JOTA-JOTI 2024

Handleiding Arduino

Voor wie en wat

Deze handleiding is geschikt voor scouts en ouder (zelfs voor (bege)leiding, en misschien als je een leergierige welp hebt ook wel). We leggen hier uit wat een Arduino is, hoe je er mee om kan gaan, en hebben een paar oefeningen om je bekend te maken met de wondere wereld van microcontrollers. We leren een beetje programmeren, wat over elektriciteit, en hoe je data uit kan lezen.

Inhoud

Wat is een Arduino?

Een Arduino is eigenlijk een hele kleine losse computer (ook wel bekend als een microcontroller). Waar een laptop eigenlijk vaak 30cm bij 30cm is, en een desktop zelfs vele malen groter, is een Arduino vrij klein: Vaak maar een paar centimeter groot. Maar eigenlijk is dat niet het belangrijkste verschil. Bij een normale computer kan je vaak interactie hebben met bijvoorbeeld een toetsenbord en een muis, en kan je ook nog zelf nog applicaties installeren. Bij de Arduino werkt het net iets anders. Je maakt een programma in een programmeertaal die hij begrijpt en die zet je vervolgens in het geheugen van de Arduino. Wanneer deze aan gaat zal deze gelijk dat programma uitvoeren. Heb je het programma niet zo gemaakt dat het ergens op kan reageren? Dan doet je Arduino dat ook niet.

Het bovenstaande klinkt niet als een enorm voordeel. Totdat je weer andere verschillen bekijkt: Zo is een Arduino vaak maar een paar euro, gebruikt hij vrij weinig stroom, en misschien wel het belangrijkste: Er zitten aansluitingen (GPIO-poorten genaamd) op waarmee je elektrische signalen kan uitsturen en lezen! Kijk, nu wordt het interessant. Zo kan je bijvoorbeeld LED-lampjes aansturen, naar het drukken op een knop luisteren en daar aan de hand van een actie uitvoeren, en eigenlijk kan je het zo gek niet maken!

Om het allemaal nog makkelijker te maken zijn er ook nog eens verschillende soorten Arduino's. Over de loop van de jaren zijn er honderden verschillende uitgegeven door het bedrijf hierachter. Eigenlijk biedt elke versie wel haar voor- en nadelen: Sommige kunnen iets met bluetooth, anderen hebben weer WiFi, sommige hebben zelfs een LED-schermpje standaard aan boord, en nog anderen hebben minder krachtige hardware zodat ze een stukje goedkoper zijn!

Ben jij ook zo enthousiast geworden en wil je meer weten? Kijk dan eens op de officiële website van Arduino: https://www.arduino.cc/ .

Open Source

Alles wat door het bedrijf achter de Arduino gemaakt wordt, is *Open Source*. Dit betekent dat alle schema's van de Arduino's en alle software waarmee je interactie hebt met de Arduino is ook voor iedereen (gratis!) toegankelijk. Het voordeel van dat alles open is, is dat er ook genoeg bedrijven zijn die het goedkoop in grote getalen kunnen produceren, waardoor prijzen laag blijven. Daarnaast stelt het ook mensen met veel technische kennis in staat om een eigen versie te ontwikkelen met meer mogelijkheden aan boord dan de "standaard" versies! Let wel op dat deze bijna exact hetzelfde werken als de "officiële" versies, maar niet gemaakt zijn door het officiële Arduino bedrijf. Als je dus vragen hebt daar over, moet je aankloppen bij de producent.

Hoe kan je beginnen?

Eigenlijk kan je heel makkelijk zelf beginnen. Je hebt eigenlijk maar 3 basisbenodigdheden echt nodig als je iets wil doen:

- Een Arduino
- Een USB-kabel om je Arduino op je computer aan te sluiten
- De Arduino IDE; hierin kan je je Arduino programmeren en je programma op je Arduino zetten). Deze kan je downloaden op https://www.arduino.cc/en/software en is beschikbaar voor Windows, MacOS en Linux.

In de IDE zitten ook al enkele voorbeeldprogramma's ingebakken die je kunnen helpen met het leren omgaan met een Arduino!

Voor de oefeningen die hier beneden staan, gaan we er van uit dat je een computer tot je beschikking hebt waar de IDE al op hebt geïnstalleerd. Download deze dus daarom en installeer hem alvast!

Beginnen met het programmeren van een Arduino

Wanneer je een nieuw programma (ook wel een Sketch genaamd) maakt voor een Arduino zie je de volgende stukken code staan:

```
void setup() {
}

void loop() {
}
```

Wat je hierboven ziet zijn 2 stukken code. Deze stukken code: De Setup en de Loop. Zo'n stuk code wordt ook wel een "functie" genoemd. Wanneer een functie aangeroepen wordt, wordt de code die bij deze functie hoort uitgevoerd. Het kan zijn dat een functie alleen wat aanstuurt, maar het kan ook zijn dat de functie een waarde teruggeeft. Het kan bijvoorbeeld zo zijn dat je een functie hebt die een berekening uitvoert waarvan je het resultaat later nog een keer wil hergebruiken, zonder dat de berekening nog een keer uitgevoerd hoeft te worden. Een functie bestaat uit enkele belangrijke onderdelen:

- De "Return Type". Dit is het type waarde dat wordt teruggegeven. Hierboven is het aangegeven als `void`. Dat betekent dat er eigenlijk niets terug wordt gegeven
- De naam. Hierboven is dat bijvoorbeeld `setup` of `loop`. Wanneer je een functie aanroept gebruik je die naam.
- De parameters. Soms moet je een bepaalde waarde doorgeven aan een functie die daar wat mee doet. Deze staan dan tussen de ronde haakjes `()`. Hierboven wordt het niet gebruikt, maar als je dat wil, kan je bijvoorbeeld het getal (een `int`) genaamd "voorbeeld" op deze manier tussen de haakjes zetten: `(int voorbeeld)`.
- De functie "body". Tussen de 2 accolades (dus tussen de { en de }) zet je de code die de functie moet uitvoeren. Wat hier kan komen kan je zien in de oefeningen.

