3. Mine første Arduino-programmer

Eksempel 1: Blinkende lysdiode

I eksempel 1 kobler du en lysdiode til pinne 7 på Arduino-kortet, og får denne til å blinke. Når programmet kjører vil lysdioden lyse i ett sekund, deretter vil den være av i ett sekund, osv.

For teste dette eksempelet trenger du:

- Et Arduino-kort
- En lysdiode / LED (light emitting diode)
- En motstand på 220 eller 330
- Et koblingsbrett og et noen koblingsledninger

En LED fungerer sånn at den lyser når den blir koblet opp med riktig spenning og riktig polaritet. Alle dioder leder strøm i bare en retning, og hvis den blir koblet opp med feil polaritet vil den ikke lyse. En LED tåler ikke 5V, som er spenningen som kommer fra utgangen på et Arduino-kort. Derfor må vi bruke en seriemotstand. Passende seriemotstand for de fleste lysdioder er 220 eller 330. Den elektriske motastanden **R** måles i (uttales ohm), og angir forholdet mellom den elektriske spenningen **U** over motstanden og strømmen **I** som går gjennom motstanden.

Ohms lov:

U=R*I

U = spenning, måles i volt (V)

R = motstand/resistans, måles i ohm ()

I = strøm, måles i ampere (A)

På bildet av lysdiodene ser vi at det ene benet er lengre enn det andre. Denne skal kobles til + (pluss) mens den korteste skal kobles til - (minus). På bildet av motstandene ser vi at de er merket med ringer med forskjellig farge. Fargekodene viser verdien på motstandene.

eksempel_1_blink

```
                                                                  // Dette
er en kommentar
const int ledPin = 7;                          // Definerer en konstant
ledPin som innehar verdien 7
                                                                  // dvs
at lysdioden skal kobles til digital pinne 7
void setup(){
  pinMode( ledPin, OUTPUT );         // Angir at digital pinne 7 skal
være en utgang
}

void loop(){
  digitalWrite( ledPin, HIGH );         // Setter pinne 7 høy
  delay(1000);                                        // Venter i 1000ms
  digitalWrite( ledPin, LOW );        // Setter pinne 7 lav
  delay(1000);                                        // Venter i 1000ms
}
```

GitHub: PLabExamples/examples/01.Basics/BlinkNorwegian.

All tekst som kommer bak tegnene **//** i programkoden er kommentarer, og vil ikke bli tolket av oversetteren (kompilatoren) som gjør om programkoden til kjørbart program. Kommentarene er nyttige for å forstå kode som andre har skrevet, eller som du selv har skrevet for den saks skyld.

Det første som skjer i koden er at vi definerer en konstant av typen **int**. At det er en konstant betyr at den ikke kan tilordnes andre verdier senere i programmet. At typen er int betyr at det er et heltall. Når denne konstanten er definert kan vi bruke ledPin i koden istedet for tallet 7. Det gir mer mening og er lettere å huske. Det kan også hende at du senere blir nødt til å bruke pinne 7 til noe annet, da blir det lettere å omdefintere ledPin til noe annet, istedet for å bytte ut 7 alle stedene i koden du bruker det.

I alle Arduino-programmer finnes funksjonene **setup()** og **loop()**. Det som gjøres i setup() kjører bare når programmet starter, mens det som gjøres i loop() kjøres om og om igjen så lenge programmet går. I setup() her definerer vi at ledPin, altså pinne 7, skal være en utgang: **pinMode(ledPin, OUTPUT);**

? Unknown Attachment

? Unknown Attachment

Lysdioder Motstander

? Unknown Attachment

Eksemplet koblet opp på koblingsbrett (breadboard)

? Unknown Attachment

? Unknown Attachment

Skjema for eksempel 1

? Unknown Attachment

Encodere og potmetere

I loop() setter vi først pinne 7 høy slik at lysdioden lyser ved hjelp av: **digitalWrite(ledPin, HIGH);** de retter kjøres **delay(1000);** som gjør at programmet venter i 1 sekund (1000 ms), deretter settes pinne 7 lav, og så venter vi i 1 sekund igjen. Dette vil fortsette så lenge programmet kjører.

