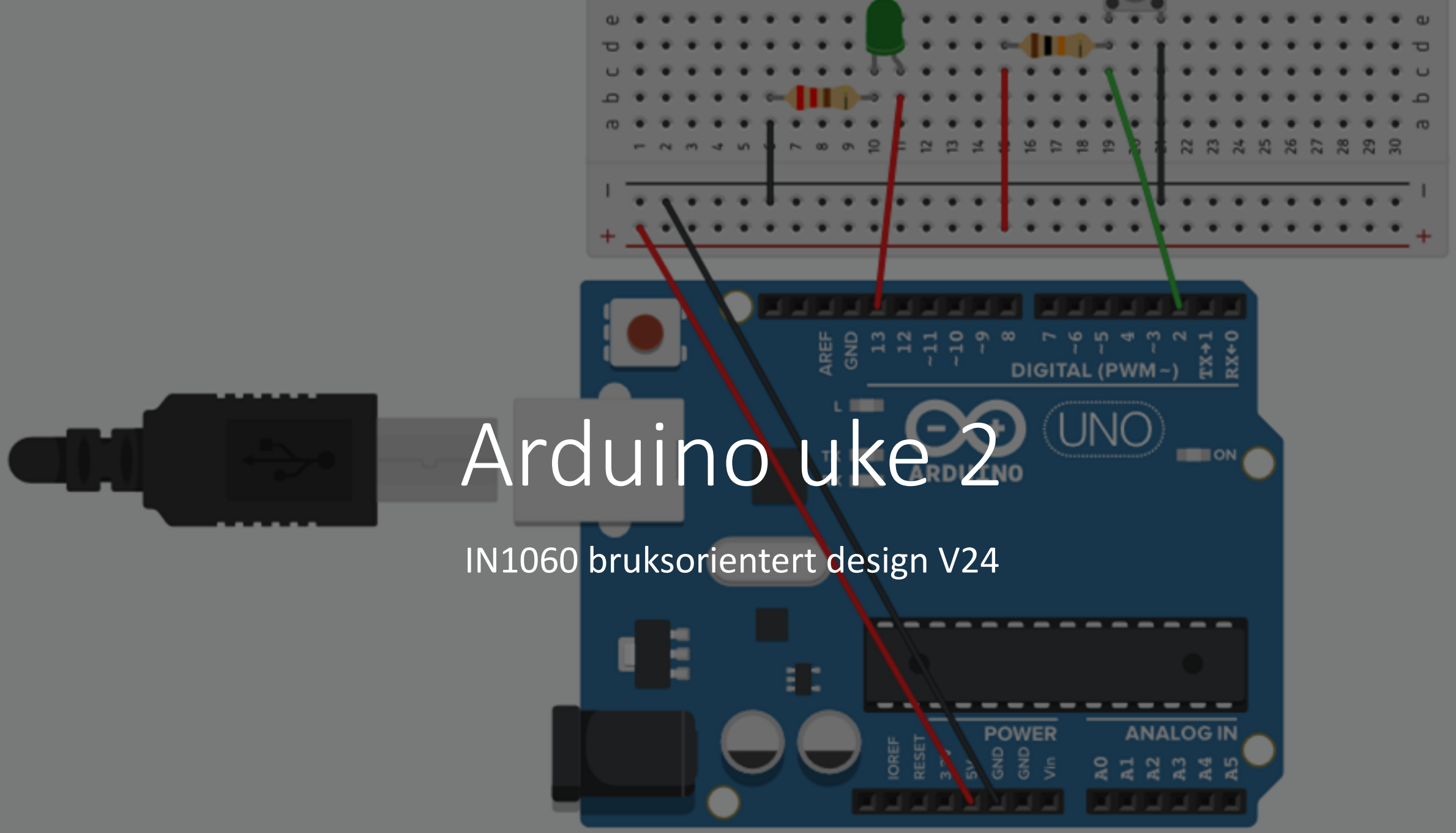
Arduino uke 2

IN1060 bruksorientert design V24



I dag

Første time

1: Kretser og breadboardet

2: Tegning av kretsdiagrammer med TinkerCad

3: Knapper og to vanlige problemer

4: Analoge og digitale signaler

Serial: et viktig verktøy for problemløsning

map(): en innebygget funksjon vi kan bruke med analoge signaler

Andre time: Kode

1: Modulere kode: ordne kode i funksjoner

2: Jobbe med tid: millis() og delay()

3: String()

4: Løkker og arrays (hvis det er tid)

Vår første elektriske krets: Blinke ekstern LED

Vi trenger:

Breadboard

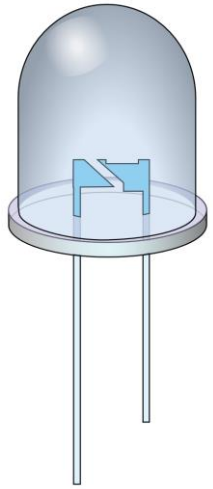
LED

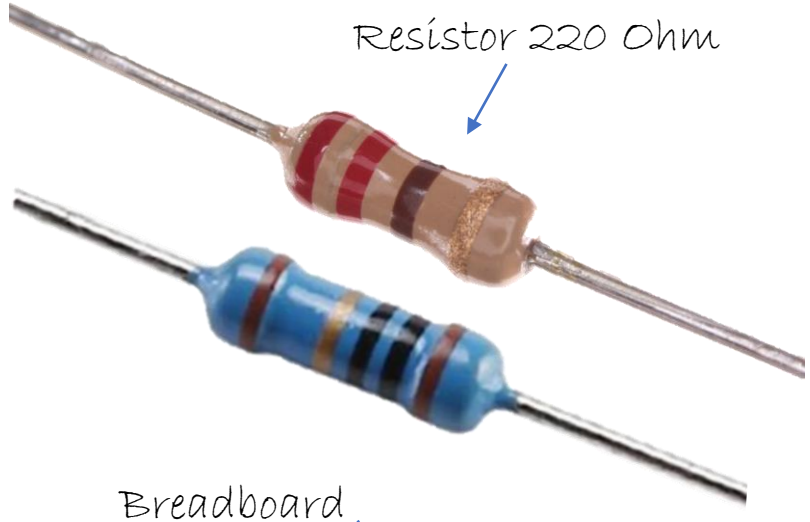
Resistor 220 Ohm

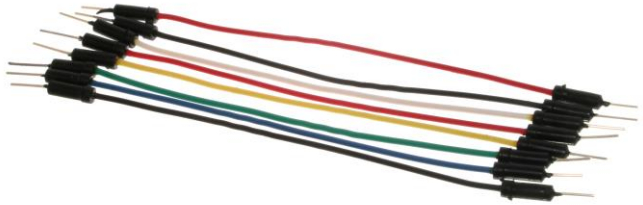
1 svart ledning

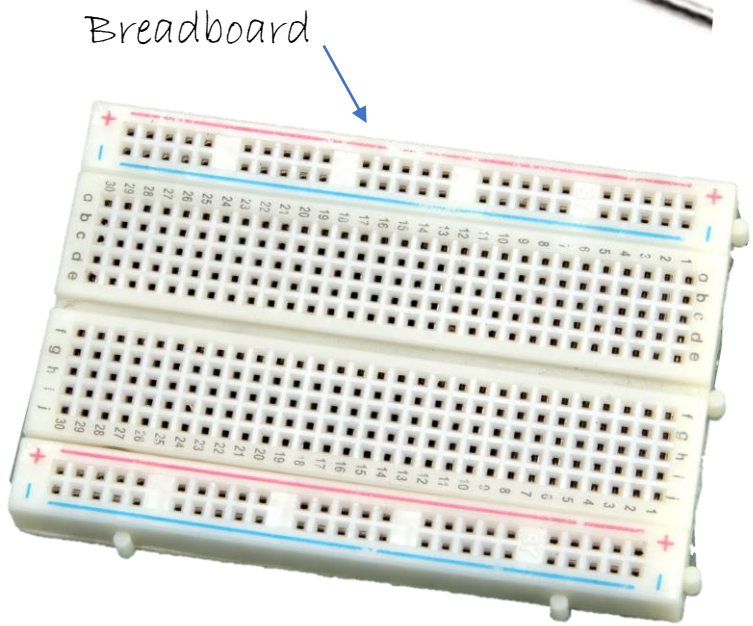
1 rød ledning

Skisse: modifisere Blink-skissen









Bygge kretsen

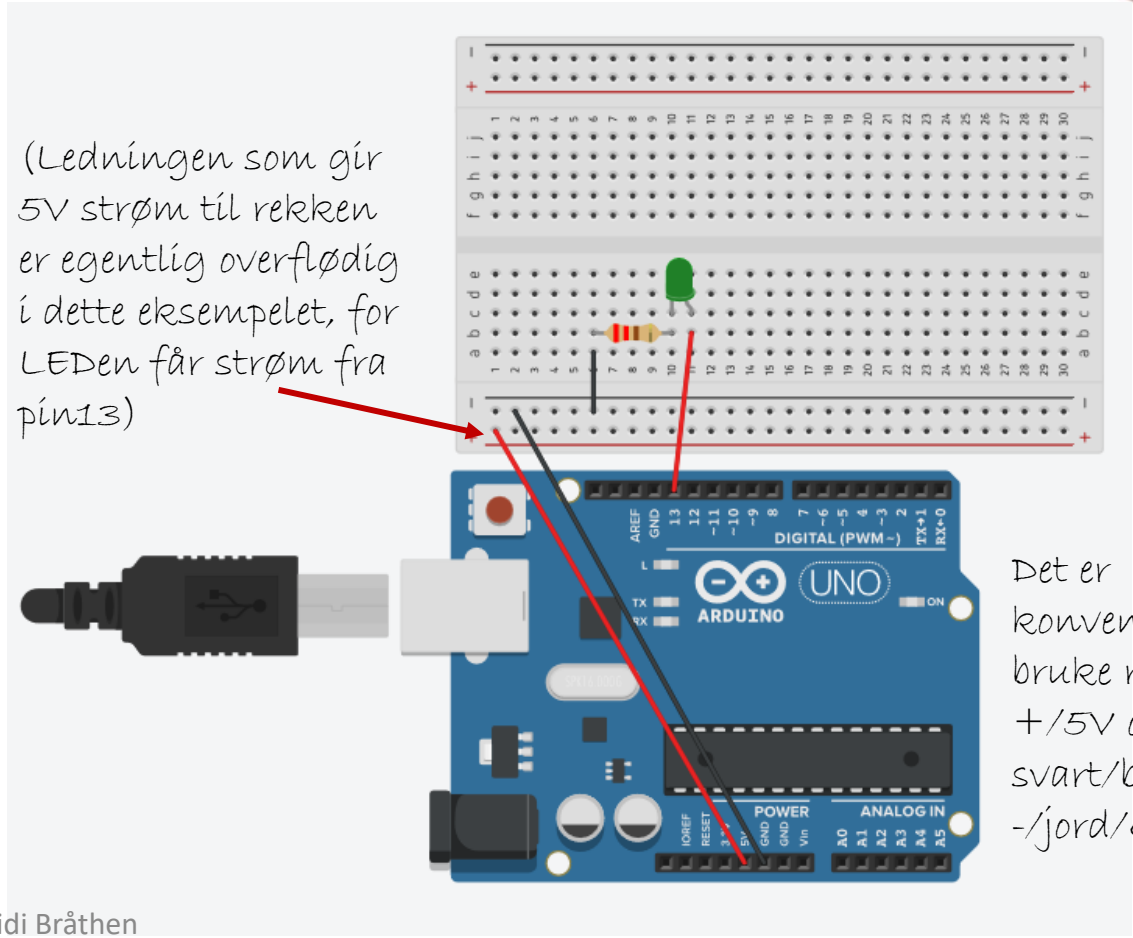
Start med
1: å sjekke at strømmen er koblet fra
2: rød ledning fra 5V til rød rekke på breadboardet
svart ledning fra GND til svart(blå) rekke

Rød ledning fra pin 13 på Arduinoen til breadboardet

Det lengste benet (+) til LED-en i samme rad som den røde ledningen (bøy det lengste benet litt)

220 ohms resistor fra det korteste benet (-) på LEDen (Resistoren kan settes begge veier)

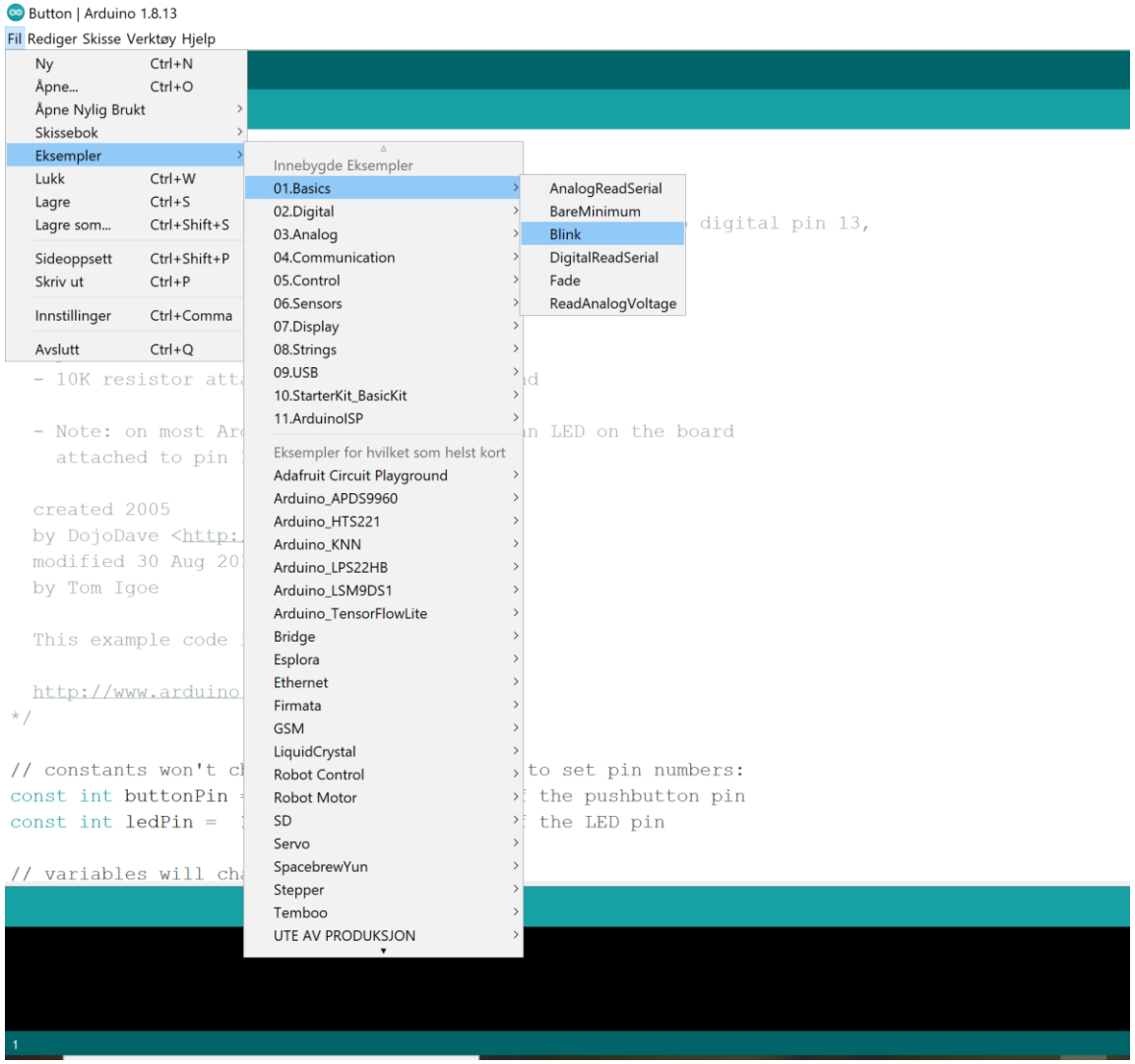
Svart ledning fra resistoren til GND på Arduinoen



Åpne Arduino IDE

I Arduino IDE på PCen, åpne
Fil – Eksempler – 01 Basics – Blink

(samme som i forrige time)



Blink-skisse til egendefinert pin

void setup() {

int ledPin = 13; //Når vi ikke bruker den innebygde LED-en må vi initialisere (lage en variabel og gi den en verdi) en variabel for pin-en vi velger

pinMode(ledPin, OUTPUT); //vi bruker pinMode til å bestemme om det er signal ut eller inn på pin-en, som tidligere

