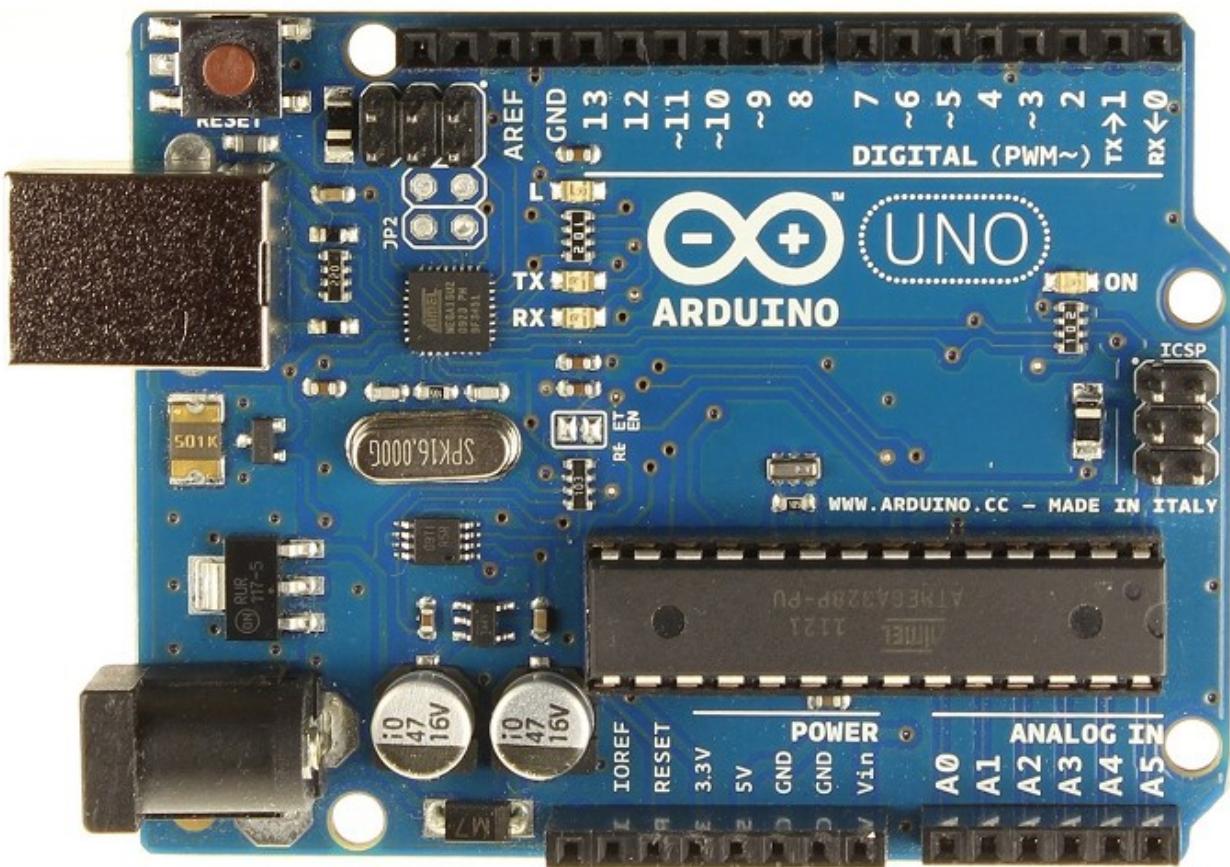


Arduino UNO projecten



Inhoudstafel

Arduino UNO projecten.....	1
Inhoudstafel.....	2
Arduino UNO R3.....	10
Arduino.....	10
Hardware.....	10
Software.....	11
Sketch.....	12
Delen van een sketch.....	12
Functies.....	13
Variabelen.....	13
Data types.....	13
Syntax regels.....	13
Instructies.....	14
Fritzing.....	16
Project 1 – Knipperende interne LED.....	17
Opgave.....	17
Benodigdheden.....	17
Sketch.....	18
Project 2 – Knipperende externe LED.....	19
Opgave.....	19
Benodigdheden.....	19
Breadboard.....	19
LED.....	19
Weerstand.....	20
Schakeling.....	21
Breadboard schakeling.....	22
Sketch.....	22
Project 3 – Ping pong LEDs.....	23
Opgave.....	23
Benodigdheden.....	23
Schakeling.....	23
Breadboard schakeling.....	24
Sketch.....	24
Project 4 – Verkeerslichten.....	26
Opgave.....	26
Benodigdheden.....	26
Schakeling.....	26
Breadboard schakeling.....	27
Sketch.....	28
Project 5 : LEDs met drukknop.....	30
Opgave.....	30
Benodigdheden.....	30
Schakelaars.....	30
Schakeling.....	31
Breadboard schakeling.....	32
Sketch.....	33

Project 6 : Helderheid LED aanpassen.....	35
Opgave.....	35
Pulse Width Modulation (PWM).....	35
Benodigdheden.....	36
Schakeling.....	36
Breadboard schakeling.....	37
Sketch.....	38
Project 7 : Driekleuren LED.....	40
Opgave.....	40
Benodigdheden.....	40
RGB LED.....	40
Schakeling.....	41
Breadboard schakeling.....	42
Sketch.....	43
Project 8 : Spanningsdeler.....	44
Opgave.....	44
Benodigdheden.....	44
Spanningsdeler.....	44
Schakeling.....	45
Breadboard schakeling.....	46
Sketch.....	47
Project 9 : Zoemer.....	49
Opgave.....	49
Benodigdheden.....	49
Piezo zoemer.....	49
Schakeling.....	50
Breadboard schakeling.....	51
Sketch.....	51
Project 10 : Eén cijfer 7 segment LED display.....	53
Opgave.....	53
Benodigdheden.....	53
Eén cijfer 7 segment LED display.....	53
Schakeling.....	54
Breadboard schakeling.....	54
Sketch.....	56
Project 11 : Elektronische thermometer.....	60
Opgave.....	60
Benodigdheden.....	60
Temperatuur sensor.....	60
Vier cijfer 7 segment LED display.....	61
Breadboard schakeling.....	62
Sketch.....	64
Project 12 : Afstandsdetectie.....	70
Opgave.....	70
Benodigdheden.....	70
HC-SR04 afstandsdetectie sensor.....	70
HC-SR04 aansluitingen op de UNO.....	70
Breadboard schakeling.....	71
Sketch.....	71

NewPing.....	73
Installatie.....	73
Declaratie.....	73
Methodes.....	73
Sketch met NewPing.....	74
Project 13 : 8x8 Dot LED Matrix.....	75
Opgave.....	75
Benodigdheden.....	75
8x8 Dot LED Matrix 1588BS.....	75
Schakeling.....	76
Breadboard schakeling.....	77
Sketch 1.....	78
Sketch 2.....	80
Project 14 : Serieel naar parallel conversie.....	84
Opgave.....	84
Benodigdheden.....	84
Serieel naar parallel convertor 74HC595.....	84
Werking.....	84
Aansluitingen.....	85
Breadboard schakeling.....	86
Sketch.....	87
Project 15 : Infrarode afstandsbediening.....	89
Opgave.....	89
Benodigdheden.....	89
Infrarode sensor VS 1838B.....	89
Nec protocol kenmerken.....	90
Nec protocol voorbeeld.....	90
Bibliotheek IRremote.....	90
Breadboard schakeling.....	91
Sketch.....	92
Project 16 : Lichtsensor GL55xx.....	94
Opgave.....	94
Benodigdheden.....	94
Lichtsensor GL55xx.....	94
Schakeling.....	95
Breadboard schakeling.....	96
Sketch.....	96
Project 17 : Sturing van DC motoren.....	99
Opgave.....	99
Benodigdheden.....	99
L298N motorsturing module.....	99
Aansluitingen.....	100
Breadboard schakeling.....	100
Sketch.....	101
Project 18 : Licht en donker detectie.....	105
Opgave.....	105
Benodigdheden.....	105
Optische sensormodule TCRT5000.....	105
Specificaties.....	105

Aansluitingen.....	105
Schakeling.....	106
Sketch.....	106
Project 19 : ST7735 TFT kleurenscherm.....	108
Opgave.....	108
Benodigdheden.....	108
ST7735.....	108
Specificaties.....	108
Aansluitingen.....	108
Bibliotheken.....	109
Breadboard schakeling.....	110
Sketch 1.....	111
Sketch 2.....	118
Project 20 : Servomotor.....	121
Opgave.....	121
Benodigdheden.....	121
Servomotor SM-S2309S.....	121
Aansluitingen.....	121
Piezo zoemer.....	122
Breadboard schakeling.....	122
Sketch.....	123
Project 21 : Sainsmart 37 sensor kit.....	127
Sensor kit.....	127
Overzicht sensors.....	127
KY001 – Digitale temperatuur sensor.....	129
Opgave.....	129
Benodigdheden.....	129
Sensor.....	129
Sketch.....	129
KY002 – Trilling sensor.....	132
Opgave.....	132
Benodigdheden.....	132
Sensor.....	132
Sketch.....	132
KY003 – Hall magnetisch veld sensor.....	134
Opgave.....	134
Benodigdheden.....	134
Sensor.....	134
Sketch.....	134
KY004 – Drukknop.....	136
Opgave.....	136
Benodigdheden.....	136
Sensor.....	136
Sketch.....	136
KY005 – Infrarode zender.....	138
Opgave.....	138
Benodigdheden.....	138
Sensor.....	138
Sketch.....	138

KY006 – Passieve zoemer.....	139
Opgave.....	139
Benodigdheden.....	139
Sensor.....	139
Sketch.....	139
KY008 – Laser diode.....	141
Opgave.....	141
Benodigdheden.....	141
Sensor.....	141
Sketch.....	141
KY009 – Drie kleuren SMD led.....	143
Opgave.....	143
Benodigdheden.....	143
Sensor.....	143
Sketch.....	143
KY010 – Licht onderbreking sensor.....	145
Opgave.....	145
Benodigdheden.....	145
Sensor.....	145
Sketch.....	145
KY011 – Grote twee kleuren led.....	147
Opgave.....	147
Benodigdheden.....	147
Sensor.....	147
Sketch.....	147
KY012 – Actieve zoemer.....	149
Opgave.....	149
Benodigdheden.....	149
Sensor.....	149
Sketch.....	149
KY013 – Analoge temperatuur sensor.....	150
Opgave.....	150
Benodigdheden.....	150
Sensor.....	150
Sketch.....	150
KY015 – Temperatuur en vochtigheid sensor.....	153
Opgave.....	153
Benodigdheden.....	153
Sensor.....	153
Sketch.....	153
KY016 – Drie kleuren led.....	156
Opgave.....	156
Benodigdheden.....	156
Sensor.....	156
Sketch.....	156
KY017 – Kwik schakelaar.....	158
Opgave.....	158
Benodigdheden.....	158
Sensor.....	158

Sketch.....	158
KY018 – LDR sensor.....	160
Opgave.....	160
Benodigdheden.....	160
Sensor.....	160
Sketch.....	160
KY019 – Relais.....	162
Opgave.....	162
Benodigdheden.....	162
Sensor.....	162
Sketch.....	162
KY020 – Tilt schakelaar.....	164
Opgave.....	164
Benodigdheden.....	164
Sensor.....	164
Sketch.....	164
KY021 – Kleine Reed schakelaar.....	166
Opgave.....	166
Benodigdheden.....	166
Sensor.....	166
Sketch.....	166
KY022 – Infrarode ontvanger.....	168
Opgave.....	168
Benodigdheden.....	168
Sensor.....	168
Sketch.....	168
KY023 – Mini X/Y joystick.....	170
Opgave.....	170
Benodigdheden.....	170
Sensor.....	170
Sketch.....	170
KY024 – Lineaire Hall sensor.....	173
Opgave.....	173
Benodigdheden.....	173
Sensor.....	173
Sketch.....	173
KY025 – Grote Reed schakelaar.....	175
Opgave.....	175
Benodigdheden.....	175
Sensor.....	175
Sketch.....	175
KY026 – Vlammen sensor.....	177
Opgave.....	177
Benodigdheden.....	177
Sensor.....	177
Sketch.....	177
KY027 – Magische led cup module.....	179
Opgave.....	179
Benodigdheden.....	179

Sensor.....	179
Sketch.....	179
KY028 – Digitale temperatuur sensor.....	181
Opgave.....	181
Benodigdheden.....	181
Sensor.....	181
Sketch.....	181
KY029 – Kleine twee kleuren led.....	184
Opgave.....	184
Benodigdheden.....	184
Sensor.....	184
Sketch.....	184
KY031 – Schok sensor.....	186
Opgave.....	186
Benodigdheden.....	186
Sensor.....	186
Sketch.....	186
KY032 – Obstakel detectie sensor.....	188
Opgave.....	188
Benodigdheden.....	188
Sensor.....	188
Sketch.....	188
KY033 – Lijn volgende sensor.....	190
Opgave.....	190
Benodigdheden.....	190
Sensor.....	190
Sketch.....	190
KY034 – Zeven kleuren led.....	192
Opgave.....	192
Benodigdheden.....	192
Sensor.....	192
KY035 – Analoge Hall sensor.....	193
Opgave.....	193
Benodigdheden.....	193
Sensor.....	193
Sketch.....	193
KY036 – Aanraking sensor.....	195
Opgave.....	195
Benodigdheden.....	195
Sensor.....	195
Sketch.....	195
KY037 – Grote microfoon.....	197
Opgave.....	197
Benodigdheden.....	197
Sensor.....	197
Sketch.....	197
KY038 – Kleine microfoon.....	199
Opgave.....	199
Benodigdheden.....	199

Sensor.....	199
Sketch.....	199
KY039 – Hartslag sensor.....	201
Opgave.....	201
Benodigdheden.....	201
Sensor.....	201
Sketch.....	201
KY040 – Rotary Encoder.....	203
Opgave.....	203
Benodigdheden.....	203
Sensor.....	203
Sketch.....	204

Arduino UNO R3

Arduino

Arduino (uitgesproken: ar-doe-wie-no) is de merknaam van een populaire reeks microcontrollerbordjes. Deze bordjes bestaan uit een Atmel ATmega microcontroller, wat ondersteunende componenten en een aantal aansluitingen. Samen vormen ze een soort kleine ‘computer’ waarmee je verschillende projecten zelf kunt aansturen.

Een Arduino bordje is gemaakt om op een voordelige manier je elektronica-projecten aan te sturen. De microcontroller op het bordje is hier dan ook op geselecteerd. Een volwaardige computer met veel rekenkracht zou namelijk overbodig zijn voor dit doel en alleen tot hoge kosten leiden. De Arduino Uno heeft bijvoorbeeld een geheugen van slechts 32kB. Veel minder dan bijvoorbeeld een laptop, maar ruim voldoende om je elektronica-projecten aan te sturen.

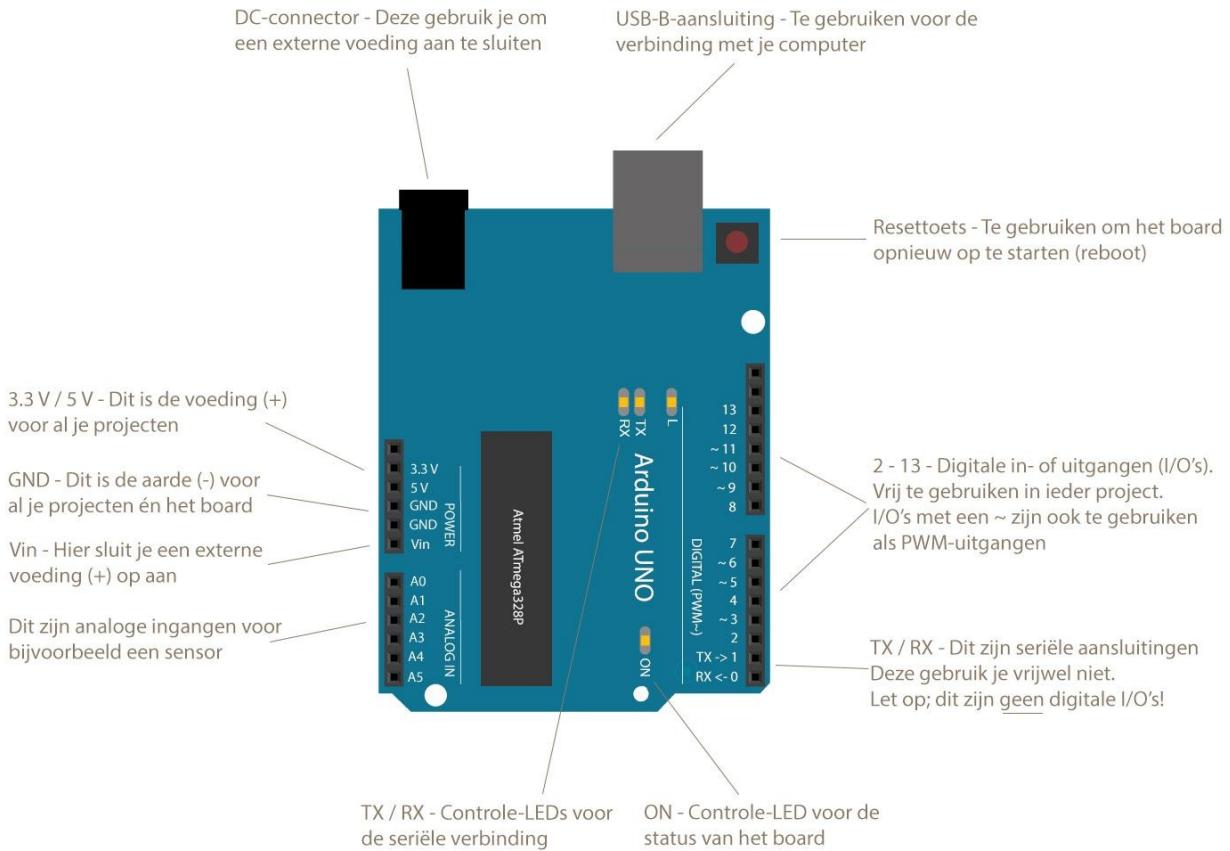
Arduino is een open-source systeem. Dit betekent dat alle ontwerpen van de diverse bordjes voor iedereen beschikbaar zijn. Wanneer je zelf je eigen Arduino bordje wilt maken, dan mag dat ook! De makers van Arduino hebben daarbij wel een voorwaarde gesteld; je mag het bordje zelf geen Arduino noemen. Het grote voordeel van open-source initiatieven als deze is dat veel gebruikers hun kennis en creativiteit in kunnen brengen. Er ontstaat zo al snel een grote groep mensen (community) die samen het originele idee kunnen verbeteren.

Hardware

Arduino bordjes zijn er in diverse uitvoeringen. De verschillende bordjes hebben elk hun eigen voordelen, maar kennen ook veel overeenkomsten. Elke Arduino bestaat namelijk uit een microcontroller met daaromheen een aantal ingangen en uitgangen, ook wel I/O's (Input/Output) genoemd. Op de input sluit je een sensor aan, op de output een actor (40 mA maximum verbruik per aansluiting). De sensor geeft de Arduino een reden om iets te doen. De actor voert vervolgens de daadwerkelijke actie uit. De software bepaalt tussen deze beide stappen in wat er moet gebeuren.

De meest eenvoudige en tegelijk ook de populairste is de Arduino Uno R3. De meest relevante onderdelen van dit bordje zijn :

- Atmel ATmega328 microcontroller
- 14 digitale input/output poorten 0-13 (6 ervan kunnen als PWM output gebruikt worden)
- 6 analoge input poorten A0-A5
- 16 MHz klok
- USB aansluiting voor voeding (5V) en communicatie met de pc
- power aansluiting voor batterij
- ICSP header
- reset knop
- power LED is aan als er spanning op het bordje staat
- pin 13 LED knippert normaal 1 maal per seconde
- Zend- en ontvangst LED (TX/RX) knipperen bij het doorsturen van het programma van de pc naar het bordje



De specificaties van de Arduino hardware zijn open-source. Dit betekent dat iedereen er gebruik van mag maken om zijn eigen versie te ontwikkelen. Dit gebeurt dan ook in ruime mate. [Sainsmart](#) is een Chinees producent van kwalitatieve Arduino compatibele bordjes en toebehoren. Die zijn bovendien een stuk goedkoper dan de originele Italiaanse bordjes.

Software

Het programmeren van een Arduino bordje doe je via de Arduino IDE (Integrated Development Environment). Deze IDE is gratis te downloaden van de [officiële Arduino-website](#) en biedt je een volledige programmeeromgeving met alle noodzakelijke elementen. De Arduino programmeertaal is gebaseerd op C/C++. De Arduino IDE is ook weer open-source. Dit betekent dat ook de Arduino IDE door iedereen vrij te gebruiken en bewerken is. Een Arduino programma noemt men een **sketch** (Nederlandse vertaling: **schets**)

De Arduino IDE omgeving bevat :

- editor met syntax controle tijdens het typen
- compileer knop om de volledige sketch te controleren
- upload knop om de sketch naar de Arduino te sturen via de USB kabel
- knoppen voor nieuwe sketch, bewaren sketch en opladen sketch
- knop voor starten seriële monitor om output van sketch te kunnen volgen
- tekst console onderaan waar status- en foutberichten worden getoond

Sketch

Een programma waarmee je het Arduino bordje vertelt wat deze moet doen heet een ‘sketch’. Een sketch bevat alle noodzakelijke elementen om jouw project goed te laten functioneren. Deze elementen worden via de IDE omgezet in concrete taken voor de hardware. Je kunt een sketch zelf schrijven, maar via de grote [Arduino-community](#) zijn ook al heel wat kant-en- klare sketches te vinden voor tal van inspirerende projecten. Je kunt deze sketches compleet overnemen of er juist delen uitpakken die voor jouw project interessant zijn.

Delen van een sketch

- *Hoofding* : dit kan commentaar bevatten. Als er meerdere lijnen zijn zet je die tussen /* en */. Voor enkele lijnen kan je ook // gebruiken. Commentaar is bedoeld voor degene die het programma gemaakt heeft maar wordt door de Arduino genegeerd. Het is wel heel nuttig om zaken te verduidelijken zodat je later nog weet wat alles doet.

```
/*
  Arduino schakelaar

Benodigde onderdelen:
  1 Rode LED
  1 Weerstand 150 Ohm
  1 Druktoets
  1 Weerstand 10 kOhm
  5 Draadbruggen
  1 Breadboard
*/
```

- *Declaraties* : hier kan je constanten en variabelen declareren die je verder in de sketch zal gebruiken. Je kan hier bvb de poorten die je zal gebruiken een logische naam geven; zo wordt het programma meer leesbaar dan als je de poortnummers gebruikt.

```
// Benoem de noodzakelijke elementen van deze sketch
const int led = 3;           // De LED is aangesloten op pin 3
const int toets = 2;          // De sensor (LDR) is aangesloten op pin 2
```

- *Setup functie* : hier plaats je code die slechts 1 maal moet uitgevoerd worden. Dit wordt typisch gebruikt om de poorten in te stellen als input of output, of om de poorten te initialiseren, ... Als je op de **reset** knop drukt, dan wordt het programma terug opgestart en wordt ook de setup weer uitgevoerd.

```
// Hier geven we eenmalig aan wat de instellingen voor deze sketch moeten zijn
void setup() {
  pinMode(led, OUTPUT);
  pinMode(toets, INPUT);
}
```

- *Loop functie* : hier plaats je de code van je sketch die steeds opnieuw moet herhaald worden. De instructies worden één voor één uitgevoerd. Na de laatste instructie wordt automatisch teruggekeerd naar de eerste instructie in de loop functie.
De code wordt uitgevoerd zolang het Arduino bord voeding heeft.

```
// Hier geven we aan welke stappen de Arduino moet afwerken in de sketch
void loop() {
  int toetsWaarde = digitalRead(toets);
```

Functies

In een Arduino sketch ga je een aantal vooraf gedefinieerde functies gebruiken. Een functie geeft een instructie aan het bordje. Met de *pinMode* functie geef je bijvoorbeeld aan of een I/O zich moet gedragen als input of output. De functie *analogRead* geeft aan dat er een analoge waarde uitgelezen moet worden. Met *digitalWrite* geef je het board de instructie om een output aan (HIGH) of uit (LOW) te zetten.

Variabelen

Een variabele bestaat uit een naam, een waarde en een type. Met een variabele sla je als het ware een waarde op zodat je deze later in de sketch weer kunt gebruiken. Er zijn verschillende variabelen, sommige maak je zelf, andere zijn al vooraf gedefinieerd.

Met de variabele ‘int ledPin = 13;’ geef je bijvoorbeeld aan dat je de aansluiting met waarde ‘13’ in de sketch gaat aanspreken met de naam ‘ledPin’. Met het type ‘int’ geef je aan dat er een geheel getal wordt opgeslagen, een integer.

Data types

Dit zijn de mogelijke types van data welke je gebruikt voor de declaratie van variabelen.

Type	Bytes	Waarden	Opmerking
<i>boolean</i>	1	True of False	Logisch data type kent slechts 2 waarden 0 of 1
<i>char</i>	1	-128 tot 127	ASCII karakter code. Letter A heeft als code 65. De negatieve waarden worden niet gebruikt.
<i>byte</i>	1	0 tot 255	Meestal gebruikt voor seriële data
<i>int</i>	2	-32768 tot 32767	Gehele getallen
<i>unsigned int</i>	2	0 tot 65535	Gehele getallen zonder negatieve waarden
<i>long</i>	4	-2147483648 tot 2147483647	Lange gehele getallen voor grote waarden
<i>unsigned long</i>	4	0 tot 4294967295	Lange gehele getallen zonder negatieve waarden
<i>float</i>	4	-3,4028235 E+38 tot 3,4028235E+38	Decimale getallen
<i>double</i>	4	-3,4028235E+38 tot 3,4028235E+38	Bij de Arduino is dit hetzelfde als type float. Normaal zijn dit decimale getallen van 8 bytes met een grotere precisie als type float.

Constanten zijn eigenlijk variabelen die een waarde krijgen die daarna niet meer kan veranderen. Bij de declaratie zet je *const* voor het datatype.

Syntax regels

- iedere code regel wordt afgesloten met een ;
- iedere functie en/of code blok begint met een { en eindigt met een }
- de code instructies zijn hoofdlettergevoelig !!!

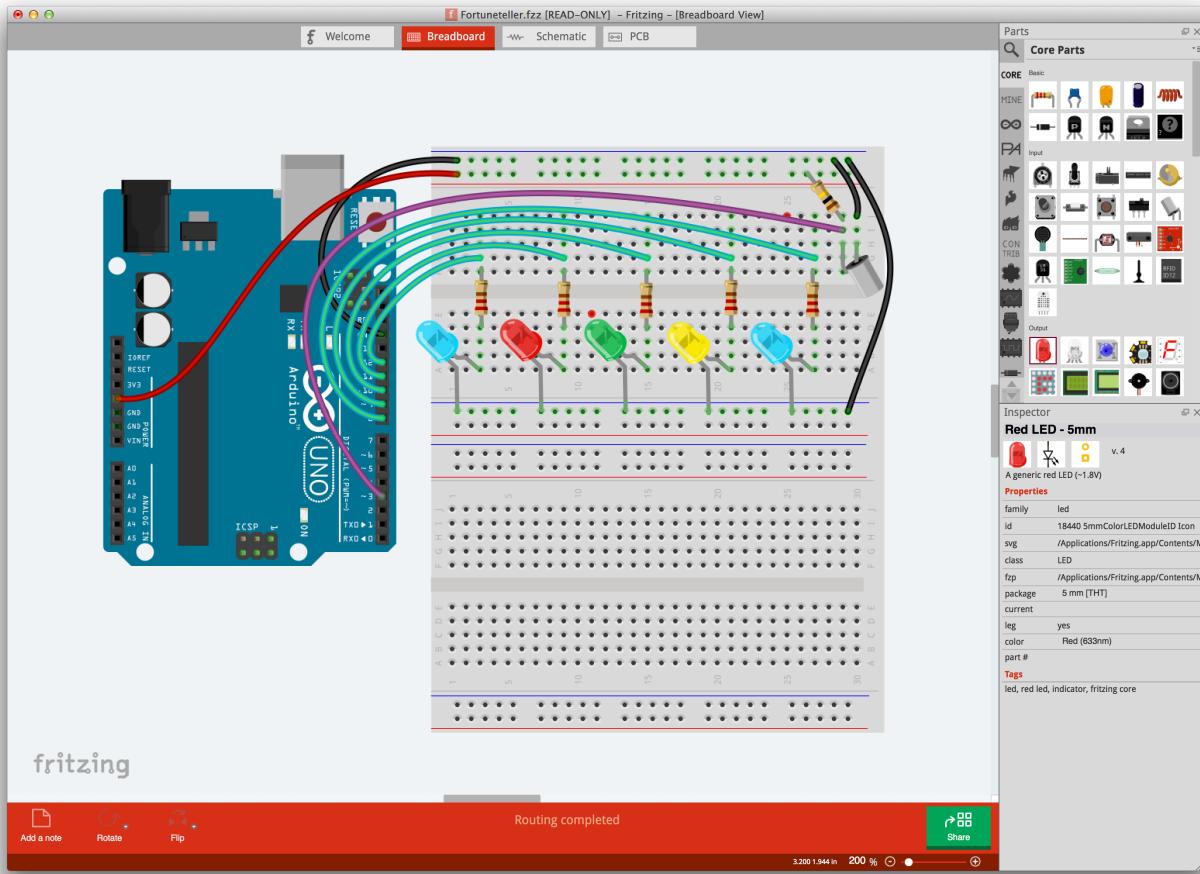
Instructies

Instructie	Uitleg
<code>/* commentaar over meer dan 1 lijn */</code>	Meerdere lijnen commentaar worden omringd door /* en */
<code>// commentaar op 1 lijn</code>	Voor commentaar op 1 lijn staat een //
<code>int helderheid = 0;</code>	Declaratie van de variabele <i>helderheid</i> van het type geheel getal en initialisatie op 0
<code>const int ledPin = 4;</code>	Declaratie van de constante <i>ledPin</i> van het type geheel getal en initialisatie op 4
<code>helderheid++;</code>	Verhoog de variabele <i>helderheid</i> met 1
<code>helderheid--;</code>	Verlaag de variabele <i>helderheid</i> met 1
<code>pinMode(ledPin, OUTPUT);</code>	De poort <i>ledPin</i> wordt geïnitialiseerd als output.
<code>pinMode(switchPin, INPUT);</code>	De poort <i>switchPin</i> wordt geïnitialiseerd als input.
<code>digitalWrite(ledPin, HIGH);</code>	De poort <i>ledPin</i> wordt hoog gezet (3,3V of 5V)
<code>digitalWrite(ledPin, LOW);</code>	De poort <i>ledPin</i> wordt laag gezet (0V)
<code>switchStatus = digitalRead(switchPin);</code>	De status van de schakelaar <i>switchPin</i> wordt toegekend aan de variabele <i>switchStatus</i> . Kan enkel hoog (HIGH) of laag (LOW) zijn.
<code>analogWrite(ledPin, helderheid);</code>	De waarde <i>helderheid</i> wordt op de PWM poort <i>ledPin</i> gezet.
<code>contrast = analogRead(potPin);</code>	De waarde van de potentiometer op <i>potPin</i> wordt toegekend aan de variabele <i>contrast</i> .
<code>Serial.begin(9600);</code>	Initialiseer de seriële monitor op 9600 baud (bits per seconde).
<code>Serial.print("waarde contrast "); Serial.println(contrast);</code>	Schrijf de tekst "waarde contrast " naar de monitor. Schrijf de waarde van de variabele <i>contrast</i> naar de monitor en begin een nieuwe lijn.
<code>frequentie = map(sensorWaarde, sensorLaag, sensorHoog, 50, 4000);</code>	De functie map zet de <i>sensorWaarde</i> met als grenzen <i>sensorLaag</i> en <i>sensorHoog</i> om naar een waarde tussen 50 en 4000 (Hz) en kent die toe aan de variabele <i>frequentie</i> .
<code>tone(buzzerPin, frequentie, duur);</code>	Speel een geluid af op <i>buzzerPin</i> met <i>frequentie</i> (Hz) en <i>duur</i> (ms)
<code>delay(1000);</code>	Wacht 1000 ms (1 seconde)
<code>if (switchPin == HIGH) { // instructies als drukknop is ingedrukt }</code>	Als de variabele <i>switchPin</i> hoog is, voer dan de instructies binnen de {} uit.

Instructie	Uitleg
<pre>if (switchPin == HIGH) { // instructies als drukknop is ingedrukt } else { // instructies als drukknop niet is ingedrukt }</pre>	Als de variabele <i>switchPin</i> hoog is, voer dan de instructies binnen de {} uit. Als dat niet het geval is, voer dan de instructies binnen de else {} uit.
<pre>switch(antwoord) { case 0: // instructies bij antwoord 0 break; case 1: // instructies bij antwoord 1 break; case 2: // instructies bij antwoord 2 break; }</pre>	Afhankelijk van de waarde van <i>antwoord</i> worden de overeenkomstige instructies van de case statement uitgevoerd. Let op dat iedere case afgesloten wordt met een break instructie om de switch test te verlaten,
<pre>for (int x = 1; x < 255; x++) { // uit te voeren instructies }</pre>	Herhaal de instructies binnen de {} voor variabele x, vanaf beginwaarde 1 tot 255 met een verhoging van 1 bij iedere herhaling.

Fritzing

[Fritzing](#) is een open-source teken programma om elektronische schakelingen te ontwerpen. Het is een uitstekende tool om je Arduino projecten mee te documenteren. De software werd ontwikkeld aan de [FH van Potsdam](#) en is beschikbaar voor Windows, Mac en Linux.



Het heeft volgende views :

1. **breadboard** : hier kan je je schakeling visueel ontwerpen op een virtueel prototype bord aan de hand van componenten uit de onderdelen catalogus. Componenten die niet in de catalogus zitten kunnen aangemaakt worden via de Onderdelen Bewerker.
2. **Schema** : hier zie je de logische schakeling van je ontwerp. De componenten worden met hun symbool voorgesteld. Er is een auto routeringsfunctie die de verschillende aansluitingen automatisch probeert te maken.
3. **PCB** : met deze view kan je zelfs een printplaatje voor je schakeling ontwerpen en dit exporteren naar de juiste formaten om het te laten produceren.

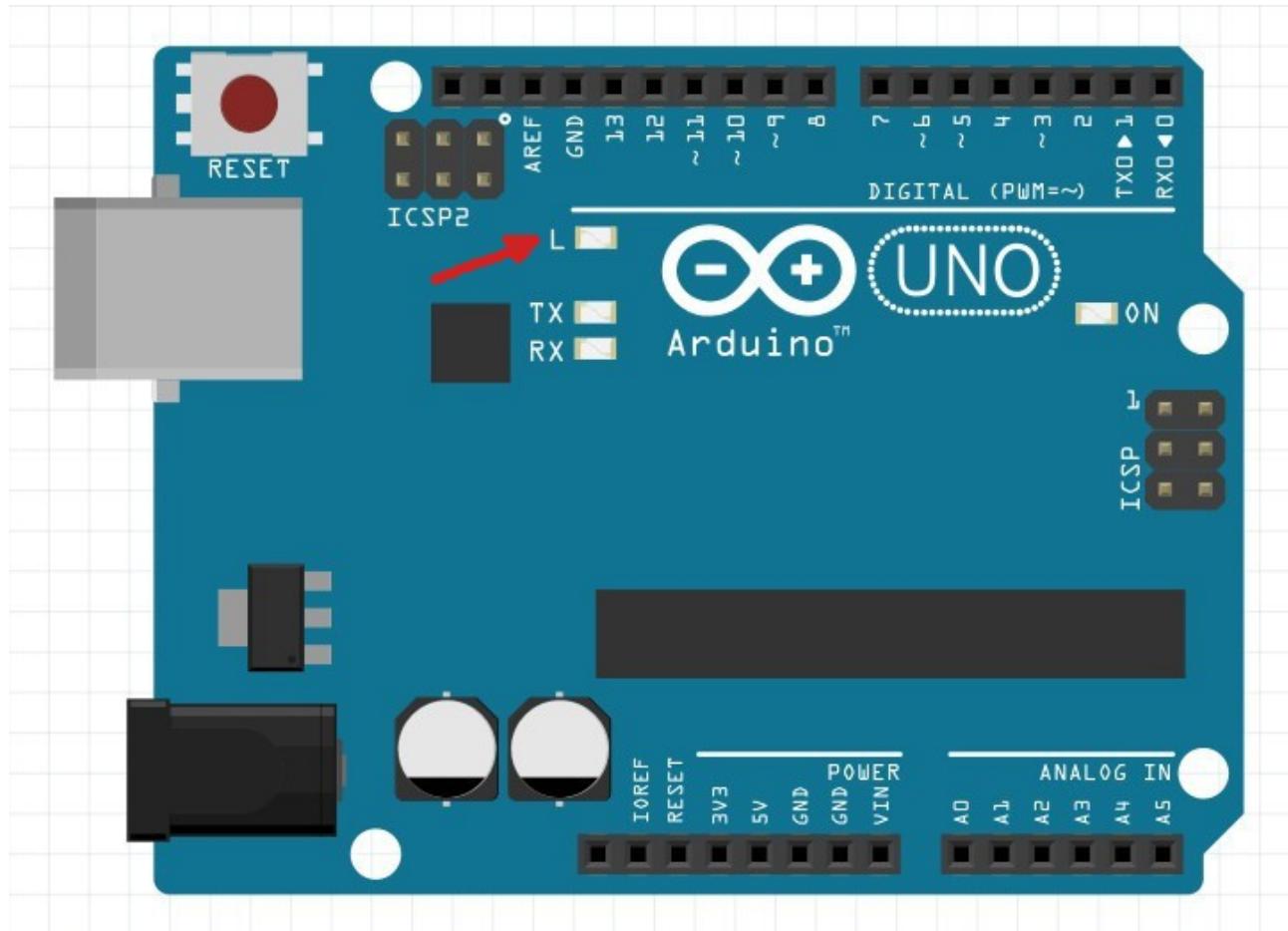
Alle wijzigingen in één van de views worden automatisch ook in de andere views aangebracht.

De website heeft naast de algemene informatie over de tool ook een uitgebreide verzameling studie materiaal, alsook een verzameling van Fritzing projecten die door de gebruikers zijn gedeeld.

Project 1 – Knipperende interne LED

Opgave

Laat de interne LED op het Arduino bordje flikkeren met een frequentie van 2 maal per seconde.



De LED op het bordje (zie rode pijl) wordt eigenlijk bestuurd door digitale poort 13. Je moet dus in dit project niets doen met je jumper kabeltjes en je breadboard. Je moet enkel een kleine sketch schrijven in de IDE omgeving, die laten compileren en uploaden naar je bordje.

Benodigdheden

- Arduino UNO

Sketch

Voor digitale poort 13 wordt de constante **ledPin** gedefinieerd. Die wordt dan in de setup functie met de instructie *pinMode* als output poort geïnitialiseerd.

In de loop functie wordt de digitale poort met de instructie *digitalWrite* eerst op hoog gezet. Vervolgens wordt met de instructie *delay* ½ sec gewacht. Daarna wordt de poort op laag gezet en wordt er terug ½ sec gewacht. Daarna begint delus opnieuw.

Compileer de sketch en als er geen fouten zijn, klik dan op de upload knop en bekijk de interne LED. Die zou nu 2 maal per seconde moeten knipperen.

```
/*
  Knipperende LED

  Laat de interne LED op het Arduino UNO bord knipperen met een frequentie van
  2 maal per seconde

*/

// constanten
const int ledPin = 13; // de digitale poort die de LED aanstuurt

// variabelen

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    // initialiseer digitale poort als output.
    pinMode(ledPin, OUTPUT);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    // zet de LED poort hoog (3.3V) zodat de LED brandt
    digitalWrite(ledPin, HIGH);

    // wacht 500 msec (=1/2 sec)
    delay(500);

    // zet de LED poort laag (0V) zodat LED dooft
    digitalWrite(ledPin, LOW);

    // wacht weer 500 msec
    delay(500);
}
```

Project 2 – Knipperende externe LED

Opgave

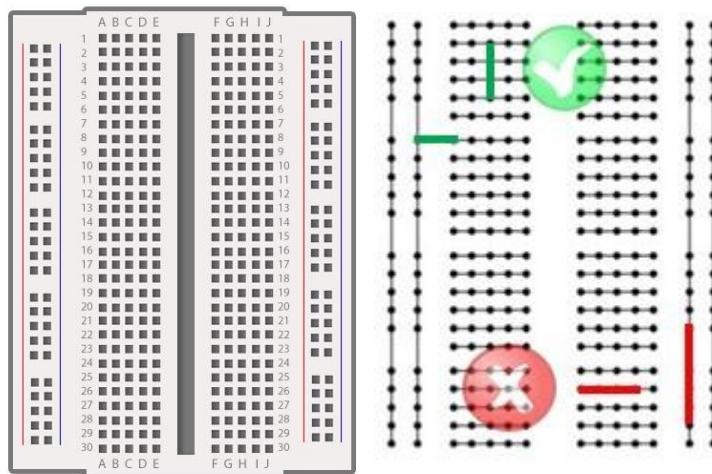
Laat een externe LED flikkeren met een frequentie van 2 maal per seconde.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 LED (om het even welke kleur)
- 1 weerstand 220 ohm

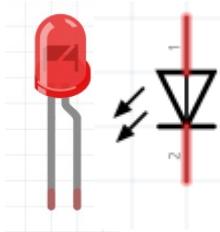
Breadboard

Een breadboard is perfect voor het maken van testopstellingen. Het is in feite een printplaat waarbij je niet hoeft te solderen. Je steekt de verschillende onderdelen eenvoudig in de gaten. Deze gaatjes zijn onderling met elkaar verbonden.



Let op; maak altijd verbindingen tussen de stroken op het breadboard. Een verbinding binnen een strook geeft kortsluiting. Zie ook de schematische weergave van het breadboard hiernaast.

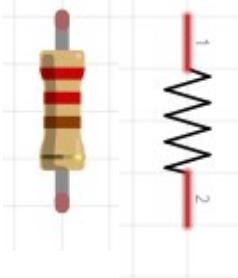
LED



Dit is de afkorting van Light Emitting Diode. Deze component zet elektrische energie om in licht energie. Zoals de gewone diode kan de elektrische stroom slechts in 1 richting lopen.

Het lange beenje is de **anode** en wordt verbonden met de +; het korter beenje is de **kathode** en wordt verbonden met de -. (Ezelsbrugje : KNAP)

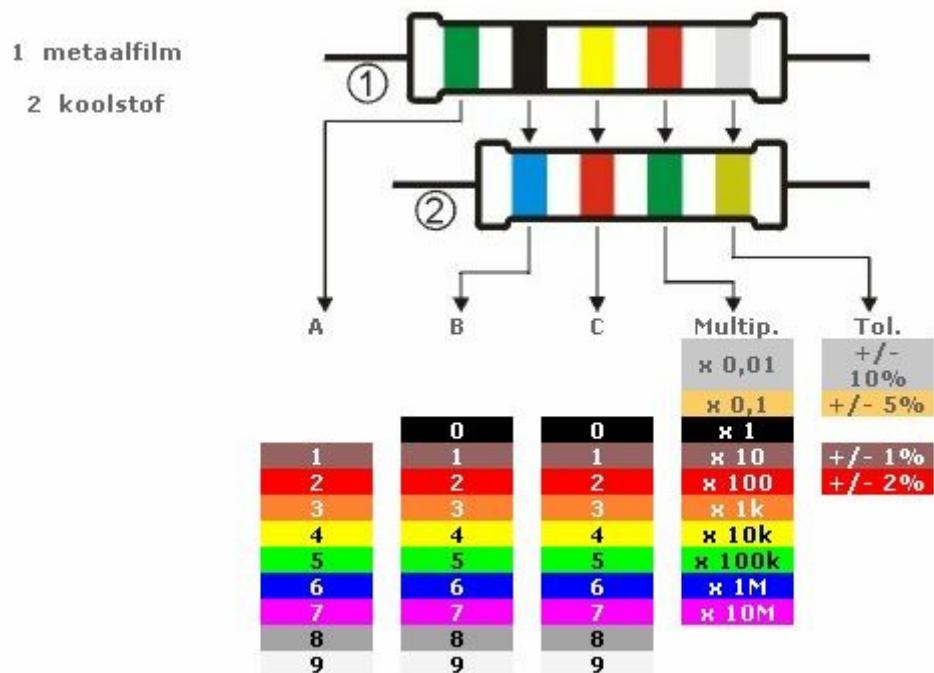
Weerstand



Een weerstand is een component die elektrische energie gedeeltelijk omzet in warmte energie. Deze component wordt meestal in serie geplaatst met andere componenten om een gedeelte van de energie te absorberen zodat de andere component minder energie ontvangt.

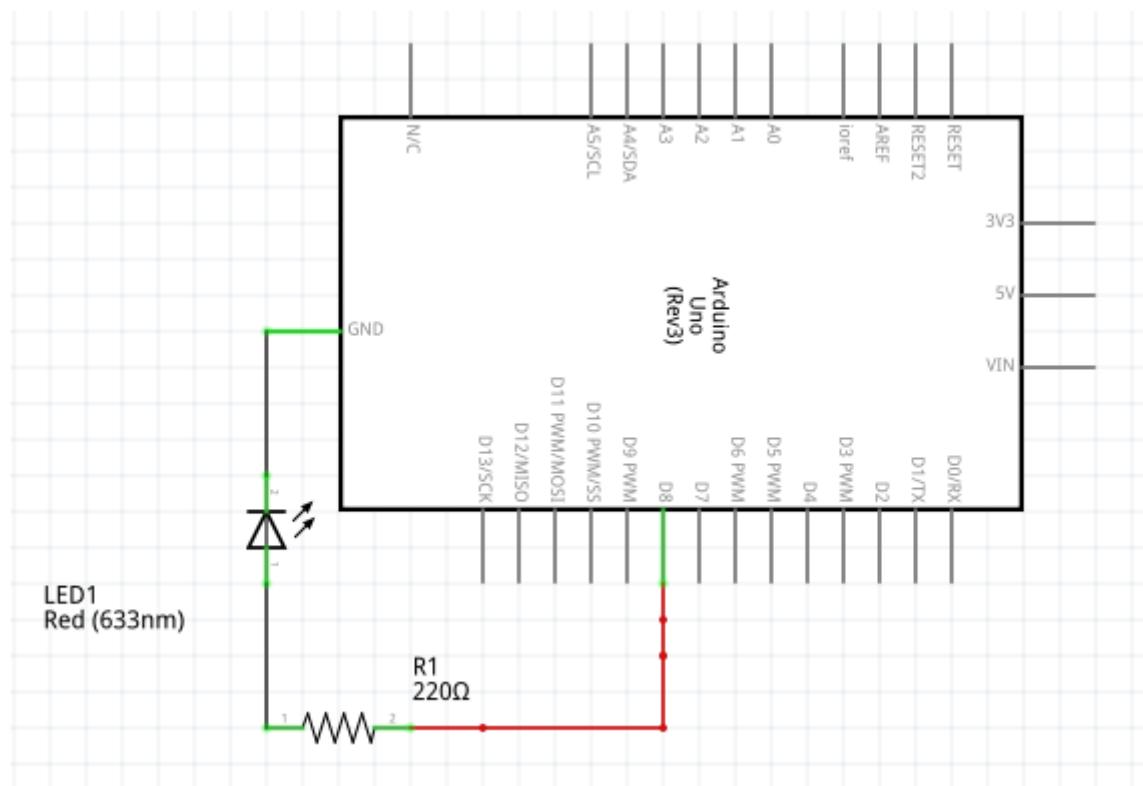
Een typisch voorbeeld is een weerstand in serie met een LED. Zonder de weerstand zou de LED even branden op een te hoge spanning en rap kapot gaan. Met de weerstand ontvangt de LED minder spanning en kan hij normaal werken.

De waarde van de weerstand wordt aangeduid door 4 of 5 gekleurde ringen. De eerste 2 of 3 ringen duiden de waarde aan, de volgende ring de vermenigvuldigingsfactor en de laatste ring duidt de tolerantie (mogelijke afwijking in %) aan.



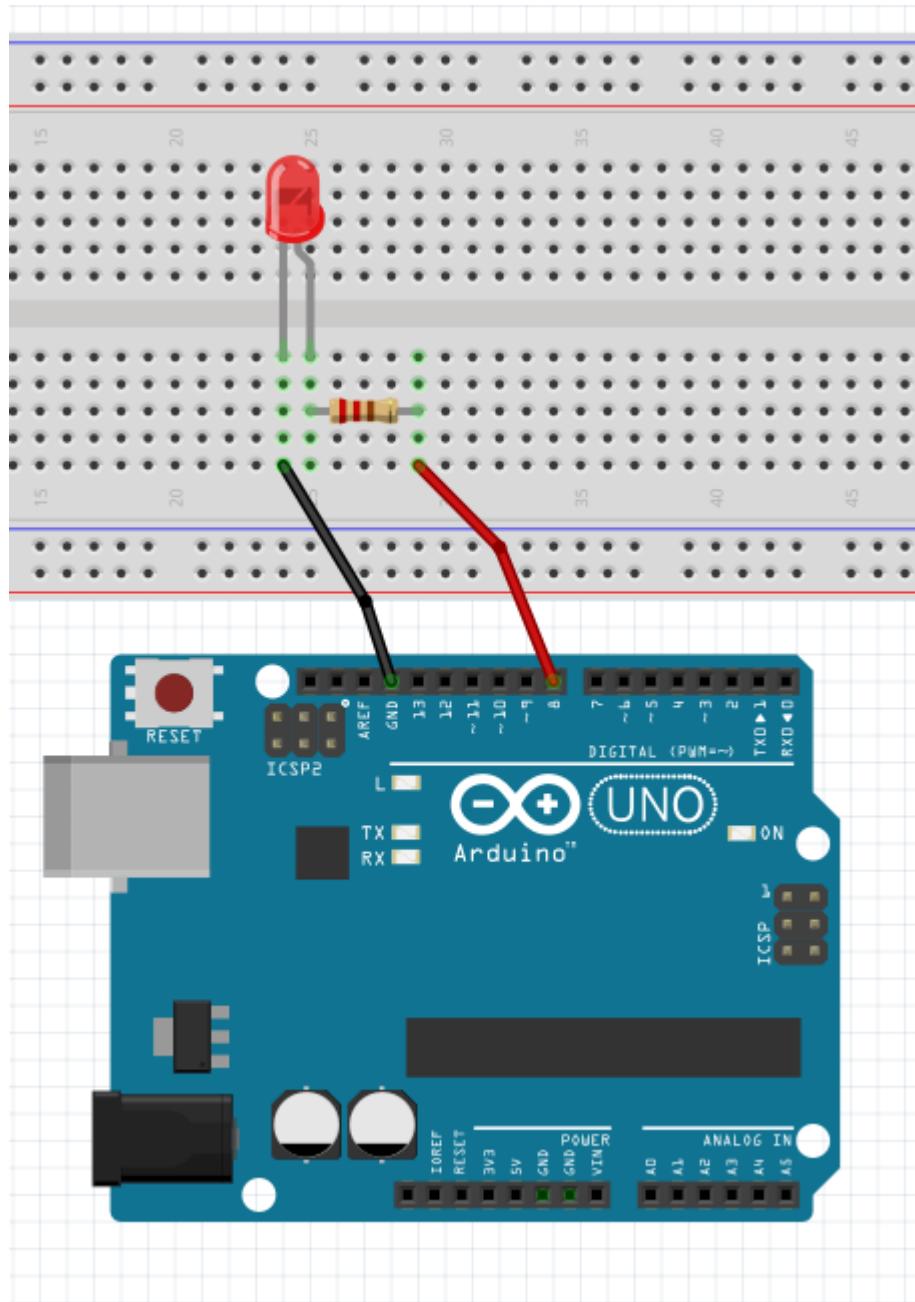
220 ohm met 5% tolerantie is dan : rood – rood – bruin – goud.

Schakeling



De Arduino UNO werkt met spanningen van 3,3V of 5V. Een typische LED kan niet zoveel spanning aan en dus moet er een **weerstand** mee in serie geplaatst worden.

