



Schakelmodule Digitale Technologie - online materiaal

Auteurs	Digitaal Redacteur ; Rudy Jonker ; H. Dirks
Team	Schakelmodule Digitale Technologie
Laatst gewijzigd	24 februari 2023
Licentie	CC Naamsvermelding-GelijkDelen 4.0 Internationale licentie
Webadres	https://maken.wikiwijs.nl/171772/



Dit lesmateriaal is gemaakt met Wikiwijs van Kennisnet. Wikiwijs is hét onderwijsplatform waar je leermiddelen zoekt, maakt en deelt.

Inhoudsopgave

Inleiding	3
Doele van de module	3
Structuur van de module	3
Combinatie met andere modules	4
Leerdoelen	4
Studeren en werken in de digitale technologie: kunstmatige Intelligentie - filmpjes van Daan Geijs	5
Ontwerpen en ontwerpcyclus	6
1 - Probleemstelling	6
2 - Pakket van eisen	6
3 - Deeluitwerkingen	6
4 - Ontwerpvoorstel	6
5 - Prototype	7
6 - Evaluatie	7
7 - Contexten	7
Hardware voor de verwerking van informatie in een digitaal systeem	9
Arduino Uno	9
Arduino Nano	10
Arduino Mega	10
Micro:bit	10
Raspberry Pi	10
Hardware voor de input van een digitaal systeem	11
LDR sensor	11
Momentdrukknop	12
Magneetcontact	14
Thermometer (LM35)	15
Potentiometer	18
Microfoon	19
Klok	21
Ultrasone sensor	21
Vochtsensor	22
Lichtsluis	22
Hardware voor de output van een digitaal systeem	24
LED	24
LED RGB	25
Neopixel LEDstrip	25
LCD display	26
Relais	28
Pulse Width Modulation PWM	29
Ventilator	32
DC motor (lineaire motor)	32
Stappenmotor	33
Pomp	33
Buzzer	33
Elektromagneet	33
Hardware voor de opslag en het transport van data	34
Draadjes (en ze bevestigen)	34
i2c	34
(micro) SD card reader	36
Wifi chip	38
Voeding voor een digitaal systeem	39
Arduino, sensoren en actuatoren voeden	39
USB stekker (5v)	39
Digitale systemen instructies geven (programmeren)	40
Programmeren	40
Een voorbeeld	44
Ronde 1	44
Wall of Fame	46

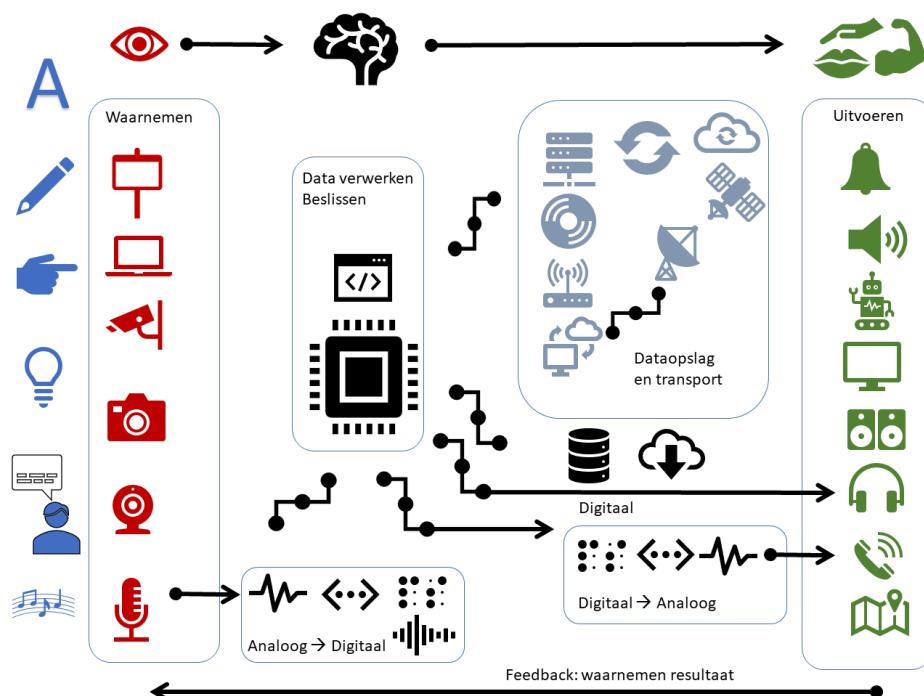
Inleiding

Doele van de module

Tijdens deze module ga je bekend raken met digitale techniek. Digitale techniek is overal.

Structuur van de module

Digitale techniek bestaat uit een aantal onderdelen. Dat wordt schematisch weergegeven in figuur 1. Vrijwel alle systemen nemen iets uit hun omgeving waar. De koelkast meet de temperatuur in de koelkast, een satelliet in de ruimte neemt bepaalde golflengtes licht waar, een telefoon neemt een signaal van de zendmast waar etc. Dat waarnemen gebeurt met **sensoren**. *Hoofdstuk 3* van de schakelmodule gaat over de waarnemingskant van een digitaal systeem. Vervolgens wordt die waarneming omgezet in een digitaal signaal. Dat betekent dat dat wat gemeten wordt omgezet wordt naar informatie waar computers mee overweg kunnen, 1'en en 0'en, wat we **digitale data** noemen. Die data worden vervolgens **verwerkt** (*hoofdstuk 4*) en **opgeslagen en getransporteerd** (*hoofdstuk 5*). Verwerken betekent dat er op basis van de data vervolgacties worden vastgesteld. Als de koelkast te warm is, moet de koelmotor aan, als een telefoon het signaal verliest, gaat hij op zoek naar een nieuw signaal, als het niveau in de brandstoffank te laag wordt, gaat het lampje aan etc. Die vervolgactie, de uitvoer, vindt plaats door **actuatoren**. Dat zijn apparaten die in de echte omgeving een effect hebben. Een lampje dat aangaat, een motor die gaat draaien, een speaker die geluid geeft, dat zijn allemaal actuatoren die aan het werk worden gezet. Deze uitvoer ('output') van een digitaal systeem komt in *hoofdstuk 6* aan bod.



Figuur 1. Alle onderdelen van digitale techniek. Dit figuur gaat vaak terugkomen in deze module

Combinatie met andere modules

Omdat er in veel andere vakgebieden digitale systemen zijn te vinden gaat de informatie van deze module je ook helpen bij andere modules. Bij de modules over het hart (Hart en vaten & Leef met je hart) is er bijvoorbeeld het ECG. Door middel van sensoren wordt er informatie verzameld over je hart en dit kan op een scherm bekijken worden. Maar digitale systemen kunnen ook oplossingen bieden voor problemen. Zo is er een module Ruimte voor de rivier, waarbij je natuurlijk met slimme elektronische systemen sluizen open en dicht kunt zetten om zo het water niveau in een rivier te reguleren. Bij Summer in the city kun je met behulp van een slimme regenton (zie de link hieronder) je tuin beter voorbereiden op weerextremen. De mogelijkheden zijn eigenlijk eindeloos.

Daarnaast is deze module een zogenaamde schakelmodule. Deze module helpt om alle leerlingen, ongeacht hun achtergrond in dit onderwerp, op een basisniveau te krijgen zodat in een verdere module daarop voortgebouwd kan worden.



Slimme regenton

<https://slimmeregenton.nl/>

Leerdoelen

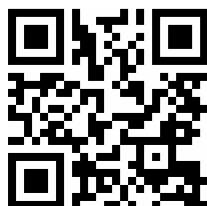
Tijdens dit hoofdstuk leer je

- Dat ontwerpen niet in 1x volmaakt zijn, maar volgens het proces van de ontwerpcyclus stap voor stap verbeteren
- Dat je door de ontwerpcyclus goed toe te passen je een overzichtelijker verslag van je ontwerpproces krijgt en een beter ontwerp
- Dat ontwerpen vaak een kwestie is van gewoon doen en dan goed evalueren en dat fouten maken eigenlijk cruciaal is om tot een goed ontwerp te komen
- Dat je leert welke praktische stappen er nodig zijn om van een ontwerp op papier naar een echt werkend prototype te komen
- Dat je leert nadenken over technische oplossingen voor problemen en deze oplossingen kunt realiseren.

Studeren en werken in de digitale technologie: kunstmatige Intelligentie - filmpjes van Daan Geijs

Daan Geijs heeft over zijn opleiding en werk vier filmpjes gemaakt. Hij vertelt over zijn opleiding en onderzoek, en waar hij later aan de slag kan met zijn kennis van digitale technologie.

- 1 [zijn vakken op de middelbare school en zijn studie aan Universiteit Twente](#)
- 2 [wat Kunstmatige Intelligentie is, en waar het voor gebruikt kan worden](#)
- 3 [welke rol kunstmatige intelligentie in het ziekenhuis kan hebben: promotieonderzoek bij het Radboud UMC](#)
- 4 [welk werk Daan hierna kan gaan doen](#)



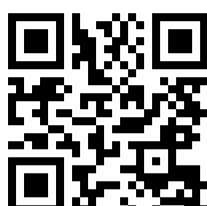
[1 Introductie digitale technologie. Daan Geijs vertelt over zijn studiekeuze en opleiding](#)



[2 Wat is kunstmatige intelligentie? Daan Geijs vertelt wat er met digitale technologie mogelijk is.](#)



[3 Kunstmatige intelligentie in het ziekenhuis. Daan Geijs vertelt over zijn onderzoek aan het herkennen van tumoren met de computer.](#)



[4 Daan Geijs over het werk dat hij in de toekomst kan gaan doen](#)

Ontwerpen en ontwerpcyclus

Jullie gaan tijdens deze module tot een prototype komen. Daar is een bepaalde manier voor om dat zo goed mogelijk te laten verlopen.

1 - Probleemstelling

Als eerste moet je vaststellen voor welk probleem je een oplossing gaat ontwerpen. Stel: je kat gaat iedere vakantie dood van de honger omdat je weg bent en er niemand eten kan geven, en jij wilt een apparaat bouwen dat de kat automatisch eten geeft. Of: de vogels in het vogelhuisje in de tuin raken oververhit in de zomer en je wilt iets bouwen dat het vogelhuisje koelt als de temperatuur te heet wordt. Of: Je wilt een wekker bouwen die licht geeft als de alarmtijd aangebroken is.

Je ziet het al: het doel van je ontwerp (nml. het oplossen van een probleem) bepaal je zelf. Het is handig om het probleem zo concreet mogelijk te omschrijven. Als je doel bijv. 'wereldvrede' is, dan is dat veel te breed en te vaag om een goed startpunt te zijn voor een ontwerpcyclus. Vaak kom je in een ontwerpcyclus later nog terug op het probleem als je er in eerste instantie achter komt dat het probleem te groot is om met je ontwerp op te lossen. In dat geval stel je je doel bij (tenzij je een opdracht hebt gekregen van een opdrachtgever, dan moet je van tevoren inschatten of het haalbaar is).

2 - Pakket van eisen

De tweede stap is dat je gaat vaststellen aan welke eisen je ontwerp moet voldoen om te zorgen dat je probleem ook echt opgelost wordt. Een voorbeeld: als je een apparaat wil bouwen dat automatisch onder water de troebelheid meet als waarschuwingsysteem voor blauwalg, moet het natuurlijk wel waterdicht zijn.

Het kan ook hierbij het geval zijn dat je gedurende je ontwerpcyclus erachter komt dat je een eis mist, die voeg je dan in de volgende ronde toe. Let op dat ook eigenschappen als prijs of afmetingen van het ontwerp belangrijke eisen kunnen zijn.

3 - Deeluitwerkingen

Een ontwerp is vaak complex en heeft verschillende aspecten en moet aan verschillende eisen voldoen. Er is de kwestie van de stroomtoevoer, het systeem moet dingen waarnemen, maar ook output geven. De code moet kloppen en de code bevat verschillende onderdelen om de verschillende onderdelen van je systeem aan te sturen. Om dit alles op een ordelijke manier te ontwerpen helpt het als je je ontwerp opbrekt in kleine stukken. Zo kun je ook makkelijker de taken verdelen binnen een team.

Bij de fase van 'deeluitwerkingen opstellen' breekt je ontwerpproces dus op in kleine stukken die je op zichzelf kan maken en testen waarna je daarna alle deeluitwerkingen (stap voor stap) samenvoegt. Dat samenvoegen is een complexe stap, en daarom moet je dit stap voor stap doen. Als je namelijk in 1x alles bij elkaar zet en er is iets fout, dan weet je namelijk niet welk onderdeel de fout veroorzaakt.

4 - Ontwerpvoorstel

In het ontwerpvoorstel ga je beschrijven welke deeluitwerkingen je samenvoegt voor je ontwerp. Dus

stel dat je bij een wake-up light in de 1e ronde alleen de code schrijft die de klok laat lopen en op een display laat verschijnen, dan beschrijf je dat. Op die manier kun je hier bij je evaluatie op terugkomen.

(in de praktijk voer je deze en de volgende stap tegelijk uit)

5 - Prototype

Nu ga je aan het bouwen. Je zorgt dat je de stappen die je hebt gemaakt werkend hebt (dus als je een code hebt die de klok laat lopen en op een display laat verschijnen, dan laad je niet alleen de code op de Arduino, maar je sluit ook alle draadjes aan voor de display, zodat het systeem ook daadwerkelijk kan werken. Dit is de leukste stap, maar ook de frustrerendste, omdat er vaak dingen niet werken of door jou slecht zijn uitgedacht of voorbereid. Dat hoort erbij, **deal with it**.

6 - Evaluatie

Dit is een hele belangrijke stap, en een stap waarvan het heel belangrijk is dat je al je waarnemingen goed oopschrijft. Je beschrijft nu namelijk wat er van je ontwerp wel en wat niet werkt. Vaak is dit in het begin meer niet dan wel. **Fail and try again**.

Om te zorgen dat je niet steeds dezelfde problemen tegenkomt is het belangrijk dat je je evaluatie gesstructureerd aanpakt. Zorg dat je de datum vermeld, wat werkte er wel, wat werkte er niet, en je vermoedens waar het aan ligt.

