# 1.0. Arduino Programmeren voor Beginners
## Deel 1: Opzet


Dit is het eerste deel in een serie artikelen die zijn geschreven voor jonge mensen die graag willen leren programmeren. Uiteraard hebben we daarvoor een leuk stukje hardware nodig (Arduino) en moeten we natuurlijk een beetje bekend worden met de programmeertaal C zoals deze voor Arduino Programmeren gebruikt wordt. Overigens: dit is slechts basis programmeren, we gaan dus geen robot bouwen …

In dit eerste artikel gaan we kijken naar de spullen die we nodig hebben om met de Arduino aan de slag te kunnen. We gaan een Arduino kiezen en kijken hoe wel de IDE, de programmeer omgeving, op moeten zetten zodat we de voorbeelden in de komende artikelen kunnen uitproberen.

Deze lessen richten zich hoofdzakelijk op het Arduino Programmeren voor beginners – gebrek aan kennis voor wat betreft de Engelse taal en wiskundige achtergrond hoeft waarschijnlijk geen probleem te zijn. Het gebruik van extra elektronica componenten blijft beperkt tot een minimum en bewaren we voor een volgende reeks.

# Inhoud

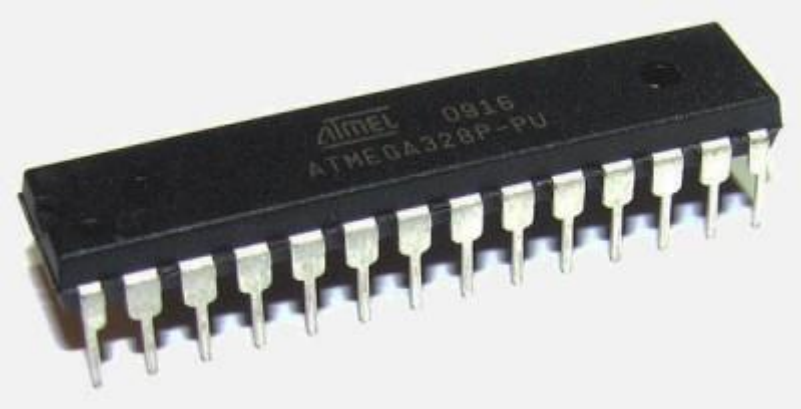
## 1.1. Wat is een Arduino?

De Arduino is een klein print plaatje, met wat elektronica onderdelen opgezet rond een zogenaamde Micro-Controller, welke we voor de Arduino Programmeren voor Beginners “cursus” willen gaan gebruiken.

Een **Micro-Controller** is in principe een simpele maar relatief complete minicomputer (niet te verwarren met “mini computer“). Het is stukken minder krachtig in vergelijk met een desktop of laptopcomputer of zelfs een mobiele telefoon. Je vindt ze echter zo’n beetje overal in huis, zoals een wekker, was machine, magnetron, etc.

In ons geval gebruiken we een specifieke Micro-Controller welke op de Arduino gebruikt wordt en wel een zogenaamde Atmel AVR. Atmel is de fabrikant, en AVR is de model serie. In principe is een Atmel AVR is een processor, met geheugen, een soort van besturingssysteem, en leuke aansluitingen om allerlei zaken aan te sturen zoals lampjes, schakelaars, motortjes, relais, sensoren, etc.

Het nadeel is echter, zeker voor beginners, dat een zo’n Micro-Controller slechts een enkele chip is – wat aansluiten en experimenteren wat lastige maakt.




*Figuur 1- 1*

*Voorbeeld van een Atmel AVR chip*

Omdat aansluiten zo lastig is, heeft de organisatie “**Arduino**” een printje gemaakt (het blauwe stuk in onderstaande afbeelding) waarbij het aansluiten stukken makkelijker gaat. Over de jaren hebben ze een aantal modellen en varianten ontwikkeld – verschillend in maat, snelheid, geheugen, prijs, etc.


*Figuur 1- 2*

*Arduino Uno R3*

Zoals je in de afbeelding kunt zien is het Arduino printje voorzien van de eerdergenoemde chip (de Micro-Controller). Daarnaast heeft het een stroom aansluiting, een USB-aansluiting en een aantal zwarte strips die het straks makkelijk maken om extra dingen aan te sluiten zoals lampjes, motortjes, sensoren, etc.

