Arduino-eksperiment	130140	*Stikord*	ADC, float, skalering, funktionsværdi	
Version	2018-05-22 / HS	*Niveau*	Grundkursus, modul 7	*p.* 1/4

Det lærer du:

- Brug den indbyggede analog-digital converter (ADC) til at måle spændinger
- Skriv en funktion, som returnerer et tal
- Regn med reelle tal (med decimaler)
- Skalér et talinterval efter behov

- Forstå ADC opløsning (bits)
- Brug et potentiometer som variabel spændingsdeler
- Brug spændingsdelere til målinger

1 – Potentiometer på breadboard

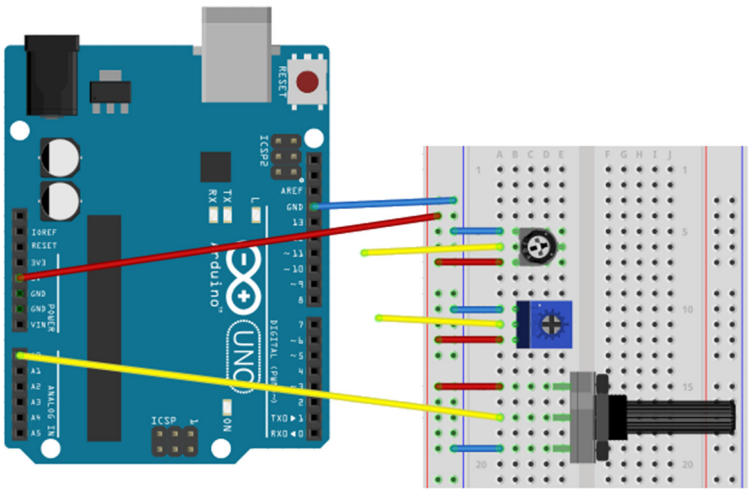
Forskellige typer

Potentiometre kommer i mange forskellige udgaver (se billedet), men fælles for dem alle er de tre ben og en del som kan drejes – enten med en knap eller ved hjælp af værktøj. (Der findes faktisk også skydepotentiometre, hvor den bevægelige del ikke drejes, men flyttes lineært.)

For de fire typer på billedet gælder, at hvis du anbringer potentiometeret symmetrisk foran dig, så er det midterste ben forbundet til den drejelige del. Hvis de andre to ben forbindes til hhv. 0 V og 5 V, vil spændingen på midterbenet variere jævnt mellem 0 og 5 V, når potentiometeret drejes.

Placering på breadboard

Billedet herunder viser *tre* forskellige slags potentiometre anbragt på boardet – du skal kun bruge ***ét***. De to lange baner til venstre er koblet til hhv. 5 V og 0 V på Arduinoen. Midterbenet på potentiometeret forbindes til ben A0.



2 – Analog input

Første test

Arduinoen har 6 ben, som kan bruges til at måle spændinger: A0 til A5. Benets nummer (uden "A") anvendes i funktionen `analogRead()`, som "aflæser" spændingen på benet.

Det følgende lille program vil hele tiden kalde `analogRead()` og udskrive resultatet i monitorvinduet på PC'en.

Tast ind, og prøv af ved at dreje potentiometeret fra den ene yderposition til den anden:

```
const byte pinIn = 0;            // Arduino ben A0

void setup() {
  Serial.begin(57600);
}

void loop() {
  Serial.println( analogRead(pinIn) );
}
```

Analog-Digital Converter (ADC)

Som du så lige før, præsenteres resultaterne som en række hele tal. Med et almindeligt godt potentiometer skulle tallene gerne kunne varieres fra 0 til 1023.

Funktionen `analogRead()` returnerer de "rå" resultater fra den *analog-digital converter* (ACD), som er indbygget i Arduinoen. ADC'en har en opløsning på 10 bit, eftersom $1023 = 2^{10} - 1$. Bemærk, at disse tal ikke kan rummes i en byte (talværdier fra 0 til 255), du er nødt til at bruge datatypen `int` (evt. `word`, `long` eller `unsigned long`).

Der findes en ganske handy funktion `map()`, som kan omsætte værdier i ét tal-interval til et andet. Antag, at du f.eks. ønsker potentiometerets stilling i % mellem 0 og 100.

Følgende lille programstump klarer denne omsætning, så `y` f.eks. får værdien 50, hvis `x` er 255:

```
int x, y;

x = analogRead(pinIn);
y = map(x, 0, 1023, 0, 100); // Parametre: input, input min, max, output min, max
```

`map()` er indbygget i systemet og kan bruges, som den er.