De standaard setup functie wordt eenmalig aangeroepen als de Arduino aan gaat. Vervolgens wordt de loop functie constant aangeroepen. Wanneer die functie dus klaar is met uitvoeren, wordt hij tot de Arduino uitgaat die functie opnieuw en opnieuw aangeroepen.

Benodigdheden bij de oefeningen

Voor de onderstaande oefeningen hebben we eigenlijk de volgende onderdelen nodig. Als je meerdere scouts de oefeningen tegelijkertijd wil laten uitvoeren, heb je dus ook een veelvoud nodig van deze onderdelen:

- 1 Arduino Uno R3 (ook wel bekend als een Arduino Uno Rev3)
- 1 USB-A naar USB-B kabel
- 1 rood LED-lampje
- 1 groen LED-lampje
- 1 geel LED-lampje
- 1 RGB LED-lampje (Dit is 1 LED-lampje dat Rood, Groen en Blauw kan worden, of zelfs een combinatie van deze kleuren)
- 1 kleine drukknop
- 1 DHT-11 temperatuursensor
- 1 Weerstand van 220Ω (220 Ohm)
- 1 Weerstand van 10kΩ (10 Kilo-ohm/10 000 Ω/10 000 Ohm)
- 1 Breadboard
- 10 Male-Male Jumper kabels
- 16x2 lcd-scherm

Als je een beetje creatief zoekt bij de grote partijen (zoals die vernoemd is naar een regenwoud of bij een bekende grote Chinese winkel) kan die al voor enkele euro's (rond een tientje) bij elkaar gesprokkeld worden. Vaak hebben deze ook zelfs een starters set tussen de 20 en 40 euro, maar daar zit natuurlijk ook veel meer in dan alleen het bovenstaande!

Ook moet je nog 1 dingetje instellen in de Arduino IDE, namelijk naar welke type Arduino je upload. Open daarvoor de IDE, ga in het Menu naar Tools > Board > Arduino AVR Boards > Arduino Uno. Hierdoor weet je IDE welke "taal" hij moet spreken tegen je Arduino.

Oefening 1: Een lampje laten knipperen (data uitsturen)

Ok, genoeg theorie, we gaat nu aan de slag met een eenvoudige oefening. We gaan een LED-lampje laten knipperen. Zoals bij alles met een Arduino bestaat het uit 2 delen: Een gedeelte programmeren en een gedeelte fysiek. Laten we beginnen met het programmeren:

Allereerst moeten we tegen de Arduino vertellen dat we 1 van de GPIO-poorten willen gebruiken als uitgang. We kiezen hiervoor voor gaatje nummer 13, dus moet de Arduino begrijpen dat we daar elektriciteit doorheen willen sturen. Dat doen we door het volgende toe te voegen aan de setup functie:

```
pinMode(13, OUTPUT);
```

Hier roepen we de functie genaamd pinMode aan. Dit is een functie die je eigenlijk niet ziet in je Sketch, maar die zit al standaard in de Arduino ingebakken! We geven hier 2 parameters aan mee, namelijk 13 en OUTPUT. Wat deze functie doet is GPIO-poort nummer 13 markeren als een uitgang.

Vervolgens willen we het LED-lampje aan zetten. Dit kunnen we doen in de functie loop. Laten we beginnen met het lampje aan te zetten door de volgende code:

```
digitalWrite(13, HIGH);
```

De functie digitalWrite geeft een digitaal signaal af over een bepaalde poort. Zoals je misschien wel weet is een digitaal signaal een 1 of een 0, oftewel Hoog en Laag, of stroom aan of uit. Als je toevallig wat Engels kan, kan je misschien wel inschatten wat deze functie met deze parameters doet: Hij stuurt een “hoog” signaal uit over poort 13, oftewel er wordt stroom uitgestuurd over poort 13. Het aansluiten op een LED-lampje komt later in deze oefening.

Vervolgens willen we dat de Arduino even 1 seconde niets doet. Als het lampje namelijk te snel knippert, zie je dit helemaal niet eens! Dit doen we met de volgende code:

```
delay(1000);
```

Aan de delay functie geef je en getal mee. Dit is het aantal microseconden dat je wil dat de Arduino niets doet. In 1 seconde zitten 1000 microseconden, dus als je wil dat hij 1 seconde niets doet, geef je de waarde 1000. Als je wil dat hij 1,5 seconde niets doet moet de waarde 1500 zijn, etc.

Als laatste willen we dat het LED-lampje ook weer uitgaat uiteraard. Dit kunnen we doen door de digitalWrite functie weer aan te roepen, maar dan met een LOW i.p.v een HIGH. Ook daarna willen we weer 1 seconde wachten.

Als je alles goed hebt gedaan ziet je code er als volgt uit:

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

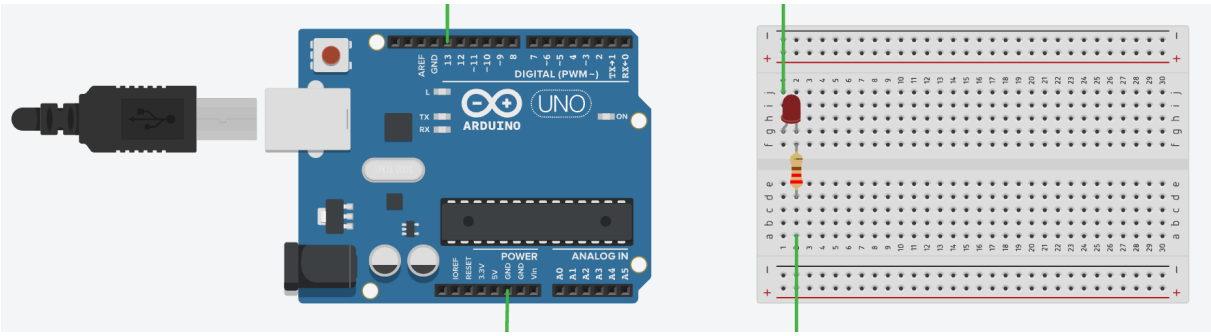
Nu we het gedeelte met de code achter de rug hebben, is de volgende stap het fysiek in elkaar zetten van je project. Je hebt hiervoor de volgende onderdelen nodig:

- De Arduino met je programma er op
- 2 Jumper kabels (Male-Male)
- Een rood LED-lampje
- Een weerstand van 220Ω (220 Ohm)
- Breadboard

Om te beginnen prik je 1 van de kabels in het gaatje in de Arduino met het nummer 13 ernaast. Zoals je waarschijnlijk al herinnert van hierboven, is dat de GPIO-poort die we gebruiken om elektriciteit uit te sturen. Prik vervolgens de andere kant van de kabel in 1 van de gaten van de breadboard. Bijvoorbeeld op plek J1. Plaats daarna een LED-lampje op het breadboard. **LET OP:** het is belangrijk hoe je deze plaatst. Een LED-lampje laat maar op 1 manier elektriciteit door. We plaatsen de lange poot (de anode) op plek G1, en de korte kant (de kathode) op plek G2.

Vervolgens plaatsen we een weerstand van 220Ω tussen plek F2 en D2. We zullen niet te diep ingaan op de theorie waarom dit nodig is, maar kort gezegd stuurt een Arduino net wat te veel stroom uit en met een weerstand ertussen brengen we dat een klein beetje naar beneden.

Als laatste moeten we het circuit nog compleet maken: Dit betekent dat er nog een kabel van de weerstand terug moet gaan naar 1 van de poorten met GND ernaast. Plaats daarom een kabel tussen die poort en plek A2 op het breadboard. Als het goed is ziet het eruit als de onderstaande afbeelding. Wanneer je je Arduino nu aan zet (door de USB-kabel aan te sluiten) zal je zien dat na een korte opstarttijd je LED-lampje begint te knipperen



Gefeliciteerd! Je hebt de eerste opdracht voltooid! Hopelijk vond je het leerzaam. We zijn natuurlijk eenvoudig begonnen, maar gaan zeker nog leukere projectjes maken!

Oefening 2: Werken met een drukknop (leren hoe een drukknop werkt)

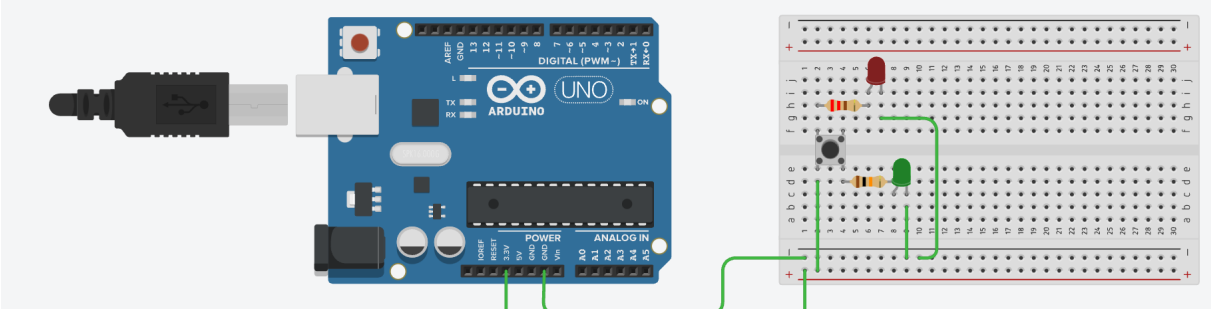
Top, we hebben oefening 1 overleefd! Wat een geweldig gevoel om dingen zelf te maken, niet? De volgende stap wat we gaan doen heeft eigenlijk geen programmeerwerk, maar is bedoeld om je bekend te maken met hoe een drukknop werkt. Ook gaan we een stapje verder met het gebruik van een breadboard.

Op de Arduino Uno zitten ook 3 poorten die altijd stroom afgeven. We gaan gebruik maken van degene waar 3.3V naast staat. Deze geeft constant 3,3 volt af. Sluit deze met behulp van een Jumper kabel aan op 1 van de rijen met een rode + ernaast. Ook willen we dat uiteindelijk de stroomkring ook volledig wordt. Sluit daarom 1 van de poorten met GND ernaast aan op de rij naast die we hebben gebruikt, namelijk die meteen zwarte – ernaast.

Voor we doorgaan eerst een kleine uitleg over hoe een drukknop werkt: Een drukknop bestaat uit 4 pinnetjes en natuurlijk de knop zelf. Wanneer je op 1 van de 4 pinnetjes een stukje stroom zet, loopt deze automatisch door naar de pin die er recht tegenover zit. Wanneer je de knop indrukt, gaat er *ook* stroom lopen naar de overige 2 pinnetjes.

Goed, met deze kennis op zak is het dus van belang dat we de knop eerst op het breadboard zetten. Druk de pinnetjes van de knop in de volgende posities: F2, F4, E2 en E4. Vervolgens moeten we de stroom vanuit de Arduino ook naar het knopje krijgen. Sluit daarom een willekeurig gat van de + rij aan op plek D2. Vervolgens gaan we 2 LED-lampjes aansluiten: Zet een rode LED op plek I6 (de lange kant van de LED) en I7 (de korte kant van de LED). Sluit vervolgens een groene LED aan op positie C8 (de lange kant) en C9 (de korte kant). Vervolgens prik je 2 draadjes van G7 en B9 naar de – rij die je aan het begin hebt aangesloten.

Zoals je misschien nu al inschat, missen we nog iets om het werkend te maken, namelijk de verbinding tussen de knop en de LED-lampjes! Omdat we niet per ongeluk een LED-lampje kapot willen maken sluiten we daarom 2 weerstanden aan: Een weerstand van 220Ω moet tussen H2 en H6 aangesloten worden, en een weerstand van 10kΩ moet tussen D4 en D8 aangesloten worden. Het zou er dan ongeveer zo uit horen te zien:



Zet je Arduino vervolgens aan. Wat gebeurt er standaard, en wat als je op het knopje drukt?

