Arduino-experiment	130140-EN	*Trefwoorden*	ADC, float, scaling, functiewaarde	
Versie	2018-07-09 / HS	*Niveau*	Basiscursus, module 7	*p.* 1/4

Wat je leert:

- Gebruik de ingebouwde analoog-digitaal conversie (ADC) voor spanningsmetingen
- Schrijf een functie die een waarde teruggeeft
- Berekenen met floating point (d.w.z. decimale) getallen
- Schaal een cijferinterval naar je behoeften

- Begrijp ADC-resolutie (bits)
- Gebruik een potentiometer als variabele spanningsdeler
- Gebruik spanningsdelers bij metingen

1 – Potentiometer op een breadboard

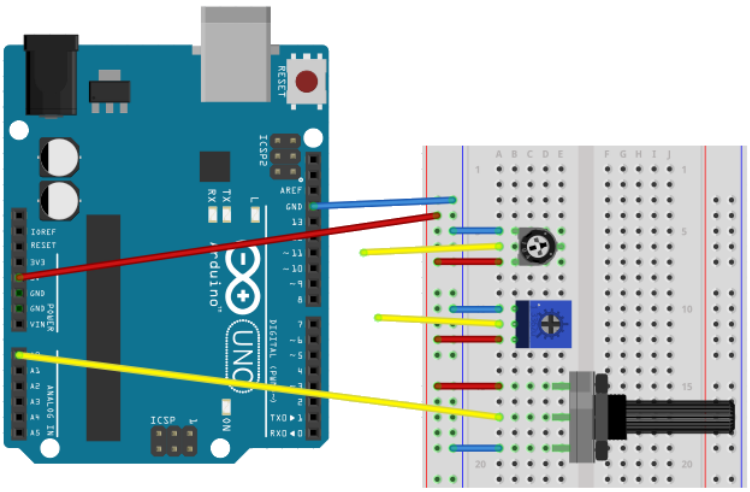
Verschillende soorten

Potentiometers zijn er in veel verschillende vormen (zie foto), maar ze hebben allemaal drie pinnen en een onderdeel dat kan draaien – met een knop of met een schroevendraaier. (Er zijn zelfs ook lineaire potentiometers, waarbij het bewegende deel glijdt in plaats van draait.)

Bij de vier types op de foto betreft het draaibare deel de pin in het midden wanneer je de potentiometer symmetrisch voor je plaatst. Als de twee andere pinnen zijn aangesloten op respectievelijk 0 V en 5 V, zal de spanning op de middelste pin soepel variëren van 0 tot 5 V wanneer de as van de potentiometer wordt gedraaid.

De potentiometer op een breadboard plaatsen

De afbeelding hieronder toont *drie* verschillende potentiometers die op de plank zijn geplaatst – je hebt er slechts ***één*** nodig. De lange rijen gaten links zijn verbonden met respectievelijk 5 V of 0 V op de Arduino. De middelste pin van de potentiometer is verbonden met pin A0.



2 – Analoge invoer

Eerste test

De Arduino heeft 6 pinnen die kunnen worden gebruikt voor spanningsmetingen: A0 tot A5. Dan wordt het pinnummer (zonder de "A") gebruikt in de functie `analogRead()` die de spanning op de pin "afleest".

Het volgende programma zal voortdurend `analogRead()` aanroepen en het resultaat tonen in het monitorvenster op de PC.
Voer het programma in en test het door de potentiometer van de ene uiterste positie naar de andere te draaien:

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

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

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

Analoog naar Digitaal Conversie (ADC)

Zoals je ziet worden de resultaten getoond als een rijtje integers. Met een gewone potentiometer moeten deze getallen variëren tussen 0 en 1023.

De `analogRead()` functie geeft de "ruwe" resultaten van de in de Arduino ingebouwde analoog naar digitaal converter (ADC) terug. De ADC heeft een resolutie van 10 bits: $1023 = 2^{10} - 1$. Merk op dat deze nummers niet in één byte passen (nummers van 0 tot 255) – je moet het gegevenstype `int` (of `word`, `long` of `unsigned long`) gebruiken.

