1.0. Arduinoprojecten

Project 1 – Aan de slag met jouw eerste Arduino project

Jullie heb kort geleden een Arduino UNO (setje)aangeschaft via de school compleet met opbergdoos, Breadboard, jumper draadjes en wat componenten om mee te beginnen.

Om jezelf bekent te maken met de Arduino, loop je eerst door de stappen heen van het simpelste project: het aan- en uitschakelen van een LEDje.
Het goede van dit eerste projectje is dat je jezelf bekent maakt met de basis stappen voor ieder Arduino project zoals het aansluiten op de computer, het gebruik van de arduino software en een paar basis stappen in het programmeren (C/C++ gebaseerd).

Inhoudsopgave

1.1. Jouw Eerste Arduino Project – Wat hebben we nodig?

Voor we met ons eerst Arduino project aan de slag kunnen, hebben we het volgende nodig:

- **Arduino** (UNO is een aanrader)
- **USB kabeltjes**
- **Arduino software**

Arduino Website is erg informatief!

Merk op dat de Arduino website geeft geweldig goede info heeft zoals programmeertaal referenties en studiematerialen.
Ik kan sterk aanrader daar eens rond te snuffelen – het is echt de moeite waard.

1.2. Arduino (UNO)

Dit project heeft niet veel hardware of software eisen en zou in principe op de meeste standaard Arduino boards moeten werken, maar ik kan de Arduino UNO (r3 is de huidige versie) sterk aanraden. De Uno kun je als standaardversie (DIL microcontroller) or als SMD versie kopen.

Ik heb de voorkeur om een board van het Arduino merk te kopen (made in Italy) om ze te ondersteunen in hun pogingen Arduino verder te ontwikkelen. Maar je kunt natuurlijk ook een goedkope Chinese variant kopen welke vaak net zo goed werken en goedkoper zijn – maar soms laat de bouw kwaliteit wat te wensen over. De Arduino in de set is een Chinese cloon.

Het voordeel van de standaard (niet-SMD) versie is dat je de microcontroller van het board kunt verwijderen en in een uiteindelijke schakeling kunt plaatsen. Of als je per ongeluk de microcontroller opblaast, dan kun je deze eenvoudig vervangen. De SMD-versie biedt deze opties niet.

Figuur 1- 1

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

Beide versies werken verder 100% identiek, maar zoals je in bovenstaande afbeelding kunt zien, is de microcontroller (een standaard Atmel ATMEGA328) in de SMD-versie (rechts) stukken kleiner dan in de standaard (DIL) versie.

Plaatsen van het Arduino board

Wanneer je aan de slag gaat met de Arduino, zeker als je er spanning op zet via de stroom stekker of via de USB-aansluiting, is het zaak dat de Arduino **op materiaal geplaatst staat welke niet stroom geleid**. Ik probeer zelf ook altijd om stroom geleidende objecten, zoals objecten van metaal en vloeistoffen, ver van de Arduino te houden om ongewenste kortsluitingen te veroorzaken.

1.3. USB Kabeltje

Voor een Arduino Uno hebben we een standaard USB kabel nodig met een type A-connector (mannetje) aan de ene kant en een type B-connector (mannetje) aan de andere kant. Dit is een typisch kabeltje wat gebruikt wordt om USB-printers of scanner aan te sluiten op de PC en sommige Arduino setjes komen met het geschikte USB kabeltje.

Figuur 1- 2

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

1.4. Arduino Software

De benodigde Arduino software kun je gratis downloaden van de Arduino Website en is beschikbaar voor Windows, MacOS X en Linux (32- en 64-bit versies).
Op het moment dat ik dit artikel schrijf is **versie 1.0.5** de meest gebruikte versie en je kunt deze ook van Tweaking4All downloaden.

Merk op dat Linux gebruikers ook apt-get kunnen gebruiken om de Arduino software te installeren:

```
sudo apt-get update
sudo apt-get install arduino arduino-core
```

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

Na download en installatie van de software, start de Arduino applicatie – onder elke besturingssysteem ziet het er een klein beetje anders uit, maar de verschillen zijn minimaal (complimenten aan het Arduino team!). Uiteraard zijn de menu balken op de standaard plaatsen te vinden: voor MacOS X boven in het scherm en voor Windows en Linux boven in het venster.

