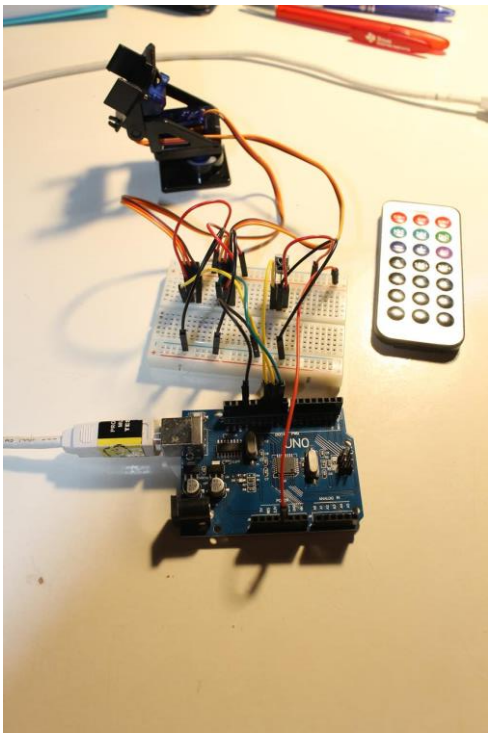
# Arduino

# voor gevorderden

# Deel 1



**Versie: 4.0**

**Datum: 13/01/2021**

**Auteur: Frank Marchal**

**Doelgroep: 14 tot 88 jaar (na basiskennis Arduino)**

# Inleiding

Dit document heeft tot doel de cursist zover te krijgen dat hij de **Arduino UNO** omgeving verder kan inzetten voor moeilijker of leukere projecten.

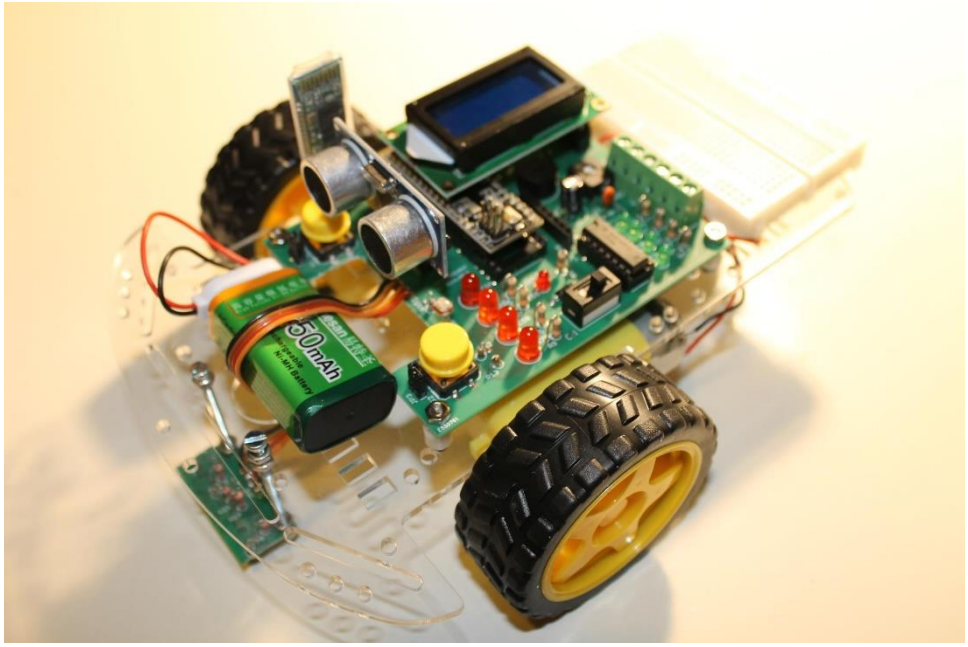
In de basiscursus werd er met **basiselektronica (LED, switch, buzzer, motor zonder/met PWM)** gewerkt en gingen we zo de programmeer omgeving leren ontdekken en uitzoeken. Daarna zochten we uit hoe we met een Arduino UNO bordje allerlei sensoren zoals een **afstandssensor** konden aansturen. Ook leerden we de robot met **bluetooth** en **lijnvolgers** besturen.

Je kan m.a.w. pas beginnen aan deze cursus als je **eerst de basis van Arduino onder de knie** hebt. We gebruiken nog Ardublock indien het kan of handig is.

In de gevorderden cursus deel 1 gaan we nu nog meer elektronica aansturen. Zo hebben we het o.a. over de **analoge inputs, LCD schermen, I2C, stappenmotoren, servo motoren, allerlei sensoren enz...**

Info hierover kan je ook vinden op [www.arduino.cc](http://www.arduino.cc)

Veel succes !



Voorbeeld van een Arduino robot met vaste PCB. Het hart van de robot is een Arduino Micro. Op het breadboard kunnen allerlei extra componenten toegevoegd worden.

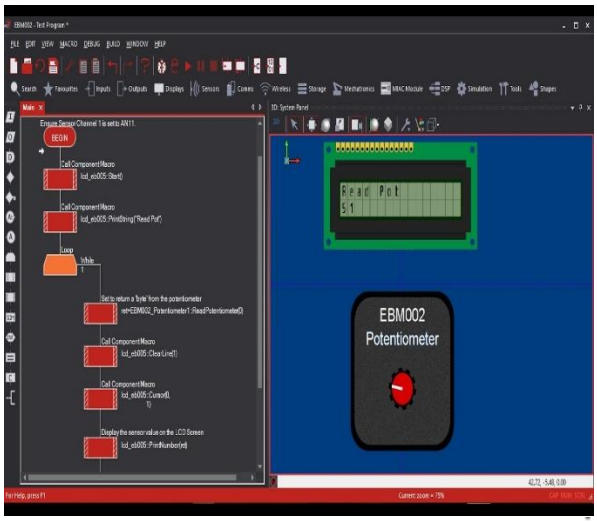
# Welke Arduino omgeving gebruiken?

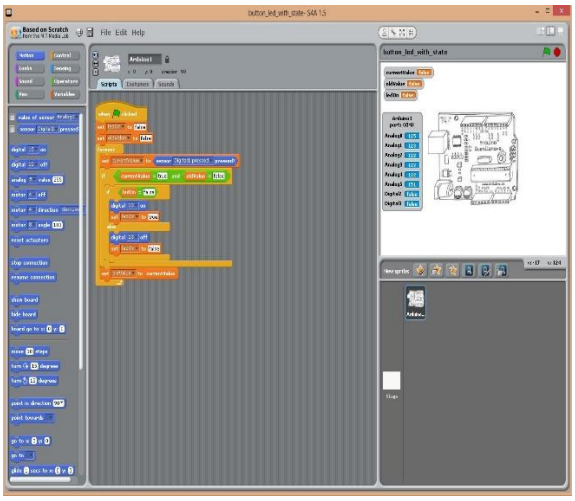
Welke interessante Arduino omgevingen bestaan er tegenwoordig?

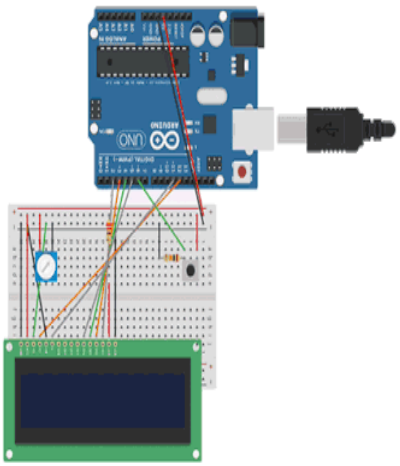
- **S4A** (Scratch for Arduino): (gratis) visuele omgeving, handig zolang je aangesloten blijft op het Arduino bordje, geeft waarden direct terug, beperkt in inputs en outputs, ideaal voor kleine beginners.
- **Ardublock**: (gratis) visuele omgeving dewelke omgezet wordt in Arduino IDE code. Handig maar als het moeilijker wordt (veel code) niet meer zo overzichtelijk. Je kan zelf blokken bijmaken.
- **Flowcode 8:** (30 dagen gratis, dan licentie), interessante visuele debugger, handige omzetting naar c-code in zelfde omgeving. Zeker goed voor beginners.
- **Tinkercad**: (gratis) handig dat je eerst jouw breadboard schakeling opbouwt in simulatie. Daarna kan je deze testen door visuele blokjes te slepen op een scratch manier en alles om te zetten in Arduino IDE. Beperkt in aantal blokjes. Goed voor beginners.
- Afgeleiden van Scratch: (gratis) (zie **M-bots**, enz ….)
- **Arduino IDE**: (gratis) eenvoudige pseudo c-code omgeving waar je libraries kan aan toevoegen en de basis van c-code programmeren toch wel onder de knie kan mee krijgen. Simulatie niet mogelijk, wel info via serial monitor ontvangen. Veel documentatie op internet.
- **Programino**: Arduino IDE in een ander jasje.
- **Fritzing**….
- **UnoArduSim**

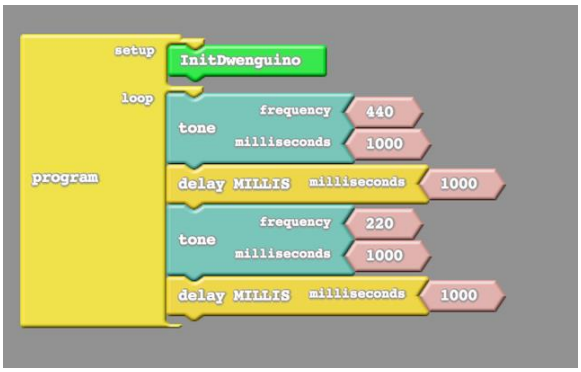
Dus programmeeromgevingen genoeg, voor ieder wat wils 😊

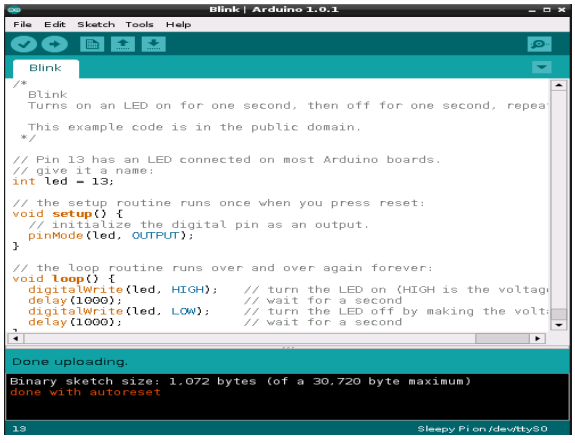
We gebruiken in deze cursus vooral **Arduino IDE**. Soms kan **Ardublock** handig zijn om programma's te vereenvoudigen. De andere omgevingen zijn zeker handig voor de beginners of als je moeilijke dingen wil debuggen. Soms duiken ook deze op in de cursus.

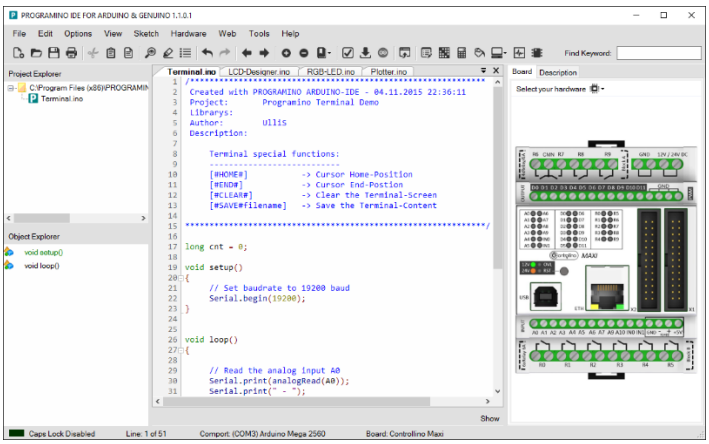






Allerlei Arduino programmeeromgevingen.

# 1. RBG leds leren aansturen: maak je eigen kleur

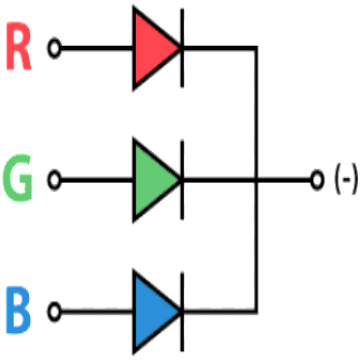
In dit hoofdstuk leren we hoe we met 1 RGB (rood – groen – blauw) LED **onze eigen kleur kunnen bepalen**. Zo kan je bijvoorbeeld bij de les fysica het kleurenspectrum laten onderzoeken. Anderzijds kan je ook jouw projecten verfraaien met verschillende kleuren. Soms zie je ook in ontwerpen dat dezelfde LED meerdere betekenissen kan hebben, afhankelijk van zijn kleur.

## 1.1 Opbouw RGB LED

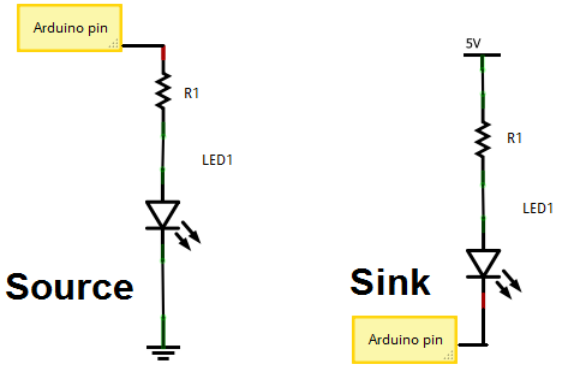
RGB LEDs kunnen op 2 manieren opgebouwd zijn: ofwel zijn de 3 kathodes met elkaar verbonden van de 3 kleuren LEDs en noemen we dit de **common kathode**. Dan stuur je aan met positieve spanningen.

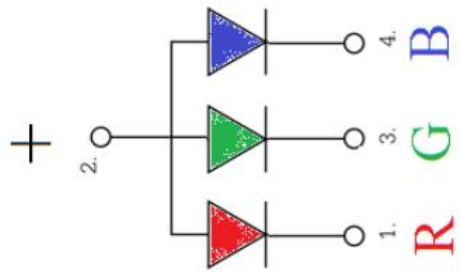
In dit geval gebruik je de **“source” techniek** (de microcontroller is de bron van de energie).

Anderszijds kunnen ook de 3 anodes samenhangen aan de common anode draad. Nu moet je 1 van de kathode pinnen naar massa trekken om de LED te laten branden. Nu werkt de microcontroller als **“sink”** (afvoer van energie).


Common kathode RGB LED




Common Anode RGB LED

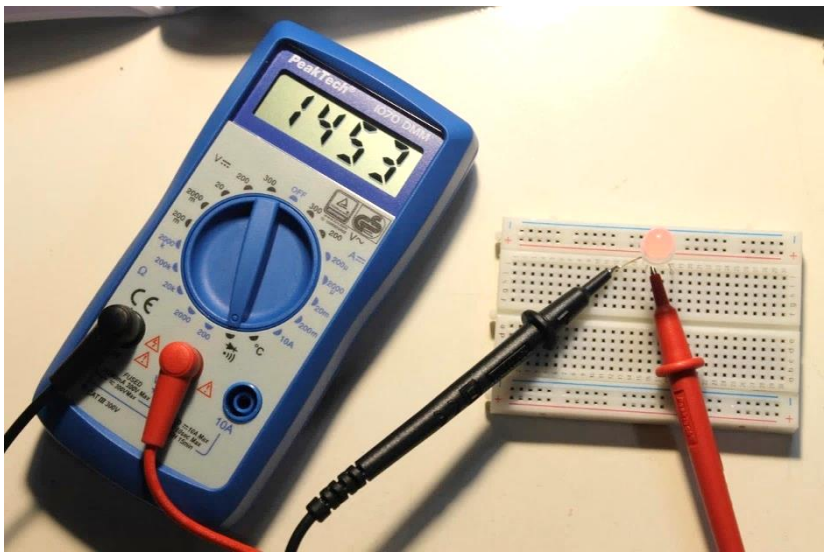
Kijk eerst goed na waar de platte kant van de LED zit. Daar zit de rode LED aangesloten. De rest kan je dan terugvinden in het aansluitschema.

➔ Hoe weet je nu of je een **common anode of kathode RGB LED** hebt?

Neem een **DMM** (digitale multimeter) en zet deze op **"diode"** stand. Wanneer je nu meet op de LED (rode DMM aansluiting op de anode en zwarte aansluiting op de kathode) moet deze gaan branden.

Experimenteer totdat je alle led aansluitingen en kleuren hebt getest.

In **ons voorbeeld** hebben we te maken met een **common anode RGB LED**.




➔ Mag ik een RGB LED rechtstreeks aansluiten op de +5V ?

Natuurlijk niet. Dit omdat de verschillende kleuren slechts werken op enkele volts en de stroom moet beperkt worden.

Met welke stroom moet ik rekening houden?

Laten we de **datasheet** eens bekijken van de RGB LEDs (als je deze al vindt van een Chinese LED).

Absolute Maximum Rating

| Item | Symbol | Absolute Maximum Rating | Unit |
|---|---|---|---|
| Forward Current | $I_F$ | 20 | mA |
| Peak Forward Current* | $I_{FP}$ | 100 | mA |
| Reverse Voltage | $V_R$ | 5 | V |
| Reverse Current | $I_R$ | 10 | uA |
| Power Dissipation | $P_D$ | 150 | mW |
| Electrostatic discharge | $E_{SD}$ | 800 | V |
| Operation Temperature | $T_{OPR}$ | -25～+80 | ℃ |
| Storage Temperature | $T_{STG}$ | -5～+45 | ℃ |
| Lead Soldering Temperature | $T_{SOL}$ | Max.260℃ for 5sec Max(3mm from the base of the epoxy bulb) | |

Typical Optical/Electrical Characteristics

| Product Type | Lens Color | Emitting Color | Forward Voltage(v) | | Prcp Wavelength (nm) | | Luminous Intensity(mcd) | | 50% Power Angle(deg) |
|---|---|---|---|---|---|---|---|---|---|
| | | | Type | Max. | Type. | Max. | Min. | Type | Type |
| 5TSRGB-A/C | Water Clear | R | 1.80 | 2.40 | 630 | 640 | 3000 | 4000 | 18 |
| | | G | 3.20 | 3.60 | 515 | 525 | 4000 | 6000 | |
| | | B | 3.20 | 3.60 | 460 | 470 | 900 | 1000 | |
| 5WSRGB-A/C | Diffused | R | 1.80 | 2.40 | 630 | 640 | 2000 | 3000 | 45 |
| | | G | 3.20 | 3.60 | 515 | 525 | 3000 | 5000 | |
| | | B | 3.20 | 3.60 | 460 | 470 | 700 | 900 | |

**Max 20mA** per LED. **Groen en blauw** kunnen we **dezelfde spanning** geven. **Rood** heeft **minder spanning nodig** en dus moet de voorschakelweerstand groter zijn.

➔ Kan een Arduino uitgang **20mA** aan of moet ik een elektronische schakelaar voorzien?

Op de Arduino UNO zit de **ATmega328P**. We kijken even in de datasheet pg 364.

### 32.1 Absolute Maximum Ratings

**Table 32-1. Absolute Maximum Ratings**

| | |
|---|---|
| Operating Temperature | -55°C to +125°C |
| Storage Temperature | -65°C to +150°C |
| Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground | -0.5V to $V_{CC}$+0.5V |
| Voltage on $\overline{\text{RESET}}$ with respect to Ground | -0.5V to +13.0V |
| Maximum Operating Voltage | 6.0V |
| DC Current per I/O Pin | 40.0mA |
| DC Current $V_{CC}$ and GND Pins | 200.0mA |

De chip kan **max 40mA** per pin aan. Dus we kunnen aan de slag. Let wel op, de ganse chip kan slechts 200mA aan.

Dit klopt ook volgens de [Arduino.cc website](Arduino.cc website).

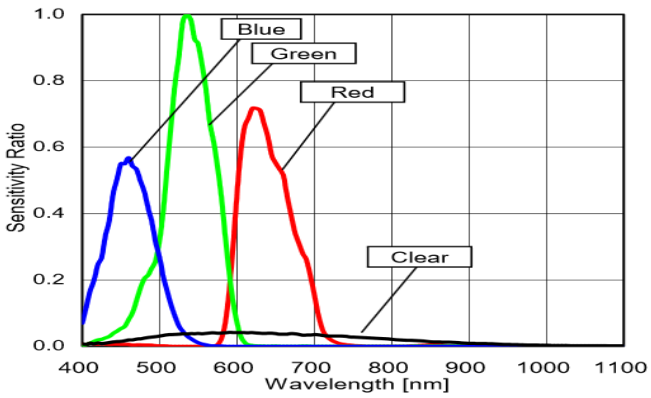
STORE SUPPORT

| | |
|---|---|
| Microcontroller | ATmega328P |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P) of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |

➔ Voorschakelweerstanden **berekenen** voor de LEDs:

Groen en blauw: R = 5V – 3.2V / 20mA = 90 ohm (**100 ohm** is een bestaande waarde)

Rood: R = 5V – 1.8V / 20mA = 160 ohm (**150 ohm** is een bestaande waarde)




Deze figuur vertelt ons iets over de golflengte van de verschillende kleuren.

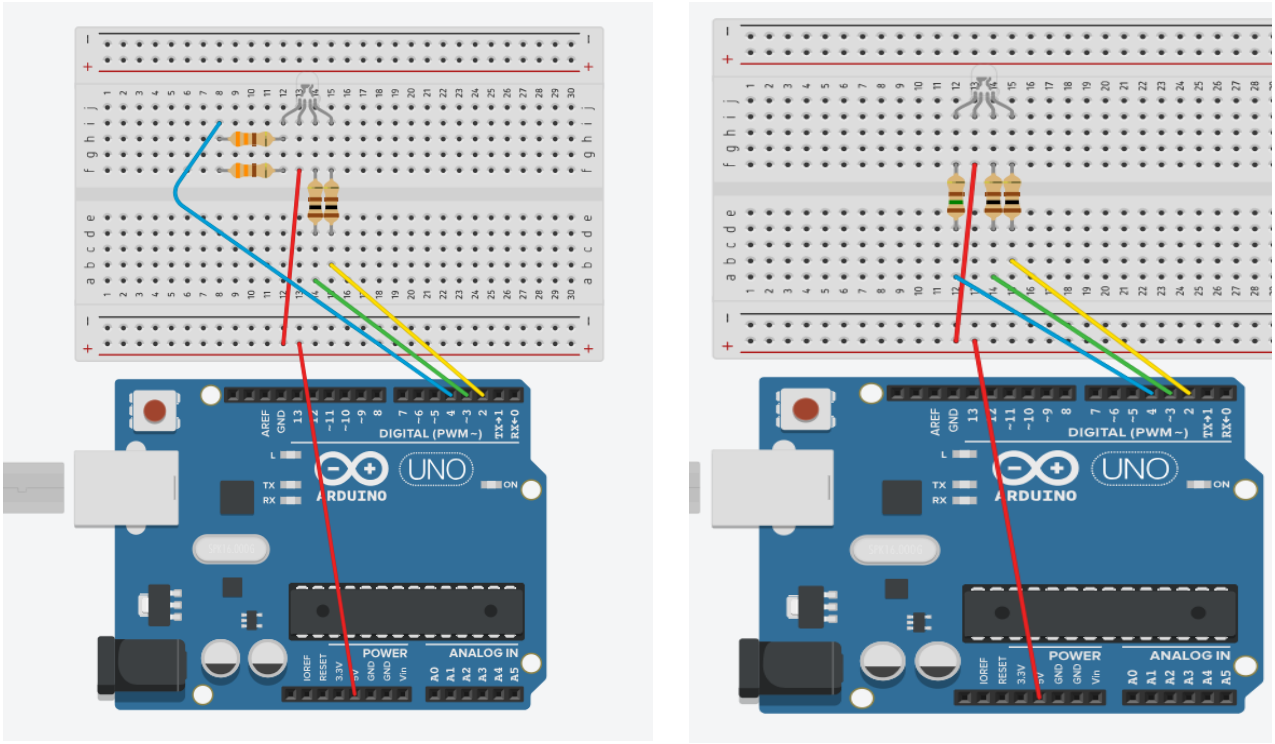
Zie ook de datasheet voor meer info.

## 1.2 RGB LED met CA aansluiten op het breadboard.

Zoek de weerstanden van **100 ohm** (bruin – zwart – bruin) en **150 ohm** (bruin – groen – bruin).

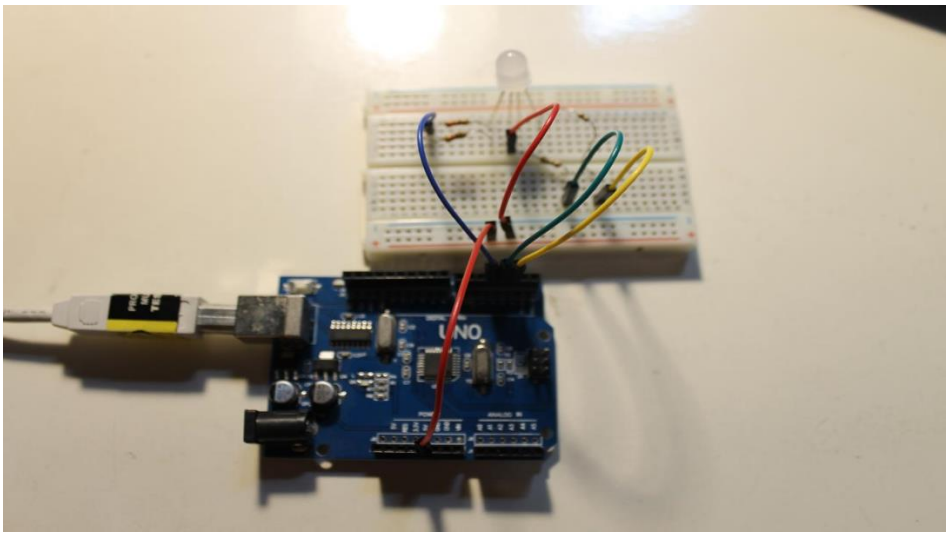
Merk op dat je 150 ohm ook kan maken door **2 x 330 ohm (oranje-oranje-bruin) in parallel** te zetten:

(R = 330 // 330 = 330 / 2 = 165 ohm). Dan zitten we ook kort bij de waarde die we zoeken.



2 mogelijke oplossingen om de RGB led met CA aan te sturen.

Let erop dat je de juiste kleurencodes gebruikt. Dat zorgt achteraf voor makkelijker debuggen van fouten. Blauw zit op pin 2, groen op pin 3 en rood op pin 4. De CA zit aan de +5V.



## 1.3 De RGB LED in Arduino code aansturen

We beginnen met het om de beurt aanzetten van de verschillende hoofdkleuren.

RGBLED_CA

```
//definiëer pinnummers

int blauw = 2;
int groen = 3;
int rood = 4;
int d = 1000; // delay in ms

void setup() {

  pinMode(blauw,OUTPUT);
  pinMode(groen,OUTPUT);
  pinMode(rood,OUTPUT);

 //zet alle uitgangen aan in het begin (common anode)
  digitalWrite(blauw,1);
  digitalWrite(groen,1);
  digitalWrite(rood,1);
}

void loop() {

// herhaal basiskleuren
// inverterend denken, CA dus enkel de gewenste kleur laag maken!
 digitalWrite(blauw, 0);
 delay(d);
 digitalWrite(blauw,1);
 digitalWrite(groen, 0);
 delay(d);
 digitalWrite(groen,1);
 digitalWrite(rood, 0);
 delay(d);
 digitalWrite(rood,1);

}
```

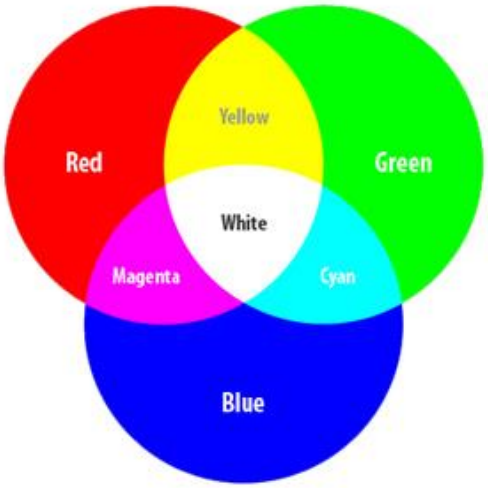
Merk op dat we **inverterend** moeten denken!

Bij de common anode RGB LED moet je enkel de uitgang laag maken van de kleur die je wenst.

Merk ook op dat we ervan uitgaan dat je weet hoe je deze code in Arduino IDE kan schrijven en hoe je deze kan downloaden in de UNO. Weet je iets niet meer, ga dan even terug kijken in de basiscursus van Arduino.

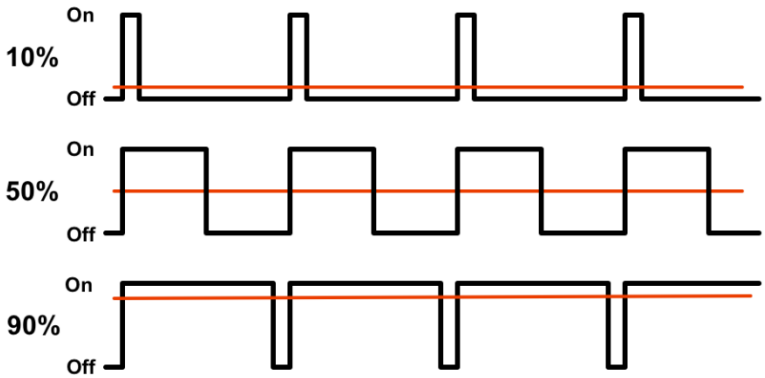
➔ **Uitdaging:**

- Pas de code aan zodat je de 7 kleuren kan mengen volgens volgend lichtspectrum.



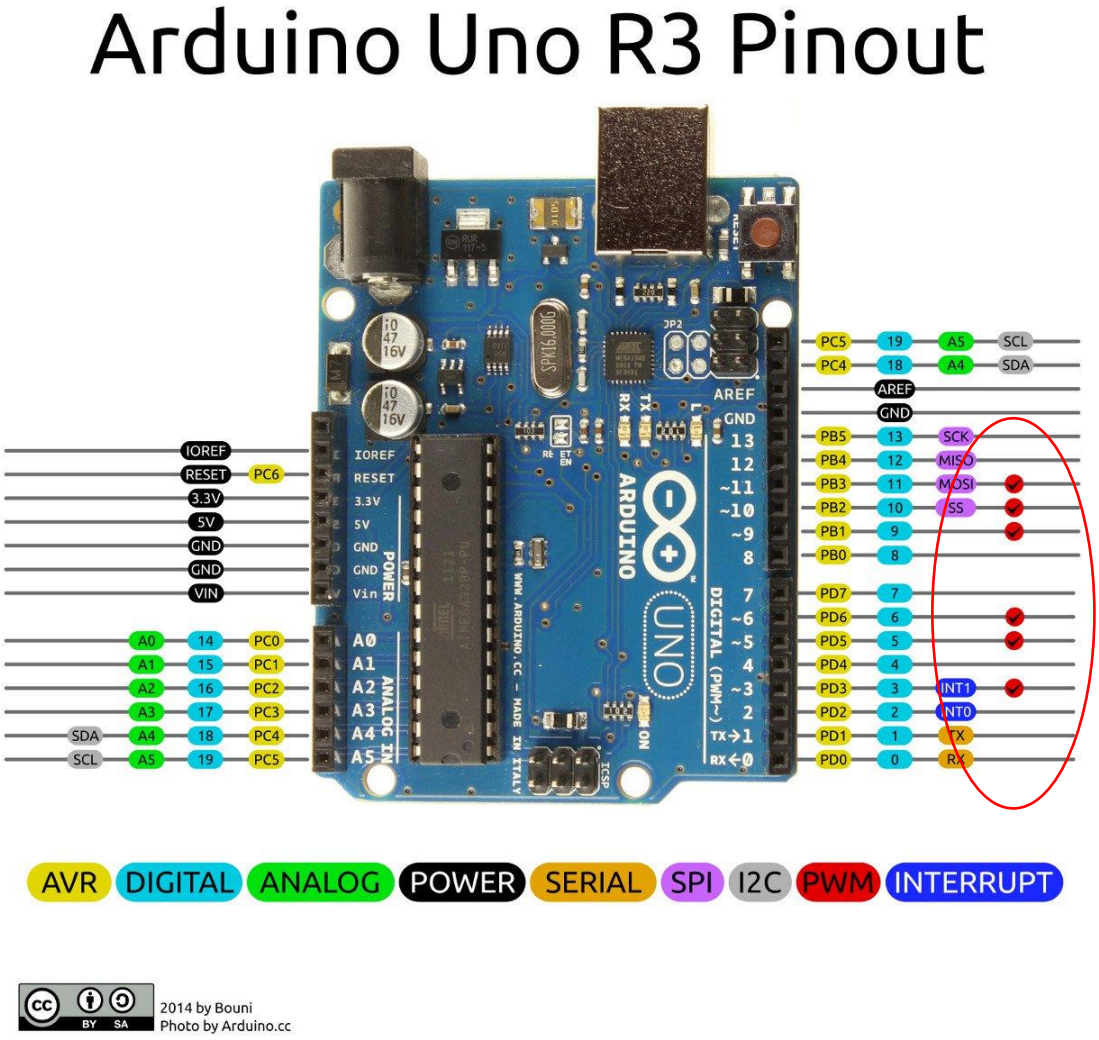

## 1.4 Met PWM de RGB kleuren dimmen

Weet je nog wat PWM betekent? We kunnen met **Pulse Width Modulation** de pin uitgang van een analoge waarde voorzien die we kunnen regelen tussen 0 en 255.




Hoe groter we de PWM waarde maken, **hoe breder de puls** wordt en **hoe groter de gemiddelde uitgangswaarde** wordt. Zo kunnen we ons LED licht dimmen.

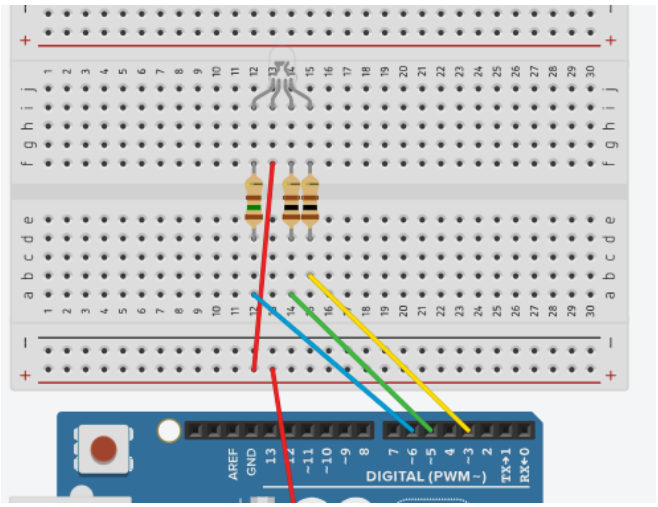
De hardware blijft bijna hetzelfde. We moeten er nu enkel voor zorgen dat we de kleuren aansluiten **op PWM uitgangen**.




De PWM uitgangen zijn de rode pinnen of pin 3, 5, 6, 9, 10 en 11.

Als voorbeeld kiezen we **blauw op pin 3, groen op pin 5 en rood op pin 6**.

Pas de bedrading aan op het breadboard.



De code voor te dimmen kan dan als volgt er uit zien:

RGBLED_CA_PWM

```
//definiëer pinnummers

int blauw = 3;//pwm pinnen
int groen = 5;
int rood = 6;
int d = 10; // delay in ms
int x; //teller maken

void setup() {

  pinMode(blauw,OUTPUT);
  pinMode(groen,OUTPUT);
  pinMode(rood,OUTPUT);

 //zet alle uitgangen op maximum in het begin (common anode)
  analogWrite(blauw,255);
  analogWrite(groen,255);
  analogWrite(rood,255);
}

void loop() {

// herhaal basiskleuren
// inverterend denken, CA dus enkel de gewenste kleur laag maken!

 for (x=255; x > 0; x--){  //lus om van 255 naar 0 af te tellen
  analogWrite(blauw, x); //pwm commando analogWrite ipv digitalWrite !
  delay(d);
 }
 for (x=0; x < 255; x++){  //lus om van 0 naar 255 op te tellen
  analogWrite(blauw, x);
  delay(d);
 }

 for (x=255; x > 0; x--){
  analogWrite(groen, x);
  delay(d);
 }
 for (x=0; x < 255; x++){
  analogWrite(groen, x);
  delay(d);
 }
 for (x=255; x > 0; x--){
  analogWrite(rood, x);
  delay(d);
 }
 for (x=0; x < 255; x++){
  analogWrite(rood, x);
  delay(d);
 }


}
```

Merk op dat je **analogWrite** gebruikt i.p.v. digitalWrite voor het activeren van de PWM uitgangen.

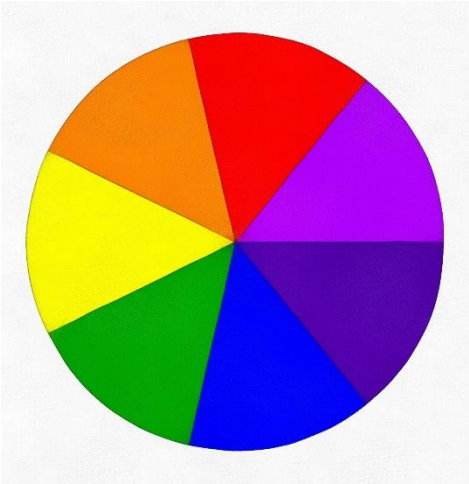
X++ en x—zijn verkorte c-code commando's voor x=x+1 en x= x – 1

De for lus dient ervoor om een commando een aantal keren te laten doorlopen. In dit geval zetten we eerst x één keer op een bepaalde waarde, dan kijken we of de actuele x groter of kleiner is dan de grenswaarde. Tot slot verkleinen of vergroten we de x waarde met 1. Dan wordt het commando tussen de haakjes uitgevoerd en wordt opnieuw de x vergeleken met de grenswaarde. Dit gaat zo door tot de vergelijking klopt. Dan springen we uit de lus.

Merk ook op dat we weer inverterend moeten denken om de common anode juist aan te sturen.

➔ **Uitdaging:**

- Maak met deze code de **schijf van Newton**. Zie oplossing in Arduino Natuurkunde boekje van Elektor pg 84. Rara welk effect dit heeft?



(we laten in feite via PWM 1 voor 1 de kleuren overgaan in elkaar: R aan, G aan, R uit, B aan, G uit, R aan, B uit, R uit, ….. De snelheid regel je met een delay in elke for lus.

- Regel de sterkte van elke kleur met een potmeter (zie deel 5).

http://www.leejoo.nl/generatoren/online_kleuren_generator.htm

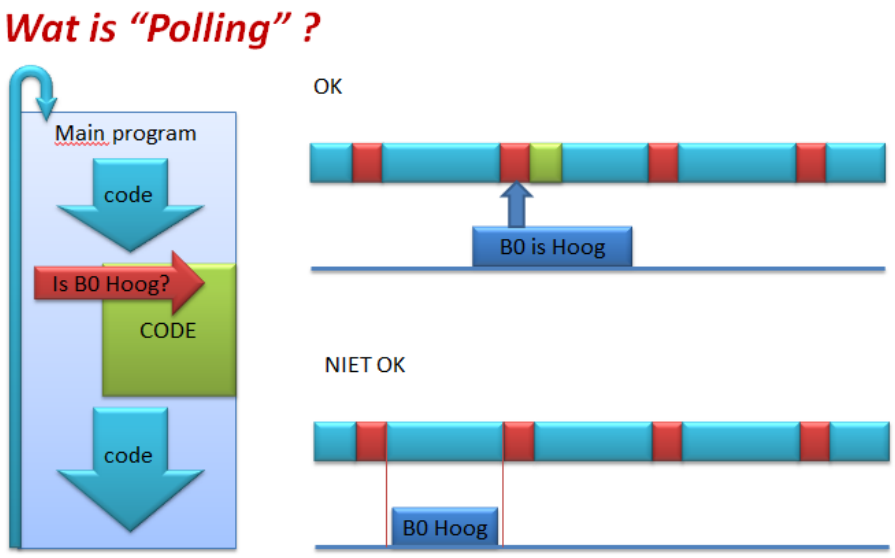
## 1.5 Hoe de dimmer pas laten werken als een knop is ingedrukt (interrupt)?

➔ Wat is een interrupt?

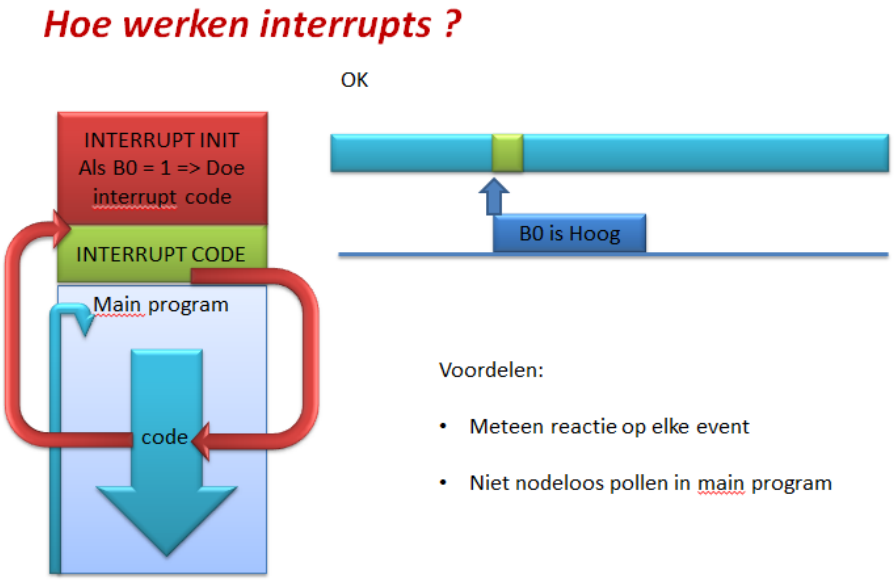
Je kan een knop op 2 verschillende manieren checken of deze is ingedrukt. Ofwel is het de “polling” methode ofwel is het via een “interrupt”.

Bij de **polling methode** ga je **af en toe eens kijken** of de knop is ingedrukt. Dit is perfect voor trage processen waarbij de tijd minder belangrijk is. De kans bestaat dat je bij deze methode al eens het indrukken van een knop hebt gemist en dat je de volgende keer via de microcontroller dit wel zal zien. Zo hebben we tot nu toe de knoppen en andere sensoren uitgelezen.

Als het indrukken van de knop dan toch is ontdekt, dan wordt de code uitgevoerd.




Bij de **interrupt** methode ga je **onmiddellijk,** nadat de knop is ingedrukt, de code uitvoeren. Er bestaat geen kans op een misser. Zeker handig bij snelle acties: als je bijvoorbeeld flesjes aan het tellen bent op een band.

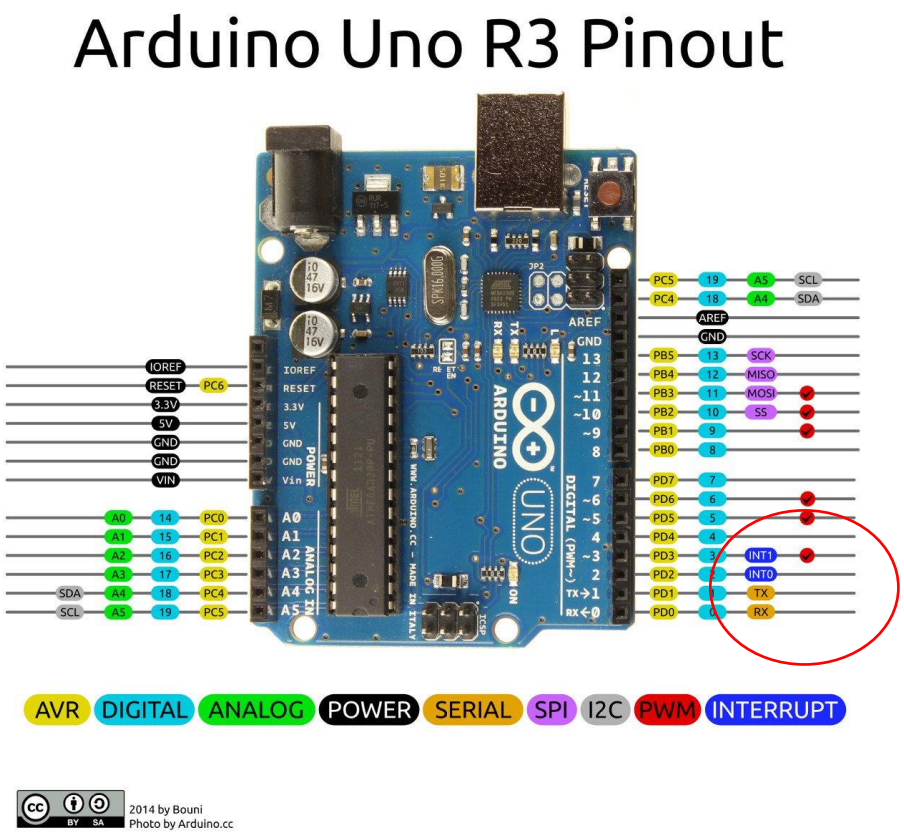


➔ Hoe een interrupt in Arduino maken?

Via het commando **attachInterrupt()** Kunnen we er voor zorgen dat we reageren met de microcontroller op een interrupt. Eens het interrupt is geweest wordt de extra code uitgevoerd (hier het dimmen van de LED).

Merk op dat **interrupts niet op alle pinnen** van de microcontroller voorkomen.

Bij de UNO kunnen we enkel op **pin 2 (INT0) en 3 (INT1)** hiervan gebruik maken.



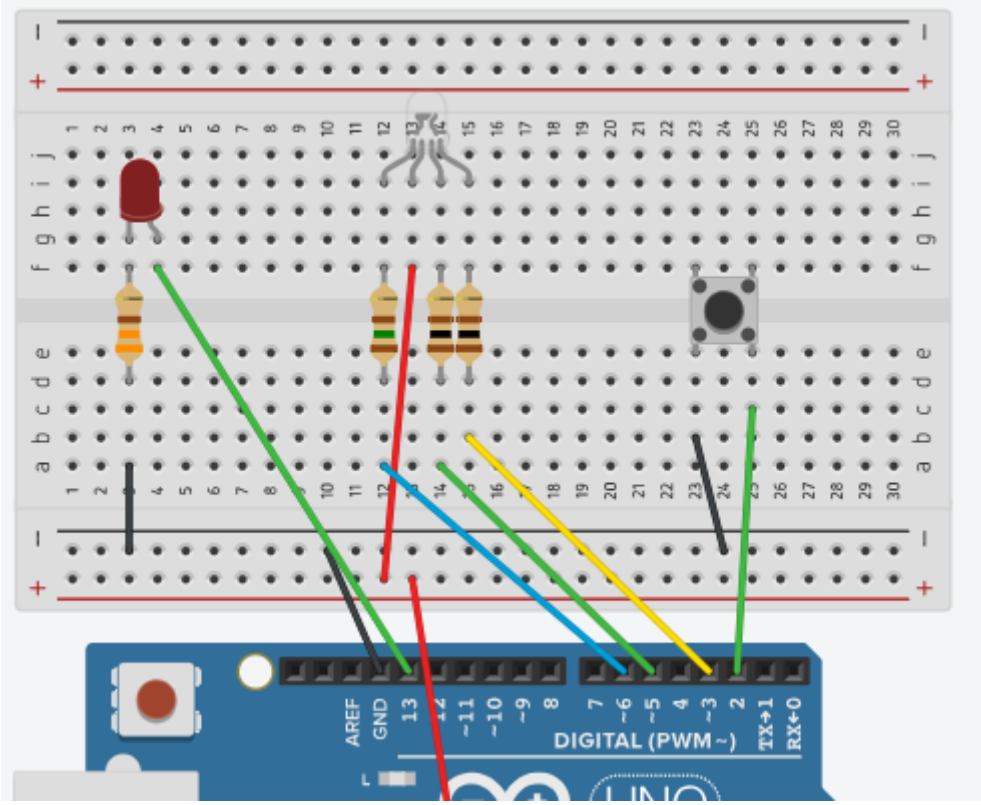

Merk ook op dat je binnen een interrupt functie (de functie waar je naar toe springt als jouw interrupt wordt gegenereerd) **geen delay()** kan gebruiken. Ook de **serial monitor** werkt hier niet juist.

Je kan reageren op verschillende veranderingen (triggers) op de INT pinnen bij de UNO:

LOW (pin gaat laag), HIGH (pin gaat hoog)(niet bij de UNO), CHANGE (als het signaal verandert aan de ingang), RISING (van laag naar hoog), FALLING (van hoog naar laag).

Meer info kan je vinden op deze plek. Ook op deze plek vind je uitbreid info over interrupts.

➔ Hoe ziet de code er dan uit wanneer de een interrupt gaan genereren na het indrukken van een knop?




We maken gebruik van **INT0 op pin 2.** We zorgen dat de **knop als laag actief** (met interne pullup weerstand) wordt aangesloten.
We gaan ook de **toestand van het interrupt** tonen op een LED **op pin 13.**
Zo kunnen we makkelijker debuggen als er nog iets fout gaat.

Merk op dat laag actieve knoppen in Tinkercad niet kunnen getest worden (INPUT_PULLUP bestaat er niet). Je kan wel een interrupt testen, maar dan met een hoog actieve knop.
Dus ofwel code en tekening aanpassen of in het echt in het labo testen.

RGBLED_PWM_INT

```
//definiëer pinnummers

int LED13 = 13;
int blauw = 3;//pwm pinnen
int groen = 5;
int rood = 6;
int d = 10; // delay in ms
int x; //teller maken
int interruptPin = 2; //int 0
volatile byte state = LOW; //variabele in interrupt functie als
                      // volatile (vluchtig) declareren !

void setup() {

  pinMode(blauw,OUTPUT);
  pinMode(groen,OUTPUT);
  pinMode(rood,OUTPUT);
  pinMode(LED13,OUTPUT);

  pinMode(interruptPin, INPUT_PULLUP);//laag actieve ingang voor INT1
  attachInterrupt(digitalPinToInterrupt(interruptPin),knop, FALLING);
  //als er een verandering op INT1 komt springen we onmiddelijk naar de ISR functie met naam   knop

 //zet alle uitgangen op maximum in het begin (common anode)
  analogWrite(blauw,255);
  analogWrite(groen,255);
  analogWrite(rood,255);

}
```

In het eerste deel van de code gaan we opnieuw de pinnen definiëren.
De interrupt pin krijgt gewoon de nummer van de pin waar het interrupt op zal gebeuren.

De variabele “state” wordt als “**volatile**” gedefiniëerd. De programmeurs van de interrupt routines raden ons aan dit type variabele te gebruiken. Zo zijn we zeker dat de inhoud van de variabele goed wordt geupdated tussen de main loop en de ISR.

De parameter “knop” is de naam die we gebruiken voor de **ISR** (interrupt service routine) of de routine waar we iets gaan aan/uitzetten voor we terug springen naar de hoofdloop. Deze naam valt vrij te kiezen.

Het **attachInterrupt()** commando dient om het interrupt op pin 2 aan te zetten, te laten reageren op neergaande flanken van de knop (handig bij laag actieve knoppen) en dan te springen naar de ISR “knop”.

```
void loop() {

  digitalWrite(LED13,state); //toestand state/interrupt op LED 13 tonen
  delay(300); //antidender

  if (state == HIGH){

       for (x=255; x > 0; x--){  //lus om van 255 naar 0 af te tellen
        analogWrite(blauw, x); //pwm commando analogWrite ipv digitalWrite !
        delay(d);
       }
       for (x=0; x < 255; x++){  //lus om van 0 naar 255 op te tellen
        analogWrite(blauw, x);
        delay(d);
       }

  }
}//einde loop

  //ISR routine waar er naar gesprongen wordt via INT0
  //deze routine zit buiten de loop gedefiniëerd
  void knop(){

  state = !state;  //inverteer de waarde van state, dit is een flipflop!

  }
```

In de loop testen we de status van “state” in de ISR door deze te tonen op LED13.