Ga in deze fase niet eindeloos lopen prutsen, maar pak de verbetering gesstructureerd aan door weer terug te gaan naar je pakket van eisen, deeluitwerkingen etc. en een goed plan te maken voor de volgende verbetering. Zo ga je dus nog een keer de ontwerpcyclus in. Bij dit project kun je zo 10-15 ontwerpcyclus doorlopen. Het is dus ook slim om je codes van je Arduinoprogramma een versienummer te geven, zodat je altijd achteraf kunt laten zien wat er steeds is veranderd, of dat als je in versie 5 iets blijkt gesloopt te hebben wat in versie 4 nog werkt, je altijd weer terug kan gaan naar de laatst werkende versie.

7 - Contexten

Er zijn ontzettend veel contexten waarvoor je een digitaal systeem kunt ontwerpen. Digitale systemen zijn namelijk overal om je heen!

In het voorbeeld in hoofdstuk 8 zie je hoe de auteur van de site een ontwerpopdracht aanpakt rondom de context van een wake-up light. Eigenlijk zijn alle systemen waarin iets van een klok of tijdwaarneming zit interessante systemen. Je moet namelijk een interessante code schrijven, er iets van een input nodig, en je hebt output op basis van die tijd.

Een andere interessante context is corona. Hoewel de meesten er natuurlijk zat van zijn, kan technologie ons helpen er beter mee te leven. Zo kun je bijvoorbeeld een apparaat bouwen waarmee je met een ultrasone sensor de afstand tot objecten om je heen kunt meten? Stel dat je met zo'n apparaat om je middel door de kantine loopt? Hoe vaak is er dan iemand binnen 1,5m? Of kun je dan een waarschuwingssignaal laten klinken als feedback?

Ook kun je op basis van een lichtsluis meten hoeveel mensen er in een winkel zijn. Is het maximum bereikt kan er een stoplicht op rood springen.

Ook sport is interessant: stel je gaat hardlopen en je meet de tijd dat je daarover doet. Misschien kun je een apparaat maken dat jou feedback geeft over je rondetijden? Je hebt misschien wel eens bij het

schaatsen op TV gezien dat als iemand een wereldrecord lijkt te schaatsten dat je dan op het ijs een lijntje geprojecteerd ziet van die tijd.

De mogelijkheden zijn eindeloos. De uitdaging zit hem vooral snel iets te gaan bouwen en niet te lang bezig zijn om de perfect context te vinden. Voor deze module is het namelijk vooral belangrijk ermee aan de gang te gaan en in de praktijk te leren.

Hardware voor de verwerking van informatie in een digitaal systeem

Arduino Uno

In deze module gebruiken we vooral de Arduino Uno.

Een Arduino (Uno) is een klein board met programmeerbare elektronica. Met een Arduino kan je verschillende elektrische componenten aansturen. Denk aan lampjes, schakelaars, motortjes en verschillende sensoren.

De Arduino Uno Revision 3 - vaak **Arduino Uno r3** of **Arduino Uno rev3** genoemd - is het meest gebruikte board van [Arduino](#). Dit board is geschikt voor beginners en gevorderde gebruikers. Mede daardoor is de Arduino Uno zo populair.

De Uno is gebaseerd op de ATmega328 chip en heeft 14 digitale input/output pinnen. Verder zijn er 6 analoge input pinnen en een USB-aansluiting. Het board is eenvoudig aan te sturen door een programmeercode te schrijven (zie elders in deze module).



Arduino uno

Je ziet in de afbeelding hierboven in rood vierkant de genummerde digitale poorten. Deze worden vaak gebruikt om aangesloten outputs (lampjes, motortjes e.d.) aan of uit te zetten). Let op: poort 0 en 1 hebben een speciale functie.

Verder zie je in een groen vierkant de analoge poorten. Waar een digitale poort alleen 1 of 0 als waarde kan aannemen, kan een analoge poort veel meer waarden aannemen. We komen hierop terug bij de verschillende sensoren.

In blauw hebben we de 2 verschillende spanningen. Soms heb je systemen die om 5V vragen en soms om 3.3V (dit is de plus).

In geel heb je de aarde (of de min) aangegeven met GND.

Arduino Nano

De Arduino Nano is vergelijkbaar met de Uno. Hij is wat kleiner, wat een voordeel is in ontwerpen die heel klein of licht moeten zijn. Het nadeel is dat hij wat minder krachtig is en minder digitale poorten heeft.



Arduino Nano

Arduino Mega

De Arduino Mega 2560 Rev3 is een microcontroller board gebaseerd op de ATmega2560. Het heeft 54 digitale input / output poorten en 16 analoge poorten. Daarmee heeft hij meer mogelijkheden dan de Uno. Omdat we meestal bij onze projecten niet zoveel poorten nodig hebben gebruiken we deze niet standaard.



Micro:bit

tekst

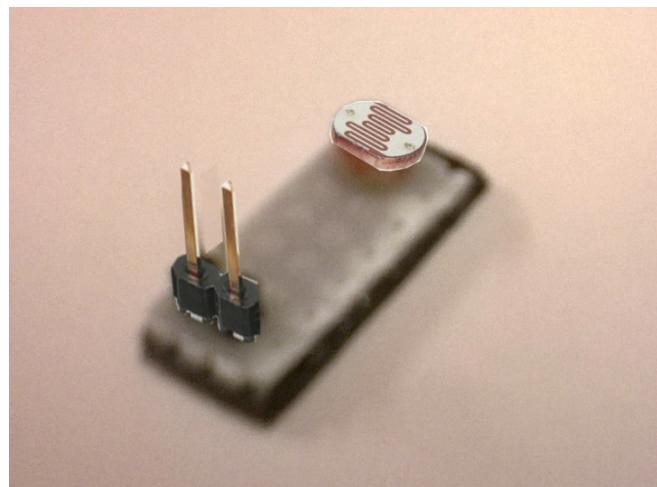
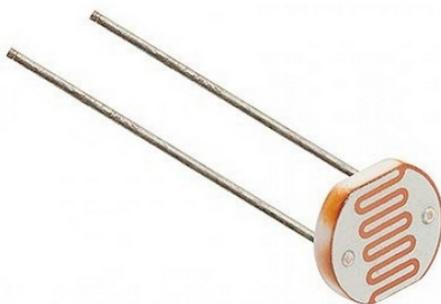
Raspberry Pi

tekst

Hardware voor de input van een digitaal systeem

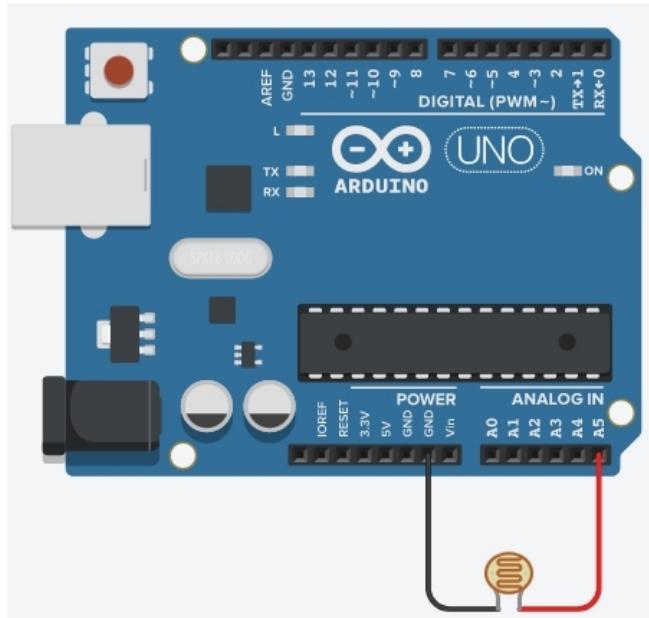
LDR sensor

Een **lichtgevoelige weerstand** of **LDR** (*light-dependent resistor*) is een elektrische component waarvan de weerstand beïnvloed wordt door de hoeveelheid licht die erop valt. De weerstandswaarde van een LDR wordt kleiner, naarmate de LDR sterker wordt belicht. Dat betekent dat als je een LDR aansluit op je analoge poort én de poort programmeert met *Pull-Up naar Vcc*, je de waarde van de spanning zult zien afnemen als er meer licht op de LDR schijnt. Wil je dat de spanning stijgt bij grotere lichtsterkte, dan heb je een *Pull-Down* weerstand naar GND nodig. Die heeft de Arduino niet aan boord dus die plaats je als fysieke weerstand extern.



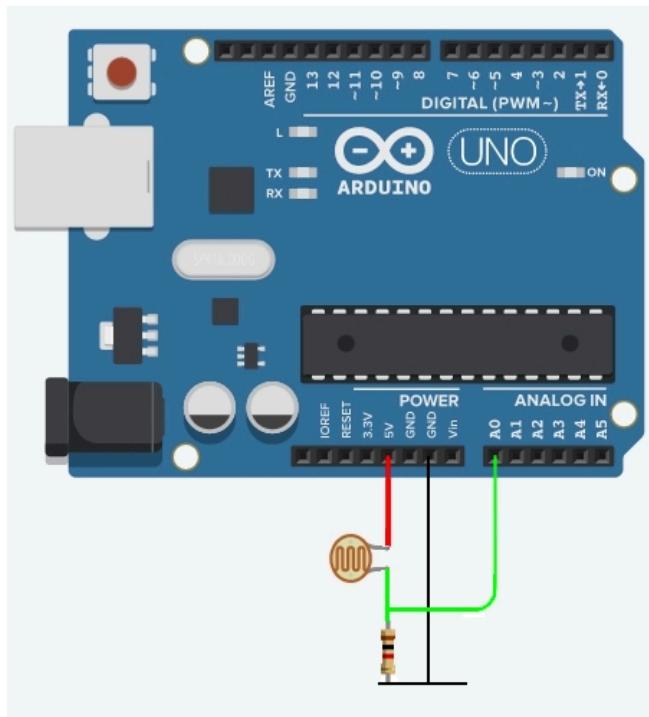
Omgekeerd evenredige spanning met lichtsterkte

De LDR sluit je aan door een draad van de GND aan het ene uiteinde te verbinden (hier zwart) en het andere uiteinde met een willekeurige analoge poort te verbinden (hier rood). Poort programmeren met Pull-Up.



Evenredige spanning met lichtsterkte

De LDR sluit je aan door een draad van de Vcc aan het ene uiteinde te verbinden (hier rood) en het andere uiteinde met een willekeurige analoge poort te verbinden (hier groen). Ook plaats je een fysieke weerstand tussen de analoge poort en de GND.

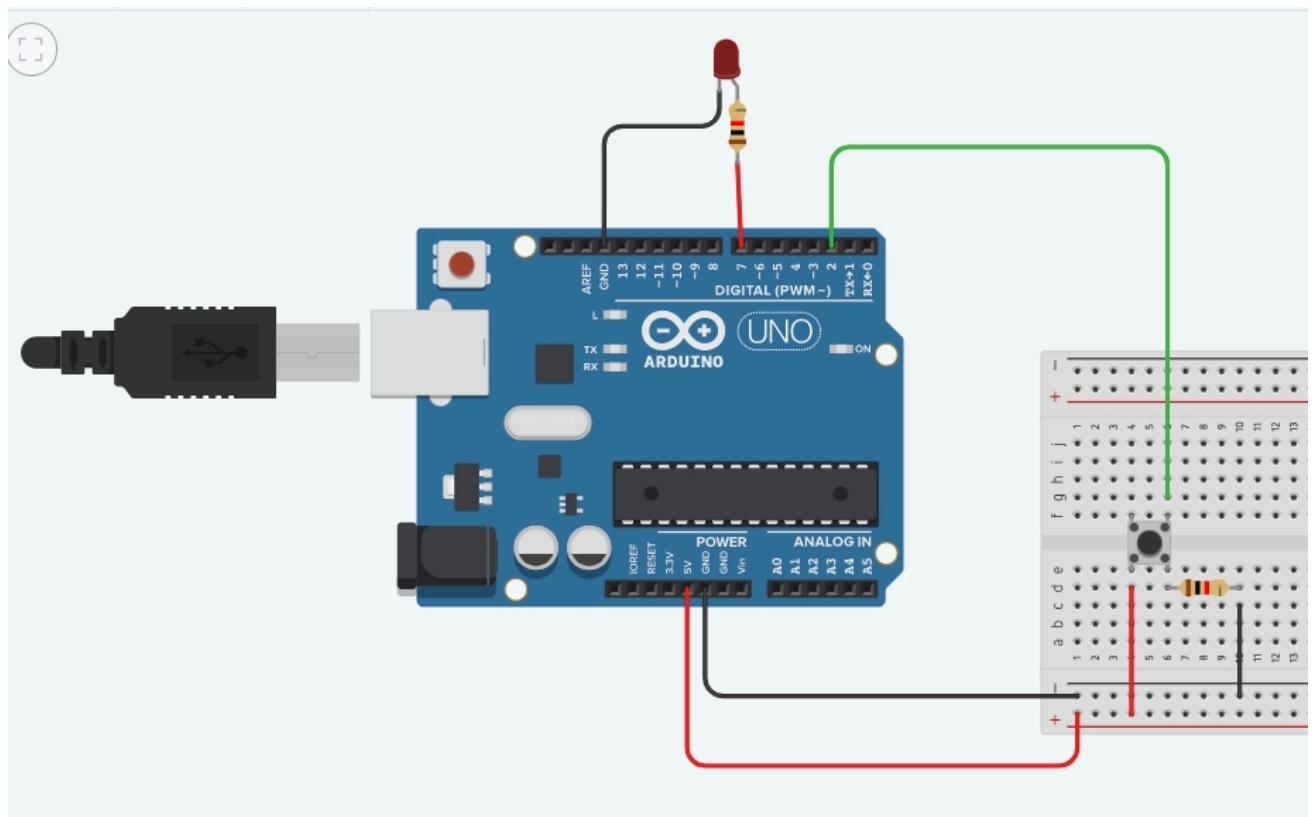
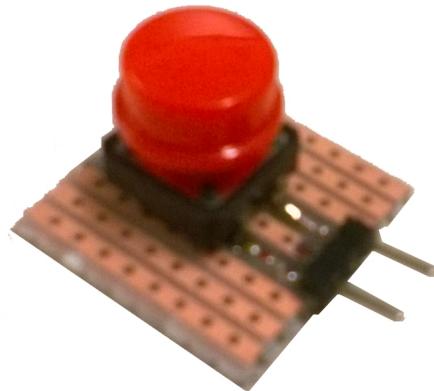


Momentdrukknop

Een drukknop is een veelgebruikte input. Je drukt op een knop en er gebeurt iets. De drukknop is gebaseerd op dat er als de knop ingedrukt is er een stroom doorheen kan lopen die minder weerstand ondervindt dan de route direct terug naar de min draad. In de schakeltekening zie je dat als de knop niet

is ingedrukt, de stroom niet door de knop kan en dan via de weerstand met de zwarte draad naar de min gaat. Als de knop wel is ingedrukt gaat het door de groene draad (via een weerstand, anders verbrandt de LED) naar de LED, die dan dus aan gaat.

