UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Prototyping med Arduino del 3

Magnus Li
magl@ifi.uio.no

INF1060
05.02.2017

Arduinoundervisningen

Forelesninger
Siste i dag.
Gjennomgang av grunnleggende temaer

Teknisk verksted (rom C)
i dag, 12.02 & 19.02
Hjelp til ukesoppgaver

Gruppetimer
Denne uken
Komme igang med prosjekt
+ Hjelp til ukesoppgaver

Obligatoriske oppgaver

1) **Frist 09.02**
   Utvalgte ukesoppgaver skal leveres

2) Frist 23.02
   Miniprosjekt skal leveres

Oblig 1 - frist fredag 23:59!

Obligatoriske oppgaver

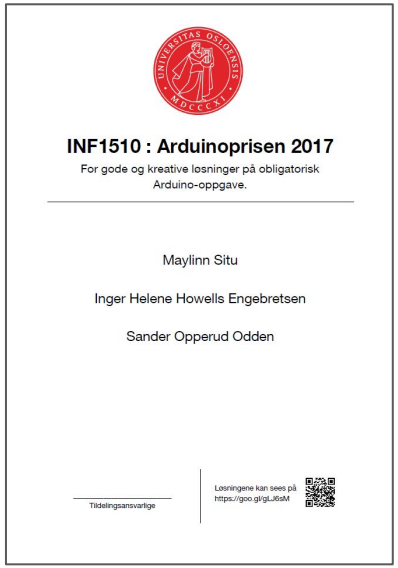
1) **Frist 09.02**
   Utvalgte ukesoppgaver skal leveres

2) Frist 23.02
   Miniprosjekt skal leveres

Oblig 2: konkurranse

Gavekort på Kjell & Company

+



mye heder og ære

```
while (true) {
    vinner.ære++;
}
```

```
for (int i = 0; i < Integer.MAX_VALUE; i++) {
    vinner.put(heder, ære);
}
```

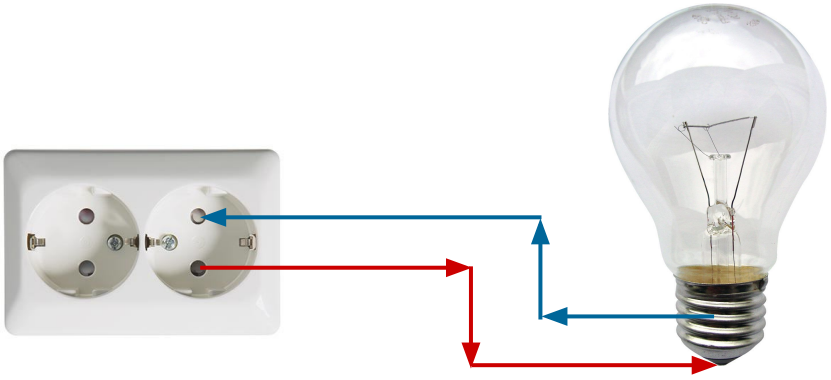
Nå

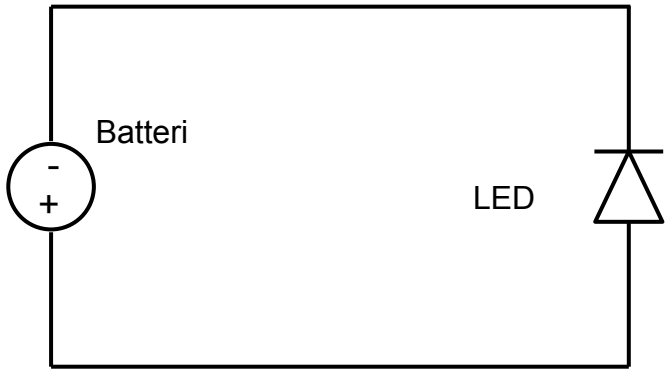
- Litt repetisjon
- Tidsutsettelse uten delay()
- Trøbbel ved knappeklikk - *debounce*
- Modularisering av kode
- Veien videre

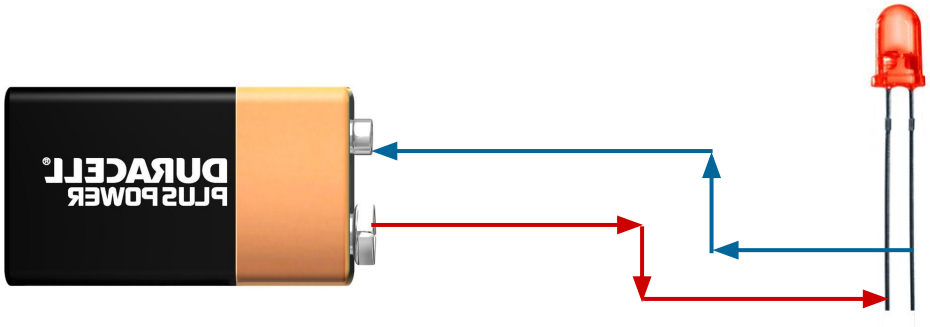
Forrige uke

Elektrisitet i kretser

Elektrisitet i kontrollerte former beveger seg alltid i det vi kaller en sluttet krets.

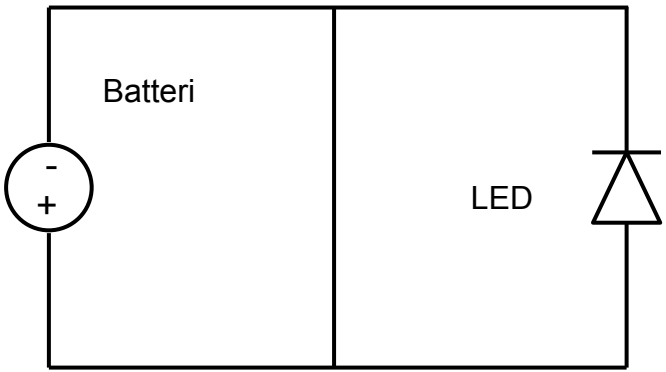


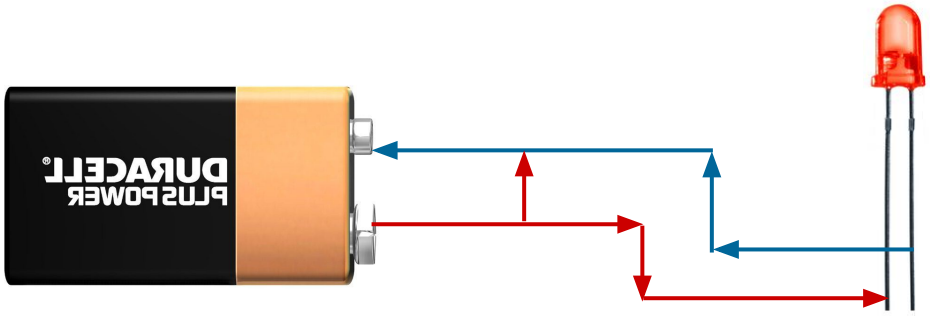




Elektrisitet i kretser

Elektrisitet i kontrollerte former beveger seg alltid i en sluttet krets, og **vil ta veien med minst motstand.**



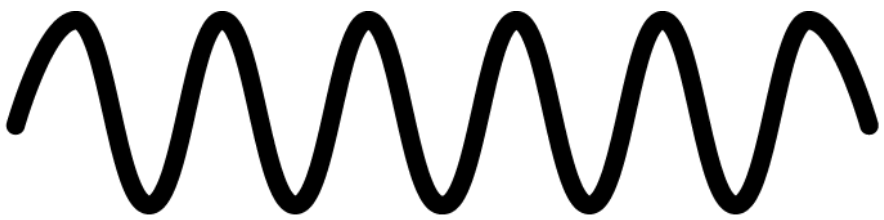


Digitale og analoge signaler

Med Arduino må vi forholde oss til to typer signaler.

Analoge, som kan ha mange verdier innenfor et bestemt spekter.

Digitale, som kun består i AV og PÅ.



