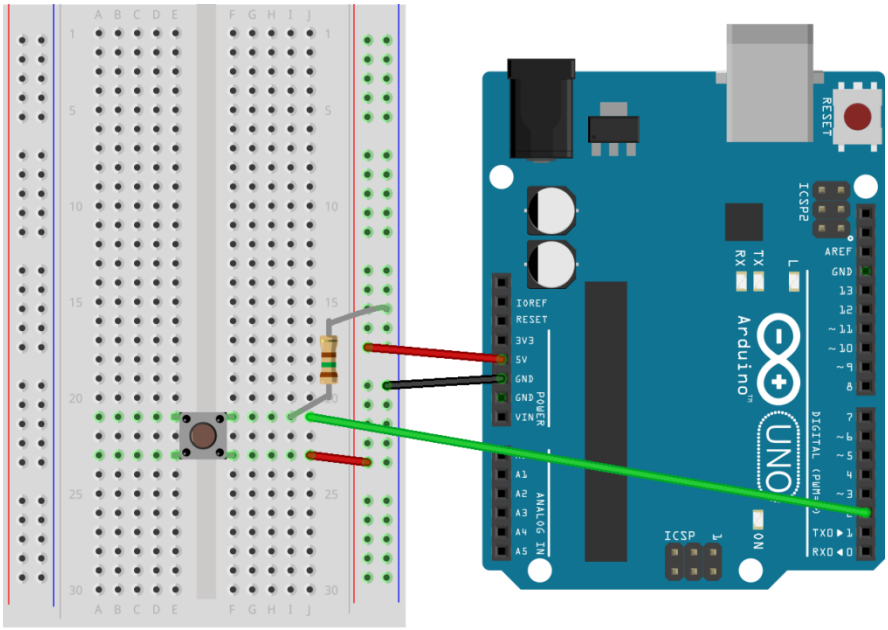
Hvis man i en betingelse ved en fejl kommer til at bruge '=' i stedet for '==', vil man sætte sin første variable lig den næste. Så

```
If (Variabel_1 = 0) {

//kommer aldrig herind

}
```

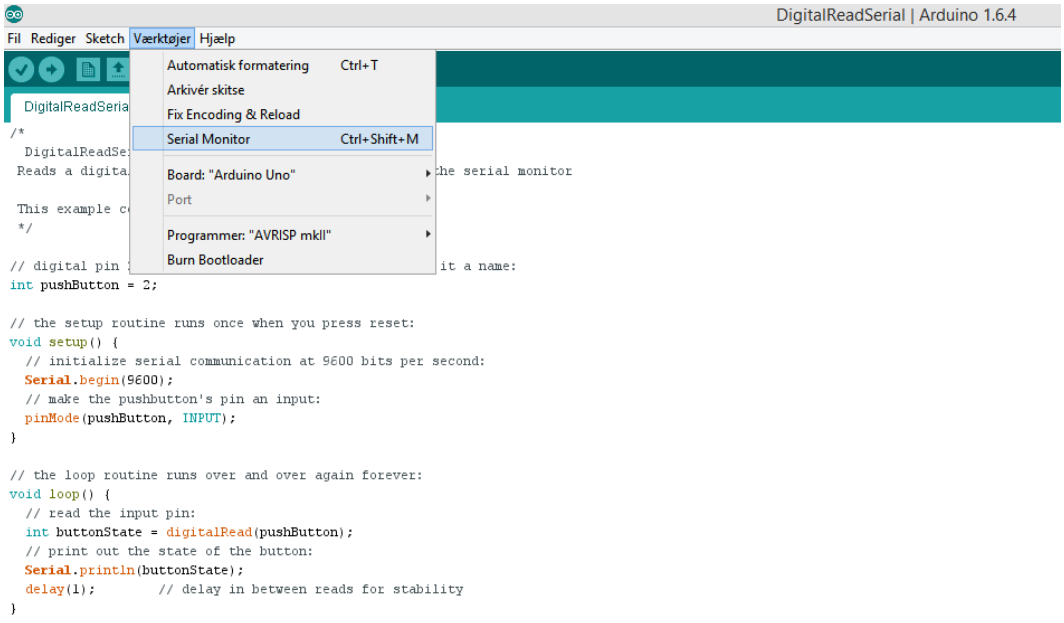
Variabel_1 bliver her sat lig 0, og resultatet bliver if (0) som er det samme som if (false). Hvis den omvendt blev sat lig et tal der var forskelligt fra 0, ville betingelsen altid være sand, og man opnår igen ikke den ønskede hensigt. Husk derfor '==' og ikke '=' i betingelser.