Verbind nu de Arduino via de USB-kabel met jouw computer!

Merk op : Gedurende de software installatie worden de benodigde USB drivers geinstalleerd, welke meteen werken voor Linux en MacOS X – Windows heeft echter een beetje hulp nodig om de drivers aan de gang te krijgen (meer hierover later).

Figuur 1- 3

Arduino onder MacOS X, Linux en Windows

1.5. Drivers voor Windows gebruikers

Helaas werken de drivers onder Windows niet meteen (en moet je ze eenmalig installeren) – automatische installatie blijkt niet goed te werken, wat je ook probeert.

Windows XP: (voor de volledigheid in dit overzicht)
Ga naar de "Device manager" (klik rechts op "My Computer" of ga direct naar de Control Panel) en zoek naar de "Other devices" "Unknown Device" en klik deze met rechts aan en kies "Update Driver".
In het volgende venster, selecteer "Install from a list or specific location" en klik op "Next".
In het volgende venster **verwijder het vinkje voor** "Search removable media" **en vink** "Include this location in the search" **aan**.

Klik vervolgens op “Browse” en ha naar “C:\Program Files\Arduino\drivers“, klik “OK“, en klik “Next“.

Windows 7 en nieuwer:
Ga naar “Control Panel” “Device Manager” en kijk onder “Ports” of “Unkown Devices” en vindt het device met het gele uitroepteken. Klik het met rechts en kies “Update driver software“. In het nieuwe venster klik je nu “Browse my computer for driver software” en in het volgende venster klik je “Browse” en ga je naar “C:\Program Files (x86)\Arduino\drivers” (de ” (x86)” alleen voor 64 bit systemen) en klik “Next“.

Windows als Virtual Machine:
Ik schrijf mijn artikelen vaak op een Mac en draai Windows in een virtuele machine (Parallels of VMWare). Tijdens mijn experimenten in deze virtuele machines kwam ik al snel tot de conclusie (en Meneer Google bevestigde dat) dat de drivers vaak niet of niet correct werken. Onder normale omstandigheden geen probleem natuurlijk omdat de Arduino software beschikbaar is voor de drie meest gangbare besturingssystemen.

1.6. Initiële Configuratie van de Arduino Software

Na installatie (en driver installatie voor Windows gebruikers), is het tijd om initiële instellingen te gaan doen voor de Arduino software.
(vergeet niet de Arduino op de computer aan te sluiten!)

Windows gebruikers:

Als eerste moeten we de verbinding instellen.
Open het menu “Tools” “Serial Port” en kies de juiste “Com” poort (USB apparaat), dit hangt natuurlijk sterk af van jouw computer en de geschikte poort **kan** dus b.v. COM4 zijn. Omdat de meeste moderne computers geen COM poorten meer hebben, kan dit echter net zo goed COM1 zijn.

Vervolgens moeten we opgeven welk Arduino board we gebruiken.
Ga naar het menu “Tools” “Board” en selecteer jouw Arduino board, in mijn geval de “Arduino Uno“.

Linux gebuikers:

We moeten eerst de verbinding instellen.
Open het menu “Tools” “Serial Port” en selecteer de USB verbinding naar de Arduino (onder Ubuntu had ik `/dev/ttyMCA0` , maar ook dit is weer verschillend per computer).

Hierna moet je het Arduino board kiezen.
Ga naar het menu “Tools” “Board” en kies jouw Arduino board, wat in mijn geval de “Arduino Uno” was.

MacOS X gebruikers:

Mac's hebben geen com-poorten, dus net als onder Linux moeten we hier het juiste device voor de verbinding kiezen.
In het menu "Tools" "Port" kies je het juiste USB device. Zoek naar zoiets als (als je een Uno hebt): `/dev/cu.usbmodem1421 (Arduino Uno)` en selecteer het.

Ook onder MacOS X moeten we het board type kiezen in het "Tools" "Board" menu, in in mijn geval was dat dus de "Arduino Uno".