- }

void loop() {

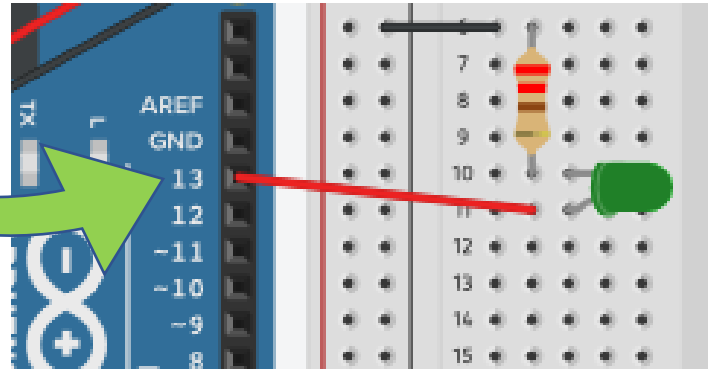
digitalWrite(ledPin, HIGH); //sender strøm til variabelen vi laget, ledPin, og slår på LED-en

delay(1000); //pauser programmet i ett sekund

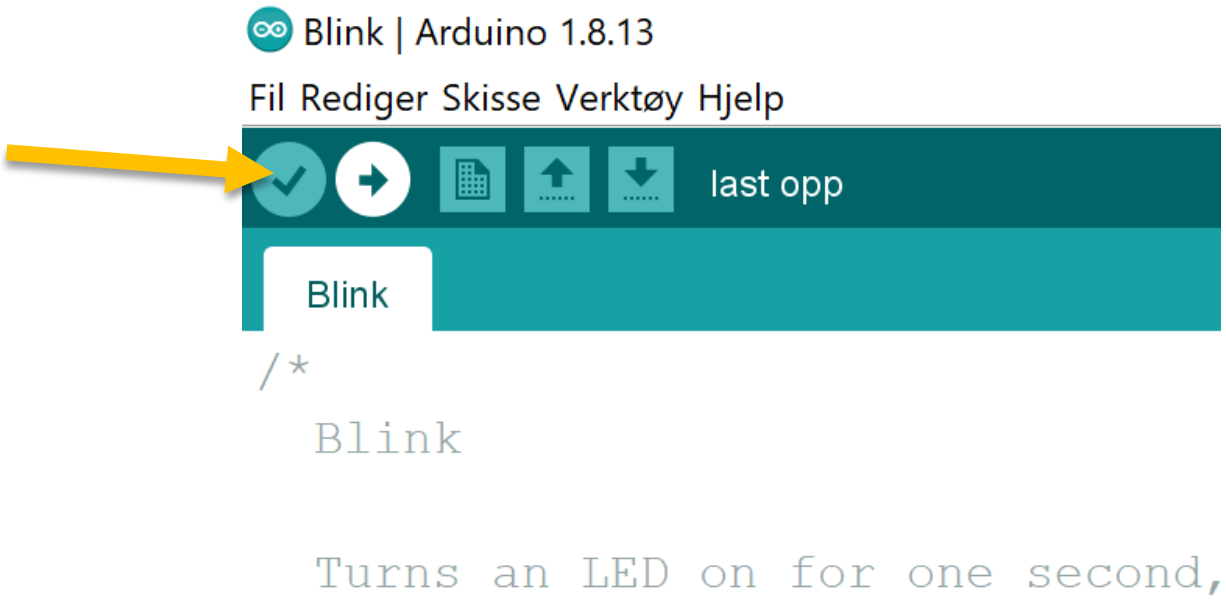
digitalWrite(ledPin, LOW); //slår av strømmen til ledPin, og slår av LED-en

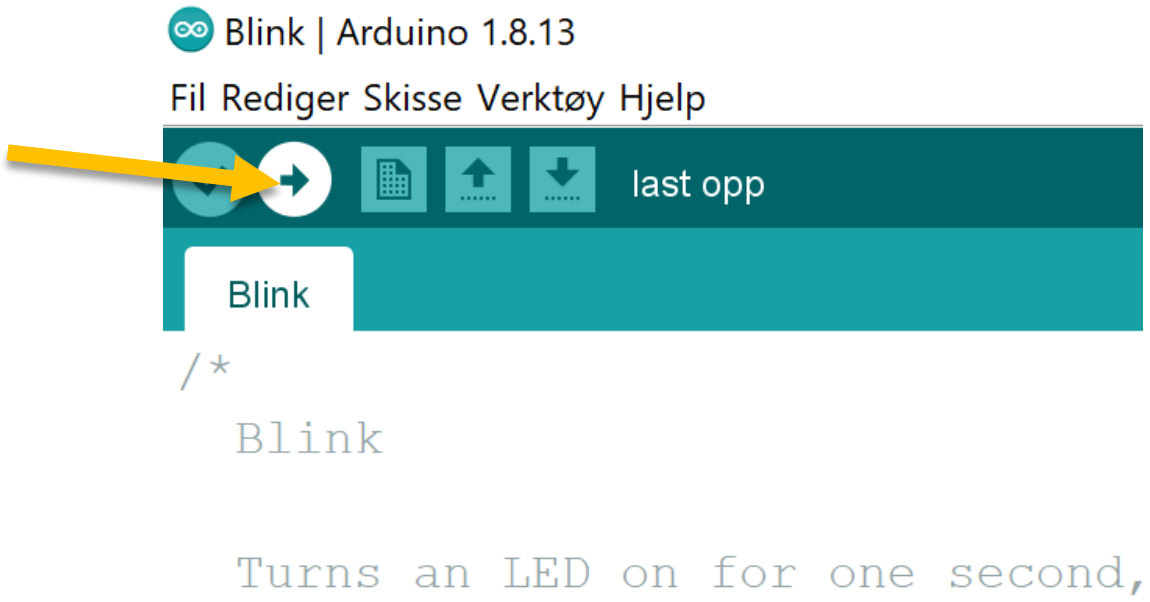
delay(1000); //pauser programmet i ett sekund

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

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



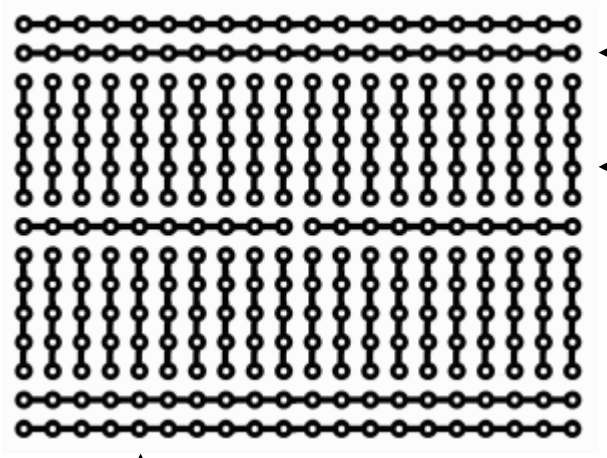
Kompiler og last opp skissen til Arduino





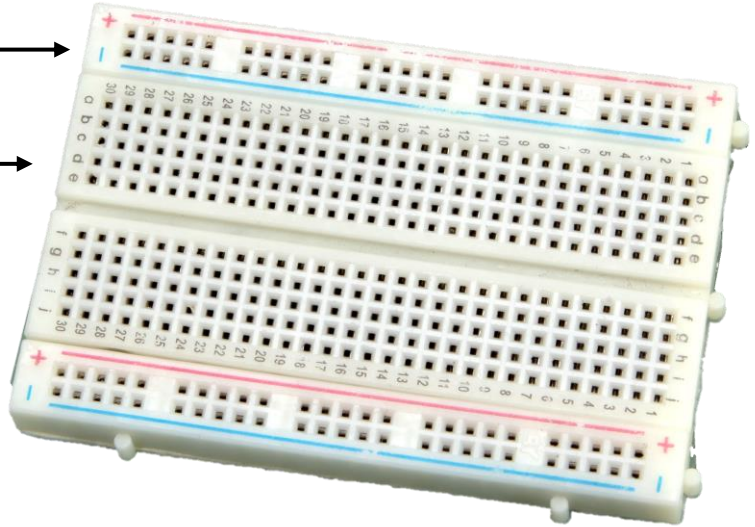
Kretser på breadboardet

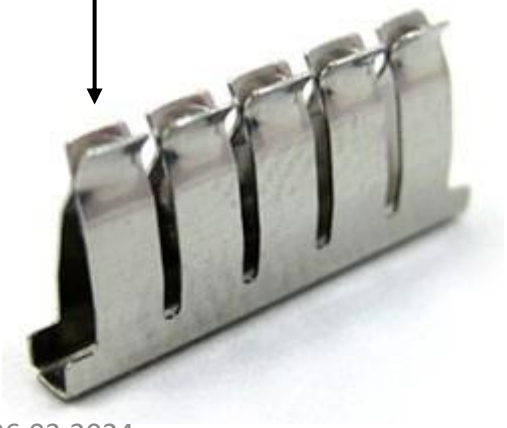
Hvordan virker breadboardet?



Skinner

Rader





Breadboardet er et system av ledninger med klips for enkel montering av komponenter i en krets

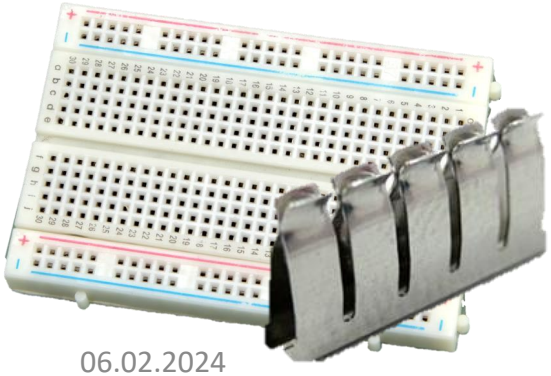
De horisontale lange stripene merket med - og + kalles skinner.

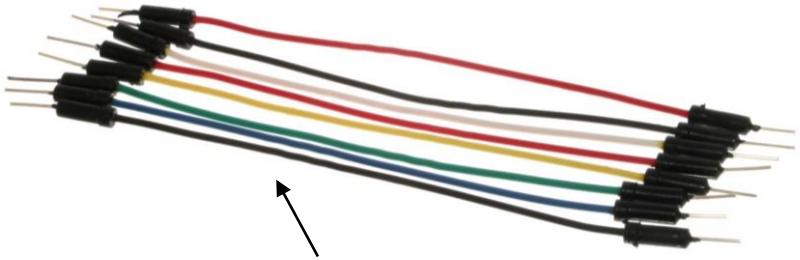
De vertikale stripene merket med tall kalles rader

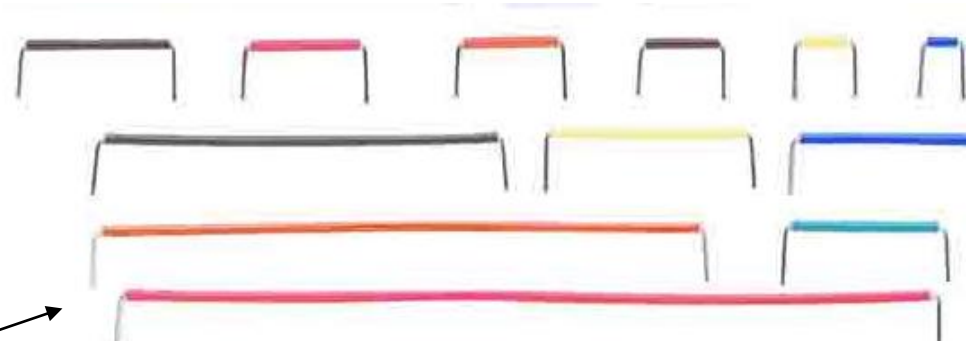
De er laget av rekker med små klips i metall som holder fast komponentene

- Hva er egentlig en ledning?

- En ledning består av et strømførende materiale, ofte et metall.
- For at ledninger til + og – ikke skal komme borti hverandre og danne kortslutning må vi isolere metallet. Plast og gummi leder strøm dårlig, og er derfor gode isolasjonsmaterialer.
- I settene deres har dere ulike typer ledninger. Noen ser litt uvanlige ut og de kan ha ulike farger, men alle virker på samme måte.



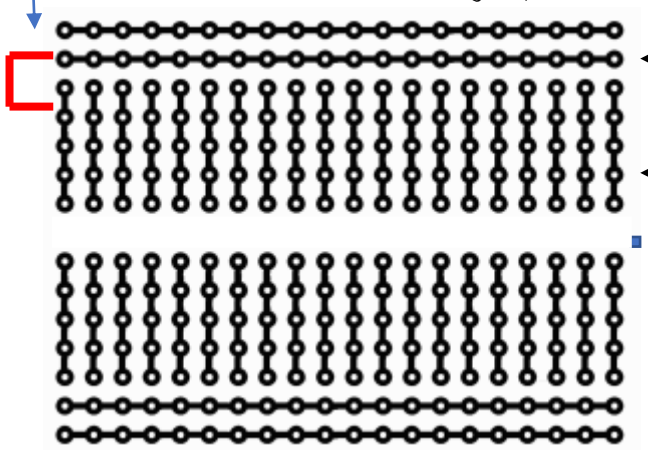




Alle disse er ledninger

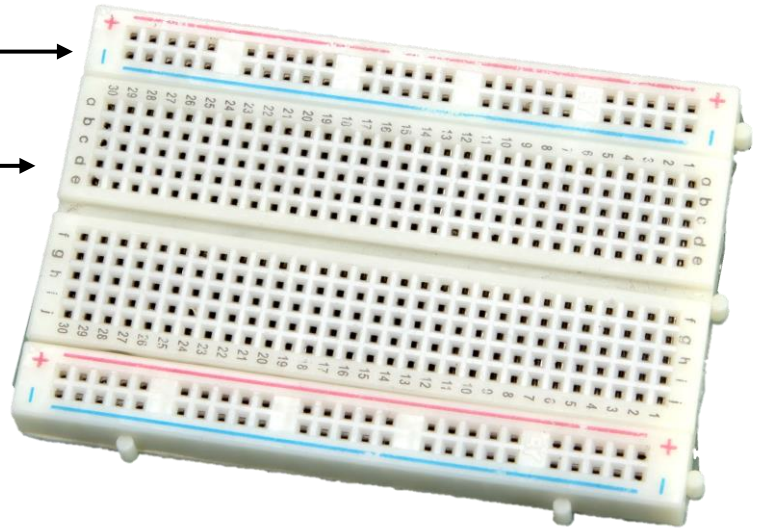
Hvordan er breadboardet ledninger?

Radene/rekkene har ikke kontakt med hverandre, og må kobles sammen ved hjelp av andre ledninger

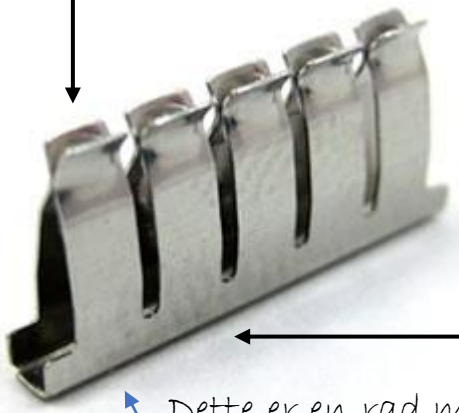


Skinner

Rader



Under plasten er det rader med metall organisert som på tegningen. Rader består av 5 kontakter/hull og skinnene har 20 (en for hver rad)



Skinnene og radene er laget av rekker med små metallklips i strømførende metall. **De virker akkurat som ledninger.** Plasten i breadboardet virker som isolasjon.

Dette er en rad med 5 kontakter/klips.

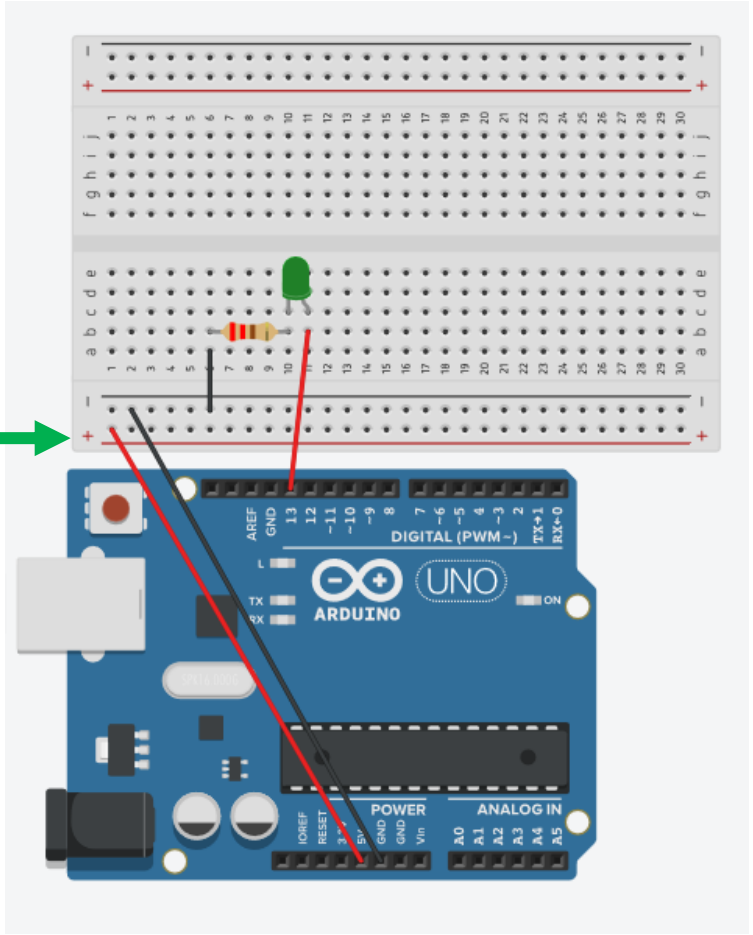
Klipsene inne i breadboardet

Hvorfor alltid trekke ledninger til rekkene?

I forrige forelesning anbefalte jeg å trekke ledninger fra Arduinoens 5V og GND til rekkene først

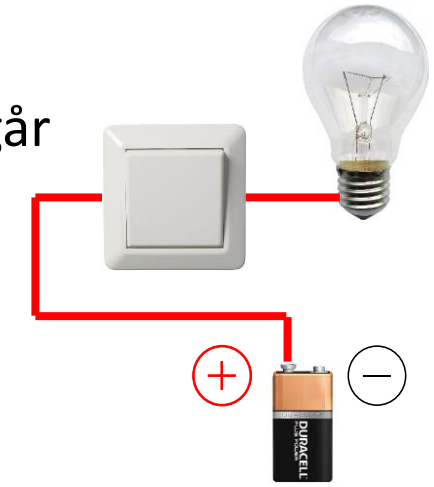
Men i eksempelet ble ikke 5V-rekken brukt til noe, den hadde ingen funksjon!

Grunnen til anbefalingen er at Arduinoen kun har to 5V-pinner og tre jord, og at man fort slipper opp for pin'er når det blir flere komponenter. Da er det nyttig å alltid ha strøm og jord tilgjengelig i rekkene



Hvorfor må kretsen gå til jord/GND?

