[Arduino Programmeren voor Beginners – Deel 1: Opzet](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/)



Overzicht van dit Hoofdstuk

* **1**[Wat is een Arduino?](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#WatiseenArduino)
* **2**[Welk model voor Beginnen met Arduino Programmeren?](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#WelkmodelvoorBeginnenmetArduinoProgrammeren)
* **3**[Wat is een IDE en wat is een Compiler?](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#WatiseenIDEenwatiseenCompiler)
  + [Vertalen ...](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#Vertalen)
    - [Interpreter - Vertalen tijdens gebruik](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#InterpreterVertalentijdensgebruik)
    - [Compiler - Vertalen voor gebruik](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#CompilerVertalenvoorgebruik)
* **4**[Hoe zetten we de Arduino IDE op?](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#HoezettenwedeArduinoIDEop)
  + [Download en Installeer de IDE](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#DownloadenInstalleerdeIDE)
  + [Initiële Configuratie van de Arduino Software](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#InitileConfiguratievandeArduinoSoftware)
    - [Windows gebruikers](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#Windowsgebruikers)
    - [Linux gebruikers](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#Linuxgebruikers)
    - [Mac OS X users:](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#MacOSXusers)
* **5**[Testen van onze Arduino](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/#TestenvanonzeArduino)

  Een volledig overzicht van de cursus vindt je hier: [**Cursus Overzicht**](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/overzicht/).

Wat is een Arduino?

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

Een [**Micro-Controller**](https://nl.wikipedia.org/wiki/Microcontroller) is in principe een simpele maar relatief complete mini computer (niet te verwarren met “[mini computer](https://nl.wikipedia.org/wiki/Minicomputer)“). Het is stukken minder krachtig in vergelijk met een desktop of laptop computer 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](https://nl.wikipedia.org/wiki/Atmel_AVR" \t "_blank). Atmel is de fabrikant, en AVR is de model serie. In principe is een Atmel AVR is een processor, met geheugen, een sorrt van besturingssysteem, en leuke aansluitingen om allerlei zaken aan te sturen zoals lampjes, schakelaars, motortjes, relays, 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.

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2016/03/Atmel-AVR-atmega328p.jpg)

Voorbeeld van een Atmel AVR chip

Omdat aansluiten zo lastig is, heeft de organisatie “[**Arduino**](https://www.arduino.cc/)” 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.

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2016/03/Arduino-Uno-R3.jpg)

Arduino Uno R3

Zoals je in de afbeelding kunt zien is het Arduino printje voorzien van de eerder genoemde 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.

Welk model voor Beginnen met Arduino Programmeren?

Ik heb zelf gekozen 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](http://www.amazon.com/s/ref=as_li_ss_tl?_encoding=UTF8&camp=1789&creative=390957&field-keywords=arduino%20uno&linkCode=ur2&tag=tweaki-20&url=search-alias%3Daps&linkId=D4XXKN3XMHRQLH6P) of [eBay](http://www.ebay.com/sch/i.html?_from=R40&_trksid=p2050601.m570.l1313.TR12.TRC2.A0.H0.Xarduino+uno.TRS0&_nkw=arduino+uno&_sacat=0), en legio andere online winkels.

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

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2014/01/usb_kabel_type_a_en_type_b.jpg)

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

Nogmaals, onze Aziatische vrienden maken goedkopere clones wat leuk is, maar soms maken ze er ook wel een potje van en kan het dus zijn dat de imitatie Arduino onverwachte problemen veroorzaakt. **Voor beginners raad ik zeer sterk aan om een originele Arduino te kopen!**

*We adviseren een****originele****Arduino Uno R3 voor beginners!*

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

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2016/03/Arduino-Uno-R3-front-and-back.jpg)

Arduino Uno R3 – Voorkant en achterkant

De Uno R3 komt in twee varianten. De gewone (links) en de zogenaamde “[SMD](https://nl.wikipedia.org/wiki/Surface-mount_technology)” 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 zo maar 1-2-3 eruit kunt trekken en vervangen.

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

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2014/01/arduino_regular_vs_smd.jpg)

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).

Wat is een IDE en wat is een Compiler?

