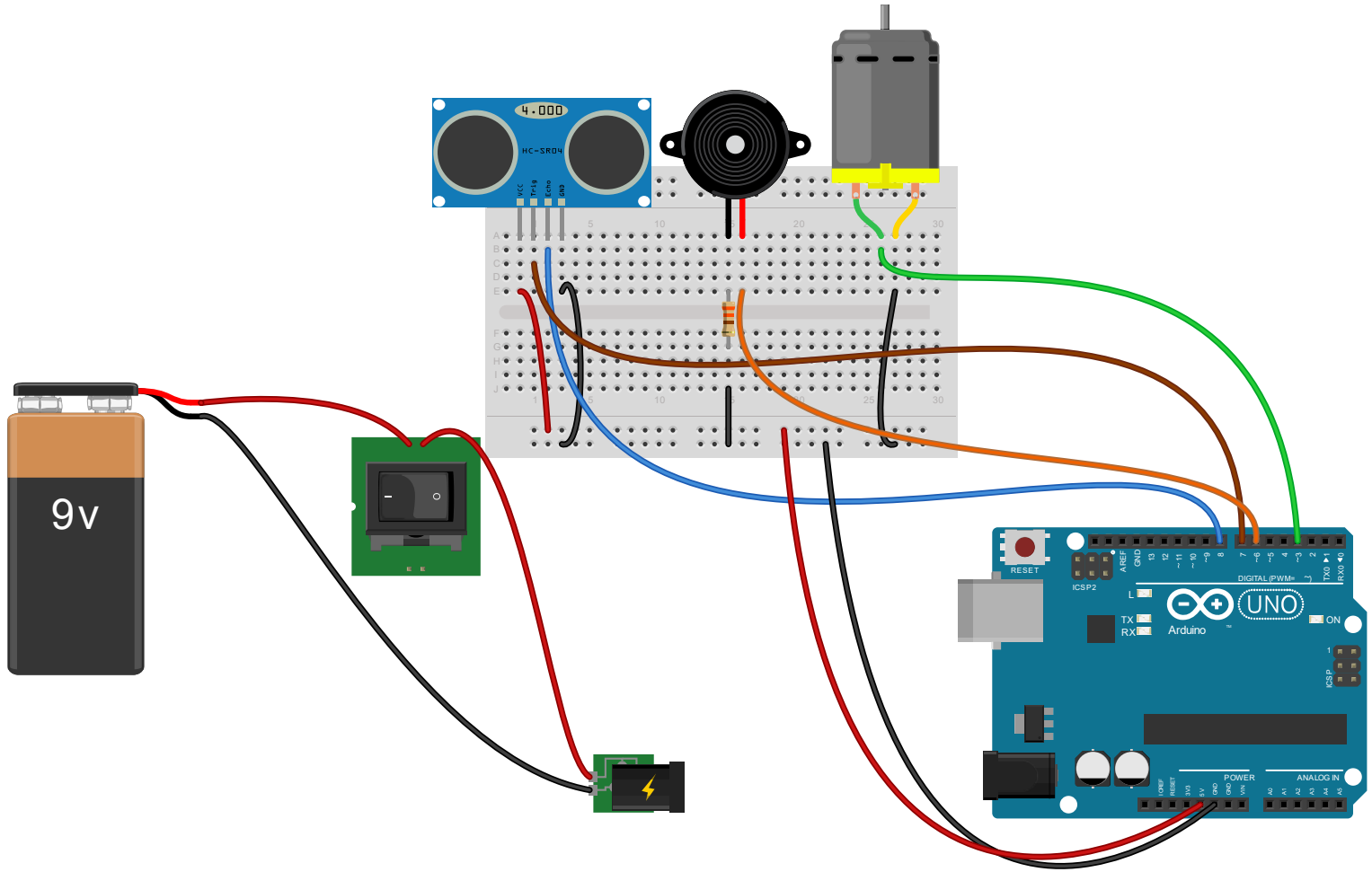
Sensorer og aktuatorer

Arduinoforelesning 3



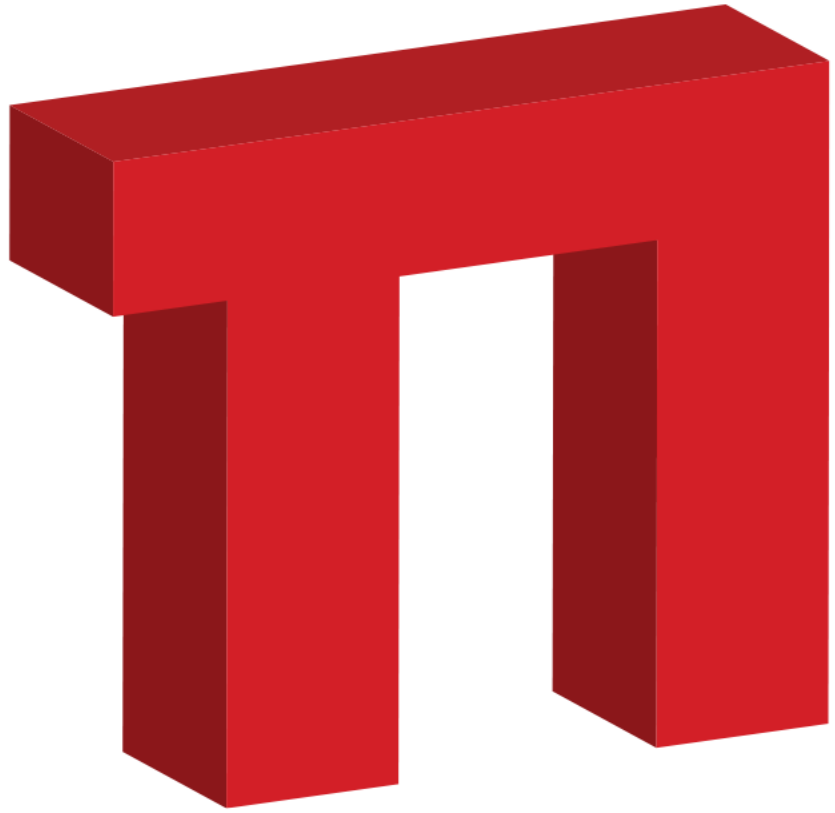
I dag:

- Beskjeder
- Arduinoprogrammering: String(), løkker og arrays
- Pullup og pulldown, repetisjon
- Sensorer og aktuatorer med Arduino
  - Bruk av sensorer og aktuatorer i settet, eksempler fra IN1060-prosjekter
  - Komponenter som ikke er i settet:
    - Hva finnes?
    - Hvor kan vi få tak i dem?
    - Hvordan bruke dem?
      - Bruke nye biblioteker
      - Lese dokumentasjon

Beskjeder

Invitasjon til Tangible Interaction utstilling på fredag

Prosjekt-kickoff neste uke



TANGIBLEINTERACTION

FRIDAY 16 FEBRUARY 2024, 12:00. 7TH FLOOR. WELCOME ALL.

RESONANCE

ECHOES OF NEBULAS, PULSES, AND PAIN

AXEL, BERNADETTE, CHRISTOPHER, EMI, LINE, LIVE, MARTE, MELISSA, OLA, ROY ANDRÉ, SANNA, SNORRE, TIFFANY, TORA, ZANDER

Ekstra nyttig for IN1060-studenter

Fredag 16. februar kl. 12.00 i 7. etg
(vi holder på til ca. 13.30)

Kom å se eksempler på:

- Bruk av Arduino i designprosjekter med tilhørende sensorer og andre komponenter.
- Bruk av øvrige prototypingsteknikker som 3D printing, laserkutting, trykkforming etc.
- Bruk av utstilling som en metode for å formidle, evaluere og regenerere designideer på.

Få en snikktitt oppe i 7. etg hvor ansatte, stipendiater og masterstudenter holder til samt inne på laben vår.

Se hvordan masterstudenter hos oss jobber med sine designprosjekter og masteroppgaver.

Det blir enkel matservering til de som møter opp kl. 12.

Prosjekt-kickoff!

Prosjekt-kickoff neste uke!

- Tirsdag og onsdag 10.00-12.00
- Møt opp der hvor forelesning og orakel vanligvis er klokka 10.15
  - NB! 3D-printekurs starter klokka 10.00 på Sonen begge dager!
- Store aud. tirsdag, Caml onsdag.

- Mål:
- Bli kjent med potensielle gruppemedlemmer
- Lære nyttige tips og triks for prosjektet

Arduinoprogrammering

String()

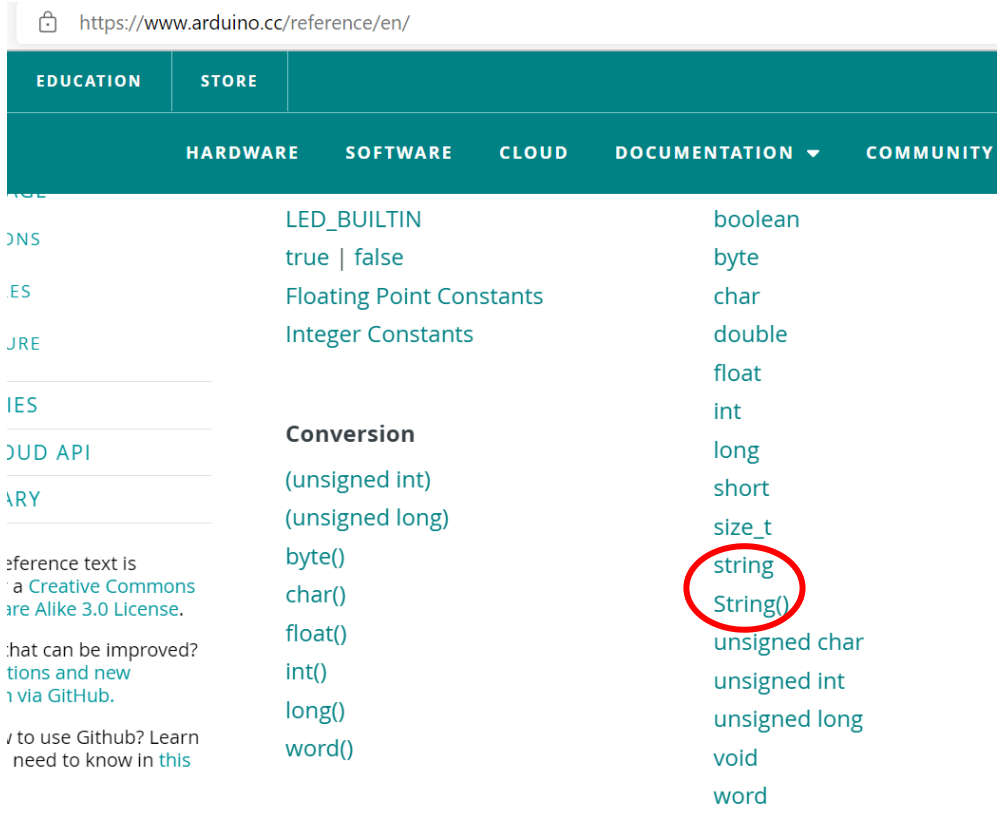
String()

Arduino har et eget String()-objekt

I Arduino language reference finner vi både String og String() under variabler (datatyper)

string er C-programmering. I C må man lage hver string som en array med forhåndsdefinert størrelse

String() er Arduinos innebygde String-objekt med en rekke funksjoner vi kan bruke for å lage, manipulere og sjekke strenger, slik vi er vant til fra Python og Java



Lage et String()-objekt i Arduino

1: Lage en variabel med String som datatype

```
String variabelNavn = «Dette er en streng»;
```

2: Bruke String-klassens konstruktørmetode og konvertere variabel til String-objekt:

```
String variabelNavn = String(‘a’); //et tegn
String variabelNavn = String(«En streng»); //en streng av tegn
String variabelNavn = String(13); //integer – String() kan lages med datatyper som int, float og long
String variabelNavn = String(variabelNavn + " mer");
// konkatenerer to strenger
```

Bruke strenger

```
String variabelNavn = «Dette er en streng»;
variabelNavn = «her skriver vi noe nytt»;
```

String()-funksjoner for tekst og tegn

Operatører i String()

+= (append) legger til data på slutten av strengen
Legger sammen to strenger på samme måte som concat()

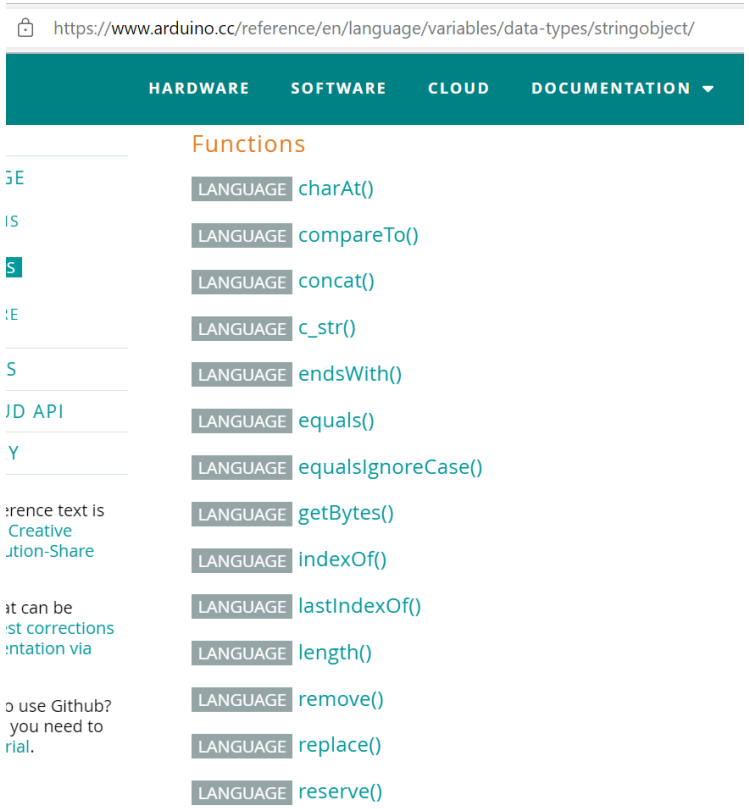
Funksjoner i String()

equals() sammenligner strenger

```
String streng1 = «tekst»;
String streng2 = «Tekst»;
If(streng1.equals(streng2){
Serial.println(«streng1 er lik streng 2»);
```

Ignorer forskjeller i store og små bokstaver med equalsIgnoreCase

```
Streng1.equalsIgnoreCase(streng2);
```



String()-funksjoner for tekst og tegn

substring(from, to) får tak i del av en streng

streng1.substring(from); //fra indeks

streng1.substring(from, to); //indeksen *etter* substringen slutter

delAvStreng = streng1.substring(3, 7);

NB! Det er smart å sjekke lengden på strengen før man prøver å hente verdier fra indekser

```
If(lengde >= to){
//hent substring
}
```

length() sjekker lengden på en streng

streng1.length();

Int lengde = streng1.length();

Løkker og arrays

Løkker og arrays

Løkker

Løkker lar oss repetere kodeblokker til en bestemt betingelse blir møtt eller et bestemt antall ganger

Forenkler repeterende oppgaver

Færre kodelinjer å skrive og lese

Bra for å behandle eller gå gjennom mange verdier

Tre typer løkker i Arduino

while-løkker

do-while-løkker

for-løkker

While-løkker

Utfører kodeblokken inne i klemmene så lenge (while) betingelsen som sjekkes er sann/true.

Sjekker ofte en variabel som oppdateres inne i kodeblokken for å kontrollere nøyaktig hvor lenge kodeblokken repeteres

Kan også sjekke f.eks. sensor-input

NB! Hvis ingenting endres og betingelsen aldri blir usann/false vil man få en evig løkke

Syntaks

```
while(betingelse){
        //gjør noe
}
```

Eksempel

```
void loop(){
variabel = 0;
// gjør dette 200 ganger:
while (variabel < 200) {
//tar verdien i variabelen og legger til 1
variabel++;
}
```

do-while-løkker

Kjører kodeblokken først og sjekker deretter om betingelsen er sann

Det sikrer at kodeblokken kommer til å kjøre minst en gang

Syntaks

```
do {
// statement block }
while (condition);
```

Eksempel

```
int x = 0;
do {
delay(50); // wait for sensors to stabilize
x = readSensors(); // check the sensors
}
while (x < 100);
```

for-løkker

«alt-i-ett-løkke»

Sjekker betingelser og oppdaterer variabler i betingelsen

Kan være mer oversiktlig å bruke enn while og do-while-løkkene

Syntaks

```
for (initialization; condition; increment) {
// gjør noe; }
```

Eksempel: Dimme en LED med en løkke

```
int ledPin = 10;
void setup() {
pinMode(ledPin, OUTPUT);
}
void loop() {
//(255 er maks verdi for analog ut-pin)
for (int i = 0; i <= 255; i++) {
analogWrite(ledPin, i);
delay(10);
}
}
```

De tre parametrene til for-løkker

Syntaks

```
for (initialization; condition; increment) {
// gjør noe;
}
```

Eksempel

```
for (int i = 0; i <= 255; i++) {
// gjør noe:
}
```

Parametere

Initialization utføres kun en gang

Condition betingelsen testes. Hvis den er true går programmet videre og utfører increment og resten av kodeblokken, hvis den er false stopper den og går ut av løkken her

Increment oppdaterer verdien før kodeblokken kjører igjen

Arrays

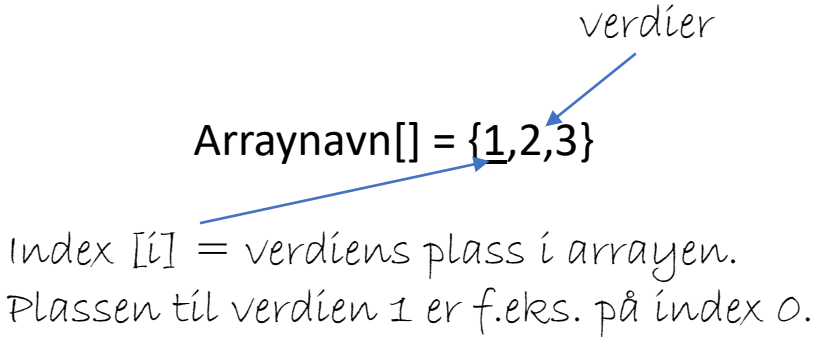
Arrays lar oss samle mange verdier under ett variabelnavn

Forenkler håndtering av store mengder data

Syntaks (flere muligheter for å deklarere arrays)

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[3] = {2, 4, -8};
char message[6] = "hello";
```

Arrays



Legg til verdi i array

arrayNavn[0] = 11;

Hent en verdi fra en array

verdi = arrayNavn[0]; //verdien er 11

NB! Vær oppmerksom på at det å hente verdier i array-plasser som ikke finnes gjør at Arduinoen henter ut info fra andre steder i minnet. Dette kan føre til feil som er vanskelige å feilsøke.

For eksempel:

Arraynavn[] = {1,2,3}

I denne arrayen er det 3 plasser, men det høyeste index-tallet er 2 fordi indexen begynner på 0!

Hvis vi prøver å hente ut

Verdi = arrayNavn[3];

vil Arduinoen hente ut informasjon fra et annet sted i minnet, fordi arrayNavn[3] ikke finnes.

Arraynavn[] = {1,2,3}

verdi	**1**	**2**	**3**
index	0	1	2

Arrays og løkker

For at løkker skal kunne gå gjennom mange ulike verdier må verdiene være samlet på ett sted

Arrays lar oss samle mange verdier under ett variabelnavn

Arrays kan manipuleres i for-løkker ved å bruke *counter* som *index* for hver verdi i arrayen

Eksempel: skrive ut alle verdiene i arrayen myPins[]

```
for (byte i = 0; i < 5; i = i + 1) {
Serial.println(myPins[i]);
}
```

Array eksempel: deklarere mange ledPins

Deklarere flere ledPiner:

```
int led1 = 2;
int led2 = 4;
int led3 = 8;
int led4 = 9;
int led5 = 10;
int led6 = 13;
```

Deklarere ledPin'er med array:

```
int ledPins[] = {2,4,8,9,10,13};
```

Array og for-løkke eksempel: sette alle ledPinene som OUTPUT

Sette pinMode()

```
void setup() {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
}
```

Sette pinMode() med for-løkke og array

```
void setup() {
  for(int i=0; i < 6; i++){
    pinMode(led[i], OUTPUT);
  }
```

Slå på alle LEDene med for-løkke og array

```
void loop(){
digitalWrite(led1, HIGH);
delay(1000);
digitalWrite(led2, HIGH);
delay(1000);
digitalWrite(led3, HIGH);
delay(1000);
digitalWrite(led4, HIGH);
delay(1000);
digitalWrite(led5, HIGH);
delay(1000);
digitalWrite(led6, HIGH);
delay(1000);
}
```

Med for-løkke og array

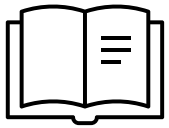
```
void loop(){
for(int i = 0; i < 6; i++){
        digitalWrite(led[i], HIGH);
}
delay(1000);
}
```

I dag

Hva vi kan bruke sensorer og aktuatorer til?

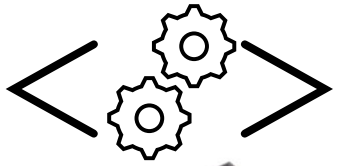
Hvordan kobler vi dem opp og programmerer dem?

lese *signaler fra sensorer*

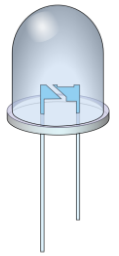


Sensorer er komponenter som tar inn signaler fra omverdenen

gjøre *noe med signalene*
Programmering/ skript



sende *eller skrive signaler ut aktuatorer*



Aktuatorer er komponenter som gjør om energi til bevegelse, som motorer. I vår modell bruker vi begrepet om output

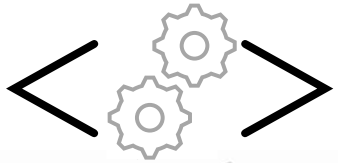
Sensorer

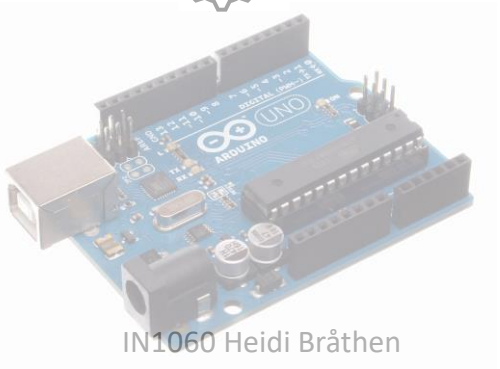
Sensorer er alle komponenter som leser inn signaler fra omverdenen og sender elektroniske signaler inn til Arduinoen

lese *signaler fra sensorer*

Sensorer er komponenter som tar inn signaler fra omverdenen

gjøre *noe med signalene*
Programmering/ skript





sende *eller skrive signaler ut aktuatorer*

Aktuatorer er komponenter som gjør om energi til bevegelse, som motorer. I vår modell bruker vi begrepet om output

Aktuatorer med Arduino

Aktuatorer er komponenter som gjør om energi for å bevege noe

I Arduino og interaksjonsdesign er aktuator en litt bredere betegnelse på all output, noe som gjør om energi til å «gjøre» noe, ikke bare bevege noe

Krever et styringssignal (elektrisk signal) og en energikilde (elektrisk strøm)

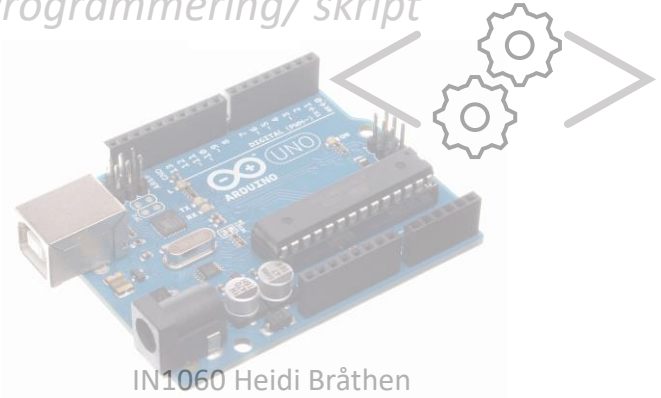
Når aktuatoren får et signal svarer den med å gjøre om elektrisk strøm til en handling («gjør noe»)

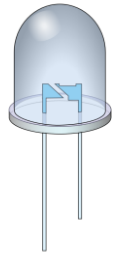
***lese** signaler fra sensorer*

***gjøre** noe med signalene*
Programmering/ skript

***sende** eller skrive signaler ut*
aktuatorer

Sensorer er komponenter som tar inn signaler fra omverdenen





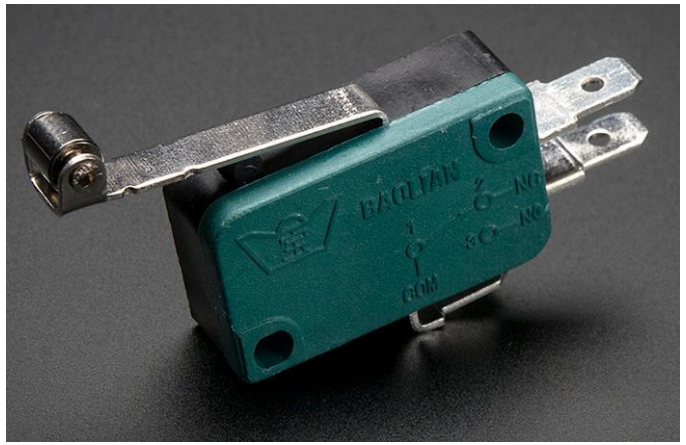
Aktuatorer er komponenter som gjør om energi til bevegelse, som motorer. I vår modell bruker vi begrepet om output

Ulike typer komponenter for Arduino

Sensorer: knapper/brytere

Det finnes mange typer brytere

Knapper er brytere, de slutter og bryter kretser.
Alle disse er digitale; de er enten av/0 eller på/1



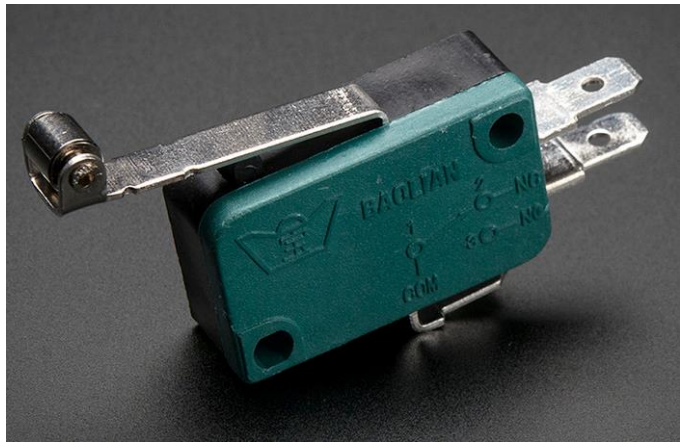
Vi kan lese av tilstanden til sensorer som knapper og andre brytere
- og bruke tilstanden til å bestemme hva som skal skje med aktuatorene

For eksempel:
Knapper kan være av eller på, og slå LED av eller på

Det finnes mange typer brytere

Knapper er brytere, de slutter og bryter kretser.
Alle disse er digitale; de er enten av/0 eller på/1

Knapper
Kalles ofte tactile buttons



Limit switch
Brukes i 3D-printere og for å slå på lys når skapdører åpnes

Brytere
«toggle switch» holder tilstand AV/PÅ ved hjelp av mekanisk oppsett

Tilt switch
Virker ved at en metallkule eller kvikksølvboble(!) kobler sammen en + og minuspol inne i knappen når vi snur på den

Vi har allerede sett på hvordan knapper og LED kobles og programmeres

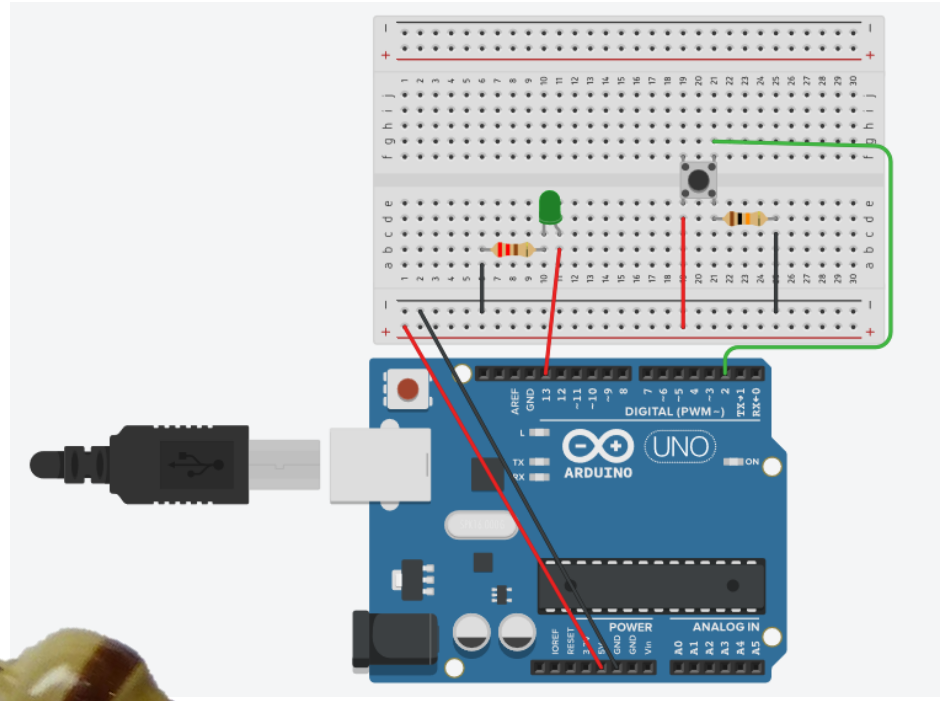
Pushbutton (bryter)

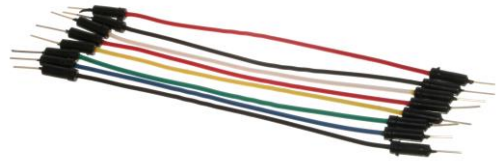
Ekstra ledning i valgfri farge

10k resistor

Eksempelskisse fra Arduino:

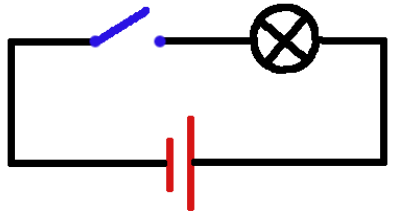
last opp fra Fil – Eksempeler – 02 Digital – Button



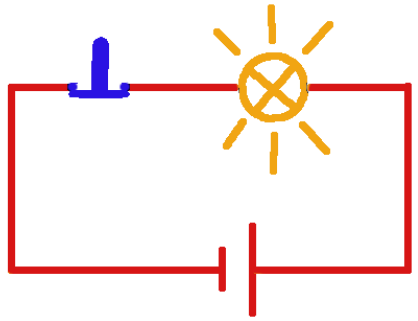


```
*/ Eksempelskisse: Button (rep.)
const int buttonPin = 2;     // the number of the pushbutton pin // constants won't change.
const int ledPin =  13;      // the number of the LED pin
int buttonState = 0;         // variable for reading the pushbutton status // variables will change:

void setup() {
pinMode(ledPin, OUTPUT); // initialize the LED pin as an output:
pinMode(buttonPin, INPUT); // initialize the pushbutton pin as an input:
}

void loop() {
buttonState = digitalRead(buttonPin); // read the state of the pushbutton value:
if (buttonState == HIGH) {// check if the pushbutton is pressed. If it is, the buttonState is HIGH:
        digitalWrite(ledPin, HIGH); // turn LED on:
                } else {
                digitalWrite(ledPin, LOW); // turn LED off:
  }
}
```

Knapper: hvordan virker de?



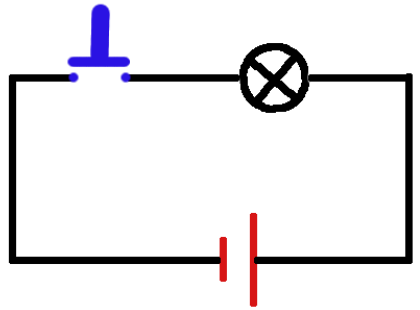
En bryter er et komponent som kobler sammen
to punkter i en krets (slutter kretsen) når den trykkes ned
og bryter kretsen når den slippes (eller omvendt)



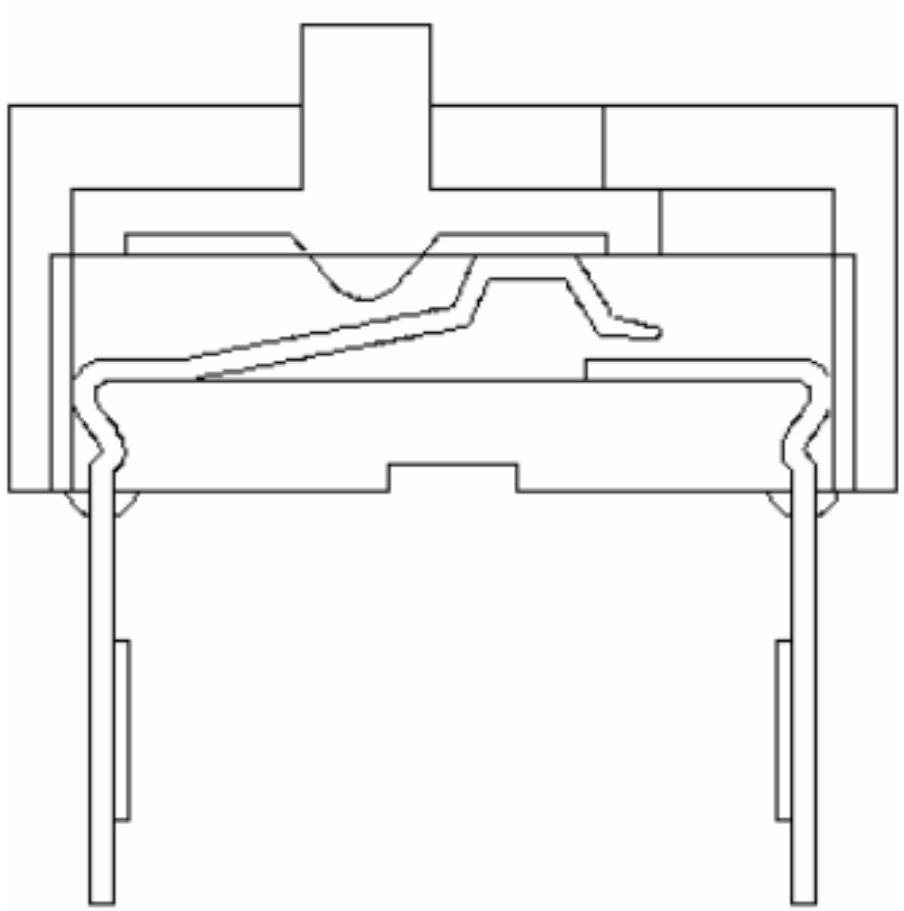
Knapper (pushbuttons) er eksempler på brytere

Enkle mekanismer: metall som kommer i kontakt for å slutte en krets

Ben 1 og 3 er koblet sammen og ben 2 og 4 er koblet sammen



Resistorer og knapper



Knappen er en bryter som setter to metallpinner i kontakt med hverandre når vi trykker den ned. Den er som en ledning i kretsen.

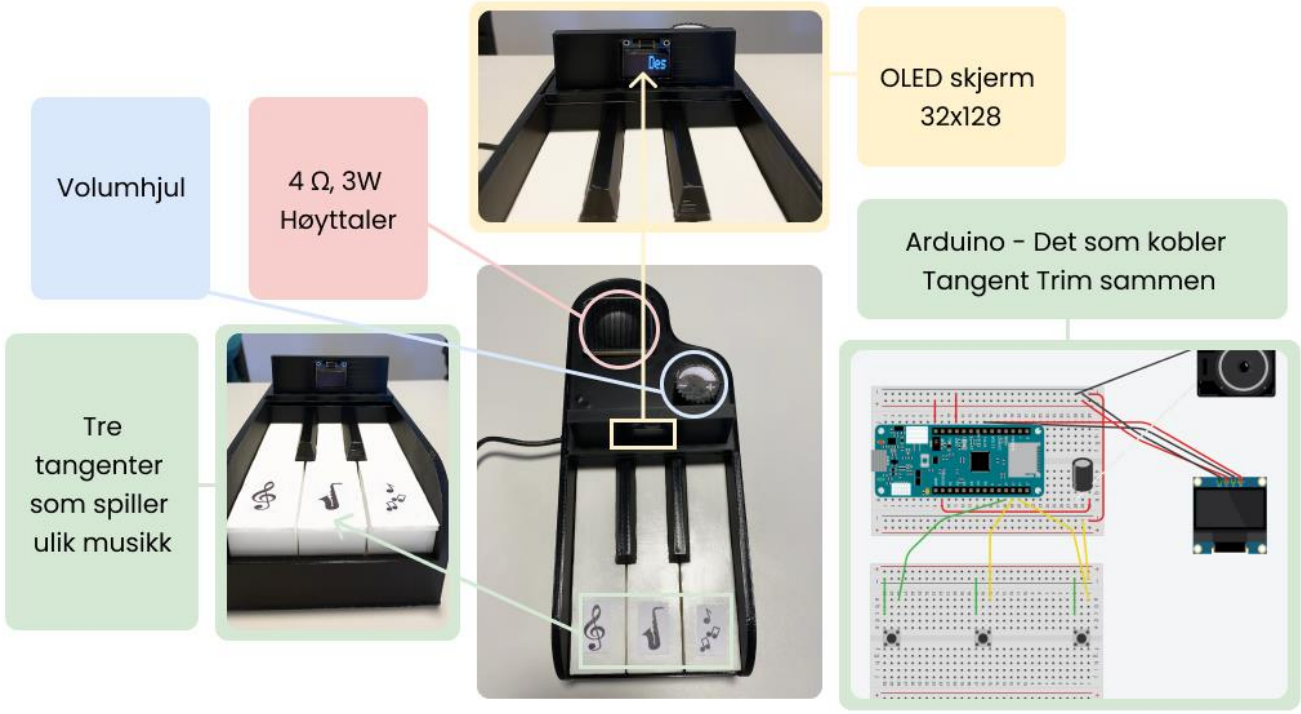
Hvis det ikke er noen andre komponenter i kretsen får vi en direkte linje fra pluss til minus og risikerer kortslutning

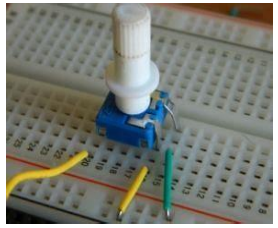
En resistor legger til ett komponent i kretsen, og motvirker en direkte krets fra pluss til minus

Eksempel: Tangent trim av Designerne.ino, 2023

Video:
https://vimeo.com/831218073

Prosjektside:
https://www.uio.no/studier/emner/matnat/ifi/IN1060/v23/prosjekter-var-2023/designerne.ino/





Sensorer

Knapper (tangenter)

Potensiometer (Volumhjul)

Aktuatorer

OLED skjerm

Høytaler

To vanlige problemer med knapper

Switch bounce og floating signals/pins

Serial: et verktøy for problemløsning (rep.)

Arduinoen har et verktøy som lar oss se signalene som går ut og inn i form av tekst

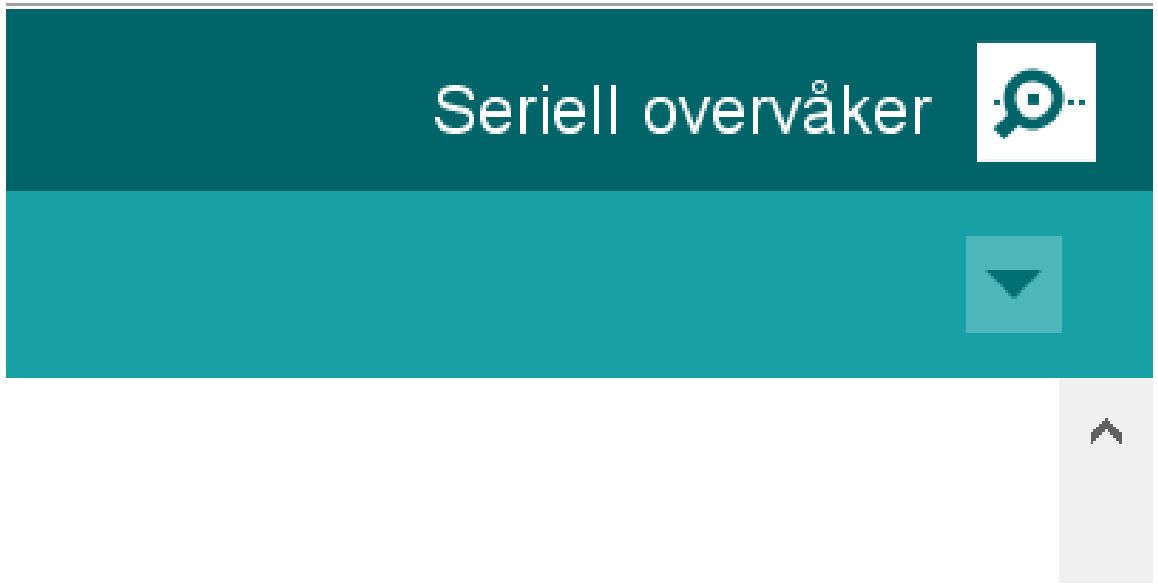
Gå til:

Verktøy – seriell overvåker eller ctrl+shift+m eller ikon øverst til høyre

Vises som popup-vindu

Seriell overvåker (serial monitor)

Skriver ut returverdier fra funksjonene når programmet kjører i form av tall eller tekst



Funksjoner for seriell kommunikasjon #1 (rep.)

For å starte seriell kommunikasjon må vi initialisere den med

Serial.begin(9600)

Serial.begin()-funksjonen tar inn en variabel for baud rate.

Baud rate refererer til hastigheten på kommunikasjonen over en datakanal. På Arduino er det vanlig å sette den til 9600.

Serial.print()- funksjonen lar oss skrive ut beskjeder til overvåkeren og er nyttig for å følge med på hva som egentlig skjer i programmet