Je moet in de code aangeven in welke poort het signaal van de drukknop binnenkomt (in de schakeling is dat digitale poort 2). Hieronder zie je ook een voorbeeldcode passend bij de schakeltekening.



Een schakeling waarmee een drukknop een LED bedient

```
// Pin 2 has an pushbutton connected on most Arduino boards.
```

```
// give it a name:
```

```
int pushButton = 2;
```

```
int led = 7;
```

```
// the setup routine runs once when you press reset:
```

```
void setup() {
```

```

//initialize serial communication at 9600 bits per second
Serial.begin(9600);
//make the pushButton's pin an input:
pinMode(pushButton,INPUT);
pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {

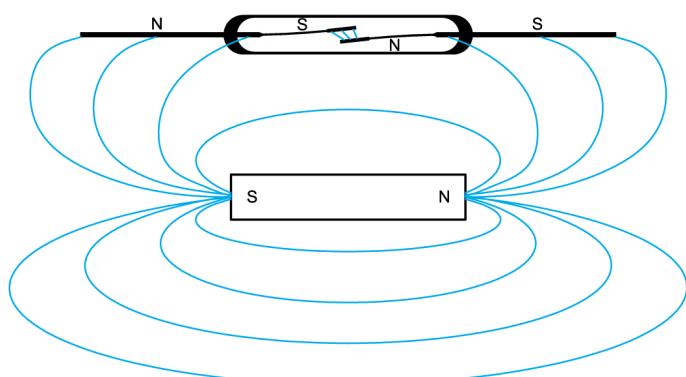
//read the input pin
int buttonState = digitalRead(pushButton);
Serial.println(buttonState);
delay(1);
//read the output pin
if(buttonState > 0 ) {
digitalWrite(led,HIGH);
delay(1000);
digitalWrite(led,LOW);
delay(1000);
} else
{}
}

```

Magneetcontact

Een magneetcontact (ook wel *magneetschakelaar* of *reedswitch* genoemd) is een schakelaar die gevoelig is voor magnetisch veld. De schakelaar kent twee posities; hij is Open of Dicht, oftewel Hoogohmig of Laagohmig (zoals je misschien weet is Ohm de eenheid voor weerstand bij elektriciteit). Dat betekent dat er een open dan wel gesloten circuit is. De werking is niet anders dan de momentdrukknop. het contact is gesloten zolang het magneetveld aanwezig is.

Het magnetisch veld breng je aan in de vorm van een magneet. In onderstaande afbeelding zie je de werking van het magneetveld. Het contact zal zich sluiten door de aantrekkingskracht.





Magneetcontact in naakte vorm.

Waar je bij de drukknop nog een tweede route voor de elektriciteit moet bieden, hoeft dat bij het magneetcontact niet. De ene kant kan aangesloten op de GND, en de andere op een digitale poort. Het nummer van die poort is dan in je Arduinocode de locatie voor de input.

Thermometer (LM35)

De LM35 is vergelijkbaar met de TMP36 temperatuursensor. Je ziet hieronder een afbeelding van de drie aansluitingen die de sensor heeft: *voedingsspanning*, *ground* én het *sensorsignaal*.

Er zijn sensoren die voorzien zijn van een lange aansluitkabel.

Rood gemerkte draad = **+5Volt**.

Zwart gemerkte draad = **GND**.

Draad zonder markering (of andere kleur) = Signaal.

In de datasheet vind je de verhouding van het signaal t.o.v. de temperatuur. Bekijk ook de link naar een voorbeeldschakeling, waar je ook de code kunt zien. Er is een TMP36 sensor aangesloten en 3 LED's. Als je op simulatie starten klikt, kun je door op de sensor te klikken de temperatuur verhogen en zien dat de LED's bij hogere temperatuur één voor één aan gaan.



Tinkercad TMP36

<https://www.tinkercad.com/things/94VFDq4ienj-copy-of-tmp36-temperatur-e-sensor-with-arduino/editel?tenant=circuits>

De code die je kunt gebruiken:

```
int baselineTemp = 0;
int celsius = 0;
int fahrenheit = 0;

void setup()
{
pinMode(A0, INPUT);
Serial.begin(9600);

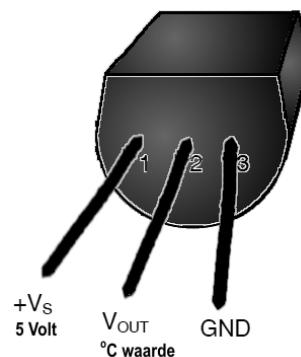
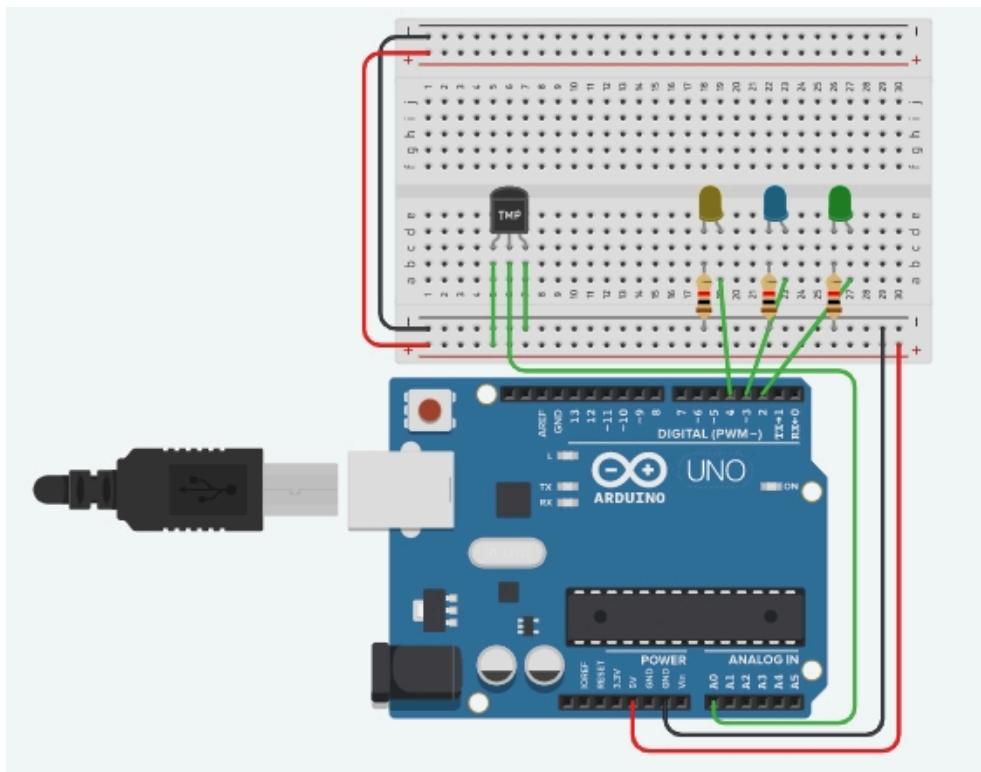
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
}

void loop()
{
baselineTemp = 40;
```

```
celsius = map(((analogRead(A0) - 20) * 3.04), 0, 1023, -40, 125);

fahrenheit = ((celsius * 9) / 5 + 32);
Serial.print(celsius);
Serial.print(" C, ");
Serial.print(fahrenheit);
Serial.println(" F");

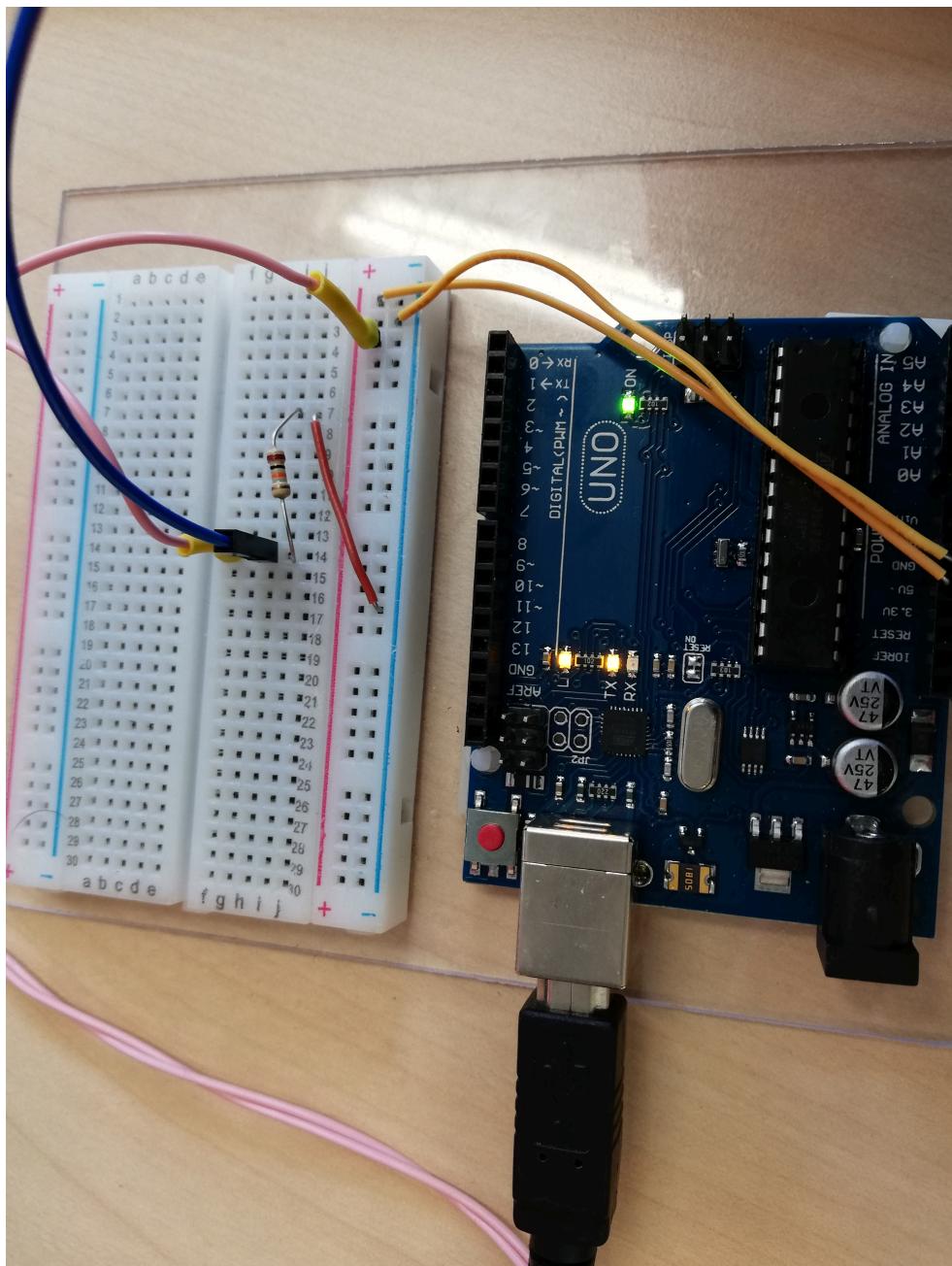
if (celsius < baselineTemp) {
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
}
if (celsius >= baselineTemp && celsius < baselineTemp + 10) {
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
}
if (celsius >= baselineTemp + 10 && celsius < baselineTemp + 20) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
}
if (celsius >= baselineTemp + 20 && celsius < baselineTemp + 30) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
}
if (celsius >= baselineTemp + 30) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
}
delay(1);
}
```



LM35 onderaanzicht



[LM35 Datasheet](#)



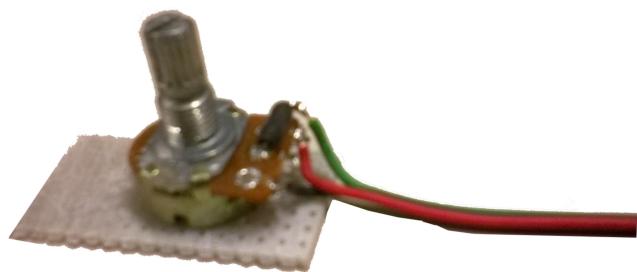
Potentiometer

De potentiometer is een weerstand waarvan je de waarde kunt variëren. Wanneer je aan de as draait verandert de positie van een loper op de weerstandsbaan (vaak carbon). Met de draaiknop kun je de spanning aan de loper regelen. De werking hiervan en het aansluiten ervan wordt in de onderstaande video uitgelegd.

Aan de potmeter die wij gebruiken is een extra weerstandje toegevoegd om te voorkomen dat de potmeter verbrand bij verkeerd aansluiten (er zou dan kortsleuteling kunnen ontstaan).



