Les 1: Kennismaken met de Arduino


Bekijk de les op YouTube door op het logo te klikken


Volg de les via een website


Materialen nodig:

- Arduino UNO
- USB-kabel

Inleiding

Het doel van een **Arduino** is om elektronica toegankelijk te maken voor iedereen, zonder dat je hiervoor een uitgebreide elektronica-opleiding hoeft te volgen. Arduino moedigt gebruikers aan om creatief aan de slag te gaan en zelf projecten te ontwikkelen.

De drijvende kracht achter Arduino, en vooral het succes ervan als **Open Source-platform**, is **Massimo Banzi**. Hij heeft een belangrijke rol gespeeld in de ontwikkeling en popularisering van Arduino. Je kunt meer over hem te weten komen door zijn inspirerende **TED Talk** te bekijken, die je kunt vinden door op het YouTube-logo te klikken.


TED-talk: How Arduino is open-sourcing imagination | Massimo Banzi

Wat is een Arduino?

Een Arduino is een **experimenteerbordje met een microcontroller**, omgeven door alle benodigde elektronica om te communiceren met sensoren, actuatoren en computers. Je programmeert de Arduino in een afgeleide van **C/C++** via de **Arduino IDE** (Integrated Development Environment).

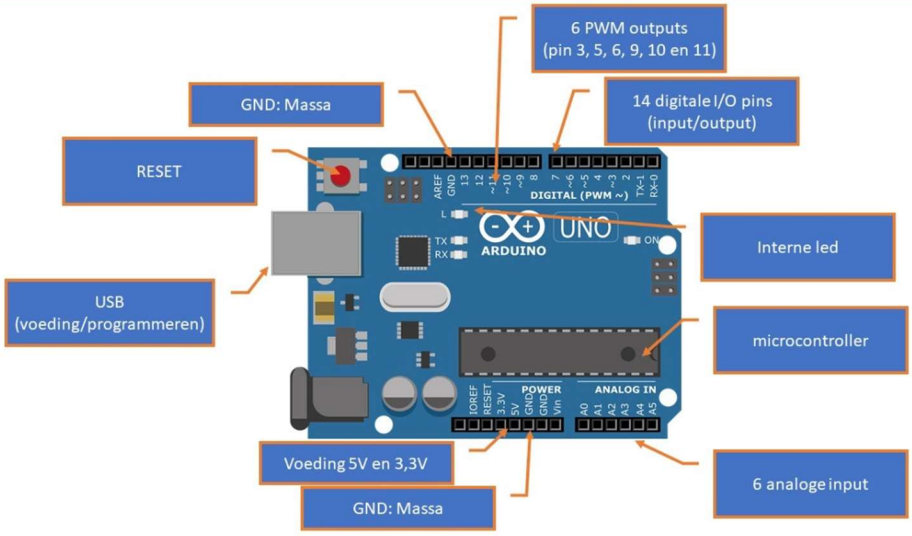
Een van de grote voordelen van Arduino is dat het **open source** is. Dit betekent dat er naast de bekende blauwe en blauwgroene bordjes ook veel **clones** en andere experimenteerbordjes op de markt zijn, die eveneens geprogrammeerd kunnen worden in de Arduino IDE.

De Arduino UNO

In deze cursus werken we met de **Arduino UNO**, een van de meest populaire en veel gebruikte experimenteerbordjes. Hoewel er inmiddels verschillende soorten Arduino-bordjes bestaan, was de UNO het eerste model dat ontwikkeld werd. Sinds de introductie in **2005** is de Arduino UNO nog steeds in productie, ondertussen in de **vierde generatie**. Dit komt vooral door de ideale vormgeving en pinindeling, die het bordje bijzonder geschikt maakt om te leren werken met microcontrollers.

Met de Arduino UNO kun je op een eenvoudige en toegankelijke manier kennismaken met de wereld van elektronica en programmeren, waardoor het een perfecte keuze is voor beginners en gevorderden alike.