Een IDE is een geïntegreerde ontwikkel omgeving of wel een “**[I](https://nl.wikipedia.org/wiki/Integrated_development_environment" \t "_blank)**[ntegrated](https://nl.wikipedia.org/wiki/Integrated_development_environment" \t "_blank) **[D](https://nl.wikipedia.org/wiki/Integrated_development_environment" \t "_blank)**[evelopment](https://nl.wikipedia.org/wiki/Integrated_development_environment" \t "_blank) **[E](https://nl.wikipedia.org/wiki/Integrated_development_environment" \t "_blank)**[nvironment](https://nl.wikipedia.org/wiki/Integrated_development_environment" \t "_blank)“, 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**](https://www.arduino.cc/en/Main/Software).

www.arduino.com

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”](https://nl.wikipedia.org/wiki/C_(programmeertaal)). “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](https://nl.wikipedia.org/wiki/Interpreter" \t "_blank)**.

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**](https://nl.wikipedia.org/wiki/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****.*

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

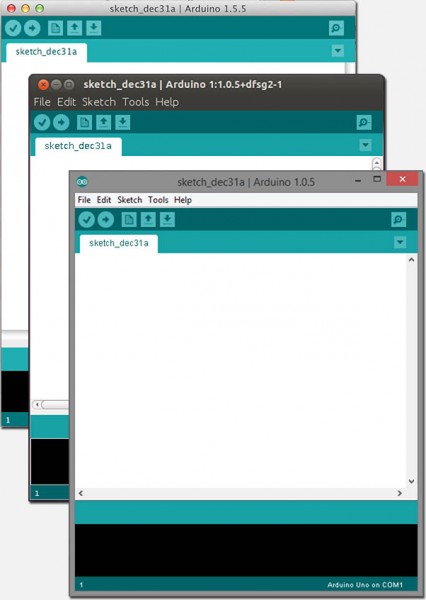
Installatie gaat in eerste instantie simpel – zoals eerder gezegd: download de versie die voor jouw computer geschikt is.

Windows gebruikers dubbel klikken 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 method.

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.

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2014/01/arduino_mac_linux_windows.jpg)

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.

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](https://www.arduino.cc/en/Tutorial/Blink) 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.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 | /\*   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.**

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2014/01/arduino_software_buttons.jpg)

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 |

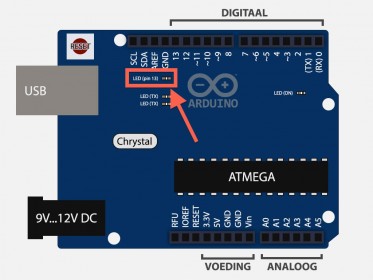
In de tabel zien je dat knop “**A**” gebruikt kan worden om jouw code te verifiëren op fouten.  
Knop “**B**” doet dit ook, maar vertaald het daarna ook nog eens (compileren) en als alles goed ging, stuurt deze de vertaling (het programma) naar de Arduino.

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)**](https://www.arduino.cc/en/Tutorial/Sketch)” een paar keer hebben gebruikt.  
Dit is de jargon (lingo) die Arduino gebruikers gebruiken om [**source code**](https://nl.wikipedia.org/wiki/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**](https://nl.wikipedia.org/wiki/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 …

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2014/01/arduino_ledje13.jpg)

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 … https://www.tweaking4all.nl/wp-content/themes/tweaking4all/css/images/emoticons/t4a_smile.png

[Arduino Programmeren voor Beginners – Deel 2: Uitvoer](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/)



* + [Arduino Programmeren voor Beginners](https://www.tweaking4all.nl/categorie/hardware/arduino/arduino-programmeer-cursus/)

[8](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#comments)

Dit is het tweede deel in een serie artikelen die ik heb geschreven voor mijn neefje Bram, welke graag zo willen leren programmeren omdat hij een robot wil gaan bouwen.

In dit deel gaan we de Arduino IDE, die we in deel 1 hebben geïnstalleerd, een beetje verkennen en gaan we kijken hoe we uitvoer (output) van de Arduino zichtbaar kunnen maken op onze computer. Dat laatste is natuurlijk handig als we willen weten of ons programma naar wens werkt, of om tussen door te kijken wat het programma doet. In tegenstelling tot een gewone computer, heeft de Arduino namelijk geen beeldscherm …

Deze reeks richt 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 electronica componenten blijft beperkt tot een minimum en bewaren we voor een volgende reeks.

