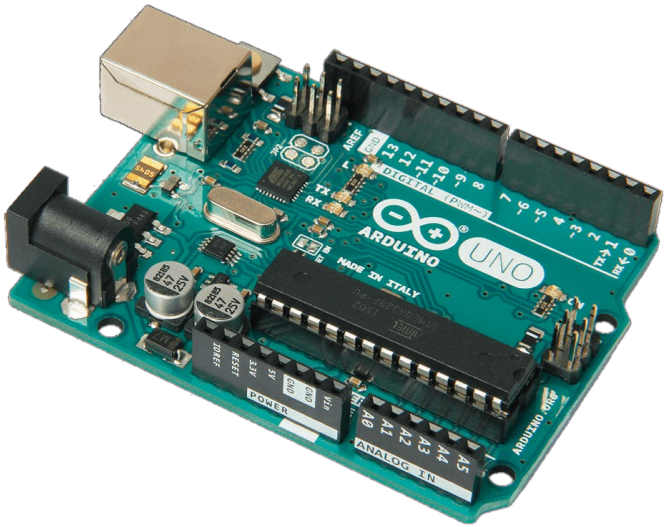
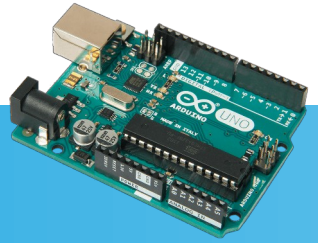
Arduino workshop #2



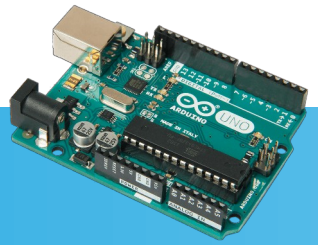
Quiz

- Hvad hedder et Arduino-program?
- Hvilke to funktioner indeholder det altid?
- Hvordan laver man kommentar-blokke?
- Og linje-kommentarer?
- Hvilket tegn slutter kommandoer med i C?



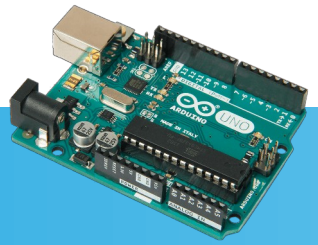
Svar

- Et Arduino-program hedder en **sketch**
- En sketch indeholder altid en **setup**-funktion og en **loop**-funktion
- Kommentarblokke starter med ***/*** og slutter med *****/***
- Linjekommentarer starter med ***//***
- Kommandoer i C slutter med semikolon **;**



Arduino i tal

- Microcontroller: Atmel Atmega 328
- 8-bit mikroprocessor
- 32kB flash (til program/sketch)
- 2kB RAM (til variable og stack)
- 16MHz clockfrekvens

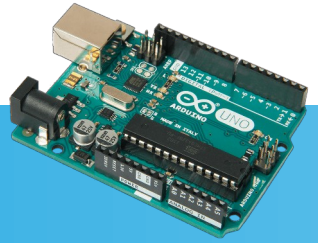


Variable og datatyper

- Hvad er en variabel?
  - Datatyper (se https://www.arduino.cc/reference/en/#variables)
  - bool – to værdier true og false
  - char – et tegn (fx ’A’), eller tal -128 til 127
  - byte (unsigned char) – tal 0 til 255
  - int – tal -32.768 til +32.767
  - word (unsigned int) – tal 0 til 65.535
  - long – tal -2.147.483.648 til 2.147.483.647
  - unsigned long – tal 0 til 4.294.967.295

Løkker

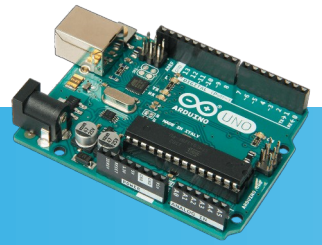
- Der findes overordnet set 3 typer løkker i C/Arduino
  - while-løkke:
    - while (betingelse) { kommandoer; }
  - do-while-løkke (bruges sjældent):
    - do { kommandoer; } while (betingelse);
  - for-løkke (*meget* generel, kan bruges til alt 😊):
    - for (init ; betingelse ; gentagelse) { kommandoer; }
    - f.eks. for (i=0; i<10; i++) { Serial.println(i); }