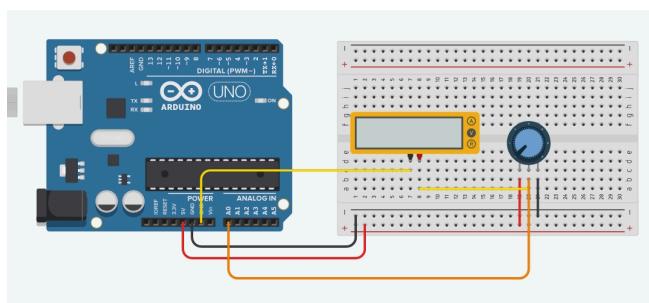
[Potentiometer](#)



Aansluitingen

De **groene** draad is verbonden met de *loper* (via extra weerstandje van 2kOhm). De **rode** is verbonden met de uiterste kant met de klok mee (*maximum*). De **zwarte** draad zit aan de nulpositie, de uiterste kant tegen de klok in (*minimum*).

Onze potmeter heeft een waarde van 10kOhm met de loper van 2kOhm.



Aansluiting van de de potentiometer met willekeurige kleuren draad(!)

Microfoon

Een microfoon is een sensor (of opnemer) die gevoelig is voor *geluidsgolven*.

Wij maken gebruik van een module met het MAX9814 circuit met de volgende mogelijkheden:

- * vast instellen van versterking
- * automatische versterking met instelbare attack / release
- * nauwkeurige opname in het frequentiegebied tussen 20Hz en 20kHz

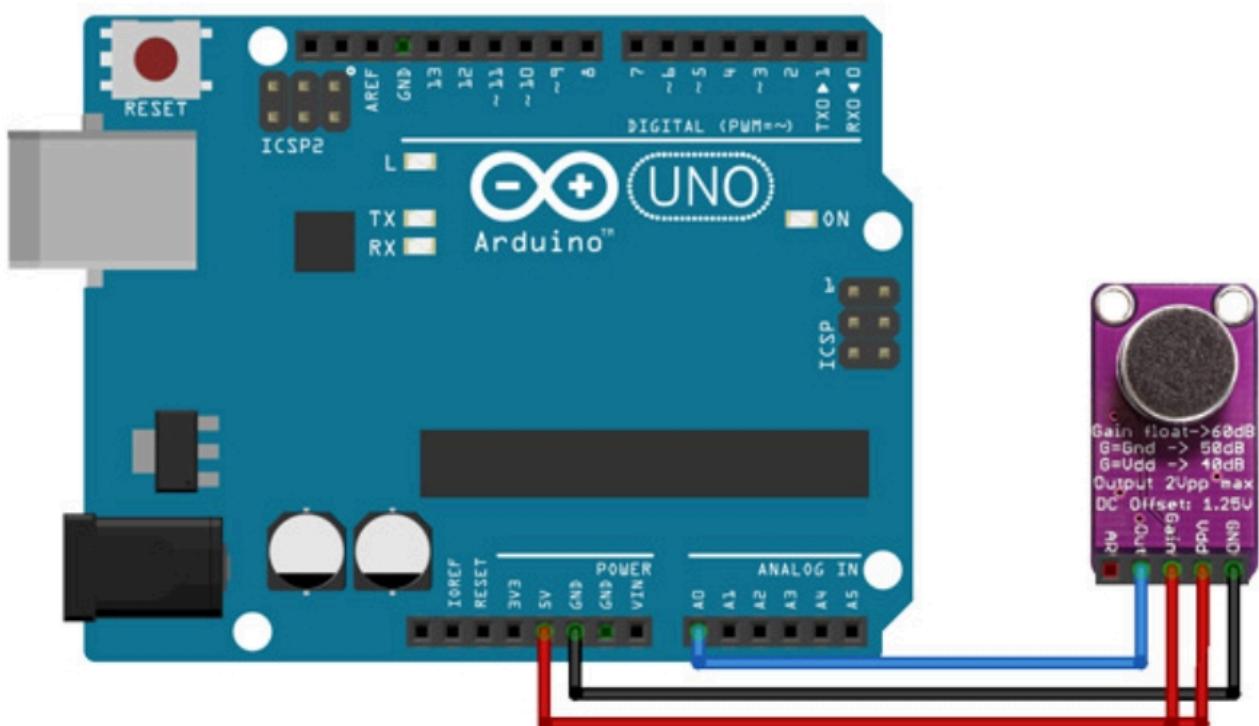
Bekijk de *datasheet* van de MAX9814 voor een schat aan informatie over de mogelijkheden.



<https://maken.wikiwijs.nl/userfiles/4/44fb5180baa209c6ad024ef21086108b1fb911c1.pdf>

Hieronder zie je een aansluitschema voor de microfoon. Een microfoon is eigenlijk een vrij simpele analoge input, en dat zie je ook aan de code. Er wordt een analoge input uitgelezen en die wordt in deze code geplot op de seriële plotter (bij hulpmiddelen, let op: de y-as wordt steeds automatisch geschaald). Let op: wat je hier ziet is de spanning die de microfoon afgeeft aan de Arduino, de golven die je ziet verschijnen zijn niet de geluidsgolven maar een elektronische representatie daarvan.

Als je echt de microfoon als input wilt gebruiken zul je nog een flinke bewerking met de gegevens moeten uitvoeren. Dat kan vrij complex zijn, en je moet wat weten over geluid. We zullen hier t.z.t. meer informatie over opnemen.



Een aansluitschema voor de microfoon die wij gebruiken

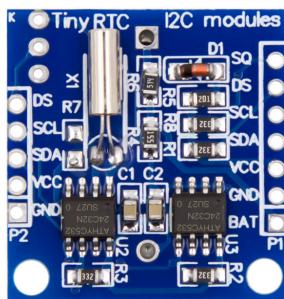
Klok

De Arduino beschikt over een ingebouwde klok die is aan te roepen met het commando `millis()`. Deze klok is handig om incidenteel een korte periode te meten, een wachttijd in te stellen of een teller op te hogen.

Wil je nauwkeurig een tijdseenheid kunnen toepassen en wil je daarin flexibeler zijn dan is het verstandig om een externe klok te gebruiken, een *Real Time Clock*.

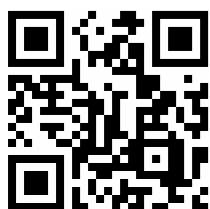
Met zo'n klok beschik te over de werkelijke tijd en vaak ook de datum en zelfs een meerjarenkalender. Je leest hem uit op het moment dat je de gegevens nodig hebt zodat je je in de programmacode met belangrijker zaken kunt bezighouden. Met een ingebouwde batterij (knoopcel o.i.d.) zorg je ervoor dat de klok blijft werken als de Arduino is uitgeschakeld.

Wij werken met de DS1307 chip en communiceren met de module via het **i2c protocol**.



Ultrasone sensor

Met de ultrasone sensor kun je de afstand tot een object meten. Het is een apparaat dat erg eenvoudig in gebruik is. Op de onderstaande site kun je voor het type HC-SR04 lezen hoe je hem moet aansluiten en gebruiken. Dit is het type dat wij op school hebben. Er zijn ook andere typen. Je ziet ook een filmpje hieronder met de ultrasone sensor in gebruik en een foto van hoe het er in de praktijk uit ziet.

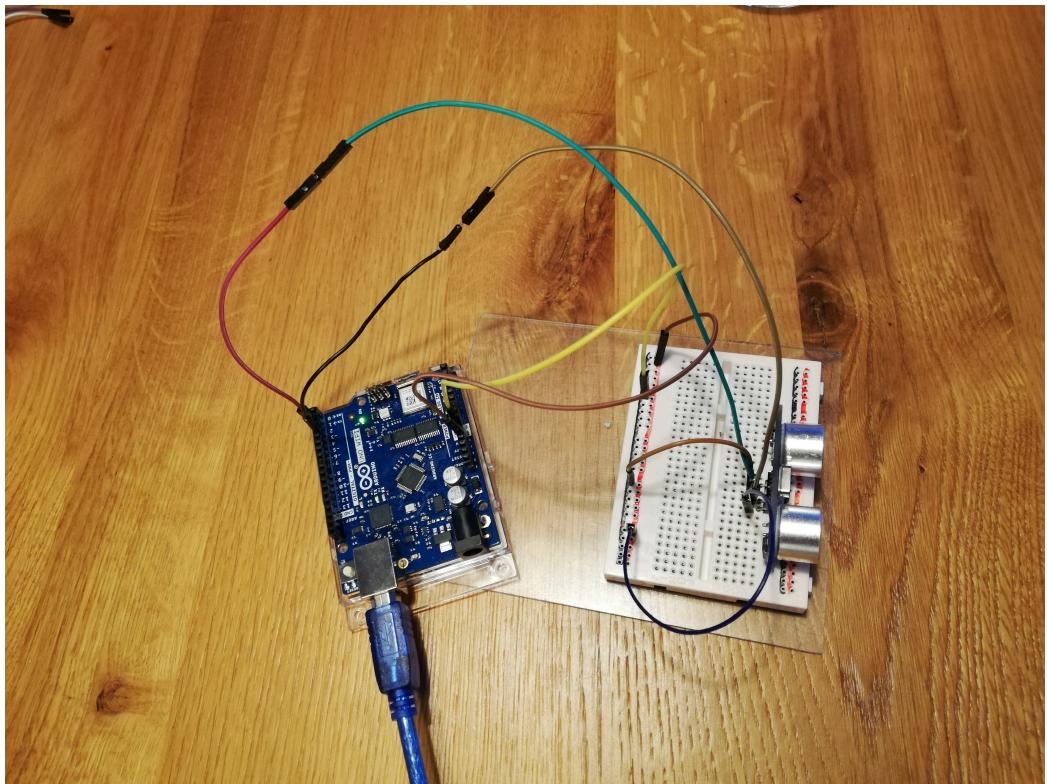


[Ultrasone sensor](#)



Tutorial voor de ultrasone sensor

<https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6>



Een voorbeeld van hoe een aangesloten ultrasone sensor eruit ziet

```
int trigPin = 9; // TRIG pin    Serial.begin (9600); void loop() {  
digitalWrite(trigPin, HIGH); duration_us = pulseIn(echoPin, HIGH); delay(500);
```

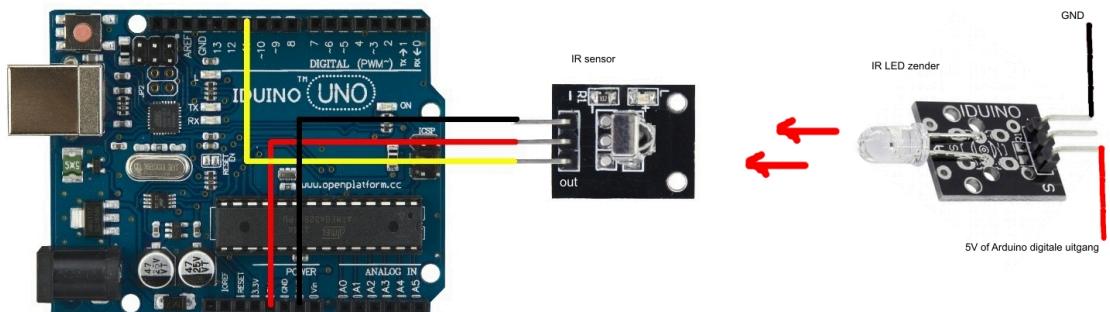
Vochtsensor

De vochtsensor is een sensor die gebaseerd is op de geleiding van stroom door (normaal) water. Als er meer water is, gaat die geleiding makkelijker, is het droog gaat de geleiding moeilijker.

Lichtsluis

Een lichtsluis is een hulpmiddel om aanwezigheid van objecten te kunnen detecteren. Een voorbeeld van een hele grote lichtsluis vind je bij een liftdeur. In de deuropening zit aan de ene zijde een lichtbron (al dan niet met sensor) en aan de andere zijde een reflector of een lichtsensor. Zodra een persoon zich tussen lichtbron en sensor bevindt zal de deur niet sluiten. Het licht is meestal onzichtbaar (infrarood).

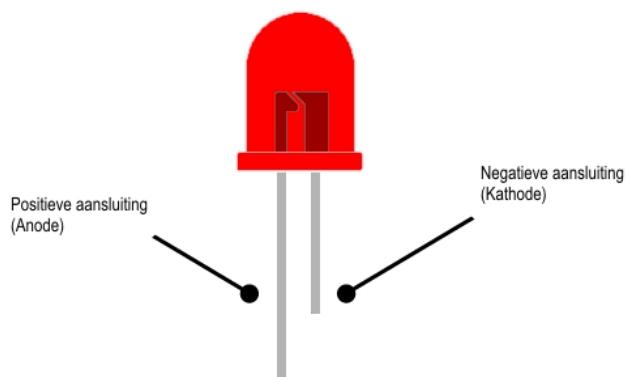
In het aansluitvoorbeeld zie je hoe je een lichtsluis kunt toepassen i.c.m. een Arduino.



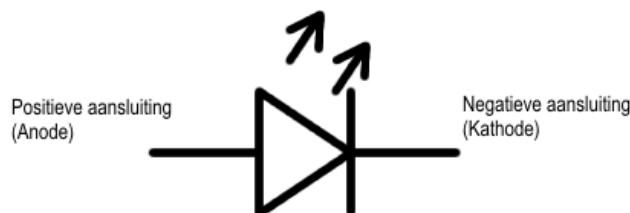
Hardware voor de output van een digitaal systeem

LED

Een LED is een diode die licht kan uitzenden. De afkorting staat voor Light Emitting Diode. Een diode is een elektronische component, een zogenaamde *halfgeleider*.



De LED wordt geleidend en gaat licht uitzenden zodra er een stroom in de juiste richting gaat lopen: de LED heeft een positieve en een negatieve aansluiting. Als de LED verkeerd om wordt gemonteerd loopt er geen stroom, de LED staat dan in *sperstand*.



Schematisch Symbool van de LED

Waar moet je op letten bij gebruik van een LED?