Opgave 4: Digital read serial – serial monitor via trykknap

I denne opgave bruges seriel kommunikation fra arduinoen og udadtil, for at kunne aflæse hvad der sker i programmet.



1. Genbrug samme opstilling med knappen som i opgave 3. Lad gerne dioden sidde forbundet, det gør ingen forskel.
2. Åben programmet Fil \ Eksempler \ 01.Basics \ DigitalReadSerial
3. Upload programmet til arduinoen via USB kablet, og åben serial monitor



4. Sæt kassen nederst i højre hjørne af monitoren til 9600 baud, passende til denne koden.

```
// Knappen er forbundet til pin 2:
int pushButton = 2;
void setup() {
  // initialisrer serial kommunikation ved 9600 bits per sekund:
  Serial.begin(9600);
  // Sætter knappens pin til input:
  pinMode(pushButton, INPUT);
}
void loop() {
  // læser input pin:
  int buttonState = digitalRead(pushButton);
  // Printer knappens tilstand i seriel monitor:
  Serial.println(buttonState);
  delay(1);        // pause mellem læsninger for stabilitet
}
```

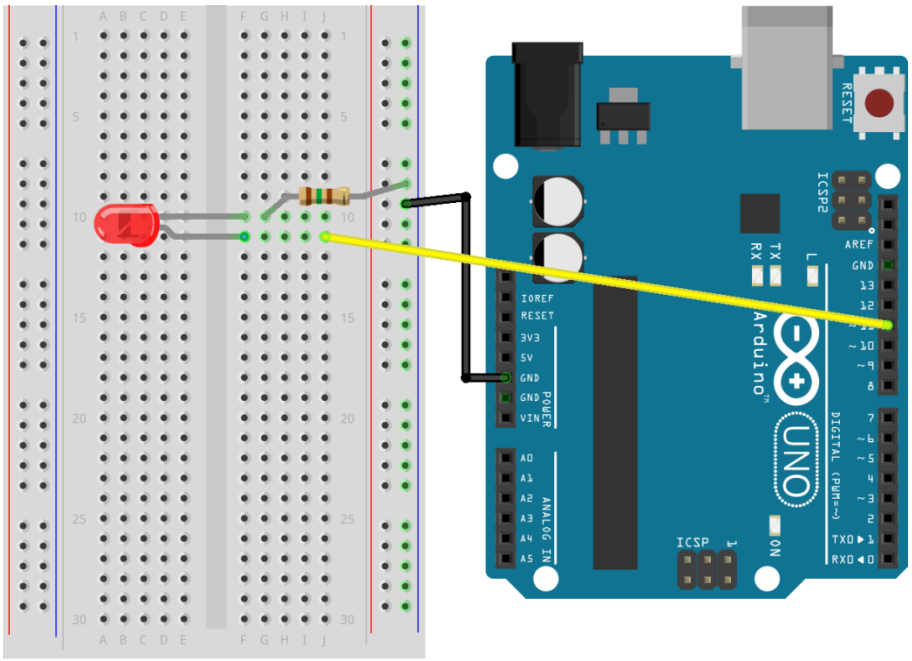
Der vil når knappen er trykket ned skrives 1 i serial monitor, og 0 når den ikke er trykket ned.

Dette kan bruges generelt til at udskrive variabler til monitoren så man kan se hvad ardunioen læser fra sine input, hvad den prøver at skrive på sine output, eller hvad en beregning i programmet måtte give af resultater. På den måde kan man fejlfinde, eller se hvad der sker i programmet.

Serial kommunikationen kan dog også bruges til at kommunikere med andre microcontrollere eller programmer på computere, og endda nogle sensorer eller andre elektriske komponenter. Det kommer vi dog ikke videre ind på i disse opgaver.

Opgave 5: For-løkke med LED

I denne opgave udvider vi blinke eksemplet med at tage både for-løkken og analogWrite i brug.



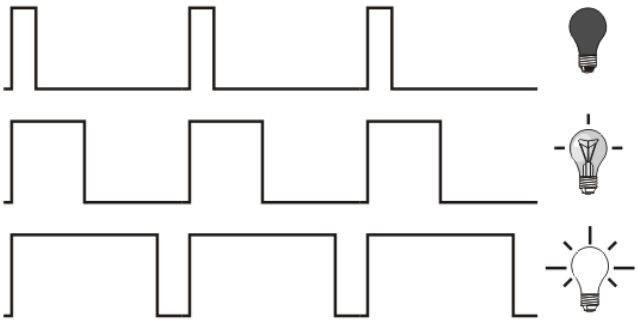
1. Lysdioden forbindes næsten på samme vis som i opgave 2, men her forbindes (+) til pin 11.
2. Kopier nedenstående program ind i IDE'en og upload det.
3. Når arduinoen får strøm og programmet er uploadet, vil dioden stå og tiltage og aftage i lysstyrke.

analogWrite

Med digitalWrite kan man på en given digital pin sætte spændingen til enten 5V eller 0V relativt til GND.

analogWrite kan til gengæld lave en spænding på mellem 0V og 5V ved at lave et PWM signal. PWM står for Pulse Width Modulation.