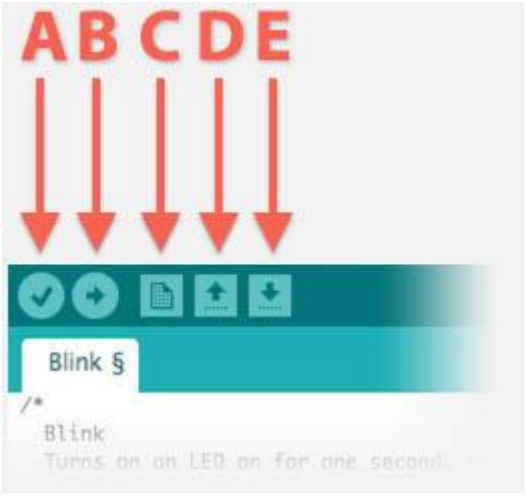
Nu dat we dat allemaal hebben ingesteld, tijd voor ons eerste programma …

1.7. Mijn Eerste Arduino Project

Voor we beginnen: voor de Arduino noemt men een "programma" een **Sketch** en een Sketch is een programma geschreven in C/C++. Ik had hier liever een taal zoals Pascal gezien, maar ik vrees dat velen deze mening niet zullen delen,… waarschijnlijk door een gebrek aan Pascal kennis, maar laten we vooral niet die discussie gaan beginnen …

Snelkoppelingen …

Een paar snelkoppeling die handig zijn, zie je hieronder.



Figuur 1- 4

Arduino Software – Handige snelkoppelingen

Handige Arduino Snelkoppelingen

Snelkoppeling	Doel
A	Verify – Code verificatie (of: compileer code)
B	Upload – Compileer en Upload het programma naar de Arduino
C	New – Maak een nieuwe Sketch
D	Open – Open een bestaande Sketch
E	Save – Sketch opslaan

Na het invoeren of wijzigen van code, kun je de "Verify" (**A**) knop gebruiken om fouten te vinden. In principe compileert deze functie jouw code en laat je eventuele fouten zien. Om het programma op de Arduino te draaien, moet de gecompileerde code naar de Arduino gestuurd worden. Hiervoor gebruiken we de "Upload" knop (**B**) welke de code compileert en als alles goed ging verstuurd naar de Arduino. Je kunt dat dus zien als op de "Verify" knop drukken en vervolgens het uploaden naar de Arduino als een enkele klik.

1.8. Basis onderdelen van een Sketch

In elke Sketch kunnen we een aantal basis onderdelen vinden …

Opmerkingen (Optioneel)

Het is een goede gewoonte om opmerkingen in jouw code te zetten. Het bespaard jou en anderen een hoop uitzoek werk als we de code willen lezen of veranderen.
Opmerkingen beginnen met twee schuine streepjes voorwaarts (//) en alles na deze twee karakters wordt als opmerking gezien.
Opmerkingen worden door de compiler genegeerd en maken het eind programma niet groter of kleiner, maar leesbaarheid wordt stukken beter.
Je kunt als alternatief een opemerking tussen '/*' en '*/' zetten – zelfs over meerdere regels.
Opmerking worden ook vaak gebruikt om snel even stukjes code uit te schakelen zonder dat je de code hoeft te verwijderen.

1	// Enkele regel opmerking
2	
3	int led = 13; // opmerking aan het einde van de regel
4	
5	// De volgende code doet niets omdat het een opmerking is
6	// int led = 14;
7	
8	/* Dit is een opmerking blok */
9	
10	/* Maar opmerking blokken
11	kunnen over meerdere
12	regels gaan */
13	
14	/* Opmerking blokken kunnen ook gebruikt worden om meerdere code regels tijdelijk uit te schakelen
15	int led = 13;
16	int MyVariable = 8;
17	*/

Hekje statements (optioneel)

Hash (Hekje, "#" of "Pound" zoals Amerikanen het noemen) statements zijn zogenaamde **compiler directives** (specifieke opdrachten voor de compiler) – welke we in ons eerste project niet gebruiken maar welke je in veel andere projecten tegen zult komen. De compiler is het programma welke jouw tekst (code) omzet in binair formaat zodat de microcontroller er mee overweg kan.

Merk op : Compiler directives hebben geen punt-komma (;) aan het einde van de regel, dit in tegenstelling tot normale C code.

