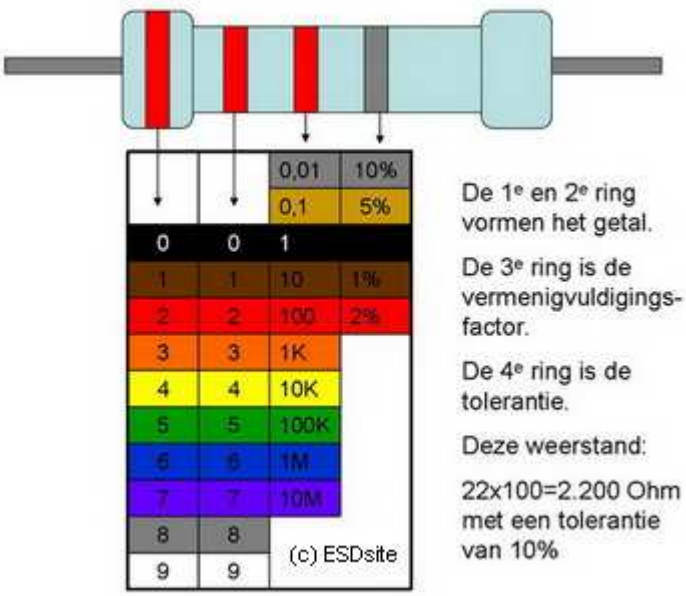
rd

Sessie 4 : “bar-graph” met Arduino.

1. In deze opstelling gaan we terug een ander item van Arduino gebruiken, namelijk een potentiometer, of regelbare weerstand. We kennen allemaal de koolweerstand (deze die normaal gebruiken). Daar is ook nog de draad weerstand (gewikkelde weerstandsdraad- wordt gebruikt voor grote vermogen schakeling). De waarde van een weestand wordt aangegeven met een kleurencode (althans bij de koolweerstand) bij de draad weerstand gaat de waarde er meestal opgedrukt staan.
2. De kleurencode wordt als volgt bepaald:



De kleuren code zelf gaat als volgt. (samen met een ‘ezelsbrugje’ om het gemakkelijk te onthouden

Zij	= Zwart	= 0
Bracht	= Bruin	= 1
Rozen	= Rood	= 2
Op	= Oranje	= 3
Gerrits	= Geel	= 4
Graf	= Groen	= 5
Bij	= Blauw	= 6
Vies	= Violet	= 7
Grauw	= Grijs	= 8
Weer	= Wit	= 9

Een voorbeeld daarvan is onze bekende weerstand va 220 ohm. Of rood-rood-bruin.

3. Er vierde ring of tolerantie ring zijn normaal zilver of goud. Zilver is 10% tolerantie en goud 5% tolerantie. Er bestaan echter ook weerstanden mat nog betere tolerantie maar deze hebben 6 ringen ipv 4. Waarbij de 4de ring de vermenigvuldigingsring is en dan 5 ring is de tolerantie ring en de 6de (iets breder) is dze dat de afhankelijkheid van de temperatuur aanduidt. Maar aangezien we dat zelden of niet tegenkomen in onze oefeningen ga ik daar niet verder op in.
4. Dan hebben we de variabele weerstand soms ook potentiometer genoemd. Daarin bestaan er ook verschillende soorten volgens het gebruik. Deze die we best kennen is de draai

rd

potentiometer. of ook de schuif potentiometer. Dit zijn gelijken en die werken als volgt. De weerstand wordt gemaakt door een koollaag van een bepaalde weerstand waarde vb 50 KOhm. Door het draaien aan de metalen uitstekend element schuift een glijder van 0 (begin) tot 50KOhm in een vloeiende beweging, steeds andere waarden afgevend. De totale waarde (eindwaarde) staat normaal gedrukt op de potentiometer. Zijn 'klein broertje, de trim-potentiometer heeft de zelfde werking. Echter deze wordt meestal ingesteld op een vaste waarde en blijft zo. De instelling daarvan gebeurt dan ook met een schroevendraaier welke de glijder op een vaste plaats zet.