Sketch med variable og løkke

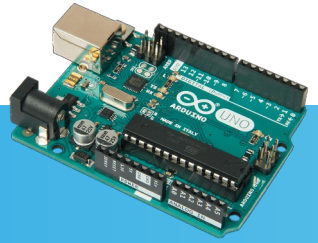
- Indtast sketch'en til højre
- Prøv at forstå hvad den gør
  - Hint: Den lægger nogle tal sammen
- Hvornår stopper sketch'en?
- Der er en fejl i koden, hvor?
  - Ret fejlen og kør (upload til Arduino)

```
1   int i = 0, sum = 0;
2
3   void setup() {
4     Serial.begin(9600);
5     while (sum < 40000) {
6       Serial.println(sum, DEC);
7       i = i + 1;
8       sum = sum + i;
9     }
10  }
11
12  void loop() {
13  }
```



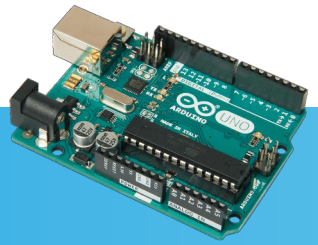
Flydende tal

- float – flydende tal (kommatal) 32-bit
  - Største tal ca. $3,4 * 10^{38}$ ~ 340.000.000.000.000.000.000.000.000.000.000.000
  - ”Mindste” tal ca. $1,17 * 10^{-38}$ ~ 0,00000000000000000000000000000000117
- Beregninger med float er mere krævende end heltal
- Husk man kan (generelt) ikke sammenligne eksakt
  - Fx if (f == 27.0)



Arrays

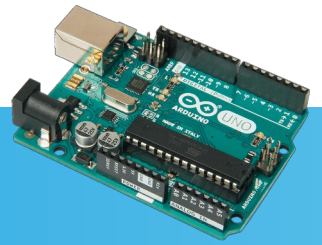
- Et array i C er et fast antal elementer af en given type
- Dvs. din kode kan ikke tilføje eller fjerne elementer
  - (men du kan have en variabel, der holder styr på hvor mange elementer du faktisk bruger)
- Eksempel på et array af 7 integers:
  - int primtal[] = { 2, 3, 5, 7, 11, 13, 17 };
- Eksempel på array, med plads til 10 tal:
  - int numre[10];
  - Indeksering starter med 0



Array, float og for-løkke

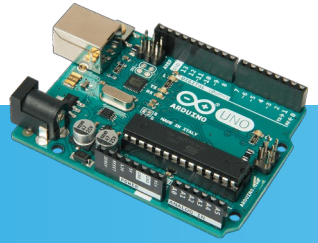
- Afprøv følgende sketch:

```
1  void setup() {
2    Serial.begin(9600);
3    while (!Serial) { }
4  }
5  float arr[] = { 1.1, 2.2, 4.4, 8.8, 16.16, 32.32, 64.64, 128.128 };
6  void loop() {
7    float sum = 0.0;
8    // put your main code here, to run repeatedly:
9    for (int i=0; i<8; i++) {
10     sum += arr[i];
11     Serial.print("arr[");
12     Serial.print(i, DEC);
13     Serial.print("] = ");
14     Serial.print(arr[i]);
15     Serial.print(", sum=");
16     Serial.println(sum);
17   }
18   delay(5000);
19 }
```



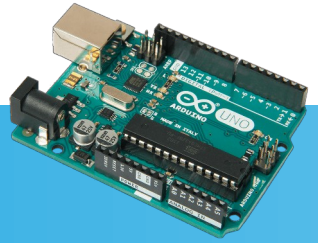
Tekststrenge

- C har ikke egentlige strenge, men bruger char array:
  - char tekststreng[] = ”Dette er en streng”;
  - Strenge er nul-terminerede (slutter med ASCII-værdi 0)
  - char tekst[] = {’S’, ’t’, ’r’, ’e’, ’n’, ’g’, ’\0’};
- Arduino har også en String-klasse:
  - String str = String(”Dette er en anden streng”);
  - Den kan bedre bruges til at manipulere strenge, men den bruger meget hukommelse (husk vi har kun 2kB)