Opbouw van de Arduino UNO



Op de afbeelding zie je de verschillende onderdelen van de Arduino UNO. Aan de **bovenzijde** van het bordje vind je **14 digitale I/O-pinnen** (Invoer/Uitvoer-pinnen). Deze pinnen zijn genummerd van 0 tot 13 en kunnen zowel als invoer (input) als uitvoer (output) worden gebruikt. Wel moet je in je sketch (het Arduino-programma) aangeven of een pin als input of output moet functioneren. De digitale pinnen 0 tot en met 13 worden ook wel aangeduid met **GPIO**. Dit staat voor General Purpose Input Output. Digitale pin 13 wordt zo **GPIO 13**.

Aan de **onderzijde** van het bordje bevinden zich **6 analoge input pinnen**, genummerd van **A0 tot A5**. Het verschil tussen analoge en digitale ingangen zullen we later bespreken, zodra we met ingangen gaan werken.

Voedingspinnen

De meeste microcontrollers werken op 3,3 volt, maar de Arduino-familie is een uitzondering: deze werkt meestal op 5 volt. Daarom vind je op de Arduino UNO zowel een **5V-pin** als een **3,3V-pin**. Daarnaast zijn er **2 GND-pinnen** (GROUND), die een potentiaal van 0 volt hebben. Aan de bovenzijde van het bord vind je nog een derde GND-pin. Als deze 3 GND-pinnen zijn intern met elkaar verbonden.

Je kunt de 5V- en GND-pinnen zien als de aansluitingen van een 5V-bron, waarbij de 5V-pin de positieve kant en de GND-pin de negatieve kant vertegenwoordigt.

Overige onderdelen

Op het bordje is ook de **microcontroller** prominent aanwezig, evenals een **USB-aansluiting**. Deze USB-poort wordt gebruikt om de microcontroller te programmeren, maar kan ook dienen om het bordje van stroom te voorzien.

Verder zijn er **4 LEDjes** op het bordje te vinden:

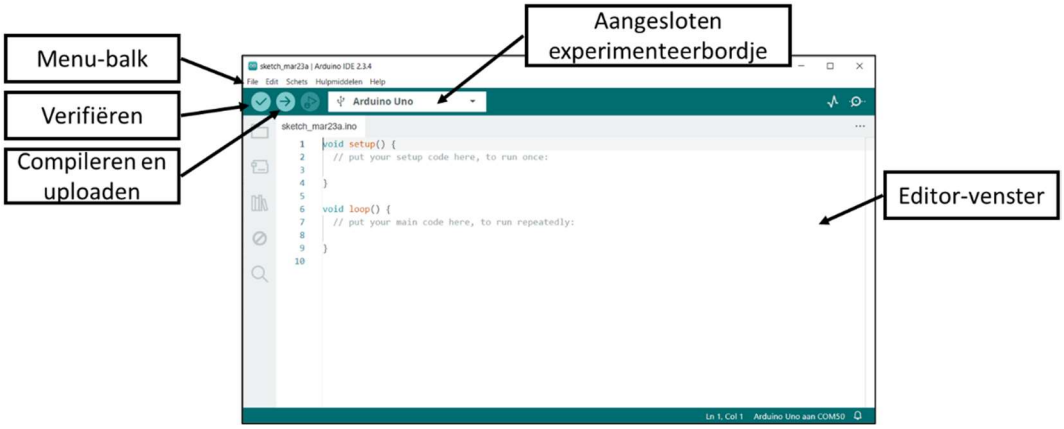
- Twee LEDjes, gelabeld met **RX** en **TX**, geven aan wanneer er data wordt verzonden (TX) of ontvangen (RX) via de USB-verbinding.
- Het derde LEDje is de **interne LED**, die verbonden is met **pin 13**. Deze LED kan handig zijn voor testdoeleinden.

- Het laatste LEDje geeft aan of de Arduino voeding krijgt.
  Is de voeding aangesloten, maar het LEDje brandt toch niet, dan zit er een grote fout in je circuit.