* Een LED is kwetsbaar, overschred de maximum stroom niet! Gebruik daarom ALTIJD een *voorschakelweerstand* in serie met de LED. Deze zal de stroom begrenzen op de gewenste waarde en de grootte van de benodigde weerstand is afhankelijk van aangeboden spanning. Het maakt niet uit of je de weerstand in de pluslijn of in de minlijn plaatst.

Als vuistregel mag je de volgende waarden hanteren:

- * Aangeboden spanning 5 Volt d.c. --> voorschakelweerstand 150 Ohm.
- * Aangeboden spanning 12 Volt d.c. --> voorschakelweerstand 560 Ohm.

Soms krijg je een LED waarbij al een weerstand in serie is gemonteerd, zie de foto voor een voorbeeld hiervan.



weerstand met waarde 150 Ohm



weerstand met waarde 560 Ohm

LED RGB

Neopixel LEDstrip

Neopixel LEDs zijn fantastisch. Het zijn LEDjes die alle kleuren en intensiteit kunnen aannemen en je kunt heel makkelijk een heleboel LEDs aansturen met een paar draadjes en wat slimme code.

Om Neopixels te gebruiken moet je een library installeren op je laptop. In die library zitten een heleboel commandos die speciaal zijn gemaakt voor de Neopixel LEDs. Hoe je de library installeert zie je op onderstaande website.



Arduino tutorial neopixel

<https://create.arduino.cc/projecthub/robocircuits/neopixel-tutorial-1ccfb9>

Op de bovenstaande site vind je ook het schema hoe je de Neopixel aansluit en hoe je met je code de verschillende 'adresseerbare' LEDs aan of uit zet en welke kleur en intensiteit ze moeten aannemen.

Met adresseerbaar bedoelen we dat je met de code verschillende LEDs kunt aansturen:

```
pixels.setPixelColor(i, pixels.Color(0, 150, 0));
```

Met bovenstaande code wordt nr i aangezet met een kleur die beschreven wordt met het pixels.Color commando. De 0,150,0 staat voor de intensiteit van de kleuren Rood Groen en Blauw (RGB). Dit zal dus een LED een milde groene kleur laten aannemen. De intensiteit kan van 0 tot 255 lopen.

Dus met:

```
pixels.setPixelColor(8, pixels.Color(255, 0, 0));
```

zetten we LED nr 8 op de strip aan met een felrode kleur.

Hieronder zie je een voorbeeld van een code waarmee je een Neopixel bestuurt. Dit betreft een Neopixel van 10 LED's, maar ook voor een Neopixel van 100 LED's kun je deze code gebruiken (alleen het getal 10 vervangen door 100).

```
#include <Adafruit_NeoPixel.h>
#ifndef __AVR__
#include <avr/power.h>
#endif
#define Neopixel 7
```

```

#define NUMPIXELS 10

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, Neopixel, NEO_GRB + NEO_KHZ800);
int delayval = 50;
void setup() {

pixels.begin();
}

void loop() {

for(int i=0;i<NUMPIXELS;i++){

pixels.setPixelColor(i, pixels.Color(0,150,0));
pixels.show();
delay(delayval);

}
for(int i=0;i<NUMPIXELS;i++){

pixels.setPixelColor(i, pixels.Color(0,0,0));
pixels.show();
delay(1);

}
}

```



[Neopixel voorbeeld](#)

LCD display

De processor (Arduino) in jullie ontwerp gaat acties uitvoeren aan de hand van signalen of gegevens die je laat *invoeren* (input). Is het op enig moment nodig om visueel informatie aan de buitenwereld kenbaar te maken dan kan dat met een eenvoudig lichtsignaal. Maar soms is de hoeveelheid informatie zo groot dat er meer nodig is. De uitkomst is dan een beeldscherm. Jullie kunnen daarvoor een LCD gebruiken. LCD staat voor Liquid Crystal Display. Dat is een stapeltje glasplaatjes waartussen vloeibare kristallen kunnen bewegen. De richting waarin de kristallen zich richten verandert door een kleine lading aan te brengen. Daardoor ontstaat een polariserend filter dat licht blokkeert. Een goed voorbeeld is je rekenmachine of een kwartshorloge; je ziet op dat schermpje cijfers verschijnen in zwart/wit.

Praktisch

De LCD die jullie kunnen gebruiken bevat 4 regels (20x04) waarop karakters kunnen worden getoond. Schrijven gebeurt via je programmacode en het is nodig dat je telkens aangeeft op welke regel en welke positie je begint met schrijven. Omdat een standaard LCD wordt aangestuurd met een behoorlijk aantal signalen is er een **i2c** circuit toegevoegd die dat voor zijn rekening neemt. Daardoor zijn er voor het opbouwen van het display *slechts vier draden* nodig: Voedingsspanning, Ground, SCL kloklijn en SDA datalijn.

Hieronder vind je een code die werkt. Het blijkt in de praktijk wel tricky te zijn om zo'n display werkend te krijgen, er staan namelijk een hoop voorbeelden op het web waar fouten in zitten. Daarnaast heb je

twee libraries nodig. Een library is een collectie van programmeercommando's die samen helpen om bepaalde taken uit te voeren. De libraries die je hier nodig hebt zijn wire.h en LiquidCrystal_I2c.h.

Wire.h is standaard geïnstalleerd, maar die andere niet. Via de link hieronder vind je een zip bestand. In de arduino omgeving kun je zo'n bibliotheek inladen vanuit een zip bestand (zie screenshot hieronder).

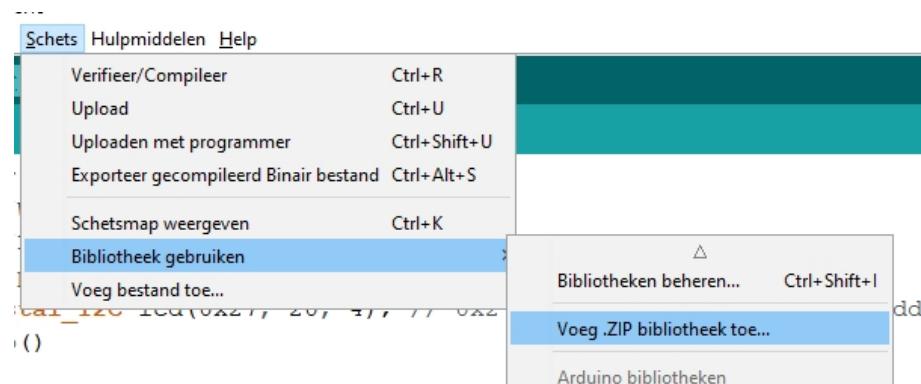


Een "kale" display met veel aansluitingen.

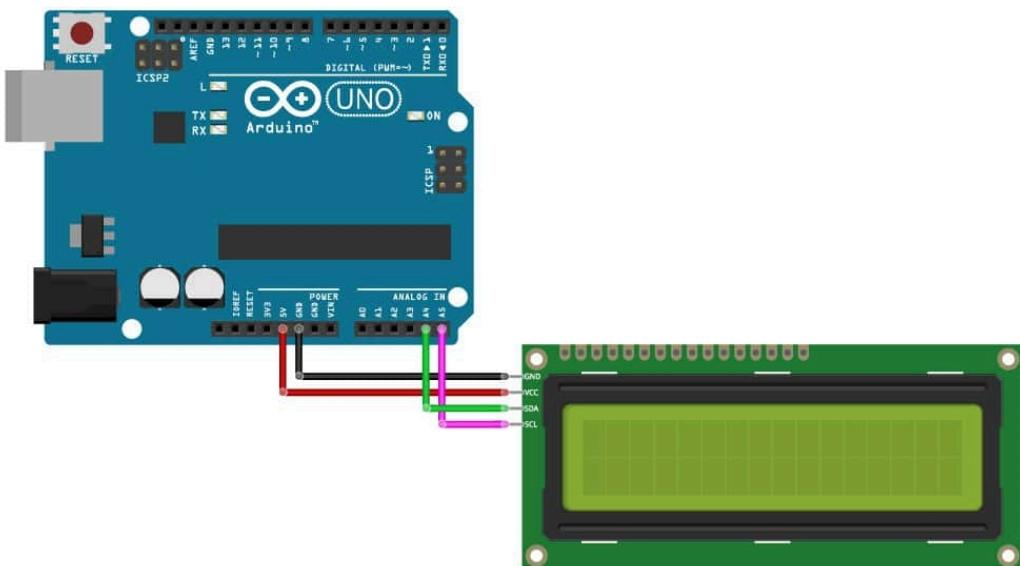


link naar liquidcrystal_i2c (datum link: 7-2-2022)

https://downloads.arduino.cc/libraries/github.com/marcoschwartz/LiquidCrystal_I2C-1.1.2.zip



Toevoegen van een zip-bibliotheek (library)



Display voorzien van i2c interface: aansluitvoorbeeld.

Een werkende code

Onderstaande code start de display, en laat de tekst 'hello world' zien. Daarna start er een teller die vrij rap oploopt (het geprinte 'nummer' wordt in elke loop met 1 opgehoogd).

```
int nummer = 0;
#include "Wire.h" // For I2C
#include "LiquidCrystal_I2C.h" // Added library*
//Set the pins on the I2C chip used for LCD connections
LiquidCrystal_I2C lcd(0x27, 20, 4); // 0x27 is the default I2C bus address
void setup()
{
// Set off LCD module
lcd.begin (20, 4); // 20 x 4 LCD module
// Turn on the blacklight and print a message.
//lcd.setBacklight(3,POSITIVE); // BL, BL_POL
lcd.setBacklight(HIGH);
lcd.setCursor(0, 3);
lcd.print("Hello, world!");
}
void loop()
{
nummer = nummer+1;
lcd.setCursor(2,2);
lcd.print(nummer);
}
```

Let op: bij mijn laptop blijkt ik de LCD display te moeten aansluiten op de 3,3V aansluiting ipv de 5V. Ook kan er een probleem ontstaan als je een verouderde versie van Arduino op je laptop hebt staan.

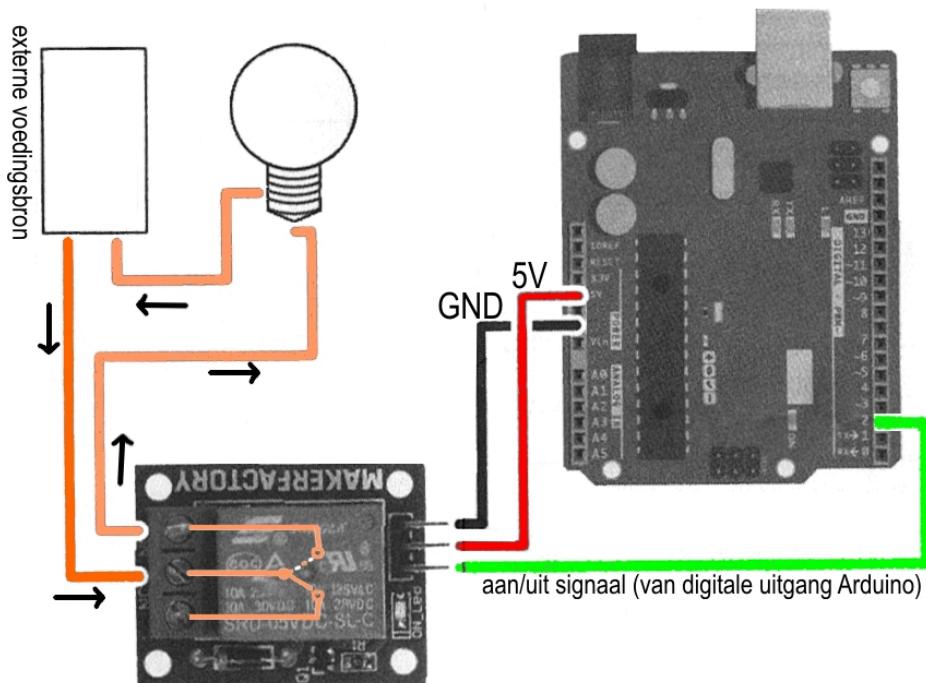
Relais

Relais

Een relais is een schakelaar om actuators te kunnen inschakelen en uitschakelen.

Je gebruikt een relais wanneer de Arduino zelf niet genoeg vermogen kan leveren voor de actuator.

Ook kun je een relais gebruiken om een circuit galvanisch gescheiden van de Arduino te kunnen schakelen, bijvoorbeeld wanneer een element op netspanning werkt, of op een andere spanning dan 5 Volt.



Aansluiten

Een relaismodule is beschikbaar met één relais of met meerdere relais.

De module wordt bediend met een digitaal stuursignaal vanaf een uitgangspin van de Arduino (HOOG / LAAG --> 5V / 0V).

Omdat het relais bekrachtigd moet worden moet je via een extra draad 5 Volt voeding aanbieden. Dat kan via de 5V pin van de Arduino. Let wel op dat deze pin begrensd is in maximale stroomcapaciteit. Gebruik je dus meerdere relais of gelijktijdig ook andere verbruikers op 5 Volt, dan kun je die beter rechtstreeks voeden uit een externe 5 Volt bron.

De relais schakelaar wordt magnetisch bekrachtigd en heeft in ons geval 3 aansluitpunten. Het is een wisselschakelaar. In onbekrachtigde toestand (in *rust*) is het *moedercontact* (*middelste aansluitpunt*) geschakeld aan contactpunt *N.C.* (*Normally Closed*). Zet je het relais "aan" dan schakelt het *moedercontact* over naar contactpunt *N.O.* (*Normally Open*). Op de printplaat zijn deze punten herkenbaar gemerkt.

Pulse Width Modulation PWM

Elektronische schakelaar in plaats van mechanisch Relais

In bepaalde situaties wil je een actuator (lamp / motor / klep o.i.d.) schakelen. Dat kan vaak met een relais (elektromagnetisch bekrachtigde schakelaar).

Maar er zijn toepassingen waarbij het **niet** met een relais kan.

Bijvoorbeeld het trapsgewijs schakelen naar een hoger vermogen lukt niet met een mechanische schakelaar, die staat óf AAN óf UIT, ALLES of NIKS.