[50](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/)Share

Overzicht van dit Hoofdstuk

* **1**[Een korte verkenning van de Arduino IDE](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#EenkorteverkenningvandeArduinoIDE)
* **2**[Communicatie met de Arduino ...](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#CommunicatiemetdeArduino)
  + [Wat USB te bieden heeft bij Arduino Programmeren](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#WatUSBtebiedenheeftbijArduinoProgrammeren)
  + [Seriële Communicatie over USB](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#SerileCommunicatieoverUSB)
  + [Seriële Monitor - Gegevens van de Arduino ontvangen](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#SerileMonitorGegevensvandeArduinoontvangen)
* **3**[De Basis Arduino Programma Lay-out](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#DeBasisArduinoProgrammaLayout)
  + [Setup() en Loop()](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#SetupenLoop)
  + [Arduino Programmeren met Opmerkingen](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#ArduinoProgrammerenmetOpmerkingen)
  + [Statements en Code blokken](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-2/#StatementsenCodeblokken)

  Een volledig overzicht van de cursus vindt je hier: [**Cursus Overzicht**](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/overzicht/).

Een korte verkenning van de Arduino IDE

Ik ga er vanuit dat je [Deel 1](https://www.tweaking4all.nl/hardware/arduino/arduino-programmeer-cursus/arduino-programmeren-deel-1/) netjes gevolgd hebt, en dus de Arduino met een USB kabeltje aan de PC hebt hangen, en de Arduino IDE (software) hebt draaien. Uiteraard ga ik er dus ook van uit dat je het test programma, het knipperende [LEDje](https://nl.wikipedia.org/wiki/Led" \t "_blank) (lampje), ook inderdaad aan de gang hebt gekregen.

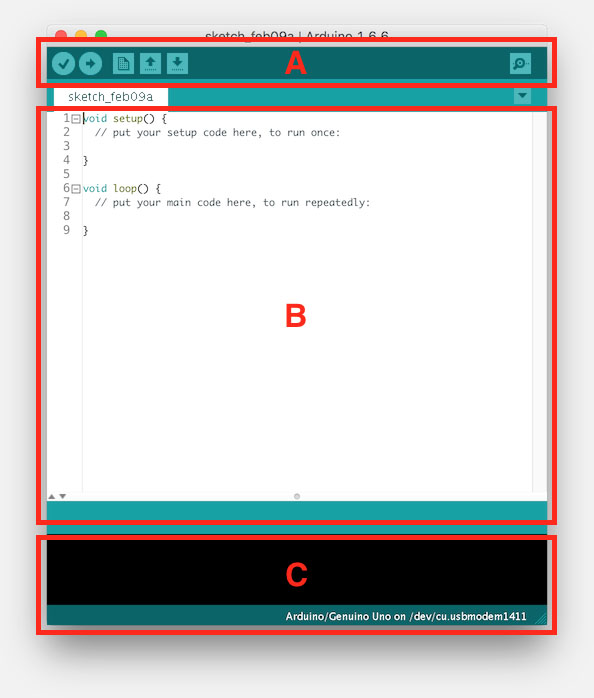
Omdat de Arduino IDE het belangrijkste stukje gereedschap is voor het Arduino Programmeren, gaan we hier het programma een beetje beter bekijken.

Laten we eerst even de Arduino IDE openen. Als dit niet de eerste keer is dat je de Arduino IDE start, klik dan op de knop “**C**“, zoals in onderstaande afbeelding weergegeven, om een nieuw “project” te starten:

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2014/01/arduino_software_buttons.jpg)

Arduino Software – Handige snelkoppelingen

Na het klikken op de “C” knop, of het voor het eerst openen van de Arduino IDE, zul je een venster zien zoals hieronder:

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2016/03/arduino-ide-sections.jpg)

Arduino IDE – Belangrijke secties

We zien 3 “belangrijke” secties (zie afbeelding 2, hierboven):

In sectie “**A**” zien we de **knoppen** die we eerder besproken hebben (zie afbeelding 1) – maar we zien nu ook nog helemaal rechts een extra knop die op een vergrootglas lijkt. Dit is de “**Serieel Monitor**” knop, welke soms ook weleens “Debug Window” knop genoemd wordt.