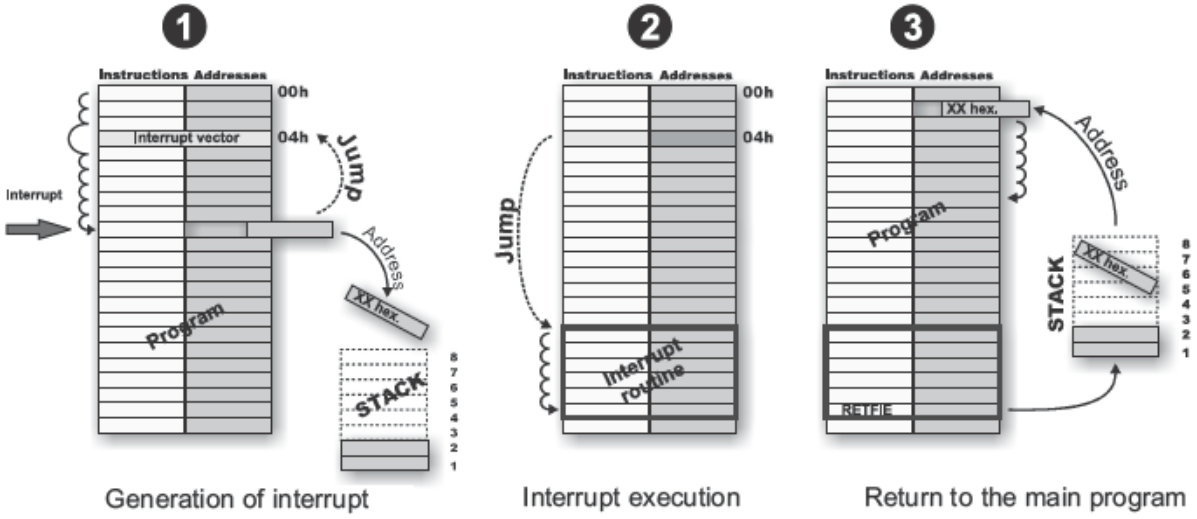
We voorzien ook wat **antidender**. Want onze knop levert wel meerdere op en neergaande pulsen na het drukken er op. Dat levert uiteraard meerdere interrupts op. En zo is de uitkomst dan niet zeker. Dus bouwen we een kleine vertraging in.

In de ISR “knop” wordt via de **“!” of NOT poort** steeds de waarde van “state” omgewisseld. Dit is een inverter en zorgt ervoor dat we een flipflop krijgen.

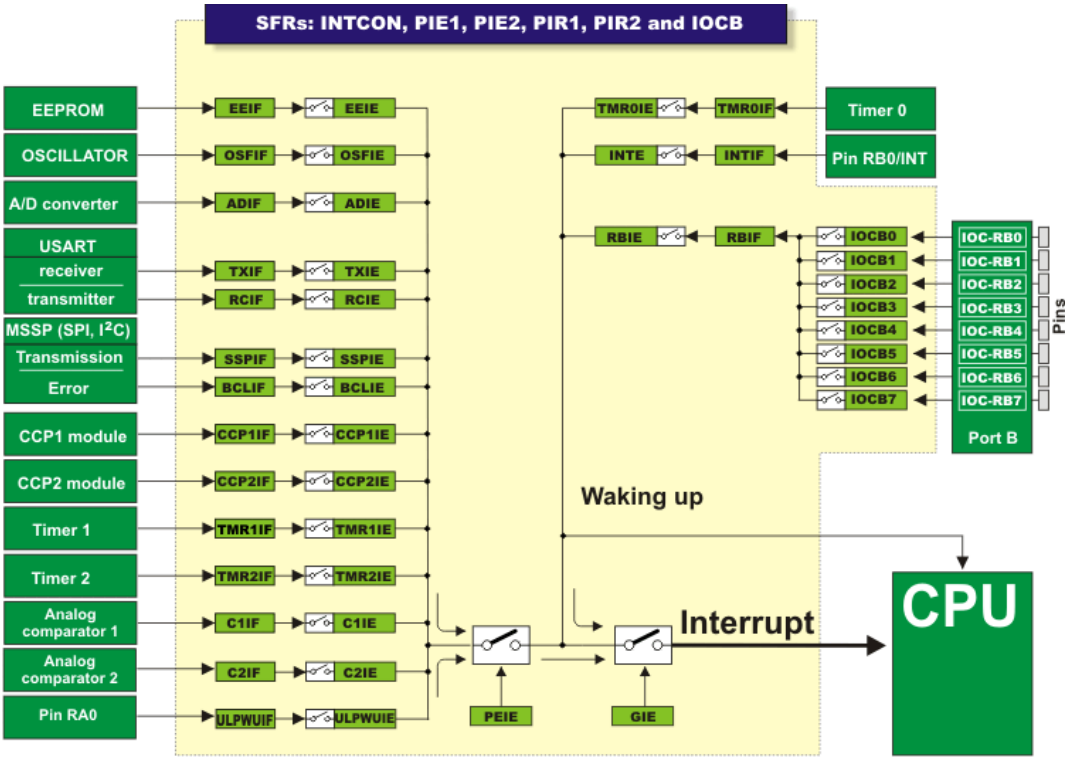
Opmerking: stel dat je in de code op een gegeven ogenblik het interrupt terug wil afzetten, dan moet je het commando “**detachInterrupt()**” gebruiken. Hier vind je meer info.

Er bestaan uiteraard niet alleen externe interrupts (interrupts door externe signalen gegenereerd) zoals INT0 en INT1.
Je kan ook interne interrupts genereren en afvragen. Dat vraagt nog meer uitdieping.
Onze cursus gaat hier niet verder op in omdat dit ons veel te ver zou brengen.
Meer info vind je uiteraard op deze plek.

**Extra info over interrupts:**

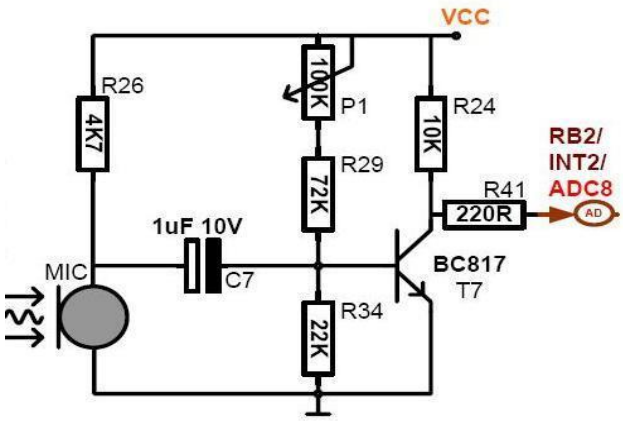



Andere voorstelling van hoe een microcontroller te werk gaat met een interrupt.



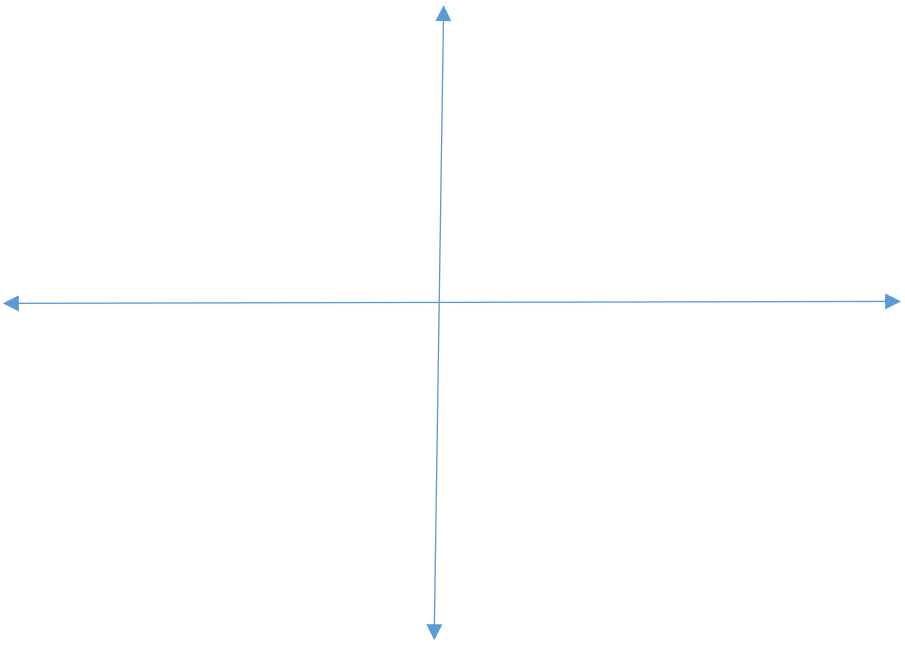

Een microcontroller kan veel verschillende interrupts hebben. We moeten zorgen dat enkel het gewenste interrupt actief maken.

➔ **Extra interrupt uitdaging: bouw een kleine versterker** en laat jouw robot pas rijden **als je geklapt hebt in je handen. Bestudeer en simuleer** de volgende versterker. Bouw deze dan op jouw **breadboard** en maak de **koppeling met de Arduino**. Gebruik daarna bovenstaande interrupt kennis om de versterker op het juiste moment te laten werken.




**Noteer hier de werking van de microfoonversterker en simuleer deze in Multisim / Tina TI:**

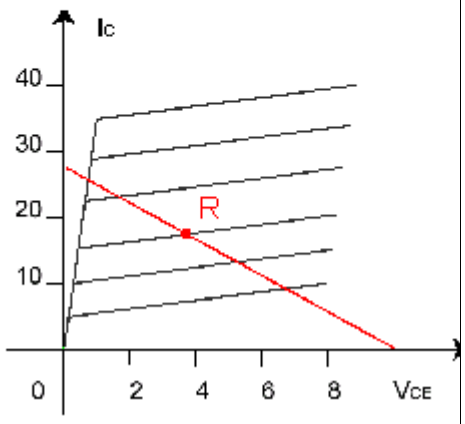
Noteer hier de klasse A transistor curve:

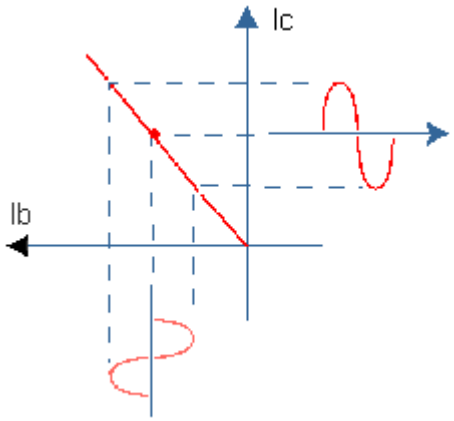


**Ter info: soorten versterkers: zie info op**
**http://cursus.radioclubleuven.be/cursus/sc/class.htm**
**http://www.popschoolmaastricht.nl/college_versterkers_klassen.php**

**Klasse A :**

Om een versterker in klasse **A** te laten w
moet je het instelpunt **"R"** in het midden
lijn plaatsen.



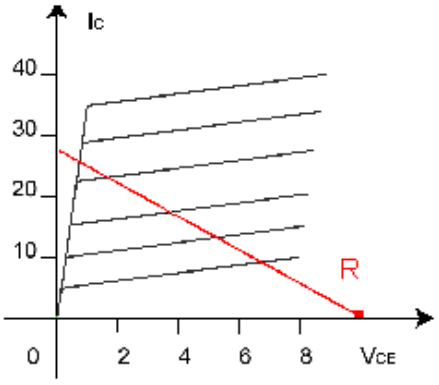

De A-klasse instelling wordt veel gebruikt bij LF (laagfrequente) signalen

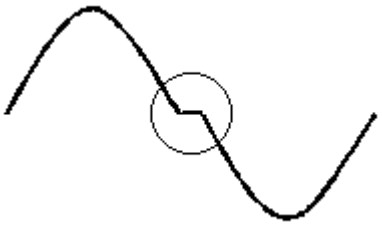
Deze werkwijze vergt veel energie zodat het rendement nauwelijks 50 % haalt

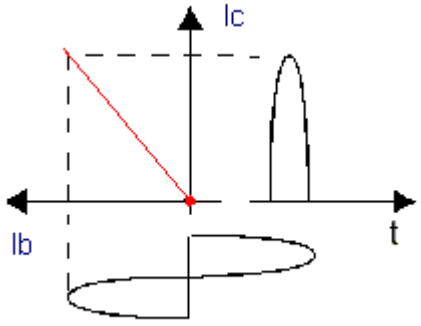
Dit geeft een goede lineariteit ( het resultaat aan de uitgang volgt rigoureus de vorm aan de ingang ).

**Klasse B :**

Het instelpunt hierboven bij Ic=0, de transistor staat volledig gesperd.






De transistorschakeling verbruikt geen onnuttige energie meer als er geen signaal wordt aangelegd. Daar tegenover staat dat we aan de uitgang slechts de helft krijgen van wat aan de ingang werd aangelegd Om beide alternanties te verwerken kunnen we beroep op de zogenaamde PUSH PULL schakeling.

**Goed rendement (60%);**
**- Vervorming bij overgang naar volgende fase;**

De vervorming ontstaat bij overgang van één alternantie naar de andere. Er is minstens 0.7V nodig om de transistor diode te laten geleiden.

**- Noodzaak van een symmetrische opstelling (LF).**

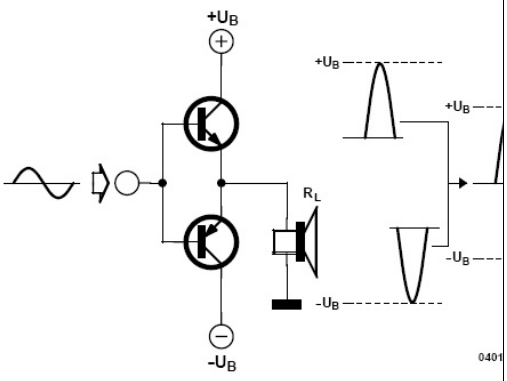
**Instellingsklasse AB :**

Vaak worden onze versterkers in deze klasse AB ingesteld. Je vermoedt het al: het instelpunt zal ergens tussen klasse A en B in liggen.

Je polariseert zodanig de schakeling voor kleine signalen in het lineair deel ( klasse A) werkt en in klasse B voor grote signalen.

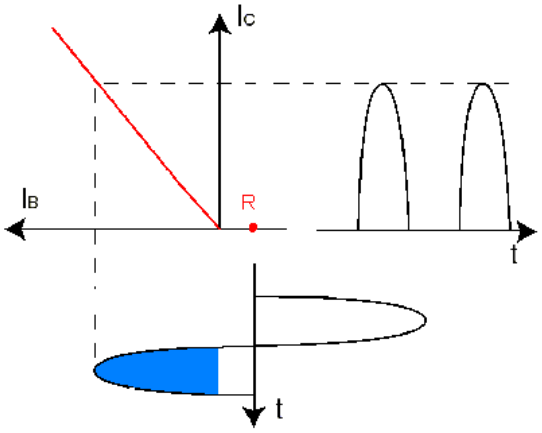
**- Gemiddeld rendement;**
**- Aanvaardbare lineariteit.**

Klasse B voorbeeld:

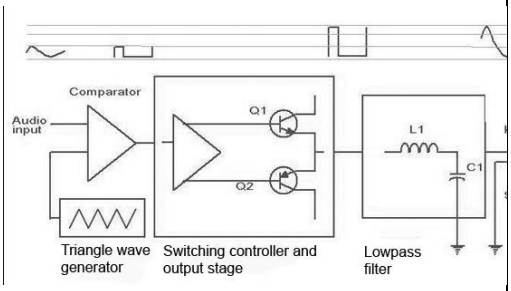



**Klasse C :**

Deze klasse is erg nuttig bij gebruik van frequentievermenigvuldigers, en in het algemeen als krachtige versterking noodzakelijk is waarbij de lineariteit niet zo belangrijk is. Voor radio is deze klasse minder geschikt, afgezien van bijzondere gevallen ( bijv. vermogensversterking).

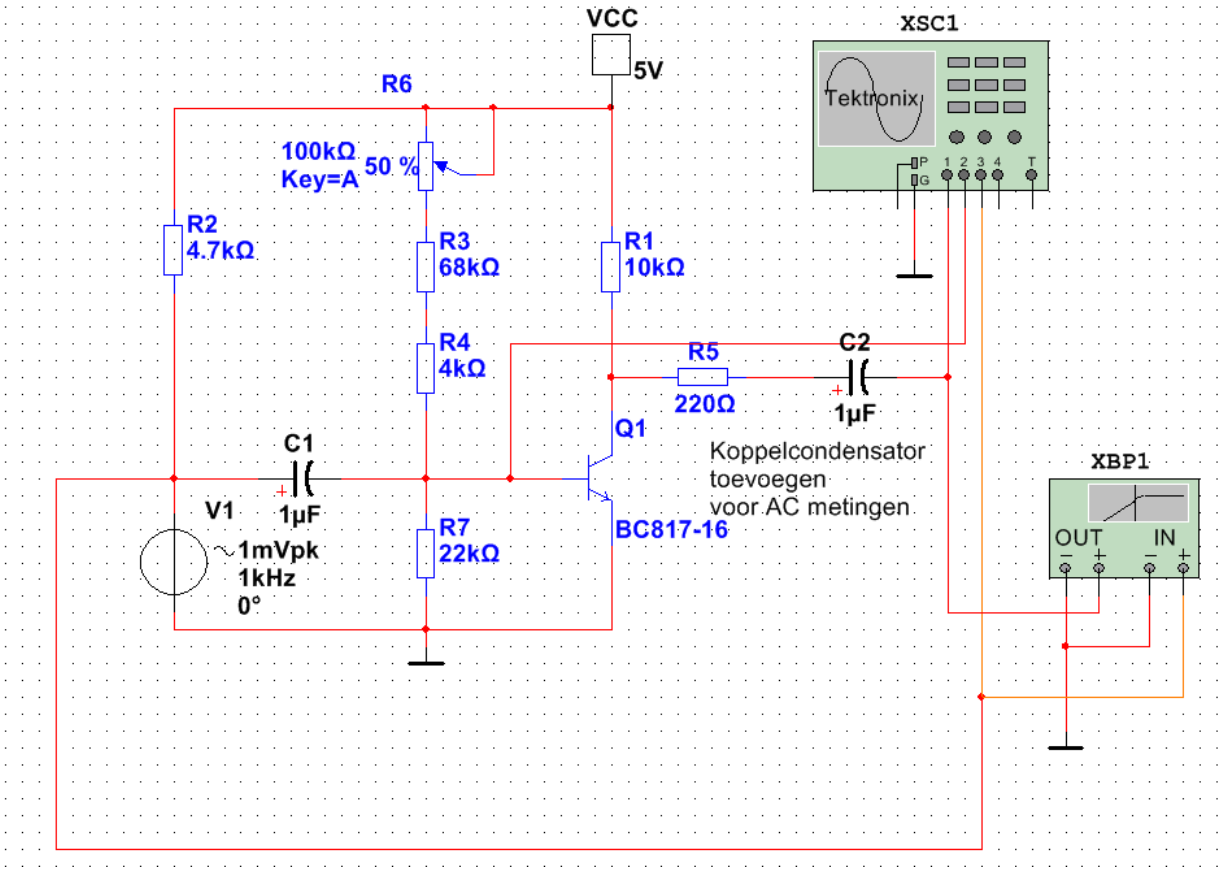


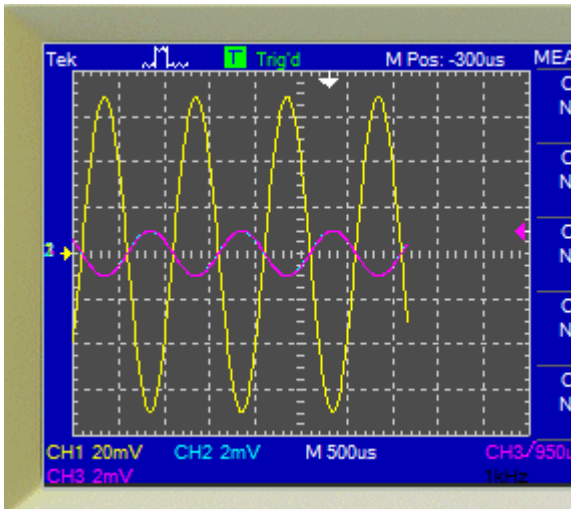

**Klasse D:**

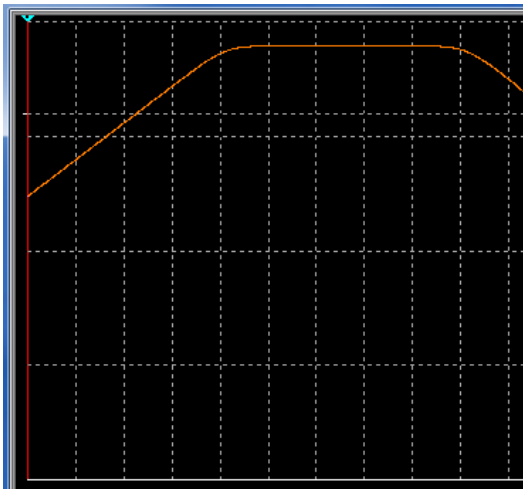



Klasse D versterkers, ook wel digitale versterkers genoemd, zijn schakelende versterkers (switching amplifiers). De uitgang wordt snel in en uitgeschakeld via pulsbreedtemodulatie (Pulse Width Modulation, PWM), tenminste enkele malen per cyclus. Aangezien de uitgang of compleet aan staat, of compleet uit, wordt er in theorie geen vermogen verspild. Omdat er helaas geen verliesvrije en oneindig snelle schakelaars bestaan, is er toch altijd wel wat verlies. Rendement ligt rond de 90% en hoger.

➔ **TAAK: simuleer de klasse A versterker in MS / Tina TI op een scope en bode plotter. Bestudeer zo de werking (signaal op ingang en uitgang) en de versterkingsfactor.**



**Zorg dat je ook zo'n scope meting en bodeplot krijgt in Multisim. Bespreek de schakeling en geef een verslag af.**

## 1.6 Wat als de LED meer dan 20mA trekt?

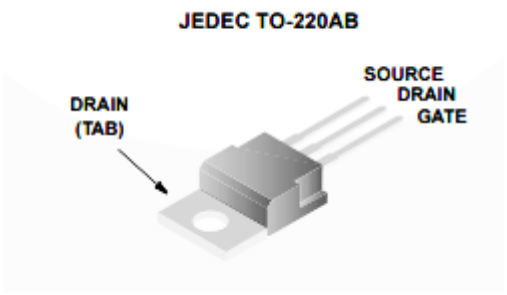
Stel dat je nu **meer LEDs wil aansturen tegelijk in parallel**, of je wil LEDs gebruiken die **meer vermogen** vragen (vb 100mA). Dan moet je een elektronische vermogenschakelaar gebruiken i.p.v. rechtstreeks de microcontroller uitgang te gebruiken.

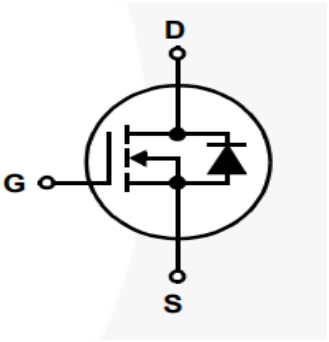
Zoals reeds aangegeven in de datasheet van onze controller kunnen we in het beste geval slechts 40mA vragen van een pin (best bij 20mA houden).

➔ Wat is een elektronische vermogenschakelaar?

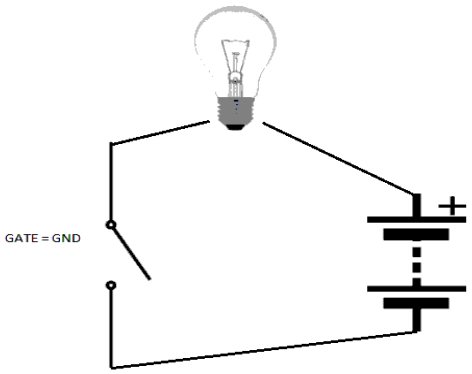
We denken hierbij in eerste instantie aan de **transistor en MOSFET** voor onze kleine toepassingen. Wil je graag 220V en grotere vermogens sturen, dan raden we de solid state relais (veilige omgeving), thyristor of triac aan (wordt niet behandeld in deze cursus).

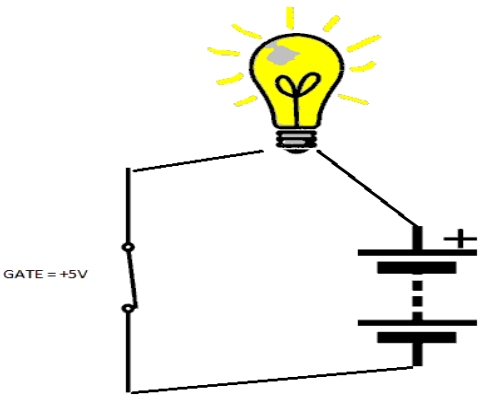
Net als we in de basiscursus de DC motor gingen aansturen met bijvoorbeeld een MOSFET, kunnen we dit principe hier ook toepassen op een vermogen LED.




Principe werking MOSFET:




We sluiten de MOSFET aan als schakelaar

Wanneer we +5V op de gate aansluiten gaat de MOSFET sluiten

(= gesloten verbinding tussen DRAIN en SOURCE)

Wanneer we GND op de gate aansluiten gaat de MOSFET open

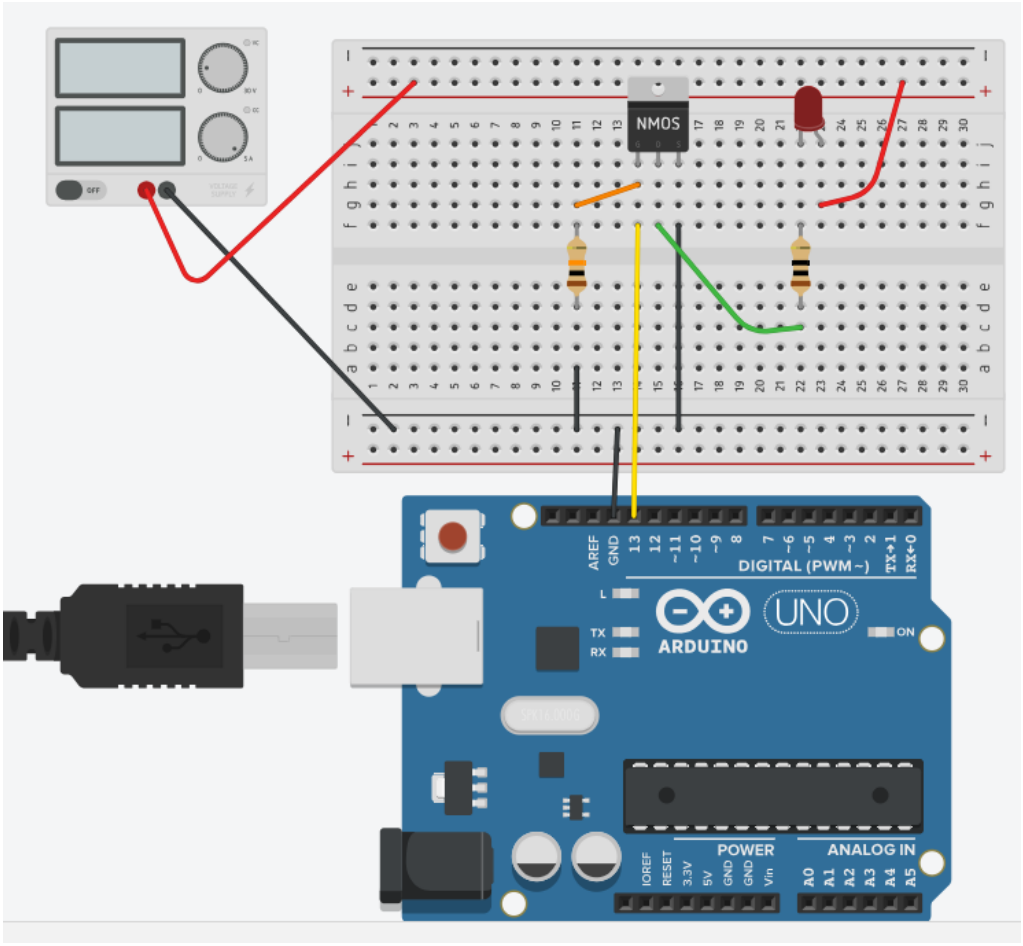
(= open verbinding tussen DRAIN en SOURCE)

Let op de juiste aansluitingen!!!

➔ Welke MOSFET moet ik gebruiken?

Als je met 500mA, 60V maximum tevreden bent kan je gebruik maken van de **BS170** (niet voor een DC motor gebruiken, is minimum 600mA).

Wil je al stromen tot 12A bedwingen bij max 100V dan gebruik je best de **RFP12N10L**.




Bovenstaande figuur geeft aan hoe je de MOSFET moet aansluiten.

Ga op zoek in de datasheet naar de aansluitingen: GATE, DRAIN en SOURCE.

De SOURCE hang je aan de massa van de Arduino.

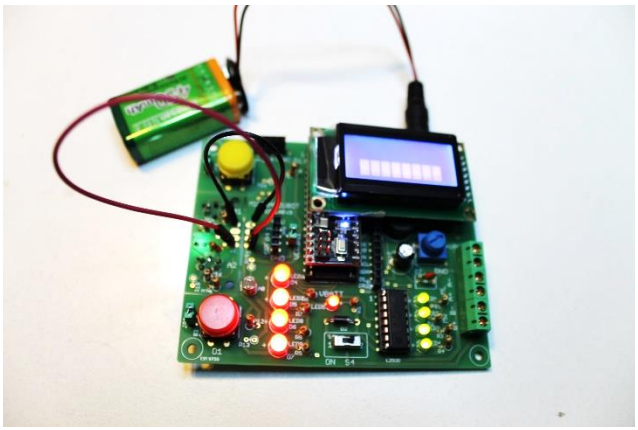
De DRAIN hangt via de verbruiker (hier de LED en voorschakelweerstand) aan **de externe voeding.**

Dit is omdat de Arduino niet zoveel stroom zelf kan leveren via de USB kabel!

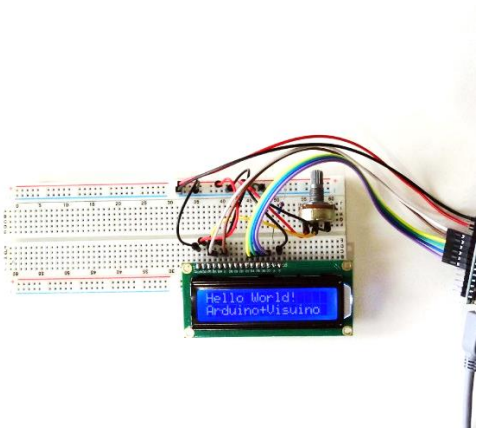
De GATE wordt op de Arduino OUTPUT pin aangesloten. Wanneer hier +5V komt op te staan zal de LED branden.

De **10K naar massa** is voorzien wanneer per ongeluk de GATE zou loskomen te zitten. Dan zou de MOSFET inschakelen zonder dat we dit wensen. Dit is een **pulldown weerstand**.

# 2. Een LCD scherm leren aansturen



2 x 8 LCD scherm op MicroArdubot PCB




2 X 16 LCD scherm op breadboard

Een LCD scherm (Liquid Crystal Display) is een klein schermpje waarop we tekst kunnen plaatsen dewelke ons kan helpen met het debuggen van programma's.

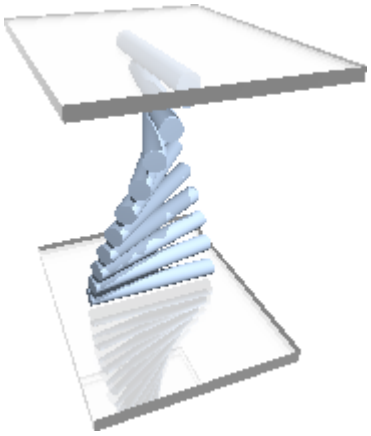
Je kan er ook meetwaardes van sensoren op plaatsen, dewelke je juist wil afstemmen, of zelfs een volledig spelletje tonen. Zo moeten lijnvolgers meestal analoog eerst gekalibreerd worden alvorens de robot juist weet wat nu zwart of wit is. Deze waardes tonen we dan eerst op het LCD scherm. Daarna stoppen we die in ons programma.

Een LCD scherm is een handige vervanger voor een serial monitor wanneer de opstelling niet toelaat dat er constant een PC moet aangesloten zijn om gegevens te verwerken of te tonen.

## 2.1 Hoe werkt een LCD scherm?

Wikipedia leert ons het volgende:

De werking berust op het effect dat de "vloeibare kristallen" in het display in staat zijn om de polarisatierichting van licht te draaien als er een elektrische spanning op wordt gezet.



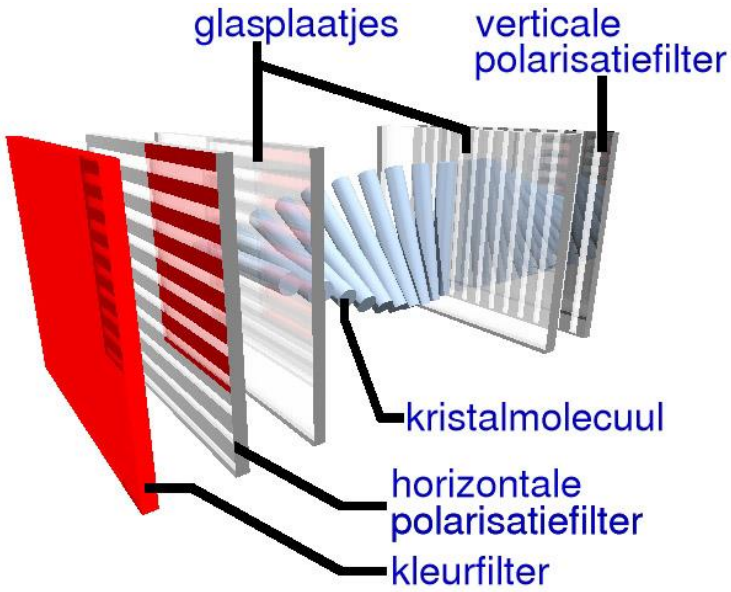
Het vloeibare kristal bestaat uit staafvormige complexe moleculen die in onderlinge interactie een helische structuur aannemen waarbij elk molecuul een stukje gedraaid ligt ten opzichte van het onderliggende molecuul.

Er bestaan **verschillende soorten** LCD's:

**Passieve** matrix LCD:
- Zwart-wit (zie het STN (super-twisted nematic) LCD type). Hierbij kunnen de moleculen van 180° tot 270° gedraaid worden ipv 90°). Deze verbruiken minder stroom en zijn goedkoper dan TFT schermen. Ze zijn wel trager in response time dan de TFT schermen.
- Grijstinten
- Kleur

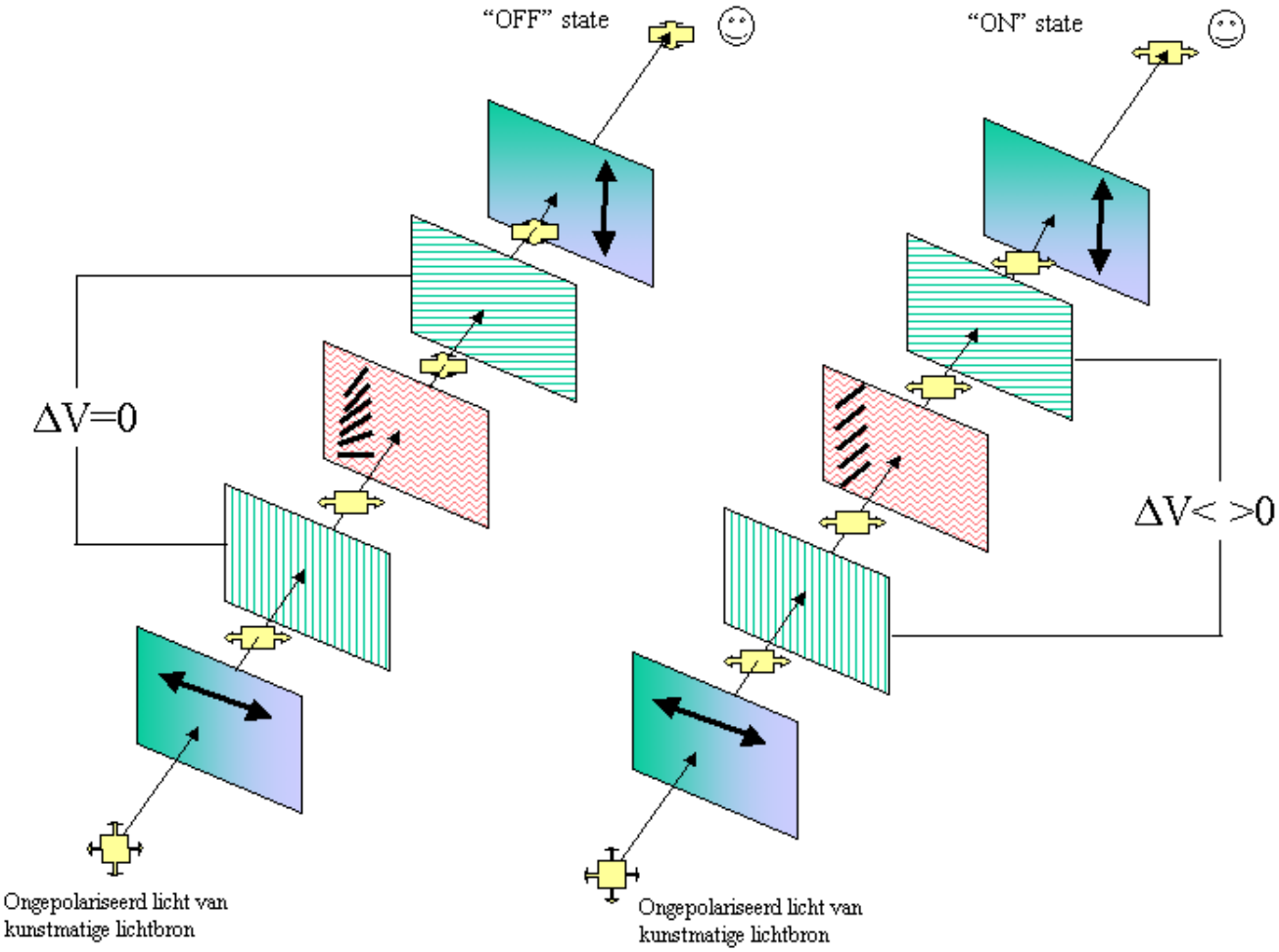
**Actieve** matrix LCD:
- TFT (Thin Film Transistor)
- LTPS (Low Temperature Polycrystalline Silicon)
- TFD (Thin Film Diode)

Een display kan ook:
- **Reflectief** (weerkaatsing met spiegel in LCD cel, zie rekenmachine)
- **Transmissief** (kunstmatige lichtbron achter het display, zie laptops)
- **Transflectief** (combinatie van beide vorige, zie GSM's)




Deze figuur toont 1 pixel van het LCD scherm

De lcd-cel bestaat uit twee glasplaten die met een fotolithografieproces voorzien zijn van elektroden van ITO (indium-tinoxide). Daartussenin zit een laagje vloeibaar kristal (LC). Aan de buitenzijden van de cel zitten twee polarisatiefilters geplakt. In het geval van een reflectief of transflectief display zit er nog een spiegel in de cel, of is deze geïntegreerd in het achterste polarisatiefilter.




We volgen de lichtstraal vanaf de kunstmatige lichtbron in het geval er geen spanning over de elektroden staat. De LC-moleculen zijn in hun natuurlijke gedraaide toestand. Het invallende licht is ongepolariseerd licht. Wanneer de lichtstraal het polarisatiefilter passeert wordt alleen het licht met één specifieke polarisatierichting doorgelaten. Er gaat hierbij dus een hoop licht verloren. Het uittredende licht is dus gepolariseerd. Vervolgens passeert de lichtstraal via de glasplaat het LC, waardoor de polarisatierichting wordt veranderd door de gedraaide structuur van het LC. Via de tweede glasplaat komt het licht bij een tweede polarisatiefilter dat wederom alleen licht doorlaat met één bepaalde polarisatierichting. Wanneer dit overeenkomt met de polarisatierichting van het licht uit de LC, dan zal er in totaal dus een hoeveelheid licht door de lcd-cel zijn gepasseerd en kan de waarnemer "licht" zien. In een reflectief display gaat het teruggekaatste licht zonder problemen terug via het polarisatiefilter, omdat de polarisatierichting van het licht overeenkomt met die van het filter.

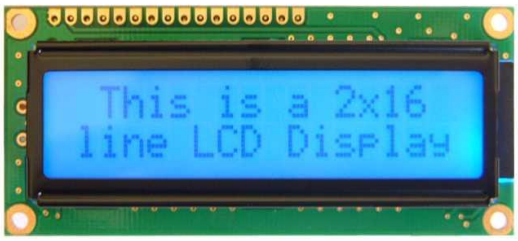
Stel nu dat je een potentiaalverschil over de LC-laag zet, dan wordt de gedraaide structuur van de LC-moleculen verstoord en zullen ze zich allemaal richten (vanwege hun eigen elektrische lading) naar het opgelegde elektrische veld. Om te voorkomen dat de moleculen zich naar één elektrode bewegen wordt een wisselspanning opgelegd. Het licht zal nu niet van polarisatierichting worden veranderd en zal dus het tweede filter niet kunnen passeren, en dus zal de waarnemer "zwart" zien. Dit effect kan ook omgedraaid worden door juist als tweede polarisatiefilter er een te kiezen dat geen licht doorlaat dat getwist is en wèl licht van een andere polarisatierichting. Dan zal de waarnemer in de "uit-toestand" zwart zien en in de

"aan-toestand" licht. Dit laatste is gebruikelijk bij kleurendisplays, terwijl in zwart-witdisplays de uit-toestand vaak "wit" is.

Door op de onderste glasplaat verticale ITO-banen te etsen *("commons")* en op de bovenste plaat horizontale banen *("segments")* krijg je een raster van ITO-sporen waarmee je op elk kruispunt een spanning kunt opleggen. Zo worden de pixels gecreëerd die elk afzonderlijk aan of uit kunnen worden gezet. Dit wordt aangestuurd door een chip, die vaak op de onderste glasplaat is gezet of op een extern stukje folie of PCB.

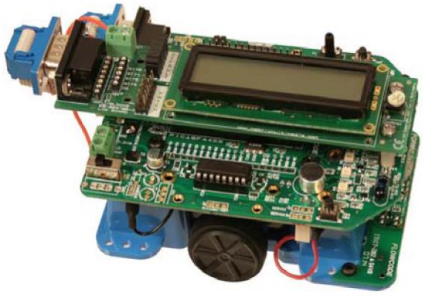
## 2.2 Welk LCD display zit er meestal in de Arduino startersdoos?

De Arduino starterskit bevat een veel gebruikte **162C-CC-BC-3LP** LCD scherm.

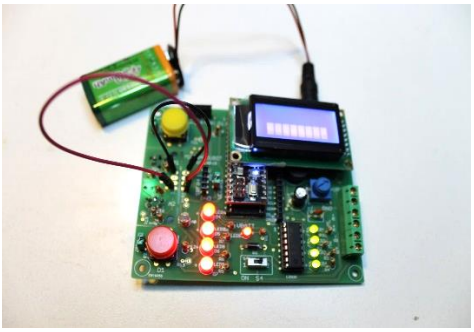



Hierbij is de achtergrond blauw, heeft deze een **backlight LED**, is de display matrix technologie **STN** en is de display mode **transflective** (zie vorig hoofdstuk).

Soms wordt ook gebruik gemaakt van het LCD scherm zonder achtergrond licht: **162C-BA-BC**



**FFrobot met zwart wit LCD**



**MicroArdubot LCD met achtergrond verlichting**

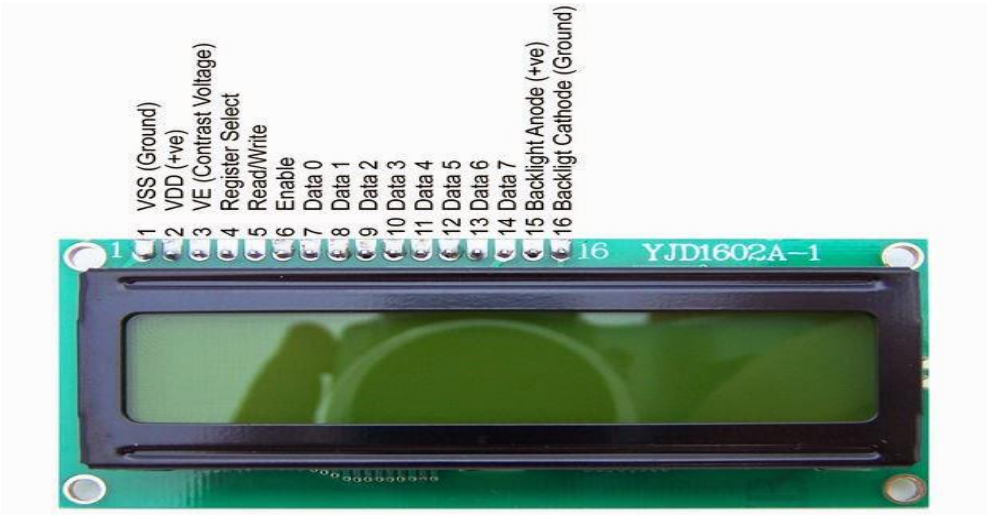
Bij de MicroArdubot maken we gebruik van een 8 x 2 LCD scherm. Ook hier is de achtergrondverlichting voorzien. Als nummer kan je hiervoor de **0802A** (14 pins) of **0802A1 rev A** (16 pins) gebruiken.

## 2.3 Aansluitingen LCD scherm 16 x 2 aan?

Het LCD scherm bevat heel wat aansluitingen. We zetten ze even op een rijtje:

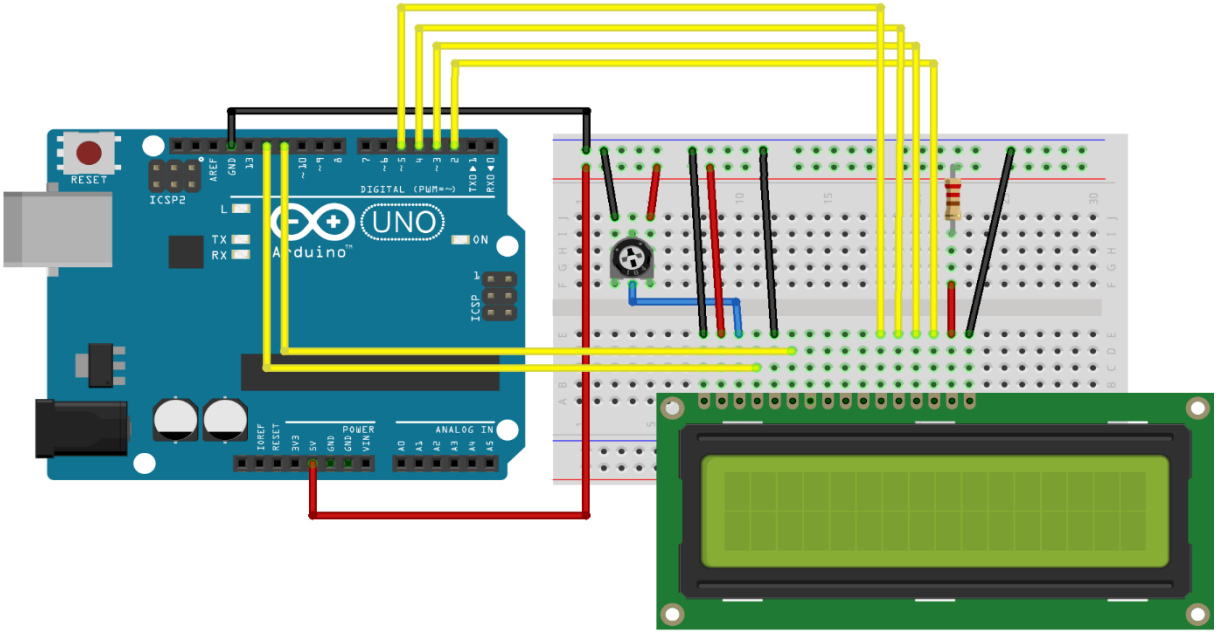
- RS: **register select** pin: via deze pin wordt het data (vb tekst) of instructie (vb scroll, clear) register van het LCD aangestuurd.
- R/W: **read /write** pin: via deze pin geven we aan het LCD te kennen of we gaan data schrijven naar het LCD of er iets uit lezen (hier zit tenslotte ook een microcontroller in, de gekende KS0066U van Samsung of de HD44780 van Hitachi. Deze bevat de o.a. de karakterset en commando's die jij gaat doorsturen via Arduino).
- E: **Enable** pin: deze pin staat toe dat we in de registers van het LCD kunnen schrijven.
- **8 data pinnen** (D0 – D7): via deze pinnen sturen we de data naar het register in het LCD of lezen we de data uit. Je kan met 4 bits mode werken of met 8 bits mode.
- **VE** (contrast): hiermee kunnen we, via een potmeter, het contrast regelen van het LCD.
- **+5V (VDD) en GND (VSS)**: de voeding van het LCD
- **Bklt+ en Bklt**-: de LED backlight spanning die moet worden aangesloten opdat het achtergrond licht zou aangaan van de LCD. Deze is 5V, maar het is aangeraden om een kleine voorschakelweerstand te gebruiken om de stroom wat te beperken voor het Arduino bordje.

Aansluitingen van het LCD scherm:

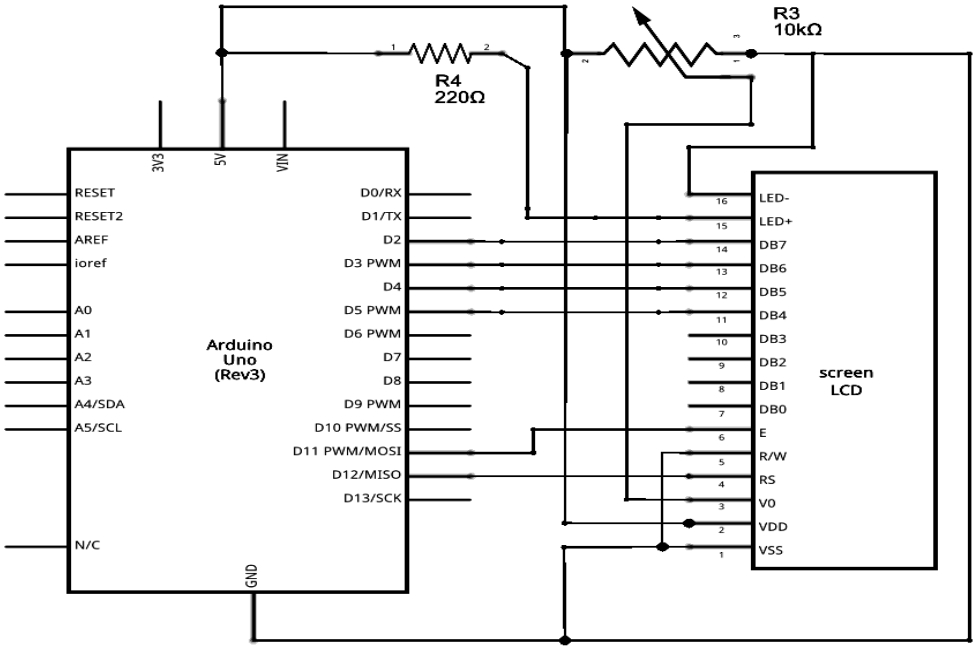


## 2.4 Hoe sluiten we het LCD aan op een breadboard?

Onderstaande opstelling geeft een voorbeeld aan van een aansluiting van een LCD scherm op een Arduino UNO.

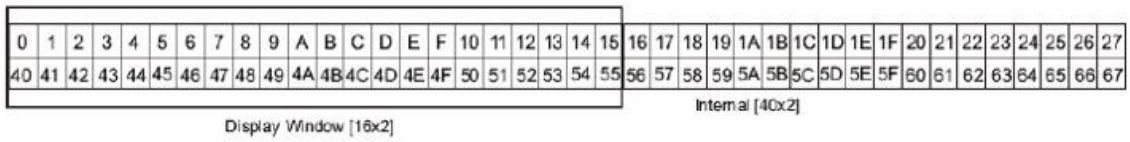


Je kan zien dat we hier gebruik maken van de 4 bits data mode. D0 – D3 zijn niet gebruikt.
Voor de backlight maken we gebruik van een 220 ohm voorschakelweerstand. Deze weerstand mag ook 100 ohm worden, als je meer achtergrond licht wil hebben. Bij de MicroArdubot gebruiken we zelfs maar 10 ohm. Natuurlijk is er dan veel meer stroomverbruik en geraakt de batterij sneller leeg.
De potmeter van 10K wordt gebruikt om het contrast af te regelen van het LCD scherm. De zit aan 1 kant aan de VDD (+5V) en de andere kant aan de VSS (GND). De middelste pin (de loper) zit aan de VE ingang.
Pin 5 (read/write) hangen we standaard aan de massa. We gaan altijd schrijven naar het scherm.

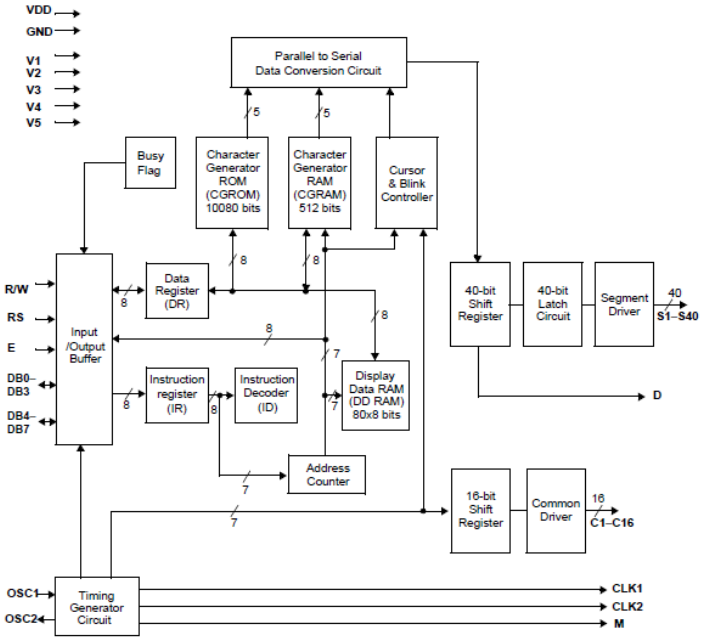


## 2.5 Wist je dat?

Jouw scherm kan in feite 20 karakters achter elkaar in het geheugen opslaan. Hij toont er telkens maar max 16.




