Arduinoøving 3 - OLED og Switch-case

Gjennom dette laboratorieøvingen skal studenten bli kjent med flere sensorer, en OLED-skjerm og switch-casen. Det vil bli gitt eksempelkode for bruk av de forskjellige sensorene, som skal brukes som utgangspunkt.

I tillegg er fokuset å bruke funksjoner for de forskjellige arbeidsoppgavene programmet skal gjøre. Siden funksjoner fortsatt kan føles litt nytt, begynner dette dokumentet med en enkel repetisjon av hvordan man bruker og lager funksjoner.

Funksjoner

Grunnstrukturen

En funksjon er et forhåndsdefinert område med kode. Formålet med funksjoner er flerfoldig, men formålet vi benytter oss av i dag er lesbarhet. Den generelle formen for en funksjon er som følger:

```
void min_funksjon() {
  // Koden inne i funksjonen.
  // Denne kan være flere linjer.
}
```

Denne er av det enkleste formatet. Koden vil ikke utføres før funksjonen kalles på. Dette gjøres på følgende måte, her som eksempel inne i void loop:

```
void loop() {
  min_funksjon();
}
```

Inngangsargument

En funksjon kan også ta inn verdier som benyttes inne i funksjonen. Disse må defineres i funksjonen med datatype og navn:

```
void legg_sammen_tall(int a, int b) {
  int sum = a + b;
}
```

Her kan man se at funksjonen tar inn to verdier, a og b, og bruker disse i koden. Måten man kaller på en slik funksjon er som følger:

```
void loop() {
  legg_sammen_tall(10, 5);
}
```

Returverdier

Nå gir det mening at funksjonen sender data tilbake igjen. Den har fått noen verdier, gjort beregninger og funnet ut noe vi mest sannsynlig ønsker å vite om. Dette bruker man nøkkelordet return for. Når return blir kjørt avsluttes funksjonen og verdien bak return sendes ut. Her er funksjonen over utvidet med retur-verdi:

```
float legg_sammen_tall(int a, int b) {
  int sum = a + b;
  return sum;
}
```

Denne funksjonen blir kalt på samme måte som forrige, men nå må vi ta vare på retur-verdien:

```
void loop() {
  int tall = legg_sammen_tall(5, 10);
}
```

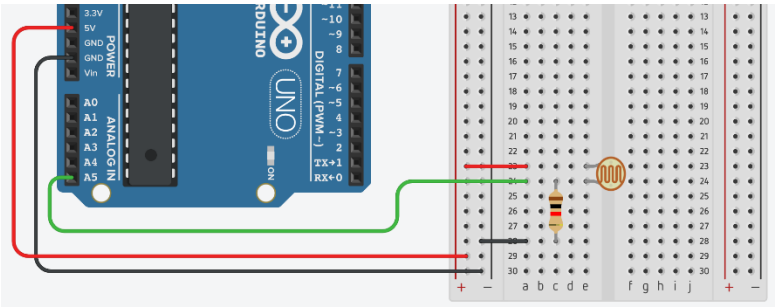
Oppgave 1 – Photoresistor

Teori

En motstand som varierer med lysstyrken den blir utsatt for. Har to pinner og må kobles som en spenningsdeler for å kunne blir avlest. Retning den brukes har ingen innvirkning på funksjonen, men man kan invertere resultatet ved å bytte plassen på motstanden og photoresistoren.

Oppkobling i Tinkercad

Valg av seriemotstand avhenger av type Photoresistor. Tinkercad sin skal være av typen 4.7KOhm, derfor bruker vi samme verdi på seriemotstanden.



Kodehjelp

Dette er en analog sensor som kan leses av ved å bruke `analogRead`. Når man bruker analogRead returneres en ADC-verdi, som kan regnes om til spenning:

```
// Leser av spenning på analog inngang
float readValue = analogRead(photoResPin);
// Regner om fra ADC-verdi til spenning
float voltage = (readValue*5)/1023;
```

Lag en funksjon for disse to kodelinjene. Her trenger vi en funksjon som returnerer en float. I tillegg skal vi inkludere et inngangsargument som lar oss bestemme hvilken pinne som leses

```
float readVoltage(int pin) {
  ...
  ...

  return voltage;
}
```