```
DigitalReadSerial eksempelskisse

int pushButton = 2;

void setup() {
Serial.begin(9600); // start seriell
kommunikasjon ved 9600 bits per
sekund
pinMode(pushButton, INPUT);
}

void loop() {
int buttonState =
digitalRead(pushButton);
Serial.println(buttonState); //skriv
ut tilstanden til knappen
  delay(1); //pause mellom
avlesningene for stabilitet
}
```

Vanlig problem med brytere #1: Sprettende input (rep)

Switchbounce

Når vi trykker ned knappen tar det noen millisekunder før metallbitene kommer i stabil kontakt med hverandre

Fordi loop() kjører veldig fort klarer Arduinoen å registere flere knappetrykk idet vi trykker ned knappen

Løsning: Debounce

Vi kan programmere Arduinoen til å vente noen millisekunder og dermed kun registrere ett av trykkene

Debounce ved å bruke delay()

Delay() pauser programmet i millisekunder: delay(antall millisekunder)

Debounce med delay() rep.

Trykk på knappen, må du holde inn en liten stund for å få LEDen til å lyse stabilt?

Hva skjer når du trykker knappen raskt inn og slipper?

Se på seriell overvåker mens du trykker ned knappen

```
*/ Eksempelskisse: Button

const int buttonPin = 2;
const int ledPin =  13;
int buttonState = 0;

void setup() {
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
}

void loop() {
 buttonState = digitalRead(buttonPin);
if (buttonState == HIGH) {
          delay(500);
          digitalWrite(ledPin, HIGH);
          } else {
                    delay(500);
                    digitalWrite(ledPin, LOW);
          }
}
```

Vanlig problem #2: Flaksende og flytende input

«Flagrende» eller flytende input er et vanlig problem når vi bruker brytere i Arduino-kretser. Signalet på porten knappen er koblet til settes vilkårlig til 5V/HIGH eller 0V/LOW, og det knappen skal kontrollere oppfører seg uforutsigbart eller virker ikke i det hele tatt

Dette er et problem som kan ta mye tid å løse hvis man ikke kjenner til fenomenet med flytende input..

Løsning:

Arduino har en innebygget pullup i form av en innebygget konstant: INPUT_PULLUP som brukes i stedet for INPUT i pinMode(). Den kan løse dette problemet.

Løsning: Pullup

Hva er pull up?

Pullups og pulldowns setter signalet til høyt(-up) eller lavt(-down)

Hvordan bruke innebygget pullup:

Vi kan sette signalet stabilt ved åpen krets/knappen ikke trykket ned med den innebyggede konstanten INPUT_PULLUP

Den brukes i stedet for INPUT i pinMode()-funksjonen:

pinMode(valgtPin, INPUT_PULLUP);

Hvordan bruke resistor som pull up eller pull down?

En resistor kan brukes som «pull up» som tilbakestiller signalet fra knappen til Arduinoen etter at den er trykket på. Om det er pull up eller down avhenger av plasseringen til resistoren (til 5V eller jord).

Problem: knapp uten noe pullup eller pulldown

Hvis du setter opp knappen uten resistor (se diagrammet) og bruker denne koden >

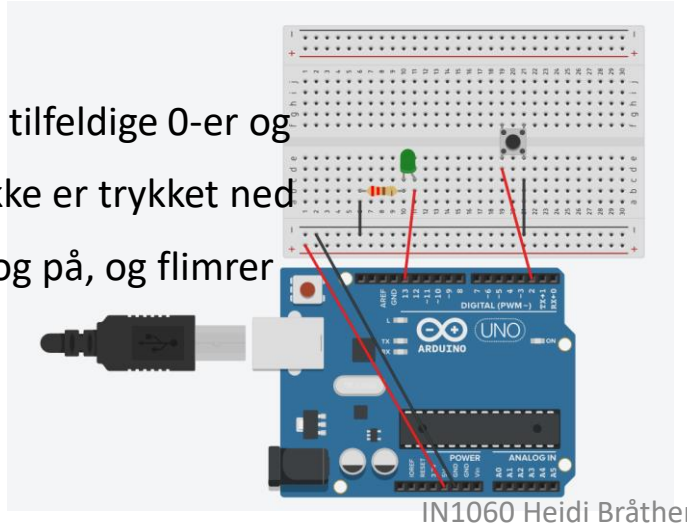
- vil du sannsynligvis se at LEDen flimrer når knappen ikke er trykket ned, og lyser vanlig når knappen trykkes ned

Du kan åpne serial monitor og se på avlesningene når knappen trykkes/ikke trykkes på, for å se hva som skjer med knappen

Knappen vil da vanligvis sende tilfeldige 0-er og

1-ere om hverandre når den ikke er trykket ned

Derfor blir LEDen slått fort av og på, og flimrer



```
int buttonPin = 2;

int Led = 13;


void setup() {

pinMode(buttonPin,INPUT);

pinMode(Led,OUTPUT);

Serial.begin(9600); //starter seriell kommunikasjon

}


void loop() {

//lagrer tilstanden til knappen (HIGH/LOW):

// hvis knappen trykkes ned er den nå i en LOW tilstand

int buttonState = digitalRead(buttonPin); if (buttonState == LOW) {

// LEDen lyser som den skal

digitalWrite(Led,HIGH);

} else {

//når knappen ikke er trykket ned flimrer LEDen

digitalWrite(Led,LOW);

//skriver ut verdien av buttonstate som 0 og 1

Serial.println(buttonState);

}}
```

Hvorfor sender knappen både 0-er og 1-ere når den ikke er trykket ned?

- Når den ikke er trykket ned er kretsen åpen, og uten noen resistor til å jorde den kan den ikke sette en tilstand som av eller på. LEDen får da ulike signaler om å slå seg av og på så fort at den sjelden eller aldri får nok strøm til å lyse klart, den kan stå og blinke veldig fort og uregelmessig. Den åpne kretsen kan reagere på statisk elektrisitet.

- Når knappen blir trykket ned blir kretsen koblet til jord/GND, og da blir signalet satt stabilt til 0. Programmet sier at LEDen skulle lyse hvis knappen var LOW, altså 0, og derfor lyser da LEDen opp som normalt.

Knapp med innebygget pullup i koden

Vi kan sette signalet stabilt ved åpen krets/knappen ikke trykket ned med den innebyggede konstanten INPUT_PULLUP

Den brukes i stedet for INPUT i pinMode()-funksjonen:

pinMode(valgtPin, INPUT_PULLUP);

Denne konstanten setter signalet til å være default 5V, altså er signalet 5V/HIGH når knappen ikke er trykket inn(!)

Litt uvant?

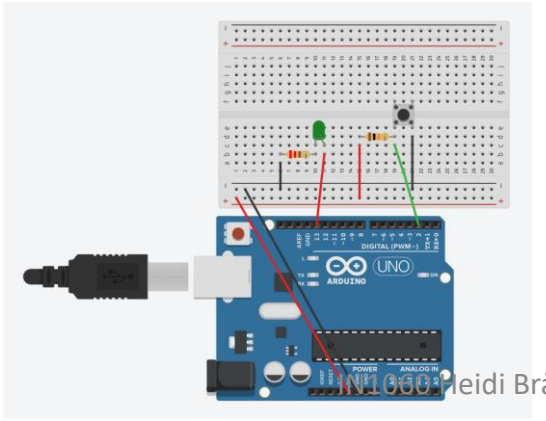
PULLUP med resistor mellom knapp og 5V

Vi kan bruke resistorer i kretsen for å oppnå det samme som med konstanten INPUT_PULLUP

Denne kretsen setter også signalet til å være default 5V, altså er signalet 5V/HIGH når knappen ikke er trykket inn(!)

Merk at signalet til pin 2 leses fra samme side som resistoren

Litt uvant?



```
/* program med pullup-resistor*/
int buttonPin = 2;
int Led = 13;

void setup() {
pinMode(buttonPin,INPUT);
pinMode(Led,OUTPUT);
Serial.begin(9600);
}