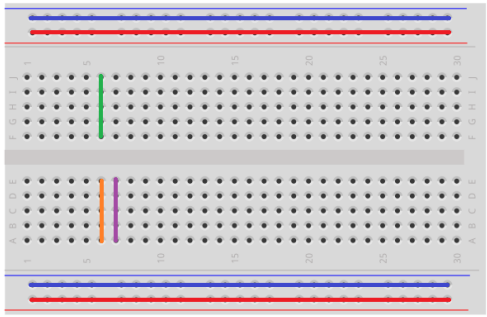
Den specielle type void

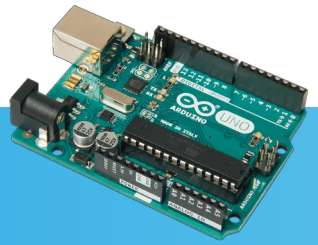
- void betyder ”ingenting”
- Bruges som ”returtype” for funktioner, der ikke returnerer noget (se fx setup og loop)
- I C bruges void også til at angive en funktion, der ikke tager nogen argumenter, men da Arduino også tillader C++ er det ikke nødvendigt her - men man *må* godt:
  - void loop(void)



Introduktion til breadboard

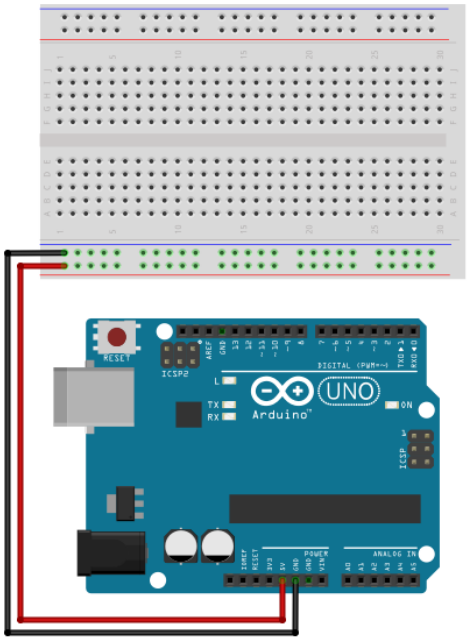
- Et breadboard bruges til let at lave elektroniske kredsløb
- Indeholder en masse huller i et gittermønster. Hullerne har små fjedderkontakter, som skaber elektrisk forbindelse imellem de komponenter/ledninger, som stikkes i hullet.
- Grupperne af 5 huller har indbyrdes forbindelse, som vist med grøn, organge og lilla
- De fleste breadboards har også to power-busser, der bruges til forsyningsspænding 5V på de røde og GND (0V) på de blå.

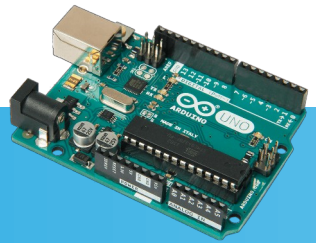




Kredsløb

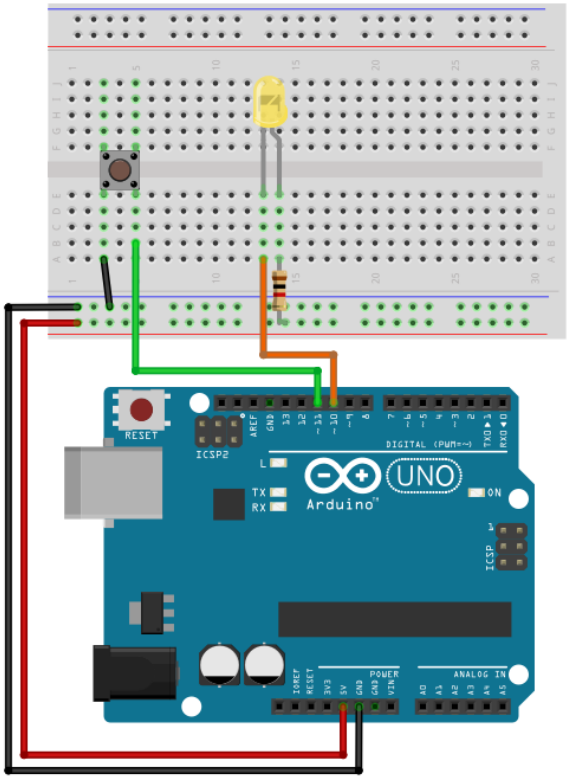
- Tag USB-kablet ud af Arduino mens der laves kredsløb på breadboardet (undgå kortslutninger)
- Forbind GND på Arduino til GND-bus på Breadboard (sort ledning)
- Forbind 5V på Arduino til + bus på breadboard (rød ledning)
- NB: brug M-M dupont-kabler