Dette signal har en frekvens på de fleste pins på omkring 490 Hz, altså tænder og slukker den på 1s/490Hz = 0,002 sekund.



Det går så hurtigt at mange komponenter, som en LED eller en motor vil "opføre" sig som om at den får en ren spænding tilsvarende til den procentdel af tiden pin'en er tændt. Tændt 10% af tiden giver 0,5 V, 50% af tiden giver 2,5V, osv.

analogWrite virker kun på nogle pins, da det kun er nogle på arduinoen der kan lave PWM output. De er angivet med en lille '~' foran pin nummeret.

Værdien som kan outputtes er 8bit, og skal dermed være mellem 0 og 255, fordelt på 0V til 5V.

```
int led = 11;

// Setup function kører kun én gang når arduinoen tændes

void setup() {

  // Sætter pin 11 til output

  pinMode(11, OUTPUT);

}


// Loop funktionen kører så længe der er strøm på arduinoen

void loop() {

  for (int i=0; i < 256; i++) {

    analogWrite(led, i);    //Skriver en analog værdi til LED pin

    delay(5);               // Venter i 5 ms = 0.005 sekund

  }

  for (int i=255; i > 0; i--) {

    analogWrite(led, i);    //Skriver en analog værdi til LED pin

    delay(5);               // Venter i 5 ms = 0.005 sekund

  }

}
```

For-løkken

Ud over hovedløkken i programmet, er der i denne opgave brugt en for-løkke. Dette er en måde hvorpå man nemt kan enten gentage den samme opgave et bestemt antal gange, eller gentage den samme opgave men med varierende parametre.

I denne opgave bruges ” `for (int i=0; i < 256; i++)` ” hvor i bruges som tæller. i sættes til en start lig 0 (`int i=0`), derefter sættes en betingelse om at for-løkken skal køre så længe i er mindre end 256 (`i < 256`), og slutteligt sættes inkrementeringen af i til at være 1 per cyklus (`i++`).

i++ er en forsimplet måde at skrive i = i +1, som ville lægge én til i og gemme dette tal oveni i. Man kan dog variere både hvilken variabel man bruger, hvad betingelsen er for at løkken skal blive ved at køre, og med hvilke skridt, variende størrelse kan bruges, som variablen skal ændre sig. Hvis man så bruger ’i’, eller hvad man nu kalder tælleren, til fx at styre hvor kraftigt dioden lyser, kan man få den til at skrue meget fint op og ned.

Kapitel 3

I dette kapitel vil der blive introduceret nogle flere komponenter, men det vil også blive demonstreret hvordan man kan bruge input fra en komponent til at styre en eller flere andre.

If..else funktionen vil også blive udvidet med mellemleddet 'else if'.

Komponenter til opgaverne i Kapitel 3

I dette kapitel skal følgende komponenter bruges for, at opgaverne kan løses:

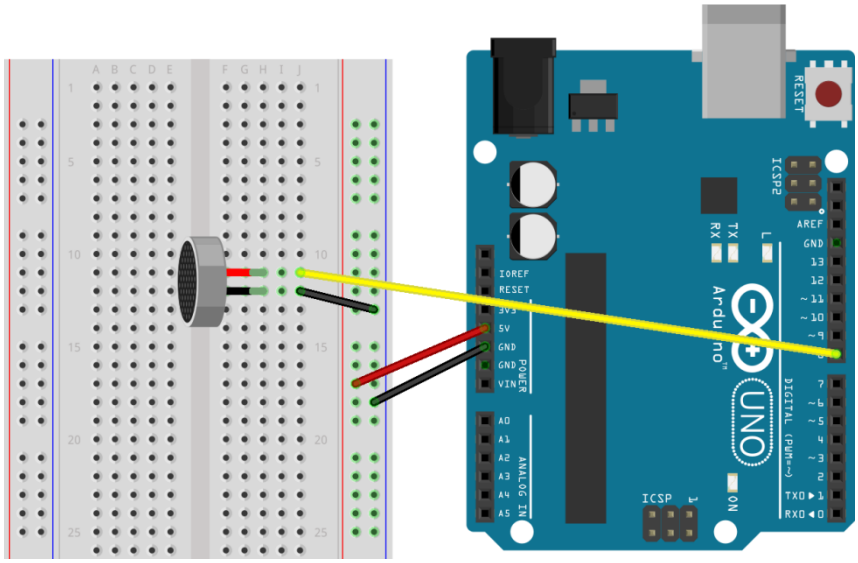
- Arduino Uno Mikrokontroller
- A-B USB kabel (medfølger som regel med Arduino'en)
- Breadboard
- Hårde ledninger
- 1 stk. Piezo højtaler
- 1 stk. Trykknap
- 1 stk. Sonarsensor
- 3 stk. Lysdioder
- 3 stk. 150 Ohms modstande

Opgave 6: Beeeep

I denne opgave skal vi få en lille 'Piezo buzzer', eller polyfonisk højtaler, til at lave en tone.