Ingenting vil virke hvis ikke hele kretsen går til samme jord/GND-punkt

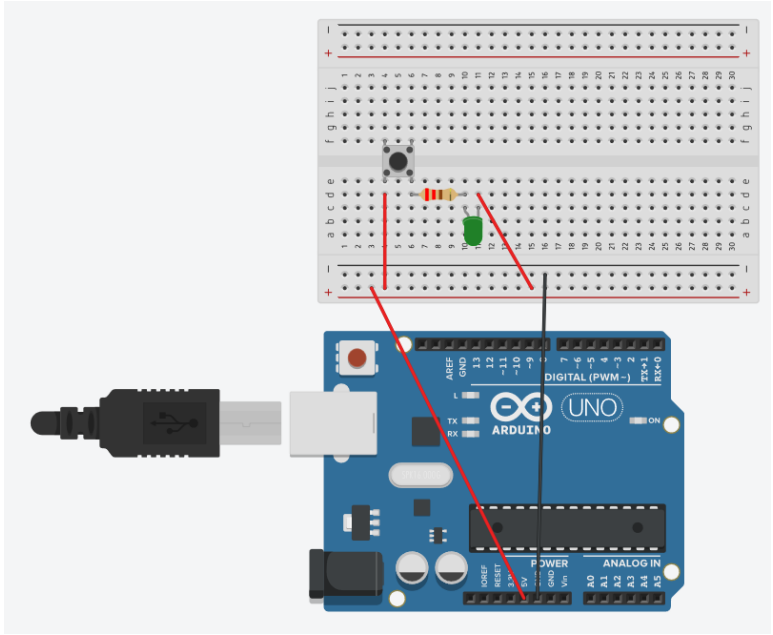


Hvis ikke kretsen går til jord, vil ikke kretsen være sluttet (hel), fordi den ikke går fra pluss til minus. Da kan ikke elektronene strømme, for det blir ingen spenningsforskjell som dytter dem.

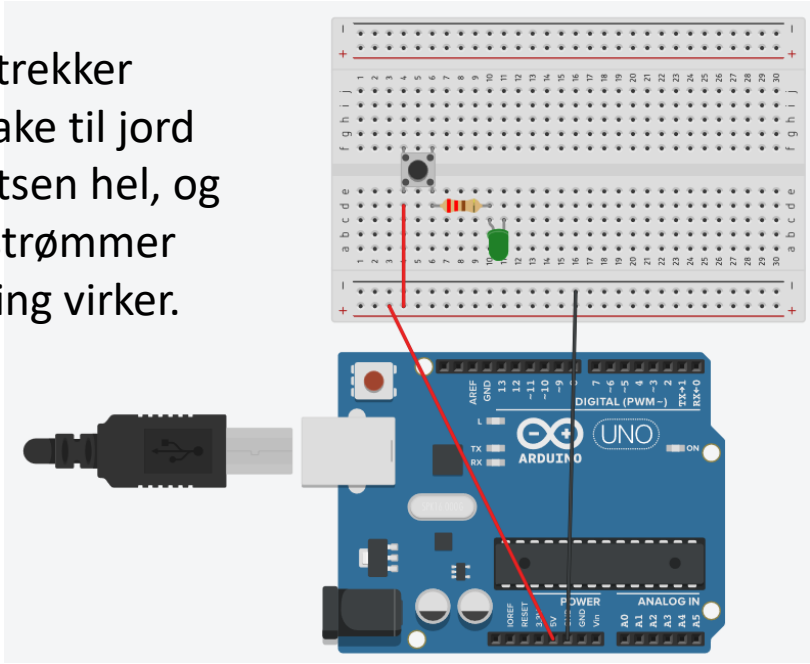
Jord/GND på Arduino er et *referansepunkt* hvor det 0 volt. Det betyr at alle andre signaler blir målt mot dette punktet som representerer 0 V, og siden alt blir målt mot samme referansepunkt, kan man måle alle verdiene i forhold til hverandre slik at de gir mening i kretsen.

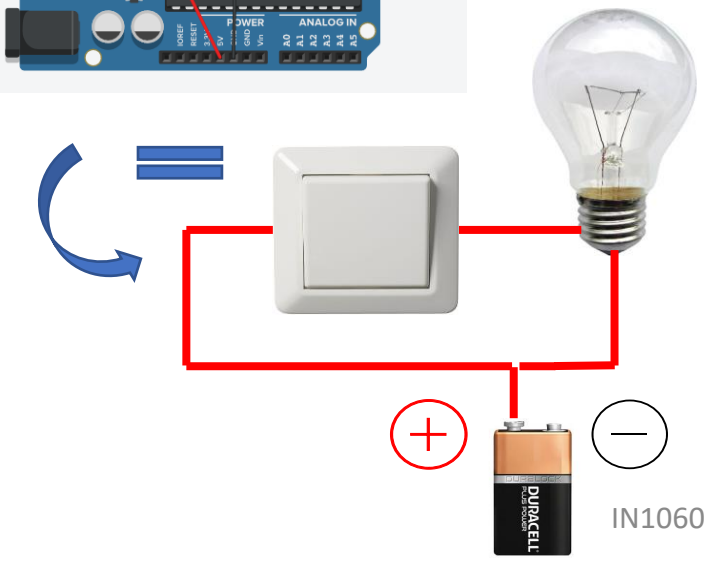
På store strømkraftverk og i elektriske systemer i hus er jord/GND ledet ned i den faktiske jorda/bakken. Det er vanlig praksis å jorde kretsen med et jordspyd, en meterlang metallbolt, som er drevet ned i den faktiske bakken

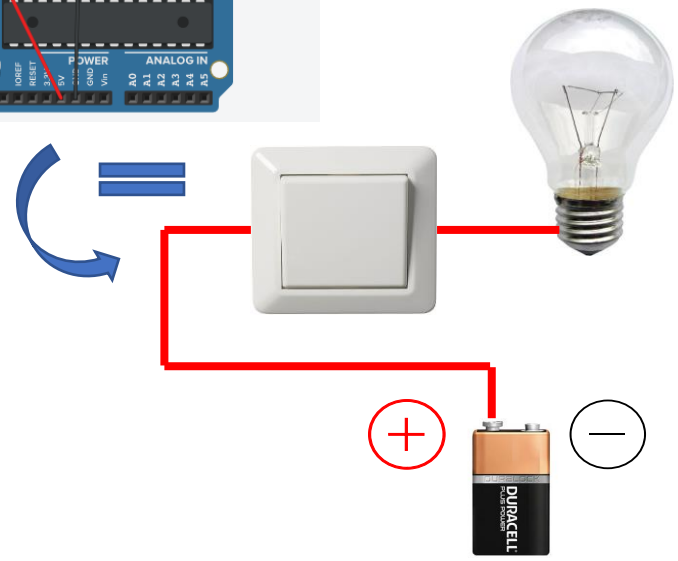
Kretsen må kobles tilbake til jord/GND for at kretsen skal bli hel



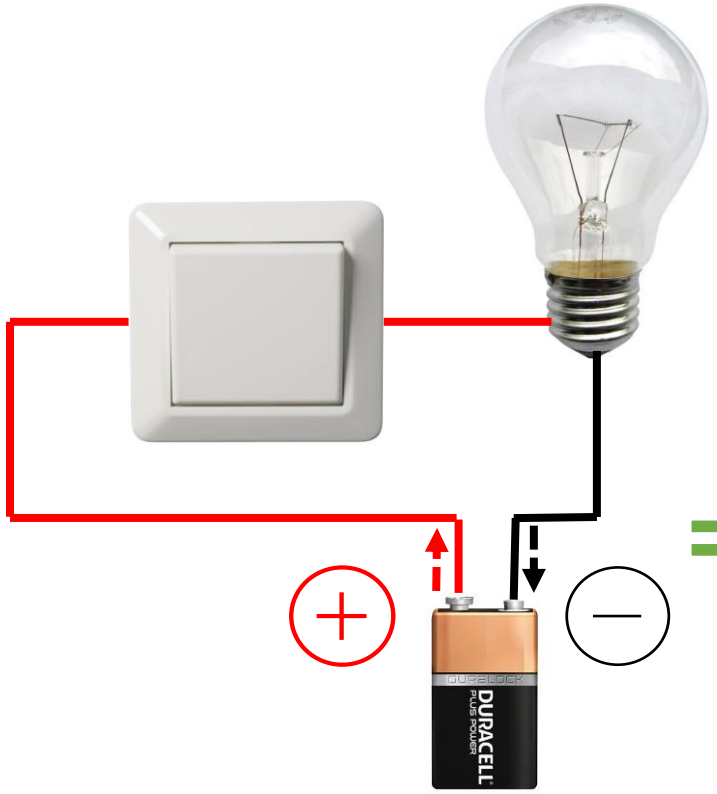
Hvis vi ikke trekker kretsen tilbake til jord blir ikke kretsen hel, og strømmen strømmer ikke, ingenting virker.



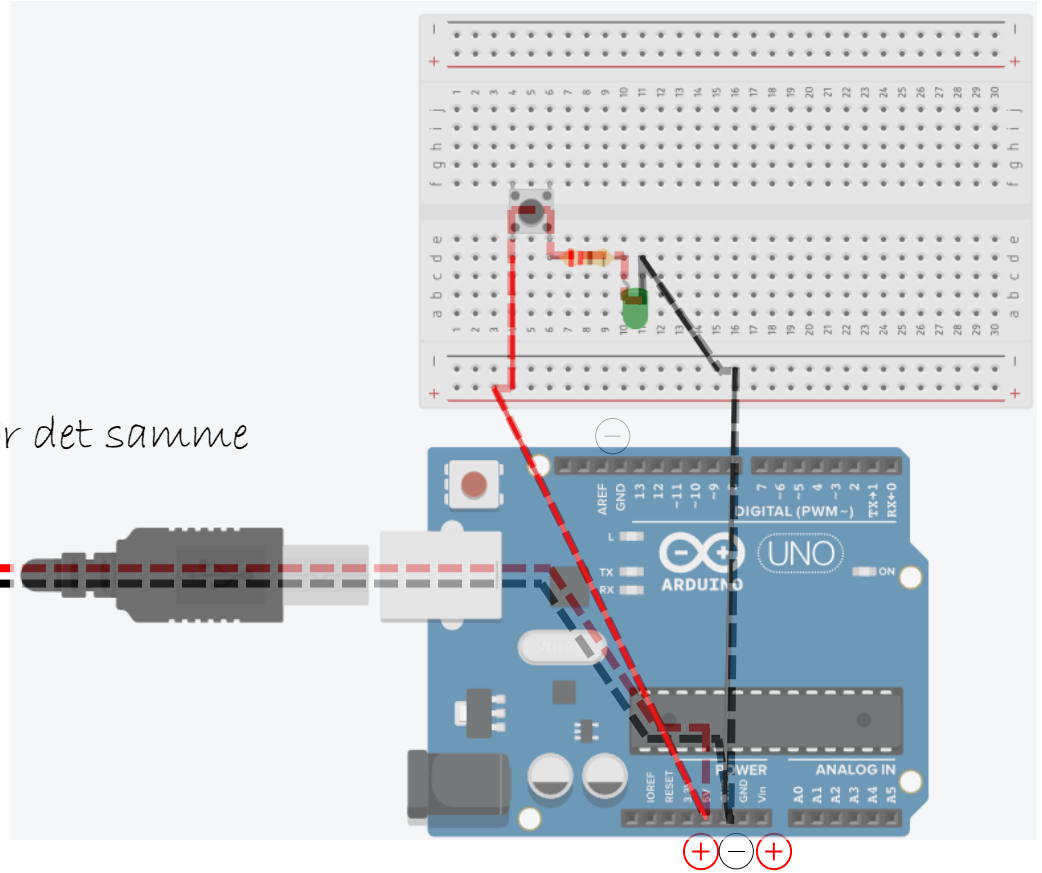




Bygge kretser på Arduino og breadboard



Disse kretsene gjør det samme

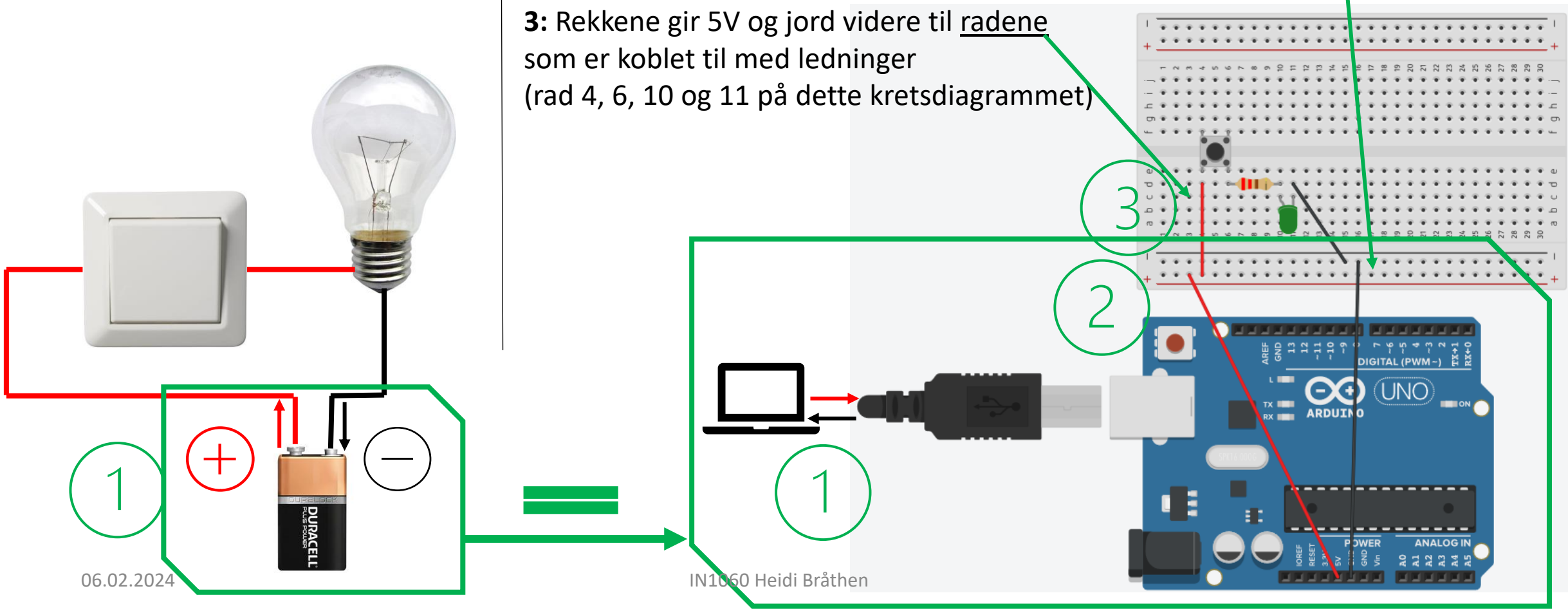


Det er kanskje ikke så lett å se at disse kretsene er like, for Arduinoen og breadboardet legger til litt flere elementer i den delen av kretsen som er spenningskilden (batteriet/ fra pluss til minus)

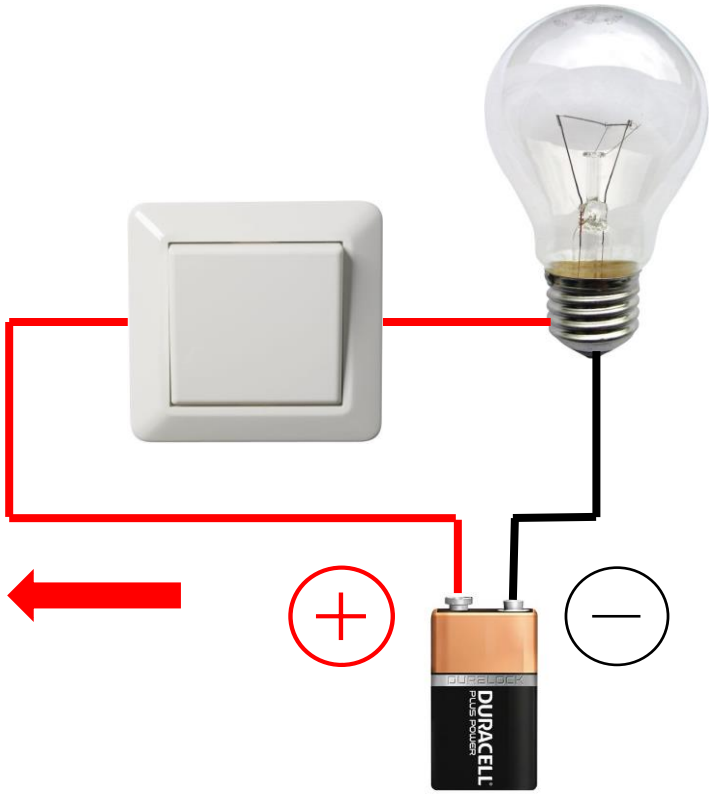
Spenningskilden får litt flere ledd

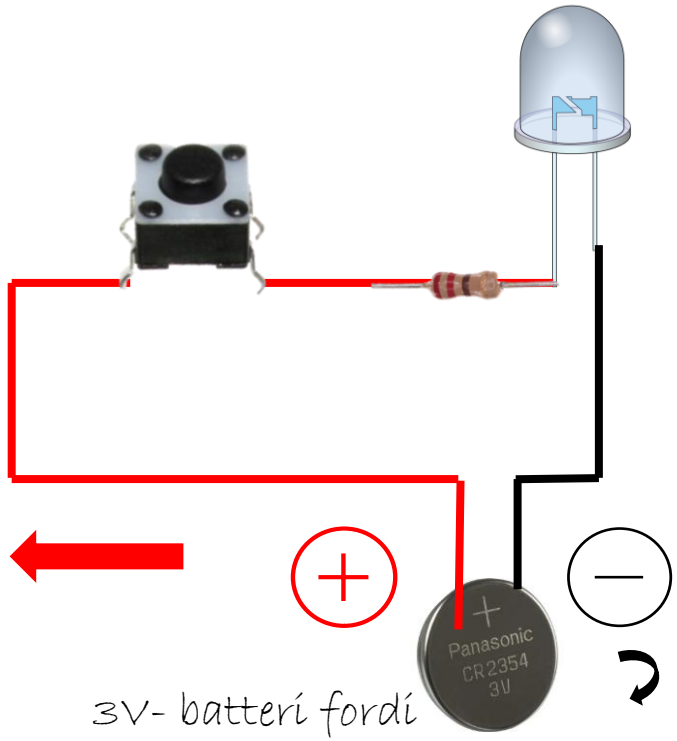
Spenningskilden til denne kretsen:
1: Batteriet gir 9 volt og jord til kretsen