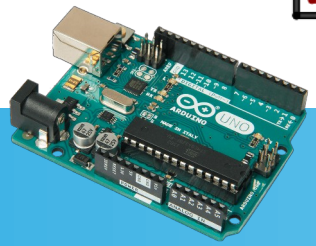




Knap og lysdiode

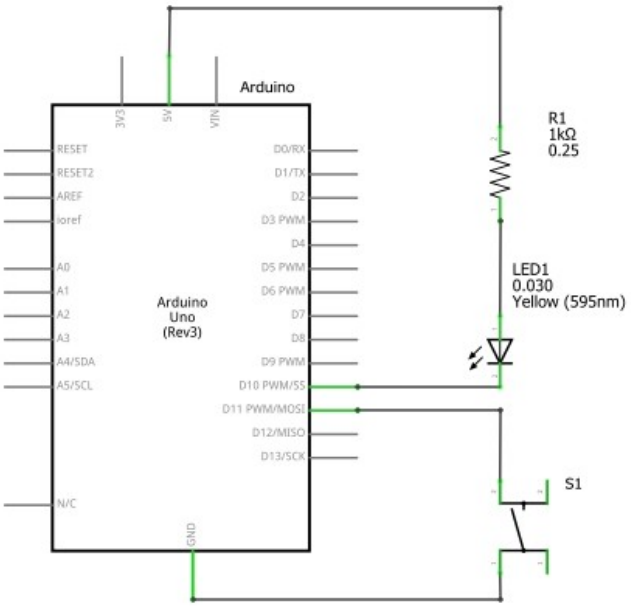
- På breadboard tilføjes:
- En knap der forbinder pin 11 til GND
- En lysdiode med formodstand på 1kΩ
  - Formodstand forbindes fra +5V på breadboard til lysdiodens anode (lange ben)
  - Lysdiodens katode (korte ben) forbindes til Arduino pin 10.

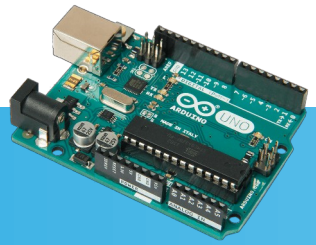




Diagram

- Her er diagrammet for vores kredsløb:

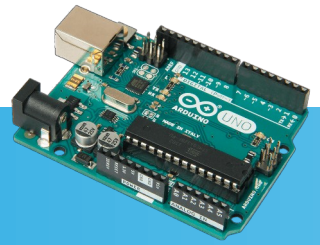




Knap til lysdiode

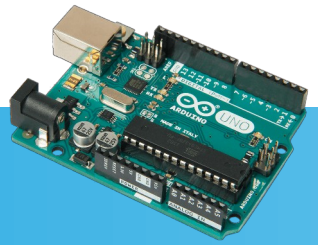
- Prøv at indtaste denne sketch:

```
1   const byte lysdiode_pin = 10;
2   const byte knap_input_pin = 11;
3
4   void setup() {
5     pinMode(lysdiode_pin, OUTPUT);
6     pinMode(knap_input_pin, INPUT_PULLUP);
7   }
8
9   void loop() {
10    if (digitalRead(knap_input_pin) == LOW) {
11      digitalWrite(lysdiode_pin, HIGH);
12    }
13  }
```



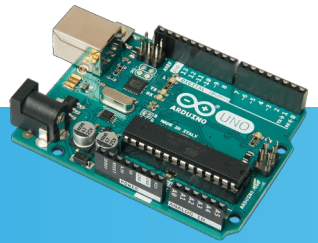
Forklaring

- Først defineres navne for de to benyttede pins
- I setup-funktionen initialiseres
  - lysdiode_pin som udgang (OUTPUT)
  - knap_input_pin som indgang med pull-up
- I loop-funktionen checkes hele tiden om knappen er aktiveret, og hvis den er sættes lysdiode_pin høj
- Knappens indgang bliver lav når man trykker
- Lysdiodens pin sættes høj for at slukke lyset



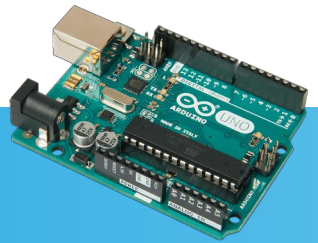
Prøv at forklare

- Hvad sker der når man trykker på knappen?
- Og hvad sker der når man slipper igen?