Met deze opbouw is de Arduino UNO een ideaal bordje om mee te experimenteren en de basisprincipes van microcontrollers te leren.

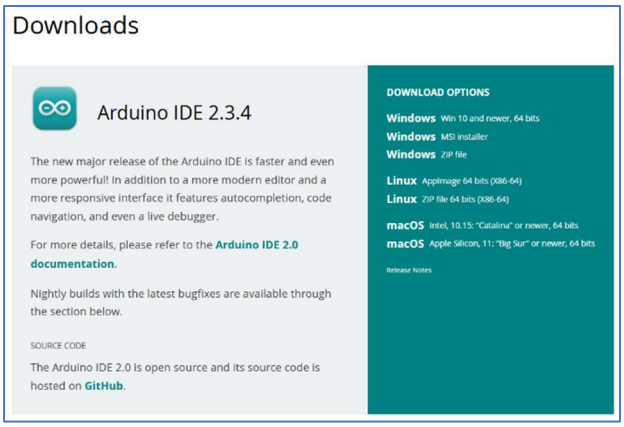
De Programmeeromgeving: Arduino IDE

Wanneer we het over **Arduino** hebben, gaat het niet alleen over de experimenteerbordjes, maar ook over de **programmeeromgeving**, de **Arduino IDE** (Integrated Development Environment). Deze software is speciaal ontwikkeld om Arduino-bordjes te programmeren. De nieuwste versie van de IDE is **Arduino IDE 2.3.4**.



De IDE installeren.

1. Surf naar https://www.arduino.cc/en/software/
2. Klik op de Download Option die bij jou past.



3. Klik op “JUST DOWNLOAD”

4. Installeer met het gedownloade bestand de IDE, of unzip de zip-file.

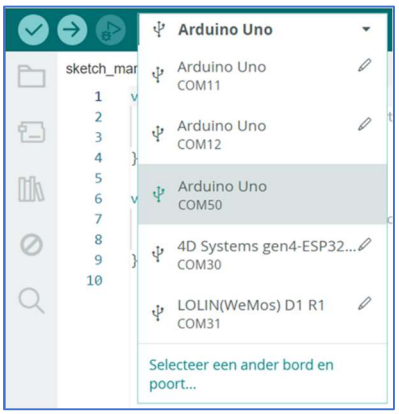
Wat kun je met de Arduino IDE?

De Arduino IDE biedt drie belangrijke functies:

1. **Schrijven van de sketch**: Je kunt je code (ook wel een **sketch** genoemd) schrijven in de editor.
2. **Compileren**: De sketch wordt omgezet naar machinetaal, zodat de microcontroller deze kan begrijpen.
3. **Uploaden**: De gecompileerde sketch wordt naar de microcontroller op het Arduino-bordje gestuurd.

Om dit goed te laten werken, moet de IDE weten welk type bordje je gebruikt en op welke **COM-poort** het bordje is aangesloten. Meestal wordt het bordje automatisch gedetecteerd, maar soms kan de COM-poort problemen opleveren. Je kunt deze instellingen handmatig aanpassen.

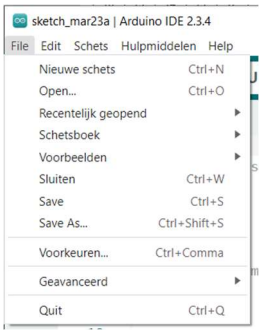
Hoe werkt de Arduino IDE?



- **Bordje en COM-poort selecteren**:
  Bovenaan de IDE kun je het type bordje (bijvoorbeeld Arduino UNO) en de COM-poort selecteren. Als je veel opties ziet in de keuzelijst, kies dan de verwijzing naar de Arduino UNO **zonder een "potlood"** ernaast. Dit is de juiste poort.
- **Verifiëren / compileren:**
  Als je op het **vinkje** klikt, wordt de sketch gecontroleerd en gecompileerd. Als er een fout in de syntax (de schrijfwijze van de code) wordt gevonden, krijg je een foutmelding.
- **Uploaden:**
  Klik je op het **pijltje**, dan wordt de sketch eerst gecompileerd en daarna geüpload naar de Arduino.