Oefening 3: Een stoplicht maken (Variabelen, waardes vergelijken, en invoer lezen)

Binnen deze oefening gaan we nog een stapje verder. We gaan namelijk een stoplicht maken. Dit stoplicht staat standaard op rood. Wanneer je op een knopje drukt, gaat het rode lampje uit en het groene aan voor 5 seconden. Daarna wordt de groene uitgezet, gaat een gele aan voor 2 seconden, en als laatste gaat die ook weer uit en gaat rood weer aan.

Om dit voor elkaar te krijgen hebben we 3 nieuwe concepten nodig: Een waarde dat we moeten instellen om te checken of er op een knop is gedrukt, en we moeten de knop uitlezen wanneer deze wordt ingedrukt.

Laten we beginnen bij het begin: We hebben in totaal 3 poorten nodig die we als uitgang gebruiken. In dit voorbeeld gebruiken we poort 9, 10 en 11 hiervoor. In oefening 1 heb je geleerd hoe je een poort instelt als uitgang. Stel poort 9 ook standaard in om aan te gaan als de Arduino aan gaat. Je wil tenslotte ook niet dat een stoplicht pas een lichtje geeft als je er pas op drukt. Daarnaast hebben we ook poort 8 nodig als ingang om te lezen of de drukknop is ingedrukt. Dit kan je doen door het volgende in de setup toe te voegen:

```
pinMode(8, INPUT);
```

Verder moeten we ook iets gebruiken om te zorgen dat we niet per ongeluk de cyclus van het stoplicht opnieuw starten terwijl hij al bezig is. Om dit te doen gaan we gebruik maken van uit wat een *variabele* heet. Een variabele is een waarde die we kunnen aanpassen naar wat we maar willen. Om hier gebruik van te maken, en om te zorgen dat deze niet elke keer wordt ingesteld op een waarde als de *loop* functie wordt aangeroepen, moeten we zorgen dat deze ingesteld wordt buiten een al bestaande functie. Voeg daarom helemaal bovenaan je sketch het volgende toe. Hierdoor maak je een variable genaamd `lock` met als waarde 0:

```
int lock = 0;
```

Vervolgens gaan we de daadwerkelijke logica maken zodat het stoplicht werkt. Dit gaan we doen in de *loop* functie. Hiervoor hebben we het volgende stuk code nodig:

```
if (lock != 1 && digitalRead(8) > 0) {
  lock = 1;

  ...

  lock = 0;
}
```

Dit is iets anders dan wat je eerder al hebt gezien. Laten we beginnen met het uitleggen wat dat gedeelte op de regel van de if doet. Computers begrijpen eigenlijk alleen 1'en en 0'en. Als we dus iets willen uitvoeren onder bepaalde omstandigheden moeten we dat dus ook bekijken. Eigenlijk wat we hier doen is kijken of de waarde van lock niet gelijk is aan 1 en kijken we of we op pin nummer 8 een signaal binnen krijgen (dat signaal is een 1 of een 0, dus als het groter is dan een 0, dan krijgen we een signaal binnen).

Vervolgens zetten we de waarde van lock op 1. Wanneer de loop dan weer wordt aangeroepen als hij bezig is, zal de check bij de if niet kloppen, en zal deze code ook niet uitgevoerd worden. De ... vullen we zo meteen in, maar eerst als laatste zetten we de waarde van lock weer terug op 0 zodat we hierna nog een keer kunnen drukken.

Om je programma af te maken gaan we nu de ...invullen. Zoals aan het begin te lezen is, moet je eerst alleen het groene lampje aanzetten, dan 5 seconden wachten, dan alleen het gele lampje aanzetten, 2 seconden wachten, en als laatste alleen het rode lampje aanzetten. In principe hebben we alles wat je hiervoor nodig hebt al in oefening 1 behandeld, namelijk een LOW signaal sturen naar de LED die uit moet, een HIGH signaal sturen naar de LED die aan moet, en dan even wachten.

Als je alles goed hebt gedaan komt je code er dan ongeveer zo uit te zien:

```
int lock = 0;

void setup()
{
  pinMode(8, INPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);

  digitalWrite(9, HIGH);
}

void loop()
{
  if (lock != 1 && digitalRead(8) > 0) {
    lock = 1;

    digitalWrite(9, LOW);
    digitalWrite(11, HIGH);
    delay(5000);

    digitalWrite(11, LOW);
    digitalWrite(10, HIGH);
    delay(2000);

    digitalWrite(10, LOW);
    digitalWrite(9, HIGH);
    lock = 0;
  }
}
```

Nu we de code hiervoor af hebben wordt het tijd om alle draadjes weer aan te sluiten! Laten we beginnen met het drukknopje in te stellen:

De drukknop aansluiten

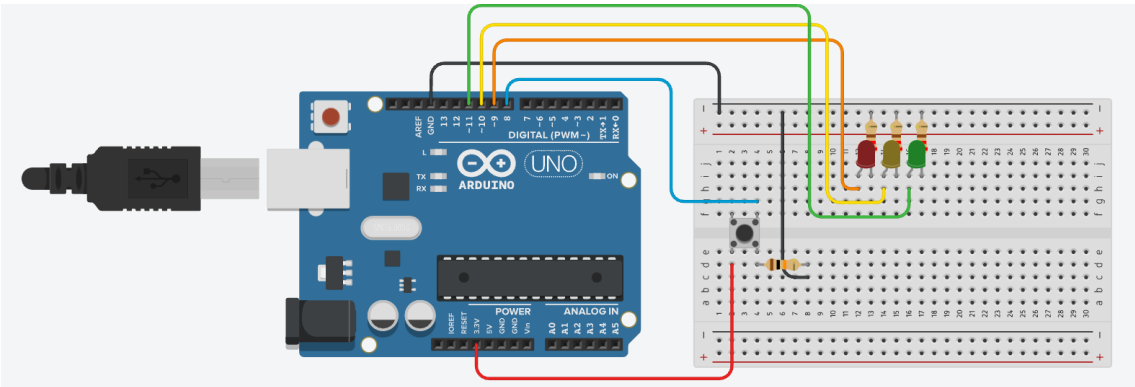
Sluit deze weer aan op punten F2, F4, E2 en E4. We laten vervolgens een draad vanuit de 3.3V poort lopen naar de rij waar de links onder poot van de drukknop in bevind. De rechts boven poot sluiten we vervolgens op dezelfde manier aan, maar dan in poort 8. Rechtsonder verbinden we met een 10 kΩ weerstand tussen positie D5 en D7. De weerstand verbinden we vervolgens met een kabeltje naar de bovenste – rij. Deze rij vervolgens weer met 1 van de GND-poorten in de Arduino.