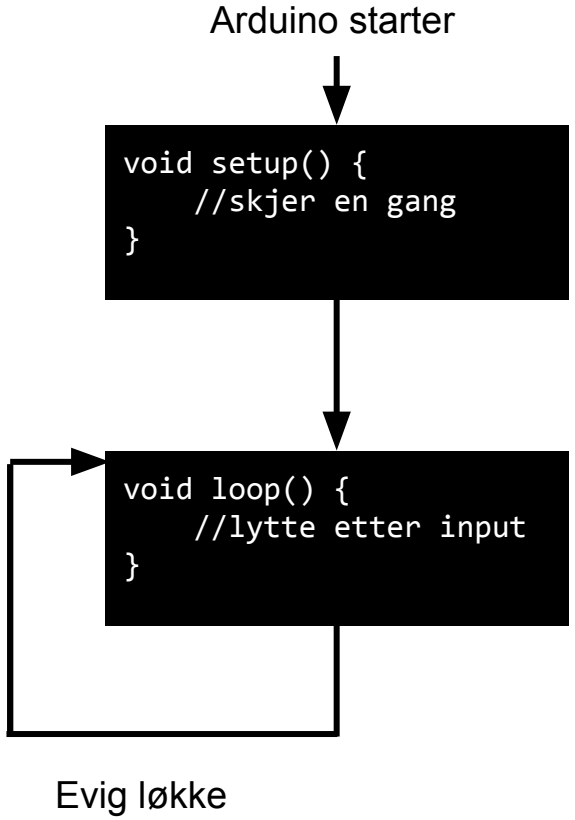
Analog Signal

Digital Signal

SETUP OG LOOP

Da loop kjører i en evig løkke vil all kode her skje igjen og igjen, noe som former måten vi programmerer på.

Løkken gjør at vi kan lytte etter signaler fra portene hele tiden.

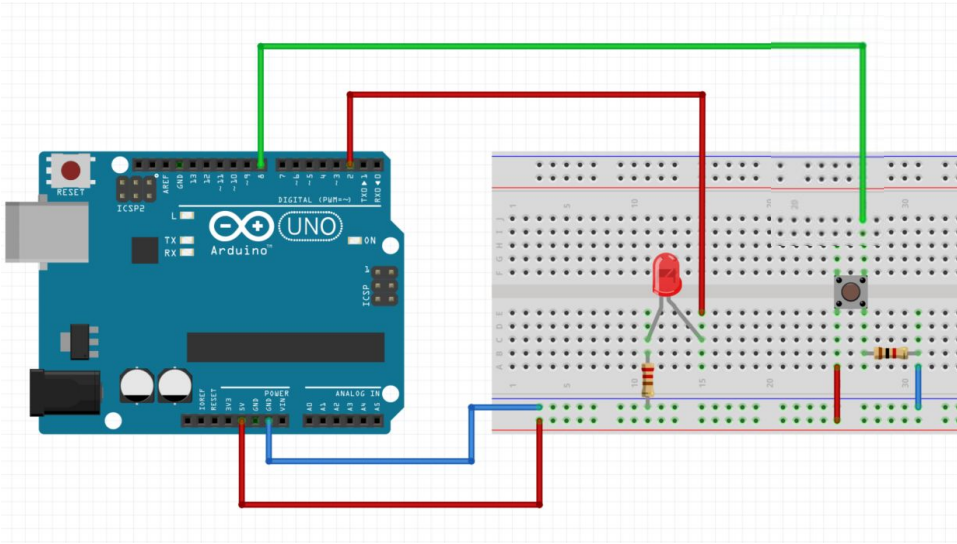


Tidsutsettelse uten delay()

Mål:

Vi ønsker å lage et system hvor et lys blinker av og på hvert sekund.

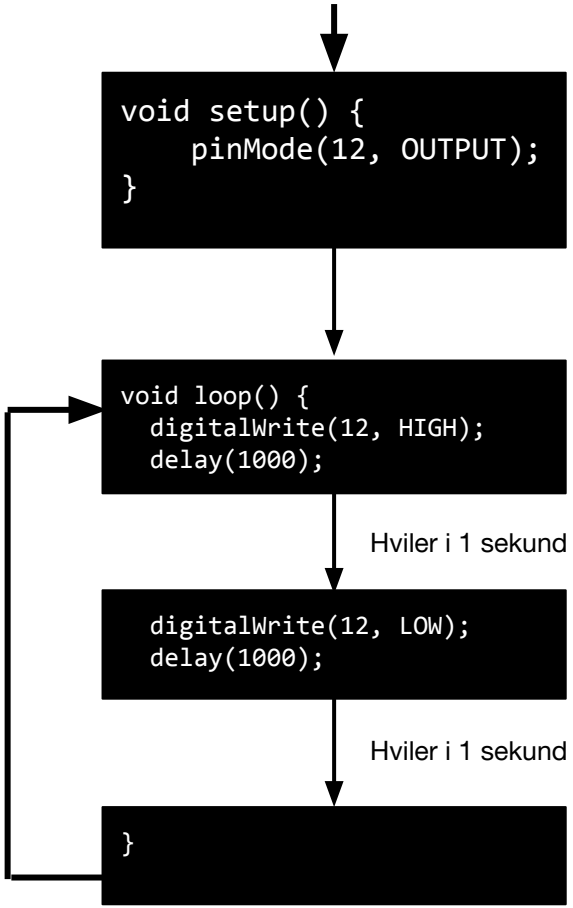
Samtidig vil vi at man skal kunne klikke på en knapp når som helst. Når knappen klikkes skal lyset slutte å blinke



DELAY()

```
int led = 6;
void setup() {
    pinMode(led, OUTPUT);
}

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

Evig løkke



DELAY()

Evig løkke

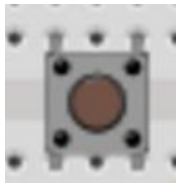
```
void loop() {
  digitalWrite(12, HIGH);
  delay(1000);
```

Hviler i 1 sekund