Breadboard schakeling



Sketch

Het sketch programma is bijna identiek aan het vorige programma met de knipperende interne LED. Je moet enkel de digitale poort aanpassen van 13 naar 8. De anode van onze LED is nu via de weerstand aangesloten op digitale poort 8.

Project 3 – Ping pong LEDs

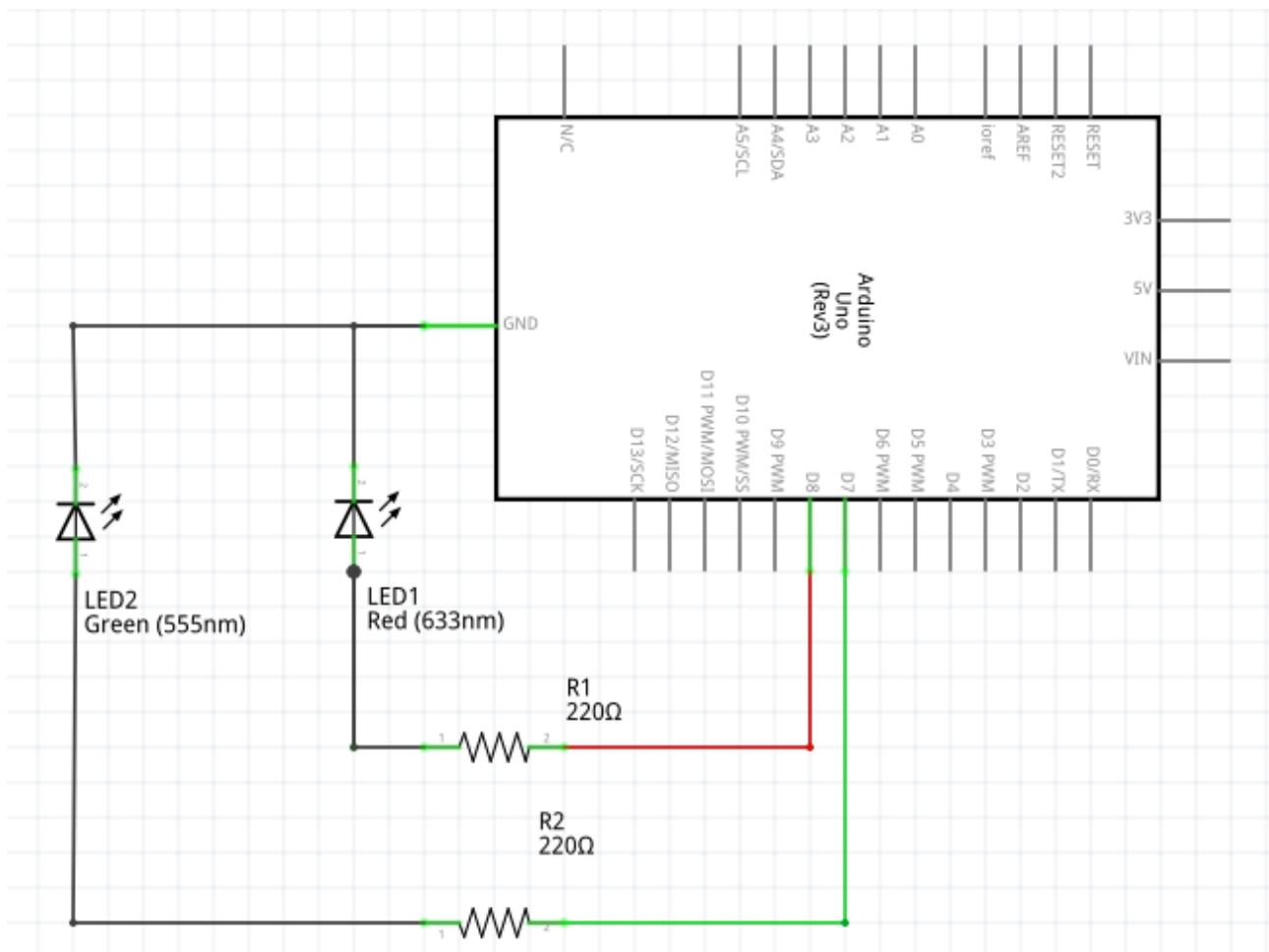
Opgave

Laat de 2 LEDs afwisselend aan en uit gaan gedurende 1 seconde.

Benodigdheden

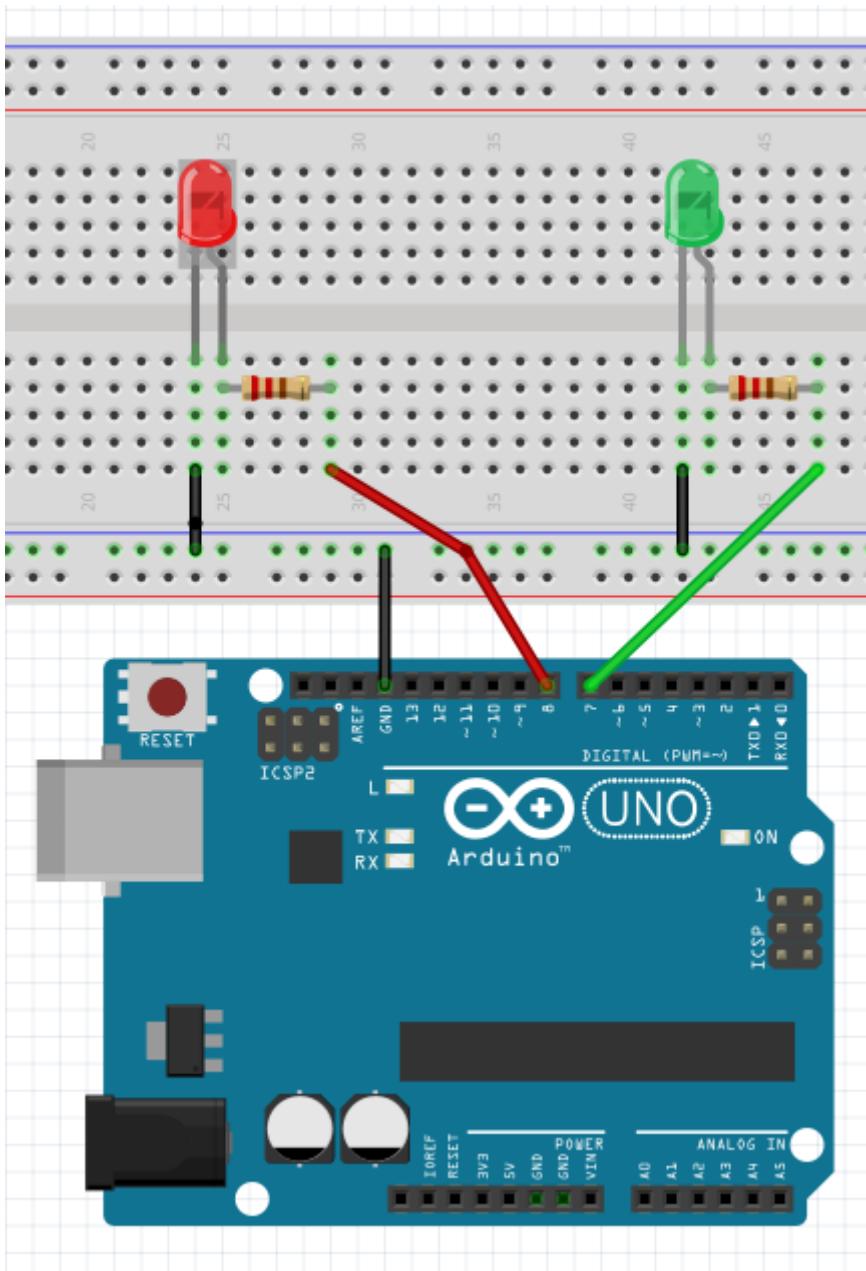
- Arduino UNO, breadboard, jumper kabeltjes
- 1 rode LED
- 1 groene LED
- 2 weerstanden 220 ohm

Schakeling



Let op de polariteit van de LEDs ! We sturen de rode LED aan via digitale poort 8 en de groene LED via digitale poort 7.

Breadboard schakeling



Sketch

We definiëren 2 constanten voor de digitale poorten die de rode en de groene LED aansturen.

In de setup functie definiëren we de beide poorten als output en zetten ze ook beide laag zodat ze gedooft zijn bij de start.

In de loop zetten we eerst de rode LED aan voor 1 sec, daarna zetten we rood uit en zetten direct de groene LED aan voor 1 sec. Daarna zetten we de groene LED uit en herbegint de lus.

```

/*
Ping pong LEDs

Laat een rode en groene LED afwisselend aan en uit gaan gedurende 1 seconde.

*/

// constanten
const int ledRoodPin = 8; // de digitale poort die de rode LED aanstuurt
const int ledGroenPin = 7; // de digitale poort die de groene LED aanstuurt

// variabelen

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    // initialiseer digitale poorten als output.
    pinMode(ledRoodPin, OUTPUT);
    pinMode(ledGroenPin, OUTPUT);

    // zet beide LEDs af
    digitalWrite(ledRoodPin, LOW);
    digitalWrite(ledGroenPin, LOW);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

    // zet de rode LED poort hoog
    digitalWrite(ledRoodPin, HIGH);

    // wacht 1 sec
    delay(1000);

    // zet de rode LED poort laag
    digitalWrite(ledRoodPin, LOW);

    // zet de groene LED poort hoog
    digitalWrite(ledGroenPin, HIGH);

    // wacht 1 sec
    delay(1000);

    // zet de groene LED poort laag
    digitalWrite(ledGroenPin, LOW);
}

```

Project 4 – Verkeerslichten

Opgave

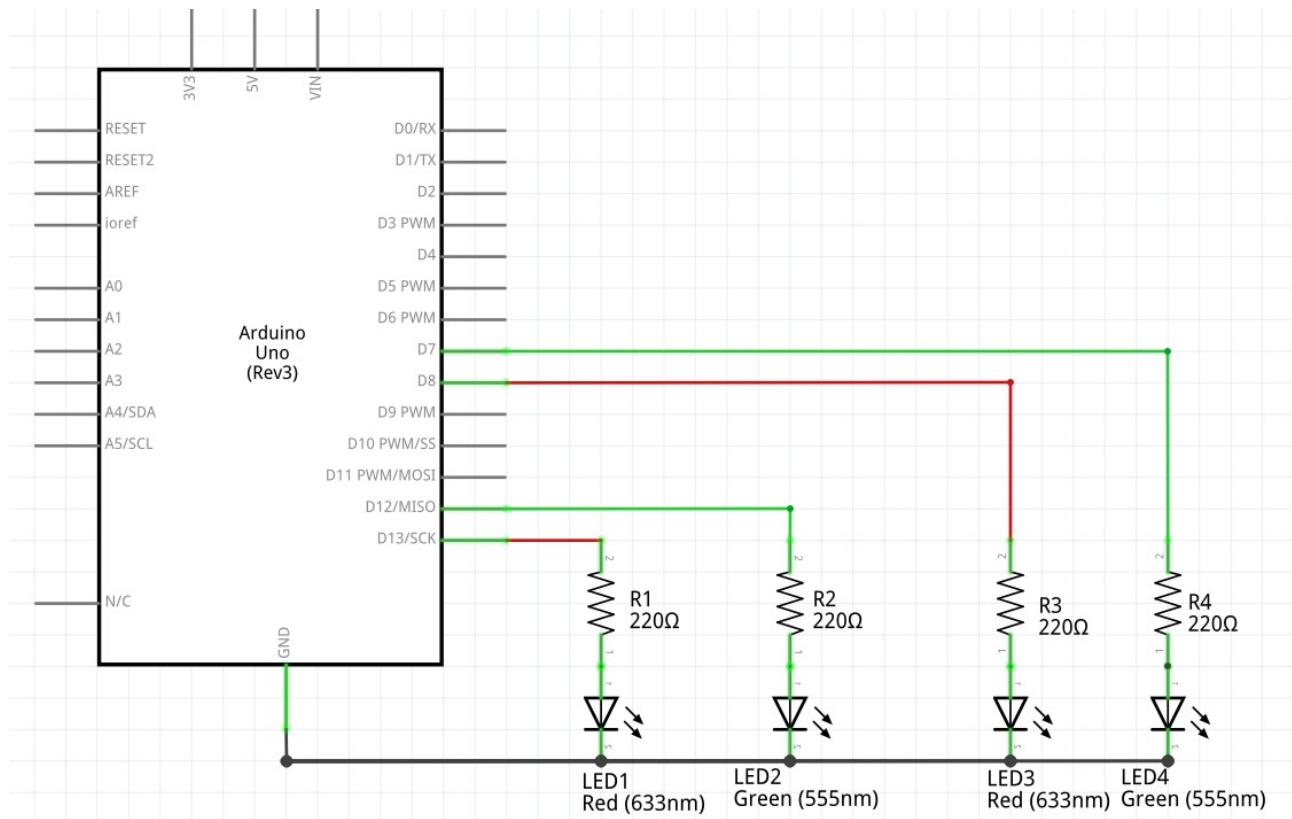
Simuleer 2 verkeerslichten. Als verkeerslicht 1 groen toont, dan moet verkeerslicht 2 rood tonen en omgekeerd. Maak ook dat als verkeerslicht 1 op rood gaat het niet onmiddellijk groen wordt op verkeerslicht 2. Omgekeerd als verkeerslicht 2 rood wordt, dan mag verkeerslicht 1 ook pas met een kleine vertraging groen worden.

De verkeerslichten schakelen om iedere 30 seconden. De vertraging bij groen aan de overkant is 2 seconden.

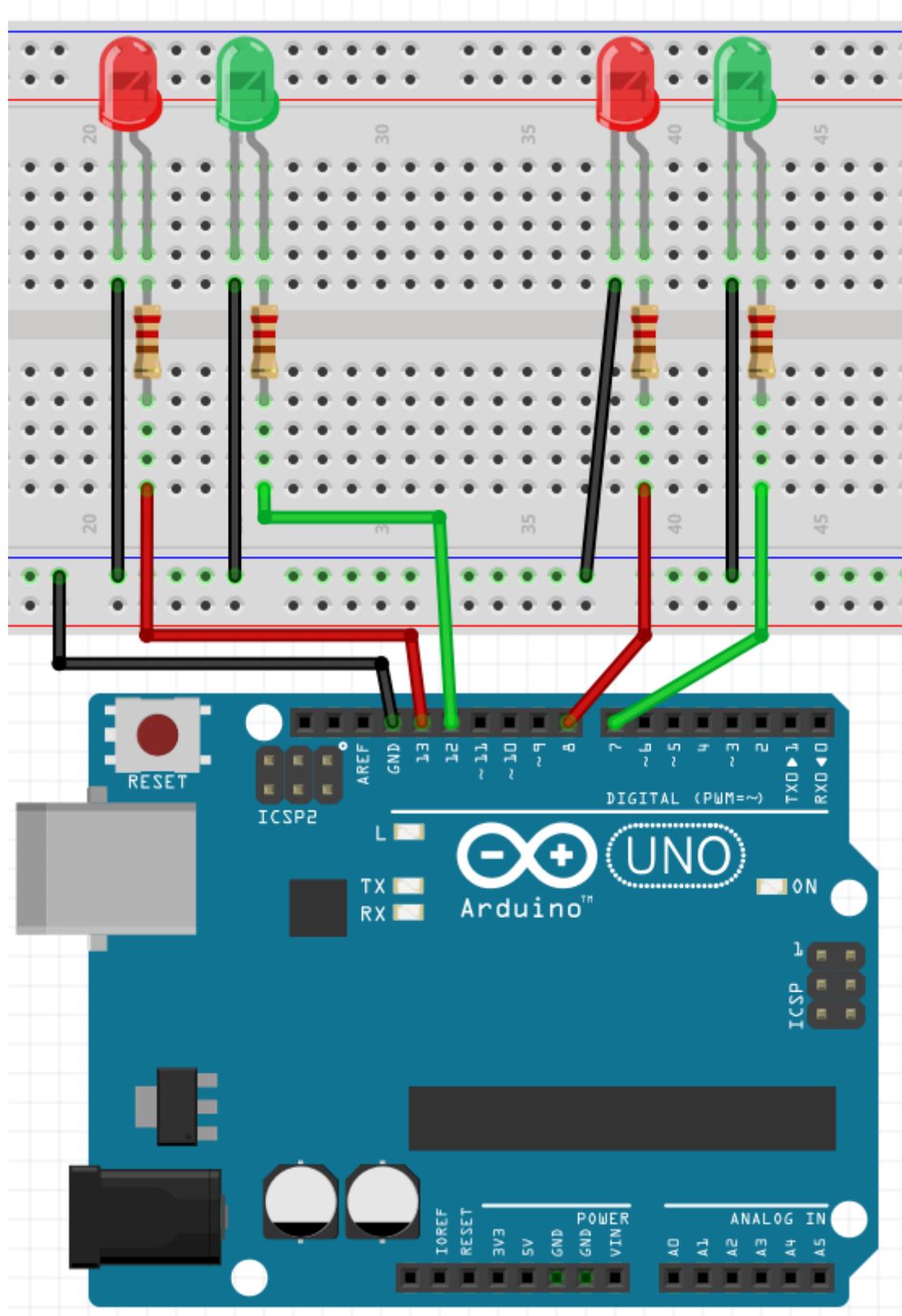
Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 2 rode LEDs
- 2 groene LEDs
- 4 weerstanden 220 ohm

Schakeling



Breadboard schakeling



We gebruiken volgende digitale poorten :

- links rood → 13 - links groen → 12
- rechts rood → 8 - rechts groen → 7

Sketch

We definiëren 4 constanten voor de digitale poorten die de rode en de groene LEDs aansturen.

In de setup functie definiëren we alle poorten als output en zetten ze ook alle laag zodat ze gedooofd zijn bij de start.

In de loop functie beginnen we met rechts rood te zetten. We wachten 2 sec. Daarna zetten we links op groen. Na 30 seconden moeten de verkeerslichten wisselen. We zetten eerst links op rood. We wachten 2 sec en zetten daarna rechts op groen. We wachten terug 30 sec om opnieuw te beginnen.

```

// constanten
const int ledRoodLinksPin = 13; // de digitale poort die de rode LED links aanstuurt
const int ledGroenLinksPin = 12; // de digitale poort die de groene LED links aanstuurt
const int ledRoodRechtsPin = 8; // de digitale poort die de rode LED rechts aanstuurt
const int ledGroenRechtsPin = 7; // de digitale poort die de groene LED rechts aanstuurt

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    // initialiseer digitale poorten als output.
    pinMode(ledRoodLinksPin, OUTPUT);
    pinMode(ledGroenLinksPin, OUTPUT);
    pinMode(ledRoodRechtsPin, OUTPUT);
    pinMode(ledGroenRechtsPin, OUTPUT);

    // zet alle leds af
    digitalWrite(ledRoodLinksPin, LOW);
    digitalWrite(ledGroenLinksPin, LOW);
    digitalWrite(ledRoodRechtsPin, LOW);
    digitalWrite(ledGroenRechtsPin, LOW);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

    // zet rood rechts
    digitalWrite(ledGroenRechtsPin, LOW);
    digitalWrite(ledRoodRechtsPin, HIGH);

    // wacht 2 sec om groen links te zetten
    delay(2000);

    // zet groen links
    digitalWrite(ledGroenLinksPin, HIGH);
    digitalWrite(ledRoodLinksPin, LOW);

    // wacht 30 sec
    delay(30000);

    // zet rood links
    digitalWrite(ledGroenLinksPin, LOW);
    digitalWrite(ledRoodLinksPin, HIGH);

    // wacht 2 sec om groen rechts te zetten
    delay(2000);

    // zet groen rechts
    digitalWrite(ledGroenRechtsPin, HIGH);
    digitalWrite(ledRoodRechtsPin, LOW);

    // wacht 30 sec
    delay(30000);
}

```

Project 5 : LEDs met drukknop

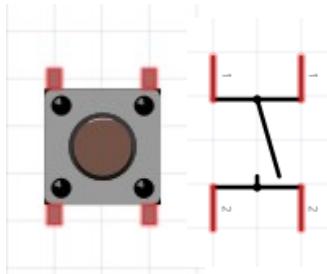
Opgave

Gebruik een drukknop om afwisselend de rode en de groene LED aan en uit te laten gaan. Bij een eerst druk gaat de groene led aan en de rode uit. Bij een volgende druk gaat de groene led uit en de rode led aan.

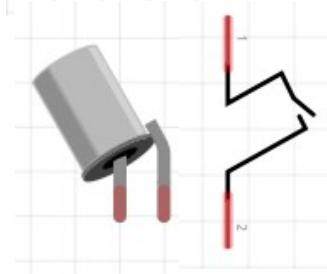
Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 rode LED
- 1 groene LED
- 2 weerstanden 220 ohm
- 1 weerstand 1K ohm
- 1 drukknop

Schakelaars

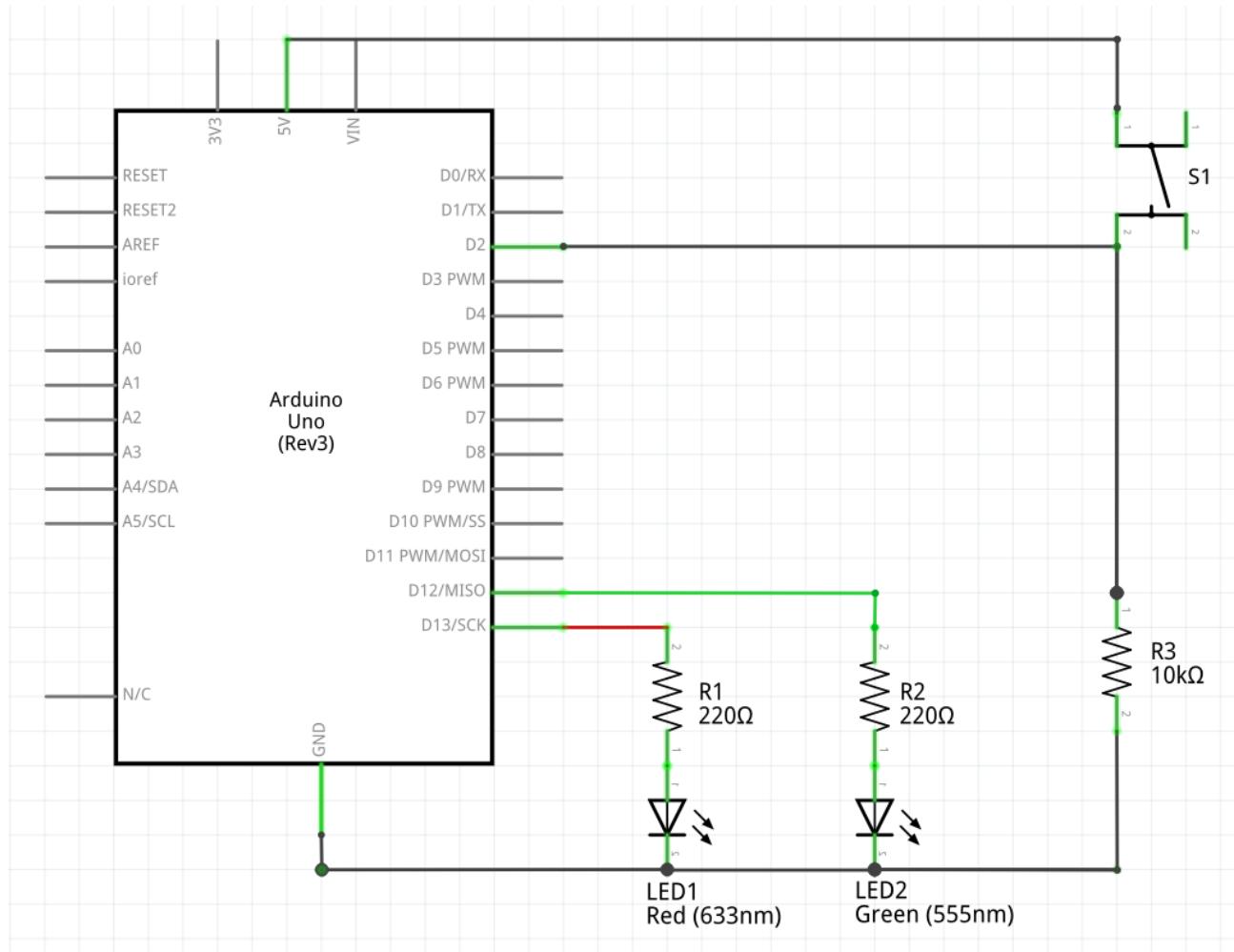


Schakelaars worden gebruikt om een schakeling te onderbreken of te sluiten. Dit type schakelaar is een drukknop. Die heeft 4 contacten die 2 aan 2 verticaal verbonden zijn. Bij het indrukken van de knop wordt een verbinding gemaakt tussen beide horizontale contacten.

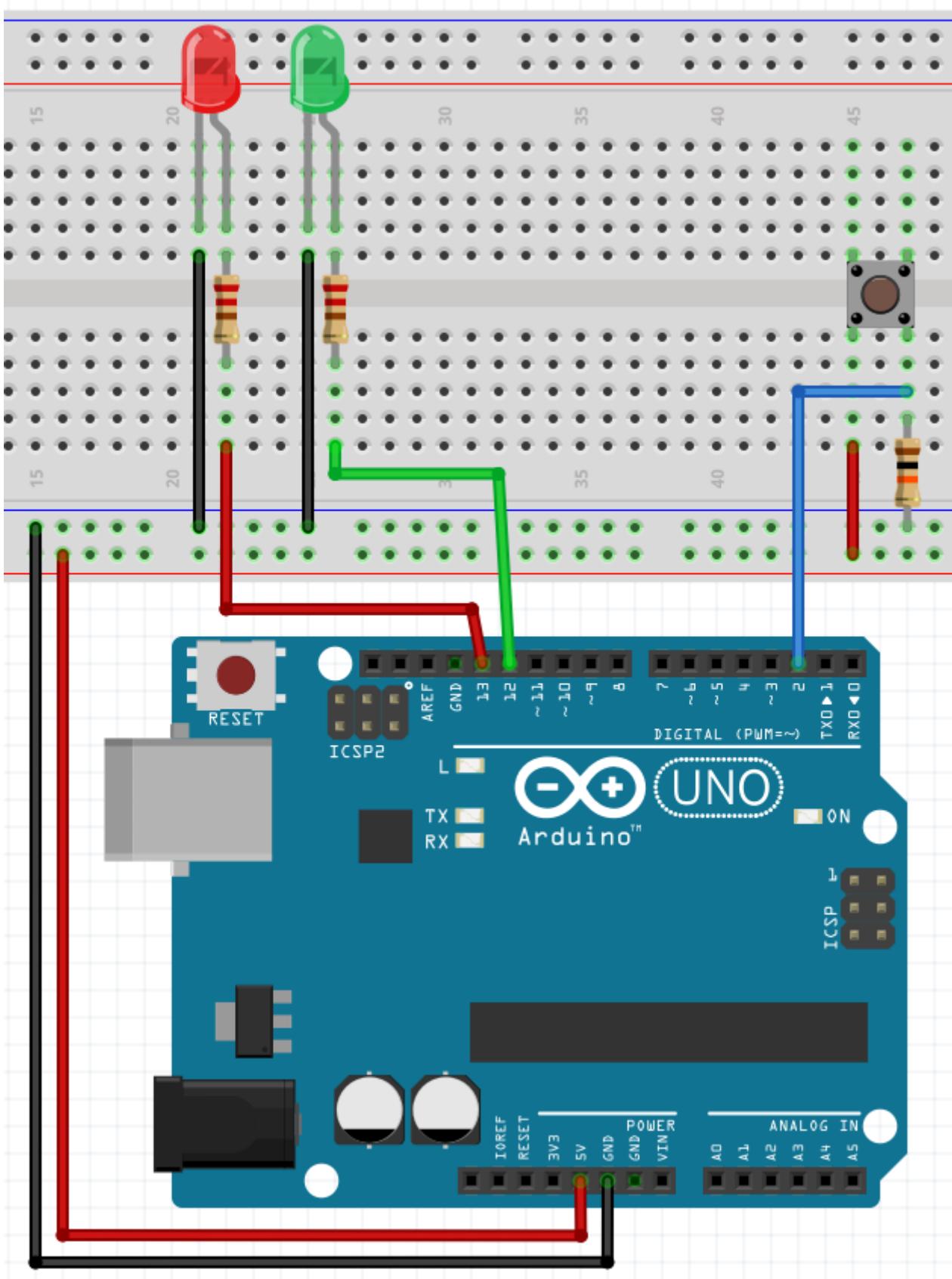


Dit is een tilt schakelaar. In het buisje zit een metalen kogeltje dat afhankelijk van de oriëntatie van de schakelaar de contacten opent of sluit. Meestal zijn de contacten gesloten als de tilt schakelaar verticaal wordt gemonteerd.

Schakeling



Breadboard schakeling



Sketch

We definiëren 2 constanten voor de digitale poorten die de rode en de groene LED aansturen, alsook voor de digitale poort die de status van de drukknop moet lezen.

In de setup functie definiëren we de led poorten als output en de drukknop poort als input.

In de loop functie beginnen we met de status van de drukknop te lezen met de instructie *digitalRead*. We bewaren die in de variabele switchStatus. We kijken vervolgens met een *if* of de drukknop al dan niet is ingedrukt.

Als de drukknop is ingedrukt, dan moeten we vervolgens controleren welke led we moeten aanzetten en welke we moeten uitzetten. Dat doen we door een 2de *if* op de variabele groeneLedAan.

Als de groen led aan is, dan zetten we eerst groen uit en daarna rood aan. In het andere geval doen we het omgekeerde : we zetten eerst rood uit en daarna groen aan. Ok, de leds zijn nu omgeschakeld.

We moeten nu nog de variabele groenLedAan bijwerken. Doordat dit een boolean is die maar 2 waarden kan hebben waar (true) of onwaar (false), kunnen we dit eenvoudig doen met een *not* operator.

Tenslotte moeten we nog een kleine wachtlus inbouwen (250 msec is voldoende) om te voorkomen dat de omschakeling van de leds zo rap gebeurt dat het lijkt of beide leds aanstaan. De UNO voert de loop functie enkele duizenden keren per seconde uit en wij kunnen zo rap de drukknop niet loslaten.

```

// constanten
const int ledRoodPin = 13; // de digitale poort die de rode LED aanstuurt
const int ledGroenPin = 12; // de digitale poort die de groene LED aanstuurt
const int switchPin = 2; // de digitale poort die de status van de drukknop leest

// variabelen
int switchStatus = 0; // houdt de status van de drukknop bij
boolean groeneLedAan = false; // houdt bij als de groene LED aan is

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {

    // initialiseer digitale poorten voor de LEDs als output.
    pinMode(ledRoodPin, OUTPUT);
    pinMode(ledGroenPin, OUTPUT);

    // initialiseer de digitale poort voor de drukknop als input.
    pinMode(switchPin, INPUT);

}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

    // lees de status van de drukknop
    switchStatus = digitalRead(switchPin);

    // is de drukknop ingedrukt ?
    if (switchStatus == HIGH) {
        // drukknop is ingedrukt - we controleren welke LED aan/uit moet
        if (groeneLedAan == true) {
            // groen is aan -> we zetten groen af en rood aan
            digitalWrite(ledGroenPin, LOW);
            digitalWrite(ledRoodPin, HIGH);
        }
        else {
            // groen is af -> we zetten rood af en groen aan
            digitalWrite(ledRoodPin, LOW);
            digitalWrite(ledGroenPin, HIGH);
        }
    }

    // status van groene led bijwerken (status omkeren)
    groeneLedAan = !groeneLedAan;

    // kleine vertraging inbouwen
    delay(250);
}
}

```

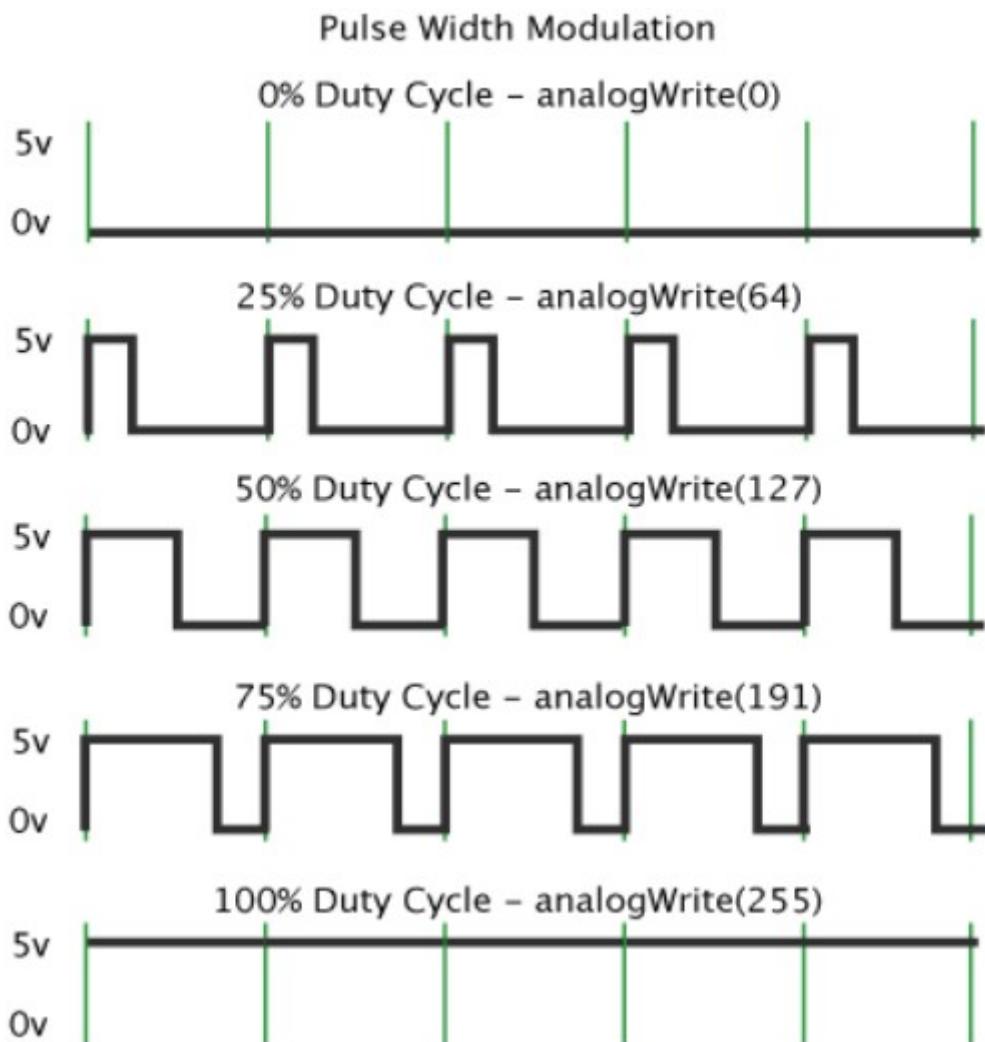
Project 6 : Helderheid LED aanpassen

Opgave

Gebruik een PWM (Pulse Width Modulation) digitale poort om de helderheid van een LED aan te passen. Laat de helderheid toenemen van 0 tot maximum en daarna terug afnemen naar 0.

Pulse Width Modulation (PWM)

Een standaard digitale poort kan normaal slechts 2 spanningen hebben : laag (0V) of hoog (5V). Met PWM kan je echter een blokgolf signaal op een digitale poort zetten. Dit is een spanning die afwisselend laag en hoog is. Afhankelijk van de tijd dat het signaal hoog is spreken we van een bepaald percentage duty cycle. Hiermee kan je ongeveer hetzelfde resultaat simuleren als bij een analoge poort waar je iedere waarde tussen 0 en 5V kan sturen. Voor de PWM digitale poort kan je slechts 256 verschillende waarden tussen 0 en 5V sturen.



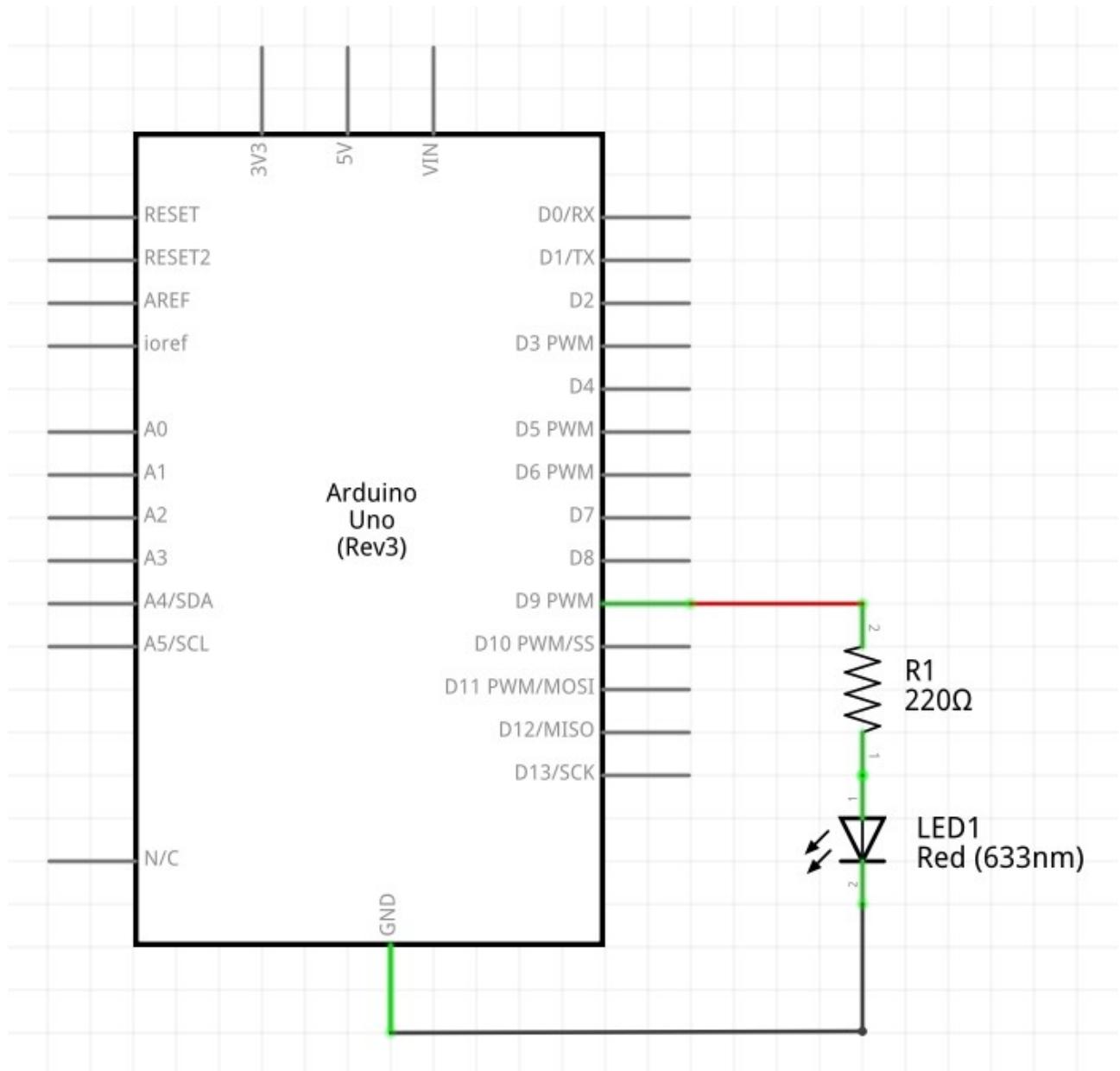
De PWM digitale poort wordt ingesteld met de instructie `analogWrite(pin, waarde)`

- pin : de PWM pin op de UNO (herkenbaar aan het ~ tekentje)
- waarde : tussen 0 (laag) en 255 (hoog)

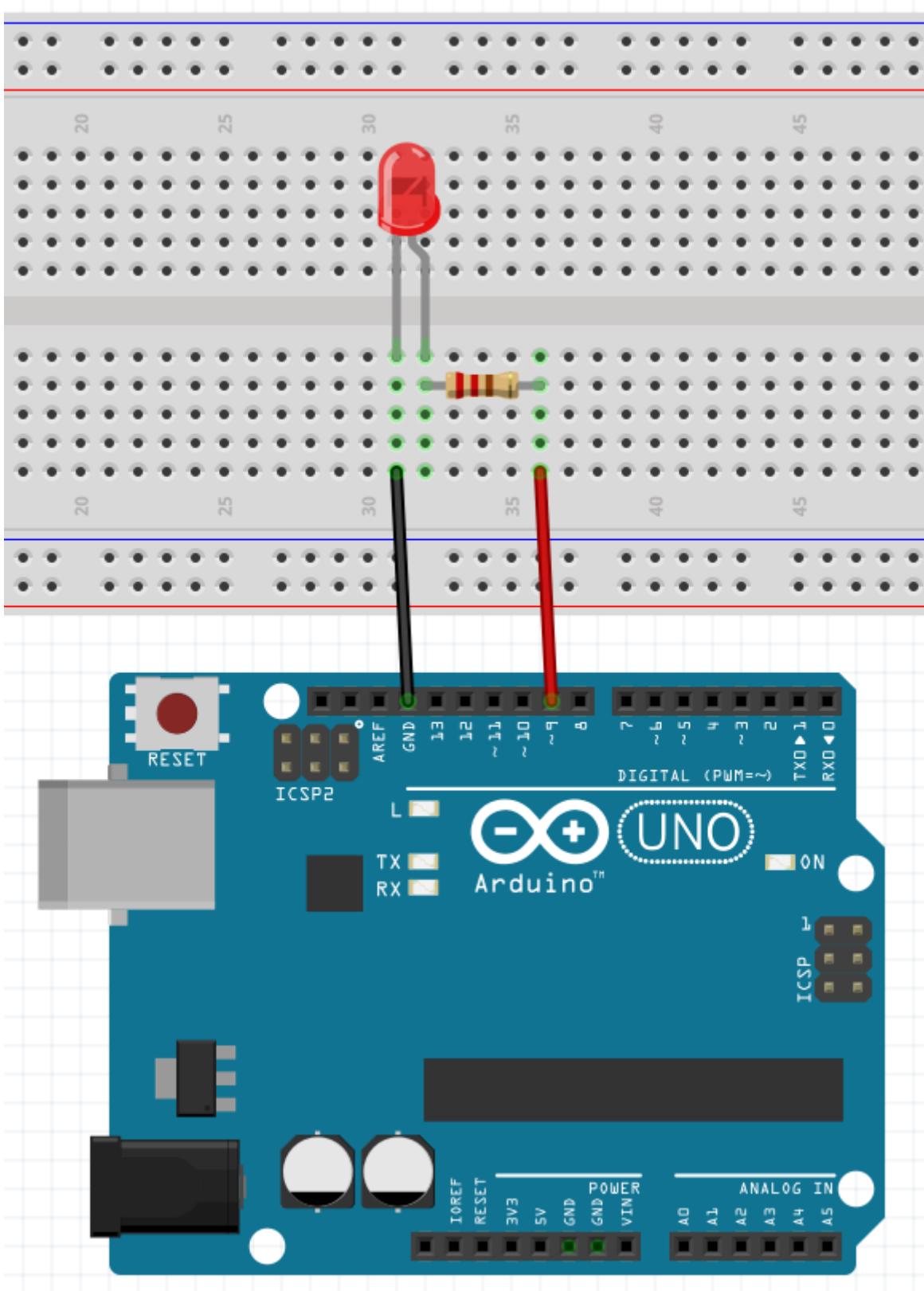
Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 LED (om het even welke kleur)
- 1 weerstand 220 ohm

Schakeling



Breadboard schakeling



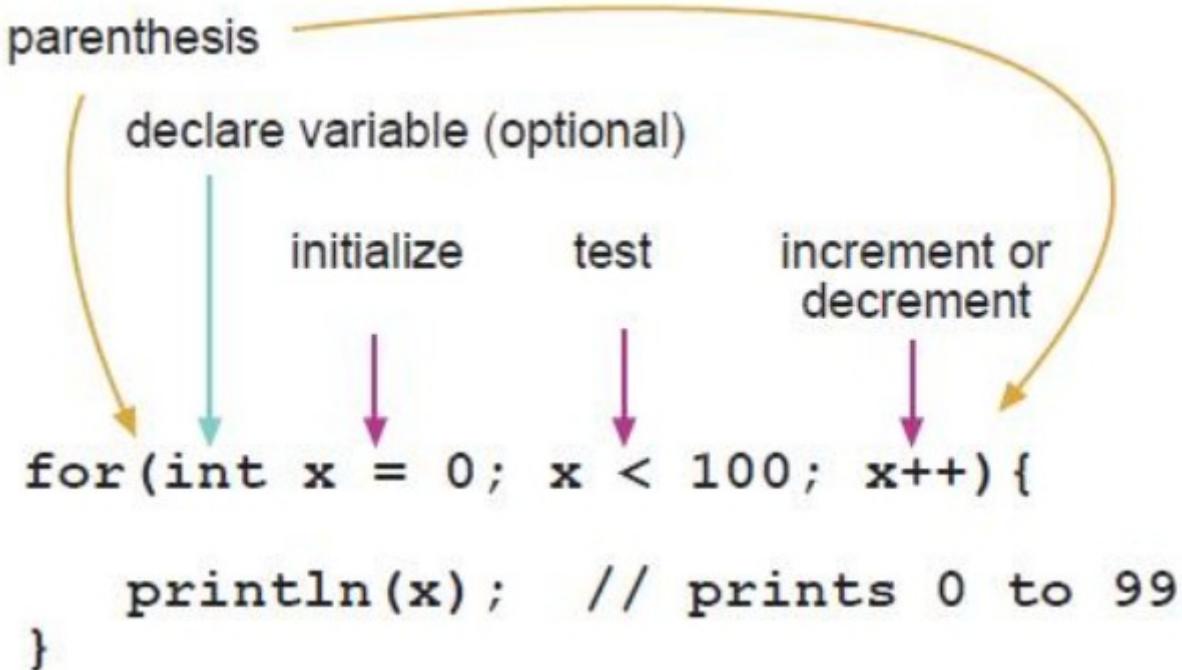
Sketch

We definiëren een constante voor de digitale PWM poort die de LED aanstuurt. We definiëren ook een variabele die de helderheid bevat die we naar de poort zullen sturen.

In de setup functie definiëren we de led poort als output.

In de loop functie gebruiken we 2 lussen om de helderheid van laag naar hoog en weer terug naar laag te brengen. We gebruiken hiervoor de *for* instructie. Die bevat volgende elementen :

- de **beginwaarde** van de lus, eventueel met de declaratie ervoor. Dit is niet verplicht als de variabele reeds gedeclareerd is.
- de **test** die iedere keer gebeurt : heeft de variabele de maximale of minimale waarde bereikt ? Als dat zo is wordt de lus gestopt.
- de waarde waarmee de variabele **verhoogd** of **verlaagd** wordt bij iedere doorloop van de lus.



In de lussen gebruiken we de instructie *analogWrite(ledPin, helderheid)* om het juiste signaal naar de ledpin te sturen. We bouwen ook een kleine vertraging in om het effect beter te kunnen zien.

```

/*
  Helderheid LED aanpassen

Gebruik een PWM (Pulse Width Modulation) digitale poort om de helderheid van een LED aan te passen.
Laat de helderheid toenemen van 0 tot maximum en daarna terug afnemen naar 0.

*/

// constanten
const int ledPin = 9; // de digitale PWM poort die de LED aanstuurt

// variabelen
int helderheid = 0; // bewaart de huidige helderheid

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
  // initialiseer digitale poort als output.
  pinMode(ledPin, OUTPUT);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

  // lus helderheid van laag (0) naar hoog (255)
  for (helderheid = 0; helderheid < 255; helderheid = helderheid + 5) {
    // stuur signaal naar de led poort
    analogWrite(ledPin, helderheid);

    // wacht even om de verandering zichtbaar te maken voor ons
    delay(50);
  }

  // wacht even
  delay(1000);

  // lus helderheid van hoog (255) naar laag (0)
  for (helderheid = 255; helderheid > 0; helderheid = helderheid - 5) {
    // stuur signaal naar de led poort
    analogWrite(ledPin, helderheid);

    // wacht even om de verandering zichtbaar te maken voor ons
    delay(50);
  }

  // wacht even
  delay(1000);
}

```

Project 7 : Driekleuren LED

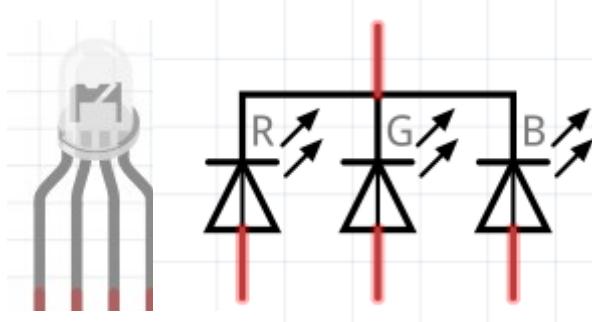
Opgave

Laat een drie kleuren LED (RGB LED) achtereenvolgens rood, groen en blauw schijnen.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Driekleuren (RGB) LED
- 3 weerstanden 220 ohm

RGB LED



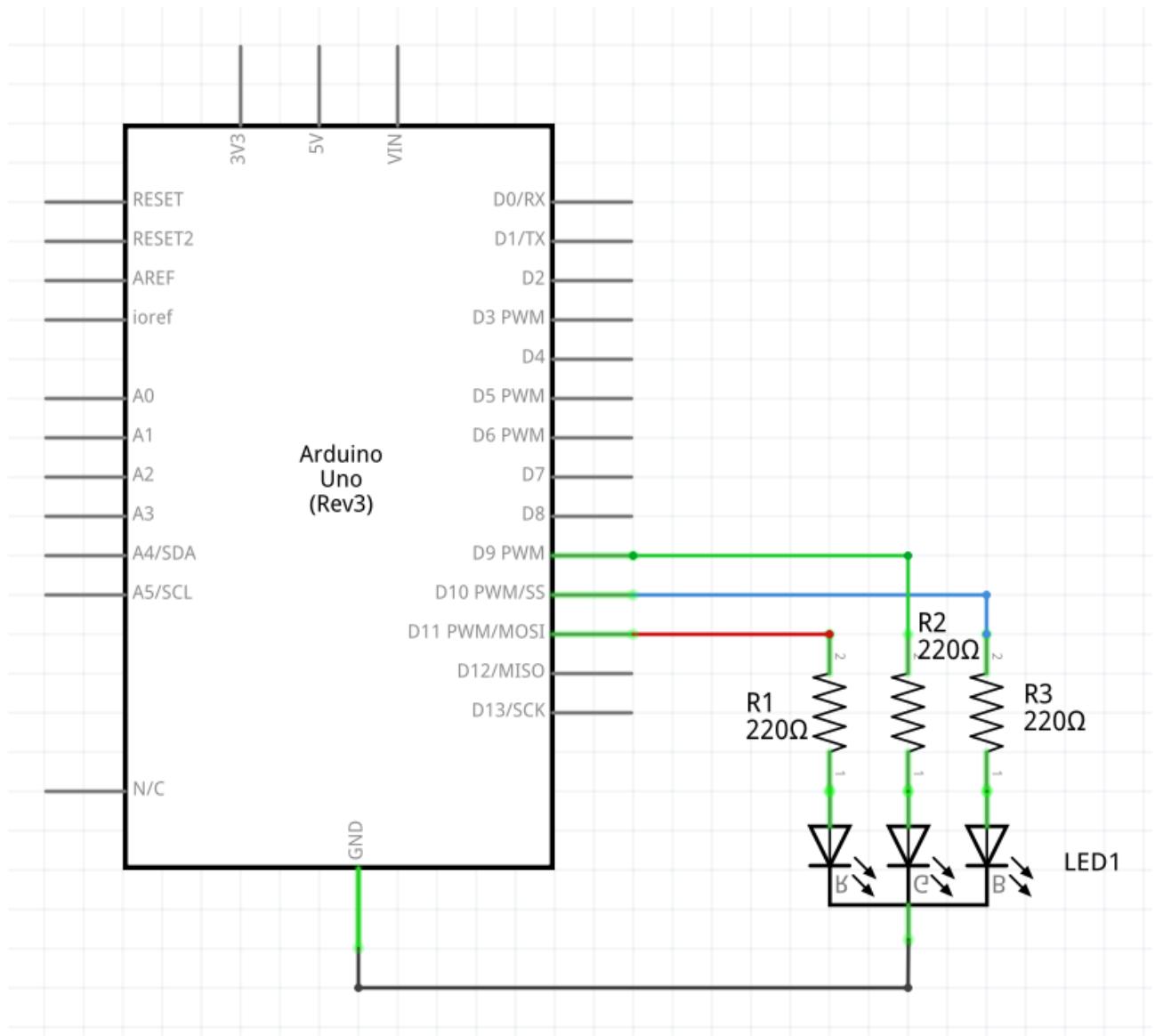
De drie kleuren LED (of RGB LED) is een combinatie van een rode, groene en blauwe led in één behuizing.