De LED-lampjes aansluiten

De volgende stap is het aansluiten van de LED-lampjes:

- Rood: I12 (lang) & I13 (kort)
- Geel: I14 (lang) & I15 (kort)
- Groen: I16 (lang) & I17 (kort)

Verbind vervolgens plekken J13, J15 en J17 met een 220Ω weerstand aan de – rij bovenaan. Als laatste stap moet de Arduino hier ook bij kunnen: Sluit daarom plek H12 aan op poort 9, H14 op poort 10 en H16 op poort 11. Als het goed is zien je ontwerp er nu uit als volgt:



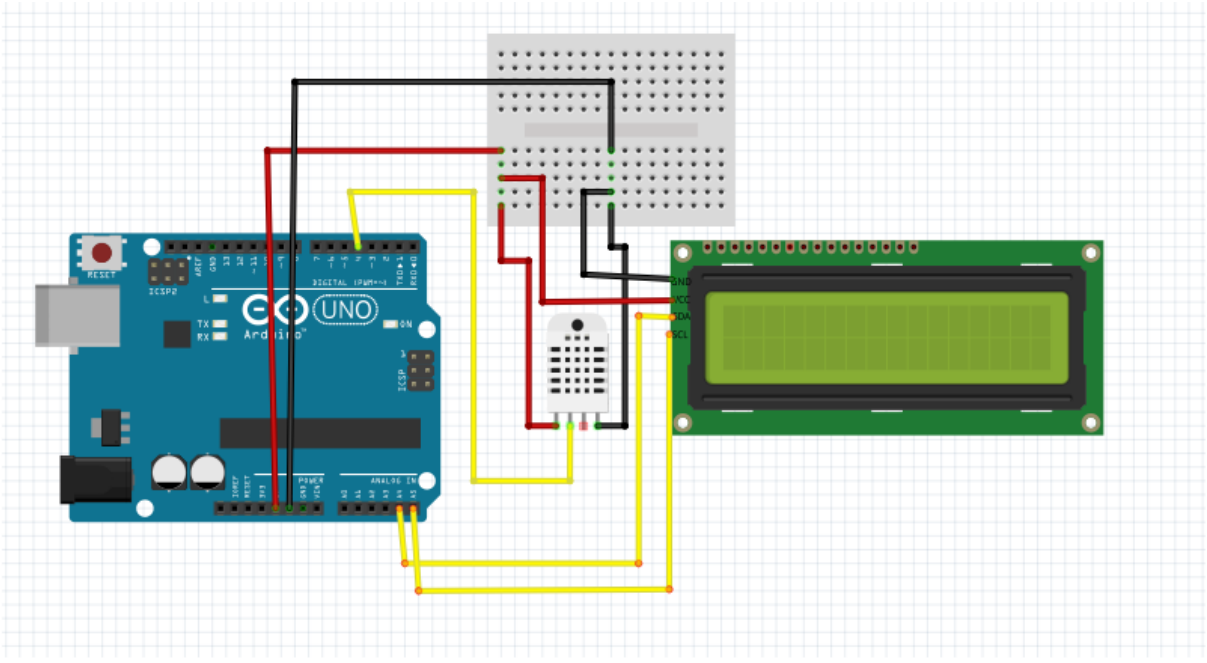
Oefening 4: Het uitlezen van de temperatuur

Top, we hebben nu een geleerd hoe we een eigen stoplicht maken, maar het kan allemaal nog geavanceerder. We gaan daarom nu nog dieper de materie in en maken met behulp van een Arduino een schermpje dat de huidige temperatuur en luchtvochtigheid weergeeft. We gaan het hier hebben over hoe je daadwerkelijk data heen en weer kan sturen, en wat Libraries zijn en hoe je kan gebruiken.

Omdat we de basis van componenten aansluiten al hebben gehad, zullen we daar wat sneller doorheen gaan dit keer:

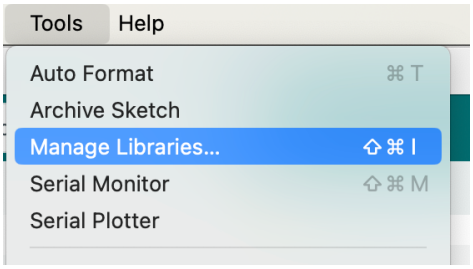
- Temperatuursensor
    - 1e pin aansluiten op 5V
    - 2e pin aansluiten op poort 4
    - 3e los
    - 4e pin op GRD aansluiten
- Lcd-scherm
    - GND-pin op GND
    - VCC-pin op 5V
    - SDA-pin op A4
    - SCL-pin op A5

Je ontwerp komt er dan ongeveer als volgt uit te zien. Het is niet heel spannend om dit aan te sluiten, want het moeilijkere werk komt bij het programmeren!