Opgave

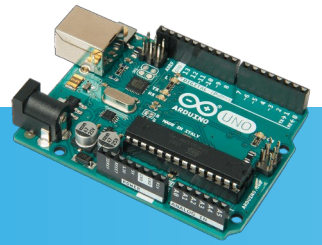
- Ret koden så lysdioden er slukket så længe knappen holdes nede, men tænder igen når der slippes



Løsningsforslag

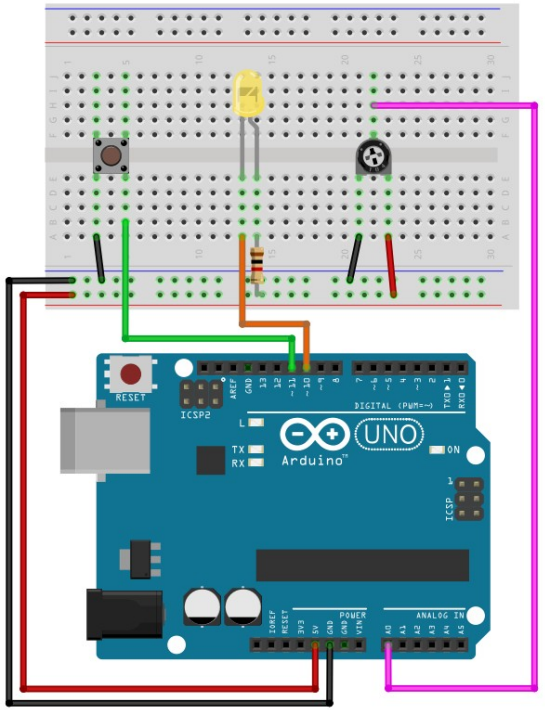
```
 9  void loop() {
10    if (digitalRead(knap_input_pin) == LOW) {
11      digitalWrite(lysdiode_pin, HIGH);
12    } else {
13      digitalWrite(lysdiode_pin, LOW);
14    }
15  }
```

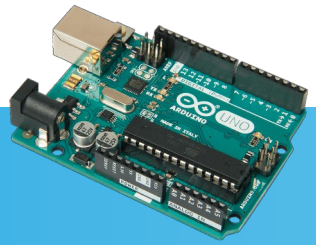
```
 9  void loop() {
10    digitalWrite(lysdiode_pin, !digitalRead(knap_input_pin));
11  }
```



Analog input

- Tag USB-kablet ud af Arduino mens kredsløbet ændres
- Tilføj et potentiometer (10K) til breadboardet
  - De yderste ben forbindes til GND og +5V
  - Det midterste ben forbindes til Arduino'ens analoge indgang A0
- Husk at tjekke forbindelserne inden USB-kablet sættes i (undgå kortslutning)

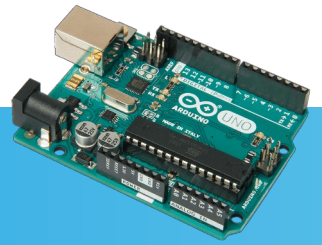




Læs analog værdi

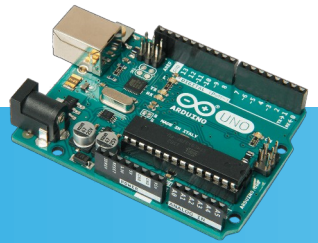
- Indtast nedenstående sketch

```
1   void setup() {
2     Serial.begin(9600);
3   }
4
5   int sensorValue;
6
7   void loop() {
8     sensorValue = analogRead(analogInPin);
9     Serial.println(sensorValue);
10  }
```



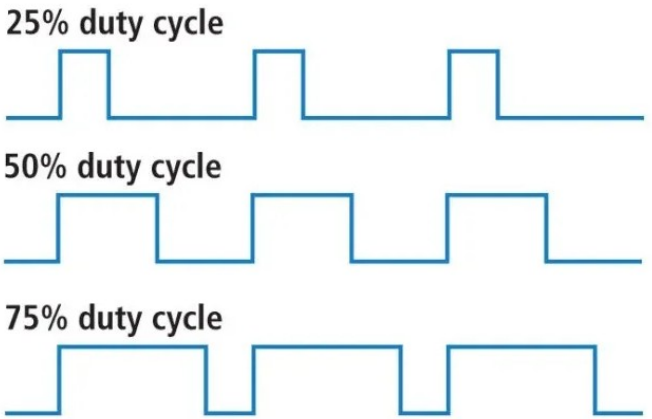
Afprøv

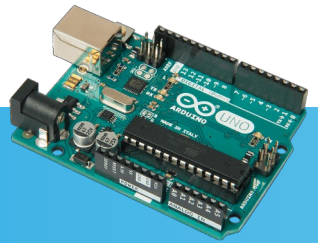
- Upload sketch’en til Arduino
- Åbn Serial Monitor
- Se hvad der sker når du drejer på potentiometeret