Er zijn 4 aansluitingen : een kathode (langste beenntje) die de gemeenschappelijk grond (-) is voor de 3 kleuren en een anode voor elk van de 3 kleuren.

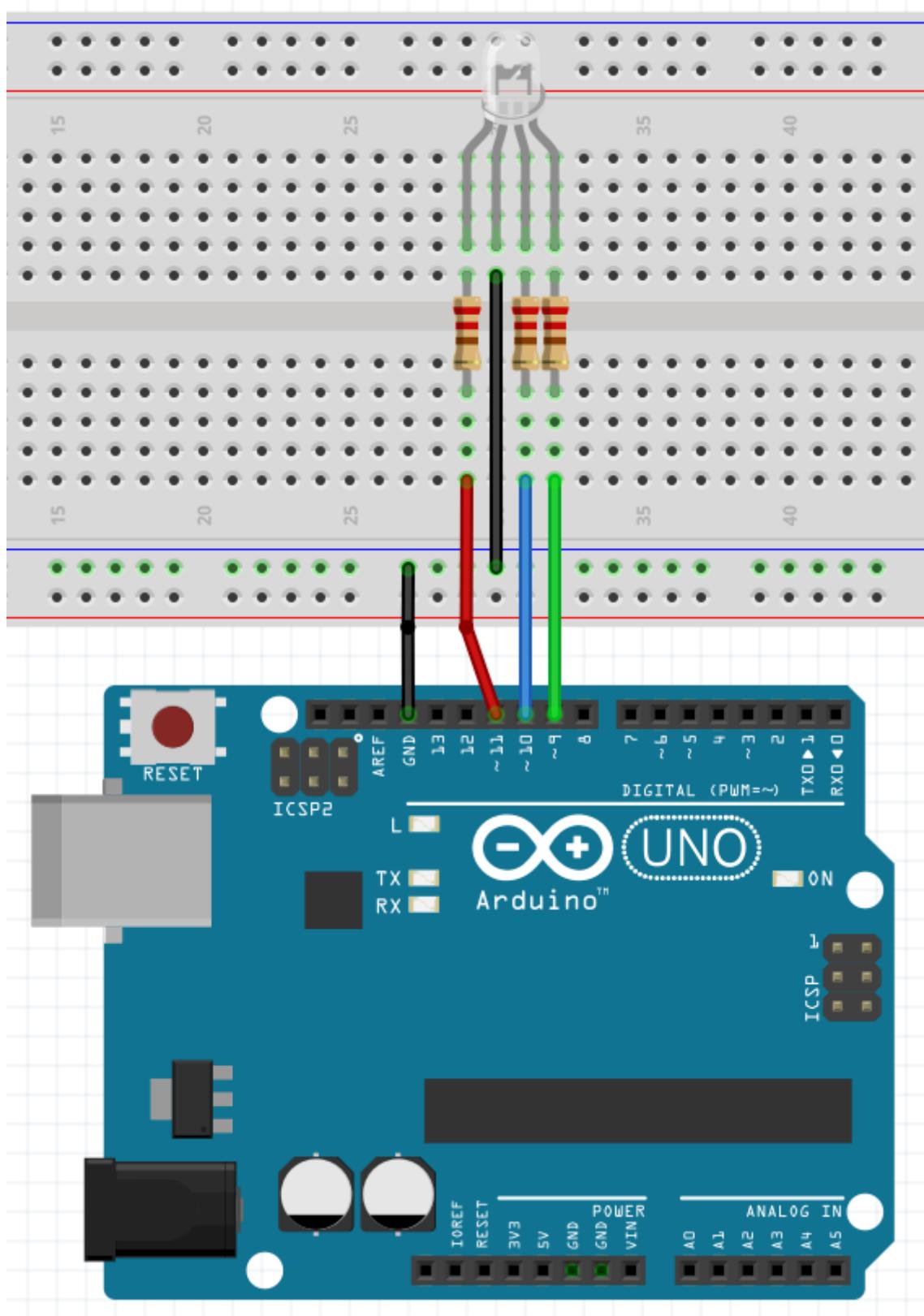


Sommige RGB LEDs zijn gemonteerd op een module met 4 even lange aansluitpinnen. De aansluitingen staan dan op de achterkant van de module aangeduid.

Schakeling



Breadboard schakeling



Sketch

We definiëren 3 constanten voor de pins waarop we de anodes van de RGB LED aansluiten.

We initialiseren deze poorten als output in de setup functie.

In de loop functie zetten we vervolgens voor elk van de 3 kleuren eerst de pin hoog, we wachten 2 sec en zetten daarna de pin terug laag.

```
/*
  Drie kleuren LED of RGB LED

  Laat een drie kleuren LED (RGB LED) achtereenvolgens rood, groen en blauw oplichten.

*/

// constanten
const int roodLedPin = 9;
const int groenLedPin = 10;
const int blauwLedPin = 11;

// variabelen

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {

    // initialisatie van de RGB led pinnen als output
    pinMode(roodLedPin, OUTPUT);
    pinMode(groenLedPin, OUTPUT);
    pinMode(blauwLedPin, OUTPUT);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

    // zet rood aan, wacht 2 sec en zet rood af
    digitalWrite(roodLedPin, HIGH);
    delay(2000);
    digitalWrite(roodLedPin, LOW);

    // zet groen aan, wacht 2 sec en zet groen af
    digitalWrite(groenLedPin, HIGH);
    delay(2000);
    digitalWrite(groenLedPin, LOW);

    // zet blauw aan, wacht 2 sec en zet blauw af
    digitalWrite(blauwLedPin, HIGH);
    delay(2000);
    digitalWrite(blauwLedPin, LOW);
}
```

Project 8 : Spanningsdeler

Opgave

Plaats een vaste weerstand en een variabele weerstand (potentiometer) in serie met elkaar en meet de spanning over de vaste weerstand. Draai aan de potentiometer en toon de resultaten in de seriële monitor.

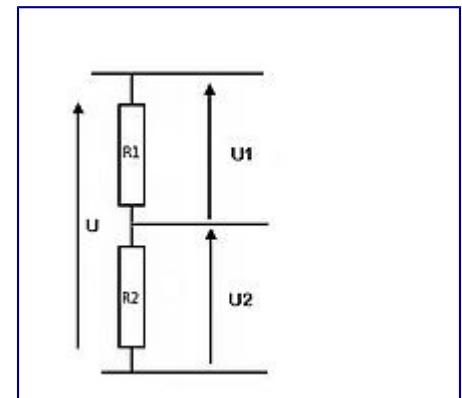
Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- weerstand 10 kohm
- potentiometer 50 kohm

Spanningsdeler

Een **spanningsdeler** is een schakeling die een elektrische spanning in delen splitst. Het doel is om van een beschikbare voedingsspanning een lagere spanning af te leiden.

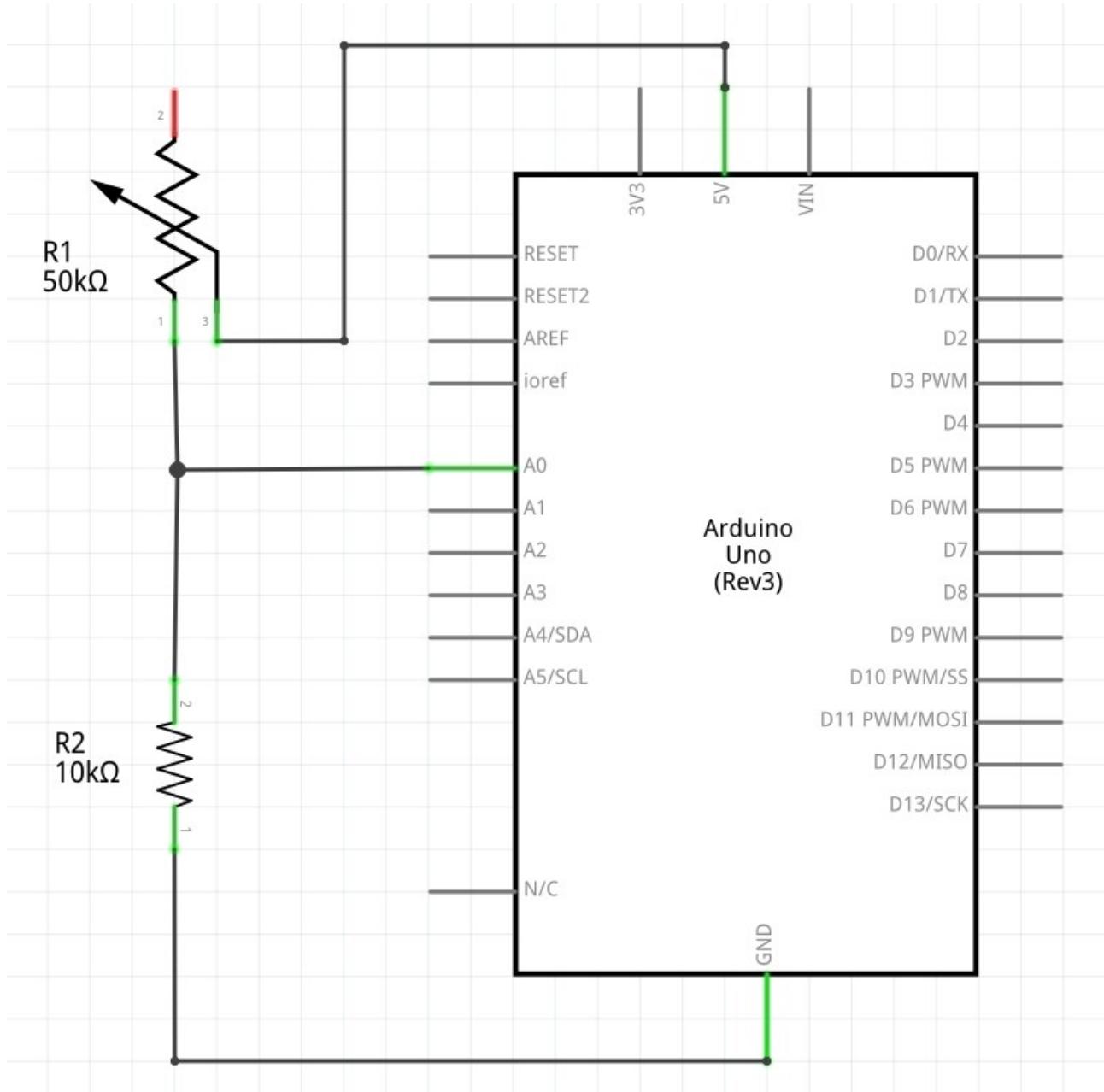
Dit gebeurt dan door de spanningsbron over twee of meer in serie geschakelde weerstanden te zetten. In de simpelste vorm ziet het schema er uit als in de figuur rechts.



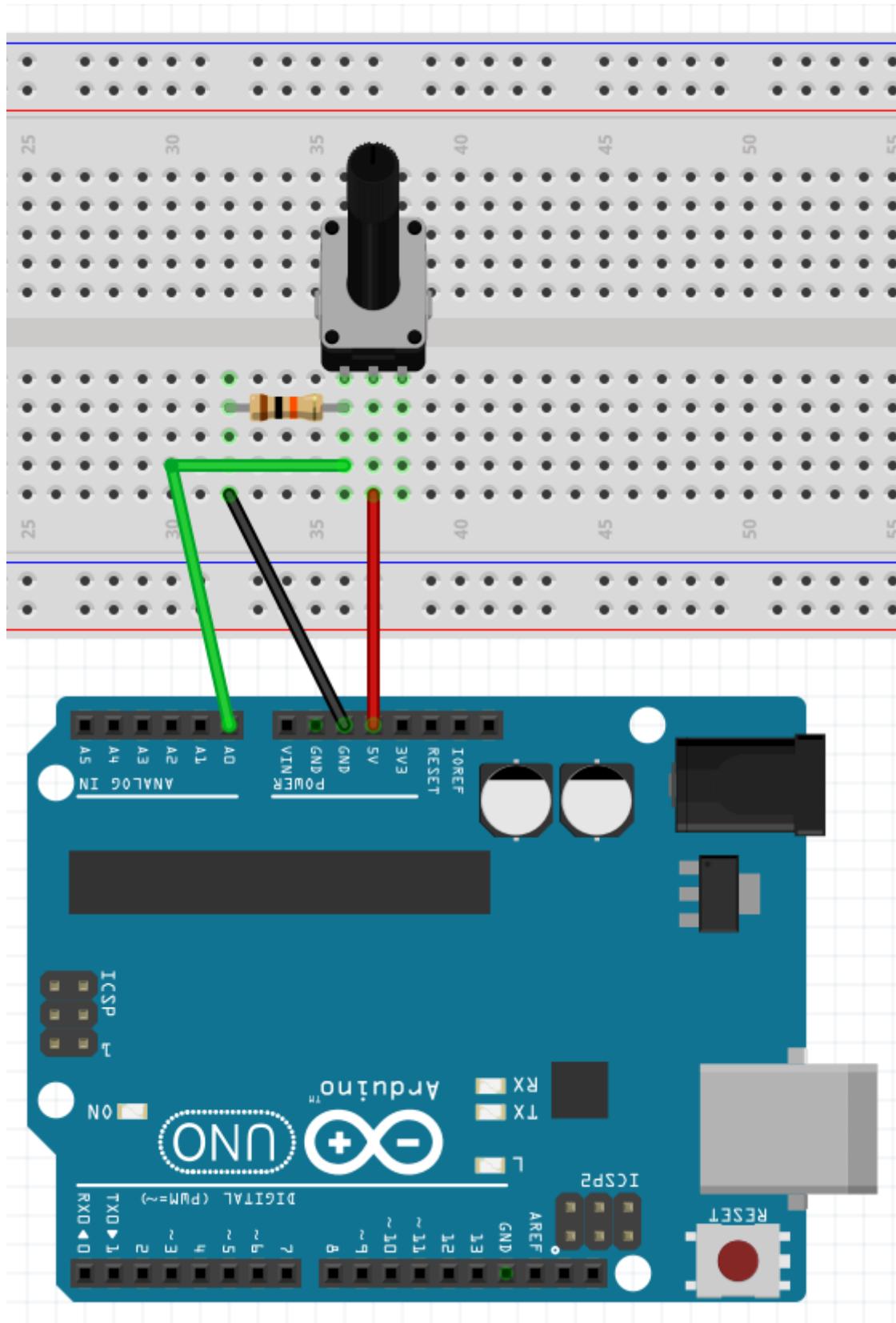
De aangelegde spanning U verdeelt zich over de beide weerstanden en wel precies in de verhouding van hun waarden. Wanneer dus $U = 10$ volt en $R_1:R_2 = 1:9$ dan valt over weerstand R_1 , 1 volt en over R_2 , 9 volt, het gaat om de *verhouding*. De werkelijke waarde doet er dus niet toe. Weerstanden van 1 en 9 ohm geven voor wat de spanningsdeling betreft precies hetzelfde resultaat als weerstanden van $100\text{ k}\Omega$ en $900\text{ k}\Omega$. De keuze van de concrete weerstandswaarden hangt af van de gewenste stroom. In de praktijk berekent men de spanning over R_2 het snelst met de formule:

$$U_2 = U * \frac{R_2}{R_1 + R_2}$$

Schakeling



Breadboard schakeling



Sketch

We declareren een constante voor de analoge poort waarmee we zullen meten, en enkele variabelen voor de meetwaarde en de spanning over de weerstanden.

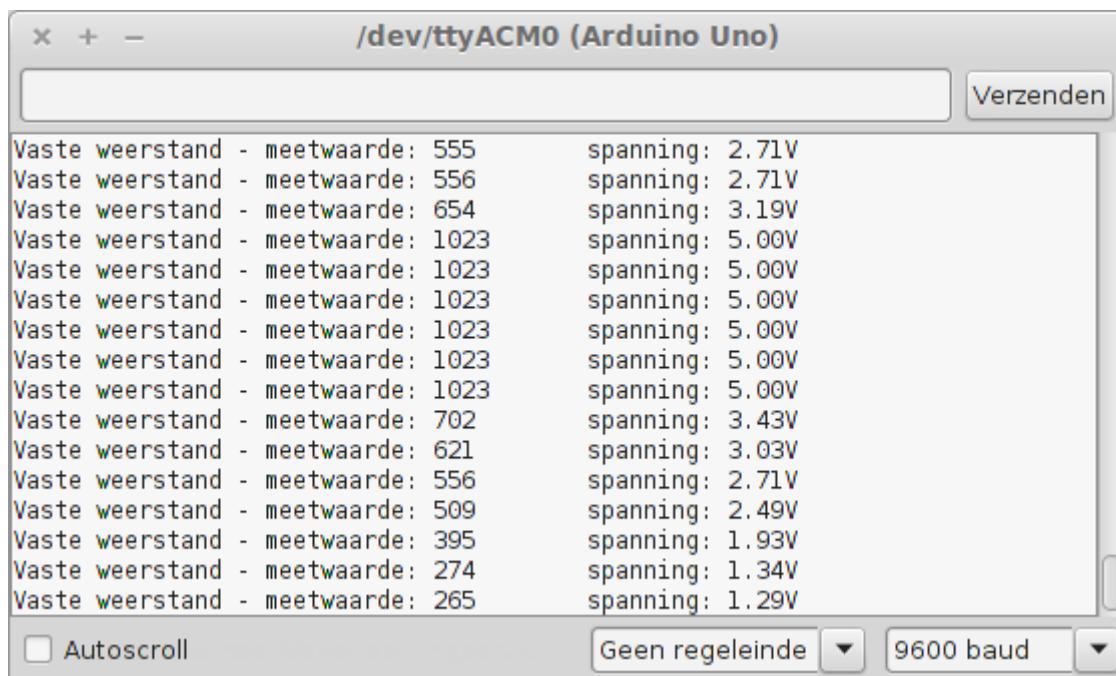
In de setup functie initialiseren we de analoge poort als input. We zetten ook de seriële monitor aan zodat we zelfs de meetwaarden op het scherm kunnen tonen.

In de loop functie beginnen we met de meetwaarde uit te lezen en deze te tonen in de seriële monitor. Daar deze meetwaarden een bereik hebben van 0 (laag) tot 1023 (hoog) moeten we de meetwaarde omzetten via deze formule om de werkelijke spanning te bekomen :

$$\text{spanning} = (\text{meetwaarde} / 1024.0) * 5.0 \text{ V}$$

Opgelet : we gebruiken decimale getallen omdat de variabele spanning als decimaal (float) is gedeclareerd.

We tonen de spanning in de seriële monitor en bouwen een vertraging van 1 sec zodat de UNO kan volgen met lezen.



The screenshot shows the Arduino Serial Monitor window titled '/dev/ttyACM0 (Arduino Uno)'. The window displays a list of measurements from a potentiometer connected to an Arduino Uno. Each row shows the raw analog value followed by its calculated voltage. The calculated voltage values are color-coded: green for values between 0 and 5.0V, yellow for values between 5.0V and 10.0V, and red for values above 10.0V. The bottom of the window includes controls for 'Autoscroll' (unchecked), 'Geen regeleinde' (selected), and baud rate selection (set to 9600).

Vaste weerstand - meetwaarde:	spanning:
555	2.71V
556	2.71V
654	3.19V
1023	5.00V
702	3.43V
621	3.03V
556	2.71V
509	2.49V
395	1.93V
274	1.34V
265	1.29V

```

/*
  Spanningsdeler

  Plaats een vaste weerstand en een variabele weerstand (potentiometer) in serie met elkaar
  en meet de spanning over de vaste weerstand. Draai aan de potentiometer en toon de
  resultaten in de seriële monitor.

*/
// constanten
const int meetPin = A0;          // de analoge poort voor de meting

// variabelen
int meetwaarde = 0;             // houdt de ruwe meetwaarde bij
float spanning = 0.0;           // spanning over de vaste weerstand

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {

  // initialiseer analoge poort voor de meting als input.
  pinMode(meetPin, INPUT);

  // initialiseer de seriële monitor op 9600 baud (bits/sec).
  Serial.begin(9600);

}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

  // lees de waarde op de meetPin
  meetwaarde = analogRead(meetPin);

  // toon de meetwaarde in de seriële monitor
  Serial.print("Vaste weerstand - meetwaarde: ");
  Serial.print(meetwaarde);

  // De meting van de analoge poort geeft een resultaat van 0 tot 1023
  // We gebruiken de regel van 3 om de meetwaarde om te zetten naar de werkelijke spanning
  spanning = (meetwaarde/1024.0) * 5.0;

  // toon de spanning in de seriële monitor
  Serial.print("\t spanning: ");
  Serial.print(spanning);
  Serial.println("V");

  // kleine vertraging inbouwen
  delay(1000);

}

```

Project 9 : Zoemer

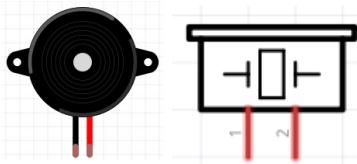
Opgave

Maak een sirene met de passieve piezo zoemer die 4 maal van laag naar hoog en terug naar laag gaat.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Passieve piezo zoemer

Piezo zoemer



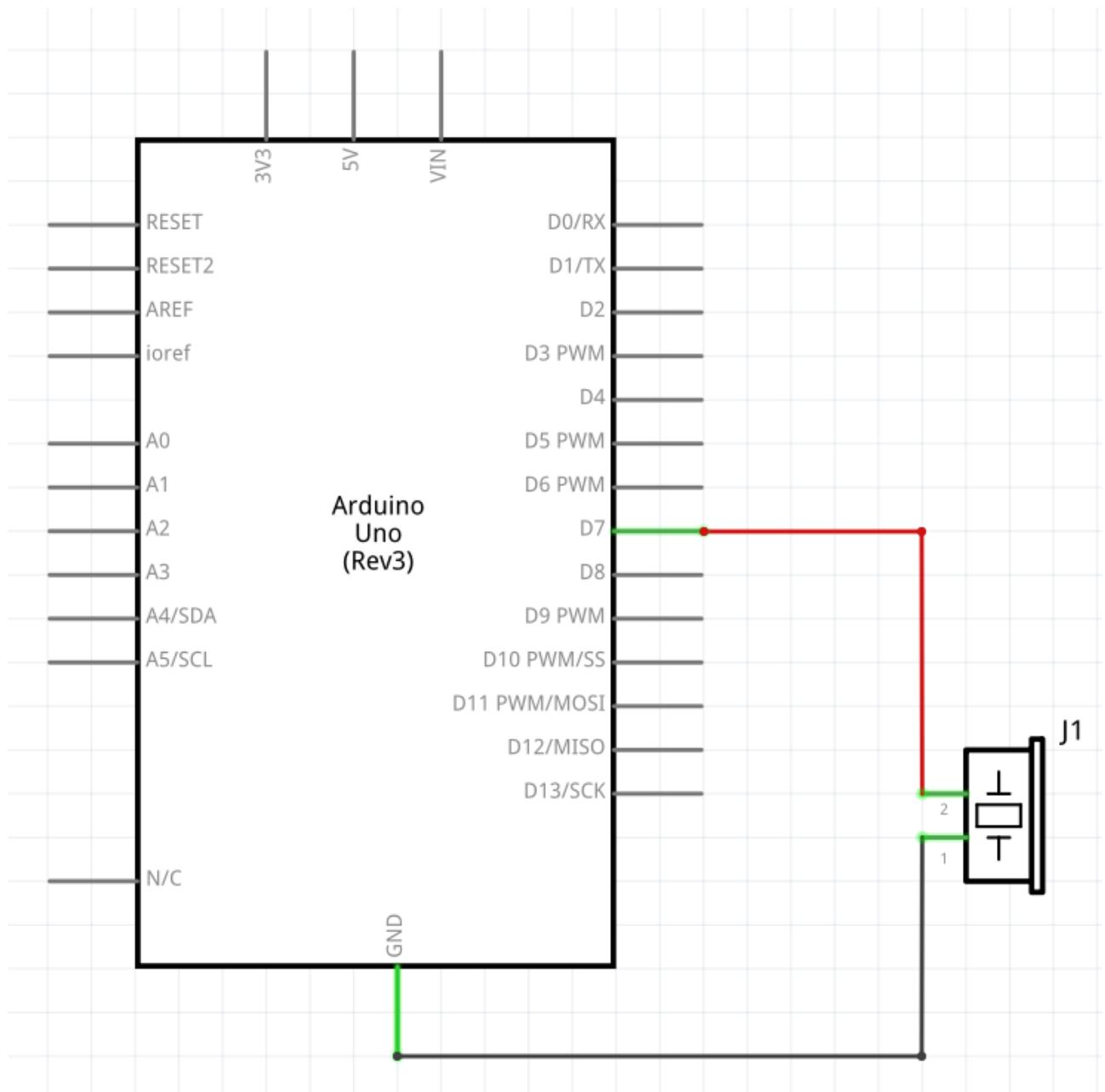
Een piezo zoemer is een elektronisch component die elektrische energie omzet in mechanische energie in de vorm van trillingen. Het zijn sinusvormige geluidsgolven die typisch in de hoorbare frequenties van 20 Hz tot 20 kHz vallen.

Er zijn 2 types piezo zoemers :

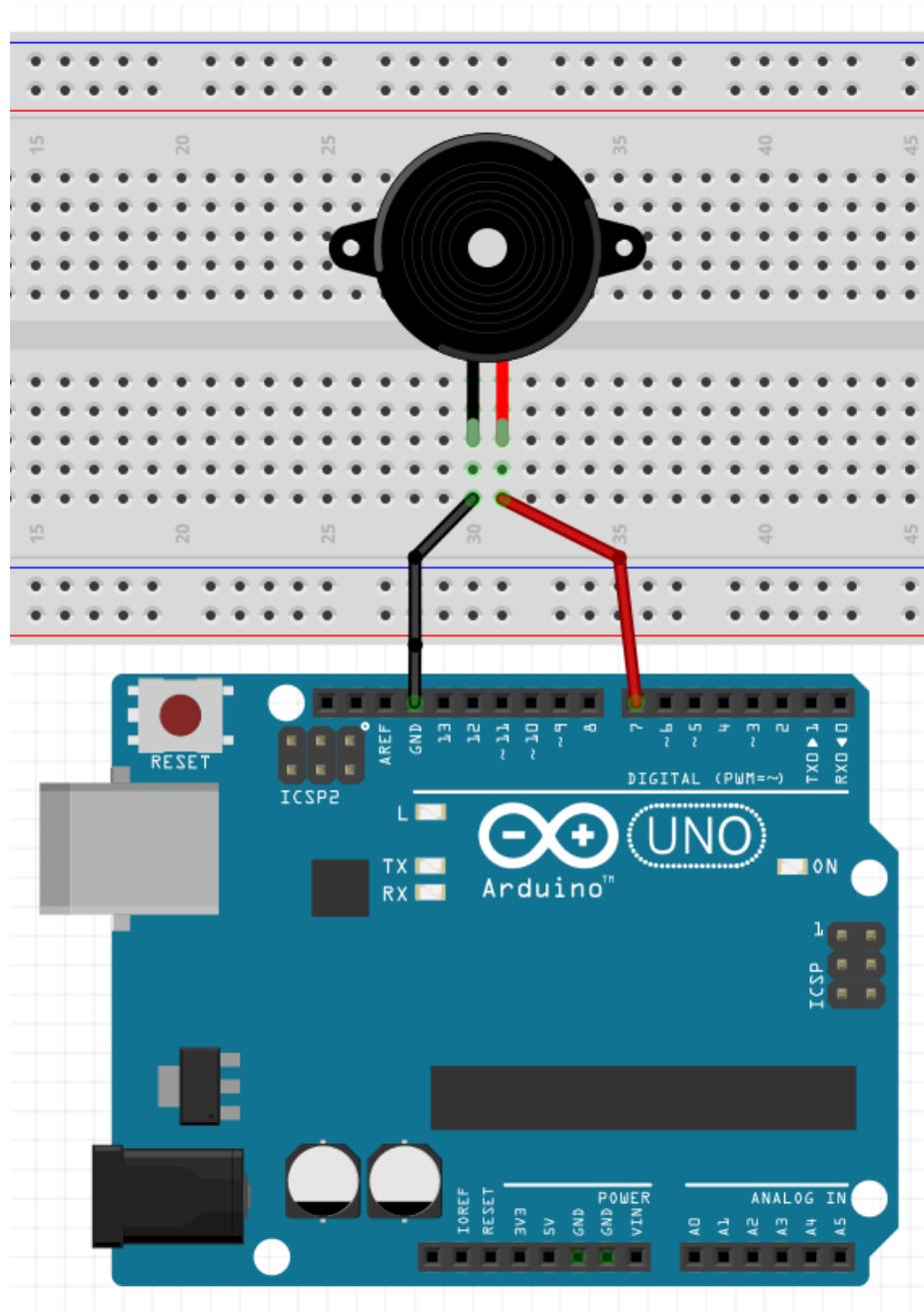
1. *passieve zoemers*: deze bevatten geen eigen oscillatie circuit en moeten dus gevoed worden met een wisselspanning om ze te doen werken. Je kan die herkennen aan de onderkant; je ziet nog het printplaatje met de pinnetjes.
2. Actieve zoemers : deze bevatten hun eigen oscillatie circuit en moeten enkel gevoed worden met een gelijkspanning om ze te doen werken. Je kan die herkennen aan de onderkant van plastic.



Schakeling



Breadboard schakeling



Sketch

We declareren een constante voor de poort waarop de positieve pin van de zoemer is aangesloten, alsook voor de lengte van de noot en de pause tussen de noten.

We declareren ook variabelen voor de lussen van de sirene en de noten.

In de setup functie wordt de hoofdlus 4 keer doorlopen. Binnen deze hoofdlus zijn 2 lussen die de de noten van laag naar hoog en van hoog naar laag afspelen met de instructies *tone (pin, frequentie, duur)* gevuld door een pauze met de instructie *delay*. Er is geen code in de loop functie.

```

/*
  Sirene

  Maak een sirene die 4 maal van laag naar hoog en terug naar laag gaat.

*/

// constanten
const int passieveZoemPin = 7;           // de pin waar de positieve pin van de zoemer is aangesloten
const int lengteNoot = 1000/8;            // lengte van de noot (msec)
const int pauseNoot = lengteNoot * 0.2;   // pauze tussen de noten (msec)

// variabelen
int aantalLoops = 0;                     // aantal loops voor de sirene
int noot = 0;                            // lus voor de noten

void setup() {
  // de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
  // wanneer op de reset knop gedrukt wordt

  // de sirene gaat 4 maal op en af
  for (aantalLoops = 0; aantalLoops < 4; aantalLoops++) {
    // speel de noten van laag naar hoog
    for (noot = 25; noot < 120; noot++) {
      // speel de noot
      tone(passieveZoemPin, 20 * noot, lengteNoot);
      // pauze voor volgende noot
      delay(pauseNoot);
    }

    // speel de noten van hoog naar laag
    for (noot = 120; noot >= 25; noot--) {
      // speel de noot
      tone(passieveZoemPin, 20 * noot, lengteNoot);
      // pauze voor de volgende noot
      delay(pauseNoot);
    }
  }
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
}

```

Project 10 : Eén cijfer 7 segment LED display

Opgave

Maak een schakeling met het één cijfer 7 segment LED display waarbij éénmalig :

- alle segmenten worden aan- en uitgezet.
- de segmenten afzonderlijk één voor één worden aan- en uitgezet.

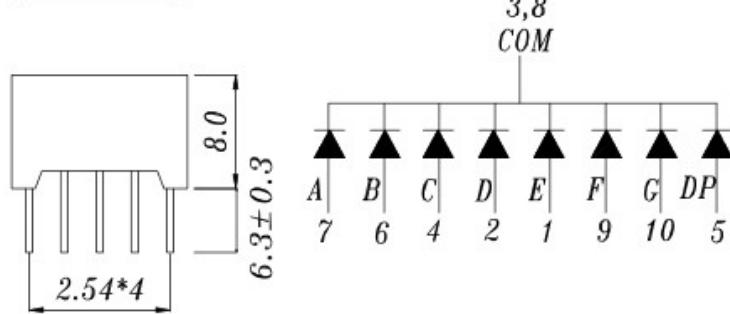
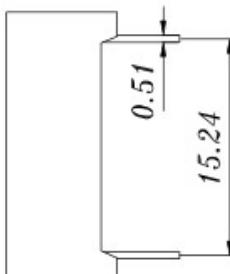
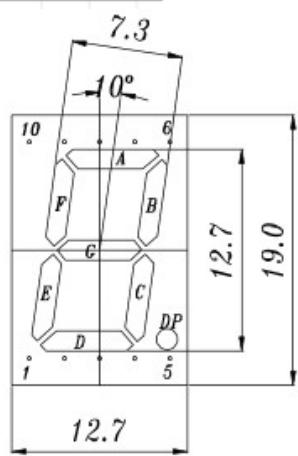
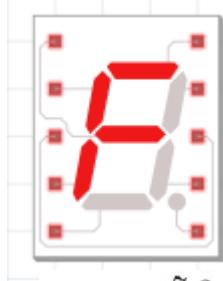
Vervolgens moet het één cijfer 7 segment LED display voortdurend optellen van 0 tot 9.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 één cijfer 7 segment LED display 5101AS
- 1 weerstand 220 ohm

Eén cijfer 7 segment LED display

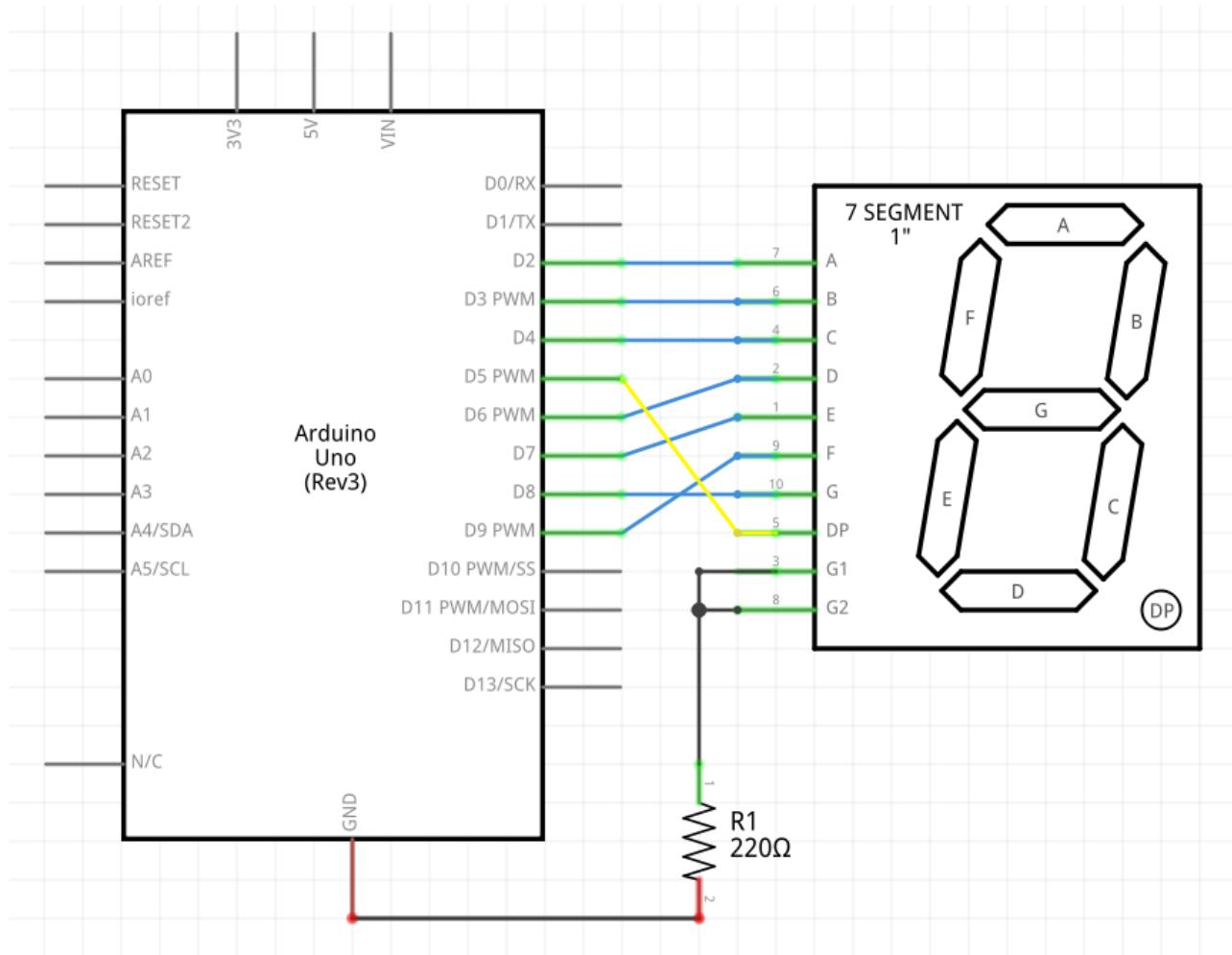
De 5101AS is één cijfer 7 segment LED display waarbij alle kathodes van de segmenten met elkaar verbonden zijn (Gemeenschappelijke kathode schakeling).



Dit is de pin lay-out van de 5101AS :

Pin	Functie
1	Segment E
2	Segment D
3	COM (GND)
4	Segment C
5	Decimaal punt
6	Segment B
7	Segment A
8	COM (GND)
9	Segment F
10	Segment G

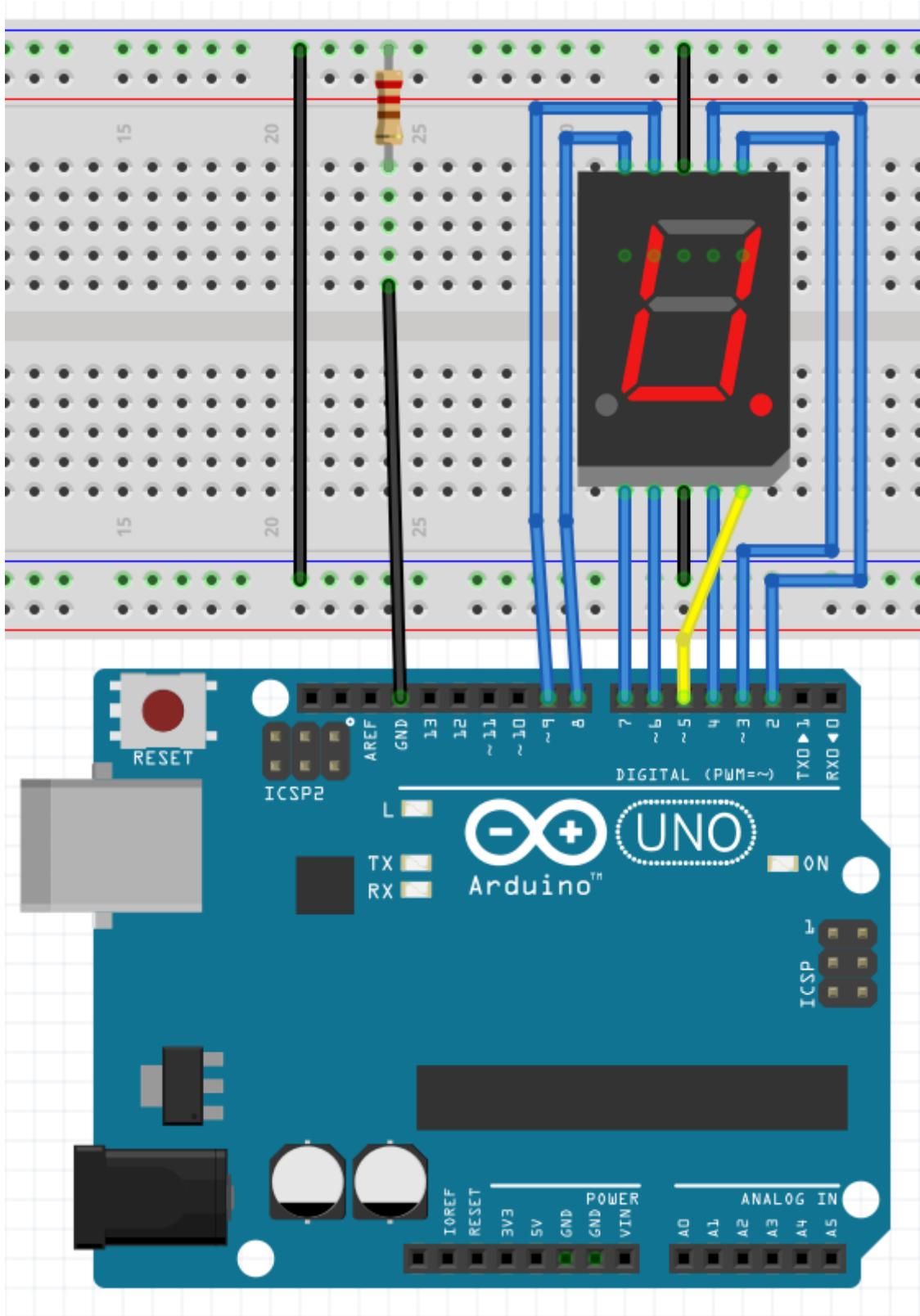
Schakeling



Breadboard schakeling

We sluiten het één cijfer LED display als volgt aan op de Arduino :

Pin 5101AS	Functie	Pin Arduino
1	Segment E	7
2	Segment D	6
3	COM	GND (via weerstand)
4	Segment C	4
5	Decimaal punt	5
6	Segment B	3
7	Segment A	2
8	COM	GND (via weerstand)
9	Segment F	9
10	Segment G	8



Sketch

We beginnen met het declareren van constanten voor de poorten van de segmenten van het één cijfer LED display, alsook de wachttijd tussen de updates.

In de setup initialiseren we de segment poorten als output. We roepen ook 2 functies op :

Stap	Wat	Functie
1	Alle segmenten tegelijk aan- en uit zetten	allesAanUit();
2	Segmenten één voor één aan- en uit zetten	segmentenAanUit();

```
/*
Het één cijfer 7 segment LED display

Maak een schakeling met het één cijfer 7 segment LED display waarbij éénmalig :
- alle segmenten samen worden aan- en uitgezet.
- de segmenten afzonderlijk één voor één worden aan- en uitgezet.
Vervolgens moet het één cijfer 7 segment LED display voortdurend optellen van 0 tot 9.

*/
// constanten
const int segA = 2;          // pin voor segment A van LED display
const int segB = 3;          // pin voor segment B van LED display
const int segC = 4;          // pin voor segment C van LED display
const int segD = 6;          // pin voor segment D van LED display
const int segE = 7;          // pin voor segment E van LED display
const int segF = 9;          // pin voor segment F van LED display
const int segG = 8;          // pin voor segment G van LED display
const int segPt = 5;         // pin voor decimaal punt van LED display

const int pauseTijd = 1000; // wachttijd tussen de updates

// variabelen

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {

    // initialisatie van de LED display poorten
    pinMode(segA, OUTPUT);
    pinMode(segB, OUTPUT);
    pinMode(segC, OUTPUT);
    pinMode(segD, OUTPUT);
    pinMode(segE, OUTPUT);
    pinMode(segF, OUTPUT);
    pinMode(segG, OUTPUT);
    pinMode(segPt, OUTPUT);

    // set alle segmenten tegelijk aan en uit
    allesAanUit();

    // set segmenten één voor één aan en uit
    segmentenAanUit();
}
```

In de loop functie tonen we de cijfers 0 tot 9 op het LED display. Hiervoor wordt gebruik gemaakt van een *for* lus en een *select* statement die de juiste functies (nul, een, twee, ..., negen) aanroeft die de cijfers vormt.

```
// de loop functie wordt steeds herhaald tot de st
void loop() {
    // lus om cijfers te tonen
    for (int digit = 0; digit < 10; digit++) {
        // select voor de cijfers
        switch(digit) {
            case 0:
                nul();
                break;
            case 1:
                een();
                break;
            case 2:
                twee();
                break;
            case 3:
                drie();
                break;
            case 4:
                vier();
                break;
            case 5:
                vijf();
                break;
            case 6:
                zes();
                break;
            case 7:
                zeven();
                break;
            case 8:
                acht();
                break;
            case 9:
                negen();
                break;
        }
        // wacht even zodat we iets kunnen zien
        delay(pauseTijd);
    }
}

// functie om cijfer 1 te tonen
void een() {
    digitalWrite(segA, LOW);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, LOW);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 2 te tonen
void twee() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, LOW);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, LOW);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 3 te tonen
void drie() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 4 te tonen
void vier() {
    digitalWrite(segA, LOW);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}
```

```

// functie om cijfer 5 te tonen
void vijf() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, LOW);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, LOW);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 6 te tonen
void zes() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, LOW);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 7 te tonen
void zeven() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, LOW);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 8 te tonen
void acht() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 9 te tonen
void negen() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, LOW);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 0 te tonen
void nul() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, LOW);
    digitalWrite(segPt, LOW);
}

// functie om alle segmenten tegelijk
void allesAanUit() {
    // alle segmenten aan
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, HIGH);

    // even wachten
    delay(pauseTijd);

    // alle segmenten uit
    digitalWrite(segA, LOW);
    digitalWrite(segB, LOW);
    digitalWrite(segC, LOW);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, LOW);
    digitalWrite(segPt, LOW);

    // even wachten
    delay(pauseTijd);
}

```

```
// functie om de segmenten één voor één aan en uit te zetten
void segmentenAanUit() {
    // segment A
    digitalWrite(segA, HIGH);
    delay(pauseTijd);
    digitalWrite(segA, LOW);
    delay(pauseTijd);
    // segment B
    digitalWrite(segB, HIGH);
    delay(pauseTijd);
    digitalWrite(segB, LOW);
    delay(pauseTijd);
    // segment C
    digitalWrite(segC, HIGH);
    delay(pauseTijd);
    digitalWrite(segC, LOW);
    delay(pauseTijd);
    // segment D
    digitalWrite(segD, HIGH);
    delay(pauseTijd);
    digitalWrite(segD, LOW);
    delay(pauseTijd);
    // segment E
    digitalWrite(segE, HIGH);
    delay(pauseTijd);
    digitalWrite(segE, LOW);
    delay(pauseTijd);
    // segment F
    digitalWrite(segF, HIGH);
    delay(pauseTijd);
    digitalWrite(segF, LOW);
    delay(pauseTijd);
    // segment G
    digitalWrite(segG, HIGH);
    delay(pauseTijd);
    digitalWrite(segG, LOW);
    delay(pauseTijd);
    // Decimaal punt
    digitalWrite(segPt, HIGH);
    delay(pauseTijd);
    digitalWrite(segPt, LOW);
    delay(pauseTijd);
}
```

Project 11 : Elektronische thermometer

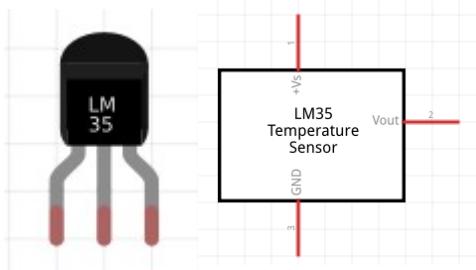
Opgave

Maak een thermometer die de temperatuur in °C toont op het vier cijfer 7 segment LED display. Gebruik hiervoor de temperatuur sensor LM35.

Benodigdheden

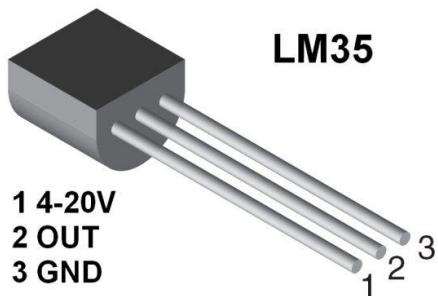
- Arduino UNO, breadboard, jumper kabeltjes
- 1 temperatuur sensor LM35
- 1 vier cijfer 7 segment LED display KYX-5461AS
- 4 weerstanden 220 ohm

Temperatuur sensor



Een temperatuur sensor is een component die de omgevingstemperatuur omzet in een variabele spanning. Het verband tussen temperatuur en spanning is **lineair**. Bij 0 °C is de spanning 0 V en bij iedere toename van 1 °C stijgt de spanning met 10 mV.

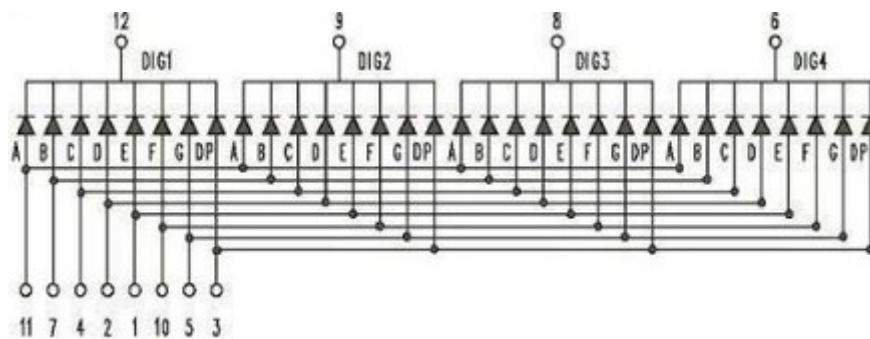
$$V_{\text{out}} (\text{mV}) = \text{temp} (\text{°C}) \times 10 (\text{mV/°C})$$



De component heeft 3 aansluitingen :

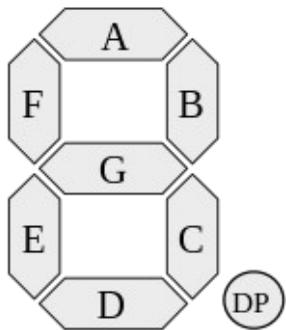
1. *anode* : sluiten we aan op 5 V
2. *out* : hier lezen we de variabele spanning af
3. *kathode* : sluiten we aan op massa (GND)

Vier cijfer 7 segment LED display



overeenkomstige segmenten van de afzonderlijke cijfers met elkaar verbonden.

De verschillende digits worden met korte tussenpozen één voor één aangestuurd om het juiste cijfer te tonen. Dit systeem heet **multiplexing**. Doordat die tussenpozen zo klein zijn lijkt het alsof alle digits hun cijfer tegelijk tonen. Dit is vergelijkbaar met een film of tv beeld dat eigenlijk ook een 30 tal stilstaande beeldjes per seconde toont, waardoor het voor ons lijkt alsof alles beweegt.