```
  digitalWrite(12, LOW);
  delay(1000);
```

Hviler i 1 sekund

```
}
```

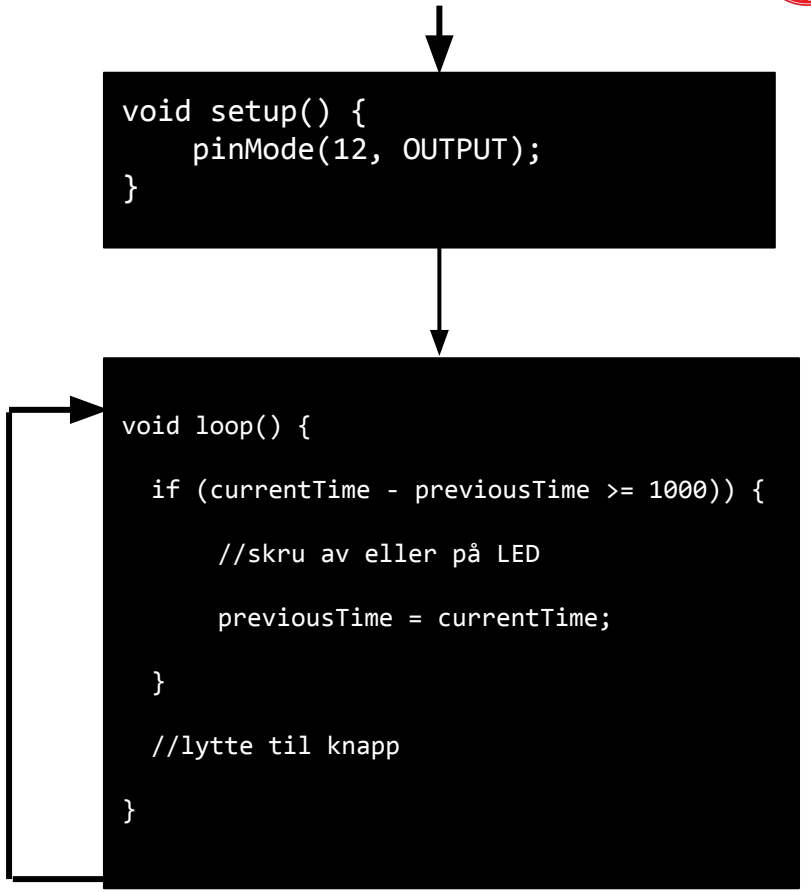


Det ikke være mulig å registrere nye signaler som kommer inn i tidsrommet til delay().

Løsning: telle tid

Vi teller hvor lang tid det har gått siden forrige gang vi skrudde av/på lyset.

Hvis det har gått 1 sekund (1000 millisekunder), så skrur vi det av/på.



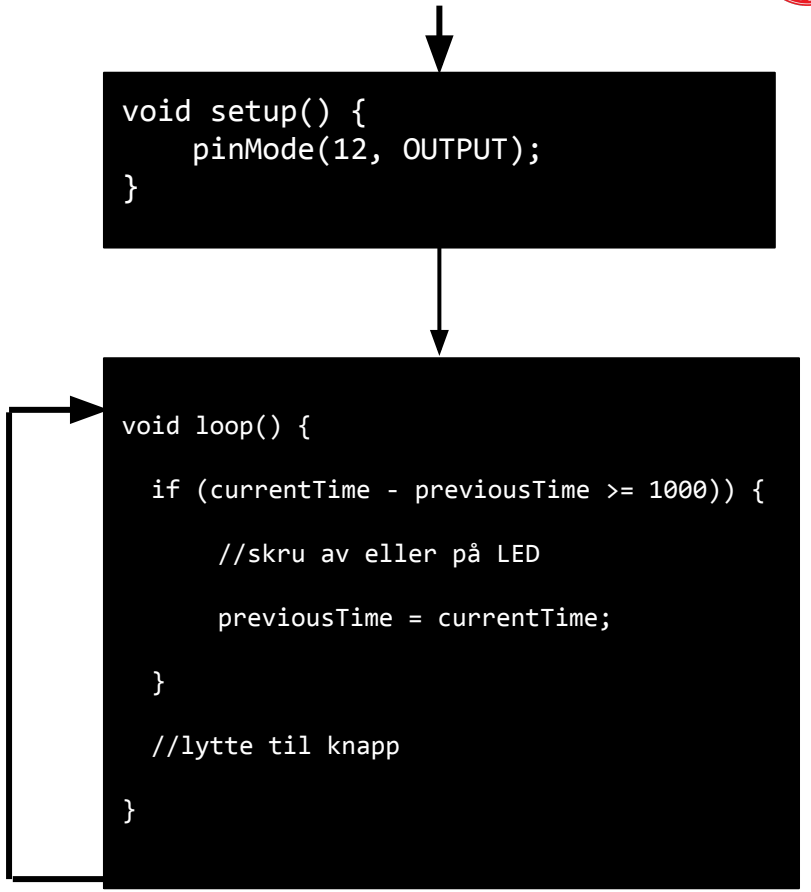
Løsning: telle tid

previousTime = 0
currentTime = 300

(currentTime - previousTime >= 1000) == ?

previousTime = 0
currentTime = 1000

(currentTime - previousTime >= 1000) == ?



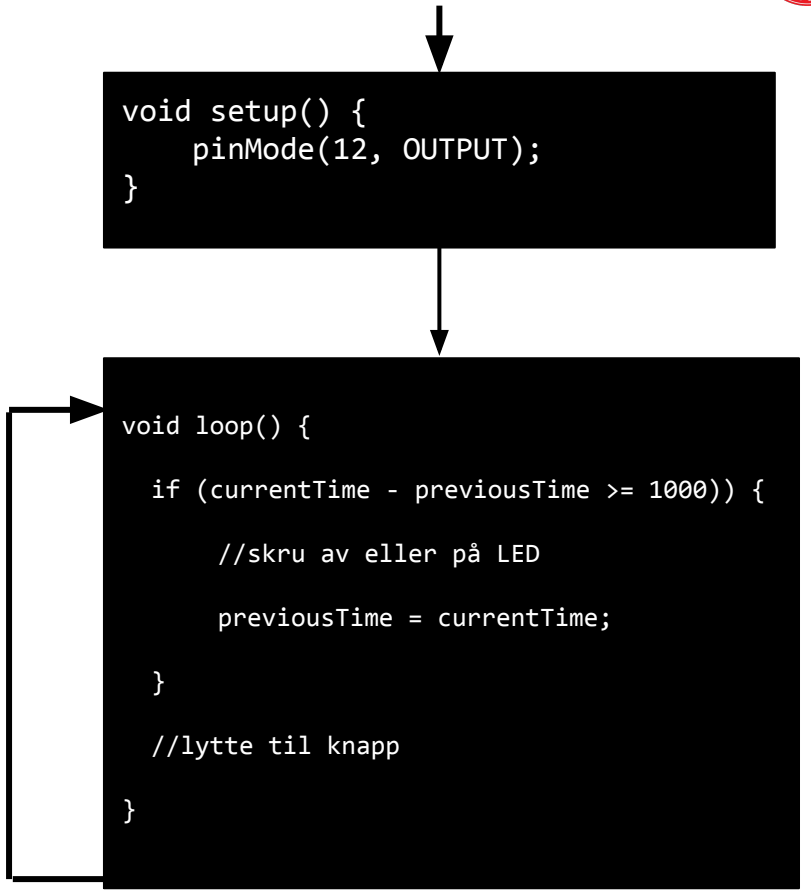
Løsning: telle tid

previousTime = 1000
currentTime = 1300

(currentTime - previousTime >= 1000) == ?

previousTime = 1000
currentTime = 2002

(currentTime - previousTime >= 1000) == ?

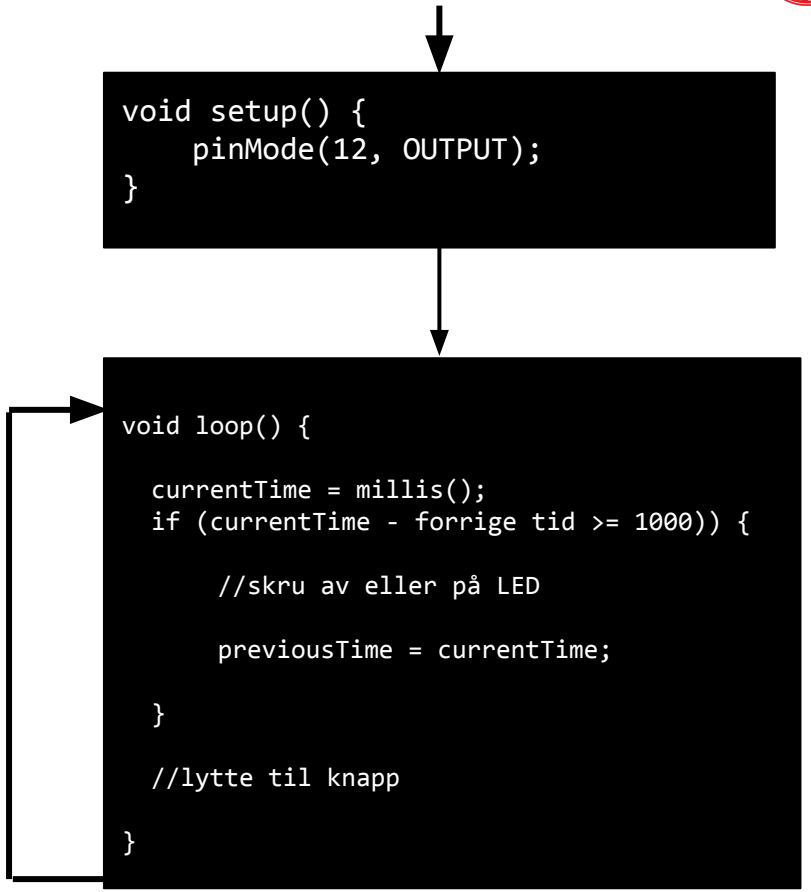


Millis()

Metoden millis() returnerer tiden siden Arduino startet opp i millisekunder.

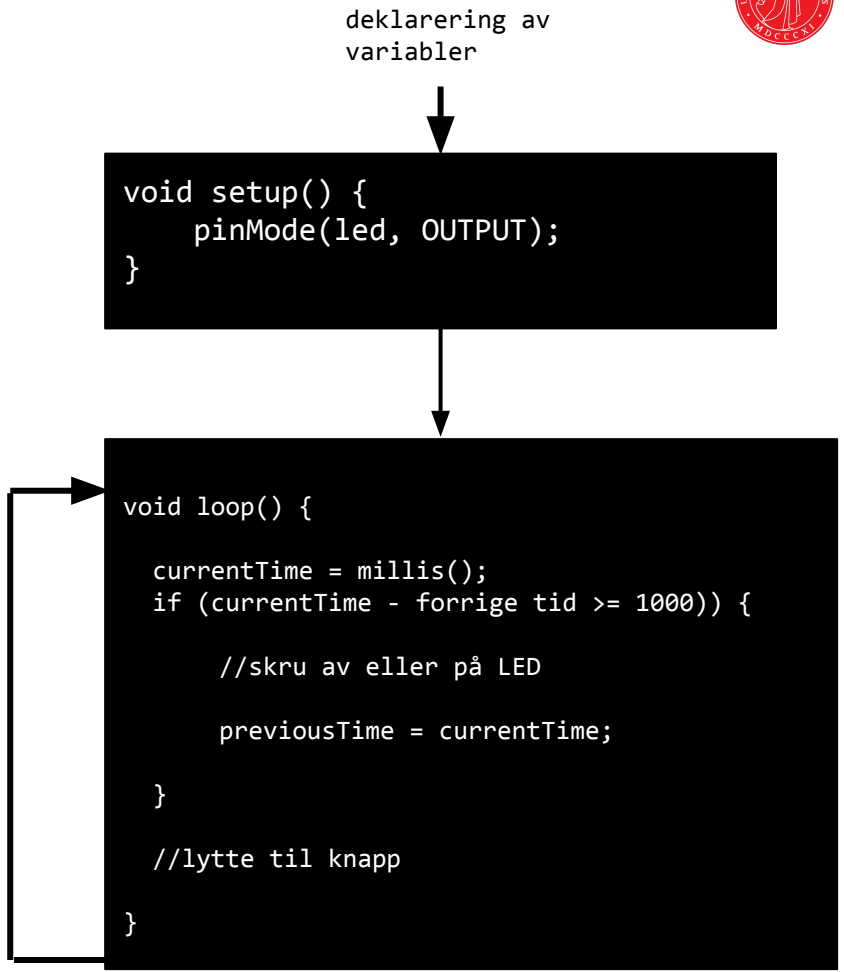
5 sekunder etter at Arduino starter vil millis() returnere 5000.

Dette kan vi bruke for å telle tid.



Tidsutsettelse med millis()