## 1.2. Welk model voor Beginnen met Arduino Programmeren?

Ik heb zelfgekozen voor de Arduino Uno R3. Deze is solide, betaalbaar, eenvoudig te gebruiken en kan een stootje hebben. Het is bovendien (op dit moment althans) een van de meest gebruikt modellen. Je kunt ze eenvoudig vinden online, zoals b.v. bij Amazon of eBay, en legio andere online winkels.

Merk wel op dat er vaak clonen (imitatie) modellen worden aangeboden welke stukken goedkoper kunnen zijn. Ik adviseer altijd om eerste met een originele Arduino te beginnen.

Je kunt de Arduino Uno R3 ook kopen als starters-kit waarbij dan allerlei leuke speeltjes zitten, maar de prijs gaat dan wel flink omhoog.

**Merk op dat we ook een USB A naar USB B kabel nodig hebben om de Arduino op de PC aan te kunnen sluiten.**

*Figuur 1- 3*

*USB Kabeltje met een Type-A en Type-B aansluiting*

De originele Arduino's kun je herkennen aan de logo's, maar vooral aan het blauwe printje met een witte achterkant.


*Figuur 1- 4*

*Arduino Uno R3 – Voorkant en achterkant*

De Uno R3 komt in twee varianten. De gewone (links) en de zogenaamde "SMD" versie, wat in principe hetzelfde is als de gewone versie, alleen is dan de chip aanzienlijk kleiner en is de chip gesoldeerd op het printje in plaats van geplaatst in een voetje zoals bij de normale versie. Het nadeel daaraan is dat als je de Micro-Controller per ongeluk beschadigd, dat je deze dus niet zomaar 1-2-3 eruit kunt trekken en vervangen.

Je ziet de verschillen hieronder meteen. Links de “gewone” en rechts de SMD-versie.

*Figuur 1- 5*

*Arduino UNO: Standaard (links) versus SMD (rechts)*

Je bent overigens niet verplicht om de Arduino Uno R3 te nemen – andere modellen zullen vergelijkbaar werken (misschien met uitzondering van een paar van de meest recente en meer complexe modellen).

## 1.3. Wat is een IDE en wat is een Compiler?

Een IDE is een geïntegreerde ontwikkelomgeving of wel een “**I**ntegrated **D**evelopment **E**nvironment“, wat eigenlijk wil zeggen dat alle programma’s en software die we nodig hebben voor het programmeren, lekker handig bij elkaar zit in een handig pakketje. Je hoeft dus niet op zoek te gaan naar allerlei onderdelen en in ons geval is dat pakketje gewoon gratis.

Je kunt het van **de Arduino website downloaden**.

We hebben ook een versie op Tweaking4All staan, maar het is beter om de versie van de Arduino website te halen – dan weet je zeker dat je de meest actuele versie hebt.

DOWNLOAD - Arduino Windows

**Platform:** Windows, 32 bits
**Bestand:** arduino-ide-windows.exe
**Versie:** 1.6.7
**Omvang:** 80.7 MiB
**Datum:** 8 feb 2016

**Download Nu**

## DOWNLOAD - Arduino MacOS X

**Platform:** Mac OS X
**Bestand:** arduino-ide-macosx.zip
**Versie:** 1.6.7
**Omvang:** 136.4 MiB
**Datum:** 8 feb 2016

Download Nu

## DOWNLOAD - Arduino Linux 32-bit

**Platform:** Linux, 32 bits
**Bestand:** arduino-ide-linux32.tar.xz
**Versie:** 1.6.7
**Omvang:** 92.1 MiB
**Datum:** 8 feb 2016

Download Nu

## DOWNLOAD - Arduino Linux 64-bit

**Platform:** Linux, 64 bits
**Bestand:** arduino-ide-linux64.tar.xz
**Versie:** 1.6.7
**Omvang:** 91.0 MiB
**Datum:** 8 feb 2016

Download Nu

Het IDE "pakket" heeft een programma om tekst in te voeren (een zogenaamde text **editor**) waarmee we de code (instructies) van ons programma mee kunnen invoeren, daarnaast de hulpmiddelen om verbinding met de Arduino te maken, een aantal bibliotheken (**libraries** – code verzamelingen om ons werk te besparen), en een zogenaamde **compiler**.