Spenningskilden til denne kretsen:
1: PCen gir 5V og jord til Arduinoen
2: Arduinoen gir 5V og jord videre til breadboardets «+ og –» skinner
3: Rekkene gir 5V og jord videre til radene som er koblet til med ledninger
(rad 4, 6, 10 og 11 på dette kretsdiagrammet)

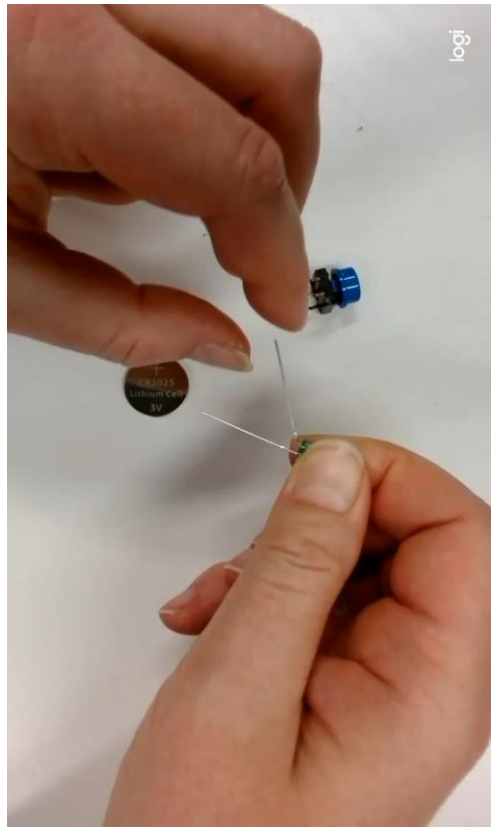


Fra krets til breadboard steg for steg





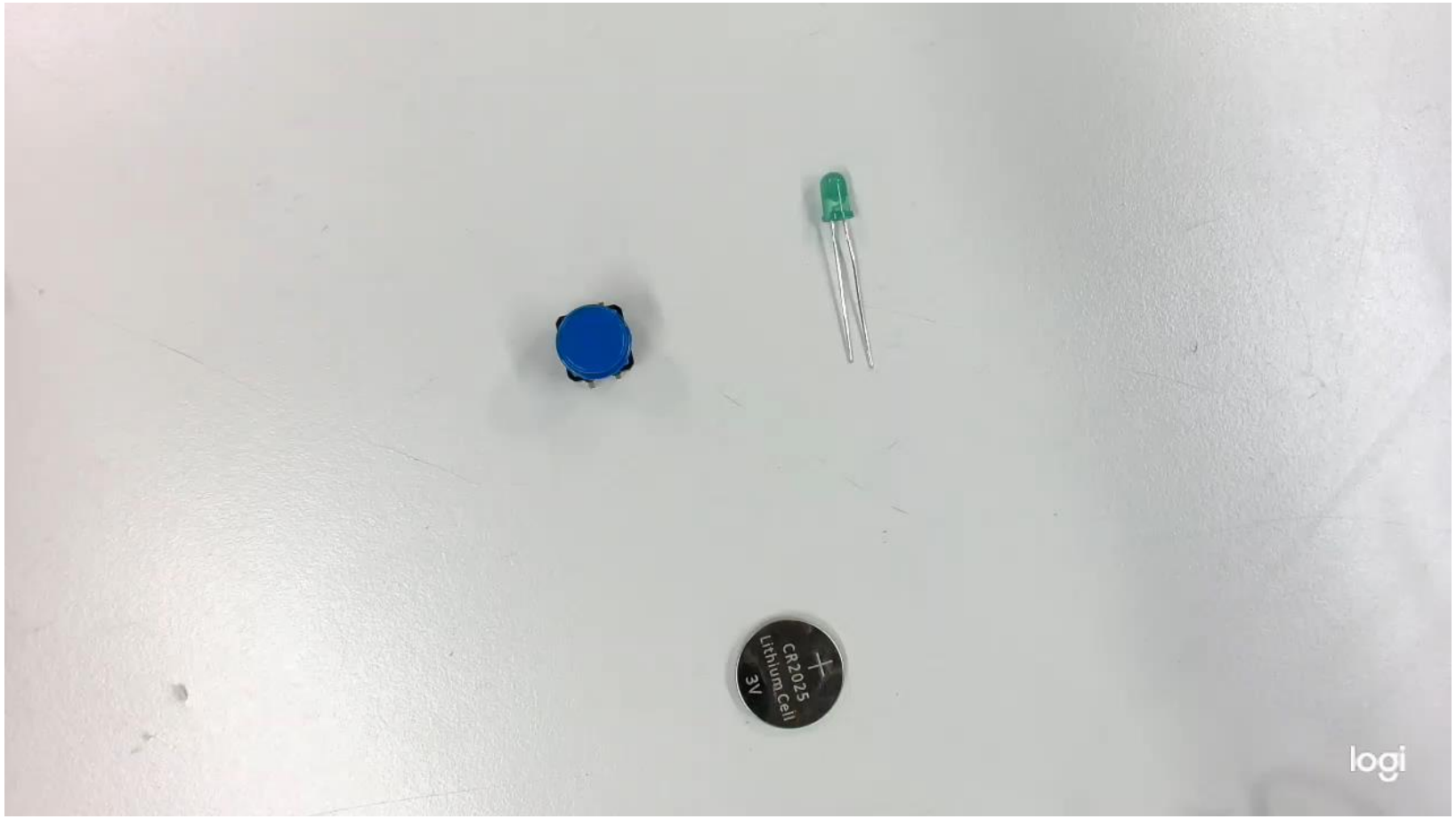
3V- batteri fordi LEDen ikke tåler mer strøm



3V-batteriet kan gi strøm rett til LEDen

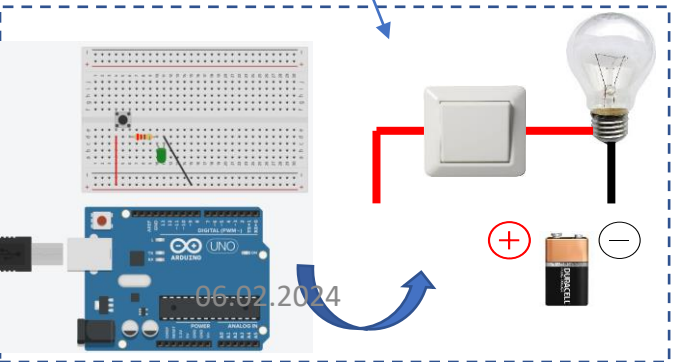
Klokkebatterier har plusspolen foran og minuspolen bak

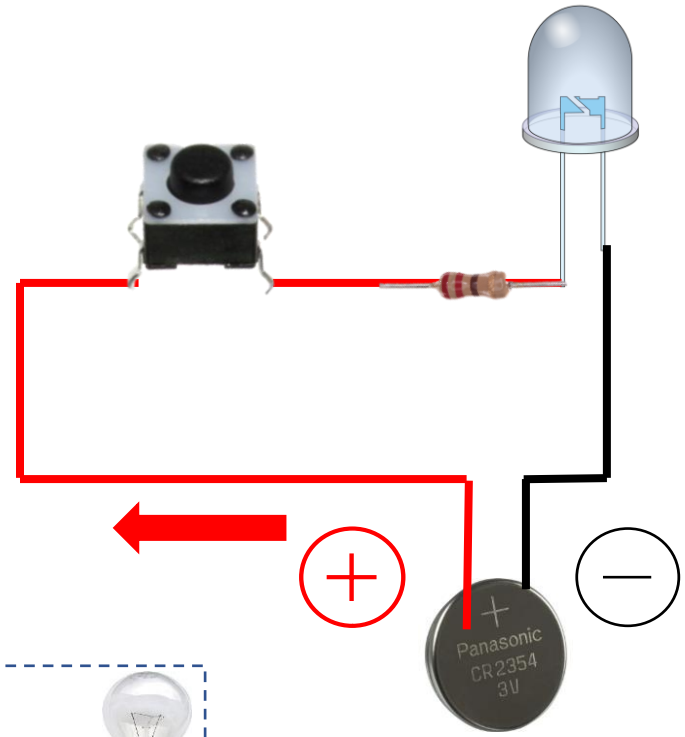
Koble en enkel krets med alligatorklips

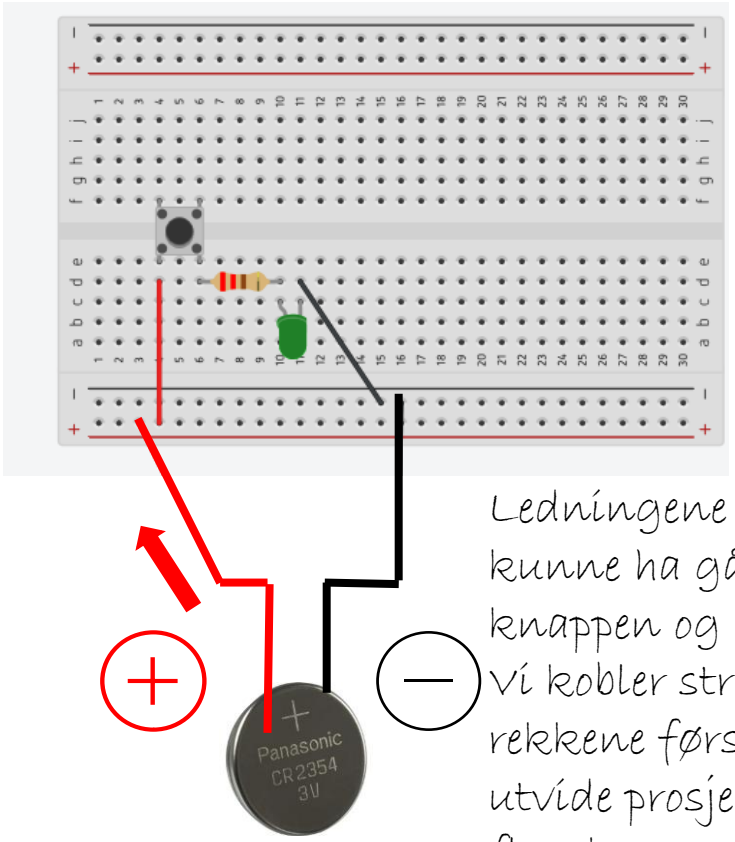


Breadboardet har ikke noen egen strømkilde

Hvis vi ikke trekker ledninger fra arduinoen til breadboardet finnes det ikke noe strøm tilgjengelig på breadboardet:

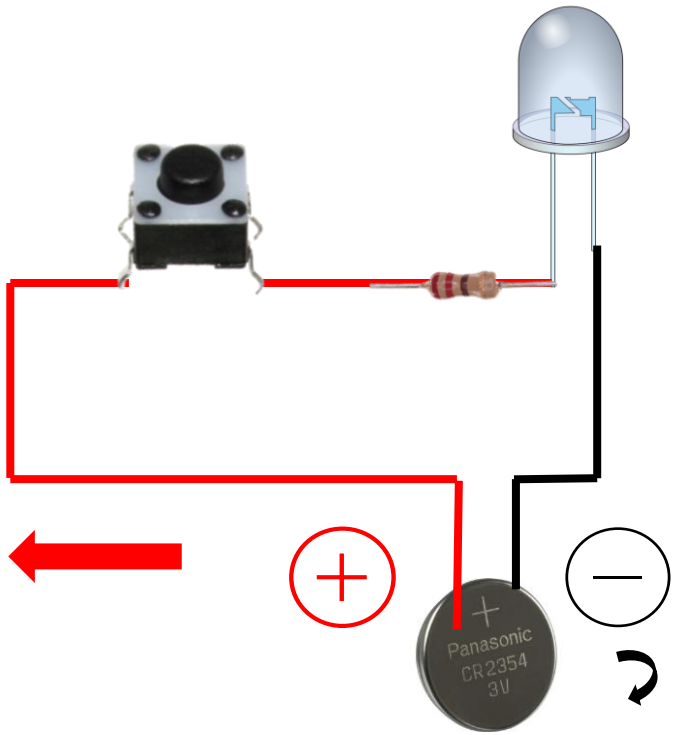


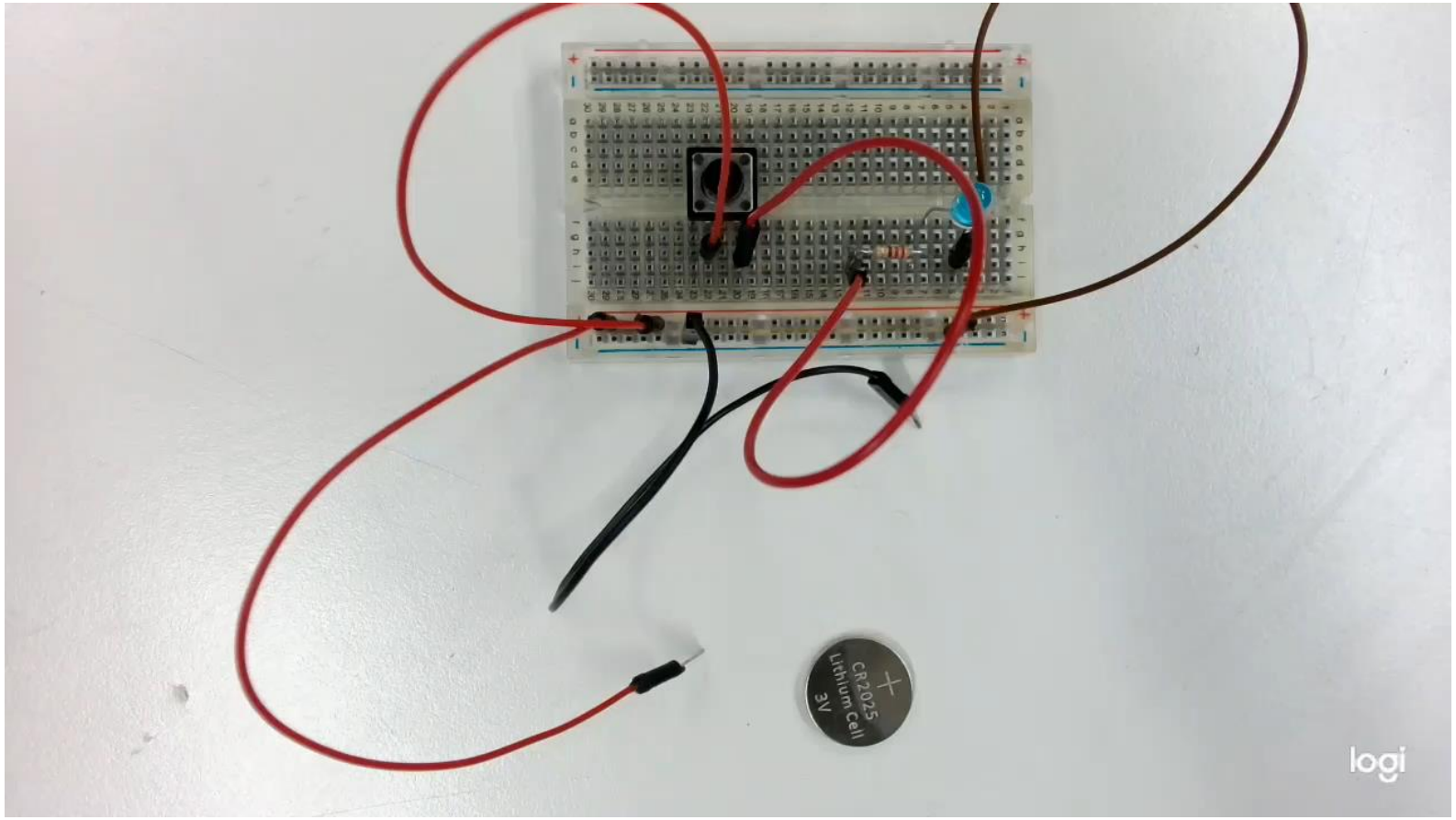




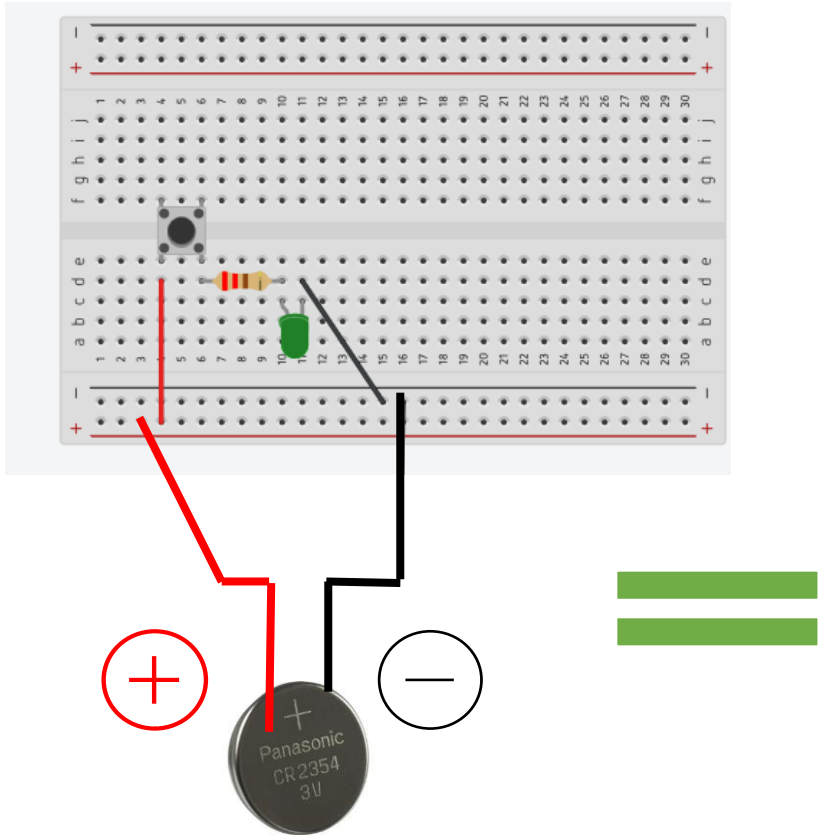
Ledningene fra batteriet kunne ha gått rett til knappen og rett fra LEDen. Vi kobler strøm og jord til rekkene først slik at vi kan utvide prosjektene våre med flere komponenter