PWM

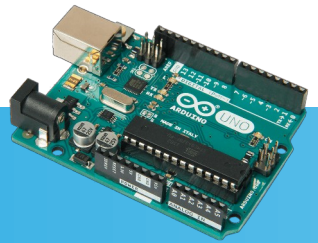
- Nogle udgange på Arduino kan bruges til PWM (Pulse Width Modulation), de er markeret med ~
- PWM betyder at man tænder og slukker meget hurtigt, og at man varierer hvor lang tid der er tændt og slukket
- Hvis vi bruger PWM på vores lysdiode kan vi skrue op og ned for den





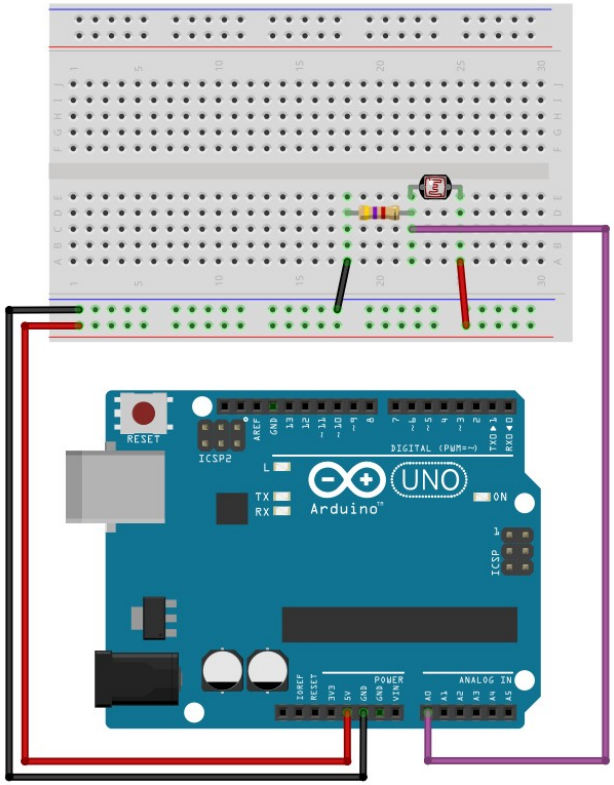
Eksempel AnalogWrite

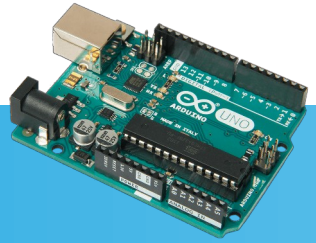
- Åbn eksemplet
  File → Examples → 03.Analog → AnalogInOutSerial
- Ret værdien af analogOutPin til 10 (på linje 25)
- Upload sketch'en til Arduino
- Se hvad der sker når du drejer på potentiometeret
- Åbn serial monitor og se output
  - Læg mærke til at jo højere værdien er, des svagere lyser lysdioden. Hvorfor?



Lysmåler

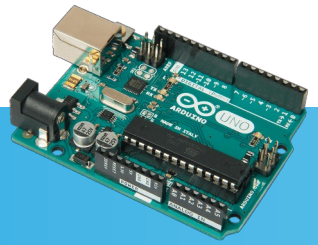
- Afbryd USB-kablet mens der ændres på kredsløbet
- Udskift potentiometeret med en lysfølsom modstand (LDR) og en fast modstand på 4,7kΩ
- Tilslut USB-kablet igen
- Åbn serial monitor, og se hvordan værdien ændrer sig når der er lys eller skygge på LDR'en





Ekstraopgaver

- Lav en sketch, der
  - Venter på at knappen er trykket ned
  - Tænder lysdioden
  - Venter på at knappen er sluppet
  - Venter på at knappen er trykket ned
  - Slukker lysdioden
  - Gentager
- OBS: Der bliver sikkert brug for et lille delay() p.g.a. kontaktprel



Stopur

- Funktionen millis() returnerer antal millisekunder siden sketchen er startet.
- Udvid sketch'en
  - Tilføj Serial.begin(9600); i setup-funktionen
  - Når lysdioden tændes gemmes tiden (millis) i en variabel
  - Når lysdioden slukkes gemmes tiden i en anden variabel
  - Udskriv forskellen på serielkonsollen, så har du et stopur
  - Kan udvides med at vise tiden i minutter:sekunder:millisekunder