At det står **void** foran setup() og loop() betyr at disse funksjonene ikke returenerer noen verdier, og de tomme parantesene **()** betyr at funksjonene ikke tar innparametere.

Eksempel 2: Blinkende lysdiode med variabel frekvens

Dette eksempelet er en modifikasjon av eksempel 1, der vi kan variere frekvensen på blinkene. Til å gjøre det bruker vi en variabel motstand eller potensiometer, ofte forkortet potmeter. Potmetere brukes typisk som volumkontroller på forsterkere og mye annet elektronisk utstyr.

De to til høyre på bildet er eksempel på potmetere. De to andre er encodere, og fungerer på en helt annen måte.

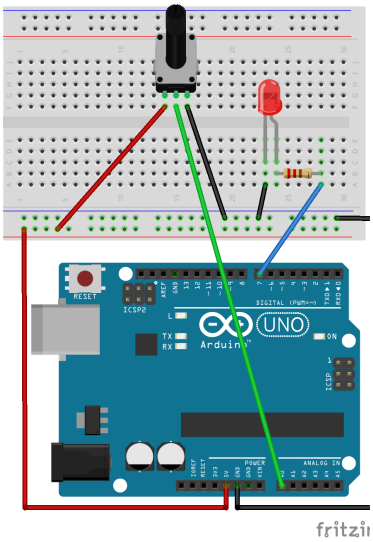
Som skjemaet viser har vi her koblet inn et potmeter på analog inngang 0. Dvs at spenningen på inngang 0 vil variere mellom 0V og 5V når vi vrir på potmeteret.

Det vi skal gjøre i programmet er er å lese av analog inngang 0. Når det er 0V på A0 vil vi lese av 0, og når det er 5V leser vi av 1023. Dvs den verdien vi leser av vil variere fra 0 til 1023 når vi vrir på potmeteret. Deretter bruker vi den avleste verdien som lengden på den tiden (i millisekunder) lysdioden lyser, og den tiden den ikke lyser. For å få til det må vi ta i bruk en **variabel** av type int. Den fungerer nesten som konstanten i eksempel 1, men en variabel kan tilordnes nye verdier mens programmet kjører.

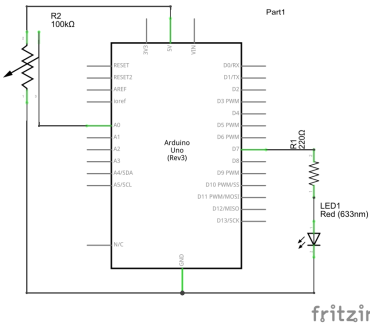
eksempel_2_blink_speed

```
const int ledPin = 7;          // LED kobles til pinne 7
const int potPin = A0;          // Potmeter kobles til analog inngang 0
(A0)
int delayTime;                          // Variabelen delayTime av type
int holder verdien vi leser av på A0

void setup(){
  pinMode( ledPin, OUTPUT );               // Setter pinne 7 til utgang
}

void loop(){
  delayTime = analogRead( potPin ); // Leser av A0 og lagrer verdien i
variabelen delayTime
  digitalWrite( ledPin, HIGH );          // Slår på LED
  delay(delayTime);                                 // Venter i
delayTime antall millisekunder
  digitalWrite( ledPin, LOW );                // Slår av LED
  delay(delayTime);                                 // Venter i
delayTime antall millisekunder
}
```

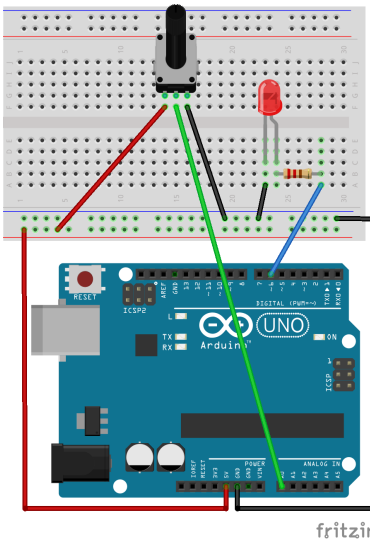
GitHub: PLabExamples/examples/01.Basics/BlinkVariatingFrequency.