```
int led = 2;
unsigned long previousTime = 0;
unsigned long currentTime;
int interval = 1000;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {

   currentTime = millis();

   if (currentTime - previousTime >= interval) {

       //skru lys av/på

       previousTime = currentTime;
   }
}
```



Få lyset til å blinke

```
void loop() {

    currentTime = millis();

    if (currentTime - previousTime >= interval) {

        //skru lys av/på

        previousTime = currentTime;

    }
}
```

Blink

```
int led = 2;
unsigned long previousTime = 0;
unsigned long currentTime;
int interval = 1000;
boolean ledState = false;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {

   currentTime = millis();

   if (currentTime - previousTime >= interval) {

        if (ledState == false) {
             digitalWrite(led, HIGH);
             ledState = true;
        } else {
             digitalWrite(led, LOW);
             ledState = false;
        }

       previousTime = currentTime;

  }

}
```

Blink

```
int led = 2;
unsigned long previousTime = 0;
unsigned long currentTime;
int interval = 1000;
boolean ledState = true;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {

   currentTime = millis();

   if (currentTime - previousTime >= interval) {

      ledState = !ledState;
      digitalWrite(led, ledState);

       previousTime = currentTime;

   }
}
```

! (utropstegn)
reverserer verdien

!true = false

(1 > 2) = false
!(1 > 2) = true

Blink

```
int led = 2;
unsigned long previousTime = 0;
unsigned long currentTime;
int interval = 1000;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {

   currentTime = millis();

   if (currentTime - previousTime >= interval) {

      digitalWrite(led, !digitalRead(led));

       previousTime = currentTime;
   }
}
```

! (utropstegn) reverserer status av porten :-)

Modularisere koden

```
int led = 2;
unsigned long previousTime = 0;
unsigned long currentTime;
int interval = 1000;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  currentTime = millis();

  if (currentTime - previousTime >= interval) {

    digitalWrite(led, !digitalRead(led));

      previousTime = currentTime;

  }

}
```

Når programmet begynner å vokse, kan det fort bli rotete i loop().

Derfor gjelder det å legge alle deler av koden som har med hverandre å gjøre, **ut i egne funksjoner.**

Det gjør det lettere å lese koden for andre. Dermed blir det bedre å samarbeide i prosjektet.

NB! Dette er veldig viktig i oblig 2, og i koden til prosjektet.

Modularisere koden

```
void loop() {
    unsigned long currentTime = millis();
    if (currentTime - previousTime >= interval) {
        digitalWrite(led, !digitalRead(led));
        previousTime = currentTime;
    }
}
```

```
void loop() {
    blink(led, 1000);
}

void blink(int led, int interval) {
    unsigned long currentTime = millis();
    if (currentTime - previousTime >= interval) {
        digitalWrite(led, !digitalRead(led));
        previousTime = currentTime;
    }
}
```

Modularisere koden

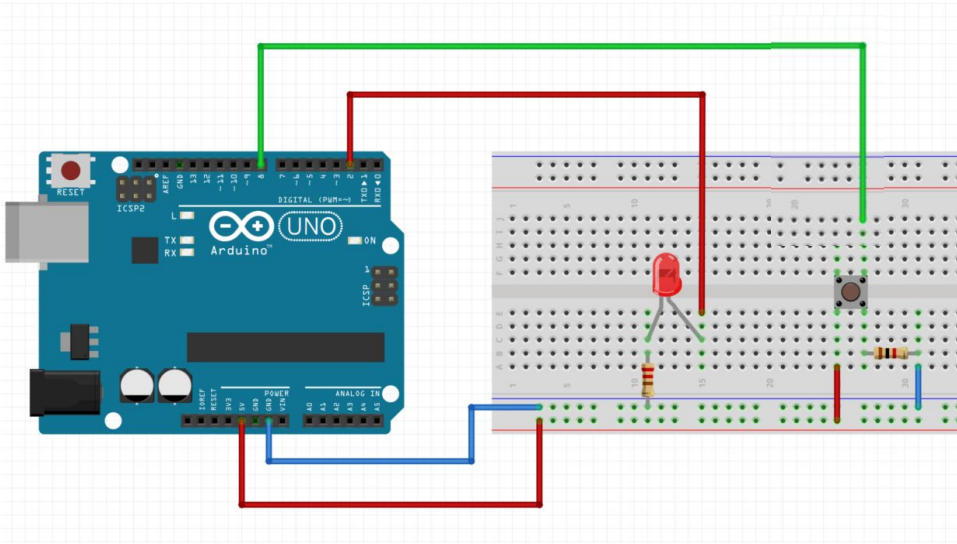
```
void loop() {

   blink(led, 1000);

}


void blink(int led, int interval) {

   unsigned long currentTime = millis();

   if (currentTime - previousTime >= interval) {

      digitalWrite(led, !digitalRead(led));
      previousTime = currentTime;
   }
}
```

```
void loop() {

   blink(led, 1000);

}

void blink(int led, int interval) {

   unsigned long currentTime = millis();

   if (currentTime - previousTime >= interval) {
      toggle(led);
      previousTime = currentTime;
   }
}

void toggle(int pin) {
   digitalWrite(pin, !digitalRead(pin));
}
```

Mål:

- Vi ønsker å lage et system hvor et lys blinker av og på hvert sekund.

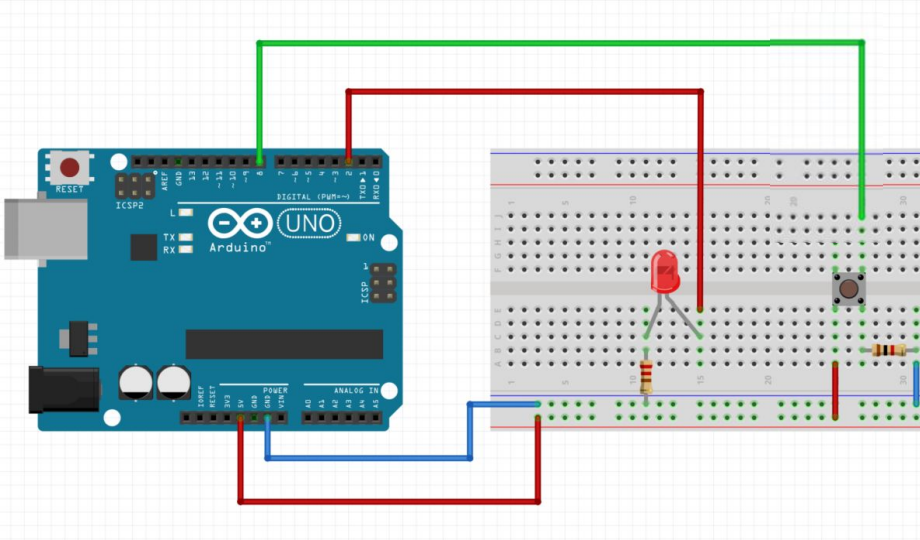
Samtidig vil vi at man skal kunne klikke på en knapp når som helst. Når knappen klikkes skal lyset slutte å blinke



Legge til funksjonalitet for knapp

```
void loop() {

   blink(led, 1000);