void loop() {
int buttonState = digitalRead(buttonPin);
if (buttonState == LOW) {
digitalWrite(Led,HIGH);
Serial.println(buttonState);
} else{
digitalWrite(Led,LOW);
} Serial.println(buttonState);
}
```

PULLDOWN med resistor mellom knapp og jord

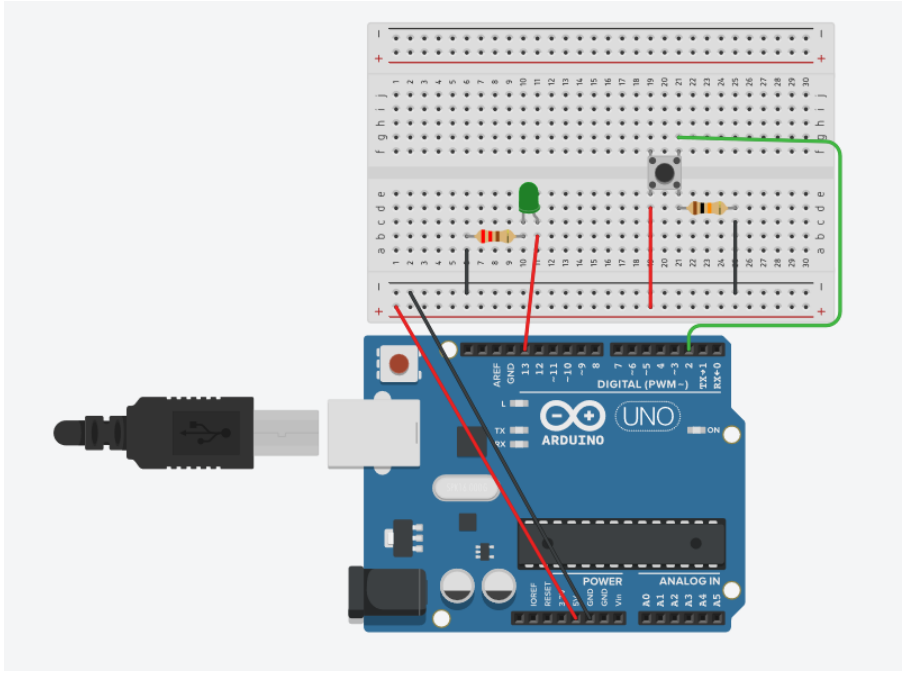
Setter signalet default til 0V

Altså er signalet 0V/LOW når knappen ikke er trykket inn

Merk at signalet til pin 2 leses fra samme side som resistoren

Litt mer som forventet?

Skisse: Button eksempelskisse virker med denne kretsen



Flere sensorer

Sensorer som måler omgivelser

Temperatursensor

- Peltier element: kan overføre varme/kulde
- Heat pad

Lyssensor

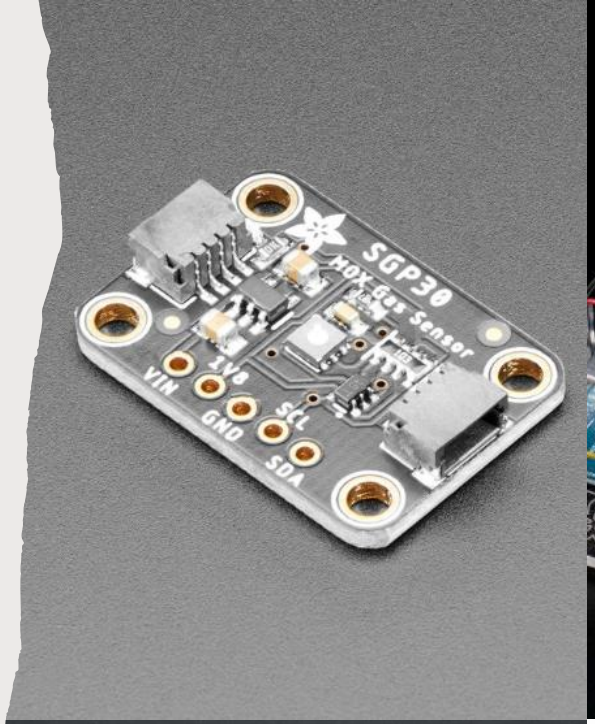
Barometrisk sensor (lufttrykk)

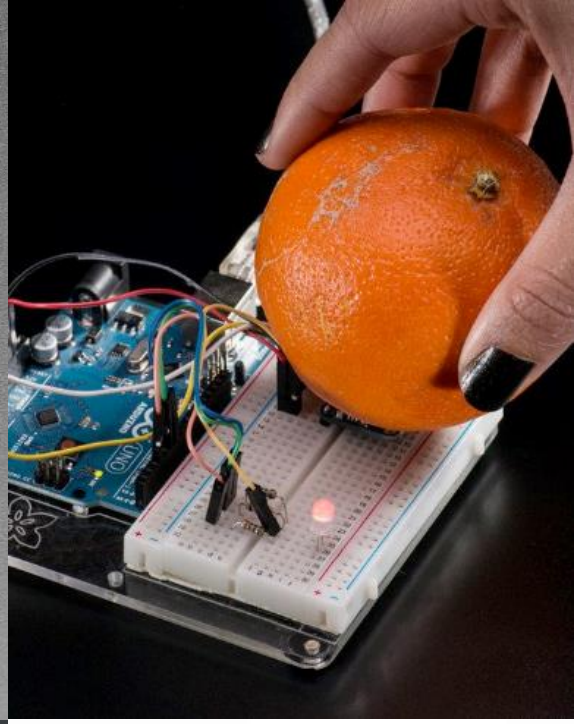
Luftkvalitet (CO2)

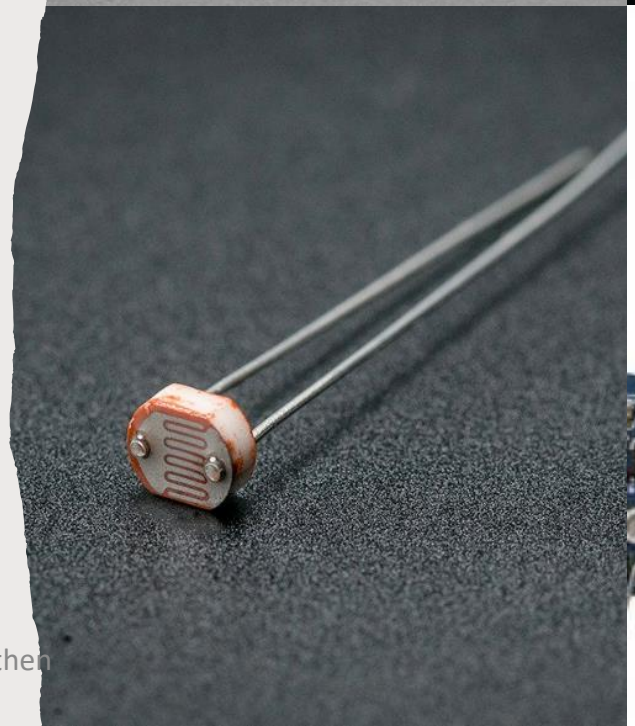
Fuktighetssensor

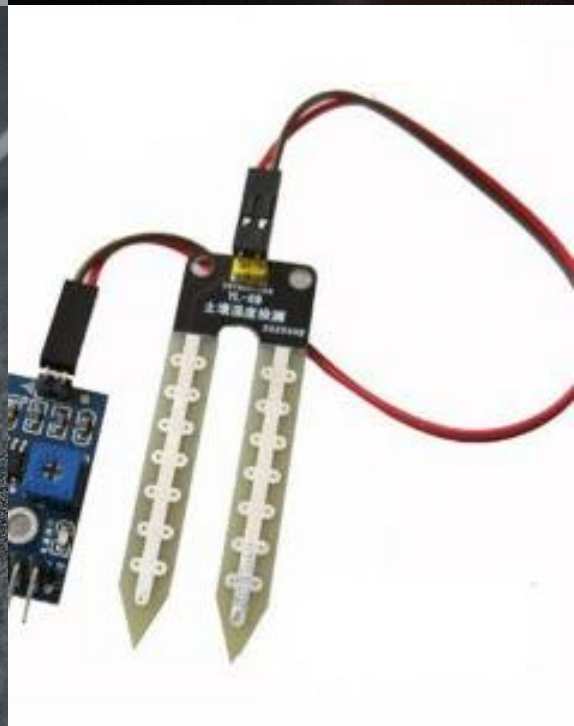
Fargesensor

Atomizer (luftfukter)









Sensorer som måler bevegelse og posisjon

Aksellerometer

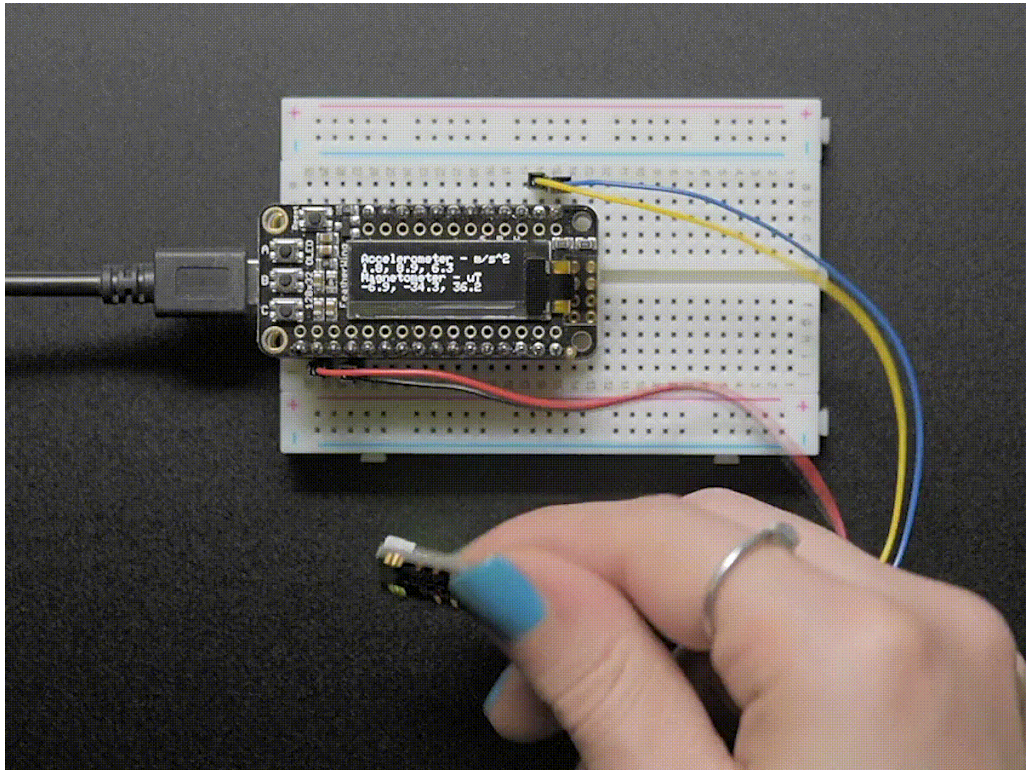
Kompass

Vibrasjonssensor

Aktuatorer som beveger seg

Motorer

Solenoid



Sensorer som kan måle egenskaper ved kroppen

Muskel

Hjerterytme/puls

Skin galvanisation

Gest

Leap

Sensorer og aktuatorer som måler og produserer lyd

Mikrofon

Lydsensor

Høytalere

Piezo (er både lydsensor og høyttaler)

Arduino Uno kan ikke spille av lydfiler, men *skjold* kan utvide med muligheten:

Adafruit music maker

Adafruit wave skjold

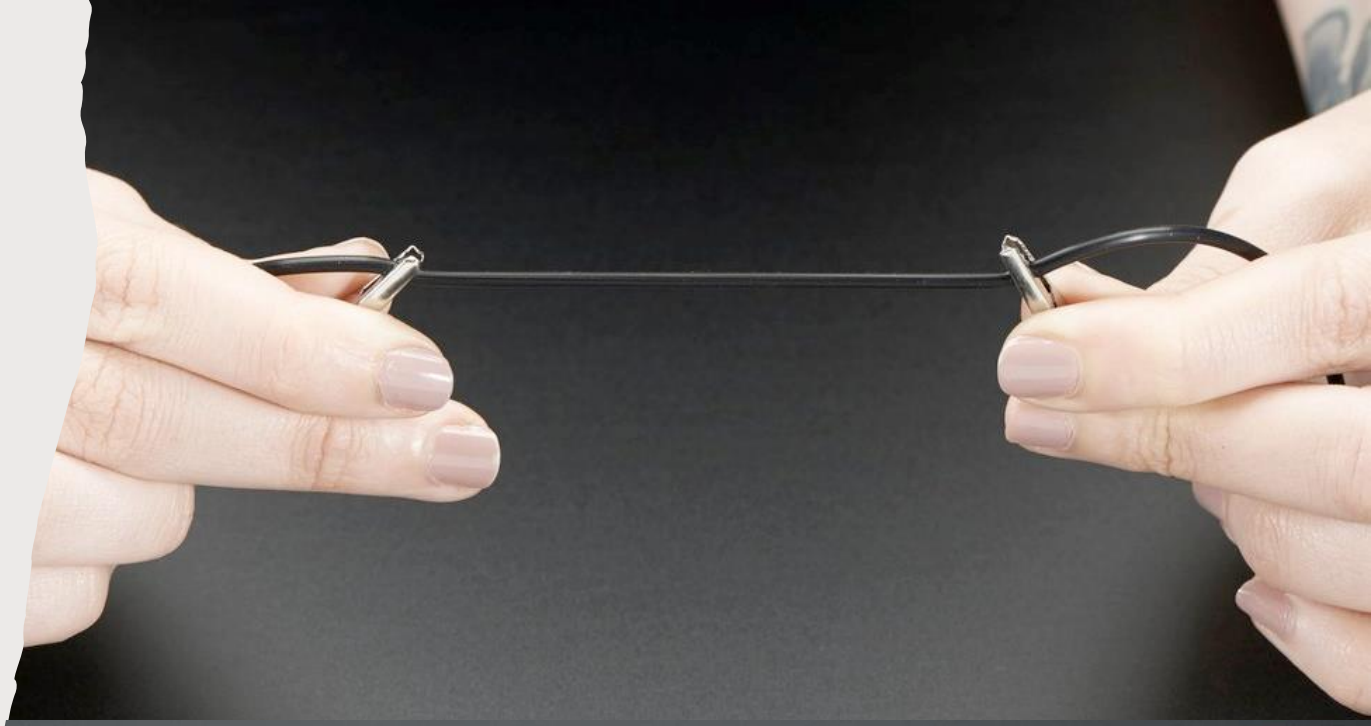
Raspberry Pi

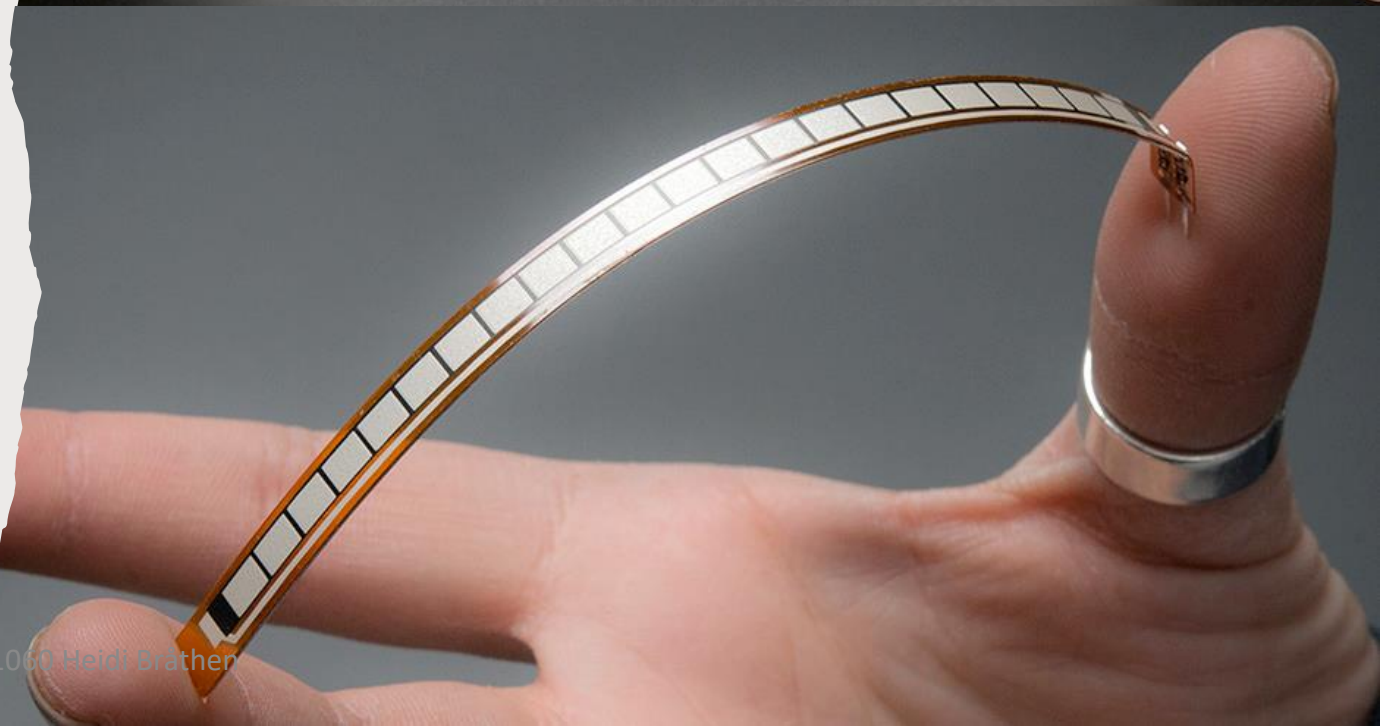
Sensorer som kan bøyes og strekkes

Strekk-sensor av gummistrikk

Fleks-sensor

Elektrisk ledende materialer



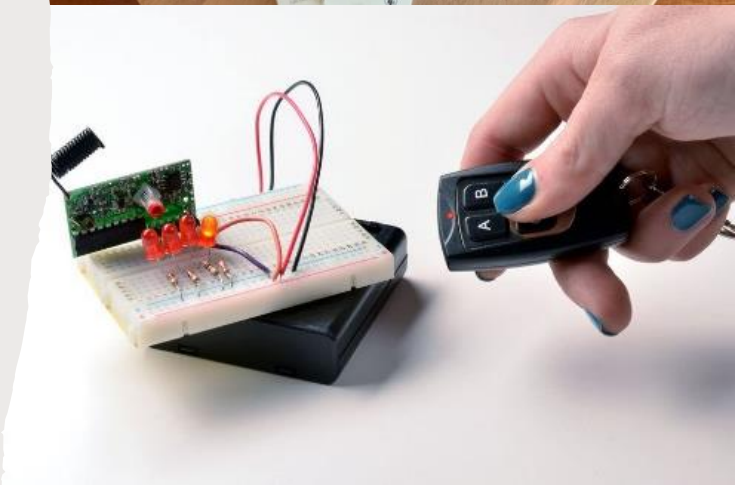


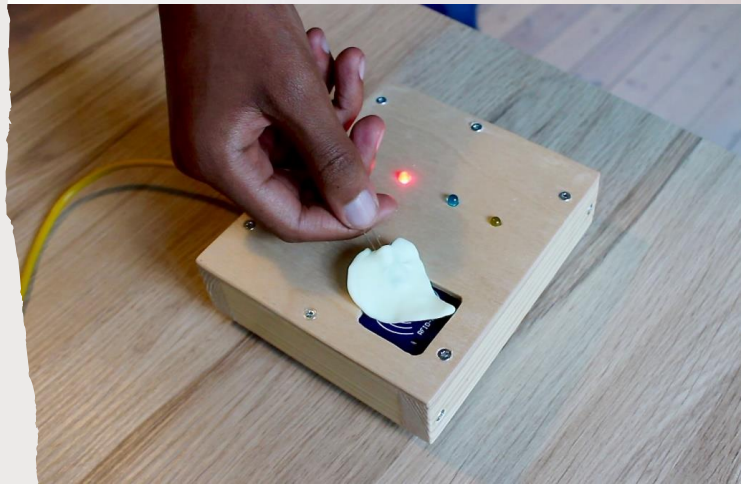
Trådløs kommunikasjon

RFID

NFC

Radiofrekvens



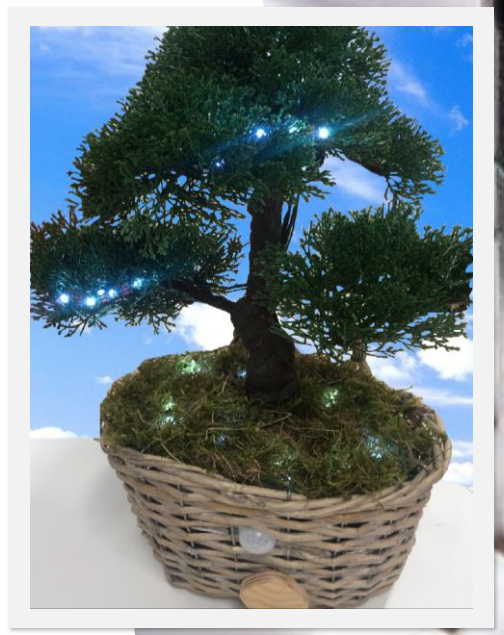


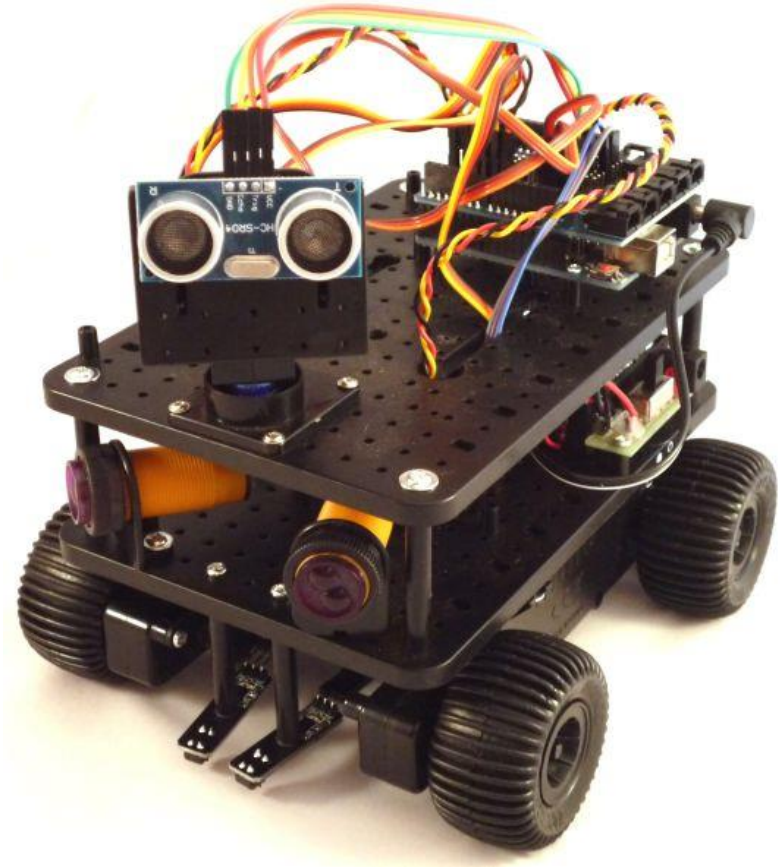
Infra-røde komponenter

Passiv infrarød sensor (bevegelsessensor)

Infrarød diode, sender (fjernstyring)

Infrarød mottaker





Robotikk

Avstandssensor ultrasonisk

Linjeleser

Motorer

Sensorer som virker med magneter

Reed switch: digital

Hall effect sensor: analog

Skjermer, prosjektorer, kamera

Skjermer

LCD

E-ink

Touchskjerm (Arduino har ikke nok minne, se Raspberry Pi)

Kamera

Kamera i bevegelse

Varmesøkende kamera

Optisk mus

Hvordan finne og sette opp nye komponenter?

- Hvor får man kjøpt komponentene?

- Hvordan finne ut hvordan de kobles opp og programmeres?

  - **Dokumentasjon**

    Dokumentasjon er tekst, bilder og tegninger(diagrammer) hvor de som har laget et komponent eller produkt beskriver hvordan det er laget og hvordan vi skal bruke det

    - Komponentene i settet har god dokumentasjon på Arduino sine sider

  - **Biblioteker**

    Ofte kan (må) vi laste ned egne biblioteker for å bruke nye komponenter

Kjøpe komponenter

Adafruit: lager komponenter og gode tutorials

Sparkfun: lager komponenter og gode tutorials

Kjell&Co: selger komponenter, for eksempel fra Adafruit og Sparkfun

Digital Impuls: selger komponenter

Bruke bibliotek med sensorer og aktuatorer

Bibliotek

Et Arduino-bibliotek er en fil hvor det er samlet relaterte funksjoner()

Alle Arduinobibliotek er en mappe som består av to standardfiler; den ene heter headerfil og den andre er en kodefil

I standardbiblioteket, eller kjernebiblioteket, til Arduino finner vi de innebygde funksjonene, variablene og strukturene vi kan se I Arduino Language reference på Arduinos hjemmeside.

Det er mange andre bibliotek som kommer installert i Arduino IDE når vi laster det ned:

Gå til skisse/sketch bibliotek/libraries og se listen over installerte bibliotek

Bruke bibliotek

Standardbiblioteket, eller kjernebiblioteket, blir alltid referert til automatisk uten at vi må skrive noen kommando. Derfor kan vi alltid bruke alle funksjonene, variablene og strukturene i det biblioteket

Når vi bruker et annet bibliotek må vi referere til en biblioteksfil fra skissen/ programmet vårt ved å skrive:

#include <libraryheader>

libraryheader er filnavnet til headerfilen for biblioteket vi ønsker å bruke og skrives som regel slik: biblioteksnavn.h

For å bruke funksjonene I biblioteket må vi fortelle kompilatoren at dette er en funksjon fra et bibliotek ved å skrive

library.function()

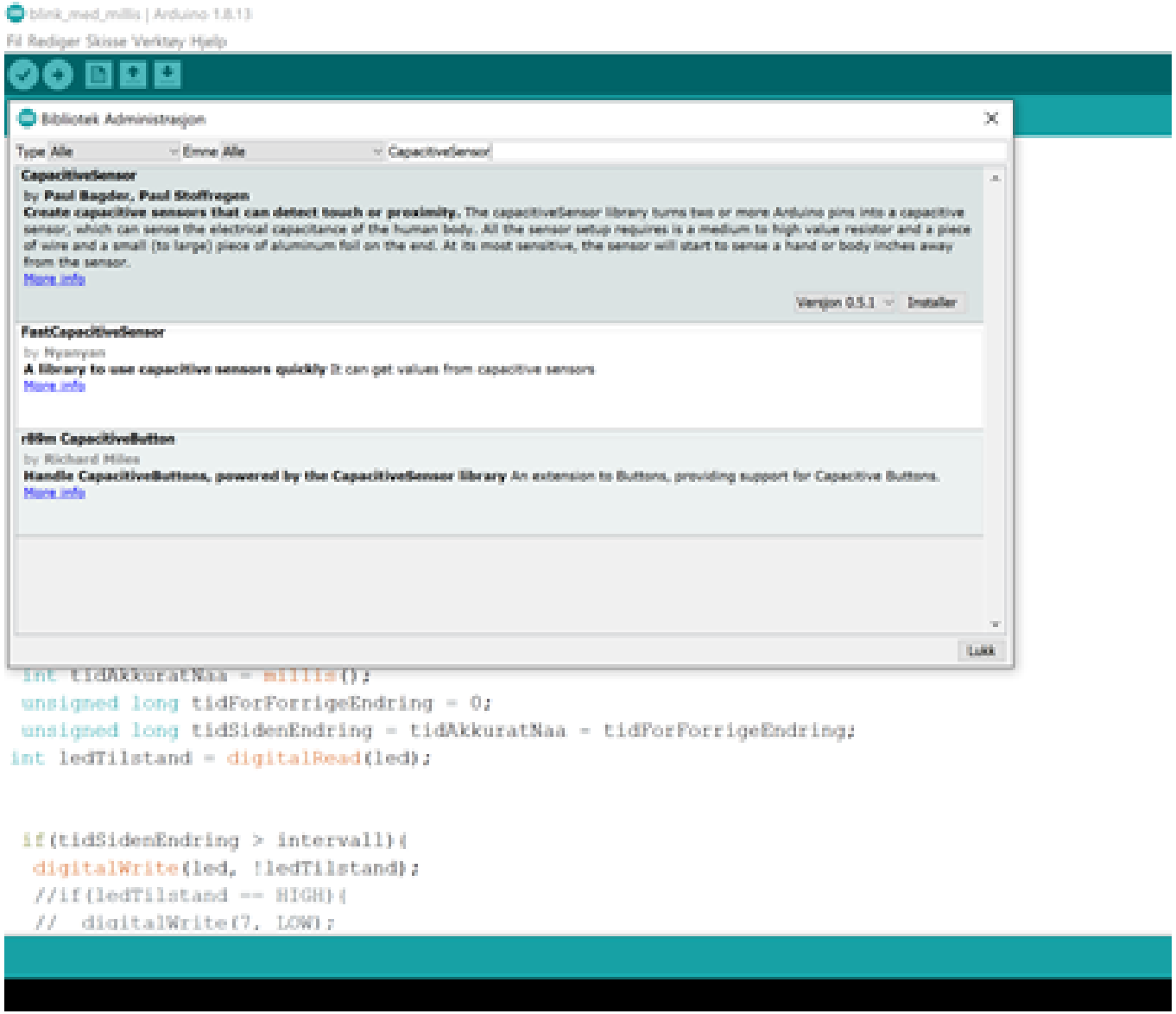
Installere inaktive forhåndsinstallerte bibliotek

En del bibliotek er forhåndsinstallert i Arduino IDEs biblioteksmappe. Men de er ikke aktive, så vi må gå inn i biblioteksmenyen og installere dem i vår versjon av Arduino IDE for å kunne ta dem i bruk

I Arduino IDE, gå til Skisse/Sketch > Inkluder bibliotek/Include Library > Administrer bibliotek > Library manager – velg bibliotek fra listen og trykk på installer siste versjon

Bruk i skisser med #include som ved standard bibliotek

Se Arduino reference eller bidragsyterens dokumentasjon for å se hva slags funksjoner som er tilgjengelige



Installere *bidratte(contributed)* bibliotek

Contributed libraries er laget av andre Arduino-byggere for å gjøre bestemte oppgaver eller blir brukt med bestemte komponenter

Mange av disse finnes ikke forhåndsinstallert i Arduino IDE, men det finnes en liste på Arduino Playground, https://playground.arduino.cc/ > se under libraries

Vi må installere disse manuelt, for eksempel ved å laste dem ned som .zip og lagre dem på PCen/MACen og laste dem opp til Arduino IDE

I Arduino IDE, gå til skisse/sketch> inkluder/import bibliotek > Legg til ZIP bibliotek > velg .zip fra dine mapper

Du finner disse bibliotekene igjen under skisse/sketch> inkluder/import bibliotek contributed/bidratt bibliotek