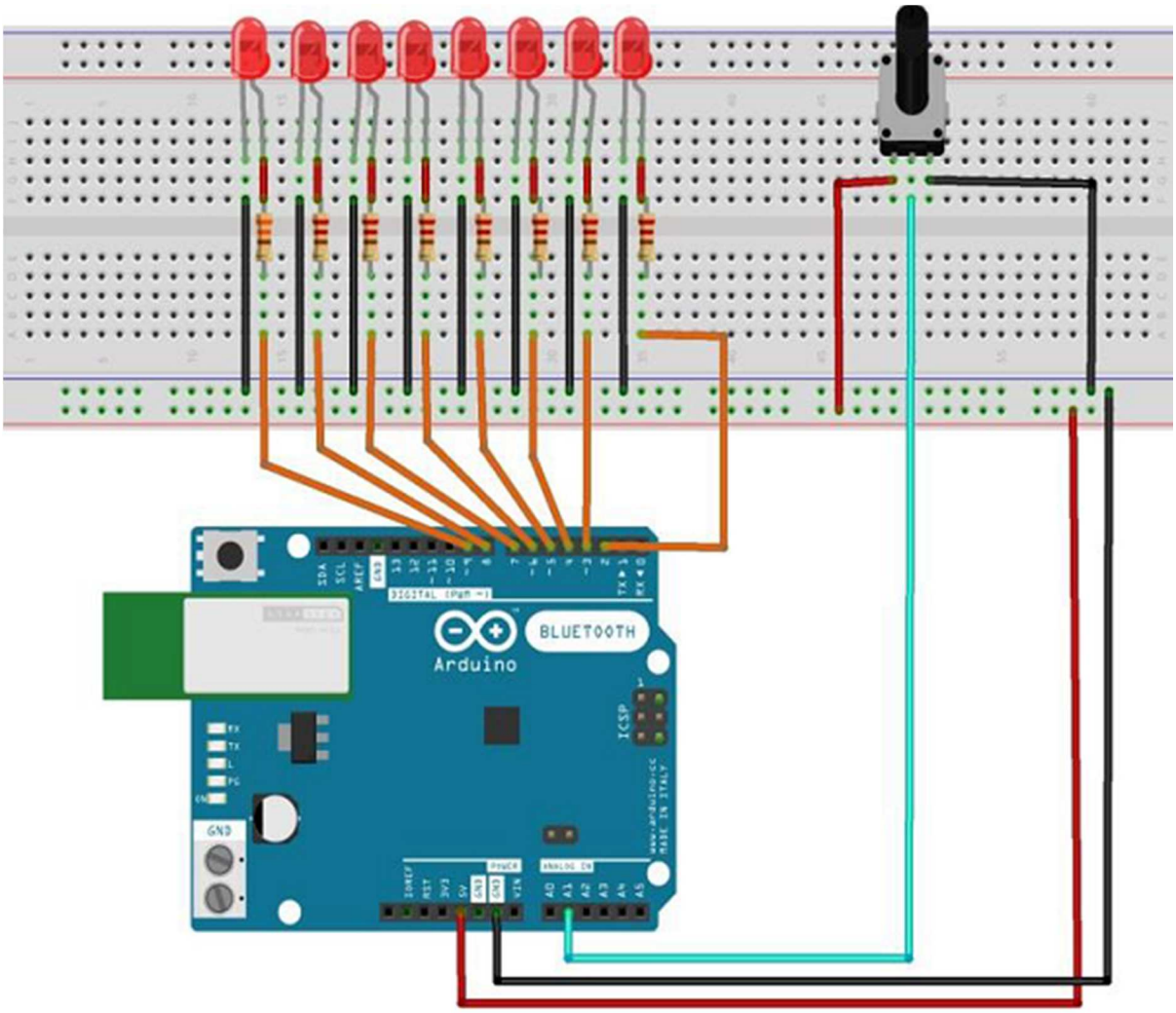
5. De schematische voor stelling is:

Rheostat Symbol

De waarde kunnen we best afmeten met een voldoende precieze Ohm meter.

6. Na deze uitleg over het wat en hoe van een weerstand, komen we terug op onze Bar-graph. Het idee is dat we een aantal LED achtereenvolgens laten branden, door verschuiven van een potentiometer.

rd



7. Code

```
const int analogPin = A0;
const int ledCount = 8;
int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9}     //tabel nieuw
void setup(){
   for(thisLed = 0; thieLed < ledPins; thisLed++){
      pinMode( ledPins[thisLed], OUTPUT);
   }
}
void loop(){
  int sensorReading = analogRead(analogPin);
  int ledLevel = map(sensorReading, 0, 1023, 0, ledCount)
  for(int thisLed = 0; thisLed < ledCount; thisLed++){
     if(thisLed < ledLevel){
        digitalWrite(ledPins[thisLed], HIGH);
     ) else{
        digitalWrite(ledPins[thisLed], LOW);
     }
   ]
}
```

8. Bovenstaande code zou moeten gaan voor het aansturen van onze graph. In deze code echter komen we een paar nieuwe dingen tegen die ik zal trachten uit te leggen.

rd

8.1 Eerst komen we terug de afkorting **const** tegen. Zoals vorige keer uit gelegd is dit hoe men een waarde aangeeft die steeds dezelfde waarde houd in het onderdeel waarin deze gedeclareerd wordt. Let op ze in **publiek** gedeclareerd, dus geldig voor gans het programma.