}

void blink(int led, int interval) {

   unsigned long currentTime = millis();

   if (currentTime - previousTime >= interval) {
      toggle(led);
      previousTime = currentTime;
   }
}

void toggle(int pin) {
   digitalWrite(pin, !digitalRead(pin));
}
```

På grunn av modulariseringen vår, slipper vi nå å forholde oss til akkurat hvordan blinkingen vår er implementert, og kan lett fokusere på å implementere knappen vår.

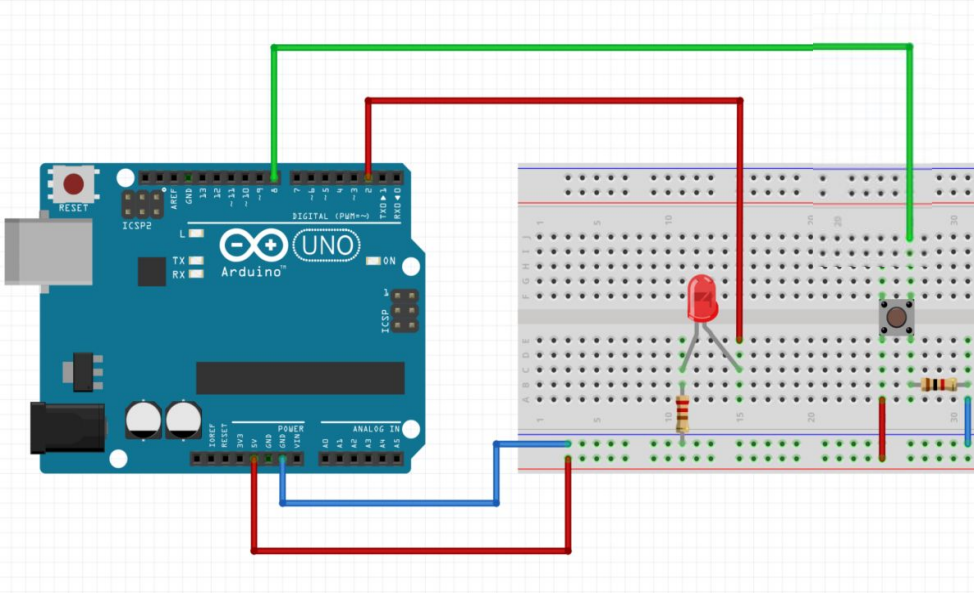


Legge til funksjonalitet for knapp

```
int button = 8;
boolean shouldBlink = true;

void loop() {

   if (shouldBlink == true) {
      blink(led, 1000);
   }
}
```

(skitten og tungvind implementasjon)

Først må vi legge til en variabel som kan ta vare på om lyset skal blinke

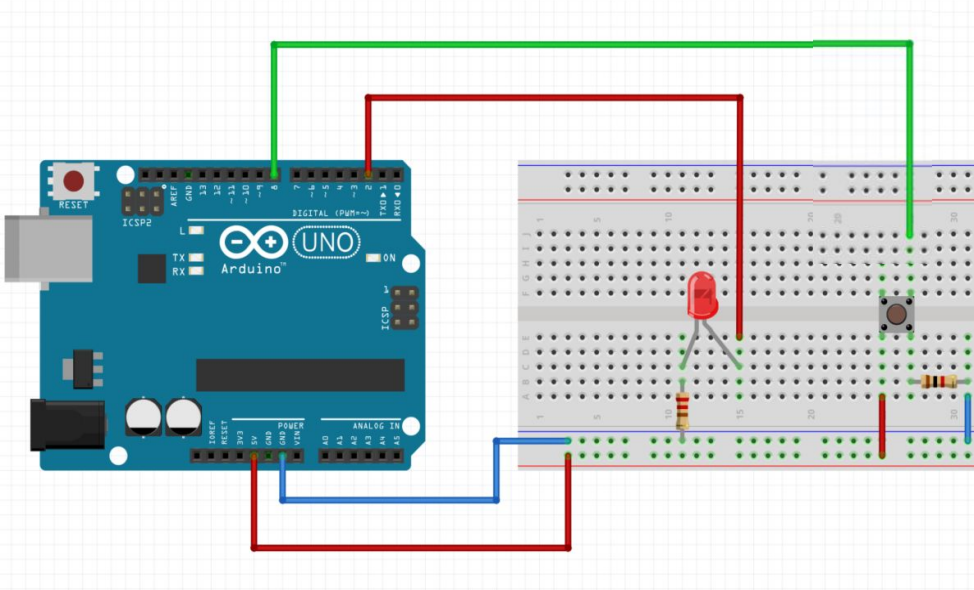


Legge til funksjonalitet for knapp

```
int button = 8;
boolean shouldBlink = true;

void loop() {

   if (shouldBlink == true) {
      blink(led, 1000);
   }
   if (digitalRead(button) == HIGH) {
      shouldBlink = false;
   }

}
```

(skitten og tungvind implementasjon)

Så legger vi til en if, hvor vi sjekker om knapp har blitt klikket. Hvis klikket setter vi shouldBlink til false.



Evaluering av boolean / HIGH og LOW

```
int button = 8;
boolean shouldBlink = true;

void loop() {

   if (shouldBlink == true) {
      blink(led, 1000);
   }
   if (digitalRead(button) == HIGH) {
      shouldBlink = false;
   }

}
```

(skitten og tungvind implementasjon)

```
int button = 8;
boolean shouldBlink = true;

void loop() {

   if (shouldBlink) {
      blink(led, 1000);
   }
   if (digitalRead(button)) {
      shouldBlink = false;
   }

}
```

(penere implementasjon)

Mer modularisering?

```
int button = 8;
boolean shouldBlink = true;

void loop() {

   if (shouldBlink) {
      blink(led, 1000);
   }
   if (digitalRead(button)) {
      shouldBlink = false;
   }

}
```

```
int button = 8;
boolean shouldBlink = true;

void loop() {

   if (shouldBlink) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      shouldBlink = false;
   }

}

boolean isClicked(int pin) {
   return digitalRead(pin);
}
```

Trøbbel med knappeklikk

debounce

Trøbbel med knappeklikk

Når vi klikker en knapp kan vi ha ulike scenario avhengig av hva vi ønsker å gjøre.

For eksempel:

1. Slå av en blinkende LED.
2. Få lampen til å blinke når vi holder inne knappen

3. Slå av eller på blinkingen hver gang knapp klikkes
4. Slå på blinkingen etter 5 klikk

Trøbbel ved knappeklikk

1. Slå av blinkende LED

```
boolean shouldBlink = true;

void loop() {

   if (shouldBlink) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      shouldBlink = false;
   }

}
```

2. Få LED til å blinke når vi holder inne knapp

```
void loop() {

   if (isClicked(button)) {
      blink(led, 1000);
   }

}
```

Trøbbel med knappeklikk

Når vi klikker en knapp kan vi ha ulike scenario avhengig av hva vi ønsker å gjøre.

For eksempel:

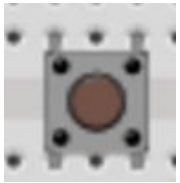
1. Slå av en blinkende LED.
2. Få lampen til å blinke når vi holder inne knappen

3. Slå av eller på blinkingen hver gang knapp klikkes
4. Slå på blinkingen etter 5 klikk

Trøbbel ved knappeklikk

3. Slå av eller på blinkingen hver gang knapp klikkes

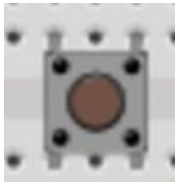
```
boolean shouldBlink = true;

void loop() {

   if (shouldBlink) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      shouldBlink = !shouldBlink;
   }

}

boolean isClicked(int pin) {
   return digitalRead(pin);
}
```



Trøbbel ved knappeklikk

3. Slå av eller på blinkingen hver gang knapp klikkes

```
boolean shouldBlink = true;

void loop() {

   if (shouldBlink) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      shouldBlink = !shouldBlink;
   }

}

boolean isClicked(int pin) {
   return digitalRead(pin);
}
```

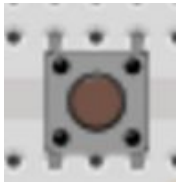


shoudBlink blir her endret mellom true og false flere ganger fordi loop rekker å gå flere runder mens vi klikker knappen

Trøbbel ved knappeklikk

4. Slå på blinkingen etter 5 klikk

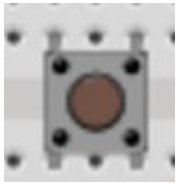
```
int clickCount = 0;

void loop() {

   if (clickCount >= 5) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      clickCount++;
   }

}

boolean isClicked(int pin) {
   return digitalRead(pin);
}
```



Trøbbel ved knappeklikk

4. Slå på blinkingen etter 5 klikk

```
int clickCount = 0;

void loop() {

   if (clickCount >= 5) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      clickCount++;
   }

}

boolean isClicked(int pin) {
   return digitalRead(pin);
}
```



Det blir her registrert flere knappeklikk fordi loop rekker å gå mange runder før vi slipper knappen

Debounce 1: med delay()

4. Slå på blinkingen etter 5 klikk

```
int clickCount = 0;

void loop() {

   if (clickCount >= 5) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      clickCount++;
   }

}

boolean isClicked(int pin) {
   return digitalRead(pin);
}
```

```
boolean isClicked(int pin) {
   if (digitalRead(pin));
      delay(50);
      return true;
   }
   return false;
}
```

Her bruker vi delay() for å pause loop litt, slik at vi ikke registrerer flere knappeklikk på ett klikk.

Problem: delay()

Debounce 2: med millis()-utsettelse

```
int clickCount = 0;

void loop() {

   if (clickCount >= 5) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      clickCount++;
   }

}
```

```
unsigned long lastButtonPush = 0;
int debounceDelay = 50;

boolean isClicked(int pin) {

   if ((millis() - lastButtonPush) >= debounceDelay) {

      if (digitalRead(button)) {
          lastButtonPush = millis();
          return true;
      }

   }

   return false;
}
```

Debounce 3: med millis()-utsettelse og lastButtonState

```
int clickCount = 0;

void loop() {

   if (clickCount >= 5) {
      blink(led, 1000);
   }
   if (isClicked(button)) {
      clickCount++;
   }

}
```

```
unsigned long lastButtonPush = 0;
int debounceDelay = 50;

boolean isClicked(int pin) {

   if ((millis() - lastButtonPush) >= debounceDelay) {

      if (digitalRead(button)) {
          lastButtonPush = millis();
          return true;
      }

   }

   return false;
}
```

Fortsatt problemer!

Hvis vi holder knappen lenger enn 50 millisekunder vil flere klikk fortsatt registreres.

For løsning se: https://www.arduino.cc/en/Tutorial/Debounce

(Mer) modularisering

Modularisering

Å modularisere koden vår i funksjoner hjelper oss med:

- Bryte opp et problem til mindre delproblemer
- Lage lettlest kode
- Koble sammen arbeidet til flere personer

Dette ved å abstrahere bort detaljene i hvordan noe er implementert

Still deg disse spørsmålene før du begynner å løse problemet, og hele tiden underveis:

- Hvilke funksjonalitet må jeg ha for å løse problemet?
  → Danner grunnlag for hovedfunksjoner

- Bruker jeg denne (del)-koden flere ganger?
  → Kanskje det bør være en egen funksjon.

- Kan denne delen av koden uttrykkes som funksjonalitet i seg selv?
  → Legg det i en egen funksjon

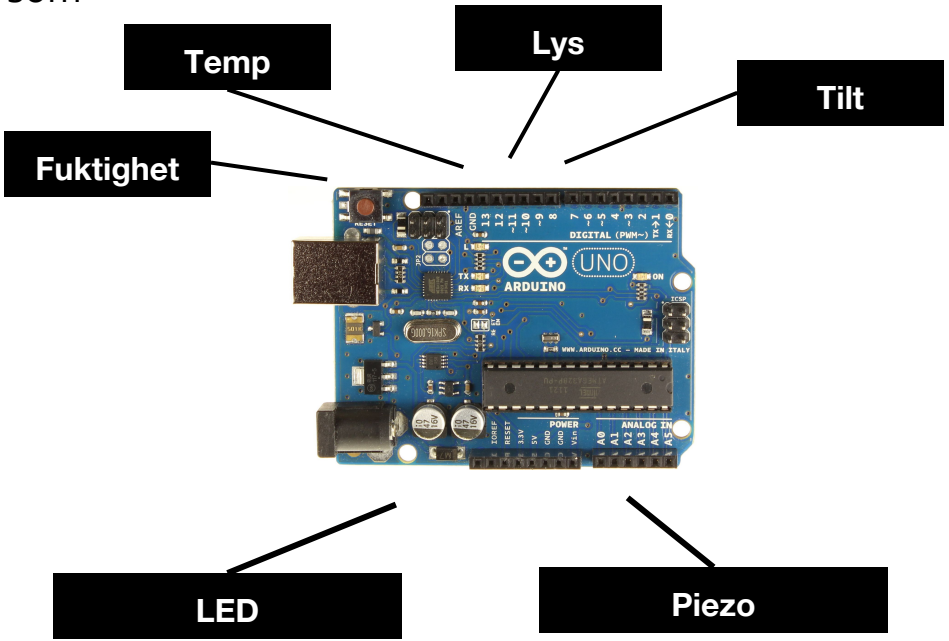
Modularisering - eksempel

Vi ønsker en boks som kan måle forskjellige faktorer i rommet den er i, og si ifra til bruker dersom noen verdier er ugunstige i forhold til hva man ser på som et godt arbeidsmiljø.

Input fra miljø
Lys, temperatur, luftfuktighet.

Input fra bruker
Tilt-sensor

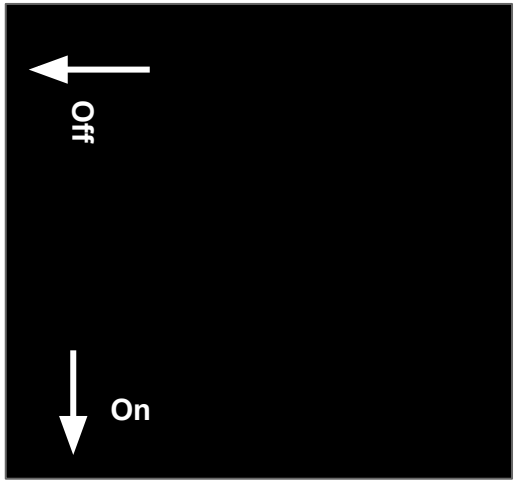
Output (for feedback til bruker)
LED-lys.
Lyd med piezo-element.



Modularisering

Tanken er at boksen skal kunne stå på et bord, og ha alle sensorer inkludert. Hvilken vei boksen står vil bestemme om den er på eller av.

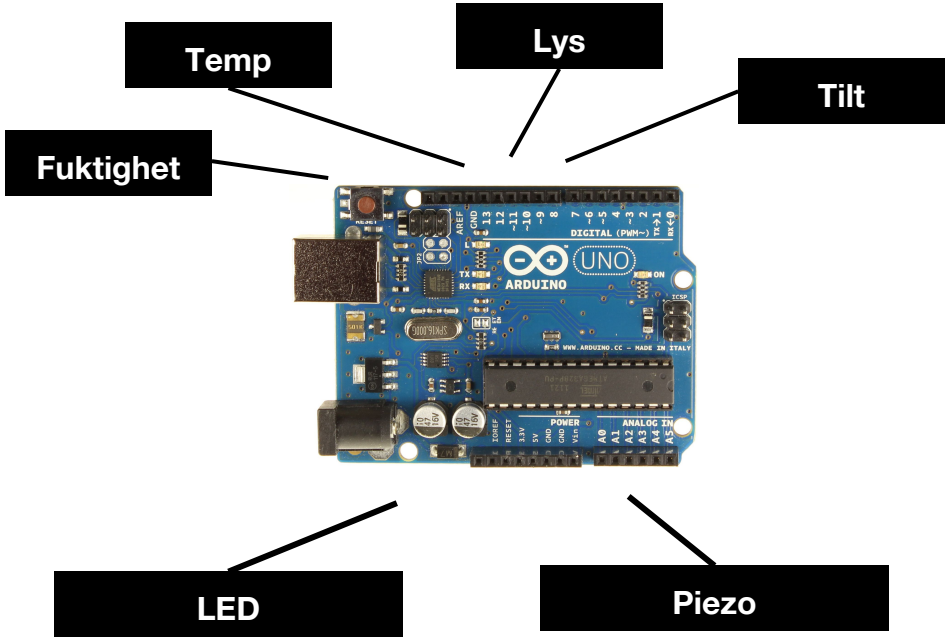
For å oppnå dette kan vi benytte en tilt-sensor.



Modularisering

For å holde oversikt i prosjektet er det nå viktig at vi tenker på hvordan vi skal strukturere koden.

Det kan være lurt å implementere hver av de ulike **funksjonene** vi har listet opp i forskjellige **funksjoner** som så kalles fra loop().



Modularisering

Her har vi satt opp et skjelett for hvilke metoder vi kan benytte for å løse problemet.

Inndelingen gjør det også lett å dele opp kodingen på flere personer.

Input fra miljø
Lys, temperatur, luftfuktighet.

Input fra bruker (er den på)
Tilt-sensor

```
boolean lightTooLow() {

}

boolean humidityTooHigh() {

}

boolean temperatureTooHigh() {

}

boolean on() {

}
```

Modularisering

Slå av og på

```
int tiltSwitch = 2;

boolean on() {

   return digitalRead(tiltSwitch);

}
```

Sjekke lysnivå

```
int lightSensor = A0;
int lightThreshold = 420;

boolean lightTooLow() {

   int light = analogRead(lightSensor);
   return (light < lightThreshold);

}
```

Modularisering

Feedback til bruker

Vi trenger også noen metoder for å gi feedback til bruker via LED-pærer og Piezo-element.