[Arduino IDE - Serieel Monitor Knop](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2016/03/arduino-ide-serial-monitor-button.jpg)

Arduino IDE – Serieel Monitor Knop

Sectie “**B**” is waar we onze [**Source Code (Broncode)**](https://nl.wikipedia.org/wiki/Broncode) kunnen  bewerken. Het is eigenlijk een simpele tekstverwerker met zogenaamde “code highlighting”. *Code highlighting* zorgt ervoor dat bepaalde trefwoorden, tekens of character reeksen een bepaalde kleur krijgen, wat ons beter helpt bij het lezen van onze broncode.

Sectie “**C**” is waar the area de Arduino IDE ons belangrijke **meldingen** zal geven. Bijvoorbeeld als er iets fout gaat als we een onze broncode compileren, of als er iets anders fout of goed gaat. In het begin zullen de meldingen hier wel eens knap onduidelijk zijn, maar naarmate we meer leren en ervaring opdoen, worden ze gemakkelijker om te volgen.

De Arduino IDE heeft natuurlijk nog veel meer functies, die we zelden gebruiken maar toch handig zijn. Je vind ze in de menu’s en we hebben er al 2 gezien bij het aansluiten van de Arduino – toen we communicatie poort en type Arduino moesten instellen.

Naarmate we meer werken met de Arduino, zul je merken wat een aantal van die functies voor ons kunnen doen, aar in het begin zijn ze niet zo van belang.

Communicatie met de Arduino …

Zoals eerder gezegd en gezien: de Arduino heeft geen beeldscherm, toetsenbord of muis. Dat maakt communicatie tussen ons mensen en de Arduino wat lastig en aan ee knipperend lampje hebben we niet al te veel. Dus hoe kunnen we zien (uitvoer!) wat de Arduino nou aan het sjouwen is?

Dit is nu waar de “**Serieel Monitor**” van pas gaat komen!

Wat USB te bieden heeft bij Arduino Programmeren

Laten we even kort kijken wat de USB aansluiting te bieden heeft als we met een Arduino aan de slag gaan.

**1. USB voorziet de Arduino van stroom.**

Zoals met de meeste electronica, heeft de Arduino natuurlijk stroom nodig om te werken.  
Onder normale omstandigheden biedt een USB aansluiting 5V, wat genoeg is voor de Arduino om te werken.  
Mochten we geen USB aansluiting gebruiken dan zullen we een adaptertje moeten gebruiken om de Arduino van stroom te voorzien (ronde connector bij de meeste Arduino modellen).

**2. USB als middel om programma’s naar de Arduino te sturen**

We schrijven ons programma op een computer, en dat programma (na vertaling) moet dus naar de Arduino overgezet worden – het zogenaamde “uploaden”. Dit gebeurt ook over de USB verbinding.

**3. USB can Receive data from the Arduino**

De USB verbinding kan echter ook gebruikt worden om gegevens (data) van de Arduino naar de PC terug te sturen, iets wat we gaan gebruiken als vervanger van het ontbrekende beeldscherm. Omdat we veel van de programmeer taal gaan uitproberen, gaat dit erg handig zijn.

Seriële Communicatie over USB

Zoals je ziet: de belangrijkste taak van de USB verbinding is communicatie.

Nu moet je weten dat vroeger een [Micro-Controller](https://nl.wikipedia.org/wiki/Microcontroller), zoals we die ook op de Arduino hebben, communiceerde via een zogenaamde seriële poort. Dat doen ze nog steeds, maar de handige mannen van het Arduino team hebben een “omzetter” op de meeste Arduino modellen gezet die dit omzet naar een USB aansluiting. Een dergelijke seriële poort werd ook wel “[**com poort**](https://nl.wikipedia.org/wiki/Seri%C3%ABle_poort)” of “[**Serieel poort**](https://nl.wikipedia.org/wiki/Seri%C3%ABle_poort)” genoemd.

Een seriële-port, of com-poort, verstuurd gegevens (data) met een bit per keer – achter elkaar dus.  
Een [**bit**](https://en.wikipedia.org/wiki/Bit) is iets wat èèn (1) of een nul (0) zijn.