Bruk i skisser med #include som med standard bibliotek

Oppsummering biblioteker

Standardbiblioteket blir automatisk inkludert, og vi kan bruke funksjonene uten videre

Fårhåndsinstallerte biblioteker kan vi enkelt velge fra en liste, og Arduino skriver include-biten for oss

Forhåndsinstallerte biblioteker kan være inaktive, da må vi gå tillisten og trykke installer før vi kan velge dem

Bidratte (contributed) biblioteker må vi installere ved å laste ned en .zip

Dokumentasjon

Lese dokumentasjon for nye komponenter

Dokumentasjon

Dokumentasjon er tekst, bilder og tegninger(diagrammer) hvor de som har laget et komponent eller produkt beskriver hvordan det er laget og hvordan vi skal bruke det

Dokumentasjon er en del av komponentene som vi trenger for å få dem til å virke

Vi kan tenke på dokumentasjonen som en usynlig del av elektroniske komponenter

Å forstå dokumentasjon gir oss mer handlingsrom

Hvis vi vet hvordan vi leser dokumentasjon er vi ikke like avhengige av å se på andres eksempler, vi kan gå rett til kilden. Da bruker vi «råmaterialet» og lager noe selv, fremfor å kopiere det noen andre allerede har laget.

Tutorials om Arduino kan være gode kilder til dokumentasjon

Ettersom Arduino er en åpen plattform er det mange som skriver dokumentasjon som er lett å forstå, og lager nyttige tabeller og diagrammer fra dokumentasjon som kanskje er litt tung og uvant for oss å lese. Men kryssjekk gjerne kjapt mot original dokumentasjon for sikkerhets skyld.

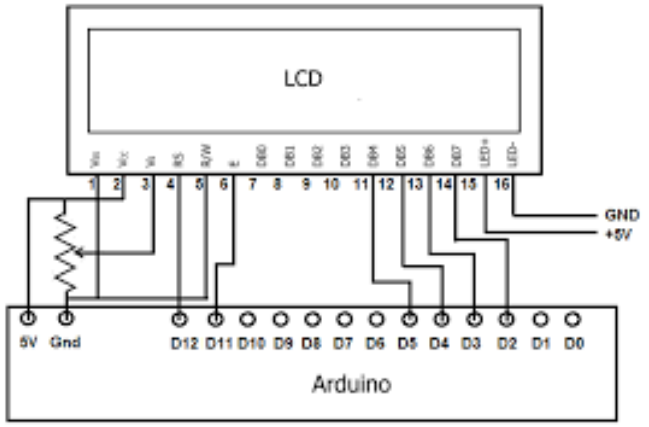
Ting vi må vite for å bruke nye komponenter

Når vi tar i bruk nye komponenter trenger vi ofte funksjoner som er spesielt utviklet for komponentet og informasjon om hvordan det skal kobles og brukes med Arduino

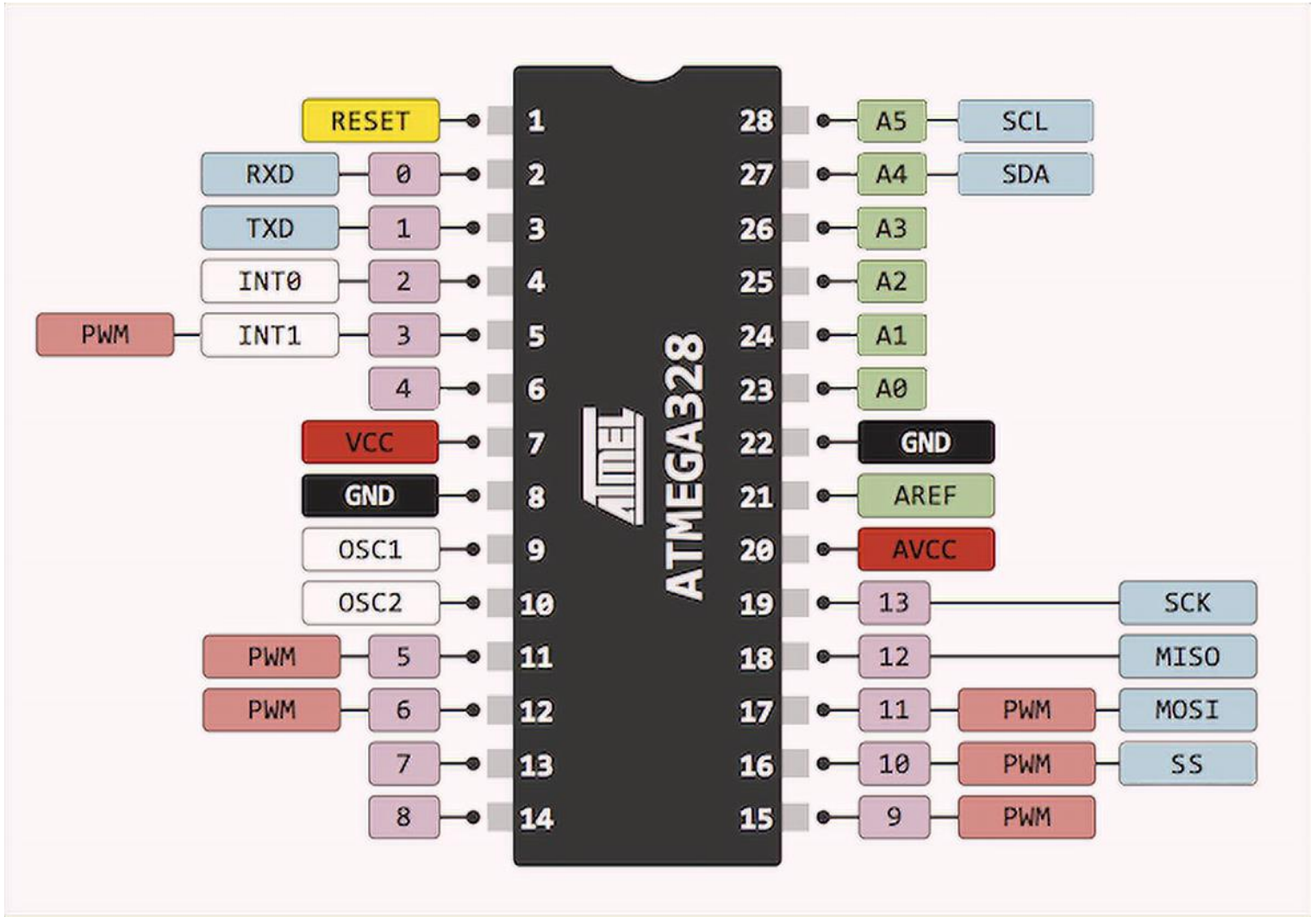
Program/kode
Ofte finnes det egne biblioteker vi kan bruke for å bruke komponenter

Dokumentasjon
For å lære om hvilke biblioteker vi kan bruke, hvordan bibliotekene skal brukes og hvordan vi skal koble kretsene våre må vi finne informasjon. Denne informasjonen er samlet i dokumentasjon

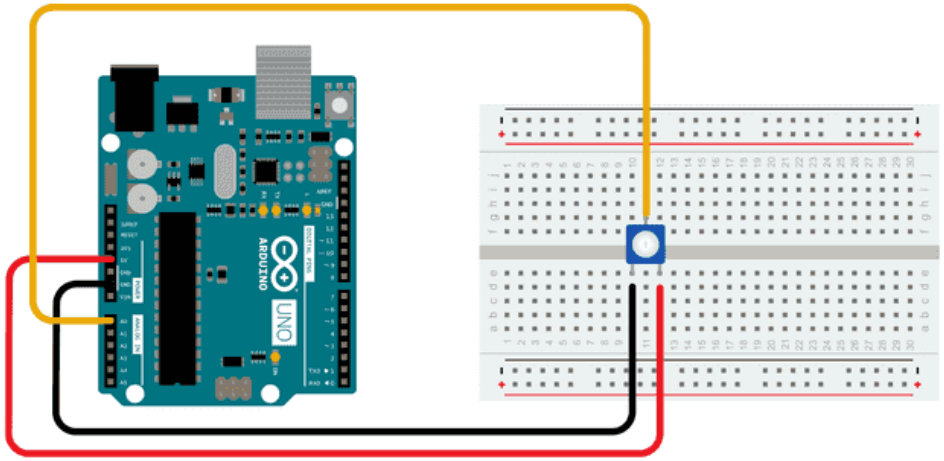
Kretser
Pinouts viser hvordan et komponent kan kobles til Arduinoens pin'er



Eksempel: pinout for Arduino

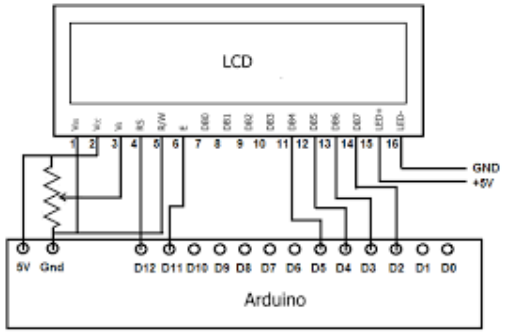


Eksempel: Wiring diagram



Dette er et eksempel på wiring diagram for potensiometeret. Her forstår vi også hva de tre pin'ene til potensiometeret gjør.

En fin ting med Arduinos åpne platform er at mange lager hjelpsomme diagrammer som oppsummerer informasjon fra datablader på en enkel måte, slik som dette fra hhv Arduino.cc og next.gr



Hva slags dokumentasjon er nyttig for oss?

Arduino Language Reference

Vi kan bruke Arduino Language reference aktivt når vi koder og bygger kretser

Tutorials/ Eksempelprogrammer og matchende kretsdiagrammer med gode kommentarer (som i boka)

Tutorials om Arduino-komponenter baserer seg ofte på dokumentasjon fra Arduino language reference, datablader og pinouts, litt avhengig av hvor mye erfaring skaperne av tutorialen har. Tutorials kan være en kilde til god dokumentasjon i mer lettfattelig form fordi informasjonen blir forklart og gjort om til onkrete eksempler. Gode kilder for tutorials er Arduino.cc, Adafruit.com og Sparkfun.com, to store forhandlere av komponenter som retter seg mot Do It Yourself/maker-miljøer, samt instructables (hvem som helst kan publisere på instructables, vurder informasjonen kritisk)

PINOUTS og wiring diagrams

For å finne ut hvordan vi kobler et nytt komponent til Arduino kan vi søke opp pinout'en til komponentet. Pinout beskriver hva hver pin på et komponent gjør. Ofte kan vi også finne pinouts som beskriver hvilke pin'er vi bruke på Arduinoen også. Noen bibliotek er programmert slik at det ikke er valgfritt hvilke pin'er vi velger for at komponentet skal virke. Ofte kan vi lese hva de ulike pinene gjør ut fra wiring diagrams.

Datablader

Er ofte i form av pdf-er og inneholder beskrivelser og tekniske spesifikasjoner som pinout, størrelser/mål på komponentet, hvor mye strøm det tåler, og kanskje maks og min temperaturer komponentet tåler etc.

Hvor finner vi datablader?

NB! Det er nyttig å vite at informasjon om komponenter kan finnes i datablader. Noen ganger er imidlertid datablader svært detaljerte og lite relevante for Arduinoprosjekter som våre. Det kan være mer nyttig å søke etter tutorials og bruke Arduino Reference-sidene

Produsenter og forhandlere av komponenter har gjerne databladet tilgjengelig via en lenke på samme side som produktet vises/selges

For å finne riktig datablad må vi vite navnet på komponentet og kanskje mer spesifikke betegnelser på hvilken versjon vi bruker

Det er ofte lønnsomt å google «*komponentnavn* for Arduino»

Hva ser vi etter i dokumentasjon?

Programmering: bibliotek, kompabilitet

Når vi lager programmer kan vi se etter om komponentet krever spesielle bibliotek. Hvis vi vurderer å kjøpe det kan vi sjekke at det kan brukes sammen med Arduino. Forhandlerne **Adafruit** og **Sparkfun** er ofte gode kilder.

Koble kretser: pinouts og elektriske egenskaper

Pinouts, krav til minimum eller maks spenning- og strømstyrke. For at det skal være kompatibelt med Arduino må det fungere på mellom 3 og 12 volt, helst mellom 3 og 5.

Fysisk prototyping: dimensjoner

Datablader er praktiske når man lager dategningner av prototyper, for eksempel til 3D-printing. Man kan få helt nøyaktige mål av alle deler av et komponent fra datablader.

NB! Manglende dokumentasjon og datablad

Det finnes en rekke billige alternativer til komponenter, som for eksempel skjold, fra store internasjonale forhandlere