For å bruke funksjonen kan vi kalle på den i void loop. Husk å lagre den returnerte verdien i en variabel:

```
void loop()
{
  float read_voltage = readVoltage(photoResPin);
  Serial.println(read_voltage);
}
```

Oppdrag

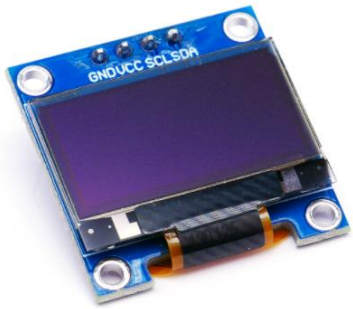
Koble opp, bruk kodeforslag som utgangspunkt og verifiser at både krets og kode oppfører seg som den skal. Sjekk at verdiene kommer opp i seriemonitor, og at de endrer seg ved endring av lysstyrke.

Du må selv deklarere variabler du trenger, og skrive en void setup, som klargjør alt som trengs.

Oppgave 2 – OLED-skjerm

Teori

I denne oppgaven skal en OLED-skjerm kobles på. OLED -skjermen dere finner i kittet er kjent som SSD1306. For å kommunisere med denne skjermen skal vi benytte I2C, og et bibliotek utviklet av Adafruit, skaperne av denne modulen. Hvordan man bruker dette blir forklart under kode-avsnittet.



For å koble opp og programmere skjermen, benytter vi oss av en guide skrevet av Random Nerd Tutorials. Denne finner du her:

https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/

Oppdrag

Bruk koden du skapte i første oppgave som base, og legg på funksjonalitet for OLED-skjerm. Dette må nå flettes inn i forrige kode. Husk at du bare kan ha én setup og én loop. Dette betyr at vi må slå sammen eksempelkoden fra denne oppgaveteksten med forrige oppgave.

Skriv deretter spenning fra photoresistor ut til OLED-skjerm. Ta alltid med benevning!