Aan een enkele bit hebben we niet zo erg veel – het kan alleen maar 2 standen aangeven. Bijvoorbeeld AAN/UIT, JA/NEE, WAAR/NIET-WAAR (dit heet overigens “boolean”, iets wat we later bespreken). Slimme mensen hebben daarom de “[**byte**](https://nl.wikipedia.org/wiki/Byte)” bedacht.  
Een “byte” combineert 8 bits in een groep, welke 256 verschillende waarden kan aannemen, waardoor het ineens wel zinvol wordt. We kunnen met 256 waarden bijvoorbeeld het hele alfabet (kleine letters en hoofd letters), nummers, en tekens omvatten.

Dus een bit kan 1 van 2 waarden aannemen, een nul (0) of een èèn (1).  
Een byte heeft 8 bits, wat wiskundig wil zeggen **28** = 2 tot de macht 8 = 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 = 256 waarden. Mocht je op school nog niet gehoord hebben van “tot de macht”, geen probleem … ik probeer het in een van de volgende delen een beetje uit te leggen, maar voor nu is het belangrijk dat je weet dat het 256 waarden kan hebben; van 0 … 255.

*“****Tot de macht****” geeft aan hoeveel keer een nummer gebruikt wordt voor vermenigvuldiging.*

*In andere woorden:****Hoe veel keer je het nummer “1” met dit nummer moet vermenigvuldigen.***

Een seriele poort “praat” met een bepaalde snelheid, dit noemt men vaak de zogenaamde **[baudrate](https://nl.wikipedia.org/wiki/Baud_(telecommunicatie)" \t "_blank)**. De baudrate geeft aan hoe snel bits verstuurd en/of ontvangen worden, dus eigenlijk “bits per seconde”. Er zijn een aantal standaard snelheden zoals bijvoorbeeld 9600, wat wil zeggen dat gegevens (data) met een snelheid van 9600 bits per second verstuurd worden, wat omgezet naar bytes wil zeggen 1200 bytes (of karakters) per seconde. Vergeet niet dat een byte uit 8 bits bestaat, dus 9600 : 8 = 1200.

Een kleine opmerking: om helemaal correct te zijn is de berekening van snelheid in bytes een beetje complexer dan wat ik hier net aangaf. Dit heeft o.a. te maken met controle of alles goed is aangekomen, indicatie wanneer een byte “klaar is”, etc. Maar … daar hoeven we ons nog niet zo druk over te maken.

Uiteraard moet men de snelheid aan beide kanten hetzelfde hebben – dus zowel de computer als de Arduino moeten met dezelfde snelheid praten anders snappen ze elkaar niet.

Een aantal standaard baudrate waarden zijn:

| *Standard Baudrates* |
| --- |
| **Baudrate** |
| 110 |
| 300 |
| 600 |
| 1200 |
| 2400 |
| 4800 |
| **9600** |
| 14400 |
| 19200 |
| 38400 |
| 56000 |
| 115200 |

Er zijn nog meer standaard waarden, en deze gaan vaak hoger dan 115200. Helaas wil het zo zijn dat niet iedere computer (met name Windows) daar altijd even goed in is, waar een Mac of een Linux computer heer altijd wel goed mee overweg kan.

OK, even genoeg geschiedenis les voor vandaag ….

Omdat de meeste computers tegenwoordig niet meer geleverd worden met een antieke seriële-poort of com-poort, gebruikt de Arduino een **USB-naar-Serieel converter**, welke op de meeste Arduino’s te vinden zijn. Deze worden dan door de computer als een seriële-poort gezien.

We weten dus nu dat via de USB verbinding ons programma naar Arduino gestuurd kan worden, tijd om eens te kijken hoe de Arduino nu gegevens terug kan sturen naar de computer.

Seriële Monitor – Gegevens van de Arduino ontvangen

Als we een programma starten, dan willen we natuurlijk graag zien wat het programma doet. Al is het maar voor onze voorbeelden of voor het zoeken naar fouten in ons programma – dat zoeken van fouten heet “**[debugging](https://nl.wikipedia.org/wiki/Debuggen" \t "_blank)**“.