Kontroller at disse har god dokumentasjon før dere bestiller, og lagre dokumentasjonen lokalt

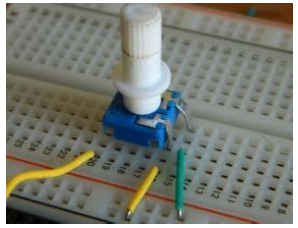
Skjold og andre komplekse komponenter uten datablad er vanskelige å bruke og nesten umulige å feilsøke.

Eksempler på bruk av komponenter i 1060-prosjekter

+ mer om bruken av et utvalg konkrete komponenter med biblioteker og dokumentasjon

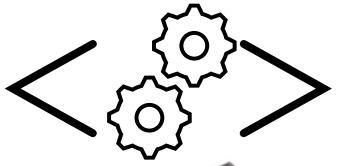
Eksempel:
Game over?
CAPY, 2022

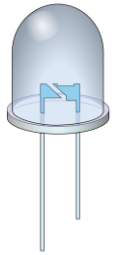
- Potensiometer
- LED



Sensorer

Potensiometer





Aktuatorer

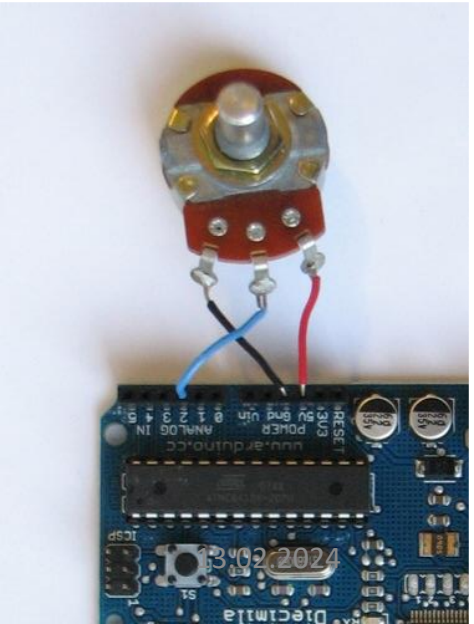
LED

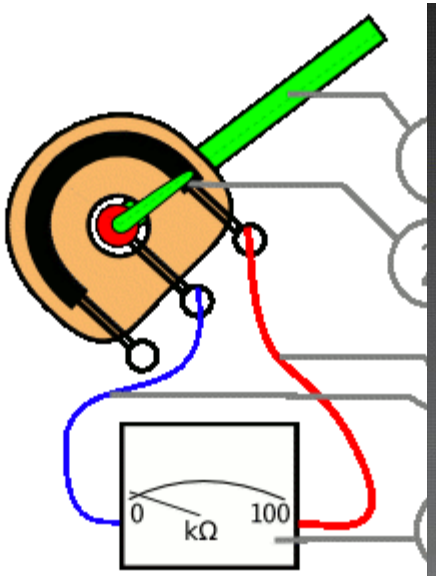
Potensiometere

Potensiometer: begrenset til 270 grader, enkel avlesning

Slidere er potensiometer med en annen form!

Rotary encoder er et lignende komponent som måler mer enn 270 grader, litt vanskeligere å sette opp





- Potensiometer og LED finnes i alle settene våre

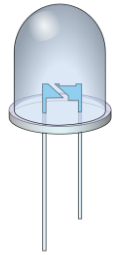
- De kan brukes med Arduinos innebygde standardbibliotek

Eksempel:
Garduino, 2022

- Temperatursensor
- LED
- Piezo/Buzzer

Sensorer

Temperatursensor



Aktuatorer

LED

Piezo/Buzzer

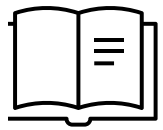
- Temperatursensor og piezo finnes i alle settene våre

- Vi kan bruke Arduinos standardbibliotek

- (mer om trådløs tilkobling neste uke)

Eksempel:
Connection, 2022

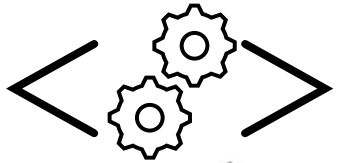
- Knapper
- Touchsensor av aluminiumsfolie
- RGB LED
- Neopixler/LED strip



Knapper

Sensorer

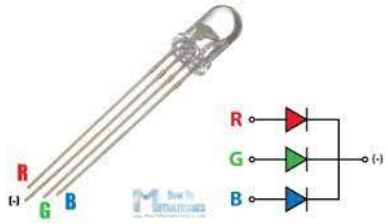
Touchsensor av aluminiumsfolie



Aktuatorer

RGB LED

Neopixler



Touch-sensorer

To vanlige typer touch-teknologi

Capasative touch

Vanlig i touch-screen teknologi

Bruker menneskekroppens evne til å lede elektrisitet (kroppens kapasitanse (NO), capitance (EN)

En liten mengde strøm trekkes til det punktet hvor f.eks. en finger berører en skjerm og målingen registrerer hvor trykket er plassert

Trenger litt mindre strøm for å registrere trykk og kan registrere trykkets lokasjon mer nøyaktig

Resistive touch

Vanlig i touch-screen teknologi

To lag med ledende materialer (materialer som leder elektrisk strøm) kommer i kontakt ved at noe presser dem sammen. Motstanden mellom de to punktene måles og trykket blir registrert på dette punktet

Capasative touch med Arduino

Capasative touch

Vi kan bruke capasative touch som sensor for å lese av både berøring fra og nærhet til mennesker, som en finger eller hånd.

Vi kan bruke biblioteket **CapacitiveSensor** for å lage vår egen capasative touch-sensor av Arduinobrettet, en resistor, en ledning og litt aluminiumsfolie (se Arduino reference-libraries):

https://www.arduino.cc/reference/en/libraries/capacitivesensor/

Med denne sensoren kan vi registrere at noen tar på ledningen eller aluminiumsfolien

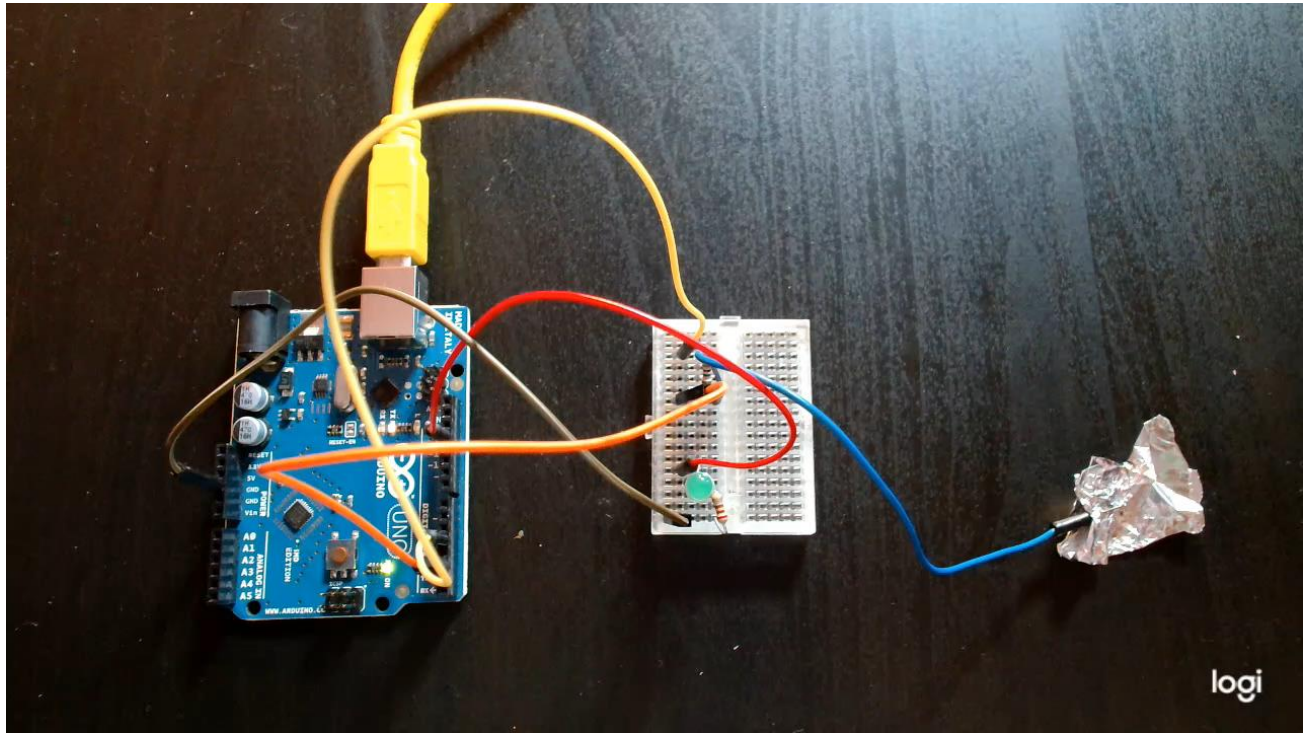
Touchsensor med Arduino

Eksempel 13 i boka >

Installer biblioteket CapacitiveSensor I Arduino IDE, gå til Skisse/Sketch > Inkluder bibliotek/Include Library > Administrer bibliotek > Library manager – velg bibliotek fra listen og trykk på installer siste versjon

Se dokumentasjon for biblioteket på

https://www.arduino.cc/reference/en/libraries/capacitivesensor/



Andre touch-sensorer for Arduino #1

Vi kan bruke en rekke materialer som har evnen til å lede strøm

Metaller og vanlige/tradisjonelle elektriske ledere

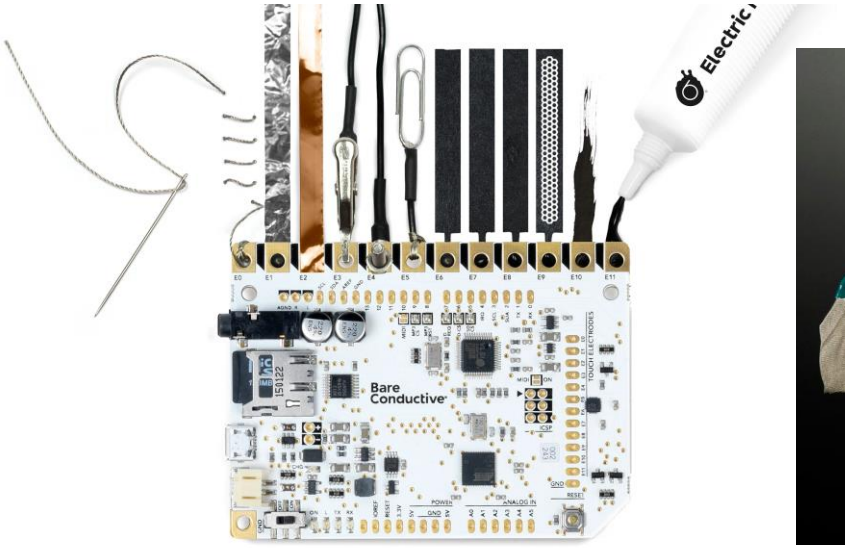
Ledende tekstiler

Frukt og planter med tykke blader (vannholdig, vannet leder strøm)

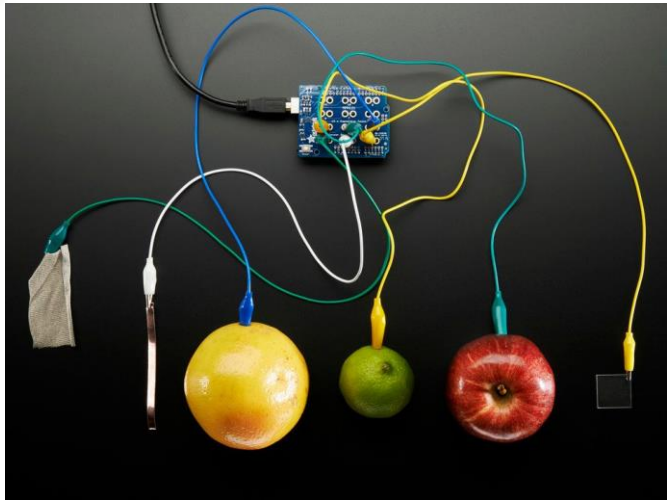
Grafitt (blyant-«bly»)

Andre touch-sensorer for Arduino #2

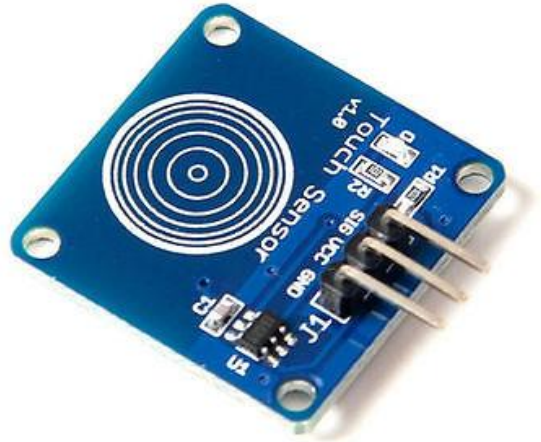
Tilgjengelige touch-sensorer man kan kjøpe (produkter)



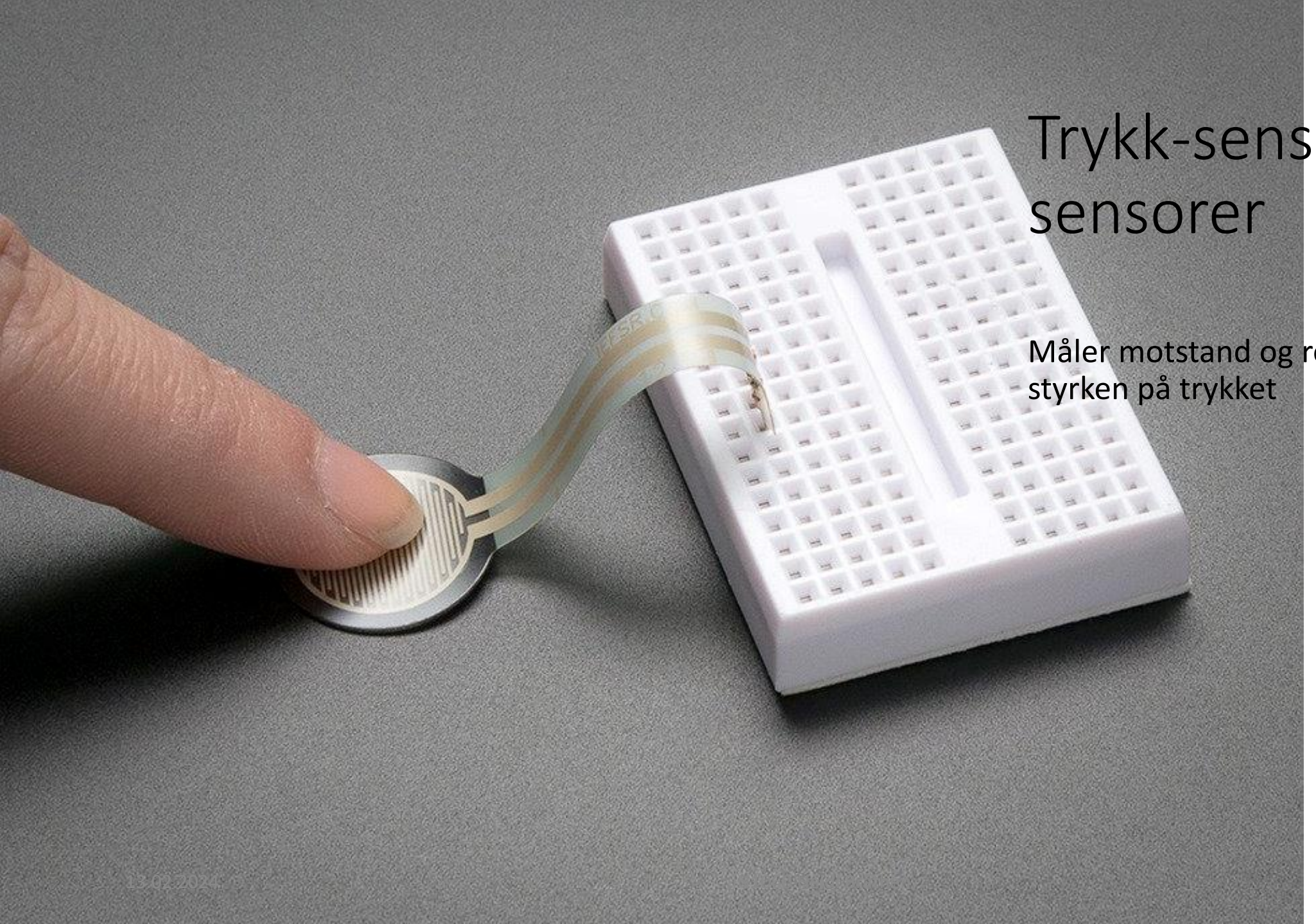
Bare conductive
(ikke kompatibelt med Arduino, kan programmeres med Arduino IDE separat system, dyrt)
https://www.bareconductive.com/



Adafruit capasative touch shields
(noen er kompatible med Arduino UNO)
https://www.adafruit.com/



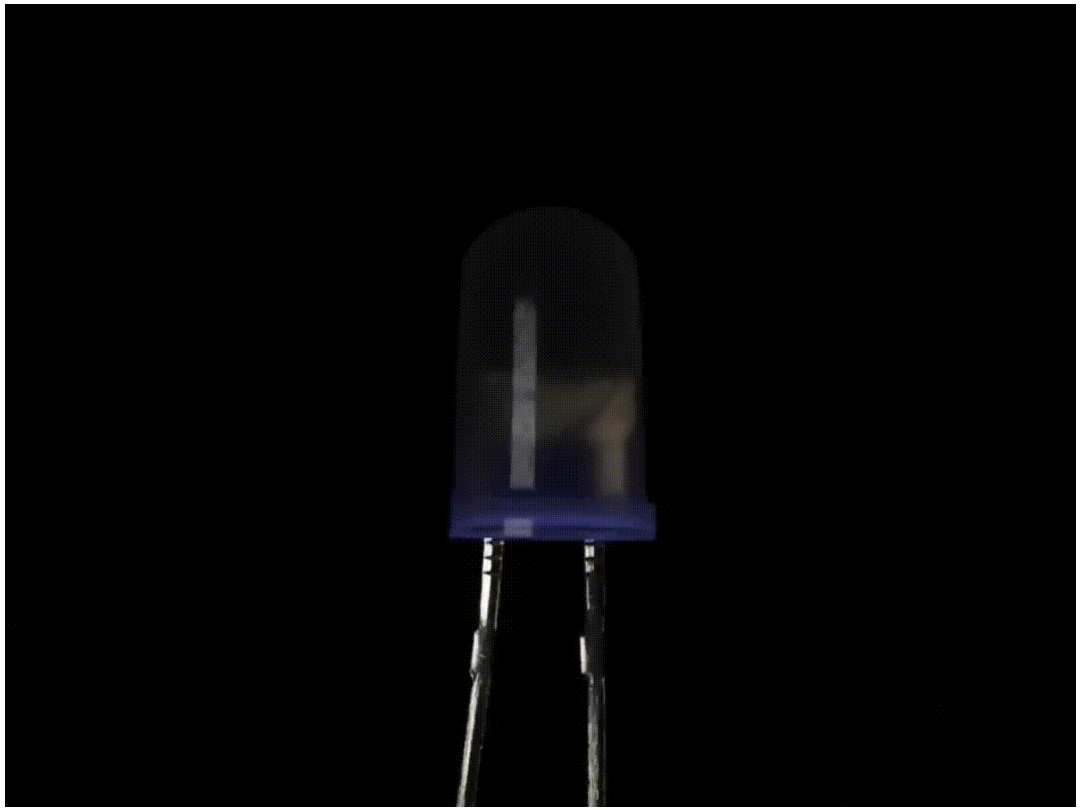
Arduino capasative touch sensor
diverse forhandlere

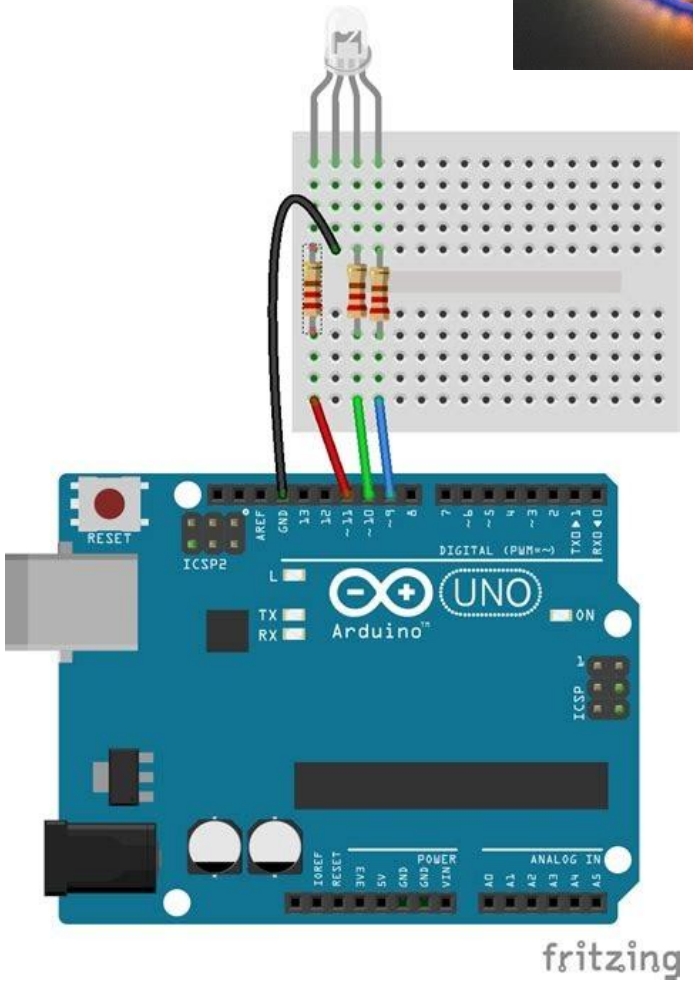


Trykk-sensitive sensorer

Måler motstand og registrerer styrken på trykket

RGB LED rød, grønn og blå LED





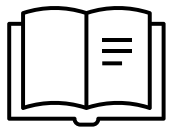
Neopixler

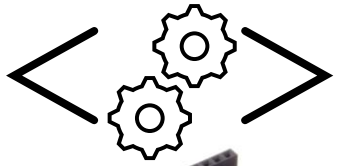
Finne dokumentasjon

Contributed bibliotek

Eksempel:
Bikuben
Svakeste LED, 2022

- **RFID**
- **Piezo**
- **LED**
- **Servomotor**

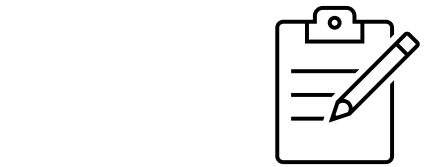




RFID leser studentkort

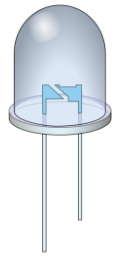
Sensorer

Piezo leser at noen banker på døren



Aktuatorer

LED lyser opp



Servomotor låser og låser opp døren

RFID og NFC

RFID: Trådløs kommunikasjonsteknologi

Radio Frequency Identification

Består av «tags» (for eksempel i adgangskort) og en RFID-leser

Det genereres et elektromagnetisk felt ved induksjon når en tag er i nærheten av leseren, dette gir strøm til tag'en som kan sende signal til leseren

Tags har unike id-er som kalles Unique Identification Digit (UID), og en chip som lagrer UID og antenne

Leseren kan både sende og motta radiosignaler og består av antenne, kontrollenhet og radiofrekvensmodul

RFID versus NFC

RFID

Radio Frequency Identification

Lang rekkevidde (flere meter)

Vanlig i bussbilletter, adgangskort og for sikker utskrift

NFC

Near Field Communication

-En type RFID-teknologi

Kort rekkevidde (10 cm)

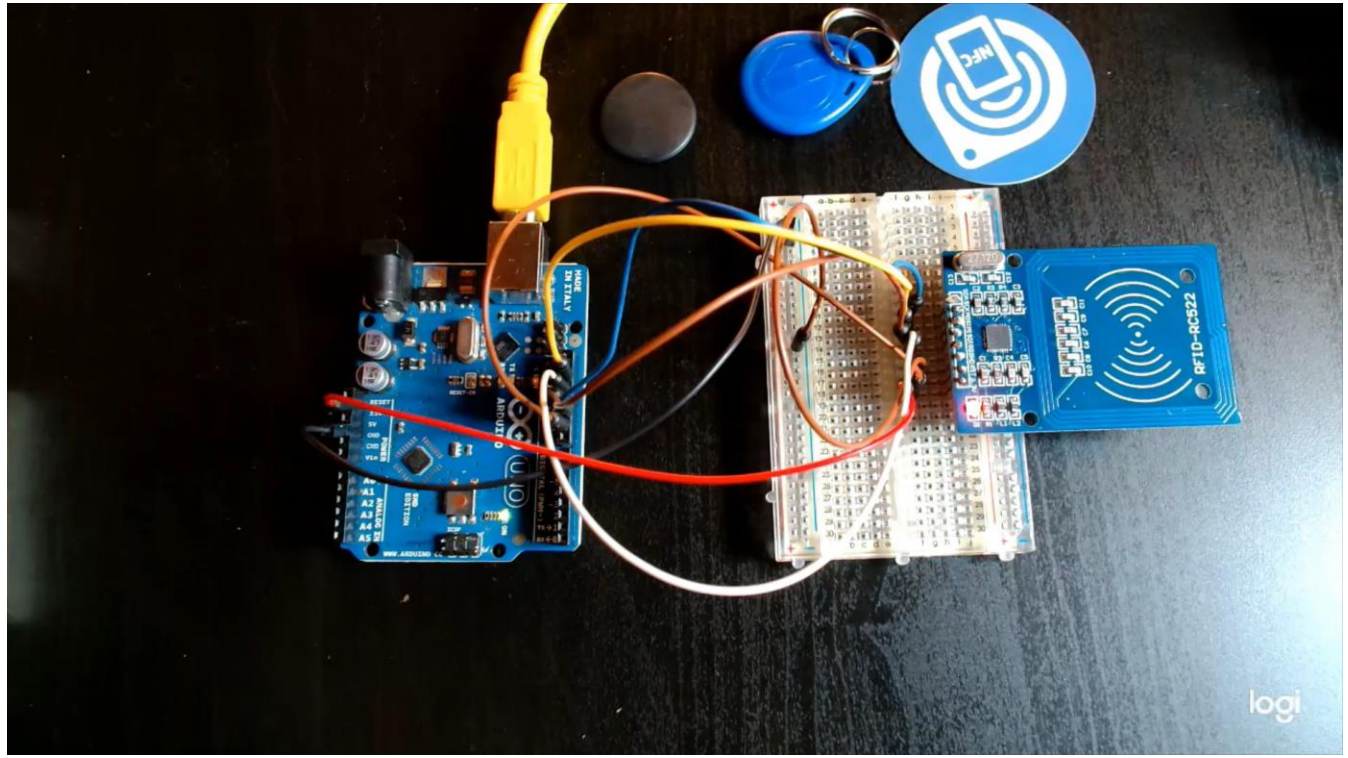
NFC-teknologi kan ofte finnes i mobiler, kan aktiveres og brukes som RFID, men med kort rekkevidde

Bruke RFID

Det finnes flere typer RFID-leser skjold for Arduino