Da højtaleren vi bruger (ABT-402-RC) fungerer ved 1 – 20 volt, kan den fint klare de 5 volt arduinoen leverer.

Den valgte højtaler har to ben og den fungerer lige meget hvilken vej man vender ledningerne til den. Dette er ikke tilfældet med alle typer piezo buzzere. Prøv derfor at vende plus og minus hvis det ikke virker i første omgang.



1. Placer højtaleren på Breadboardet, så hvert ben sidder i ikke-forbundne huller.
2. Forbind det ene ben til GND, og det andet ben med en ledning til digital pin 8.
3. Kopier nedenstående ind i arduino IDE'en og upload til arduinoen.

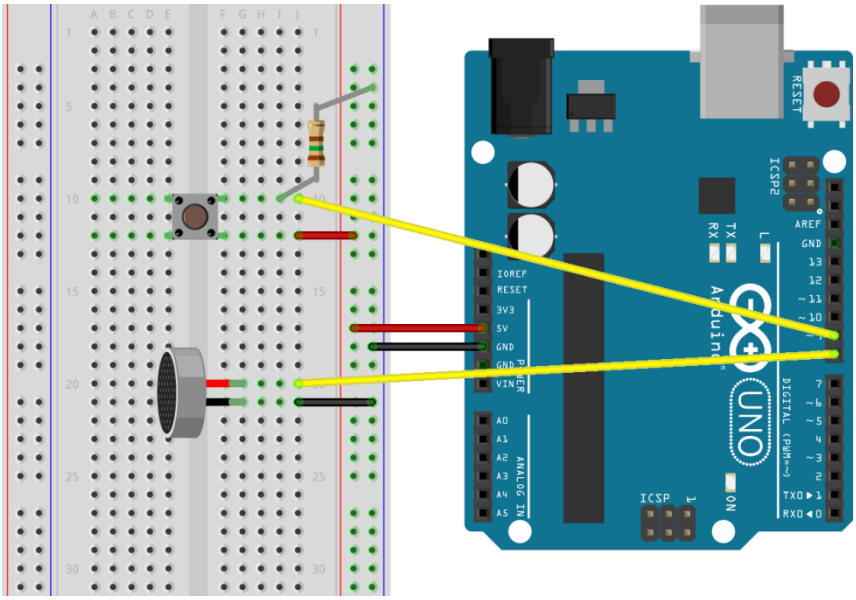
```
void setup() {

}

void loop() {

    tone(8, 440, 200);      //Laver en tone på pin 8 i 200 ms

    delay(1000);            //Pause i 1000ms = 1 s

    noTone(8);              //Slår tone-funktionen fra på pin 8

}
```

Opgave 7: Den irriterende knap

I denne opgave skal vi få piezo buzzeren til at lave en tone, når den aktiveres med en trykknap.

Problemet er præcis det samme som i opgave 3, men denne gang bruges en højtaler i stedet for en LED diode.

Sæt en 150 ohm modstand mellem GND og signal pin (pin 9) til knappen. Denne kaldes en pull-down resistor. Den bruges til holde spændingen på input signalet på 0 v, indtil knappen trykkes ned, og 5 volt signalet får lov at løbe igennem. Den er nødvendig hvis man vil undgå støj på signalet. Forsøg også at se hvad der sker hvis den ikke er på.