Bron: https://projecthub.arduino.cc/Druhi_C/temperature-and-humidity-sensor-with-lcd-1602-i2c-display-a2861e

Om te beginnen moeten we een manier hebben om met de temperatuursensor te praten. Nu kunnen we helemaal van begin af aan beginnen met daar tegenaan te programmeren, maar dat is eigenlijk behoorlijk complex. Gelukkig zijn er genoeg andere mensen (of producenten van de sensoren) die dit werk al voor ons hebben gedaan. Die hebben en stel functies gemaakt en beschikbaar gesteld aan iedereen die wil. Al dit hebben zij gebundeld in een *Library* (het Engelse woord voor Bibliotheek). Deze kan je simpel toevoegen aan je sketch en gebruik van maken.



Om van een Library gebruik te maken moet je beginnen met deze te installeren. Er zijn gelukkig al vele voorgeselecteerde libraries beschikbaar via de Arduino IDE. Ga hiervoor in het menu naar Tools en dan naar Manage Libraries.

Er opent hier dan een scherm waarin je de libraries kan zoeken. Zoek hier naar de libraries die “DHT11” en “LiquidCrystal I2C” heten en klik op installeren.

Om vervolgens gebruik te maken van deze libraries in je sketch moet je helemaal bovenin de volgende 2 regels toe te voegen:

```
#include "DHT11.h"
#include <LiquidCrystal_I2C.h>
```

Hiermee geef je aan je programma aan “Hey, die stukken code die ik niet op mijn computer heb gedownload, gebruik deze in mijn sketch”.

De volgende stap is eigenlijk direct daar onder de temperatuursensor en het LCD scherm te initialiseren. Dit doe je door het volgende stuk code:

```
DHT11 dht(4);
LiquidCrystal_I2C lcd(0x27,16,2);
```

Hiermee creëer je variabele genaamd dht en lcd die je kan gebruiken om bepaalde functionaliteit aan te roepen op je lcd-scherm en je temperatuursensor. Zodra je Arduino aan gaat, wil je ook dat je sensor en lcd-scherm beginnen te werken. Dit kan je doen door de volgende regels toe te voegen aan de *setup* functie. Die geven aan dat je temperatuursensor moet beginnen met meten, zeten het achterlicht aan van je LCD-scherm en zegt dat hij informatie kan ontvangen.

```
dht.begin();
lcd.backlight();
lcd.init();
```

Vervolgens moet je de volgende regels toevoegen binnen de *loop* functie:

```
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Luchtv=");
lcd.print((float) dht.readHumidity());
lcd.print("%");
lcd.setCursor(0,1);
lcd.print("Temp=");
lcd.print((float) dht.readTemperature());
lcd.print("C");
delay(2000);
```

Laten we dat stukje even opbreken in de verschillende delen:

Hiermee zeg je dat het LCD scherm leeg gemaakt moet worden:

```
lcd.clear();
```

Vervolgens zeggen we dat we beginnen met tekst weergeven op de 1e regel:

```
lcd.setCursor(0,0);
```

Hierna geven we aan dat we de het volgende stukje tekst daar willen weergeven: Luchtv=10.0% (waar die 10.0 natuurlijk vervangen wordt door de echte waarde

```
lcd.print("Luchtv=");
lcd.print((float) dht.readHumidity());
lcd.print("%");
```

Nadat we de luchtvochtigheid hebben weergegeven zeggen we dat we het volgende stukje tekst willen schrijven op de 2e regel:

```
lcd.setCursor(0,1);
```

Op de 2e regel schrijven we vervolgens: Temp=19.5C

```
lcd.print("Temp=");
lcd.print((float) dht.readTemperature());
lcd.print("C");
```

En wachten we nog 2 seconden voordat we de volgende meting doen:

```
delay(2000);
```

Er zijn hier boven een paar punten die misschien je al opgevallen zijn die we wat verder moeten uitlichten:

- Als we op regel 1 willen schrijven, zeggen we dat het regel 0 is, en als je op de 2e regel wil schrijven is dat regel 1. Dat klinkt misschien raar, maar je moet je realiseren dat bij het programmeren we niet beginnen te tellen bij 1, maar beginnen met tellen bij 0.
- Je ziet ook het volgende: (float). Het ziet er misschien een beetje raar uit, maar zoals eerder aangegeven uit zijn er verschillende types data. Je hebt eerder al int gezien. Een int (kort voor integer) is een geheel getal, dus niets achter de komma. Een float is een getal met getallen achter de komma. Door (float) voor de aangeroepen functie te zetten zorgen we ervoor dat wat die functie ook teruggeeft.
- Om een stuk tekst weer te geven, wordt er een functie gebruikt genaamd print. Dat is eigenlijk raar, want het is toch geen printer? Toch is deze bewoording vanuit het begin van de computers overgebleven en zit vastgeroest binnen zo'n beetje alle programmeertalen
- Als laatste is het je misschien opgevallen dat we niet alleen een functie aanroepen, maar dat het er als volgt uit ziet: dht.readTemperature(). dht is een variabele die we eerder hebben aangemaakt. Deze variabele is van het type DHT11. Wanneer deze variabele wordt aangemaakt worden daar een paar functies in gestopt. Een daarvan is de readTemperature functie. Wanneer je dus bovenstaand stukje code aanroept, roep je dus eigenlijk die functie aan op die variabele.