We hebben compiler directives niet altijd nodig, maar dit zijn de twee meest gebruikte …

LIBRARY INCLUDES

```
#include <MijnFile.h>
```

Deze directive vertelt de compiler dat het een bestand met de naam “MijnFile.h” moet toevoegen aan jouw code. Dit is een zogenaamde C include bestand of library. Een Library of Include file bevatten vaak code welke vaak hergebruikt kan worden in andere Sketches. Vaak zijn libraries (bibliotheken) een verzameling samenhangende functies gericht op een bepaalde toepassing. Bijvoorbeeld om een strip met LEDjes aan te sturen, speciale wiskundige functies of bijvoorbeeld om een lcd-schermpje aan te sturen.

CONSTANTEN

```
#define MijnConstante 6
```

Deze directive definieert een constante met de naam “MijnConstante” als de waarde “6”, welke de compiler eigenlijk de opdracht geeft om elk voorkomen van “MijnConstante” te vervangen door het nummer “6”.
Het doel van dit soort constanten is dat jouw code veel beter leesbaar wordt (slechts een nummer zegt niet veel) en het zorgt ervoor dat als je dit nummer moet veranderen dat je dit maar op 1 plaats hoeft te doen i.p.v. door de hele code te zoeken waar je het allemaal moet veranderen. Constanten worden vaak in het begin van de code gedefinieerd.

1.9. Definitie van globale variabelen (optioneel)

Buiten de functies die we zo meteen gaan bespreken, kunnen we zogenaamde “globale variabelen” definiëren. Dat wil zeggen: variabelen die gelezen en veranderd kunnen worden op welke locatie in de Sketch dan ook. Zo’n variabele kan dus ook benaderd worden in de functies setup() en loop() – dit in tegenstelling to de eerder genoemde constanten (`#define`) tijdens de afloop van het programma NIET kunnen veranderen.

In onderstaand voorbeeld definiëren we de variabele “MijnVariable”, welke een geheel nummer is (int = Integer = geheel nummer), met een initiële waarde van 8.

```
int MijnVariable = 8;
```

1.10. Functie: Setup()

Deze functie is verplicht in iedere Sketch die je voor de Arduino schrijft – het is dat deel van het programma dat initiële initialisatie stappen uitvoert voor het werkelijk "programma" gaat draaien. Hier kun je bevoorbeeld definieren dat een bepaalde pin een OUTPUT (uitgang) is.
De term "*void* " voor de functie naam is een standaard C kreet die aangeeft dat er geen waarde terugverwacht wordt.

```
1  void setup() {
2    ...
3  }
```

1.11. Functie: Loop()

Ook deze functie is verplicht in iedere Sketch en is het deel dat blijft herhalen in de mircocontroller – in tegen stelling tot een PC, blijft een microcontroller het programma aflopen tot jet de stroom naar de microcontroller verwijdert.
In dit deel zul je zien dat we in ons eerste Arduino project een LEDje AAN en UIT schakelen.

```
1  void loop() {
2    ...
3  }
```

1.12. Onze eerste Sketch

Merk op : dit voorbeeld kan in de Arduino software gevonden worden onder "Examples".

In ons eerste Arduino projectje laten we een LEDje knipperen.
Simpel en misschien totaal zinloos, maar het doel is dat we bekend geraken met onze nieuwe Arduino en niet dat we de volgende Mars Rover gaan bouwen.

In ons voorbeeld kunnen we gebruik maken van een LED dat op de meeste Arduino boards gevonden kan worden: Het LEDje dat aan PIN 13 zit.
Je kunt er ook voor kiezen om een LED toe te voegen en dat laten we in de laatste stap zien.

De code:

```
1  /*
2    Blink
3    Schakelt een LED 1 seconde aan, en dan 1 seconde uit, en blijft dit herhalen.
4
5    Dit voorbeeld is in de public domain.
6   */
7
```

```
8  // Pin 13 heeft een LED op de meeste Arduino boards.
9  // Geef het een naam:
10 int led = 13;
11
12 // de setup routine draait 1x als we de Arduino aanzetten of op Reset drukken:
13 void setup() {
14   // initialiseer de pin als digitale uitgang.
15   pinMode(led, OUTPUT);
16 }
17
18 // de loop routine blijft zich onendig herhalen:
19 void loop() {
20   digitalWrite(led, HIGH);   // schakel LED AAN (HIGH is het voltage level)
21   delay(1000);               // wacht een seconde
22   digitalWrite(led, LOW);    // schakel LED UIT door voltage op LOW te zetten
23   delay(1000);               // wacht een seconde
24 }
```

Zoals je ziet, zowel **setup()** als **loop()** bestaan in deze Sketch.