### Vertalen …

De compiler is een van de belangrijkste onderdelen omdat we ons programma in een hogere taal schrijven. Daarmee bedoelen we dat de "taal" of "programmeer taal" dichter bij de menselijke taal zit dan bijvoorbeeld machine taal. In ons geval gebruiken we de programmeertaal "C". "C" is een taal die wat weg heeft van de Engelse taal, waardoor het voor (Engel sprekende mensen) makkelijker te lezen is. De taal "C" bestaat al erg lang, en er

zijn legio varianten in omloop – de zogenaamde dialecten, net zoals we dat zien in de menselijke talen.

Het probleem met computers is dat ze de menselijke taal niet spreken, een programmeer taal zoals “C” komt al dichterbij maar ook die taal begrijpt de Micro-Controller helemaal niet. We hebben daarom dan ook een tolk vertaler nodig.

Dit vertalen kan op twee manieren gedaan worden.

## Interpreter – Vertalen tijdens gebruik

Het vertalen terwijl we het programma “draaien” of aflopen, wordt gedaan door een zogenaamde **interpreter**.

Zie het als het lezen van een boek, geschreven in een taal die jij niet kent. Om dat boek te kunnen lezen moet er een vertaler naast je zitten, die eerst wat tekst leest en deze dan vertaald en hardop uitspreekt zodat jij het verhaal kunt volgen.

Het voordeel is dat we meteen aan de slag kunnen met het lezen van dat boek in die vreemde taal. Het nadeel is dat we dus wel even een tolk in huis moeten hebben, die dan ook nog eens steeds een stukje moet lezen, vertalen en hand hardop uitspreken … dus zeg maar regel voor regel.

Dit is dus langzamer dan wanneer het hele boek al in het Nederlands zou zijn. Dus snelle start, maar langzaam lezen.

## Compiler – Vertalen voor gebruik

Hierbij komt de zogenaamde **compiler** om de hoek kijken.

Een compiler kun je zien als een vertaler die het hele boek gelezen en vertaald heeft – je hebt dus een Nederlandse versie van het boek in he handen.

Het voordeel is dus dat we geen tolk in de voorraad kast hoeven te hebben, en dat we meteen kunnen lezen als we dat zouden willen, zonder dat er iets vertaald hoeft te worden.

Het nadeel is echter dat het boek eerst vertaald dient te worden, in z’n geheel, voor dat we het kunnen gaan lezen.
Dus langzame start, maar snel lezen.

Dit geldt ook voor de computer. Het vertalen van het hele programma, voor we het starten, kost wat extra tijd, maar eenmaal vertaald hebben we geen hulp meer nodig. Daarbij komt dan ook nog eens dat het “lezen” sneller gaat, wat zicht laat zien door een programma wat veel sneller is.

*De Arduino IDE gebruikt dus een **COMPILER**.*

## 1.4. Hoe zetten we de Arduino IDE op?

Na het downloaden van de Arduino IDE voor jouw computer computer, tijd om het te installeren.

### Download en Installeer de IDE

Installatie gaat in eerste instantie simpel – zoals eerder gezegd: download de versie die voor jouw computer geschikt is.

Windows gebruikers dubbelklikken vervolgens op het EXE bestand en volgen de instructies, Mac en Linux gebruikers zullen eerst het bestand moeten uitpakken. Vervolgens slepen de Mac OS X gebruikers de Arduino applicatie naar de Applicaties directory, en Linux gebruikers gebruiken de voor hun gebruikelijke methode.

Eenmaal geïnstalleerd ziet het beeld er voor Windows, Mac OS X en Linux, vergelijkbaar uit.

**Sluit nu de Arduino, via de USB kabel, aan op jouw computer!**

***N.b.*** : Drivers zullen geen probleem zijn voor zowel Windows, Linux als Mac OS X. De recentere versie handelen dit ook onder Windows goed voor je af.


*Figuur 1- 6*

*Arduino onder MacOS X, Linux en Windows*

## Initiële Configuratie van de Arduino Software

Na installatie moeten we even kijken naar een initiële instelling zodat onze Arduino correct gekozen wordt (niet vergeten de Arduino aan te sluiten op de computer!).