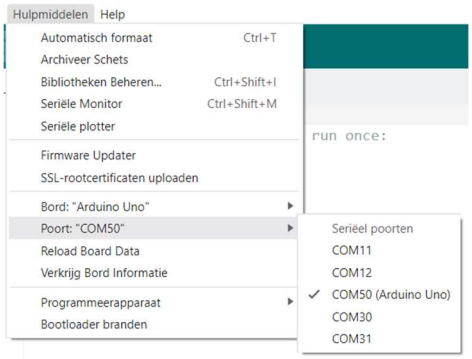
De Menubalk

De Arduino IDE heeft een handige menubalk met verschillende opties:



- **File-menu:**
  Hier kun je een nieuwe sketch starten, een sketch opslaan, of een eerder opgeslagen sketch openen. Ook kun je vanuit dit menu een aantal voorbeelden openen, die handig zijn om te leren en te experimenteren.

- **Hulpmiddelen-menu:**
  In dit menu kun je het type bordje en de COM-poort selecteren. Hier is het vaak duidelijker te zien met welke COM-poort de Arduino verbonden is.



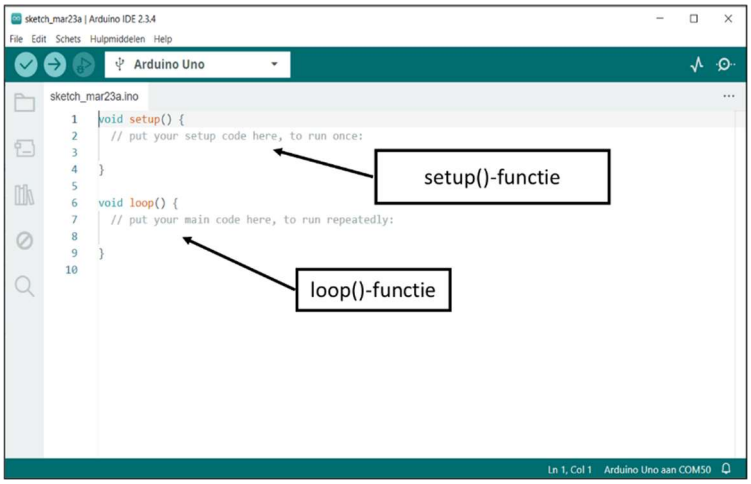
Je eerste sketch

"**Hello World!**" is een eenvoudig programma dat vaak wordt gebruikt om te controleren of een nieuwe compiler of interpreter correct is geïnstalleerd en werkt. Het programma toont de tekst "Hello World!" op het scherm, wat aangeeft dat de ontwikkelomgeving goed functioneert en dat je klaar bent om meer complexe code te schrijven. Dit programma wordt beschouwd als een traditionele eerste stap bij het leren van een nieuwe programmeertaal.

In de wereld van **microcontrollers** is de tegenhanger van "Hello World!" het "**Blink**"-programma. Hierbij wordt een LEDje op een experimenteerbordje aan en uit geschakeld met een frequentie van **0,5 Hz**, wat betekent dat het LEDje elke seconde knippert. Dit eenvoudige programma controleert of de microcontroller correct is geprogrammeerd en of de basis hardware, zoals de **GPIO-pinnen**, goed functioneert. Net zoals "Hello World!" is "Blink" vaak de eerste stap bij het werken met microcontrollers.

Het algoritme

Wanneer je een **lege sketch** opent in de Arduino IDE, staat er al een beetje code klaar. Dit zijn twee functies, oftewel blokken code die kunnen worden aangeroepen. Deze functies vormen de basis van elk Arduino-programma.



De setup()- en loop()-functies

De twee functies die standaard in de sketch staan, zijn de **setup()**- en **loop()**-functie.