In het LCD ook een microcontroller zit. Hier kan je duidelijk zien dat je tot 40 karakters kan gaan.




Je kan op jouw LCD scherm **karakters tonen dewelke je zelf hebt ontworpen**. Pixel per pixel geef je deze in en stuur je deze door. Test deze code maar eens:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

byte smiley[8] = {
  B00000,
  B10001,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
};

void setup() {
  lcd.createChar(0, smiley);
  lcd.begin(16, 2);
  lcd.write(byte(0));
}

void loop() {}
```

https://www.arduino.cc/en/Reference/LiquidCrystalCreateChar

## 2.6 De LiquidCrystal bib toevoegen aan Arduino

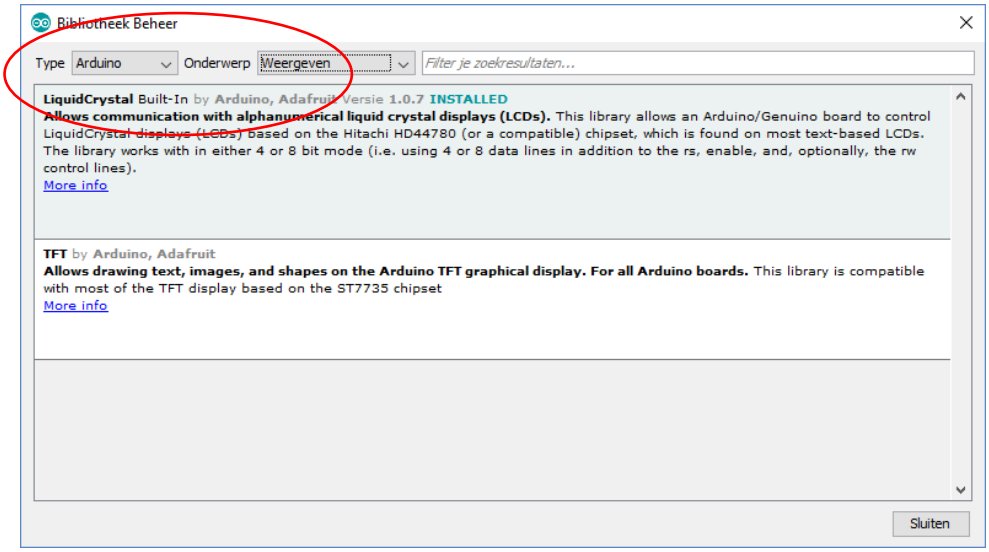
Voordat we kunnen gaan compileren moeten we de LiquidCrystal bib toevoegen aan jouw Arduino omgeving. Deze bib bevat een verzameling van functies die kunnen gebruikt worden bij het LCD scherm. Vele bibliotheken zijn reeds al in Arduino toegevoegd. Soms moet er eentje bijgevoegd worden. Normaal is de LiquidCrystal bib al toegevoegd.

**Zo niet, hoe doen we dat dan?**

We controleren eerst of onze bib al geïnstalleerd is.

Klik hiervoor in Arduino op **Schets -> Bibliotheek gebruiken -> Bibliotheek beheren**

Als je nu de filter instelt op type = “Arduino” en Onderwerp = “Weergeven”, dan zie je dat “**LiquidCrystal**” verschijnt. Hierachter moet “**INSTALLED**” staan.




Dit geeft aan dat de bib reeds voorzien is en dat we al aan de slag kunnen.

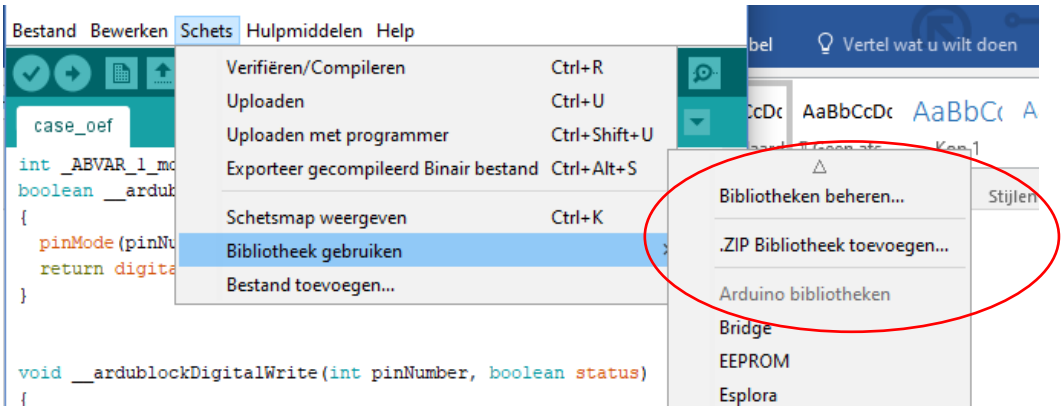
Stel dat deze niet verschijnt dan kan je het volgende doen:

Zoek op https://github.com/arduino/Arduino/issues in de zoekfunctie naar jouw library.

Download de library als een zip file. Pak deze file niet uit! Deze bevat een *.h (header file) en een *.cpp file. Soms zit er ook een *.text of andere voorbeelden bij.

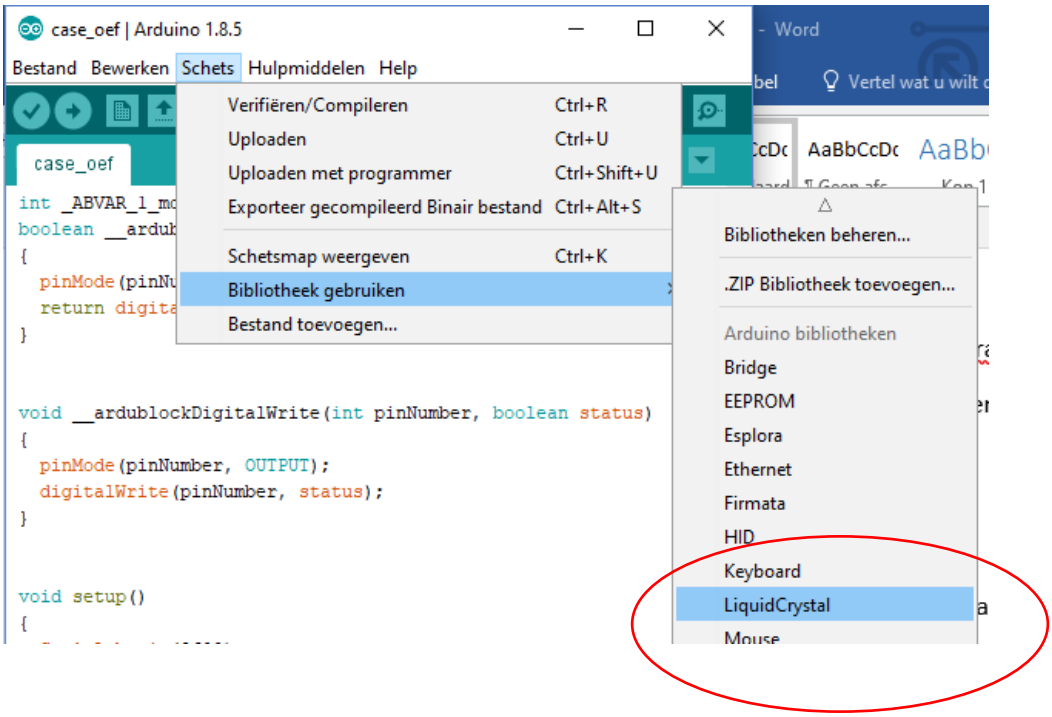
Plaats de file in een directory op jouw PC (maakt niet uit waar).

Ga nu in Arduino naar **Schets -> bibliotheek gebruiken -> .ZIP toevoegen**




Blader naar de juiste ZIP file en open deze vanop de plaats waar je deze net had gesaved.

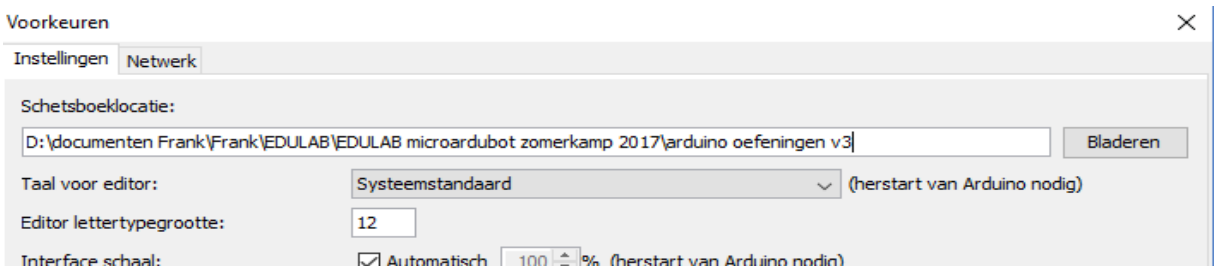
Als je nu in de **Schets -> bilbliotheek gebruiken -> zie lijstje**




Nu zie je de library terugkomen in het lijstje en kan je aan de slag.

Ps: Je kan ook de zip file uitpakken en in de sketchbook location plaatsen, dan moet de library ook gevonden worden. De sketchbook location stel je in via Arduino. Ga hiervoor naar

**Bestand -> voorkeuren -> sketchbook location**



## 2.7 Welke LCD functies kan ik gebruiken?

Deze info kan je terugvinden op de volgende plek van de **LiquidCrystal bibliotheek**:
https://www.arduino.cc/en/Reference/LiquidCrystal

Deze bib moeten we vooraf in onze code inladen. Zodoende weet Arduino hoe hij moet gebruik maken van deze LCD functies. Hoe je deze bib in Arduino binnenhaalt lees je in hoofdstuk 3.6.

Op het begin van de code staat **#include <LiquidCrystal.h>** . Dit is de manier om de lib in de code binnen te halen.

Hier volgt kort een overzicht van de LCD functies:

| | |
|---|---|
| LiquidCrystal lcd(…) | Deze functie moet op het **begin van de code** staan om Arduino aan te geven dat we gaan werken met een LCD.<br>Je geeft hier de naam van het LCD (vb lcd) mee en op welke pinnen je alles hebt aangesloten. Voor ons scherm geeft dit het volgende:<br>**LiquidCrystal lcd(rs, enable, d4, d5, d6, d7)**<br><br>We moeten hier onze pinnen invullen:<br>**LiquidCrystal lcd(12, 11, 5, 4, 3, 2);** |
| lcd.begin(K, R) | Ons lcd bestaat uit x kolommen en y rijen (lijnen). Deze moet je op het begin eenmalig aangeven in de void setup() code.<br><br>Dan krijgen we voor ons 16 x 2 scherm: **lcd.begin(16,2);** |
| lcd.setCursor(K, R) | Kolom 0 … Kolom 7<br>Lijn 0: 0,0 1,0 2,0 3,0 4,0 5,0 6,0 7,0<br>Lijn 1: 0,1 1,1 2,1 3,1 4,1 5,1 6,1 7,1<br>Vb van een 2 x 8 LCD scherm<br><br>Via deze functie kunnen we de cursor op een **startpositie** plaatsen.<br>Merk op dat we **steeds van 0 beginnen** te tellen.<br><br>Wil je bijvoorbeeld het eerste character printen op kolom 0, rij (lijn) 1, dan type je de code: **lcd.setCursor( 0,1);** |
| lcd.print(“….”) | Deze functie dient om tekst op het scherm te printen. Alles tussen de “ “ wordt afgeprint.<br><br>Vb: **lcd.print(“hello world”);** |
| lcd.print(var) | Wil je graag een variabele afprinten, dan gebruik je geen “ “. Je kan direct de variabele naam plaatsen tussen de ().<br><br>Vb: **lcd.print(teller);** |

Als je alle code samen zet in 1 programma dan krijg je het volgende:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

Test deze code uit en print een seconde teller op het LCD scherm.

## 2.8 Nog meer functies voor het LCD scherm

Hier volgt nog een lijstje van meer uitgebreide functies voor het LCD scherm. Test zeker deze ook eens uit. Klik op de link voor meer info.

| | |
|---|---|
| • Autoscroll: Shift text right and left.<br>• Blink: Control of the block-style cursor.<br>• Cursor: Control of the underscore-style cursor.<br>• Display: Quickly blank the display without losing what's on it.<br>• Scroll: Scroll text left and right.<br>• Serial Display: Accepts serial input, displays it.<br>• Set Cursor: Set the cursor position.<br>• Text Direction: Control which way text flows from the cursor. | clear()<br>home()<br>write()<br>noCursor()<br>noBlink()<br>noDisplay()<br>scrollDisplayLeft()<br>scrollDisplayRight()<br>autoscroll()<br>noAutoscroll()<br>leftToRight()<br>rightToLeft()<br>createChar() |

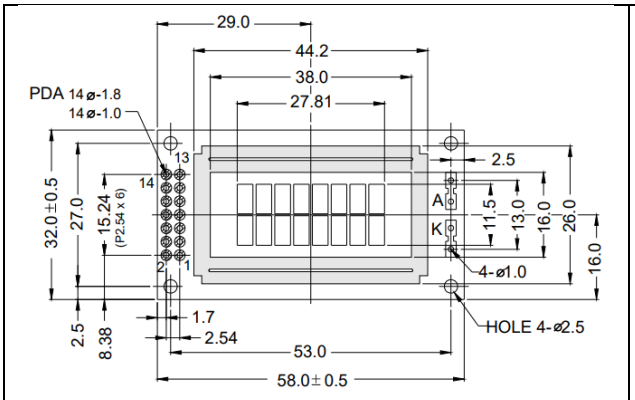
## 2.9 Wat met een 2 x 8 LCD?

Bij de MicroArdubot maken we gebruik van een 2 x 8 LCD (zie onderstaande datasheet van de 0802A).

Hier moet je bijvoorbeeld de volgende code gebruiken:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(8, 2);
  // Print a message to the LCD.
  lcd.print("hello");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```


Enkel de aansluitpinnen en de afmetingen van het LCD zijn aangepast.




| PIN ASSIGNMENT | | |
|---|---|---|
| Pin no. | Symbol | Function |
| 1 | Vss | Power supply(GND) |
| 2 | Vdd | Power supply(+) |
| 3 | Vo | Contrast Adjust |
| 4 | RS | Register select signal |
| 5 | R/$\overline{W}$ | Data read / write |
| 6 | E | Enable signal |
| 7 | DB0 | Data bus line |
| 8 | DB1 | Data bus line |
| 9 | DB2 | Data bus line |
| 10 | DB3 | Data bus line |
| 11 | DB4 | Data bus line |
| 12 | DB5 | Data bus line |
| 13 | DB6 | Data bus line |
| 14 | DB7 | Data bus line |

**Wat als je enkel blokjes op het scherm krijgt?**

Dan is er meestal nog iets mis met de datalijnen bedrading.

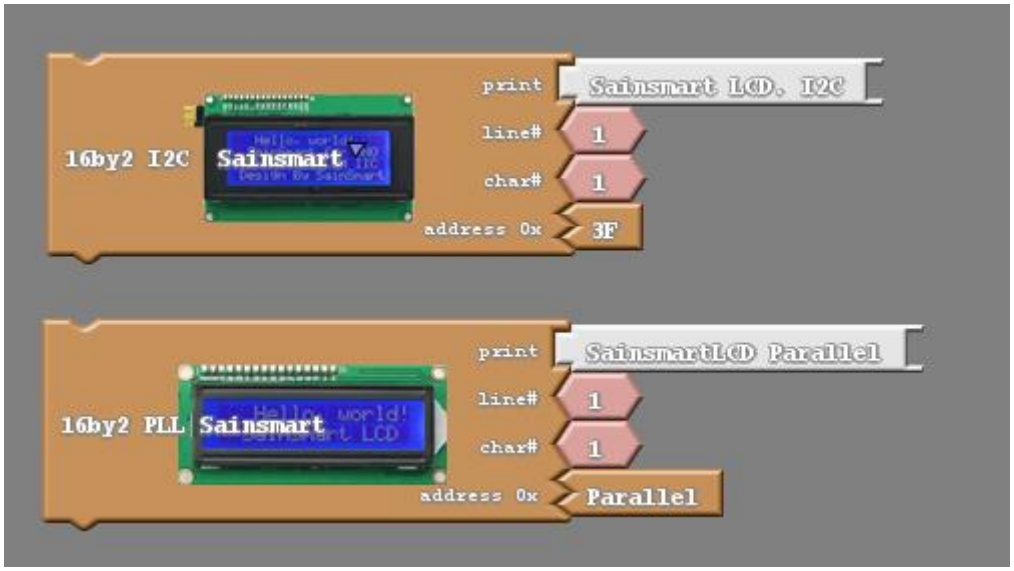
Kijk deze nog eens na of meet het na met een logic analyser.

## 2.10 Kan ik ook een 16 x 2 scherm aansturen via Ardublock?

Dit kan via de reeds uitgelegde **parallel versie** of via een **I2C versie van het LCD**. Bij deze laatste is het parallel gestuurde LCD eerst aangesloten op een I2C module. Daarna wordt deze pas aangesloten op de Arduino. De I2C versie bestuderen we **verder in deel 3 van deze Arduino cursus** voor gevorderden.

In Ardublock kan je het LCD scherm vinden in de "**generic hardware**" tool.

We hebben hier 2 versies:

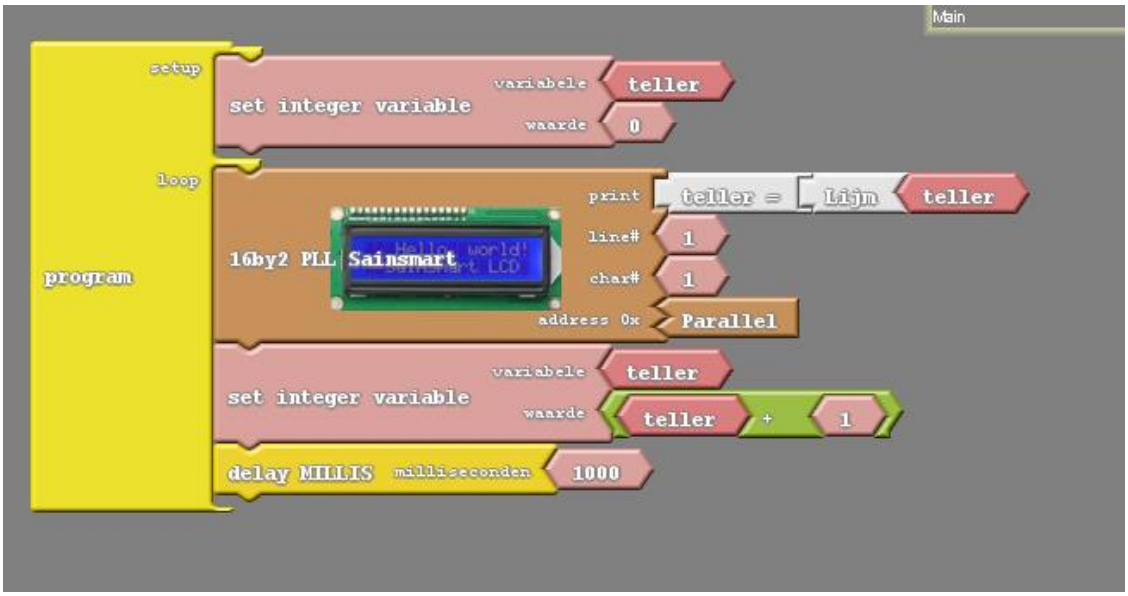



Het bovenste icoon geeft de **I2C versie** aan. Merk op dat je via een pijltje kan kiezen tussen een 4 x 20 en 16 x 2 LCD. Wij gebruiken hier de 2$^{de}$ versie, namelijk 16 x 2, als we de I2C omzet module zouden aansluiten (zie volgend hoofdstuk).

Het onderste icoon geeft de **parallel versie** van het LCD weer (met 4 datadraden) voor het 16 x 2 LCD.

De onderste **parallel versie** is net wat wij nu nodig hebben 😊

Nu kan je bijvoorbeeld een teller maken en deze tonen op jouw LCD scherm. Maak hiervoor volgende code in Ardublock.




Via line# en char# bepaal je de rij en de kolom waar het eerste karakter moet komen. In dit voorbeeld is dit links boven. Merk op dat ze in Ardublock niet vanaf nul beginnen te tellen, in c-code gebeurd dit wel. Daarvoor passen ze tijdens de download in c-code de waarde aan “-1”.

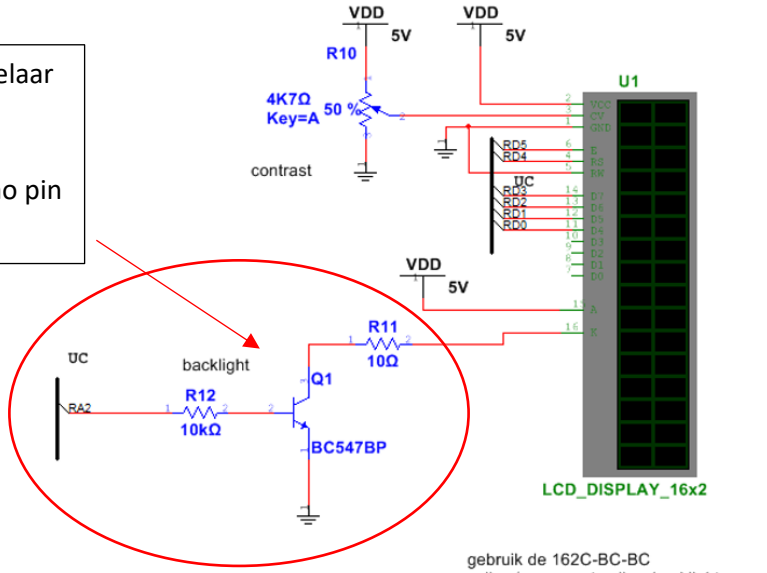
LCD_parallel_v2.abp

```
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal.h>

int _ABVAR_1_teller = 0 ;
// For these LCD controls to work you MUST replace the standard LCD library wit
// https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home
// Direct download https://bitbucket.org/fmalpartida/new-liquidcrystal/download
// Your project will not compile until this is done.
//
//                            RS  EN  d0  d1  d2  d3  LED
LiquidCrystal lcd_I2C_Parallel(12, 11,  5,  4,  3,  2,  7, POSITIVE);

void setup()
{
  lcd_I2C_Parallel.begin (16, 2);
  lcd_I2C_Parallel.setBacklight(HIGH);
  _ABVAR_1_teller = 0 ;

}

void loop()
{
  lcd_I2C_Parallel.setCursor( (5) - 1, (2) - 1 );
  lcd_I2C_Parallel.print( "teller =" );
  lcd_I2C_Parallel.print( _ABVAR_1_teller );
  _ABVAR_1_teller = ( _ABVAR_1_teller + 1 ) ;
  delay( 1000 );
}
```

Merk op dat je voor deze oefening ook nog de **achtergrond verlichting kan aansturen** via pin 7. Hiervoor moet je dan wel een **extra transistor toevoegen** in de schakeling, anders kan je nooit op een veilige manier de backlight aan/uitzetten. Zie voorbeeld in onderstaande schakeling.

Zie ook de "-1" bij de setCursor functie. Hierdoor zorgt Ardublock er toch voor dat het LCD scherm op 0,0 start i.p.v. 1,1

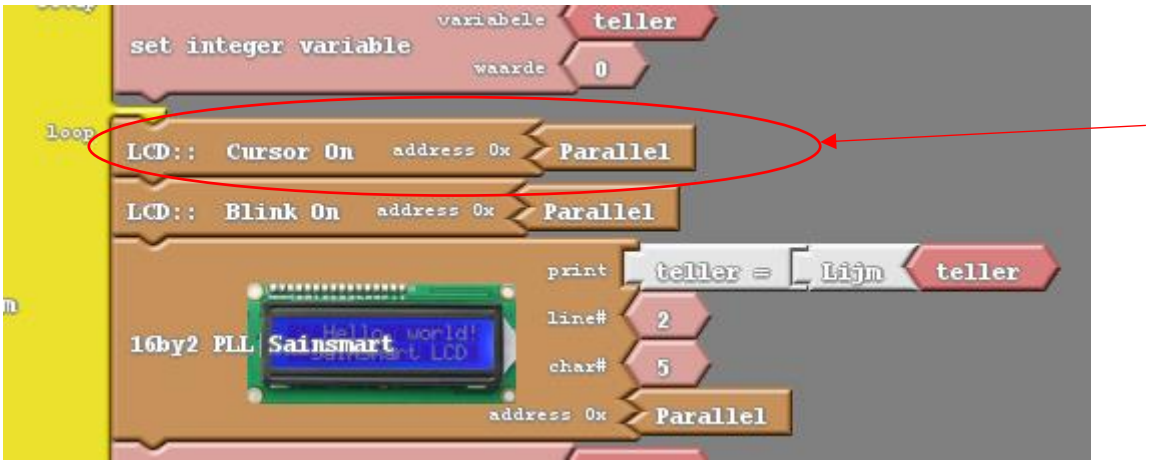
Transistor als schakelaar ingesteld om de backlight aan/uit te zetten via de Arduino pin 7

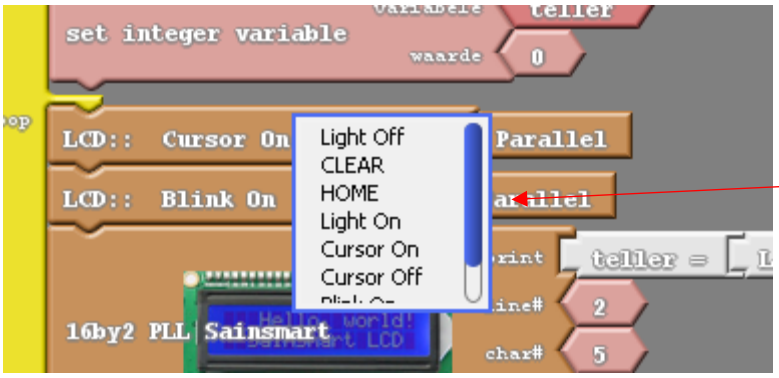



gebruik de 162C-BC-BC
yellow/green met yellow backlight
slechts 4 pinnen aangesloten voor data

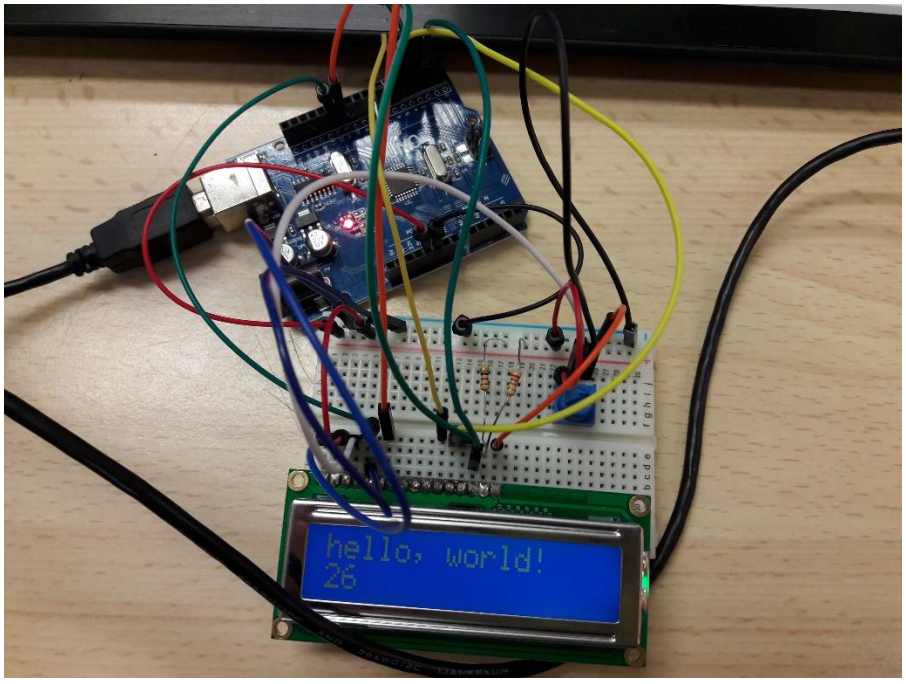
Ardublock kent ook **andere LCD functies** zoals: clear, blink, ….

Klik maar eens op het pijltje van de volgende icoon (ook in de generic hardware te vinden):






Vergeet niet van de het adres op “**parallel**” te zetten i.p.v. een I2C adres zoals “3F”.




Resultaat van een parallelle LCD opstelling via een Chinese UNO

## 2.11 Uitdagingen:

1. Laat een tekst naar keuze verschijnen op lijn 2.
2. Telkens je op S1 drukt komt de tekst “S1 = ON” op lijn 1.

   Druk je op S2 dan verschijnt op lijn 2 “S2 = ON” .
   Druk je beide knoppen dan verschijnt er “AND” op lijn 1.

3. Maak een optelsom en print deze op lijn 1 (vb 8 + 4). Schrijf het resultaat gecentreerd op lijn 2.

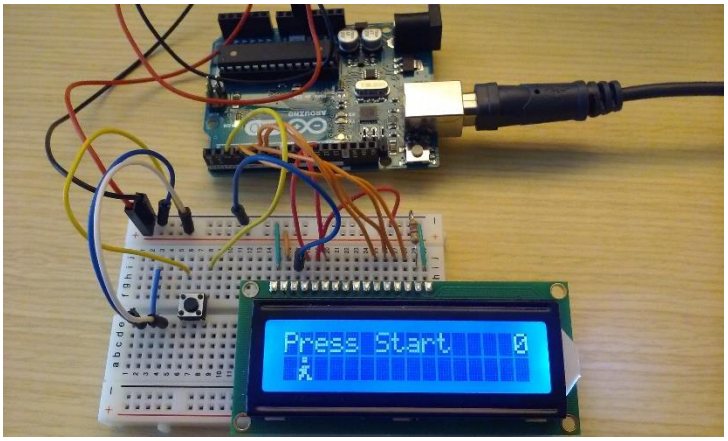
4. Uitbreiding op oef 3: telkens je drukt op S1 verhoogt het rechtse getal op lijn 1 met 2. Uiteraard veranderd dan ook het resultaat op lijn 2.
   Maak hier gebruik van een variabele = geheugenplaats.

5. Maak een secondenteller en toon de waardes op het LCD.

6. Maak een chronometer (start, stop, reset). Opmerking: omdat we geen TIMER interrupt gebruiken kan de tijd mogelijk een beetje afwijken. De oplossing hiervoor wordt aangeleerd in de Arduino gevorderden deel 2 cursus.

7. Gebruik het LCD in verdere oefeningen om waardes op het scherm te printen:
   vb de LDR, PWM of lijnvolger waarde. Het LCD kan net als de serial monitor gezien worden als een debugtool.

8. Ontwerp een spelletje en toon dit op het LCD. Gebruik bijvoorbeeld 2 knoppen om de cursor links en rechts te laten bewegen. Nu brandt de groene LED. Raakt deze de rand dan komt er “BOEM” op het LCD en een pieptoon. Dan brandt ook de rode LED. Je kan zelfs een motortje laten draaien wanneer de “BOEM” voorkomt. Schroef op de as een lusterklem. Zodoende krijg je een concentrische beweging en lijkt het of het spelletje trilt.
9. Test eens een arduino game (zie http://www.instructables.com/id/Arduino-LCD-Game/ )



## 3. LCD scherm aansturen met I2C

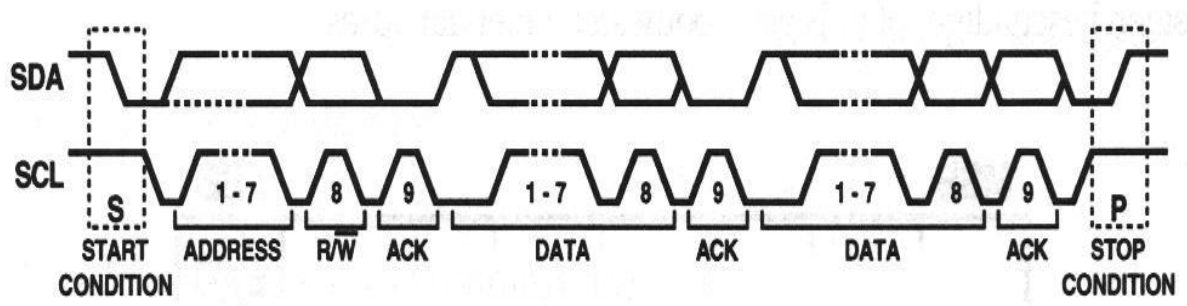
In dit deel gaan we uitleggen hoe we een LCD scherm kunnen aansturen via de I2C module.

Wanneer we te weinig pinnen ter beschikking hebben in ons project (een LCD scherm dat parallel wordt aangestuurd heeft al snel 4 data + 3 controle draden nodig), dan kunnen we ook van I2C gebruik maken. Hierbij zijn **slechts 2 draden (SDA en SCL)** nodig en kunnen we evengoed alles aansturen.

➔ Wat is I2C (Inter IC) voor een communicatie technologie?

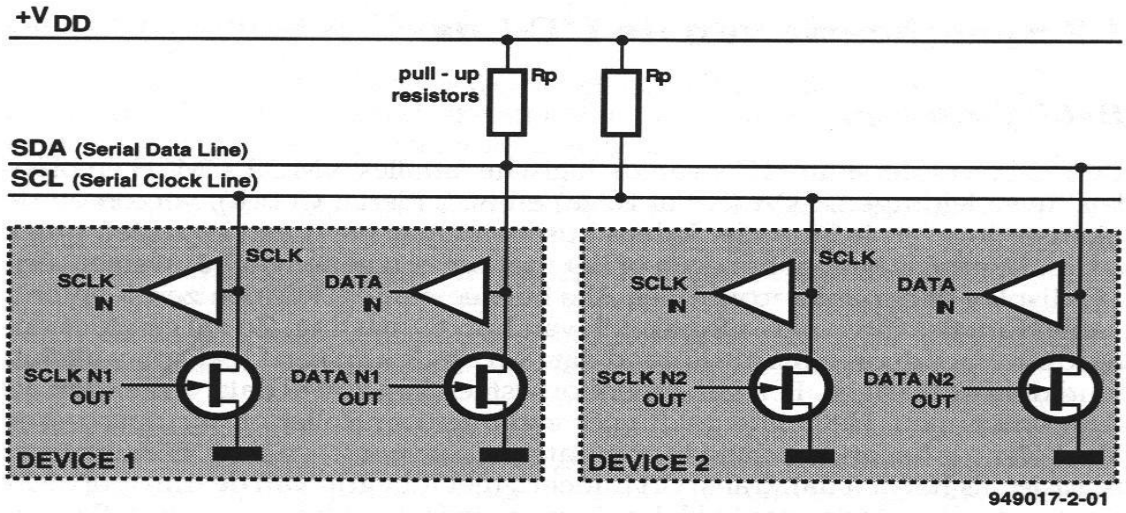
Philips heeft reeds in de jaren 80 van de vorige eeuw deze technologie uitgevonden. Om te voorkomen dat er **te veel draden** waren die op printbanen moesten gelegd worden, **overspraak** tussen draden ontstond of dat de draden voor **EMC** (elektro magnetische storingen) gevoelig waren heeft men deze techniek uitgevonden. We sturen de **data serieel** (bits achter elkaar over de bus) i.p.v. parallel (meerdere bits tegelijk).

We hebben 1 draad waar de clock op hangt (SCL). Deze wordt aangestuurd door de **master** (meestal de microcontroller). Hiermee wordt de data op de andere datalijn (SDA) netjes achter elkaar ingeklokt in de I2C chip aan de **slave** kant. Dit is een **synchrone** bus. De clock zit niet in de data verwerkt maar loopt synchroon met de data mee.




*Het verloop van een transfer.*

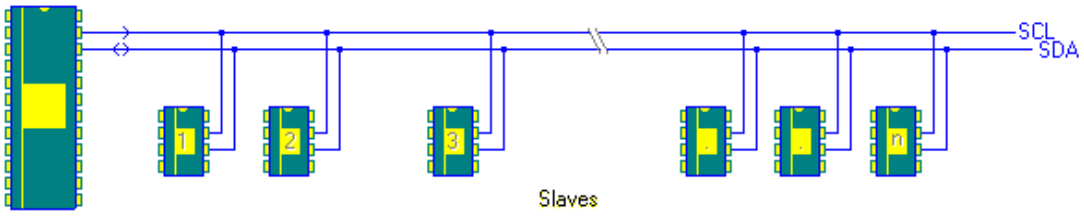
Belangrijk is dat we ook weten dat de **beide lijnen steeds** met een **pullup weerstand** van ongeveer 4K7 aan de +5V moeten hangen. Dit komt omdat de lijn **omlaag getrokken** wordt als je data wil versturen. Anders laat je de lijn hoog.




*De wired-AND-schakeling.*

De data draad wordt ook **bidirectioneel** gebruikt. D.w.z. dat de master iets kan sturen, maar dat de slave ook kan antwoorden door op zijn buurt de lijn even laag te maken (**ACK** = acknowledge). Merk op dat deze databus **halfduplex** werkt, d.w.z. dat je om de beurt iets kan zeggen op de bus (net als bij een **walky-talky**).

Elke I2C chip van een bepaalde toepassing heeft een **unieke nummer** gekregen van de fabrikant. Zo luisteren alle IC's, die parallel hangen op de draden, naar het **juiste adres (nummer van IC + ADRES van chip)**. Als dit voorbij komt dan reageert enkel de chip met hetzelfde adres.




De meest voorkomende I2C adressen voor ons **LCD scherm I2C omzetter** zijn: **0x27, 0x20 of 0x3F**

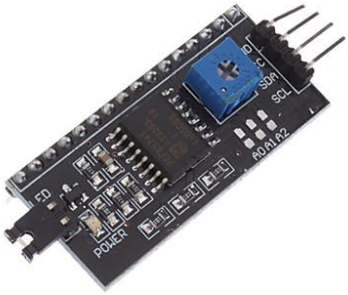
Die gaan we moeten invullen in ons programma.

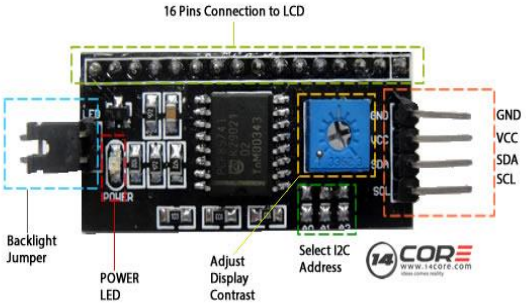
➔ Hoe het I2C adres ontdekken als je het niet weet?

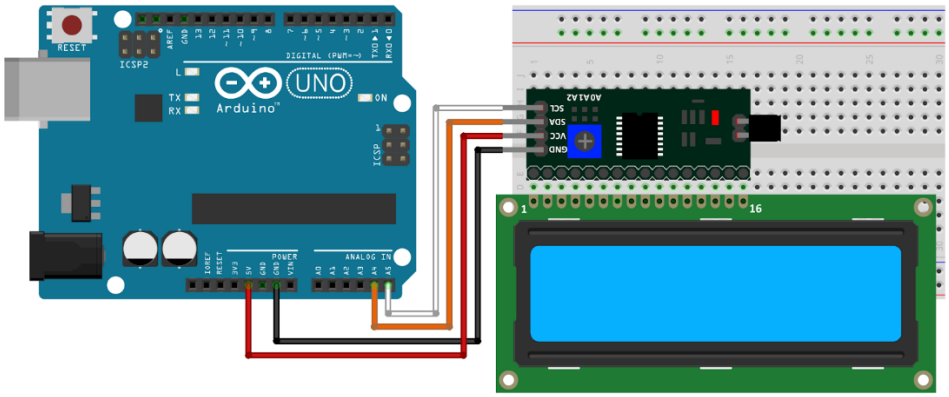
Indien deze niet werken kan je nog de **I2C adres scanner** gebruiken:

http://playground.arduino.cc/Main/I2cScanner

Download deze sketch in jouw Arduino, sluit de I2C pinnen aan en kijk via de **serial monitor** op **welk adres** het device is aangesloten. Dan kan je dit adres gebruiken in jouw Ardublock programma.

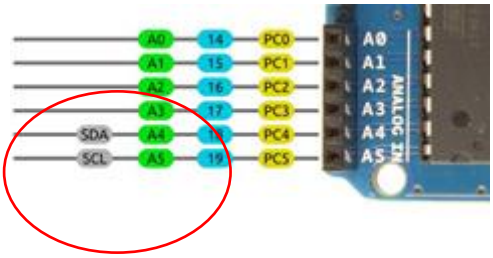




Merk op dat je gewoon de +5V en massa kan aansluiten tussen het LCD scherm en de UNO.

Daarnaast hangt de **SDA pin (data) op A4** van de UNO en de **SCL pin (CLK) op A5** van de UNO.



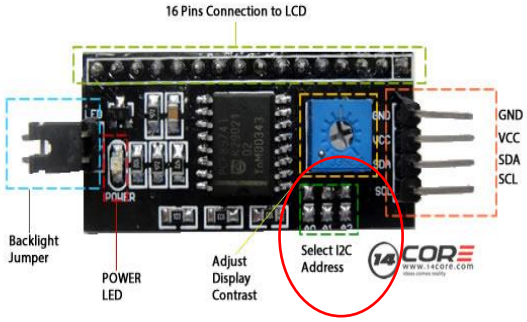

Op deze manier kan je ontdekken welk I2C adres je moet gaan gebruiken in de verdere code.

Je kan het **achtergrond licht (backlight) aan/uitzetten** via de jumper. Het **contrast** van het scherm kan je regelen via de blauwe potmeter op de achterkant.

Het resultaat van de scanner kan bijvoorbeeld zijn:

```
Scanning...
I2C device found at address 0x27  !
done
```

Opmerking: stel dat je **meerdere LCD schermen** wil aansturen via I2C, dan moet je elk scherm een ander adres geven. Dit kan door **brugjes te solderen** op het I2C bordje.




## Pin/Control Descriptions:

| Pin # | Name | Type | Description |
|---|---|---|---|
| 1 | GND | Power | Supply & Logic ground |
| 2 | VCC | Power | Digital I/O 0 or RX (serial receive) |
| 3 | SDA | I/O | Serial Data line |
| 4 | SCL | CLK | Serial Clock line |
| A0 | A0 | Jumper | Optional address selection A0 - see below |
| A1 | A1 | Jumper | Optional address selection A1 - see below |
| A2 | A2 | Jumper | Optional address selection A2 - see below |
| Backlight | | Jumper | Jumpered - enable backlight, Open - disable backlight |
| Contrast | | Pot | Adjust for best viewing |

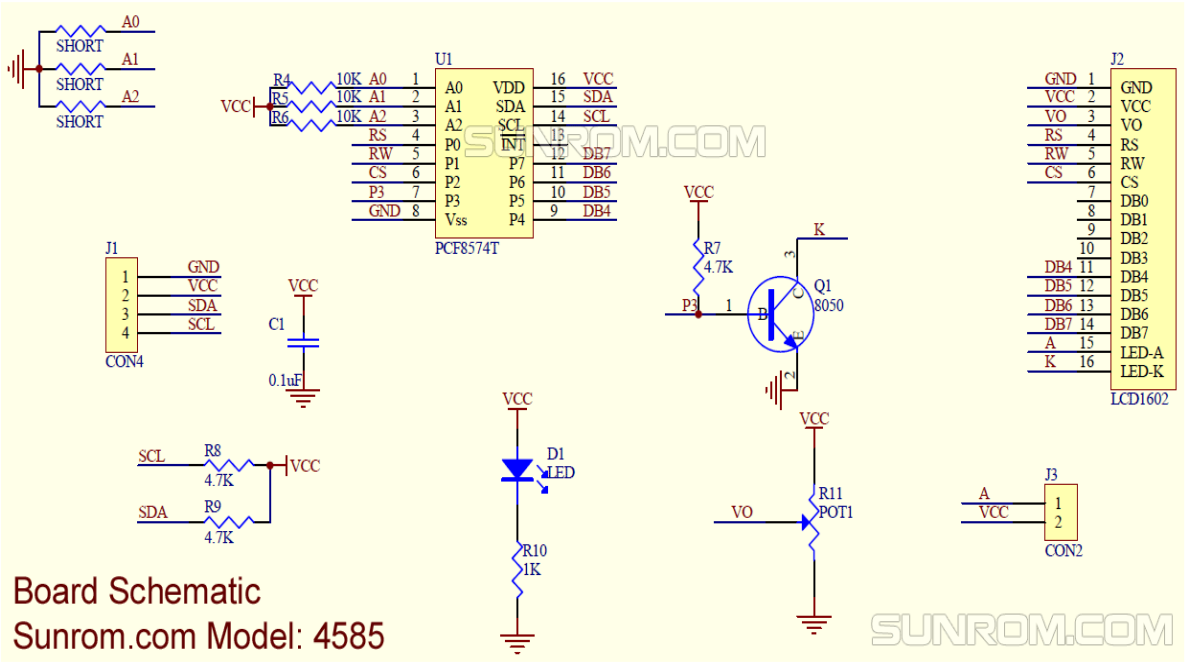
## Addressing:

| A0 | A1 | A2 | Address |
|---|---|---|---|
| Open | Open | Open | 0x27 |
| Jumper | Open | Open | 0x26 |
| Open | Jumper | Open | 0x25 |
| Jumper | Jumper | Open | 0x24 |
| Open | Open | Jumper | 0x23 |
| Jumper | Open | Jumper | 0x22 |
| Open | Jumper | Jumper | 0x21 |
| Jumper | Jumper | Jumper | 0x20 |

Als de adreslijnen **niet gesoldeerd** zijn worden ze via een weerstand op de **PCB hoog getrokken**.

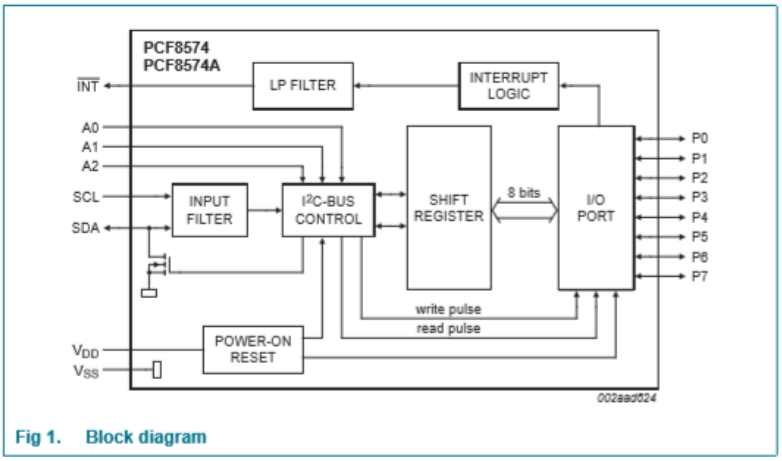
Opmerking 2: **er zitten al 2 pull-up weerstanden** op het I2C LCD printje gesoldeerd. Dus hier moet je geen rekening meer mee houden.

➔ Wat zit er in feite op het printje?




De **PCF8574** is een Philips I2C chip (of vergelijkbare) die gebruikt wordt **als IO expander.** Werkt op **100KHz** maximum snelheid en kan 8 pinnen als input of output aansturen. Werkt op 5V en kan maximum **80mA** (type afhankelijk) hebben verdeeld **over de 8 IO pinnen**. Liefst niet meer dan **10mA gebruiken per pin.**

Meer info zie datasheet.




Merk op dat de echte Philips chips een adres hebben dat begint met 0x**7**x . Onze versie start met 0x**2**x . Dus het is een andere fabrikant.

Je kan deze **chip PCF8574** ook **zelf via een library** leren aansturen. Zie meer info hier.

➔ Software voor ons I2C LCD scherm maken.

Op de volgende plek op www.arduino.cc kan je de laatste "**LiquidCrystal_I2C" library** downloaden en als zip file binnenhalen in Arduino.

https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/

Kies de laatste versie. Niet unzippen!

Voeg deze weer toe aan jouw bibliotheek in Arduino (zie vorig hoofdstuk hoe je dit moet aanpakken).

Nu schrijven we code om tekst op het LCD scherm te tonen:

HelloWorld

```
//Compatible with the Arduino IDE 1.0
//Library version:1.1
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
int teller = 0;

LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line display

void setup()
{
  lcd.init();                      // initialize the lcd

  lcd.backlight();//zet backlight aan
  lcd.setCursor(3,0);//kolom 3, lijn 0 , start tekst
  lcd.print("Hello, world!");
  lcd.setCursor(3,1);
  lcd.print("Arduino!");
  delay(1000);
  lcd.noBacklight();//zet backlight uit
  delay(1000);
}

void loop()
{
  lcd.clear();//wis scherm
  lcd.backlight();
  lcd.setCursor(3,0);
  lcd.print("teller = ");
  lcd.print(teller);
  delay(1000);
  teller++;
}
```

Merk op dat we in de code het juiste I2C adres ingeven: 0x27.

Zorg dat de 2 libraries Wire.h en LiquidCrystal_I2C.h zijn ingeladen in jouw Arduino voor je gaat compileren en uploaden.

De rest van de commando's zijn gelijkaardig aan die van het gewoon sturen van een LCD scherm.

Leuk is dat we nu ook de backlight kunnen aan en uit zetten via het aansturen van de transistor op P3.

| | Kolom 0 | | | | | | Kolom 7 |
|---|---|---|---|---|---|---|---|
| Lijn 0 | 0,0 | 1,0 | 2,0 | 3,0 | 4,0 | 5,0 | 6,0 | 7,0 |
| Lijn 1 | 0,1 | 1,1 | 2,1 | 3,1 | 4,1 | 5,1 | 6,1 | 7,1 |

Het commando setCursor (kolom, rij) is nog steeds hetzelfde als bij een gewoon LCD scherm.

Altijd van 0 beginnen te tellen.

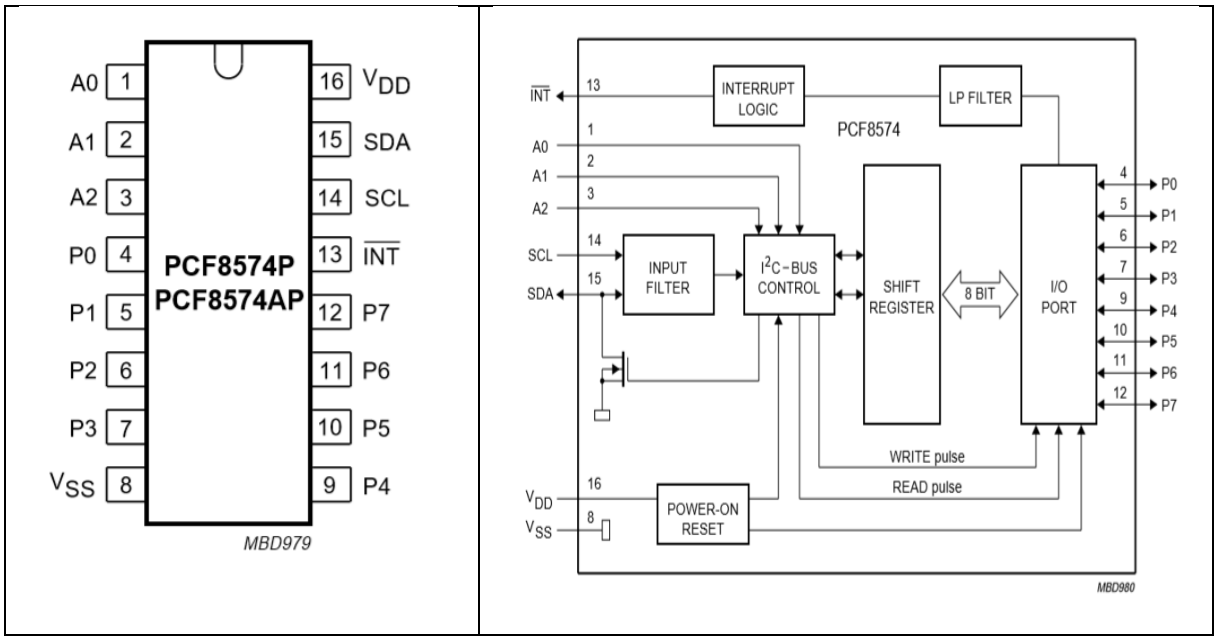
➔ **Uitdagingen:**

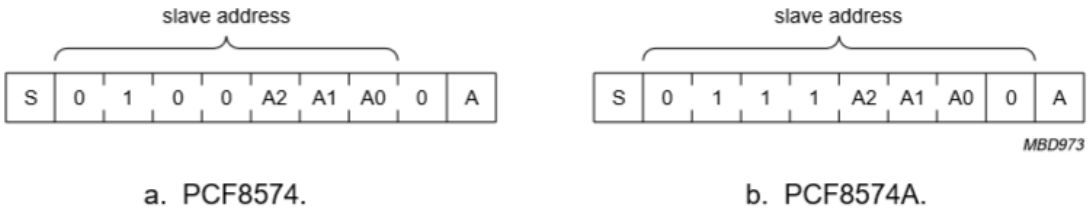
- Toon de PWM waarde van jouw RGB LED dimmer op het scherm
- Test de code uit waarbij je een karakter moet invullen in de serial monitor van Arduino. Dit wordt daarna getoond op het LCD scherm (zie oplossing op de USB stick bij LCD_I2C_test).
- Combineer een scherm met I2C in jouw project waar je weinig draden over hebt voor het display. Je kan ook debug waardes sturen naar het scherm om beter te begrijpen wat er gebeurd met jouw sensoren.

- **Extra: Onderzoek hoe de I/O expander** met I2C **ook kan ingezet worden voor andere projecten** dan met het LCD scherm. Wanneer je bijvoorbeeld op een knop drukt (aangesloten als laag actief op P2 of P3) moet er een laag actieve LED op dezelfde IO expander aangaan op P0 of P1.
  **Onderzoek eerst met de I2C scanner welk het adres is van de expander.** Vul dit dan in in de onderstaande code.
  **Doe ook een meting met de logic analyzer van het I2C signaal.**
  Zie schema en uitleg op volgende pagina's.

  Zie ook de website van instructables voor meer info en code!

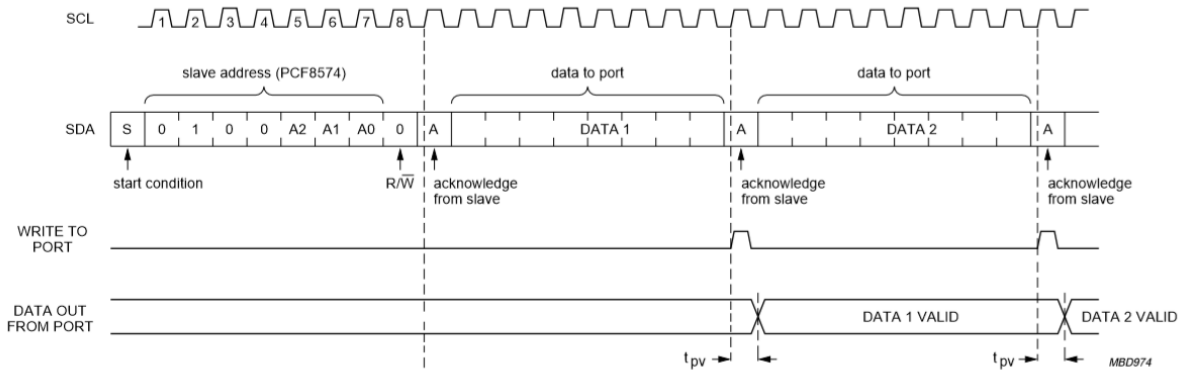
Chip:



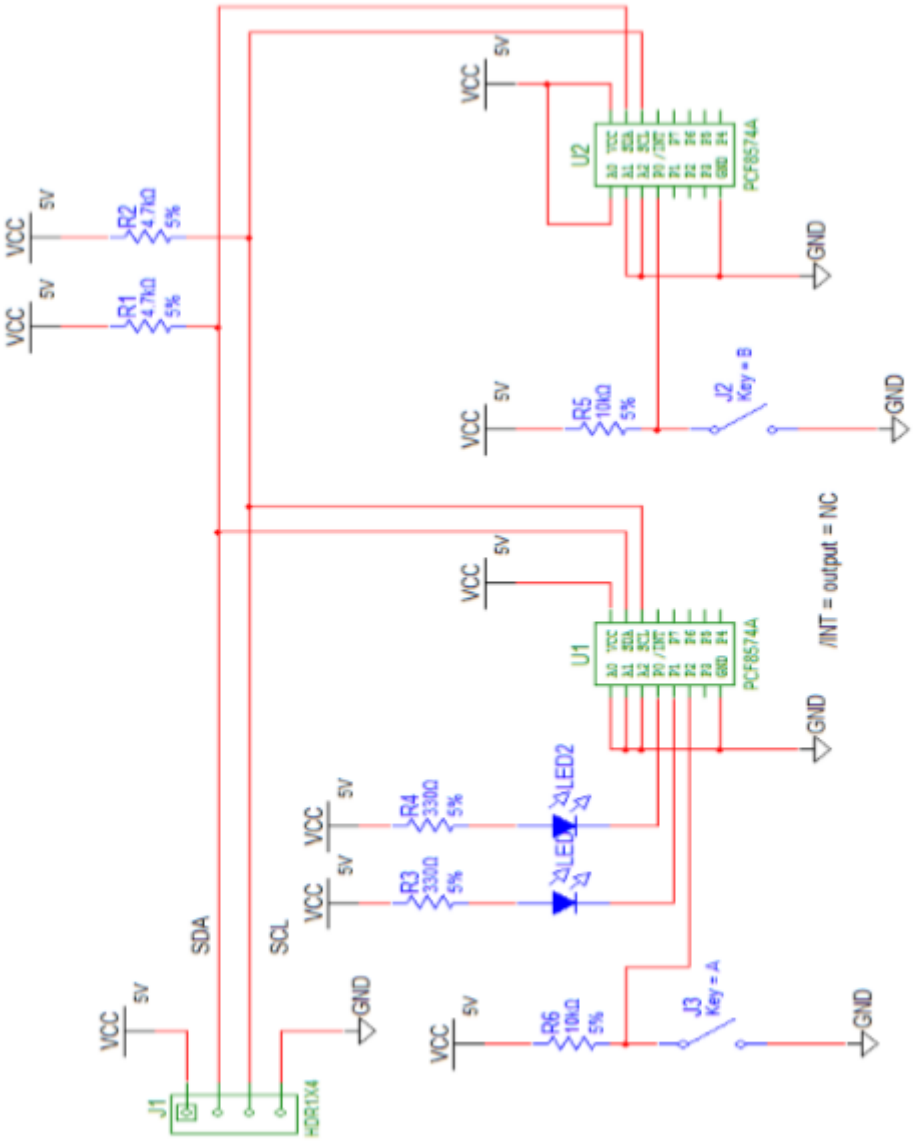




a. PCF8574. b. PCF8574A.

Adres? Stel A2A1A0 = 000 -> 0x70 (laatste bit op 0 = write)




Schema:




Voorstel van aansluitingen van LEDs en knoppen op de IO expander.

Vergeet de 2 pull-up weerstanden niet!

Aansluiten I2C op Arduino

SDA (pin 15 op PCF8574A) = A4 op Arduino

SCL (pin 14 op PCF8574A) = A5 op Arduino

i2c_pcf8574

```
/// Experiment van PCF8574 IO-expander

# include <Wire.h>
#define DEVICE_1 B0111000 //0x70  , werkt ook met 0x38  011 1000
                                  //omdat ze de R/W bit niet mee tellen
byte knop;

// Schrijf een byte naar de IO-uitbreiding
void IOexpanderWrite (byte byteadres, byte _data)
{
 Wire.beginTransmission (byteadres);
 Wire.write (_data);
 Wire.endTransmission ();
}

// Lees een byte van de IO-uitbreiding
byte IOexpanderRead (int adres)
{
 byte _data;
 Wire.requestFrom (adres, 1);
 if (Wire.available ()) {
   _data = Wire.read ();
 }
 return _data;
}

void setup ()
{
    Wire.begin ();
    IOexpanderWrite (DEVICE_1, 0xFF);
}

void loop ()
{

      knop = IOexpanderRead(DEVICE_1);

      if (knop == 0bllllllOll){//lees knop 1 op P2

          IOexpanderWrite (DEVICE_1, 0bllllllOl);     //stuur led op P1
          delay (500);
          IOexpanderWrite (DEVICE_1, 0bllllllll0);     //stuur led op P0
          delay (500);
      }

      else if (knop == 0bllll0llll){//lees knop  op P3

          IOexpanderWrite (DEVICE_1, 0b00000001);
          delay (200);
          IOexpanderWrite (DEVICE_1, 0b00000010);
          delay (200);
      }

      else {

          IOexpanderWrite (DEVICE_1, 0xFF);
      }

}
```

# 4. Een OLED scherm aansturen via I2C

Op dit moment zijn we reeds in staat om LCD (Liquid Crystal Displays) aan te sturen.

Ondertussen zijn er nieuwere schermpjes op de markt. De OLED's of **organic light-emitting diode.**

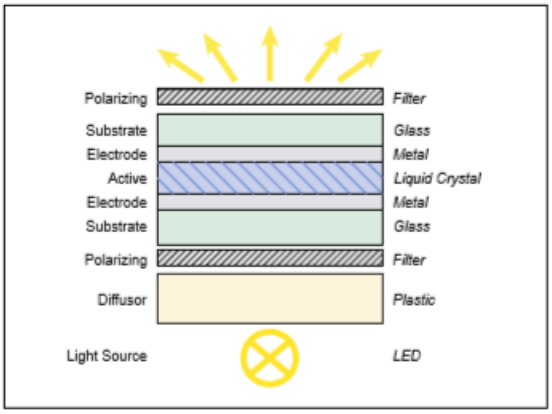
Deze schermpjes combineren alle **voordelen** van de LCD, LED en plasma schermen met elkaar:

- Geen response time (bij LCD moet de polarisatie fysiek echt omwisselen, kost veel energie, bij een OLED kan men tot 1KHz schakelfrequentie gaan)
- Onbeperkte kijkhoek (180 graden, dus vanaf de zijkant te bekijken)
- Natuurgetrouwe kleuren
- Contrast is beter (achterkant van de OLED is zwart, geen reflectie)
- Zeer platte schermen
- Je kan meerdere lijnen op 1 schermpje maken en zit niet vast aan de karakters van het LCD met elk hun pixels.
- Het begint betaalbaar te worden (zeker de kleine schermpjes)
- Het rendement van de OLED is beter, omdat bij een LCD scherm er nog veel energie in de achtergrondverlichting wordt gestopt.

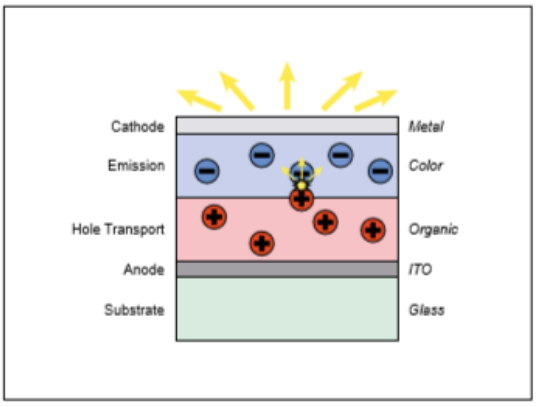
**Nadelen** aan OLED's?

- Stabiliteit van het organisch materiaal. Halfgeleiders verouderen gewoon langzamer en zijn robuster.
- Moeten nog goedkoper worden.

Het artikel over OLED's in het maandblad Elektor van januari 2018 vertelt heel wat over deze nieuwe schermpjes.




Figuur 1. Schematische opbouw van een LCD.




Figuur 2. Schematische opbouw van een OLED-display.

Bovenstaande foto's geven meer info over de opbouw van de OLED.

Meer info kan je ook [terugvinden op wiki.]

➔ Meer technische info over het OLED scherm

De OLED die we gaan gebruiken is een schermpje met **een 0.96 inch** diagonale.

Deze heeft een **128 x 64 pixels scherm** en werkt met I2C.

De MCU chip in het schermpje is de **SSD1306**.

In dit geval is het geen kleurenscherm maar zijn de **letters wit**. Op de markt kan je ook meer kleuren versies vinden.

De makers van de SSD1306 (driver van de OLED) zijn **Solomon Systech.**

**128 x 64, Dot Matrix OLED/PLED**
**Segment/Common Driver with Controller**

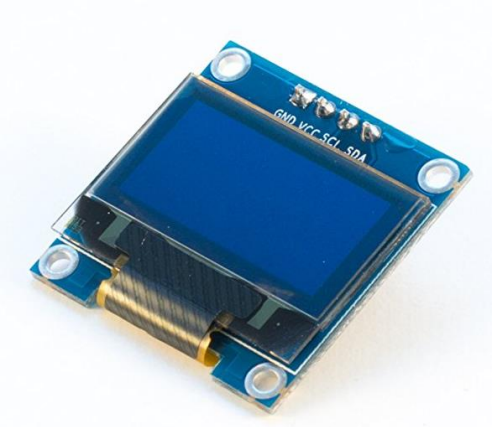
**CMOS**

SSD1306 is a single-chip CMOS OLED/PLED driver with controller for organic / polymer light emitting diode dot-matrix graphic display system. It consists of 128 segments and 64 commons. This IC is designed for Common Cathode type OLED panel.

The SSD1306 embeds with contrast control, display RAM and oscillator, which reduces the number of external components and power consumption. It has 256-step brightness control. Data/Commands are sent from general MCU through the hardware selectable 6800/8000 series compatible Parallel Interface, I2C interface or Serial Peripheral Interface. It is suitable for many compact portable applications, such as mobile phone sub-display, MP3 player and calculator, etc.

Features

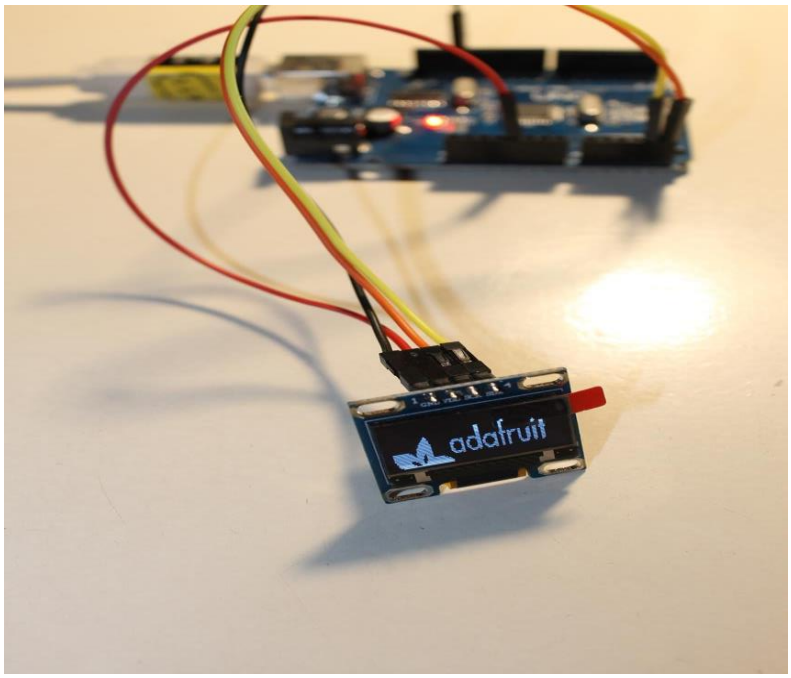
- Resolution: 128 x 64 dot matrix panel
- Power supply
  - VDD= 1.65V - 3.3V, <VBAT for IC Logic
  - VBAT= 3.3V - 4.2V for charge pump regulator circuit
  - VCC= 7V - 15V for Panel driving
- For matrix display
  - Segment maximum source current: 100µA
  - Common maximum sink current: 15mA
  - 256 step contrast brightness current control
- Embedded 128 x 64 bit SRAM display buffer
- Pin Selectable MCU Interfaces:
  - 8-bit 6800/8080-series parallel interface
  - 3/4-wire Serial Peripheral Interface
  - I2C interface
- Screen saving continuous scrolling function in both horizontal and vertical direction
- Internal charge pump regulator
- RAM write synchronization signal
- Programmable Frame Rate and Multiplexing Ratio
- Row Re-mapping and Column Re-mapping
- On-Chip Oscillator
- Chip layout for COG & COF
- Wide range of operating temperature -40°C to 85°C




Je kan de I2C draden ook hier terugvinden: SDA en SCL.

Sluit de **SDA weer aan op A4 en SCL op A5** van de UNO.

De voeding is 5V (VCC) of 3V3 als je een adafruit OLED hebt !




➔ Neem opnieuw **het I2C scanner programma** en check naar welk adres jouw schermpje luistert.

Normaal krijg je dit: **0x3C**

```
I2C Scanner
Scanning...
I2C device found at address 0x3C  !
done
```

➔ Nu moeten we de juiste library toevoegen van de SSD1306. We gebruiken deze van ADAFRUIT.
Je kan eerst gaan kijken in jouw Arduino of de libraries al geïnstalleerd zijn.
Ga weer naar **Schets -> bibliotheken gebruiken -> bibliotheken beheren**
Zoek op de term **SSD1306.**


Je zal de **ADAFRUIT SSD1306** zien staan. Mogelijk moet je nog even op "**installeren**" drukken.

Daarna staat er in het blauw "installed".

Belangrijk is dat we voor de OLED sturing de library **adafruit_ssd1306-master** en **adafruit-gfx-library-master** toevoegen aan Arduino.

Je kan ze via deze link vinden.

Voeg ze toe zoals reeds eerder zip files zijn toegevoegd als libraries.

Opmerking: je kan deze libs ook terugvonden onder de naam Adafruit GFX in bibliotheek beheer. Deze zouden stabieler en makkelijker werken dan deze aangegeven via de link.

Nu kunnen we ons startvoorbeeld inladen in Arduino. Het ADAFRUIT voorbeeld is reeds mee ingeladen via de zip file van de libraries.

Dit kan je nu terugvinden in jouw Arduino omgeving **bij bestand -> voorbeelden -> adafruit_ssd1306 -> ssd1306 128x64 I2C**

Klik er op en laat de code in.

Dit voorbeeld bevat heel wat verschillende commando's om allerlei figuren op het schermpje te tonen.

We starten met **een punt en het logo van Adafruit**. De rest van de code zetten we in commentaar via de /* …. */

We moeten in de originele code ook nog 2 dingen aanpassen!

**Het I2C adres** is geen 0x3D maar **0x3C**

Pas dit aan in de code:

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
```

We moeten ook de **error afzetten**(in tekst) die ze in de code als veiligheid hebben ingebouwd:

```
/*
#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif
*/
```

Ons scherm is 64 pixels hoog, dus dat zit goed.

ssd1306_128x64_i2c_test

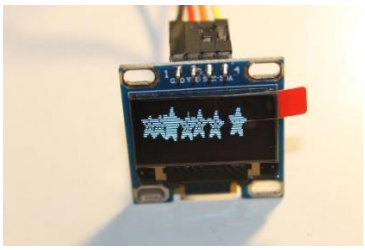
```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2


#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH  16
static const unsigned char PROGMEM logo16_glcd_bmp[] =
{ B00000000, B11000000,
  B00000001, B11000000,
  B00000001, B11000000,
  B00000011, B11100000,
  B11110011, B11100000,
  B11111110, B11111000,
  B01111110, B11111111,
  B00110011, B10011111,
  B00011111, B11111100,
  B00001101, B01110000,
  B00011011, B10100000,
  B00111111, B11100000,
  B00111111, B11110000,
  B01111100, B11110000,
  B01110000, B01110000,
  B00000000, B00110000 };
```

Wire.h voor de I2C chips
De 2 speciale adafruit files om niet alles van nul te moeten maken.

Reset scherm (ook al gebruiken we pin 4 niet met dit scherm?) en verbind “display” met de adafruit bib

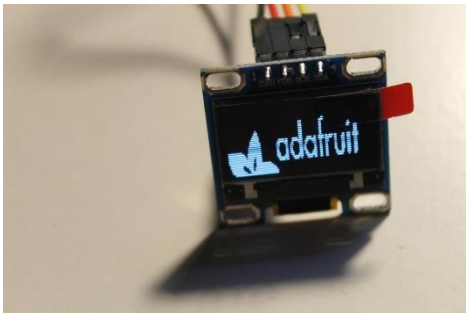
Dit is een bitmap van een **ster**. Je kan deze vervangen door andere bitmaps.

Er zijn ook tools om makkelijk bitmaps te maken van een figuur, zoals **LCD assistant**. Klik eens op deze link.



Vergeet niet vooraf jouw bmp file als **monochrome file** in **paint te saven**. Dan pas omzetten!

```
/*
#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif
*/

void setup()   {

  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3C (for the 128x64)
  // init done

  // Show image buffer on the display hardware.
  // Since the buffer is intialized with an Adafruit splashscreen
  // internally, this will display the splashscreen.
  display.display();
  delay(2000);

  // Clear the buffer.
  display.clearDisplay();

  // draw a single pixel
  display.drawPixel(10, 10, WHITE);
  // Show the display buffer on the hardware.
  // NOTE: You _must_ call display after making any drawing commands
  // to make them visible on the display hardware!
  display.display();
  delay(2000);
  display.clearDisplay();
```




In dit 2de deel van de code zetten we het **I2C adres** juist: 0x3C en zetten we de **juiste spanning** aan in de OLED.

Blijkbaar heeft de OLED een buffergeheugen waar je een foto kan in opslaan, in dit geval heeft **Adafruit zijn logo** hierin gestopt.

We kunnen dit na het tonen wissen en zetten dan 1 punt op het scherm.

Merk op dat je telkens eerst jouw tekening of tekst maakt, en het dan pas op het display toont met **display.display();**

Ik heb de andere code even uitgezet en test alleen de vallende sterren 😊

```
  // draw a bitmap icon and 'animate' movement
  testdrawbitmap(logo16_glcd_bmp, LOGO16_GLCD_HEIGHT, LOGO16_GLCD_WIDTH);
}

void loop() {

}
```

➔ **Uitdaging:** Test zeker zelf de code eens uit en **zet enkele lijnen aan en uit.** Zo kan je al de commando's leren gebruiken. Zie bijvoorbeeld het test voorbeeld waar enkel en ster getoond wordt of de code hieronder waar we met tekst spelen.

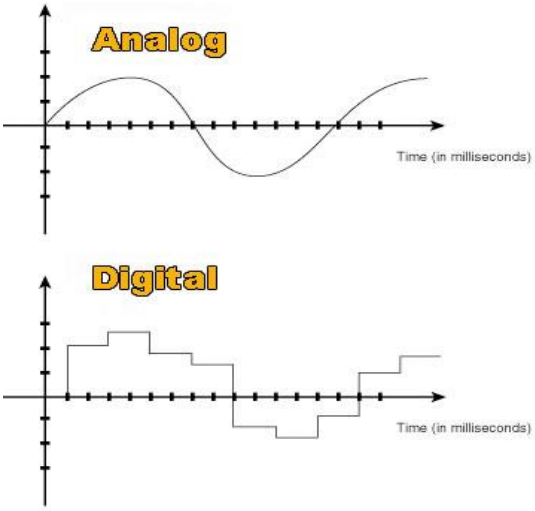
ssd1306_128x64_i2c_testv3_tekst

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>


#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);


void setup()   {

  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3C (for the 128x64)
  // init done

  display.display();
  delay(2000);

  // Clear the buffer.
  display.clearDisplay();


    // text display tests
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.println("Hello, world!");
  display.setTextColor(BLACK, WHITE); // 'inverted' text
  display.println(3.141592);
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.println("EDULAB");
  display.display();
  delay(2000);
  display.clearDisplay();


  // invert the display
  display.invertDisplay(true);
  delay(1000);
  display.invertDisplay(false);
  delay(1000);
  display.clearDisplay();

//scroll text
testscrolltext();


}


void loop() {

}
```

# 5. Werken met analoge ingangen

Wanneer we het aansturen van het LCD onder de knie hebben, kunnen we nu nog een stapje verder gaan, namelijk waardes van o.a. analoge ingangen tonen op het LCD scherm. Ook andere waardevolle info kan getoond worden op een LCD scherm om bijvoorbeeld code te debuggen.

## 5.1 Wat is het verschil tussen digitaal en analoog?




Zoals de bovenstaande figuur al aangeeft is er wel degelijk een groot verschil tussen analoge en digitale signalen.

Bij een analoog signaal telt de waarde van 0 tot een max vastgelegde waarde. Bij een Arduino kan je zo de referentie spanning instellen (Aref). Deze is standaard ingesteld op +5V.

Wanneer je dan met een analoge functie **analogRead()** gaat werken, dan krijg je een getal terug tussen 0 en 1023. Dit komt omdat de **ADC** (analoge digitale converter) een **10-bitter** is. 2^10 = 1024.

Dus het **kleinste stapje** van de analoge ingang van de Arduino is 5V / 1024 = **5 millivolt.**

Dit noemen we ook de **resolutie**.

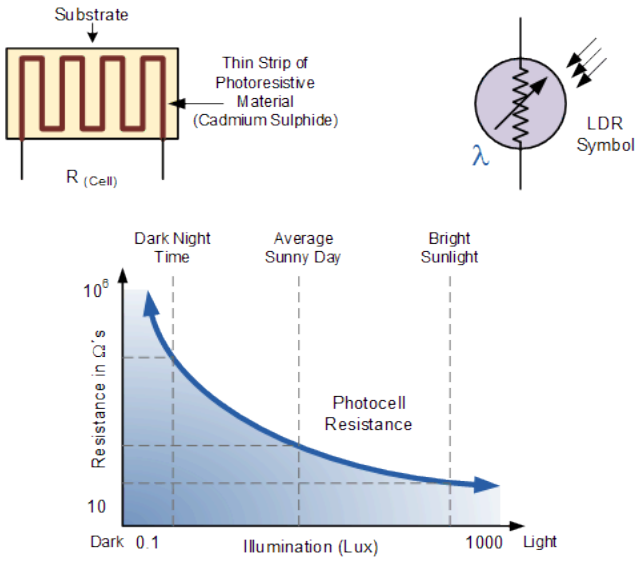
De **digitale ingang** heeft geen tussenwaardes. Het is 0V of 5V. Dus de ingang is **'0' of '1'** en de waarde kan in 1 bit gestopt worden.

Typische voorbeelden voor een **analoog signaal** zijn de waarde van een LDR (licht gevoelige weerstand), een NTC (negatieve temperatuurs coëfficient sensor) of een potentiometer.

Hiermee gaan we nu verder aan de slag om schakelingen te bouwen.

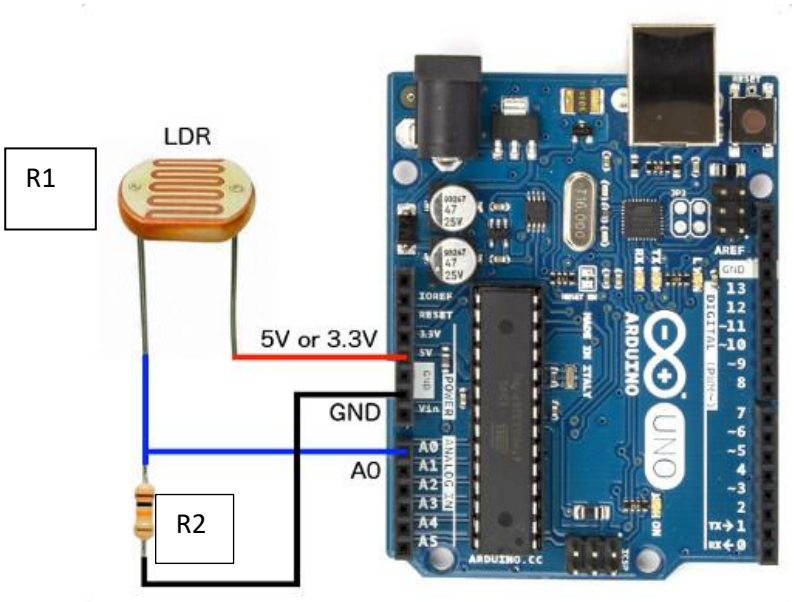
## 5.2 Een LDR inlezen en tonen op het PC scherm

Hoe werkt een LDR (light-dependent resistor)?




De figuur geeft de werking aan: als er **veel licht valt** op de sensor gaat de **weerstand dalen**.

Nu hangt het er vanaf hoe de LDR is aangesloten in de kring.

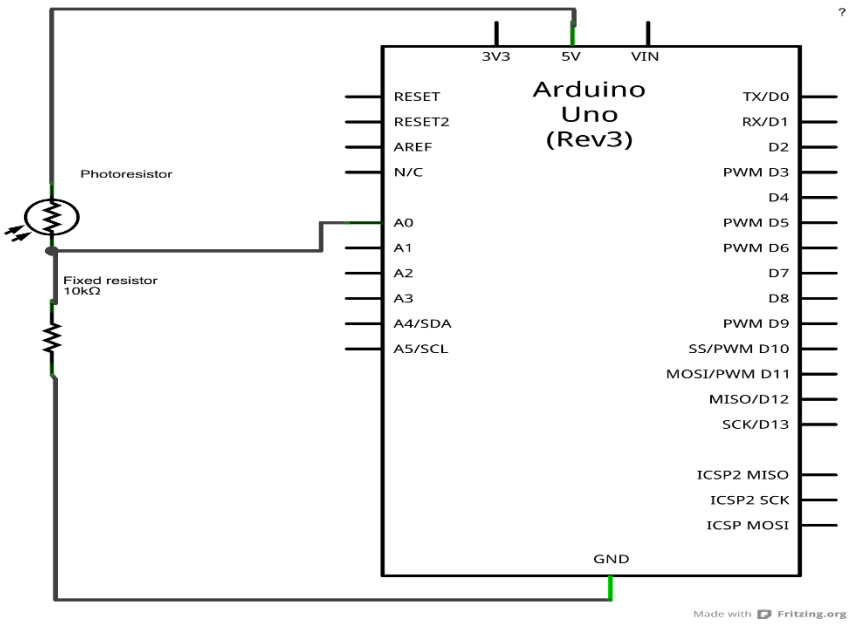



Wanneer de LDR aan de +5V is aangesloten en een pull-down weerstand van 10K zit aan de massa, dan kunnen we zeggen dat wanneer er **meer licht** valt op de LDR, de **spanning op de ingang van de microcontroller daalt.**

De formule achter de schakeling is een spanningsdeler:

$$Vout=Vin*(R2/(R1+R2))$$

Waarbij Vin = 5V, R1 = LDR, R2 = 10K




Schema van de aansluitingen van de LDR op de Arduino

## 5.3 Code om de LDR in te lezen

De volgende code leest de LDR in op pin A0 en toont de waarde op de **serial monitor.**

LDR_test

```
int LDR = 0 ;
int pin_LDR = A0;//enkel een cijfer is ook voldoende

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  LDR = analogRead(pin_LDR) ;
  Serial.print("LDR =");
  Serial.print(LDR);
  Serial.println();
  delay( 1000 );
}
```

COM18

```
LDR =368
LDR =378
LDR =386
LDR =388
LDR =387
LDR =385
LDR =382
LDR =384
LDR =815
```

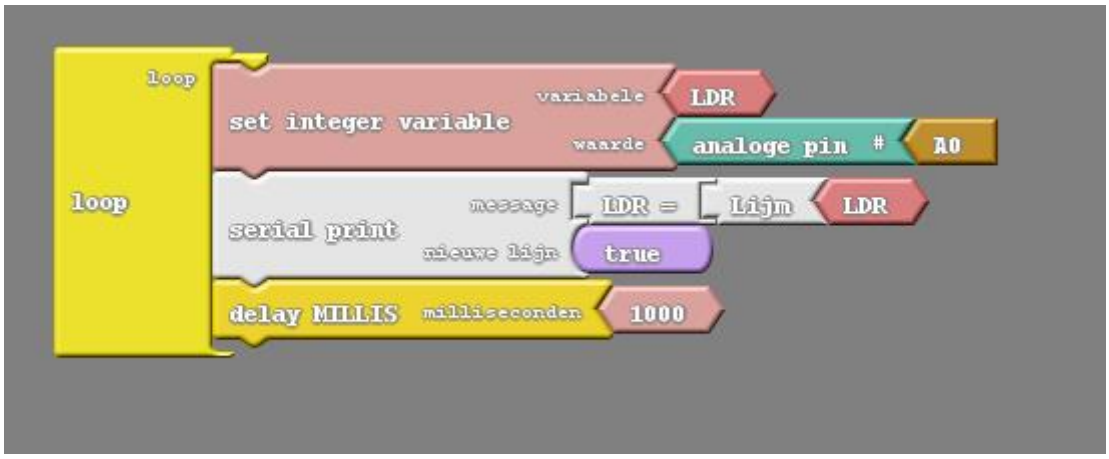
Merk op dat we nu **analogRead()** gebruiken i.p.v. **digitalRead()**

Test eens uit wat er gebeurd als je meer of minder licht op de LDR plaatst.

Normaliter moet de **waarde dalen** als je **minder licht** voorziet. Minder licht geeft dat de LDR weerstand stijgt (R1 in de noemer) en zodoende Vout daalt.

De waarde op de serial monitor zit **tussen 0 en 1023** (denk aan de 10 bits)

In Ardublock ziet de code er dan zo uit:




We lezen in de bovenstaande code **pin A0** uit. In de oudere Ardublock versie moet je **0 schrijven i.p.v. A0**, want anders lukt het niet. In de laatste versie is dit geen probleem en kies je A0.

Dan wordt deze in een variabele LDR gestopt. Deze wordt dan naar jouw PC scherm gestuurd om de seconde.

## 5.4 Een potentiometer inlezen en op de serial monitor tonen

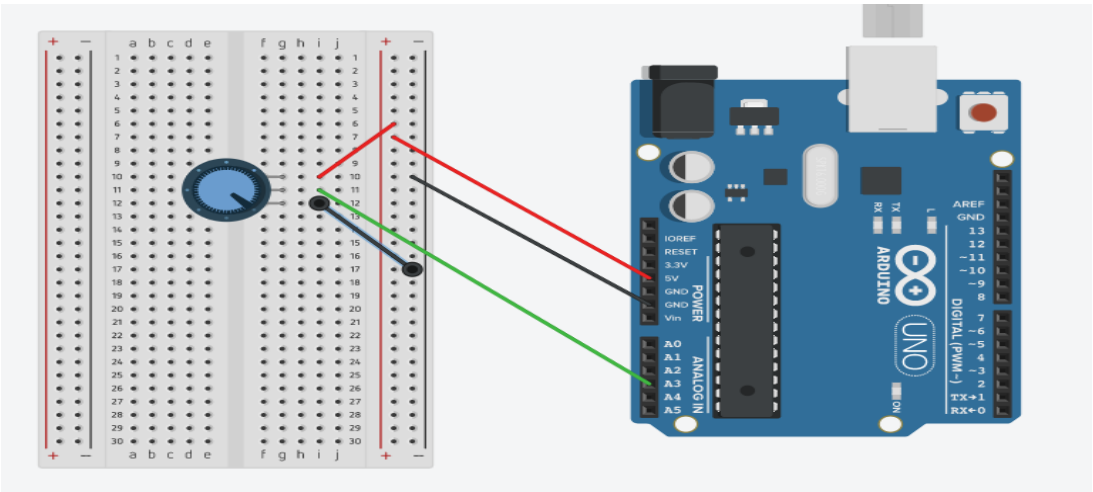
Nu gaan we een schakeling maken waarbij we de potmeter gaan gebruiken om de tijd in te stellen dat een LED gaat knipperen.

**Eerst** kijken we even of de schakeling werkt en **printen we de potmeterwaarde** af op de **serial monitor.**

We sluiten de **loper** van de potmeter aan op pin **A3** (dit is de middelste pin).

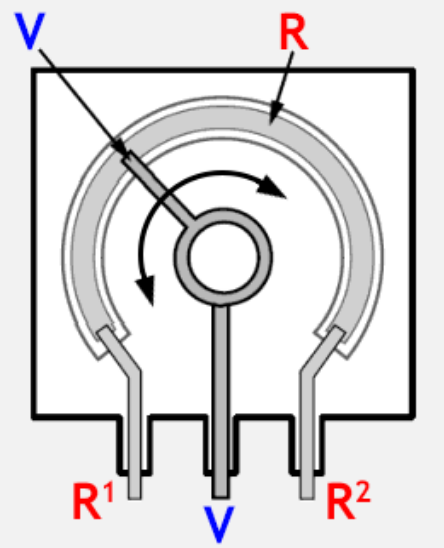
Eén zijde zit aangesloten aan de **GND** en de andere zijde aan de **+5V.**

Zo kunnen we een spanning tussen 0V en +5V regelen met onze potmeter.



**Hoe werkt de pot(entio)meter?**

Dit is een regelbare weerstand. Je kan deze regelen van 0 ohm tot de maximum waarde.

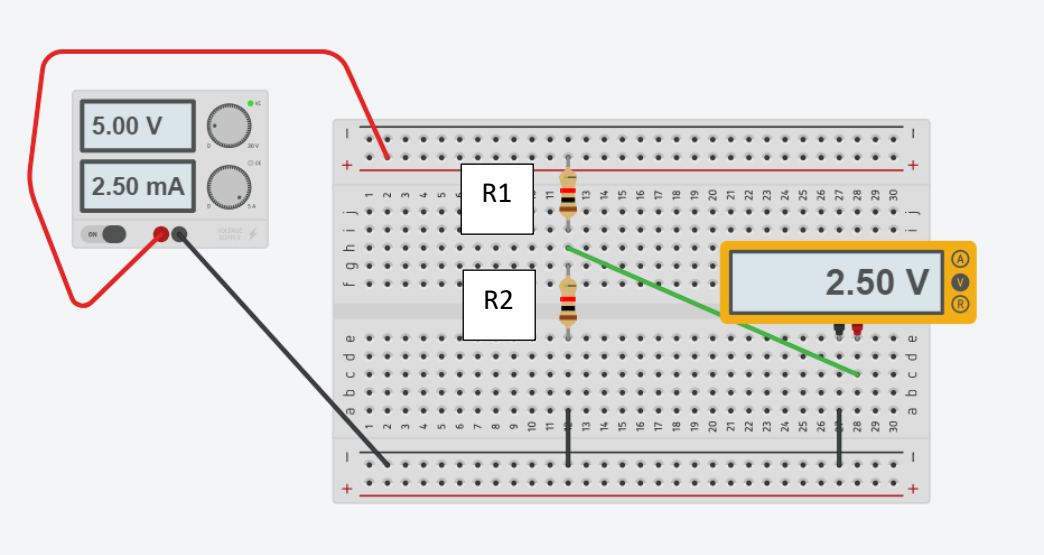



Op de bovenstaande figuur kan je zien dat de potmeter met de totale waarde R via de loper (het middelste pootje) **opgedeeld wordt in 2 weerstandswaarden** R1 (afstand R1 pin tot loper) en R2 (afstand R2 pin tot loper).

**R = R1 + R2**

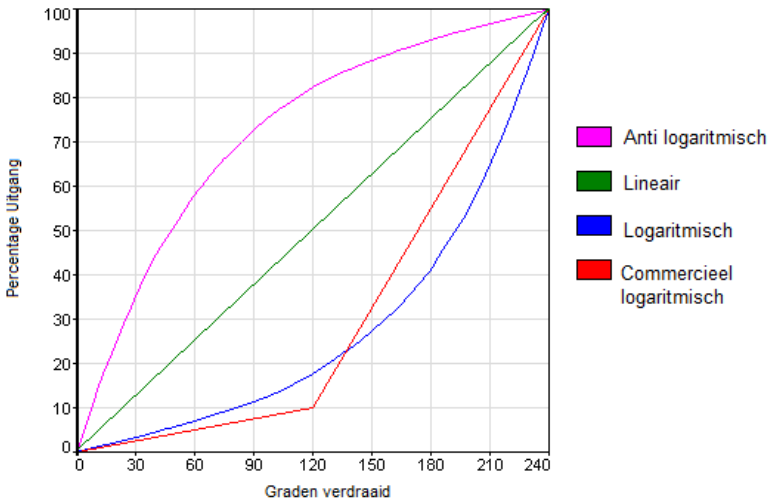
Wanneer we nu spanning aanleggen aan R2 (GND) en R1 (+5V), dan vormen de 2 weerstanden in de potmeter een spanningsdeler.

**V = 5V * (R2/R1 + R2)**




Voorbeeld potmeter opgedeeld in 2 echte weerstanden: 5 x (1K / 1K + 1K) = 2.5V

Zo kunnen we dus de loper van plaats schuiven. Dan veranderen de R1 en R2 in verhouding tot elkaar en kunnen we zo V gebruiken bij onze microcontroller.




Merk op dat een potmeter op verschillende manieren kan opgebouwd zijn. Wij gebruiken meestal een **lineaire potmeter** (LIN in potmeter gedrukt). Bij elke verandering zal de uitgang evenredig mee veranderen met de hoekverdraaiing van de as t.o.v. de beginpositie (zie groene curve).

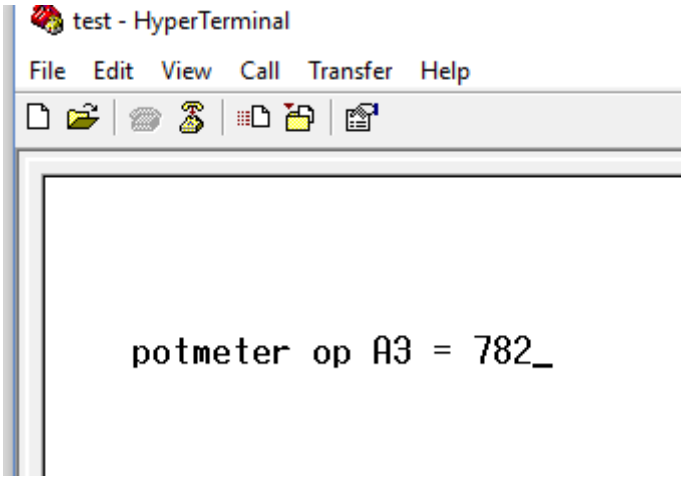
Je kan voor o.a. audio versterkers ook een **logaritmische potmeter** (LOG in potmeter gedrukt) gebruiken (zie blauwe curve). We gebruiken deze potmeter eerder bij een audioversterker omdat het menselijk oor ook logaritmisch is. Verdere info moet je maar eens googelen 😊

https://nl.wikipedia.org/wiki/Potentiometer

potmeter_serial_monitor

```
int pot = A3; //analoge ingang A3
int value = 0;

void setup() {

  Serial.begin(9600);
  Serial.write(27);//ESC
  Serial.print("[?25l");//maak cursor onzichtbaar
  Serial.write(27);//ESC
  Serial.print("[2J");//wis gans scherm

  //merk op dat de analoge ingang niet via pinMode wordt ingesteld!
}

void loop() {

    value = analogRead(pot);//getal tussen 0 en 1023 inlezen

    Serial.write(27);//ESC
    Serial.print("[1K");//veeg alles van begin regel tot aan de cursor uit
    Serial.write(27);//ESC
    Serial.print("[5;5H");//zet cursor op positie 5,5
    Serial.print("potmeter op A3 = ");
    Serial.print(value);//toon analoge waarde op scherm

    delay(1000);
}
```

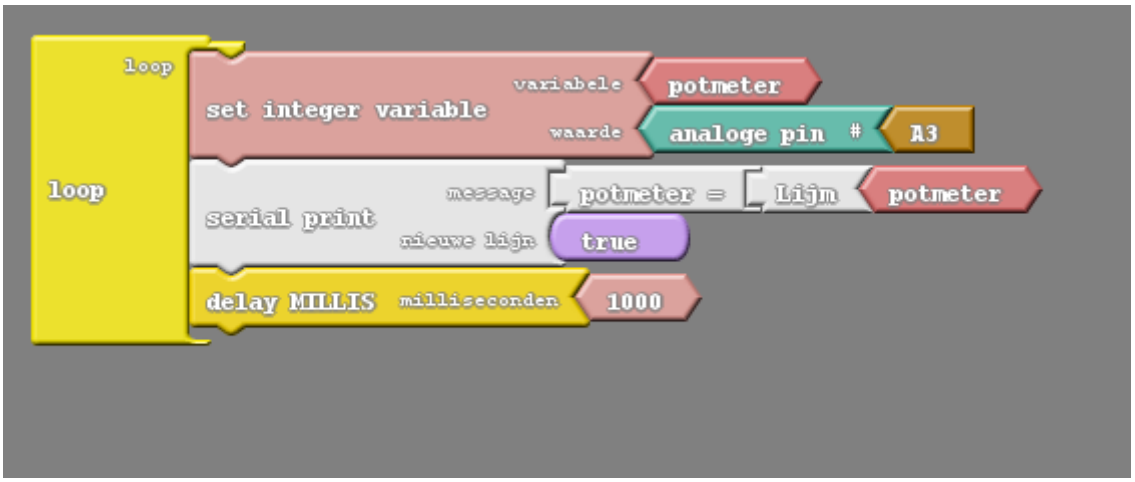
Download deze code in de Arduino.



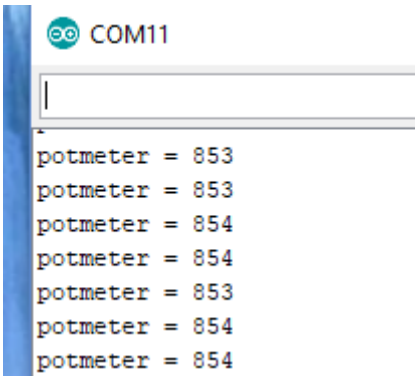

**Opmerking:** De serial monitor van Arduino gaat blokjes tonen en fouten maken omdat deze de ESC commando's niet kent (zie deel 1 van de basiscursus, serial monitor hoofdstuk). Gebruik daarom de **hyperterminal private edition** als je deze speciale commando's wil gebruiken.

Nu wordt de tekst steeds netjes herhaald en staat hij niet meer links in de bovenhoek.

Zonder ESC commando's ziet de code er in **C-code** / **Ardublock** als volgt uit:




De potmeter is aangesloten op A3 en wordt getoond op de serial monitor.



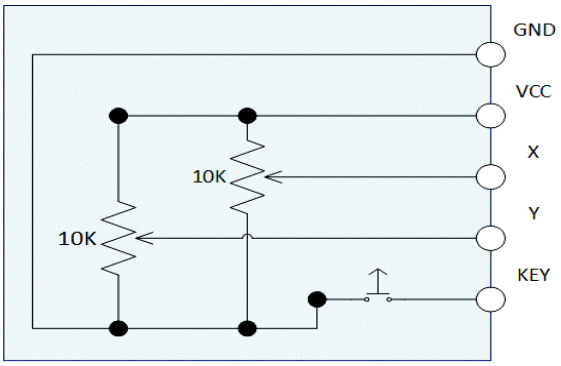
potmeter

```
int _ABVAR_1_potmeter = 0 ;
int __ardublockAnalogRead(int pinNumber)
{
  pinMode(pinNumber, INPUT);
  return analogRead(pinNumber);
}



void setup()
{
  Serial.begin(9600);
}

void loop()
{
  _ABVAR_1_potmeter = __ardublockAnalogRead(A3) ;
  Serial.print("potmeter =");
  Serial.print(" ");
  Serial.print(_ABVAR_1_potmeter);
  Serial.print(" ");
  Serial.println();
  delay( 1000 );
}
```

C-code van bovenstaande Ardublock zonder ESC commando's (nu wordt alles links onder elkaar gedrukt).

## 5.5 Een joystick leren gebruiken

De joystick, die ook voorkomt in de Playstation consoles zijn leuke inputs waarmee we 2 analoge waardes kunnen inlezen tegelijk en ook nog eens kunnen detecteren of de knop is ingedrukt (joystick omlaag drukken). Hier kan je heel leuke projecten mee bedenken 😊



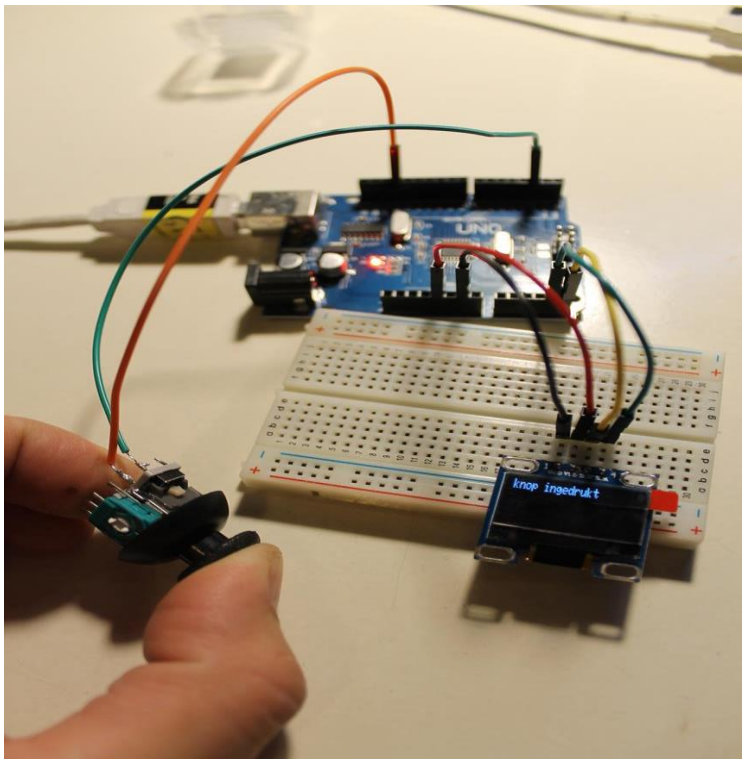

➔ Eerst testen we de knop.

We **solderen 2 draden** op de schakelaar van de joystick (tenzij de knop al op een PCB zit). Als draad gebruiken we 2 breadboard draden waarvan je telkens 1 connector afknipt.
Meet eerst door met de multimeter welke draden de knop onderbreken.

We gaan testen in de code of de knop is ingedrukt (**pin 2**). We zorgen dat deze **laag actief** is geschakeld. Als deze is ingedrukt tonen we tekst op een OLED scherm, anders niets.

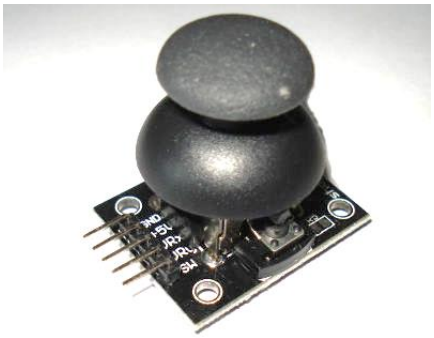
ssd1306_128x64_i2c_testv3_tekst§

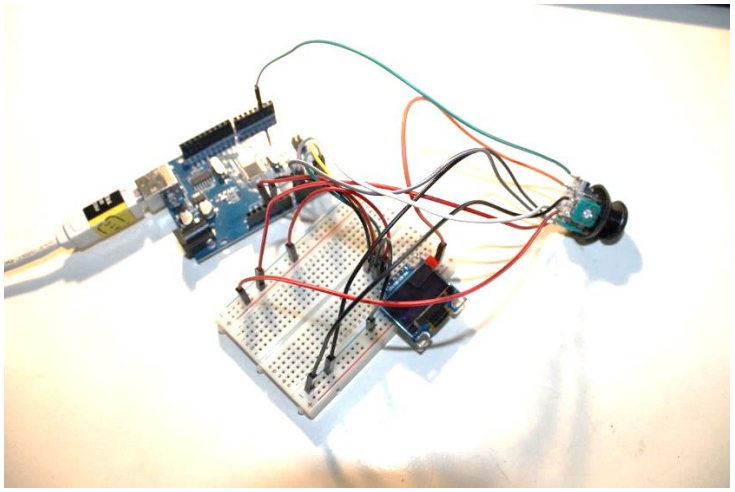
```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

int knop = 2;

void setup()   {

  pinMode(knop,INPUT_PULLUP);//laag actief

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3C (for the 128x64)