5. Brug opsætningen fra opgave 6.
6. Tilføj en trykknap ind over midten af breadboardet, med et ben forbundet til 5v, og det andet ben forbundet til pin 9. Forbind også pin 9 til ground med en 150 ohm modstand.
7. Prøv selv at koble opgave 6 sammen med opgave 3 i koden. Brug gerne lidt tid ☺
8. Hvis punkt 3 fejler: Kopier nedenstående ind i arduino IDE'en og upload til arduinoen.

```
// Konstanter: sæt pin numre:
const int buttonPin = 9;     // Tal på knap pin
const int ledPin =  13;      // Tal på indbygget LED pin
const int buzzerPin =  8;      // Tal på højtaler pin
// variabler der kan ændres:
int buttonState = 0;         // variabelt tal til at læse knap

void setup() {
  pinMode(ledPin, OUTPUT);  //Sæt LED pin til output
  pinMode(buzzerPin, OUTPUT);  //Sæt buzzerpin til output
  pinMode(buttonPin, INPUT);//Sæt knap pin til input
}

void loop() {
    buttonState = digitalRead(buttonPin); // Læs knap værdi:

  // check if the pushbutton is pressed.
  if (buttonState == HIGH) {  // Hvis, buttonState er høj:
    digitalWrite(ledPin, HIGH); // tænd LED:
    tone(buzzerPin, 440);        // Afspil tone:
  }
  else { // Hvis, buttonState ikke er høj:
    digitalWrite(ledPin, LOW); // sluk for LED:
    noTone(buzzerPin);         //Sluk for tone
  }
}
```

Opgave 8: Sonar sensor

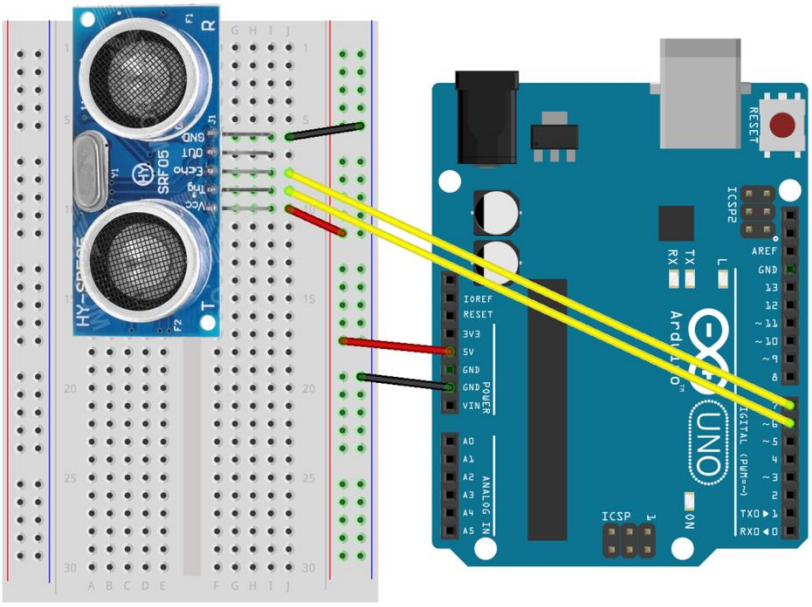
I denne opgave skal vi bruge en sonar sensor (HY-SRF05) til at bestemme afstand.

Sonar sensoren virker ved at udsende nogle højfrekvente (8stk, 40 kHz) lyd bølger, for derefter at bestemme hvor lang tid der går før lydbølgerne når tilbage til sensoren. Da lydens hastighed i luft ved forskellige temperaturer er kendt, kan man ud fra lydens rejsetid beregne afstanden til objektet foran sensoren. Denne rejsetid outputter sensoren på sin Echo pin, ved at lave en 5v puls der varer den samme tid som lyden rejste. Da lyden rejser både frem og tilbage, skal man dividere med to for at få den halve rejsetid.

$$\text{distance} = (\text{T(pulslængde)} * (340\ \text{m/s})) / 2$$

Hvis præcision er vigtig, kan man tage højde for luftens temperatur. Det er dog altid relevant at tjekke kalibreringen af sin sensor. Hvis man forbandt en termister (temperatursensor) kunne man endda man få sin microprocessor til selv at tilpasse sig temperaturen!

$$C_{lyd} = 331.3\frac{m}{s} + 0.606\frac{\frac{m}{s}}{°C} \times \text{Temperatur\_i\_°C}$$