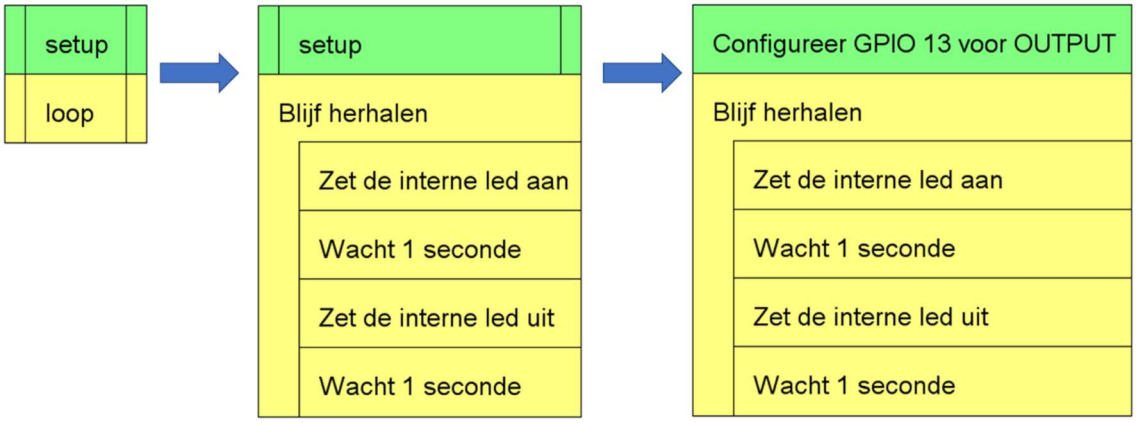
- Wanneer je de Arduino opstart, wordt eerst de **setup()**-functie automatisch aangeroepen. Deze functie wordt **één keer** uitgevoerd en wordt typisch gebruikt om instellingen te configureren, zoals het definiëren van pinnen als input of output.

- Na de setup() wordt de **loop()**-functie aangeroepen. Deze functie is een **lus** en blijft continu herhaald worden zolang de Arduino aanstaat. Dit is een belangrijk verschil tussen het programmeren van een microcontroller en een "gewone" computer. Bij een gewone computer wil je nooit in een oneindige lus terechtkomen, maar bij een microcontroller is dit juist de standaard. Dit is logisch, want een microcontroller wordt vaak gebruikt voor taken die continu moeten worden uitgevoerd, zoals het regelen van verkeerslichten.

Zowel de **setup()**- als de **loop()**-functie moeten in je sketch aanwezig zijn, ook al zijn ze leeg. Ze worden automatisch aangeroepen door de Arduino, dus hun aanwezigheid is verplicht.

Grafische weergave met een Nassi-Shneidermann diagram

Om het algoritme dat we nodig hebben duidelijk weer te geven, gebruiken we een N**assi-Shneidermann diagram**. In dit diagram worden functies aangegeven door een rechthoek waarvan de verticale zijden uit dubbele lijnen bestaan.



Uitwerking van de loop()-functie

We beginnen met het uitwerken van de **loop()**-functie. Deze functie is een iteratie (herhaling) waarin we het volgende doen:

1. Het interne LEDje aanzetten.
2. Eén seconde wachten.
3. Het LEDje uitzetten.
4. Weer één seconde wachten.

Dit proces blijft zich oneindig herhalen zolang de Arduino aanstaat.

Configuratie van de GPIO-pin

Het interne LEDje op de Arduino UNO is verbonden met **GPIO 13**. Om dit LEDje te kunnen gebruiken, moeten we GPIO 13 configureren als een **output**. Dit doen we in de **setup()**-functie, omdat deze instelling maar één keer hoeft te worden uitgevoerd bij het opstarten van de Arduino.

De sketch

Een sketch, script of programma is de vertaling van een algoritme naar code. We gaan dit ook blokje per blokje doen.

Configureren van een GPIO (setup):

```
Configureer GPIO 13 voor OUTPUT
```

Wanneer we een GPIO willen configureren als input of output hebben we 3 dingen nodig:

1. De instructie om de pin te configureren
2. De nummer van de pin die we gaan configureren
3. Weten of de pin als OUTPUT of als INPUT moet geconfigureerd worden.