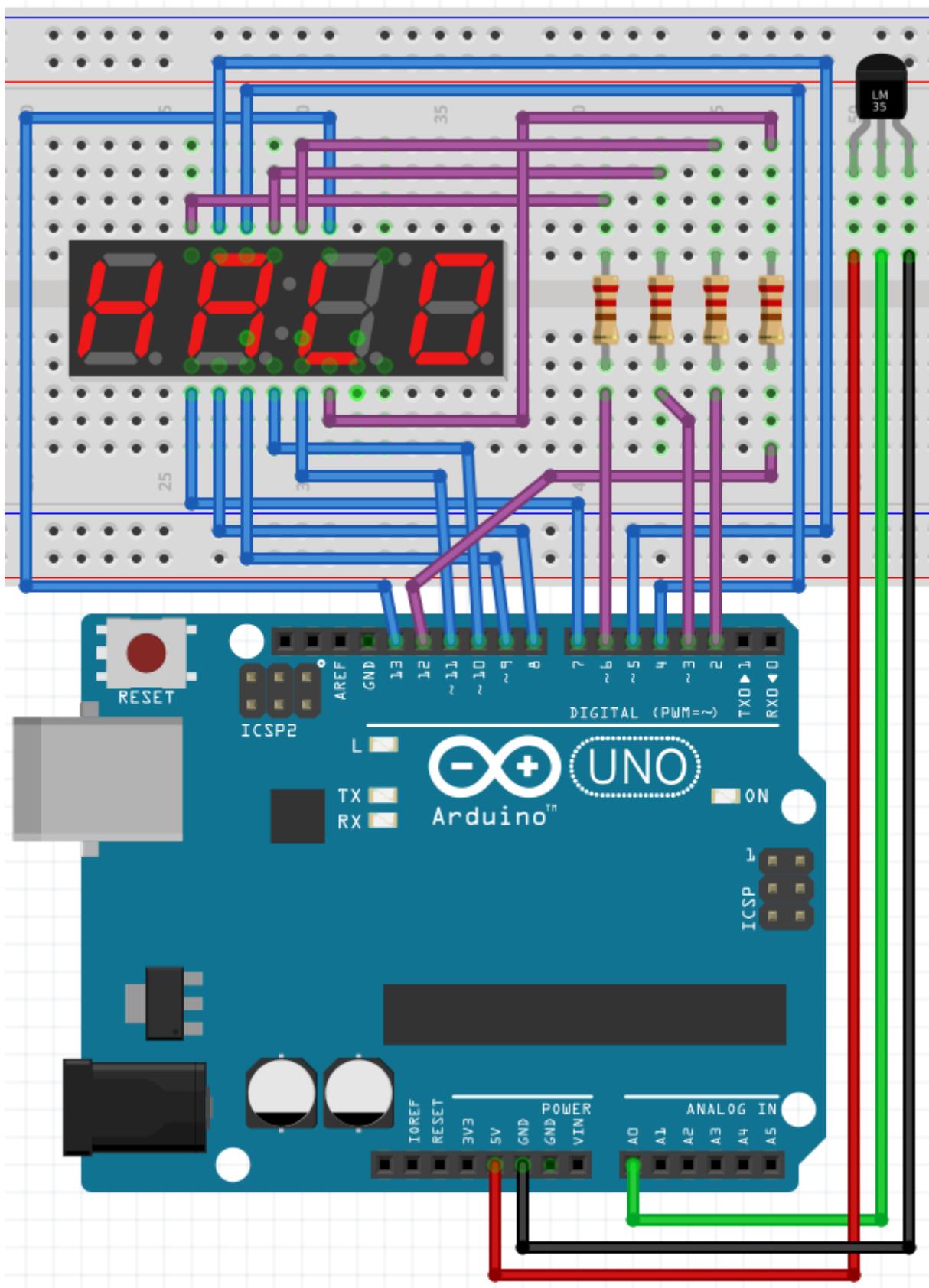


Dit is de pin lay-out van de KYX-5461AS :

1	Segment E
2	Segment D
3	Decimaal punt
4	Segment C
5	Segment G
6	Digit 4
7	Segment B
8	Digit 3
9	Digit 2
10	Segment F
11	Segment A
12	Digit 1



Breadboard schakeling



We sluiten de temperatuur sensor als volgt aan op de Arduino :

Pin LM35	Functie	Pin Arduino
1	anode	5 V
2	out	A0
3	kathode	GND

We sluiten het vier cijfer LED display als volgt aan op de Arduino :

Pin KYX-6461AS	Functie	Pin Arduino
1	Segment E	7
2	Segment D	8
3	Decimaal punt	9
4	Segment C	10
5	Segment G	11
6	Digit 4	12 (via weerstand)
7	Segment B	13
8	Digit 3	2 (via weerstand)
9	Digit 2	3 (via weerstand)
10	Segment F	4
11	Segment A	5
12	Digit 1	6 (via weerstand)

Sketch

We beginnen met het declareren van constanten voor de poorten van de temperatuur sensor en de digits en de segmenten van het vier cijfer LED display. We declareren ook de offset waarde voor de ijking van de thermometer. Dit is een waarde die we experimenteel moeten vaststellen. We definiëren ook enkele variabelen voor het uitlezen van de temperatuur sensor en het besturen van de loop.

In de setup functie initialiseren we de seriële console voor het debuggen van onze sketch. De poort van de temperatuur sensor wordt als input en deze van de vier cijfer LED display als output geïnitialiseerd.

```
// constanten
const int sensorPin = A0;      // pin van de temperatuur sensor
const float offset = 0.13;      // offset waarde voor temperatuur sensor

const int segA = 5;             // pin voor segment A van LED display
const int segB = 13;            // pin voor segment B van LED display
const int segC = 10;            // pin voor segment C van LED display
const int segD = 8;             // pin voor segment D van LED display
const int segE = 7;             // pin voor segment E van LED display
const int segF = 4;             // pin voor segment F van LED display
const int segG = 11;            // pin voor segment G van LED display
const int segPt = 9;            // pin voor decimaal punt van LED display
const int dig1 = 6;             // pin voor digit 1 van LED display
const int dig2 = 3;             // pin voor digit 2 van LED display
const int dig3 = 2;             // pin voor digit 3 van LED display
const int dig4 = 12;            // pin voor digit 4 van LED display

// variabelen
int pauseTijd = 900;           // wachttijd tussen de updates van de digits
int teller = 0;                // aantal loops
int sensorWaarde = 0;          // waarde van temperatuur sensor (0-1023)
float spanning = 0.0;           // spanning op temperatuur sensor (V)
float temperatuur = 0.0;        // temperatuur decimaal (°C)

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {

    // start seriële monitor voor debugging
    Serial.begin(9600);

    // initialisatie van de temperatuur sensor poort
    pinMode(sensorPin, INPUT);

    // initialisatie van de LED display poorten
    pinMode(segA, OUTPUT);
    pinMode(segB, OUTPUT);
    pinMode(segC, OUTPUT);
    pinMode(segD, OUTPUT);
    pinMode(segE, OUTPUT);
    pinMode(segF, OUTPUT);
    pinMode(segG, OUTPUT);
    pinMode(segPt, OUTPUT);
    pinMode(dig1, OUTPUT);
    pinMode(dig2, OUTPUT);
    pinMode(dig3, OUTPUT);
    pinMode(dig4, OUTPUT);
}
```

In de loop functie lezen we pas een nieuwe temperatuur waarde in om de 500 cyclussen. We gebruiken *analogRead* om de waarde van de poort A0 uit te lezen. Dat geeft ons een waarde tussen 0 en 1023. Vervolgens zetten we deze waarde om naar spanning met deze formule :

$$\text{spanning} = (\text{sensorWaarde} / 1024.0) * 5.0$$

Dat geeft ons de waarde in Volt. Nu moeten we deze spanning nog omzetten naar de temperatuur in °C. Dat doen we met deze formule :

$$\text{temperatuur} = (\text{spanning} - \text{offset}) * 100.0$$

De waarde offset moet experimenteel bepaald worden. Begin met offset = 0 en bekijk hoeveel de temperatuur afwijkt van de juiste. Waarschijnlijk zal de berekende temperatuur te hoog zijn. Pas nu offset aan tot de temperatuur juist is. Gebruik hiervoor de seriële console.

```
// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

    // lees de temperatuur maar om de 500 loops
    if (teller%500 == 0) {

        // lees de temperatuur sensor
        sensorWaarde = analogRead(sensorPin);

        // converteer de sensorwaarde (0-1023) naar spanning
        spanning = (sensorWaarde / 1024.0) * 5.0;

        // bereken de temperatuur - iedere 10 mV = 1 °C
        temperatuur = (spanning - offset) * 100.0;

        // toon waarden in seriële monitor voor debugging
        Serial.print("Sensorwaarde: ");
        Serial.print(sensorWaarde);
        Serial.print("\t Spanning: ");
        Serial.print(spanning);
        Serial.print("\t Temperatuur : ");
        Serial.print(temperatuur);
        Serial.println(" C");

        // reset teller
        teller = 0;
    }
}
```

De digits van het vier cijfer LED display worden als volgt opgevuld :

Digit	Waarde	Voorbeeld 21,6 °C
1	Tiental(temperatuur)	2
2	Eenheid(temperatuur) + decimaal punt	.
3	Decimaal(temperatuur)	6
4	Letter "C"	C

Het updaten van het vier cijfer LED display met de juiste temperatuur waarde gebeurt via een systeem dat **multiplexing** heet. Iedere digit wordt één na één upgedate met een korte tussenpauze. Ons oog ziet die korte onderbrekingen niet en voor ons lijkt het alsof het display altijd onmiddellijk wordt bijgewerkt bij iedere temperatuursverandering.

Iedere digit wordt als volgt bijgewerkt :

Stap	Wat	Functie
1	Selecteer de digit d (1-4)	selectDigit(d);
2	Stuur waarde x naar digit	stuurDigit(x);
3	Pauzeer gedurende pauseTijd (msec)	delayMicroseconds(pauseTijd);
4	Toon de digit d (1-4)	digitalWrite(d, HIGH);

```
// we sturen de digits van het LED display één voor één aan met de juiste informatie

// digit 1 bevat het tiental van de temperatuur waarde
selectDigit(1); // selecteer digit 1
stuurDigit(tiental(temperatuur)); // stuur tiental van temperatuur door
delayMicroseconds(pauseTijd); // wacht even
digitalWrite(dig1, HIGH); // toon digit 1

// digit 2 bevat de eenheden van de temperatuur waarde
selectDigit(2); // selecteer digit 2
stuurDigit(eenheid(temperatuur)); // stuur eenheid van temperatuur door
decimaalpunt(); // stuur decimale punt door
delayMicroseconds(pauseTijd); // wacht even
digitalWrite(dig2, HIGH); // toon digit 2

// digit 3 bevat de decimalen van de temperatuur waarde
selectDigit(3); // selecteer digit 3
stuurDigit(decimaal(temperatuur)); // stuur decimaal van temperatuur door
delayMicroseconds(pauseTijd); // wacht even
digitalWrite(dig3, HIGH); // toon digit 3

// digit 4 toont de hoofdletter C voor Celsius
selectDigit(4); // selecteer digit 4
cee(); // stuur de C door
delayMicroseconds(pauseTijd); // wacht even
digitalWrite(dig4, HIGH); // toon digit 4

// teller voor de loops verhogen
teller++;

}
```

In de loop functie worden verschillende functies gebruikt die verderop in de sketch staan : een, twee, drie, vier, vijf, zes, zeven, acht, negen, nul, decimaalpunt, cee, selectDigit, stuurDigit, tiental, eenheid en decimaal.

```

// functie om cijfer 1 te tonen
void een() {
    digitalWrite(segA, LOW);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, LOW);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 2 te tonen
void twee() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, LOW);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, LOW);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 3 te tonen
void drie() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 4 te tonen
void vier() {
    digitalWrite(segA, LOW);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 5 te tonen
void vijf() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, LOW);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, LOW);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 6 te tonen
void zes() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, LOW);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 7 te tonen
void zeven() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, LOW);
    digitalWrite(segE, LOW);
    digitalWrite(segF, LOW);
    digitalWrite(segG, LOW);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 8 te tonen
void acht() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

```

```

// functie om cijfer 9 te tonen
void negen() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, LOW);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, HIGH);
    digitalWrite(segPt, LOW);
}

// functie om cijfer 0 te tonen
void nul() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, HIGH);
    digitalWrite(segC, HIGH);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, LOW);
    digitalWrite(segPt, LOW);
}

// functie om letter C te tonen
void cee() {
    digitalWrite(segA, HIGH);
    digitalWrite(segB, LOW);
    digitalWrite(segC, LOW);
    digitalWrite(segD, HIGH);
    digitalWrite(segE, HIGH);
    digitalWrite(segF, HIGH);
    digitalWrite(segG, LOW);
    digitalWrite(segPt, LOW);
}

// functie om decimaal punt te tonen
void decimaalpunt() {
    digitalWrite(segPt, HIGH);
}

// functie om het teken x naar de juiste digit (1-4) te sturen
void stuurDigit(int x) {
    switch(x) {
        case 1:
            een();
            break;
        case 2:
            twee();
            break;
        case 3:
            drie();
            break;
        case 4:
            vier();
            break;
        case 5:
            vijf();
            break;
        case 6:
            zes();
            break;
        case 7:
            zeven();
            break;
        case 8:
            acht();
            break;
        case 9:
            negen();
            break;
        case 10:
            cee();
            break;
        default:
            nul();
            break;
    }
}

```

```

// functie om de juiste digit (1-4) te selecteren
void selectDigit(int d) {
    // we zetten eerst de digit laag voordat we die updaten
    switch (d) {
        case 1:
            digitalWrite(dig1, LOW);
            break;
        case 2:
            digitalWrite(dig2, LOW);
            break;
        case 3:
            digitalWrite(dig3, LOW);
            break;
        default:
            digitalWrite(dig4, LOW);
            break;
    }
}

// functie om het tiental van een decimaal getal x te berekenen
int tiental(float x) {
    float divided = x / 10.0;
    return (int)divided;
}

// functie om de eenheid van een decimaal getal x te berekenen
int eenheid(float x) {
    float divided = x - (10.0 * tiental(x));
    return (int)divided;
}

// functie om de decimaal van een decimaal getal x te berekenen
int decimaal(float x) {
    float divided = x - (10.0 * tiental(x)) - eenheid(x);
    divided *= 10;
    return (int)divided;
}

```

Project 12 : Afstandsdetectie

Opgave

Experimenteer met de HC-SR04 afstandsdetectie sensor en toon de resultaten in de seriële console.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 HC-SR04 afstand sensor

HC-SR04 afstandsdetectie sensor



De HC-SR04 werkt zoals de sonar van een duikboot.
De sensor heeft een bereik van 2 tot 400 cm

De Trigger pin wordt gedurende minstens $10 \mu\text{s}$ hoog gezet. Hierdoor worden 8 ultrasone geluidsignalen van 40 kHz uitgezonden vanaf de linker sensor (T van Transmit) die werkt als een mini luidspreker. De geluidsignalen worden door obstakels gedeeltelijk teruggekaatst en opgevangen door de rechter sensor (R van Recieve). Hierdoor wordt de Echo pin hoog gezet.

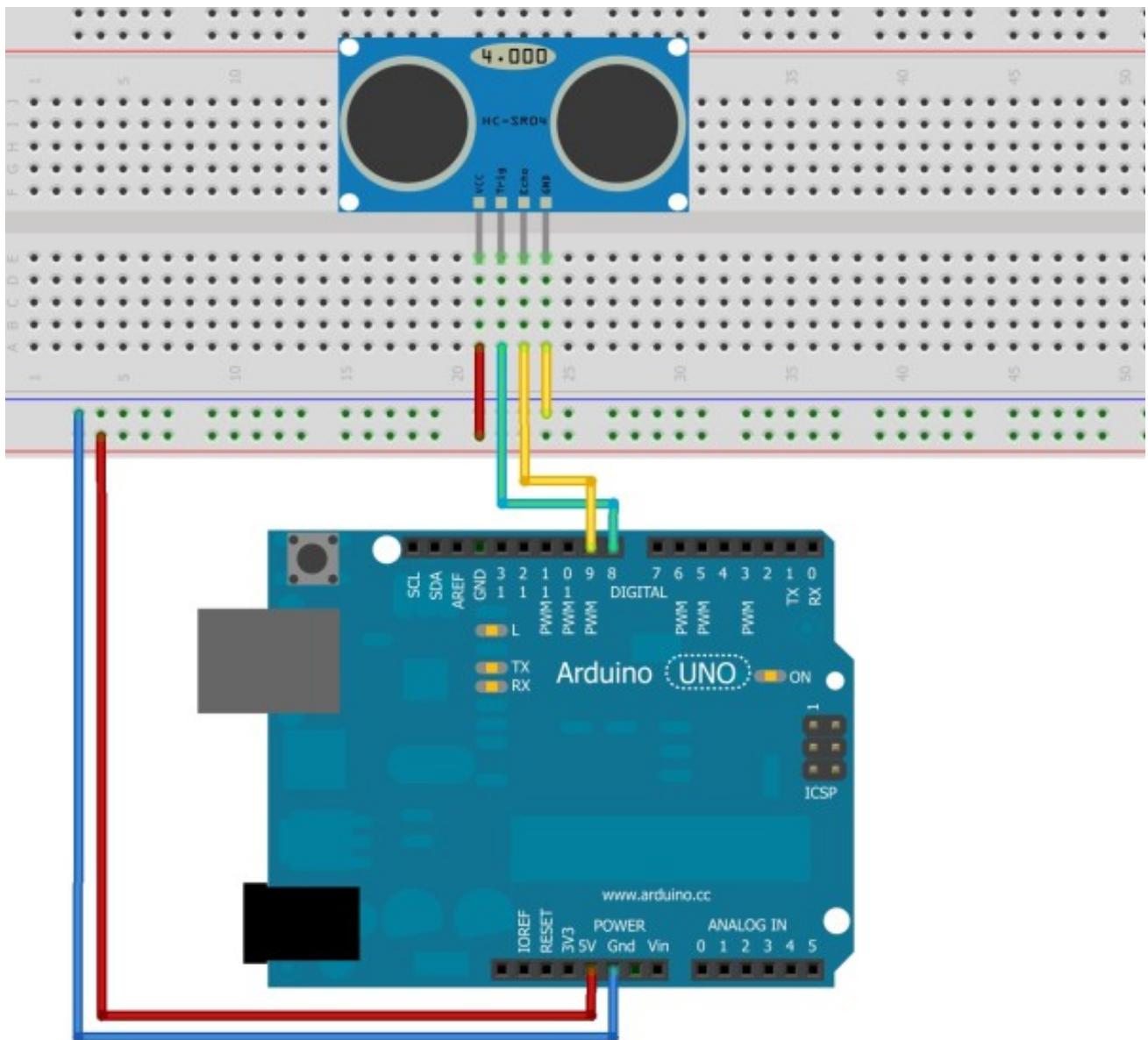
Geluidsignalen hebben een snelheid van 340 m/s of omgezet $29,1 \text{ cm}/\mu\text{s}$. De tijd van de teruggekaatste geluidsignalen is equivalent met $2 \times$ de afstand tot het obstakel. De formule om de afstand te berekenen wordt dus :

$$\text{Afstand (in cm)} = \text{Echo tijd (in } \mu\text{s}) / 29,1 \text{ (in cm}/\mu\text{s}) / 2$$

HC-SR04 aansluitingen op de UNO

Pin HC-SR04	Functie	Pin UNO
VCC	Voeding +	+5V
Trigger	Output	8
Echo	Input	9
GND	Voeding -	GND

Breadboard schakeling



Sketch

We verbinden de afstand sensor met de UNO volgens de tabel op vorige pagina.

We declareren enkele constanten voor de pinnen van de afstand sensor en enkele variabelen voor tijd en afstand.

In de setup functie initialiseren we de seriële monitor en de pinnen van de afstand sensor.

In de loop functie sturen we de ultrasone geluidsignalen uit met de trigger sequentie. We bepalen de tijd met het commando pulseIn. Als de echo pin hoog is wordt de tijd variabele opgevuld. Via de formule wordt deze tijd omgezet naar afstand die we vervolgens in de seriële monitor afbeelden. Er volgt dan nog een wachttijd van $\frac{1}{2}$ seconde voor de volgende loop.

```

/*
Afstandssensor

Experimenteer met de HC-SR04 afstand sensor en toon de resultaten in de seriële console.

*/
// constanten
const int TriggerPin = 8;
const int EchoPin = 9;

// variabelen
int tijd, afstand;

void setup() {
    // de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
    // wanneer op de reset knop gedrukt wordt

    // Initialiseer de seriële monitor
    Serial.begin(9600);

    // Declaratie afstand sensor pinnen
    pinMode(TriggerPin, OUTPUT);
    pinMode(EchoPin, INPUT);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    // Stuur de geluidsignalen uit
    digitalWrite(TriggerPin, HIGH);      // Start Trigger
    delayMicroseconds(1000);           // wacht 1 ms
    digitalWrite(TriggerPin, LOW);      // Stop Trigger

    // bereken de echo tijd
    tijd = pulseIn(EchoPin, HIGH);     // als de EchoPin hoog is wordt de tijd opgeslagen in de variabele

    // bereken de afstand
    afstand = (tijd/2) / 29.1;        // zie formule

    // toon resultaten in seriële monitor
    if (afstand >= 200 || afstand <= 0) {
        // ongeldige afstand
        Serial.println("Ongeldig bereik");
    } else {
        // geldige afstand
        Serial.print("Afstand: ");
        Serial.print(afstand);
        Serial.println(" cm");
    }

    // Even wachten
    delay(500);
}

```



NewPing

NewPing is een objectgeoriënteerde bibliotheek met handige methodes (functies) geschreven door Tim Eckel. Deze kan gebruikt worden bij verschillende types van afstand sensoren, onder andere ook de HC-SR04.

Installatie

Surf naar de [homepage van NewPing](#) voor de uitleg en download de bibliotheek via de link op deze pagina. Pak het zip bestand uit en kopieer de NewPing folder naar de folder libraries van je Arduino installatie.

Declaratie

In de Arduino IDE kies je *Schets – Bibliotheek importeren – NewPing*. Er wordt automatisch een #include ingevoegd : `#include <NewPing.h>`

Met de instructie `NewPing sonar(TriggerPin, EchoPin, MaximumBereik);` wordt een nieuwe instance sonar aangemaakt met als parameters : trigger pin, echo pin en maximum bereik in cm van de afstand sensor.

Methodes

Methode	Uitleg
<code>sonar.ping();</code>	Stuur een ping en geef de tijd terug in μ s. Geef 0 als er geen echo is binnen het max afstandsbereik.
<code>sonar.ping_in();</code>	Stuur een ping en geef de afstand terug in inches. Geef 0 als er geen echo is binnen het max afstandsbereik.
<code>sonar.ping_cm();</code>	Stuur een ping en geef de afstand terug in cm. Geef 0 als er geen echo is binnen het max afstandsbereik.
<code>sonar.ping_median(iterations);</code>	Stuur iterations pings (standaard 5) en geef de gemiddelde tijd terug in μ s. 0 als er geen echo's zijn binnen het afstandsbereik.
<code>sonar.convert_in(echoTime);</code>	Zet de tijd echoTime (in μ s) om naar afstand (in inches).
<code>sonar.convert_cm(echoTime);</code>	Zet de tijd echoTime (in μ s) om naar afstand (in cm).
<code>sonar.ping_timer(function);</code>	Stuur een ping en roep de functie function op om te controleren of de ping compleet is.
<code>sonar.check_timer();</code>	Controleer als een echo is ontvangen binnen het afstandsbereik.
<code>timer_us(frequency, function);</code>	Roep functie function op iedere frequency μ s.
<code>timer_ms(frequency, function);</code>	Roep functie function op iedere frequency ms.
<code>timer_stop();</code>	Stop de timer.

Sketch met NewPing

```
#include <NewPing.h>
/*
   Afstandssensor

   Experimenteer met de HC-SR04 afstand sensor en toon de resultaten in de seriële console.
*/

// constanten
const int TriggerPin = 8;
const int EchoPin = 9;
const int MaxBereik = 200;      // in cm

// NewPing instance
NewPing sonar(TriggerPin, EchoPin, MaxBereik);

// variabelen
int tijd, afstand;

void setup() {
    // de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
    // wanneer op de reset knop gedrukt wordt

    // Initialiseer de seriële monitor
    Serial.begin(9600);

    // Declaratie afstand sensor pinnen
    pinMode(TriggerPin, OUTPUT);
    pinMode(EchoPin, INPUT);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    // Stuur de ping uit en geeft de tijd terug
    tijd = sonar.ping();

    // bereken de afstand in cm
    afstand = sonar.convert_cm(tijd);

    // toon resultaten in seriële monitor
    if (afstand >= MaxBereik || afstand <= 0) {
        // ongeldige afstand
        Serial.println("Ongeldig bereik");
    } else {
        // geldige afstand
        Serial.print("Afstand: ");
        Serial.print(afstand);
        Serial.println(" cm");
    }

    // Even wachten
    delay(500);
}
```

Project 13 : 8x8 Dot LED Matrix

Opgave

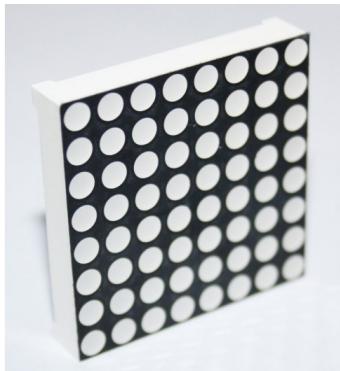
Laat op de 8x8 Dot LED matrix :

- alle leds, de afzonderlijke leds, de rij leds en de kolom leds één voor één oplichten.
- de letters van “HALLO” gevuld door een smiley :-) van rechts naar links scrollen

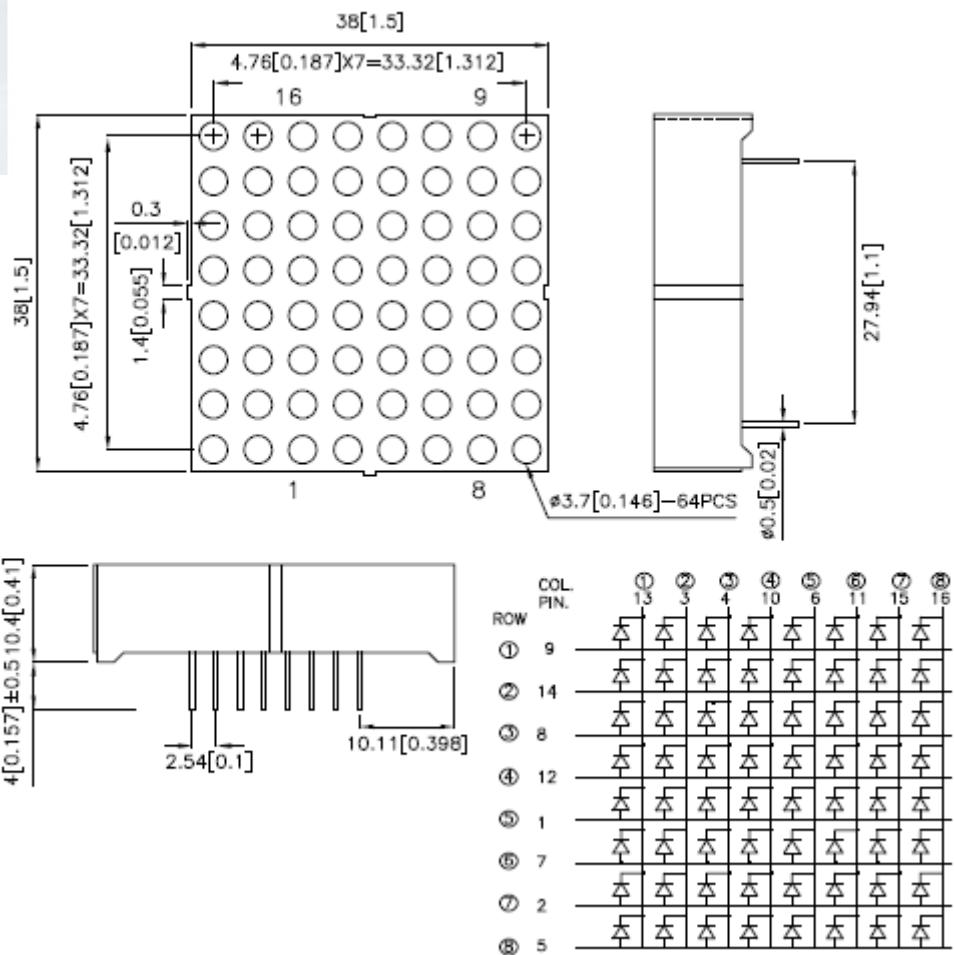
Benodigdheden

- 8 weerstanden 220 ohmArduino UNO, breadboard, jumper kabeltjes
- 1 8x8 Dot LED Matrix 1588BS
- 8 weerstanden 220 ohmArduino UNO, breadboard, jumper kabeltjes

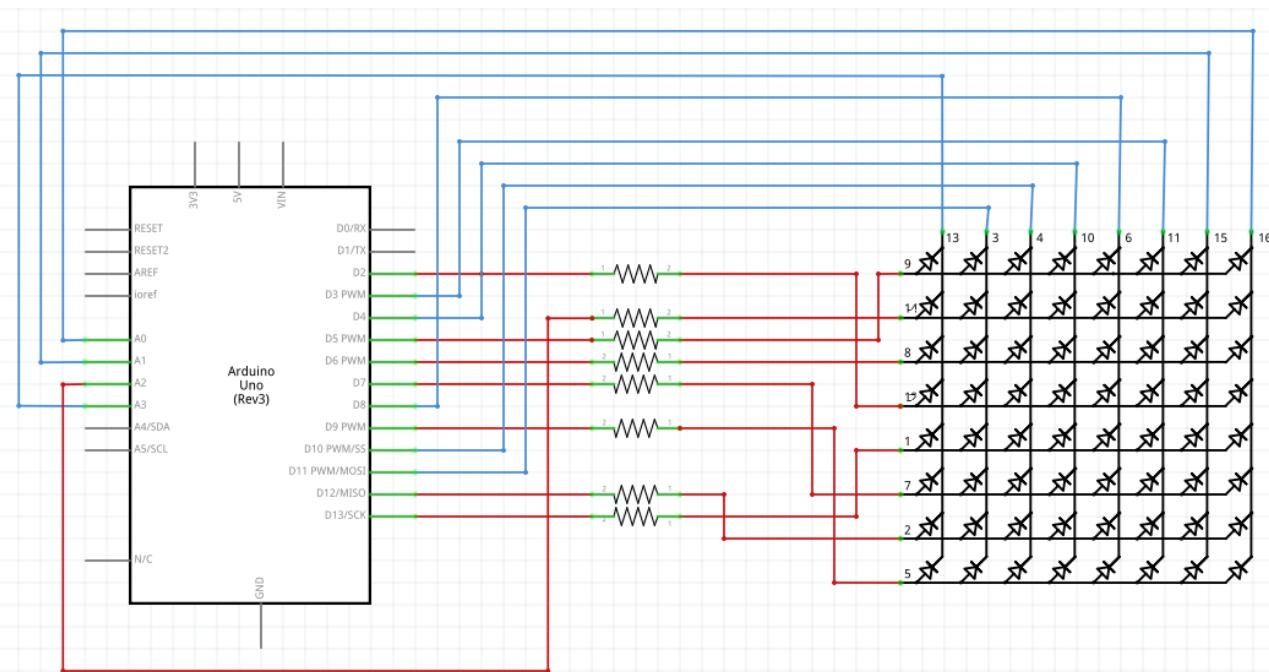
8x8 Dot LED Matrix 1588BS



De 1588BS is een 8x8 LED matrix die samengesteld is uit 64 afzonderlijk aanstuurbare LEDs. Hierbij zijn de anodes van de LEDs in dezelfde rij met elkaar verbonden. De kathodes van de LEDs in dezelfde kolom zijn eveneens met elkaar verbonden.



Schakeling

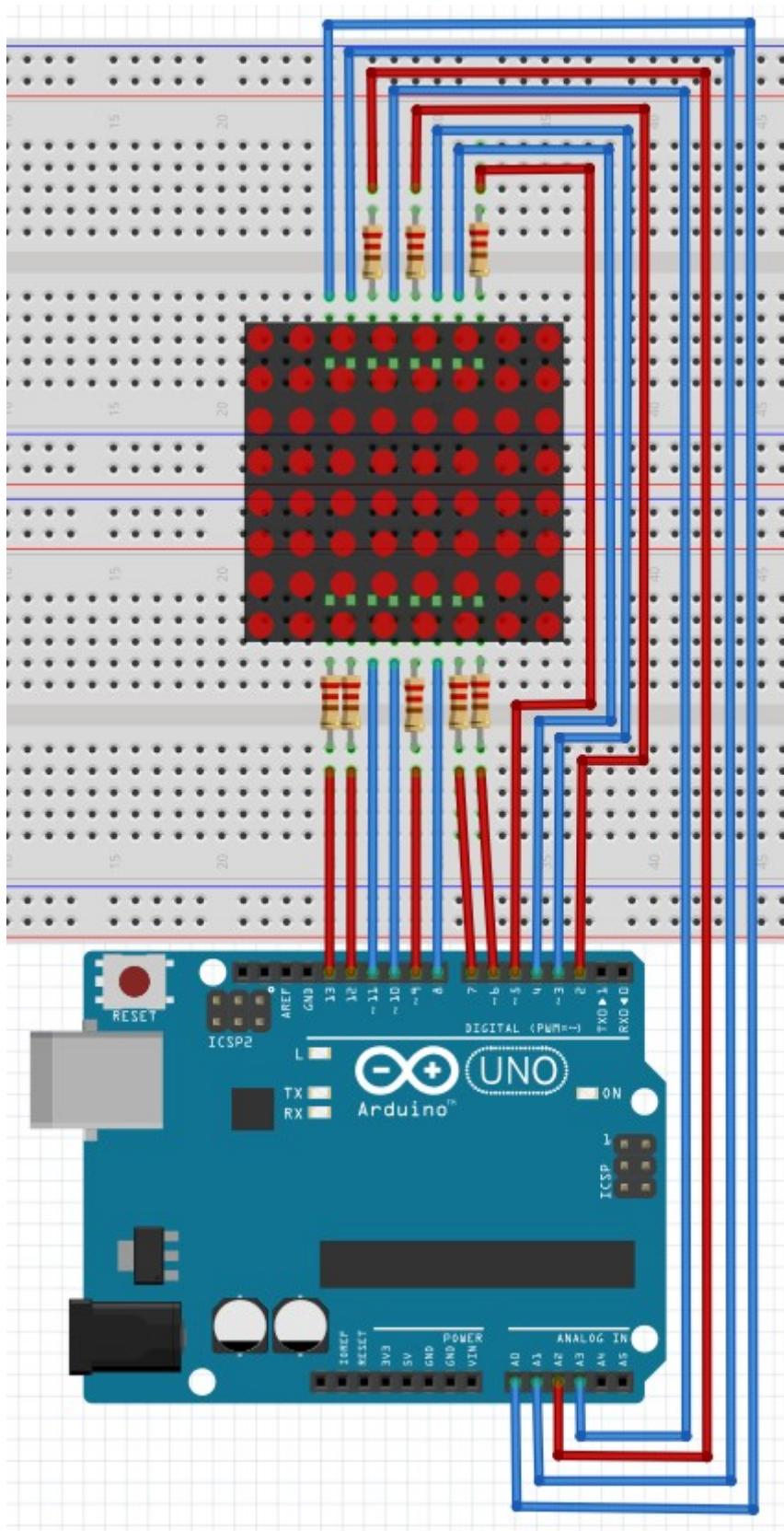


We verbinden de 1588BS LED matrix met de UNO volgens onderstaand schema. De verbindingen voor de rij LEDs doen we via een weerstand.

1588BS pin	LED rij	LED kolom	UNO pin
1	5		13 (via weerstand)
2	7		12 (via weerstand)
3		2	11
4		3	10
5	8		9 (via weerstand)
6		5	8
7	6		7 (via weerstand)
8	3		6 (via weerstand)
9	1		5 (via weerstand)
10		4	4
11		6	3
12	4		2 (via weerstand)
13		1	17*
14	2		16* (via weerstand)
15		7	15*
16		8	14*

* De analoge poorten A0 tot A5 zijn op de Arduino UNO ook gekend als 14 tot 19.

Breadboard schakeling



Sketch 1

We beginnen met het declareren van 2 reeksen voor de pinnen die de rijen en de kolommen van de LED matrix aansturen. Een reeks is een verzameling van variabelen waarvan de waarden kunnen opgevraagd worden via een index.

De declaratie heeft de volgende vorm :

```
reeks[aantal_elementen] = { element1, element2, ... };
```

De index van een reeks begint steeds bij index 0. Als je bvb een reeks declareert met 8 waarden, dan moet je de 1ste waarde opvragen als reeks[0], de 3de waarde als reeks[2], enz...

Als je de pinnen voor de rijen en kolommen in een reeks stekt, dan kan je die makkelijk manipuleren via een *for* statement.

In de setup functie declareren we de rij en kolom pinnen als output. We initialiseren tevens de waarde voor de pinnen. Merk hierbij op dat de rij pinnen laag moeten staan; de anodes van de verschillende leds zijn immers per rij met elkaar verbonden ! Analoog moeten de kolom pinnen hoog omdat de kathodes van de leds per kolom met elkaar verbonden zijn.

```
/*
Laat op de 8x8 Dot LED matrix de afzonderlijke LEDs één voor één oplichten,
alsook alle rijen en kolommen.

*/
// constanten
const int rij[8] = { 5, 16, 6, 2, 13, 7, 12, 9 }; // pinnen die de rijen (anodes) sturen
const int kol[8] = { 17, 11, 10, 4, 8, 3, 15, 14 }; // pinnen die de kolommen (kathodes) sturen
const boolean Aan = true; // matrix aan
const boolean Uit = false; // matrix uit
const int tijd = 300; // wachttijd

// variabelen
int r = 0; // loop variabele voor reeks rij
int k = 0; // loop variabele voor reeks kol

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup(){

    // declaratie pinnen
    for (int i=0; i<8; i++) {
        pinMode(rij[i], OUTPUT); // rij pinnen als output
        pinMode(kol[i], OUTPUT); // kolom pinnen als output
    }

    // Matrix uit
    InitMatrix(Uit);
}
```

In de loop functie gebruiken we intensief *for* lussen om leds aan en uit te zetten. We maken ook nog een functie *InitMatrix* met een boolean parameter. Als die waar is (true), dan worden alle leds van de matrix aangezet; in het ander geval (false) worden alle leds uitgezet. We gebruiken hiervoor de 2 boolean constanten Aan en Uit.

```

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

    // Alle leds aan- en uitzetten
    InitMatrix(Aan);                                // alles aan
    delay(tijd);                                    // wacht tijd

    InitMatrix(Uit);                                // alles uit
    delay(tijd);                                    // wacht tijd

    // Alle leds één voor één aan- en uitzetten
    for (r=0; r<8; r++) {                          // voorwaarts
        for (k=0; k<8; k++) {
            digitalWrite(rij[r], HIGH);
            digitalWrite(kol[k], LOW);
            delay(tijd);
            digitalWrite(rij[r], LOW);
            digitalWrite(kol[k], HIGH);
        }
    }
    delay(tijd);                                    // wacht tijd

    for (r=7; r>=0; r--) {                          // achterwaarts
        for (k=7; k>=0; k--) {
            digitalWrite(rij[r], HIGH);
            digitalWrite(kol[k], LOW);
            delay(tijd);
            digitalWrite(rij[r], LOW);
            digitalWrite(kol[k], HIGH);
        }
    }
    delay(tijd);                                    // wacht tijd

    // Alle rijen één voor één aan- en uitzetten
    for (k=0; k<8; k++) {                          // alle kol pinnen uit
        digitalWrite(kol[k], LOW);
    }
    for (r=0; r<8; r++) {                          // voorwaarts
        digitalWrite(rij[r], HIGH);
        delay(tijd);
        digitalWrite(rij[r], LOW);
    }
    delay(tijd);                                    // wacht tijd

    for (r=7; r>=0; r--) {                          // achterwaarts
        digitalWrite(rij[r], HIGH);
        delay(tijd);
        digitalWrite(rij[r], LOW);
    }
    for (k=0; k<8; k++) {                          // alle kol pinnen aan
        digitalWrite(kol[k], HIGH);
    }
    delay(tijd);                                    // wacht tijd
}

```

```

// Alle kolommen één voor één aan- en uitzetten
for (r=0; r<8; r++) {
    digitalWrite(rij[r], HIGH);
}
for (k=0; k<8; k++) {
    digitalWrite(kol[k], LOW);
    delay(tijd);
    digitalWrite(kol[k], HIGH);
}
delay(tijd);

for (k=7; k>=0; k--) {
    digitalWrite(kol[k], LOW);
    delay(tijd);
    digitalWrite(kol[k], HIGH);
}
for (r=0; r<8; r++) {
    digitalWrite(rij[r], LOW);
}
delay(tijd);

// initialisatie van de LED matrix
void InitMatrix(boolean Aan) {
    if (Aan) {
        for (int i=0; i<8; i++) {
            digitalWrite(rij[i], HIGH);
            digitalWrite(kol[i], LOW);
        }
    } else {
        for (int i=0; i<8; i++) {
            digitalWrite(rij[i], LOW);
            digitalWrite(kol[i], HIGH);
        }
    }
}

```

Sketch 2

Voor het afbeelden van de scrollende tekst op de LED matrix gebruiken we de *FrequencyTimer2* bibliotheek. Met deze bibliotheek kan je makkelijk om de zoveel μ sec een bepaalde functie laten lopen. We gebruiken deze truuk voor onze scrollende tekst.

Voor meer uitleg, download, installatie instructie en voorbeeld code surf je naar [deze pagina](#).

De verschillende letters worden gedefinieerd in een 2 dimensionale reeks. Ze bevat 8 elementen die de rijen van de leds van de matrix voorstellen. Ieder element is op zijn beurt een reeks die alle individuele leds van de rij voorstellen. Als de waarde 0 is moet de led uit zijn, als de waarde 1 is moet de led aan zijn.

```

/*
Laat op de 8x8 Dot LED matrix de letters van "HALLO" van rechts naar links scrollen
gevolgd door een smiley :-)

*/
#include <FrequencyTimer2.h>

#define H { \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0} \
}

#define A { \
    {0, 0, 0, 1, 1, 0, 0, 0}, \
    {0, 0, 1, 0, 0, 1, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0} \
}

#define L { \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0} \
}

#define O { \
    {0, 0, 0, 1, 1, 0, 0, 0}, \
    {0, 0, 1, 0, 0, 1, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 0, 1, 0, 0, 1, 0, 0}, \
    {0, 0, 0, 1, 1, 0, 0, 0} \
}

```

We voegen de bibliotheek toe via het menu Schets – Bibliotheek importeren – FrequencyTimer2. Hierdoor wordt het juiste #include statement toegevoegd.

We declareren zoals in de vorige sketch een aantal constanten voor de rij- en kolompinnen, alsook voor de patronen voor de letters en de smiley. Merk op dat dit een 3 dimensionale reeks is : 6 patronen van 8 rijen en 8 kolommen.

We declareren eveneens een aantal variabelen voor manipulatie van de verschillende reeksen, alsook een led variabele die steeds 1 patroon zal bevatten (scroll manipulatie !)

In de setup functie declareren we de rij en kolom pinnen als output. We initialiseren tevens de waarde voor de pinnen. We initialiseren de FrequencyTimer2 methodes : interval 2000 μ s en interrupt routine is onze eigen *TekenMatrix* functie. Daarna wordt het eerste patroon opgeladen met de functie *LaadPatroon(p)*.

```
// constanten
const int rij[8] = { 5, 16, 6, 2, 13, 7, 12, 9 };           // pinnen die de rijen (anodes) sturen
const int kol[8] = { 17, 11, 10, 4, 8, 3, 15, 14 };         // pinnen die de kolommen (kathodes) sturen
const int AantalPatronen = 6;                                // aantal patronen
const byte pat[AantalPatronen][8][8] = \
{ H, A, L, L, O, Smiley };                                    // af te beelden patronen
const boolean Aan = true;                                     // led matrix aan
const boolean Uit = false;                                    // led matrix uit

// variabelen
int r = 0;                                                 // loop variabele voor reeks rij
int k = 0;                                                 // loop variabele voor reeks kol
int p = 0;                                                 // loop variabele voor reeks pat
byte led[8][8];                                           // led matrix reeks
byte row = 0;                                              // bijhouden rij voor interrupt routine

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup(){
    // declaratie pinnen
    for (int i=0; i<8; i++) {
        pinMode(rij[i], OUTPUT);                            // rij pinnen als output
        pinMode(kol[i], OUTPUT);                            // kolom pinnen als output
    }

    // initialisatie matrix
    InitMatrix(Uit);                                     // led matrix uitzetten

    // setup library
    FrequencyTimer2::disable();                         // afzetten gebruik pin 11
    FrequencyTimer2::setPeriod(2000);                   // interrupt iedere 2000  $\mu$ s
    FrequencyTimer2::setOnOverflow(TekenMatrix);        // interrupt routine

    // patroon ophalen
    LaadPatroon(p);                                    // patroon opladen
}

}
```

In de loop functie verhogen we de variabele van het patroon en zorgen ervoor dat de cyclus steeds herhaald wordt; als het laatste patroon (smiley) afgebeeld wordt starten we terug met het eerst (H). Dit gebeurt door de instructie :

$p = ++p \% AantalPatronen;$

Het scrollen van de letters gebeurt door de instructie :

ScrollPatroon(p, 150);

```

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    p = ++p % AantalPatronen;
    ScrollPatroon(p, 150);
}

// initialisatie van de LED matrix
void InitMatrix(boolean Aan) {
    if (Aan) { // alle leds aan
        for (int r=0; r<8; r++) {
            for (int k=0; k<8; k++) {
                led[r][k] = 1; // led aan
                digitalWrite(rij[r], HIGH); // rij pin anode aan
                digitalWrite(kol[k], LOW); // kol pin kathode uit
            }
        }
    } else { // alles leds uit
        for (int r=0; r<8; r++) {
            for (int k=0; k<8; k++) {
                led[r][k] = 0; // led uit
                digitalWrite(rij[r], LOW); // rij pin anode uit
                digitalWrite(kol[k], HIGH); // kol pin kathode aan
            }
        }
    }
}

// opladen patroon
void LaadPatroon(int p) {
    for (int r=0; r<8; r++) {
        for (int k=0; k<8; k++) {
            led[r][k] = pat[p][r][k]; // patroon naar led kopieren
        }
    }
}

// interrupt routine - teken het patroon op de LED matrix
void TekenMatrix() {
    digitalWrite(rij[row], LOW); // vorige rij pin anode af
    row++; // rij variabele + 1
    if (row == 8) { // overflow
        row = 0; // rij variabele resetten
    }
    for (k=0; k<8; k++) {
        if (led[row][k] == 1) { // kol pin kathode uit
            digitalWrite(kol[k], LOW);
        } else { // kol pin kathode aan
            digitalWrite(kol[k], HIGH);
        }
    }
    digitalWrite(rij[row], HIGH); // rij pin anode aan
}

// scroll patroon rechts naar links
void ScrollPatroon(int p, int tijd) {
    for (int l = 0; l < 8; l++) {
        for (int r=0; r<8; r++) {
            for (int k=0; k<7; k++) { // lus kolommen
                led[r][k] = led[r][k+1]; // steek volgende kol in huidige
            }
        }
        for (int r=0; r<8; r++) { // lus rijen
            led[r][7] = pat[p][r][0 + l]; // steek 1ste kol patroon in laatste kol
        }
        delay(tijd); // wacht tijdje
    }
}

```

Project 14 : Serieel naar parallel conversie

Opgave

Maak met behulp van de 74HC595 serieel naar parallel convertor een running light met 8 rode leds (à la Knightrider).

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 serieel naar parallel convertor 74HC595
- 8 rode leds
- 8 weerstanden 220 ohm

Serieel naar parallel convertor 74HC595



Om de 8 leds aan te sturen zouden we natuurlijk iedere led met een poortje op de UNO kunnen verbinden.

Als we een 74HC595 gebruiken kunnen we via een serieel naar parallele omzetting het aantal nodige poorten beperken tot 3. De chip heeft 8 output poorten en 3 input poorten waarmee we de data bit per bit kunnen aanleveren.

Werking

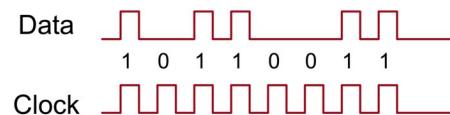
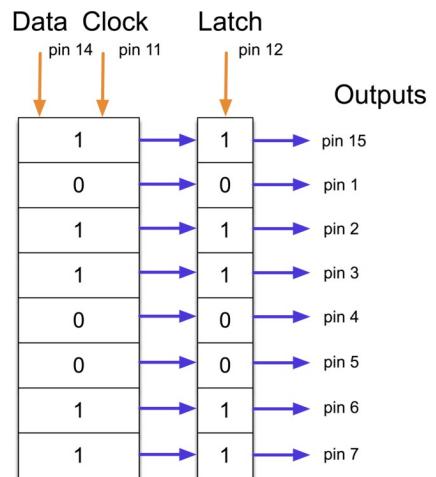
Het *shift register* van de chip kunnen we zien als 8 geheugen locaties die een 0 of een 1 kunnen bevatten.

Om die locaties op te vullen worden de *data pin 14* en de *clock pin 11* gebruikt.

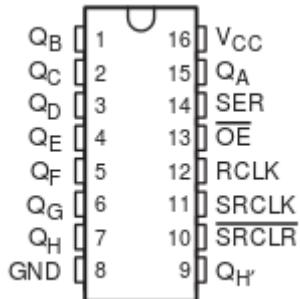
Op de clock pin komt een blokgolf van 8 pulsen. Als de clock pin hoog is wordt gekeken naar de data pin. Is die op dat moment hoog, dan wordt een 1 in het shift register geduwd, in het ander geval een 0. Bij de volgende hoge clock pin wordt dat proces herhaald en wordt een nieuwe 1 of 0 in het shift register geduwd. De bit van de vorige puls wordt mee naar beneden geduwd; vandaar de naam shift register.

Als alle 8 bits ontvangen zijn, kan de *latch pin 12* hoog gezet worden. Hierdoor worden de waarden in het shift register gekopieerd naar het *latch register*, en komen zo beschikbaar op de *output pinnen*.

Er is ook nog een OE (output enable) pin om alle output ineens aan- of af te zetten. Deze kan gebruikt om via een PWM poort op de UNO de helderheid van de leds aan te passen. Opgelet : om die te activeren moet hij laag staan.



Aansluitingen

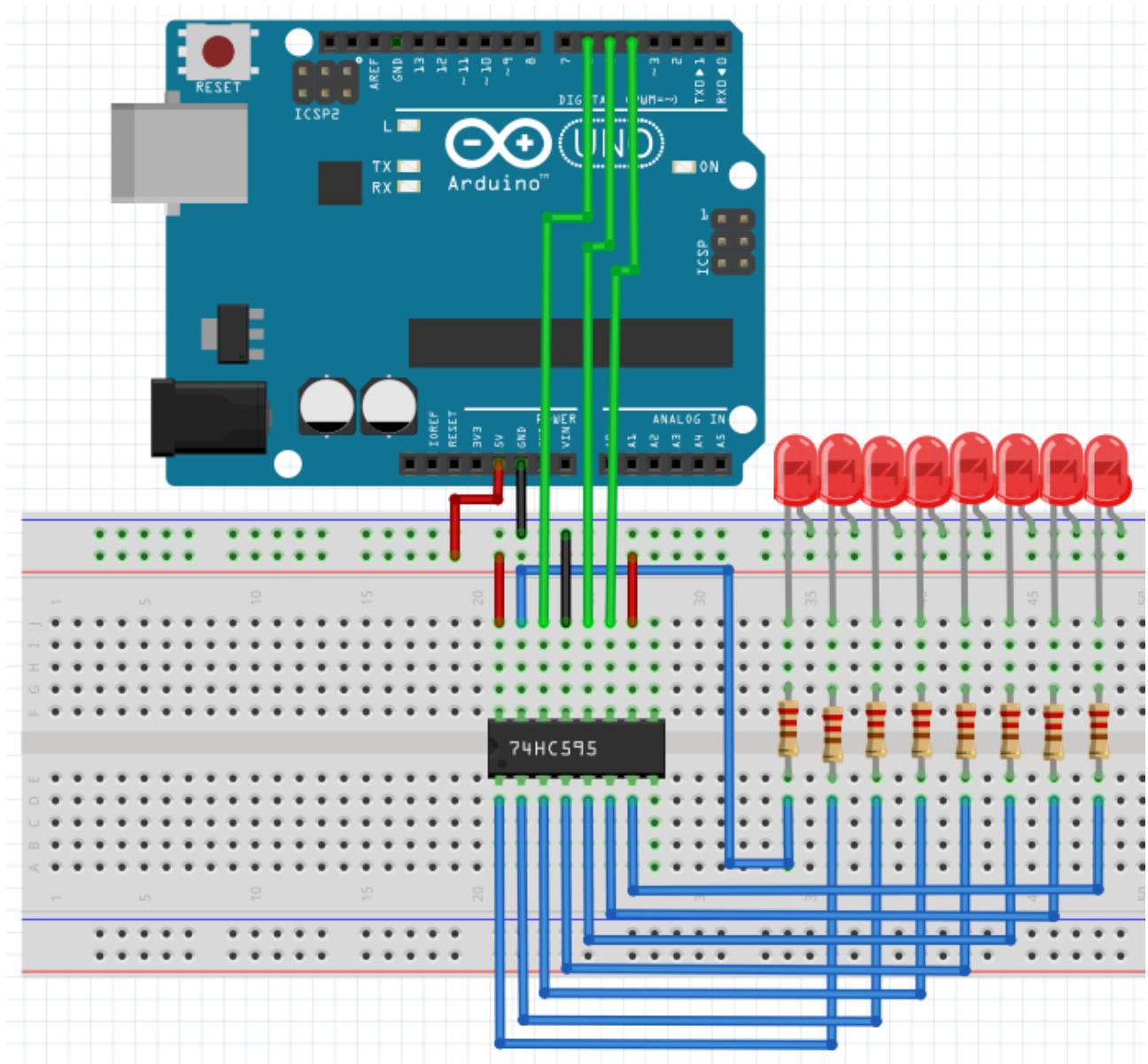


Pin #	Pin naam	I/O	Omschrijving
15, 1-7	Qa – Qh	O	Parallelle output data
8	GND	I	IC massa in
9	Qh'	O	Seriële bit uit
10	!SRCLR	I	Master clear voor seriële registers (actief laag)
11	SRCLK	I	Master klok voor seriële registers ('clock')datasheet english
12	RCLK	I	Master klok voor output registers ('latch')
13	!OE	I	Master controle voor output buffers aanzetten (actief laag)
14	SER	I	Seriële data bit in ('data')
16	Vcc	I	IC positieve voeding in

FUNCTION TABLE

INPUTS					FUNCTION
SER	SRCLK	SRCLR	RCLK	OE	
X	X	X	X	H	Outputs Q _A -Q _H are disabled.
X	X	X	X	L	Outputs Q _A -Q _H are enabled.
X	X	L	X	X	Shift register is cleared.
L	↑	H	X	X	First stage of the shift register goes low. Other stages store the data of previous stage, respectively.
H	↑	H	X	X	First stage of the shift register goes high. Other stages store the data of previous stage, respectively.
X	X	X	↑	X	Shift-register data is stored in the storage register.