Udfordring 1

Omskriv programmet, så output bliver tal mellem 0 og 100.

Indsæt også et `delay()`-kald, så der kun skrives for hver tiendedel sekund.

Når det virker, så skriv programmet om igen, så det er spændingen i volt, der vises.

Du har formentlig bemærket, at `map()` åbenbart returnerer heltal. Man kan diskutere det fornuftige i at vise en spænding mellem 0 og 5 V uden decimaler…

Det viser sig, at omsætningsfunktionen er defineret som vist nederst på siden. Se også værktøjs-boks til højre.

Alle værdier – indgående parametre og det afleverede resultat – er altså af typen `long` – det vil sige heltal. Så det kan ikke undre, at der blev rundet af.

Hvis det skal bruges til praktiske formål, vil det være nyttigt at kunne regne med decimaltal.

Værktøj: Funktion, som returnerer en værdi

Eksempel:

Definér funktionen:

```
long triple(int x) {
  return 3*x;
  x = 42; // Dumt. Udføres ikke!
}
```

Kald funktionen:

```
Serial.println( triple(7) );
```

Denne linje giver udskriften: 21

Forklaring:

Funktionsværdiens datatype angives. (Her: `long`)

Udtrykket, som står efter ordet `return`, beregnes og returneres som funktionsværdi.

Funktionen *springer over* evt. ekstra kommandoer mellem `return` og den afsluttende krølleparentes.

```
long map(long x, long in_min, long in_max, long out_min, long out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

3 – Datatypen `float`

Arduinoen har netop én type til at gemme decimaltal i: `float`. Se værktøjs-boksen til højre.

Datatypen fylder 4 bytes på en Arduino Uno.
Præcisionen ligger på 6 til 7 betydende cifre.
Talområdet går fra $-3{,}4028235 \cdot 10^{38}$ til $3{,}4028235 \cdot 10^{38}$.

De sidste to egenskaber er ikke så imponerende. Men til mindre krævende opgaver er det OK.

Der eksisterer også en såkaldt dobbelt-præcisions type decimaltal, kaldet `double`. Den triste nyhed er, at på en Arduino Uno er det *det samme* som en `float`.

Andre typer microcontrollere, f.eks. Arduino Due, afsætter 8 bytes til en `double`. Såvel præcision som talområde forbedres væsentligt! – Men det kan vi ikke bruge hér.

Taler vi om at angive en spænding som "4,73 V", er præcisionen dog i hvert fald tilstrækkelig.

For at forbedre programmet fra *Udfordring 1*, skal du lave en ny funktion `mapf()`, som gør det samme som den oprindelige `map()`-funktion – men som bruger decimaltal overalt.

Værktøj: `float`

Eksempel:

```
float x = 37;
float y = 7;
float z = x/y;
Serial.println(z);
Serial.println(100*z);
Serial.println(10000*z);
Serial.println(1000000*z);
Serial.println(100000000*z);
```

Denne programstump udskriver følgende:

```
5.29
528.57
52857.14
5285714.00
528571424.00
```

Forklaring:

Der oprettes tre variabler til decimaltal. Efter den tredje linje bør `z` indeholde tallet

5,285714285714285714285714285143 osv.

men vi kan se tre ting af udskriften:

1. `float`-tal afrundes til 2 decimaler, når de udskrives med `Serial.println()`.
2. Arduino regner kun ca. 7 cifre rigtigt.
3. At gange med en ti-potens er ikke nogen eksakt operation, som det er i titalssystemet! (Sammenlign de to nederste linjer i resultatet.)

Udfordring 2

Omskriv programmet til at bruge en decimaltals-omsætning, så spændingen udlæses i volt med to decimaler.

Udskriften skal være med enhed (dvs. f.eks. "2,61 V" og ikke bare "2,61").

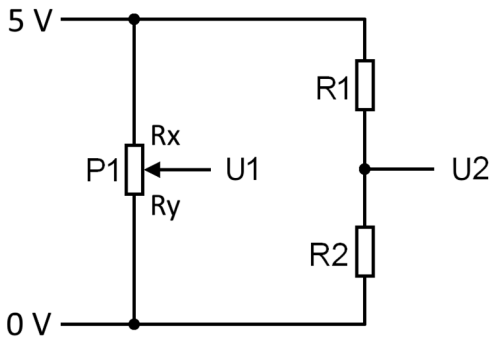
4 – Spændingsdelere

Et potentiometer, som forbindes ligesom i dette kursusmodul, er en speciel type af en *spændingsdeler*: To modstande i serieforbindelse, hvor man bruger spændingen i deres fælles punkt. Se diagramsymbolerne for begge dele herunder.

Potentiometeret (til venstre) er indsat mellem 0 V og 5 V, ligesom vi har gjort i praksis. Midterbenet er tegnet som en pil, som man kan forestille sig kan glide op og ned (svarende til, at der drejes på potentiometeret). Modstanden af den del, som ligger over glideren, har fået betegnelsen Rx og den resterende del kaldes Ry.

Tilsammen udgør de Rx + Ry, som netop er hele modstanden mellem de to yderben på potentiometeret. Man kan beregne (vha. Ohms lov), at spændingen U1 på midterbenet er givet ved ligningen

$$U1 = 5\ \text{V} \cdot \frac{Ry}{Rx + Ry}$$



Spændingsdeleren (til højre) er også forbundet mellem 0 V og 5 V. "Midterbenet" er her fast, men spændingen U2 er givet af et fuldstændigt tilsvarende udtryk:

$$U2 = 5\ \text{V} \cdot \frac{R2}{R1 + R2}$$

5 – Sensorer, som ændrer modstand

Der findes flere typer sensorer, som fungerer ved at de ændrer modstand, når de påvirkes udefra af f.eks. lys eller temperatur. Ved at lade sensoren indgå i en spændingsdeler vil modstandsændringerne omsættes til spændings-variationer, som kan måles med Arduinoen.

Som eksempel vil vi se på en lys-sensor af typen LDR (Light Dependent Resistor).
Se billedet – føleren er ca. 5 mm i diameter.

Som den anden modstand i spændingsdeleren vil vi bruge en 10 kΩ modstand. Værdien af modstande visses med farvekoder, hvor der er to næsten ens systemer i brug:

For modstande med 5 % tolerance angives værdien med 3 farvede ringe. 10 kΩ vises som **brun, sort, orange**. Tolerancen markeres med en guld-farvet fjerde ring.

For modstande med 1 % tolerance angives værdien med fire farvede ringe. 10 kΩ vises som **brun, sort, sort, rød**. Tolerancen angives med en brun femte ring.

I praksis kan du anvende et ohmmeter. Slut det til modstanden med krokodillenæb – hvis du holder den fast med fingrene, vil din egen modstand i hud og krop påvirke resultatet.

Modstande leveres ofte på tape som vist på billedet. Når du skal bruge en af modstandene, klippes benene over med en skævbider tæt på tapen (vist med en rød streg). Når du ved, hvor modstanden skal sidde, kan du evt. korte et eller begge ben af, så den ikke står og flagrer (og laver kortslutninger).

Klip. *Lad være* med at hive benene ud af tapen! Der vil altid sidde nogle lim-rester på komponentbenene, som ender i breadboardet. Det er umuligt at rense af, hvis det sker.

Klip modstandene af tapen:

Udfordring 3

Lav en lysmåler!

Der er ingen tegning at følge denne gang, men du kan jo tage udgangspunkt i den eksisterende opstilling med potentiometeret. Lad LDR-sensoren indgå som ”R2” (tættest på 0 V) i spændingsdeleren.

Mens opstillingen afprøves, kan du bruge samme program på Arduinoen som før.

Når du ved, at opstillingen fungerer, skal du ændre programmet, så der udskrives teksten ”Mørke”, når du skygger for LDR’en ved at sætte en finger på. Når der falder lys på LDR’en, skal teksten ”Lys” udskrives. Brug `if` til dette.

Gem programmet under et nyt navn og lav det om, så du får flere trin på skalaen: ”Skummelt”, ”Halvmørkt”, ”Gråvejr” osv. – eller hvad du nu finder på. Programmet skal udskrive disse betegnelser i rækkefølge, når du skygger mere eller mindre for sensoren.

Udfordring 4 (kræver en lux-sensor)

Hvis du råder over en lux-sensor, kan du prøve, om du ved hjælp af din `mapf()`-funktion og et par passende valgte kalibreringspunkter kan få din lyssensor til **nogenlunde** at vise lysintensiteten i lux – i et begrænset område.

(Et tip: Hvis du bytter om på ét par af ”max” og ”min” -værdier, når du kalder `mapf()` , vil omsætningsfunktionen blive aftagende, så mindre og mindre input giver større og større output.)

En LDR har en meget ulineær respons, så en simpel omsætningsfunktion vil kun have begrænset gyldighed.

Har du god tid – og er god til matematik – kan du prøve, om du kan lave en helt anden omsætningsfunktion, som rammer bedre i et større område.