***Debugging*** *is het proces waarbij men fouten (“*[***bugs***](https://en.wikipedia.org/wiki/Software_bug)*“) probeert te vinden en probeert te verhelpen in programma’s.*

Hier is de “**Seriële Monitor**” erg handig …

Tip: Herstart een Arduino Programma met de Seriële Monitor

Elke keer als je de Seriële Monitor opent, krijgt de Arduino een “herstart” (reset) signaal waardoor het opnieuw opstart alsof je de Arduino net hebt aangezet, zodat het programma netjes schoon opstart.

Je kunt dit dus gebruiken als truc om de Arduino opnieuw op te starten: Sluit de Seriële Monitor en open de Seriële monitor weer opnieuw en de Arduino start weer opnieuw.

Om er nu voor te zorgen dat de Arduino ons informatie stuurt, moeten er dus voor zorgen dat zowel computer als Arduino op de zelfde snelheid babbelen. We moeten zowel de Seriele Monitor als de Arduino dus vertellen met welk tempo we gaan communiceren.

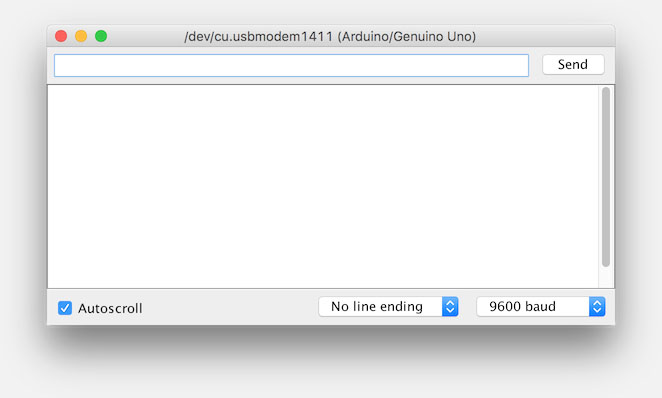
Voor de Arduino wil dat zeggen dat we in ons programma moeten opgeven welke baudrate we willen gaan gebruiken. Meestal is de Arduino IDE slim genoeg om dat vanzelf te ontdekken, maar soms moeten we het handmatig even doorgeven.

Laten we er even een voorbeeld bij gaan halen. Uiteraard moeten we hiervoor de **Arduino IDE open hebben**, en de **Arduino verbonden** met de computer, en natuurlijk op “**Seriële Monitor**” knop (het vergrootglas-knopje rechts boven in de Arduino IDE).

[Arduino IDE - Serieel Monitor Knop](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2016/03/arduino-ide-serial-monitor-button.jpg)

Arduino IDE – Serieel Monitor Knop

Nadat je op de “Seriële Monitor” knop hebt gedrukt, krijg je of een foutmelding in de berichten sectie (Arduino niet correct aangesloten?) of een venstertje zoals hieronder:

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2016/03/arduino-ide-serial-monitor.jpg)

Arduino Seriële Monitor venster

In dit venster kunnen we gegevens (data) ontvangen van de Arduino, maar (later meer hierover) ook naar Arduino toe sturen.

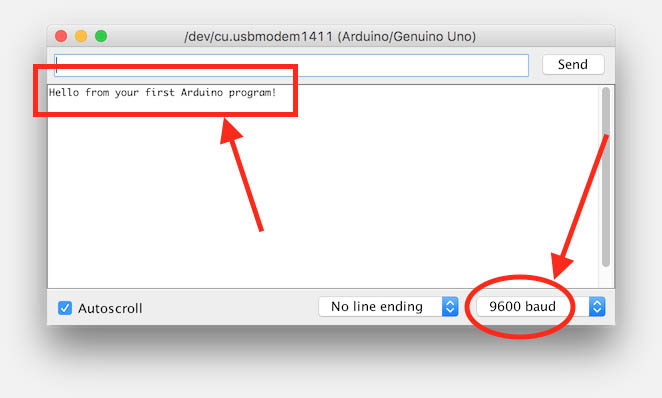
Laten we even ons eerste “praat” programma bekijken …  
In de onderstaande code heb ik 2 regels aan de standaard code toegevoegd: regels 3 en 4.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | void setup() {   // put your setup code here, to run once:   Serial.begin(9600);   Serial.println("Hello from your first Arduino program!"); }  void loop() {   // put your main code here, to run repeatedly:  } |

De regel “Serial.begin(9600); ” vertelt de Arduino dat het over de seriële verbinding moet praten met een snelheid van 9600 baud.  
Als je nu in de volgende afbeelding kijkt, dan zie je “9600 baud” rechts onderin – dezelfde snelheid als wat we in het programma opgeven.