Je uiteindelijke code zou er dan als volgt uit moeten komen te zien uiteindelijk:

```
#include "DHT11.h"
#include "LiquidCrystal_I2C.h"

DHT11 dht(4);
LiquidCrystal_I2C lcd(0x27,16,2);

void setup()
{
  dht.begin();
  lcd.backlight();
  lcd.init();
}

void loop()
{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Humidity=");
    lcd.print((float)dht.readHumidity());
    lcd.print("%");

    lcd.setCursor(0,1);
    lcd.print("Temp=");
    lcd.print((float)dht.readTemperature());
    lcd.print("C");

    delay(2000);
}
```

Begrippenlijst

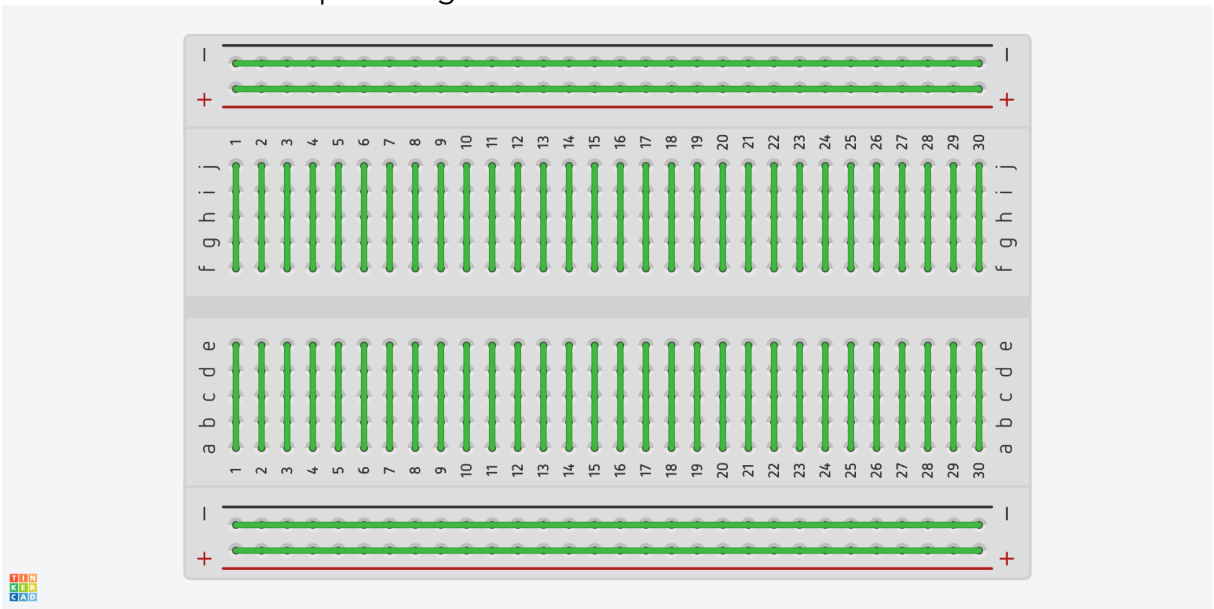
- **Arduino** De microcomputer die we gebruiken voor deze oefeningen.
- **Library** Een verzameling van code dat door iemand beschikbaar gemaakt is.
- **IDE** Het programma waarin je de code moet typen.
- **Sketch** De code welke je up de Arduino zet.

Addendum

Addendum 1: Hoe werkt een breadboard?

Een breadboard, letterlijk vertaald een broodplank, is een plankje met allerlei gaten. In deze gaten kan je kabels of elektronica prikken. Daarom wordt het in het Nederlands ook wel eens een prikbord genoemd. Deze gaten zijn onderling met elkaar verbonden, zodat je als je stroom op een bepaald gaatje zet je dat ook door verschillende andere gaten stuurt. Hierdoor kan je dus componenten aan elkaar koppelen.

Onder water is alles op de volgende manier met elkaar verbonden:



De groene lijnen staan met elkaar in verbinding. Om het even concreet te maken: Als je bijvoorbeeld een kabel op plek A1 prikt, en daar stroom doorheen stuurt, dan staat er ook stroom op de plekken B1, C1, D1 en E1. Wil je bijvoorbeeld ook op F1, G1, H1, I1 en J1 stroom hebben op datzelfde moment? Dan moet je 1 van de open plekken van de onderste rij verbinden met 1 van de plekken van de bovenste rij door bijvoorbeeld een kabel of een component ertussen te zetten.

Addendum 2: Welke componenten zijn er?
Hieronder hebben we een lijstje gemaakt met de verschillende componenten die je waarschijnlijk tegenkomt bij projecten met een Arduino. Ook staat er een kleine beschrijving bij waar en hoe deze voor gebruikt kan worden. Deze lijst is langer dan wat er voor de voorbeelden gebruikt wordt. Ze probeert je voornamelijk een idee te geven wat er allemaal is

- **Weerstand** Een weerstand is bedoeld om de voltage dat door je stroomkring loopt te verlagen. Een weerstand heeft een bepaald getal, dat ook wel Ohm wordt genoemd (of aangegeven met het symbool Ω). Hoe meer Ohm een weerstand heeft, hoe minder stroom hij doorlaat. Hoeveel stroom dit is is afhankelijk van hoeveel stroom je erin stuurt.
    - **Lichtgevoelige weerstand (LDR)** Naast de gewone weerstand zijn er ook variabele weerstanden. Deze hebben niet een vaste waarde (in Ohm/Ω), maar die waarde veranderd aan de hand een bepaalde factor. In het geval van een LDR is dit door middel van licht. Hoe meer licht erop schijnt, hoe meer weerstand die heeft. Hierdoor kan je bijvoorbeeld meten hoe licht of donker het in je omgeving is. Handig om een lamp automatisch aan of uit te zetten!

- **Diodes** Een diode is een speciaal soort component. Hij laat stroom maar op 1 manier door! Als je de stroom van de "verkeerde" kant erin probeert te sturen, dan komt er aan de andere kant niets uit. Wanneer je de stroom door de juiste kant stuurt, komt er aan de andere kant wel wat uit. Het is dus belangrijk dat je deze altijd juist aansluit, anders werkt je project niet.
    - **LED-lampje** De meest bekende versie van een diode is de Light Emitting Diode, ook wel bekend als een LED. Dit is eigenlijk niets anders dan een bepaald soort diode dat licht uitstraalt. Soms kan dit zelfs infrarood licht zijn dat je zelf niet kan zien.
    - **Zener diode** Een andere variant van een diode wordt ook wel een Zener diode genoemd. Deze houdt de stroom vanaf een bepaalde kant tegen. Tot het moment dat er een bepaald voltage is bereikt. Hierbij wordt eigenlijk alles wat boven dat voltage komt door de diode heen gestuurd (ja, dus de "verkeerde" kant op!). Stel je hebt 10 volt, en dat splits je op een kabel en over een Zener diode van 3,7 volt, dan gaat er door de andere kabel maar 3,7 volt en door de Zener diode 10 – 3,7 = 6,3 volt!

