

PLANTENONDERZOEK ARDUINO

# Arduino Plantenonderzoek

In dit project leer je hoe je gegevens van sensoren kan opslaan voor het project plantenonderzoek. De sensoren moeten worden aangesloten op een Arduino-bord. Het Arduino-bord verwerkt de sensorsignalen. De gegevens die Arduino produceert, worden vervolgens via het datalogger shield op een SD kaart opgeslagen.

We gaan alle sensoren en functies van de data logger apart testen. Dit maakt het begrijpen en het vinden van fouten in het circuit en code gemakkelijker. De stappen die we gaan volgen zijn:

1. Testen DHT11 temperatuur- en luchtvochtigheidsensor
2. Testen SD card
3. Testen RTC Real Time Clock
4. SD en RTC samen
5. Sensor data logger

Voor het volgen van deze uitleg heb je de volgende onderdelen nodig:

* Arduino
* Arduino kabel
* Breadboard
* Data logger shield
* DHT11 luchtvochtigheid en temperatuur sensor module
* Kabeltjes
* SD kaart

## Arduino

Arduino is een open-source elektronicaplatform gebaseerd op eenvoudig te gebruiken hardware en software. Arduino-boards kunnen inputs van een sensor lezen en er een output van maken. Je kunt je board instrueren wat te doen door een set instructies naar de microcontroller op het board te sturen.

De Arduino heeft een aantal in en uitgangen die we eerst gaan bekijken:

Afbeelding met elektronica, circuit

Automatisch gegenereerde beschrijving**USB**: de USB-verbinding kan twee dingen doen. Ten eerste kan het de Arduino van stroom voorzien vanaf de computer. Ten tweede kan het communiceren met de computer.

**TX / RX**: als uploaden en communiceren niet via USB plaatsvinden, kunnen pins D0 en D1 worden gebruikt.

**Power in**: als de USB niet als voeding wordt gebruikt, kunnen de voedingsaansluiting (5,5 mm - 2,1 mm) of de voedingspinnen Vin en GND worden gebruikt. Arduino kan een stroom aan met een spanning van 5-20V.

**Power out**: voeding van externe componenten kan worden gegeven met de voedingspinnen 5V en 3.3V. De maximale stroom is 1A.

**Analoog in**: Arduino kan analoge spanning lezen van 0V tot 5V.

**Digitaal in / uit**: deze pinnen kunnen worden aangegeven als invoerpinnen of uitvoerpinnen. Als ingangspin kan het ingangsspanning lezen: <2.5V is LAAG,> 2.5V is HOOG. Als uitgangspin is LAAG 0V, HOOG is 5V. Pin 3, 5, 6, 9, 10 en 11 kunnen een PWM-signaal leveren.

Je kunt de Arduino programmeren met Arduino IDE. Je kunt Arduino IDE downloaden op: www.arduino.cc/en/Main/Software

# Verbinding maken met Arduino

Zorg ervoor dat je de Arduino IDE correct hebt geïnstalleerd. Open Arduino IDE en verbind de Arduino UNO via USB. Als stuurprogramma's correct zijn geïnstalleerd, herkent de computer het Arduino-bord.

Ga naar 'Hulpmiddelen' en verbind de COM-poort van het Arduino-bord door te klikken. Voor het uploaden naar de Arduino moet de COM-poort waar de Arduino op is aangesloten aangevinkt zijn.

Afbeelding met schermafbeelding

Automatisch gegenereerde beschrijving

In dit voorbeeld is het COM3

# Void setup en loop

De code waarmee de Arduino geprogrammeerd wordt bestaat standaard uit twee delen: de setup en de loop.

## Void Setup(){ … }

In het blok “Void setup “ staat tussen de accolades {} de informatie. Hier staat wat we allemaal gaan gebruiken. De setup wordt maar één keer ingelezen en vervolgens onthouden.

|  |
| --- |
| void setup() {   // Tussen de accolades wordt de code één keer uitgevoerd.   // Dit kan handig zijn om bijvoorbeeld te zeggen waar een led aangesloten is.  }  // Alles wat buiten de accolades staat is niet geldig en zal een ERROR geven. |

## Void loop(){ … }

In het blok “Void loop” staat de informatie ook tussen de accolades {}. In deze informatie staat wat we allemaal gaan doen. De “Void loop” wordt niet één keer ingelezen, maar iedere keer herhaald. Binnen het blok “Void loop” kun je kleinere blokjes maken, die staan dan ook weer binnen twee accolades. Het aantal accolades is dus altijd even. Bij het schrijven van een programma kun je natuurlijk wel eens een accolade vergeten. Als je in de sketch (je programmacode) naast een accolade gaat staan laat het programma zien waar de tweede staat, dit ter controle.

|  |
| --- |
| void loop(){   // Alle code tussen de accolades in de loop worden eeuwig herhaald.   // Dit kan bijvoorbeeld handig zijn om een lampje te laten knipperen.  }  // Alle code buiten de accolades zijn niet geldig en geven een ERROR |

## Speciale Tekens

Arduino gebruikt een aantal verplichte speciale tekens in de code:

**; Puntkomma.** Iedere programmaregel moet worden afgesloten met een puntkomma. Een vergeten puntkomma zorgt ervoor dat een programma niet werkt.

**{} Accolades.** Een accolade geeft het begin en einde van een blok informatie weer. Binnen een blok accolades kunnen meerdere kleine blokken met eigen accolades voorkomen.

**// Regel commentaar.** In elke regel kan je // plaatsen. De regel achter de // wordt niet meer gelezen door Arduino. Handig voor uitleg/commentaar. Maar het kan ook gebruikt worden om een programmaregel (tijdelijk) uit te schakelen.

**/\*….\*/ Commentaarblok.** Als je uitleg of commentaar over meerdere regels wilt schrijven kan je gebruik maken van de tekens voor commentaar blok. Uitleg over het programma is, zeker voor anderen, heel handig.

Variabelen

Een variabele is een manier om een waarde een naam te geven en op te slaan voor later gebruik door het programma, zoals gegevens van een sensor of een tussenwaarde die wordt gebruikt in een berekening. Een variabele heeft een naam, waarde en type.

Variabelen opgeven

Voordat ze worden gebruikt, moeten alle variabelen worden opgegeven. Een variabele opgeven betekent het type definiëren en optioneel een beginwaarde instellen (de variabele initialiseren). Variabelen hoeven niet te worden geïnitialiseerd (een waarde toegewezen) wanneer ze worden opgegeven, maar het is vaak handig.

Je moeten bij het kiezen van het type variabele rekening houden met wat je wilt opslaan (hele getallen, kommagetallen, tekst ,…) en de grootte van de getallen.

|  |
| --- |
| int ledRood = 13 |

maakt een variabele waarvan de naam pin is, waarvan de waarde 13 is en waarvan het type int is. Later in het programma kun je naar deze variabele verwijzen door de naam, waarna de waarde wordt opgezocht en gebruikt. In deze verklaring bijvoorbeeld:

|  |
| --- |
| pinMode (ledRood, OUTPUT); |

Voordeel van zo’n variabele is dat het de code overzichtelijker maakt: Je hoeft in het voorbeeld de waarde 13 maar één keer op te geven, daarna kun je de naam gebruiken.

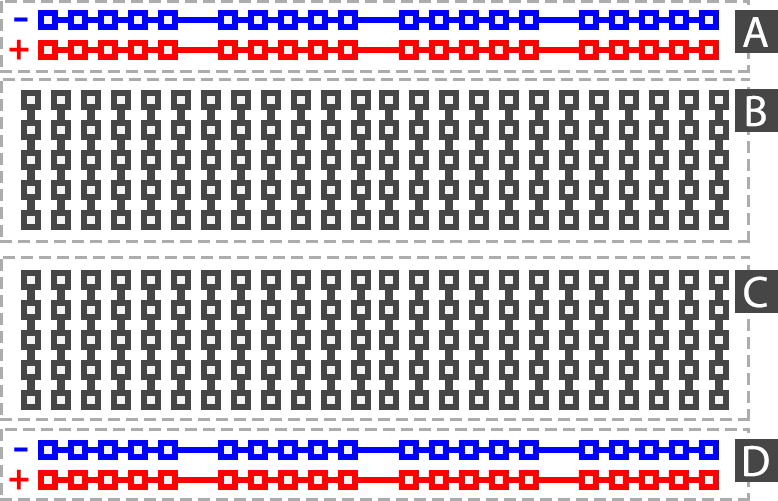
De volgende tabel bevat de types die je tijdens het programmeren met Arduino kan gebruiken.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| void | Boolean | char | Unsigned char | byte | int | Unsigned int | word |
| long | Unsigned long | short | float | double | array | String-char array | String-object |

# Breadboard

Met het breadboard kun je componenten met elkaar verbinden en zo circuits maken, zonder dat je hoeft te solderen. Het breadboard heeft verschillende gebieden:

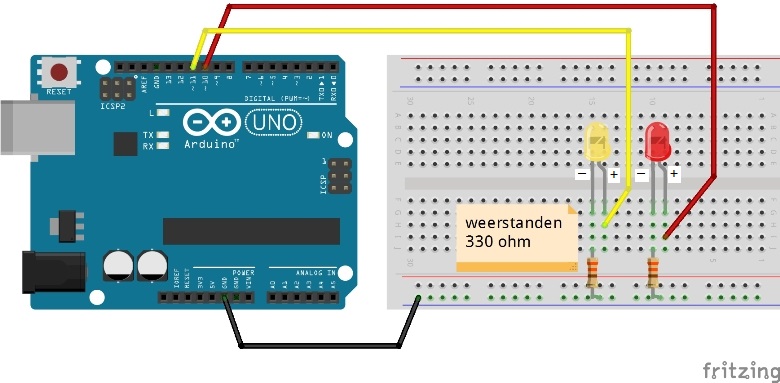
* in de onderstaande illustratie A en D zijn horizontaal met elkaar verbonden en worden gebruikt voor stroomvoorzieningen
* in diezelfde illustratie zijn B en C het raster voor componenten. Deze zijn vertikaal verbonden. Tussen B en C zit een scheiding.



Door de 5V van de Arduino met een draadje in de + van het breadboard te steken kun je deze hele rij van stroom voorzien. Hetzelfde met de GND en de – rij van het breadboard.

# Stoplicht (<https://training.funmetelectronica.nl/docs/arduino-basis/lessen/les-2-meerleds/>)

Laten we klein beginnen. Een eenvoudig, enkel verkeerslicht is een goede plek om te beginnen. Hier is het circuit: Als de opstelling correct gebouwd is volgens de bouwtekening en de code geüpload is zullen twee lampjes gaan knipperen.



|  |
| --- |
| /\*   Autheur:      Fun met Electronica   Link:         https://training.funmetelectronica.nl/docs/arduino-basis/lessen/les-2-meerleds/   Beschrijving: Laat eens meer leds aan en uit schakelen, keer op keer.  \*/    int ledRood = 10;   // Rode led op pin 10  int ledGeel = 11;   // Gele led op pin 11    /\*   Het programmablok void setup wordt eenmaal beschreven  \*/  void setup() {   // We geven aan dat de led-pin-aansluitingen output moet zijn (spanning moet geven)   pinMode(ledRood, OUTPUT);   pinMode(ledGeel, OUTPUT);  }    /\*   Het programmablok void loop herhaalt zich keer op keer  \*/  void loop() {   digitalWrite(ledRood, HIGH);  // zet spanning op de led-pin   digitalWrite(ledGeel, HIGH);  // zet spanning op de led-pin   delay(1000);                  // wacht een seconde (1000 miliseconden)   digitalWrite(ledRood, LOW);   // zet de spanning van de led-pin af   digitalWrite(ledGeel, LOW);   // zet de spanning van de led-pin af   delay(1000);                  // wacht een seconde  } |

# Opdrachten

**Opdracht:**

Zorg dat de lampjes om en om knipperen.

**Opdracht:**

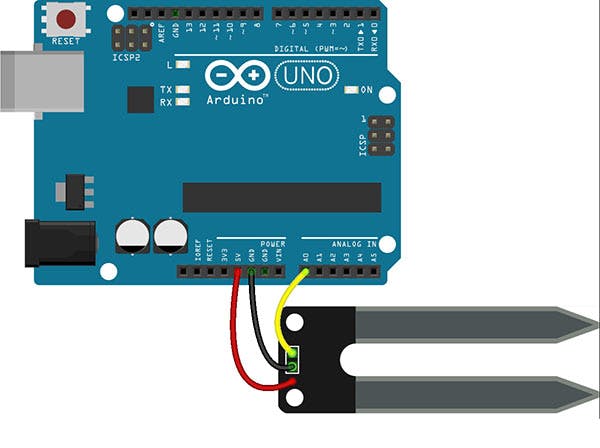
Voeg een groen lampje toe op pin 12 en laat deze mee knipperen.

**Opdracht:**

Maak van deze opstelling een werkende stoplicht met geloofwaardige delays.

Bodemvochtigheidsensor

De bodemvochtigheidsensor is een sensor die de vochtigheid van de bodem kan meten voor planten in huis en op het balkon. De weerstand van deze sensor verandert aan de hand van de vochtigheid van de grond. Hoe lager de weerstand hoe hoger de vochtigheid van de grond.



Gebruik de volgende code om een waarde voor de bodemvochtigheid te meten.

Voor elke bodemvochtmeting nemen we gemiddeld 100 sensorgegevens om de gegevens stabieler en nauwkeuriger te maken.

De code bevat een “for” structuur. Hiermee laat je een stukje programma dat tussen de { } staat, een bepaald aantal keren uitvoeren.

(int i = 0; i <= 100; i++) Hiermee wordt een variabele i aangemaakt die begint bij 0, het gaat door totdat i kleiner of gelijk is aan 100, i++ betekent dat er steeds 1 bij i wordt opgeteld.

Code:

|  |
| --- |
| #define SensorPin A0 //stelt in dat de sensor op A0 is aangesloten  float sensorValue = 0;  //maakt een variabele sensorValue aan en geeft de beginwaarde 0  void setup() {  Serial.begin(9600); //start de communicatie tussen de arduino en pc  }  void loop() {  for (int i = 0; i <= 100; i++) //doet het volgende 100x  {  sensorValue = sensorValue + analogRead(SensorPin);  //leest de waarde van de SensorPin en telt deze op bij de vorige sensorwaarde  delay(1);  }  sensorValue = sensorValue/100.0; //deelt de waarde door 100  Serial.println(sensorValue); //print de waarde in de seriele monitor  delay(30); //wacht 30ms  } |

Seriele monitor

In de seriele monitor kun je de waardes van de luchtvochtigheid en temperatuur zien verschijnen. Open hiervoor de seriele monitor die je rechts in de Arduino IDE vindt.



# Wat zijn libraries?

Libraries zijn een verzameling code waarnaar je kan verwijzen in je sketch. Hierdoor kun je je eenvoudig verbinding kunt maken met een sensor, display, module, enz. De ingebouwde LiquidCrystal-bibliotheek maakt het bijvoorbeeld gemakkelijk om met LCD-displays met tekens te praten. Een aantal libraries zijn standaard geïnstalleerd en op internet zijn er honderden extra bibliotheken beschikbaar om te downloaden. Om de extra bibliotheken te gebruiken, moet je deze installeren.

# Hoe een library te installeren

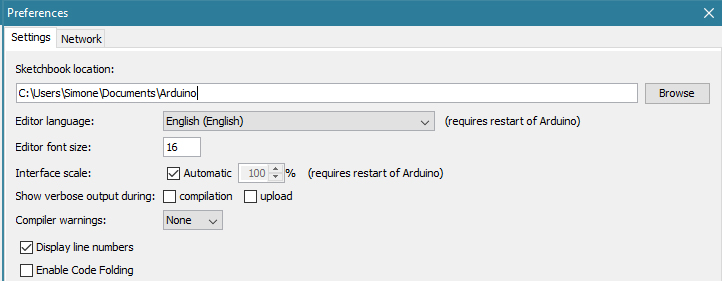
Er zijn 3 manieren om een library te installeren:

* Via de Library Manager
* Een .zip Libary installeren
* En handmatig installeren

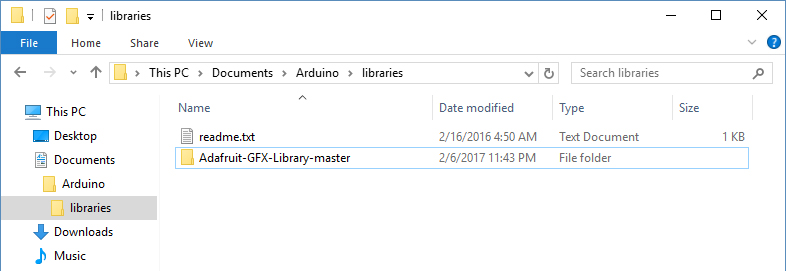
De libraries voor dit project kun je downloaden via It’s Learning en vervolgens handmatig installeren.

Handmatige installatie

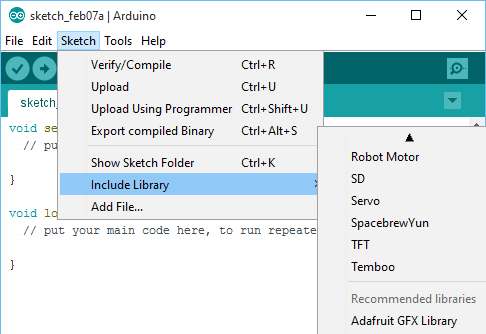
Als je een bibliotheek handmatig wilt toevoegen, moet je deze downloaden als een ZIP-bestand, deze uitpakken en in de juiste map plaatsen. Je kunt de locatie van uw schetsboekmap vinden of wijzigen via file> preferences> sketchbook location.



Kopieer de uitgepakte libraries van It’s Learning naar de map ‘libaries’ bij je sketchbook location.



Start de Arduino Software (IDE), ga naar Sketch> Include Library. Controleer of de libraries die je zojuist hebt toegevoegd beschikbaar zijn in de lijst.



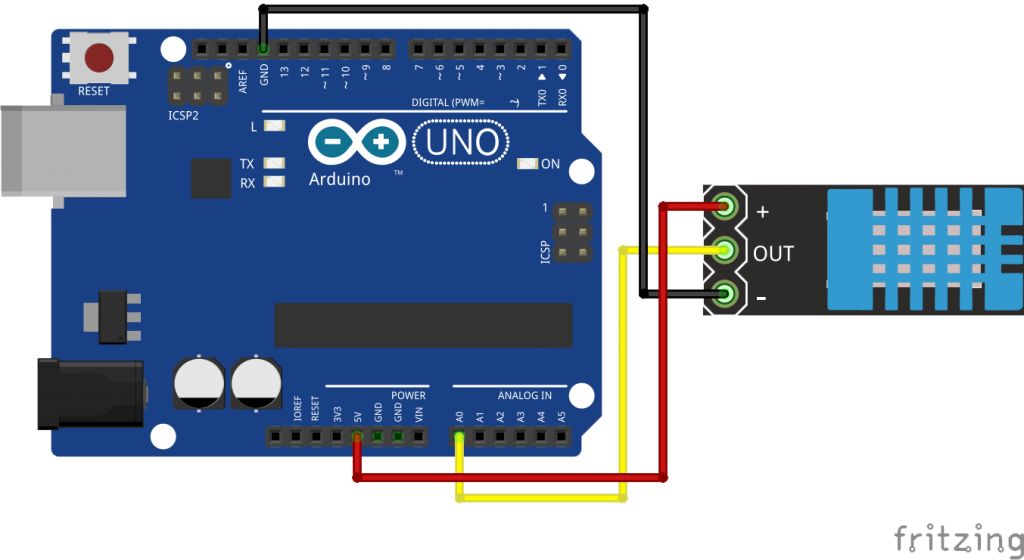
Wanneer je een library in je code wilt gebruiken zul je die moeten toevoegen (include). Dit doe je aan het begin van de code. Bijvoorbeeld:

|  |
| --- |
| #include "Adafruit\_CCS811.h" |

DHT11 TEMPERATUUR EN LUCHTVOCHTIGHEIDSENSOR   
(<https://www.brainy-bits.com/dht11-tutorial/>)

# In dit deel zullen we leren hoe je de DHT11 temperatuur- en vochtigheidssensor gebruikt. Deze sensor is nauwkeurig genoeg voor de meeste projecten die de vochtigheids- en temperatuurmetingen moeten bijhouden. We zullen een library gebruiken die speciaal is ontworpen voor deze sensoren en die ervoor zorgt dat onze code kort en gemakkelijk te schrijven is.

# Circuit



Zoals je kunt zien, hebben we slechts 3 verbindingen met de sensor nodig. De verbindinen zijn: spanning (5V), Aarde (Gnd) en signaal (A0). De laatste kan worden aangesloten op elke analoge pin op de Arduino.

DE CODE

Omdat we een library gebruiken die beschikbaar is voor deze sensor, is onze code erg kort en eenvoudig.

Zorg dat je de DHT library geinstalleerd hebt. En upload de volgende op de Arduino

*#include "dht.h" // Toevoegen van de DHT library aan de code*

*#define dht\_apin A0 // Analoge pin waarop de sensor is aangesloten*

dht DHT;

void setup(){

Serial.begin(9600); //Start de seriele communicatie van de arduino met de pc

delay(500); //Halve seconde wachten om op te starten

Serial.println("DHT11 Humidity & temperature Sensor");

//print tekst in de seriele monitor

delay(1000); //Wacht seconde om op te starten

} //Einde van de setup

void loop(){

DHT.read11(dht\_apin); //Leest de sensor op A0

Serial.**print**("Current humidity = "); //print tekst in de seriele monitor

Serial.**print**(DHT.humidity); //print luchtvochtigheidwaarde in monitor

Serial.**print**("% "); //print tekst % in seriele monitor

Serial.**print**("temperature = "); //print tekst in de seriele monitor

Serial.**print**(DHT.temperature); //print temperatuurwaarde

Serial.println("C ");//print tekst C in de seriele monitor en start nieuwe regel

delay(5000); //Wacht 5 seconden alvoor de sensor weer te lezen.

} // Einde van de loop

# DATA LOGGER SHIELD - REAL TIME CLOCK

Het datalogger shield bevat een RTC (Real Time Clock). Een realtime klok is een klok die de huidige tijd bijhoudt en die kan worden gebruikt om acties op een bepaald tijdstip te programmeren.

Om de tijd en de RTC bij te blijven houden, is het noodzakelijk om de klok ingeschakeld te houden. Hiervoor zit er op het de data logger shield een knoopcel battarij.



|  |
| --- |
| #include <Wire.h> //voegt library toe aan de code  #include "RTClib.h"    **RTC\_DS1307** RTC;    void setup (){  **Serial**.begin(9600); //start verbinding arduino en pc     Wire.begin();     RTC.begin(); //start de real time clock     if (! RTC.isrunning()) //als de klok niet loopt dan de volgende regels uitvoeren   {  **Serial**.println("RTC is NOT running!");     //zet de klok op de datum en tijd waarop deze code is gecompiled     RTC.adjust(**DateTime**(\_\_DATE\_\_, \_\_TIME\_\_));   }    }    void loop ()  {  **DateTime** now = RTC.now(); //leest de tijd van de RealTimeClock uit    **Serial**.print(now.year(), DEC); //print het huidige jaar in de monitor  **Serial**.print('/'); //print een / in de monitor  **Serial**.print(now.month(), DEC); //print de huidige maand in de monitor  **Serial**.print('/');  **Serial**.print(now.day(), DEC);  **Serial**.print(' ');  **Serial**.print(now.hour(), DEC);  **Serial**.print(':');  **Serial**.print(now.minute(), DEC);  **Serial**.print(':');  **Serial**.print(now.second(), DEC);  **Serial**.println(); //start een nieuwe regel     delay(1000); //wacht een seconde  } |

# Opdrachten

**Opdracht:**

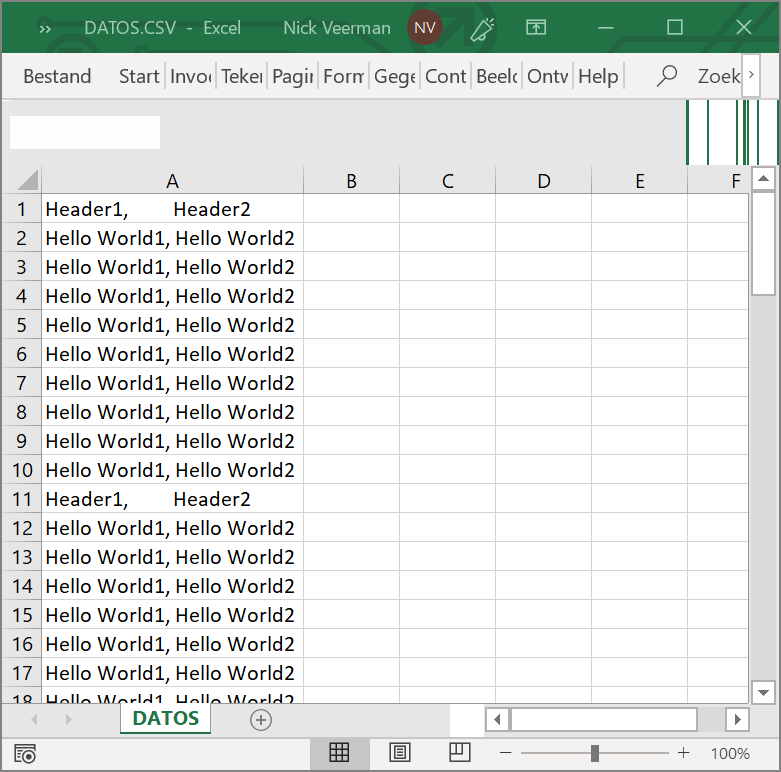
Druk het shield op de Arduino en plaats de knoopcelbatterij (CR1220) in het shield. Zorg ervoor dat de RTC library is geïnstalleerd en upload de code. Open de seriele monitor om de tijd te zien.

**Opdracht:**

Zorg ervoor dat de tijd om de 2 seconden verschijnt.

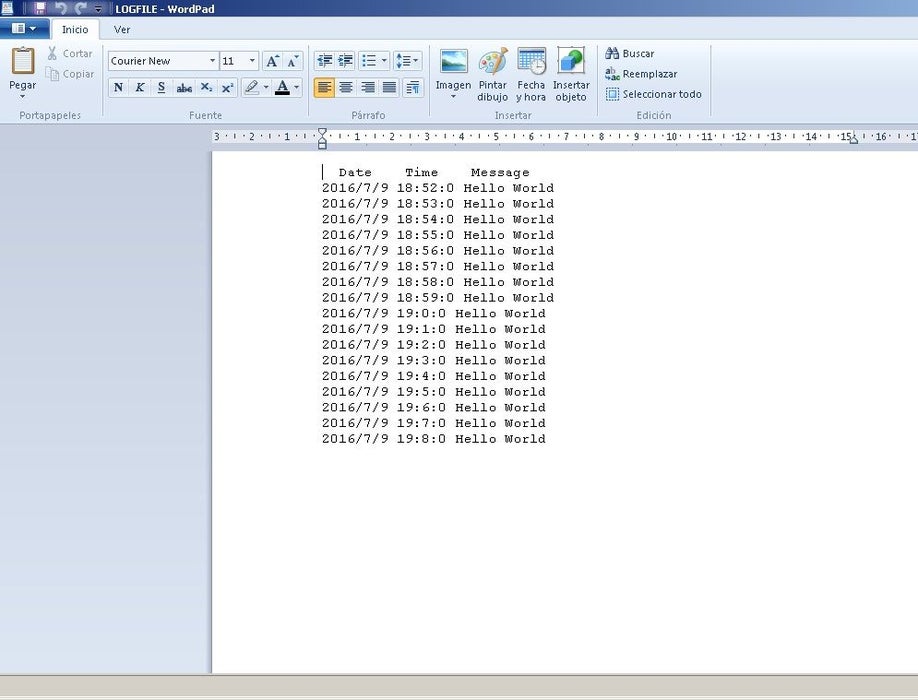
# Testen SD kaart

Deze schets maakt het bestandje “Datos.csv” op het sd kaartje, schrijft twee koppen in de void setup. “Hello world1” en “Hello world2” wordt elke seconde (zie void loop) op je sd-kaart. Geef het een minuutje en controleer je SD-kaart. Controleer of je het bestand op de sd kaart staat. Je zult zoiets als op de volgende foto krijgen.



|  |
| --- |
| //Test SD card  //NJarpa  #include <**SD**.h>                                  //libraries die we nodig hebben  #include <**SPI**.h>  #include <Wire.h>  const int chipSelect = 10;  File Datos;                      //Naam van het bestand dat je op de sdkaart aanmaakt  void setup() {  pinMode(10 , OUTPUT);  **SD**.begin(chipSelect);  Datos=**SD**.open("Datos.csv",FILE\_WRITE);    //Opent het bestand om de koppen te printen  Datos.println("Kop1,Kop2");   //Print koppen (nog niet opgeslagen)  Datos.close();                                   //slaat print op  }  void loop() {    Datos = **SD**.open("Datos.csv", FILE\_WRITE);       //Opent bestand  Datos.println("Hello World1, Hello World2");    //Print tekst  Datos.close();                                  //slaat print op  delay(1000);                                    //wachten  } |

# SD en RTC samen



Nu gaan we "Hello World" naar de sd-kaart schrijven met de datum en tijd erbij.

Als alles goed werkt, krijg je hetzelfde als op de vorige afbeelding.

|  |
| --- |
| //Data Logger NJarpa  #include <**SD**.h>                      //Libraries we need  #include <Wire.h>  #include <RTClib.h>  #include <**SPI**.h>  **RTC\_DS1307** RTC;                      // Define the Real Time Clock object  const int chipSelect = 10;            //CS pin of your shield  File Logfile;                        //Name of the file  void setup() {  pinMode(10 , OUTPUT);  **SD**.begin(chipSelect);  RTC.begin();  Wire.begin();  Logfile=**SD**.open("Logfile.csv",FILE\_WRITE);        //Will open and will write once just for headers  Logfile.println("  Date,Time,Message");     //Print headers (not saved yet)  Logfile.close();                                  //Print saved  }  void loop(){    **DateTime** now = RTC.now();                        //Clock call  if(now.second()==0){                            //Sample every minute    Logfile=**SD**.open("Logfile.csv",FILE\_WRITE);       //Will open and will write date  now =RTC.now();  Logfile.print(now.year(), DEC);                //Print date and time(not saved yet)  Logfile.print("/");  Logfile.print(now.month(), DEC);  Logfile.print("/");  Logfile.print(now.day(), DEC);  Logfile.print(", ");  Logfile.print(now.hour(), DEC);  Logfile.print(":");  Logfile.print(now.minute(), DEC);  Logfile.print(":");  Logfile.print(now.second(), DEC);  Logfile.print(", Hello World");                   //Print message(not saved yet)  Logfile.println();                               //Jump to the next line  Logfile.close();                                 //Print saved  }    delay(1000);                                    //One data per second  } |

# Data logger met sensor

Nu voegen we de code voor de sensor toe. Laten we starten met de DHT11-sensormodule. We nemen de code van de laatste stap en voegen de regels toe die we nodig hebben.