De daarop volgende regel “Serial.println("Hello from your first Arduino program!"); ” (Engels voor: Hallo van jouw eerste Arduino Programma) vertelt de Arduino dat het tekst moet versturen over de seriële verbinding.

OK, kopieer nu de code en plak het in de Arduino IDE, waarbij alle bestaande tekst vervangen wordt.  
Klik vervolgens op de “Compileer en Upload” knop (knop “B” in afbeelding 1).  
De Arduino zal weer wat knipperen met de LEDjes, als dat afgelopen is dan zul je het volgende in het venster van de Seriële Monitor zien:

[](https://www.tweaking4all.nl/wp-content/uploads/sites/2/2016/03/first-arduino-output.jpg)

Onze eerste Arduino Output

Ziet er toch best eenvoudig uit hè? Als je het eenmaal gezien hebt althans …

De Basis Arduino Programma Lay-out

Setup() en Loop()

Ieder Arduino programma heeft dezelfde basis lay-out en dat zie je meteen als je een nieuw project start (klik op de “C” knop in afbeelding 1).

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 | void setup() {   // put your setup code here, to run once: }  void loop() {   // put your main code here, to run repeatedly:  } |

Hier zien we twee basis elementen: “**setup()**” en “**loop()**“, en beiden, ook al laat je ze leeg, moeten bestaan!

Als de Arduino namelijk start, dan zal het eerst naar “setup()” gaan zoeken en deze slechts èèn keer doorlopen.  
Daarna zal de Arduino “loop()” uitvoeren en deze “loop()” oneindig blijven herhalen. Vandaar dus de naam “loop” (lus).

De Arduino makers wisten dat natuurlijk al, dus vandaar dat de Arduino IDE deze code al automatisch gaat aanmaken.

*Elk Arduino programma heeft een “setup()” functie welke tijdens het opstarten slechts èèn maal de startup() doorloopt, en vervolgens de “loop()” functie eindeloos blijft herhalen tot de Arduino uitgezet wordt.*

Om dit te illustreren zouden we de code die de tekst uitvoert naar de seriële verbinding, in de “loop()” kunnen zetten in plaats van in de “setup()”. Het gevolg zou zijn dat het venster van de Seriële Monitor herhaaldelijk dezelfde tekst blijft weergeven tot we zeg maar de stroom van de Arduino af halen (USB kabel er uit trekken).

We laten “Serial.begin(9600); ” in de “setup()” staan, want instellen van de snelheid hoeft natuurlijk maar 1 keer.  
De regel “Serial.println("Hello from your first Arduino program!"); ” verplaatsen we nu naar de “loop()” wat dus als gevolg heeft dat die zich blijft herhalen.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | void setup() {   // put your setup code here, to run once:   Serial.begin(9600); }  void loop() {   // put your main code here, to run repeatedly:   Serial.println("Hello from your first Arduino program!"); } |

Arduino Programmeren met Opmerkingen

Het toevoegen van opmerkingen in de code van ons programma is zeker sterk aan te raden.  
Als eerst maken de opmerkingen onze code aanzielijk duidelijker, wat handig is als we op een later tijdstip iets willen veranderen in onze code.  
Nog belangrijker is dat opmerkingen uitleg bieden aan anderen die misschien jouw code willen bewerken of aanpassen..

***Plaats altijd Opmerkingen (uitleg) in jouw code !***

In een aantal van de eerdere code voorbeelden heb je al wat opmerkingen in de code zien staan, het zijn de regels die met 2 schuine strepen, ook wel [slashes](https://nl.wikipedia.org/wiki/Schuine_streep" \l "Computers" \t "_blank) (“**//**“) genoemd, beginnen.

Maar er zijn meer opties om opmerkingen te plaatsen.

Als eerste weten we dat opmerkingen beginnen met 2 schuine strepen “**//**“, wat wil zeggen dat alles achter deze tekens (in deze regel) als commentaar of opemerking gezien moet worden. Dit wil dus ook zeggen dat de vertaler (de compiler) alles na de 2 schuine strepen negeert op betreffende regel.  
Dit wil ook zeggen dat een opmerking achter commando’s kan staan.