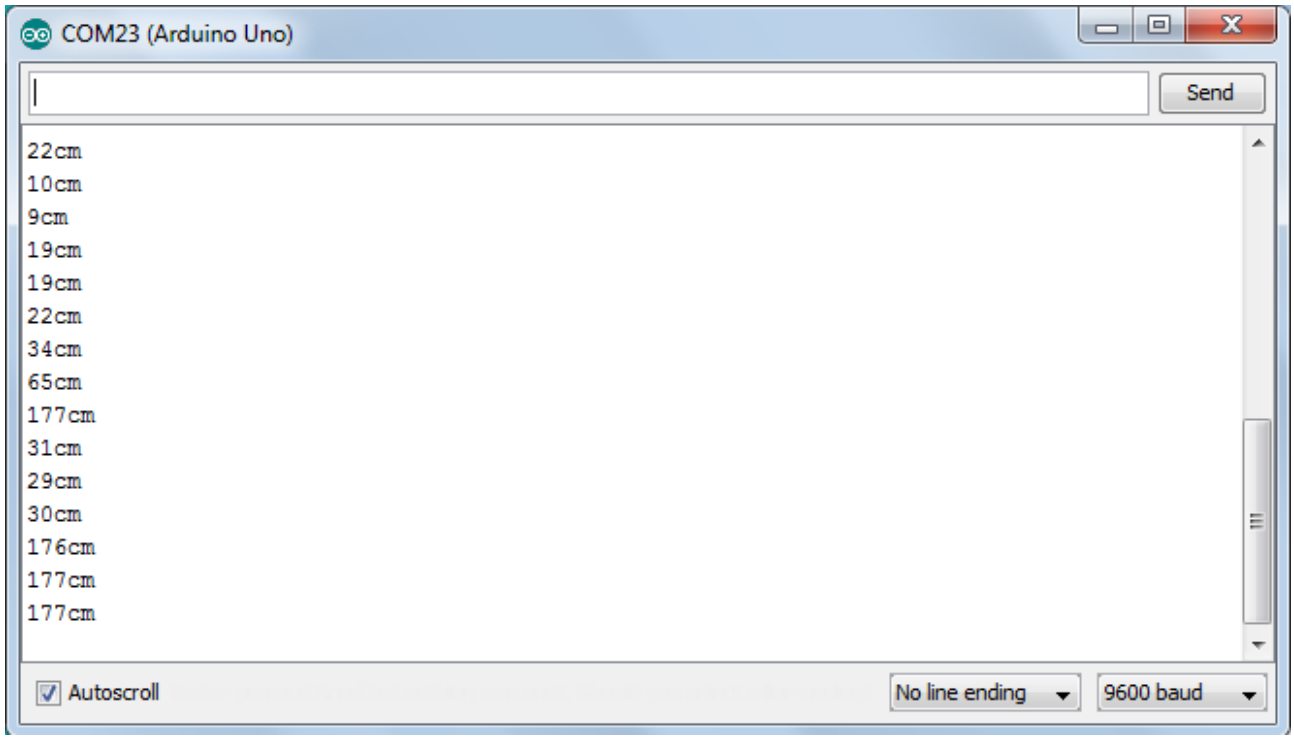


1. Forbind sonar sensoren som vist på tegningen. Echo til pin 7, Trig til pin 6, Vcc til 5v og GND til jord.
2. Kopier nedenstående ind i arduino programmet, upload til arduinoen, og åbn /Værktøjer/Serial monitor
3. Forsøg at sæt noget foran sensoren og se om de afstande der vises i terminalen passer med virkeligheden.

```
//Antager 20 grader celcius rumtemperatur

const int TRIG_PIN = 6;

const int ECHO_PIN = 7;


void setup() {

  // initialiser serial kommunikation:

  Serial.begin(9600);

  pinMode(TRIG_PIN,OUTPUT);

  pinMode(ECHO_PIN,INPUT);

}

void loop()

{

   long duration, distanceCm;

  // Laver en kort lav puls først, for derefter at lave en ren
høj 10ms puls:

  digitalWrite(TRIG_PIN, LOW);       //Sætter den lav

  delayMicroseconds(2);              //Vent 2ms

  digitalWrite(TRIG_PIN, HIGH);      //Sætter den høj

  delayMicroseconds(10);             //Venter => 10ms lang puls

  digitalWrite(TRIG_PIN, LOW);       //Sætter den lav

  duration = pulseIn(ECHO_PIN,HIGH);//Venter på, og læser
puls-længde på echo-pin.
```