### Windows gebruikers

Als eerste moeten we de juiste USB "poort" kiezen.
Open het menu "Tools" "Serial Port" en kies de juiste "Com" poort (USB), dit is helaas sterk afhankelijk van jouw computer en kan dus gerust COM4 zijn op de ene computer en COM2 op een andere. Omdat de meeste computers tegenwoordig geen com-poorten meer hebben, kan dit zelfs COM1 zijn.

Vervolgens moeten we het juiste Arduino type instellen of controleren.
In het menu “Tools” “Board” kiezen we (of controleren we) het juiste Arduino board wat in mijn geval de “Arduino Uno” was.

### Linux gebruikers

Als eerste moeten we de juiste USB “poort” kiezen.
Open het menu “Tools” “Serial Port” en kies het juiste the serieel device voor de USB verbinding van jouw Arduino (mijn Ubuntu setup noemde het `/dev/ttyMCA0` , maar op jouw computer kan dit dus een andere naam hebben).

Vervolgens moeten we het juiste Arduino type instellen of controleren.
In het menu “Tools” “Board” kiezen we (of controleren we) het juiste Arduino board wat in mijn geval de “Arduino Uno” was.

### Mac OS X users:

Als eerste moeten we de juiste USB “poort” kiezen.
Open het menu “Tools” “Serial Port” en kies het juiste the serieel device voor de USB verbinding van jouw Arduino, bij mij was dit zoiets als `/dev/cu.usbmodem1421 (Arduino Uno)` maar dat kan op jouw computer dus anders zijn.

Vervolgens moeten we het juiste Arduino type instellen of controleren.
In het menu “Tools” “Board” kiezen we (of controleren we) het juiste Arduino board wat in mijn geval de “Arduino Uno” was.

Nu dat we de Arduino IDE hebben opgezet … tijd om te kijken of het werkt.

## 1.5. Testen van onze Arduino

We gaan alles even testen met een klein programma dat een van de lampjes (LED) op het printje van de Arduino laat knipperen. Het programma heb ik even snel van de Arduino website gehaald – het is maar een test.

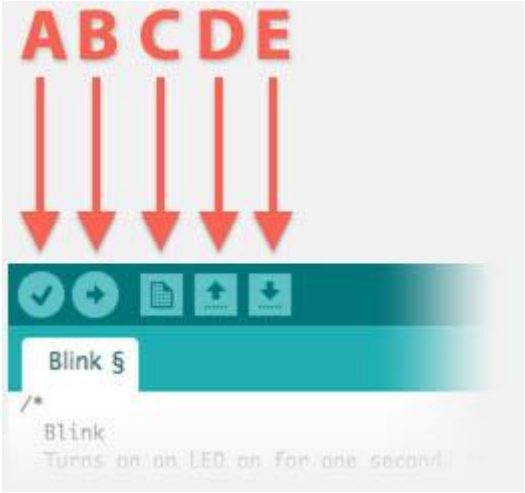
Als eerste moet je de volgende code kopiëren en in de tekst editor van de Arduino IDE plakken waarbij het alle bestaande tekst in de editor vervangt.

```
/*
  Blink
  Herhaaldelijk aan schakelen van een LED aan voor 1 seconde, dan weer uit voor een
seconde.

  Dit voorbeeld staat in het public domain.
 */

// Pin 13 heeft een LEDje op de meeste Arduino modellen.
// geef het een naam:
int led = 13;

// de setup functie draaid 1x als je de Arduino aanzet of op reset drukt:
void setup() {
  // Initialiseer de digitale pin as als output.
  pinMode(led, OUTPUT);
}

// De loop routine herhaald zichzelf oneindig:
void loop() {
  digitalWrite(led, HIGH);   // LED aan (HIGH is het voltage niveau - 5V)
  delay(1000);               // wacht een seconde
  digitalWrite(led, LOW);    // LED uit (LOW is stroom uitzetten)
  delay(1000);               // wacht een seconde
}
```

De volgende stap, mits we geen type fouten of plak fouten hebben gemaakt, is dat we dit programma moeten vertalen (dit doet de compiler en wordt ook wel “compilen” of “compileren” genoemd), en vervolgens moet het vertaalde resultaat naar de Arduino gestuurd worden. Gelukkig kan de Arduino IDE dit allemaal voor ons regelen met een enkele klik.