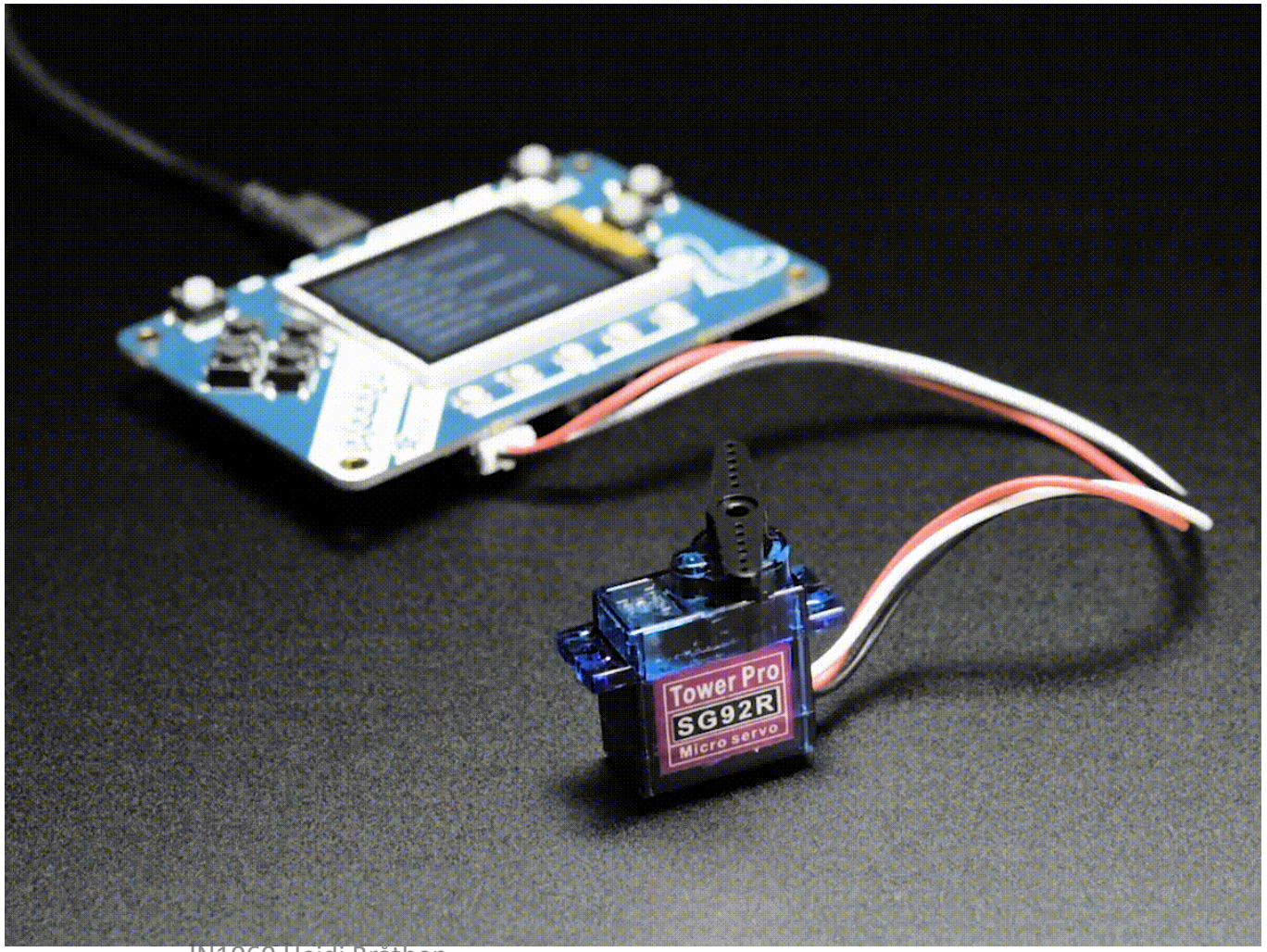
Avhengig av typen skjold må man finne pinout og de riktige bibliotekene (ofte må de lastes ned som .zip og installeres i Arduino IDE)

Anbefaling: kjøp fra Adafruit, de har god kvalitet på komponenter, mange forhåndsinstallerte biblioteker i Arduino IDE og ofte gode tutorials



Servomotorer

Servomotor



Styre servomotorer med Servo-biblioteket

Vi må bruke Servo-biblioteket (standard i Arduino IDE)

Servo-biblioteket er et eksempel på et bibliotek som er installert i standardversjonen av Arduino IDE. **Vi trenger ikke å installere det selv**, vi kan bare bruke det.

Vi kan se det i listen over bibliotek i Arduino IDE:

I Arduino IDE, gå til Skisse/Sketch > Inkluder bibliotek/Include Library > se på listen

Hvis du trykker på biblioteket fra listen, **legger Arduino til kode for å inkludere biblioteket** i den skissen du har åpen

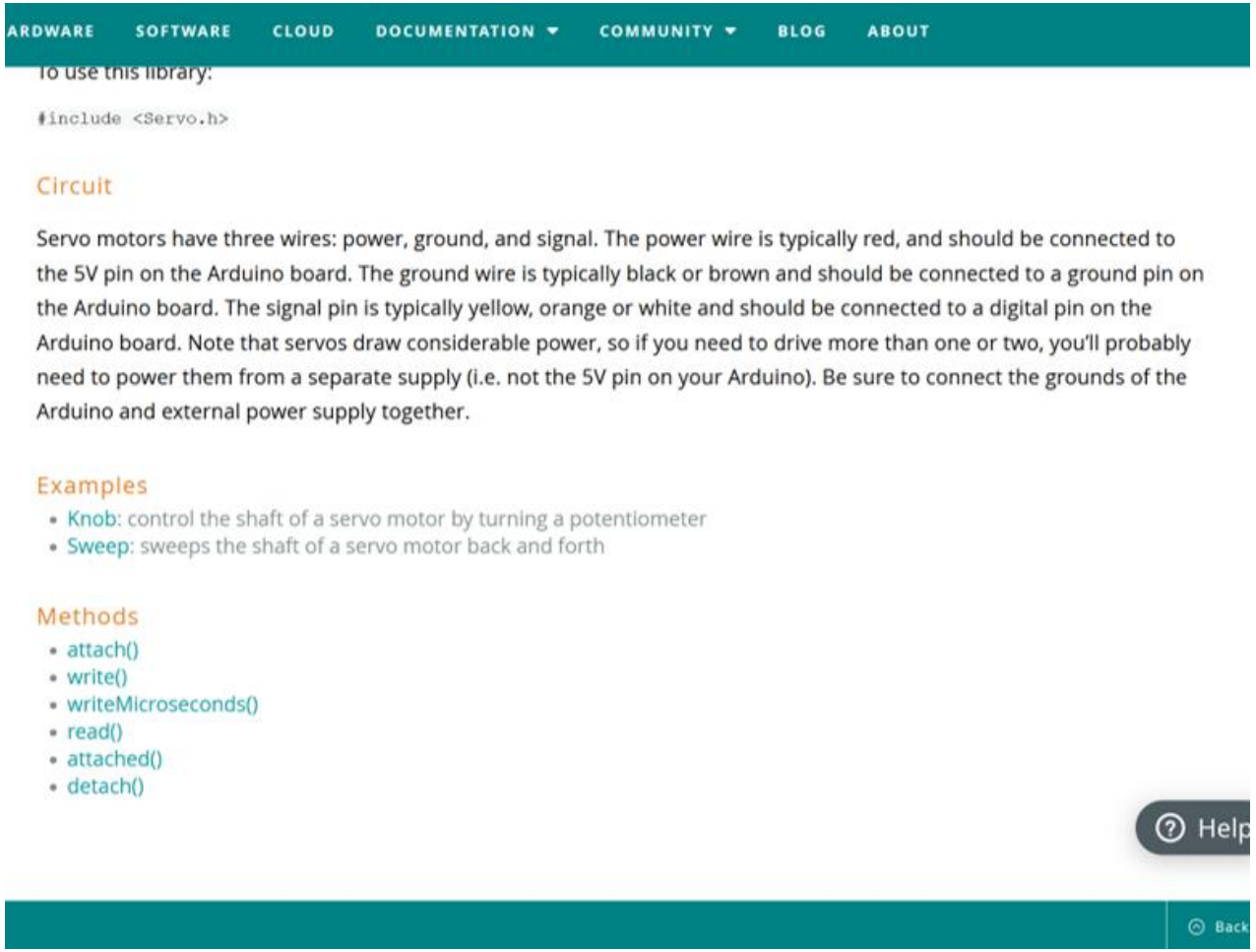
Du kan også bruke biblioteket i skissen ved å skrive :

#include<Servo.h> akkurat som i *#include <libraryheader>*

Dokumentasjon for Servo-biblioteket

https://www.arduino.cc/reference/en/libraries/servo/

I Arduino reference kan vi bl.a. lese om hvordan vi kobler servomotor i en krets, hvilke funksjoner som er tilgjengelig i biblioteket, og hvilke eksempler vi kan se på i Arduino IDE



Styre servomotorer med Servo-biblioteket

1: Bruk biblioteket i skissen: #include<Servo.h> akkurat som i *#include <libraryheader>*

2: Lag et servo-objekt, for eksempel Servo minServo;

- Dette biblioteket forutsetter at vi lager et servo-objekt for hver servomotor
- Vi må bruke objektnavnet til å vise kompilatoren at funksjonene er fra et bibliotek i stedet for å bruke bibliotekets eget navn: minServo.funksjon(); i stedet for Library.function()

3: Bestem pin for signalet ut til servomotoren med funksjonen attach() i Servo-biblioteket: minServo.attach(pin);

Servo.attach() tilsvarer pinMode() for servomotorer

4: Bestem posisjon på motoren med funksjonen write() i Servo-biblioteket myServo.write(90);

Servo.write() tilsvarer analogWrite() for servomotorer og gjør at motoren roterer til en gitt posisjon

Vibrasjonsmotorer

En annen type enkle motorer

Bruke vibrasjonsmotorer med Arduino

Enkle å bruke

Kan styres med analogWrite() for å justere strømstyrke og dermed vibrasjonsstyrke

Bra (og vanlig) for å skape «haptic feedack»

Sensorer og aktuatorer som vi ikke har i settet

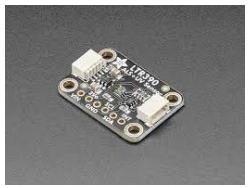
Eksempel: UV-briller av Korts/utning, 2023

Video: https://vimeo.com/831665347

Prosjektside: https://www.uio.no/studier/emner/matnat/ifi/IN1060/v23/prosjekter-var-2023/kortslutning/

fysisk prototype: foran
- knapp på siden
- UV-sensor i midten

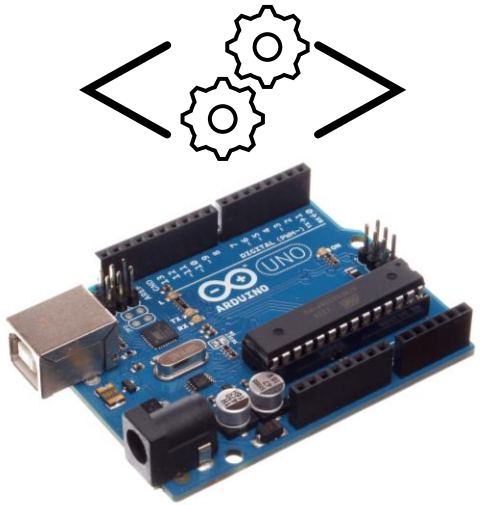
fysisk prototype: bak
- skjerm til venstre
- 3 LED-lys til høyre



Sensorer

Knapp

UV-sensor



Aktuatorer

Neopixler (LED)

Skjerm

Skjold

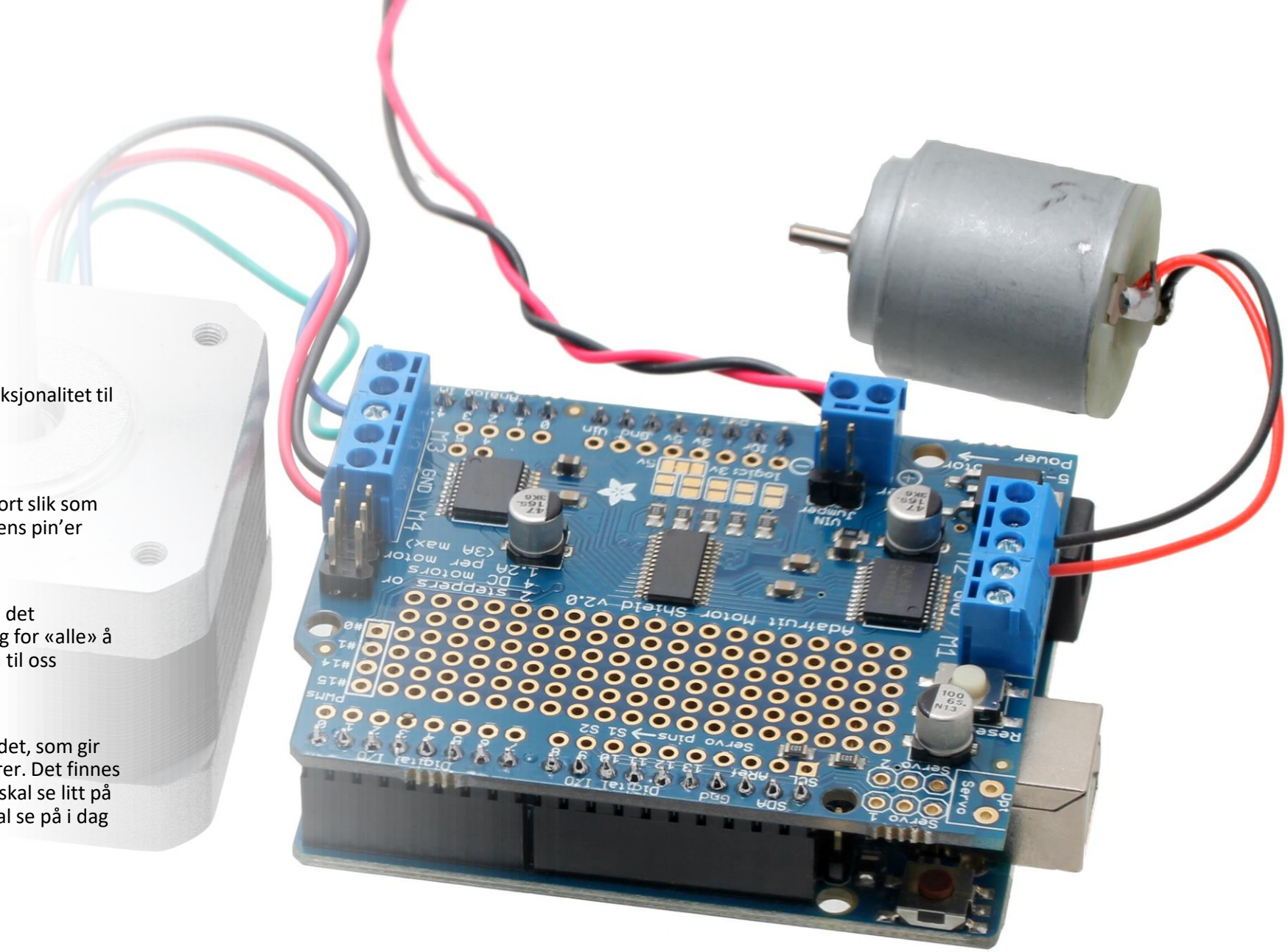
Skjold

- Et skjold er en ekstern krets som gir ekstra funksjonalitet til Arduinoens standard grensesnitt

- Et skjold er en krets som er laget på et brett/kort slik som Arduinoen og som passer å koble til oppå Arduinoens pin'er

- En av fordelene med open source maskinvare: det standardiserte og åpne grensesnittet gjrød et mulig for «alle» å lage skjold for kretser de ofte bygger og selge dem til oss

- Et eksempel på populære skjold er motorskjoldet, som gir flere pwm-pinner slik at man kan styre flere motorer. Det finnes også ethernet-skjold og prototypingsskjold som vi skal se litt på neste gang, og LCD-skjold og RFID-skjold som vi skal se på i dag



Skjold og bibliotek

Skjold kommer som regel med egne bibliotek for å utnytte de spesielle tilleggsfunksjonene som skjoldet tilbyr

En del slik skjold er så populære at bibliotekene er forhåndsinstallert i Arduino IDEs biblioteksmappe. Men de er ikke aktive, så vi må gå inn i biblioteksmenyen og installere dem i vår versjon av Arduino IDE for å kunne ta dem i bruk

I Arduino IDE, gå til Skisse/Sketch > Inkluder bibliotek/Include Library > Administrer bibliotek > Library manager – velg bibliotek fra listen og trykk på installer siste versjon

Informasjon om slike bibliotek finnes på nettsiden Arduino reference – libraries (samme sted som Language reference)

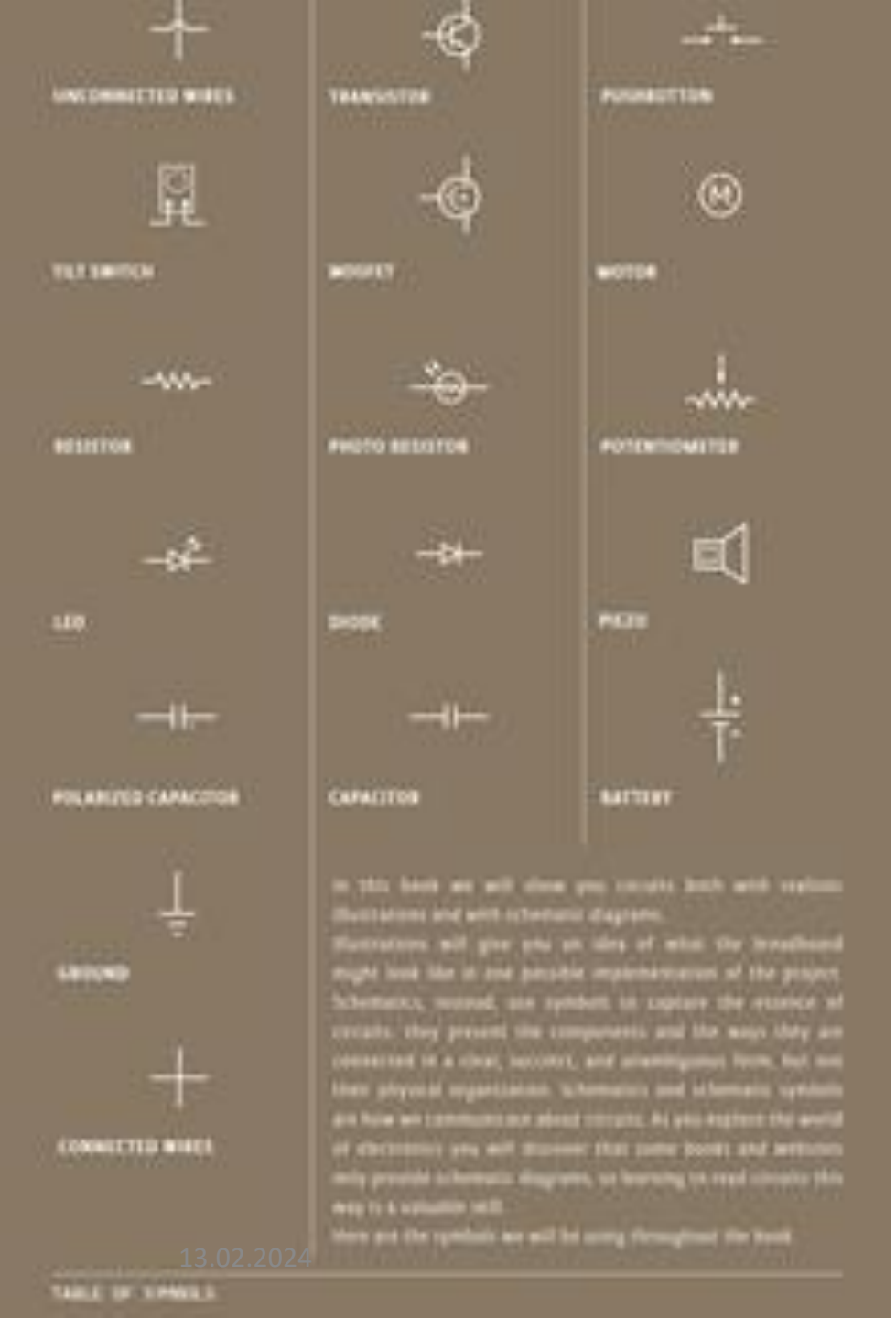
Se for eksempel motorskjold-biblioteket her: https://www.arduino.cc/reference/en/libraries/adafruit-motor-shield-library/

Ofte stilte spørsmål (OSS)

Lese kretsdiagrammer

Resistorer

Serie- og parallellkoblinger



Lese og tegne kretsdiagrammer

Kretsdiagram

en skjematisk tegning av en krets som beskriver hvordan kretsen skal bygges

Symboler

Vi bruker enkle symboler for å representere komponentene/ delene i kretsen. Dere kan se en oversikt over symboler for komponentene i settene deres i Arduinoboka på side 10

Spenningskilde og ledninger er I alle kretstegningene

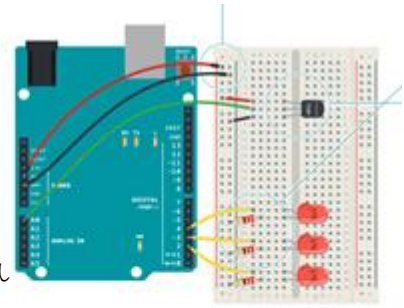
Alle kretsene vi bygger inneholder en spenningskilde (se symbolet for batter) som er representert med en lengre og en kortere strek sammen med + og -. Ledninger tegnes med rette streker som viser hvordan ting er koblet sammen.

Tegne kretsdiagrammer

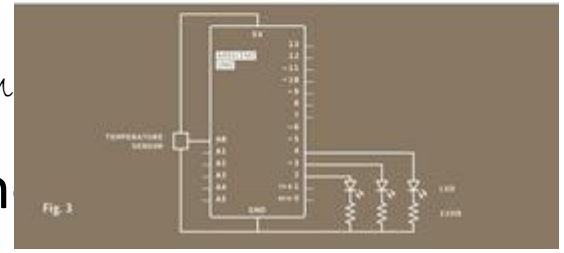
Det kan være lurt å tegne symbolene først og ledningene til slutt

Visuelle versus grafiske kretsdiagram

Visuelt kretsdiagram



Grafisk kretsdiagram



Grafisk kretsdiagram

Hvordan kan vi beskrive at kretsen går til en bestemt pin på Arduin

For å representere arduinoen i et grafisk kretsdiagram tegner vi et rektangel med nummererte pin'er