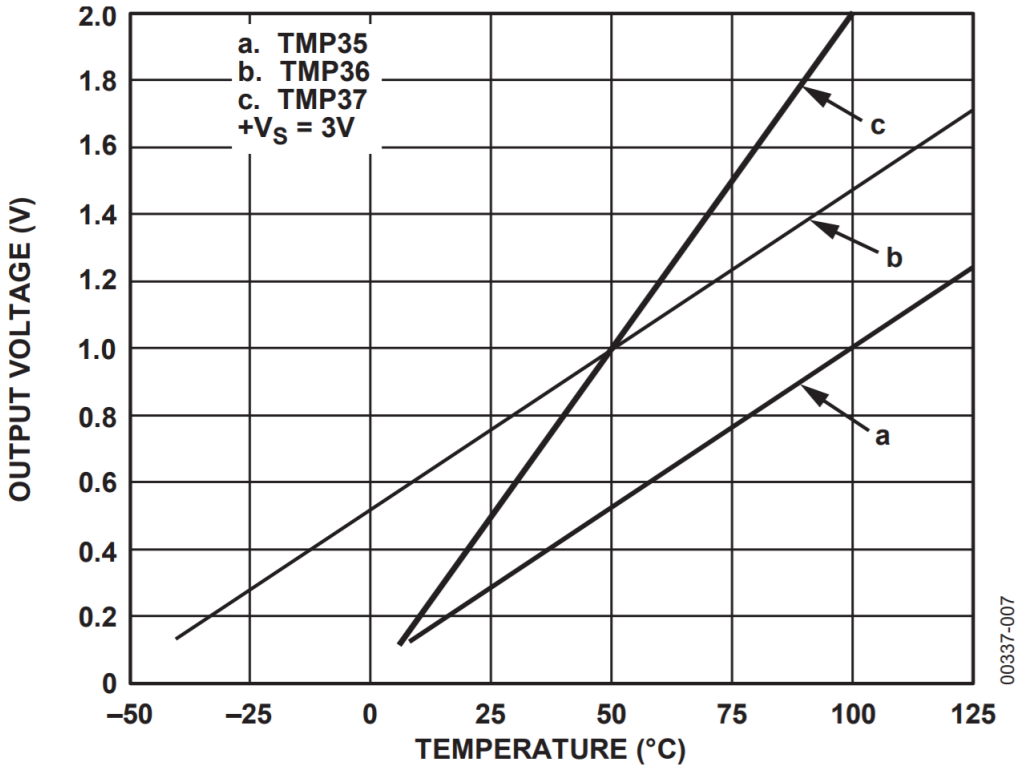
Oppgave 3 – TMP36

Teori

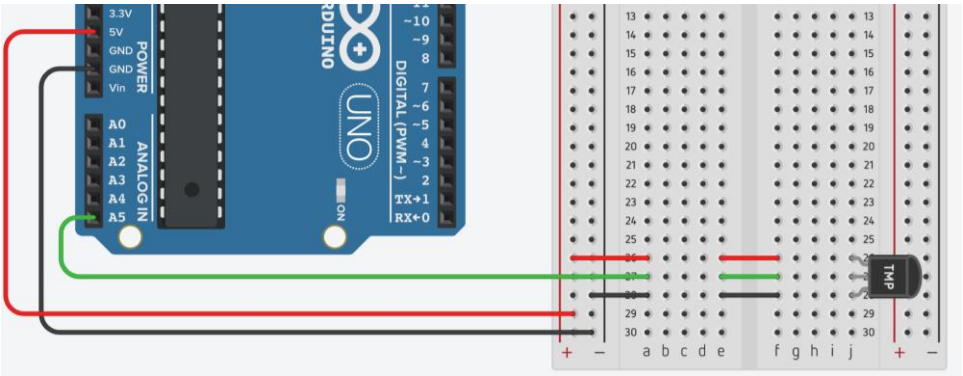
En analog temperatursensor. Denne har tre pinner, som kan angitt på illustrasjon. Se tabell for informasjon om hva hver pinne gjør.

VIN	Forsyingsspenning. Kan være mellom 2.7 V to 5.5 V.
VOUT	Utgangsspenning, varierer med temperatur.
GND	Referanse til jord.

For å regne om spenningen vi leser av til temperatur, må vi vite litt mer om sensoren. Fra datablad kan vi lese at følsomheten til en TMP36 er $\frac{10mV}{°C}$. I tillegg må vi vite hva spenningen er ved 0 grader. Siden responsen til denne sensoren er lineær, er dette nok informasjon til å ekstrapolere ut temperaturen ved en gitt spenning. Se under for responskurven til TMP36



Oppkobling i Tinkercad



Kodehjelp

Disse tre linjene kan brukes for å finne temperaturen i celsius. Les kommentarene for å få mer innblikk i hva hver enkelt linje gjør. Hvis vi studerer disse kodelinjene, kan vi se at de to første er helt like for TMP36 og photoresistor, sett bort fra pinnen vi bruker i funksjonskallet til `analogRead`.

```
// Leser av TMP36
float readValue = analogRead(tmpPin);

// Regner om fra ADC-verdi til spenning
float voltage = (readValue*5)/1023;

// Regner om fra spenning til celsius
float celsius = (voltage-0.5)*100;
```

Vi kan derfor benytte funksjonen vi lagde i forrige oppgave, når vi lager en funksjon for temperaturen.

```
float readTemp() {
  ...
  ...

  return celsius;
}
```

Oppdrag

Koble opp og programmer inn koden for å lese av TMP36-sensoren. Bruk deretter seriemonitor til å verifisere at alt fungerer som det skal. Sjekk at verdien endrer seg når vi påtrykker en temperaturendring.

Når du har verifisert dette, kan du fortsette videreutviklingen av koden fra tidligere oppgaver, og utvide slik at både TMP36 og Photoresistor leses av. Fra forrige oppgave er det bare Photoresistor som skrives til OLED. Nå skal du bytte slik at TMP36 er den som får vise frem verdien sin på OLED-skjerm. Ikke slett koden som viste spenningen fra photoresistoren! Denne skal vi ta i bruk igjen senere.