Breadboard oppkobling for eksempel 2



Koblingsskjema for eksempel 2



Breadboard oppkobling for eksempel 3

Eksempel 3: Lysdiode med variabel styrke

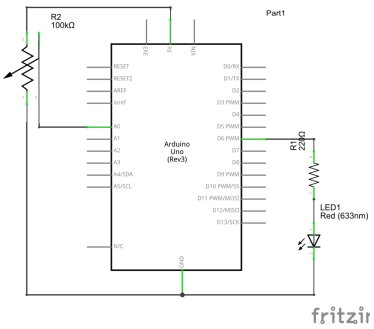
I dette eksempelet skal vi bruke potmeteret til å variere lysstyrken til lysdioden. Arduino har ikke analoge utganger, men vi kan få til dette ved å bruke en teknikk som kalles pulsbreddemodulasjon, på engelsk pulse-width modulation eller PWM. Arduino UNO har 6 utganger som kan kjøre PWM: 3, 5, 6, 9, 10 og 11. Når vi bruker PWM kommer det et pulstog med ca 500Hz på den utgangen vi velger å bruke. Det vil si at den slås av og på 500 ganger per sekund. For å sende et PWM-signal brukes funksjonen analogWrite(pin, value). Her angir pin den utagangen vi ønsker å bruke, og value er et heltall mellom 0 og 255. Value angir hvor lang tid pulsen er høy (5V) i forhold til hvor lang tid den er lav (0V). På den måten kan vi få et slags analogt utsignal. Dette fungerer fint på lysdioder og mange motorer. For å få et ekte analogt utsignal må vi bruke en ekstern D/A-omformer.

Koblingsskjemaet blir nesten som på eksempel 2, men vi må flytte LED til en utgang som støtter PWM. For eksempel D6.

eksempel_3_led_intensity

```
const int ledPin = 6;
const int potPin = A0;
int lightIntensity;


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

void loop(){
  lightIntensity = analogRead( potPin );
  analogWrite( ledPin, lightIntensity / 4 );  // Deler lightIntensity på
4 for å få en verdi mellom 0 og 255
}
```

GitHub: PLabExamples/examples/01.Basics/LedIntensity.

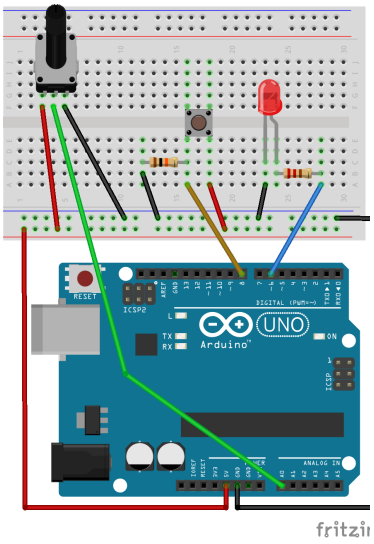


Koblingsskjema for eksempel 3

Eksempel 4: Lysdiode med variabel styrke og trykkbryter

Dette eksempelet er veldig likt eksempel 3, men vi må holde nede en trykkbryter for å få lysdioden til å lyse. Dette gjøres for å forklare et fenomen som kan oppstå når knapper og brytere brukes for å gi signal inn til Arduino.