Zie je de 5 knoppen, boven in de Arduino IDE, zoals weergegeven in figuur 7?

**Na het plakken van de code moeten we op de “B” knop klikken.**




*Figuur 1- 7*

*Arduino Software – Handige snelkoppelingen*

*Useful Arduino Shortcuts*

| Knop | Doel |
|---|---|
| **A** | Verify – Verificatie van onze Code |
| **B** | Upload – Verificatie, Compileren, en Uploaden naar de Arduino |
| **C** | New – Maak een nieuwe Sketch (code) |
| **D** | Open – Open een bestaande Sketch |
| **E** | Save – Sla een Sketch op |

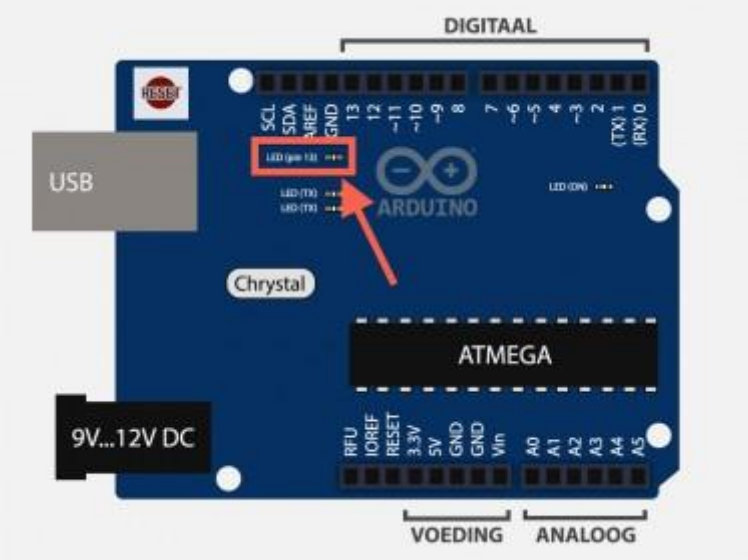
In de tabel zien je dat knop "**A**" gebruikt kan worden om jouw code te verifiëren op fouten. Knop "**B**" doet dit ook, maar vertaald het daarna ook nog eens (compileren) en als alles goed ging, stuurt deze de vertaling (het programma) naar de Arduino.

De knoppen "**C**", "**D**" en "**E**" wordt gebruikt om met bestanden te werken, zoals het maken van een nieuw document, opslaan (Save) en inlezen (Open).

Je ziet dat we het woord “**Sketch – (Engelstalig)**” een paar keer hebben gebruikt.
Dit is de jargon (lingo) die Arduino gebruikers gebruiken om **source code**, of te wel “code”, aan te geven – in andere woorden: ons programma, geschreven in de taal C.

*Arduino **Source code** in een bestand, noemt men in Arduino jargon een **SKETCH***

Nadat je op de “B” knop hebt geklikt, zal jouw programma (source code) gecontroleerd worden, vertaald worden en naar de Arduino gestuurd worden, mits alles goed ging natuurlijk. Tijdens het uploaden (naar de Arduino sturen) zul je de LEDjes (**LED** = Light Emitting Diode) van de Arduino wat zenuwachtig zien knipperen. Dit wordt gedaan om kenbaar te maken dat de Arduino bezig is. Vervolgens start het programma: 1 seconde AAN, 1 seconde UIT, 1 seconde AAN, 1 seconde UIT etc …




*Figuur 1- 8*

*Arduino Uno – Het LEDje dat aan pin 13 zit*

Gefeliciteerd! Je hebt een Arduino aangesloten op jouw computer, de Arduino IDE geïnstalleerd en een eerste test programma naar de Arduino gestuurd.

Tijd voor deel 2 … 

Als je vragen hebt: stel ze dan hieronder, en bedenk dat er geen domme vragen zijn, behalve dan natuurlijk de vraag die niet gesteld is. We zijn allemaal een keer bij nul begonnen!

Volgende hoofdstuk: **Arduino Programmeren voor Beginners – Deel 2: Uitvoer**