}

void loop() {

  if(digitalRead(knop)==LOW){
    //knop aan
      display.setTextSize(1);
      display.setTextColor(WHITE);
      display.setCursor(3,0);
      display.println("knop ingedrukt");
      display.display();
  }
  else{
     display.clearDisplay();
     display.display();
  }
}
```



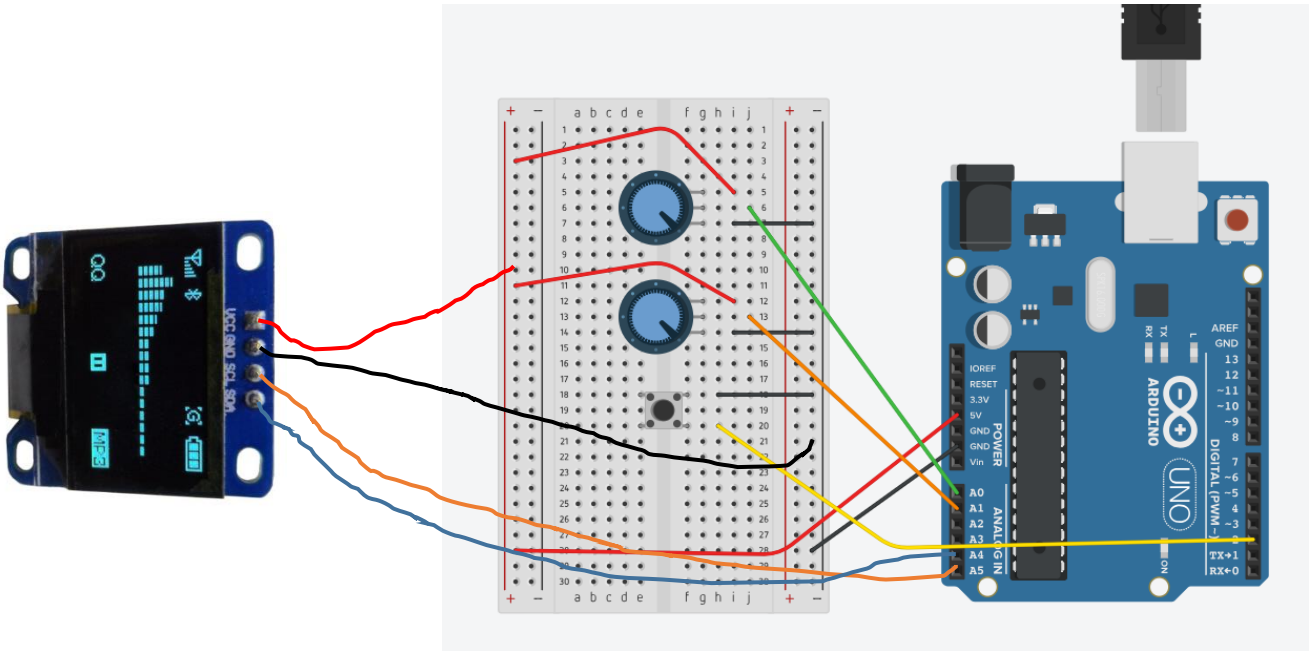

➔ Nu voegen we de potmeters toe.

Ideaal zou het zijn dat we een PCB maakten waar we onze joystick zouden op solderen en alle draden dan laten leiden naar een header. Je vind deze bijvoorbeeld via RobotLinking.
Je kan ook 3 draden per potmeter bijsolderen.
Ik heb beide uitgetest.





+5V, GND, X-as (analoog), Y-as (analoog) en knopuitgang (digitaal)



Merk op dat de joystick hier voorgesteld is als de 3 verschillende onderdelen die er in zitten: 2 analoge potmeters van 10k en een knopje.

➔ We testen code op deze hardware en tonen de analoge waardes op het scherm, telkens we op de knop gedrukt hebben. Ik merkte wel dat af en toe de UNO bij gebruik van de joystick crasht ?

joystick_oled_ganse_joystick

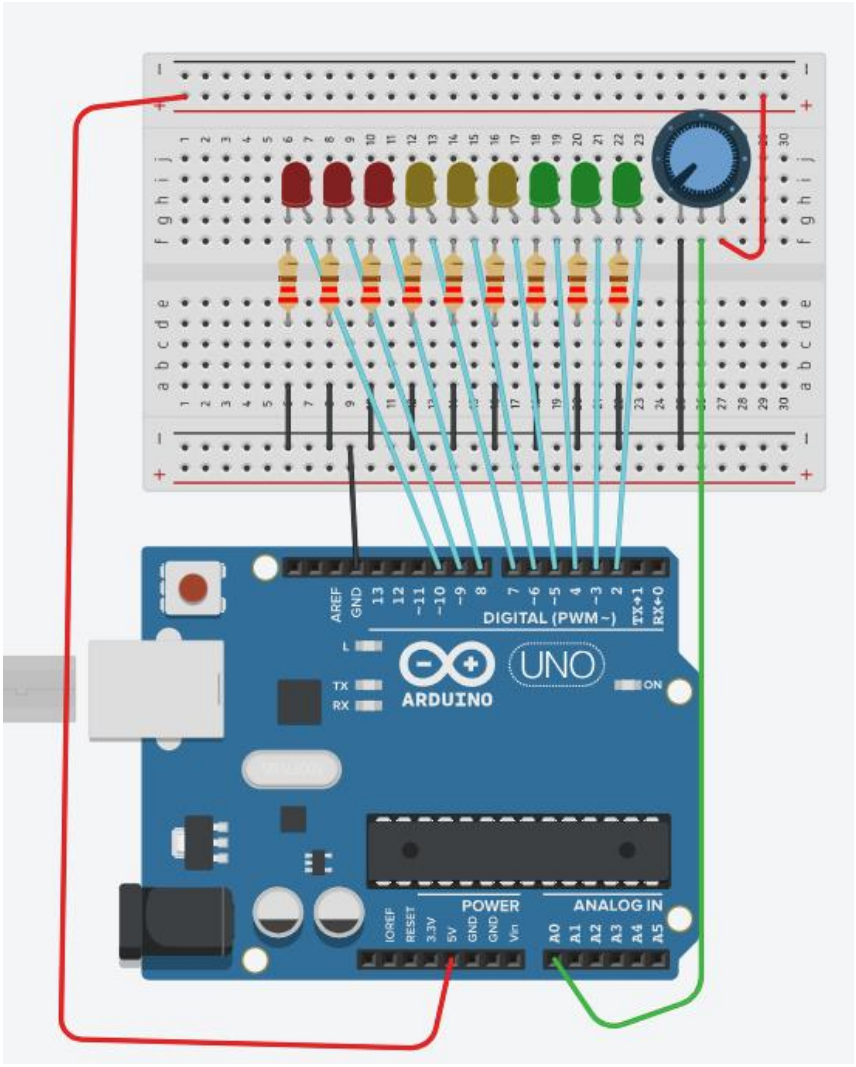
```
void loop() {

  if(digitalRead(knop)==LOW){
    //knop aan

   int analog_x = analogRead(x_as);//waarde tussen 0 en 1023 ingelezen
   int analog_y = analogRead(y_as);

       display.clearDisplay();
       display.display();
       display.setCursor(3,0);
       display.println("knop ingedrukt");
       display.display();
       display.print("x-as = ");
       display.display();
       display.println(analog_x);
       display.display();
       display.print("y-as = ");
       display.display();
       display.println(analog_y);
       display.display();
       delay(300);

  }
  else{

       display.setCursor(3,0);
       display.println("druk knop in");
       display.display();
       delay(300);
       display.clearDisplay();
       display.display();
  }
}
```

➔ Stel dat je de waardes 0 – 1023 (10 bits) moet omzetten naar 0 – 255 (8 bits)?

Hiervoor kunnen we het commando **map**(analoge ingangspin,0,1023,0,255); gebruiken.

Met dit commando wordt de analoge waarde die normaal ingelezen wordt door een 10 bits ADC (analoge digitale converter) omgezet naar een schaal van 0 tot 255 (behorende bij een 8 bits ADC).

We nemen als uitdaging **één potmeter van de joystick (A0) waarvan de waarde dan op 9 LEDs** wordt getoond. Hier vervangen we de 255 door de maximum waarde van het aantal leds die meedoen.




In de code gaan we ook eens een extra trucje toepassen. In plaats van de leds 1 voor 1 aan te spreken gaan we gebruik maken van een **array.** Dit is een geheugenplek of variabele die dezelfde naam heeft voor meerdere geheugenplekjes, maar elke plek heeft wel een nummer (ook **index** genoemd). Met de index kunnen we dus de plek in het geheugen aanduiden. Dit voorkomt veel schrijfwerk in de code.

ledPins[] = {2,3,4,5,6,7,8,9,10};

Dit is een voorbeeld van een array. Alle pinnummers worden op die manier in ledPins opgeslagen. Typisch gebruiken we de [] haakjes om dit aan te duiden.

| ledPins (inhoud) | Index (positie) |
|---|---|
| 2 | 0 |
| 3 | 1 |
| 4 | 2 |
| 5 | 3 |
| 6 | 4 |
| 7 | 5 |
| 8 | 6 |
| 9 | 7 |
| 10 | 8 |

➔ Probeer de volgende code zeker eens uit:

map_joystick_leds §

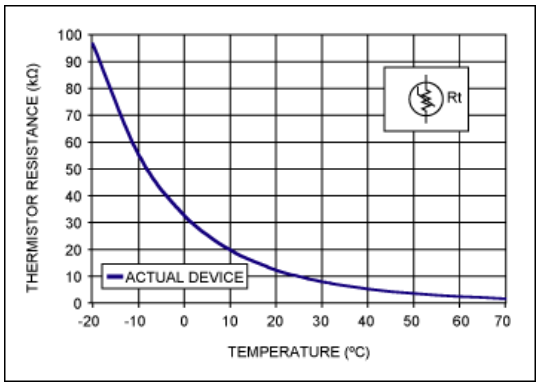
```
int analogPin = A0; //potmeter ingang
int ledCount = 9; //aantal leds
int ledPins[] = {2,3,4,5,6,7,8,9,10}; //de verschillende pinnen
                                       //in 1 keer gedefinieerd
int sensor = 0;
int ledlevel = 0;

void setup() {

  for(int thisLed= 0; thisLed < ledCount; thisLed++){

    pinMode(ledPins[thisLed],OUTPUT); //stel alle pinnen 1 voor 1 in als OUTPUTs
  }
}

void loop() {

 sensor = analogRead(analogPin); //potmeter inlezen
 ledlevel = map(sensor,0,1023,0,ledCount); //de mapping van analoge waarde naar max aantal leds
                                           // 1023 wordt hier in 10 gelijke stukken gedeelt
 for(int thisLed = 0; thisLed < ledCount; thisLed++){

    if (thisLed < ledlevel){ //schakel LEDs in op volgorde
        digitalWrite(ledPins[thisLed],HIGH);
    }
    else{
         digitalWrite(ledPins[thisLed],LOW);//schakel LEDs in volgorde uit
    }
 }

}
```

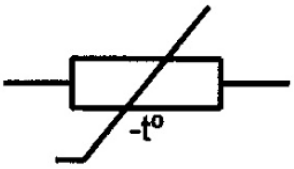
Merk op dat thisLed een **locale variabele** is (enkel in die functie loop of setup aanwezig en aanpasbaar), terwijl de **globale variabele** (voor alle functies geldig) bovenaan staan.

## 5.6 Een NTC inlezen en op de serial monitor tonen

Wat is een NTC?

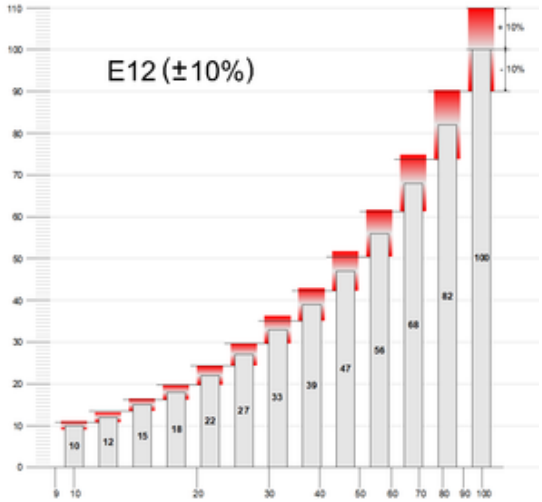
NTC staat voor **Negatieve Temperatuurcoëfficiënt**.
Dit wil zeggen dat de weerstand gaat afnemen als de temperatuur gaat stijgen.
Op wiki kan je hier een mooie vergelijking van terugvinden:
https://nl.wikipedia.org/wiki/NTC-weerstand




Op deze curve kan je heel makkelijk het gedrag van de NTC aflezen.
Hoe groter de temperatuur, hoe kleiner de weerstand.

We gaan de NTC in serie met een weerstand zetten op ons breadboard.
De waarde van een NTC kan je aflezen aan zijn kleurencode of je kan deze ook nameten met de ohmmeter.
De waarde van de NTC wordt vastgelegd bij 25 graden celsius en heeft dan een waarde uit de E12 weerstanden reeks.

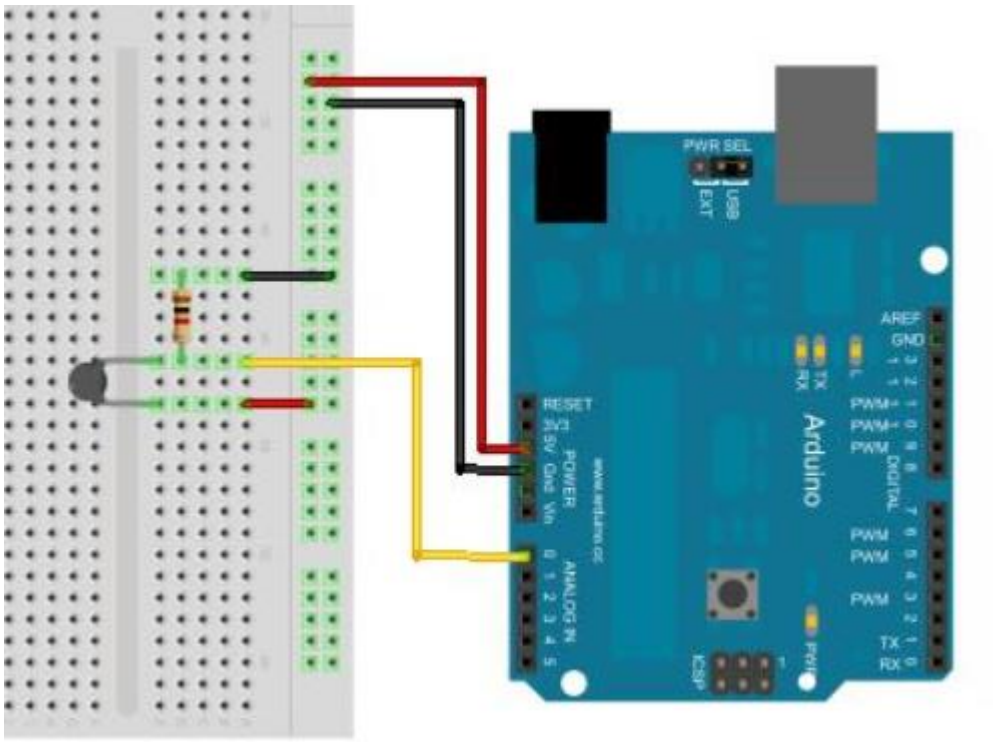



Dit is een voorstelling van de E12 weerstanden reeks.
De 12 waarden zijn op een logaritmische schaal gelijk verdeeld over 1 decade.
https://nl.wikipedia.org/wiki/E-reeks
Zo krijgen we weerstanden van bv 1K, 1K2, 1K5, 1K8 …

In mijn meetopstelling heb ik een NTC gebruikt van **1K5 (bruin groen rood**)

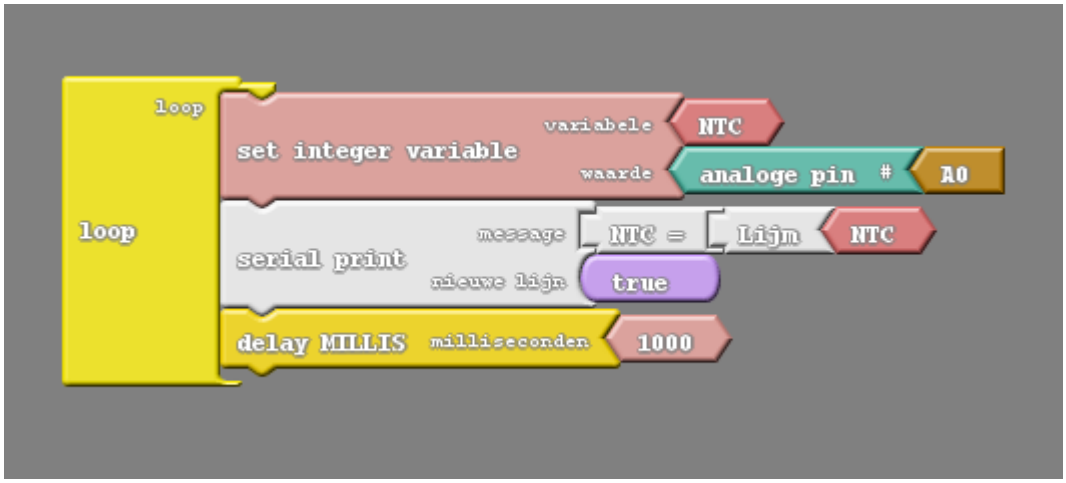
| | zilver | goud | zwart | bruin | rood | oranje | geel | groen | blauw | violet | grijs | wit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mantisse, eerste 2 of 3 ringen | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Vermenigvuldigingsfactor, volgende ring | $10^{-2}$ | $10^{-1}$ | $10^{0}$ | $10^{1}$ | $10^{2}$ | $10^{3}$ | $10^{4}$ | $10^{5}$ | $10^{6}$ | $10^{7}$ | | |
| Tolerantie, meestal laatste ring | 10% | 5% | | 1% | 2% | | | 0,5% | 0,25% | 0,1% | 0,05% | |
| Temperatuurafhankelijkheid, meestal vijfde ring | | | | 1% | 0,1% | 0,01% | 0,001% | | | | | |
| Ezelsbruggetje | | | **Z**ij | **BR**engt | **RO**zen | **O**p | **GE**rrits | **GR**af | **B**ij | **V**les | **GRIJS** | **W**eer |

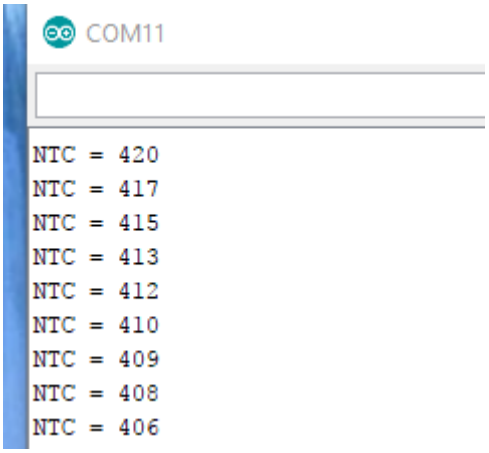
De opstelling ziet er dan met de Arduino als volgt uit:




De NTC zit met 1 kant aan de +5V en de andere aan de analoge ingang van de Arduino. Een weerstand van **1K** is hier gebruikt om een **spanningsdeler** te maken en hangt met 1 zijde aan de massa en de andere ook aan de analoge ingang.

We schrijven de volgende Ardublock code:




Als we onze hand leggen op de NTC (dus **warmte toevoegen**), gaat de weerstandswaarde van de NTC dalen, maar de **spanning aan de microcontroller ingang gaat stijgen**.

$$V_{NTC} = 5 * (1K / NTC + 1K)$$

Vb als NTC = 1000 ohm, dan is $V_{NTC}$ = 2.5V

Als NTC = 1200 ohm, dan is $V_{NTC}$ = 2.27V

De resolutie van de 10 bit analoge ingang is 5/1023 = 5mV = 0.005V

Als je 2.5V meet dan krijg je de waarde ADC = 2.5 / 0.005 = 500 op de serial monitor te zien.

Opmerking: kies altijd een weerstand in serie met de NTC die ongeveer dezelfde waarde heeft. Dan is er voldoende variatie bij een temperatuursverandering aan de ingang van de UNO.

NTC

```
int _ABVAR_1_NTC = 0 ;
int __ardublockAnalogRead(int pinNumber)
{
  pinMode(pinNumber, INPUT);
  return analogRead(pinNumber);
}



void setup()
{
  Serial.begin(9600);
}

void loop()
{
  _ABVAR_1_NTC = __ardublockAnalogRead(A0) ;
  Serial.print("NTC =");
  Serial.print(" ");
  Serial.print(_ABVAR_1_NTC);
  Serial.print(" ");
  Serial.println();
  delay( 1000 );
}
```

Dit is de C-code dewelke de NTC waarde op de serial monitor toont.

NTC_serial_monitor

```
int NTC = A5; //analoge ingang A5
int value = 0;

void setup() {

  Serial.begin(9600);
  Serial.write(27);//ESC
  Serial.print("[?25l");//maak cursor onzichtbaar
  Serial.write(27);//ESC
  Serial.print("[2J");//wis gans scherm

  //merk op dat de analoge ingang niet via pinMode wordt ingesteld!
}

void loop() {

    value = analogRead(NTC);//getal tussen 0 en 1023 inlezen

    Serial.write(27);//ESC
    Serial.print("[1K");//veeg alles van begin regel tot aan de cursor uit
    Serial.write(27);//ESC
    Serial.print("[5;5H");//zet cursor op positie 5,5
    Serial.print("NTC op A5 = ");
    Serial.print(value);//toon analoge waarde op scherm

    delay(1000);
}
```

Deze code zorgt dat de NTC waarde slechts op 1 locatie op de serial monitor verschijnt (via de ESC commando's). Zeker eens testen.

## 5.7 Hoe zet ik de NTC gemeten waarde om in graden celsius?

We willen nu graag de gemeten NTC waarde omzetten in een bruikbare waarde in graden celsius.

Hiervoor moet je gebruik maken van de **Steinhart-Hart** vergelijking.

https://en.wikipedia.org/wiki/Steinhart%E2%80%93Hart_equation

De eenvoudige versie hiervan ziet er zo uit:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R}{R_0}\right)$$

De parameters zijn als volgt:

T = temperatuur resultaat in Kelvin

$T_0$ = kamertemperatuur (met een voltmeter met temperatuuraansluiting te meten)

B = de coëfficiënt die hoort bij de NTC (zie datasheet van jouw NTC), bij mij “3528” (zie tabel).

**ELECTRICAL DATA AND ORDERING INFORMATION**

| $R_{25}$ (Ω) | $R_{25}$-TOL. (± %) | $B_{25/85}$ (K) | $B_{25/85}$-TOL. (± %) | UL RECOGNIZED (Y/N) | SAP MATERIAL NUMBER NTCLE100E3....B0/T1/T2 RoHS COMPLIANT WITH EXEMPTIONS (1) | SAP MATERIAL NUMBER NTCLE100E3....B0A/T1A/T2A RoHS COMPLIANT WITHOUT EXEMPTIONS (1) | COLOR CODE (2) I | II | III |
|---|---|---|---|---|---|---|---|---|---|
| 330 | 2, 3, 5 | 3560 | 1.5 | Y | 331*B0 | 331*B0A | Orange | Orange | Brown |
| 470 | 2, 3, 5 | 3560 | 1.5 | Y | 471*B0 | 471*B0A | Yellow | Violet | Brown |
| 680 | 2, 3, 5 | 3560 | 1.5 | Y | 681*B0 | 681*B0A | Blue | Grey | Brown |
| 1000 | 2, 3, 5 | 3528 | 0.5 | Y | 102*B0 | 102*B0A | Brown | Black | Red |
| 1500 | 2, 3, 5 | 3528 | 0.5 | Y | 152*B0 | 152*B0A | Brown | Green | Red |
| 2000 | 2, 3, 5 | 3528 | 0.5 | Y | 202*B0 | 202*B0A | Red | Black | Red |
| 2200 | 2, 3, 5 | 3977 | 0.75 | Y | 222*B0 | 222*B0A | Red | Red | Red |
| 2700 | 2, 3, 5 | 3977 | 0.75 | Y | 272*B0 | 272*B0A | Red | violet | Red |
| 3300 | 2, 3, 5 | 3977 | 0.75 | Y | 332*B0 | 332*B0A | Orange | Orange | Red |

$R_0$ = de weerstand van de NTC bij kamertemperatuur (25 graden = 1K5 bij mij, zie kleurencode op NTC)

R = gemeten weerstand met de Arduino (wel eerst de spanningswaarde terug omzetten in de code naar weerstand). Hiervoor moet je de volgende formule gebruiken:

R = 1K / (1023 / ADC – 1)

ADC = gemeten waarde van de UNO.

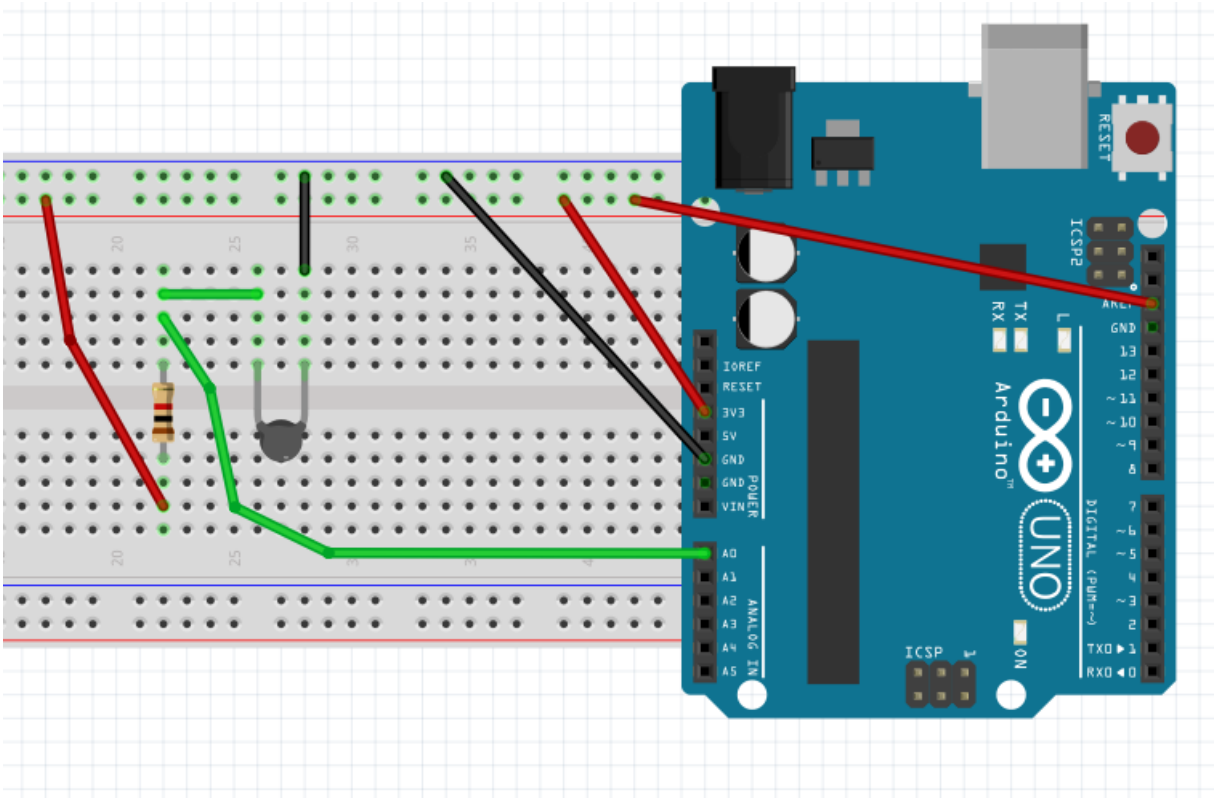
Deze formule is gehaald uit de berekeningen die je kan terugvinden op

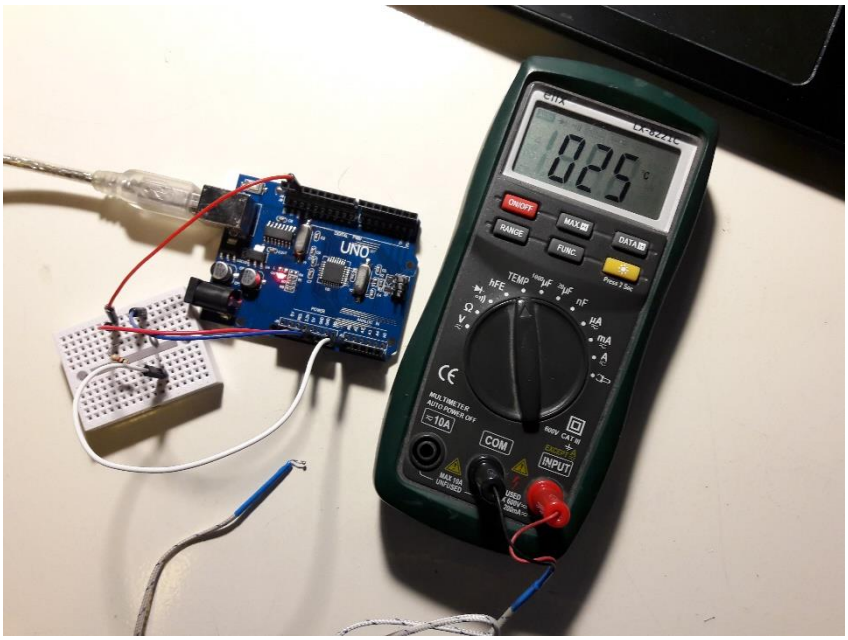
https://learn.adafruit.com/thermistor/using-a-thermistor

We bouwen nu de volgende schakeling:

Merk op dat we de **1K weerstand en de NTC van plaats wisselen** tov de vorige NTC schema's.

We gaan ook de **AREF** gebruiken en hangen deze aan **3V3** op het Arduino bord. Deze 3V3 gebruiken we ook voor de NTC schakeling. Dit zou een stabielere spanning moeten geven als referentie dan de 5V die rechtstreeks van de USB komt. De 5V wordt via een extra filter opgezet naar 3V3. Zodoende wordt de ruis weg gefilterd die nog op de USB voeding zit.




Opstelling UNO met NTC. De kamertemperatuur wordt hier gemeten en vergeleken met de gemeten waarde van de NTC, getoond op de serial monitor. Zo kan je testen of je goed bezig bent.

De C-code ziet er dan als volgt uit:

NTC_graden_celsius

```
// which analog pin to connect
#define THERMISTORPIN A0
// resistance at 25 degrees C
#define THERMISTORNOMINAL 1500
// temp. for nominal resistance (almost always 25 C)
#define TEMPERATURENOMINAL 25
// how many samples to take and average, more takes longer
// but is more 'smooth'
#define NUMSAMPLES 5
// The beta coefficient of the thermistor (usually 3000-4000)
#define BCOEFFICIENT 3528
// the value of the 'other' resistor
#define SERIESRESISTOR 1000

uint16_t samples[NUMSAMPLES];

void setup(void) {
Serial.begin(9600);
analogReference(EXTERNAL);
}

void loop(void) {
uint8_t i;
float average;

// take N samples in a row, with a slight delay
for (i=0; i< NUMSAMPLES; i++) {
samples[i] = analogRead(THERMISTORPIN);
delay(10);
}

// average all the samples out
average = 0;
for (i=0; i< NUMSAMPLES; i++) {
average += samples[i];
}
average /= NUMSAMPLES;
```

AREF wordt aangezet !

Zie de 3V3

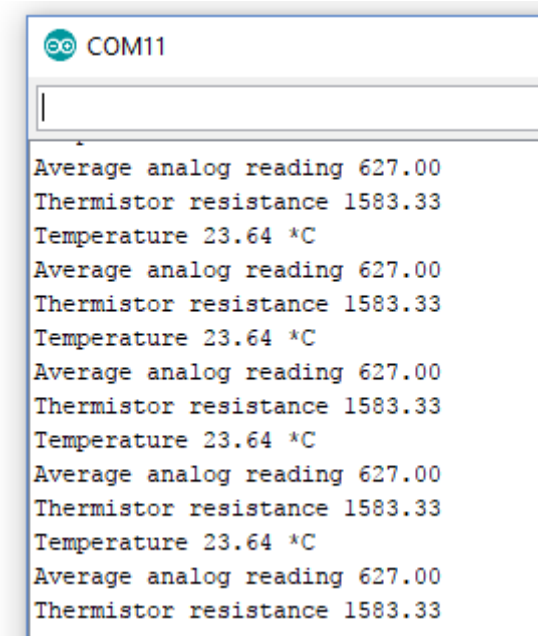
Meet 5 keer de waarde,

Tel deze op en deel door 5

```
Serial.print("Average analog reading ");
Serial.println(average);

// convert the value to resistance
average = 1023 / average - 1;
average = SERIESRESISTOR / average;
Serial.print("Thermistor resistance ");
Serial.println(average);

float steinhart;
steinhart = average / THERMISTORNOMINAL;     // (R/Ro)
steinhart = log(steinhart);                  // ln(R/Ro)
steinhart /= BCOEFFICIENT;                   // 1/B * ln(R/Ro)
steinhart += 1.0 / (TEMPERATURENOMINAL + 273.15); // + (1/To)
steinhart = 1.0 / steinhart;                 // Invert
steinhart -= 273.15;                         // convert to C

Serial.print("Temperature ");
Serial.print(steinhart);
Serial.println(" *C");

delay(1000);
}
```

Bovenstaande code geeft aan hoe de formule van STEINHART er uit ziet.

Het resultaat van onze gemiddelde meting wordt getoond op de serial monitor in graden celsius.



## 5.8 Toon de temperatuur op het LCD

Combineer nu de LCD kennis met het inlezen van de NTC en toon de graden op het schermpje.

Hier is de code en de opstelling:

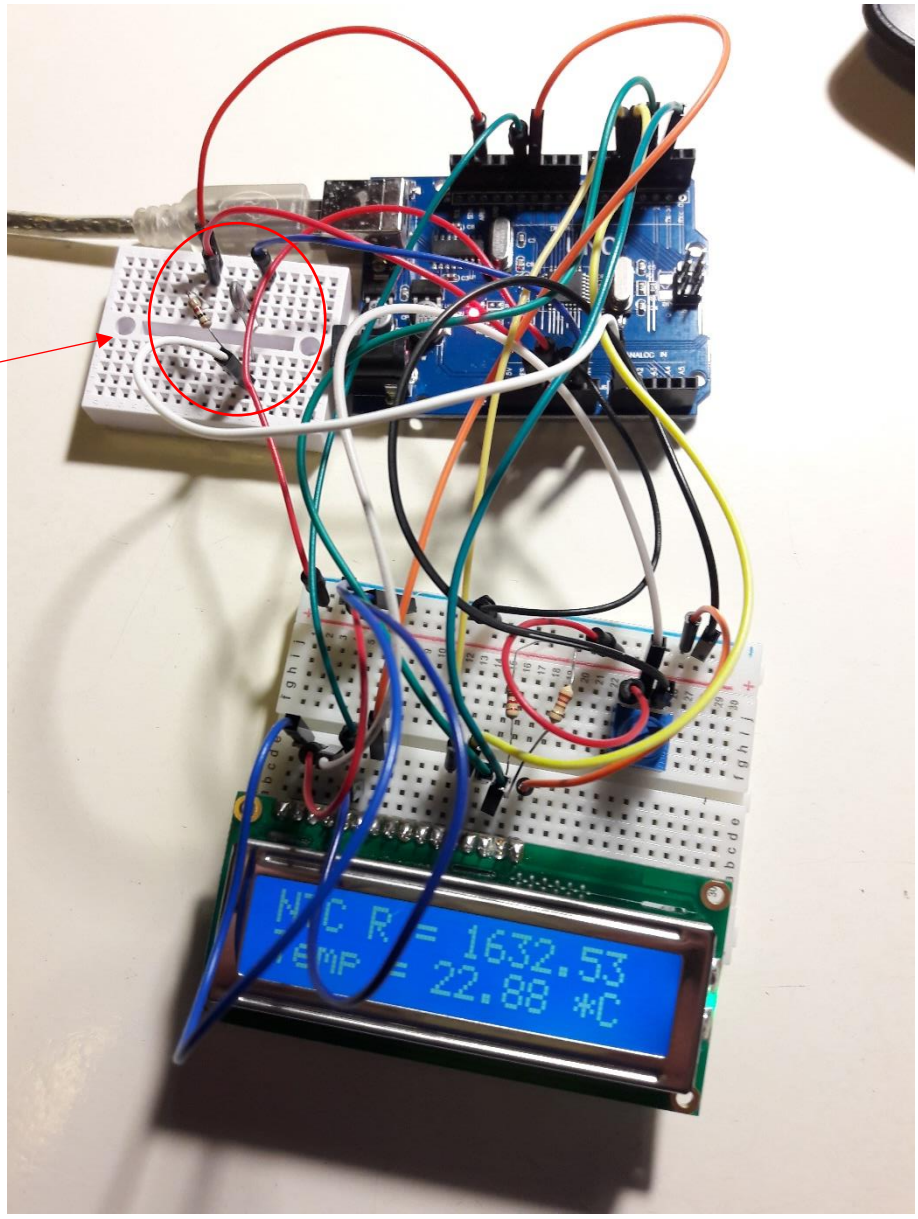
NTC_graden_celsius_op_LCD

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// which analog pin to connect
#define THERMISTORPIN A0
// resistance at 25 degrees C
#define THERMISTORNOMINAL 1500
// temp. for nominal resistance (almost always 25 C)
#define TEMPERATURENOMINAL 25
// how many samples to take and average, more takes longer
// but is more 'smooth'
#define NUMSAMPLES 5
// The beta coefficient of the thermistor (usually 3000-4000)
#define BCOEFFICIENT 3528
// the value of the 'other' resistor
#define SERIESRESISTOR 1000

uint16_t samples[NUMSAMPLES];

void setup(void) {

analogReference(EXTERNAL);

lcd.begin(16,2);

}

void loop(void) {
uint8_t i;
float average;

// take N samples in a row, with a slight delay
for (i=0; i< NUMSAMPLES; i++) {
samples[i] = analogRead(THERMISTORPIN);
delay(10);
}
```

```
// average all the samples out
average = 0;
for (i=0; i< NUMSAMPLES; i++) {
average += samples[i];
}
average /= NUMSAMPLES;


// convert the value to resistance
average = 1023 / average - 1;
average = SERIESRESISTOR / average;
lcd.clear();
lcd.print("NTC R = ");
lcd.print(average);

float steinhart;
steinhart = average / THERMISTORNOMINAL;     // (R/Ro)
steinhart = log(steinhart);                  // ln(R/Ro)
steinhart /= BCOEFFICIENT;                   // 1/B * ln(R/Ro)
steinhart += 1.0 / (TEMPERATURENOMINAL + 273.15); // + (1/To)
steinhart = 1.0 / steinhart;                 // Invert
steinhart -= 273.15;                         // convert to C

lcd.setCursor(0,1);
lcd.print("Temp = ");
lcd.print(steinhart);
lcd.print(" *C");

delay(1000);
}
```

NTC opstelling

aangesloten op 3V3




Totale opstelling thermometer met LCD scherm

## 5.9 Een NTC van 10K i.p.v. 1K5 ?

Wanneer de NTC een 10K waarde heeft, dan gaan we al eerste op zoek naar de datasheet.

We hebben hier te maken met een Vishay NTC, nummer **NTCLE413.** Zie de zwarte kleine NTC in jouw doos.

**Table 3**

| PART IDENTIFICATION | $R_{25}$ | | $B_{25/85}$ | |
|---|---|---|---|---|
| | kΩ | ± % | K | ± % |
| NTCLE413-428 10K 1 % B3435 K | 10 | 1 | 3435 | 1.0 |

Uit de tabel moeten we de **B** waarde halen. Deze is **3435** voor 10K.

Deze parameter passen we aan in de code, net als de serieweerstand van 10K en de waarde van de NTC bij 25 graden, ook 10K.

Als je deze parameters invult dan krijg je de volgende code. We gebruiken hier een LCD scherm met I2C. De buzzer gaat af als het meer dan 30 graden wordt. Let op de polariteit bij de buzzer!

NTC_graden_celsius_op_LCD_10k

```
#include <Wire.h> // bibliotheek om met I2C chips te praten
#include <LiquidCrystal_I2C.h> //bib voor het LCD scherm

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address 0X27

// which analog pin to connect
#define THERMISTORPIN A0
// resistance at 25 degrees C
#define THERMISTORNOMINAL 10000
// temp. for nominal resistance (almost always 25 C)
#define TEMPERATURENOMINAL 25
// how many samples to take and average, more takes longer
// but is more 'smooth'
#define NUMSAMPLES 5
// The beta coefficient of the thermistor (usually 3000-4000)
#define BCOEFFICIENT 3435 //zie datasheet
// the value of the 'other' resistor
#define SERIESRESISTOR 10000

uint16_t samples[NUMSAMPLES];

int buzzer = 2; //op pin 2 de buzzer aansluiten
int duration = 100;

void setup(void) {

   pinMode(buzzer,OUTPUT);

   analogReference(EXTERNAL); //3V3 ipv 5V

   lcd.begin(16,2);
   lcd.backlight();//BACKLIGHT AAN


}
```

```
void loop(void) {
uint8_t i;
float average;

// take N samples in a row, with a slight delay
for (i=0; i< NUMSAMPLES; i++) {
samples[i] = analogRead(THERMISTORPIN);
delay(10);
}

// average all the samples out
average = 0;
for (i=0; i< NUMSAMPLES; i++) {
average += samples[i];
}
average /= NUMSAMPLES;

// convert the value to resistance
average = 1023 / average - 1;
average = SERIESRESISTOR / average;
lcd.clear();
lcd.print("NTC R = ");
lcd.print(average);

float steinhart;
steinhart = average / THERMISTORNOMINAL;     // (R/Ro)
steinhart = log(steinhart);                  // ln(R/Ro)
steinhart /= BCOEFFICIENT;                   // 1/B * ln(R/Ro)
steinhart += 1.0 / (TEMPERATURENOMINAL + 273.15); // + (1/To)
steinhart = 1.0 / steinhart;                 // Invert
steinhart -= 273.15;                         // convert to C

lcd.setCursor(0,1);
lcd.print("Temp = ");
lcd.print(steinhart);
lcd.print(" *C");
```

```
if (steinhart > 30){

  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print("hittegolf");

  for (long i = 0; i < duration; i++){

    digitalWrite(buzzer,HIGH);
    delayMicroseconds(1000);//1ms hoog en 1ms laag = 2KHZ
    digitalWrite(buzzer,LOW);
    delayMicroseconds(1000);
  }

}

delay(1000);
}
```

## 5.10 Uitdagingen

1. Doe een meting met een LDR en toon het resultaat op een LCD scherm/OLED.
2. Doe een meting van de potmeter en toon deze op het LCD scherm/OLED.
3. Neem een joystick en toon beide waardes op een LCD scherm, enkel als je op de knop van de joystick drukt.
4. Maak een VU meter met LEDs. Hoe verder je de potmeter opendraait, hoe meer LEDs er zullen aangaan. Gebruik hier de map() functie.
5. Meet via een NTC een analoge waarde en zet deze om naar een decimale waarde, in graden, op een LCD scherm.

   **Extra:**

6. Maak een dimmer van een LED met PWM en toon ondertussen de waarde op het LCD (oplossing zie volgende pagina).
7. Meet de lijnvolgers analoog en toon beide op een andere lijn van het scherm.
8. Gebruik oef 3 om lijnvolgers voor verschillende kleuren te kalibreren. Gebruik deze zo als kleurensensor en laat de robot bij elke kleur/combinatie van kleuren iets anders doen. Zoek eens wat inspiratie bij de OZOBOT. Zie http://ozobot-benelux.nl/
9. Wat als mijn NTC 10K is i.p.v. 1K5? Pas de code instellingen en opstelling aan zodat alles terug goed werkt (oplossing zie vorige pagina's).

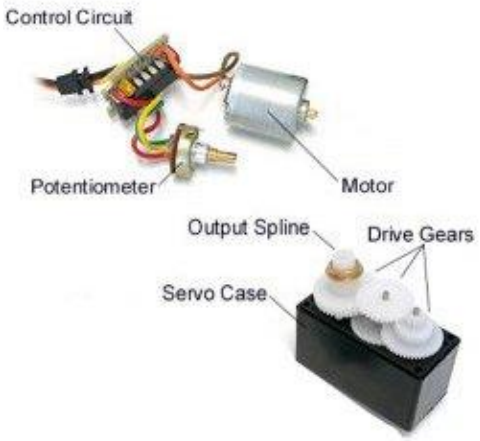
# 6. Servo's leren aansturen

➔ Wat is een servo?

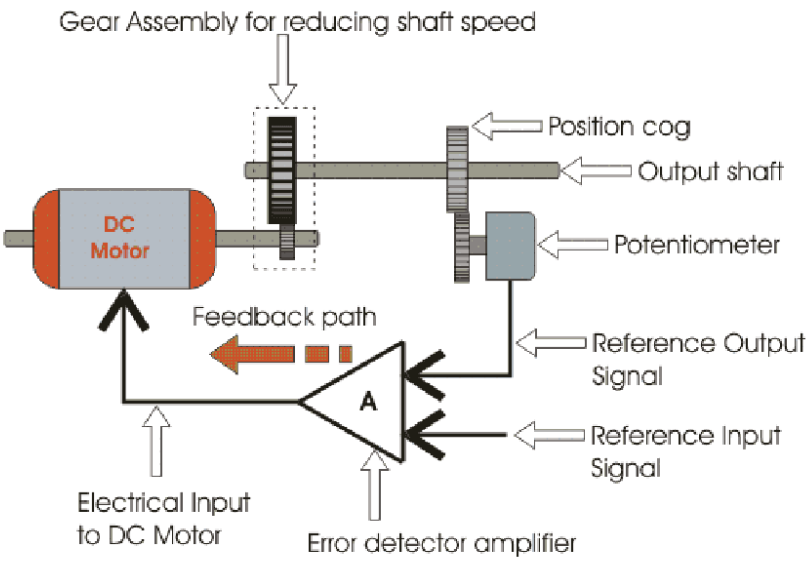
Een servo is een kleine motor waar je armpjes op kan monteren die in heel juiste hoek kunnen aannemen tussen 0 en 180 graden. Je kan deze monteren in een pan-en-tilt behuizing. Zo kan je dan een camera of laser bedienen en sturen in de juiste richting. Ook in kleine robotjes komen deze servo's vaak voor, zeker als de robot uit meerdere bewegingsvrijheden bestaat.

➔ Wat zit er in de servo?

De servo bestaat hoofdzakelijk uit een DC motor, een tandwieloverbrenging, een potentiometer die meedraait met de motor en een klein microcontroller circuit waarmee je communiceert met de Arduino.




Als je de potmeter zou verwijderen, kan je in principe de motor laten ronddraaien.

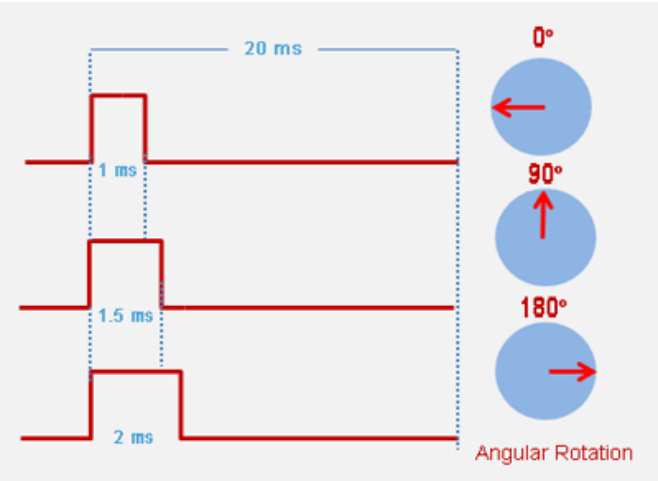



Hier zie je nog beter de terugkoppeling (closed loop systeem) van de potentiometer naar de comparator. Via de Arduino wordt een **referentie signaal** aangelegd aan 1 kant van de opamp. T.o.v. het signaal dat is teruggekoppeld via de potmeter gaat er een **vergelijking** gebeuren in de opamp. Als de signalen verschillen gaat de uitgang van de opamp naar de DC motor een **error correctie signaal** sturen. De motor draait de assen en de **potentiometer wordt bijgestuurd** tot dat het verschil op de opamp uitgang nul is. Als de PWM puls van de Arduino de referentie input weer verandert begint alles weer van vooraf aan.

Je kan ook zien dat de potmeter meedraait op dezelfde as, als de motor ook beweegt.

➔ Welke aansluitingen zijn er aan een servo en hoe stuur je deze aan?

Een servo heeft typisch 3 aansluitingen: een +5V (rode draad), GND (bruine draad) en de stuursignaaldraad (geel of oranje).




Op de signaaldraad moeten we een puls via de Arduino aanleggen. De **breedte van de puls** bepaald in welke richting de motor zal draaien. De pulsen komen **om de 20ms**.

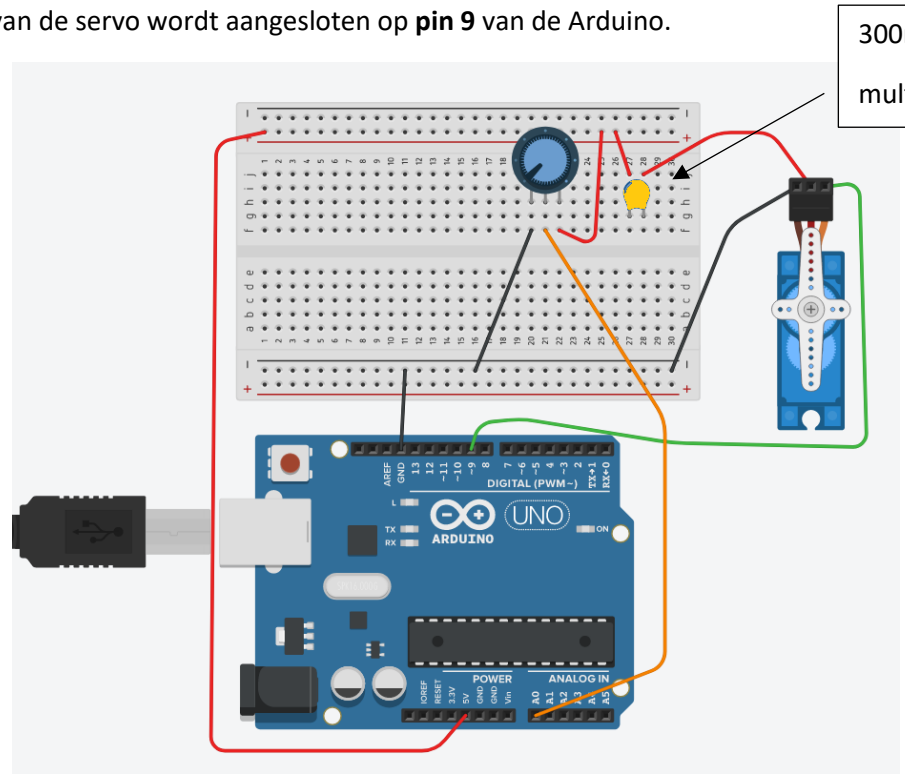
Tip: voor de lange levensduur van de servo is het altijd goed van een **multifuse in serie** te zetten met de +5V. Als de motor eens vastzit of teveel stroom vraagt, dan beschermt hij zichzelf.

## 6.1 Hoe stuur je een servo aan in Arduino?

We gaan 1 analoge potmeter van de **joystick (x-as)** gebruiken om de servo van 0 naar 180 graden te bewegen. We sluiten deze aan op de **A0** analoge ingang van de Arduino.

De **x-as** gaan we de **tilt servo** noemen. Straks komt er een **y-as** bij en wordt dat de **pan servo**.

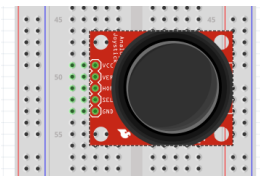
Het signaal van de servo wordt aangesloten op **pin 9** van de Arduino.




In mijn opstelling sluit ik ook nog een I2C LCD scherm aan op pin A4(SDA) en A5(SCL). Zo kan ik goed volgen hoe de servo wordt aangestuurd.

Merk op dat je in het tekenpakket **fritzing** de originele joystick kan terugvinden in de componenten.

Vergeet niet van een 300mA multifuse te voorzien in de +5V van de servo ter stroombeperking.



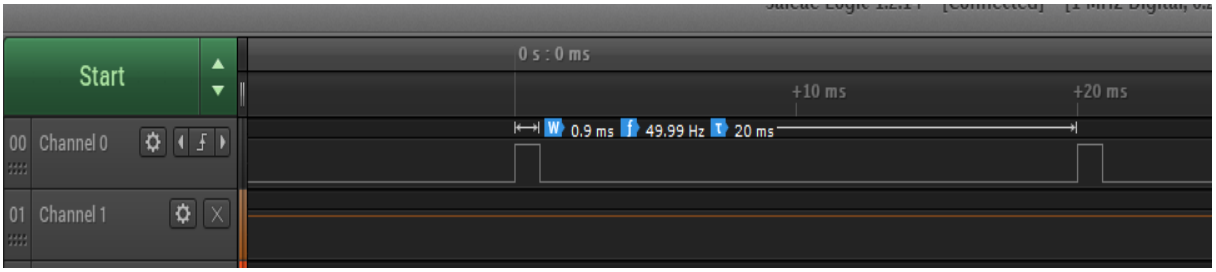
servo_test §

```
#include <Servo.h>

#include <Wire.h> // bibliotheek om met I2C chips te praten
#include <LiquidCrystal_I2C.h> //bib voor het LCD scherm
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

Servo tilt; //servo object creëren
int joyX = A0; //analoge joystick pin x verbinden met A0
int x;

void setup() {
  tilt.attach(9); // sluit tilt servo aan op pin 9

  lcd.begin(16,2);   // initialize the lcd for 16 chars 2 lines
  lcd.backlight();   // turn on backlight

}

void loop() {

 x = map(analogRead(joyX),0,1023,900,2500);
 //lees analoge waarde potmeter in (0 - 1023) en zet deze
 //naar een handige servo waarde om tussen 900 en 2500 microsec

tilt.write(x);//stel servo positie in van x-as

lcd.clear();
lcd.setCursor(0,0); //Start at character 0 on line 0
lcd.print("pin joyX= ");
lcd.print(joyX);
lcd.setCursor(0,1);
lcd.print("x= ");
lcd.print(x);
delay(100);

}
```

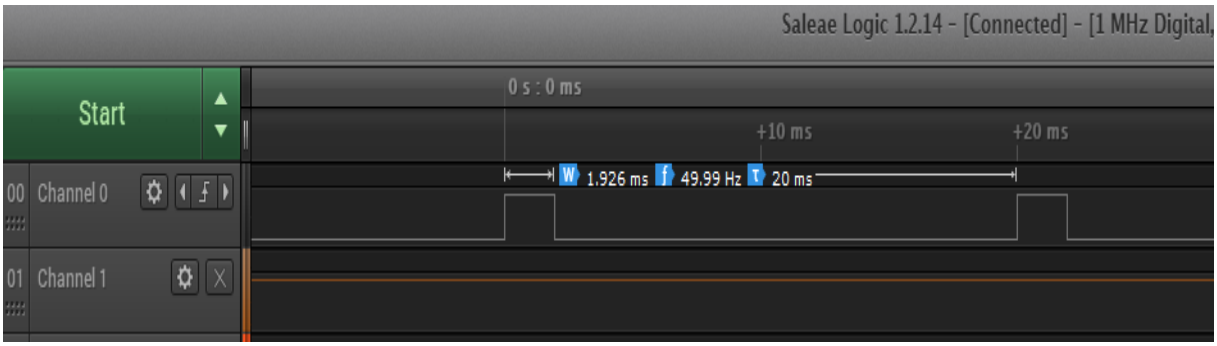
Dit is een mogelijkheid om eenvoudig servo's te sturen.