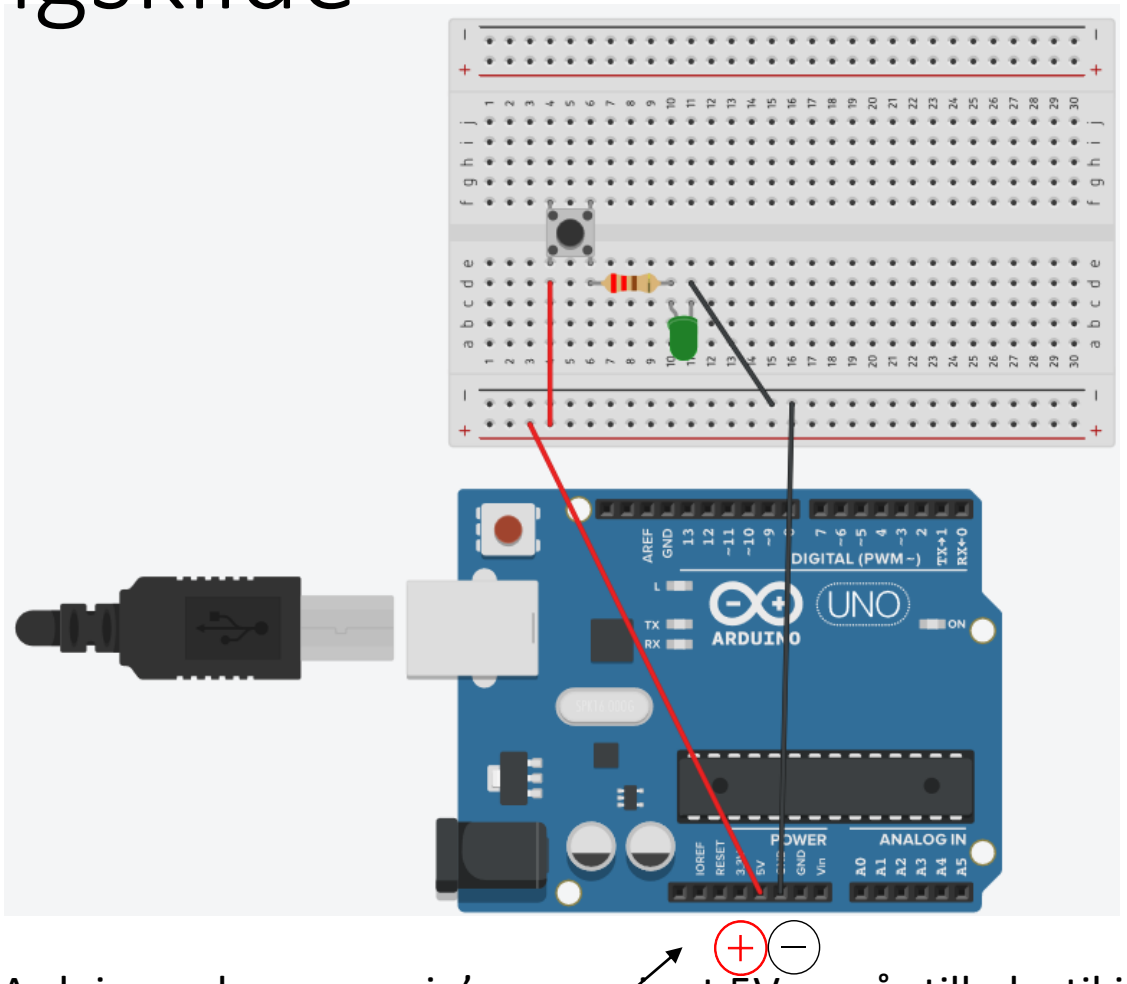
Vi bruker breadboard fordi det gjør det enkelt å koble sammen mange komponenter på en gang. Breadboardet har ingen egen strømkilde, så vi må koble på batteri eller gi strøm fra Arduino.


Panasonic
CR2354
3V


CR2025
Lithium Cell
3V
logi

Arduinoen som spenningskilde

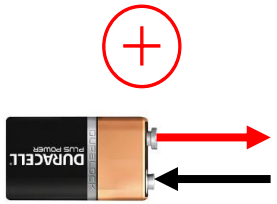




Hvis vi skal programmere knappen og LEDen til å gjøre mer enn å skru LEDen på når knappen trykkes ned, og skru av LEDen når knappen slippes opp, må vi programmere den. Da trenger vi Arduino.

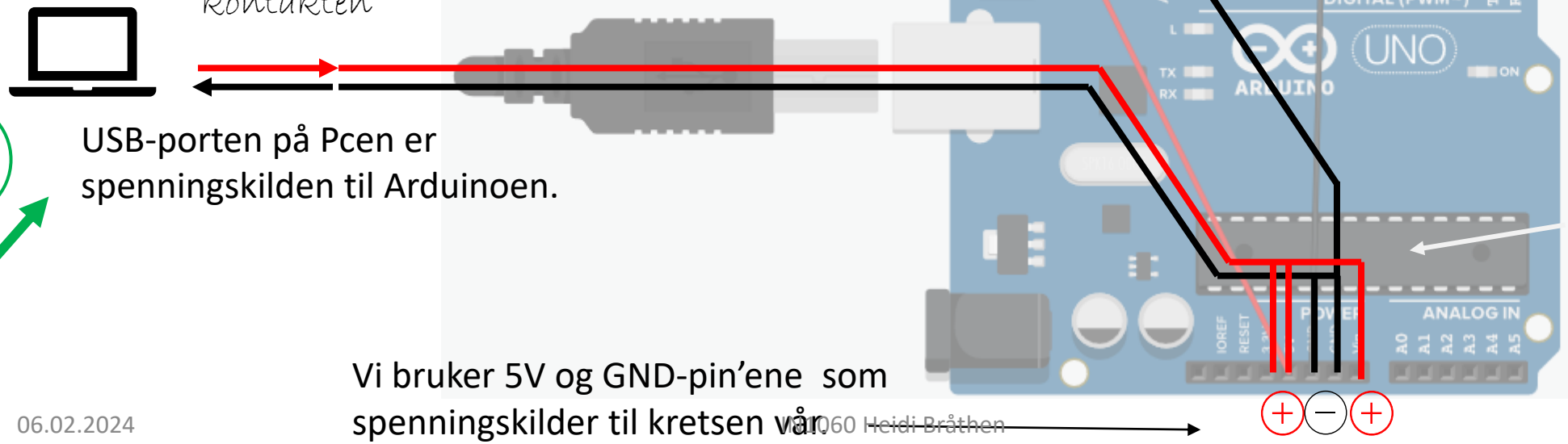
Arduinoen har egne pin'er som gir ut 5V og går tilbake til jord. Disse pin'ene er spenningskilden slik som et batteri. Vi kan hente strøm fra Arduinoen over til breadboardet ved å koble ledninger fra 5V og GND til den lange røde og svarte rekken

1: PCen/USBen er Arduinoens spenningskilde



Batteriet var spenningskilden i den første kretsen

USBen har en strømledning og en jordledning inne i kontakten



1

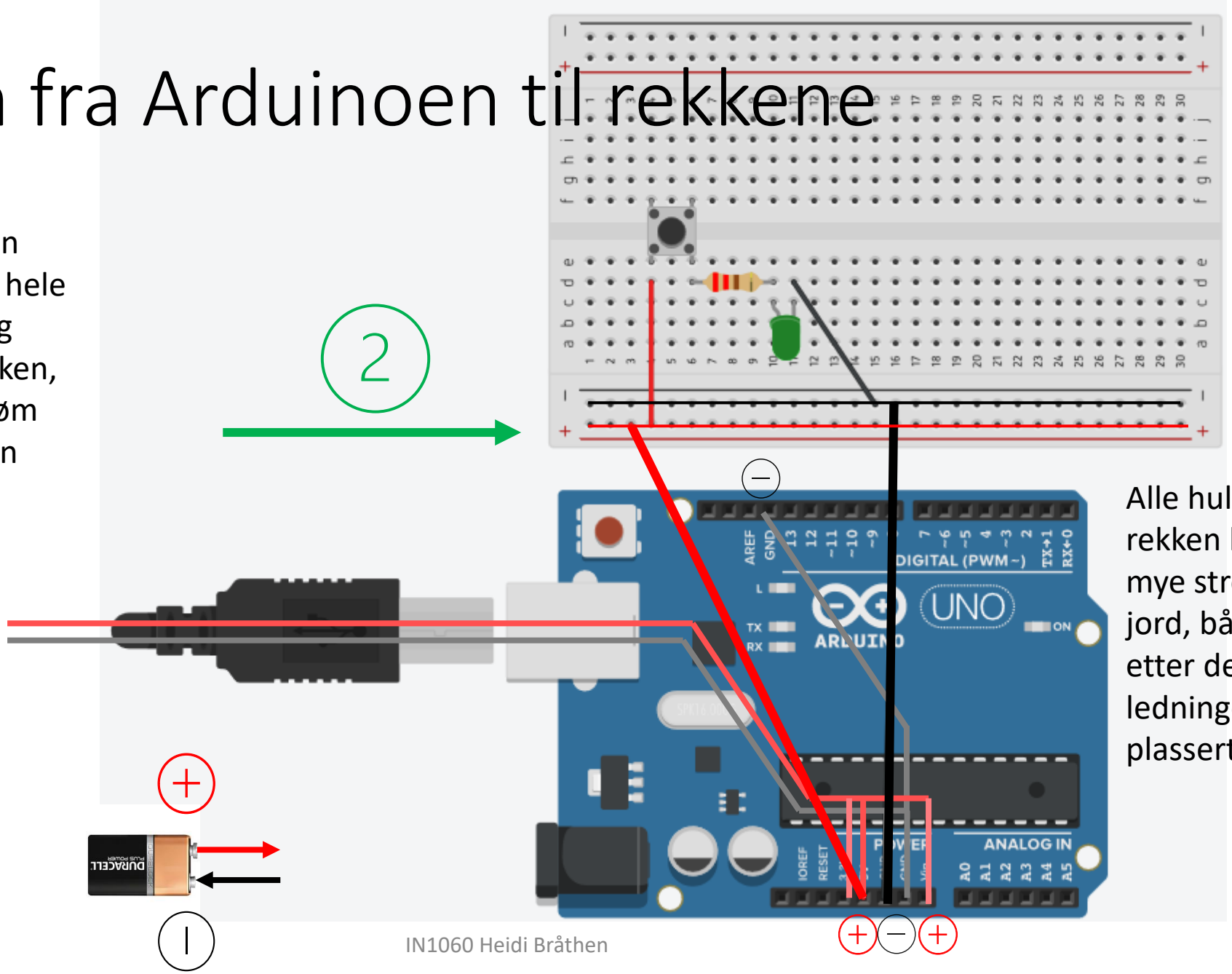
USB-porten på Pcen er spenningskilden til Arduinoen.

Vi bruker 5V og GND-pin'ene som spenningskilder til kretsen vår.

Mikrokontrolleren har dedikerte bein som håndterer strøm som går ut fra 5 og inn til GND

2: Strøm fra Arduinoen til rekkene

Ettersom det er en ledning under plasten som kobler sammen hele den røde +-rekken og hele den svarte –rekken, gir Arduinoen nå strøm og jord til hele rekken

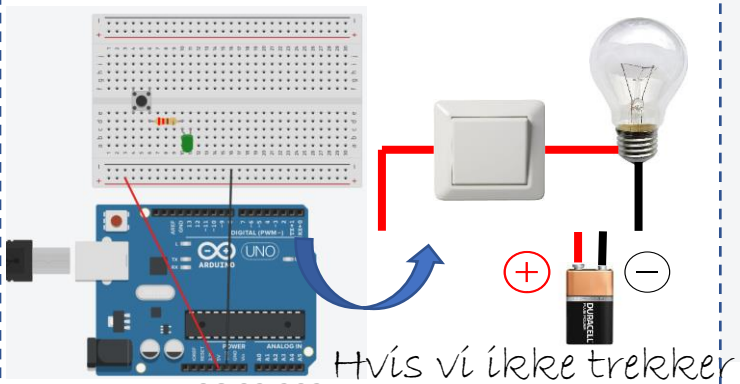


Alle hullene i rekken har nå like mye strøm eller jord, både før og etter der hvor ledningen er plassert.

3: Strøm fra rekker til rader

For at knappen skal få strøm, må vi trekke en ledning fra strøm-rekken + over til den raden hvor knappen er koblet i (her er det rekke 4). For at kretsen skal være sluttet må neste komponent ned i samme rad som der siste komponent slutter, slik som knappen og resistoren er på samme rad nr. 6

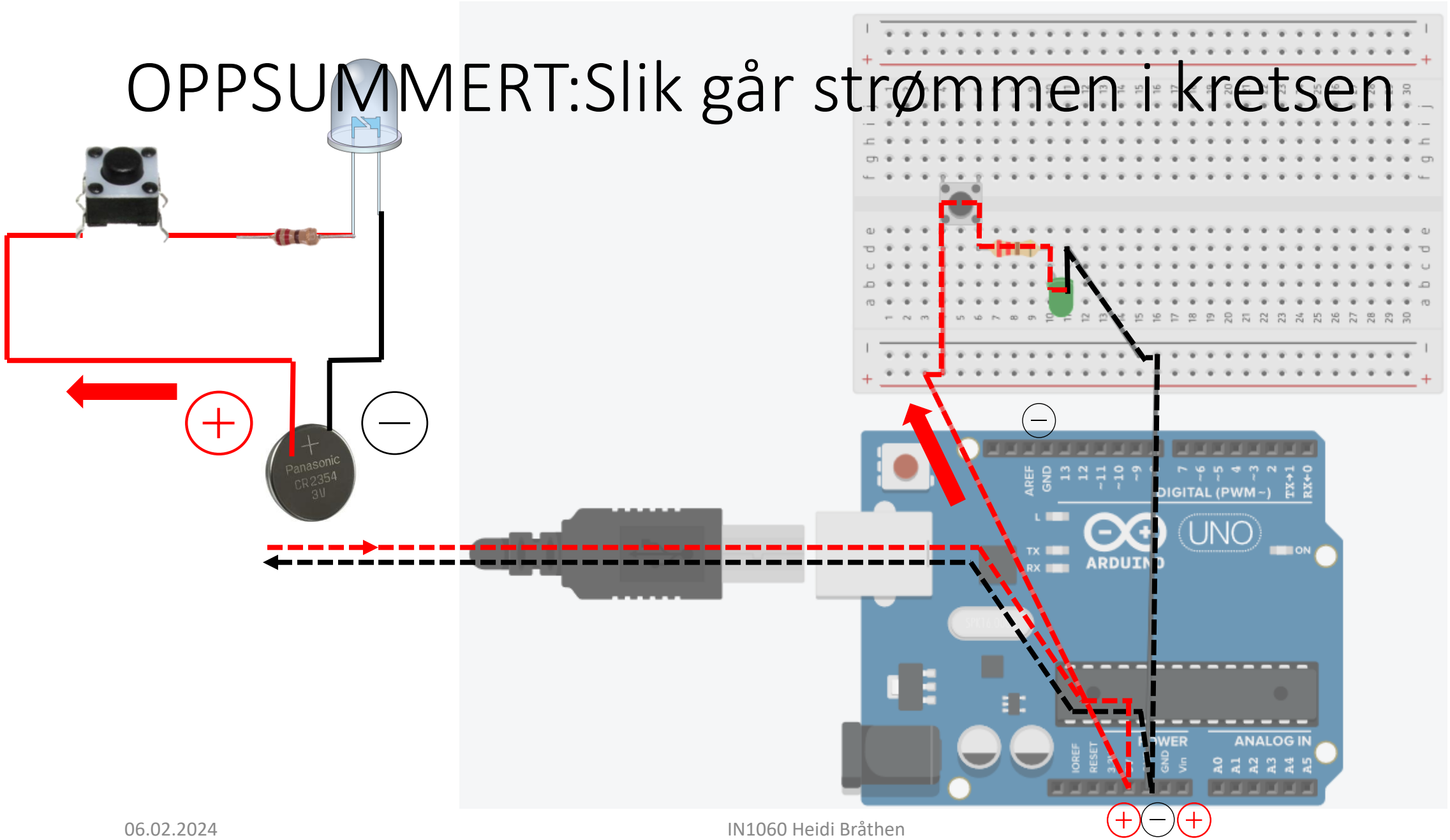
Man kan velge hvilken rad som helst, men man må passe på at ting som skal kobles sammen er i samme rad, ellers får de ikke kontakt



Hvis vi ikke trekker ledninger fra rekkene til radene kan ikke strømmen komme frem til komponentene

Alle hullene i de radene som er koblet i (4, 6, 10 og 11) har nå like mye strøm eller jord, både før og etter der hvor ledningen eller komponentet er plassert

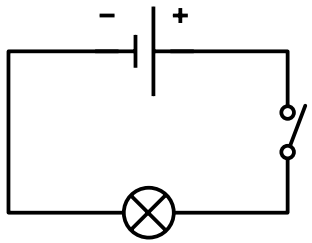
OPPSUMMERT:Slik går strømmen i kretsen



Tegne kretsdiagrammer

Med TinkerCad

Tegning av kretsdiagrammer



Vi bygger kretser etter arbeidstegninger som kalles kretsdiagrammer

Et kretsdiagram er en grafisk fremstilling av hele kretsen som viser hvordan alle delene i kretsen er koblet sammen