Breadboard schakeling



Pin 74HC595	Functie	Pin UNO
15, 1-7	Qa-Qh	(anodes leds via weerstand)
8	GND	GND
9	Qh'	-
10	!SRCLR	5V
11	SRCLK (clock)	4
12	RCLK (latch)	5
13	!OE	GND
14	SER (data)	6
16	Vcc	5V

Sketch

We declareren constanten voor de clock-, data- en latchpin. Vervolgens declareren we een reeks met 16 patronen waarin we onze leds willen laten oplichten om het running light na te bootsen. We gebruiken hiervoor byte variabelen. 1 byte bevat 8 bits en is juist voldoende om de toestand van onze 8 leds te sturen. Merk op dat we onze bytes in binaire vorm declareren door de hoofdletter B voor de binaire notatie te zetten.

In de setup functie definiëren we onze pinnen als output en zetten we alle leds uit met behulp van de functie *updateLeds(leds)*. De variabele *leds* is weerom een byte met een patroon dat moet worden afgebeeld. Hier is dit patroon decimaal nul; binair zijn dit 8 nullen en dus gaan alle leds uit.

In de loop functie lopen we met een for lus doorheen onze 16 patronen en beelden ze achtereenvolgens af met dezelfde *updateLeds(leds)* functie gevuld door een korte wachttijd.

```
/*
 Maak met behulp van de 74HC595 een running light met 8 rode leds (à la Knightrider).

*/
// constanten
const int latchPin = 5;
const int clockPin = 4;
const int dataPin = 6;

const int aantalPatronen = 16;
const byte pat[aantalPatronen] = { \           // byte patronen in binaire vorm
  B00000001, B00000011, B00000110, B00000110, \
  B00011000, B00110000, B01100000, B11000000, \
  B10000000, B11000000, B01100000, B00110000, \
  B00011000, B00001100, B00000110, B00000011 };

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
  byte leds;

  // pinnen als output declareren
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);

  // alle leds afzetten
  leds = 0;
  updateLeds(leds);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
  byte leds;
  // lus door patronen
  for (int i=0; i<16; i++) {
    leds = pat[i];
    updateLeds(leds);
    delay(100);
  }
}
```

De functie `updateLeds(leds)` doet het eigenlijke werk om de 74HC595 aan te sturen en onze leds te laten oplichten :

1. Eerst wordt de latchpin laag gezet zodat we tijdens het doorsturen van de data geen flikkerende leds krijgen.
2. Daarna wordt de variabele `leds` (die het patroon van de 8 leds bevat) doorgestuurd naar de chip via het commando : `shiftOut(dataPin, clockPin, LSBFIRST, leds);`

`LSBFIRST` wil zeggen dat eerst de minst belangrijke bit (rechtse) wordt gepushed in het shift register. Hierdoor komt het bitpatroon eigenlijk omgekeerd in het register terecht. We gebruiken hier dan ook `MSBFIRST`; de meest belangrijke bit (links) wordt eerst gepushed en het bitpatroon komt juist terecht in het register.

3. Alle 8 bits zitten nu klaar in het shift register. We zetten de latchpin hoog zodat de data gekopieerd wordt naar het latch register. Afhankelijk van het bitpatroon worden de output pinnen hoog of laag gezet en die sturen op hun beurt de anodes van de leds via een weerstand aan.

```
// data van leds seriëel inlezen en parallel uitsturen naar output pinnen
void updateLeds(byte leds) {
    // latch pin laag zetten; we beginnen met inlezen
    digitalWrite(latchPin, LOW);

    // data van leds seréel inlezen van datapin
    // LSBFIRST - Least Significant Bit First = minst belangrijke bit eerst
    // MSBFIRST - Most Significant Bit First = meest belangrijke bit eerst
    shiftOut(dataPin, clockPin, MSBFIRST, leds);

    // alles is ingelezen; zet latch hoog zodat shift register beschikbaar komt op output pinnen
    digitalWrite(latchPin, HIGH);
}
```

Project 15 : Infrarode afstandsbediening

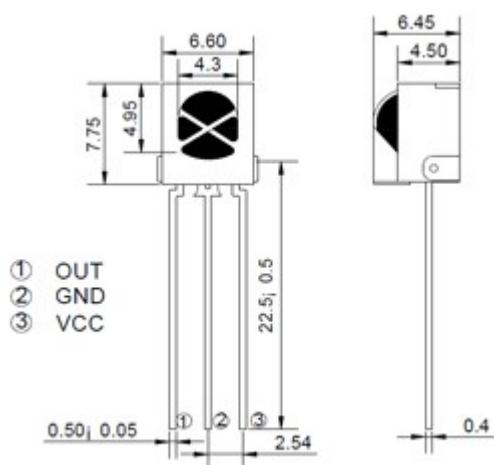
Opgave

Gebruik de VS 1838B infrarode (IR) sensor om drie leds aan en uit te zetten met de UNO. Gebruik hiervoor de toetsen 1, 2 en 3 van de afstandsbediening. Met toets 4 schakelen we alle leds terug uit.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 IR sensor VS 1838B
- 1 afstandsbediening (NEC protocol)
- 3 leds (verschillende kleuren)
- 3 weerstanden 220 ohm

Infrarode sensor VS 1838B



Afstandsbedieningen voor tv, dvd, ... gebruiken binaire elektronische pulsen die door de IR zender led omgezet worden naar infrarood licht.

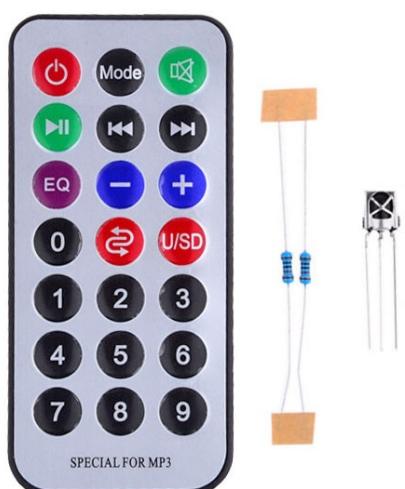
De infrarode sensor VS 1838B vangt dit IR licht op en zet ze terug om in elektronische pulsen die door onze UNO geïnterpreteerd worden.

De VS 1838B werkt op een frequentie van 38 kHz (38.000 pulsen per sec).

Om communicatie mogelijk te maken moeten IR afstandsbediening en IR ontvanger :

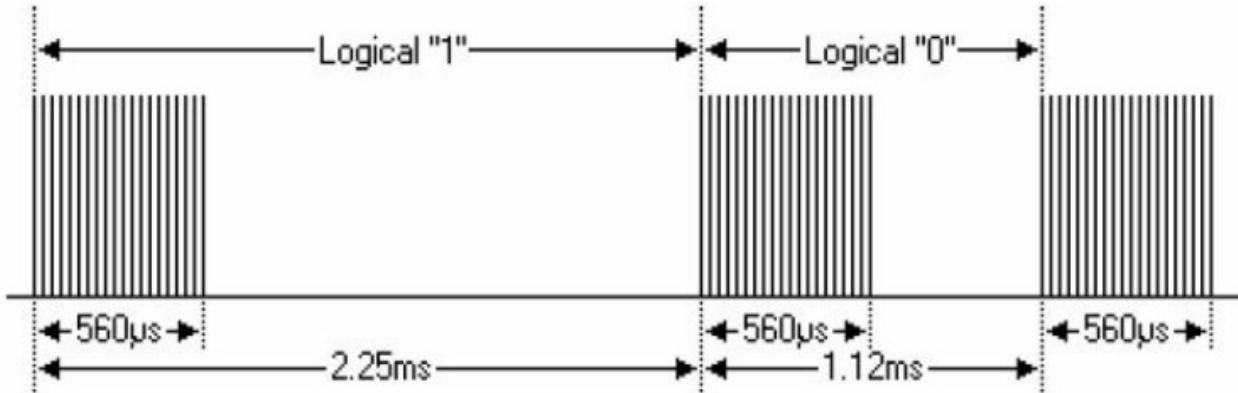
1. op dezelfde frequentie werken.
2. elkaar kunnen zien (geen obstakels ertussen).

Er zijn verschillende protocollen ontwikkeld voor IR communicatie. Degene die we hier gaan gebruiken in combinatie met de onze afstandsbediening hiernaast is het NEC protocol.



Nec protocol kenmerken

1. Adressen en commando's van 8 bits.
2. Adressen en commando's worden dubbel verzonden om de betrouwbaarheid te verhogen.
3. Puls afstandsmodulatie.
4. Draaggolf frequentie : 38 kHz.
5. Bit '1' duurt 2,25 ms; bit '0' duurt 1,12 ms (half zolang).



Nec protocol voorbeeld



Dit is een typische NEC protocol puls sequentie. Die begint met een marker waarbij de IR led gedurende 9 ms wordt aangezet en vervolgens gedurende 4,5 ms wordt uitgezet. Daarna wordt het adres 0x59 (hex notatie) of 1001 1010 (binair) doorgestuurd. Vervolgens het commando 0x16 (hex) of 1000 0110 (binair). Na ieder adres en commando wordt de geïnverteerde waarde doorgestuurd; die kan als controle gebruikt worden.
motor

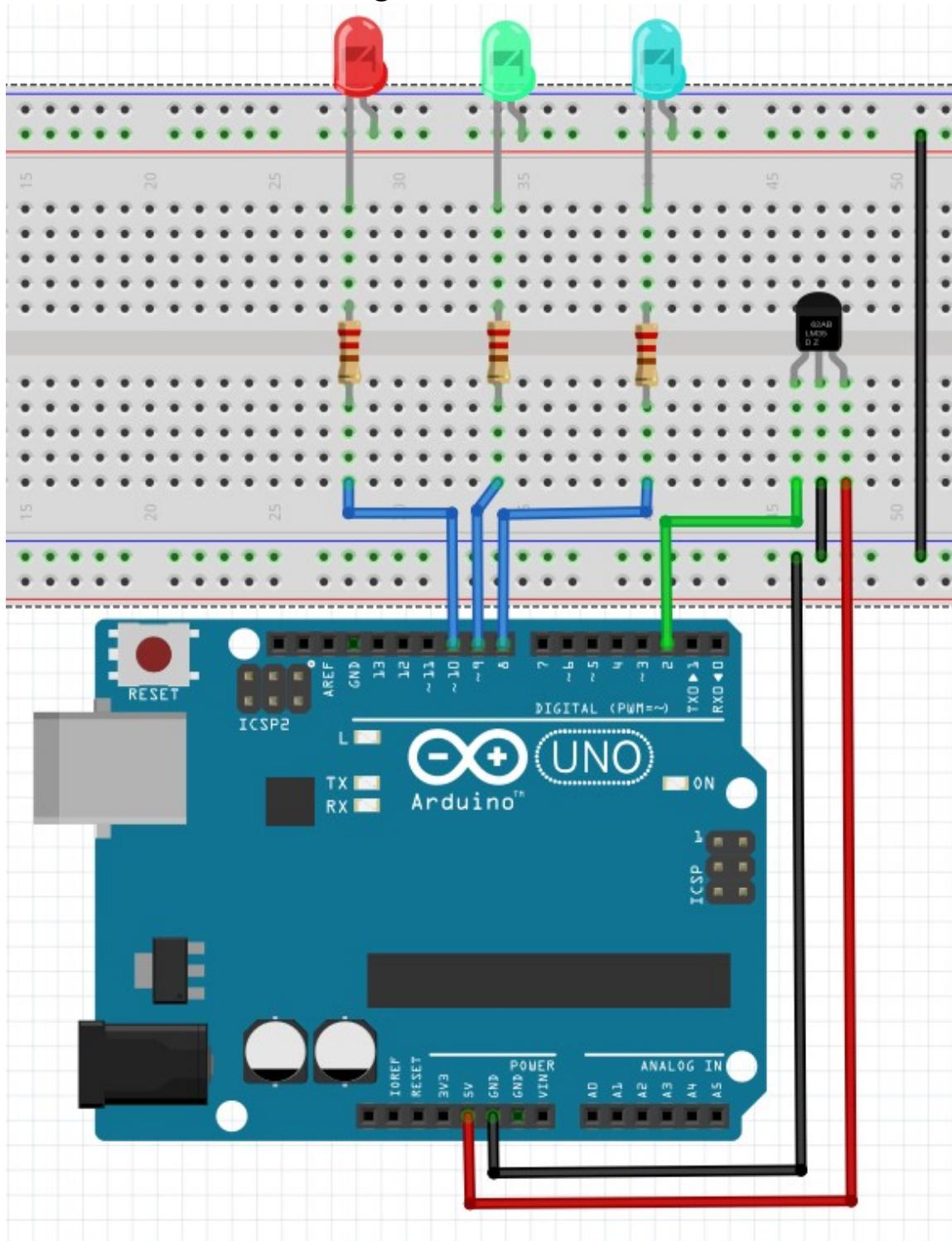
Bibliotheek IRremote

Deze bibliotheek bevat een aantal handige functies om de IR communicatie een stuk te vereenvoudigen. Het bevat ook de verschillende protocollen die gebruikt worden voor IR communicatie.

Ga naar [deze webpagina](#) om de laatste versie te downloaden. Pak het zip bestand uit en kopieer de map *Arduino-IRremote-master* naar de map *libraries* van je Arduino IDE installatie.

Vergeet niet de map *RobotIRremote* te verwijderen ! Deze bevat gelijkaardige functies maar veroorzaakt fouten !

Breadboard schakeling



Pin VS1838B	Functie	Pin UNO
1	Out	2
2	GND	GND
3	Vcc	5V

Sketch

We voegen de IRremote bibliotheek toe op de reeds gekende manier : menu Schets – Bibliotheek importeren – IRremote. We declareren constanten voor de ontvangst pin van onze IR sensor en voor de pinnen van de 3 leds. We declareren tevens 3 boolean variabelen om de toestand van onze leds bij te houden (aan of uit).

We declareren vervolgens 2 objecten : een IR ontvanger object *IRontvanger* en een object *resultaten* dat de vertaalde IR codes bevat.

In de setup functie wordt de IR ontvangst pin als input gedeclareerd en de 3 ledpinnen als output. We zetten de leds uit en initialiseren de seriële monitor voor debugging. Vervolgens starten we de IR ontvanger op.

```
/*
Gebruik de VS 1838B infrarode (IR) sensor om drie leds aan en uit te zetten met de UNO.
Gebruik hiervoor de toetsen 1, 2 en 3 van de afstandsbediening.
Met toets 4 schakelen we alle leds terug uit.

*/
#include <IRremoteInt.h>
#include <IRremote.h>

// constanten
const int IRPin = 2;                                // pin van de IR sensor
const int LedRPin = 10;                             // pin voor de rode led
const int LedGPin = 9;                               // pin voor de groene led
const int LedBPin = 8;                               // pin voor de blauwe led

// variabelen
boolean Rood = false;                            // status rode led
boolean Groen = false;                           // status groene led
boolean Blauw = false;                           // status blauwe led

// declaratie objecten
IRrecv IRontvanger(IRPin);                      // IR ontvanger object
decode_results resultaten;                        // vertaalde IR kodes

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    // de pinnen initialiseren
    pinMode(IRPin, INPUT);
    pinMode(LedRPin, OUTPUT);
    pinMode(LedGPin, OUTPUT);
    pinMode(LedBPin, OUTPUT);

    // de leds uitzetten
    digitalWrite(LedRPin, LOW);
    digitalWrite(LedGPin, LOW);
    digitalWrite(LedBPin, LOW);

    // de seriële monitor initialiseren
    Serial.begin(9600);
    Serial.print("IR HEX data");
    Serial.println("\tIR vertaalde data");

    // start de IR ontvanger
    IRontvanger.enableIRIn();
}

}
```

In de loop functie testen we met `if (IRontvanger.decode(&resultaten))` of er iets ontvangen is. Is dat zo dan tonen we de rauwe data in de seriële monitor. Vervolgens wordt de IRdispatcher functie aangeroepen die afhankelijk van de waarde bepaalde acties zal ondernemen. Daarna wordt de volgende waarde opgehaald.

```
// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    if (IRontvanger.decode(&resultaten)) { // IR signaal ontvangen
        Serial.print(resultaten.value, HEX); // toon de rauwe data in de seriële monitor
        IRdispatcher(); // IR dispatcher
        IRontvanger.resume(); // haal volgende data op
    }
}

// dispatcher functie die acties doet bij bepaalde IR codes/toetsen
void IRdispatcher() {
    switch(resultaten.value) {
        case 0xFF30CF:
            Serial.println("\ttoets 1 - ROOD aan/uit");
            if (Rood) {
                digitalWrite(LedRPin, LOW);
            } else {
                digitalWrite(LedRPin, HIGH);
            }
            Rood = !Rood;
            break;
        case 0xFF18E7:
            Serial.println("\ttoets 2 - GROEN aan/uit");
            if (Groen) {
                digitalWrite(LedGPin, LOW);
            } else {
                digitalWrite(LedGPin, HIGH);
            }
            Groen = !Groen;
            break;
        case 0xFF7A85:
            Serial.println("\ttoets 3 - BLAUW aan/uit");
            if (Blauw) {
                digitalWrite(LedBPin, LOW);
            } else {
                digitalWrite(LedBPin, HIGH);
            }
            Blauw = !Blauw;
            break;
        case 0xFF10EF:
            Serial.println("\ttoets 4 - ALLES uit");
            Rood = false;
            Groen = false;
            Blauw = false;
            digitalWrite(LedRPin, LOW);
            digitalWrite(LedGPin, LOW);
            digitalWrite(LedBPin, LOW);
            break;
        default:
            Serial.println("\tandere toets");
    }
}
```

The screenshot shows the Arduino IDE's serial monitor window titled '/dev/ttyACM0 (Arduino Uno)'. The window displays a list of raw IR codes followed by their corresponding actions:

Raw IR Code	Action
FFFFFFF	andere toets
FF18E7	toets 2 - GROEN aan/uit
FFFFFFF	andere toets
FF7A85	toets 3 - BLAUW aan/uit
FFFFFFF	andere toets
FF10EF	toets 4 - ALLES uit
FFFFFFF	andere toets
FF7A85	toets 3 - BLAUW aan/uit
FFFFFFF	andere toets
FF18E7	toets 2 - GROEN aan/uit
FFFFFFF	andere toets
FF30CF	toets 1 - ROOD aan/uit
FFFFFFF	andere toets
FF10EF	toets 4 - ALLES uit
FFFFFFF	andere toets

Project 16 : Lichtsensor GL55xx

Opgave

Maak met behulp van een lichtsensor GL55xx een schakeling die afhankelijk van de lichtsterkte van de omgeving een led feller of zwakker laat oplichten. Hoe groter de lichtsterkte, hoe zwakker de led moet branden. Gebruik de potentiometer om de lichtsterkte waarbij de led begint te branden te regelen.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 lichtsensor GL55xx
- 1 led
- 1 weerstand 220 ohm
- 1 weerstand 10k ohm
- 1 potentiometer 10k ohm

Lichtsensor GL55xx

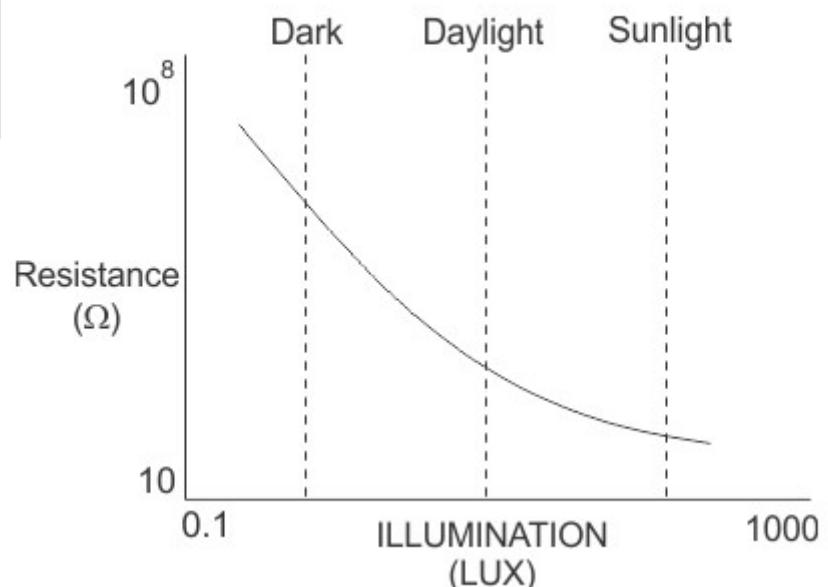


Een lichtsensor zoals de GL55 serie is ook gekend onder de naam fotocel.

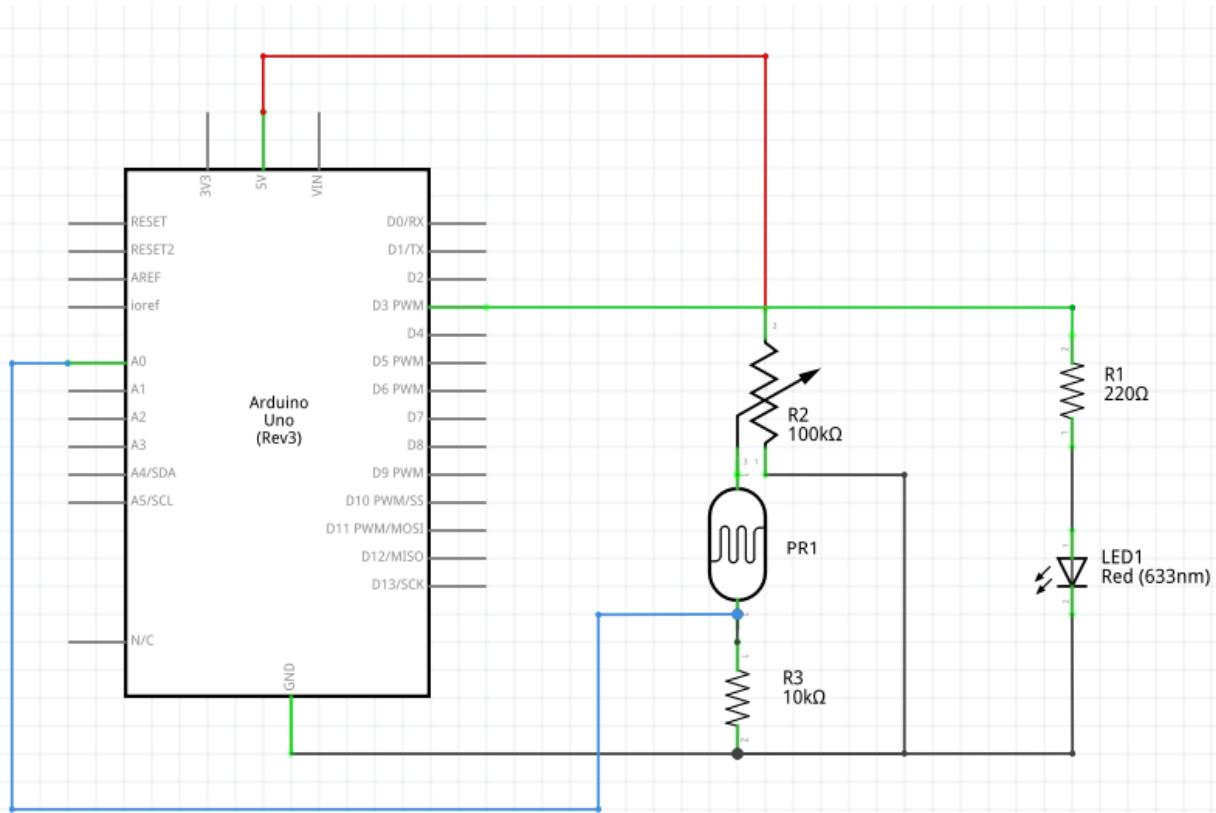
De Engelse benaming Light Dependent Resistor (LDR) geeft eigenlijk beter aan wat deze component doet.

De component is een variabele weerstand. De hoeveelheid licht die op de bovenkant van de sensor valt, verandert de weerstand tussen de 2 pinnen.

Hoe meer licht, hoe lager de weerstand. De weerstand kan variëren van enkele 100M ohm (in het donker) tot enkele 10 ohm (in zonlicht).



Schakeling

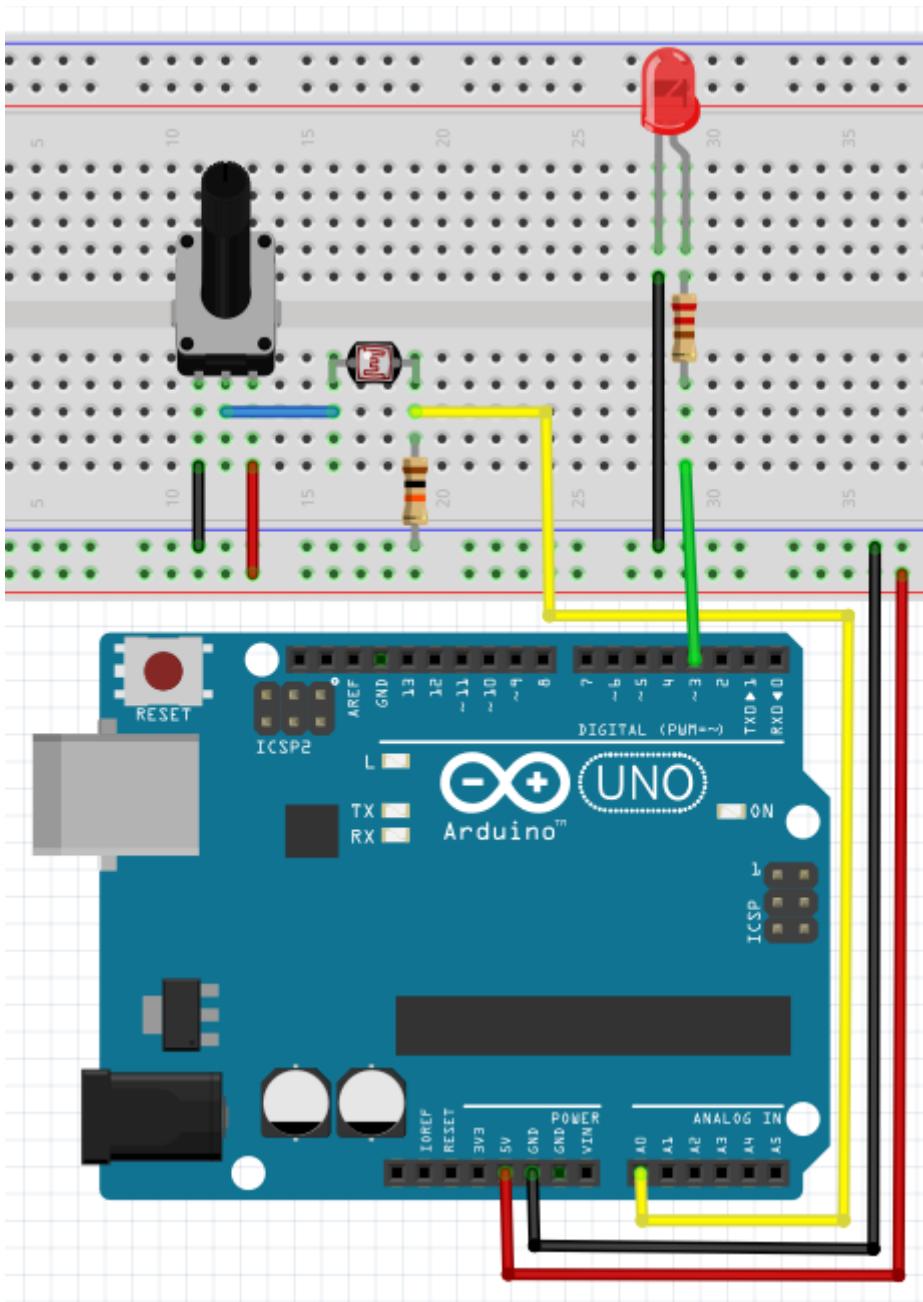


We zetten onze lichtsensor in serie met een 10k ohm vaste weerstand en een 10k ohm potentiometer. We verbinden de analoge poort A0 tussen de lichtsensor en de vaste weerstand (zie project 8 over de spanningsdeler).

De potentiometer kunnen we gebruiken om het punt waarop de led overdag moet uit zijn bij te regelen.

De led verbinden we via een 220 ohm weerstand met de PWM poort 3.

Breadboard schakeling



Sketch

We declareren 2 constanten voor de pin van de lichtsensor en de pin van de led. We declareren ook variabelen voor de huidige, het minimum en het maximum van de meetwaardes van de lichtsensor. In de setup functie declareren we de sensor pin als input en de led pin als output. We zetten de led uit en we initialiseren de seriële monitor voor het debuggen van de meetwaardes.

In de loop functie lezen we de sensorwaarde uit met `analogRead(sensorPin);` Deze geeft een waarde tussen 0 en 1023. Voor het aansturen van de PWM poort voor de led hebben we een waarde nodig tussen 0 en 255. Die conversie wordt gedaan in de functie `autoTune()`. Vervolgens wordt de led aangestuurd met `analogWrite(ledPin, lichtsterkte);` gevolgd door een korte pauze.

```

/*
Maak met behulp van een lichtsensor GL55xx een schakeling die afhankelijk van de lichtsterkte
van de omgeving een led feller of zwakker laat oplichten. Hoe groter de lichtsterkte,
hoe zwakker de led moet branden. Gebruik de potentiometer om de lichtsterkte waarbij de led
begint te branden te regelen.

*/
// constanten
const int sensorPin = A0; // pin van de lichtsensor
const int ledPin = 3; // pin van de rode led

// variabelen
int lichtsterkte; // lichtsterkte sensor meetwaarde
int minLicht = 1023, maxLicht = 0; // variabelen voor min en max lichtsterkte

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    // de pinnen initialiseren
    pinMode(sensorPin, INPUT);
    pinMode(ledPin, OUTPUT);

    // de led uitzetten
    digitalWrite(ledPin, LOW);

    // de serële monitor initialiseren
    Serial.begin(9600);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    // lees de lichtsterkte (0-1023)
    lichtsterkte = analogRead(sensorPin);

    // toon gemeten lichtsterkte
    Serial.print("sensor: ");
    Serial.print(lichtsterkte);

    // zet de waarde om naar PWM range (0-255) voor de led aansturing
    autoTune();

    // toon geconverteerde stuurwaarde
    Serial.print("\tled: ");
    Serial.println(lichtsterkte);

    // laat de led branden op lichtsterkte
    analogWrite(ledPin, lichtsterkte);

    // even wachten
    delay(500);
}

```

In de functie autoTune() zit de eigenlijke intelligentie voor het aansturen van de led. De analoge poort A0 geeft een waarde tussen 0 en 1023. Voor de PWM poort van de led hebben we een waarde nodig tussen 0 en 1023.

De lichtsensor zal echter nooit de uiterste waarden weergeven bij dag of nacht. Het zullen eerder waarden zijn tussen 300 en 900. Hoeveel die minimum en maximum waarde is zou je experimenteel kunnen vaststellen door de lichtsensor te verlichten met een zaklamp en te verduisteren met je hand bijvoorbeeld. We zullen die waarden echter door de UNO zelf laten bijhouden in de variabelen minLicht en maxLicht. Initieel staan die op 1023 en 0. Tijdens iedere cyclus worden deze

variabelen bijgewerkt met de 2 *if* constructies.

Met behulp van de *map* functie wordt de eigenlijke conversie gedaan. sainsmart

```
waarde = map(waarde, bron_min, bron_max, doel_min, doel_max);
```

We gebruiken voor het bronbereik minLicht tot maxLicht met een speling van 30. Het doelbereik is uiteraard de 0 en 255 van de PWM led poort. Dit wordt dan :

```
lichtsterkte = map(lichtsterkte, minLicht+30, maxLicht-30, 0, 255);
```

Soms kan het uitlezen van de lichtsensor een ongeldige waarde opleveren. We filteren deze eruit met de functie *constrain()*. Negatieve waarden worden 0, waarden hoger dan 255 worden 255.

```
lichtsterkte = constrain(lichtsterkte, 0, 255);
```

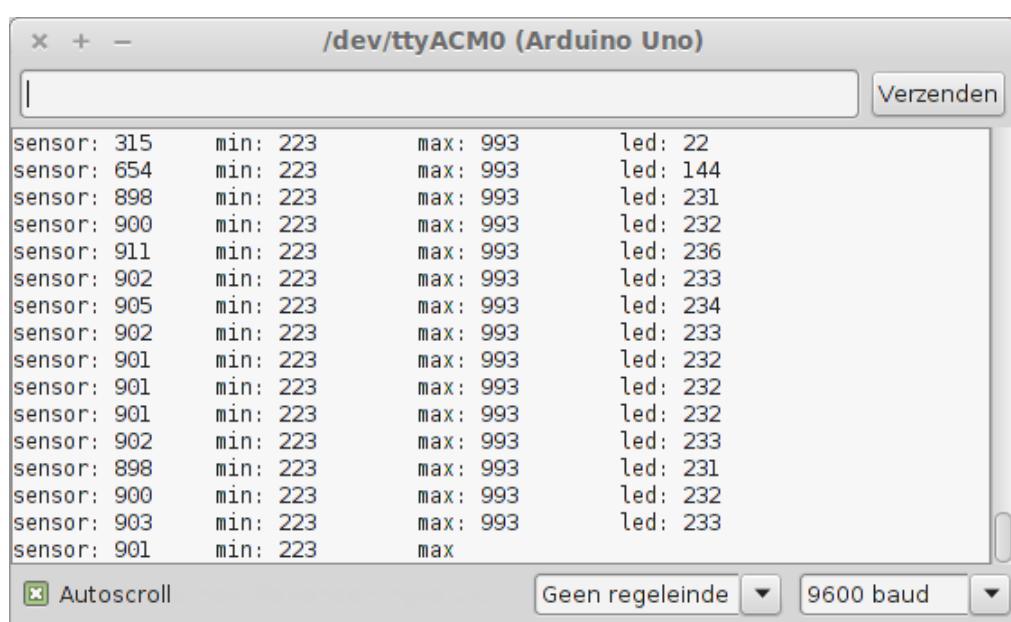
```
// functie om lichtsterkte automatisch om te zetten naar de juiste waarde  
// bij nacht moet led maximaal branden; bij dag moet led uit zijn.  
void autoTune() {
```

```
// pas minimum gemeten lichtsterkte aan  
if (lichtsterkte < minLicht) {  
    minLicht = lichtsterkte;  
}
```

```
// pas maximum lichtsterkte aan  
if (lichtsterkte > maxLicht) {  
    maxLicht = lichtsterkte;  
}
```

```
// toon min en max in seriële monitor  
Serial.print("\tmin: ");  
Serial.print(minLicht);  
Serial.print("\tmax: ");  
Serial.print(maxLicht);
```

```
// zet lichtsterkte om naar optimale waarde voor PCM poort led (0-255)  
lichtsterkte = map(lichtsterkte, minLicht+30, maxLicht-30, 0, 255);  
lichtsterkte = constrain(lichtsterkte, 0, 255);
```



The screenshot shows the Arduino Serial Monitor window titled '/dev/ttyACM0 (Arduino Uno)'. It displays a series of sensor readings in a tabular format. The columns are labeled 'sensor', 'min', 'max', and 'led'. The data is as follows:

sensor	min	max	led
315	223	993	22
654	223	993	144
898	223	993	231
900	223	993	232
911	223	993	236
902	223	993	233
905	223	993	234
902	223	993	233
901	223	993	232
901	223	993	232
901	223	993	232
902	223	993	233
898	223	993	231
900	223	993	232
903	223	993	233
901	223	max	

At the bottom of the window, there are buttons for 'Autoscroll' and 'Verzenden', and dropdown menus for 'Geen regeleinde' and '9600 baud'.

Project 17 : Sturing van DC motoren

Opgave

Gebruik de L298N motorsturing module om 2 DC motoren aan te sturen. We bedienen de motoren via toetsaanslagen in de seriële monitor :

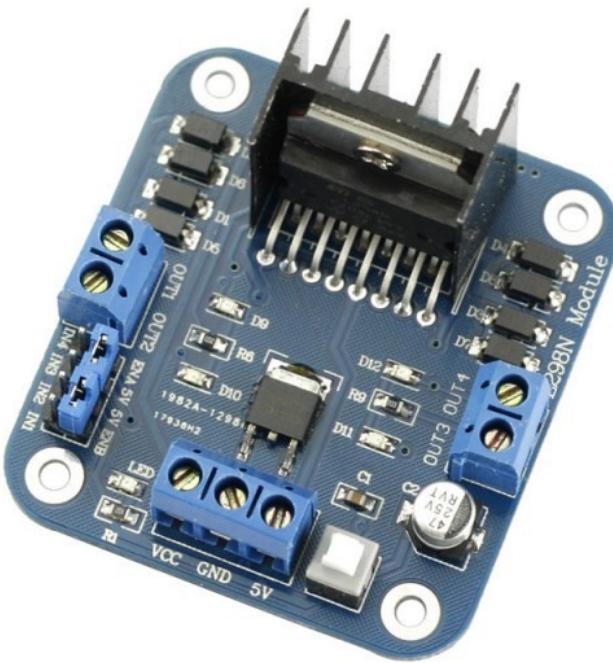
Motor A : '1' = vooruit – '2' = stoppen – '3' = achteruit

Motor B : '4' = vooruit – '5' = stoppen – '6' = achteruit

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 L298N module
- 2 DC motoren
- 1 9V batterij

L298N motorsturing module



De L298N (gemonteerd op de grote koelvin) is een zogenaamde H-bridge. Die worden veel gebruikt om motoren aan te sturen.

Deze module kan dienen voor zowel het sturen van 2 stappenmotoren (servomotoren) als van 2 gewone DC motoren.

Voor het aansturen van een stappenmotor moet de jumper over de enable pin en de 5V pin geplaatst zijn.

Voor het aansturen van een DC motor kan je een PWM signaal op de enable pin plaatsen om de draaisnelheid aan te passen. Voor meer uitleg over Pulse Width Modulation (PWM) zie project 6.

De draairichting van de motor wordt bepaald door de signalen die op de In aansluitingen staan. De combinatie LAAG – HOOG laat de motor vooruit draaien; de combinatie HOOG – LAAG achteruit.

Wanneer een externe voeding (>5V) is aangesloten op VCC, dan kan je met de 5V aansluiting op de module de UNO voeden. Vergeet uiteraard ook niet de GND met de UNO te verbinden om het circuit te sluiten.

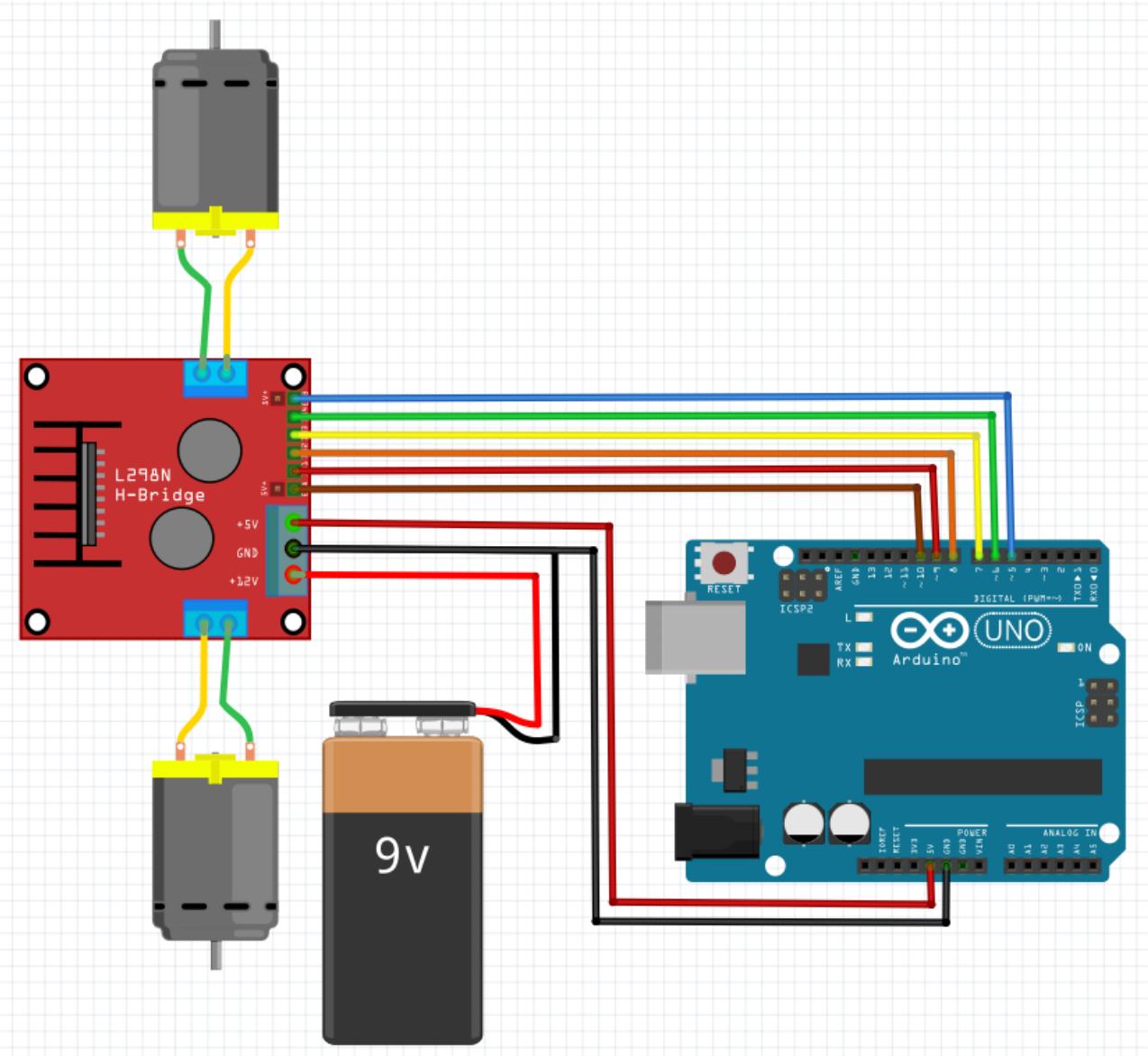
Aansluitingen

Pin L298N	Functie
Out1	Motor A aansluiting +
Out2	Motor A aansluiting -
Out3	Motor B aansluiting +
Out4	Motor B aansluiting -
VCC	Voedingsspanning (5V – 35V)
GND	Massa
5V	5V output als VCC reeds spanning heeft > 5V
EnA	Motor A : met jumper = stappenmotor – zonder = DC motor snelheid via PWM
EnB	Motor B : met jumper = stappenmotor – zonder = DC motor snelheid via PWM
In1, In2	Motor A draairichting : vooruit = LAAG-HOOG / achteruit = HOOG-LAAG
In3, In4	Motor B draairichting : vooruit = LAAG-HOOG / achteruit = HOOG-LAAG

Breadboard schakeling

We maken de volgende verbindingen tussen de L298N module en de andere componenten :

Pin L298N	Functie	Wat	Pin
Out1	Motor A aansluiting +	Motor A	+
Out2	Motor A aansluiting -	Motor A	-
Out3	Motor B aansluiting +	Motor B	+
Out4	Motor B aansluiting -	Motor B	-
VCC	Voedingsspanning (5V – 35V)	Batterij 9V	+
GND	Massa	Batterij 9V UNO	– GND
5V	5V output als VCC reeds spanning heeft > 5V	UNO	5V
EnA	Motor A snelheid	UNO	10
In1	Motor A draairichting	UNO	6
In2	Motor A draairichting	UNO	7
EnB	Motor B snelheid	UNO	5
In3	Motor B draairichting	UNO	8
In4	Motor B draairichting	UNO	9



Sketch

We declareren voor iedere motor 3 constanten : 2 voor de pinnen om de draairichting te sturen en 1 voor de PWM pin die de snelheid stuurt. We declareren ook een variabele voor de snelheid.

In de setup functie initialiseren we alle pinnen als output. We initialiseren tevens de seriële console die we zullen gebruiken om via toetsaanslagen verschillende commando's te geven om de motoren aan te sturen :

Draairichting	Motor A	Motor B
Vooruit	1	4
Stoppen	2	5
Achteruit	3	6

```

/*
Gebruik de L298N motorsturing module om 2 DC motoren te sturen.
We besturen de motoren via toetsaanslagen in de seriële monitor :
    Motor 1 : '1' = vooruit - '2' = stoppen - '3' = achteruit
    Motor 2 : '4' = vooruit - '5' = stoppen - '6' = achteruit

*/
// constanten
const int In1Pin = 6;                                // In1 pin motor A (draairichting)
const int In2Pin = 7;                                // In2 pin motor A (draairichting)
const int EnAPin = 10;                               // EnA pin motor A (snelheid - PWM!)

const int In3Pin = 8;                                // In3 pin motor B (draairichting)
const int In4Pin = 9;                                // In4 pin motor B (draairichting)
const int EnBPin = 5;                                // EnB pin motor B (snelheid - PWM!)

// variabelen
int snelheid;                                         // variabele voor snelheid

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    // de pinnen initialiseren
    pinMode(In1Pin, OUTPUT);
    pinMode(In2Pin, OUTPUT);
    pinMode(In3Pin, OUTPUT);
    pinMode(In4Pin, OUTPUT);
    pinMode(EnAPin, OUTPUT);
    pinMode(EnBPin, OUTPUT);

    // Seriele monitor initialiseren
    Serial.begin(9600);
    Serial.println("Motor A: 1=vooruit - 2=stoppen - 3=achteruit");
    Serial.println("Motor B: 4=vooruit - 5=stoppen - 6=achteruit");
    Serial.println("-----");
}

}

```

In de loop functie testen we met `Serial.available()` of er een toets is doorgestuurd via de seriële monitor. Als dat zo is wordt die met `Serial.read()` ingelezen in de variabele `toets`. Vervolgens is er een switch statement die afhankelijk van de toets de juiste code uitvoert.

De draairichting van de motor wordt met 2 `digitalWrite(motorpin, waarde)` statements ingesteld aan de hand van volgende tabel :

Draairichting	Motorpin 1	Motorpin 2
Vooruit	laag	hoog
Stoppen	laag	laag
Achteruit	hoog	laag

De snelheid wordt met een `for` lus ingesteld met `analogWrite(enablepin, snelheid)`. Let hierbij op dat de gebruikte pinnen van het type Pulse Width Modulation (PWM) moeten zijn !

```

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    if (Serial.available() > 0) { // toetsaanslag ontvangen ?
        int toets = Serial.read(); // toets inlezen

        switch(toets) {

            case '1':
                Serial.println("Motor A vooruit");
                digitalWrite(In1Pin, LOW); // vooruit - laag
                digitalWrite(In2Pin, HIGH); // vooruit - hoog
                for (snelheid=0; snelheid<256; snelheid++) {
                    analogWrite(EnAPin, snelheid); // opspinnen
                    delay(20); // korte pauze
                }
                break;

            case '2':
                Serial.println("Motor A stoppen");
                for (snelheid=255; snelheid>=0; snelheid--) {
                    analogWrite(EnAPin, snelheid); // afspinnen
                    delay(20); // korte pauze
                }
                digitalWrite(In1Pin, LOW); // stoppen
                digitalWrite(In2Pin, LOW); // stoppen
                break;

            case '3':
                Serial.println("Motor A achteruit");
                digitalWrite(In1Pin, HIGH); // achteruit - hoog
                digitalWrite(In2Pin, LOW); // achteruit - laag
                for (snelheid=0; snelheid<256; snelheid++) {
                    analogWrite(EnAPin, snelheid); // opspinnen
                    delay(20); // korte pauze
                }
                break;

            case '4':
                Serial.println("Motor B vooruit");
                digitalWrite(In3Pin, LOW); // vooruit - laag
                digitalWrite(In4Pin, HIGH); // vooruit - hoog
                for (snelheid=0; snelheid<256; snelheid++) {
                    analogWrite(EnBPin, snelheid); // opspinnen
                    delay(20); // korte pauze
                }
                break;
        }
    }
}

```

```

        case '5':
            Serial.println("Motor B stoppen");
            for (snelheid=255; snelheid>=0; snelheid--) {
                analogWrite(EnBPin, snelheid);           // afspinnen
                delay(20);                            // korte pauze
            }
            digitalWrite(In3Pin, LOW);               // stoppen
            digitalWrite(In4Pin, LOW);               // stoppen
            break;

        case '6':
            Serial.println("Motor B achteruit");
            digitalWrite(In3Pin, HIGH);              // achteruit - hoog
            digitalWrite(In4Pin, LOW);               // achteruit - laag
            for (snelheid=0; snelheid<256; snelheid++) {
                analogWrite(EnBPin, snelheid);         // opspinnen
                delay(20);                            // korte pauze
            }
            break;

        default:
            Serial.println("Ongeldige toets");
    }
}
}

```

The screenshot shows the Arduino Serial Monitor window titled '/dev/ttyACM0 (Arduino Uno)'. The window has a text input field at the top and a 'Verzenden' (Send) button. Below the input field is a scrollable text area displaying the following interactions:

- Initial setup output:
Motor A: 1=vooruit - 2=stoppen - 3=achteruit
Motor B: 4=vooruit - 5=stoppen - 6=achteruit
- Sequence of commands entered:
Motor A vooruit
Motor A stoppen
Motor A achteruit
Motor A stoppen
Motor B vooruit
Motor B stoppen
Motor B achteruit
Motor B stoppen
Motor A vooruit
Motor B achteruit
Motor A stoppen
Motor B stoppen
- Serial monitor settings at the bottom:
Autoscroll checked
Baud rate set to 9600

Project 18 : Licht en donker detectie

Opgave

Gebruik de TCRT5000 optische sensor module om zwart en wit te detecteren op een blad papier. Toon de resultaten in de seriële monitor. Experimenteer ook eens met de potentiometer om de gevoeligheid aan te passen.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 TCRT5000 optische sensor module
- 1 blad papier met een dikke zwarte lijn

Optische sensormodule TCRT5000



De TCRT5000 optische sensormodule dient voor het detecteren van reflecterende materialen.