Het getal moet tussen 900 en 2500 microseconden liggen.

**Origineel moet je 2000** als maximum getal ingegeven, maar dan gaat de servo net niet ver genoeg in 1 richting. **1000** zou het **minimum** moeten zijn. Best even experimenteren.

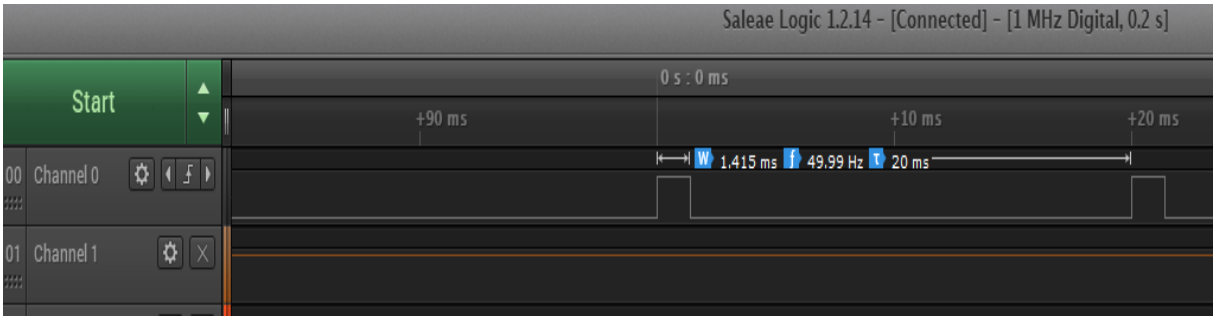



Logic Analyser : bij potmeter op 900 wordt de puls **0.9 ms.** Motor draait **helemaal naar 1 kant**. Normaal moet dit bij 1ms = 1000 al voldoende zijn. De **periode** van de pulsen is **20ms.**




Is de potmeter naar de andere kant gedraaid dan krijgen we bij 1926 ook **1.926 ms** als puls.

De motor draait nu volledig **naar de andere kant**. Normaal moet 2000 = 2 ms het uiterste zijn.




Bij een losse potmeter staat de servo **in het midden.** 1415 op het scherm geeft **1.415 ms** als puls.

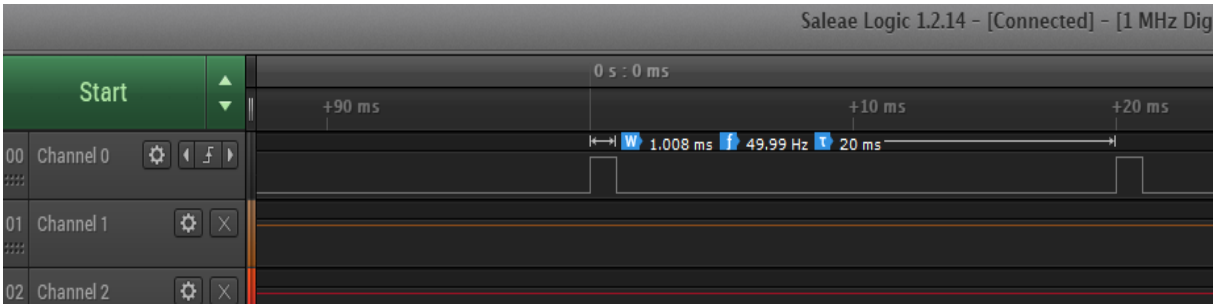
Je kan ook gaan werken met de **graden**. Dan krijg je de volgende code:

servo_testv2

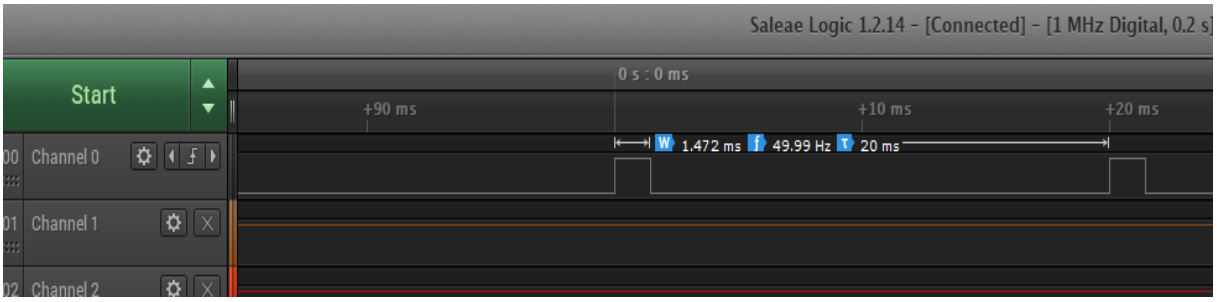
```
#include <Servo.h>

int servoPin = 9;
Servo servo;
int servoAngle = 0;   // servo position in degrees

void setup()
{
  servo.attach(servoPin);
}

void loop()
{
//control the servo's direction and the position of the motor

   servo.write(45);      // Turn SG90 servo Left to 45 degrees
   delay(1000);          // Wait 1 second
   servo.write(90);      // Turn SG90 servo back to 90 degrees (center position)
   delay(1000);          // Wait 1 second
   servo.write(135);     // Turn SG90 servo Right to 135 degrees
   delay(1000);          // Wait 1 second
   servo.write(90);      // Turn SG90 servo back to 90 degrees (center position)
   delay(1000);

//end control the servo's direction and the position of the motor
```

```
//control the servo's speed

//if you change the delay value (from example change 50 to 10), the speed of the servo changes
  for(servoAngle = 0; servoAngle < 180; servoAngle++)  //move the micro servo from 0 degrees to 180 degrees
  {
    servo.write(servoAngle);
    delay(50);
  }

  for(servoAngle = 180; servoAngle > 0; servoAngle--)  //now move back the micro servo from 0 degrees to 180 degrees
  {
    servo.write(servoAngle);
    delay(10);
  }
  //end control the servo's speed
}
```

In het 2de deel van de code laten we de servo langzaam roteren van 0 tot 180 graden.




Bij 45 graden wordt een puls van 1ms uitgestuurd. De servo draait 45 graden naar 1 zijde.




Bij 90 graden staat de servo in het midden. Puls is nu ongeveer 1.5ms.

Om dus **naar 0 graden en 180 graden** te gaan moet je de servo brengen tot **< 1ms en > 2ms pulsen**.

Best zelf eens mee experimenteren.

➔ Uitdaging: het is interessant om eens zelf een **digitale scope** / **logic analyser** aan te sluiten op de servo uitgang.
Zo kan je dan het echte stuursignaal bestuderen en kijken of het overeenkomt met de theorie.

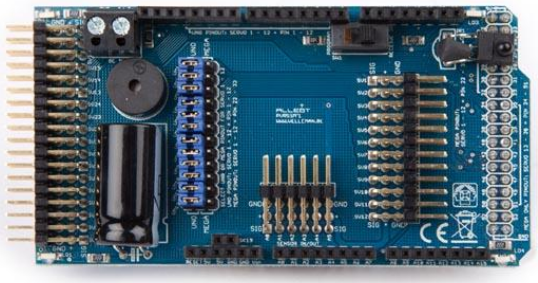
## 6.2 Servo's sturen mechanica aan (zie bouwkits)

Wanneer we meerdere vrijheden of bewegingen willen bekomen van onze robotarm of dergelijke, dan moeten we dus meer servo's inschakelen.

De Velleman robot is op dat gebied een kunstwerkje. Elke poot heeft 2 servo's te sturen. Dus 8 in totaal (je kan nog 2 poten extra aansluiten als je dat wil). Het sturen van zo'n robot wordt dan een beetje makkelijker gemaakt, voor de bedrading, door een extra shield te plaatsen op de UNO. Bouwinstructie vind je hier.

Merk op dat je met een 3D printer alle onderdelen zelf kan printen.

ALLBOT

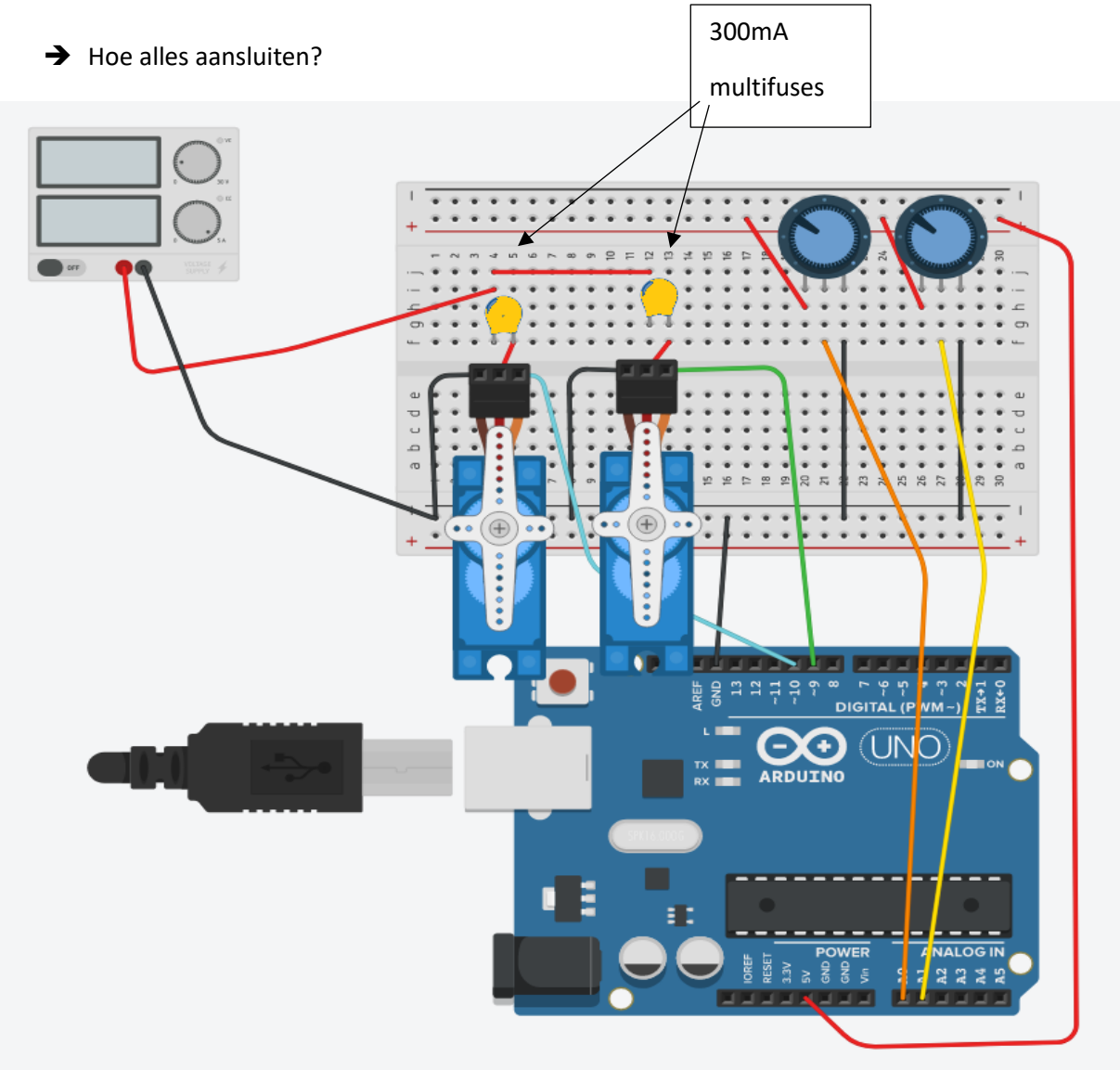


VRSSM shield

We gaan ons bezig houden met 2 servo's. Deze gaan we **pan en tilt** laten doen op een kleine beugelconstructie. Een video handleiding voor de montage kan je hier vinden.

Ook bij adafruit kan je info voor de montage vinden. Er zit ook een **pdf versie en foto's** op de USB stick.

➔ Hoe alles aansluiten?




We gebruiken de **x-as en y-as van de joystick**. Dus beide potmeters aansluiten.

De signalen van de servo worden aangesloten op pin 9 en pin 10.

Merk op dat de servo library tot 12 motoren ondersteund. Wanneer deze library wordt gebruikt dan is wel de PWM functie op pin 9 en pin 10 (via analogWrite() ) niet beschikbaar!

Bij de Arduino Mega kan je 12 servo's naast 12 PWM functies gebruiken 😊

We zijn dus **niet verplicht om een PWM uitgang te gebruiken voor de servo uitgangen**. Je kan alle digitale uitgangen gebruiken.

Meer info vind je hier.

Om onze Arduino niet te overbelasten gaan we de +5V servo spanningen aftappen van een externe voeding. In elke +5V van de servo's zit een multifuse van 300mA als stroombegrenzer!

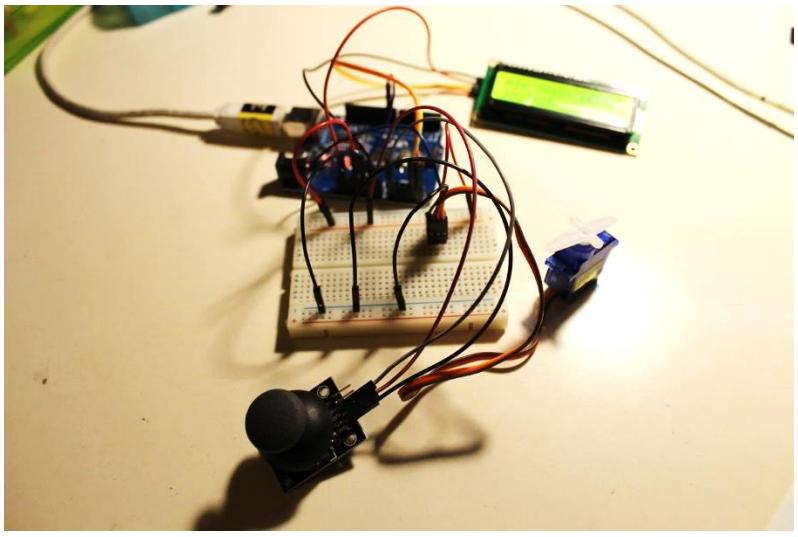
➔ Hoe ziet de code er dan uit?

2_servo_pan_tilt

```
#include <Servo.h>

Servo tilt, pan; //servo objecten creëren
int joyX = A0; //analoge joystick pin x verbinden met A0
int joyY = A1; // analoge joystick pin y verbinden met A1
int x, y;

void setup() {
  tilt.attach(9); // sluit tilt servo aan op pin 9
  pan.attach(10); // sluit pan servo aan op pin 10


}

void loop() {

 x = map(analogRead(joyX),0,1023,900,2500);
 //lees analoge waarde potmeter in (0 - 1023) en zet deze
 //naar een handige servo waarde om tussen 900 en 2500 microsec

 y = map(analogRead(joyY),0,1023,900,2500);

tilt.write(x);//stel servo positie in van x-as
pan.write(y);
delay(15); //wacht op nieuwe positie servo

}
```

Dit keer gaan we zowel x als y inlezen via de potmeters.

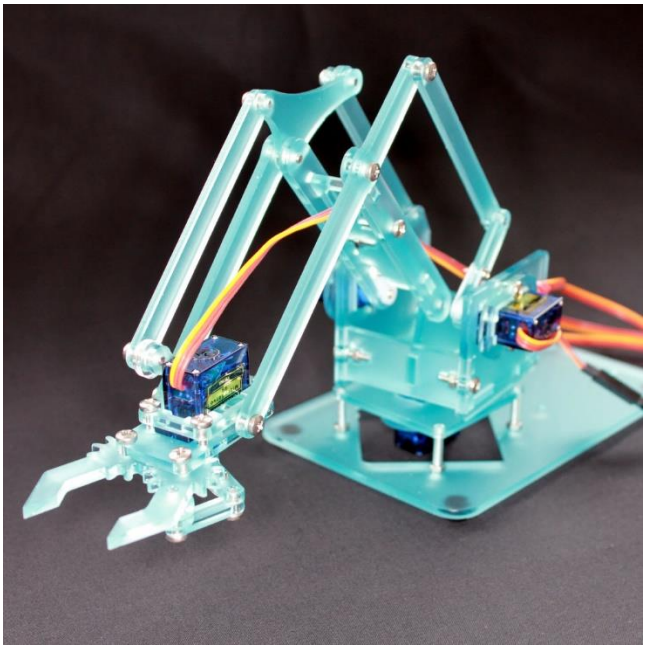
Na de nodige mapping wordt de waarde omgezet naar servo waardes.

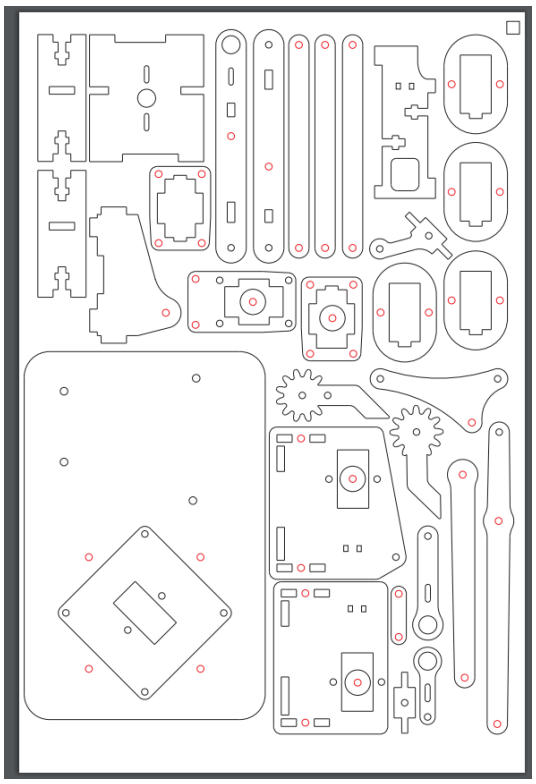


Koop een Arm-04 en bouw deze robotarm. Hij wordt gestuurd door 4 servo's, met 2 joysticks.

Je kan de ganse robot bij Ali kopen of zelf lasercutten. Daarna monteren. Hier een youtube filmpje.

Je kan op de website van instructables ook de montage netjes in stapjes volgen.





## 6.3 Met een servo een laser aansturen

Nu we toch een pan tilt robotarm hebben kunnen we hier allerlei **elektronica op monteren**.

Je kan er aan afstandssensor, camera of zelfs een laser op monteren.

**Let wel op dat met je veilig omspringt met de laser. Deze is 5mW 650nm 5V (klasse 3R). Als je deze in je oog krijgt kan je tijdelijk verblind zijn. Dus best niet op iemand richten of in de laser kijken!**

Je kan hierover info vinden op het web.

Of bij wikipedia.

- **Klasse 3R**: deze laser wordt **beschouwd als ongevaarlijk**, maar kan **potentieel gevaarlijk zijn wanneer direct in de laserbundel wordt gekeken**. De limiet is 5 keer de waarde van de limiet die geldt voor klasse 1 (voor onzichtbaar licht) of klasse 2 (voor zichtbaar licht).

Dus best de opstelling inbouwen in **een veilige ruimte** om te testen.

Wist je dat Laser staat voor: **L**ight **A**mplification by **S**timulated **E**mission of **R**adiation, in het Nederlands: lichtversterking door gestimuleerde uitzending van straling.

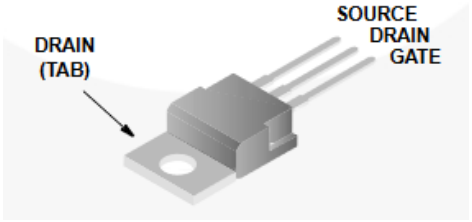
➔ Wat verbruikt de laser?

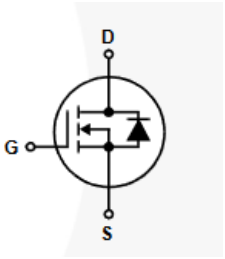
Volgens de websites verbruikt onze laser **max 25mA** (gemeten met de Ampére meter is het 24.2 mA). Dit is echter te veel voor onze microcontroller, die bij deze stroom zijn max bereikt.

Oplossing: weer een **elektronische schakelaar** gebruiken zoals een transistor (BC547, tot 100 mA), een optokoppelaar of een **MOSFET** (BS170 kan 500mA aan, of de **RFP12N10L**).

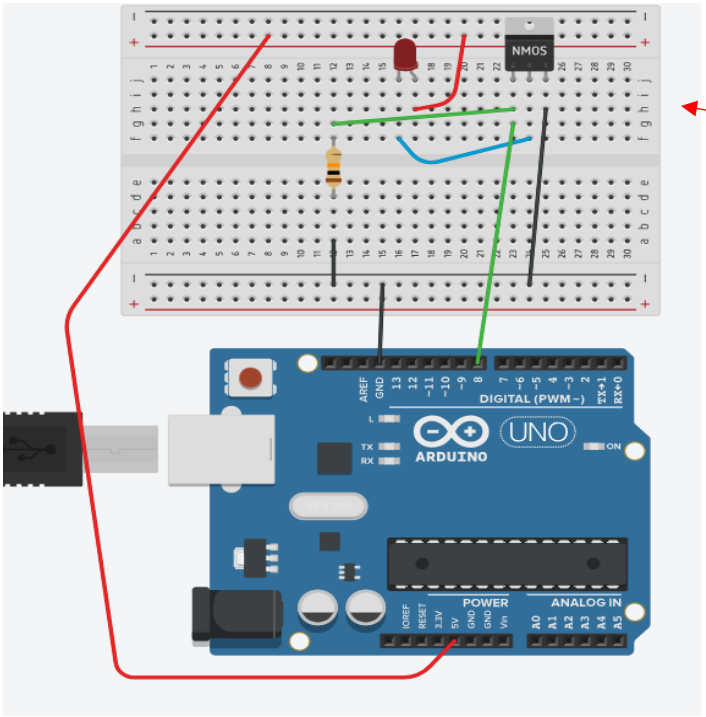
Voor meer info over de MOSFET kan je in **hoofdstuk 1.6** terecht van deze cursus.

Merk op dat de laser een diode is en dus ook een polariteit heeft: **GND (blauwe draad) en +5V (rode draad).** Vermits deze laserdiode reeds op 5V werkt moet je er **geen voorschakelweerstand** voor zetten. Merk op dat er al een SMD voorschakelweerstand van **16 ohm** op de laser is voorzien om de stroom te beperken. Soldeer 2 headers op de draadjes.




➔ Aansluitschema:




Laserdiode

5V 5mW

25mA

Eigen stroombeperking

Opgelet met de ogen!!!

De 10k is er weer als pulldown om de MOSFET niet te laten werken als hij niet gestuurd wordt.

Gewoon via een **digitalWrite() op pin 8** aansturen om de laser te laten branden.

➔ **Uitdagingen servo's:**

1. Bouw de schakeling op een breadboard waarbij we met 1 joystick een pan en tilt systeem aansturen. Toon de x en y waardes bijvoorbeeld op een LCD scherm, OLED of serial monitor ter controle.
2. Bouw met 2 joysticks een schakeling waar bij je de ME Arm kan aansturen (4 servo's). De servo's bewegen een laserstraal (schijn hier enkel mee op de muur, niet in iemand zijn oog!). De laser mag pas aangaan als je op de knop van de joystick drukt. Zorg er ook voor dat de servo's zachte bewegen doorlopen!
3. Pas nu de code van de ME Arm zo aan dat de servo blijft staan op de plek waar je hem naar toe stuurt. Dus als je de joystick loslaat blijft de servo toch nog staan op de laatste plek en zwiept deze niet terug naar zijn beginpositie !

# 7. Stappenmotor

Na de DC motoren en de servo motoren gaan we nu de stappenmotor onder de loop nemen.

Deze motor kan **heel precies** (in kleine stapjes) zijn **as laten bewegen**.

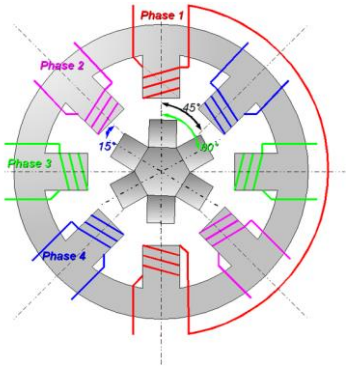
Ze worden ingezet op plaatsen waar men **heel precies** wil gaan werken. Denk maar aan de lade die open/toe ging van de grote floppydrive of de 3D printers en lasercutters.


Toepassingen genoeg van stappenmotoren

De stappenmotor is een borstelloze synchrone elektromotor volgens wikipedia. We kunnen hoekverdraaiingen tot 1.8 graden instellen. De rotor bevat permanente magneten. De stator is opgebouwd uit elektromagneten.

De hoge resolutie wordt bereikt door zowel de stator als de rotor van een hoog aantal polen te voorzien, zodanig dat zij geen gemeenschappelijke factor bezitten. Telkens als een wikkeling bekrachtigd wordt, komt een pool in de rotor recht tegenover een pool in de stator te staan, waardoor de rotor een klein stukje draait. Door middel van pulsbreedtemodulatie kan een nog hogere resolutie worden bereikt.

De belangrijkste karakteristieke eigenschap van een stappenmotor is het **koppel** dat hij kan leveren. De motor kan ook koppel leveren als hij stilstaat en kan daarom als standrem fungeren.




Er bestaan 2 types stappenmotoren: unipolair en bipolair.

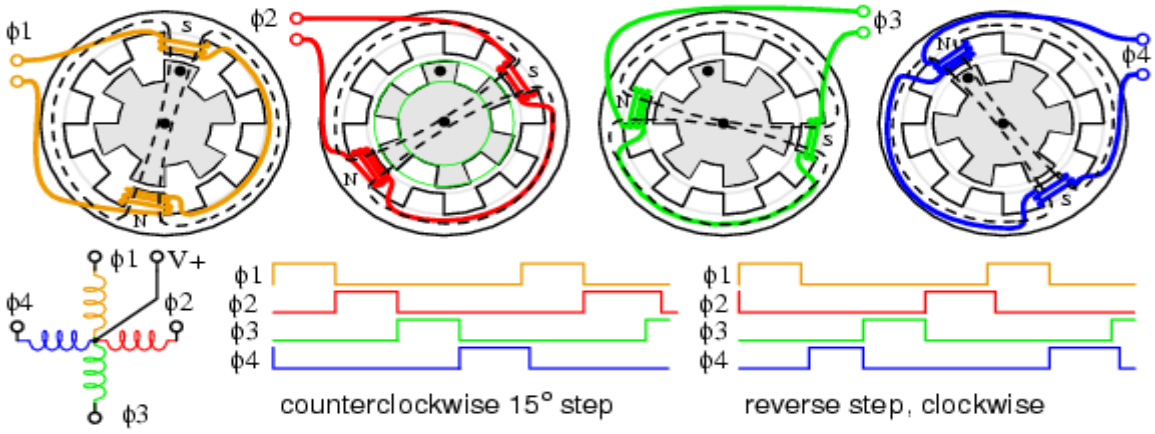
## 7.1 Unipolaire type

De unipolaire stappenmotor bezit 2 wikkelingen met elk een middenaftakking en heeft dus **6 aansluitingen**.

De middenaftakkingen (*common*) worden **permanent aan één kant van de voedingsspanning** aangesloten. In sommige gevallen zijn beide common-aansluitingen samengenomen en heeft de motor 5 aansluitingen.

De 4 andere aansluitingen worden door de besturing steeds wel of niet met het andere eind van de spanning verbonden. Op deze wijze worden **hoogstens 2 halve wikkelingen tegelijk bekrachtigd**, zodat op enig gegeven moment **minstens de helft van de wikkelingen ongebruikt blijft**.

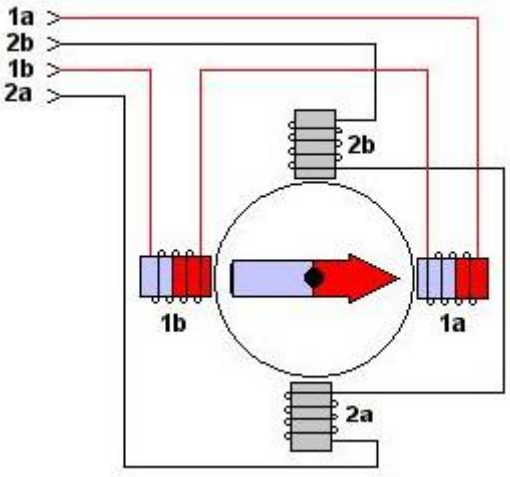
Daardoor is een compacte bouw niet mogelijk. Het voordeel is echter de relatief **eenvoudige besturing**; er zijn maar 4 vermogenstransistoren nodig. Met de komst van de hoog-geïntegreerde elektronica is dit echter niet langer een economisch voordeel; dit **type motor raakt in onbruik**.




Deze figuur toont de 4 stappen van de unipolaire stappenmotor. Merk op dat elke spoel is aangesloten met 1 zijde aan de V+. Dit is dus een 5-draads verbinding. Je kan dus slechts aan 1 kant van de spoel maar een stroom aanbieden. Deze zal nooit in de andere richting vloeien. **Unipolair** dus. Merk op dat de common aan de massa moet hangen om bovenstaand tijdsdiagram te laten werken (Het lijkt een foutje in tekening maar je moet denken dat je een **transistor** stuurt. Deze geeft het **geïnverteerd effect** )!

## 7.2 Bipolaire type

De bipolaire stappenmotor bezit **2 wikkelingen zonder middenaftakking** en heeft dus 4 aansluitingen. Tegenwoordig is dit het **gangbare type**. De wikkelingen worden **in beide richtingen bekrachtigd** en moeten daarom elk in een transistorbrugschakeling geplaatst worden. Zie hiervoor onze L293D (H-brug).




Rood = zuid, Blauw = noord

Noord en zuid trekken elkaar aan

Zie de rechterhandregel om te achterhalen hoe de stroom vloeit, en zo ook de magneten worden beïnvloed in de stappenmotor.

(vingers over spoel laten bewegen in richting stroom, duim wijst naar noorden)

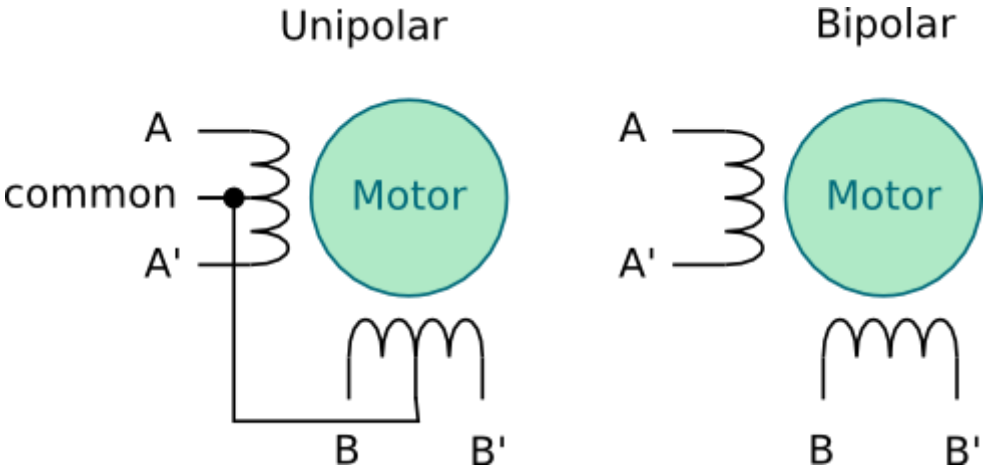
https://phet.colorado.edu/en/simulation/legacy/magnets-and-electromagnets

In deze figuur kan je zien dat de spoelen **per 2 gekoppeld** zijn. Als er een positieve spanning is op 1a dan wordt de rotor aangetrokken door 1a en afgestoten door 1b. De kunst is van vele wikkelingen te hebben zodat kleine nauwkeurige stappen kunnen gemaakt worden.

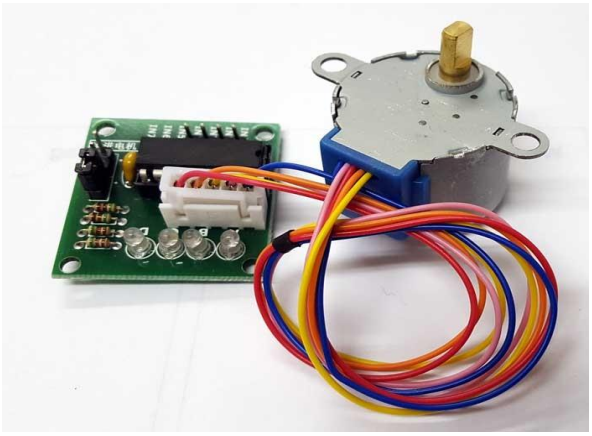
We kunnen hier aan de spoel een stroom in de ene richting (van 1a naar 1b) of in de andere richting (van 1b naar 1a) aanbieden. **Bipolair** dus.

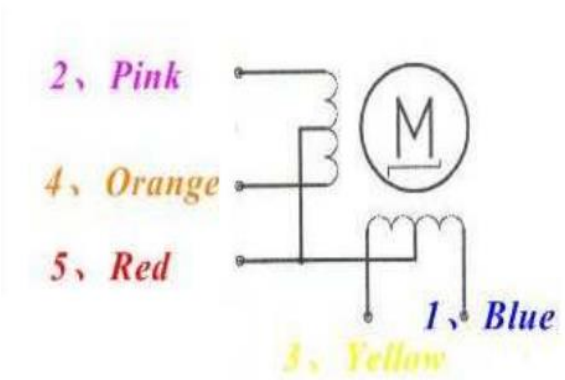
## 7.3 Hybride stappenmotoren?

Tegenwoordig bestaan er motoren die zowel als unipolaire en bipolaire stappenmotor kunnen aangesloten worden. Het is kwestie van de middenaftakkingen wel of niet te gebruiken. Belangrijk wel is dat de **2 middenaftakkingen aan de buitenkant** van de motor zitten. Anders kan je deze niet aanpassen.

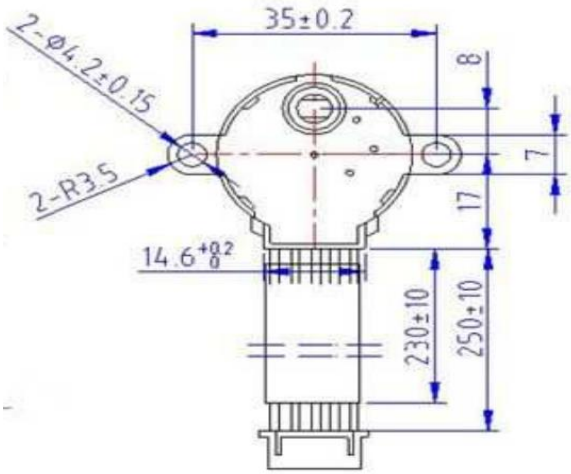


Onze **stappenmotor 28BYJ-48** is een motor met 5 draden. Ze hebben intern in de motor **de rode draad** verbonden **met beide middenaftakkingen** van de spoelen. Als je nu een bipolaire stappenmotor wil maken moet je de motor gaan openen en de rode draad aanpassen. Hier zijn handleidingen van te vinden op internet. Wij gaan hem **gebruiken als unipolaire motor**.






| | |
|---|---|
| Rated voltage : | 5VDC |
| Number of Phase | 4 |
| Speed Variation Ratio | 1/64 |
| Stride Angle | 5.625°/64 |
| Frequency | 100Hz |
| DC resistance | 50Ω±7%(25℃) |
| Idle In-traction Frequency | > 600Hz |
| Idle Out-traction Frequency | > 1000Hz |
| In-traction Torque | >34.3mN.m(120Hz) |
| Self-positioning Torque | >34.3mN.m |
| Friction torque | 600-1200 gf.cm |
| Pull in torque | 300 gf.cm |
| Insulated resistance | >10MΩ(500V) |
| Insulated electricity power | 600VAC/1mA/1s |
| Insulation grade | A |
| Rise in Temperature | <40K(120Hz) |
| Noise | <35dB(120Hz,No load,10cm) |
| Model | 28BYJ-48 – 5V |




➔ Hoe stuur je een **unipolaire stappenmotor** aan?

We kunnen makkelijk tot **250mA per winding** aan stroom gaan verbruiken. Dus we kunnen de motor niet rechtstreeks aansluiten op de Arduino (25mA max per pin).

Omdat we hier 4 windingen willen aansturen gaan we niet met aparte MOSFETS werken maar neem we een kant en klaar bordje waar de **ULN2803 driver** op zit.

**ULN2803A**
**DARLINGTON TRANSISTOR ARRAY**

SLRS049C – FEBRUARY 1997 – REVISED AUGUST 2004

- **500-mA Rated Collector Current (Single Output)**
- **High-Voltage Outputs . . . 50 V**
- **Output Clamp Diodes**
- **Inputs Compatible With Various Types of Logic**
- **Relay Driver Applications**
- **Compatible with ULN2800A Series**

**description/ordering information**

The ULN2803A is a high-voltage, high-current Darlington transistor array. The device consists of eight npn Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads. The collector-current rating of each Darlington pair is 500 mA. The Darlington pairs may be connected in parallel for higher current capability.

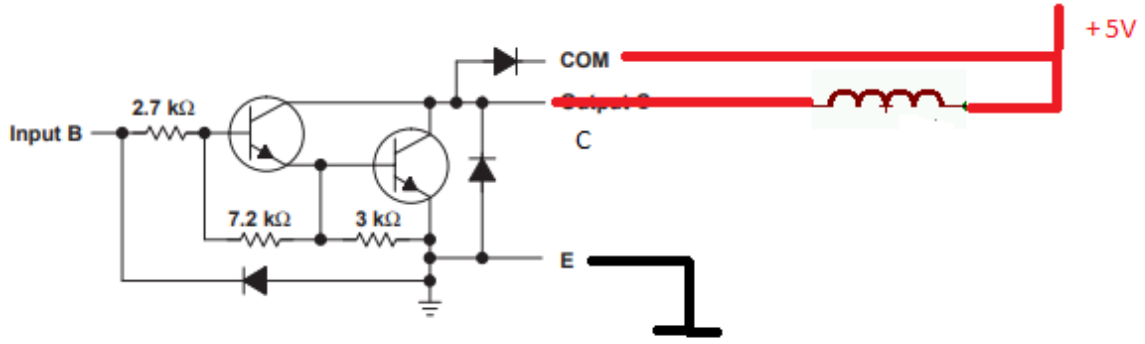
**DW OR N PACKAGE**
**(TOP VIEW)**

| | | | |
|---|---|---|---|
| 1B | 1 | 18 | 1C |
| 2B | 2 | 17 | 2C |
| 3B | 3 | 16 | 3C |
| 4B | 4 | 15 | 4C |
| 5B | 5 | 14 | 5C |
| 6B | 6 | 13 | 6C |
| 7B | 7 | 12 | 7C |
| 8B | 8 | 11 | 8C |
| GND | 9 | 10 | COM |

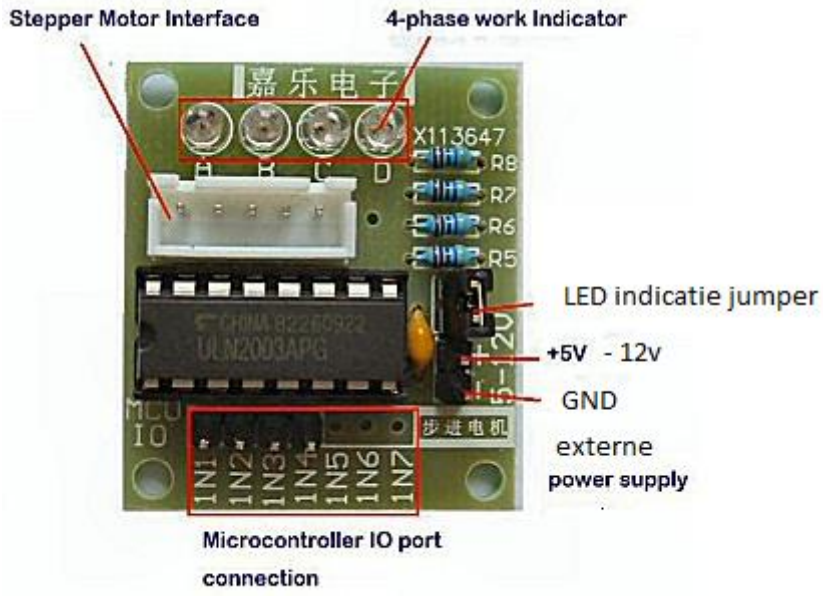
Dit is een driver chip met **open collectoren**. De rode draad (middenaftakking) van de motor gaan we aan de +5V hangen. Wanneer we nu 1 van de transistoren in de ULN2803A aansturen trekt deze de 5V via 1 van de spoelen naar massa. Dus laag actief denken!

De COM pin hangen we aan de +5V. Deze zorgt ervoor dat er een beveiliging is van de **overspanning (> 5.7V)** door bijvoorbeeld de opwekking van **inductie** van de motoren op de collector pinnen (uitgangen van de chip). De diode in de chip is dus een **overspanningsdiode**.

De diode tussen de emitter en collector is ter bescherming van de **negatieve spanningen (< 0.7V).**




De ULN2803 zit reeds op een voorgemaakte PCB

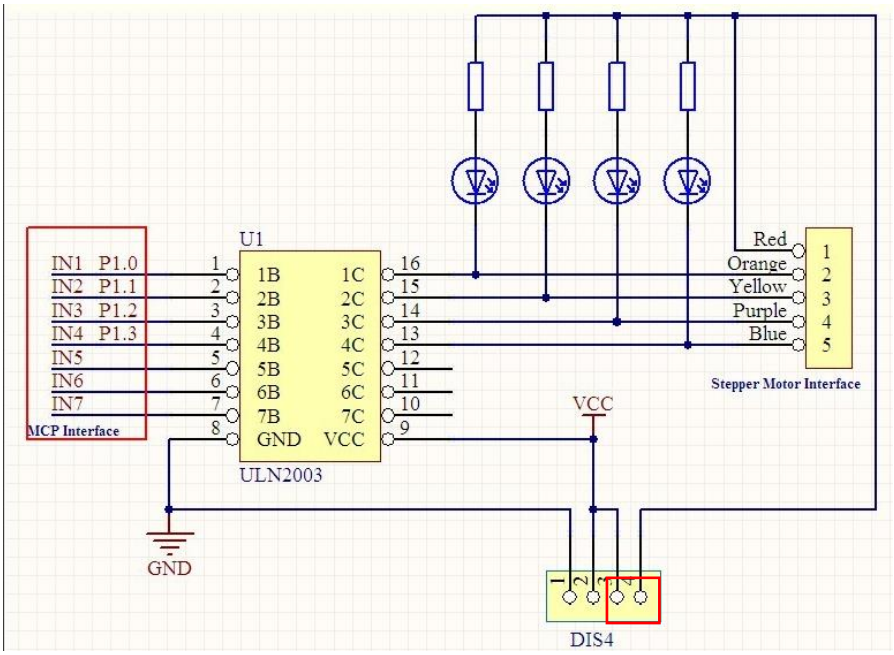



Zorg dat de jumper zo kort mogelijk zit aan de kant van de weerstanden!

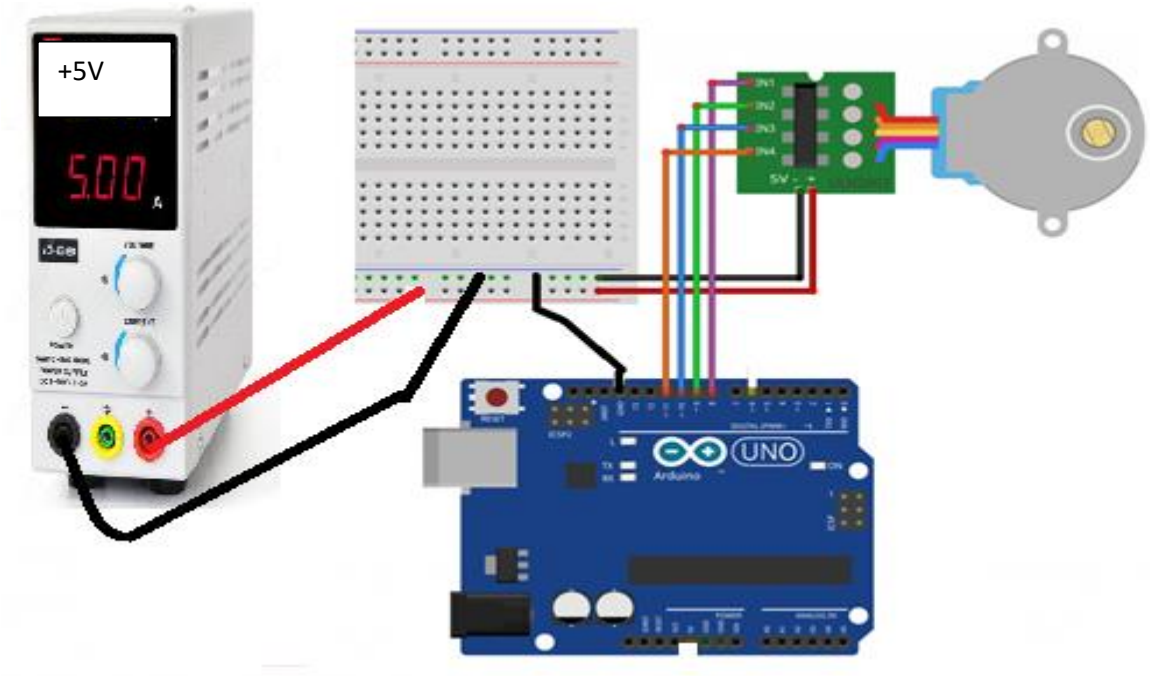
De jumper dient enkel om de LEDs te activeren.

De GND sluit je aan op de linker pin 1, de +5V sluit je aan (via een **externe voeding**), op de 2de pin.

De UNO kan niet voldoende stroom leveren om de stappenmotor te voeden!




Schema van het ULN2803 driverbordje




Sluit de UNO zo aan op de stappenmotor.

| | |
|---|---|
| +5V | Aan externe +5V voeding |
| GND | Aan externe voeding en UNO (referentie) |
| IN1 (oranje op motor) | Pin 8 UNO |
| IN2 (geel op motor) | Pin 9 UNO |
| IN3 (roze op motor) | Pin 10 UNO |
| IN4 (blauw op motor) | Pin 11 UNO |

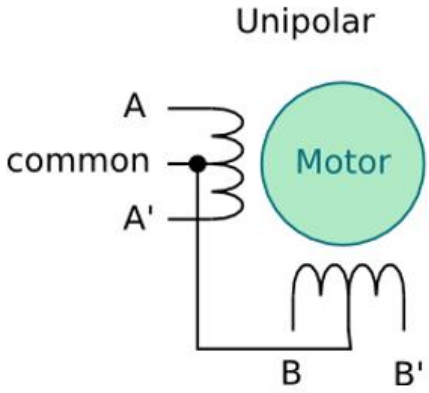
➔ Welke signalen moeten we aanleggen aan de UNIPOLAIRE servo motor?

We kunnen op 3 manieren de servo motor aansturen:

**One phase full-step** mode

| Step | A | B | C | D |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |

Elke motor winding wordt 1 keer aangesproken per keer. Dit heeft het **nadeel** dat het **draaimoment** (torsie of koppel, kracht op de as) **laag** is.




**Two Phase full-step** mode

| Step | A (IN1) | B (IN2) | C (IN3) | D (IN4) |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 0 | 0 | 1 | 1 |

2 windingen worden tegelijk bekrachtigd. Dus **meer draaimoment** mogelijk!

**Two Phase half-step** mode

| Step | A | B | C | D |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 1 |

Af en toe 2 windingen, af en toe 1 winding, dus minder torsie. Maar we hebben nu 8 stapjes i.p.v. 4, dus meer precisie.

De keuze is dus aan de maker om de voor hem beste methode te gebruiken.

➔ Hoe ziet de 2 phase full-step code eruit?

stappenmotor_2phase_FS

```
//2 FASE FULL STEP
#define IN1 8  //overal waar IN1 wordt gebruikt is dit 8
               // oranje
#define IN2 9 //geel
#define IN3 10 //roze
#define IN4 11//blauw

int RPM = 12;  //rotaties per minuut, snelheid
int StepsPerCycle = 512; //aantal stappen per 360 graden
int StepDelay; // in ms
int FullMode[4] = {B01100,B00110,B00011,B01001};
//zie 2 phase full-step tabel voor de bits
//dit is een array met de 4-bits voorstelling van de
//signalen op de windingen

void CLOCKWISE (int count)
  {
    for (int j = 0; j < count; j++) //doe de 4 stapjes "count" keer, zie parameter
                                    //van clockwise
    {
      for (int i = 4; i >= 0; i--) //steeds de 4 combinaties afgaan
      {
        SendPulse(i);
        delayMicroseconds(StepDelay);
       }
    }
  }

void SendPulse(int k)
{
  digitalWrite(IN1,bitRead(FullMode[k],0)); //haal bit met positie 0 uit variabele k van array en zet op pin IN1
  digitalWrite(IN2,bitRead(FullMode[k],1)); //haal bit met positie 1 uit variabele k van array en zet op pin IN2
  digitalWrite(IN3,bitRead(FullMode[k],2)); //haal bit met positie 2 uit variabele k van array en zet op pin IN3
  digitalWrite(IN4,bitRead(FullMode[k],3)); //haal bit met positie 3 uit variabele k van array en zet op pin IN4

}


void setup() {

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  StepDelay = 29300 / RPM;
  //meer details over de berekening in het boek "motorcontrol" van elektor

  //laat de motor met de clock meedraaien

}

void loop() {

  CLOCKWISE(2);//ga 2 toeren clockwise
  delay(100);
}
```

We maken hier weer gebruik van een array om de 4 binaire combinaties van de 2 fase full step tabel in de code makkelijk te bekomen. Hier kan je de tabel ook makkelijk aanpassen naar andere combinaties.

Elke bit uit 1 plek van de array wordt dan naar de juiste pin gestuurd (zie SendPulse functie).

De RPM is een vaste berekende waarde. De snelheid van de motor hangt af van de tijd tussen 2 pulsen. De formules hier achter kan je terugvinden in het elektor boek “motorcontrol”.