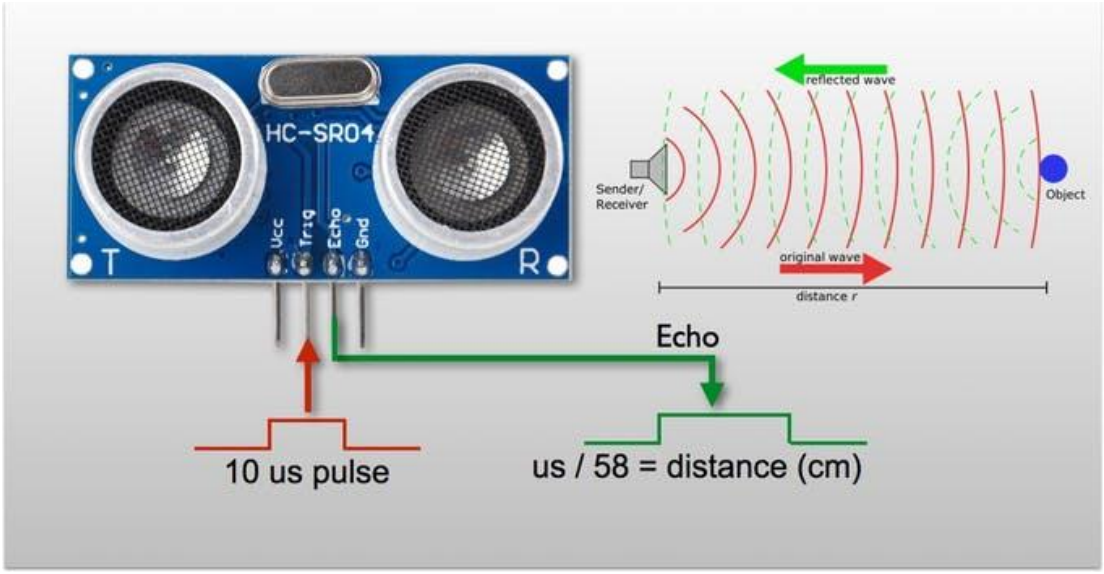
Oppgave 4 – HC-SR04

Teori

En ultrasonisk avstandssensor, kan sammenliknes med sonaren til en flaggermus. Denne har fire pinner, som angitt på illustrasjon. For å starte en måling med denne sensoren må man skru trigger-pinnen høy i 10µs. Deretter leser man av hvor lang pulsen på echo-pinnen er, og regner om til avstand ved hjelp av lydens hastighet.

VCC	Forsyingsspenning. Kan være mellom 2.7 V to 5.5 V.
ECHO	Utgang, blir høy i en periode tilsvarende avstand fra objekt til sensor.
TRIG	Inngang, settes høy i 10µs for å starte måling.
GND	Referanse til jord.

Her er en illustrasjon som viser hva som foregår ved måling.



For å finne ut hvor lenge Echo-pinnen er høy, må vi anvende en ny funksjon. Denne er bygd inn i Arduino og heter PulseIn. Denne returnerer antall µs en pinne har vært på.

Deretter må vi lage en formel som regner om denne verdien til avstand. Her må vi bruke hastigheten til lyd gjennom luft.

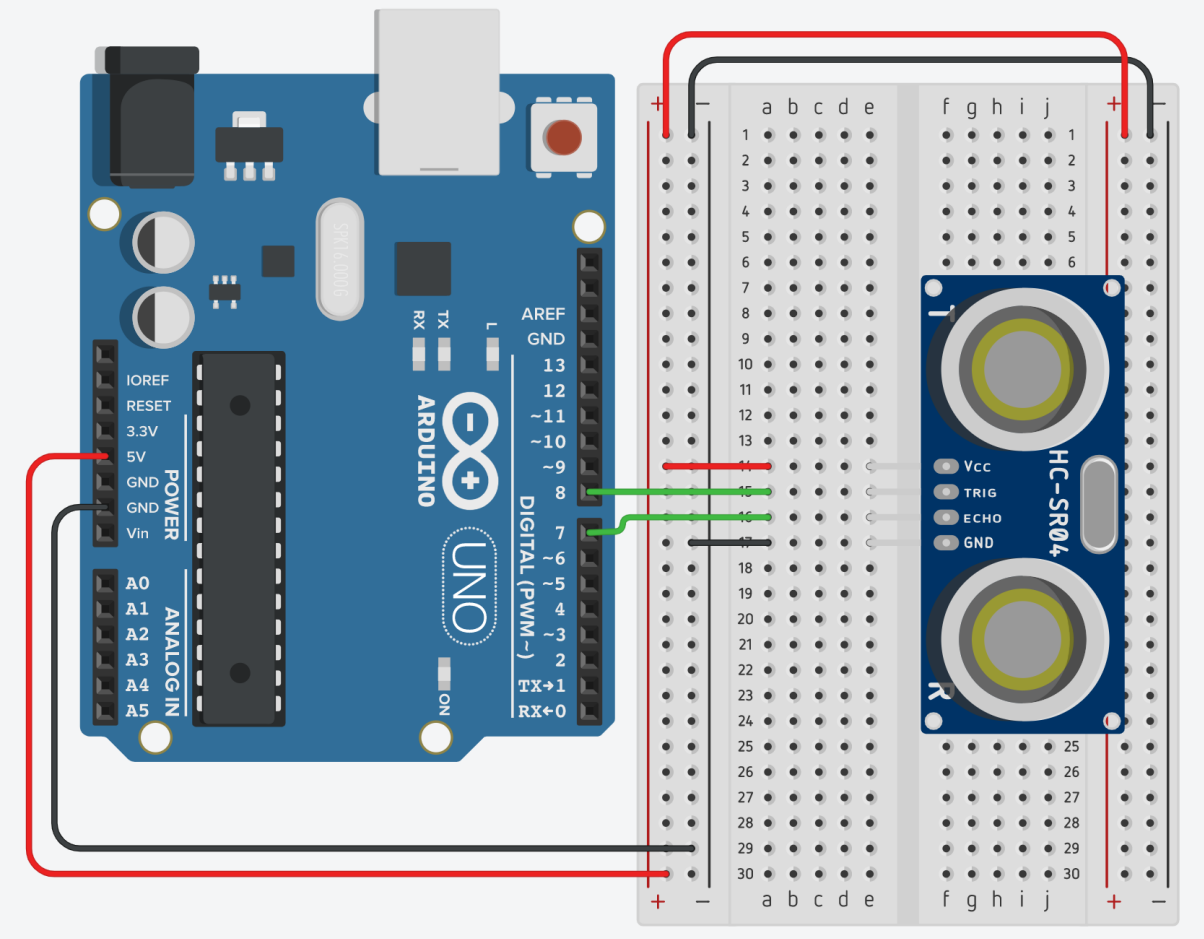
$$c = 343\frac{m}{s}$$

Men her må det justeres til vårt tilfelle. Vi ser etter cm, og jobber med µs. Dette må vi justere for.

$$c = 343\frac{m}{s} = 34\ 300\frac{cm}{s} = 3.43\frac{cm}{ms} = 0.0343\frac{cm}{\mu s}$$

Nå når vi vet hvor mange cm lyden reiser på 1 µs, kan vi benytte dette for å avgjøre hvor langt unna sensoren har merket et objekt. Husk at lyden både må frem og tilbake! Så om lyden har brukt 2µs, er avstanden den halve!

Oppkobling i Tinkercad



Kodehjelp

Dette er sensoren som krever mest kode. To pinner må deklareres:

```
const int trigger = 8;
const int echo = 7;
```

Disse må i tillegg få definert retning i void setup:

```
pinMode(trigger,OUTPUT);
pinMode(echo,INPUT);
```

Her er koden som skal kjøres hver gang sensoren skal leses av. Denne skrur på trigger, leser av echo og regner om til distanse i cm:

```
// Skrur trigger lav for å være sikker på at
// høypulsen er "ren"
digitalWrite(trigger, LOW);
delayMicroseconds(2);

// Skrur på triggerpin i 10µs, deretter av igjen
digitalWrite(trigger, HIGH);
delayMicroseconds(10);
digitalWrite(trigger, LOW);

// Leser av echopinne, og returnerer lydbølgens reisetid
// i mikrosekunder
long duration = pulseIn(echo, HIGH);

// Utregning av distanse. 0.0343 er lydens hastighet i cm/µs.
// Deles på to da lyden skal både frem og tilbake igjen
float distance = duration * 0.0343 / 2;
```

Skriv om koden for sensoravlesning (ikke deklarering og setup) inn i en funksjon. Denne funksjonen skal returnere distansen i centimeter. Du velger selv om trigger og echo skal være inngangsargument, eller være globale variabler.

Oppdrag

Utvid koden du har jobbet med gjennom dagen med kode til HC-SR04. Bruk seriemonitor til å teste systemet. Sjekk at verdiene endres når du endrer treffpunktet til HC-SR04. Skriv nå på samme måte som forrige oppgave en kode som viser frem verdien til HCSR-04 på OLED. Denne skal vises frem i cm.

Oppgave 5 – OLED-karusell

Kodehjelp – Switch Case

Nå som du har verifisert at avstandssensoren fungerer som den skal, er det på tide å få vist frem verdiene den finner på OLED-skjermen. Men siden det er tomt for plass, skal vi nå utvikle en ny måte for OLED-skjermen å vise frem data på.

Grunntanken er å danne en switch-case. Det er denne switch-casen som skal stå ansvarlig for å skrive informasjon til LCD-skjerm. Se på følgende eksempel:

```
switch (LCDstate) {
   case 0:
     // Denne koden kjører hvis LCDstate er 0
     break;
   case 1:
     // Denne koden kjører hvis LCDstate er 1
     break;
   case 2:
     // Denne koden kjører hvis LCDstate er 2
     break;
 }
```

Her har vi gjort klart tre kodeområder, ett for hver stadie OLED-skjermen kan ha. Akkurat nå er det litt forvirrende at OLEDstate skal være tall-verdier, så vi benytter oss av noe som kalles define. Dette er ikke variabler, men omdøping av ord kun for lesers skyld.

```
#define HCSR    0
#define TMP     1
#define PHOTO   2

...

  switch (OLEDstate) {
    case HCSR:
      // Denne koden kjører hvis OLEDstate er 0
      break;
    case TMP:
      // Denne koden kjører hvis OLEDstate er 1
      break;
    case PHOTO:
      // Denne koden kjører hvis OLEDstate er 2
      break;
  }

...
```

Nå er det litt mer oversiktlig hvilken case som gjør hva. Start enkelt med å test at du lykkes med én switch-case. Merk at en switch-case bare er en forenklet måte å skrive en lang if/else-if på.

PS: Du kan også velge andre teknikker her. Direkte tallverdier eller Enums for å nevne noen.

Kodehjelp – Knappetriks

Det neste som må utvikles er en måte for OLEDstate å bytte verdi. Her er tanken at når en knapp blir trykket på, skal OLEDstate økes med én. Her er det viktig at OLEDstate aldri får være større enn 2 i vårt tilfelle. Se på dette trikset for å kjøre en kode bare én gang per knappetrykk:

```
void loop() {
  if (digitalRead(buttonPin)) {
    while (digitalRead(buttonPin));

      // Her kan vi skrive kode som bare vil kjøre én gang for hvert
         tastetrykk
  }
}
```

Vi har nå laget en løsning som gjennomfører handlingen vi ønsker på det som kalles fallende flanke. Når knappen trykkes inn, går vi inn i if-setningen. Det første vi møter på her er en tom while-løkke. Her sitter vi fast så lenge knappen er inne. Når knappen slippes, går koden videre og kjører det resterende av if-setningen. Ønsker du å gjøre dette på stigende flanke, kan du bytte plass på handlingen og while-setningen.

Oppdrag

Din oppgave blir nå å bruke disse kodetipsene til å utvikle en karusellmodus for LCD-skjermen. Knappetrykket skal øke verdien til OLEDstate, så lenge OLEDstate ikke er 2 (da skal vi resette til 0). Deretter skal en switch case bruke denne variabelen til å bestemme hva som skal vises på LCD-skjerm.

Oppgave 6 – Alarm

Oppdrag

I denne oppgaven skal du utvide programmet og koblingen med en alarm. Det er opp til deg hvilken sensor du ønsker å bruke for å slå ut en alarm, og hva alarmen skal være. Enten det er buzzer, LED-blink eller noe annet du kommer på. Du kan også legge på en ny sensor dersom det er noe du ønsker.

OLED-sekvensen skal nå utvides med enda en tilstand, alarmkonfigurasjon. Når OLED-skjermen er i denne tilstanden skal man kunne velge om alarm-delen av koden er aktivert, eller deaktivert. Dette kobles det opp en knapp for å kunne styre. Når knappen trykkes **OG** OLED-skjermen står på alarmkonfigurasjon, skal alarmen deaktiveres/aktiveres, og teksten på skjermen oppdateres.