Syntax:

```
pinMode(pin, mode)
```

Parameters

- `pin`: de nummer van de Arduino-pin (GPIO) die we gaan configureren
- `mode`: `OUTPUT`, `INPUT` of `INPUT_PULLUP`
  - `OUTPUT`: pin wordt geconfigureerd als OUTPUT
  - `INPUT`: pin wordt geconfigureerd als INPUT
  - `INPUT_PULLUP`: pin wordt geconfigureerd als INPUT maar met een interne pull-up weerstand. Wat dit is gaan we in later behandelen.

Bron:
https://docs.arduino.cc/language-reference/en/functions/digital-io/pinMode/

GPIO 13 configureren als output:

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

Merk op dat elke instructie moet afgesloten worden met een punt-komma ";". Een blok code wordt geplaatst tussen twee accolades. Omdat een functie een blok code is die kan aangeroepen worden, moet ook deze code tussen twee accolades geplaatst worden.

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

LEDje laten knipperen (loop)

Zet de interne led aan

Zet de interne led uit

Wanneer we een digitale pin als OUTPUT configureren, dan kan je dit voorstellen alsof die aansluiting, via een schakelaar, verbonden is met de +5V. In de sketch kunnen we zeggen dat de schakelaar moet gesloten worden, en dan wordt de uitgang hoog (HIGH), of dat de schakelaar moet openen, en dan wordt de uitgang laag (LOW).

Sluiten we een LEDje aan op deze uitgang, dan zal dit LEDje aan gaan als de uitgang HOOG is, en uit gaan als de uitgang LAAG is. Omdat het interne LEDje, het LEDje op het experimenteerbordje, verbonden is met uitgang 13, kunnen we dit op deze manier laten aan en uit gaan.

Om een uitgang hoog of laag te maken moet je:

1. De instructie om de pin hoog of laag te maken kennen
2. De nummer van de pin die we wilt veranderen
3. Of de pin hoog of laag moet worden

Syntax:

```
digitalWrite(pin, waarde)
```

Parameters

- `pin`: de nummer van de Arduino-pin (GPIO) die we gaan configureren
- `waarde: HIGH of LOW`

Bron :
https://docs.arduino.cc/language-reference/en/functions/digital-io/digitalwrite/

Je maakt uitgang 13 hoog met:

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

en laag mag:

```
digitalWrite(13, LOW);
```

Wacht 1 seconde

Het laatste stukje dat we moeten vertalen is de vertraging van 1 seconde. Dit kan met de delay()-functie.

Syntax:

```
delay(waarde)
```

Parameters:

- `waarde`: de tijd in milliseconden

Bron :
https://docs.arduino.cc/language-reference/en/functions/time/delay/

De code om 1 seconde, of 1 000 ms te wachten is:

```
delay(1000);
```

Alle blokjes zijn nu gekend. Nu moeten we ze gewoon gaan samenvoegen.

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

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

Compileren en uploaden

Als laatste stap vertalen we de sketch naar machinetaal en uploaden deze naar de Arduino.

1. **Voer de sketch in of download deze:**

- Als je de sketch nog niet hebt ingetypt in de Arduino IDE, doe dit dan nu.
- Of klik op het Arduino-logo om de sketch te downloaden.

2. **Controleer de instellingen:**

- Zorg dat **"Arduino UNO"** als bord is geselecteerd.
- Kies de juiste **COM-poort**.
    - Let op: Als je de verbinding tussen de computer en het Arduino-bord hebt verbroken en opnieuw aangesloten, kan het nummer van de COM-poort veranderen.
- In de rechter benedenhoek van het IDE-venster zie je of het bord succesvol is verbonden.

3. **Upload de sketch:**

- Klik op de **pijlknop (→)** om de sketch te compileren en naar de Arduino te uploaden.

Lift off!

De laatste stap is de sketch testen.

Testen van de sketch