Som vist på skjemaet til høyre har vi koblet opp en trykkbryter, S1, som gir 5V til D8 når den holdes nede. I tillegg er det koblet opp en ny motstand på 10k (10 kiloohm = 10000). Denne kalles en nedtrekksmotstand, eller pull-down. Denne er der for å forsikre om at signalet på D8 er 0V når knappen ikke er trykket ned. Hvis denne ikke hadde vært tilstede ville vi fått "floating", og verdien på D8 ville blitt tilfeldig når knappen ikke var trykket ned. Det er viktig at verdien på denne motstanden er ganske høy. 10k er vanlig for slike motstander.

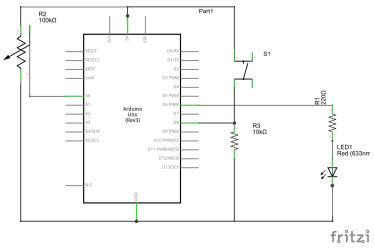


Breadboard oppkobling for eksempel 4

eksempel_4_button

```
const int ledPin = 6;
const int potPin = A0;
const int switchPin = 8;
int lightIntensity;


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

void loop(){
  if( digitalRead( switchPin ) == HIGH ){                // Her tester
vi om bryteren er trykket inn
    lightIntensity = analogRead( potPin );                // Leser
verdien fra potmeteret
    analogWrite( ledPin, lightIntensity / 4 );        // Setter
lysstyrken på LED
  } else {
          analogWrite( ledPin, 0 ); // Setter lysstyrken til 0 hvis
bryteren ikke er trykket inn
  }
}
```

GitHub: PLabExamples/examples/01.Basics/LedIntensityWithButton.



Koblingsskjema for eksempel 4

Her har vi innført en if-setning. Den tester en betingelse og utfører noe hvis betingelsen er oppfylt, deretter kan den også utføre noe annet dersom betingelsen ikke er oppfylt.

En annen ting vi må være klar over med brytere er an vi kan få prell, eller bounce på engelsk. Det som skjer er at når du trykker inn bryteren og kontaktflatene i bryteren nærmer seg hverandre vil de sprette litt fram og tilbake før det endelig blir kontakt. Dette skjer i nesten alle type brytere, og veldig fort slik at det i en lysbryter for eksempel ikke vil ha noen praktisk betydning. Arduino kan oppfatte dette som flere trykk på bryteren etter hverandre. Dette problemet kan fikses i programmet eller med en elektronisk krets.

Eksempel 5: Bruk av serial monitor

Serial monitor er en veldig nyttig feature i Arduino IDE. Den kan for eksempel brukes for å feilsøke programmer eller kretser, eller hvis du bare skal teste en sensor eller lignende. Serial monitor fungerer bare når Arduinoen er tilkoblet datamaskinen, og brukes til å sende tekst til et eget vindu i IDE, men kan også brukes til å lese inn tekst fra tastaturet som sendes til programmet. Serial monitor startes fra Tools-menyen på IDE, eller med Ctrl+Shift+M.I eksempelet under tar vi utgangspunkt i eksempel 3 der vi har potmeter og LED med variabel lysstyrke. Det som er nytt er at verdien fra potmeteret og verdien som sendes til LED sendes til serial monitor.

Serial monitor

```
const int ledPin = 6;
const int potPin = A0;
int lightIntensity, potReading;


void setup(){
  pinMode( ledPin, OUTPUT );
  Serial.begin(9600); // Initierer serial monitor med
overføringshastigheten satt til 9600 baud
}

void loop(){
  potReading = analogRead(potPin);
  lightIntensity = potReading / 4; // Deler potReading på 4 for å få en
verdi mellom 0 og 255
  Serial.print("Potmeterverdi: "); // Serial.print sender tekststrengen
til serial monitor uten linjeskift
  Serial.println(potReading);      // Serial.println sender verdien av
potReading som en tekststreng med linjeskift
  Serial.print("LED-verdi: ");
  Serial.println(lightIntensity);
  analogWrite(ledPin, lightIntensity);  // Setter lysstyrken på LED
}
```