➔ Kan de code ook makkelijker zijn voor de steppermotor?

steppermotor_step_lib

```
#include <Stepper.h>

const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
  // set the speed at 60 rpm:
  myStepper.setSpeed(60);
  // initialize the serial port:
  Serial.begin(9600);
}

void loop() {

  // step one revolution  in one direction:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(500);

  //dit deel werkt niet?

  // step one revolution in the other direction:
  Serial.println("counterclockwise");
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

We kunnen inderdaad gewoon gebruik maken van een bestaande library “Stepper”. Nu moeten we niet wakker liggen over de aansturing van de bits. Alle rekenwerk is op de achtergrond gebeurt in de library. Enkel nog de StepsPerRevolution instellen, het aantal RPM (rotations per minute) en je bent vertrokken.

Merk op dat er ook code voorzien is om de motor in de tegenrichting te laten draaien. Deze werkt echter niet?

Info van deze code kan je hier vinden.

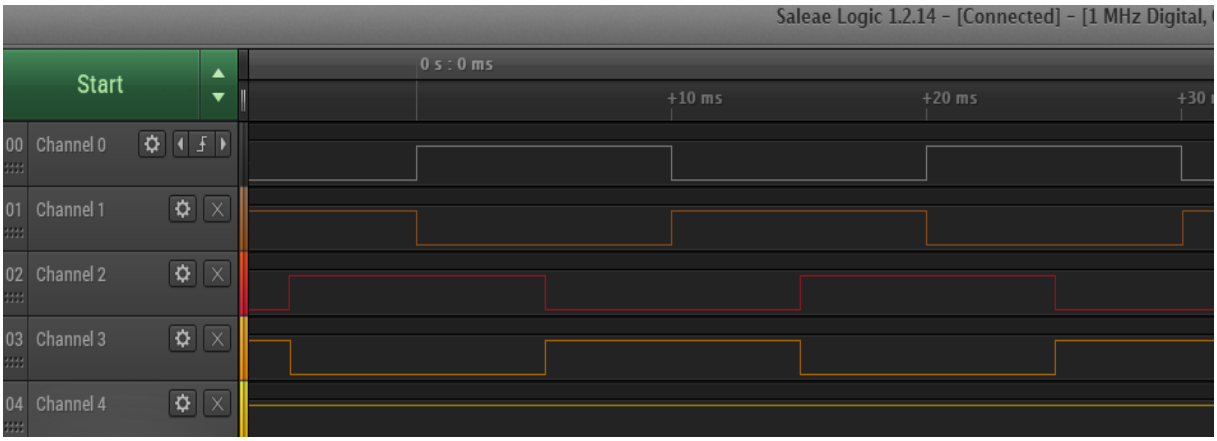
➔ Dit code voorbeeld voorziet dat de steppermotor snelheid kan aangestuurd worden met een joystick, aangesloten op A0. Via de map() functie wordt de motorspeed waarde geschaald naar een waarde tussen 0 en 100.

stepper_analoge_pot

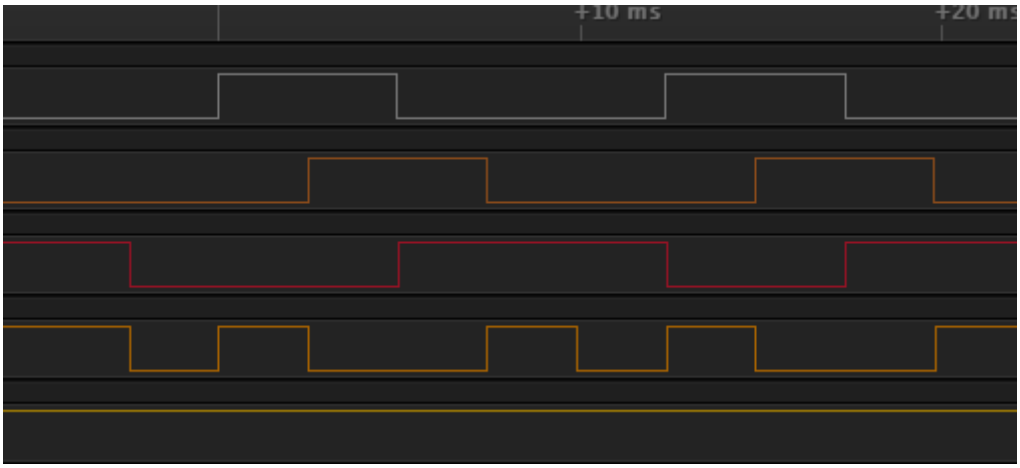
```
const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor


// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

int stepCount = 0;  // number of steps the motor has taken

void setup() {
  // nothing to do inside the setup
}

void loop() {
  // read the sensor value:
  int sensorReading = analogRead(A0);
  // map it to a range from 0 to 100:
  int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
  // set the motor speed:
  if (motorSpeed > 0) {
    myStepper.setSpeed(motorSpeed);
    // step 1/100 of a revolution:
    myStepper.step(stepsPerRevolution / 100);
  }
}
```

➔ **Uitdagingen:**

- Laat de motor 2 fase full step in de tegenrichting draaien.
- Laat de motor 2 fase half step draaien.
- Doe een meting met de **logic analyzer** op jouw 4 stepper motoruitgangen. Zo begrijp je nog beter de werking.

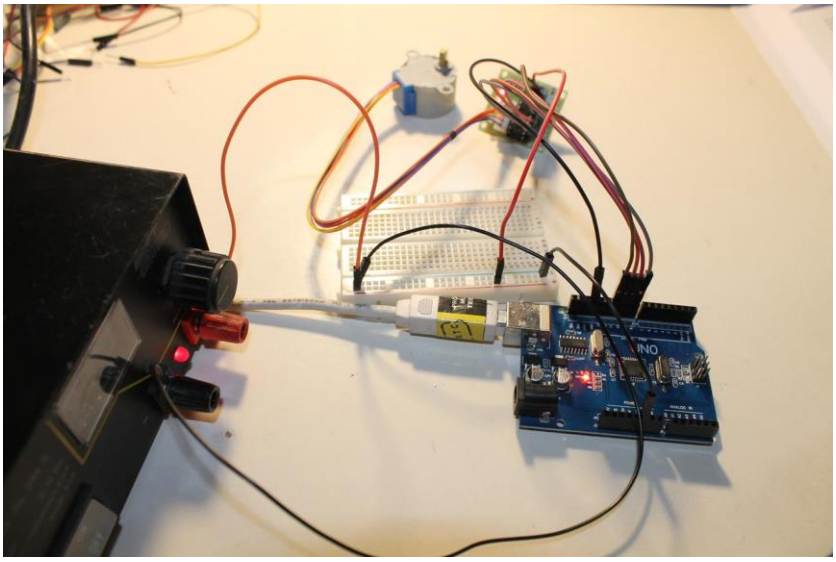


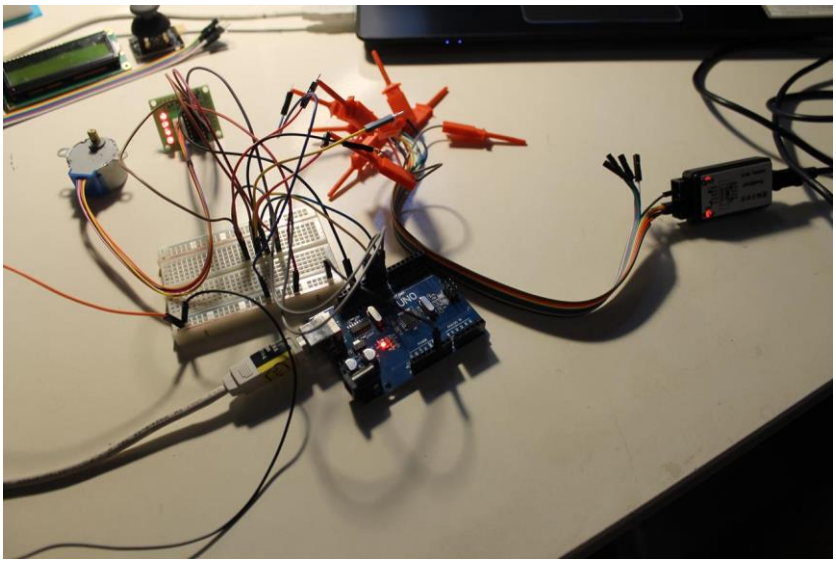

Via de logic analyzer meting kunnen we mooi de combinaties aflezen (vertikaal): 0110 , 1010, 1001, 0101 (dit zijn de combinaties van de Stepper library).




Dit is de meting als we 2 fase full step meten op onze zelfgemaakte code (zonder stepper library).

Nu zie je mooi vertikaal 1001, 1100, 0110 en 0011 verschijnen. Dan is er een moment dat niet duidelijk is (de microprocessor springt dan terug naar het begin van de for lus en begint opnieuw).





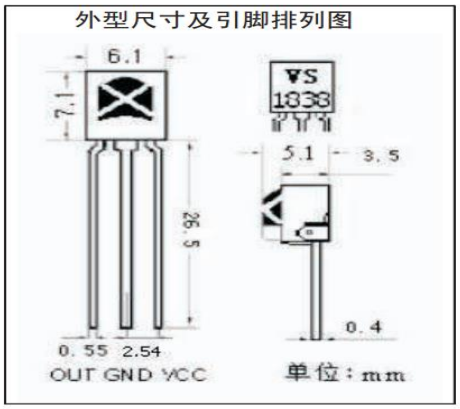
Opstelling zonder en met logic analyser.

## 8. IR of RF signalen sturen/ontvangen?

Nu we heel wat motoren kunnen aansturen met joysticks en bluetooth (zie basiscursus), willen we de servo motoren ook nog op een andere manier draadloos leren aansturen.

We gaan hiervoor aan de slag met een IR of RF ontvanger (**infrarood of radio frequentie**) en een afstandsbediening.

We gaan gebruik maken van een **VS1838B 38kHz RF** ontvanger.




De behuizing van de sensor is zo voorzien dat enkel het 38KHz **RF signaal** wordt gefilterd.

Deze ontvanger heeft 3 aansluitingen: **uitgangssignaal (OUT), massa (GND) en de +5V.**

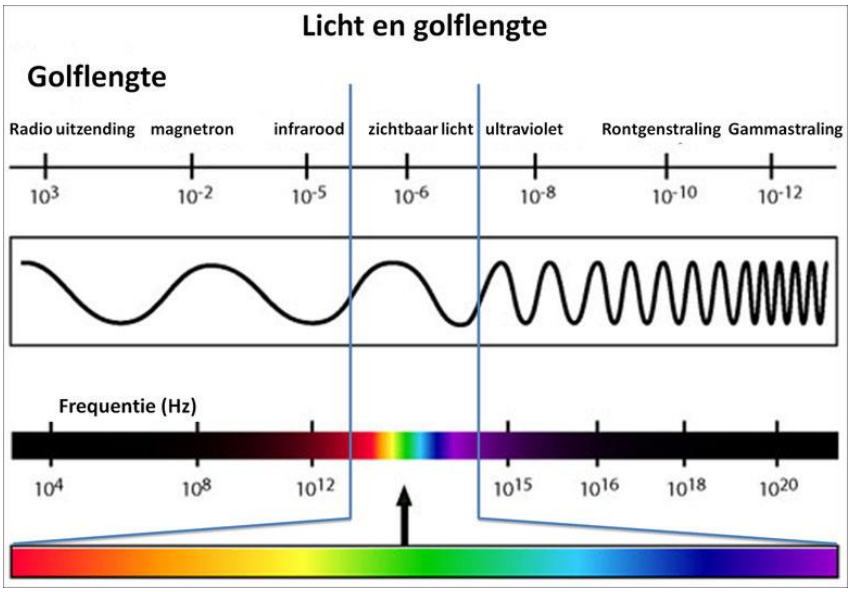
➔ **Hoe weten we dat dit een RF en geen IR signaal is?**

- Als je een universele afstandsbediening wilt aankopen, bedenk dan dat er 2 soorten afstandsbedieningen bestaan: afstandsbedieningen die werken op RF (Radio Frequency) of IR (Infrarood). Het is dus belangrijk om dit verschil te zien zodat je geen miskoop doet.
- Een **RF afstandsbediening hoef je niet te richten naar je apparatuur om deze te bedienen**. Een IR afstandsbediening moet je wel richten naar je toestel. Dat is al een belangrijk onderscheid. Maar hoe kun je nu eenvoudig het verschil zien? Want met het blote oog kun je deze signalen niet zien.
- **Richt met je digitale camera naar je afstandsbediening** terwijl je op een knop drukt. Als je een **IR afstandsbediening hebt, zul je dit zichtbaar krijgen op je digitale camera.**

- Je kunt deze techniek ook gebruiken, als je toestel met IR ontvanger niet meer reageert op je afstandsbediening en je wilt eenvoudig zien of je afstandsbediening nog signalen uitzendt.

➔ **Welke type afstandsbediening hebben wij (test uit)? …….**

**FM signaal of CW ?**




Golflengte = c / f = 300 000 000 / 38 000 = 300 000 / 38 **= 7850 Hz**

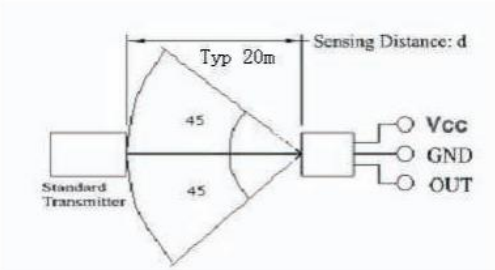
Uit deze berekening merken we dat we eerder in het RF (**radio gebied)** zitten dan in het IR gebied.

En dit klopt ook, maar de datasheet zegt het volgende:

Infrared Receiver Module 红外线接收器

型号: VS1838B

7. 接收角度:

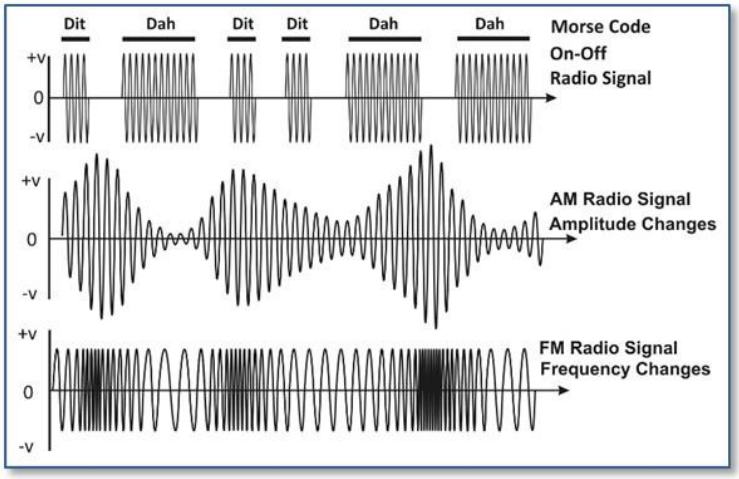



8. 推荐使用条件:

| 项目 | 符号 | Min | Typ | Mnx | 单位 |
|---|---|---|---|---|---|
| 工作电压 | Vcc | 2.7 | ------ | 5.5 | V |
| 输入频率 | FM | | 38 | | kHz |
| 工作温度 | Topr | -20 | 25 | 80 | ℃ |

Het is verwarrend om te lezen dat het een IR signaal is in het radio FM gebied?

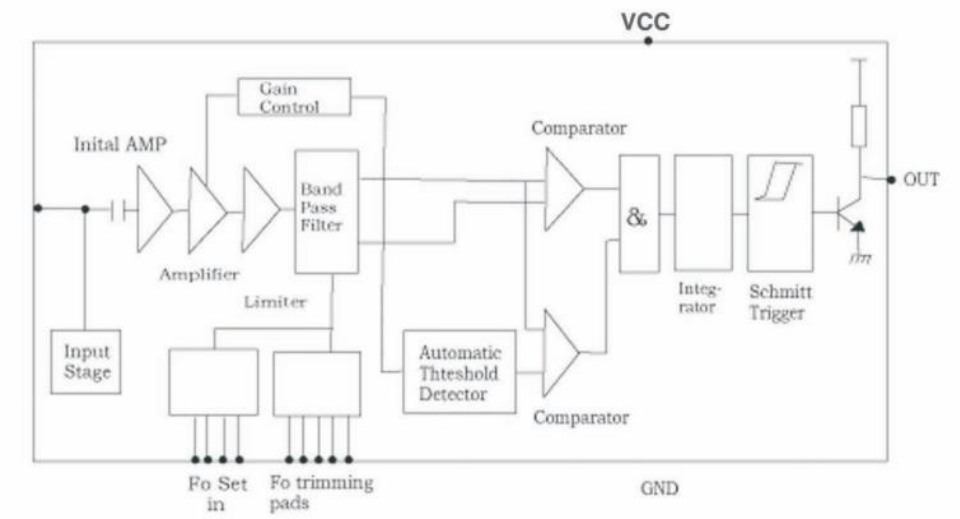
Het gekke is dat het signaal steeds 1 frequentie is die aan of uit wordt gezet. Dan spreken we eerder van een **CW (Continuous wave) of morse code** signaal dan een FM radio signaal.




Conclusie: we sturen radio golven i.p.v. IR golven in CW modulatie 😊

Wat zit er in de VS1838B?

In de ontvanger zit een IC met een volledige regelkring. Het RF signaal dat binnenkomt, **gemoduleerd op een frequentie van 38kHz**, wordt omgezet tot een eenvoudige 0 of 1 aan de open collector uitgang. We kunnen deze direct aansluiten op de UNO.




➔ Leg de werking uit van de verschillende onderdelen en noteer ook het verband tussen de onderdelen. Duid op de bovenste figuur de nummers aan en welk signaal er geproduceerd wordt:

1. Condensator aan ingang:

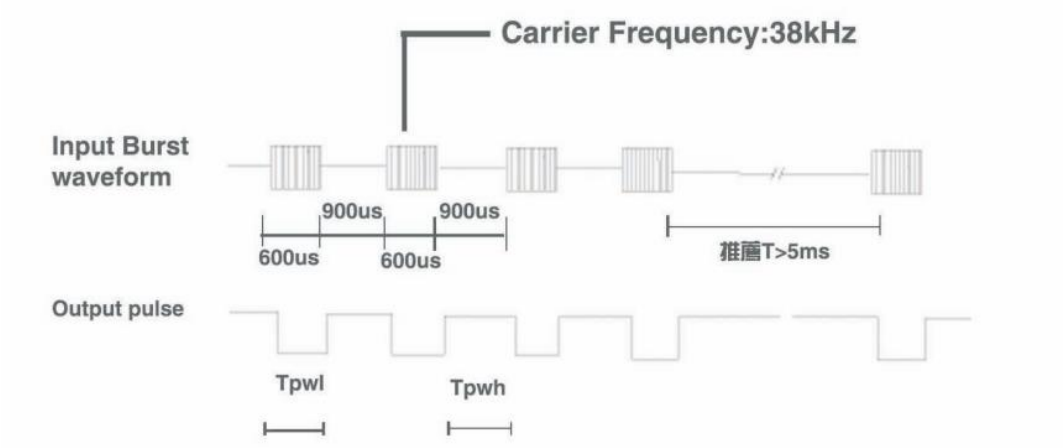
2. Amplifier en AGC :

3. Band Pass Filter:

4. Integrator:

5. Schmitt-Trigger:

6. Transistor:



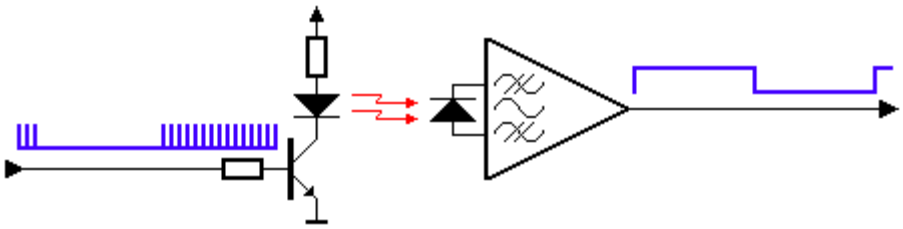

Op deze figuur zie je hoe de draaggolf van 38kHz wel of niet wordt verstuurd. Als deze verstuurd wordt levert dit een “0” op aan de uitgang. Als deze niet verstuurd wordt krijgen we een “1”.
Gelukkig moeten we in Arduino ons hier weer geen zorgen over maken. We gaan gebruik maken van de **IRremote library**. Deze zal de binaire code die binnenkomt weer omzetten in begrijpbare en hapklare data voor ons. Gek dat dit niet de RFremote lib noemt?

➔ Waarom een draaggolf van 38kHz?

We willen **voorkomen dat ons signaal beïnvloed wordt door invloeden van buitenaf**. In het gewone zonlicht zit bijvoorbeeld ook UV. Ook in de verschillende soorten lampen die overal schijnen. Om ervoor te zorgen dat het IR / RF signaal hier geen invloed van heeft moet men verschillende technieken uit de kast halen om dit op te vangen. Bij RF storen nog eens alle andere elektronische apparaten die in de buurt staan (zie EMC storingen).

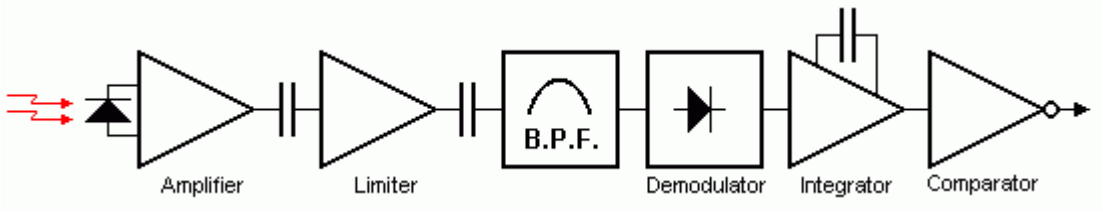
De eerste truc is de **puls sturen via een hogere frequentie**. De pulsen bestaan letterlijk uit een stukje draaggolf van 36KHz. We passen hier dus een **FM modulatietechniek** toe, meer bepaald **CW** .

De ontvanger (transmitter) zal zich alleen richten op deze frequentie en deze doorlaten:

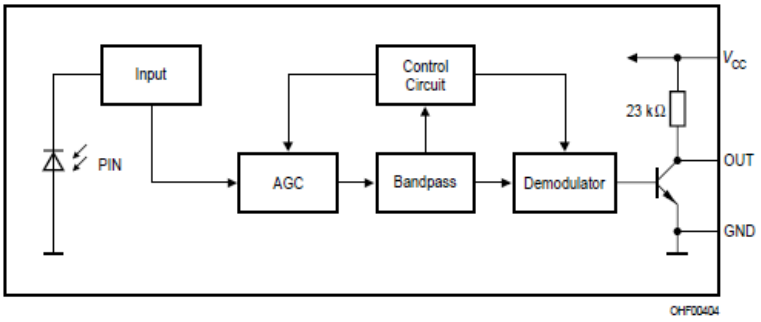


In de bovenstaande figuur zie je hoe de zender een signaal van Marks (de transmitter staat aan) en Spaces (transmitter staat default af) stuurt. Werkelijk worden er kleine draaggolf pulsen verstuurt. De RF ontvanger pakt deze frequentie op (tussen 30 en 60KHz) en maakt er terug een volle puls van.

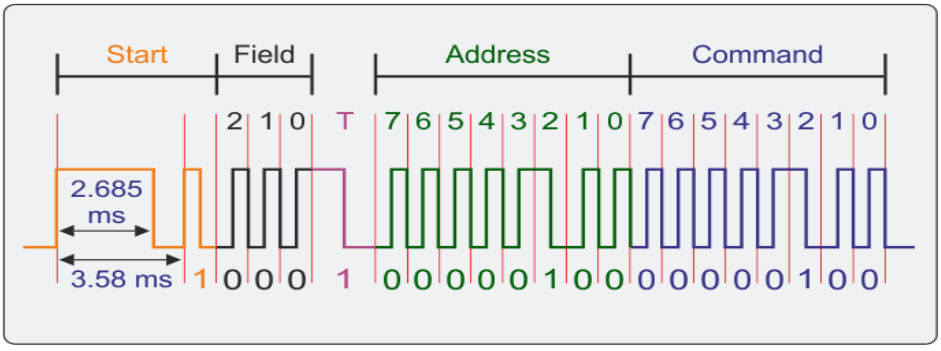
De ontvanger ziet er in een vereenvoudigt blokschema als volgt uit:




Eerst wordt het ontvangen signaal versterkt en begrensd om constante pulsen te bekomen. Dan wordt de AC van 30 tot 60KHz uitgefilterd. De demodulator (pulsen uit draaggolf afleiden), integrator (pulsen omzetten naar zaagtand) en comparator (zaagtand omzetten naar pulsen) zorgen ervoor dat wanneer er pulsen zijn de uitgang hoog wordt gemaakt.




➔ Waar zit de data verstopt in de rij bits?



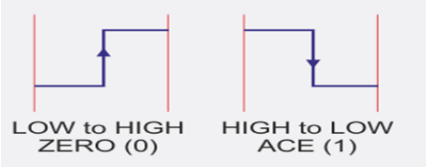

Bij Philips RC6 ziet de verstuurde code er zo uit. Merk op dat elke bit voortkomt uit een burst van 36kHz. De data komt er ook geïnverteerd uit als je zou meten met de scope (open collector).

Opbouw signaal:

Weet dat elke bit een lengte heeft van 0.895ms = 895us.

1. De **start bit** bestaat uit een hoge puls van 2.685ms (4 bits lang) gevolgd door een lage periode van 0.895ms (1 bit). Daarna komt nog 1 normale startbit van 895us. Deze is altijd “1”. Dit betekent dat deze altijd van “hoog” naar “laag” gaat.

   We werken hier met de **MANCHESTER code**. Dat betekent dat de puls veranderd in het midden van de periode. Een “1” is een “hoog” naar “laag” moment en een “0” is een “laag” naar “hoog” moment. Als nu de klok **frequentie een beetje “out of sync” is door omstandigheden, dan is de klok toch nog binnen de bit tijd.**




MANCHESTER code

2. Na de startbit volgen **3 field bits.** Ze zijn altijd “0” en zijn elk 1 bit lang . Dit is voor de kloksynchronisatie van dit asynchrone systeem. De klok zit namelijk in het data signaal verstopt en wordt niet synchroon meegestuurd zoals bij I2C.

3. Dan volgt de **toggle** bit (2 bits lang). Deze bit dient om de software te laten detecteren of de knop langer als 1 code ingedrukt blijft. Wanneer je m.a.w. **1 knop blijft indrukken zal de toggle bit niet veranderen.** Pas als je een andere knop indrukt verandert de toggle bit. Deze aanpassing moet je in software opvangen.

   Let wel op: de lengte van de laag en hoog zijn elk 1 bit lang. Dus de ganse toggle bit is 2 x 895us = 1.79ms.

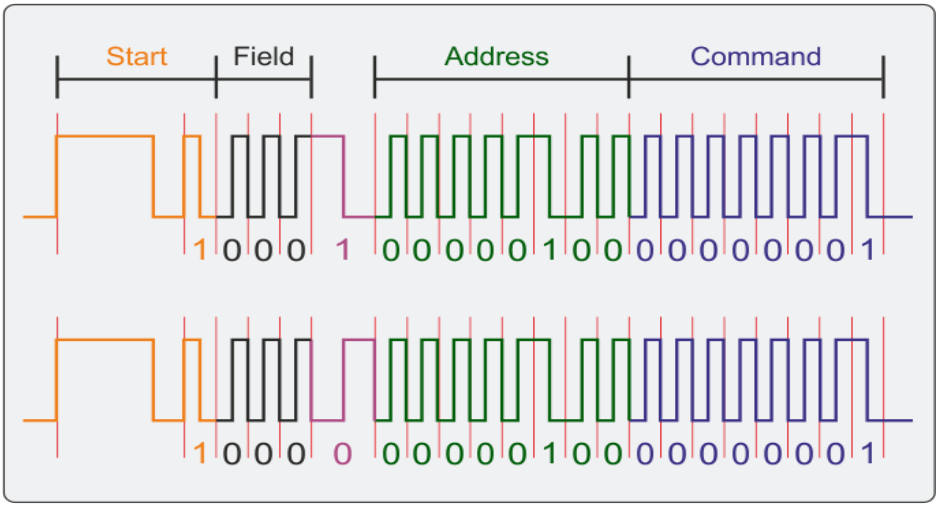
4. Dan volgen **de 8 address bits** (MSB First). Elk systeem heeft een uniek adres. Zo luistert bijvoorbeeld de DVD niet naar de afstandsbediening van de CD speler. Hier volgt de lijst:

| System Number | Description | Command Tables |
|---|---|---|
| 0 | TV 1 (TV receiver 1) | 2, 3, 4a |
| 1 | TV 2 (functions and command numbers as system 0) | 2, 3, 4a |
| 2 | Txt (teletext) | 2, 3, 5 |
| 3 | Extension to TV 1 and TV 2 | 2, 4b |
| 4 | LV (LaserVision player) | 2, 3, 6 |
| 5 | VCR 1 (video cassette recorder 1) | 2, 3, 7a |
| 6 | VCR 2 (functions and commands as system 5) | 2, 3, 7a |
| 7 | Reserved | |
| 8 | Sat 1 (satellite TV receiver 1) | 2, 3, 8 |
| 9 | Extension to VCR 1 and VCR 2 | 2, 7b |
| 10 | Sat 2 (functions and commands as system 8) | 2, 3, 8 |
| 11 | Reserved | |
| 12 | CD-Video (compact disc video player) | 2, 3, 9 |
| 13 | Reserved | |
| 14 | CD-Photo (photo on compact disc player) | 2, 3, 10 |
| 15 | Reserved | |
| 16 | Preamp 1 (audio preamplifier 1) | 2, 11 |
| 17 | Tuner (radio tuner) | 2, 12 |
| 18 | Rec 1 (analog cassette recorder) | 2, 13 |
| 19 | Preamp 2 (functions and commands as system 16) | 2, 11 |
| 20 | CD (compact disc player) | 2, 14 |
| 21 | Combi (audio stack or record player) | 2, 15 |
| 22 | Sat (audio satellite) | 2, 16 |
| 23 | Rec 2 (functions and commands as system 18) | 2, 11 |
| 24 | Reserved | |
| 25 | Reserved | |
| 26 | CD-R (compact disc recorder) | 2, 17 |
| 27 | Reserved | |
| 28 | Reserved | |
| 29 | Reserved | |
| 30 | Reserved | |
| 31 | Reserved | |

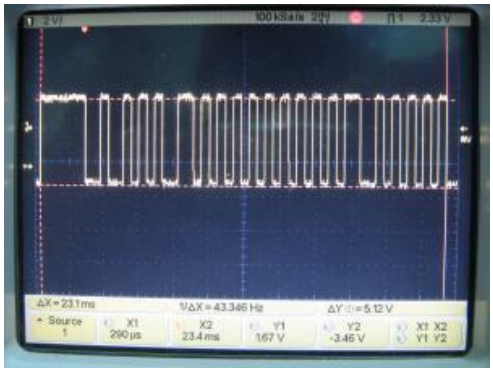
Merk op dat DVD's en TV's op het zelfde adres reageren.

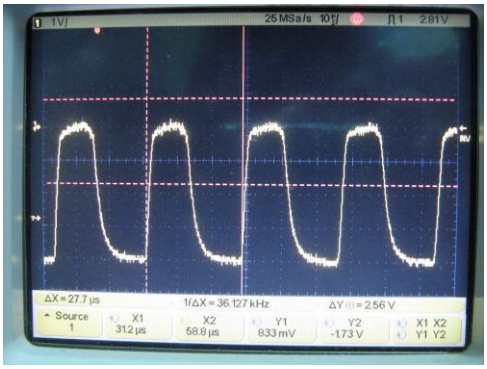
5. Tot slot hebben we nog **8 commando bits** (MSB First). Deze commando's bevatten wat de DVD speler bijvoorbeeld moet doen (eject, play …). Deze lijst vind je terug op wikipedia.

Voorbeeld van code:



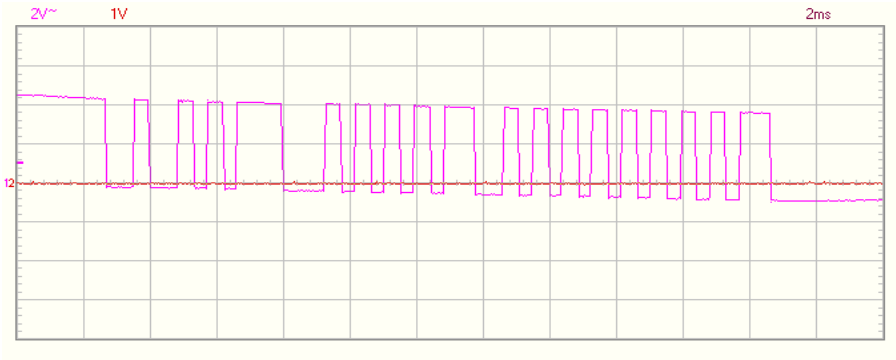

Merk de toggle bit op. Er wordt hier voor de DVD (of laser vision) speler 2 keer op de "1" gedrukt met een tijdspanne tussen.





Links het gemeten RC6 IR signaal op de OUT van een SFH5110-36 ontvanger.

Rechts de burst van 36kHz gemeten na het inzoomen op 1 bit.



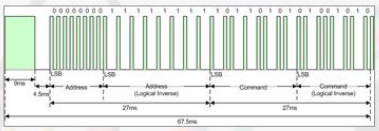
Voorbeelden van andere afstandsbedieningsprotocollen kan je hier vinden:

http://www.diegm.uniud.it/bernardini/Laboratorio_Didattico/2016-2017/2017-Telecomando/ir-protocols.html

Wij hebben vermoedelijk te maken met het NEC protocol in onze afstandsbediening.

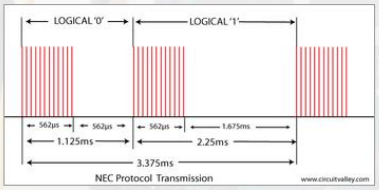
NEC PROTOCOL

**NEC** is maybe the best known one and the **most widespread**. It has been developed by NEC and adopted by many companies (perhaps it's royalty-free, but I didn't find anything certain about this). Its **waveform** is made of a start frame, a bit stream and a final terminator.



The **start frame** is a PWM burst 9ms long, equivalent to 16 times the base timing ***T*** of 562.5us. In this phase the IR Led blinks and then it follows a pause of 4.5ms.

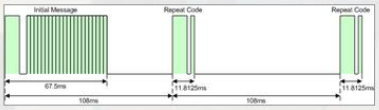
Now the **bitstream** begins. Four bytes are transmitted. Only two of them contain the proper data, the address of the receiver and the command, and they are interleaved with their logical inverse bytes (bitwise not). This is like an error check implementation. All the information bits are streamed LSB first.




A **logical 1** consists of a 562.5us (1x ***T***) pulse burst followed by a pause of 1687.5us (3x ***T***), for a total duration of 2250us (4x ***T***).

A **logical 0** consists of a 562.5us (1x ***T***) pulse burst followed by a pause of another 562.5us (1x ***T***), for a total duration of 1125us (2x ***T***).

Since the bitstream contains also the logical inverse bytes, its **total duration** is constant and lasts 67.5ms plus the terminator which is just another standard pulse of 1x ***T***.



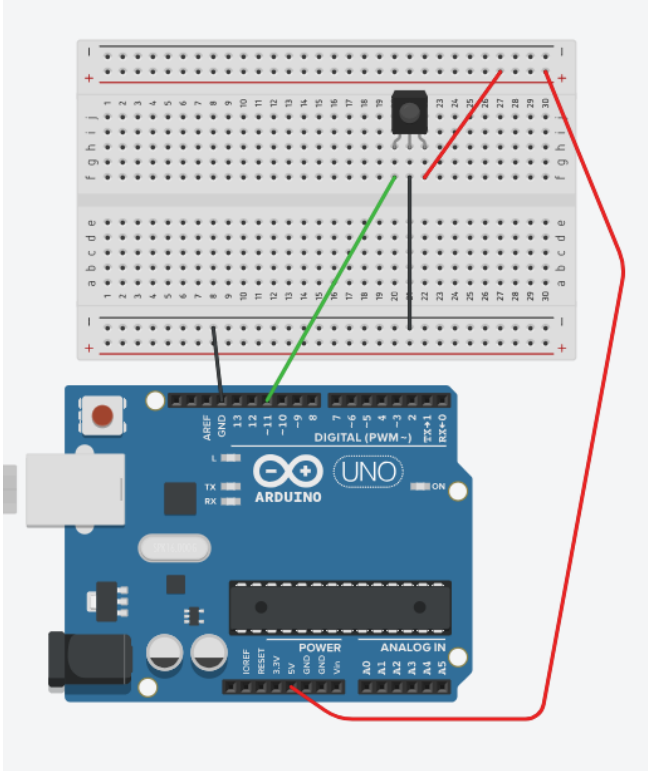

It is stated that **two consecutive streams** must be separated by a total time of 108ms from start edge to start edge.

There is also a dedicated code that means "**repeat** last command". It is made by a 9ms start burst followed by 2.25ms of pause and a standard pulse of 1x ***T***.

Each 562.5us pulse is realized with **PWM** technique with a suggested duty cycle of 1/4 or 1/3. The **carrier frequency** is set at 38KHz.

We gaan het signaal in een verdere uitdaging op een digitale scope proberen zichtbaar te maken en zo het protocol ontleden.

## 8.1 Hoe de IR / RF ontvanger aansluiten op de UNO?



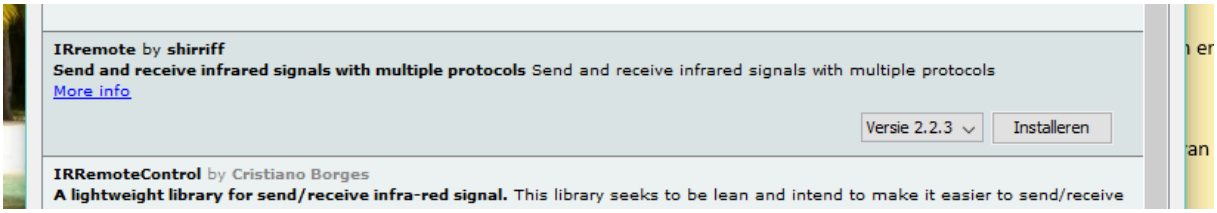

We sluiten de OUT van de ontvanger aan op pin 11 van de UNO.

De GND hangen we aan de massa van de UNO en de VCC aan de +5V van de UNO.

Nu moeten we een library bij installeren. We moeten op zoek gaan in Arduino of de **IRremote** library al is toegevoegd.

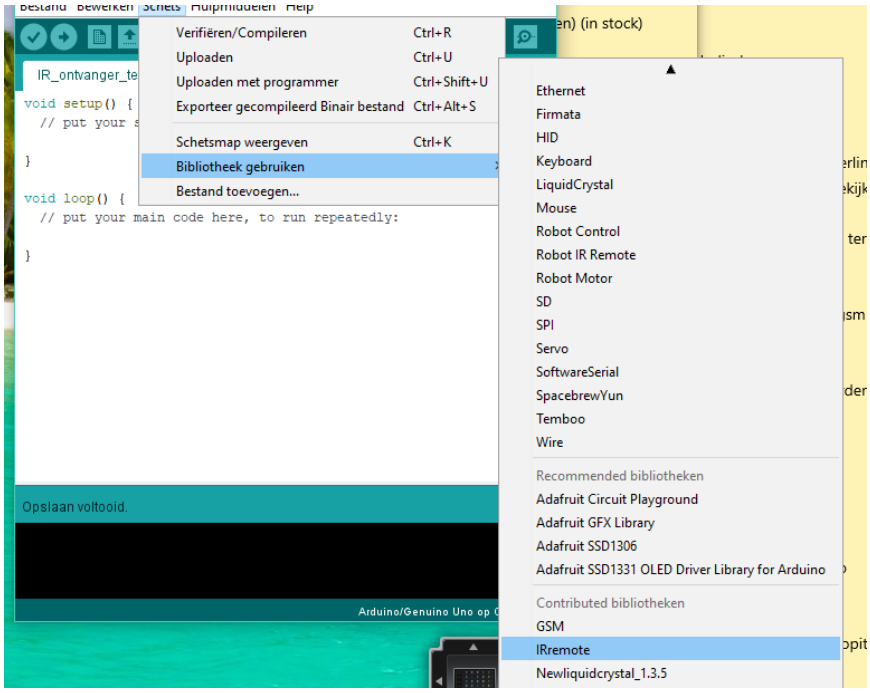
Je kan gaan kijken in **schets -> bibliotheek** gebruiken en kijken of hier IRremote tussen staat.

Je kan ook via **schets -> bibliotheek gebruiken -> bibliotheek beheer** zoeken op IRremote.

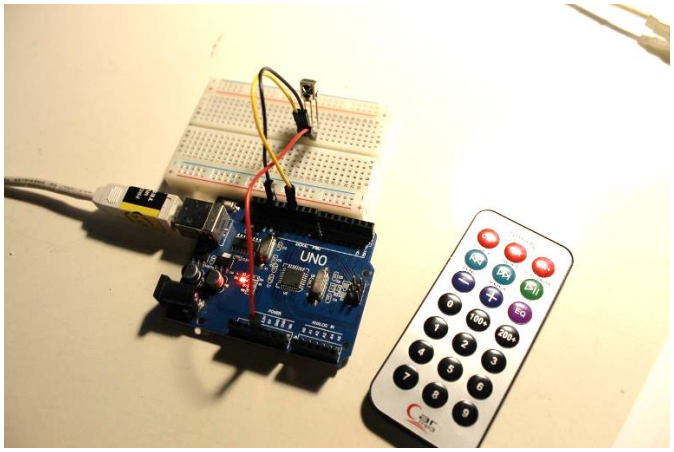



Druk op “more info” en dan op “installeren”.

Na de installatie sluit je dit venster.




Nu moet de IRremote library wel te zien zijn in het lijstje.

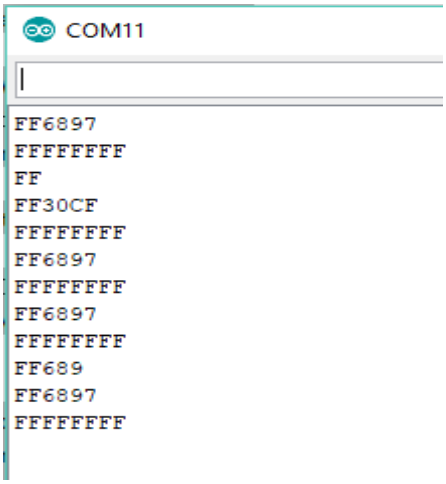


➔ De Arduino testcode

IR_ontvanger_test

```
#include <IRremote.h>

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);//maak object irrecv
decode_results results;

void setup()
{
  Serial.begin(9600); //toon resultaat in serial monitor
  irrecv.enableIRIn(); // start de ontvanger op
}

void loop() {
  if (irrecv.decode(&results)) //als er signaal aan de ingang is
  {
    Serial.println(results.value, HEX);//print HEX versie op serial monitor
    irrecv.resume(); // ontvang volgende waarde
  }
}
```

Wanneer je op de afstandsbediening drukt gaat het signaal ontvangen worden. Naar de UNO wordt nu een reeks bits gestuurd. De if houd in het oog of er een resultaat is binnengekomen. Als dit zo is wordt dit gedecodeerd via de IRremote library. Het resultaat wordt dan in HEX vorm naar de serial monitor gestuurd. Zo kan je voor elke knop een combinatie zoeken en deze ergens noteren voor de volgende software oefening.




Opmerkingen bij code:

- Er wordt pin 11 gekozen voor het aansluiten van de RF ontvanger. Dit mag ook op een andere pin. Het zijn gewone digitale pulsen.
- &results heeft te maken met het adres van results waar naar verwezen wordt. We noemen dit ook een pointer. Hierover kan je meer lezen op https://www.arduino.cc/reference/en/language/structure/pointer-access-operators/reference/
  Een pointer wordt gebruikt bij complexe datastructuren.

Na het drukken op “0” krijg ik de hexadecimale waarde 0xFF6897. **Hexadecimaal** wil zeggen dat bijvoorbeeld de “6” bestaat uit 4 bits 0b0110 maar dat we dit proberen **korter te schrijven**.

We gebruiken in het hexadecimaal stelsel ook de A tot F letter om 10 tot 15 voor te stellen.

15 = 0b1111 = 0xF

Meer over deze omzettingen vind je zeker terug op internet.

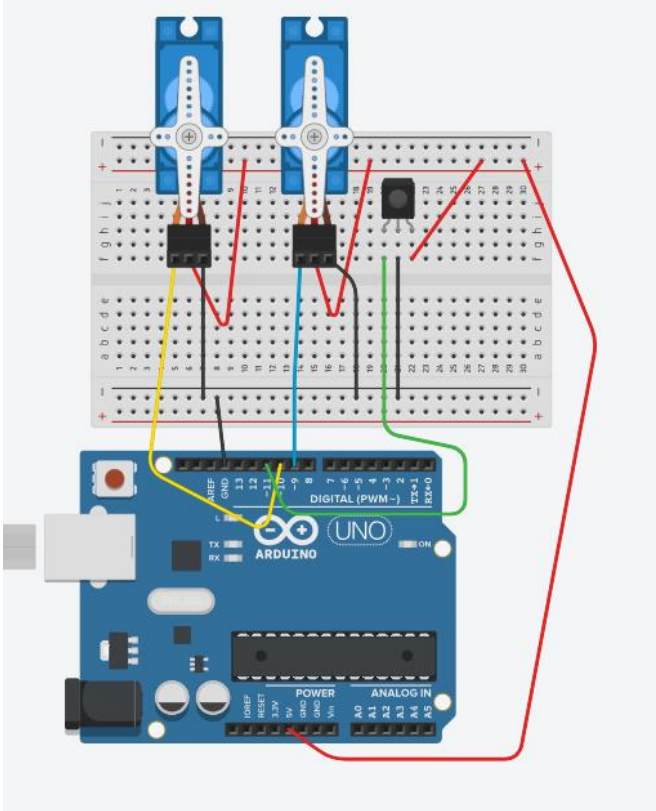
Wanneer je **enkel F-jes** op het scherm ziet dan wil dat zeggen dat je **te lang de knop** van de zender indrukt.

## 8.2 Met de afstandsbediening de pan-tilt opstelling sturen

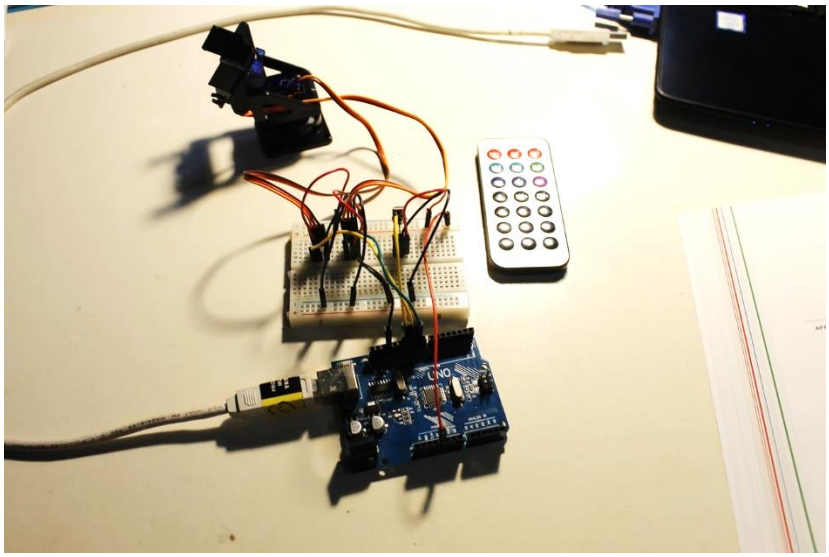
Nu gaan we onze pan-tilt opstelling er weer bij nemen en gaan we de 2 servo's sturen met de afstandsbediening.

Laat de opstelling van de vorige oefening zijn en bouw hierop verder. De **IR output** is nog steeds aangesloten op **pin 11.**

De **2 servo's** gaan we aansluiten **op pin 9 (X-as** = pan) en **pin 10 (Y-as** = tilt).




Merk op dat je de voeding van de servo's best elk met een 300mA multifuse op een externe voeding aansluit. De massa verbindt je met die van de arduino.



Let goed op de kleurencodes, dan zijn al die draadjes wel te doen en makkelijker te debuggen 😉

➔ De code voor de pan-tilt sturing met IR ontvanger

IR_ontvanger_pan_tilt

```
#include <Servo.h> //servo bib toevoegen
#include <IRremote.h> //IR ontvanger bib toevoegen

unsigned long Value1 = 0xFF10EF; //waarde van knop 4 invullen (links)
unsigned long Value2 = 0xFF5AA5; //waarde van knop 6 invullen (rechts)
unsigned long Value3 = 0xFF18E7; //waarde van knop 2 invullen (boven)
unsigned long Value4 = 0xFF4AB5; //waarde van knop 8 invullen (beneden)

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);//maak object irrecv
decode_results results;

Servo servo1; //x-as object
Servo servo2; //y-as object

void setup()
{
  Serial.begin(9600); //toon resultaat in serial monitor
  irrecv.enableIRIn(); // start de ontvanger op
  servo1.attach(10); //x-as, pan
  servo2.attach(9);// y-as, tilt
}

void loop() {
  if (irrecv.decode(&results)) //als er signaal aan de ingang is
  {
    Serial.println(results.value, HEX);//print HEX versie op serial monitor
    irrecv.resume(); // ontvang volgende waarde
  }


  if(results.value == Value1){ //knop 4 ingedrukt
     servo1.write(150);//servo ga naar 160 graden
     Serial.println("links");
  }
  else if(results.value == Value2){ //knop 6 ingedrukt
     servo1.write(60);//servo ga naar 70 graden
      Serial.println("rechts");
  }
  else if(results.value == Value3){ //knop 2 ingedrukt
     servo2.write(60);//servo ga naar 70 graden
      Serial.println("boven");
  }
  else if(results.value == Value4){ //knop 8 ingedrukt
     servo2.write(150);//servo ga naar 160 graden
      Serial.println("onder");
  }

}
```

Mogelijk moet je de waarde van de code aanpassen voor jouw afstandsbediening!

Via de serial monitor kan je zien welke knop is ingedrukt. Handig bij het debuggen en instellen van jouw pan-tilt constructie.

➔ Uitdagingen afstandsbediening:

1. Sluit de RF/IR ontvanger aan op een uno. Zorg dat de code op een **serial monitor** de waardes weergeeft wanneer je drukt op een knop.
2. Zorg ervoor dat je de **2 servo's (pan/tilt)** kan aansturen met de afstandsbediening (boven (2) , onder (8) , links (4 ), rechts (6 )). Toon tekst op de serial monitor. Denk aan het gebruik van multifuses en een externe voeding!
3. Neem een **OLED of LCD scherm** en toon hier de waardes van oefening 2 op.
4. Stuur nu de **MeArm** met afstandsbediening aan. Daag je zelf uit en probeer een blokje te verplaatsen van de ene naar een andere plek. Maak er een GSM filmpje van.
5. Meet met een **logic analyser of digitale scope** zowel de burst als een ganse periode van het verzonden protocol. Analyseer de data en probeer de inhoud uit te leggen (zie soorten protocollen).
6. Verwerk de werking en de oefeningen in jouw verslag en trek er besluiten uit.

Extra:

7. Stuur een **Arduino robot aan met een afstandsbediening.** Laat hem in alle richtingen rijden. Bescherm mogelijk de robot met een afstandssensor zodat hij niet botst tegen een rand.
8. Onderzoek het **protocol van andere** afstandsbedieningen.
9. Pas jouw code aan en maak gebruik van een **switch case** structuur: https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/