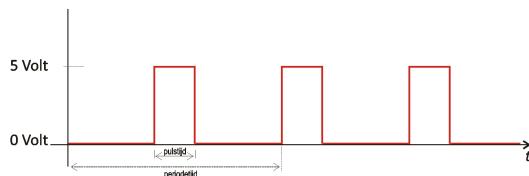
Een elektronische schakelaar kan veel sneller schakelen, bevat geen mechanische onderdelen en is dus heel geschikt om middels korte snelle pulsen te schakelen.

Dat geeft nieuwe mogelijkheden! Door met korte pulsen de schakelaar in- en uit te schakelen kan het *gemiddelde vermogen* geregeld worden. Zo'n pulserend regelsignaal heet *pulsbreedtemodulatie*, oftewel **Pulse Width Modulation PWM**.

Door bijvoorbeeld iedere seconde afwisselend aan- en uit te schakelen wordt er het *halve vermogen* toegevoerd, want 50% van de tijd is de schakelaar AAN en 50% van de tijd is deze UIT. Men zegt dan: de *Duty cycle* is 50%.

Maar: iedere halve seconde is een erg lage frequentie. Dan zou een lamp bijvoorbeeld erg vervelend knipperen, wel handig als knipperlicht, niet handig als je de lamp juist zou willen dimmen.

Dus, een bruikbaar PWM signaal heeft een veel hogere frequentie, want dan krijg je een heel goed regelbaar signaal.



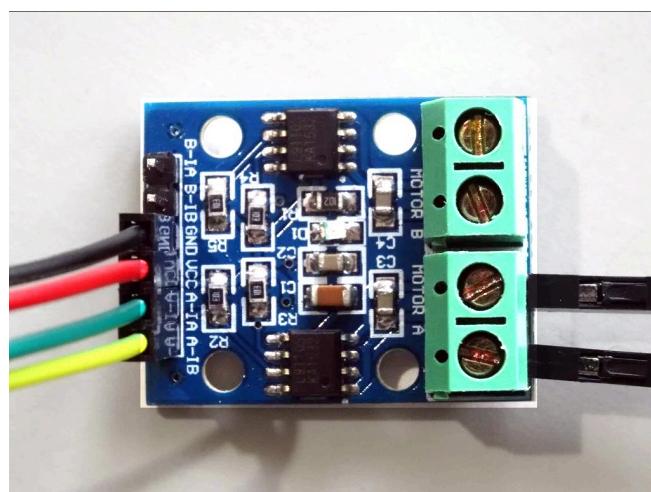
Hiernaast zie je een PWM signaal. Een digitale uitgang van de Arduino wordt op een *vaste frequentie* geschakeld tussen laag en hoog. De verhouding tussen de duur van laag en hoog bepaal je zelf. De duur van de puls (*hoog*) noem je de *PulsBreedte (PulseWidth)*. In dit geval is de *Dutycycle* 33%.

PWM poorten

De Arduino Uno bezit een paar poorten die je kunt gebruiken om een PWM signaal te genereren. Deze poorten werken automatisch op commando en staan op een vaste frequentie.

Arduino Uno pin 3, 5, 6, 9, 10 en 11 kunnen als PWM uitgang gebruikt worden.

(Pin 5 en 6 werken met een 980 Hz frequentie. Poorten 3, 9, 10 en 11 werken met 490 Hz.)



Solid State Switch (Elektronische Schakelaar) om 2 motoren te regelen.

Code voorbeeld Arduino UNO

Met het commando **analogWrite(naam,waarde)** geef je de opdracht om het signaal aan de uitgangspoort beschikbaar te maken. **naam** Is de naam die je aan de poort hebt gegeven. **Let er op dat niet alle uitgangen van de Arduino geschikt zijn voor een PWM signaal.**

waarde Is de gemiddelde waarde die je wilt dat het signaal heeft. Deze is instelbaar tussen 0% (UIT) en 100% (AAN).

formule: Duty cycle = (waarde / 255) [%]

Duty cycle 0% = 0

Duty cycle 50% = 127

Duty cycle 100% = 255

onderstaand een voorbeeldcode voor de combinatie Arduino / Chip L9110 / DC motor

Het IC L9110 heeft 2 ingangen waarmee je 2 uitgangen schakelt (de twee draden die naar de motor gaan). Het IC schakelt volgens een waarheidstabell:

inputA inputB outputA outputB motor

0 0 0 0 uit

1 0 1 0 rechtsom

0 1 0 1 linksom

1 1 1 1 uit

```
// pwm analogWrite op pin D05 490Hz  
// pwm analogWrite op pin D06 490Hz  
const int LINKS = 5; // motor pwm signaal op pin D04  
const int RECHTS = 6; // motor pwm signaal op pin D05
```

```
void setup()  
{  
pinMode(LINKS, OUTPUT); //  
pinMode(RECHTS, OUTPUT); //  
}
```

```
void loop()  
{  
analogWrite(LINKS,0);  
analogWrite(RECHTS,0);  
delay(500); // wacht xx ms
```

```
analogWrite(RECHTS,125);  
delay(2000); // wacht xx ms
```

```
analogWrite(RECHTS,200);  
delay(2000); // wacht xx ms
```

```
analogWrite(RECHTS,255);  
delay(2000); // wacht xx ms
```

```
analogWrite(LINKS,0);  
analogWrite(RECHTS,0);  
delay(500); // wacht xx ms
```

```
analogWrite(LINKS,200);  
delay(2000); // wacht xx ms  
}
```

Ventilator

DC motor (lineaire motor)

DC motor

Een standaard DC motor (gelijkstroom elektromotor) wordt toegepast in allerlei soorten aandrijvingen. Denk aan een accuboormachine of een polijstmachientje bij de pedicure bijvoorbeeld.

De motor schakel je *niet* rechtstreeks vanuit de poort van de Arduino maar je gebruikt tussen Arduino en motor een relais of een solid-state-switch. Hiermee schakel je dan een *externe voeding* aan- of uit.



Voorbeeldschakeling met Arduino, Relais en Lamp.
https://maken.wikiwijs.nl/152360/NLT__Digitale_Techniek_TCC#Ipage-7105936

Linksom of Rechtsom?

De dc-motor draait altijd in één richting, afhankelijk van de polariteit (+ pool en - pool) van de aangeboden voedingsspanning. Draai je de polariteit om, dan gaat de motor in de andere richting draaien.

Door het toepassen van *twee relais* wordt het mogelijk om de draairichting van de motor te kiezen.

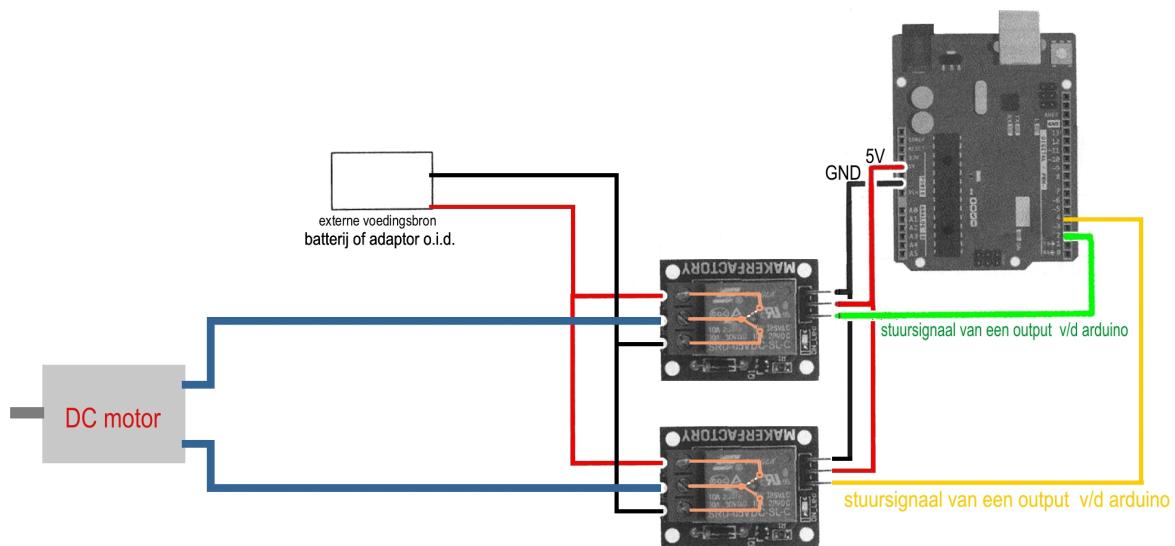
Zie onderstaande voorbeeldschakeling. Naast de twee relais heb je uiteraard ook een externe voeding nodig én je gebruikt een extra uitgang van de Arduino.

Hoe werkt het?

In rust staan beide relais in de ruststand. De motor is dan met beide aansluitingen verbonden met de - pool van de voeding.

Er gebeurt dan dus niks.

De truuuk zit hem in het afzonderlijk aansturen van één van de relais. Dan krijgt de motor energie toegevoerd, via het ene relais zal de motor met de klok mee draaien, met het andere relais tegen de klok in!



Stappenmotor

Pomp

Buzzer

De buzzer is een soort luidspreker. Deze kan een geluid produceren. Vaak een pieptoon of een zoemtoon, die ontstaat doordat een membraan gaat trillen onder invloed van een spanning of stroom.

De buzzer die hier als voorbeeld is afgebeeld kan direct worden aangestuurd door een digitale uitgang van de Arduino, 5 Volt gelijkspanning, dus let op de positieve en negatieve (GND) aansluiting.



Miniatuurbuzzer

Elektromagneet

Hardware voor de opslag en het transport van data

Draadjes (en ze bevestigen)

i2c

Sensoren en Actuatoren slimmer aansluiten in de praktijk

Wanneer je al wat schakelingen hebt bedacht en gebouwd aan de hand van de oefenopdrachten, zul je gemerkt hebben dat deze kunnen uitmonden in een wirwar aan verbindingen en draadjes. Vooral als je wat meer met de Arduino gaat doen dan alleen een temperatuur meten of een lampje aan- of uitzetten.

Zodra je project groeit krijg je te maken met ingewikkelder sensoren, meer ingangssignalen en ook meer uitvoer.

Voor weergave van gegevens kun je bijvoorbeeld gebruik maken van een display. Voor opslag op een geheugenkaart zijn er mooie insteekmodules. Typisch onderdelen die veel verschillende signalen nodig hebben om te kunnen werken.

Om je project overzichtelijk te houden, slimmer op te zetten, beter te kunnen onderhouden en makkelijker te kunnen ontwerpen ga je gebruik maken van een communicatieprotocol. Hiermee zijn een heleboel moeilijke ontwikkelstappen al gedaan en kun je op relatief eenvoudige manier grote slagen gaan maken.

Het communicatieprotocol waarvoor de Arduino al is voorbereid is het i2c protocol (spreek uit als "i kwadraat c").

Dit systeem werkt met maar 2 signaallijnen om gegevens uit te wisselen tussen processor en sensor of actuator. Samen noemen we deze twee draden de "bus". In dit geval dus de i2c bus.

i2c is een seriële bus, omdat alle informatie in blokken van een aantal bits achter elkaar wordt getransporteerd. Eén draad is de referentie en voert een kloksignaal, de kloklijn: SCL (Serial CLock). De andere draad voert de informatie, de data en heet de datalijn: SDA (Serial DAta). Beide lijnen kennen twee niveaus, een logische "1" en een logische "0".

Een heel groot praktisch voordeel van deze communicatiebus is dat er een hele grote hoeveelheid verschillende sensoren en actuatoren parallel aan deze twee draden kunnen worden aangesloten. Dat scheelt enorm veel werk en bespaart veel hardware (bedrading).

Het protocol zorgt er voor dat alle aangesloten eenheden herkenbaar zijn aan een adres. Bij i2c bestaat elk adres uit 7 bits (een getal bestaande uit 7 nullen en/of enen). Met deze methode kun je dus maar liefst 128 sensoren/actuatoren verbinden met de Arduino en dat met maar twee draden! (Ga na of je begrijpt waarom er maximaal 128 unieke adressen bestaan bij i2c).

Geschikt voor i2c ?

Niet alle randapparaten die wij beschikbaar hebben zijn voorbereid voor i2c. Let hier op door van elke sensor of actuator die je wilt gebruiken bijbehorende informatie te zoeken! (uit de datasheet of andere bronnen).

Master en Slave

i2c werkt op basis van een Master-Slave constructie. De processor (Arduino) is de baas, de Master. De sensor of actuator is de slaaf (Slave). Dat houdt in dat een sensor of actuator pas actie onderneemt nadat de master daarvoor de opdracht heeft gegeven. Het zou anders een onbegrijpelijke brei van signalen worden op de bus.

Pull-Up

De beide buslijnen hebben in de ruststand een hoog niveau: logische "1". (5 Volt of 3,3 Volt, afhankelijk van de gebruikte voedingsspanning). Daarom moeten er altijd pull-up weerstanden worden gebruikt tussen de buslijnen en de voeding. Deze weerstanden plaats je bij een fysiek korte bus dicht bij de Arduino (je kunt ze ook via de code programmeren, dan hoef je geen externe weerstanden te paatsen). Als je een langere bus gebruikt (meerdere meters) óf deze zich bevindt in een omgeving met veel ruis (sterke stoorsignalen) dan plaats je meerdere weerstanden, verspreid over de totale lengte.