|  |
| --- |
| //Data Logger DHT11 NJarpa  #include <dht.h>  #include <**SD**.h>                      //Libraries we need  #include <Wire.h>  #include <RTClib.h>  #include <**SPI**.h>  **RTC\_DS1307** RTC;                      // Define the Real Time Clock object  #define DHTPIN A1                     // what digital pin we're connected to  #define DHTTYPE DHT11                // DHT 11  DHT dht(DHTPIN, DHTTYPE);  const int chipSelect = 10;            //CS pin of your shield  File Logfile;                        //Name of the file  void setup() {  pinMode(10 , OUTPUT);           //Uncomment if you need  **SD**.begin(chipSelect);  RTC.begin();  Wire.begin();  dht.begin();  Logfile=**SD**.open("Logfile.csv",FILE\_WRITE);                //Will open and will write once just for headers  Logfile.println("  Date    Time    Humidity   Temp");     //Print headers (not saved yet)  Logfile.close();                                          //Print saved  }  void loop(){    float h = dht.readHumidity();                 //Read humidity    float t = dht.readTemperature();              //Read temperature celcius    **DateTime** now = RTC.now();                        //Clock call  if(now.second()==0){                            //Sample every minute    Logfile=**SD**.open("Logfile.csv",FILE\_WRITE);       //Will open and will write date  now =RTC.now();  Logfile.print(now.year(), DEC);                //Print date and time(not saved yet)  Logfile.print("/");  Logfile.print(now.month(), DEC);  Logfile.print("/");  Logfile.print(now.day(), DEC);  Logfile.print(" ");  Logfile.print(now.hour(), DEC);  Logfile.print(":");  Logfile.print(now.minute(), DEC);  Logfile.print(":");  Logfile.print(now.second(), DEC);  Logfile.print("  ");                            //Space between data  Logfile.print(h);                               //Print humidity(not saved yet)  Logfile.print("%");                             //Print simbol  Logfile.print("  ");                            //Space between data  Logfile.print(t);                               //Print temperature  Logfile.print("°C");                            //Print simbol  Logfile.println();                               //Jump to the next line  Logfile.close();                                 //Print saved  }    delay(1000);                                    //One data per second  } |

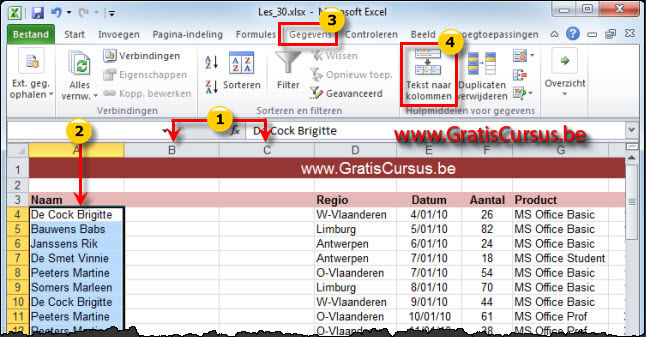
# Opdrachten

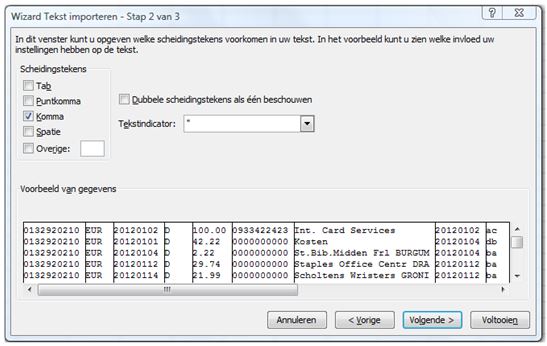
**Opdracht:**

Upload de code en wacht een paar minuten. Check vervolgens de SD-kaart om te zien of de sensorwaarden goed worden opgeslagen.

Excel tekst naar kolommen

De gegevens op de SD kaart staan in één kolom. In excel kun je de functie ‘tekst naar kolommen’ gebruiken om de tekst in meerdere kolommen te krijgen.





1. Selecteer de cel of de kolom met de tekst die u wilt splitsen.
2. Selecteer **gegevens**> **tekst naar kolommen.**
3. Selecteer in de wizard **Tekst naar kolommen** de opties **Gescheiden** > **Volgende**.
4. Selecteer de **Scheidingstekens** voor uw gegevens. Bijvoorbeeld **Komma** of **Spatie**. In het venster **Voorbeeld van gegevens** ziet u een voorbeeld van de gegevens.
5. Selecteer **Volgende**.
6. Selecteer **Gegevenstype per kolom** of gebruik de voorgestelde keuze.
7. Selecteer de **Bestemming**, dus de plaats waar u de gesplitste gegevens in het werkblad wilt weergeven.
8. Selecteer **Voltooien**.