8.2 Het volgende dat we tegen komen is een **tabel**, beter gekent in het programmeren als een **array.** Dit is een systeem om data op te slaan op een georganiseerde wijze en welke de mogelijkheid bied om deze data altijd op te roepen. Een tabel bestaat , zoals in de wiskunde, uit een naam en een index dar kennen we uit het bekende programma Excel. De data wordt uit deze tabel gehaald door de naam van de tabel aan te duiden, samen met de plaats van de data. Dit word mogelijk gemaakt door gebruik van de index. Een voorbeeld. Stel we hebben een tabel met als data 0 , 1, 2 en met naam getal. In dit geval zal men het cijfer '2' uit de tabel halen door de naam van de tabel samen met de index (plaats in de tabel)in deze tabel) te geven. Dus tabel[3].

8.3 Ook een commando welke we nog niet tegengekomen zijn is de beperkte lus of **de for lus.** De tegenstelling met de continous loop of doorlopende lus. Het verschil tussen beide is de wijze hoe dze in het programmeren benaderd wordt. Elk van de lussen zal werken met een variabele welke de mogelijkheid biedt om de lus actie op een welbepaald moment te stoppen.

8.4 de formule of methode van de beperkte lus zit hem erin on de stuur variabele een beperkte en vastgestelde levensduur te geven. Ik snap dat dit op chinees of andere taal lijkt maar is in feite zeer simpel:

- in eerste instantie zullen we een variabele declareren (benoemen). Voor het gemak noemen we deze 'x'. let wel kleine of grote letter heeft geen belang zolang u bij zijn vorm blijft voor gans zijn gebruik. Gezien dit een getal is weten we van voeger dat we dit de prefix int moeten geven. Dus **int x.** Moeten we die x een begin waarde vb. 0 toekennen. In principe hier niet. In sommige toepassingen is dat wel noodzakelijk (verder meer).
- De magische formule is nu:
  for (x=0; x niet gelijk is aan een controle waarde; x++){……}
  for is de aanroep van de functie (loop). **Steeds in kleine letters!** Open de ronde haakjes (denk aan het woord functie).
  Dan zullen we de variabele x gebruiken. U ziet dat hij hier direct een begin waarde krijgt, het was daarom dat het niet nodig is om hen voor dien een waarde te geven. Dit wordt afgesloten met een punt-komma. Dan gaan we deze variabele vergelijken met een voordien ingegeven waarde. Dat is een van de verschillen met de ander lussen, tot dat lus. (while) en tot wanner lus(until). Dit zijn normaal doorlopende lussen. Maar later meer daarover als we die nodig hebben. Nu kunnen we ons behelpen met de for-lus.
  De vergelijking methoden zijn in hoofzaak **binair** dat wil zeggen de uitslag is steeds 1 (true) of 0 (false). De methoden, en we kennen ze uit de wiskunde zijn groter dan (>) kleiner dan (<) groter dan of gelijk aan (>=)kleine dan of gelijk aan (<=) of gelijk aan (==). En daar zit een adder onder het gras. Bij gelijk aan kennen we het gelijkheid teken uit de wiskunde. In informatica, althans in de 3de graadstalen (C,VB,Delphi…enz) gebruik man de dubbel gelijkheids-teken als we willen kijken als iets gelijk is aan. Maar als we een waarde aan iets willen toekennen zullen we het enkele gelijkheid teken gebruiken. (Vb x = 0, maar 3 == 3)
  Nadien hebben we nog dta iedere maal dat de lus doorlopen is de variabele 1 (of meerdere) waarde moet verhogen. Daarvoor kunnen we x = x+1 gebruiken. Maar

hoe men dat schrijft is verschillend. Schrijft men x ++ is het zelfde. Later zal dat wel duidelijker worden.
Nadien sluiten we de ronde haakjes. Natuurlijk tijdens die lus (loop) functie moet er iets gebruren. Dat zetten we zoals we reeds vroeger gedaan hebben tussen accolades ({}) en sluite nadien het geheel af met punt-komma. Ik weet in het begin is dat wel een beetje verwarrend maar dat komt wel!.

9. Dat was een ganse boterham om te veteren. Maar in de loopp van de oefeningen zal u daar mee spelen en dan is dat maar klein grut met het geen we later nog zullen tegen komen!
10. Met deze 4-de sessie zijn we al een redelijk eind geschoven in de lessen reeks. Ik hoop dat u de moed nog niet opgeeft want leukere dingen volgen.