Als je I2C apparatuur aan gaat sluiten krijgt elk apparaat een uniek adres. Dit is nodig omdat alle info naar alle aangesloten apparaten via hetzelfde draadje moet. In je code moet je dan dus aangeven van elk aangesloten I2C apparaat wat zijn adres is. Maar hoe weet je dan het adres? Met onderstaande code scan je je systeem voor alle adressen van aangesloten I2C apparatuur. Voor een LCD display is het bijv. 0x27.

```
/ ----- /  
// Arduino I2C Scanner  
// Re-written by Arbi Abdul Jabbaar  
// Using Arduino IDE 1.8.7  
// Using GY-87 module for the target  
// Tested on 10 September 2019  
// This sketch tests the standard 7-bit addresses  
// Devices with higher bit address might not be seen properly.  
/ ----- /  
  
#include <Wire.h> //include Wire.h library  
  
void setup()  
{  
    Wire.begin(); // Wire communication begin  
    Serial.begin(9600); // The baudrate of Serial monitor is set in 9600  
    while (!Serial); // Waiting for Serial Monitor  
    Serial.println("\nI2C Scanner");  
}  
  
void loop()  
{  
    byte error, address; //variable for error and I2C address  
    int nDevices;  
  
    Serial.println("Scanning...");  
  
    nDevices = 0;  
    for (address = 1; address < 127; address++ )  
    {  
        // The i2c_scanner uses the return value of  
        // the Write.endTransmisstion to see if  
        // a device did acknowledge to the address.  
        Wire.beginTransmission(address);  
        error = Wire.endTransmission();  
  
        if (error == 0)  
        {  
            Serial.print("I2C device found at address 0x");  
            if (address < 16)  
                Serial.print("0");  
        }  
    }  
}
```

```

Serial.print(address, HEX);
Serial.println(" !");
nDevices++;
}
else if (error == 4)
{
Serial.print("Unknown error at address 0x");
if (address < 16)
Serial.print("0");
Serial.println(address, HEX);
}
}
if (nDevices == 0)
Serial.println("No I2C devices found\n");
else
Serial.println("done\n");

delay(5000); // wait 5 seconds for the next I2C scan
}

```

(micro) SD card reader

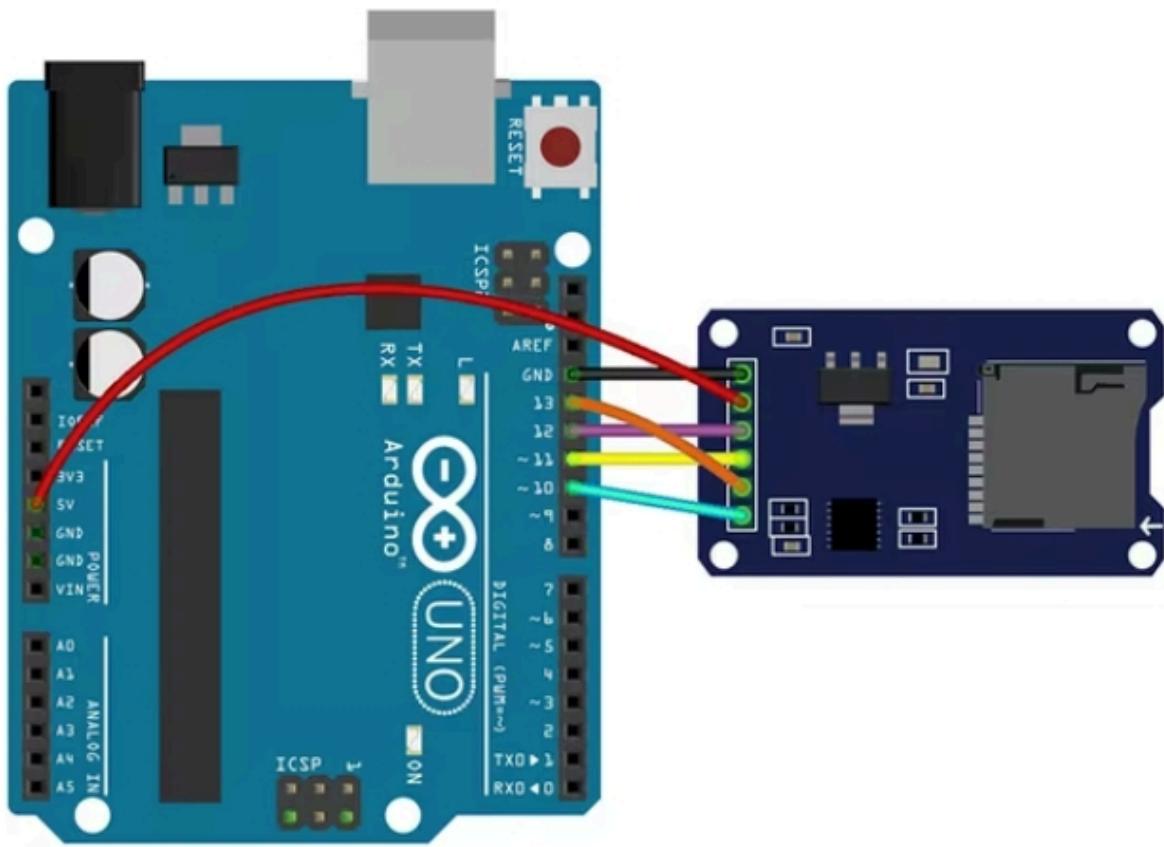
Als je systeem gegevens verzamelt wil je die bewaren. De makkelijkste manier om dit te doen is door het weg te schrijven naar een (micro)SD kaart. Deze zijn makkelijk aan te sluiten op de Arduino en met een cardreader kun je vervolgens de gegevens op de computer bekijken.

Het vergt wel wat voorbereiding. De gegevens schrijf je namelijk weg, zoals je gegevens naar de seriële monitor schrijft. Op de SD kaart wordt een tekstbestandje opgeslagen met daarin de tekst die je naar dat bestand toeschrijft.

Stel dat je een reeks metingen wilt wegschrijven is het bijv. belangrijk om elke meting steeds op een nieuwe regel te beginnen, en andere informatie die belangrijk is voor de meting (bijv. het tijdstip) op dezelfde regel op te schrijven. Verschillende soorten informatie kun je scheiden door spaties, maar beter is het om dit te doen door puntkomma's. Je kunt dan in Excel het bestand openen en de kolommen scheiden op basis van puntkomma's (wordt uitgelegd).

Het goed gebruiken van de SD kaart is een van de pittigste onderdelen van je systeem, maar het is het wel het onderdeel dat het systeem meer maakt dan een dom apparaatje. Nu kun je er namelijk onderzoek mee doen en daarmee je apparaat veel nauwkeuriger evalueren, wat een belangrijke stap is in de ontwerpcyclus.

Daarnaast is de code voor het wegschrijven naar de SD kaart best complex (zie onder). Het vergt enige oefening om al je andere code op de juiste plek in de code neer te zetten waardoor alle verschillende onderdelen samen werken. In hoofdstuk 8 ga je hier vanzelf een voorbeeld van zien. Omdat jullie allemaal iets anders gaan maken, zal voor iedereen de uiteindelijke oplossing anders zijn.



Aansluitschema voor een micro-sd cardreader

```
#include <SPI.h>
#include <SD.h>
File myFile;
void setup() {
// Open serial communications and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
Serial.print("Initializing SD card...");
if (!SD.begin(10)) {
Serial.println("initialization failed!");
while (1);
}
Serial.println("initialization done.");
// open the file. note that only one file can be open at a time,
// so you have to close this one before opening another.
myFile = SD.open("test.txt", FILE_WRITE);
// if the file opened okay, write to it:
if (myFile) {
Serial.print("Writing to test.txt... ");
myFile.println("This is a test file :)");
myFile.println("testing 1, 2, 3.");
for (int i = 0; i < 20; i++) {
myFile.println(i);
}
}
```

```
}

// close the file:
myFile.close();
Serial.println("done.");
} else {
// if the file didn't open, print an error:
Serial.println("error opening test.txt");
}
}

void loop() {
// nothing happens after setup
}
```

Wifi chip

Voeding voor een digitaal systeem

Arduino, sensoren en actuatoren voeden

USB stekker (5v)

Digitale systemen instructies geven (programmeren)

Programmeren

Een Arduinocode noemen we een sketch. Deze sketch wordt omgezet naar computertaal in enen en nullen (de binaire code, zie hierboven). Wij hoeven ons niet druk te maken over de binaire code, maar moeten wel weten hoe we zo'n sketch schrijven.

Een sketch maakt gebruik van **objecten**. Wat zijn objecten? Een object is iets dat je in een code gedefinieerd hebt en dat een geheugenplaats inneemt. Als je bijvoorbeeld in je code iets wilt doen met een **LED** (aanzetten of uitzetten) moet je wel tegen die Arduino vertellen dat er zoiets als een **LED** is. Dat slaat de Arduino dan op in zijn geheugen zodat hij op een later moment weet wat jij bedoelt met het woordje **LED**. Het woordje **LED** snapt de Arduino niet, dus je moet hem vertellen dat het woordje **LED** iets te maken heeft met de poort waar de **LED** op aangesloten is, en dat dit een **OUTPUT** is. Er komt namelijk licht uit. Hier komt de structuur van een sketch om de hoek kijken.

De structuur van een code ziet er globaal zo uit:

1. < ruimte voor definiëren van objecten >
2. < ruimte voor het duidelijk maken wat de objecten zijn, dit wordt eenmalig uitgevoerd >
3. < ruimte waar verteld wordt wat er met de objecten moet gebeuren, dit wordt steeds opnieuw uitgevoerd >

Het eerste gedeelte heeft geen naam, het 2^e gedeelte noemen we de **void setup**, het 3^e gedeelte noemen we **void loop**. We gaan dit verduidelijken met behulp van onderstaande figuur

```

1 const int LAMP = 8;
2 const int KNOOP = 2;
3
4 void setup()
{
5     pinMode(LAMP, OUTPUT);
6     pinMode(KNOOP, INPUT);
7 }
8
9
10 void loop()
11 {
12     int KnopStatus = digitalRead(KNOOP);
13     if(KnopStatus==1){
14         digitalWrite(LAMP, HIGH);
15     } else {
16         digitalWrite(LAMP, LOW);
17     }
18     delay(100);
19 }
```

Voorbeeld van een code

Deel 1 - Objecten definiëren

Je ziet in regel 1 en 2 dat er wordt verteld dat de **LAMP** gelijk is aan 8 en de **KNOOP** gelijk is aan 2. Aan het begin van de regel staat het woordje **const**, dat betekent dat de waarde van dit object niet kan veranderen tijdens het programma. Stel dat je een plekje in het geheugen wilt inruimen voor iets dat wel kan veranderen, bijvoorbeeld een object waarin de waarde van een thermometer steeds opnieuw opslaat, dan moet er natuurlijk geen **const** voor staan. We komen later terug op het woordje **int**.

Deel 2 - Void setup

In het tweede gedeelte zie je het commando **pinMode**. Dat is een commando waarmee je de Arduino vertelt wat voor soort apparaat is aangesloten op de pin (er zitten genummerde pins op de Arduino). Je ziet dat het object **LAMP** een **OUTPUT** is, dat is logisch: er komt licht uit. Je ziet ook dat **KNOOP** een **INPUT** is. Dat is ook logisch: een knop verwerkt een invloed van buitenaf (jij drukt wel of niet op de knop). Er zijn nog veel meer commando's die in void setup gebruikt kunnen worden, maar die volgen later. Je ziet ook dat er na het commando void setup een openingsaccoade staat **{** en aan het einde van void setup een sluitingsaccoade **}**. Dat betekent dat alles wat tussen die accolades staat bij **void setup** hoort. Commando's waarbij accolades gebruikt worden (zoals bij void setup, maar ook bij if en ifelse) hebben geen ; aan het einde van de zin, alle andere zinnen wel. Als je die ; niet plaats denkt de computer dat je zin nog niet af is en krijg je in de volgende regel een foutmelding. Eerste tip bij foutmeldingen is dus even te checken of je overal ; hebt staan waar dat hoort.

Deel 3 - Void loop

In regel 12 wordt een object gedefinieerd met de naam **KnopStatus**. Je kunt ook in dit gedeelte van de code objecten definieren, dat gebeurt dan steeds opnieuw omdat de void loop steeds opnieuw wordt uitgevoerd. Stel dat je in regel 3 (in het eerste gedeelte) **KnopStatus** al zou definiëren zou deze regel hetzelfde zijn, maar dan zonder **int** ervoor (dus: **KnopStatus = analogRead(KNOOP);**). In deze regel wordt met het commando **analogRead** de waarde uitgelezen van de KNOOP. Het systeem checkt dus of de knop ingedrukt wordt of niet. Als dat zo is, geeft **analogRead** een 1, anders een 0. Die 1 of 0 wordt met deze regel opgeslagen in het object **KnopStatus** en dat gebeurt dus steeds opnieuw. Omdat dit een hele korte void loop is, gaat het heel snel en wordt er dus eigenlijk continu gekeken of de knop wordt ingedrukt.

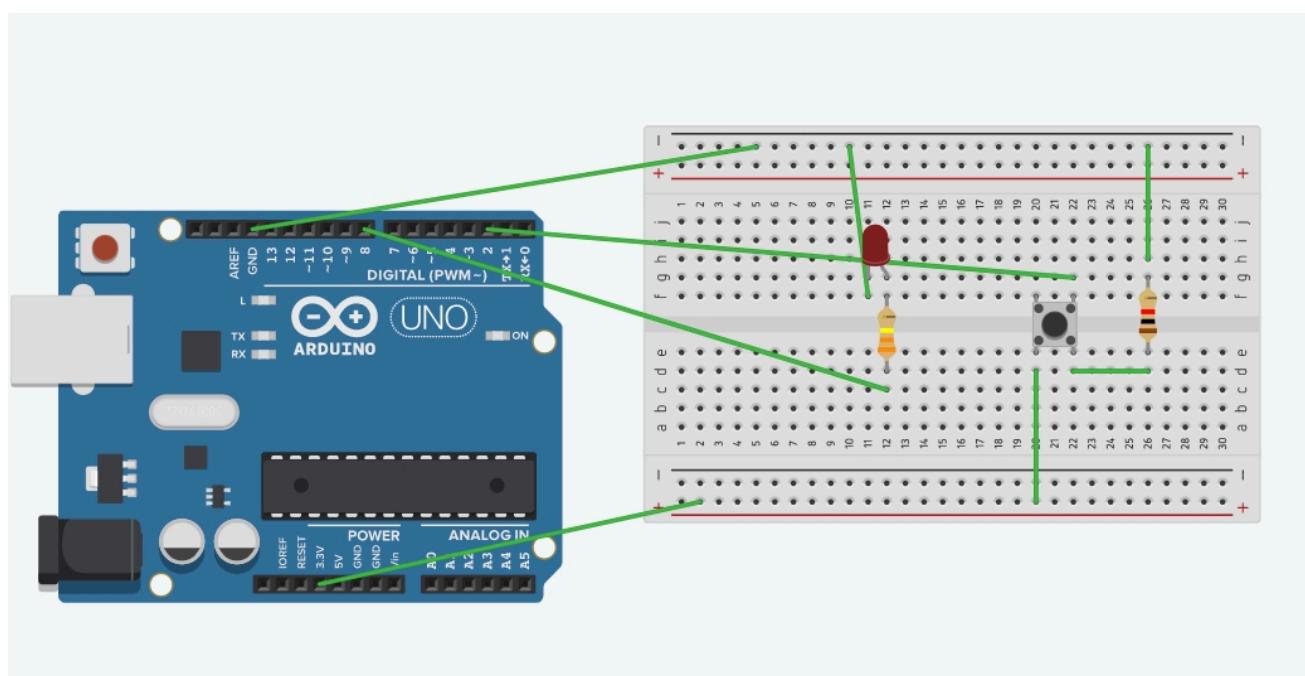
In regel 12 staat een **if** commando. Dat commando kijkt of de waarde tussen de haakjes erachter waar is of niet. In dit geval checkt hij of de waarde van **KnopStatus** gelijk is aan 1.