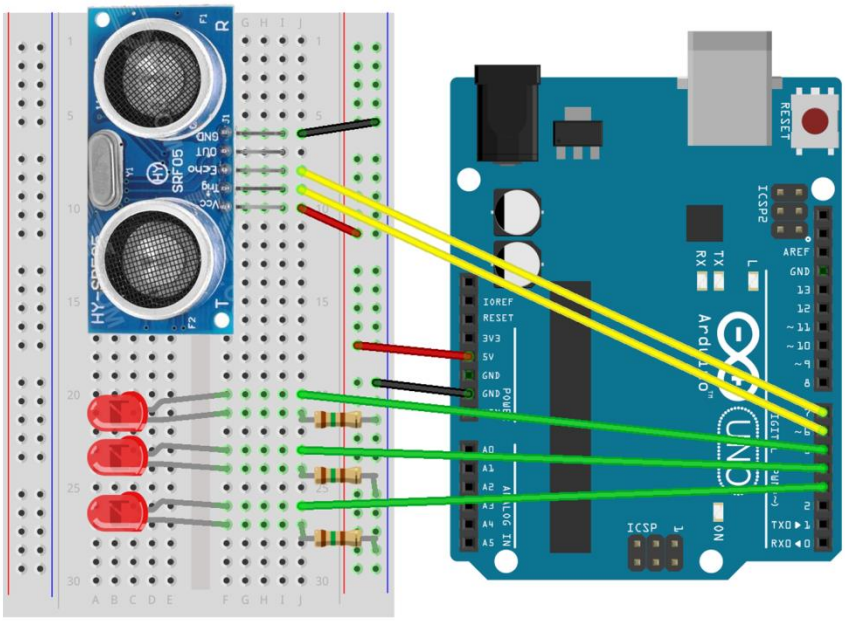
```
// konverterer tid til distance
  distanceCm = duration / 29.1 / 2 ;  //Beregner distancen.
//Burde være: duration / 34 / 2;
//Men 29.1 var en bedre kalibrering med min sensor.


if (distanceCm <= 0){
    Serial.println("Out of range");
  }
  else {
    Serial.print(distanceCm);
    Serial.print("cm");
    Serial.println();
  }
  delay(1000);
}
```

Serial monitoren med output på afstanden:



Opgave 9: Sonar blinklys

I denne opgave skal vi bruge en sonar sensor (HY-SRF05) og tre dioder til at angive hvor tæt sensoren er på en genstand.



1. Forbind sonar sensoren som vist på tegningen. Echo til pin 7, Trig til pin 6, Vcc til 5v og GND til jord.
2. Forbind 3 LED dioder med plus ben til henholdsvis port 3, 4 og 5, og minus ben forbindes med en 150 ohms modstand til jord.
3. Der skal ændres en smule i programmet fra opgave 10, men ikke meget. Kan I selv omskrive det? Ellers så prøv at plukke de rigtige ting ud af nedenstående:

```
//Antager 20 grader celcius rumtemperatur
const int TRIG_PIN = 6;
const int ECHO_PIN = 7;
const int ledPIN1 = 3;
const int ledPIN2 = 4;
const int ledPIN3 = 5;


void setup() {
  // initialiser serial kommunikation:
  Serial.begin(9600);
  pinMode(TRIG_PIN,OUTPUT);
  pinMode(ECHO_PIN,INPUT);
  pinMode(ledPIN1,OUTPUT);
  pinMode(ledPIN2,OUTPUT);
  pinMode(ledPIN3,OUTPUT);
}
void loop()
{
   long duration, distanceCm;
  // Laver en kort lav puls først, for derefter at lave en ren
høj 10ms puls:
  digitalWrite(TRIG_PIN, LOW);       //Sætter den lav
  delayMicroseconds(2);              //Vent 2ms
  digitalWrite(TRIG_PIN, HIGH);      //Sætter den høj
  delayMicroseconds(10);             //Venter => 10ms lang puls
  digitalWrite(TRIG_PIN, LOW);       //Sætter den lav
  duration = pulseIn(ECHO_PIN,HIGH);//Venter på, og læser
puls-længde på echo-pin.
```

```
// konverterer tid til distance
  distanceCm = duration / 29.1 / 2 ;  //Beregner distancen.
//Burde være: duration / 34 / 2;
//Men 29.1 er en bedre kalibrering.

  if (distanceCm > 0 && distanceCm <= 10){
    digitalWrite(ledPIN1, HIGH);
    digitalWrite(ledPIN2, HIGH);
    digitalWrite(ledPIN3, HIGH);
  }
  else if (distanceCm > 10 && distanceCm <= 20){
    digitalWrite(ledPIN1, HIGH);
    digitalWrite(ledPIN2, HIGH);
    digitalWrite(ledPIN3, LOW);
  }
  else if (distanceCm > 20 && distanceCm <= 30){
    digitalWrite(ledPIN1, HIGH);
    digitalWrite(ledPIN2, LOW);
    digitalWrite(ledPIN3, LOW);
  }
  else{
    digitalWrite(ledPIN1, LOW);
    digitalWrite(ledPIN2, LOW);
    digitalWrite(ledPIN3, LOW);
  }
  delay(100);
}
```

Else if –funktion

Hvis man har brug for mere end én if, men hvor de andre er betinget af udfaldet af første kan man indskyde en ”else if”, eller ”ellers, hvis” funktion imellem en if og en else.

Formen kan ses i eksemplet herover. Her er formålet at dioderne skal tændes alt efter hvor langt væk nærmeste genstand er på sensoren. If – else if – else, er opbygget i lag, og man kan kun komme til næste del hvis den forrige betingelse var falsk. Betingelserne kunne i dette tilfælde derfor godt forkortes til

```
if (distanceCm > 0 && distanceCm <= 10){
}
else if (distanceCm <= 20){
}
else if (distanceCm <= 30){
}
else{
}
```

Da det der underforstået at afstanden efter if ikke er under 10, og efter første else if ikke er under 20, osv.