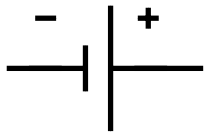
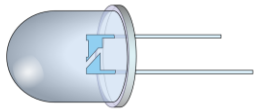
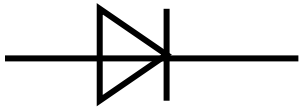
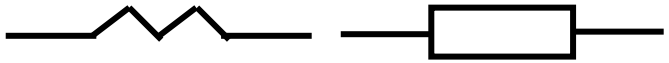
Tegnes med standardiserte symboler for komponenter

Tegnes etter regler og konvensjoner

Hvis vi designer kretsen selv, tegner vi et kretsdiagram som dokumentasjon slik at vi og andre kan gjenskape og feilsøke kretsen

Vi kan tegne kretsdiagrammer i TinkerCad https://www.tinkercad.com

Konvensjoner

Komponent	Navn på komponent	Symbol
	Spenningskilde/batteri	
	Ledning	
	LED	
	Resistor	

Strømmen flyter fra pluss til minus

+ tegnes med rødt
- tegnes med svart

Ett komponent kan ha flere litt ulike symboler

Ressurser for å tegne kretsdiagrammer

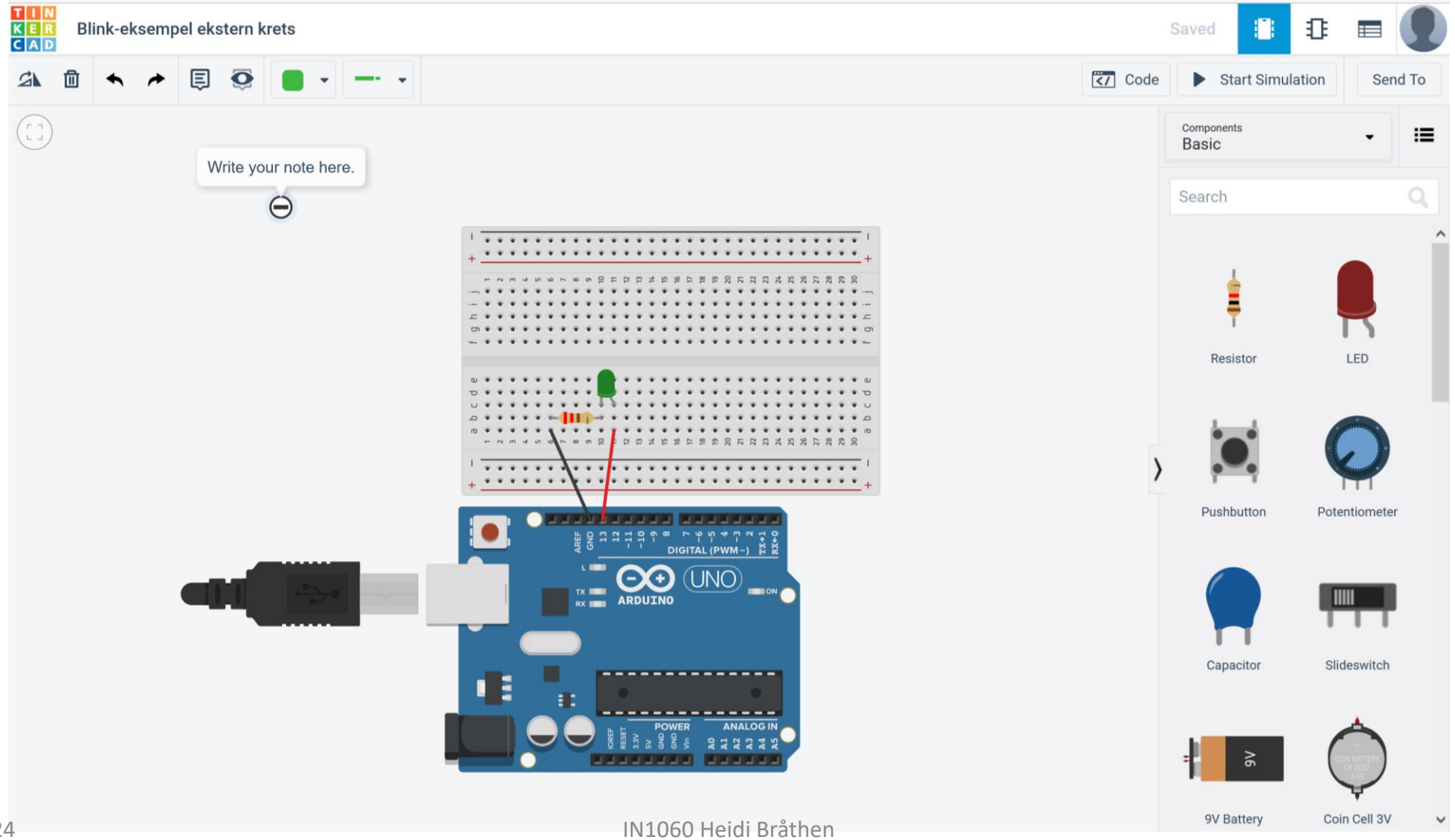
CAD står for Computer Aided Design
CAD er software som lar oss tegne ting som kan produseres med CAM – Computer Aided Manufacturing, for eksempel 3D-printere

TinkerCad https://www.tinkercad.com

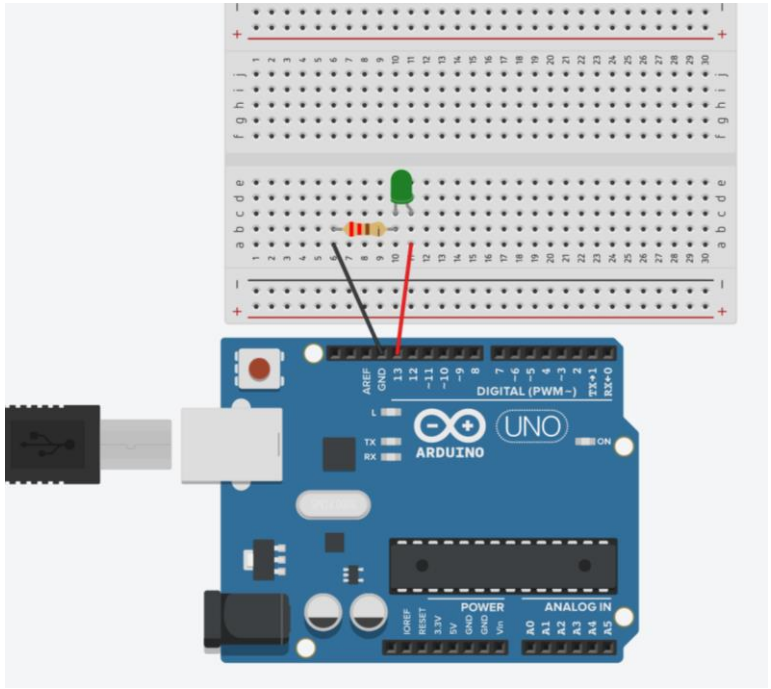
- Anbefalt, gratis og enkelt (må opprette en bruker)
- Tegne visuelle kretsdiagrammer med drag&drop
- Simulere kretsen
- Konvertere til grafisk kretsdiagram (det er lov å bruke det visuelle kretsdiagrammet i alle innleveringer)
- Kodeblokk-funksjon hvor man kan bygge kode med drag&drop
- Kan dele og duplisere filer, men flere brukere kan ikke samarbeide på samme fil

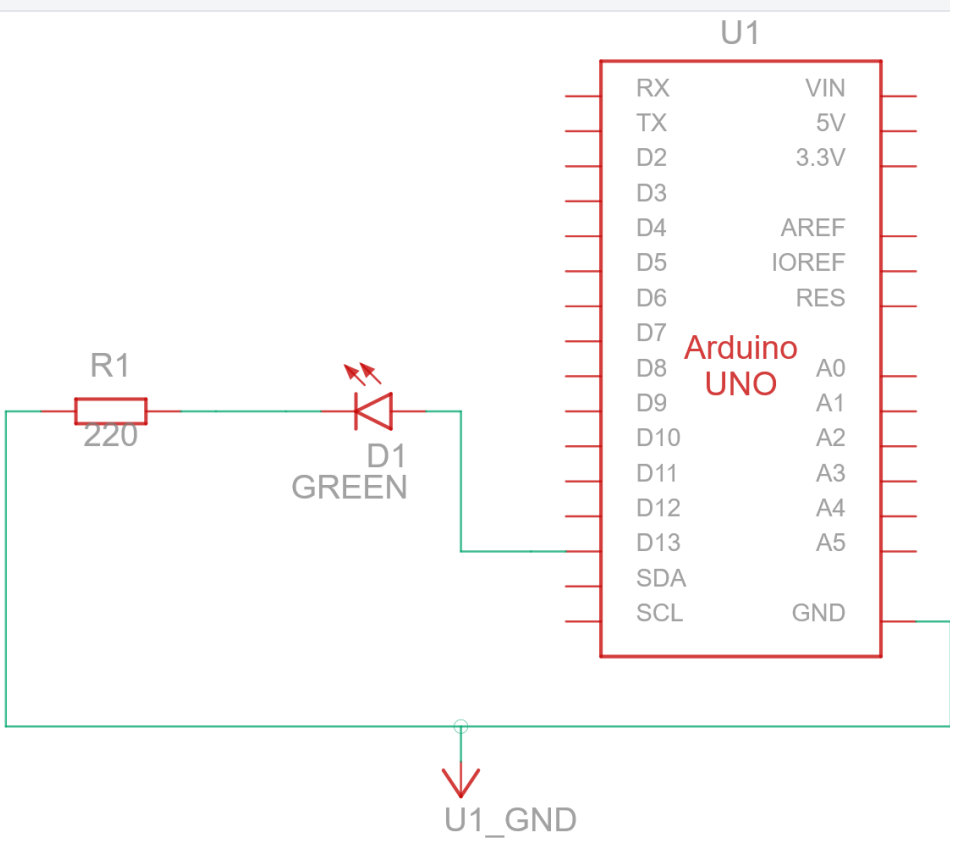
- **Andre ressurser man kan tegne kretsdiagrammer i (se semestersiden)**
- Fritzing (koster litt penger, men enkelt) https://fritzing.org/
- KiCad (gratis, men avansert arbeidsflyt, for de som er interessert i å designe egne kretskort) https://www.kicad.org/
- EagleCad (ligner på KiCad, samme utviklere som TinkerCad, har større begrensninger i gratisversjon)

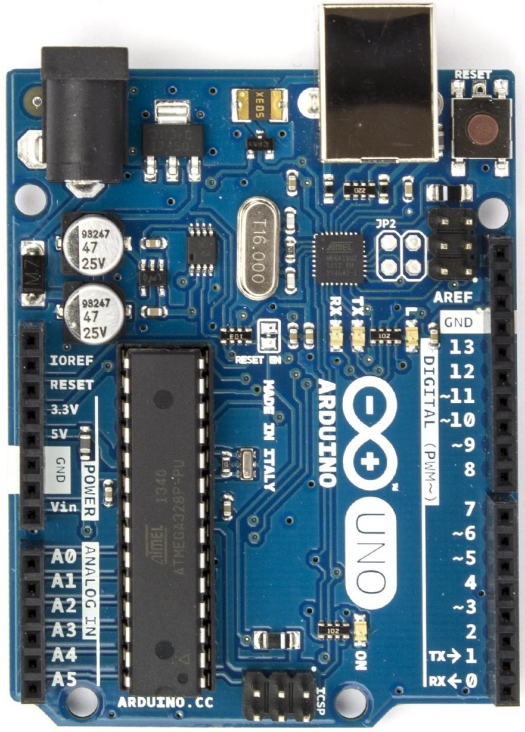
Eksempel på tegning av et visuelt kretsdiagram i TinkerCad



Visuelt og skjematisk kretsdiagram







Utvide kretsen med en bryter

Utvide kretsen vår med en bryter

Trenger nå

Kretsen fra i stad

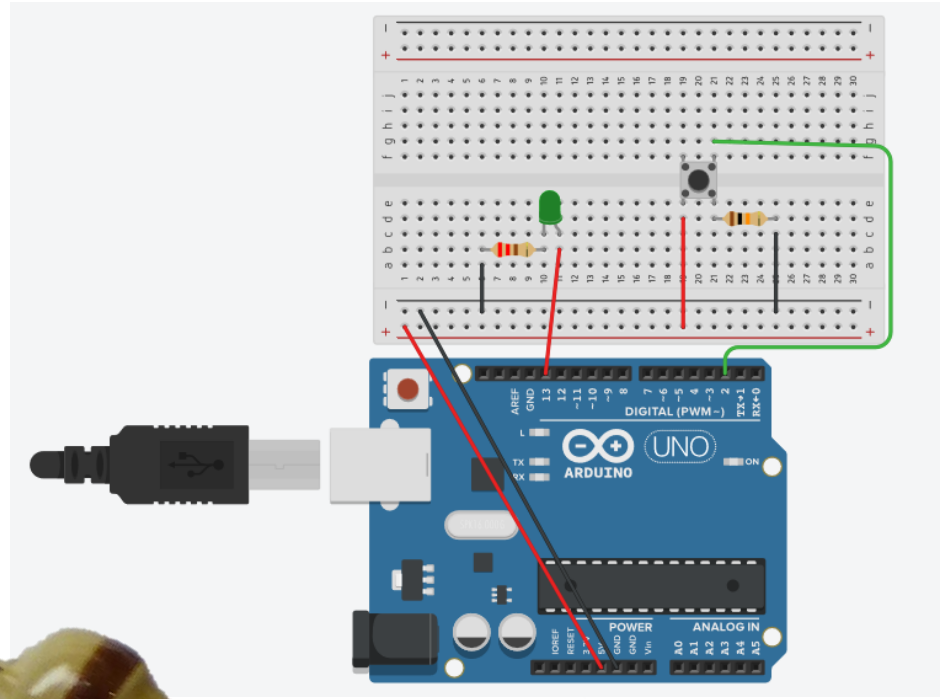
Pushbutton (knapp/bryter)

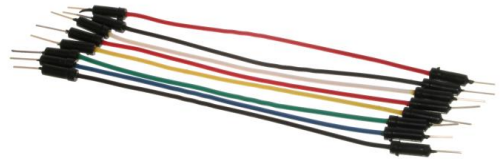
Ekstra ledning i valgfri farge

10k resistor

Eksempelskisse fra Arduino:

last opp fra Fil – Eksempler – 02 Digital – Button

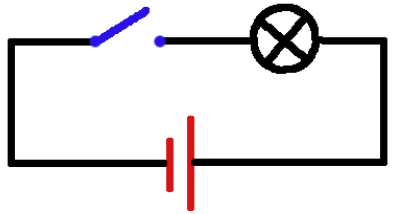




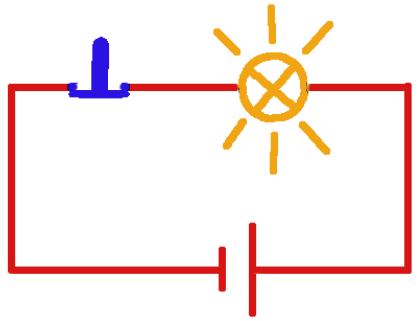
```
*/ Eksempelskisse: Button

const int buttonPin = 2;     // the number of the pushbutton pin // constants won't change. They're used here to set pin numbers:

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:

  }

}
```

Bryter

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)

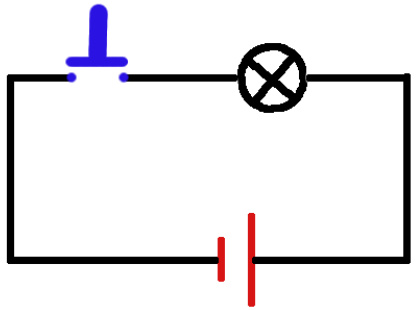


Knapper (pushbuttons) er eksempler på brytere

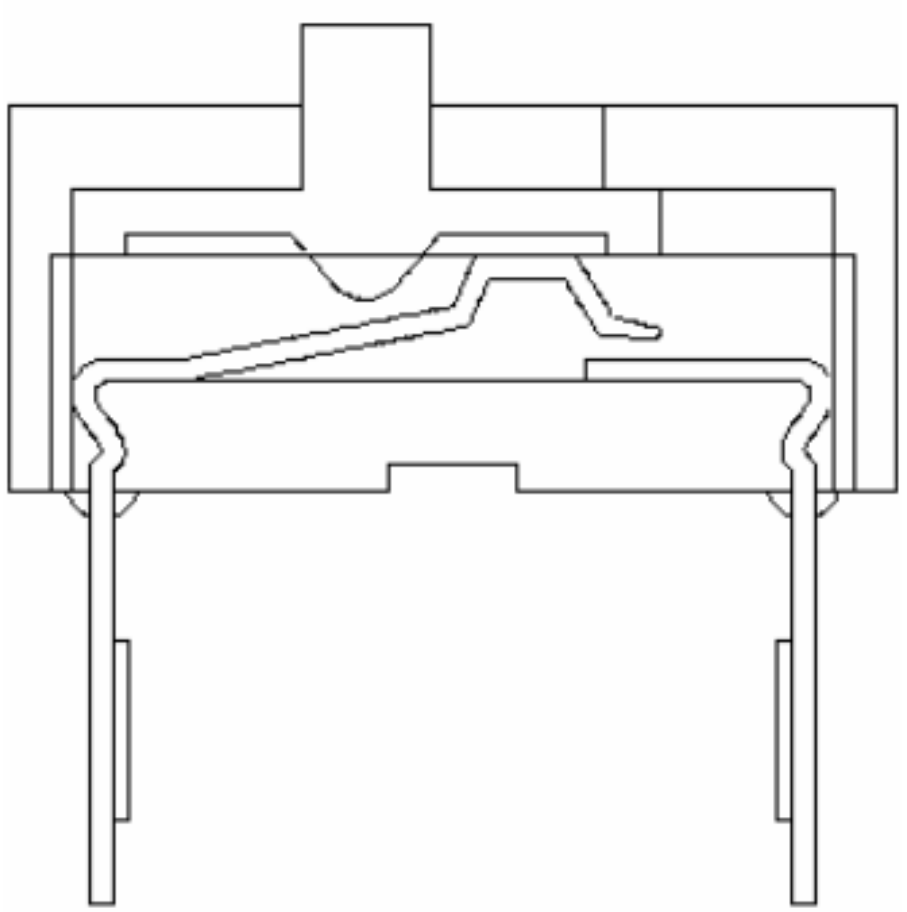


Enkle mekanismer: metall som kommer i kontakt for å slutte en krets

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

Problemløsning og feilsøking

+ To vanlige problemer med knapper

Serial: et verktøy for problemløsning

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

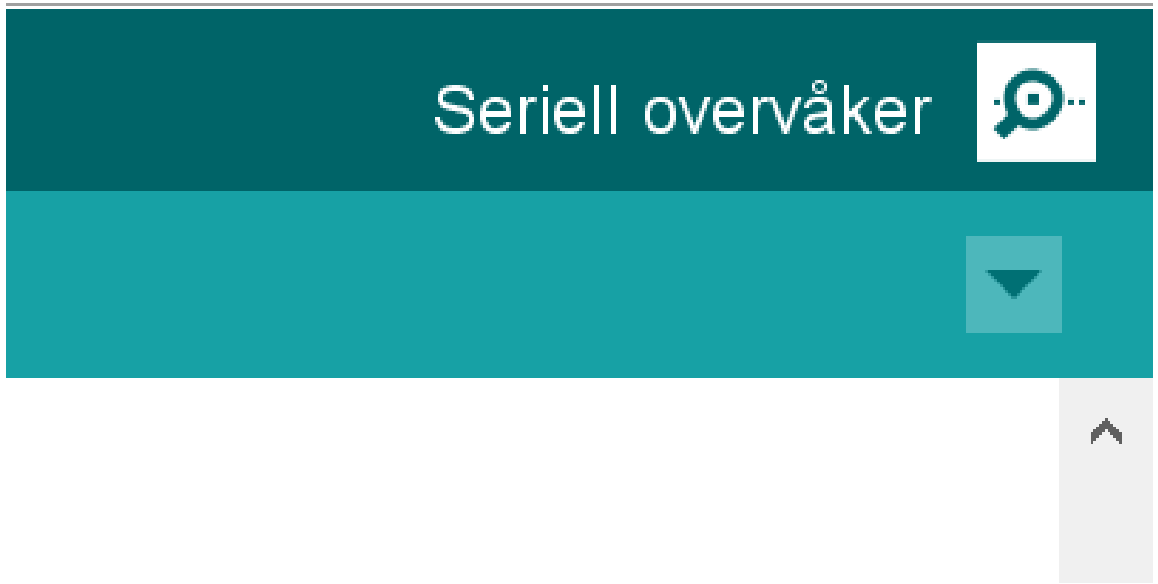
Gå til:

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

Vises som popup-vindu (integrert i nye IDE 2.0)

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
}
```

Vanlig problem med brytere #1: Sprettende input

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()

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 p lem #2: Flak og flytende

**«Flagrende» ** r et vanlig problem n Arduino-kretse na er koblet til set GI 0V/LOW, og de tro re oppfører seg uf ir det hele tatt

Løsning: Pullu

Arduino har en gget innebygget kons NPU brukes i stedet

Pullup med resistor og innebygget PULLUP

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

Bruke den innebygde pull-upen:

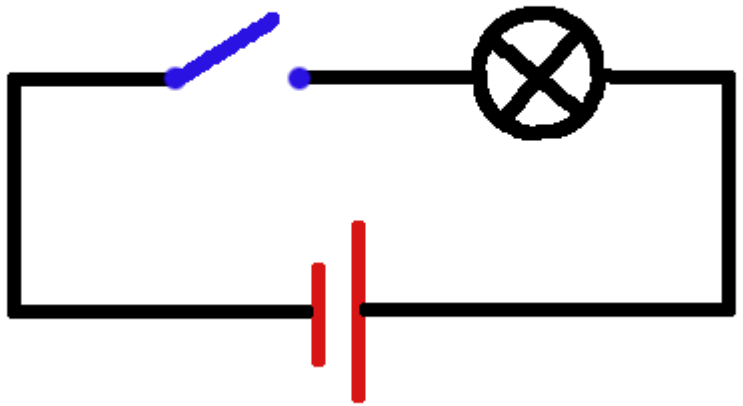
INPUT_PULLUP brukes i stedet for INPUT i pinMode()

Bruke komponentet resistor som pull up:

En resistor kan brukes som «pull up» som tilbakestiller signalet fra knappen til Arduinoen etter at den er trykket på ved å kobles mellom 5V og knappen

Digitale og analoge signaler

Digitale og analoge signaler



Digitale signaler: av og på

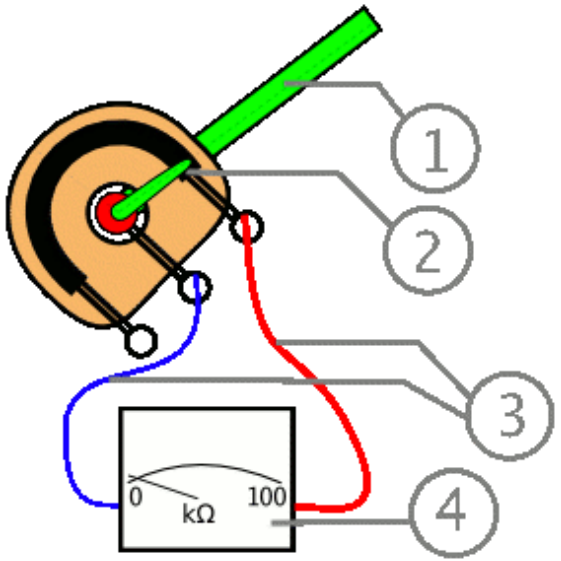
Av er det samme som 0 volt, LOW, false, 0

På er det samme som 5 volt, HIGH, true, 1

Analoge signaler

En skala av verdier/tall innenfor et spekter

Verdier som endres kontinuerlig



Digitale og analoge porter på Arduinoen

Arduinoen er egentlig digital, men har noen innebygde deler som kan lese og sende analoge signaler. Disse delene virker på noen spesielle pin'er på mikrokontrolleren.

Når vi skal lese og sende analoge signaler med Arduinoen må vi velge bestemte pin'er som er laget for analoge signaler

Analoge inn-porter: A0-A5

innebygget analog-til-digital omformer (analog-to-digital converter ADC) som regner om analoge innsignaler til digitale. ADC-en virker på alle portene som er merket A for analog

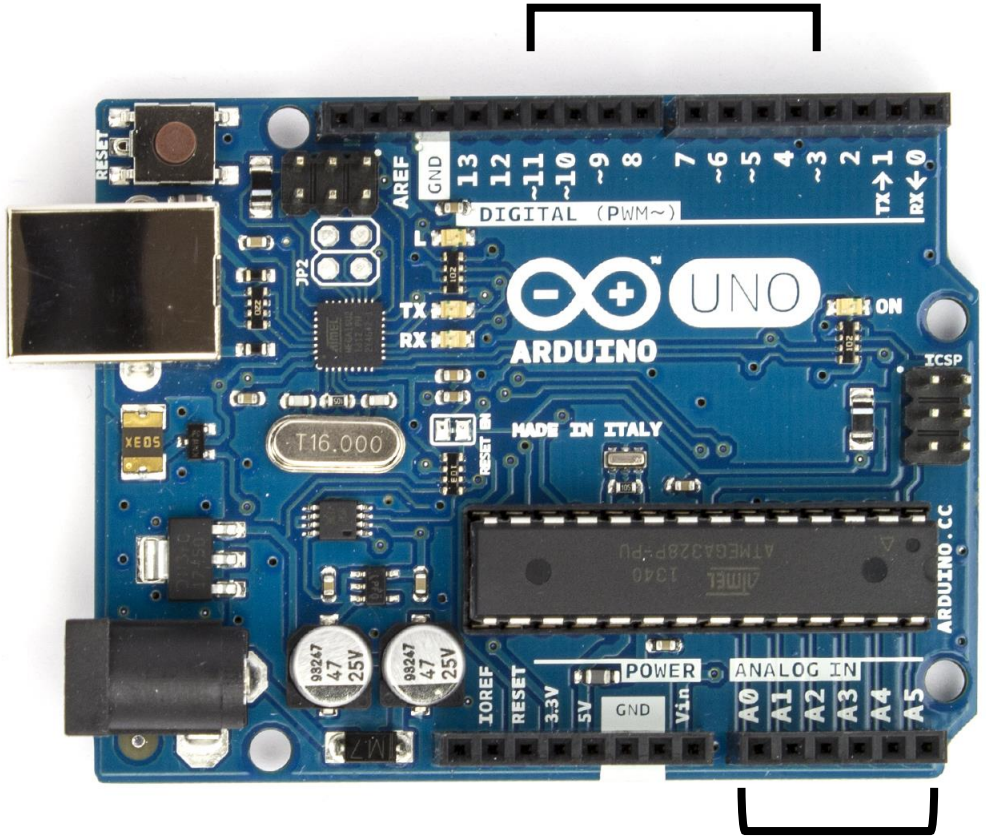
Analoge ut-porter: merket med tilde ~ 3, 5, 6, 9, 10 og 11

Arduinoen bruker en teknologi for å simulere analoge signaler ut som kalles Pulse Width Modulation. Den skrur spenningen veldig fort av og på slik at spenningen blir lavere.

Derfor kalles portene som er merket med tilde ~ også PWM-porter.

Oversikt over analoge pinner på Arduino Uno

Analog UT (PWM-porter) ~11 ~10 ~9 ~6 ~5 ~3



Analog INN: Analoge porter: A0-A5

PWM: Analog UT

Arduinoen bruker en teknologi for å simulere analoge signaler ut som kalles **P**ulse **W**idth **M**odulation. Den skrur spenningen veldig fort av og på slik at spenningen blir lavere.

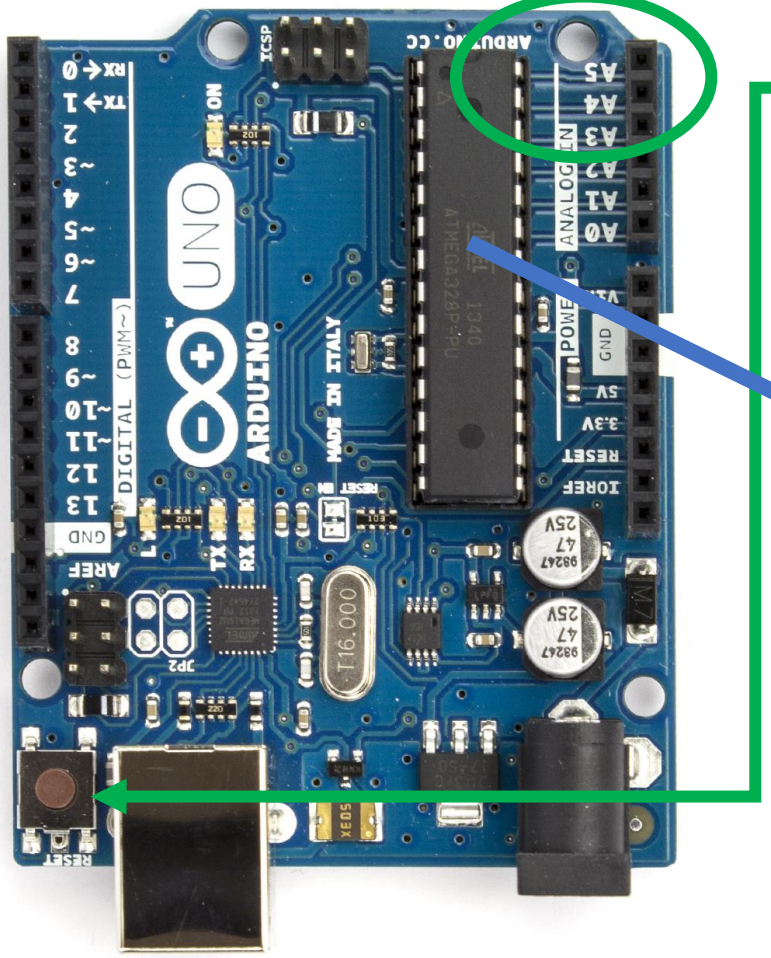
Derfor kalles pin'ene som er merket med tilde ~ også **PWM**-pin'er. De kan sende ut analoge signaler. Det kan ikke de andre digitale pin'ene, som for eksempel pin 2.

ADC: Analog INN

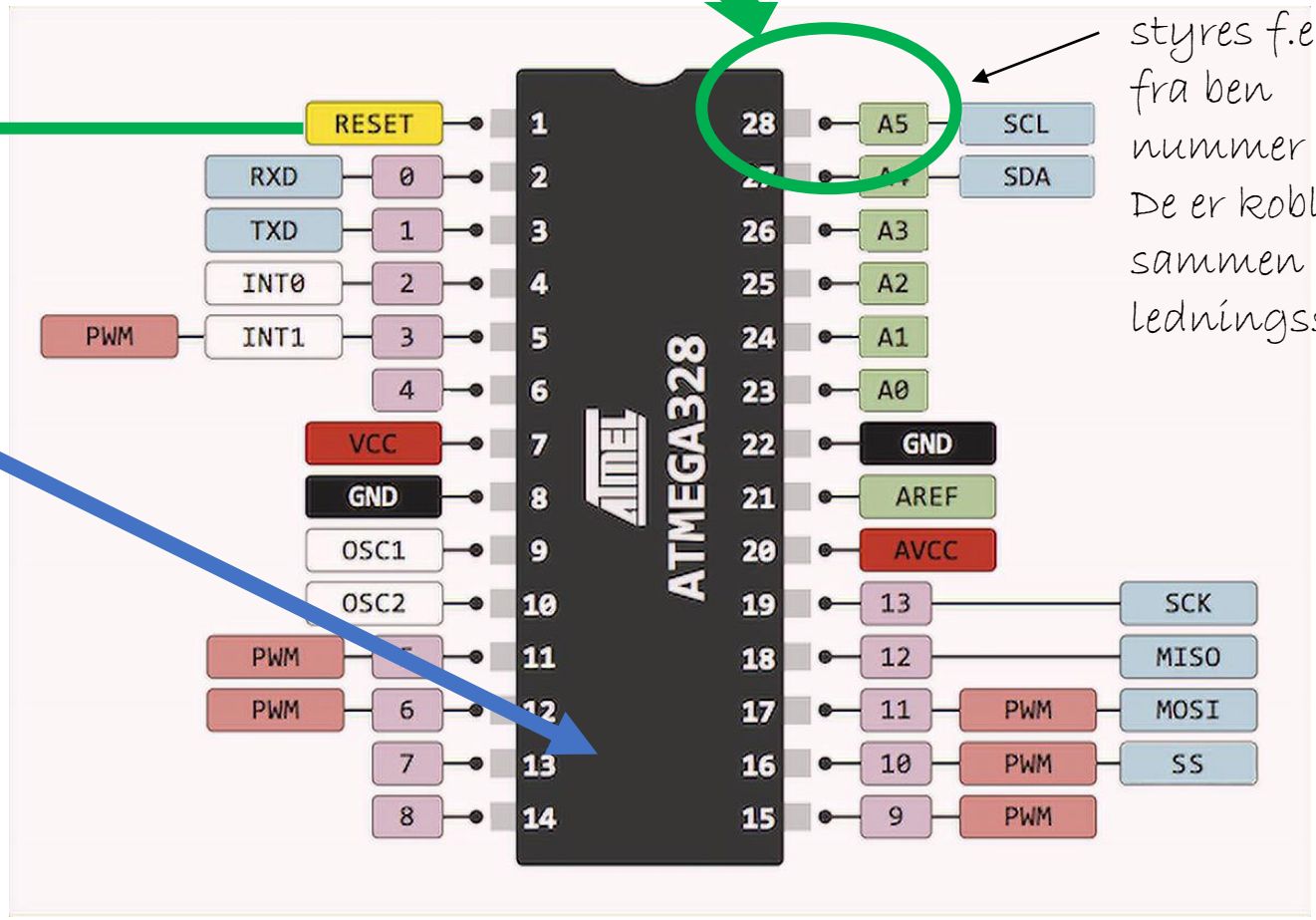
En innebygget analog-til-digital omformer (analog-to-digital converter ADC) regner om analoge innsignaler til digitale

ADC-en virker gjennom alle portene som er merket A for analog

Pin'ene på microcontrolleren har hver sine oppgaver



Reset-knappen styres f.eks. fra ben nummer 1



Pin A5 styres f.eks. fra ben nummer 28. De er koblet sammen med ledningsspor.

Dette er et diagram over mikrokontrolleren på Arduinobrettet og hva hvert av de 28 bena gjør

Digitale signaler

Digital inn (repetisjon)

Når vi leser av sensorer som er på eller av, f.eks knapp

digitalRead(valgt pin)

leser digitale inn-signaler og returnerer tilstand av eller på

Av er det samme som 0 volt, LOW, false, 0

På er det samme som 5 volt, HIGH, true, 1

Digital ut (repetisjon)

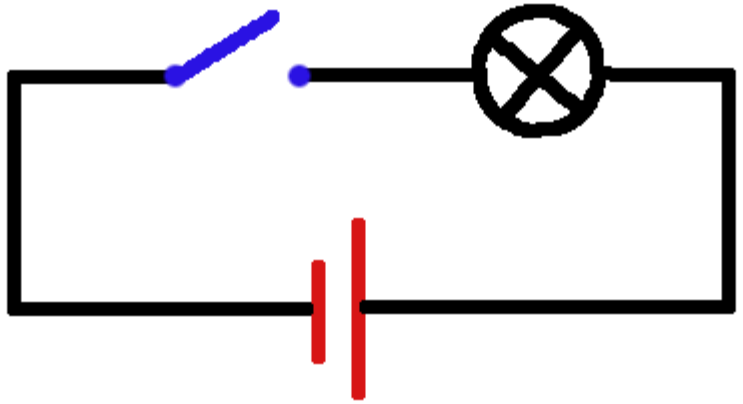
Når vi sende signal til aktuatorer som skal være av eller på, f.eks en LED

digitalWrite(valgt pin, tilstand: av eller på)

skriver eller sender ut **digitale** signaler om tilstand av eller på

Av er det samme som 0 volt, LOW, false, 0

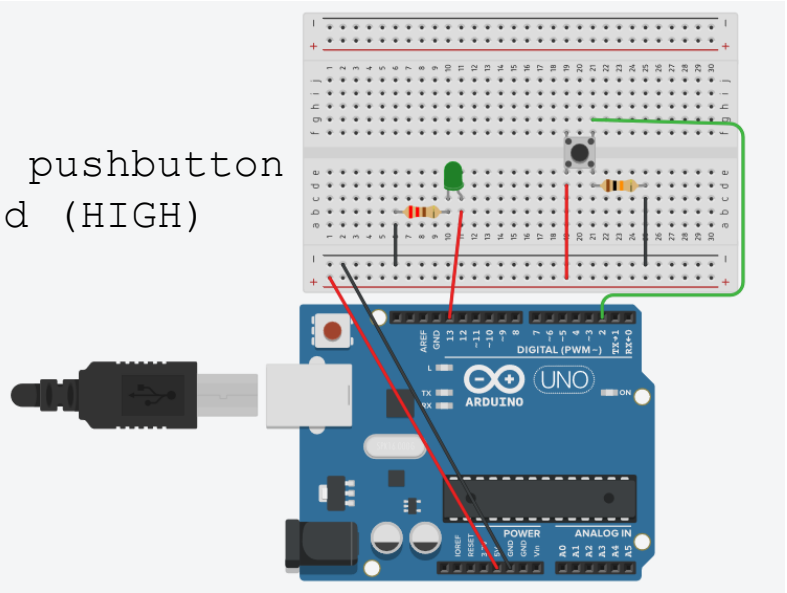
På er det samme som 5 volt, HIGH, true, 1



Digitale signaler eksempel: knapp og LED

***/ Eksempelskisse: Button under fil> eksempler> 02. Digital> Button**