- **Drukknop** De kans dat je weet wat dit doet is eigenlijk al vrij groot. Je hebt het namelijk vast wel al een keer zelf gebruikt, bijvoorbeeld om een apparaat aan te zetten. Wanneer je een drukknop indrukt, gaat er een stroompje door lopen. Hierdoor kan je iets aanzetten zolang je die knop indrukt, of je kan het beetje elektriciteit aflezen en daar aan de hand van een bepaalde actie met je Arduino doen!

- **Condensator** De condensator slaat tijdelijk stroom op zolang deze gevoed wordt. Wanneer er dan geen stroom meer op staat, laat hij alles wat weer opgeslagen is weer vrij.

Addendum 3: Programmeer referenties

Een GPIO-poort als uitgang gebruiken

```
pinMode(13, OUTPUT);
```

Hierboven staat de code om GPIO-poort nummer 13 als uitgang te gebruiken. Wanneer je een andere poort wil gebruiken, hoef je simpelweg alleen maar het getal 13 te veranderen.

Een GPIO-poort als ingang gebruiken

```
pinMode(13, INPUT);
```

Hierboven staat de code om GPIO-poort nummer 13 als ingang te gebruiken. Wanneer je een andere poort wil gebruiken, hoef je simpelweg alleen maar het getal 13 te veranderen.

De Arduino even laten wachten

```
delay(1000);
```

Door delay te gebruiken laat je de Aruino even wachten. Hoe lang hij wacht is aan de hand van het getal dat je meegeeft. Dit is een getal in microseconden (1 seconde = 1000 microseconde). Hierboven vertellen we de Arduino dus dat hij 1 seconde moet wachten voordat hij weer door moet gaan.

Een digitaal signaal uitsturen

```
digitalWrite(13, HIGH);
```

Een digitaal signaal is een 1 of een 0, of beter gezegd in elektrische bewoording: Een hoog of een laag. Aan deze functie geef je eerst aan over welke poort je het signaal wil sturen. Vervolgens na de komma geef je een HIGH of LOW aan om aan te geven of je wel of niet iets wil uitsturen.

Een digitaal signaal uitlezen

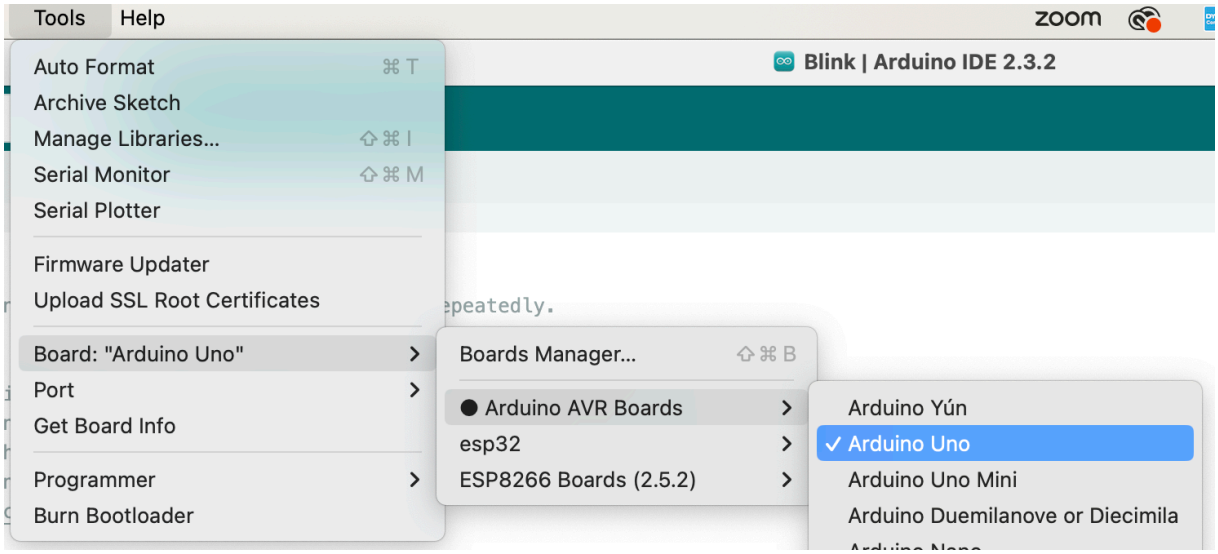
```
digitalRead(13);
```

Met deze functie kunnen we pin nummer 13 uitlezen. Wanneer er stroom binnenkomt op poort 13 zal deze functie de waarde 1 teruggeven. Als er niets op binnen komt, zal dit een 0 zijn.

Addendum 4: Hoe zet je je sketch op een Arduino?

Allereerst is het belangrijk dat je je IDE goed configureert. Je wil namelijk wel dat het ook daadwerkelijk bij de Arduino aankomt. Het is hiervoor noodzakelijk dat je je Arduino met een USB-kabel aan je computer hebt aangesloten. Vervolgens ga je in de IDE naar Tools (in het menu) en onder Board moet je het type Arduino selecteren welke je hebt aangesloten. Daar onder (nog steeds in Tools in het menu) staat ook "Port". Hier moet je degene selecteren waar je Arduino op aangesloten zit. Vaak ik hier ook maar 1 optie beschikbaar, maar controleer het wel voor de zekerheid!

Hier onder zie je hoe dat er uit kan zien:



Vervolgens ga je in het menu naar Sketch en druk je op "Upload", of druk je op het pijltje naar rechts linksboven in je IDE. Er wordt vervolgens gekeken of je code helemaal klopt, en wanneer dit zo is wordt het op de Arduino geüpload.

Het uploaden kan dus als volgt gedaan worden: