IELET1002 Datateknikk

Øving 3 : Programutvikling + Oppdrag 1

Et Arduino reaksjonsspill

Introduksjon av øvingen
+
Oppdrag 1

Oppgaven

- Det skal utvikles et Arduino reaksjonsspill. Kode + krets
- Koble kretsen med Arduinokortet, koblingsbrett, og komponenter fra kit’et ditt
- Sett opp pseudokode for koden (tenk moduler og flyt)
- Implementer løsningen modul for modul i kjørbar kode
- Demonstrer en fungerende løsning med fysisk oppkobling og kode

Oppdragene

Jobb dere gjennom Oppdrag 1 - 4 for å få litt hjelp på veien

Oppdrag 1 finner dere i denne filen, mens 2-4 ligger i hver sin separate fil

Merk!
Løsningene på oppdragene gir ikke nødvendigvis et eksakt svar på hvordan koden for den tilsvarende utfordringen må bli i sluttløsningen for Reaksjonsspillet, men det hjelper dere å stykke opp problemstillingene og skal langt på vei gi dere svaret.

Sagt med andre ord: dere kan ikke regne med at det bare er å sy koden fra løsningen av oppdragene direkte inn i sluttløsningen.

Oppgaven diskuteres i gruppene *)

men hver enkelt må koble sin egen spillkrets
..skrive sine egen kode og kommentarer
..og få løsningen til å fungere

*) Det må jobbes i gruppene slik som beskrevet tidligere, dvs. med kompetanse-overføring som en klar målsetting. Gruppene må sikre at alle i gruppa klarer å løse øvingen på egen hånd.

Innleveringen

Lever din egen kode på Blackboard som PDF eller .ino-fil (Arduino kodefil), fullt kommentert. Pluss en kort videosnutt som viser at spillet fungerer.

Lever et refleksjonsnotat (PDF) hvor du reflekterer over hvordan kompetanseoverføringen og samarbeidet i gruppen fungerte nå. Forbedring siden forrige gang? Var det noe som ikke fungerte forrige gang som dere fikk til denne gangen? Noe dere skal jobbe med som gruppe for å få til bedre neste gang? Klarer du å dra nytte av gruppearbeids-biten? Etc.

(Husk å repetere hvordan dere skal skrive refleksjonsnotater. Det skal ikke være en ren tilstandsrapport fra felten, den må inneholde refleksjoner!)

Funksjonsbeskrivelse

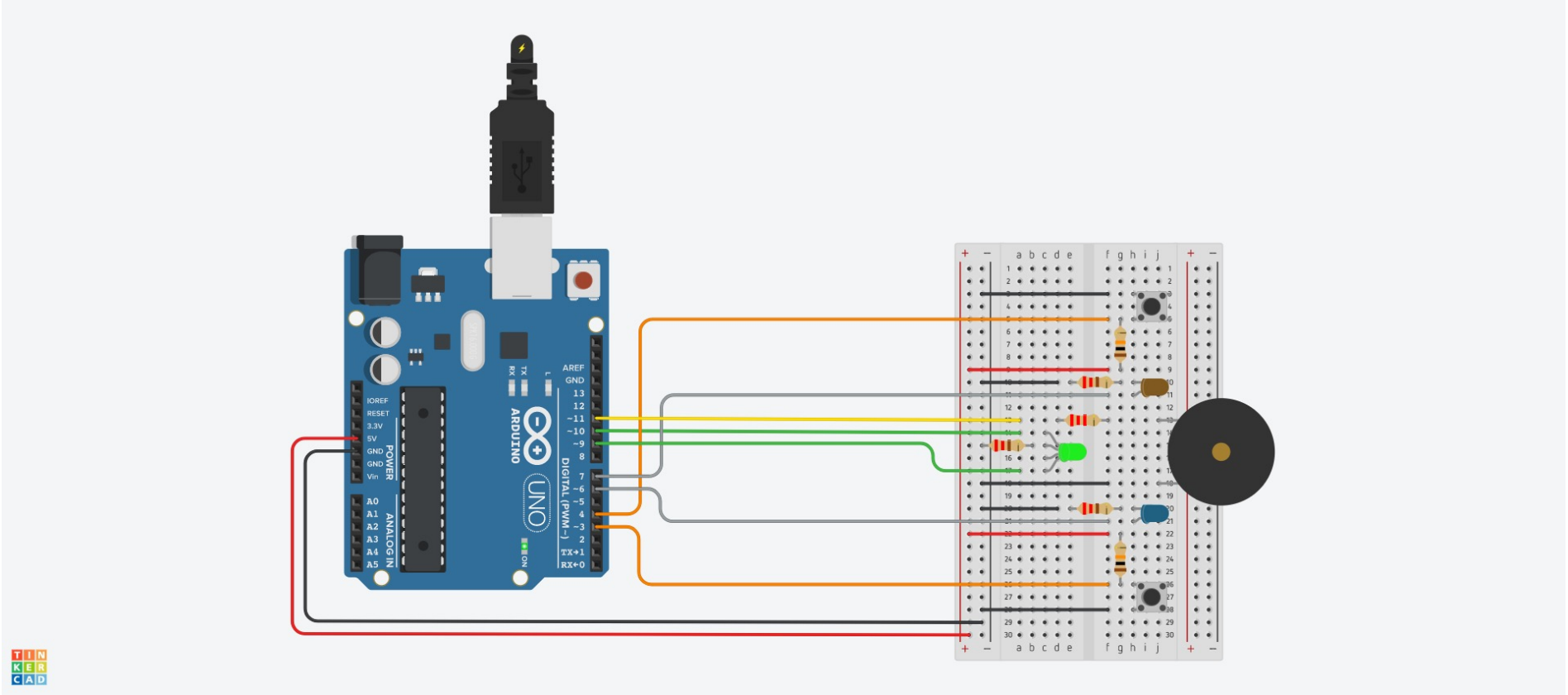
- To spillere med hver sin spillknapp og indikator-LED. En RGB-diode styrer spillet.
- Når RGB-dioden tidsrandomisert (3-6 sek) skifter fra Rød til Grønn, så vinner den som først trykker på spillknappen sin.
- Er du først til å trykke etter at RGBen har blitt grønn, får du en fanfare fra buzzeren, mens RGBen og din indikator-LED blinker raskt flere ganger
- Trykker du før grønn RGB, taper du runden og får en «feillyd» fra buzzeren, mens RGBen og din indikator-LED blinker raskt flere ganger

Krav: Fanfare og feillyd

- Både vinnerfanfare og feillyd skal løses med en ***for-løkke***
- Vinnerfanfaren skal ha suksessivt økende «pitch» som starter på 750, samtidig som grønn i RGB-dioden og vinnerens LED blinker med rask takt
- Feillyden skal ha konstant «pitch» i de lavere frekvensområder, samtidig som rød i RGB-dioden og taperens LED blinker med rask takt
- La begge vare opp til et par sekunder

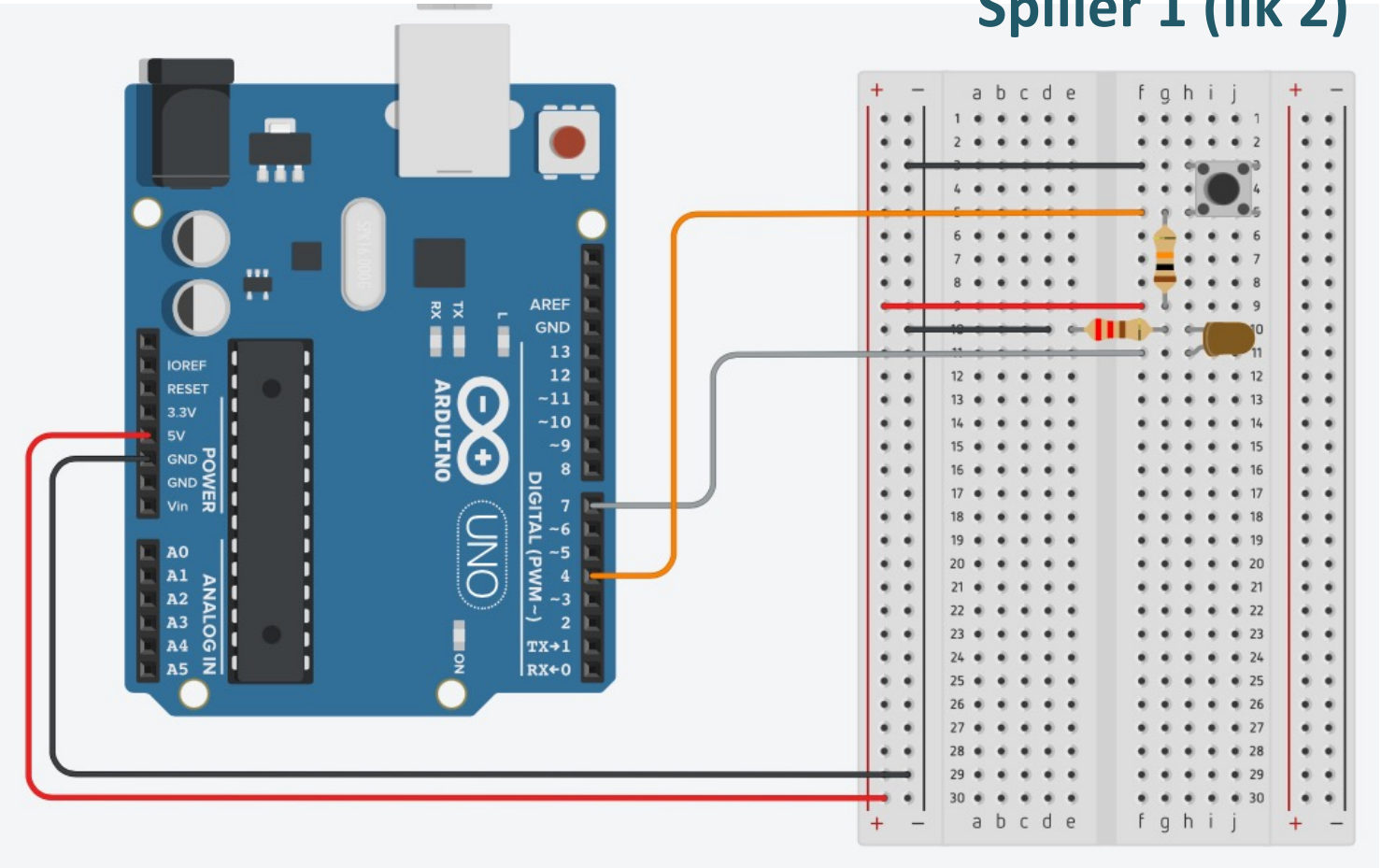
Kretsen

http://tinyurl.com/y3gu6vzp



Oppbyggingen av hele kretsen (unntatt Spiller 2)

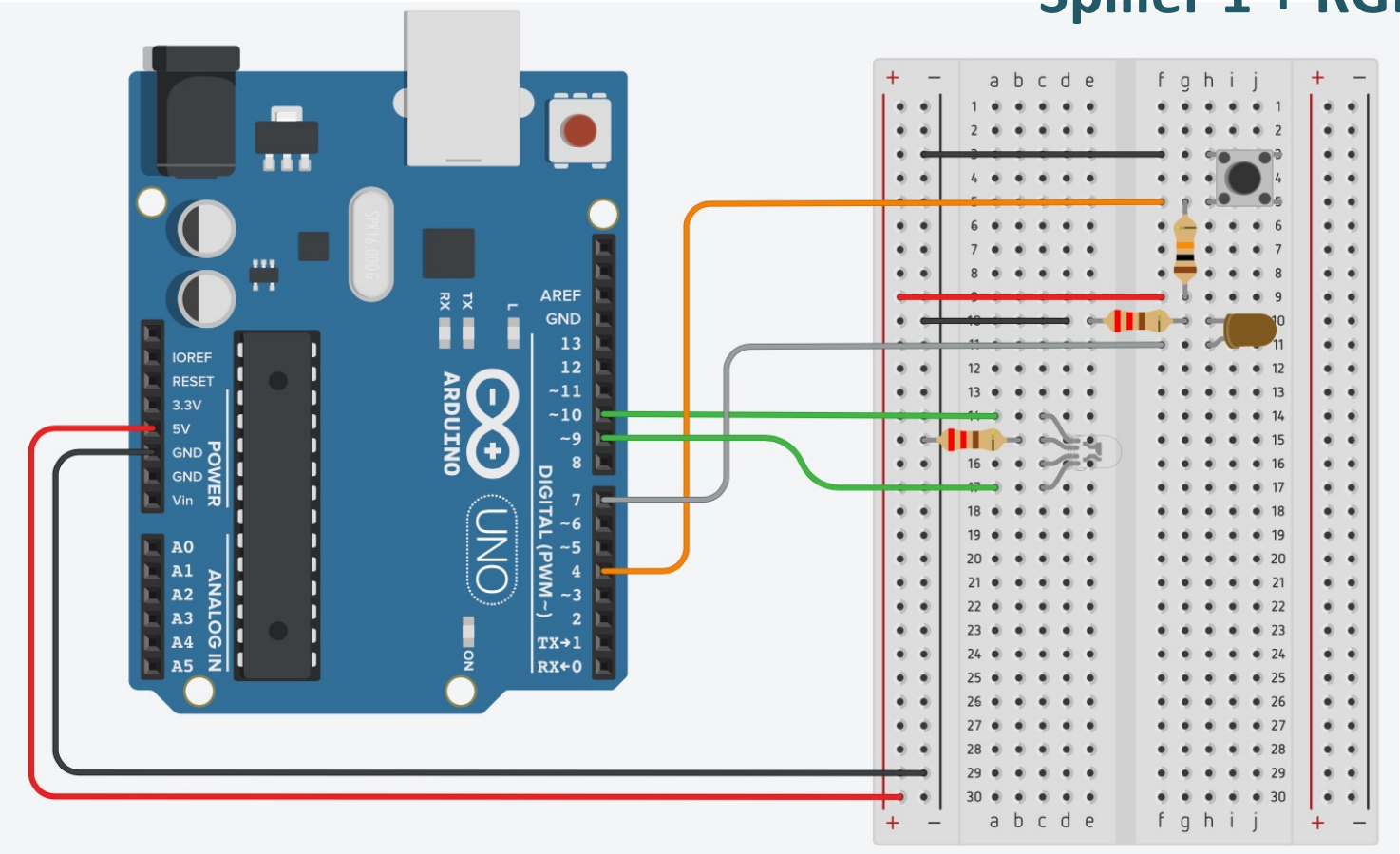
Knappe-krets + LED for spiller 1



Oppbyggingen av hele kretsen (unntatt Spiller 2)

Spiller 1 + RGB

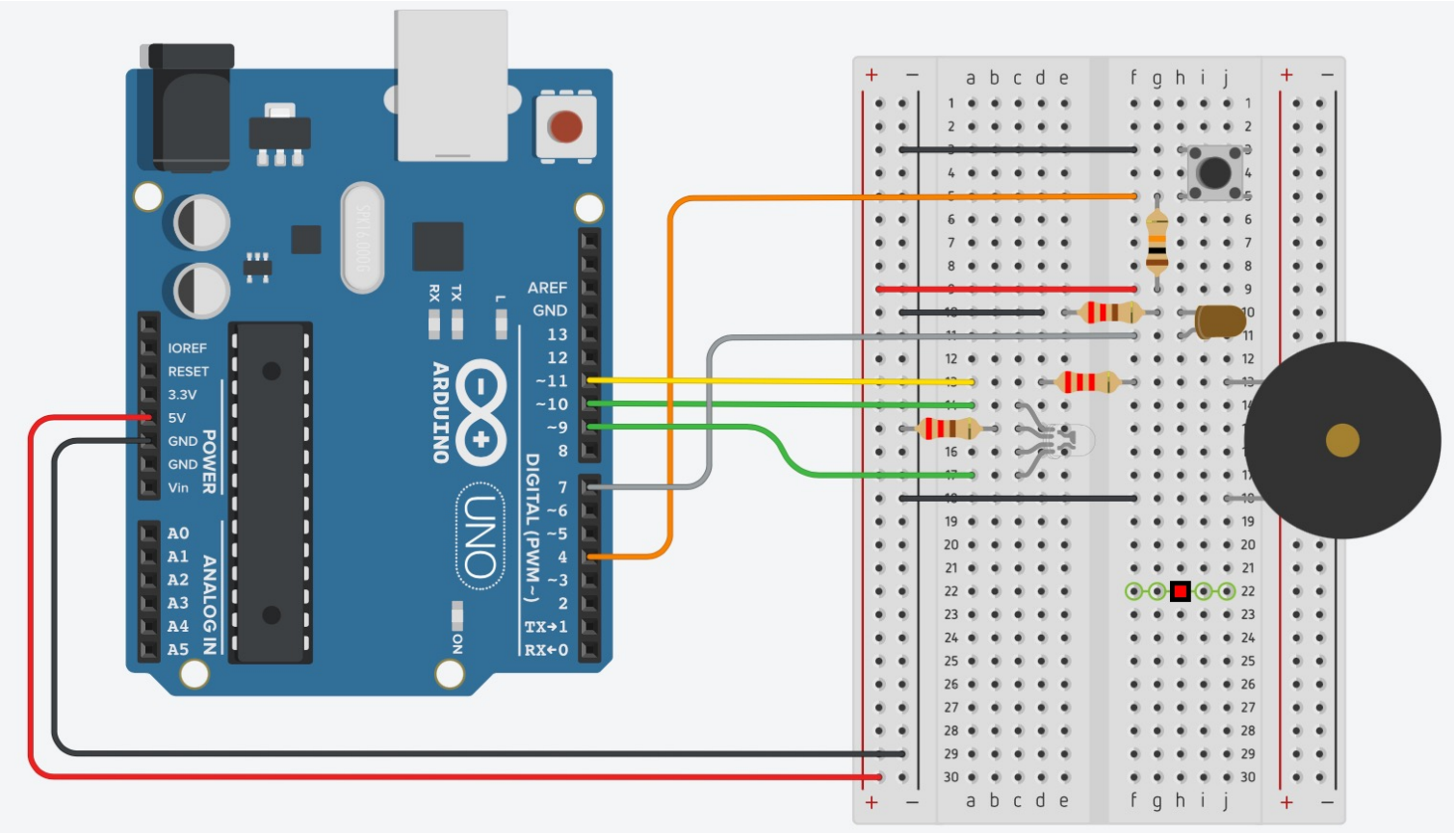
+ RGB



Oppbyggingen av hele kretsen (unntatt Spiller 2)

Spiller 1 + RGB + Buzzer

+ Buzzer



Litt hjelp på veien

På de neste to slides får dere oppgitt de variabler og konstanter som vi tenker dere trenger. Bruk disse – det vil forenkle veiledningen dramatisk om alle gjør det ☺

Lenken i forrige slide førte dere til Tinkercad, og dere kan godt jobbe der, med utprøving og simulering, men oppgaven skal sluttføres vha. Arduino-kit'et.

Konstanter og variabler

Benytt følgende konstanter og variabler:
(hjelper dere på vei, og forenkler veiledningen)

RGB-dioden:	*redLED, greenLED*
Spill-knapper:	*SW1, SW2* (for henholdsvis spiller 1 og 2)
Indikator-LEDer:	*LED1, LED2* (spiller 1 og 2)
Buzzer:	*buzzerPin*
Resultat-lyd:	*winnerBeep, faultBeep* (holder pitch-verdiene)

Hvem vant:	*winner*
Registrering av feiltrykk:	fault
Variabel for «random»:	*wait, now (NB! datatype «unsigned long»)*

Konstanter og variabler

Kodeforslag

RGB-dioden

```
const int redLED = 9;
const int greenLED = 10;
```

Spillknapper og LEDs

```
const int LED1 = 6;
const int LED2 = 7;
const int SW1 = 3;
const int SW2 = 4;
```

Buzzer + vinnerfanfare og feillyd

```
const int buzzerPin = 11;
int winner = 0;
int winnerBeep = 750;
int fault = 0;
int faultBeep = 200;
```

Randomisert ventetids-variabler

```
unsigned long wait = 0;
unsigned long now = 0;
```

Først litt om hvordan lage et program som er litt mer enn en programsnutt

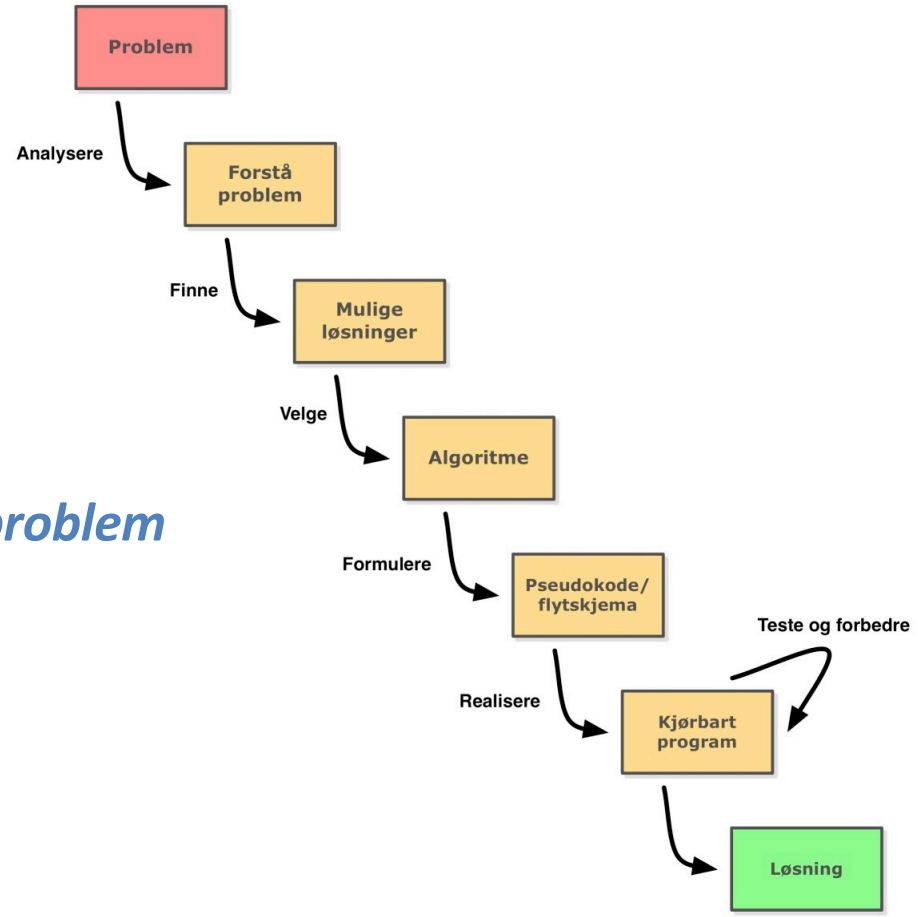
Programdesign

- -Formulere løsninger
- -Algoritmer
- -Pseudokode
- -Flytskjema

Programdesign

Fossefallmetoden:

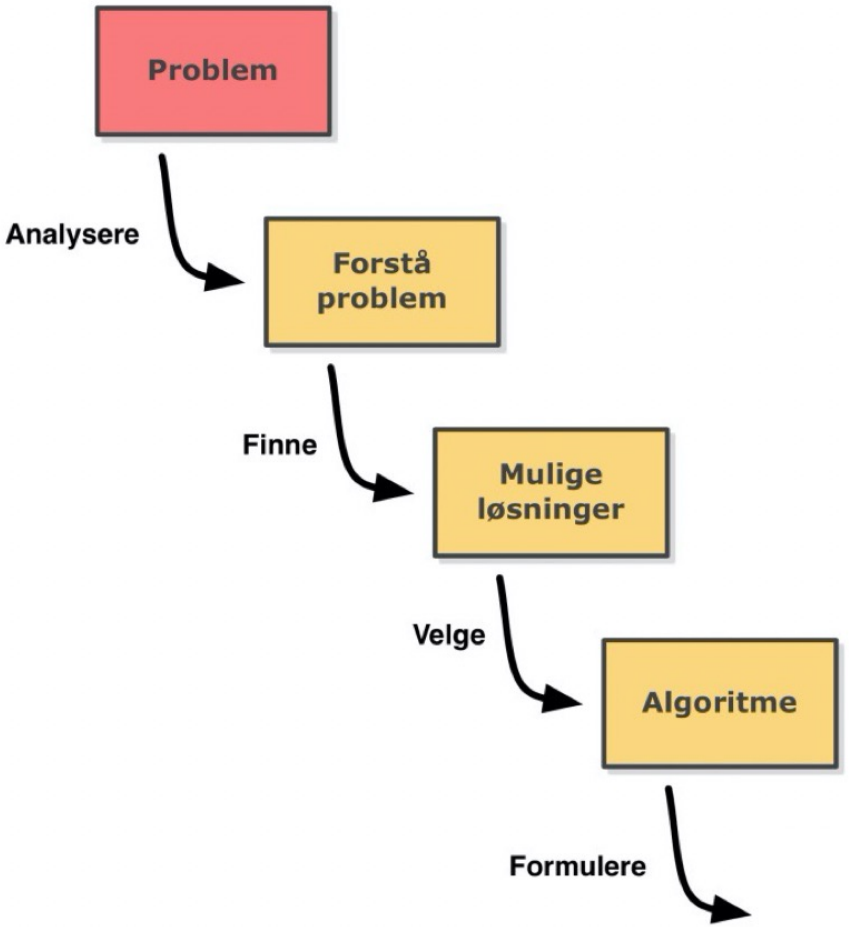
Problemløsning – Fra idé/problem til kjørbart program



Program-design

Problemløsning:

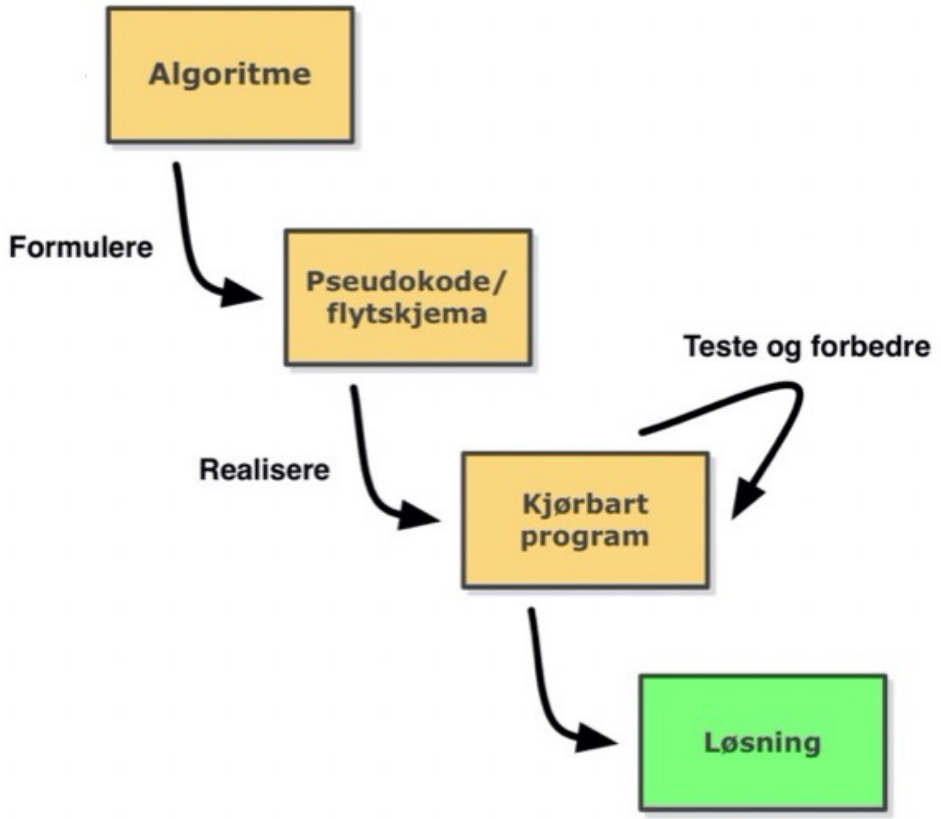
Fra idé/problem
til
kjørbart program



Program-design

Problemløsning:

Fra idé/problem
til
kjørbart program



Algoritmer

- **Hva er en algoritme?**

  En presis beskrivelse av en endelig serie operasjoner som skal utføres for å løse et problem

- **IRL-Eksempel:**

- **Effektiv sortering av en fullstendig blandet kortstokk**
- **Bake en kake**
- **Utføre polynomdivisjon på papiret**

Programløsning / Planskisse

- Her er **to vanlige metoder** i den programutviklingsfasen hvor du skal *formulere en* ***programløsning*** **-** dvs. en ***planskisse*** for programmet ditt:

- **Pseudokode:** Strukturert tekstlig beskrivelse av programmet
  - ✓ Nyttig for å få oversikt over logikken i programløsningen
  - ✓ Trenger ikke å forholde oss til korrekt syntaks
  - ✓ Gir en "slagplan" før vi starter kodingen

- **Flytdiagram:** Skjematisk/visuell framstilling av programmet
  - ✓ Tydeliggjør hvilke valg som tas i et program, og følgene av det
  - ✓ Striktere elementer og en grovere framstilling enn i pseudokode

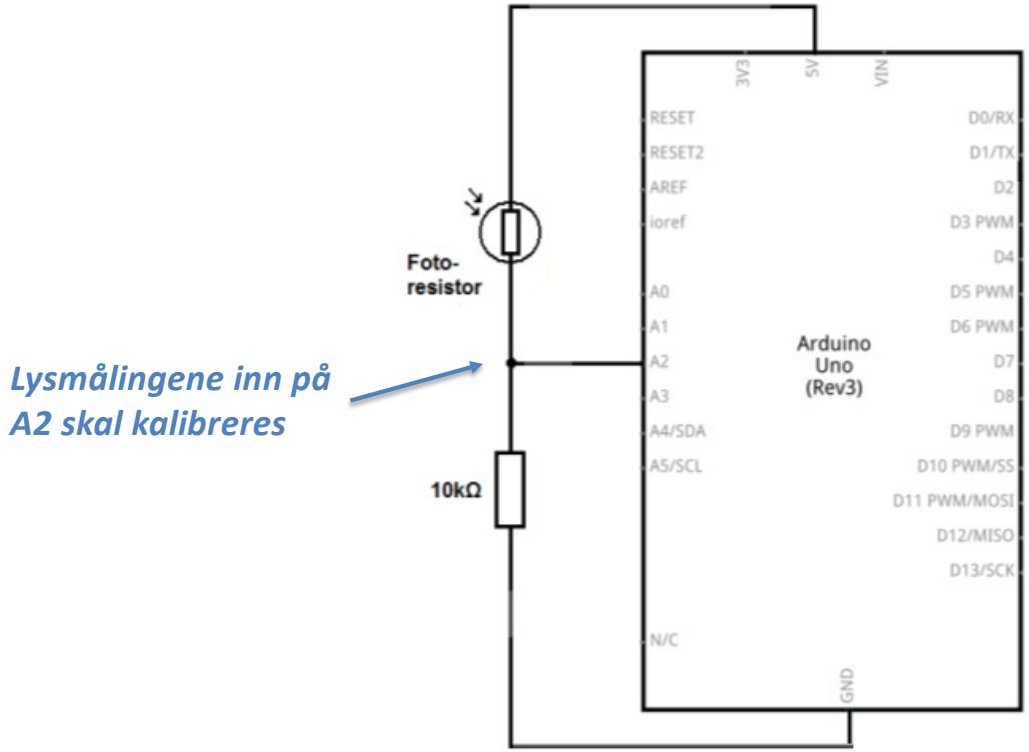
Eksempel: Pseudokode og Flytskjema

Kalibrering av målinger: Skriv en kodesnutt som skal lese inn stadig nye verdier fra en ***lyssensor*** tilkoblet Arduinoen, mens den hele tiden oppdaterer en ***max-*** og en ***min-Verdi***.

Vi kan litt forenklet si at vi ved å kalibrere systemet ***finner arbeidsområdet*** for sensoren i den aktulle anvendelsen – eller at vi ***stiller inn*** sensoren.

(Se koblingsskjemaet på neste slide)

Eksempel: Pseudokode og Flytskjema



Lysmålingene inn på A2 skal kalibreres

Problemløsing med Pseudokode

Vi antar at kodesnutten ligger i hovedprogrammet, *void loop,* og at den repeteres i det "uendelige".

Pseudokode:

Initialiser variablene ***sensorMax/sensorMin***

Start innlesing:

Les inn ***sensorVerdi***

Hvis ny *sensorVerdi* er større enn *sensorMax*:

Oppdater ***sensorMax*** med ny verdi

Hvis ny *sensorVerdi* er mindre enn *sensorMin*:

Oppdater ***sensorMin*** med ny verdi

Gå tilbake til «Start innlesing» og gjenta

Programkode:

```
void loop()  {
int sensorMax = 0;
int sensorMin = 1023;

sensorVerdi = analogRead(sensorPin);
   if (sensorVerdi > sensorMax) {
        sensorMax = sensorVerdi;
     }
   if (sensorVerdi < sensorMin) {
        sensorMin = sensorVerdi;
     }
}
```

Quiz

Før du går videre til neste slide:

Om du ikke la merke til det første gangen – det er et problem ved programkoden slik den er satt opp. Ser du hva som er problemet?

Finner du ikke ut av det, så er svaret på neste side

Problemløsing med Pseudokode

Vi antar at kodesnutten ligger i hovedprogrammet, *void loop,* og at den repeteres i det "uendelige".

Pseudokode:

Initialiser variablene ***sensorMax/sensorMin***

Start innlesing:

Les inn ***sensorVerdi***

Hvis ny *sensorVerdi* er større enn *sensorMax*:

Oppdater ***sensorMax*** med ny verdi

Hvis ny *sensorVerdi* er mindre enn *sensorMin*:

Oppdater ***sensorMin*** med ny verdi

Gå tilbake til «Start innlesing» og gjenta

Programkode:

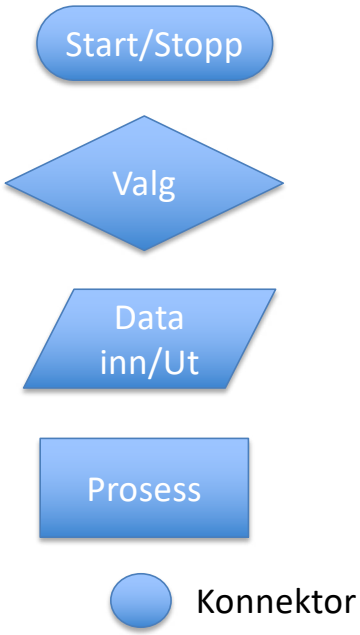
Må ligge utenfor loop()!
Hvis ikke så resettes Max og Min for hver runde!

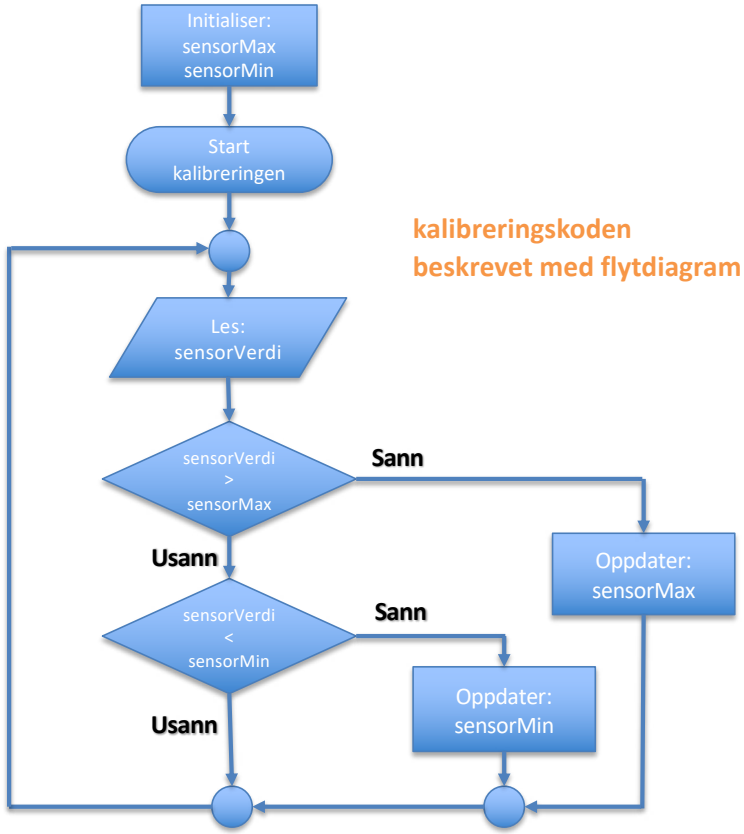
```
void loop()  {
int sensorMax = 0;
int sensorMin = 1023;

sensorVerdi = analogRead(sensorPin);
   if (sensorVerdi > sensorMax) {
        sensorMax = sensorVerdi;
     }
   if (sensorVerdi < sensorMin) {
        sensorMin = sensorVerdi;
     }
}
```

Problemløsing med Flytskjema

Vanlige Symboler/ Elementer





kalibreringskoden beskrevet med flytdiagram

Oppdrag 1: Pseudokode og programskisse

1. Sett opp komplett pseudokode som viser programflyten og ønsket funksjonalitet

2. Konverter pseudokoden til en programskisse (programstruktur)