```
void alertUser(int ledPort) {

    blink(ledPort);
    makeSound(seconds);

}

void blink(int ledPort) {
     //implementasjon
}

void makeSound(int seconds) {
    //implementasjon
}
```

Modularisering

```
void loop() {

   if (on()) {

       if (lightTooLow()) {
           alertUser(lightLed);

       } else if (humidityTooHigh()) {
          alertUser(humidityLed);

       } else if (temperatureTooHigh()) {
           alertUser(temperatureLed);

       }
   }
}
```

Koden til venstre gir et eksempel på hvordan loop kan se ut.

Siden vi har brukt selvforklarende navn på metodene, er det lett for alle å se hva de gjør, og alle er ikke nødt til å sette seg inn i detaljer for hver metode.

Oppsummering - modularisering

Å modularisere koden vår i funksjoner hjelper oss med:

- Bryte opp et problem til mindre delproblemer
- Lage lettlest kode
- Koble sammen arbeidet til flere personer

Dette ved å abstrahere bort kompleksitet

Still deg disse spørsmålene før du begynner å løse problemet, og hele tiden underveis:

- Hvilke funksjonalitet må jeg ha for å løse problemet?
  → Danner grunnlag for hovedfunksjoner

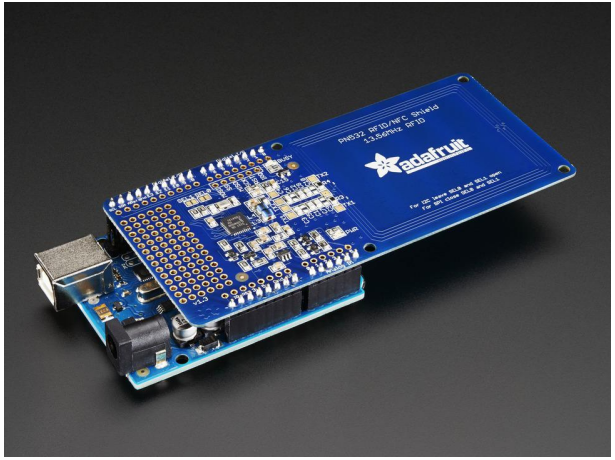
- Bruker jeg denne (del)-koden flere ganger?
  → Kanskje det bør være en egen funksjon.

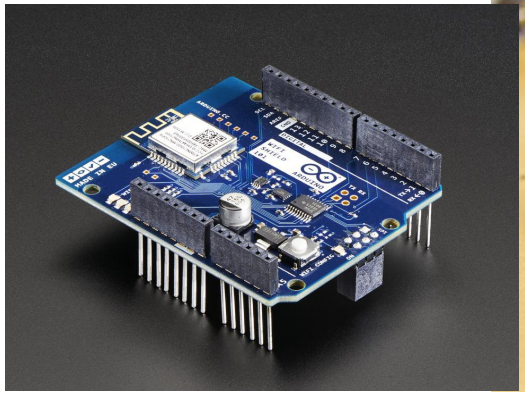
- Kan denne delen av koden uttrykkes som funksjonalitet i seg selv?
  → Legg det i en egen funksjon

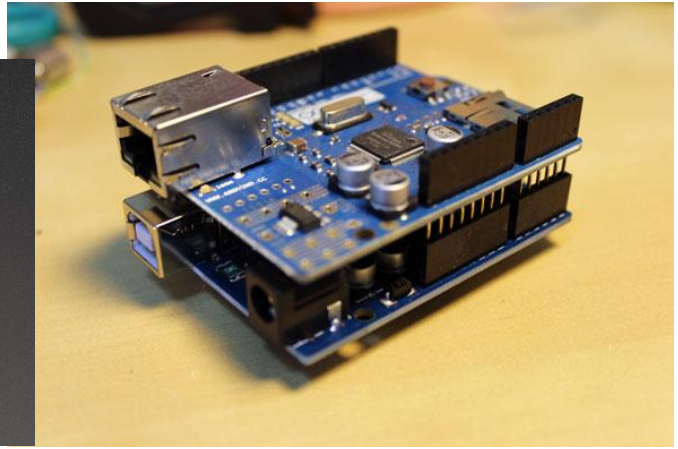
Arduinoprototypen i prosjektet

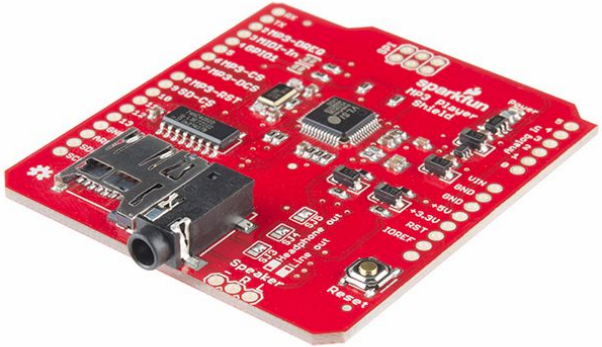
Shields

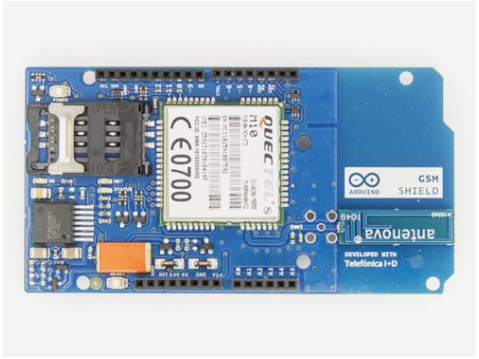
Shields


adafruit






SparkFun
mp3 player
Shield


GSM
SHIELD
antenova


Keyes_SR1y
SONGLE
10A 250VAC 10A 125VAC
10A 30VDC 10A 28VDC
SRD-05VDC-SL-C

Shields

Kilde: learn.adafuit.com

Kjøpe Shields

Det finnes mange steder å handle komponenter til Arduino.

Sparkfun
Kjell & company
Adafruit.com

Det er lurt å sjekke hva slags API/ bibliotek som er utviklet til komponenten på forhånd. Dette kan man som regel finne på nettsiden til produktet.