Als je goed kijkt zie je dat hier **==** staat. **==** betekent het dat de computer moet kijken of hetgene links en rechts van het **==**-teken gelijk zijn. **=** betekent dat je een waarde toekent aan een object. **=** gebruiken binnen de haakjes van **if** levert dus een foutmelding op.

Als datgene wat tussen de staat haakjes waar is, wordt dat wat tussen de accolades staat van **if** uitgevoerd. Net als bij **void setup** en **void loop** eindigt een **if** niet met ; maar staan er accolades. Dit kan bij complexere code ingewikkeld worden omdat je dan een **if** binnen een **if** binnen een **if** kunt krijgen en dat je erg goed moet bijnouden hoe het zit met de accolades. Een tip hierbij is om de accolades die bij elkaar horen op dezelfde afstand van de kantlijn te plaatsen. Je ziet bijvoorbeeld dat de accolade op regel 11 even dicht bij de kantlijn staat als de accolade in regel 19. Die van regel 19 sluit dus die van regel 11 af. Vaak wordt de openingsaccoorde direct achter de **if** of **else** geplaatst. De openingsaccoarde van regel 13 hoort bij de sluitingsaccoarde van regel 15 en de openingsaccoarde van regel 15 hoort bij de sluitingsaccoarde van regel 17.

Tussen de accolades van **if** staat het commando **digitalWrite(LAMP, HIGH);** Dat betekent dat er een signaal naar de lamp wordt gestuurd en wel een 1, dat zorgt er dus voor dat de lamp aangaat. De lamp blijft dan aanstaan tot dat er een 0 naar de lamp wordt gestuurd. Vergelijk het met een koelkast. Je stuurt nu een pak melk in de koelkast in. Dat pak blijft in de koelkast totdat je het er actief uit haalt. Doe je niets, blijft het pak melk in de koelkast (en wordt het op een gegeven moment slecht, maar dat hoor niet bij deze module).

In regel 15 staat het woordje **else**. Dat zorgt ervoor dat er ook iets gebeurt als de waarde tussen de haakjes bij **if** niet waar is. In dit geval stuurt hij dan een 0 naar de lamp en gaat de lamp uit.



Deze schakeling hoort bij de bovenstaande code

Verwerkingsopdracht

Hierboven wordt aan de hand van een voorbeeld een stuk code uitgelegd. Ook wordt er een schakeling getoond die bij deze code hoort. De opdracht is het volgende: zorg dat je de schakeling aanpast zodat er nog een led wordt toegevoegd en nog een knop die deze led aan op uitzet. Je moet dus én de schakeling aanpassen én de code. Doe dit in je werkdocument in Tinkercad.



Gegevens die verzameld zijn moeten vaak opgeslagen worden om later te kunnen gebruiken. Ook zullen ze soms getransporteerd moeten worden via een netwerk. Het internet is zo'n netwerk en steeds meer apparaten staan in verbinding met het internet (zogenaamde "Internet of things" (IOT)). Daar hoort ook beveiliging bij. Dit hoofdstuk gaat over hoe gegevens opgeslagen worden en hoe systemen met elkaar communiceren.

Tijdens dit hoofdstuk leer je het volgende

- Dat er vuistregels zijn voor het ordenen van gegevens binnen bestanden het ordenen van bestanden en dat het gebruiken van die vuistregels je tijd bespaart en de kans op fouten vermindert.
- Dat metadata de informatie bevat over wat er in bestanden te vinden is en hoe deze geordend zijn.
- Dat databases georganiseerd zijn op basis van de metadata en dat dat het zoeken in databases, en het samenvoegen of splitsen van databases mogelijk maakt
- Dat gegevens fysiek opgeslagen worden op harde schijven (in het klein) of in datacenters (in het groot) en hoe deze opslag functioneert
- Dat gegevens op verschillende manieren beveiligd kunnen worden door wachtwoorden en encryptie, en dat de wiskunde hier een grote rol in speelt.
- Dat het kraken van codes aan de wieg heeft gestaan van de ontwikkeling van computers
- Wat de rol van Alan Turing was in het ontstaan van computers
- Dat het internet een combinatie is van ip-adressen en TCP/IP protocollen en hoe dit werkt
- Hoe een wifi-verbinding werkt en hoe je zelf een wifi-ontvanger en zender kunt bouwen met een Arduino en daarmee gegevens kunt doorsturen naar andere digitale systemen
- Dat je informatie van een apparaat ook naar een website kunt sturen zodat je op afstand mee kunt kijken met de sensoren die jouw digitale systeem bevat.

Als er door jouw systeem 'besloten' wordt dat er iets in gang gezet moet worden (motor aan, koeling aan, ventilator uit etc.) moet dat natuurlijk wel gebeuren. Het is allemaal geen magie. In dit hoofdstuk gaan we in op hoe je je digitale systeem daadwerkelijk invloed kunt laten hebben op zijn omgeving.

Tijdens dit hoofdstuk leer je het volgende

- Dat digitale systemen invloed kunnen hebben op hun omgeving door middel van actuatoren
- Dat actuatoren zoals lampjes, speakers, motoren, RF zenders, servomotoren in veel verschillende digitale systemen voorkomen en dat je die (relatief) eenvoudig zelf kunt bouwen
- Dat je in je programmeercode in je digitale systeem soms bepaalde *bibliotheken* moet gebruiken om actuatoren te kunnen gebruiken
- Dat je voor het aansturen en monteren van actuatoren online veel informatie kunt vinden op websites als *stackoverflow* en *reddit*.
- Dat het aansturen van actuatoren gevolgen heeft voor het stroomverbruik van je digitale systeem en dat ontwerpkeuzes dus invloed hebben op de duurzaamheid van je systeem
- Dat je door te kijken hoe goed jouw actuatoren functioneren en hun omgeving beïnvloeden kunt beoordelen of jouw digitale systeem voldoet aan de ontwerpeisen die je hebt gesteld.

Een voorbeeld

Het bovenstaande kan allemaal erg onhaalbaar lijken. Daarom wordt hier een voorbeeld uitgewerkt. Ikzelf (de auteur: Rudy Jonker) ga tijdens de eerstvolgende ronde van deze module gelijktijdig met de leerlingen een ontwerp maken. Ik deel alle stappen hier op deze plek zodat je een beeld krijgt hoe zo'n ontwerp tot stand komt en welke problemen worden tegengekomen.

Ronde 1

Doelstelling ontwerp:

Ik wil een wake-up light maken. Dat wil zeggen een wekker waarbij als de **ingestelde** tijd is **aangebroken** er een lamp steeds **feller** en steeds **witter** gaat branden. Als het goed is wordt de persoon die slaapt dan ook **wakker** daardoor. Ik wil kunnen **meten** of dit ook zo is.

Pakket van eisen

- Er moet een tijd lopen op een klok
- Er moet een alarmtijd ingesteld kunnen worden
- Er moet een lamp aan kunnen gaan met een variabele felheid en kleur
- De persoon die wakker wordt moet ergens op kunnen drukken om aan te geven dat hij/zij wakker is.
- Dit moet door het systeem opgeslagen worden op een SD-kaart, zodat ik de prestaties van de lamp kan onderzoeken.

Deeluitwerkingen

Ik begin met de klok die moet lopen. Ik wil de uren en minuten laten zien, en een uur moet ook echt een uur duren, anders is de klok niet lang 'houdbaar'. Ik start met de code en check deze in de seriële monitor van arduino. Daardoor hoef ik me nog even niet druk te maken over de display.

Ontwerp:

```
long int mins = 0;  
long int hrs = 0;  
long int seconds = 0;  
  
void setup() {  
// put your setup code here, to run once:  
Serial.begin(9600);  
  
}  
void loop() {  
delay(1000); //hier verstrijkt 1000ms, dat is dus 1 seconde  
//long int t1 = millis(); //deze regel staat nu even uit  
seconds = seconds +1;  
if(seconds==60){  
seconds = 0; // als het aantal seconde 60 wordt, reset hem naar 0 en doe de minuten +1  
mins = mins+1;  
}  
if(mins==60){  
mins = 0; // als het aantal minuten 60 wordt, reset hem naar 0 en doe de uren +1  
hrs = hrs+1;  
}  
if(hrs==24){
```

```

hrs=0; //als het aantal uren 24 wordt, reset hem naar 0
}
//long int t2 = millis(); //deze regel staat nu even uit
//long int calctime= t2-t1; //deze regel kan ik aanzetten als ik wil weten wat de tijd is die verstrijkt tussen
// deze regel en de regel onderaan de code. Stel dat dat te lang is, dan moet ik de delay aanpassen
Serial.print (hrs);
Serial.print(" ");
Serial.print(mins);
Serial.print(" ");
Serial.print(seconds);
Serial.print(" ");
//Serial.println(calctime);
}

```



[De klok in werking](#)

Evaluatie

De klok doet het. Ik wilde eerst iets doen met het vastleggen van een tijd, en dan als die tijd een minuut geleden was, de minuten +1 doen, maar dat bleek veel te veel geklooi op te leveren. Daarom heb ik gekozen voor een delay. Mocht de code later nou zo ingewikkeld worden dat het uitvoeren van de code lang duurt (waardoor de tijd van 1 seconde op de klok dus 1000ms PLUS de verwerkingsstijd van de code zou duren) heb ik twee regels ingebouwd waarmee ik kan meten hoe lang de code duurt. Als het dan nodig is kan ik de delay verkleinen om voor de verwerkingsstijd van de code te corrigeren. Op naar de display en de knopjes!

Wall of Fame

Omdat deze module door veel scholen gebruikt wordt en daar allemaal leerlingen coole ontwerpen maken zijn er ongetwijfeld prachtige ontwerpen om hier te presenteren. Wil je jouw (beste) ontwerp op deze site? Stuur dan een stukje tekst met wat foto's of evt. een linkje naar een youtubefilmpje. Zorg dat het doel van je ontwerp duidelijk is en voor welke oplossingen je gekozen hebt.

Over dit lesmateriaal

Colofon

Auteurs	Digitaal Redacteur ; Rudy Jonker ; H. Dirks
Team	Schakelmodule Digitale Technologie
Laatst gewijzigd	24 februari 2023 om 13:06
Licentie	De Internationale Creative Commons 4.0 licentie waarbij de gebruiker het werk mag kopiëren, verspreiden en doorgeven en aangeleide werken mag maken onder de voorwaarden: Naamsvermelding en Gelijk Delen, zie http://creativecommons.org/licenses/by-sa/4.0/ . Meer informatie over de CC Naamsvermelding-GelijkDelen 4.0 Internationale licentie licentie.

Aanvullende informatie over dit lesmateriaal

Van dit lesmateriaal is de volgende aanvullende informatie beschikbaar:

Eindgebruiker	leerling/student
Studiebelasting	4 uur en 0 minuten

Bronnen

Slimme regenton

<https://slimmeregenton.nl/>

1 Introductie digitale technologie. Daan Geijs vertelt over zijn studiekeuze en opleiding
<https://youtu.be/H94a2UCkYXY>

2 Wat is kunstmatige intelligentie? Daan Geijs vertelt wat er met digitale technologie mogelijk is.
<https://youtu.be/p9vdykaXKg4>

3 Kunstmatige intelligentie in het ziekenhuis. Daan Geijs vertelt over zijn onderzoek aan het herkennen van tumoren met de computer.
https://youtu.be/VqiOqBVe_08

4 Daan Geijs over het werk dat hij in de toekomst kan gaan doen
<https://youtu.be/3t5nQqr28II>

Tinkercad TMP36

<https://www.tinkercad.com/things/94VFdq4ienj-copy-of-tmp36-temperature-sensor-with-arduino/editel?tenant=circuits>

Potentiometer

<https://youtu.be/H3hSQgZxNe0>

Ultrasone sensor

https://youtu.be/eYJg_Yp-Fys

Tutorial voor de ultrasone sensor

<https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6>

Arduino tutorial neopixel

<https://create.arduino.cc/projecthub/robocircuits/neopixel-tutorial-1ccfb9>

Neopixel voorbeeld

<https://youtu.be/ANoG6DoSFHA>

link naar liquidcrystal_i2c (datum link: 7-2-2022)

https://downloads.arduino.cc/libraries/github.com/marcoschwartz/LiquidCrystal_I2C-1.1.2.zip

Voorbeeldschakeling met Arduino, Relais en Lamp.

https://maken.wikiwijs.nl/152360/NLT__Digitale_Techniek_TCC#!page-7105936

De klok in werking

<https://youtu.be/nmrz2QJYan4>