Det kan dog hjælpe til overskueligheden at have det med i koden alligevel.

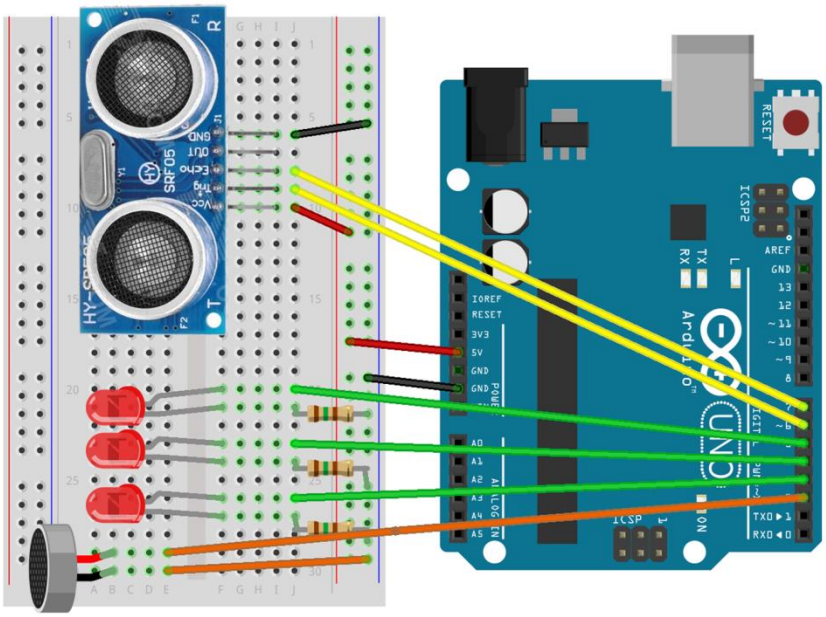
En alternativ funktion til if – else if, hvor man vil have flere mulige udfald afhængig af tilstanden på en variabel er at bruge en Switch Case.

https://www.arduino.cc/en/Reference/SwitchCase

Et eksempel på brugen af en else if kunne være, at du skal bruge noget brød. Hvis du har mel og mælk så bager du noget, ellers hvis bageren er åben går du derned og køber noget, eller hvis nærmeste dagligvarebutik er åben går du derned og køber noget, eller hvis . . .

Opgave 10: Sonar blinklys med alarm

I denne opgave skal vi bruge en sonar sensor (HY-SRF05) og tre dioder til at angive hvor tæt sensoren er på en genstand og en piezo buzzer der alarmerer når genstanden nærmer sig.



1. Forbind sonar sensoren som vist på tegningen. Echo til pin 7, Trig til pin 6, Vcc til 5v og GND til jord.
2. Forbind 3 LED dioder med plus ben til henholdsvis port 3, 4 og 5, og minus ben forbindes med en 150 ohms modstand til jord. Forbind nu også en buzzer til henholdsvis jord og pin 2.
3. Der skal ændres en smule i programmet fra opgave 11, men meget lidt. Kan I selv omskrive det?
4. Det er kun nødvendigt at tilføje henholdsvis tone() og noTone() i if, else if og else funktionerne, for at få buzzeren til at afgive toner der stiger i alarmeringsgrad.
5. Se tilføjelserne i nedenstående, og leg selv med at lave både mere og mindre larmende toner og bip mønstre ☺

```
if (distanceCm > 0 && distanceCm <= 10){
    digitalWrite(ledPIN1, HIGH);
    digitalWrite(ledPIN2, HIGH);
    digitalWrite(ledPIN3, HIGH);
    tone(2, 5000,50);
    delay(50);
    tone(2, 5000,50);
  }
  else if (distanceCm > 10 && distanceCm <= 20){
    digitalWrite(ledPIN1, HIGH);
    digitalWrite(ledPIN2, HIGH);
    digitalWrite(ledPIN3, LOW);
    tone(2, 1000);
  }
  else if (distanceCm > 20 && distanceCm <= 30){
    digitalWrite(ledPIN1, HIGH);
    digitalWrite(ledPIN2, LOW);
    digitalWrite(ledPIN3, LOW);
    tone(2, 200);
  }
  else{
    digitalWrite(ledPIN1, LOW);
    digitalWrite(ledPIN2, LOW);
    digitalWrite(ledPIN3, LOW);
    noTone(2);
  }
  delay(100);
}
```