Deze sensor wordt veelal toegepast voor :

- detectie van stroommeter pulsen
- detectie van papier in papier shredders en fax apparaten
- detectie van obstakels
- lijn detectie zwart/wit (maken van een lijn volgende robot)

Specificaties

- gebruikt de infrarode reflectieve sensor TCRT5000
- gevoeligheid in te stellen via potentiometer
- optimale detectieafstand : 1 – 25 mm
- voeding : 3,3 – 5 V
- detectie led (rood)
- gebruikt de LM393 voltage comparator

Aansluitingen

Pin TCRT5000	Functie
G	Massa
V+	Voeding +
S	Analoge output (0 – 5V)

Schakeling

We sluiten de TCRT5000 als volgt aan op de UNO :

Pin TCRT5000	Functie	Pin UNO
G	Massa	GND
V+	Voeding +	5V
S	Output	A0

Deze keer geen afbeelding van de schakeling; het is te eenvoudig 8-)

Sketch

We declareren enkel een constante voor de sensor pin en een variabele voor de sensorwaarde. In de setup initialiseren we de pin als input en de seriële monitor voor het tonen van de resultaten.

In de loop functie lezen we met *analogRead(sensorPin)* de waarden in en tonen ze in de seriële monitor.

```
/*
Gebruik de TCRT5000 optische sensor module om zwart en wit te detecteren op een blad papier.
Toon de resulaten in de seriële monitor.
Experimenteer ook eens met de potentiometer om de gevoeligheid aan te passen.

*/
// constanten
const int sensorPin = A0; // Pin TCRT5000

// variabelen
int sensorWaarde = 0; // sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    // de pinnen initialiseren
    pinMode(sensorPin, INPUT);

    // Seriele monitor initialiseren
    Serial.begin(9600);
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    // lees de sensorwaarde
    sensorWaarde = analogRead(sensorPin);

    // toon sensorwaarde in seriële monitor
    Serial.print("sensorwaarde: ");
    Serial.println(sensorWaarde);

    // korte pauze
    delay(200);
}
```

The screenshot shows a terminal window titled "/dev/ttyACM0 (Arduino Uno)". The window contains the following text output:

```
sensorwaarde: 1023
sensorwaarde: 29
sensorwaarde: 29
sensorwaarde: 29
sensorwaarde: 29
sensorwaarde: 29
sensorwaarde: 29
```

At the bottom of the window, there are three buttons: "Autoscroll" (unchecked), "Geen regeleinde" (selected), and "9600 baud".

Project 19 : ST7735 TFT kleurenscherm

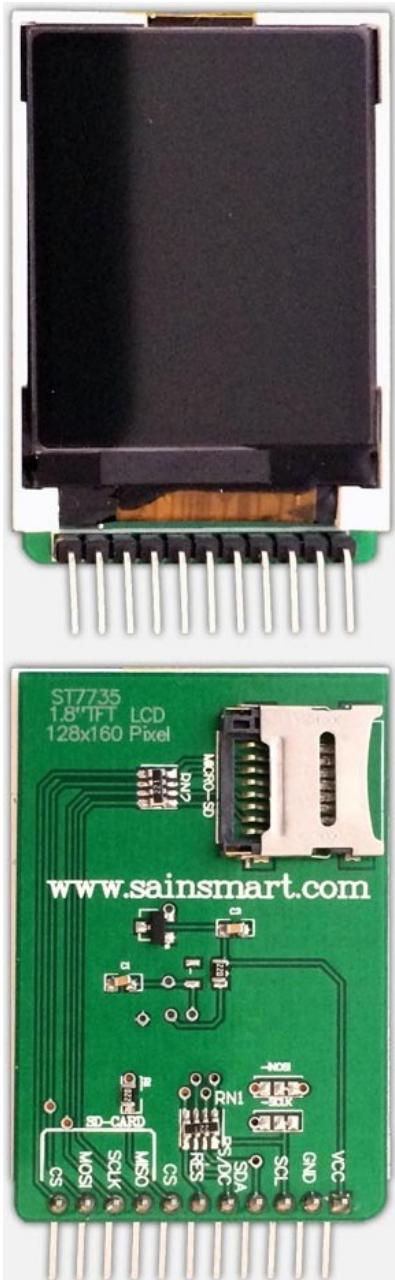
Opgave

Experimenteer met het ST7735 TFT kleurenscherm.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 ST7735 LCD scherm
- 1 microSD kaartje

ST7735



Specificaties

- 1.8" TFT kleurenscherm
- SPI interface (seriële synchrone data tussen master en slave)
- Resolutie : 128 x 160
- 262144 kleuren (18 bit)
- LED achtergrondverlichting
- Afmetingen module : 5.0 x 3.4 cm
- Afmetingen scherm : 3.5 x 2.8 cm
- 3.3 of 5V voedingsspanning
- slot voor micro SD kaart

Aansluitingen

Groep	Pin	Functie
Achtergrond verlichting	VCC	Voeding 3.3 of 5V
	GND	Massa
Scherm	SCL	Klok scherm
	SDA	Data naar scherm (SPI)
	RS/DC	Mode Commando of Data
	RES	Reset
	CS (TFT)	Chip Select scherm
SD kaart	MISO	Master In Slave Out (SPI) Lees data van SD kaart
	SCLK	Klok SD kaart (SPI)
	MOSI	Master Out Slave In (SPI) Schrijf data naar SD kaart
	CS (SD)	Chip Select SD kaart

Bibliotheken

Er zijn verschillende bibliotheken voor de aansturing van dit lcd scherm beschikbaar. Bij de standaard installatie van de Arduino software zijn deze reeds aanwezig :

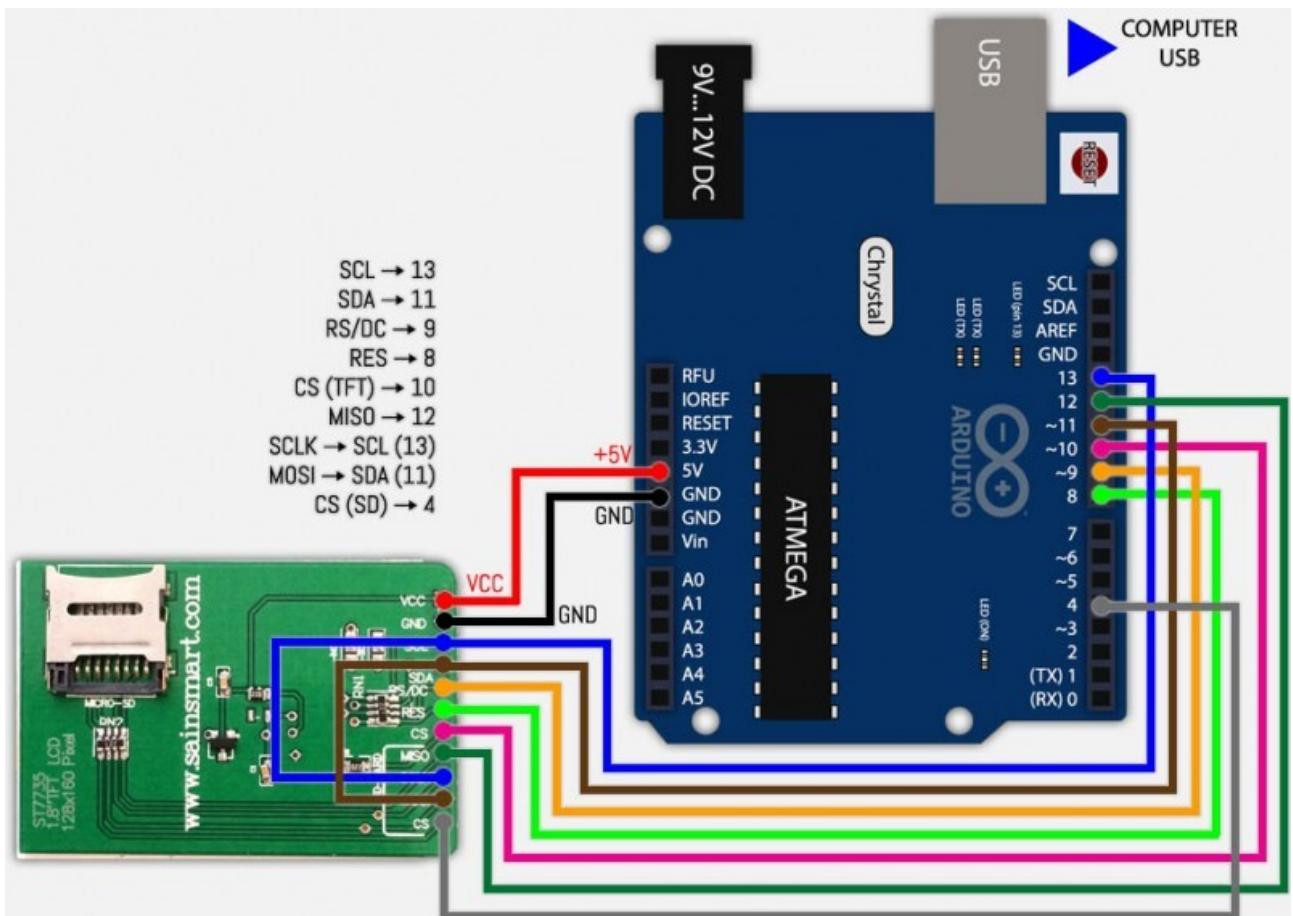
Bibliotheek	Functie
TFT	Grafische functies om lijnen, cirkels, tekst, ... te tekenen
SPI	Driver voor synchrone seriële data communicatie
SD	Functies voor het lezen van en schrijven naar een SD kaart

Dit zijn de beschikbare grafische functies van de TFT bibliotheek :

Functie	Uitleg
<i>Color565(rood, groen, blauw);</i>	Stelt een kleur samen uit <i>rood, groen en blauw</i>
<i>drawPixel(x, y, kleur);</i>	Tekent een pixel op coördinaten <i>(x,y)</i> in <i>kleur</i>
<i>drawLine(x0, y0, x1, y1, kleur);</i>	Tekent een lijn van <i>(x0,y0)</i> naar <i>(x1,y1)</i> in <i>kleur</i>
<i>drawFastHLine(x, y, breedte, kleur);</i>	Tekent een horizontale lijn van <i>(x,y)</i> van <i>breedte</i> in <i>kleur</i>
<i>drawFastVLine(x, y, hoogte, kleur);</i>	Tekent een verticale lijn van <i>(x,y)</i> van <i>hoogte</i> in <i>kleur</i>
<i>drawRect(x, y, breedte, hoogte, kleur);</i>	Tekent een rechthoek op <i>(x,y)</i> van <i>breedte</i> en <i>hoogte</i> in <i>kleur</i>
<i>fillRect(x, y, breedte, hoogte, kleur);</i>	Tekent een opgevulde rechthoek op <i>(x,y)</i> van <i>breedte</i> en <i>hoogte</i> in <i>kleur</i>
<i>drawCircle(x, y, straal, kleur);</i>	Tekent een cirkel op <i>(x,y)</i> met <i>straal</i> in <i>kleur</i>
<i>fillCircle(x, y, straal, kleur);</i>	Tekent een opgevulde cirkel op <i>(x,y)</i> met <i>straal</i> in <i>kleur</i>
<i>drawTriangle(x0, y0, x1, y1, x2, y2, kleur);</i>	Tekent een driehoek tussen <i>(x0,y0), (x1,y1)</i> en <i>(x2,y2)</i> in <i>kleur</i>
<i>fillTriangle(x0, y0, x1, y1, x2, y2, kleur);</i>	Tekent een opgevulde driehoek tussen <i>(x0,y0), (x1,y1)</i> en <i>(x2,y2)</i> in <i>kleur</i>
<i>drawRoundRect(x, y, breedte, hoogte, straal, kleur);</i>	Tekent een afgeronde rechthoek op <i>(x,y)</i> van <i>breedte</i> en <i>hoogte</i> in <i>kleur</i> . Afronden met <i>straal</i>
<i>fillRoundRect(x, y, breedte, hoogte, straal, kleur);</i>	Tekent een afgeronde opgevulde rechthoek op <i>(x,y)</i> van <i>breedte</i> en <i>hoogte</i> in <i>kleur</i> . Afronden met <i>straal</i>
<i>drawBitmap(x, y, *bitmap, breedte, hoogte, kleur);</i>	Tekent de <i>bitmap</i> op <i>(x,y)</i> met <i>breedte</i> en <i>hoogte</i> in <i>kleur</i>
<i>drawChar(x, y, karakter, kleur, achtergrondkleur, fontgrootte);</i>	Tekent <i>karakter</i> op <i>(x,y)</i> in <i>kleur</i> met <i>achtergrondkleur</i> in <i>fontgrootte</i>
<i>setCursor(x, y);</i>	Zet de cursor op positie <i>(x,y)</i>
<i>print(tekst);</i>	Tekent de <i>tekst</i>
<i>println(tekst);</i>	Tekent de <i>tekst</i> gevolgd door een nieuwe lijn

Functie	Uitleg
<code>setTextColor(kleur);</code>	Verander de tekstkleur naar <i>kleur</i>
<code>setTextColor(kleur, achtergrondkleur);</code>	Verander de tekstkleur naar <i>kleur</i> met <i>achtergrondkleur</i>
<code>setTextSize(fontgrootte);</code>	Verander de tekstgrootte naar <i>fontgrootte</i>
<code>setTextWrap(boolean);</code>	Zet tekstwrap <i>aan</i> (True) of <i>af</i> (False)
<code>setRotation(rotatie);</code>	Roteer het beeld <i>rotatie</i> (0=0°, 1=90°, 2=180°, 3=270°)
<code>fillScreen(kleur);</code>	Vul het scherm met <i>kleur</i>
<code>invertDisplay(boolean);</code>	Inverteer het scherm <i>wel</i> (True) of <i>niet</i> (False)

Breadboard schakeling



We verbinden de ST7735 met onze UNO op de volgende manier :

Pin ST7735	Functie	Pin UNO
VCC	Voeding 3.3 of 5V	5V
GND	Massa	GND
SCL	Klok scherm	13
SDA	Data naar scherm (SPI)	11
RS/DC	Mode Commando of Data	9
RES	Reset	8
CS (TFT)	Chip Select scherm	10
MISO	Master In Slave Out (SPI) – lees data van SD kaart	12
SCLK	Klok SD kaart (SPI)	(13)
MOSI	Master Out Slave In (SPI) – schrijf data naar SD kaart	(11)
CS (SD)	Chip Select SD kaart	4

Sketch 1

In deze sketch worden de grafische routines van de TFT bibliotheek gedemonstreerd.

We voegen deze bibliotheken toe aan onze sketch via het menu Schets – Bibliotheek importeren – <naam bibliotheek>

Bibliotheek	Functie
TFT	Grafische functies om lijnen, cirkels, tekst, ... te tekenen
SPI	Driver voor synchrone seriële data communicatie

We definiëren de constanten voor de gebruikte pinnen op onze UNO volgens de tabel hierboven, alsook enkele reeksen met rotatie- en kleurenstrings.

We declareren variabelen voor de basiskleuren alsook voor x- en y-coordinates en k voor kleur.

We declareren een object *scherm* van het type TFT van onze bibliotheek.

In de setup functie wordt het scherm geïnitialiseerd en worden de variabelen voor de basiskleuren opgevuld met de functie *scherm.Color565(rood, groen, blauw)*;

```

/*
  ST7735 1.8" TFT kleurenscherm

  Experimenteer met het ST7735 1.8" TFT kleurenscherm.

*/

// bibliotheken
#include <TFT.h>
#include <SPI.h>

// constanten
const int sclk = 13;
const int mosi = 11;
const int cs = 10;
const int dc = 9;
const int rst = 8;

const String rot[4] = { " 270 gr ", " 0 gr ", " 90 gr ", " 180 gr "};
const String kleur[8] = { " Rood ", " Groen ", " Blauw ", " Geel ", \
                        " Magenta ", " Cyaan ", " Wit ", " Zwart "};

// variabelen
int ST7735_zwart, ST7735_wit, ST7735_rood, ST7735_groen, ST7735_blaauw, \
ST7735_geel, ST7735_magenta, ST7735_cyaan;           // variabelen voor basiskleuren

int x, y;                                              // variabelen voor x, y
int k = 0;                                              // kleurindex

// object declaratie
TFT scherm = TFT(cs, dc, rst);                      // lcd scherm

void setup() {
  // de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
  // wanneer op de reset knop gedrukt wordt

  scherm.begin();                                       // initialisatie scherm

  ST7735_rood = scherm.Color565(0, 0, 255);          // initialisatie basiskleuren
  ST7735_groen = scherm.Color565(0, 255, 0);
  ST7735_blaauw = scherm.Color565(255, 0, 0);
  ST7735_geel = scherm.Color565(0, 255, 255);
  ST7735_magenta = scherm.Color565(255, 0, 255);
  ST7735_cyaan = scherm.Color565(255, 255, 0);
  ST7735_wit = scherm.Color565(255, 255, 255);
  ST7735_zwart = scherm.Color565(0, 0, 0);
}

```

In de loop functie worden achtereenvolgens volgende demo's gegeven :

- achtergrondkleuren (fillScreen)
- lettergroottes en letterkleuren (setTextSize, setTextColor)
- tekst overloop (setTextWrap)
- lijnen (drawLine)
- horizontale/verticale lijnen (drawFastHLine, drawFastVLine)
- rechthoeken/opgevulde rechthoeken (drawRect, fillRect)
- afgeronde rechthoeken/opgevulde afgeronde rechthoeken (drawRoundRect, fillRoundRect)
- driehoeken/opgevulde driehoeken (drawTriangle, fillTriangle)

- cirkels/opgevulde cirkels (drawCircle, fillCircle)
- scherm rotatie (setRotation)
- schermkleuren inverteren (invertDisplay)

```
// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

    scherm.setCursor(10, 50);                                // cursor positie
    scherm.setTextColor(ST7735_wit);                          // tekstkleur wit
    scherm.setTextSize(2);                                    // tekst grootte
    scherm.fillRect(ST7735_zwart);                           // zwarte achtergrond
    scherm.println("Achtergrond");                           // achtergrond
    scherm.println(" kleuren");                            // kleuren
    delay(2000);                                            // wacht

    for (int i=0; i<7; i++) {                                // lus achtergrondkleur
        scherm.setCursor(10, 50);                            // cursor positie
        scherm.setTextColor(ST7735_zwart,ST7735_wit);        // tekstkleur zwart
        scherm.fillRect(haalkleur(i));                      // kleur achtergrond
        scherm.println(kleur[i]);                            // toon kleurnaam
        delay(2000);                                         // wacht
    }

    scherm.setCursor(10, 50);                                // cursor positie
    scherm.setTextColor(ST7735_wit);                          // tekstkleur wit
    scherm.setTextSize(2);                                    // tekst grootte
    scherm.fillRect(ST7735_zwart);                           // achtergrond
    scherm.println("Fontsize");                            // tekst grootte
    delay(2000);                                            // wacht

    for (int f=1; f<11; f++) {                            // lus fontsize
        scherm.fillRect(ST7735_zwart);                     // achtergrond zwart
        scherm.setCursor(10, 40);                            // cursor positie
        k = volgendkleur();                                // volgend kleur
        scherm.setTextColor(haalkleur(k));                  // tekst kleur
        scherm.setTextSize(f);                            // tekst grootte f
        scherm.println(f);                                // print f
        delay(2000);                                         // wacht
    }

    scherm.setCursor(0, 0);                                // cursor positie
    scherm.setTextSize(1);                                // letter grootte
    scherm.setTextWrap(true);                            // zet tekstwrap aan
    k = volgendkleur();                                // volgend kleur
    scherm.setTextColor(haalkleur(k));                  // tekstkleur wit
    scherm.fillRect(ST7735_zwart);                         // zwarte achtergrond
    scherm.println("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec commodo i");
    scherm.setTextWrap(false);                            // zet tekstwrap af
    delay(2000);                                         // wacht

    scherm.setCursor(10, 50);                                // cursor positie
    scherm.setTextSize(2);                                // letter grootte
    scherm.setTextColor(ST7735_wit);                          // tekstkleur wit
    scherm.fillRect(ST7735_zwart);                           // zwarte achtergrond
    scherm.println("Lijnen");                            // Lijnen demo
}
```

```

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

k = volgendkleur();                         // volgend kleur
for (x=0; x<scherm.width(); x=x+5) {        // lus x LB
    scherm.drawLine(0, 0, x, scherm.height(), haalkleur(k));
}
k = volgendkleur();                         // volgend kleur
for (y=scherm.height(); y>0; y=y-5) {        // lus y LB
    scherm.drawLine(0, 0, scherm.width(), y, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

k = volgendkleur();                         // volgend kleur
for (x=0; x<scherm.width(); x=x+5) {        // lus x LO
    scherm.drawLine(0, scherm.height(), x, 0, haalkleur(k));
}
k = volgendkleur();                         // volgend kleur
for (y=0; y<scherm.height(); y=y+5) {        // lus y LO
    scherm.drawLine(0, scherm.height(), scherm.width(), y, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

k = volgendkleur();                         // volgend kleur
for (x=scherm.width(); x>0; x=x-5) {        // lus x RB
    scherm.drawLine(scherm.width(), 0, x, scherm.height(), haalkleur(k));
}
k = volgendkleur();                         // volgend kleur
for (y=scherm.height(); y>0; y=y-5) {        // lus y RB
    scherm.drawLine(scherm.width(), 0, 0, y, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

k = volgendkleur();                         // volgend kleur
for (x=scherm.width(); x>0; x=x-5) {        // lus x RO
    scherm.drawLine(scherm.width(), scherm.height(), x, 0, haalkleur(k));
}
k = volgendkleur();                         // volgend kleur
for (y=0; y<scherm.height(); y=y+5) {        // lus y RO
    scherm.drawLine(scherm.width(), scherm.height(), 0, y, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

```

```

for (y=0; y<scherm.height(); y=y+5) {           // lus y horizontaal
    k = volgendkleur();                         // volgend kleur
    scherm.drawFastHLine(0, y, scherm.width(), haalkleur(k));
}
for (x=0; x<scherm.width(); x=x+5) {           // lus x vertikaal
    k = volgendkleur();                         // volgend kleur
    scherm.drawFastVLine(x, 0, scherm.height(), haalkleur(k));
}

delay(2000);                                     // wacht
scherm.fillScreen(ST7735_zwart);                // zwarte achtergrond

scherm.setCursor(10, 50);                        // cursor positie
scherm.setTextSize(2);                          // letter grootte
scherm.setTextColor(ST7735_wit);                 // tekstkleur wit
scherm.fillScreen(ST7735_zwart);                // zwarte achtergrond
scherm.println("Rechthoeken");                  // rechthoeken

delay(2000);                                     // wacht
scherm.fillScreen(ST7735_zwart);                // zwarte achtergrond

for (y=0; y<scherm.height()/2; y=y+5) {         // lus y rechthoeken
    k = volgendkleur();                         // volgend kleur
    scherm.drawRect(y, y, scherm.width()-2*y, scherm.height()-2*y, haalkleur(k));
}

delay(2000);                                     // wacht
scherm.fillScreen(ST7735_zwart);                // zwarte achtergrond

for (y=0; y<scherm.height()/2; y=y+5) {         // lus y opgevulde rechthoeken
    k = volgendkleur();                         // volgend kleur
    scherm.fillRect(y, y, scherm.width()-2*y, scherm.height()-2*y, haalkleur(k));
}

delay(2000);                                     // wacht
scherm.fillScreen(ST7735_zwart);                // zwarte achtergrond

scherm.setCursor(10, 50);                        // cursor positie
scherm.setTextSize(2);                          // letter grootte
scherm.setTextColor(ST7735_wit);                 // tekstkleur wit
scherm.fillScreen(ST7735_zwart);                // zwarte achtergrond
scherm.println("Afgeronde");                   // afgeronde
scherm.println(" rechthoeken");                 // rechthoeken

delay(2000);                                     // wacht
scherm.fillScreen(ST7735_zwart);                // zwarte achtergrond

for (y=0; y<scherm.height()/2-10; y=y+5) {     // lus y rechthoeken
    k = volgendkleur();                         // volgend kleur
    scherm.drawRoundRect(y, y, scherm.width()-2*y, scherm.height()-2*y, 10, haalkleur(k));
}

```

```

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

for (y=0; y<scherm.height()/2-5; y=y+5) {    // lus y opgevulde rechthoeken
    k = volgendkleur();                      // volgend kleur
    scherm.fillRoundRect(y, y, scherm.width()-2*y, scherm.height()-2*y, 10, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

scherm.setCursor(10, 50);                    // cursor positie
scherm.setTextSize(2);                      // letter grootte
scherm.setTextColor(ST7735_wit);              // tekstkleur wit
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond
scherm.println("Cirkels");                  // cirkels

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

for (y=0; y<scherm.height()/2; y=y+5) {      // lus y cirkels
    k = volgendkleur();                      // volgend kleur
    scherm.drawCircle(scherm.width()/2, scherm.height()/2, scherm.height()/2-y, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

k = 0;
for (y=0; y<scherm.height()/2; y=y+5) {      // lus y opgevulde cirkels
    k = volgendkleur();                      // volgend kleur
    scherm.fillCircle(scherm.width()/2, scherm.height()/2, scherm.height()/2-y, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

scherm.setCursor(10, 50);                    // cursor positie
scherm.setTextSize(2);                      // letter grootte
scherm.setTextColor(ST7735_wit);              // tekstkleur wit
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond
scherm.println("Driehoeken");                // driehoeken

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

for (y=0; y<scherm.height()/2; y=y+5) {      // lus y driehoeken
    k = volgendkleur();                      // volgend kleur
    scherm.drawTriangle(scherm.width()/2, y, y, scherm.height()-y, scherm.width()-y, scherm.height()-y, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

for (y=0; y<scherm.height()/2; y=y+5) {      // lus y opgevulde driehoeken
    k = volgendkleur();                      // volgend kleur
    scherm.fillTriangle(scherm.width()/2, y, y, scherm.height()-y, scherm.width()-y, scherm.height()-y, haalkleur(k));
}

delay(2000);                                // wacht
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond

scherm.setCursor(10, 50);                    // cursor positie
scherm.setTextSize(2);                      // letter grootte
scherm.setTextColor(ST7735_wit);              // tekstkleur wit
scherm.fillRect(ST7735_zwart);               // zwarte achtergrond
scherm.println("Roteer");                   // roteer
scherm.println(" scherm");                  // scherm

for (x=0; x<4; x++) {                      // lus rotatie
    scherm.setCursor(10, 50);                  // cursor positie
    scherm.fillRect(ST7735_zwart);             // zwarte achtergrond
    scherm.setRotation(x);                   // roteer
    scherm.setTextColor(ST7735_zwart, ST7735_wit); // zwart op wit
    scherm.println(rot[x]);                  // toon rotatie graden
    delay(2000);                            // wacht
}

```

```

scherm.setRotation(1);                                // rotatie 90°
scherm.fillRect(ST7735_zwart);                      // zwarte achtergrond

scherm.setCursor(10, 50);                            // cursor positie
scherm.setTextSize(2);                             // letter grootte
scherm.setTextColor(ST7735_wit);                     // tekstkleur wit
scherm.fillRect(ST7735_zwart);                      // zwarte achtergrond
scherm.println("Inverteer");                        // inverteer
scherm.println(" scherm");                          // scherm

delay(2000);                                       // wacht
scherm.fillRect(ST7735_zwart);                      // zwarte achtergrond

scherm.setCursor(10, 50);                            // cursor positie
scherm.invertDisplay(true);                         // inverteer
scherm.println(" Inverted ");                      // toon tekst

delay(2000);                                       // wacht
scherm.fillRect(ST7735_zwart);                      // zwarte achtergrond

scherm.setCursor(10, 50);                            // cursor positie
scherm.invertDisplay(false);                        // normaal
scherm.println(" Normaal ");                       // toon tekst

delay(2000);                                       // wacht
scherm.fillRect(ST7735_zwart);                      // zwarte achtergrond
}

```

Tenslotte nog enkele functies om het volgende kleur te bepalen en de waarde van het kleur op te halen.

```
// functie om volgend kleur te bepalen
int volgendkleur() {
    k++;
    k=k%7;
    return k;
}

// functie waarde kleur ophalen
int haalkleur(int k) {
    int kleur;

    switch(k) { // kleur selecteren
        case 0:
            kleur = ST7735_rood;
            break;
        case 1:
            kleur = ST7735_groen;
            break;
        case 2:
            kleur = ST7735_blauw;
            break;
        case 3:
            kleur = ST7735_geel;
            break;
        case 4:
            kleur = ST7735_magenta;
            break;
        case 5:
            kleur = ST7735_cyaan;
            break;
        case 6:
            kleur = ST7735_wit;
            break;
        default:
            kleur = ST7735_wit;
    }

    return kleur; // return waarde
}
```

Sketch 2

In deze sketch experimenteren we met de micro SD kaart op de ST7735 module. We gebruiken hiervoor de volgende bibliotheken :

Bibliotheek	Functie
TFT	Grafische functies om lijnen, cirkels, tekst, ... te tekenen
SPI	Driver voor synchrone seriële data communicatie
SD	Functies voor het lezen van en schrijven naar een SD kaart

De bibliotheken voegen we toe via het menu Schets – Bibliotheek importeren – <naam bibliotheek>. We declareren de gebruikelijke constanten voor de gebruikte arduino UNO pinnen. We declareren enkele variabelen voor de kleuren wit en zwart. We declareren tevens een scherm object van de klasse TFT en een bitmap object van de klassen Pimage.

In de setup functie initialiseren we de kleuren wit en zwart. We starten het scherm en proberen de SD kaart te initialiseren. Als dit fout loopt stoppen we. Daarna proberen we onze bitmap file (parrot.bmp) op te laden. Als dit lukt tonen we nog eerst de afmetingen van de bitmap. Daarna wordt de bitmap op het scherm afgebeeld. De loop functie bevat geen code.

```
/*
  ST7735 1.8" TFT kleurenscherm

  Experimenteer met het ST7735 1.8" TFT kleurenscherm
  Afbeelden van een bitmap afbeelding van de SD kaart op het scherm

*/

// bibliotheken
#include <SPI.h>
#include <SD.h>
#include <TFT.h>

// constanten
const int sclk = 13;
const int mosi = 11;
const int cs = 10;
const int dc = 9;
const int rst = 8;
const int cs_sd = 4;

const int PixelBuffer = 20; // buffer voor pixels

// variabelen
int ST7735_wit, ST7735_zwart; // kleuren

// object declaratie
TFT scherm = TFT(cs, dc, rst); // lcd scherm
PImage logo; // bitmap afbeelding

void setup() {
  // de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
  // wanneer op de reset knop gedrukt wordt

  ST7735_wit = scherm.Color565(255, 255, 255); // wit
  ST7735_zwart = scherm.Color565(0, 0, 0); // zwart

  scherm.begin(); // initialisatie scherm
  scherm.setCursor(10, 50); // cursor positie
  scherm.setTextColor(ST7735_wit); // tekstkleur wit
  scherm.setTextSize(2); // tekst grootte
  scherm.fillScreen(ST7735_zwart); // zwarte achtergrond
  scherm.println("Init SDcard"); // Initialisatie SD kaart
  if (SD.begin(cs_sd)) {
    scherm.println(" OK"); // ok
  } else {
    scherm.println(" FOUT"); // fout
    return; // stoppen
  }
  delay(1000); // wachten
}
```

```

scherm.setCursor(10, 50);           // cursor positie
scherm.fillRect(ST7735_zwart);     // zwarte achtergrond
scherm.println("Zoeken");          // Zoeken
scherm.println(" parrot.bmp");     // bitmap afbeelding
logo = scherm.loadImage("parrot.bmp"); // bitmap openen
if (logo.isValid()) {              // ok
    scherm.println(" OK");
} else {                           // fout
    scherm.println(" FOUT");
    return;
}
delay(1000);                      // wachten

scherm.setCursor(10, 50);           // cursor positie
scherm.fillRect(ST7735_zwart);     // zwarte achtergrond
scherm.println(" Resolutie");      // Resolutie
scherm.print(" ");
scherm.print(logo.width());         // breedte
scherm.print(" x ");               // x
scherm.println(logo.height());      // hoogte
delay(1000);                      // wachten

scherm.setRotation(0);             // logo rechtop
scherm.image(logo, 0, 0);          // bitmap tekenen
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
}

```

Project 20 : Servomotor

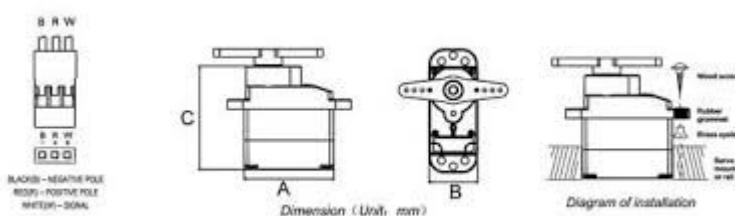
Opgave

Maak een slot met behulp van de servo motor en het piezo element. De piezo wordt in dit geval gebruikt als input element : de trillingen van het kloppen worden omgezet in elektrische spanning die door de analoge poort van de UNO kunnen worden gedetecteerd. Gebruik 3 leds om de status van het slot weer te geven : rood = gesloten, groen = open, geel = geldige klop. Gebruik een drukknop om het slot bij het begin te sluiten. Je kan het daarna terug openen met 3 geldige kloppen op de "deur".

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- 1 servomotor SM-S2390S
- 1 drukknop
- 1 piezo zoemer
- 1 rode led
- 1 groene led
- 1 gele led
- 1 capaciteit $100 \mu\text{F}$
- 3 weerstanden 220 ohm
- 1 weerstand 10K ohm
- 1 weerstand 1M ohm

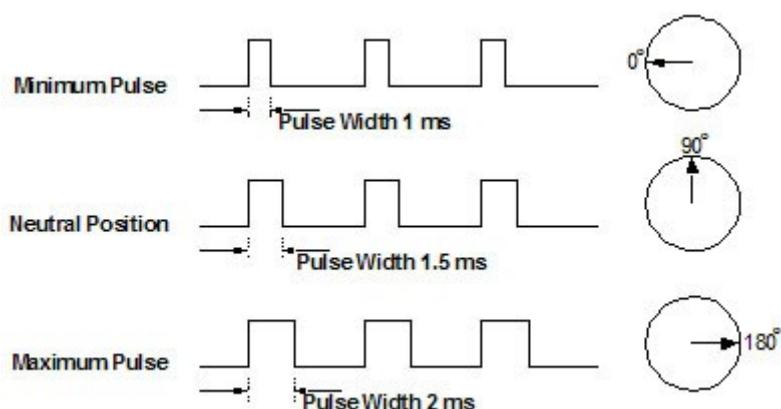
Servomotor SM-S2309S



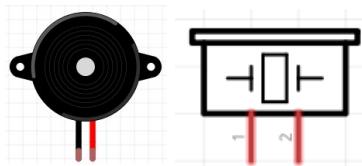
De SM-S2309S is een hoge precisie servomotor waarvan de rotor 180° kan draaien. De richting en hoeveelheid van rotatie wordt bepaald door de breedte van de elektrische pulsen (1-2 ms).

Aansluitingen

Pin	Functie
Rood	Vcc
Zwart	GND
Wit	Signaal



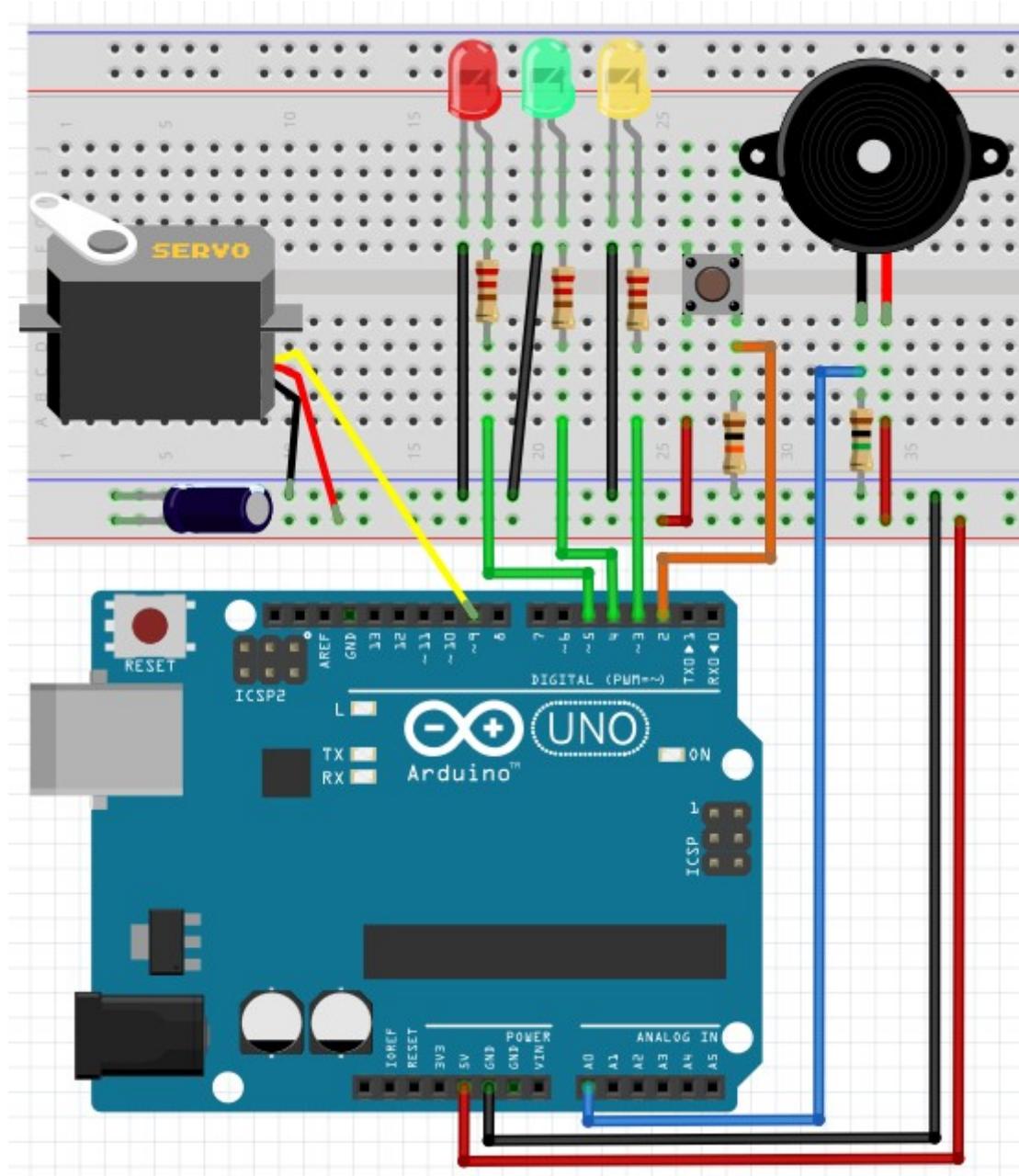
Piezo zoemer



Een piezo zoemer is een elektronisch component die elektrische energie omzet in mechanische energie in de vorm van trillingen. Het zijn sinusvormige geluidsgolven die typisch in de hoorbare frequenties van 20 Hz tot 20 kHz vallen.

Het element kan ook omgekeerd werken : trillingen omzetten in elektrische energie. Dit werkt het best door de platte kant tegen bijvoorbeeld een deur of wand te monteren. Doordat de elektrische signalen zeer klein zijn moet een hoge weerstand van 1M ohm gebruikt worden als spanningsdeler.

Breadboard schakeling



We verbinden de verschillende componenten als volgt met de UNO :

Component	Pin component	Pin UNO
SM-S2309S	rood	5V*
	zwart	GND*
	wit	9
Piezo zoemer	+	5V
	-	A0**
Drukknop	rechts	5V
	links	2***
Rode led	anode	5****
	kathode	GND
Groene led	anode	4****
	kathode	GND
Gele led	anode	3****
	kathode	GND

* capaciteit 100 µF om fluctuaties op te vangen

** spanningsdeler met 1M ohm weerstand

*** spanningsdeler met 10K ohm weerstand

**** via weerstand 220 ohm

Sketch

We laden de Servo bibliotheek op via het menu Schets – Bibliotheek importeren – Servo. Deze bibliotheek laat ons toe om op een eenvoudige manier de stand van de rotor arm van de servo te sturen, namelijk in graden.

We declareren constanten voor de gebruikte poorten van de servomotor, de piezo en de 3 leds, alsook de uiterste meetwaarden voor een geldige klop. Die kunnen we proefondervindelijk vaststellen via de seriële monitor.

Verder declareren we nog enkele variabelen voor de drukknop, de meetwaarde van het kloppen, de status van het servoslot en het aantal geldige kloppen. We declareren ook onze servomotor.

In de setup functie initialiseren we de servomotor, de piezo en de drukknop als input en de leds als output. We initialiseren tevens de seriële monitor. We zetten de servo in zijn neutrale stand (0°) en zetten de groene led aan en de rest uit. We tonen de status 'open' in de console.

```

/*
  Servo slot

Maak een slot met behulp van de servo motor en het piezo element. De piezo wordt in dit geval gebruikt
element : de trillingen van het kloppen worden omgezet in elektrische spanning die door de analoge po
kunnen worden gedetecteerd. Gebruik 3 leds om de status weer te geven : rood = gesloten, groen = open,
Gebruik een drukknop om het slot te sluiten. Je kan het terug openen met het juiste aantal geldige klo
*/



#include <Servo.h>

// constanten
const int piezo = A0;                                // analoge poort voor piezo
const int switchPin = 2;                             // digitale poort voor switch
const int geleLed = 3;                               // digitale poort voor gele led
const int groeneLed = 4;                            // digitale poort voor groene led
const int rodeLed = 5;                               // digitale poort voor rode led
const int servoPin = 9;                             // digitale poort voor servo

const int zachteKlop = 10;                           // zachte klop waarde
const int hardeKlop = 100;                          // harde klop waarde

// variabelen
int klopWaarde;                                     // klopwaarde
int switchWaarde;                                    // switchwaarde
boolean gesloten = false;                          // status slot
int aantalKloppen = 0;                            // aantal kloppen

// objecten
Servo servoslot;                                  // servo object

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {

    servoslot.attach(servoPin);                     // servo init

    pinMode(piezo, INPUT);                         // piezo inout
    pinMode(switchPin, INPUT);                    // switch input
    pinMode(geleLed, OUTPUT);                      // led output
    pinMode(groeneLed, OUTPUT);                   // led output
    pinMode(rodeLed, OUTPUT);                     // led output

    digitalWrite(geleLed, LOW);                    // led afzetten
    digitalWrite(groeneLed, LOW);                  // led afzetten
    digitalWrite(rodeLed, LOW);                    // led afzetten

    Serial.begin(9600);                           // seriële monitor
    servoslot.write(0);                           // servo 0°
    digitalWrite(groeneLed, HIGH);                // led afzetten
    Serial.println("OPEN");                        // bericht
}

}

```

Zolang de status van het slot open is, lezen we onze drukknop uit. Als die ingedrukt wordt 'sluiten' we onze deur : we draaien de servo naar 90°, zetten de groene led uit en de rode aan, passen onze gesloten variabele aan en updaten onze console.

Vanaf nu luisteren we op de analoge poort van de piezo zoemer om 3 geldige klopwaarden binnen te krijgen. We gebruiken hiervoor de functie *geldigeKlop(klopwaarde)* die een *True* teruggeeft bij een geldige en een *False* bij een ongeldige klopwaarde. Het resultaat wordt in de console getoond. Bij een geldige klop wordt ook eventjes de gele led aangezet voor visuele feedback.

Als 3 geldige kloppen zijn ontvangen 'openen' we onze deur : we draaien de servo naar 0°, zetten de groene led aan en de rode uit, passen de gesloten variabele aan en updaten de console.

```

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {

    if (not gesloten) {                                // open
        switchWaarde = digitalRead(switchPin);          // lees switch
        if (switchWaarde == HIGH) {                       // ingedrukt
            gesloten = true;                            // status aanpassen
            servoslot.write(90);                         // servo 90°
            digitalWrite(groeneLed, LOW);                // groen aanzetten
            digitalWrite(rodeLed, HIGH);                 // rood aanzetten
            Serial.println("GESLOTEN");                  // bericht
            delay(1000);                                // wachten
        }
    }

    if (gesloten) {                                    // gesloten
        klopWaarde = analogRead(piezo);                // lees piezo
        if (aantalKloppen < 3 && klopWaarde > 1) {
            if (geldigeKlop(klopWaarde)) {              // klop is ok
                aantalKloppen++;                         // tellen ophogen
            }
            Serial.print("Nog ");
            Serial.print(3-aantalKloppen);
            Serial.println(" kloppen te gaan");
        }
        if (aantalKloppen >= 3) {                      // aantal kloppen ok
            gesloten = false;                           // status aanpassen
            servoslot.write(0);                         // servo 0°
            digitalWrite(groeneLed, HIGH);               // groen aanzetten
            digitalWrite(rodeLed, LOW);                 // rood aanzetten
            Serial.println("OPEN");
            aantalKloppen = 0;                          // reset aantal kloppen
        }
    }
}

// functie om te controleren of de klop geldig is
boolean geldigeKlop (int waarde) {
    if (waarde > zachteKlop && waarde < hardeKlop) { // geldige klop
        digitalWrite(geleLed, HIGH);                   // geel aanzetten
        delay(50);                                    // even wachten
        digitalWrite(geleLed, LOW);                   // geel afzetten
        Serial.print("Geldige klop waarde ");
        Serial.println(waarde);
        return true;                                 // geldige returnwaarde
    } else {
        Serial.print("Ongeldige klop waarde ");
        Serial.println(waarde);
        return false;                                // geldige returnwaarde
    }
}

```

```
x + - /dev/ttyACM1 (Arduino Uno)
| Verzenden

Nog 0 kloppen te gaan
OPEN
GESLOTEN
Ongeldige klop waarde 2
Nog 3 kloppen te gaan
Geldige klop waarde 11
Nog 2 kloppen te gaan
Ongeldige klop waarde 4
Nog 2 kloppen te gaan
Geldige klop waarde 12
Nog 1 kloppen te gaan
Ongeldige klop waarde 4
Nog 1 kloppen te gaan
Geldige klop waarde 11
Nog 0 kloppen te gaan
OPEN

 Autoscroll Geen regeleinde ▾ 9600 baud ▾
```

Project 21 : Sainsmart 37 sensor kit

Sensor kit

Deze [sensor kit](#) is beschikbaar bij diverse leveranciers, maar ze hebben allemaal één ding gemeen : het totaal gebrek aan meegeleverde documentatie. Gelukkig is er op internet nogal wat informatie te vinden al is het soms van wisselende kwaliteit. Verder experimenteren resulteerde in dit meer uitgebreide project.



Overzicht sensors

- [KY001](#) – Digitale temperatuur sensor
- [KY002](#) – Trilling sensor
- [KY003](#) – Hall magnetisch veld sensor
- [KY004](#) – Drukknop
- [KY005](#) – Infrarode zender
- [KY006](#) – Passieve zoemer
- [KY008](#) – Laser diode
- [KY009](#) – Drie kleuren SMD led
- [KY010](#) – Licht onderbreking sensor
- [KY011](#) – Grote twee kleuren led
- [KY012](#) – Actieve zoemer
- [KY013](#) – Analoge temperatuur sensor
- [KY015](#) – Temperatuur en vochtigheid sensor
- [KY016](#) – Drie kleuren led

- [KY017](#) – Kwik schakelaar
- [KY018](#) – LDR sensor
- [KY019](#) – Relais
- [KY020](#) – Tilt schakelaar
- [KY021](#) – Kleine Reed schakelaar
- [KY022](#) – Infrarode ontvanger
- [KY023](#) – Mini X/Y joystick
- [KY024](#) – Lineaire Hall sensor
- [KY025](#) – Grote Reed schakelaar
- [KY026](#) – Vlammen sensor
- [KY027](#) – Magische led cup module
- [KY028](#) – Digitale temperatuur sensor
- [KY029](#) – Kleine twee kleuren led
- [KY031](#) – Schok sensor
- [KY032](#) – Obstakel detectie sensor
- [KY033](#) – Lijn volgende sensor
- [KY034](#) – Zeven kleuren led
- [KY035](#) – Analoge Hall sensor
- [KY036](#) – Aanraking sensor
- [KY037](#) – Grote microfoon
- [KY038](#) – Kleine microfoon
- [KY039](#) – Hartslag sensor
- [KY040](#) – Rotary Encoder

KY001 – Digitale temperatuur sensor

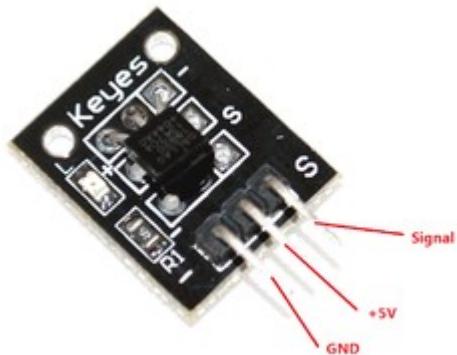
Opgave

Gebruik de sensor om de omgevingstemperatuur te bewaken en geef alarm bij het overschrijden van 25 °C. Toon de resultaten in de seriële monitor.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- KY001 temperatuur sensor
- 4.7K ohm weerstand

Sensor



Deze module bevat de [DS18B20](#) sensor. Dit is een digitale sensor in tegenstelling tot de analoge [TMP36](#) die veel gebruikt wordt voor temperatuurmetingen.

Iedere module heeft een uniek adres en er kunnen meerdere in parallel geschakeld worden. Ze kunnen dan individueel uitgelezen worden via hun adres.

Sketch

Voor het uitlezen van de sensorwaarden en het instellen van de alarm temperaturen wordt de [OneWire](#) en de [Dallas Temperature](#) bibliotheken gebruikt. We voegen de bibliotheken toe via het menu Schets – Bibliotheek importeren – <naam bibliotheek>.

Voor we verder kunnen moeten we het adres van onze DS18B20 sensor achterhalen.

Gelukkig bestaat er hier een eenvoudige voorbeeldschets voor. Die zit onder Bestand – Voorbeelden – OneWire – DS18x20_Temperature.

In de seriële console zien we dan het adres van onze sensor.

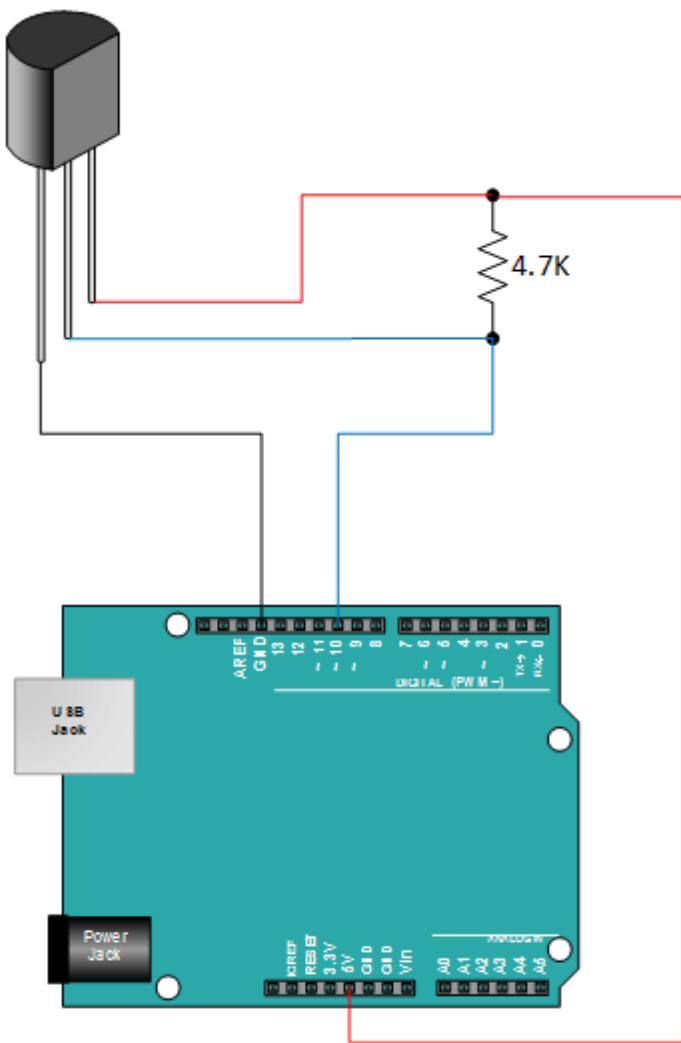
```
/dev/ttyACM0 (Arduino Uno)
No more addresses.

ROM = 28 FF 80 55 62 15 2 9F
Chip = DS18B20
Data = 1 3B 1 4B 46 7F FF C 10 E6  CRC=E6
Temperature = 19.69 Celsius, 67.44 Fahrenheit
No more addresses.

ROM = 28 FF 80 55 62 15 2 9F
Chip = DS18B20
Data = 1 3B 1 4B 46 7F FF C 10 E6  CRC=E6
Temperature = 19.69 Celsius, 67.44 Fahrenheit
No more addresses.

ROM = 28 FF 80 55 62 15 2 9F
Chip = DS18B20
Data = 1 3B 1 4B 46 7F FF C 10 E6  CRC=E6
```

Autoscroll Geen regeleinde 9600 baud



Vergeet de 4.7K ohm weerstand niet tussen de signaal pin en de voeding.