Vi bruker arduinoens 5V og GND for å representere spenningskilden i slike kretser i stedet for et eget symbol for spenningskilden

Men vi representerer ikke breadboardet (breadboardet er som ledninger)

Vi kan tegne flere 5V og jord/GND fra samme 5V og jord/GND fra + og – på Arduino

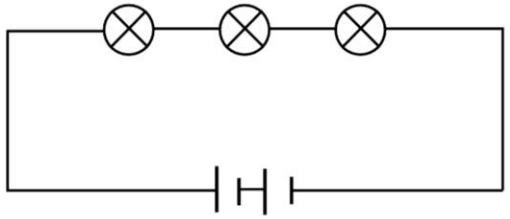
Kretser: serie- og parallellkobling

Seriekobling

Når vi kobler komponenter etter hverandre slik at strømmen går fra pluss og gjennom hvert komponent etter tur før den kommer til jord, har vi koblet dem i serie

Ved mange komponenter i serie kan vi få trøbbel med å få nok strøm og LED kan reagere med å ikke lyse

Hvis vi kobler knapper i serie må alle trykkes på for at kretsen skal sluttes



Parallellkobling

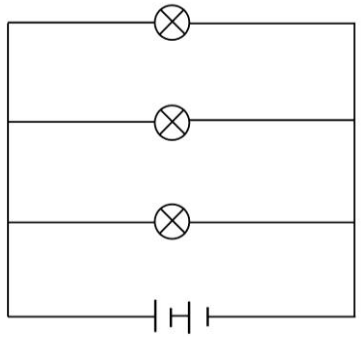
Når vi kobler komponenter på en måte som gjør at strømmen deles mellom komponentene

Hvis en serie med LED ikke virker kan vi prøve å fordele strømmen

Hvis vi parallellkobler knapper kan vi trykke på bare en av dem, og kretsen vil sluttes

Se s 28-29

i boka



Oblig 1: Hva betyr «R1», R2» etc i kretstegningene?

OSS: Hva betyr «R1», «R2» osv på kretstegningene i oblig1, er det kode for ulike styrker?

Svar: Det er vanlig å gi komponenter navn i kretsdiagrammer for oversiktens skyld, og R1 betyr bare «resistor nummer 1 i denne kretsen». Det er også vanlig å merke R1 og R2 videre med styrke, men dere kunne ikke se styrken i oblig1.

Mindre mostand

Mer mostand

Hva er forskjellen på de ulike resistorene?

Resistorene er fargekodet, og fargene representerer ulik styrke på resitorene

Diagrammet på side 41 i Arduino-boka viser hvordan styrken på resistoren regnes ut ved hjelp av fargebåndene og hvilke resistorer som er i deres sett

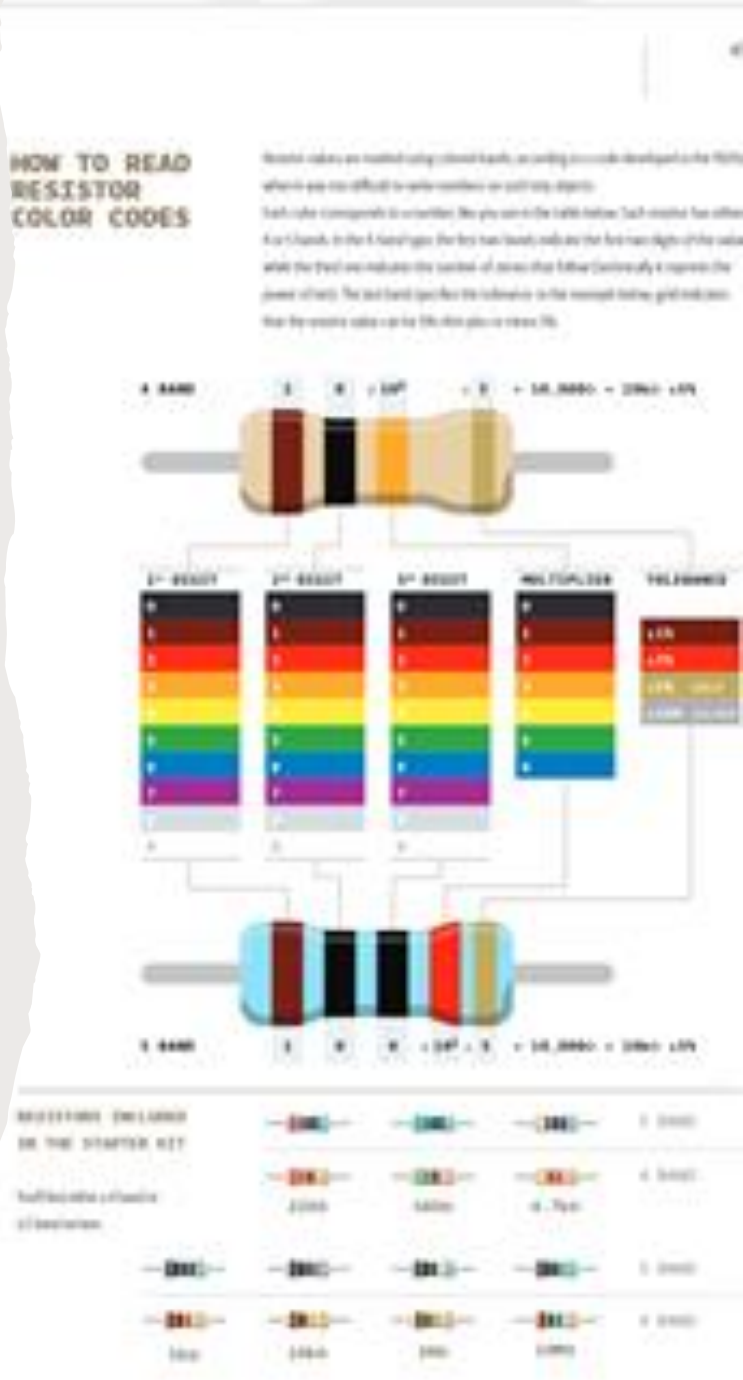
En online kalkulator:

https://www.translatorscafe.com/unit-converter/JV/calculator/resistor-color-code/

NB! Det skal være et diagram i boka deres, men det ser dessverre ut til å ha falt ut hos flere i år

Boka ligger som PDF på semestersiden under ressurser, direkte lenke:

https://www.uio.no/studier/emner/matnat/ifi/IN1060/v21/arduino/arduino-projects-book.pdf



Hva skjer hvis vi bruker feil resistor?

Hvis vi bruker for liten resistor/for lite motstand, får komponentet for mye strøm. Det kan ødelegge komponenter

Gå heller litt opp enn ned i motstand, men vær obs på at også for høy motstand kan gjøre at komponenter ikke virker som de skal

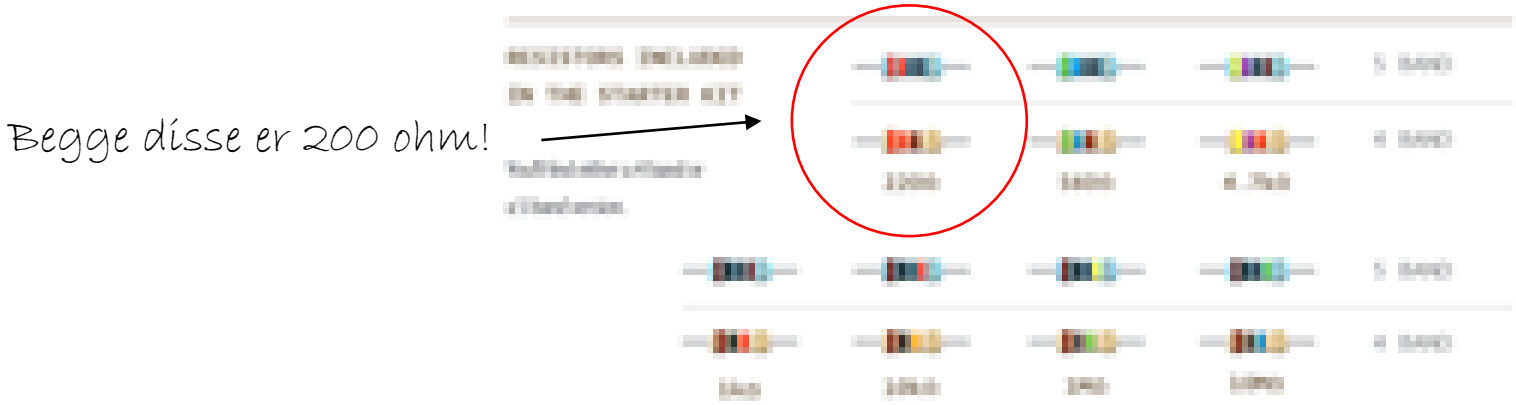
Hvis vi bruker for mye motstand kan vi få for lite strøm til komponentet

En LED vil for eksempel lyse synlig svakere hvis motstanden vi velger å bruke er veldig høy

220 Ohms resistoren i settene deres er blå!

På de fleste kretsdiagrammene i forelesninger og i boka er 220 ohms-resistoren som gjerne brukes sammen med LEDene hvit/beige med to røde, et brunt og et gull-bånd

Men i mange av settene deres er det en utgave som er blå, se nevnte diagram i boka



Skrive egne programmer og feilsøking

Oblig 2

Feilsøking

Det er veldig mange feilkilder når vi har både krets og kode

Det er veldig vanlig å få noen feil når vi setter opp nye prosjekter

Det er ofte vanskelig å se hvor feilen er

Feilen kan være i koden selv om vi ikke får feilmeldinger:

Ikke alle feil blir fanget opp av kompilatoren

Ikke alle feil er feil for Arduino: hvis vi for eksempel har brukt digitalWrite for å styre et potensiometer så vet ikke Arduioen at vi ønsker å bruke et potensiometer

Feilsøking: Seriell monitor

Ved å bruke seriell monitor kan vi se hva som skjer i programmet selv når kretsen ikke virker slik vi forventer

For å bruke seriell monitor:

Sett inn følgende kode i programmet ditt:

I setup(): serial.begin(9600);

I void(): serial(println(en eller annen test);

Du kan skrive ut for eksempel en verdi ved å sende inn variabelnavnet, eller bare skrive ut en tekst for se om programmet kjører dit du tester

Etter at du har lastet opp de endrede programmet kan du åpne seriell monitor og se om testene dine skrives ut og hva de inneholder

Serial: et verktøy for problemløsning (rep)

Arduinoen har et verktøy som lar oss se signalene som går ut og inn i form av tekst

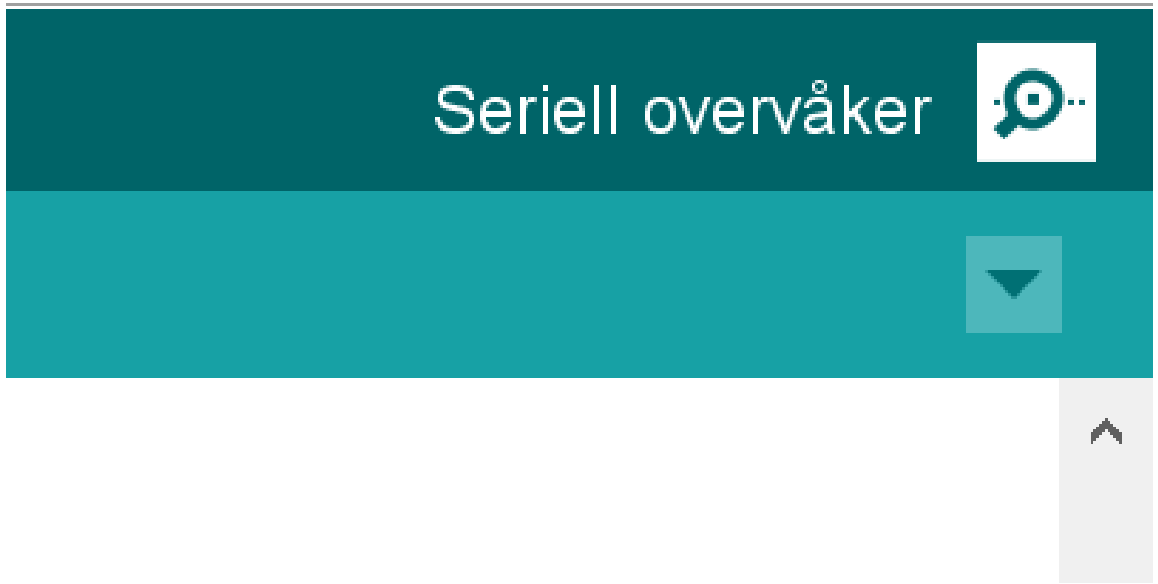
Gå til:

Verktøy – seriell overvåker eller ctrl+shift+m eller ikon øverst til høyre

Vises som popup-vindu

Seriell overvåker (serial monitor)

Skriver ut returverdier fra funksjonene når programmet kjører i form av tall eller tekst



Funksjoner for seriell kommunikasjon #1

For å starte seriell kommunikasjon må vi initialisere den med

Serial.begin(9600)

Serial.begin()-funksjonen tar inn en variabel for baud rate.

Baud rate refererer til hastigheten på kommunikasjonen over en datakanal. På Arduino er det vanlig å sette den til 9600.

Serial.print()- funksjonen lar oss skrive ut beskjeder til overvåkeren og er nyttig for å følge med på hva som egentlig skjer i programmet

```
DigitalReadSerial eksempelskisse

int pushButton = 2;

void setup() {
Serial.begin(9600); // start seriell
kommunikasjon ved 9600 bits per
sekund
pinMode(pushButton, INPUT);
}

void loop() {
int buttonState =
digitalRead(pushButton);
Serial.println(buttonState); //skriv
ut tilstanden til knappen
  delay(1); //pause mellom
avlesningene for stabilitet
}
```

Tips for å be om hjelp via teams

Husk å beskrive problemet

- Bilder av kretsen
- kodefil (enda bedre enn skjermbilder)
- beskriv kort: hva er problemet?

  Hva skjer, og hva forventet du at skulle skje?
- Beskriv kort: hva har du forsøkt for å løse problemet?

  Hva skjedde da du forsøkte disse tingene?

Det er superfrustrerende å ikke finne feilen når ting *burde virke*, men husk at vi finner ut av det på en eller annen måte ☺