Voor we een functie starten zien we de regel met `int led = 13;` welke een globale variabele definieert (zichtbaar in **setup()** en **loop()**) met de naam “led” en de initiële waard “13”. De waarde van deze variabele zou men later in setup() of loop() kunnen veranderen, maar in ons voorbeeld lezen we alleen maar de waarde uit en had een constante ook prima gewerkt (`#define led 13`).

In **setup()** zetten we pin 13 (de variable **led**) als **OUTPUT** (uitgang). Output hoeft niet alleen maar voor LEDs gebruikt te worden maar kan bijvoorbeeld gebruikt worden voor een zoemertje of bijvoorbeeld een servo motor, een lcd-schermpje, etc. Dit in tegenstelling tot een INPUT (ingang) welke gebruikt kan worden voor een schakelaar, een licht sensor, etc.

Als de **setup()** functie is afgerond, zal de Arduino het “programma” in de **loop()** functie starten en blijven herhalen.

De volgende stappen vinden we in de loop:

Steps of our First Arduino Project

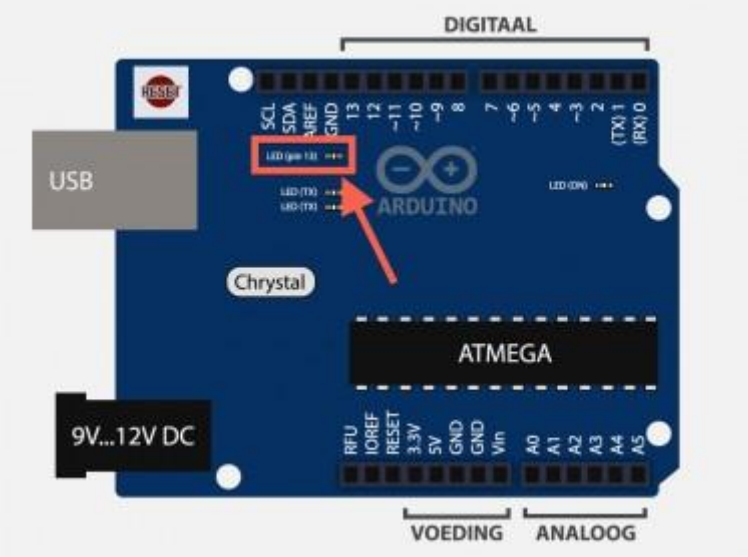
Code	Actie
digitalWrite(led, HIGH);	Zet **pin 13** (led) op **HIGH** – zet +5V op de pin zodat de **LED AAN** gaat
delay(1000);	**Wacht** tijd in milliseconden, hier **1 seconde** (1000 milliseconden = 1 seconde)
digitalWrite(led, LOW)	Zet **pin 13** (led) op **LOW** – de voltage wordt nu 0V en het **LED** gaat **UIT**
delay(1000);	**Wacht** tijd in milliseconden, hier **1 seconde** (1000 milliseconden = 1 seconde)

OK nu dat we weten wat we kunnen verwachten, kun je de code handmatig intypen (zie hieronder een versie zonder opmerkingen) of je kunt het inladen omdat het bij de Arduino software onder “Examples” zit (“Files” “Example” “01.Basics” “Blink“):

```
1  int led = 13;
2
3  void setup() {
4    pinMode(led, OUTPUT);
5  }
6
7  void loop() {
8    digitalWrite(led, HIGH);
9    delay(1000);
10   digitalWrite(led, LOW);
11   delay(1000);
12 }
```

Als je even kijkt naar de eerdergenoemde snelkoppelingen: na het invoeren van de code, klik op de “Upload” knop.