We declareren de signaal pin van onze temperatuur sensor KY001. Verder declareren we het object `oneWire` voor de seriële communicatie met de sensor en de `DallasTemperature sensors` reeks. In ons geval hebben we maar 1 sensor en die declareren we met
`DeviceAddress tempSensor = { adres };`
Verder declareren we nog enkele variabelen voor de alarm temperaturen en de sensorwaarden.

In de setup functie initialiseren we de signaal pin als input en de seriële console. Vervolgens initialiseren we de sensor en zetten de maximum en minimum alarm temperaturen alsook de gevoeligheid van de sensor.

In de loop functie lezen we de alarm temperaturen en de huidige temperatuur en tonen die in de seriële console.

```
x + - /dev/ttyACM0 (Arduino Uno) Verzenden
Min: 15 C Max: 25 C Temperatuur: 21.12 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 22.06 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 23.81 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 24.37 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 26.19 C Alarm
Min: 15 C Max: 25 C Temperatuur: 26.44 C Alarm
Min: 15 C Max: 25 C Temperatuur: 27.12 C Alarm
Min: 15 C Max: 25 C Temperatuur: 26.44 C Alarm
Min: 15 C Max: 25 C Temperatuur: 26.25 C Alarm
Min: 15 C Max: 25 C Temperatuur: 25.37 C Alarm
Min: 15 C Max: 25 C Temperatuur: 24.56 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 24.06 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 23.75 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 23.62 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 23.50 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 23.37 C Geen Alarm
Min: 15 C Max: 25 C Temperatuur: 23.31 C Geen Alarm

 Autoscroll Geen regeleinde 9600 baud
```

```

/*
    KY001 Digitale temperatuur sensor

    Gebruik de sensor om de omgevingstemperatuur te bewaken en geef alarm bij
    het overschrijden van 25 °C. Toon de resultaten in de seriële monitor.

    Benodigdheden :
    - Arduino UNO, breadbord, jumper kabeltjes
    - Digitale temperatuur sensor KY001
    - 4.7K ohm weerstand
*/

#include <OneWire.h>
#include <DallasTemperature.h>

// constanten
const int tempPin = 10;                                // KY001 signaal pin

// object declaraties
OneWire oneWire(tempPin);                            // OneWire object
DallasTemperature sensors(&oneWire);                  // KY001 sensor
DeviceAddress tempSensor = {0x28, 0xFF, 0x80, 0x55, 0x62, 0x15, 0x02, 0x9F};

// variabelen
int minTemp = 15;                                    // alarm min temperatuur
int maxTemp = 25;                                    // alarm max temperatuur
float tempC;                                         // temperatuur in °C
char alarmTemp;                                      // alarm temperaturen

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(tempPin, INPUT);                         // sensor is input
    Serial.begin(9600);                             // init seriële console
    sensors.begin();                               // init Dallas sensors
    sensors.setHighAlarmTemp(tempSensor, maxTemp); // alarm min temperatuur
    sensors.setLowAlarmTemp(tempSensor, minTemp);  // alarm max temperatuur
    sensors.setResolution(tempSensor, 12);          // gevoeligheid (9-12)
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    sensors.requestTemperaturesByAddress(tempSensor); // request temperatuur
    alarmTemp = sensors.getLowAlarmTemp(tempSensor); // lees min alarm
    Serial.print("Min: ");                           // toon
    Serial.print(alarmTemp, DEC);                   // min alarm
    Serial.print(" °C");                            // in °C
    alarmTemp = sensors.getHighAlarmTemp(tempSensor); // lees max alarm
    Serial.print("\tMax: ");                         // toon
    Serial.print(alarmTemp, DEC);                   // max alarm
    Serial.print(" °C");                            // in °C
    tempC = sensors.getTempC(tempSensor);           // lees temperatuur
    Serial.print("\tTemperatuur: ");                 // toon
    Serial.print(tempC, 2);                          // temperatuur
    Serial.print(" °C");                            // in °C
    if (sensors.hasAlarm(tempSensor)) {             // zijn er alarmen ?
        Serial.println("\tAlarm");                  // toon Alarm
    } else {
        Serial.println("\tGeen Alarm");            // toon Geen alarm
    }
    delay(1000);                                  // wachten
}

```

KY002 – Trilling sensor

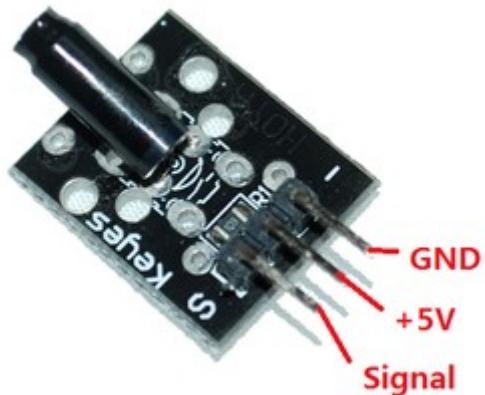
Opgave

Laat de ingebouwde led 13 flikkeren bij het detecteren van trillingen.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Trilling sensor KY002

Sensor



Deze sensor is een [SW-18015P](#). Die bestaat uit een koperen spoeltje dat gemonteerd is in een metalen buisje. Bij trillingen maken beiden contact en wordt het circuit gesloten.

De module heeft een ingebouwde weerstand die de signaal pin hoog houdt. Bij trillingen wordt de signaal pin laag.

Deze sensor werkt beter dan de tilt schakelaars gebaseerd op een metalen balletje in een buisje met contacten zoals de [KY020](#).

Sketch

We declareren constanten voor de interne led en de trilling sensor KY002 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de sensor. Als de sensor een trilling detecteert wordt de waarde laag (!) en zetten we de ingebouwde led aan. In het andere geval blijft de waarde hoog en zetten we de led af.

```

/*
  KY002 Trilling sensor

  Laat de ingebouwde led 13 flikkeren bij het detecteren van trillingen.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Trilling sensor KY002

*/

// constanten
const int led = 13;                                // interne led pin
const int trilling = 10;                            // KY002 signaal pin

// variabelen
int waarde;                                         // KY002 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(led, OUTPUT);                            // led is output
  pinMode(trilling, INPUT);                       // trilling sensor is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  waarde = digitalRead(trilling);                // lees sensor
  if (waarde == LOW) {                           // bij trilling
    digitalWrite(led, HIGH);                      // led aanzetten
  } else {                                       // anders
    digitalWrite(led, LOW);                       // led afzetten
  }
}

```

KY003 – Hall magnetisch veld sensor

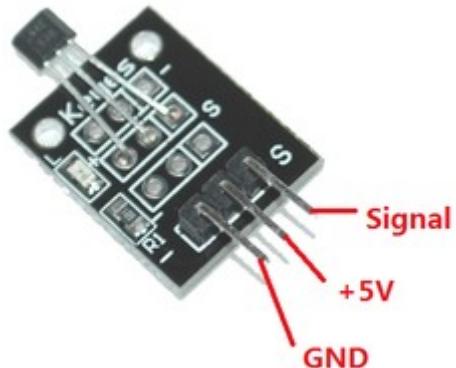
Opgave

Laat de ingebouwde led 13 flikkeren bij het detecteren van een magnetisch veld.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Hall magnetisch veld sensor KY003

Sensor



Deze module bevat een [A3144](#) Hall sensor die gebruikt wordt om magnetische velden te detecteren. Een ingebouwde weerstand houdt de signaal pin hoog. Bij detectie van een magnetisch veld wordt de signaal pin laag.

De polariteit van de magneet speelt hierbij een rol : de voorkant van de sensor moet een omgekeerde polariteit hebben tegenover de achterkant. De module heeft ook een ingebouwde led die oplicht bij detectie.

Deze kit bevat ook nog 2 andere Hall sensors die gebaseerd zijn op de SS49E, namelijk de lineaire Hall sensor [KY024](#) en de analoge Hall sensor [KY035](#).

Sketch

We declareren constanten voor de interne led en de Hall magnetisch veld sensor KY003 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de sensor. Als de sensor een magnetisch veld detecteert wordt de waarde laag (!) en zetten we de interne led aan. In het andere geval blijft de waarde hoog en zetten we de led af.

```

/*
  KY003 Hall magnetisch veld sensor

  Laat de ingebouwde led 13 flikkeren bij het detecteren van een magnetisch veld.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Hall magnetisch veld sensor KY003
  - Magneetje

*/

// constanten
const int led = 13;                      // interne led pin
const int emveld = 10;                    // KY003 signaal pin

// variabelen
int waarde;                            // KY003 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(led, OUTPUT);                  // led is output
  pinMode(emveld, INPUT);                // Hall sensor is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  waarde = digitalRead(emveld);          // lees sensor
  if (waarde == LOW) {                   // bij magnetisch veld
    digitalWrite(led, HIGH);              // led aanzetten
  } else {                                // anders
    digitalWrite(led, LOW);                // led afzetten
  }
}

```

KY004 – Drukknop

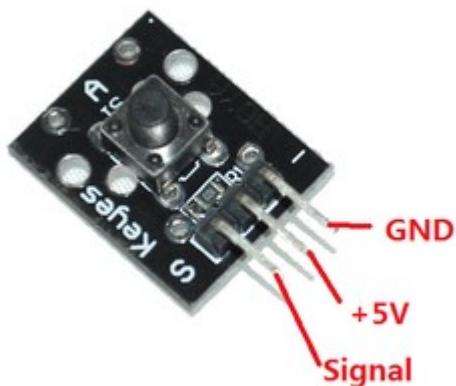
Opgave

Laat de ingebouwde led 13 oplichten wanneer de drukknop ingedrukt wordt.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Drukknop KY004

Sensor



Deze module bevat een drukknop. Een ingebouwde weerstand houdt de signaal pin hoog. Bij het indrukken van de drukknop wordt de signaal pin laag.

Sketch

We declareren constanten voor de interne led en de drukknop module KY004 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de module. Als de drukknop ingedrukt is wordt de waarde laag (!) en zetten we de interne led aan. In het andere geval blijft de waarde hoog en zetten we de led af.

```

/*
KY004 Drukknop

Laat de ingebouwde led 13 oplichten wanneer de drukknop ingedrukt wordt.

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- Drukknop KY004

*/

// constanten
const int led = 13;                                // interne led pin
const int drukknop = 10;                            // KY004 signaal pin

// variabelen
int waarde;                                       // KY004 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                         // led is output
    pinMode(drukknop, INPUT);                     // drukknop is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = digitalRead(drukknop);                // lees drukknop
    if (waarde == LOW) {                           // bij indrukken drukknop
        digitalWrite(led, HIGH);                   // led aanzetten
    } else {                                         // anders
        digitalWrite(led, LOW);                    // led afzetten
    }
}

```

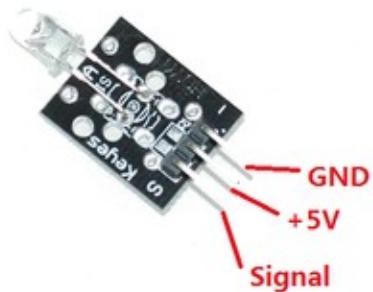
KY005 – Infrarode zender

Opgave

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Infrarode zender KY005

Sensor



Deze module bevat een infrarode zender waarmee een infrarood signaal kan worden verstuurd. Deze worden gebruikt in afstandsbedieningen voor tv's, DVD's, MP3 spelers, ...

Sketch

We voegen de IRRemote bibliotheek toe met Schets – Bibliotheek importeren – IRRemote. We declareren *IRzender* en in de loop functie sturen we de Sony power code om de tv af te zetten.

```
/*
 * KY005 Infrarode zender

 Zet je tv uit met de IR zender.

 Benodigdheden :
 - Arduino UNO, breadbord, jumper kabeltjes
 - Infrarode zender KY005
 - TV
 */

#include <IRremoteInt.h>
#include <IRremote.h>

// declaratie objecten
IRsend IRzender; // KY005 IR zender

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    for (int i = 0; i < 50; i++) {
        IRzender.sendSony (0xa90, 12); // Sony TV power code
        delay (40); // even wachten
    }
}
```

KY006 – Passieve zoemer

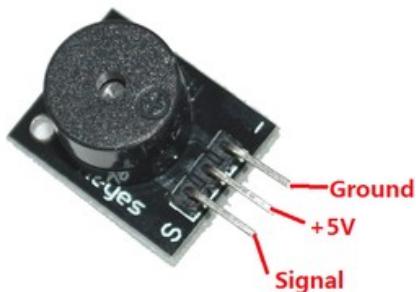
Opgave

Maak een sirene die van laag naar hoog en terug naar laag gaat.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Passieve zoemer KY006

Sensor



Deze module bevat een passieve piezo zoemer. Die heeft geen eigen oscillatie circuit en heeft een 'wisselspanning' nodig op de signaal pin om te werken.

Afhankelijk van de frequentie en duty cycle van de blokgolf kunnen verschillende tonen worden voortgebracht.

Sketch

We declareren constanten voor de signaal pin van de passieve zoemer KY006 en de lengte en duur van de noten die we gaan afspelen. We initialiseren de zoemer pin als output in de setup functie.

In de loop functie spelen we de noten af van laag naar hoog en terug naar laag. We gebruiken hiervoor de ingebouwde functie *tone(poort, frequentie, duur)*;

Deze produceert een blokgolf op *poort* met *frequentie* (in Hz) en *duur* (in msec). Een passieve zoemer heeft namelijk geen eigen oscillatiefrequentie en heeft dus een 'wisselspanning' nodig om te werken.

```

/*
  KY006 Passieve zoemer

  Maak een sirene die van laag naar hoog en terug naar laag gaat.

  Benodigdheden :
  - Arduino UNO, breadboard, jumper kabeltjes
  - Passieve zoemer KY006

*/

// constanten
const int zoemPin = 7;                      // KY006 signaal pin
const int lengteNoot = 125;                   // lengte van de noot (msec)
const int pauseNoot = 25;                     // pause tussen de noten (msec)

// variabelen
int noot;                                     // lus voor de noten

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(zoemPin, OUTPUT);                    // zoemer is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
  for (noot=25; noot<120; noot++) {          // laag naar hoog
    tone(zoemPin, 20*noot, lengteNoot);      // speel noot
    delay(pauseNoot);                        // pause noot
  }
  for (noot=120; noot>=25; noot--) {          // hoog naar laag
    tone(zoemPin, 20*noot, lengteNoot);      // speel noot
    delay(pauseNoot);                        // pause noot
}
}

```

KY008 – Laser diode

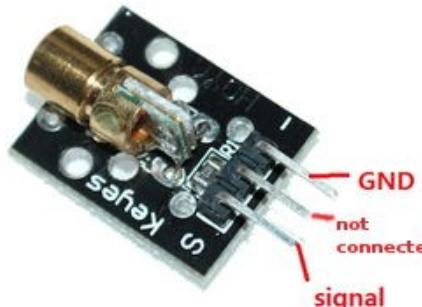
Opgave

Laat de ingebouwde led 13 oplichten wanneer de laser diode actief is.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Laser diode KY008

Sensor



Deze module bevat een rode laser zend diode. Deze werkt op een spanning van 5V, een stroom van 30 mA en een golflengte van 650 nm.

De **middelste** pin van deze module is **niet** aangesloten; de laser werkt wanneer 5V op de signaal pin wordt gezet.

Opgelet : KIJK NOOIT RECHTSTREEKS IN DE LASERSTRAAL !!!!

Sketch

We declareren constanten voor de ingebouwde led en de laser diode KY008 en initialiseren beide pinnen als output in de setup functie.

In de loop functie zetten we de laser diode en de interne led afwisselend aan en af gedurende 1 seconde.

```

/*
KY008 Laser diode

Laat de ingebouwde led 13 oplichten wanneer de laser diode actief is.
OPGELET : KIJK NOoit RECHTSTREEKS IN DE LASERSTRAAL !!!!!!

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- Laser diode module KY008

*/

// constanten
const int led = 13;                                // interne led pin
const int laser = 10;                             // KY008 signaal pin

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                      // led is output
    pinMode(laser, OUTPUT);                     // laser is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    digitalWrite(laser, HIGH);                // laser aanzetten
    digitalWrite(led, HIGH);                  // led aanzetten
    delay(1000);                            // wacht 1 sec
    digitalWrite(laser, LOW);                 // laser afzetten
    digitalWrite(led, LOW);                  // led afzetten
    delay(1000);                            // wacht 1 sec
}

```

KY009 – Drie kleuren SMD led

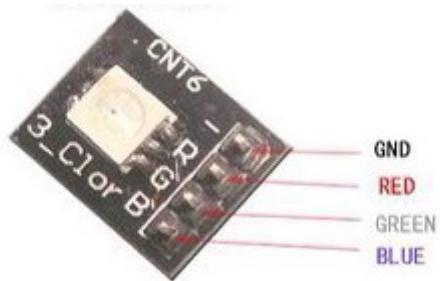
Opgave

Laat de drie kleuren led verschillende kleuren tonen door de rode, groene en blauwe pin aan te sturen met verschillende waarden. Gebruik hiervoor PWM poorten op je Arduino.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Drie kleuren SMD led KY009

Sensor



Dit is de [LL-R5050RGBC](#) drie kleuren SMD led module waarmee iedere kleur als een combinatie van verschillende intensiteiten (0 tot 255 decimaal, 00 tot FF hexadecimaal) van Rood, Groen en Blauw kan worden gemaakt.

Sketch

We declareren constanten voor de RGB pinnen van de drie kleuren led KY009 en initialiseren alle pinnen als output in de setup functie.

In de loop functie tonen we afwisselend alle HTML kleuren via de functie *kleur()*.

```

/*
KY009 Drie kleuren SMD led

Laat de drie kleuren led verschillende kleuren tonen door de
rood, groen en blauw pinnen aan te sturen met verschillende waarden.
OPGELET : Gebruik hiervoor PWM poorten op de Arduino.

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- Drie kleuren SMD led KY009

*/

// constanten
const int rPin = 9;                                // KY009 rode pin
const int gPin = 10;                               // KY009 groene pin
const int bPin = 11;                               // KY009 blauwe pin

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(rPin, OUTPUT);                         // rood is output
    pinMode(gPin, OUTPUT);                         // groen is output
    pinMode(bPin, OUTPUT);                         // blauw is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    // HTML kleuren
    kleur(0xFF, 0x00, 0x00);                      // rood
    kleur(0x00, 0xFF, 0x00);                      // limoen
    kleur(0x00, 0x00, 0xFF);                      // blauw
    kleur(0xFF, 0x00, 0xFF);                      // magenta
    kleur(0xFF, 0xFF, 0x00);                      // geel
    kleur(0x00, 0xFF, 0xFF);                      // cyaan
    kleur(0xFF, 0xFF, 0xFF);                      // wit
    kleur(0xC0, 0xC0, 0xC0);                      // zilver
    kleur(0x80, 0x00, 0x00);                      // bordeaux
    kleur(0x00, 0x80, 0x00);                      // groen
    kleur(0x00, 0x00, 0x80);                      // marine
    kleur(0x80, 0x00, 0x80);                      // purper
    kleur(0x80, 0x80, 0x00);                      // olijf
    kleur(0x00, 0x80, 0x80);                      // groenblauw
    kleur(0x80, 0x80, 0x80);                      // grijs
}

// functie om de kleur in te stellen
void kleur(int rood, int groen, int blauw) {
    analogWrite(rPin, rood);                       // rood
    analogWrite(gPin, groen);                      // groen
    analogWrite(bPin, blauw);                      // blauw
    delay(1000);                                 // wacht 1 sec
}

```

KY010 – Licht onderbreking sensor

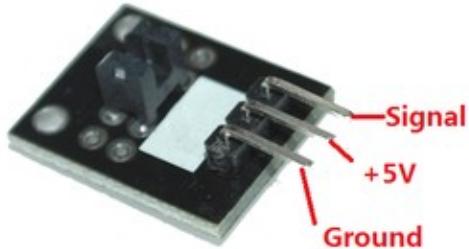
Opgave

Laat de ingebouwde led 13 oplichten wanneer het licht onderbroken wordt in de sensor.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Licht onderbreking sensor KY010

Sensor



Deze module is een licht onderbreking sensor module.
Wanneer er zich een voorwerp bevindt in de U-vormige uitsparing van de sensor, dan wordt de signaal pin hoog.

Opgelet : deze module is verkeerd gelabeld : signaal pin en – pin zijn omgewisseld !

Sketch

We declareren constanten voor de ingebouwde led en de licht onderbreking sensor KY010 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie zetten we de interne led oplichten wanneer het licht onderbroken wordt in de sensor (voorwerp tussen U-vormige uitsparing). In dit geval wordt de signaal pin hoog.

```

/*
  KY010 Licht onderbreking sensor

  Laat de ingebouwde led 13 oplichten wanneer het licht onderbroken wordt in de sensor.
  OPGELET : deze module is verkeerd gelabeld : signaal pin en - pin zijn omgewisseld !!

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Licht onderbreking sensor KY010

*/

// constanten
const int led = 13;                      // interne led pin
const int onderbreking = 10;                // KY010 signaal pin

// variabelen
int waarde;                                // KY010 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                  // led is output
    pinMode(onderbreking, INPUT);          // onderbreking sensor is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = digitalRead(onderbreking);    // lees sensor
    if (waarde == HIGH) {                   // bij onderbreking
        digitalWrite(led, HIGH);            // led aanzetten
    } else {                               // anders
        digitalWrite(led, LOW);             // led afzetten
    }
}

```

KY011 – Grote twee kleuren led

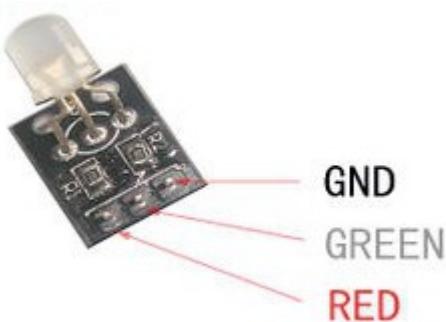
Opgave

Laat de twee kleuren led afwisselend groen en rood in- en uitfaden.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Grote twee kleuren led KY011

Sensor



Dit is een twee kleuren led module. Afhankelijk van de signalen op de rode en groene pin brandt de led rood, groen of een mengeling van deze kleuren.

Er is ook een kleine twee kleuren led [KY029](#).

Opgelet : de labeling van de pinnen op de module is verkeerd.
– = GND / midden = rood / signaal = groen

Sketch

We declareren constanten voor de rode en groene pinnen van de twee kleuren led KY011 en initialiseren beide pinnen als output in de setup functie.

In de loop functie laten we de twee kleuren led afwisselend groen en rood in- en uitfaden.

```

/*
  KY011 Grote twee kleuren led

  Laat de twee kleuren led afwisselend groen en rood in- en uitfaden.
  OPGELET : de pin layout van deze module is als volgt :
    - = GND / midden = rood / signaal = groen

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Twee kleuren led KY011

*/

// constanten
const int rPin = 10;                      // KY011 rode pin
const int gPin = 11;                      // KY011 groene pin

// variabelen
int waarde;                                // KY011 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(rPin, OUTPUT);                    // rode pin is output
  pinMode(gPin, OUTPUT);                    // groene pin is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  for (waarde=255; waarde>0; waarde--) {
    analogWrite(gPin, waarde);              // groen
    analogWrite(rPin, 255-waarde);          // rood
    delay(15);                            // even wachten
  }
  for (waarde=0; waarde<255; waarde++) {
    analogWrite(gPin, waarde);              // groen
    analogWrite(rPin, 255-waarde);          // rood
    delay(15);                            // even wachten
}
}

```

KY012 – Actieve zoemer

Opgave

Laat de zoemer afwisselend geluid produceren.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Actieve zoemer KY012

Sensor



Een actieve zoemer heeft zijn eigen oscillatiecircuit en produceert een geluid met een vaste frequentie. Deze kan niet aangepast worden. Voor verschillende tonen met variabele frequenties moet je een passieve zoemer gebruiken zoals de [KY006](#).

Sketch

We declareren een constante voor de signaal pin van de actieve zoemer module KY012. We initialiseren de zoemer pin als output in de setup functie. In de loop functie zetten we de signaal pin afwisselend hoog en laag gedurende 50 msec.

```
/*
KY012 Actieve zoemer

Laat de zoemer afwisselend geluid produceren.
OPGELET : de pin lay-out van deze module is als volgt :
- = GND / midden = niet aangesloten / signaal = 5V

Benodigdheden :
- Arduino UNO, breadboard, jumper kabeltjes
- Actieve zoemer KY012

*/
// constanten
const int zoemPin = 7; // KY012 signaal pin

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(zoemPin, OUTPUT); // zoemer is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    digitalWrite(zoemPin, HIGH); // geluid aan
    delay(50); // voor 50 msec
    digitalWrite(zoemPin, LOW); // geluid uit
    delay(50); // voor 50 msec
}
```

KY013 – Analoge temperatuur sensor

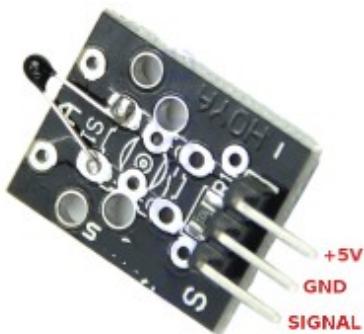
Opgave

Meet de omgevingstemperatuur met de sensor en toon de resultaten in de seriële console.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Analoge temperatuur sensor KY013

Sensor

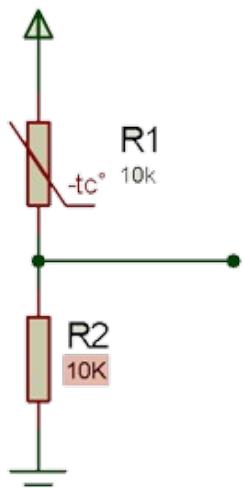


Dit is een analoge temperatuur sensor bestaande uit een spanningsdeler van een 10K ohm NTC (negatieve temperatuur coëfficiënt) thermistor met een 10K ohm weerstand.

De weerstand van de thermistor daalt als de temperatuur stijgt.

De spanning op de signaal pin kan als volgt berekend worden :

$$V_{out} = \frac{R_2}{R_1 + R_2} * V_{in}$$



In ons geval zal de spanning op de signaal pin dus stijgen bij een stijgende temperatuur omdat de waarde van de weerstand R1 zal dalen.

Opgelet : deze module is verkeerd gelabeld !

Links = signaal / Midden = GND / Rechts = 5V

Een andere variant van deze temperatuur sensor is de digitale temperatuur sensor [KY028](#).

Sketch

We declareren een constante voor de signaal pin van de analoge temperatuur sensor module KY013. We initialiseren de signaal pin als input en starten de seriële console in de setup functie.

In de loop functie lezen we de waarde van de sensor en zetten deze om naar temperatuur met de functie *thermistor(waarde)*; Deze gebruikt de logaritme functie uit de ingebouwde *math* bibliotheek van Arduino. We tonen de resultaten in de seriële console.

x + - /dev/ttyACM0 (Arduino Uno) Verzenden

```
Sensor: 452 Temperatuur: 19.72 C
Sensor: 508 Temperatuur: 24.64 C
Sensor: 558 Temperatuur: 29.16 C
Sensor: 576 Temperatuur: 30.83 C
Sensor: 584 Temperatuur: 31.58 C
Sensor: 588 Temperatuur: 31.96 C
Sensor: 591 Temperatuur: 32.24 C
Sensor: 593 Temperatuur: 32.43 C
Sensor: 579 Temperatuur: 31.11 C
Sensor: 565 Temperatuur: 29.80 C
Sensor: 553 Temperatuur: 28.70 C
Sensor: 542 Temperatuur: 27.70 C
Sensor: 533 Temperatuur: 26.88 C
Sensor: 525 Temperatuur: 26.16 C
```

Autoscroll Geen regeleinde ▾ 9600 baud ▾

```

/*
KY013 Analoge temperatuur sensor

Meet de omgevingstemperatuur met de sensor en toon de resultaten
in de seriële console.
OPGELET : Foute labeling module : midden = GND / - = 5V !!!!

Benodigdheden :
- Arduino UNO, breadboard, jumper kabeltjes
- Analoge temperatuur sensor KY013
*/

#include <math.h>

// constanten
const int sensorPin = A5; // signaal pin temperatuur sensor

// variabelen
int waarde; // KY013 sensorwaarde
double temp; // temperatuur

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(sensorPin, INPUT); // temperatuur is output
    Serial.begin(9600); // seriële console initialiseren
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    waarde = analogRead(sensorPin); // lees sensor
    temp = thermistor(waarde); // waarde omzetten naar temperatuur
    Serial.print("Sensor: ");
    Serial.print(waarde); // toon sensor waarde
    Serial.print("\tTemperatuur: ");
    Serial.print(temp); // toon temperatuur
    Serial.println(" C"); // in graden Celsius
    delay(1000); // even wachten
}

// de thermistor functie doet de omzetting van de sensorwaarde naar temperatuur
double thermistor(int waarde) {
    double Temp;
    Temp = log(10000.0 * ((1024.0 / waarde) - 1));
    Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp)) * Temp);
    Temp = Temp - 273.15; // Converteer Kelvin naar Celcius
    return Temp;
}

```

KY015 – Temperatuur en vochtigheid sensor

Opgave

Toon temperatuur en vochtigheid in de seriële console.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Temperatuur en vochtigheid sensor KY015

Sensor



Deze [DHT11](#) temperatuur en vochtigheid sensor levert een gekalibreerde digitale output van temperatuur- en vochtigheidswaarden via een ingebouwde 8-bit microcontroller.

Sketch

Voor deze test maken we gebruik van de Adafruit [DHT sensor library](#). Hierdoor wordt de ingewikkelde communicatie met de DHT11 sensor met ingebouwde microcontroller een stuk eenvoudiger :-)

We voegen de bibliotheek toe via het menu Schets – Bibliotheek importeren – DHT sensor library.

We declareren een constante voor de signaal pin van de temperatuur en vochtigheid sensor module KY015. We declareren onze sensor met *DHT sensor(dhtpin, dhttype);*

In de setup functie initialiseren we de sensor en de seriële console voor het tonen van de resultaten.

In de loop functie lezen we achtereenvolgens de vochtigheid en de temperatuur in. We berekenen vervolgens de gevoelstemperatuur met de bibliotheek functie *computeHeatIndex()*. Alle waarden worden dan in de seriële console getoond.

Opgelet : de DHT11 is een trage sensor. Daarom bouwen we een 2 seconde vertraging in tussen iedere meting.

```
x + - /dev/ttyACM0 (Arduino Uno) Verzenden  
Vochtigheid: 58.00 % Temperatuur: 22.00 C Gevoelstemperatuur: 21.77 C  
Vuchtigheid: 57.00 % Temperatuur: 23.00 C Gevoelstemperatuur: 22.84 C  
Vuchtigheid: 58.00 % Temperatuur: 22.00 C Gevoelstemperatuur: 21.77 C  
Vuchtigheid: 57.00 % Temperatuur: 23.00 C Gevoelstemperatuur: 22.84 C  
Vuchtigheid: 58.00 % Temperatuur: 24.00 C Gevoelstemperatuur: 23.97 C  
Vuchtigheid: 59.00 % Temperatuur: 24.00 C Gevoelstemperatuur: 24.00 C  
Vuchtigheid: 59.00 % Temperatuur: 25.00 C Gevoelstemperatuur: 25.10 C  
Vuchtigheid: 60.00 % Temperatuur: 25.00 C Gevoelstemperatuur: 25.12 C  
Vuchtigheid: 61.00 % Temperatuur: 26.00 C Gevoelstemperatuur: 26.96 C  
Vuchtigheid: 61.00 % Temperatuur: 27.00 C Gevoelstemperatuur: 28.15 C  
Vuchtigheid: 61.00 % Temperatuur: 27.00 C Gevoelstemperatuur: 28.15 C  
Vuchtigheid: 63.00 % Temperatuur: 26.00 C Gevoelstemperatuur: 27.04 C  
Vuchtigheid: 63.00 % Temperatuur: 26.00 C Gevoelstemperatuur: 27.04 C  
Vuchtigheid: 63.00 % Temperatuur: 25.00 C Gevoelstemperatuur: 25.20 C  
Vuchtigheid: 63.00 % Temperatuur: 25.00 C Gevoelstemperatuur: 25.20 C  
Vuchtigheid: 64.00 % Temperatuur: 24.00 C Gevoelstemperatuur: 24.13 C  
 Autoscroll Geen regeleinde ▾ 9600 baud ▾
```

```

/*
KY015 Temperatuur en vochtigheid sensor

Toon temperatuur en vochtigheid in de seriële console.

Benodigdheden :
- Arduino UNO, breadboard, jumper kabeltjes
- Temperatuur en vochtigheid sensor KY015

*/

#include <DHT.h>                                // DHTxx bibliotheek
#define DHTTYPE DHT11                            // wij gebruiken een DHT11

// constanten
const int dht11Pin = 9;                          // KY015 signaal pin

// object declaratie
DHT sensor(dht11Pin, DHTTYPE);

// variabelen
float vochtigheid;                             // vochtigheid
float temperatuur;                            // temperatuur (°C)
float gevoelstemp;                           // gevoelstemperatuur (°C)

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    Serial.begin(9600);                         // init seriële console
    sensor.begin();                            // init KY015 sensor
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    delay(2000);                               // even wachten - trage sensor !
    vochtigheid = sensor.readHumidity();       // lees vochtigheid
    temperatuur = sensor.readTemperature();    // lees temperatuur
    // bereken gevoelstemperatuur (parameter 3 : false voor °C, true voor °F)
    gevoelstemp = sensor.computeHeatIndex(temperatuur, vochtigheid, false);
    Serial.print("Vochtigheid: ");             // toon
    Serial.print(vochtigheid);                 // vochtigheid
    Serial.print(" %");                       // in procent
    Serial.print("\tTemperatuur: ");           // toon
    Serial.print(temperatuur);                 // temperatuur
    Serial.print(" C");                        // in Celsius
    Serial.print("\tGevoelstemperatuur: ");     // toon
    Serial.print(gevoelstemp);                 // gevoelstemperatuur
    Serial.println(" C");                      // in Celsius
}

```

KY016 – Drie kleuren led

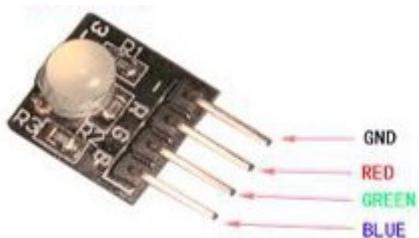
Opgave

Laat de drie kleuren led verschillende kleuren tonen door de rode, groene en blauwe pin aan te sturen met verschillende waarden. We gebruiken hiervoor PWM poorten op de Arduino.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Drie kleuren led KY016

Sensor



Dit is een drie kleuren led module waarmee iedere kleur als een combinatie van verschillende intensiteiten van Rood, Groen en Blauw kan worden gemaakt.

Sketch

We declareren constanten voor de RGB pinnen van de drie kleuren led KY016 en initialiseren alle pinnen als output in de setup functie.

In de loop functie tonen we afwisselend alle HTML kleuren via de functie *kleur()*.

```

/*
  KY016 Drie kleuren led

  Laat de drie kleuren led verschillende kleuren tonen door de
  rood, groen en blauw pin aan te sturen met verschillende waarden.
  OPGELET : Gebruik hiervoor PWM poorten op de Arduino.

  Benodigdheden :
  - Arduino UNO, breadboard, jumper kabeltjes
  - Drie kleuren led module KY016

*/

// constanten
const int rPin = 9;                      // KY016 rode pin
const int gPin = 10;                      // KY016 groene pin
const int bPin = 11;                      // KY016 blauwe pin

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(rPin, OUTPUT);                  // rood is output
  pinMode(gPin, OUTPUT);                  // groen is output
  pinMode(bPin, OUTPUT);                  // blauw is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  // HTML kleuren
  kleur(0xFF, 0x00, 0x00);                // rood
  kleur(0x00, 0xFF, 0x00);                // limoen
  kleur(0x00, 0x00, 0xFF);                // blauw
  kleur(0xFF, 0x00, 0xFF);                // magenta
  kleur(0xFF, 0xFF, 0x00);                // geel
  kleur(0x00, 0xFF, 0xFF);                // cyaan
  kleur(0xFF, 0xFF, 0xFF);                // wit
  kleur(0xC0, 0xC0, 0xC0);                // zilver
  kleur(0x80, 0x00, 0x00);                // bordeaux
  kleur(0x00, 0x80, 0x00);                // groen
  kleur(0x00, 0x00, 0x80);                // marine
  kleur(0x80, 0x00, 0x80);                // purper
  kleur(0x80, 0x80, 0x00);                // olijf
  kleur(0x00, 0x80, 0x80);                // groenblauw
  kleur(0x80, 0x80, 0x80);                // grijs
}

// functie om de kleur in te stellen
void kleur(int rood, int groen, int blauw) {
  analogWrite(rPin, rood);                // rood
  analogWrite(gPin, groen);                // groen
  analogWrite(bPin, blauw);                // blauw
  delay(1000);                           // wacht 1 sec
}

```

KY017 – Kwik schakelaar

Opgave

Laat de ingebouwde led 13 branden bij het sluiten van de schakelaar.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Kwik schakelaar KY017

Sensor

Dit is een kwik schakelaar. Afhankelijk van de tilt positie komt de signaal pin hoog of laag.



Andere schakelaars met gelijkaardige werking zijn de trilling sensor [KY002](#) en de tilt schakelaar [KY020](#).

Sketch

We declareren constanten voor de interne led en de kwik schakelaar module KY017 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de module. Als de kwik schakelaar gesloten is (verticale positie) is de waarde op de signaal pin laag (!) en zetten we de interne led aan. In het andere geval (horizontale positie) is de waarde hoog en zetten we de led af.

```

/*
  KY017 Kwik schakelaar

  Laat de ingebouwde led 13 branden bij het sluiten van de schakelaar.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Kwik schakelaar KY017

*/

// constanten
const int led = 13;                      // interne led pin
const int switchPin = 8;                   // KY017 signaal pin

// variabelen
int waarde;                             // KY017 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(led, OUTPUT);                  // led is output
  pinMode(switchPin, INPUT);             // schakelaar is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  waarde = digitalRead(switchPin);       // lees sensor
  if (waarde == LOW) {                  // bij sluiten schakelaar
    digitalWrite(led, HIGH);            // led aanzetten
  } else {                            // anders
    digitalWrite(led, LOW);             // led afzetten
  }
}

```

KY018 – LDR sensor

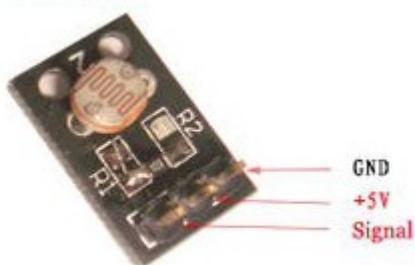
Opgave

Laat de ingebouwde led 13 sneller knipperen naarmate het donkerder wordt.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- LDR sensor KY018

Sensor



De [GL55](#) LDR sensor is eigenlijk een licht afhankelijke variabele weerstand. In het Engels staat LDR voor *Light Dependant Resistor*.

Hoe meer licht op de bovenkant van de sensor valt, hoe lager de weerstand wordt.

Sketch

We declareren constanten voor de ingebouwde led en de LDR sensor KY018 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de sensor uit met *analogRead(sensorPin)*. Deze geeft een waarde tussen 0 en 1023. Hoe meer licht, hoe lager de waarde. We laten we de interne led flikkeren met een frequentie met behulp van volgende delay functie :

```
delay(1024 - sensorwaarde);
```

Hierdoor gaat de led rapper flikkeren in een donkere omgeving.

```

/*
KY018 LDR sensor

Laat de ingebouwde led 13 sneller knipperen naarmate het donkerder wordt.

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- LDR sensor KY018

*/

// constanten
const int led = 13;                                // interne led pin
const int sensorPin = A5;                          // KY018 signaal pin

// variabelen
int waarde;                                     // KY018 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                         // led is output
    pinMode(sensorPin, INPUT);                    // licht sensor is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = analogRead(sensorPin);               // lees sensor
    digitalWrite(led, HIGH);                      // led aanzetten
    delay(1024-waarde);                           // wachten
    digitalWrite(led, LOW);                       // led afzetten
    delay(1024-waarde);                           // wachten
}

```

KY019 – Relais

Opgave

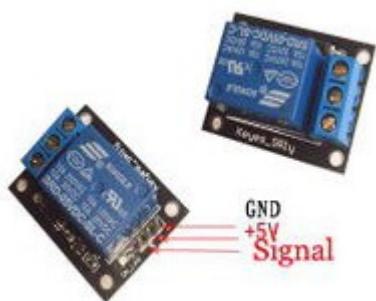
Laat de ingebouwde led 13 knipperen wanneer het relais schakelt. Test het NO en NC contact van het relais met een multimeter.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Relais KY019
- Multimeter

Sensor

De [Songle SRD-05VDC-SL-C](#) is een 1 kanaal relais waarmee apparaten op 220V/10A wisselspanning kunnen geschakeld worden. De module bevat een NO en NC contact.



Sketch

We declareren constanten voor de interne led en de relais KY019 en initialiseren beide pinnen als output in de setup functie.

In de loop functie zetten we om de 2 seconden het relais en de interne led aan en uit. We controleren de NO en NC contacten van het relais met de multimeter.

```

/*
  KY019 Relais

  Laat de ingebouwde led 13 knipperen wanneer het relais schakelt.
  Test het NO en NC contact van het relais met een multimeter.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Relais KY019
  - Multimeter

*/

// constanten
const int led = 13;                                // interne led pin
const int relaisPin = 10;                            // KY019 signaal pin

// variabelen
int waarde;                                         // KY019 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(led, OUTPUT);                             // led is output
  pinMode(relaisPin, OUTPUT);                      // relais pin is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  digitalWrite(led, HIGH);                          // led aanzetten
  digitalWrite(relaisPin, HIGH);                   // relais aanzetten
  delay(2000);                                    // wacht 2 seconden
  digitalWrite(led, LOW);                           // led afzetten
  digitalWrite(relaisPin, LOW);                   // relais afzetten
  delay(2000);                                    // wacht 2 seconden
}

```

KY020 – Tilt schakelaar

Opgave

Laat de ingebouwde led 13 branden bij het sluiten van de tilt schakelaar.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Tilt schakelaar KY020

Sensor



Dit is een tilt schakelaar. Afhankelijk van de tilt positie komt de signaal pin hoog of laag.

Deze tilt schakelaar werkt met een metalen balletje in een buisje met contacten. Dit is niet zo nauwkeurig als de trilling sensor [KY002](#).

Een andere variant is de kwik schakelaar [KY017](#), maar deze is niet milieuvriendelijk wegens het giftige kwik.

Sketch

We declareren constanten voor de interne led en de tilt schakelaar KY020 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de module. Als de tilt schakelaar gesloten is (verticale positie) is de waarde op de signaal pin laag (!) en zetten we de interne led aan. In het andere geval (horizontale positie) is de waarde hoog en zetten we de led af.

```

/*
  KY020 Tilt schakelaar

  Laat de ingebouwde led 13 branden bij het sluiten van de tilt schakelaar.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Tilt schakelaar KY020

*/

// constanten
const int led = 13;                                // interne led pin
const int switchPin = 10;                            // KY020 signaal pin

// variabelen
int waarde;                                       // KY020 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(led, OUTPUT);                           // led is output
  pinMode(switchPin, INPUT);                      // schakelaar is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  waarde = digitalRead(switchPin);                // lees sensor
  if (waarde == LOW) {                           // bij sluiten schakelaar
    digitalWrite(led, HIGH);                     // led aanzetten
  } else {                                         // anders
    digitalWrite(led, LOW);                      // led afzetten
  }
}

```

KY021 – Kleine Reed schakelaar

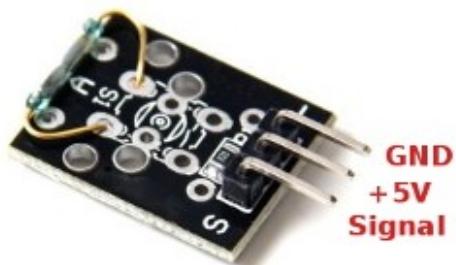
Opgave

Laat de ingebouwde led 13 oplichten wanneer de Reed schakelaar gesloten is.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Kleine Reed schakelaar KY021
- Magneetje

Sensor



Deze module bevat een kleine Reed schakelaar die onder invloed van een magnetisch veld open of sluit.

Een andere uitvoering is de grote Reed schakelaar [KY025](#); deze heeft naast een digitale ook een analoge signaal pin.

Sketch

We declareren constanten voor de ingebouwde led en de kleine reed schakelaar KY021 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de module. Als de schakelaar gesloten wordt door een magnetisch veld wordt de waarde laag (!) en zetten we de interne led aan. In het andere geval blijft de waarde hoog en zetten we de led af.

```

/*
KY021 Kleine Reed schakelaar

Laat de ingebouwde led 13 oplichten wanneer de Reed schakelaar gesloten is.

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- Kleine Reed schakelaar KY021
- Magneetje

*/

// constanten
const int led = 13;                                // interne led pin
const int emveld = 10;                             // KY021 signaal pin

// variabelen
int waarde;                                       // KY021 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                          // led is output
    pinMode(emveld, INPUT);                      // Reed sensor is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = digitalRead(emveld);                // lees sensor
    if (waarde == LOW) {                           // bij magnetisch veld
        digitalWrite(led, HIGH);                  // led aanzetten
    } else {                                      // anders
        digitalWrite(led, LOW);                   // led afzetten
    }
}

```

KY022 – Infrarode ontvanger

Opgave

Test de IR ontvanger module met een IR afstandsbediening en toon de resultaten in de seriële console.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Infrarode ontvanger KY022
- Infrarode afstandsbediening

Sensor



Deze module bevat de [AX-1838HS](#) infrarode ontvanger.
Specificaties : hoek 90°, spanning 5V, frequentie 37.9 kHz,
maximum afstand 18 m.

Sketch

Voor deze test maken we gebruik van de IRremote bibliotheek die standaard geïnstalleerd is met je Arduino software. Hierdoor wordt de ingewikkelde communicatie met de 1838 IR ontvanger KY022 een stuk eenvoudiger :-) We voegen de bibliotheek toe via het menu Schets – Bibliotheek importeren – IRremote.

We declareren een constante voor de signaal pin van de IR ontvanger KY022. We declareren onze sensor met *IRrecv Irontvanger(irPin)*; en tevens de gedecodeerde resultaten met *decode_results resultaten*;

In de setup functie initialiseren we de IR pin als input en de sensor alsook de seriële console voor het tonen van de resultaten.

In de loop functie tonen we de hexadecimale voorstelling de ontvangen signalen in de seriële console.

The screenshot shows the Arduino Serial Monitor window titled '/dev/ttyACM1 (Arduino Uno)'. The window displays a list of received infrared signal codes in hexadecimal format. The codes listed are: FFFFFFFF, FF18E7, FFFFFFFF, FF7A85, FFFFFFFF, FF42BD, FFFFFFFF, FF5AA5, FFFFFFFF, 92B25BC3, 6825E53E, 731A3E02, 731A3E02, F5999288, F5999288, and 2C452C6C. Below the list, there are settings for 'Autoscroll' (unchecked), 'Geen regeleinde' (selected), and '9600 baud'.

```
FFFFFFF
FF18E7
FFFFFFF
FF7A85
FFFFFFF
FF42BD
FFFFFFF
FF5AA5
FFFFFFF
92B25BC3
6825E53E
731A3E02
731A3E02
F5999288
F5999288
2C452C6C
```

```
/*
 * KY022 Infrarode ontvanger

Test de IR ontvanger module met een IR afstandsbediening en
toon de resultaten in de seriële console.

Benodigdheden :
- Arduino UNO, breadboard, jumper kabeltjes
- Infrarode ontvanger KY022
- Infrarode afstandsbediening
*/
#include <IRremoteInt.h>
#include <IRremote.h>

// constanten
const int irPin = 10; // KY022 signaal pin

// declaratie objecten
IRrecv IRontvanger(irPin); // KY022 IR ontvanger
decode_results resultaten; // gedecodeerde resultaten

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(irPin, INPUT); // IR ontvanger is input
    IRontvanger.enableIRIn(); // IR ontvanger starten
    Serial.begin(9600); // seriële console initialiseren
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    if (IRontvanger.decode(&resultaten)) { // zijn er signalen ?
        Serial.println (resultaten.value, HEX); // toon ze
        IRontvanger.resume(); // ontvang volgende
    }
}
```

KY023 – Mini X/Y joystick

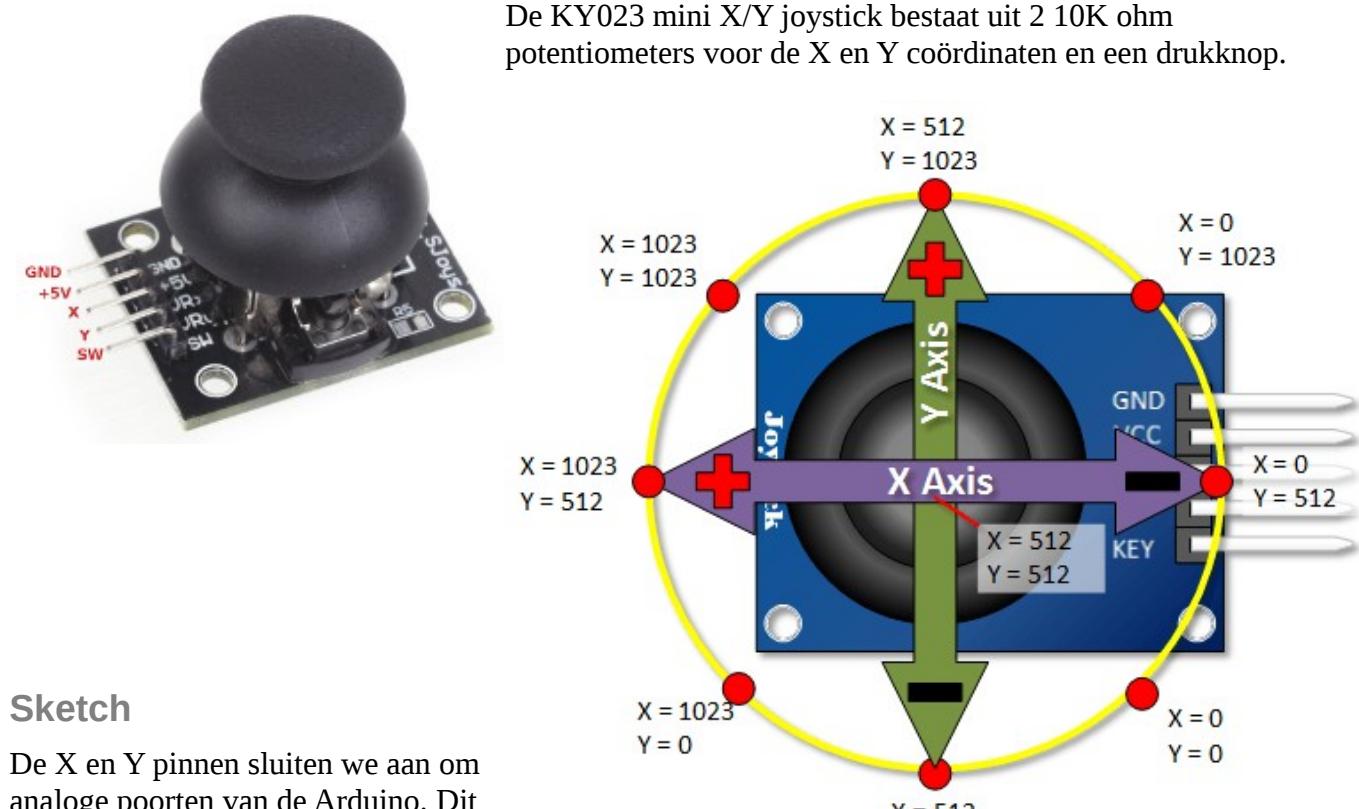
Opgave

Test de joystick module en toon de X en Y coördinaten alsook de status van de drukknop in de seriële console.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Mini X/Y joystick KY023
- 1K ohm weerstand

Sensor



Sketch

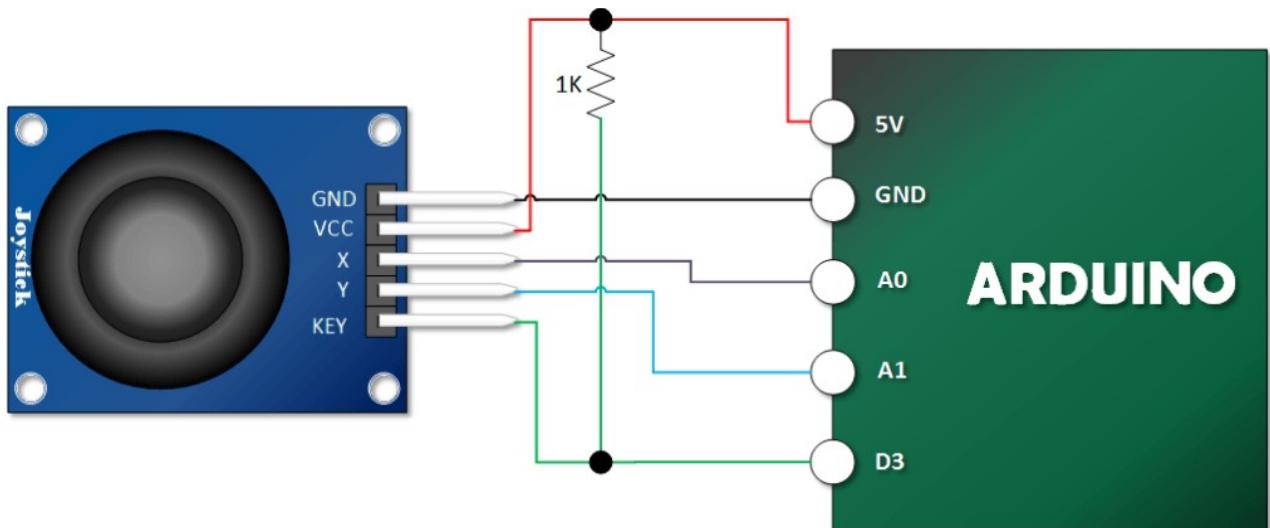
De X en Y pinnen sluiten we aan om analoge poorten van de Arduino. Dit geeft ons coördinaten die variëren van 0 tot 1023. De middenpositie van de joystick geeft ons als coördinaten X=512 en Y=512.