Det er også viktig å lese nøye hvordan komponenten skal kobles. Noen komponenter skal for eksempel benytte 3.3V-port i stedet for 5V, og kan bli ødelagt om man blander disse.

Arduinoprototypen i prosjektet

Processing

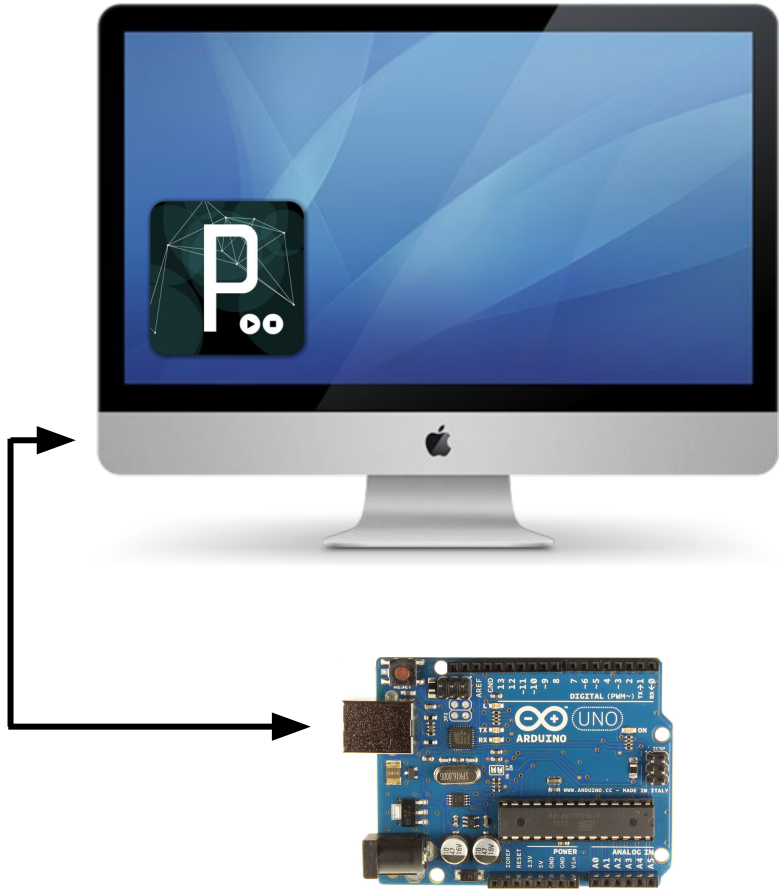
Processing

Processing er et språk og en verktøykasse hvor vi enkelt kan lage visuelle elementer og interaksjon mellom disse og bruker.

Det er forholdsvis enkelt å få dette til å snakke med Arduino-brett.

Dette gjør at vi kan lage skjermbaserte visualiseringer og interaksjonselementer som kan styres fra, eller styre en Arduino.

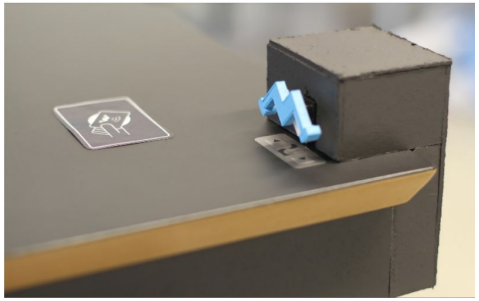
Les mer og lær her.

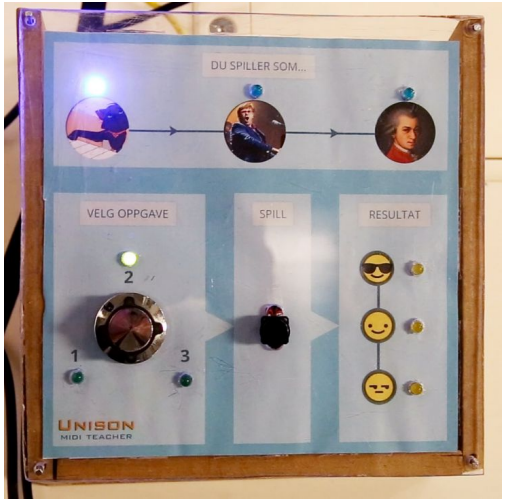


Arduinoprototypen i prosjektet

Fra breadboard til prototype

Fysisk utforming





Fysisk utforming

Handler om å være kreativ.

- Lodding
- Ledninger
- Kontakter
- 3D-printing
- Treverk f.eks. kryssfinér
- Lim

Det er mye utstyr til dette på Sonen.

Husk: Dimensjonene vi ønsker å teste er viktig for valg i utformingen av prototypen.

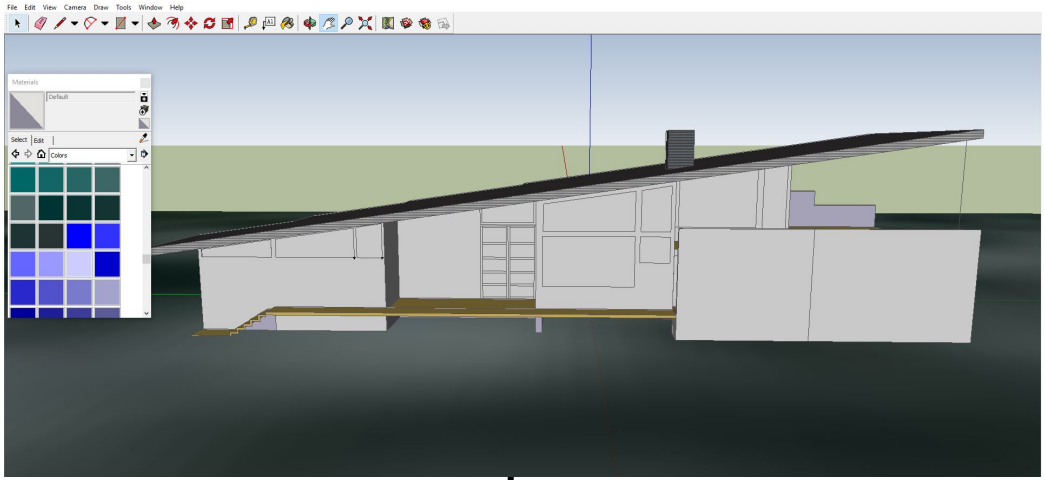
3D-printing

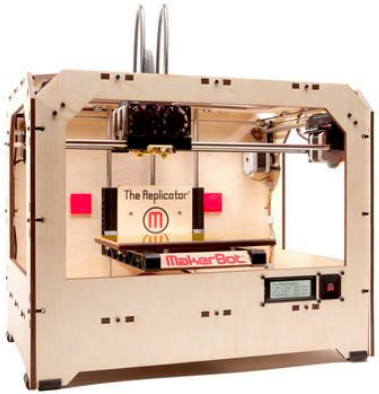
En 3D-printer bruker et passende materiale til å skrive ut lag på lag for å printe i tre dimensjoner.

Det er en slik printer på Sonen som bruker plast i forskjellige farger som materiale.

Denne kan dere bruke, og det er bare å spørre de som er der om assistanse.

For å lage modellene brukes et 3D-skisseprogram som for eksempel Sketchup.



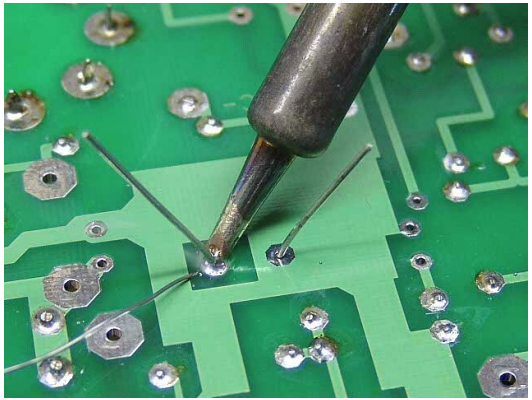


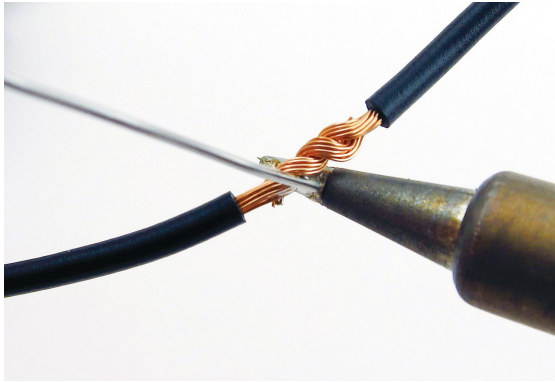
Lodding

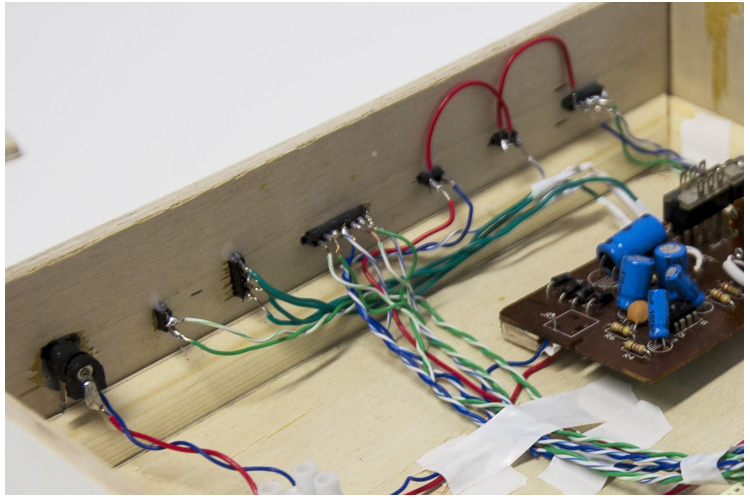
Lodding er en teknikk vi bruker for å feste deler, kort, kontakter og kabler sammen.

Loddebolten er varm, og smelter dermed loddetinn.

Når dette blir kaldt holder det sammen det vi ønsker å feste.







Fremover

Fremover

1. **Gjør ukesoppgaver**
2. **Husk oblig 1 på fredag**
3. **Teknisk verksted fra kl 12:15 i dag på rom C**

Om dere lurer på noe er det bare å sende meg en epost
magl@ifi.uio.no

Takk for meg!