***Opmerkingen maken een programma niet groter of trager****,*

*Opmerkingen maken alleen de Source Code groter, maar ook meteen aanzienlijk beter leesbaar en duidelijker.*

Opmerkingen kunnen ook tussen “**/\***” en “**\*/**” geplaatst worden en dat mag zelfs over meerdere regels. Een aantal voorbeelden:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | // Enkele regel opmerking  Serial.begin(9600);  // Opmerking na een commando  // De volgende regel schakelt tijdelijk een commando uit // Serial.begin(9600);  /\* Dit is een opmerking blok die slechts 1 regel beslaat \*/  /\* Maar een opmerking blok    kan ook meerdere    regels beslaan      \*/  /\* Met een opmerking blok kun je ook meerdere regels met commando's tijdelijk uitschakelen Serial.begin(9600); Serial.println("Hello from your first Arduino program!"); \*/ |

Hier zien we dus opmerking als enkele regel, aan het einde van een regel, of meerdere regels beslaan.

Wat we ook zien is dat het ook handig gebruikt kan worden om bepaalde stukken van onze code uit te schakelen – bijvoorbeeld als we wat aan het testen zijn tijdens de ontwikkeling van ons programma.

Statements en Code blokken

Misschien heb je het gemerkt, misschien niet, maar ik zat net even te klungelen met het juiste woord voor “commando’s” want een echt goede vertaling in het Nederlands voor “statement” kon ik zo 1-2-3 niet vinden. Elke regel kan namelijk leeg zijn, een opmerking of een instructie ([**statement**](https://nl.wikipedia.org/wiki/Statement)) bevatten. Een instructie, of statement, wordt altijd afgesloten met een punt-komma  ( **;**). Dit moet je goed onthouden want dit kan een van de meest voorkomende type fouten zijn in een programma.

*Een instructie of****statement******eindigt altijd met een punt-komma****( ; ) …*

Het woord “instructie” is beter Nederlands, maar in de computer wereld gebruikt men vaak de Engelse taal en zeggen we dus vaker “statement”.

Code blokken is weer zo’n leuke term en die hebben we al een paar keer gezien. In eerder voorbeelden zagen we bijvoorbeeld 2 van dit soort “code  blokken” ingesloten met accolades.

|  |  |
| --- | --- |
| 1 2 3 4 | void setup() {   // put your setup code here, to run once:   Serial.begin(9600); } |

Als we dit voorbeeld eens goed gaan bekijken dan zien we in de eerste regel **“void setup() {**“.  
Dit wil zeggen dat we hier een functie definiëren met de naam “setup”, welke geen waarde terug stuurt (void) en geen parameters neemt (de twee ronde haakjes) – dit gaat al verder dan wat we behandeld hebben en in het hoofdstuk “functies” gaan we hier dieper op in. Voor nu onthouden we:  
“*Hier start de definitie van de functie setup, welke geen parameters gebruikt en niets terug geeft*“.

De twee ronde haakjes **()**, geven aan wat voor informatie we naar de functie willen sturen – en omdat er niets staat: niks dus!  
De term “**[void](https://en.wikipedia.org/wiki/Void_type" \t "_blank)**” is zoiets als luchtledig, niks, noppes, nada, niets, … en omdat het voor de functie naam “setup” staat wil dit zeggen dat de functie geen informatie terug zal geven. Je hoeft dit nog niet te onthouden – zoals gezegd; we behandelen de details later als we naar functies gaan kijken.

*Een code blok start en eindigt met accolades .(* ***{*** *…* ***}*** *)..*

De accolade open, start de zogenaamde “code blok”. Alles wat hierna geschreven wordt, tot het punt waar we accolade sluiten tegen komen, wordt als 1 enkele blok gezien. Dus in het voorbeeld de code die bij de functie setup hoort “setup”. De opmerking en de regel met “Serial.begin(9600);” horen dus bij elkaar, in een blok, welke bij de functie “setup” hoort.

Nu hebben we wat meer basis informatie, dus tijd om eens te gaan kijken naar wat interessantere programmeer zaken ….

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!