We declareren constanten voor de x, y en drukknop pin van de joystick KY023.

In de setup functie initialiseren we alle pinnen als input alsook de seriële console voor het tonen van de resultaten.

In de loop functie tonen we de X en Y waarden alsook de status van de drukknop in de seriële console.

Opgelet : Plaats een weerstand van 1K ohm tussen de 5V en de drukknop pin !



```
x + - /dev/ttyACM0 (Arduino Uno) Verzenden
X: 525 Y: 523 Drukknop: GESLOTEN
X: 525 Y: 524 Drukknop: GESLOTEN
X: 525 Y: 523 Drukknop: GESLOTEN
X: 525 Y: 523 Drukknop: GESLOTEN
X: 525 Y: 523 Drukknop: OPEN
X: 525 Y: 524 Drukknop: OPEN
X: 525 Y: 523 Drukknop: GESLOTEN
X: 526 Y: 524 Drukknop: GESLOTEN
X: 525 Y: 523 Drukknop: OPEN
X: 525 Y: 524 Drukknop: OPEN
X: 522 Y: 0 Drukknop: OPEN
X: 525 Y: 1 Drukknop: OPEN
X: 525 Y: 0 Drukknop: OPEN
X: 525 Y: 1 Drukknop: OPEN
X: 525 Y: 1 Drukknop: OPEN
X: 525 Y: 0 Drukknop: OPEN
X: 525 Y: 0 Drukknop: OPEN
X: 525 Y: 1 Drukknop: OPEN
Geen regeleinde 9600 baud
 Autoscroll
```

```

/*
  KY023 Mini X/Y Joystick

  Test de joystick module en toon de X- en Y-coordinaten alsook de status van
  de drukknop in de seriële console.
  OPGELET: plaats een weerstand van 1K tussen de 5V en de drukknop pin !

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Mini X/Y joystick KY023
  - 1K ohm weerstand

*/

// constanten
const int xPin = A0;                      // KY023 X pin
const int yPin = A1;                      // KY023 Y pin
const int swPin = 10;                     // KY023 switch pin

// variabelen
int x_wrde, y_wrde, sw_wrde;           // KY023 variabelen

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(xPin, INPUT);                // X pin is input
    pinMode(yPin, INPUT);                // Y pin is input
    pinMode(swPin, INPUT);               // switch pin is input
    Serial.begin(9600);                  // seriële console initialiseren
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    x_wrde = analogRead (xPin);          // X waarde
    y_wrde = analogRead (yPin);          // Y waarde
    sw_wrde = digitalRead (swPin);       // switch waarde
    Serial.print("X: ");                // toon
    Serial.print(x_wrde, DEC);          // X waarde
    Serial.print ("\tY: ");              // toon
    Serial.print (y_wrde, DEC);          // Y waarde
    Serial.print ("\tDrukknop: ");        // toon
    if (sw_wrde == HIGH){               // drukknop niet gedrukt
        Serial.println ("OPEN");         // toon OPEN
    } else {                           // anders
        Serial.println ("GESLOTEN");     // toon GESLOTEN
    }
    delay (500);                      // wacht 1/2 sec
}

```

KY024 – Lineaire Hall sensor

Opgave

Laat de ingebouwde led 13 flikkeren bij het detecteren van een magnetisch veld.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Lineaire Hall sensor KY024
- Magneetje

Sensor



Deze module bevat een [SS49E](#) lineaire Hall sensor, die de aanwezigheid van een magnetisch veld detecteert. De module heeft zowel een analoge als een digitale signaal pin.

De analoge Hall [KY035](#) bevat tevens een SS49E sensor maar heeft geen digitale signaal pin.

De Hall magnetisch veld sensor [KY003](#) is gelijkaardig maar is gebaseerd op een [A3144](#).

Sketch

We declareren constanten voor de interne led en de lineaire Hall sensor KY024 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de sensor. Als de sensor een magnetisch veld detecteert wordt de waarde laag (!) en zetten we de interne led aan. In het andere geval blijft de waarde hoog en zetten we de led af.

```

/*
KY024 Lineaire Hall sensor

Laat de ingebouwde led 13 flikkeren bij het detecteren van een magnetisch veld.

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- Lineaire Hall sensor KY024
- Magneetje

*/

// constanten
const int led = 13;                                // interne led pin
const int emveld = 10;                             // KY024 signaal pin

// variabelen
int waarde;                                     // KY024 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                          // led is output
    pinMode(emveld, INPUT);                        // Hall sensor is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = digitalRead(emveld);                // lees sensor
    if (waarde == LOW) {                         // bij magnetisch veld
        digitalWrite(led, HIGH);                  // led aanzetten
    } else {                                       // anders
        digitalWrite(led, LOW);                   // led afzetten
    }
}

```

KY025 – Grote Reed schakelaar

Opgave

Laat de ingebouwde led 13 oplichten wanneer de Reed schakelaar gesloten is.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Grote Reed schakelaar KY025
- Magneetje

Sensor



Deze module bevat een grote Reed schakelaar die onder invloed van een magnetisch veld open of sluit. De module heeft zowel een analoge als een digitale signaal pin.

De kleine Reed schakelaar [KY021](#) heeft enkel een digitale signaal pin.

Sketch

We declareren constanten voor de interne led en analoge en digitale pin van de grote reed schakelaar KY025 en initialiseren de pinnen als respectievelijk output en twee maal input in de setup functie.

In de loop functie lezen we de digitale en analoge waarden van de module. Als de schakelaar gesloten wordt door een magnetisch veld wordt de digitale waarde hoog en zetten we de interne led aan. In het andere geval blijft de digitale waarde laag en zetten we de led af.

De analoge waarden variëren van 1023 bij een niet gesloten Reed schakelaar naar ongeveer 20 bij een gesloten schakelaar. Die waarde kan slechts in mindere mate beïnvloed worden door de potentiometer.

```

/*
  KY025 Grote Reed schakelaar

  Laat de ingebouwde led 13 oplichten wanneer de Reed schakelaar gesloten is.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Grote Reed schakelaar KY025
  - Magneetje
*/

// constanten
const int led = 13;                                // interne led pin
const int emdigitaal = 10;                          // KY025 digitale signaal pin
const int emanaloog = A5;                           // KY025 analoge signaal pin

// variabelen
int wrde_digitaal;                               // KY025 digitale sensorwaarde
int wrde_analoog;                                // KY025 analoge sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                         // led is output
    pinMode(emdigitaal, INPUT);                  // Reed sensor is input
    pinMode(emanaloog, INPUT);                   // Reed sensor is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    wrde_digitaal = digitalRead(emdigitaal); // lees sensor digitaal
    wrde_analoog = analogRead(emanaloog);   // lees sensor analog
    if (wrde_digitaal == HIGH) {             // bij magnetisch veld
        digitalWrite(led, HIGH);            // led aanzetten
    } else {                                // anders
        digitalWrite(led, LOW);             // led afzetten
    }
}

```

KY026 – Vlammen sensor

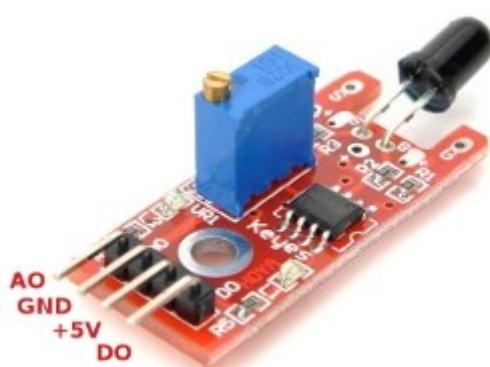
Opgave

Laat de ingebouwde led 13 oplichten wanneer een vlam gedetecteerd wordt. Experimenteer met de potentiometer om de gevoeligheid aan te passen. Toon de resultaten in de seriële console.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Vlammen sensor KY026
- Aansteker

Sensor

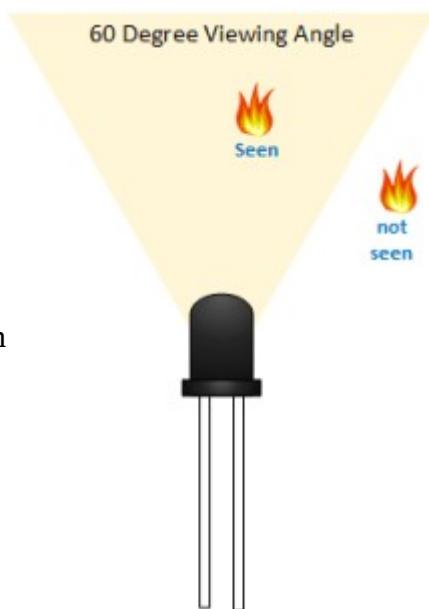


Deze vlammen sensor detecteert vlammen met golflengtes tussen 760 en 1100 nm in een 60° hoek.

De module heeft zowel een digitale als een analoge signaal pin.

De module heeft ook een ingebouwde detectie led.

De detectie gevoeligheid kan ingesteld worden met de potentiometer. De gevoeligheid verhoogt bij het draaien in wijzerzin.



Sketch

We declareren constanten voor de interne led en de analoge signaal pin van de vlammen sensor KY026 en initialiseren de pinnen als respectievelijk output en input in de setup functie. We initialiseren ook de seriële console voor het tonen van de resultaten.

In de loop functie lezen we de digitale waarde van de sensor. Als een vlam gedetecteerd wordt gaat de sensorwaarde laag en zetten we de interne led aan. In het andere geval blijft de sensorwaarde hoog en zetten we de led af. We beelden eveneens een boodschap af in de seriële console.

The screenshot shows the Arduino IDE's Serial Monitor window titled '/dev/ttyACM0 (Arduino Uno)'. The window displays the following text in a loop:

```

GEVAAR VLAM!
Alles OK
GEVAAR VLAM!
Alles OK
GEVAAR VLAM!
GEVAAR VLAM!
GEVAAR VLAM!
GEVAAR VLAM!
GEVAAR VLAM!
Alles OK

```

Below the text, there are several control buttons: 'Autoscroll' (unchecked), 'Geen regeleinde' (selected), '9600 baud' (selected), and two dropdown menus for baud rate and line endings.

```

/*
 * KY026 Vlammen sensor

Laat de ingebouwde led 13 oplichten wanneer een vlam gedetecteerd wordt.
Experimenteer met de potentiometer om de gevoeligheid aan te passen.

Benodigdheden :
- Arduino UNO, breadboard, jumper kabeltjes
- Vlammen sensor KY026
- Aansteker
 */

// constanten
const int led = 13; // interne led pin
const int vlamPin = 8; // KY026 signaal pin

// variabelen
int waarde; // KY026 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT); // led is output
    pinMode(vlamPin, INPUT); // vlammen sensor is input
    Serial.begin(9600); // seriële console initialiseren
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = digitalRead(vlamPin); // lees sensor
    if (waarde == LOW) { // vlam gedetecteerd
        digitalWrite(led, HIGH); // led aanzetten
        Serial.println("GEVAAR VLAM!"); // toon in console
    } else {
        digitalWrite(led, LOW); // led afzetten
        Serial.println("Alles OK"); // toon in console
    }
    delay(1000); // even wachten
}

```

KY027 – Magische led cup module

Opgave

Laat de led op de module flikkeren naargelang de status van de kwik schakelaar.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Magische led cup sensor KY027

Sensor

Deze module bevat een kwik tilt schakelaar en een led. Hiermee kan je naargelang de stand van de kwik schakelaar de led aan of afzetten.



Sketch

We declareren constanten voor de signaal en de led pin van de magische led cup module KY027 en initialiseren de pinnen als respectievelijk input en output in de setup functie.

In de loop functie lezen we de digitale waarde van de sensor. Als de tilt schakelaar sluit gaat de sensorwaarde laag en zetten we de led op de module aan. In het andere geval blijft de sensorwaarde hoog en zetten we de led af.

```

/*
KY027 Magische led cup module

Laat de led op de module flikkeren naargelang de stand van de kwik tilt schakelaar.

Benodigdheden :
- Arduino UNO, breadboard, jumper kabeltjes
- Magische led cup module KY027
*/

// constanten
const int swPin = 8;                                // KY027 signaal pin
const int ledPin = 9;                               // KY027 led pin

// variabelen
int waarde;                                         // KY027 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(swPin, INPUT);                      // switch is input
    pinMode(ledPin, OUTPUT);                     // led is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = digitalRead(swPin);                  // lees sensor
    if (waarde == LOW) {                         // bij schakeling
        digitalWrite(ledPin, HIGH);                // led aanzetten
    } else {                                       // anders
        digitalWrite(ledPin, LOW);                 // led afzetten
    }
}

```

KY028 – Digitale temperatuur sensor

Opgave

Toon de omgevingstemperatuur in de seriële monitor.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Digitale temperatuur sensor KY028

Sensor

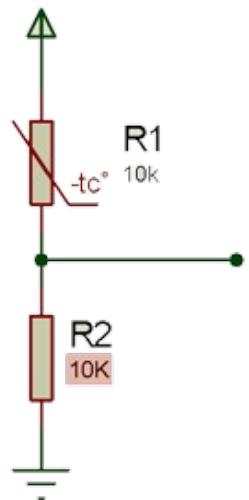


Dit is een digitale temperatuur sensor bestaande uit een spanningsdeler van een 10K ohm PTC (positieve temperatuur coëfficiënt) thermistor met een 10K ohm weerstand.

De weerstand van de thermistor stijgt als de temperatuur stijgt.

De spanning op de anaologe signaal pin kan als volgt berekend worden :

$$V_{out} = \frac{R_2}{R_1 + R_2} * V_{in}$$



In ons geval zal de spanning op de signaal pin dus dalen bij een stijgende temperatuur omdat de waarde van de weerstand R1 zal stijgen.

Deze module heeft zowel een digitale als een analoge signaal pin. Voor de digitale output kan de threshold voor hoog/laag ingesteld worden met de potentiometer. De module is ook voorzien van een power led en een temperatuur alarm led.

Een andere variant van deze temperatuur sensor is de analoge temperatuur sensor [KY013](#).

Sketch

We declareren een constante voor de analoge signaal pin van de digitale temperatuur sensor KY018. We initialiseren de signaal pin als input en starten de seriële console in de setup functie.

In de loop functie lezen we de waarde van de sensor en zetten deze om naar temperatuur met de functie *thermistor(waarde)*. Deze gebruikt de logaritme functie uit de ingebouwde *math* bibliotheek van Arduino. We tonen de resultaten in de seriële console.

We gebruiken de digitale output van deze module niet.

```
x + - /dev/ttyACM0 (Arduino Uno) Verzenden  
Sensor: 472 Temperatuur: 28.61 C  
Sensor: 474 Temperatuur: 28.42 C  
Sensor: 486 Temperatuur: 27.33 C  
Sensor: 507 Temperatuur: 25.45 C  
Sensor: 516 Temperatuur: 24.64 C  
Sensor: 524 Temperatuur: 23.93 C  
Sensor: 531 Temperatuur: 23.32 C  
Sensor: 537 Temperatuur: 22.79 C  
Sensor: 543 Temperatuur: 22.26 C  
Sensor: 549 Temperatuur: 21.73 C  
Sensor: 553 Temperatuur: 21.38 C  
Sensor: 558 Temperatuur: 20.94 C  
Sensor: 561 Temperatuur: 20.68 C  
Sensor: 564 Temperatuur: 20.42 C  
Sensor: 567 Temperatuur: 20.15 C  
Sensor: 569 Temperatuur: 19.98 C  
Sensor: 571 Temperatuur: 19.80 C  
 Autoscroll Geen regeleinde ▾ 9600 baud ▾
```

```

/*
  KY028 Digitale temperatuur sensor

  Meet de omgevingstemperatuur met de sensor en toon de resultaten in de seriële
  console. De sensor kan gecalibreerd worden met de potentiometer.

  Benodigdheden :
  - Arduino UNO, breadboard, jumper kabeltjes
  - digitale temperatuur sensor KY028
*/

#include <math.h>

// constanten
const int sensorPin = A5;           // signaal pin temperatuur sensor

// variabelen
int waarde;                      // KY028 sensorwaarde
double temp;                      // temperatuur

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(sensorPin, INPUT);      // temperatuur is output
    Serial.begin(9600);            // seriële console initialiseren
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt.
void loop() {
    waarde = analogRead(sensorPin); // lees sensor
    temp = thermistor(waarde);     // waarde omzetten naar temperatuur
    Serial.print("Sensor: ");
    Serial.print(waarde);          // toon sensor waarde
    Serial.print("\tTemperatuur: ");
    Serial.print(temp);            // toon temperatuur
    Serial.println(" C");          // in graden Celsius
    delay(1000);                  // even wachten
}

// de thermistor functie doet de omzetting van de sensorwaarde naar temperatuur
double thermistor(int waarde) {
    double Temp;
    Temp = log(10000.0 / ((1024.0 / waarde - 1)));
    Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp)) * Temp);
    Temp = Temp - 273.15;          // Converteer Kelvin naar Celcius
    return Temp;
}

```

KY029 – Kleine twee kleuren led

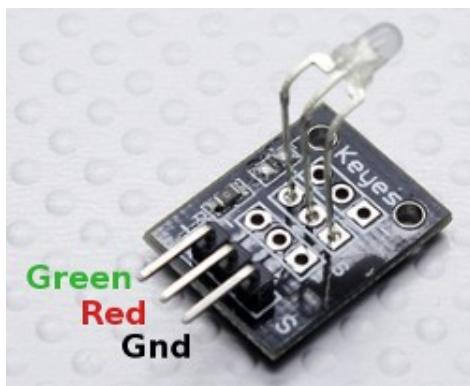
Opgave

Laat de twee kleuren led afwisselend groen en rood in- en uitfaden.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Kleine twee kleuren led KY029

Sensor



Dit is een twee kleuren led module. Afhankelijk van de signalen op de pinnen brandt de led rood of groen.

Een andere versie is de grote twee kleuren led [KY011](#).

Ogelet : de labeling van de pinnen op de module is verkeerd.
– = groen / midden = rood / signaal = GND

Sketch

We declareren constanten voor de rode en groene pinnen van de twee kleuren led KY029 en initialiseren beide pinnen als output in de setup functie.

In de loop functie laten we de twee kleuren led afwisselend groen en rood in- en uitfaden.

- Arduino UNO, breadboard, jumper kabeltjes
- Schok sensor KY031

```

/*
  KY029 Kleine twee kleuren led

  Laat de twee kleuren led afwisselend groen en rood in- en uitfaden.
  OPGELET : de pin layout van deze module is als volgt :
    - = groen / midden = rood / signaal = GND

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Twee kleuren led KY029
*/

// constanten
const int rPin = 10;                                // KY029 rode pin
const int gPin = 11;                                // KY029 groene pin

// variabelen
int waarde;                                         // KY029 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(rPin, OUTPUT);                            // rode pin is output
  pinMode(gPin, OUTPUT);                            // groene pin is output
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  for (waarde=255; waarde>0; waarde--) {
    analogWrite(gPin, waarde);                      // groen
    analogWrite(rPin, 255-waarde);                  // rood
    delay(15);                                     // even wachten
  }
  for (waarde=0; waarde<255; waarde++) {
    analogWrite(gPin, waarde);                      // groen
    analogWrite(rPin, 255-waarde);                  // rood
    delay(15);                                     // even wachten
  }
}

```

KY031 – Schok sensor

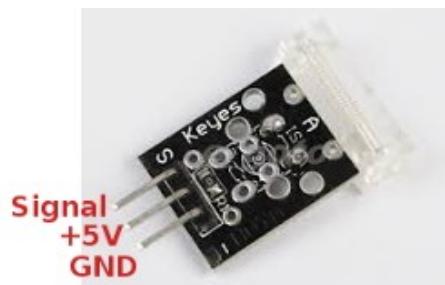
Opgave

Laat de ingebouwde led 13 flikkeren bij het detecteren van schokken.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Schok sensor KY031

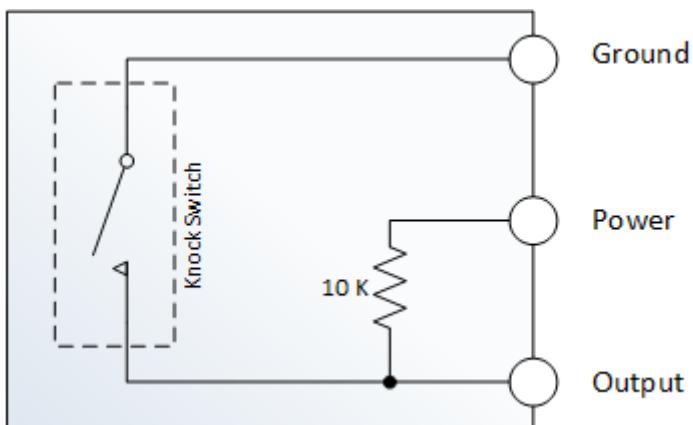
Sensor



Deze sensor bestaat uit een koperen spoeltje dat gemonteerd is rond een metalen buisje. Bij trillingen maken beiden contact en wordt het circuit gesloten.

De module heeft een ingebouwde weerstand die de signaal pin hoog houdt. Bij trillingen wordt de signaal pin laag.

Deze sensor is gelijkaardig als de trilling sensor [KY002](#), maar is een stuk minder gevoelig.



Sketch

We declareren constanten voor de interne led en de schok sensor KY031 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de sensor. Als de sensor een trilling detecteert wordt de waarde laag (!) en zetten we de ingebouwde led aan. In het andere geval blijft de waarde hoog en zetten we de led af.

```

/*
KY031 Schok sensor

Laat de ingebouwde led 13 flikkeren bij het detecteren van schokken.

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- Schok sensor KY031

*/

// constanten
const int led = 13;                                // interne led pin
const int schok = 10;                               // KY031 signaal pin

// variabelen
int waarde;                                       // KY031 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                          // led is output
    pinMode(schok, INPUT);                         // schok sensor is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = digitalRead(schok);                  // lees sensor
    if (waarde == LOW) {                           // bij trilling
        digitalWrite(led, HIGH);                   // led aanzetten
    } else {                                       // anders
        digitalWrite(led, LOW);                    // led afzetten
    }
}

```

KY032 – Obstakel detectie sensor

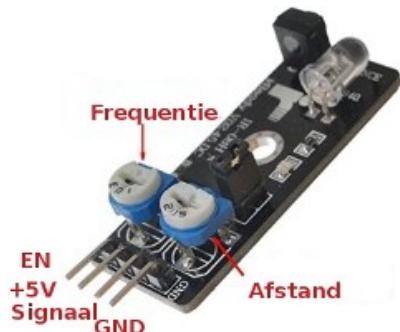
Opgave

Laat de ingebouwde led 13 flikkeren bij het detecteren van een obstakel. Experimenteer met de potentiometer om de afstand van de detectie te variëren.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Obstakel detectie sensor KY032

Sensor



Deze IR-08H module combineert een infrarode zender en ontvanger om obstakels te detecteren. Het signaal komt laag als er een obstakel gedetecteerd wordt. De detectie afstand is instelbaar met de potentiometer tussen 2 tot 30 cm.

We gebruiken de EN pin niet in deze test. Er is bitter weinig informatie over deze module te vinden op internet.

Sketch

We declareren constanten voor de interne led en de digitale pin van de obstakel detectie sensor KY032 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de sensor waarde van de sensor. Als een obstakel gedetecteerd wordt, gaat de signaal pin van hoog naar laag en zetten we de ingebouwde led aan. In het andere geval blijft de digitale waarde hoog en zetten we de led af.

```

/*
  KY032 Obstakel detectie sensor

  Laat de ingebouwde led 13 flikkeren bij het detecteren van een obstakel.
  Experimenteer met de potentiometer om de afstand van de detectie te varieren.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Obstakel detectie sensor KY032

*/

// constanten
const int led = 13;                                // interne led pin
const int obstakel = 9;                             // KY032 out pin

// variabelen
int waarde;                                       // KY032 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(led, OUTPUT);                            // led is output
  pinMode(obstakel, INPUT);                       // out pin is input
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  waarde = digitalRead(obstakel);                // lees sensor
  if (waarde == LOW) {                            // bij obstakel
    digitalWrite(led, HIGH);                      // led aanzetten
  } else {                                         // anders
    digitalWrite(led, LOW);                       // led afzetten
  }
  delay(1000); |                                  // even wachten
}

```

KY033 – Lijn volgende sensor

Opgave

Laat de ingebouwde led 13 trager flikkeren bij het detecteren van een zwarte lijn. Experimenteer met de potentiometer om de gevoeligheid van de detectie te variëren. Toon de resultaten in de seriële monitor.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Lijn volgende sensor KY033

Sensor

Deze module combineert een infrarode zender en ontvanger en is speciaal bedoeld voor licht/donker detectie. Wordt veel gebruikt bij lijn volgende robots.

De gevoeligheid van de licht/donker detectie kan worden aangepast met de potentiometer.



Sketch

We declareren constanten voor de interne led en de signaal pin van de lijn volgende sensor KY033 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de sensor. We zetten de ingebouwde led aan en af met als delay de sensorwaarde. Bij de detectie van donker is de sensorwaarde hoog en zal de led dus langzaam flikkeren, bij licht is de waarde laag en zal de led rap flikkeren. We tonen de waarden in de seriële monitor.

x + - /dev/ttyACM0 (Arduino Uno) Verzenden

```

Sensorwaarde: 18 WIT
Sensorwaarde: 18 WIT
Sensorwaarde: 18 WIT
Sensorwaarde: 18 WIT
Sensorwaarde: 743 ZWART
Sensorwaarde: 796 ZWART
Sensorwaarde: 812 ZWART
Sensorwaarde: 800 ZWART
Sensorwaarde: 794 ZWART
Sensorwaarde: 785 ZWART
Sensorwaarde: 790 ZWART
Sensorwaarde: 16 WIT
Sensorwaarde: 17 WIT
Sensorwaarde: 18 WIT
Sensorwaarde: 18 WIT
Sensorwaarde: 18 WIT
Sensorwaarde: 17 WIT

```

```

/*
 * KY033 Lijn volgende sensor

Laat de ingebouwde led 13 trager flikkeren bij het detecteren van een zwarte lijn.
Experimenteer met de potentiometer om de gevoeligheid van de detectie te varieren.
Toon de resultaten in de seriële monitor.

Benodigdheden :
- Arduino UNO, breadboard, jumper kabeltjes
- Lijn volgende sensor KY033
*/

// constanten
const int led = 13; // interne led pin
const int lijn = A5; // KY033 signaal pin

// variabelen
int waarde; // KY033 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT); // led is output
    pinMode(lijn, INPUT); // sensor is input
    Serial.begin(9600); // seriële console initialiseren
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = analogRead(lijn); // lees sensor
    digitalWrite(led, HIGH); // led aanzetten
    delay(waarde); // wachten
    digitalWrite(led, LOW); // led afzetten
    delay(waarde); // wachten
    Serial.print("Sensorwaarde: "); // toon
    Serial.print(waarde); // sensorwaarde
    if (waarde > 500) {
        Serial.println("\tZWART"); // zwart
    } else {
        Serial.println("\tWIT"); // wit
    }
}

```

KY034 – Zeven kleuren led

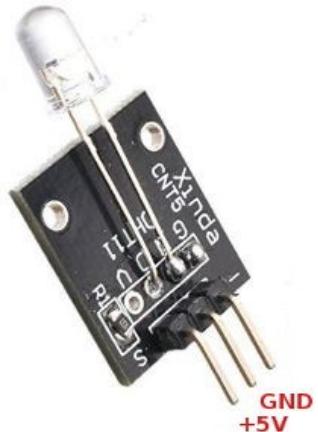
Opgave

Test de zeven kleuren led.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Zeven kleuren led KY034

Sensor



Dit is een zeven kleuren 5mm led met een ingebouwde chip. De 7 kleuren worden achtereenvolgens knipperend afgebeeld. Het kleurenpatroon wordt iedere 15 seconden herhaald.

Opgelet: de module is verkeerd gelabeld !

S = +5V / midden = GND / - = niet gebruikt

Voor deze sensor is geen sketch nodig. Sluit enkel de led aan op de UNO.

KY035 – Analoge Hall sensor

Opgave

Laat de ingebouwde led 13 sneller flikkeren bij detectie van een magnetisch veld en toon de analoge Hall sensorwaarde in de seriële monitor.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Analoge Hall sensor KY035
- Magneetje

Sensor



Deze module bevat een [SS49E](#) lineaire Hall sensor, die de aanwezigheid van een magnetisch veld detecteert. De module heeft een analoge signaal pin.

De analoge Hall [KY024](#) bevat eveneens een SS49E sensor maar heeft naast een analoge ook een digitale signaal pin.

De Hall magnetisch veld sensor [KY003](#) is gelijkaardig maar is gebaseerd op een A3144.

Sketch

We declareren constanten voor de interne led en de analoge Hall sensor KY035 en initialiseren de pinnen als respectievelijk output en input in de setup functie.

In de loop functie lezen we de waarde van de sensor. We zetten de interne led afwisselend aan en af met een delay van de sensorwaarde. Daar de sensorwaarde vermindert als er een magnetisch veld in de buurt komt gaat ook de interne led als gevolg sneller flikkeren. We tonen de sensorwaarde in de seriële console.

Sensorwaarde: 211
 Sensorwaarde: 221
 Sensorwaarde: 233
 Sensorwaarde: 238
 Sensorwaarde: 258
 Sensorwaarde: 251
 Sensorwaarde: 194
 Sensorwaarde: 191
 Sensorwaarde: 190
 Sensorwaarde: 190
 Sensorwaarde: 190
 Sensorwaarde: 190
 Sensorwaarde: 190
 Sensorwaarde: 589
 Sensorwaarde: 517
 Sensorwaarde: 517

Autoscroll Geen regeleinde 9600 baud

```
/*
 * KY035 Analoge Hall sensor

 Laat de ingebouwde led 13 sneller flikkeren bij detectie van een magnetisch veld en
 toon de analoge Hall sensorwaarde in de seriële monitor.

 Benodigdheden :
 - Arduino UNO, breadboard, jumper kabeltjes
 - Analoge Hall sensor KY035
 - Magneetje
 */

// constanten
const int led = 13;                                // interne led pin
const int emveld = A5;                             // KY035 Hall signaal pin

// variabelen
int waarde;                                         // KY035 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                            // led is output
    pinMode(emveld, INPUT);                         // Hall sensor is input
    Serial.begin(9600);                            // seriële console initialiseren
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    waarde = analogRead(emveld);                  // lees sensor
    digitalWrite(led, HIGH);                        // led aanzetten
    delay(waarde);                               // wacht waarde msec
    digitalWrite(led, LOW) ;                      // led afzetten
    delay(waarde);                               // wacht waarde msec
    Serial.print("Sensorwaarde: ");
    Serial.println(waarde, DEC);                  // sensorwaarde
}
```

KY036 – Aanraking sensor

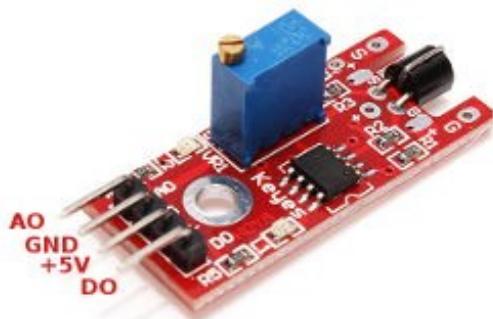
Opgave

Laat de ingebouwde led oplichten bij aanraken van de sensor en bij de volgende aanraking terug uitdoven. Experimenteer met de potentiometer om de gevoeligheid bij te regelen.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Aanraking sensor KY036

Sensor



Deze module bevat een metalen sensor die bij aanraking de signaal pin DO hoog zet.

De gevoeligheid van de sensor kan aangepast worden met de potentiometer.

Sketch

We declareren constanten voor de interne led en de digitale signaal pin van de aanraking sensor KY036. We declareren ook enkele variabelen : *laatsteTijd* om de tijd van de laatste aanraking en *ledAan* om de status van de led bij te houden.

In de setup functie initialiseren we de led en de signaal pin als respectievelijk output en input. We zetten ook de led af.

In de loop functie lezen we de waarde van de sensor. Bij aanraking wordt deze waarde hoog. We testen nu of er meer als 50 ms verstrekken is sedert de laatste aanraking. Als dit het geval is toggelen we de status van de led. We bewaren tevens de tijd in onze *laatsteTijd* variabele.

```

/*
KY036 Aanraking sensor

Laat de ingebouwde led oplichten bij aanraken van de sensor en bij de volgende aanraking
terug uitdoven. Experimenteer met de potentiometer om de gevoeligheid bij te regelen.

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- Aanraking sensor KY036
*/

// constanten
const int touch = 2;                      // KY036 D0 signaal pin
const int led = 13;                        // led pin

// variabelen
unsigned long laatsteTijd = 0;            // tijd laatste aanraking
boolean ledAan = false;                   // status van de led

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                 // led is output
    pinMode(touch, INPUT);               // KY036 pin is input
    digitalWrite(led, LOW);              // led afzetten
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    int touchStatus = digitalRead(touch); // lees touch sensor
    if (touchStatus == HIGH) {           // als aangeraakt
        if (millis() - laatsteTijd > 50) { // en meer dan 50 ms geleden
            ledAan = !ledAan;             // boolean led status aanpassen
            digitalWrite(led, ledAan?HIGH:LOW); // led omschakelen
        }
        laatsteTijd = millis();          // tijd bewaren
    }
}

```

KY037 – Grote microfoon

Opgave

Laat de ingebouwde led oplichten bij detectie van geluiden door de microfoon. Experimenteer met de potentiometer om de gevoeligheid van de detectie bij te regelen.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Grote microfoon KY037

Sensor



Deze microfoon module kan gebruikt worden om geluiden te detecteren. Er is een analoge en digitale output.

De module heeft 2 ingebouwde leds : één voor power en één voor detectie. Deze laatste kan je helpen om de gevoeligheid met behulp van de potentiometer in te stellen.

Een andere variant van deze sensor is de kleine microfoon [KY038](#).

Sketch

We declareren constanten voor de ingebouwde led en de digitale signaal pin van de microfoon sensor KY037.

In de setup functie initialiseren we de led en de signaal pin als respectievelijk output en input. We zetten ook de led af.

In de loop functie lezen we de waarde van de sensor. Bij detectie van een geluid (bvb handgeklap) wordt deze waarde hoog en zetten we de ingebouwde led aan. In het andere geval is de waarde laag en zetten we de led uit.

```

/*
  KY037 Grote microfoon

  Laat de ingebouwde led oplichten bij detectie van geluiden door de microfoon.
  Experimenteer met de potentiometer om de gevoeligheid van de detectie bij te regelen.

  Benodigdheden :
  - Arduino UNO, breadboard, jumper kabeltjes
  - Grote microfoon KY037
*/

// constanten
const int geluid = 10;                      // KY037 D0 signaal pin
const int led = 13;                          // led pin

// variabelen
int sensorwaarde;                         // KY037 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(led, OUTPUT);                  // led is output
    pinMode(geluid, INPUT);                // KY037 pin is input
    digitalWrite(led, LOW);                // led afzetten
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    sensorwaarde = digitalRead(geluid);   // lees sensor
    if (sensorwaarde == HIGH) {            // geluid gedetecteerd
        digitalWrite(led, HIGH);          // led aan
    } else {                            // anders
        digitalWrite(led, LOW);          // led af
    }
}

```

KY038 – Kleine microfoon

Opgave

Laat de ingebouwde led oplichten bij detectie van geluiden door de microfoon. Experimenteer met de potentiometer om de gevoeligheid van de detectie bij te regelen.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Grote microfoon KY037

Sensor

Deze microfoon module kan gebruikt worden om geluiden te detecteren. Er is een analoge en digitale output.



De module heeft 2 ingebouwde leds : één voor power en één voor detectie. Deze laatste kan je helpen om de gevoeligheid met behulp van de potentiometer in te stellen.

Een andere variant van deze sensor is de grote microfoon [KY037](#)

Sketch

We declareren constanten voor de ingebouwde led en de digitale signaal pin van de microfoon sensor KY038.

In de setup functie initialiseren we de led en de signaal pin als respectievelijk output en input. We zetten ook de led af.

In de loop functie lezen we de waarde van de sensor. Bij detectie van een geluid (bvb handgeklap) wordt deze waarde hoog en zetten we de ingebouwde led aan. In het andere geval is de waarde laag en zetten we de led uit.

```

/*
  KY038 Kleine microfoon

  Laat de ingebouwde led oplichten bij detectie van geluiden door de microfoon.
  Experimenteer met de potentiemeter om de gevoeligheid van de detectie bij te regelen.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Kleine microfoon KY038
*/

// constanten
const int geluid = 10;                      // KY038 D0 signaal pin
const int led = 13;                          // led pin

// variabelen
int sensorwaarde;                         // KY038 sensorwaarde

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(led, OUTPUT);                     // led is output
  pinMode(geluid, INPUT);                  // KY038 pin is input
  digitalWrite(led, LOW);                  // led afzetten
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
  sensorwaarde = digitalRead(geluid);      // lees sensor
  if (sensorwaarde == HIGH) {               // geluid gedetecteerd
    digitalWrite(led, HIGH);                // led aan
  } else {                                // anders
    digitalWrite(led, LOW);                 // led af
  }
}

```

KY039 – Hartslag sensor

Opgave

Laat de ingebouwde led oplichten in het ritme van je hartslag. Plaats hiervoor je vinger tussen de infrarode sensor en de fototransistor van de module.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Hartslag sensor KY039

Sensor

Deze hartslag sensor module combineert een infrarode led en een fototransistor. Plaats je vinger tussen beide elementen.



Sketch

Met behulp van de seriële monitor bekijken we de output op de signaal pin van de module. De rauwe sensorwaarde wordt via een speciale formule omgezet naar een waarde. Hiervan wordt het maximum bewaard en vergeleken met volgende waarden. Bij de maxima wordt de ingebouwde led aangezet, in het ander geval afgezet.

```
/*
KY039 Hartslag sensor

Laat de ingebouwde led oplichten in het ritme van je hartslag. Plaats hiervoor je vinger
tussen de infrarode sensor en de fototransistor van de module.

Benodigdheden :
- Arduino UNO, breadbord, jumper kabeltjes
- Hartslag sensor KY039
 */

// constanten
const int sensorPin = A0; // KY039 signaal pin
const int led = 13; // led pin

// variabelen
float alpha = 0.75; // 
float change = 0.0; // 
float maxval = 0.0; // 
int period = 50; // 

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
  pinMode(led, OUTPUT); // led is output
  pinMode(sensorPin, INPUT); // KY039 pin is input
  Serial.begin(9600); // seriële monitor initialiseren
  Serial.println("Hartslag detectie");
}
```

```

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    static float oldValue=1009;
    static float oldChange=0.2;

    // This is generic code provided with the board.
    int rawValue=analogRead(sensorPin);
    float value= alpha*oldValue +(1-alpha)* rawValue;
    change=value-oldValue;

    // Display data on the LED via a blip:
    // Empirically, if we detect a peak as being X% from
    // absolute max, we find the pulse even when amplitude
    // varies on the low side.

    // Reset max every time we find a new peak
    if (change >= maxval) {
        maxval = change;
        Serial.println(" | ");
        digitalWrite(led, HIGH);
    } else {
        Serial.println(" | ");
        digitalWrite(led, LOW);
    }
    // Slowly decay max for when sensor is moved around
    // but decay must be slower than time needed to hit
    // next heartbeat peak.
    maxval = maxval * 0.98;

    // Display debug data on the console
    Serial.print(value);
    Serial.print(", ");
    Serial.print(change);
    Serial.print(", ");
    Serial.println(maxval);

    oldValue=value;
    oldChange=change;
    delay(period);
}

```

KY040 – Rotary Encoder

Opgave

Toon de positie en draairichting van de rotary encoder in de seriële monitor.

Benodigdheden

- Arduino UNO, breadboard, jumper kabeltjes
- Rotary encoder KY040
- 2 capaciteiten 0,1 μ F
- 1 weerstand 10K ohm

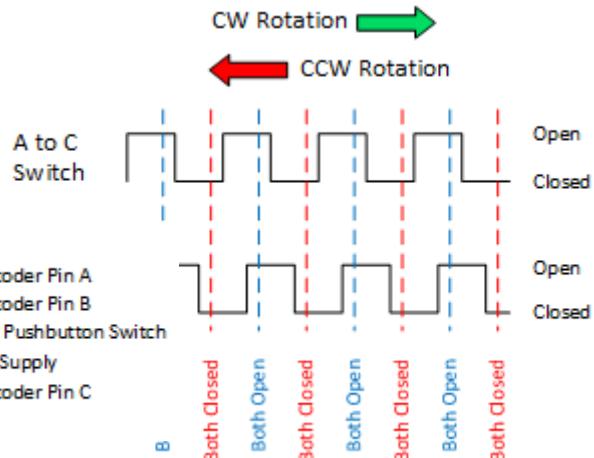
Sensor



Een rotary encoder lijkt op een potentiometer, maar is dit helemaal niet ! Het is een component met 2 interne schakelaars die ofwel beiden open of gesloten zijn. Afhankelijk van de draairichting van de encoder wordt eerst de ene en daarna de volgende schakelaar omgeschakeld.

Deze component wordt vooral gebruikt in stappenumotoren schakelingen of als digitale potentiometer.

Bij het draaien in wijzerzin schakelt eerst de pin CLK en daarna de pin DT om. In tegenwijzerzin schakelen beide pinnen in omgekeerde volgorde om.



Verder heeft deze module ook nog een NO druckschakelaar met als output pin SW. Die wordt geschakeld door de encoder draai as in te drukken. Deze druckschakelaar kan typisch gebruikt worden om de functie van de schakeling aan te passen : bvb het schakelen tussen normale en fijne afstemmingen.

Sketch

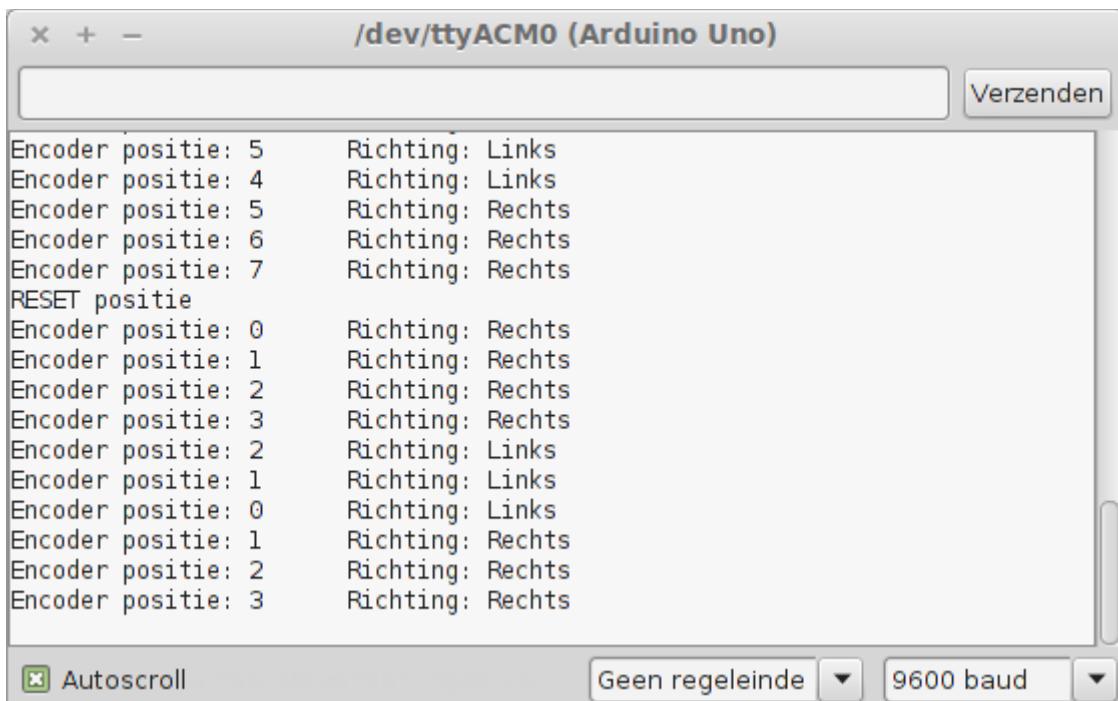
We sluiten en encoder als volgt aan op de UNO :

Pinnen KY040	Pinnen UNO
Encoder pin A – CLK	2
Encoder pin B – DT	3
Drukschakelaar – SW	4
+	+5V
Encoder pin C – GND	GND

Om de stabiliteit van de schakeling te waarborgen plaatsen we een $0,1 \mu\text{F}$ capaciteit tussen massa en de pinnen CLK en DT. Hierdoor wordt het debouncen vermindert; het willekeurig omschakelen van deze signaalpinnen wegens interferentie. We plaatsen ook nog een 10K ohm weerstand tussen de pinnen SW en +.

We declareren constanten voor de CLK, DT en SW pinnen van de rotary encoder KY040 en initialiseren de pinnen als input in de setup functie. We definiëren ook enkele variabelen voor de encoder positie en draairichting. Verder wordt ook nog een interrupt functie *isr* verbonden met interrupt 0 mbv deze instructie : *attachInterrupt(0, isr, FALLING)*; Deze interrupt routine past de encoder positie en draairichting variabelen aan.

In de loop functie lezen we de waarde van de SW pin van de encoder. Als die ingedrukt wordt resetten we de encoder positie variabele. Verder wordt bij het draaien aan de encoder de positie en draairichting van de encoder in de seriële console afgebeeld.



The screenshot shows the Arduino Serial Monitor window titled '/dev/ttyACM0 (Arduino Uno)'. The window displays a series of text entries representing encoder data. Each entry consists of two parts: 'Encoder positie:' followed by a value (e.g., 5, 4, 5, 6, 7, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3) and 'Richting:' followed by either 'Links' or 'Rechts'. The data shows a sequence of positions followed by a 'RESET positie' command, then a series of positions and directions alternating between 'Links' and 'Rechts'. At the bottom of the window, there are buttons for 'Autoscroll' (checked), 'Geen regeleinde' (unchecked), and a baud rate selector set to '9600 baud'.

```
Encoder positie: 5      Richting: Links
Encoder positie: 4      Richting: Links
Encoder positie: 5      Richting: Rechts
Encoder positie: 6      Richting: Rechts
Encoder positie: 7      Richting: Rechts
RESET positie
Encoder positie: 0      Richting: Rechts
Encoder positie: 1      Richting: Rechts
Encoder positie: 2      Richting: Rechts
Encoder positie: 3      Richting: Rechts
Encoder positie: 2      Richting: Links
Encoder positie: 1      Richting: Links
Encoder positie: 0      Richting: Links
Encoder positie: 1      Richting: Rechts
Encoder positie: 2      Richting: Rechts
Encoder positie: 3      Richting: Rechts
```

```

/*
  KY040 Rotary encoder

  Toon de positie en draairichting van de rotary encoder in de seriële monitor.

  Benodigdheden :
  - Arduino UNO, breadbord, jumper kabeltjes
  - Rotary encoder KY040
  - 2 capaciteiten 0.1µF (debounce)
  - 1 weerstand 10K ohm
*/

// constanten
const int pinA = 2;                      // KY040 CLK pin
const int pinB = 3;                      // KY040 DT pin
const int pinSW = 4;                     // KY040 SW pin

// variabelen
volatile int encoderPositie = 0;          // positie
boolean encoderRechts;                   // richting

// interrupt routine bij rotatie encoder
// uitgevoerd wanneer pinB van hoog naar laag gaat
void isr() {
    if (!digitalRead(pinB)) {           // pin B laag
        encoderPositie++;              // positie vermeerderen
        encoderRechts = true;          // richting rechts
    } else {                           // pin B hoog
        encoderPositie--;              // positie verminderen
        encoderRechts = false;         // richting links
    }
}

// de setup functie wordt 1 maal uitgevoerd bij de start van de sketch
// of wanneer op de reset knop gedrukt wordt
void setup() {
    pinMode(pinA, INPUT);             // pin A is input
    pinMode(pinB, INPUT);             // pin B is input
    pinMode(pinSW, INPUT);            // pin SW is input
    Serial.begin(9600);               // seriële console initialiseren
    attachInterrupt(0, isr, FALLING); // interrupt 0 voor rotatie
}

// de loop functie wordt steeds herhaald tot de stroom wegvalt
void loop() {
    int laatstePositie = 0;           // laatste positie
    while (true) {                   // blijven uitvoeren
        if (!digitalRead(pinSW)) {   // switch ingedrukt
            encoderPositie = 0;       // reset positie
            while (!digitalRead(pinSW)) {} // wacht tot switch wordt losgelaten
            delay(10);                // debounce
            Serial.println("RESET positie"); // toon RESET
        }
        if (encoderPositie!=laatstePositie) { // rotatie
            laatstePositie = encoderPositie; // laatste positie updaten
            Serial.print("Encoder positie: "); // toon
            Serial.print(encoderPositie);      // positie
            Serial.print("\tRichting: ");       // toon richting
            if (encoderRechts)
                Serial.println("Rechts");      // rechts
            else
                Serial.println("Links");      // anders
        }
    }
}

```