```
const int buttonPin = 2;      // the number of the pushbutton pin // const
int ledPin =  13;             // the number of the LED pin
int buttonState = 0;          // variable for reading the pushbutton status

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
if (buttonState == HIGH) {// check if the pushbutton is pressed (HIGH)
        digitalWrite(ledPin, HIGH); // turn LED on:
                } else {
                digitalWrite(ledPin, LOW); // turn LED off:
  }
}
```



Analoge signaler

Analoge signaler inn

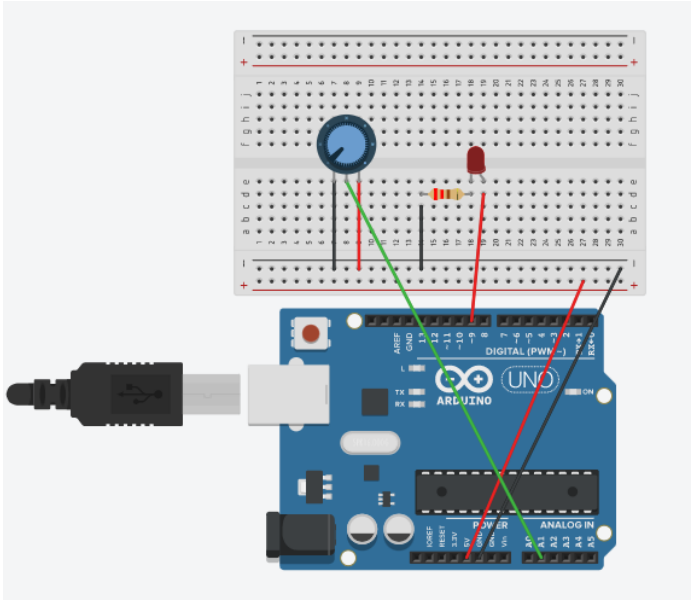
Når vi skal lese av sensorer som gir mange verdier, som en temperatursensor

analogRead(valgt pin)

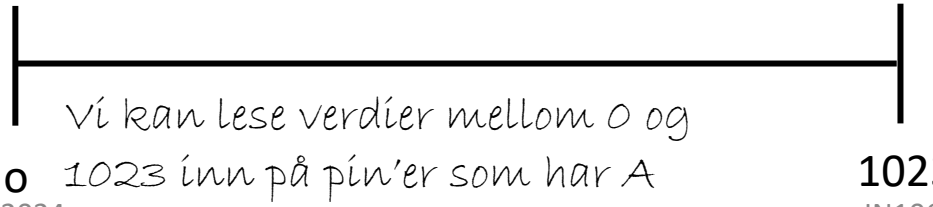
leser analoge inn-signaler og returnerer en int mellom 0 og 1023

For å lese analoge signaler må vi bruke en analog port, merket med A

Programmere analog inn



analogRead() tar imot variabler om hvilken pin det skal leses fra og returnerer hvilken verdi som leses av




```
int potentiometer = A0;


void setup(){
}
void loop(){
analogRead(potentiometer)
}
```

Analoge signaler ut

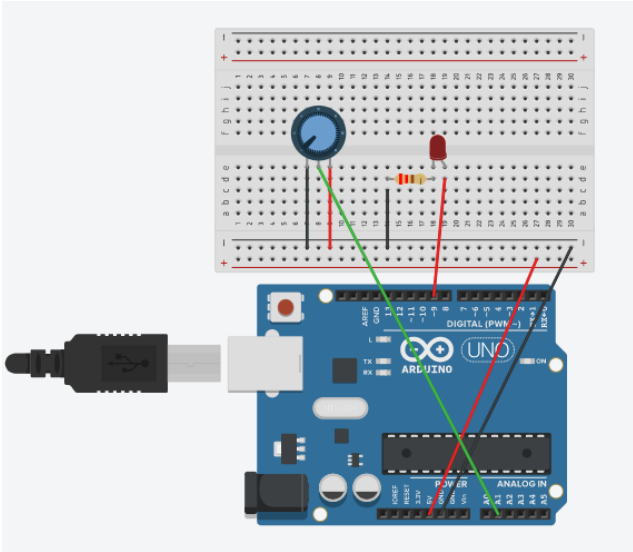
Brukes når vi skal styre aktuatorer som gjør mer enn å være av eller på, for eksempel styre volum eller dimme en led

analogWrite(valgt PWM-pin, styrke)

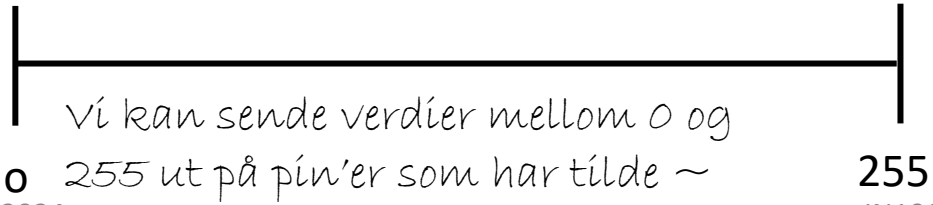
skriver eller sender ut analoge signaler om styrke i form av et tall mellom 0 og 255

For å sende analoge signaler må vi bruke en PWM-port som er merket med en tilde ~

Programmere analog ut



analogWrite() tar imot variabler om hvilken pin det skal sendes noe til og hvilken verdi som skal sendes

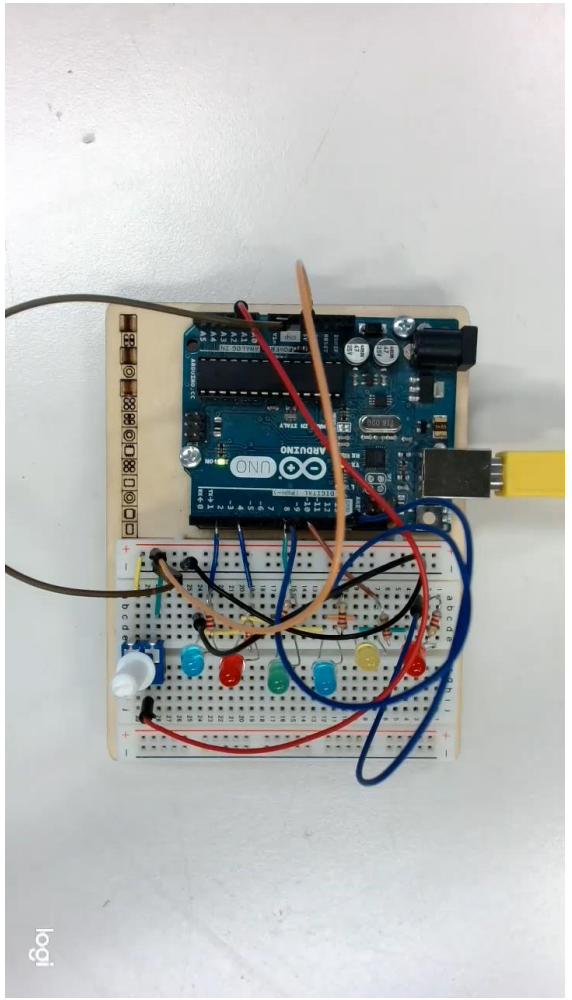



```
int ledPin = 9;


void setup(){
}
void loop(){
analogWrite(ledPin, verdi)
}
```

Slå på flere LED med et potensiometer #1

```
int led[] = {2,4,8,9,10,13};
int potentiometer = A1;

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

Serial.begin(9600);
}
```

```
void loop() {
 int potentiometerVerdi =
analogRead(potentiometer);
 Serial.println(potentiometerVerdi);
 for(int i = 0; i < 6; i++){
  if(i < potentiometerVerdi){
    digitalWrite(led[i], HIGH);
  }
  else{
     digitalWrite(led[i], LOW);
  }
 }
}
```



Seriell overvåker

Hvorfor er LEDene bare på?

Hva sender potensiometeret?

Fordi indeksen i løkken går fra 1 til 7 blir oppløsningen på potensiometeret (0 til 1023) for finkornet og det blir vanskelig å treffe.

I tillegg oppstår det en feil på potensiometeret som starter på vilkårlige verdier og ikke går til 0 når potensiometeret vris helt ned til venstre (av)

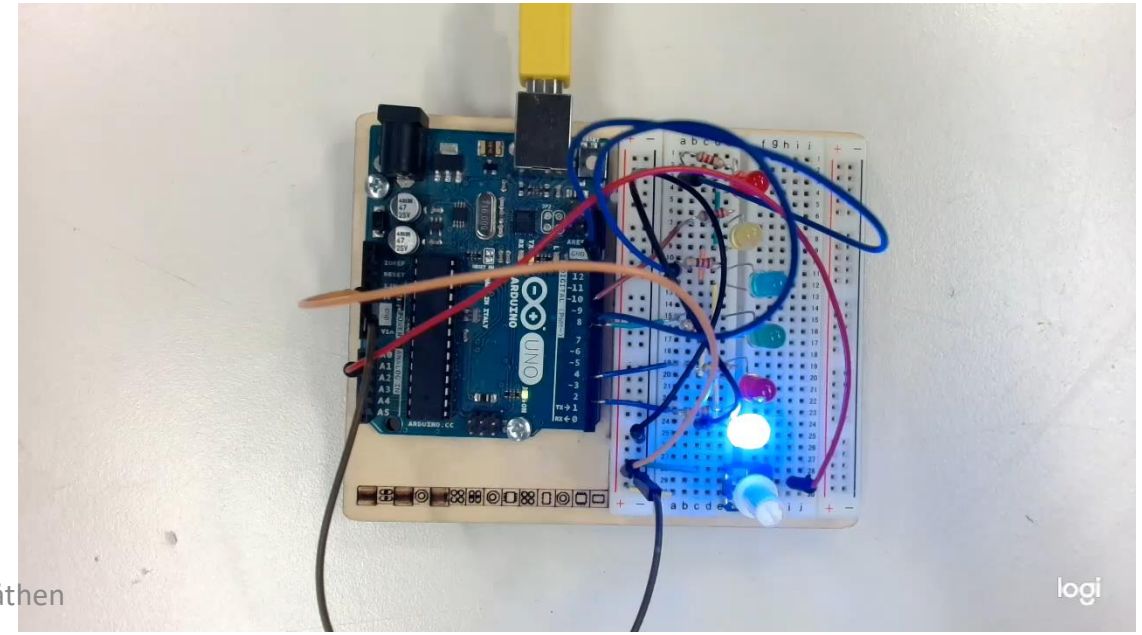
```
void loop() {
 int potentiometerVerdi = 
analogRead(potentiometer);
 Serial.println(potentiometerVerdi);
 for(int i = 0; i < 6; i++){
  if(i < potentiometerVerdi){
    digitalWrite(led[i], HIGH);
  }
  else{
     digitalWrite(led[i], LOW);
  }
 }
}
```

Slå på flere LED med et potentiometer

```
int led1 = 2;
int led2 = 4;
int led3 = 8;
int led4 = 9;
int led5 = 10;
int led6 = 13;
int potentiometer = A1;

void setup() {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
pinMode(potentiometer, INPUT);

Serial.begin(9600);
}
```

```
void loop() {
 int potentiometerVerdi = analogRead(potentiometer);
 Serial.println(potentiometerVerdi);
  if(potentiometerVerdi < 170){
    digitalWrite(led1, HIGH);
  }
   else{
     digitalWrite(led1, LOW);
  }

  if(potentiometerVerdi > 170 && potentiometerVerdi < 340){
    digitalWrite(led2, HIGH);
  }
   else{
     digitalWrite(led2, LOW);
  }

     if(potentiometerVerdi > 340 && potentiometerVerdi < 510){
    digitalWrite(led3, HIGH);
  }
   else{
     digitalWrite(led3, LOW);
  }
....etc
```

1023/6 leds =170



map()

En innebygget funksjon

Justere analoge signaler med **map()**

Analoge signaler på Arduinoen leses innenfor forhåndsbestemte verdier:

analogRead() returnerer er verdi mellom 0 og 1023

analogWrite() kan sende en verdi mellom 0 og 255

Noen ganger må vi justere skalaen slik at den passer med sensorene og aktuatorene vi bruker

map()- funksjonen kan brukes til å justere skalaen

Justere lesningen av potensiometeret med **map()**

Skalaen som analogRead() leser har høy oppløsning, fra 0 til 1023

Potensiometeret har lavere oppsløsning; det er vanskelig å treffe 1023 punkter når vi vrir på potensiometeret

For å justere lesingen av det analoge signalet regner vi om til en lavere oppløsning som passer til et potensiometer. For eksempel 6 trinn?

Map()

Omregning med map()

map(value, fromLow, fromHigh, toLow, toHigh)

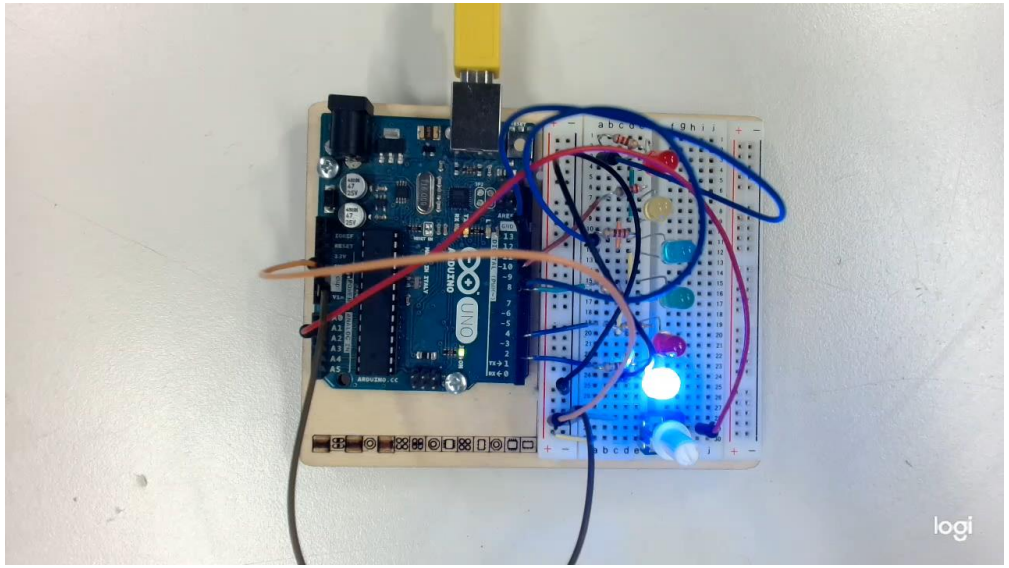
map() tar inn 5 variabler:
value er verdien vi leser av, den må vi først deklarere i setup()
fromLow og **fromHigh** er den opprinnelige skalaen, her 0 til 1023
toLow og **toHigh** er verdiene vi ønsker å sette, her 0 til 6

verdi = 0
map(verdi, 0, 1023, 0, 6)

Slå på flere LED med et potensiometer #2: Vi prøver igjen med lavere oppløsning

```
int led[] = {2,4,8,9,10,13};
int potentiometer = A1;
int potentiometerVerdi;

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

Serial.begin(9600);
}
```

```
void loop() {
 //int potentiometerVerdi = analogRead(potentiometer);
 potentiometerVerdi = map(analogRead(potentiometer), 0, 1023, 0, 6);
 Serial.println(potentiometerVerdi);
 for(int i = 0; i < 6; i++){
  if(i < potentiometerVerdi){
    digitalWrite(led[i], HIGH);
  }
  else{
     digitalWrite(led[i], LOW);
  }
 }
}
```



Analoge komponenter i settet

Analoge komponenter i settet

Sensorer

Potensiometer

Lyssensor

Temperatursensor

Lydsensor (piezo)

Aktuatorer

LED (kan dimmes, f.eks)

RGB LED

Piezo (høyttaler/lydsensor)

DC motor *(Starter kit, ikke Student kit)*

Servo motor

NB! Sett deg inn i kravene til motorene, de trekker mye strøm og kan skade Arduinoen hvis de blir satt opp feil. Vi går gjennom motorer i neste forelesning, men det er gode eksempler i Arduinoboka.