Als je geen rode foutmeldingen ziet in het zwarte veld onderin het Arduino Software venstertje, dan zul jee zien dat het LEDje gaat knipperen …
(tijdens de upload knipperen twee ledjes kort, onder de LED die aan pin 13 zit)



Figuur 1- 5

Arduino Uno – Het LEDje dat aan pin 13 zit

1.13. Wat extra “Hardware” voor ons eerste Arduino Project

In onze eerste stap gebruiken we alleen maar een Arduino board. Het LEDje dat aan pin 13 zit was hierbij handig.

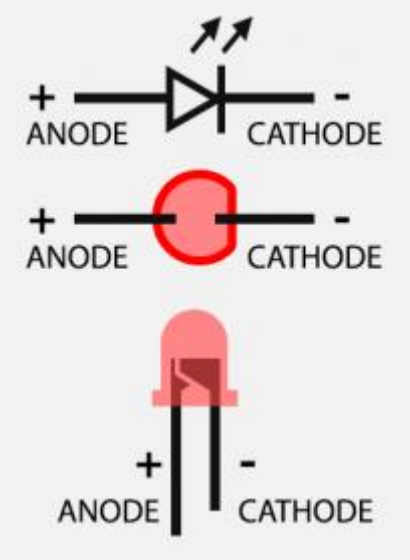
Nu kunnen we dit ook doen met een externe LED. De toepassing die ik hier laat zien is niet helemaal volgens de regels, maar het houdt alles eenvoudig en we hebben geen breadboard nodig.

Pas Op …!

Onder normale omstandigheden gebruiken we een extra weerstand (ongeveer 330Ω) tussen het LEDje en de stroomvoorziening (pin 13), voor een **kort** testje kunnen we dat achterwege laten, maar neem dit zeker niet als een voorbeeld hoe het correct gedaan moet worden.
Zie het Hoe gebruik je een Breadboard artikel voor een correctere aansluiting van een LED.

Pak een LEDje en bestudeer het Arduino board. Vindt pin 13 (links bovenin in bovenstaande afbeelding) die als plus gaat functioneren. Je zult zien dat het pinnetje ernaast GND (Ground of the wel aarde) is. Simpel gezegd: de min.

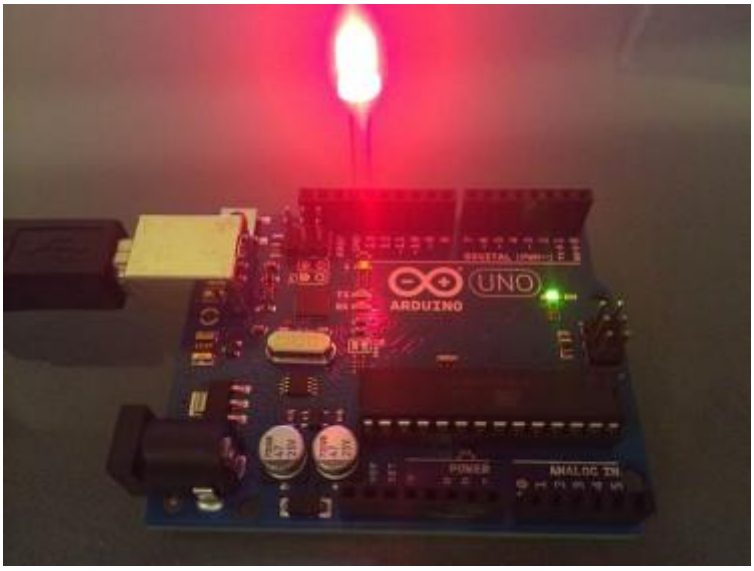
Kijk vervolgens goed naar het LEDje dat je net gepakt hebt. Bepaalde de Kathode pin en plaats deze boven de GND pin. De andere pin, Anode, schuiven we vervolgens in het gaatje van pin 13.



Figuur 1- 6

LED pinout

Je kunt dit zelfs doen terwijl de microcontroller draait, maar het is een goede gewoonte om eerst de stroom uit te zetten (USB stekker uit de computer trekken).
Let er op dat we dit maar voor een korte tijd doen, we willen voorkomen dat het LEDje uitbrandt. Zoals je in onderstaande foto ziet: het LEDje brandt best fel.
Klik op de foto om een grotere afbeelding te zien zodat je de pinnetjes in detail kunt zien.



Figuur 1- 7

Arduino UNO met extra LED