Er is een vrij handige functie, `map()`, die getallen in het ene interval omzet naar het andere. Stel dat je de rotatie van de potentiometer als percentage tussen 0 en 100 wilt hebben.
Het volgende stukje code zorgt hiervoor; een `x`-waarde van 512 geeft `y` de waarde 50:

```
int x, y;

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

`map()` is een ingebouwde functie.

Uitdaging 1

Herschrijf het programma zodat de uitvoergetallen tussen 0 en 100 worden.

Voeg een `delay()` toe om de uitvoersnelheid te verminderen tot eenmaal per tiende van een seconde.

Als dit goed werkt, herschrijf dan het programma nog een keer om de uitgang de spanning in volt te laten weergeven.

Toolbox: Functie met een uitgangswaarde

Voorbeeld:

Definieer de functie:

```
long triple(int x) {
  return 3*x;
  x = 42; // Silly. Never executes!
}
```

Roep de volgende functie aan:

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

Deze regel zal de volgende output opleveren: 21

Uitleg:

Het gegevenstype van de functiewaarde wordt gespecificeerd (hier: `long`).

De uitdrukking na het trefwoord `return` wordt berekend en als functiewaarde terruggegeven.

De functie *negeert* mogelijke extra instructies tussen de `return` en de afsluitende accolade **}**.

Het is je vast opgevallen dat `map()` een integer lijkt terug te geven. Het is echter nauwelijks zinvol om spanningen tussen 0 en 5 V zonder decimalen weer te geven …

Het blijkt dat `map()` is gedefinieerd zoals hieronder weergegeven. – Zie ook de toolbox aan de rechterkant.

Alle waarden – zowel de inkomende parameters als het resultaat – blijken van het type `long` te zijn – het zijn dus integers.
Geen wonder dat de decimalen werden afgerond…

Het zou mooi zijn om deze conversies met decimale getallen te kunnen maken.

```
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 – Het gegevenstype `float`

Arduino heeft slechts één type voor het opslaan van decimale getallen: `float`. Zie de toolbox rechts.

Dit gegevenstype gebruikt 4 bytes op een Arduino Uno. De nauwkeurigheid is ongeveer 6 tot 7 significante cijfers. Het bereik gaat van $-3.4028235 \cdot 10^{38}$ tot $3.4028235 \cdot 10^{38}$.

De laatste twee eigenschappen zijn niet indrukwekkend. Maar voor minder veeleisende taken is het nog wel OK.

Er is ook een zogenaamd decimaal getal met dubbele nauwkeurigheid dat `double` wordt genoemd. Het trieste nieuws is dat dit op een Arduino Uno precies *hetzelfde* is als een `float`.

Andere soorten microcontrollers, zoals de Arduino Due, gebruiken 8 bytes voor een `double`. Zowel de nauwkeurigheid als het bereik zijn enorm verbeterd! – Maar dit heeft in onze situatie geen zin.

Als we een spanning willen opgeven als ”4.73 V”, dan is de nauwkeurigheid in ieder geval voldoende.

Om het programma van *Uitdaging 1* te verbeteren, moet je een nieuwe functie schrijven, `mapf()`, die dezelfde taak uitvoert als de originele `map()` functie – maar dan overal met behulp van decimale getallen.

Toolbox: `float`

Voorbeeld:

```
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);
```

Dit stukje code produceert de volgende output:

```
5.29
528.57
52857.14
5285714.00
528571424.00
```

Uitleg:

Er worden drie decimale getalsvariabelen aangemaakt. Na de 3e regel moet `z` de volgende waarde hebben:

5.285714285714285714285714285143 etc.

maar we kunnen drie dingen leren van de uitvoer:

1. `float` getallen worden afgerond op twee decimalen wanneer ze worden getoond met `Serial.println()`.
2. Arduino krijgt maar ongeveer 7 cijfers goed.
3. Vermenigvuldiging met een macht van 10 is geen exacte bewerking zoals in het decimale systeem! (Vergelijk de twee laatste regels in de uitvoer.)

Uitdaging 2

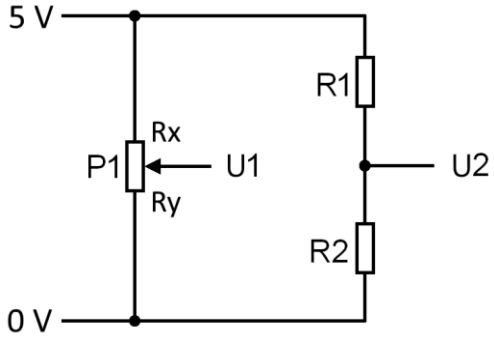
Herschrijf het programma om decimale getallen te gebruiken, zodat de spanning met twee decimalen kan worden weergegeven.

De resultaten moeten samen met de eenheid worden gepresenteerd (dus bijvoorbeeld ”2.61 V” en niet alleen ”2.61”).

4 – Spanningsverdelers

Een potentiometer, aangesloten zoals hierboven beschreven, is een speciaal soort *spanningsdeler*: twee weerstanden in serieschakeling waarbij je de spanning verkrijgt uit hun gemeenschappelijk punt. Zie hieronder de symbolen voor in schema's.