```
switch (var) {
  case 1:
    //do something when var equals 1
    break;
  case 2:
    //do something when var equals 2
    break;
  default:
    // if nothing else matches, do the default
    // default is optional
    break;
}
```

# 9. Het 595 schuifregister en 7-segmenten

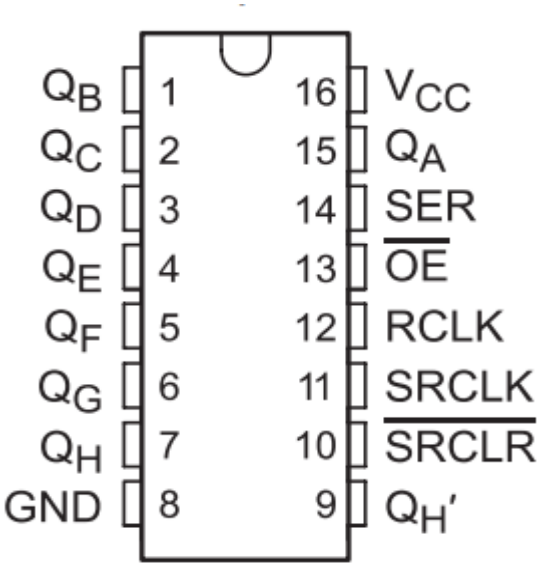
Soms is er nood aan het aansturen van meerdere uitgangen (vb LEDs, 7-segmenten) terwijl je te weinig pinnen hiervoor vrij hebt aan jouw microcontroller.

Dan is uiteraard I2C een handige oplossing (slechts 2 draden en voeding).

Stel dat de microcontroller de I2C niet ondersteunt dan kan je ook te werk gaan met een **schuifregister zoals de 74HC595.**

In dit hoofdstuk leer je omgaan met het schuifregister en gaan we een **elektronische dobbelsteen** tonen op een **7-segment display.**

➔ Wat is een schuifregister?

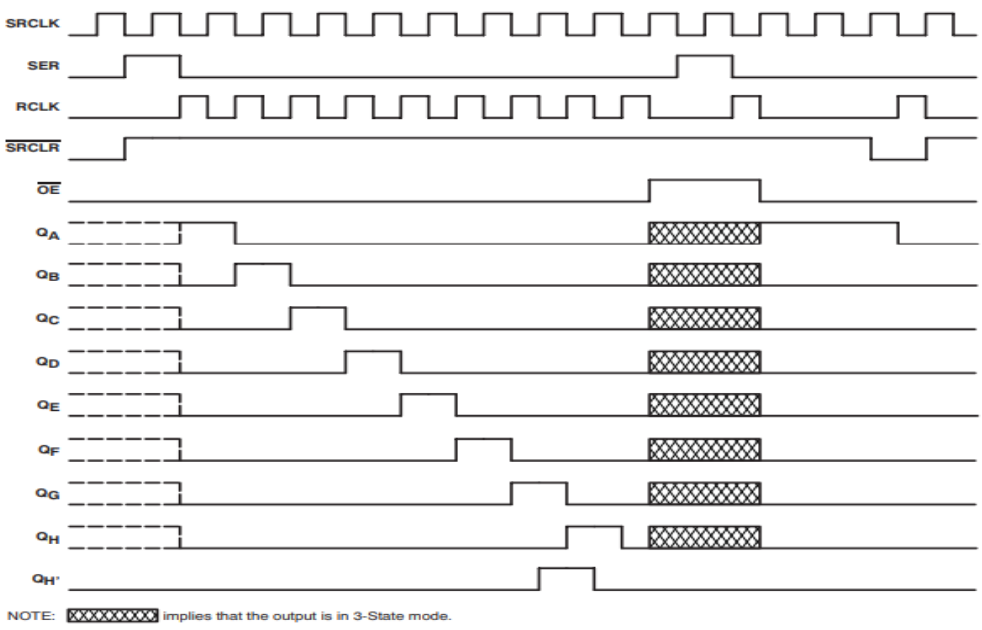



Pin 1 zit aan de linkerkant van het boogje. Deze kan je ook makkelijk terug vinden op jouw IC.

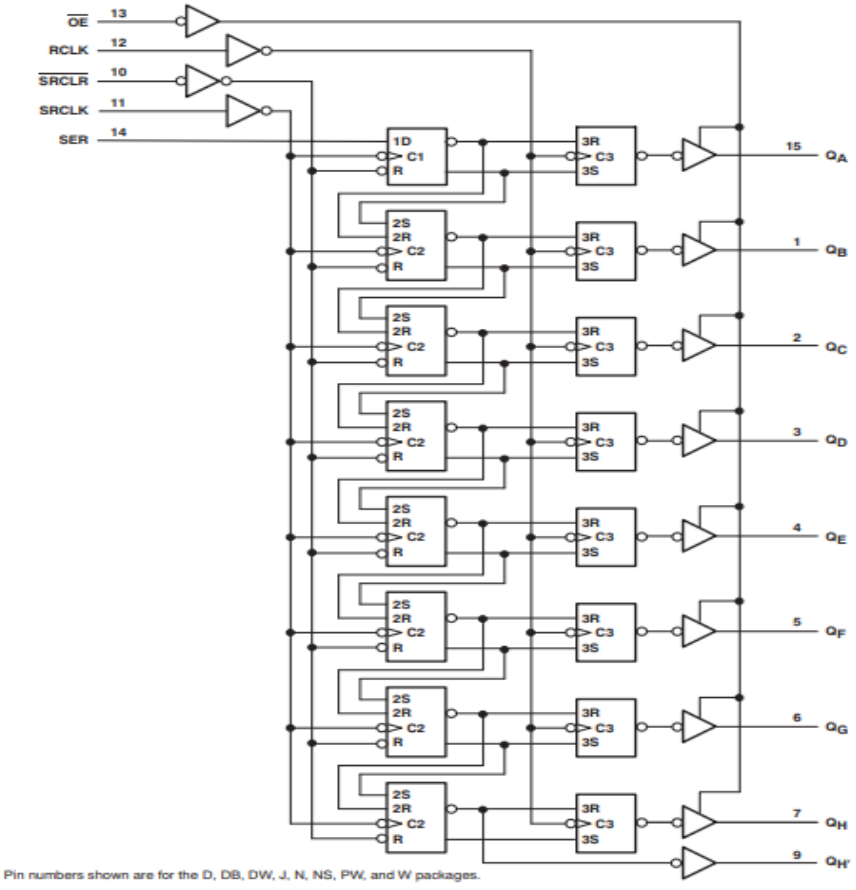
| QA – QH<br>Q0 – Q7 | Uitgangen |
|---|---|
| QH'<br>Q7S | Seriële uitgang<br>Naar volgende 74HC595 |
| $\overline{SRCLR}$ | Master clear<br>Actief laag |
| SRCLK | Schuifregister klok (CLOCK) |
| RCLK | Opslagregister klok (LATCH) |
| $\overline{OE}$ | Output Enable<br>Actief laag |
| SER | Seriële data ingang (DATA) |
| GND, VCC | Massa en +5V |

Deze chip stelt ons in staat om op een seriële manier (bit per bit) 8 bits data te versturen via de UNO. Eerst zal de data **bit per bit ingeklokt** worden, met **behulp van SRCLK, langs de SER pin** in de chip. In de tussentijd is de OUTPUT nog niet enabled (OE = hoog) en zien we nog niet op de uitgang wat er gebeurd. **Na elke ingeklokte bit** wordt een **LATCH** (RCLK) uitgevoerd om de **data te bewaren** in een volgend register. Deze actie zet dus steeds de 8 bits tegelijk in het geheugen van het schuifregister. Tenslotte tonen we het resultaat op de uitgang door **OE laag** te maken.

Je kan ook meerdere schuifregisters achter elkaar hange en zo nog meer outputs aansturen. Daarvoor dient de QH' uitgang. Die sluit je dan aan op SER van de 2$^{de}$ chip. De SRCLK, OE en LATCH hang je parallel aan beide chips en bedien je dus voor beide chips samen.



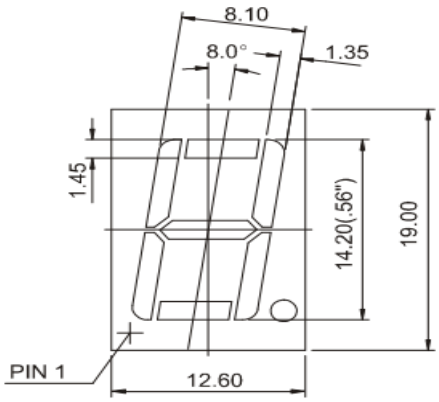

Het tijdsdiagram van de 74HC595

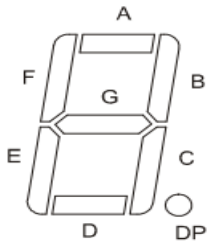



Logisch diagram van de 74HC595 (8 FlipFlops in cascade (op een rij) als schuifregister).

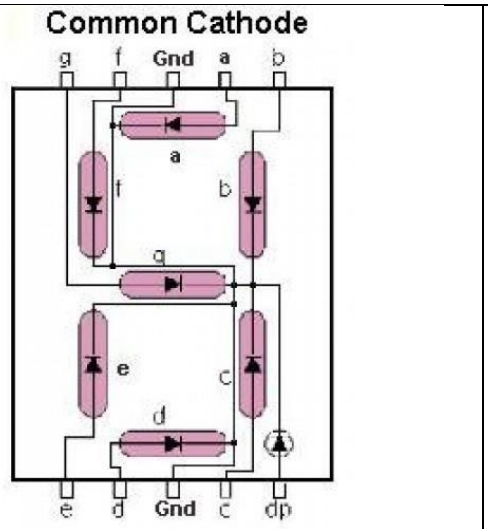
➔ Wat is een 7-segment?

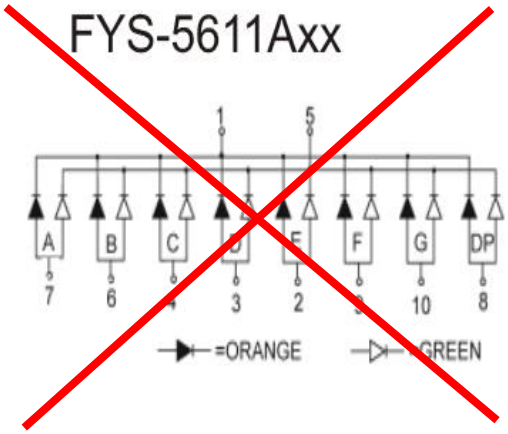
We hebben al gewerkt met gewone LEDs, RGB LEDs en lasers. Een 7-segment display is niet meer als een **verzameling van 7 LEDs** waarmee je een **cijfer of letter** kan vormen. Er is ook een **DP** (decimaal punt) voorzien zodat je komma getallen kan gaan voorstellen. We gaan werken met de **5611AH.**




Net als bij een IC ga je beginnen tellen vanaf pin 1 (zie op figuur). Dan tel je naar rechts op. Daarna spring je naar de overkant van het segment en tel je verder van rechts naar links tot je aan 10 komt.



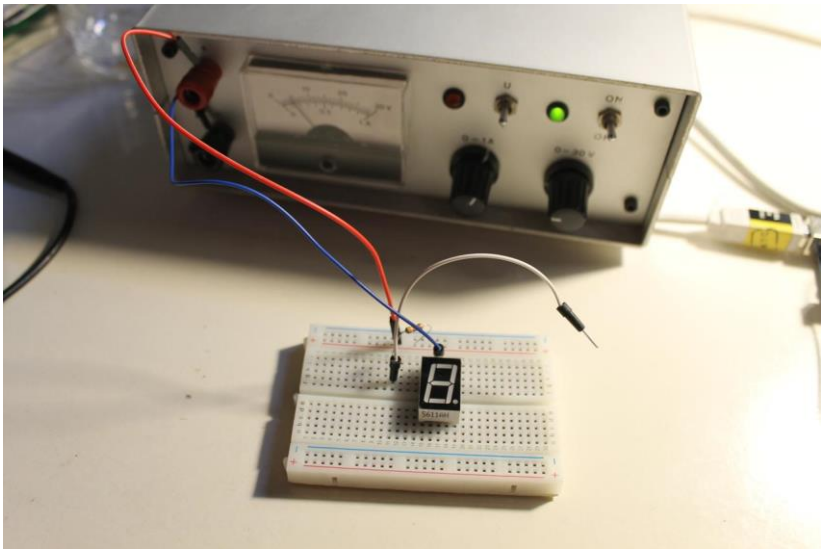

Foutieve datasheet op internet te vinden, zie jij het verschil?

Je kan aan het inwendige schema zien dat de kathodes met elkaar zijn verbonden. Het is dus een **common kathode (CK)** display. We moeten de LEDs **hoog actief** aansturen.

| | |
|---|---|
| E | 1 |
| D | 2 |
| GND (CK) | 3 |
| C | 4 |
| Decimaal punt | 5 |
| B | 6 |
| A | 7 |
| GND | 8 |
| F | 9 |
| G | 10 |

Merk op dat elke LEDs van het 7-segment een **voorschakelweerstand** nodig heeft. Vermits ze maar gemiddeld op 2V, 10mA werken moeten we een weerstand van 5 – 2 / 0.01 = 300 ohm voorzien. **220 ohm of 330 ohm** is dus prima om te gebruiken.

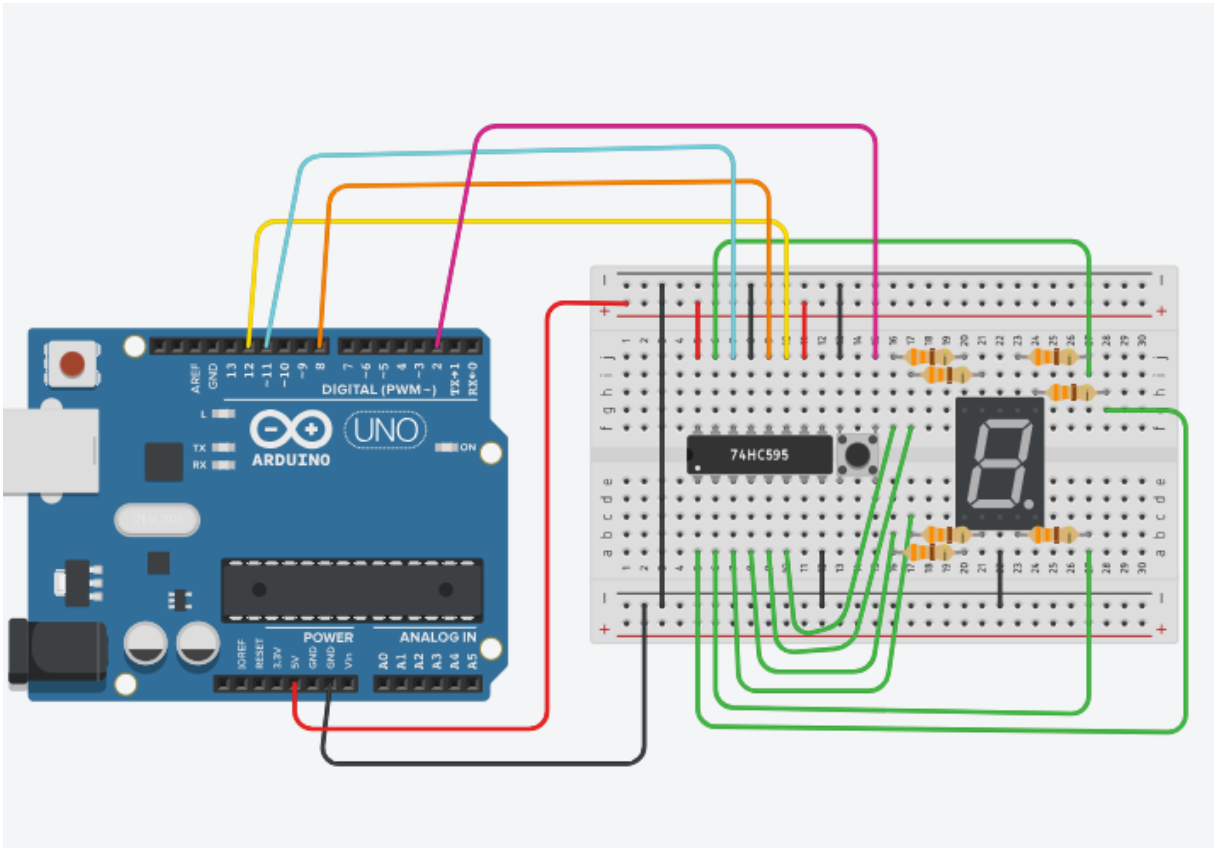
Tijdens het knutselen merkte ik dat de datasheet van het 7-segment display niet overeen kwam met het segment op mijn werktafel. Ik ben dan uiteindelijk een stap terug gaan zetten en heb **elk segment apart gaan uitmeten**. Gewoon 5V via een 220 ohm weerstand aanleggen aan het display. De **CK aan massa** leggen. Zo kan je elk segment per pin doormeten.



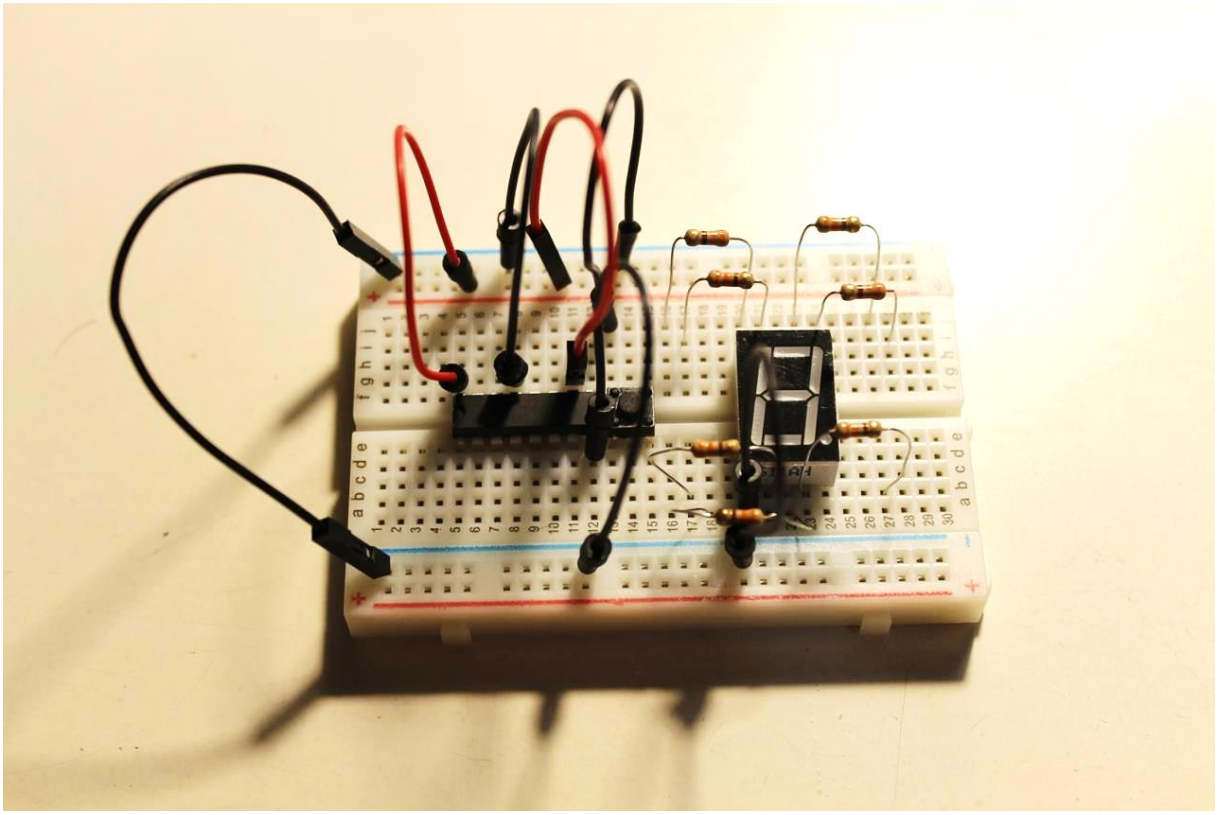
➔ Hoe pakken we dit in Arduino aan?

| **Register 74HC595** | **7-segment (via een 220/330 ohm weerstand)** |
|---|---|
| Pin 15 (QA) | Pin 7 (A) |
| Pin 1 (QB) | Pin 6 (B) |
| Pin 2 (QC) | Pin 4 ( C ) |
| Pin 3 (QD) | Pin 2 (D) |
| Pin 4 (QE) | Pin 1 ( E ) |
| Pin 5 (QF) | Pin 9 (F ) |
| Pin 6 (QG) | Pin 10 (G) |
| Pin 7 (QH) | Niet gebruikt |
| Pin 9 (QH') | Niet gebruikt |
| | Pin 3 aan GND |
| | |
| **Register 74HC595** | **UNO** |
| Pin 8 | massa |
| Pin 10 (SRCLR), reset | +5V (nooit reset) |
| Pin 11 (SRCLK), Clock | Pin 12 |
| Pin 12 (RCLK), Latch | Pin 8 |
| Pin 13 (OE) | GND (altijd uitgangen zichtbaar) |
| Pin 14 (SER) ,Data | Pin 11 |
| Pin 16 | +5V |
| | |
| **Laag actieve knop** | Pin 2 van UNO |

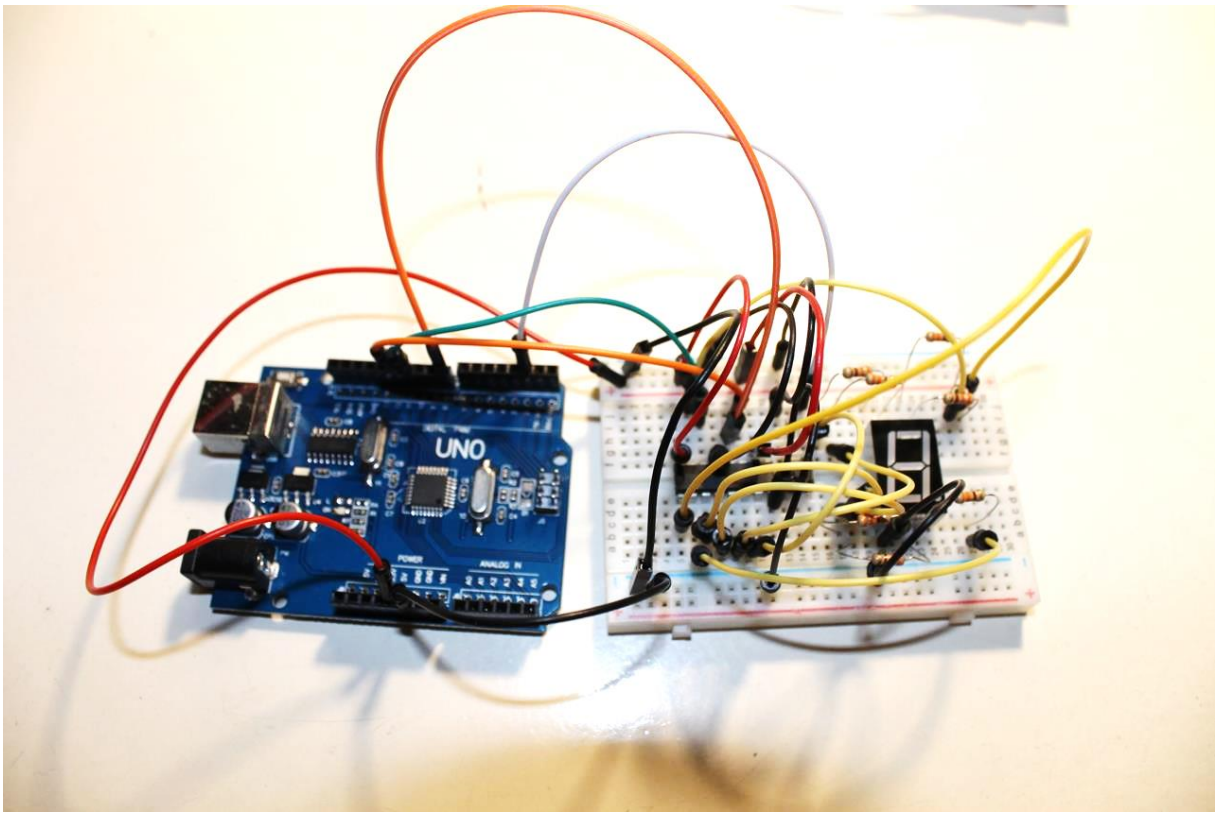
Deze tabel helpt jou de verbindingen te maken tussen de componenten en de UNO.




Bedraad alles voorzichtig. Let op de kleurencode!



Eerst de componenten plaatsen en de voedingen bedraden.




Daarna stap voor stap de tabel volgen en elke draad voorzichtig toevoegen. Wat een spaghetti.

➔ Nu we zo hard gewerkt hebben gaan we de Arduino code testen 😊

Gelukkig moeten we het warm water weer niet meer zelf uitvinden. Het serieel aansturen van de 74HC595 gaan we doen via het **shiftOut()** commando.

Eerst testen we **elk segment** na.

74HC595_led_test

```
// Arduino pins used to interface to 74HC595
const int latchPin = 8;
const int clockPin = 12;
const int dataPin = 11;
// Arduino pin interfaced to switch
const int buttonPin = 2;

// 1 to 6 and DP (decimal point) on 7-seg display, start met te tellen vanaf 0
unsigned char lookup_7seg[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40};
                              //A     B     C     D     E     F     G
                    //array POS 0     1     2     3     4     5     6

// displays the dice number on the 7-seg display
void DiceNumber(unsigned char num)
{
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, lookup_7seg[num]);
    digitalWrite(latchPin, HIGH);
}


void setup() {

  Serial.begin(9600);
  // output pins for controlling the 74HC595
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  // pin for reading button state
  pinMode(buttonPin, INPUT_PULLUP);//LAAG ACTIEF

  }


void loop() {

    for (int i=0; i < 7; i++){
      DiceNumber(i);
      Serial.print("LED");
      Serial.println(i);
      delay(1000);
    }

}
```

Ik heb de dobbelsteen vereenvoudigt tot enkel het aanzetten van de verschillende segmenten. Zo kan je snel ontdekken of de hardware goed gemaakt is. Daarna kunnen we moeilijker gaan doen.

We gebruiken weer een array om de LEDs van het 7 segment makkelijk aan te sturen.

De data, clock en latch pin, voor de aansturing van het schuifregister, zitten ook in de code verwerkt.

Altijd dezelfde volgorde aanhouden.

➔ Nu laten we het 7-segment een draaiende dobbelsteen voorstellen. Druk je op de laag actieve knop, dan telt de dobbelsteen van 1 tot 6.

74HC595_test

```
// Arduino pins used to interface to 74HC595
const int latchPin = 8;
const int clockPin = 12;
const int dataPin = 11;
// Arduino pin interfaced to switch
const int buttonPin = 2;

// 1 to 6 and DP (decimal point) on 7-seg display, start met te tellen vanaf 0
unsigned char lookup_7seg[] = {0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x80};
                              //1     2     3     4     5     6     DP
                    //array POS 0     1     2     3     4     5     6

unsigned char roll_dice[] = {0x1C, 0x58, 0x54, 0x4C}; //rollende dobbelsteen voorgesteld

// displays the dice number on the 7-seg display
void DiceNumber(unsigned char num)
{
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, lookup_7seg[num]);
    digitalWrite(latchPin, HIGH);
}

void RollDiceNumber(unsigned char num)
{
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, roll_dice[num]);
    digitalWrite(latchPin, HIGH);
}


void setup() {

  Serial.begin(9600);
  // output pins for controlling the 74HC595
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  // pin for reading button state
  pinMode(buttonPin, INPUT_PULLUP);//LAAG ACTIEF

  // display DP on seven-seg display at startup
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, lookup_7seg[6]);
  digitalWrite(latchPin, HIGH);

  }
```

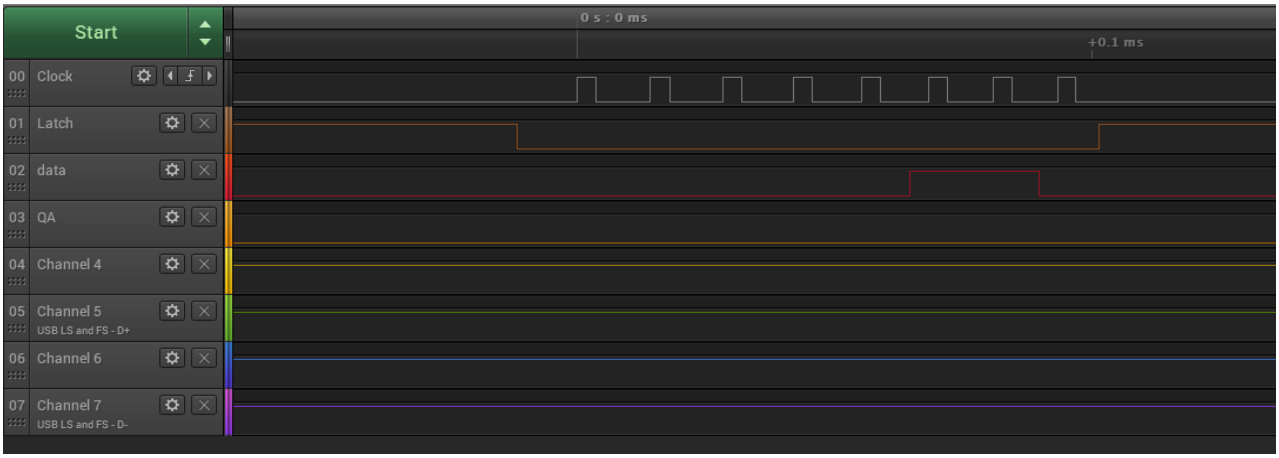
```
void loop() {
  if (digitalRead(buttonPin)== 0) { //laag actief

    for (int i=0; i < 6; i++){
      DiceNumber(i);
      Serial.print("dicenumber");
      Serial.println(i);
      delay(300);
    }

  }
  else {

      for (int i=0; i < 4; i++){
        RollDiceNumber(i);
        Serial.print("role dice number");
        Serial.println(i);
        delay(200);
      }
    }
}
```

Hoe komen ze aan 0x06 ? Wanneer je segment B en C wil aandoen om een “1” te krijgen, dan moet je QB en QC aanzetten op het schuifregister. De rest blijft “0”. Als je de ganse byte bekijkt is dit 0b0000 0110 = 0x06 . Merk op dat we van rechts naar links moeten lezen (segment A (LSB = low significant bit) ) zit aan de rechterkant. De software stuurt eerst de MSB (Most significant bit) uit. Deze moet in QG terecht komen na een 8-tal klokken.

Op de USB stick vind je uiteindelijk nog de derde versie van de ganse dobbelsteen terug.

Kijk ook deze code eens na en test uit.

➔ Uitdaging: meet met de logic analyzer op de data en kloklijnen van het schuifregister. Zo kan je dan de aansturing beter leren begrijpen.




De “1” werd naar het 7-segment gestuurd. Je kan de 0b0000 0110 terugvinden op de data lijn.

De klok toont de 8 pulsen. Op het einde wordt de Latch even actief en verschijnt de nieuwe data op de uitgang.

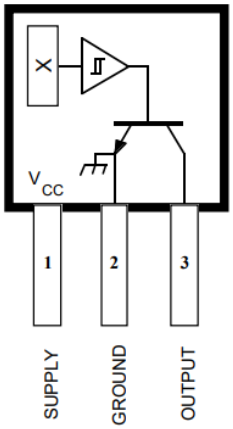
- Je kan ook 2 74HC595 chips in cascade zetten en 16 LEDs aansturen. Leuke uitdaging.

# 10. Hall sensors

Er bestaan heel wat sensoren waar langs je kan meten of iets aanwezig is of open/toe. Soms is het handig dat je **geen contact maken tussen 2 voorwerpen**, zoals een ronddraaiend fietswiel en de sensor of de ventilator in jouw desktop PC. Dan wil je toch te weten komen hoe snel deze draait, zonder dat de sensor schuurt. Dit lossen we op met een Hall sensor.

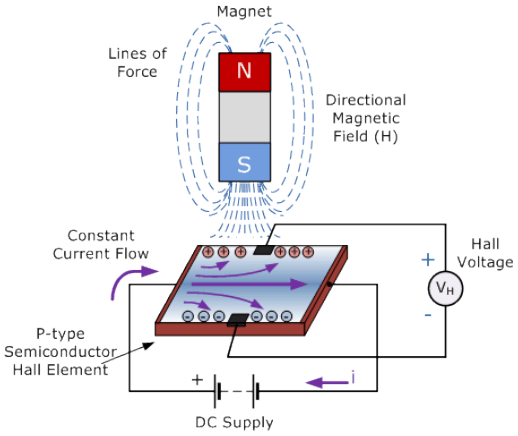
➔ Hoe werkt een Hall sensor?

We gaan gebruik maken van de **3144** Hall sensor van Allegro.

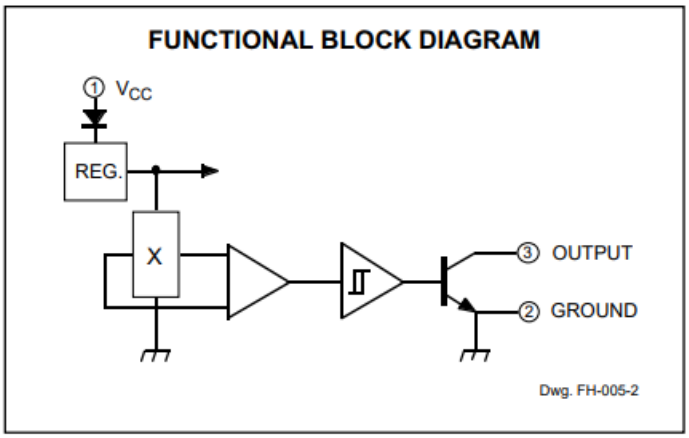




De sensor werkt heel eenvoudig. Wanneer je een **magnetisch veld laat naderen** (vb via een magneet op een stang van een fiets), dan gaat de **sensor inschakelen.** Wanneer nu de **tegengestelde pool** of **geen magnetisch veld** nadert zal de sensor **uitschakelen.**

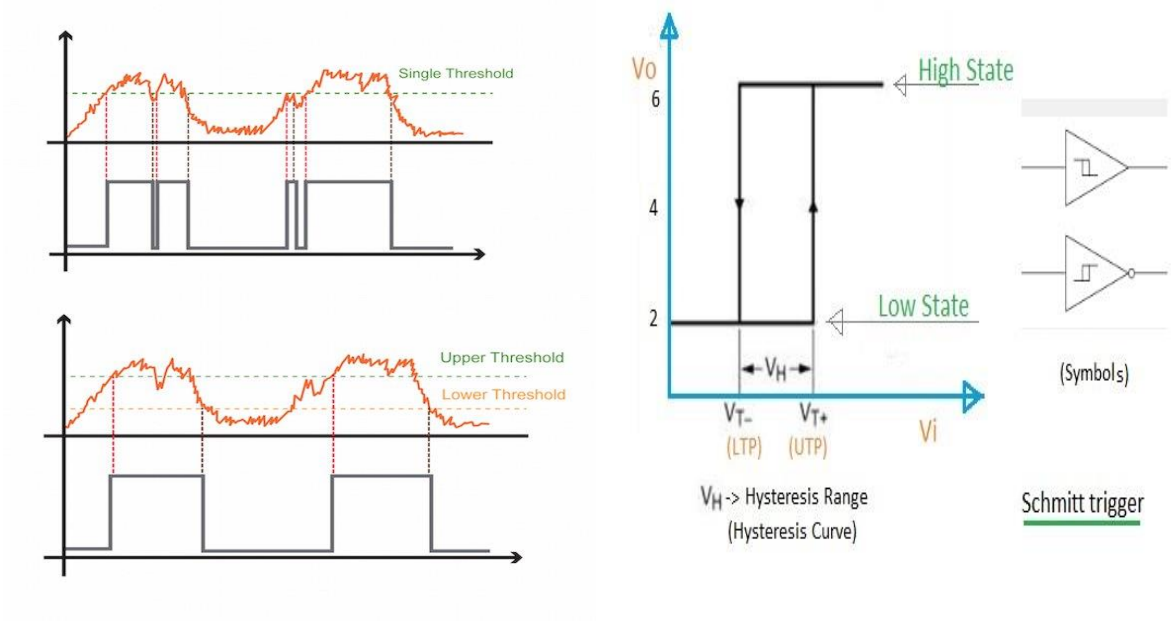



We laten door een **halfgeleider** constant, via de voeding, **stroom vloeien.** Wanneer we nu een **magnetisch veld** in de buurt van dit materiaal brengen gaat er een **spanning meetbaar** worden aan de zijkant van dit materiaal. Dit kunnen we meten of **versterken** tot het bruikbaar is voor ons. De versterker met **Schmitt-Trigger**, ingebouwd in de Hall sensor, beslist of de **open collector** uitgang wel of niet zal schakelen.

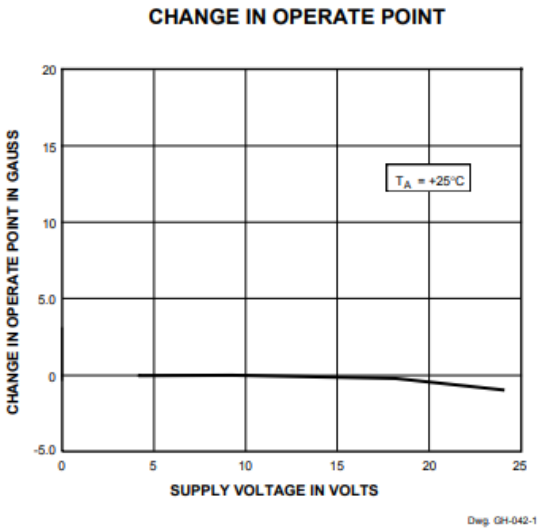



➔ Wat is een Schmitt-Trigger?

De Schmitt-Trigger is een effect waarbij de **uitgang hoog** wordt als de spanning (of hier het magnetisch veld) aan de ingang **boven een bepaalde waarde** komt. De uitgang kan pas **omlaag** als de ingang **onder een bepaalde spanning** (of magnetisme) daalt.




Je kan tot 25mA via de open collector naar massa laten vloeien. We gebruiken een 10k weerstand als pullup en komen op die manier in het ergste geval slechts aan 5 / 10k = 500uA.




## OPERATION

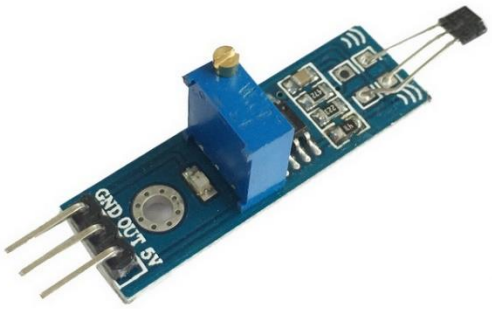
The output of these devices (pin 3) switches low when the magnetic field
the Hall element exceeds the operate point threshold ($B_{OP}$). At this point, the
ıtput voltage is $V_{OUT(SAT)}$. When the magnetic field is reduced to below the
lease point threshold ($B_{RP}$), the device output goes high. The difference in
e magnetic operate and release points is called the hysteresis ($B_{hys}$) of the
vice. This built-in hysteresis allows clean switching of the output even in
e presence of external mechanical vibration and electrical noise.

Hier kan je de datasheet nog eens nalezen van de 3144 Hall sensor.

Bij veel magnetisme (hier uitgedrukt in Gauss) schakelt de sensor in (de uitgang wordt dan laag). Is het aantal Gauss onder een bepaalde waarde wordt de uitgang hoog.

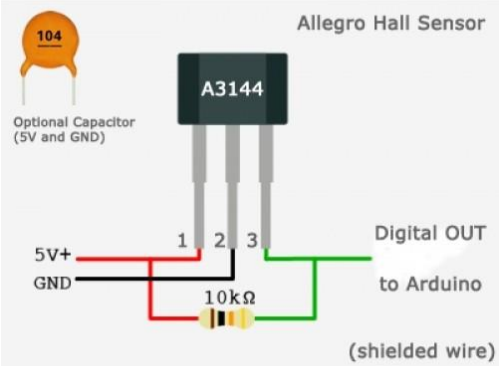
➔ Hoe de Hall sensor aansluiten in Arduino?

Hall sensor is aangekocht geweest op een breakout board, maar het **analoge signaal aan de uitgang verandert niet** als er een magnetisch signaal wordt aangeboden.



Daarom hebben we de **sensor er afgeknipt en sluiten we deze direct** aan op de UNO.

Vergeet niet de **10k pullup weerstand** te voorzien naar de +5V. Dit is omdat we anders een zwevende open collector uitgang hebben.




Er zijn 3 aansluitingen: +5V, GND, OUT (digitale uitgang voor UNO) met de 10k

We sluiten OUT aan op pin 2 van de UNO.

➔ De code om de Hall sensor te detecteren ziet er dan als volgt uit:

hallsensor_test

```
//RECHSTREEKS AANGESLOTEN op de 3144
//vergeet de externe 10K niet aan te sluiten tussen OUPUT en +5V

int latch = 2; //hall sensor digitale ingang
int led = 13;// zie led 13 op UNO

void setup() {

   pinMode(latch, INPUT);
   pinMode(led, OUTPUT);
   Serial.begin(9600);

}

void loop() {

  if (digitalRead(latch) == LOW){  //LAAG ACTIEVE UITGANG

     digitalWrite(led, HIGH);
     Serial.println("MAGNETISCH FIELD gevonden");
  }
  else {

     digitalWrite(led,LOW);
     Serial.println("OMGEKEERD MAGNETISCH FIELD gevonden");
  }

delay (500);
}
```

Als je de magneet in de ene richting houdt voor de sensor (of een magneet van bijvoorbeeld een luidspreker), dan gaat de LED op pin 13 branden. Anders gaat de LED terug uit.

# 11. Maak een weerstation met de DHT11

We gaan nu een weerstation bouwen waarbij we zowel de temperatuur als de luchtvochtigheid kunnen meten. We tonen de waardes op de serial monitor.

Om dit project te realiseren maken we gebruik van **de DHT11.** Hij bevat een capacitieve luchtvochtigheidssensor en een resistieve-type temperatuursensor om een meting van de omgeving te doen.

De metingen worden naar de Arduino gestuurd als een spanning. Het beste resultaat krijg je als je de sensor plaatst op een buitenmuur en er heel wat ruimte rond open laat.

Het LCD scherm kan je bijvoorbeeld in huis monteren.

➔ Hoe werkt de DHT11?

De sensor heeft 3 draden. 1 Pin wordt niet gebruikt.

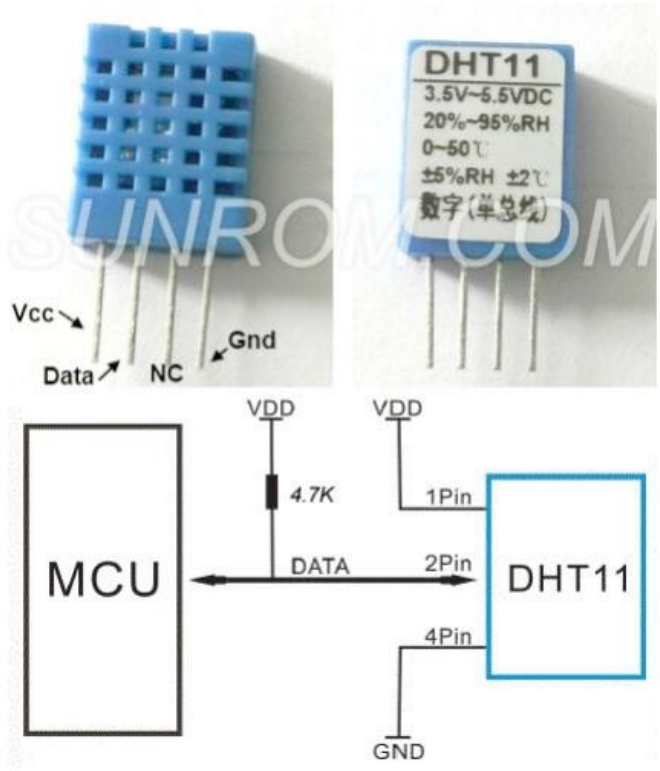
Buiten de +5V en de GND is er nog 1 data draad. Hier langs kan er in beide richtingen gecommuniceerd worden volgens het **1-wire** principe.

We moeten een **pullup weerstand** voorzien op de signaal uitgang naar +5V van **4K7** tot 10k. Dit is omdat we weer werken met een **open collector** uitgang. Wanneer er niets gestuurd wordt is de bus normaal gezien hoog. Als de zender of ontvanger wil praten trekt hij de lijn omlaag (lijkt op I2C, maar in dit geval is er geen aparte klok voorzien).

Men gaat altijd in **3 stappen** te werk om data te versturen (dit is gelukkig al door een Arduino library voorzien): request, response, data reading.

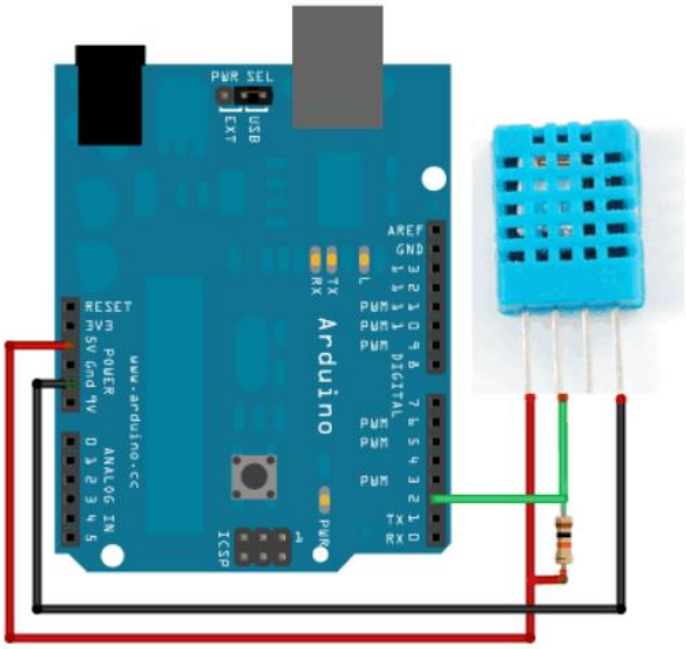

Je kan hier een logic analyzer meting vinden van de databus. In stapjes worden alle bits over de draad gestuurd.

Meer info kan je vinden in de datasheet op de USB stick.




Pin 1 is +5V, pin 2 is de data lijn, pin 3 is niet gebruikt, pin 4 is de massa.

➔ Hoe aansluiten in Arduino?



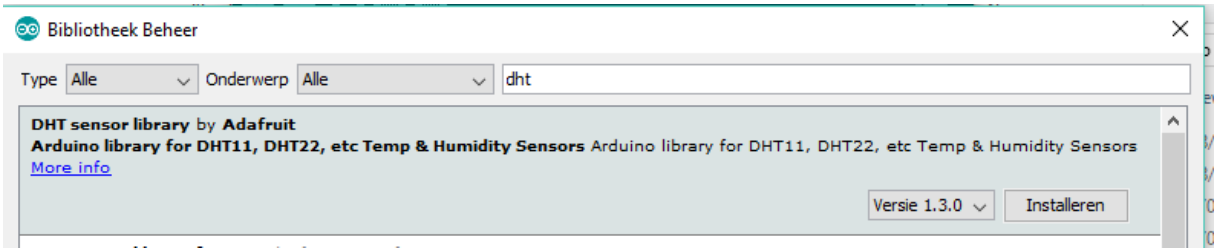

Let op de pullup weerstand van 10k

➔ Code om de uitlezing naar de serial monitor te sturen.

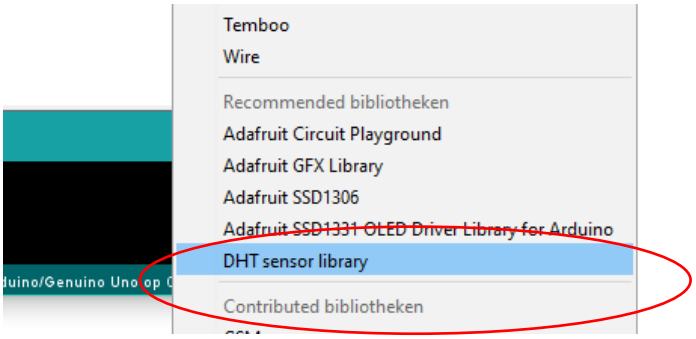
We moeten er voor zorgen dat we eerst de library weer gaan toevoegen aan Arduino. In dit geval moet je kijken of de DHT library is toegevoegd.

Ga naar **schets -> bibliotheek gebruiken -> bibliotheek beheren**

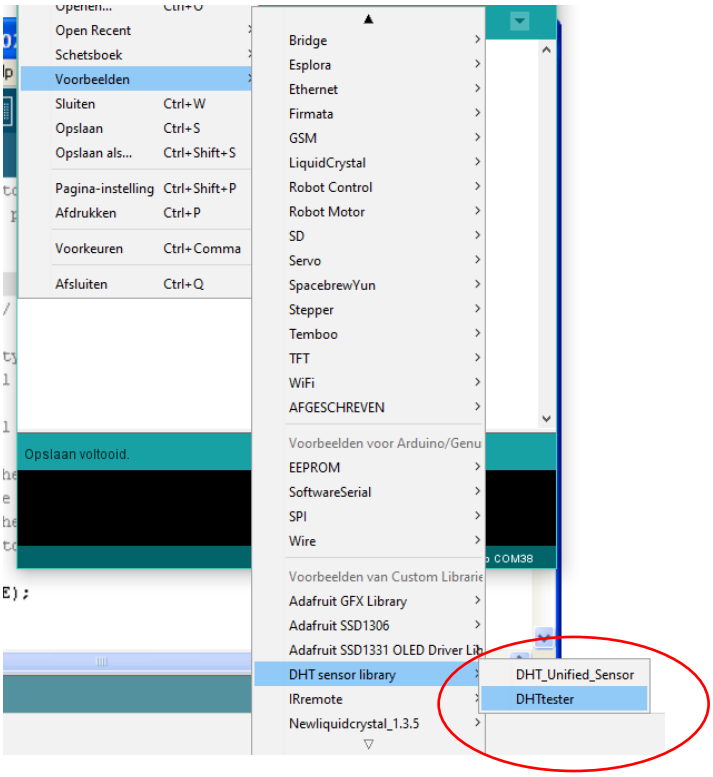
Zoek op **DHT11** en installeer de **Adafruit** versie.




Nu moet de library in het lijstje staan van jouw Arduino:



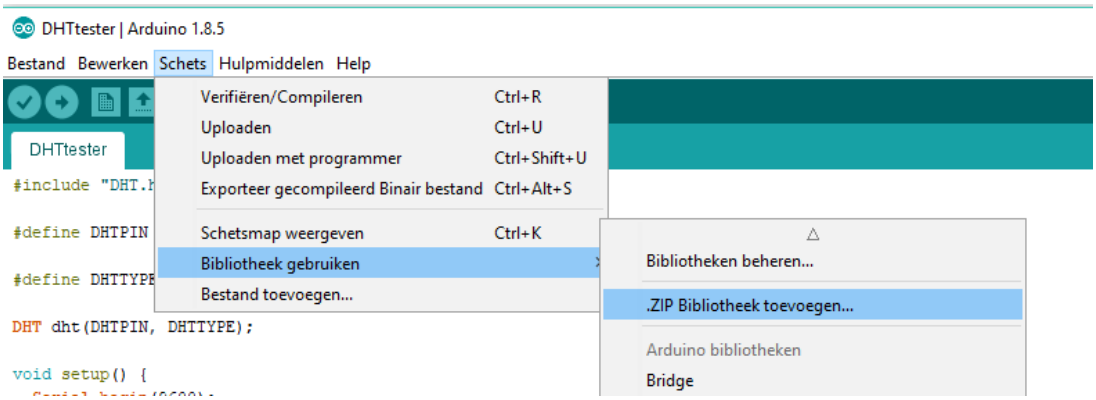

Ga nu naar de voorbeelden in Arduino en kies de **DHTtester.**



Nu moet je ook nog de **Adafruit_Sensor-master** zip file toevoegen. Deze vind je op jouw USB stick.

Ga naar **Schets -> bibliotheek gebruiken -> .ZIP bibliotheek toevoegen**

Selecteer nu de ZIP file Adafruit_Sensor-master op jouw USB stick en laat Arduino deze toevoegen.




Nu ben je klaar om de code te uploaden en testen:

DHTtester

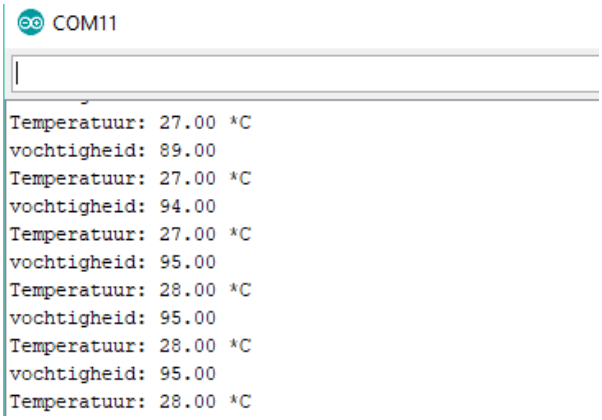
```
#include "DHT.h"

#define DHTPIN 2     // what digital pin we're connected to

#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHT11 test!");

  dht.begin();//init DHT11
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();//lees vochtigheid uit
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();//lees temperatuur uit

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) { //controleer of de DHT11 goed heeft gewerkt
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
```

```
  else {

      Serial.print("vochtigheid: ");//vochtigheid
      Serial.println(h);
      Serial.print("Temperatuur: ");
      Serial.print(t);
      Serial.println(" *C ");
  }


}
```

Via de DHT bibliotheek kunnen we heel makkelijk de sensor aansturen.

Er zijn voorgemaakte commando's waardoor we de vochtigheid en temperatuur kunnen uitlezen.

Deze code stuurt de waardes naar de serial monitor.




➔ Uitdagingen:

- Stuur de waardes naar een LCD scherm zonder / met I2C of een OLED scherm.
- Zorg dat er een alarm af gaat en een rode LED brand van de RGB LED als je boven de 30 graden komt. Onder de 30 graden is de RGB LED groen en is er geen geluid.
- Laat ook boven de 30 graden een DC motor draaien als ventilator.
- Deze ventilator staat op het pan-tilt mechanisme en wordt netjes omhoog en omlaag, links en rechts bewogen
- Met een afstandsbediening kan je de snelheid van de motor aanpassen of het systeem uitschakelen
- ….

Nog niet genoeg van Arduino?

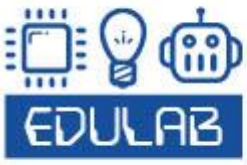
Schrijf je dan volgende zomer zeker in voor de workshop “**Arduino voor gevorderden deel 2**”.

Wat gaan we dan doen?

Hier is een greep uit de komende workshop.

1. SD card leren uitlezen en muziek data sturen naar versterker
2. WIFI shield leren gebruiken (netwerken)
3. Met een RF module, via een joystick, servo’s leren aansturen
4. Multitasking met Arduino? Hoe werkt dat?
5. Timers
6. USB device : een game controller leren maken
7. RS232
8. Een Arduino zelf leren bouwen op een breadboard

Zorg dat je zeker de **nieuwsbrief** krijgt van EDULAB. Je kan deze altijd verkrijgen door in te schrijven via de nieuwsbriefknop op [www.edulab.be](http://www.edulab.be)




FRANK MARCHAL
HOMMELHEIDE 45
B-3500 HASSELT
+32 (0) 498 82 15 90
INFO@EDULAB.BE
WWW.EDULAB.BE