De potentiometer (links afgebeeld) is, net als in de praktische opstelling, tussen 0 V en 5 V aangesloten. De middelste pin is getekend als een pijl die je kunt zien als “op en neer glijdend” (wanneer de as draait). De weerstand van het deel boven de pijl wordt Rx genoemd en het resterende deel heet Ry.



Samen vormen ze Rx + Ry , wat simpelweg de weerstand is tussen de twee buitenste pinnen van de potentiometer. Met de wet van Ohm kun je zo de spanning U1 op de middelste pin berekenen:

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

De spanningsdeler (rechts afgebeeld) is ook aangesloten tussen 0 V en 5 V. De “centre pin” staat hier vast, maar de spanning U2 is op precies dezelfde manier te berekenen:

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

5 – Sensoren die de weerstand veranderen

Een aantal soorten sensoren zorgen voor veranderende weerstand onder invloed van bepaalde externe effecten, bijvoorbeeld licht of temperatuur. Door de sensor deel uit te laten maken van een spanningsdeler worden de weerstandsveranderingen omgezet in spanningsvariaties die door de Arduino kunnen worden gemeten.

Als voorbeeld bekijken we een lichtsensor van het type LDR (Light Dependent Resistor). Zie de afbeelding – de sensor heeft een diameter van ongeveer 5 mm.

Als de andere weerstand in de spanningsdeler zullen we een weerstand van 10 kΩ gebruiken. De weerstand wordt weergegeven met een kleurcode, waarvoor twee vrijwel identieke systemen worden gebruikt:

Voor weerstanden met een tolerantie van 5 % wordt de waarde gegeven door drie gekleurde ringen. 10 kΩ wordt weergegeven als **bruin, zwart, oranje**. De tolerantie wordt weergegeven als een goudkleurige vierde ring.

Voor weerstanden met een tolerantie van 1 % wordt de waarde gegeven door vier gekleurde ringen. 10 kΩ wordt weergegeven als **bruin, zwart, zwart, rood**. De tolerantie wordt weergegeven als een bruine vijfde ring.

In de praktijk kun je een ohmmeter gebruiken. Sluit hem aan op de weerstand met krokodillenklemmen – als je hem met je handen vasthoudt, zullen je eigen huid en lichaamsweerstand de meting beïnvloeden.

Weerstanden worden vaak op tape geleverd, zoals te zien op de foto. Wanneer je een van de weerstanden nodig hebt kun je simpelweg de draden doorknippen met een diagonaalschaar dicht bij de tape (aangegeven met de rode lijnen). Als je weet waar de weerstand geplaatst moet worden kan het zijn dat je één of beide draden nog inkort om het risico op kortsluiting te vermijden.

Knip. Trek *niet* alleen het lood uit de tape! Er zullen altijd wat lijmresten zijn die in de breadboard terecht kunnen komen. Als dit eenmaal is gebeurd is dit onmogelijk weer schoon te krijgen.

Knip de weerstanden van de tape af:

Uitdaging 3

Bouw een lichtmeter!

Je krijgt deze keer geen tekening, je kunt namelijk de bestaande opstelling met de potentiometer gebruiken. Geef de LDR-sensor de rol van “R2” (het dichtst bij 0 V) in de spanningsdeler.

Voor het testen van de opstelling kun je hetzelfde programma op de Arduino gebruiken als voorheen.

Zodra je weet dat de opstelling werkt, pas je het programma aan zodat het de tekst “Dark” weergeeft wanneer je de LDR bedekt door er een vinger op te leggen. Wanneer er licht op de LDR valt, moet de tekst “Light” worden weergegeven. Gebruik hier `if` voor.

Sla het programma op met een nieuwe naam en maak verdere aanpassingen om meer stappen op de schaal mogelijk te maken: “Murky”, “Dim”, “Cloudy” etc. – of wat voor namen je ook maar kan bedenken. Het programma moet deze termen in de juiste volgorde weergeven wanneer je de sensor meer of minder schaduw geeft.

Uitdaging 4 (vereist een lichtsensor)

Als je toegang hebt tot een lichtsensor, kun je proberen of je, door middel van je `mapf()` functie en een paar geschikte calibratiepunten, je lichtsensor **ongeveer** de intensiteit in lux kan laten weergeven – binnen een beperkt bereik.

(Tip: Als je *één* paar “max” en “min” waarden in de `mapf()` functie verwisselt, wordt de conversiefunctie verkleinend, d.w.z. steeds kleinere invoer geeft een steeds grotere uitvoer.)

Een LDR heeft een zeer niet-lineaire respons, dus een eenvoudige conversiefunctie heeft slechts een beperkte geldigheid.

Als je tijd genoeg hebt – en goed bent in wiskunde – kun je proberen een totaal andere conversiefunctie te maken die beter werkt over een groter bereik.