

Projet Domotique

Arrosage automatique

Dans le cadre de nos études nous avons dû réaliser un projet pour le cours de Domotique enseigné et dirigé par M. Salvat.

Parmi différents choix de projets proposés nous avons choisis et préférés réaliser celui sur une pompe qui automatiquement arrose la terre d'une plante lorsque celle-ci manque d'humidité.

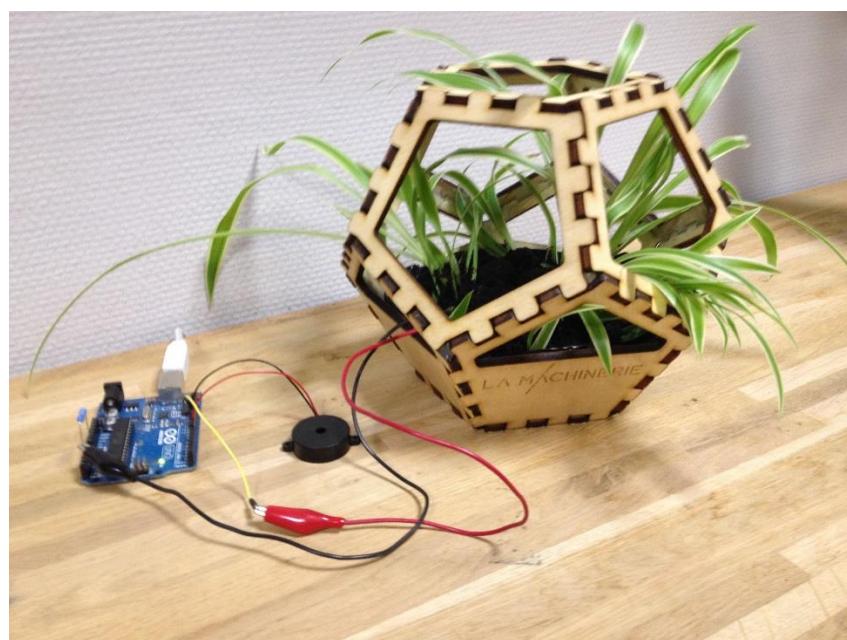
Il était essentiel durant ce projet de pouvoir récupérer les valeurs données par les différents capteurs, que ce soit des capteurs d'humidité, de poussière ou encore de qualité d'air comme la mesure de gaz CO₂.

Pour réaliser ce projet nous avons dû dans un premier temps documenter chacun des capteurs à notre disposition.

Par la suite notre but était de faire fonctionner ces différents éléments en les connectant à une carte Arduino suivant les branchements indiqués.

Pour finir, il était question de faire fonctionner ces capteurs ensemble afin de pouvoir mettre en forme notre projet dans son état final.

Voici donc un document qui recense chacun des matériels utilisés avec leurs caractéristiques, leurs branchements et un code Arduino pour chacun permettant de les faire fonctionner.



SOMMAIRE

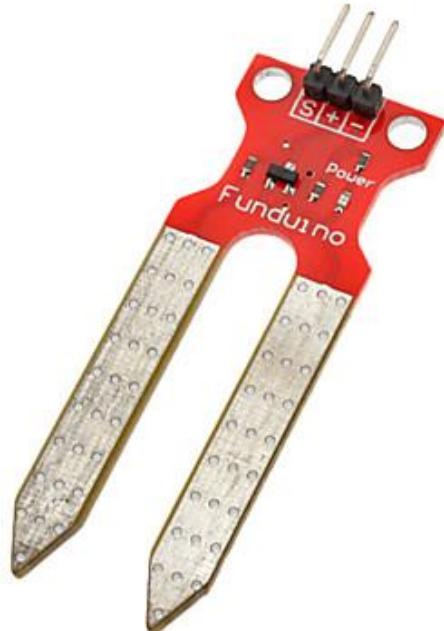
Capteur d'humidité pour sol	3
Funduino	3
FC-28	6
Ecran LCD & OLED	9
2 lignes	9
Adafruit	12
Capteur de poussière	17
Samyoung DSM501	17
DéTECTEUR optique de poussière	20
Bandeau LED	23
Adaptateur micro SD pour Arduino	26
Capteur de CO ₂	29
MISIR-5000	29
MiCS-VZ-89	32
OGM 200	36
Datalogger sur WeMos D1	40

GitHub :

Code : <https://github.com/NicolasGreaux/ProjetDomotique2016>

Wiki: <https://github.com/NicolasGreaux/ProjetDomotique2016/wiki>

Capteur d'humidité pour sol Funduino

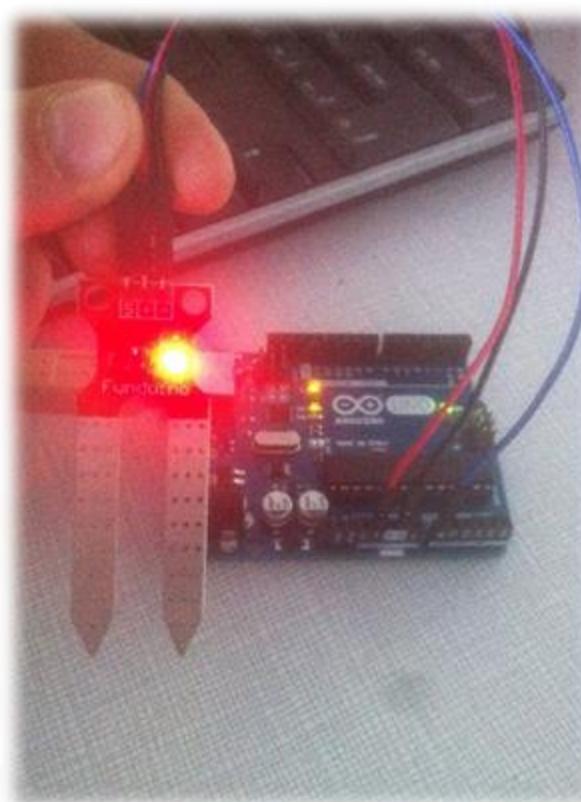


Courant de fonctionnement: moins de 20mA

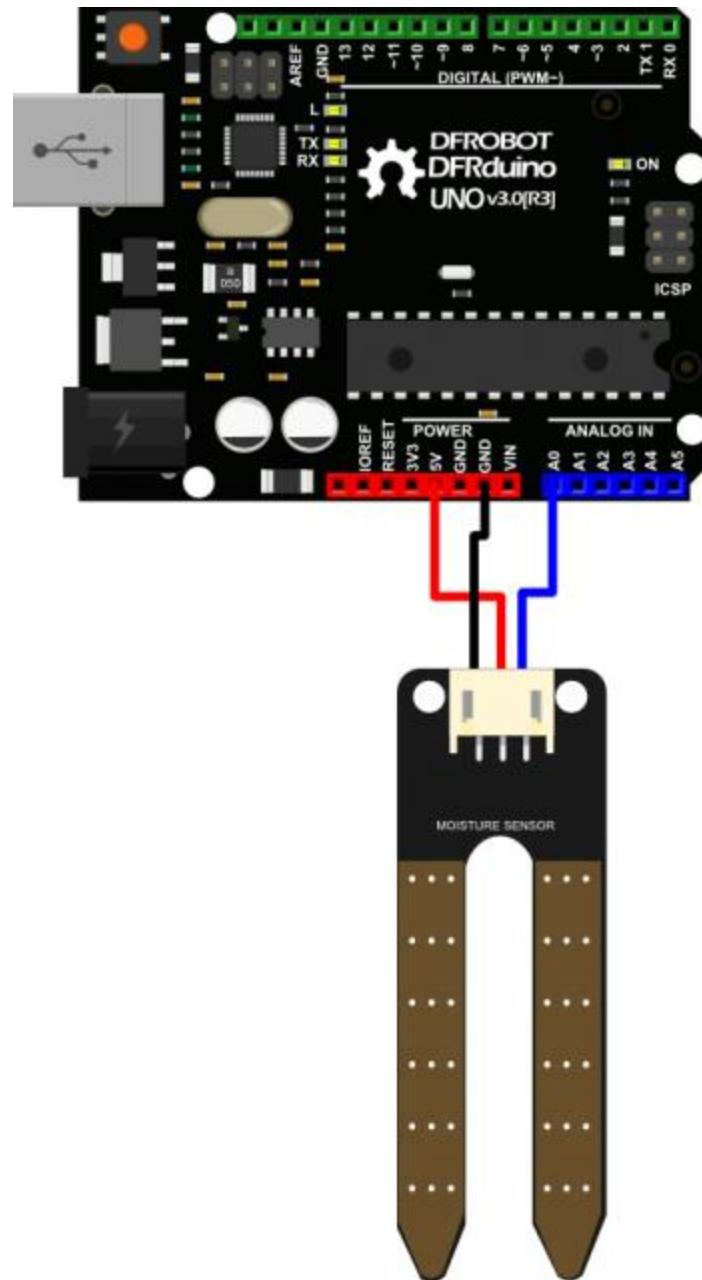
Tension de sortie: 0 ~ 2.3V (2.3V lorsque le capteur complètement trempé dans l'eau)

Tension d'alimentation: 5v

Type de capteur: Sortie analogique



Branchements du capteur Funduino sur Arduino

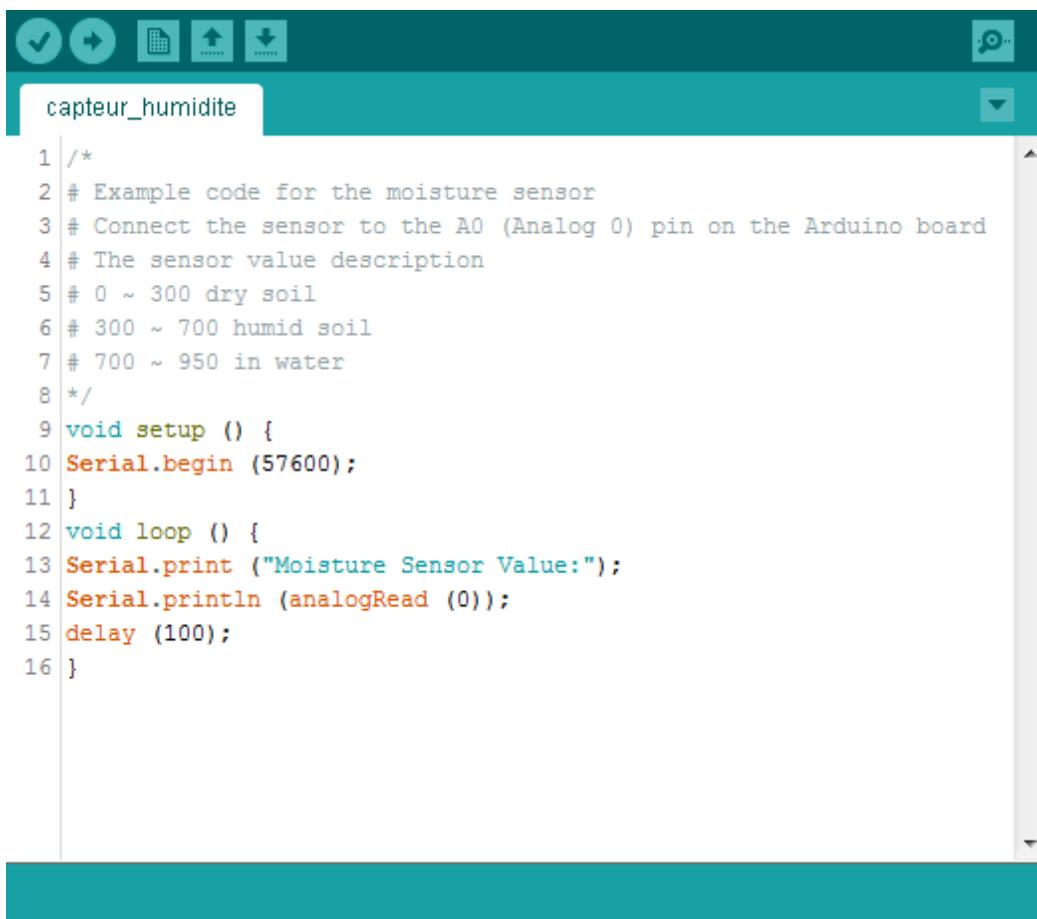


Le pin + se branche sur le 5V de la carte arduino

Le pin - se branche sur le GND de la carte arduino

Le pin S se branche sur le A0 de la carte arduino

Code utilisé pour tester le fonctionnement de ce capteur d'humidité du sol :



The screenshot shows the Arduino IDE interface with a teal header bar. The title bar says "capteur_humidite". Below the header is a code editor containing the following C++ code:

```
1 /*
2 # Example code for the moisture sensor
3 # Connect the sensor to the A0 (Analog 0) pin on the Arduino board
4 # The sensor value description
5 # 0 ~ 300 dry soil
6 # 300 ~ 700 humid soil
7 # 700 ~ 950 in water
8 */
9 void setup () {
10 Serial.begin (57600);
11 }
12 void loop () {
13 Serial.print ("Moisture Sensor Value:");
14 Serial.println (analogRead (0));
15 delay (100);
16 }
```

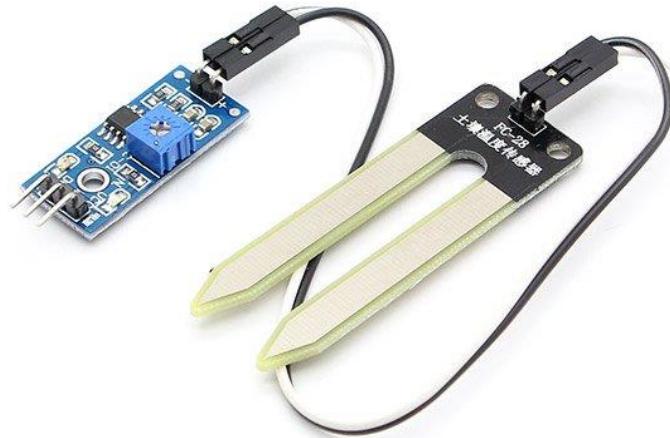
Avec ce code et en plaçant le capteur dans de l'eau on obtient une variation des valeurs données par le programme

```
moisture sensor value.
Moisture Sensor Value:0
Moisture Sensor Value:2
Moisture Sensor Value:7
Moisture Sensor Value:21
Moisture Sensor Value:41
Moisture Sensor Value:54
Moisture Sensor Value:60
Moisture Sensor Value:64
Moisture Sensor Value:68
```

Moment où l'on a trempé
le capteur dans de l'eau

Validation du test du capteur d'humidité Funduino

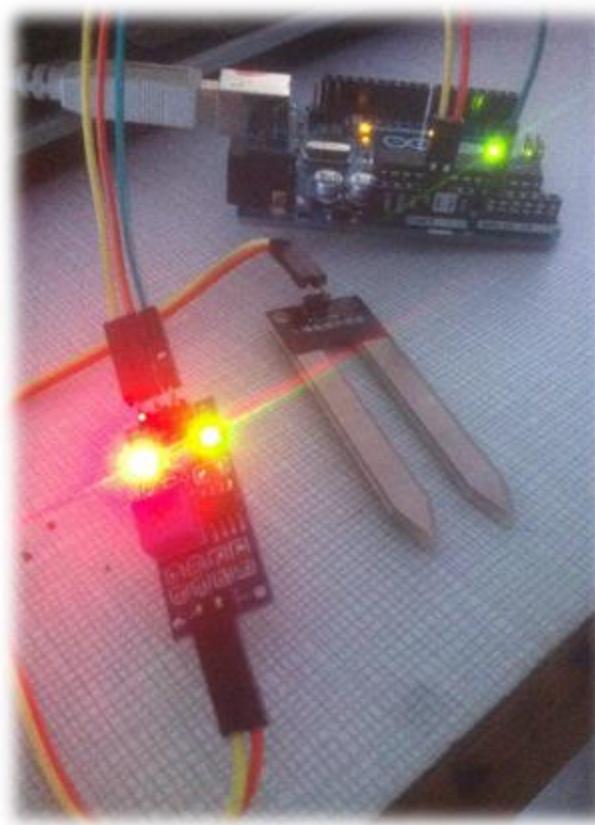
FC-28



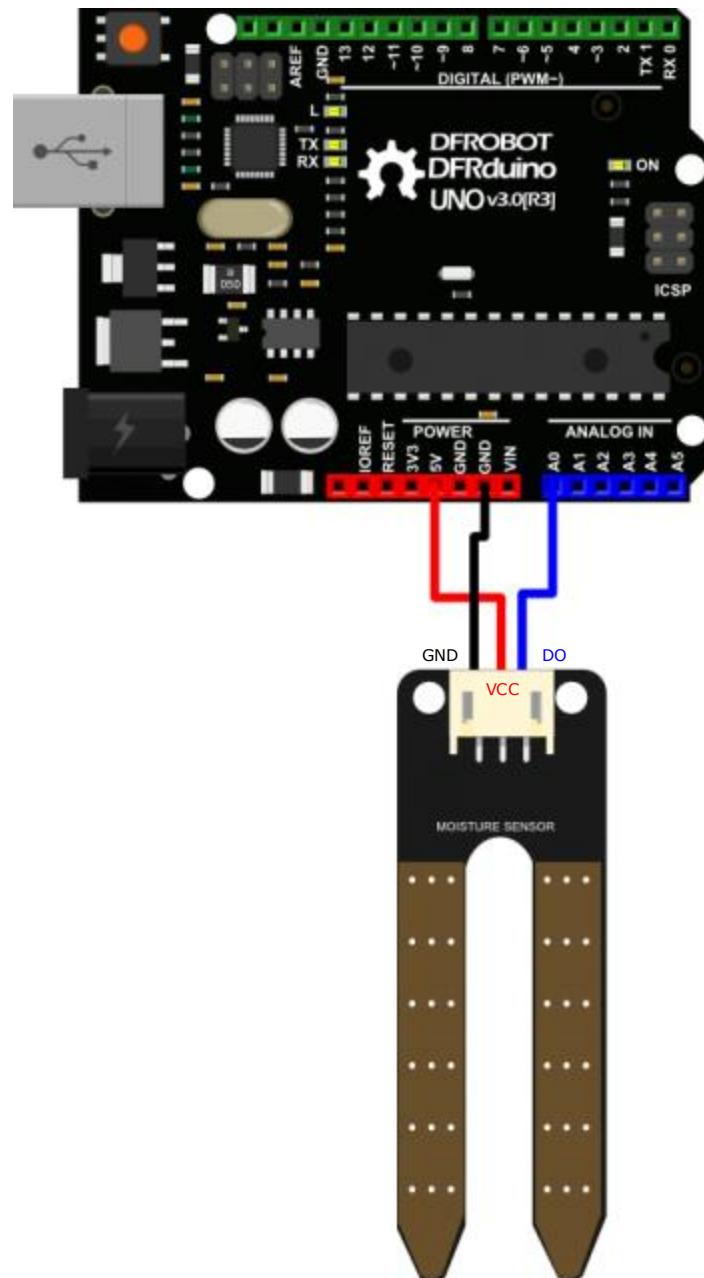
Tension de fonctionnement: 3,3 ~ 5V

Mode double sortie: sortie analogique plus précis

Voyant d'alimentation **rouge** et l'indicateur de sortie de commutation numérique **vert**



Les branchements pour faire fonctionner ce capteur sont les mêmes que pour le capteur précédent, c'est-à-dire:

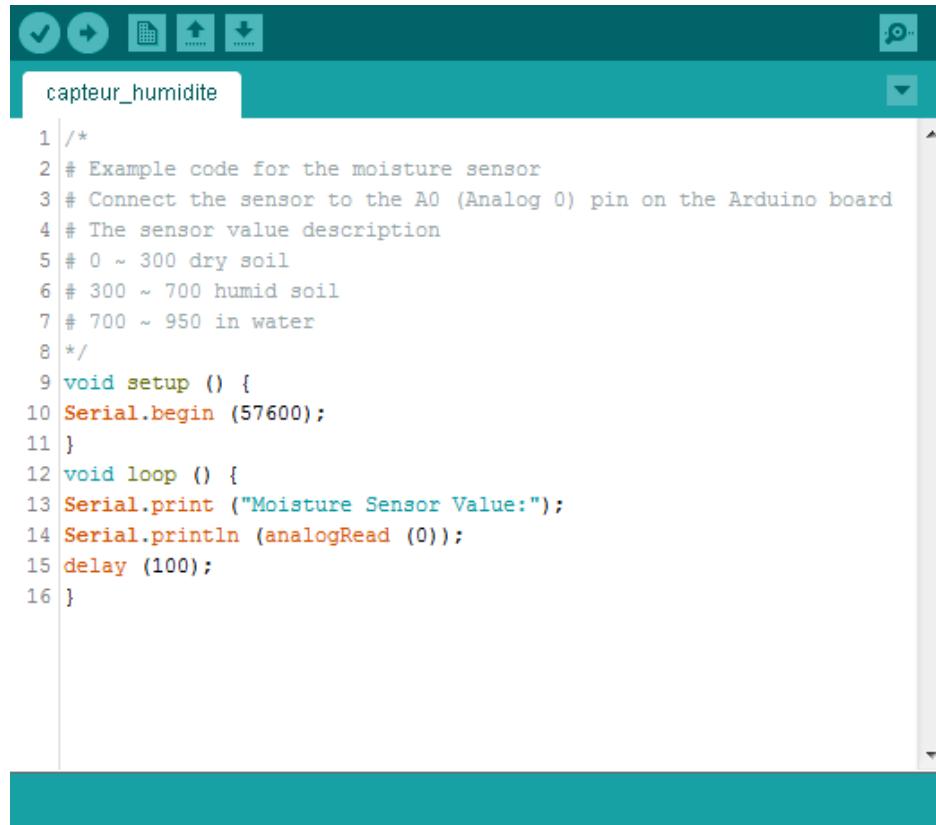


Le pin VCC se branche sur le 5V de la carte arduino

Le pin TERRE se branche sur le GND de la carte arduino

Le pin DO se branche sur le A0 de la carte arduino

On a utilisé le même code que précédemment pour tester ce capteur d'humidité FC-28:



The screenshot shows the Arduino IDE interface with a teal header bar. The title bar says "capteur_humidite". Below the header is a code editor containing the following C++ code:

```
1 /*  
2 # Example code for the moisture sensor  
3 # Connect the sensor to the A0 (Analog 0) pin on the Arduino board  
4 # The sensor value description  
5 # 0 ~ 300 dry soil  
6 # 300 ~ 700 humid soil  
7 # 700 ~ 950 in water  
8 */  
9 void setup () {  
10 Serial.begin (57600);  
11 }  
12 void loop () {  
13 Serial.print ("Moisture Sensor Value:");  
14 Serial.println (analogRead (0));  
15 delay (100);  
16 }
```

Comme précédemment en plaçant ce capteur d'humidité dans de l'eau on obtient une variation des valeurs obtenues, mais les résultats sont moins significatifs:

```
moisture sensor value:34  
Moisture Sensor Value:29  
Moisture Sensor Value:28  
Moisture Sensor Value:28  
Moisture Sensor Value:28  
Moisture Sensor Value:28
```

Moment où l'on a trempé
le capteur dans de l'eau

Validation du test du capteur d'humidité Funduino

Le capteur Funduino semble plus précis que le capteur FC-28

Ecran LCD & OLED

2 lignes

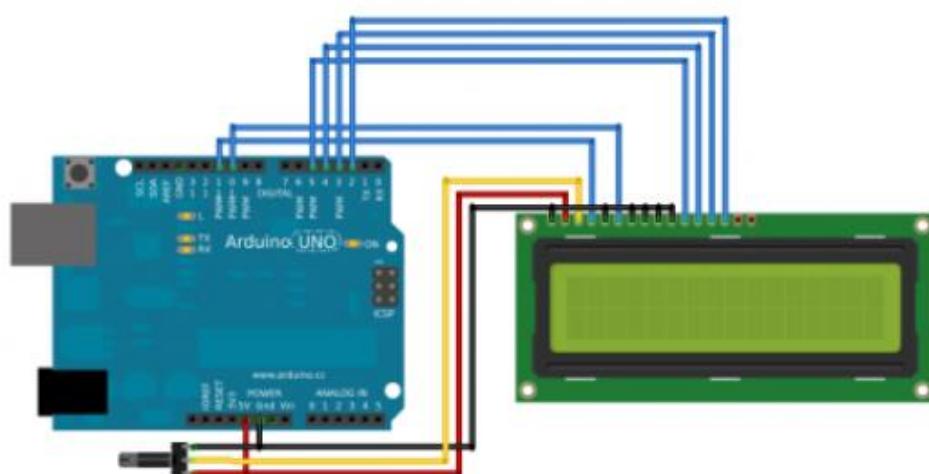
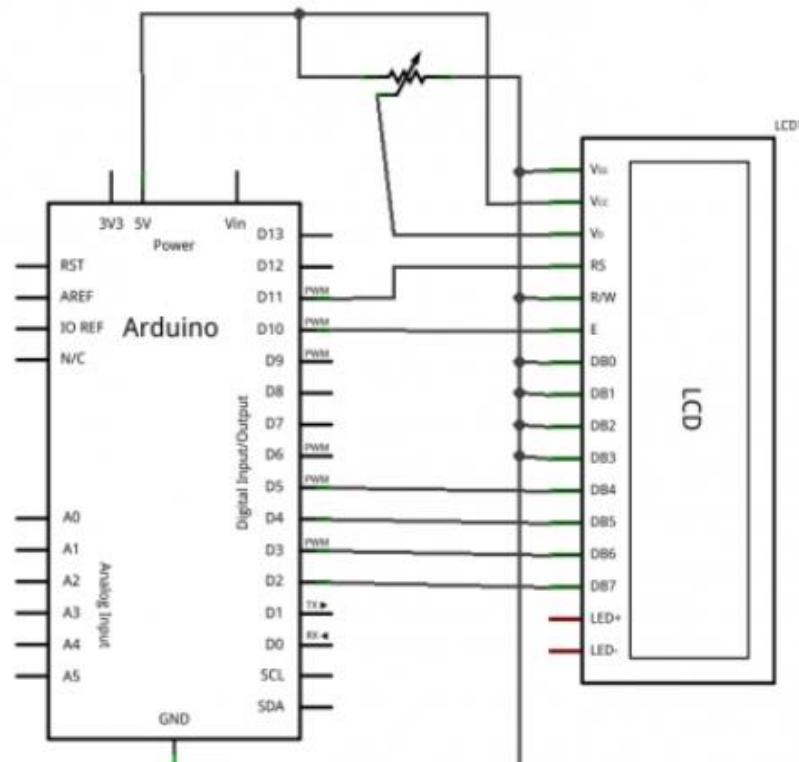
Site détaillé: <http://eskimon.fr/26-arduino-701-les-ecrans-lcd>



N°	Nom	Rôle
1	VSS	Masse
2	Vdd	+5V
3	V0	Réglage du contraste
4	RS	Sélection du registre (commande ou donnée)
5	R/W	Lecture ou écriture
6	E	Entrée de validation
7 à 14	D0 à D7	Bits de données
15	A	Anode du rétroéclairage (+5V)
16	K	Cathode du rétroéclairage (masse)

Test de fonctionnement de l'écran LCD

Le montage à 4 broches de données



Code Arduino pour tester l'écran LCD

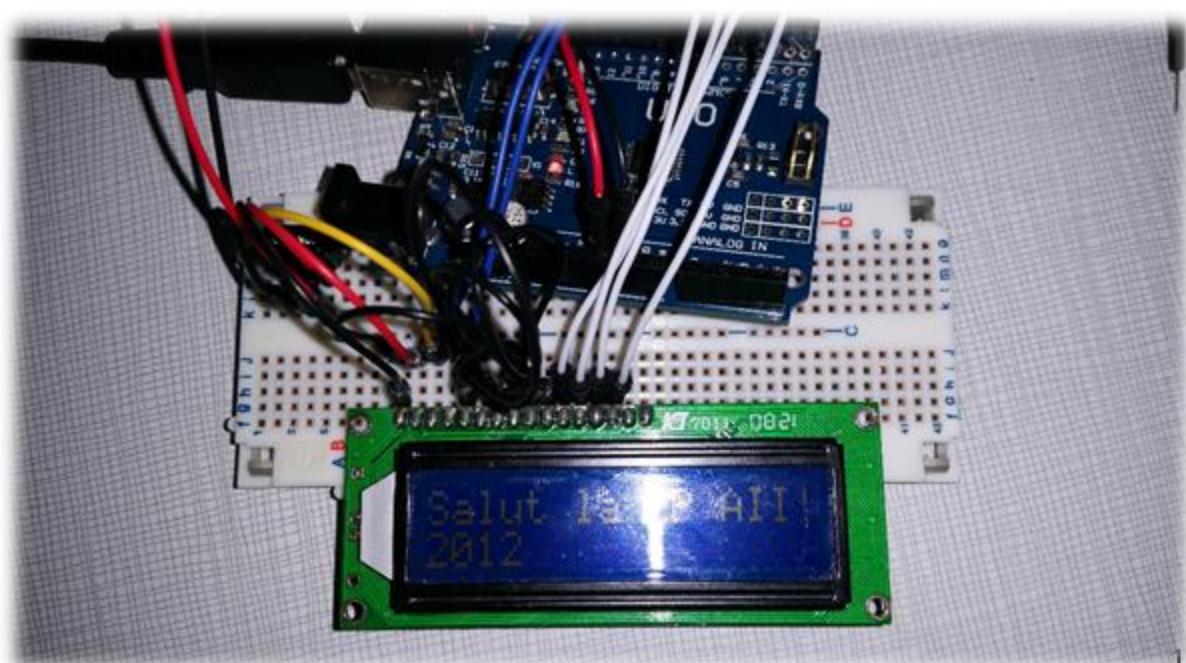
Ce code permet d'afficher du texte (1^{ère} ligne) et un compteur de secondes (2^{ème} ligne)



```
ecran_LCD

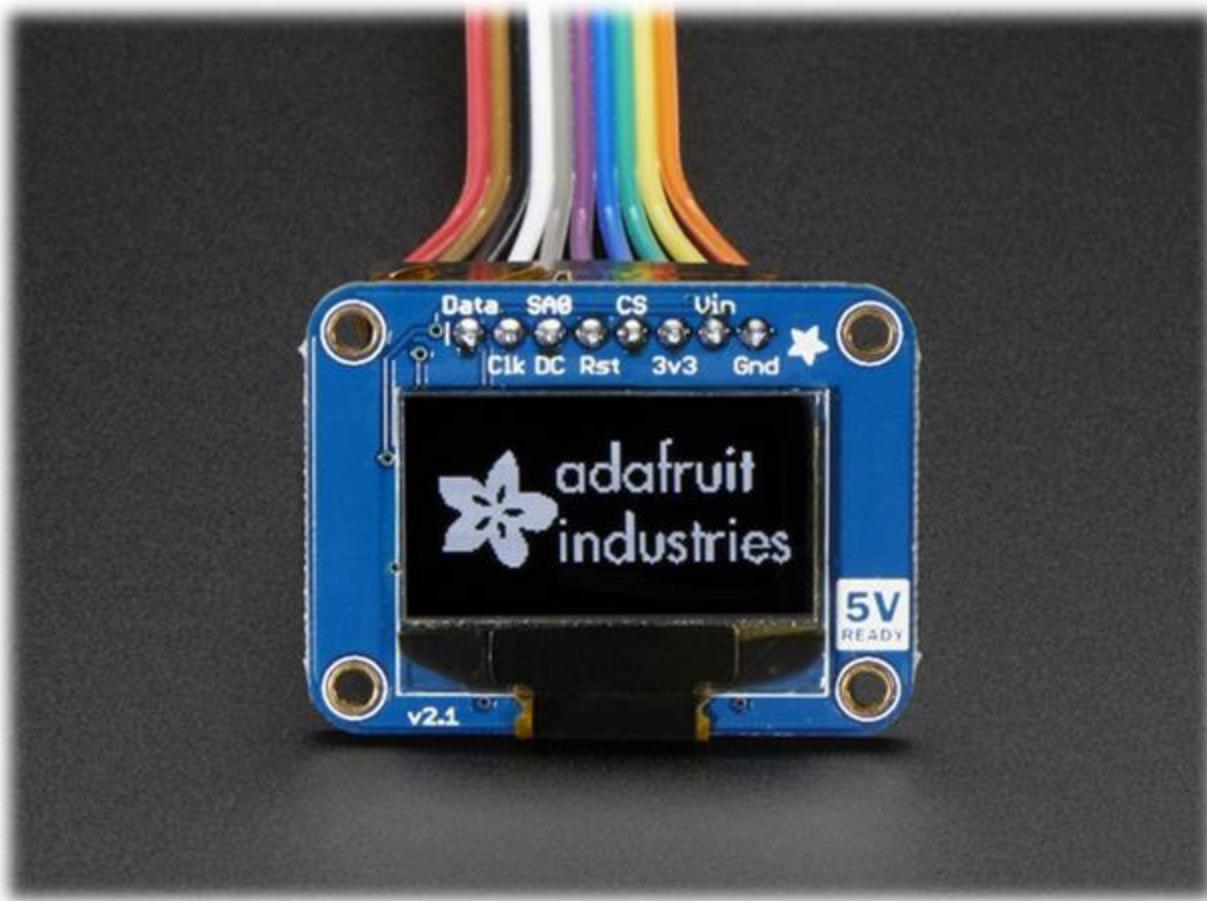
1 #include "LiquidCrystal.h" //ajout de la librairie
2
3 //Vérifier les broches !
4 LiquidCrystal lcd(11,10,5,4,3,2); //liaison 4 bits de données
5
6 void setup()
7 {
8     lcd.begin(16,2); //utilisation d'un écran 16 colonnes et 2 lignes
9     lcd.write("Salut la LP AII!"); //petit test pour vérifier que tout marche
10 }
11
12 void loop() {
13     // set the cursor to column 0, line 1
14     // (note: line 1 is the second row, since counting begins with 0):
15     lcd.setCursor(0, 1);
16     // print the number of seconds since reset:
17     lcd.print(millis()/1000);
18 }
19
```

Enregistrement terminé.



Adafruit

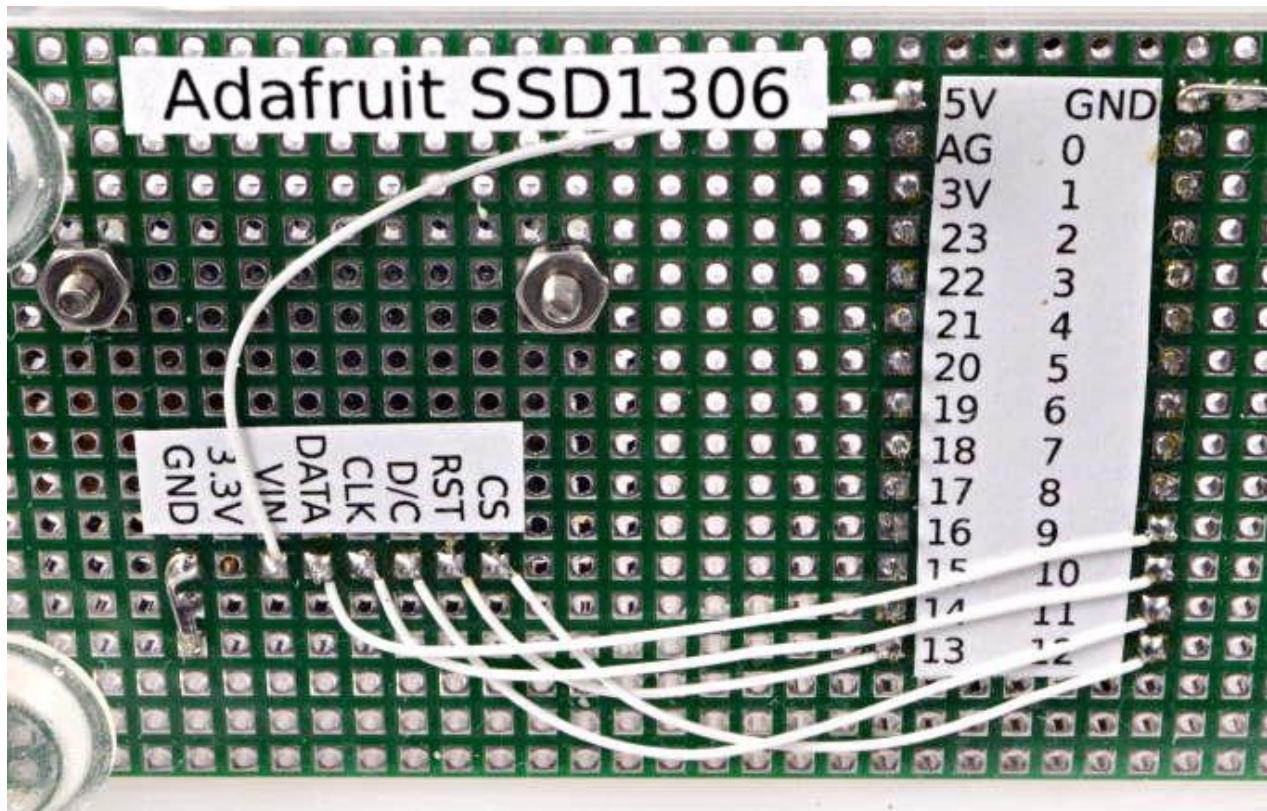
Site constructeur: <https://www.adafruit.com/products/326>



Détails techniques

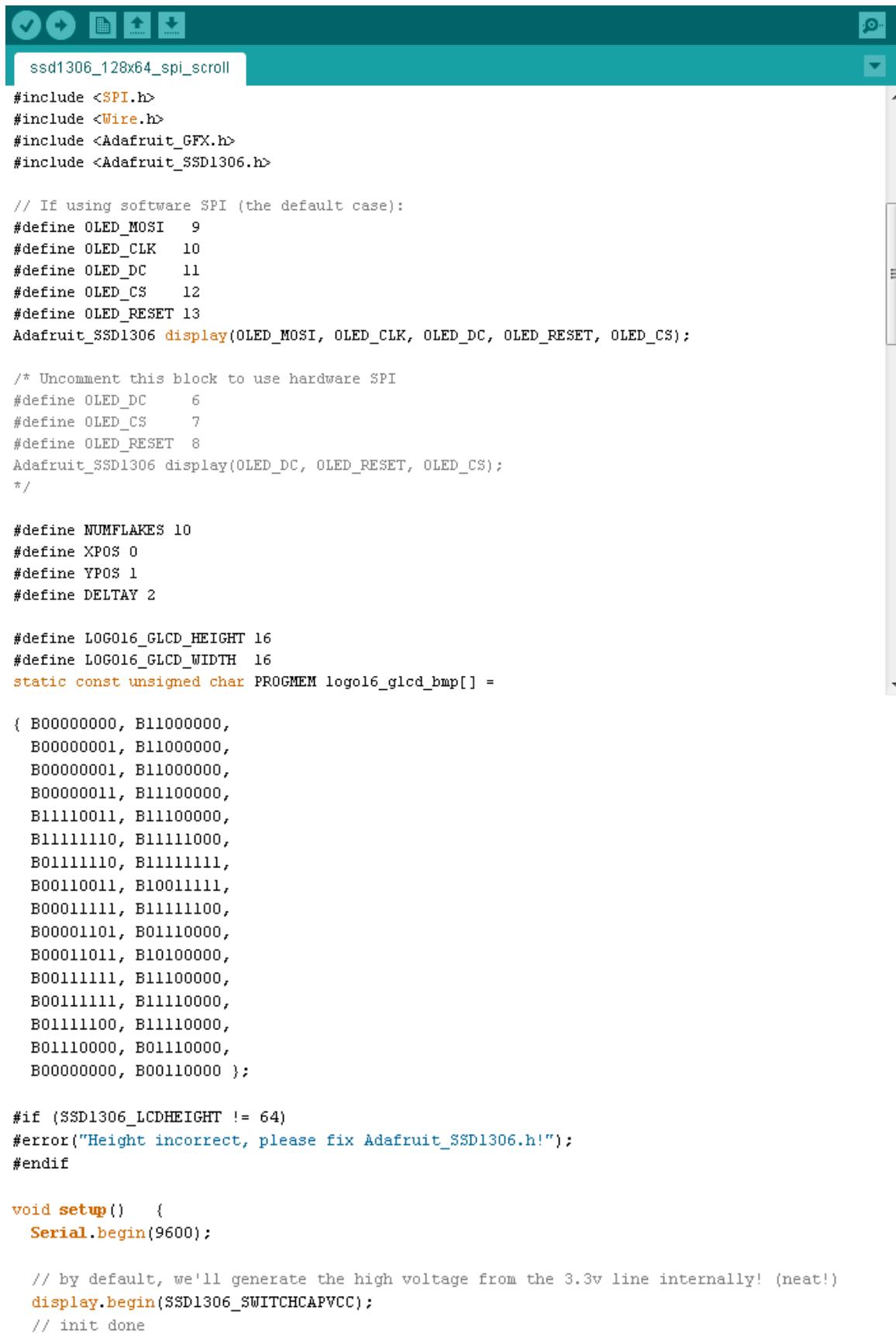
- PCB: 38mm x 29mm (1.5" x 1")
- Screen: 25mm x 14mm
- Thickness: 4mm
- Weight: 8.5g
- Display current draw is completely dependent on your usage: each OLED LED draws current when on so the more pixels you have lit, the more current is used. They tend to draw ~20mA or so in practice but for precise numbers you must measure the current in your usage circuit.
- This board/chip uses I2C 7-bit address between 0x3C-0x3D, selectable with jumpers

Les connexions à établir



Ecran LCD Adafruit	Carte Arduino
VIN	5V
DATA	PIN 9
CLK	PIN 10
D/C	PIN 11
CS	PIN 12
RST	PIN 13

Code Arduino pour tester l'écran OLED



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** ssd1306_128x64_spi_scroll
- Code Area:** The main text area contains the Arduino sketch for the SSD1306 display. It includes header includes for SPI, Wire, Adafruit_GFX, and Adafruit_SSD1306, and defines pins for software SPI (OLED_MOSI=9, OLED_CLK=10, OLED_DC=11, OLED_CS=12, OLED_RESET=13) or hardware SPI (OLED_DC=6, OLED_CS=7, OLED_RESET=8). It also defines NUMFLAKES as 10, and XPOS, YPOS, and DELTAY. A static const unsigned char variable logol16_glcd_bmp[] is defined with a large binary string representing a 16x16 pixel logo. Error handling is provided for LCDHEIGHT != 64. The void setup() function initializes the serial port at 9600 bps and begins the display using the specified parameters.
- Toolbars:** Standard Arduino IDE toolbars for file operations (New, Open, Save, etc.) and a search bar.
- Status Bar:** Not explicitly visible in the screenshot.

```

// Clear the buffer.
display.clearDisplay();
}

void loop() {

    // draw scrolling text
    testscrolltext();
    delay(2000);
    display.clearDisplay();

}

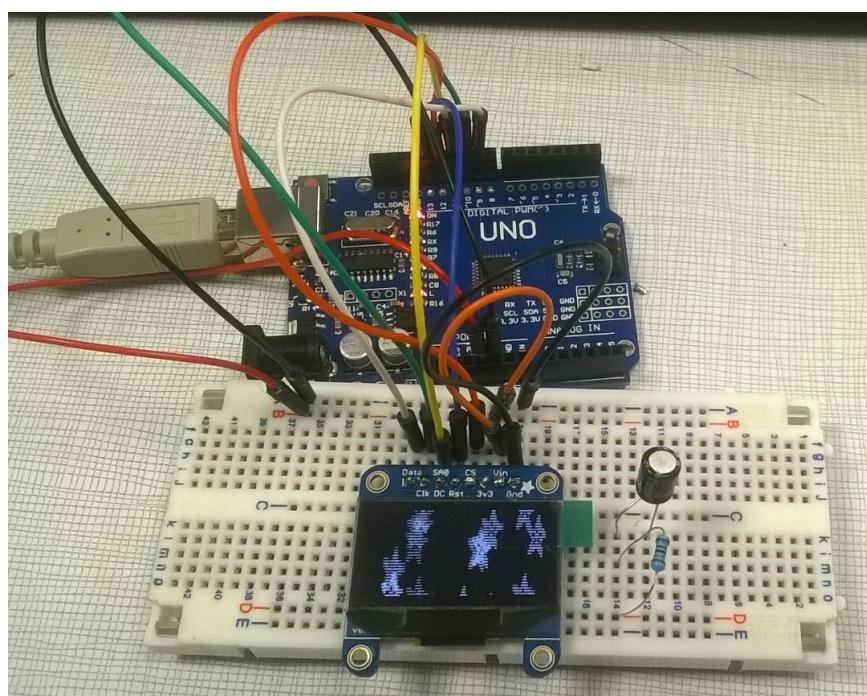
void testscrolltext(void) {
    display.setTextSize(2);
    display.setTextColor(WHITE);
    display.setCursor(10,0);
    display.clearDisplay();
    display.println("scroll");
    display.display();

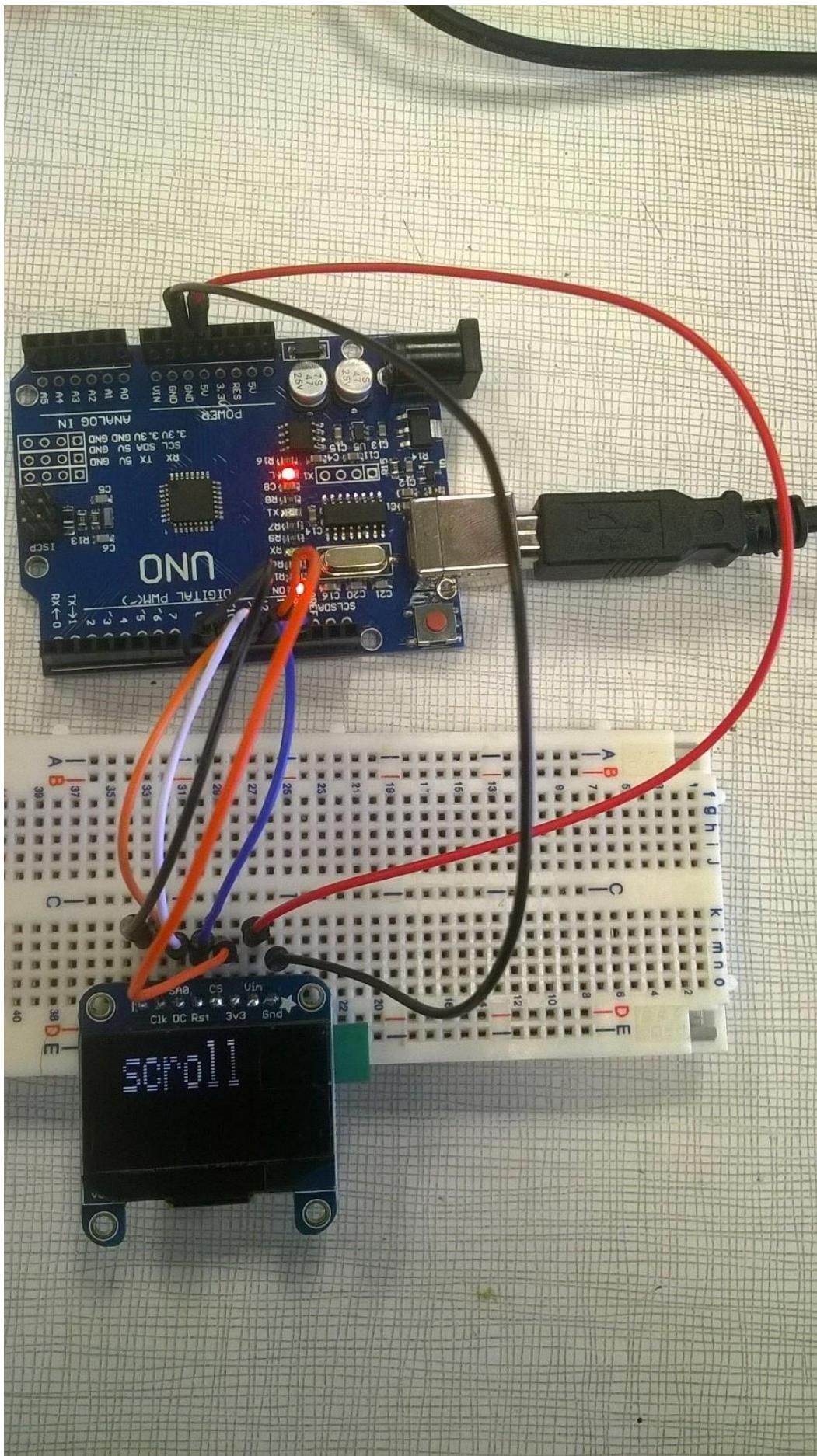
    display.startscrollright(0x00, 0x0F);
    delay(2000);
    display.stopscroll();
    delay(1000);
    display.startscrollleft(0x00, 0x0F);
    delay(2000);
    display.stopscroll();

    delay(1000);
    display.startscrolldiagright(0x00, 0x07);
    delay(2000);
    display.startscrolldiagleft(0x00, 0x07);
    delay(2000);
    display.stopscroll();
}

```

Quelques photos suite au test du code:

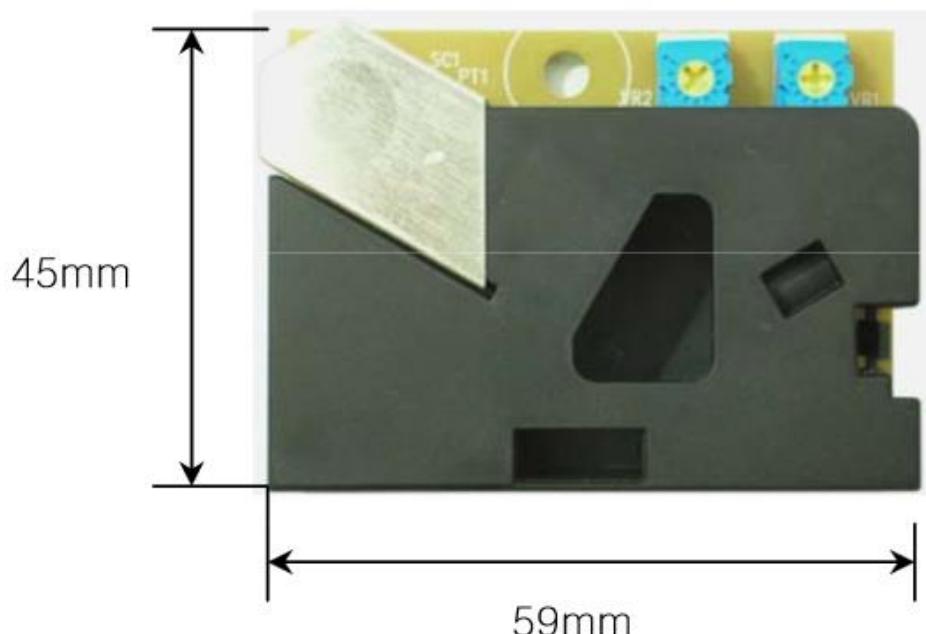




Capteur de poussière Samyoung DSM501

Site détaillé:

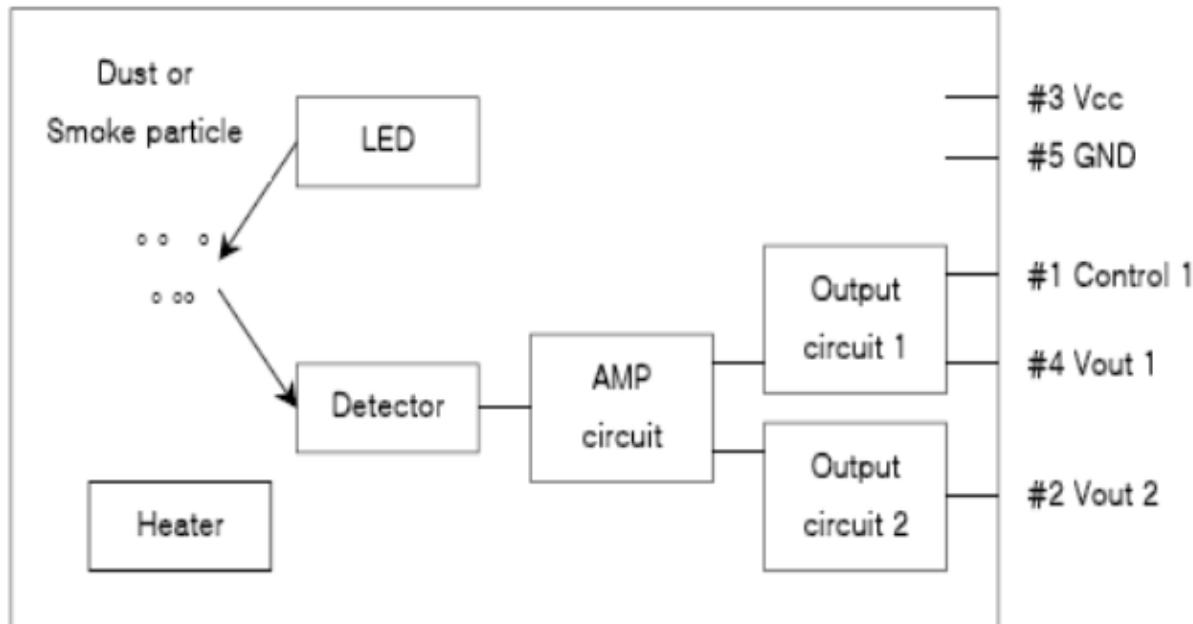
<http://www.samyoungsnc.com/products/3-1%20Specification%20DSM501.pdf>



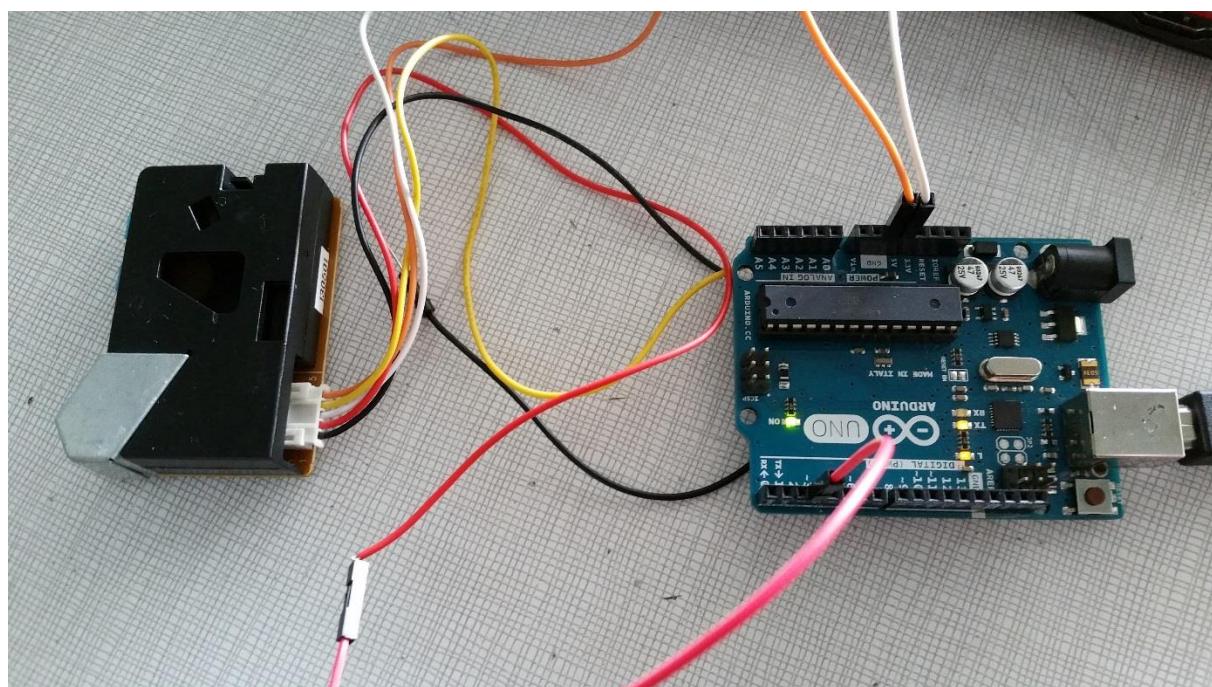
Détails techniques

- Supply voltage: DC 5V ± 10%
- Power consumption: 90mA
- Operating temperature range: -10~ +65°C
- Operating humidity range: 95% RH or less (without dew condensation)
- Recommend storage condition: -20~ +80°C
- Dimension : W59 * H45 * D20 (mm)
- Detectable particle size : approximately $1\mu\text{m}$ (minimum)
- Detectable range of concentration : $0 \sim 1.4 \text{ mg/m}^3$
- Output signal : PWM (pulse width modulation)
- Time for stabilization : 1 minute after power turned on
- Sensor characteristics : To be maintained in between the upper limit and lower limit of the standard dust sensor unit

Branchements du capteur sur la carte Arduino



Pin number	Pin name	Description
#1	Control	Vout 1 control
#2	Vout 2	Vout 2 output (PWM)
#3	Vcc	Positive power supply
#4	Vout 1	Vout 1 output (PWM)
#5	GND	Ground



Code Arduino pour tester le capteur



The screenshot shows the Arduino IDE interface. The top menu bar includes icons for file operations (checkmark, arrow, file, up, down) and a gear icon. The title bar displays the sketch name: "capteur_poussiere". The main area contains the following Arduino code:

```
int pin = 3;

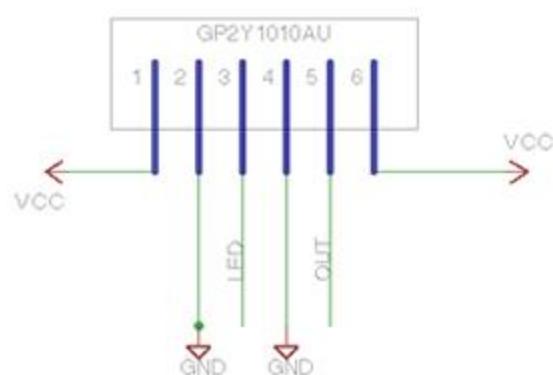
unsigned long duration;

void setup()
{
    pinMode(pin, INPUT);
    Serial.begin(9600);
}

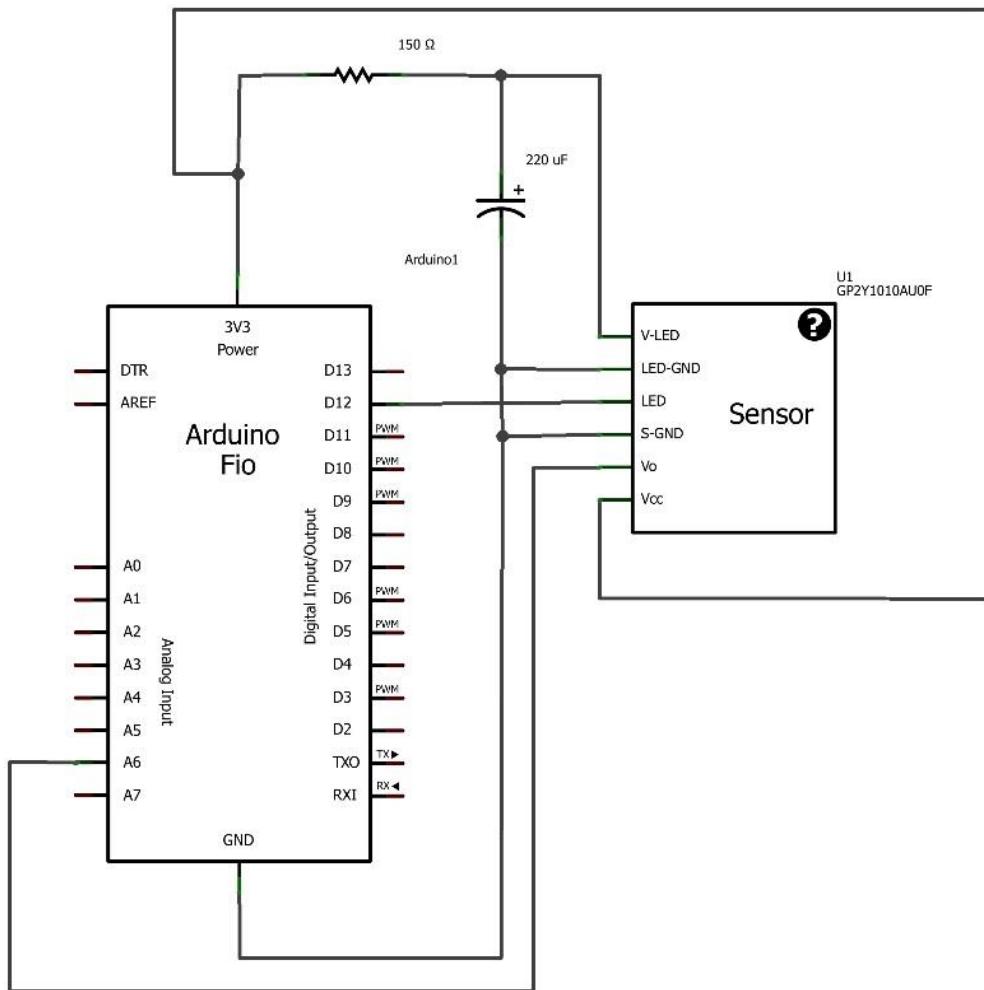
void loop()
{
    duration = pulseIn(pin, HIGH);
    Serial.println(duration);
}
```

The code uses pin 3 as an input for a dust sensor, measures the pulse duration using the pulseIn() function, and prints the result to the Serial monitor.

Détecteur optique de poussière

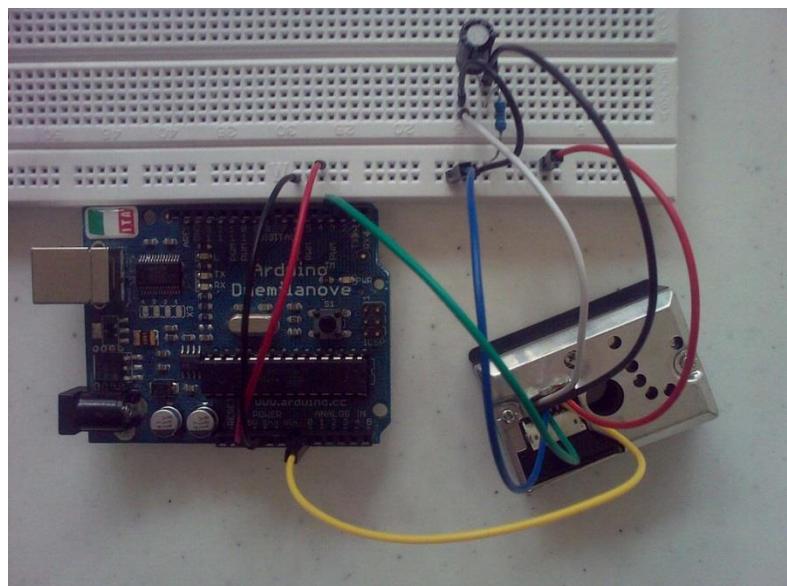


Branchements du capteur sur la carte Arduino



Made with Fritzing.org

	Sensor Pin		Arduino Pin
1	Vled	→	5V (150ohm resistor)
2	LED-GND	→	GND
3	LED	→	Digital pin 2
4	S-GND	→	GND
5	Vo	→	Analog pin 0
6	Vcc	→	5V



Code Arduino pour tester le capteur



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Shows the file name "capteur_poussiere §".
- Toolbar:** Standard Arduino IDE icons for file operations.
- Code Editor:** The main area contains the following Arduino sketch:

```
int dustPin=0;
int dustVal=0;

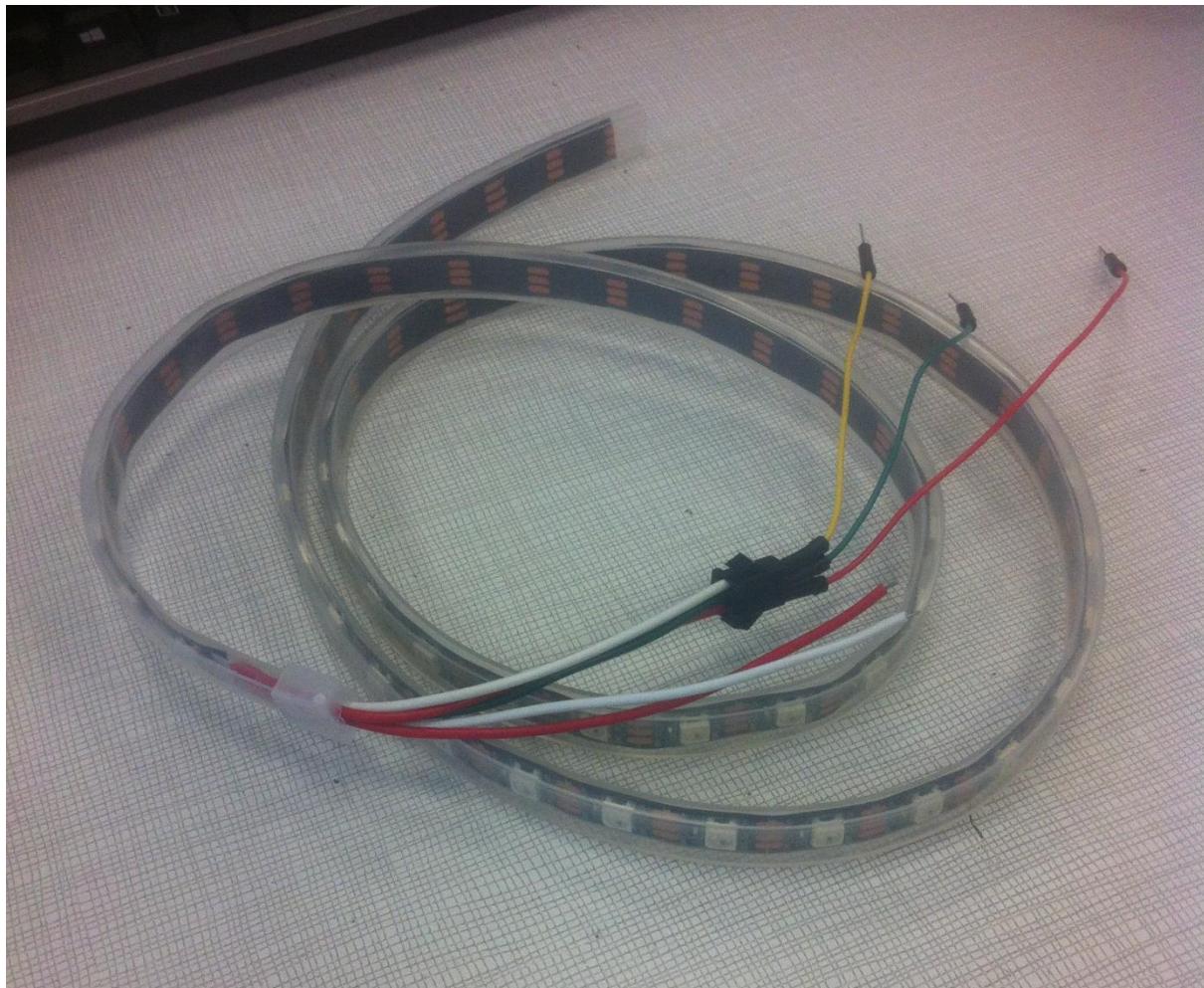
int ledPower=2;
int delayTime=280;
int delayTime2=40;
float offTime=9680;

void setup(){
  Serial.begin(9600);
  pinMode(ledPower,OUTPUT);
  pinMode(4, OUTPUT);
}

void loop(){
  // ledPower is any digital pin on the arduino connected to Pin 3 on the sensor
  digitalWrite(ledPower,LOW); // power on the LED
  delayMicroseconds(delayTime);
  dustVal=analogRead(dustPin); // read the dust value via pin 5 on the sensor
  delayMicroseconds(delayTime2);
  digitalWrite(ledPower,HIGH); // turn the LED off
  delayMicroseconds(offTime);

  delay(3000);
  Serial.println(dustVal);
}
```

Bandeau LED



Branchements du bandeau sur la carte Arduino

Bandeau LED	Carte Arduino
Pin Rouge	5V
Pin Jaune	GND
Pin Vert	Digital Output

Code Arduino pour tester le capteur



```

simple

// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// released under the GPLv3 license to match the rest of the AdaFruit NeoPixel library

#include "Adafruit_NeoPixel.h"
#ifndef __AVR__
  #include <avr/power.h>
#endif

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define PIN          6

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS    62

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.
// Note that for older NeoPixel strips you might need to change the third parameter--see the strandtest
// example for more information on possible values.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

int delayval = 500; // delay for half a second

void setup() {
  // This is for Trinket 5V 16MHz, you can remove these three lines if you are not using a Trinket
/*#if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif*/
  // End of trinket special code

  pixels.begin(); // This initializes the NeoPixel library.
}

void loop() {

  // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to the count of pixels minus one.
  while(1){

    for(int i=0;i<NUMPIXELS;i++){

      // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
      pixels.setPixelColor(i, pixels.Color(255,0,0)); // Moderately bright green color.

      pixels.show(); // This sends the updated pixel color to the hardware.

      delay(delayval); // Delay for a period of time (in milliseconds).

    }

    for(int i=NUMPIXELS;i>=0;i--){

      // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
      pixels.setPixelColor(i, pixels.Color(0,255,0)); // Moderately bright green color.

      pixels.show(); // This sends the updated pixel color to the hardware.

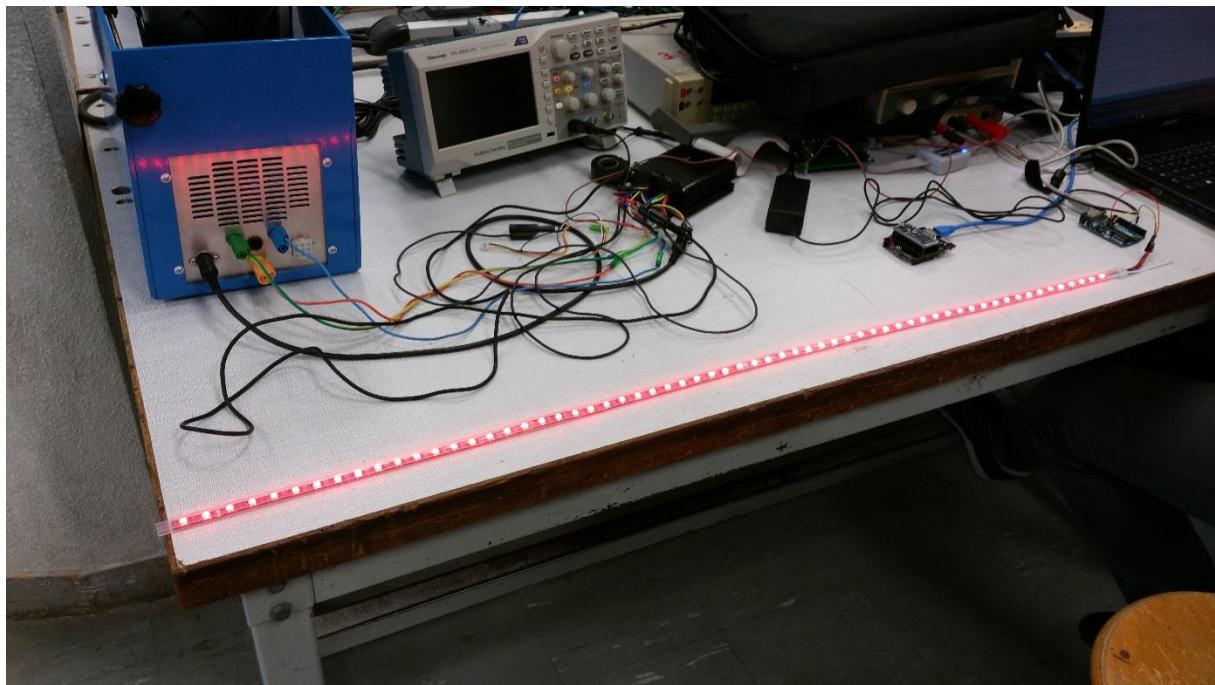
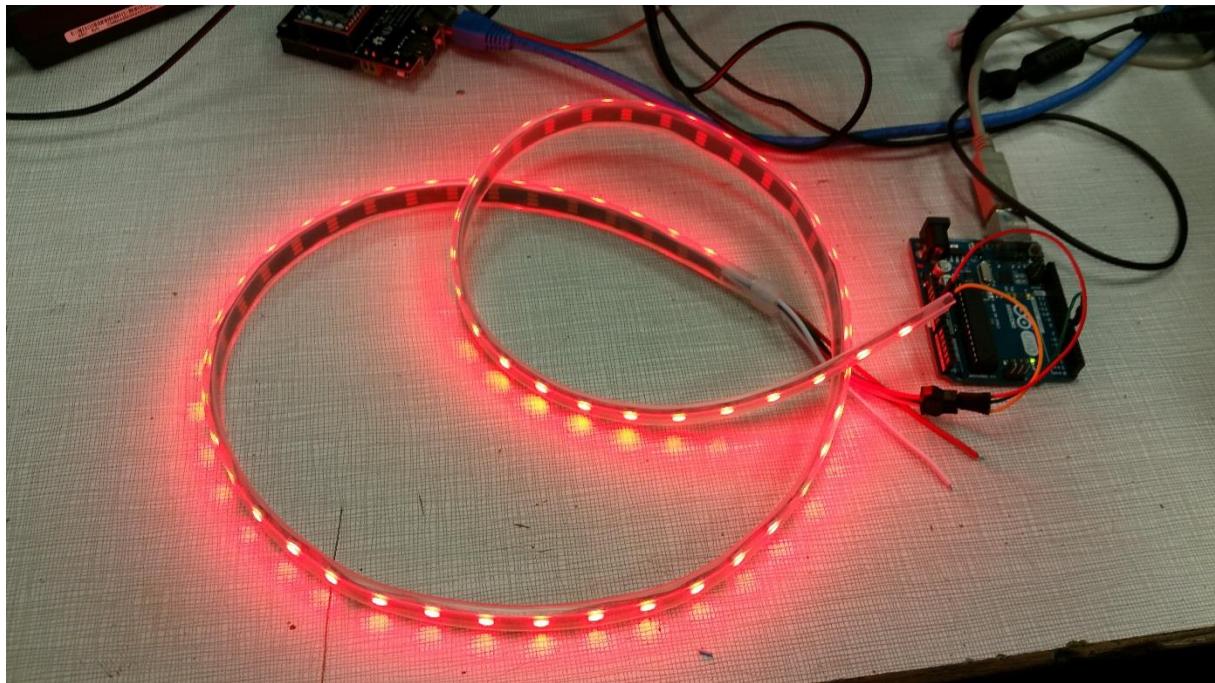
      delay(delayval); // Delay for a period of time (in milliseconds).

    }

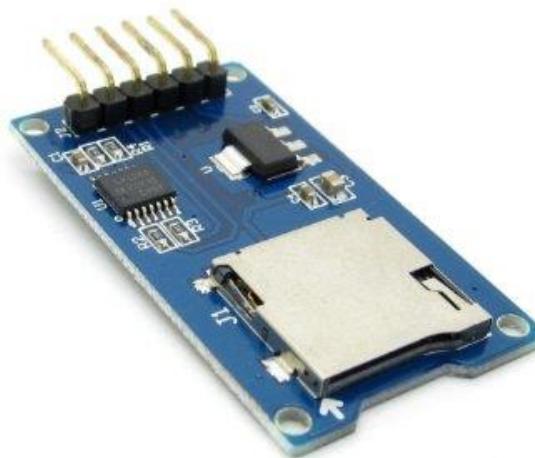
  }
}
}
}

```

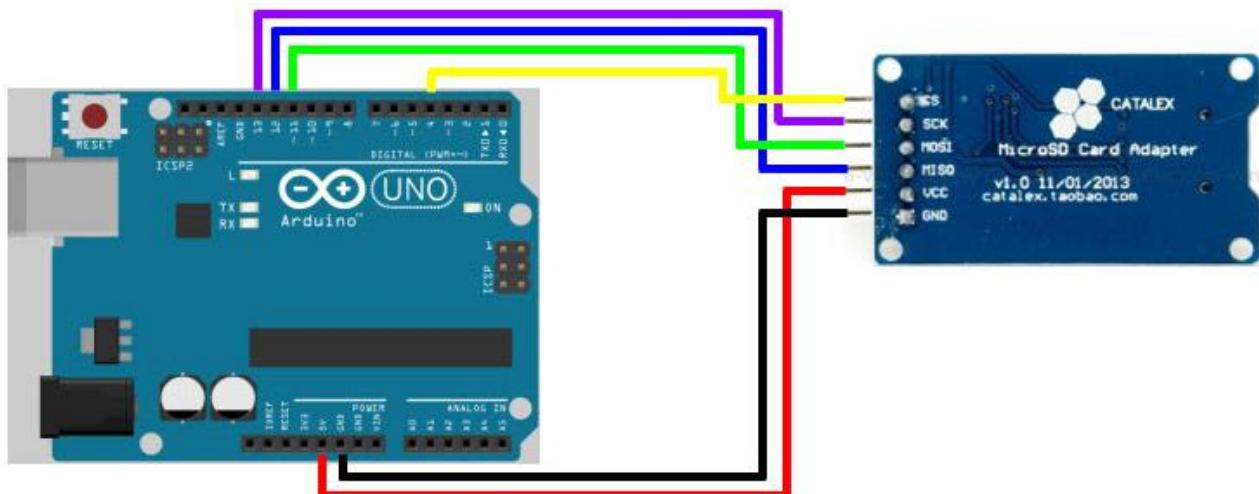
Les résultats du test en photos



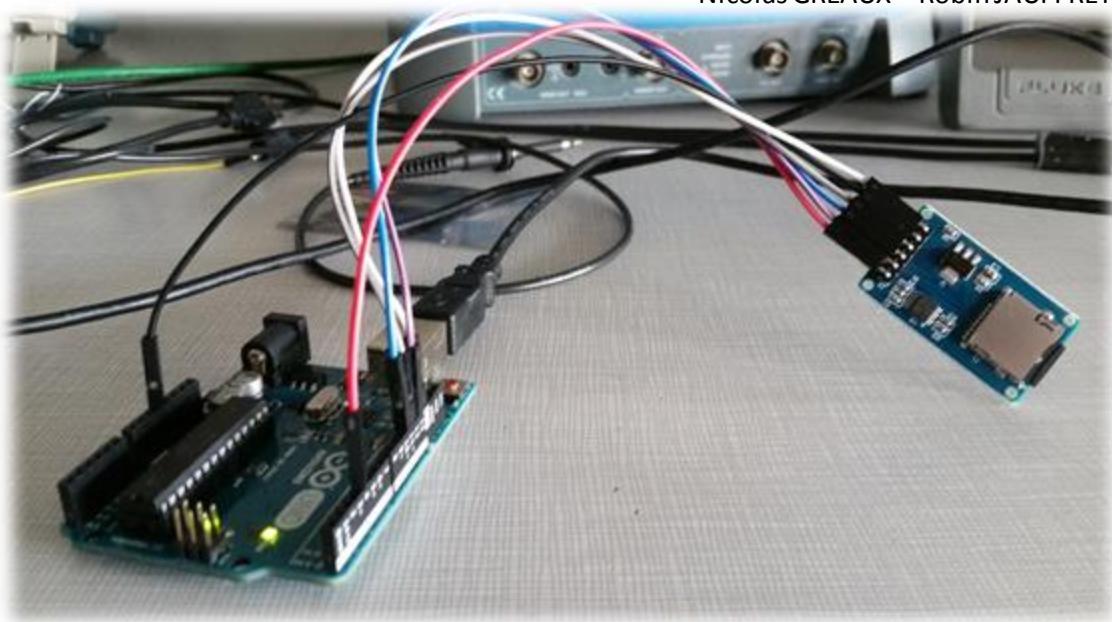
Adaptateur micro SD pour Arduino



Branchements de l'adaptateur sur la carte Arduino



Adaptateur micro SD	Carte Arduino
CS	PIN 4
SCK	PIN 13
MOSI	PIN 11
MISO	PIN12
VCC	5V
GND	GND



Code Arduino pour tester l'adaptateur

```

CardInfo

#include <SPI.h>
#include <SD.h>

// set up variables using the SD utility library functions:
Sd2Card card;
SdVolume volume;
SdFile root;

// change this to match your SD shield or module;
// Arduino Ethernet shield: pin 4
// Adafruit SD shields and modules: pin 10
// Sparkfun SD shield: pin 8
const int chipSelect = 4;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.print("\nInitializing SD card...");

  // we'll use the initialization code from the utility libraries
  // since we're just testing if the card is working!
  if (!card.init(SPI_HALF_SPEED, chipSelect)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("* is a card inserted?");
    Serial.println("* is your wiring correct?");
    Serial.println("* did you change the chipSelect pin to match your shield or module?");
  }
}

```

```

    return;
} else {
  Serial.println("Wiring is correct and a card is present.");
}

// print the type of card
Serial.print("\nCard type: ");
switch (card.type()) {
  case SD_CARD_TYPE_SD1:
    Serial.println("SD1");
    break;
  case SD_CARD_TYPE_SD2:
    Serial.println("SD2");
    break;
  case SD_CARD_TYPE_SDHC:
    Serial.println("SDHC");
    break;
  default:
    Serial.println("Unknown");
}

// Now we will try to open the 'volume'/'partition' - it should be FAT16 or FAT32
if (!volume.init(card)) {
  Serial.println("Could not find FAT16/FAT32 partition.\nMake sure you've formatted the card");
  return;
}

// print the type and size of the first FAT-type volume
uint32_t volumesize;
Serial.print("\nVolume type is FAT");

Serial.println(volume.fatType(), DEC);
Serial.println();

volumesize = volume.blocksPerCluster();           // clusters are collections of blocks
volumesize *= volume.clusterCount();             // we'll have a lot of clusters
volumesize *= 512;                               // SD card blocks are always 512 bytes
Serial.print("Volume size (bytes): ");
Serial.println(volumesize);
Serial.print("Volume size (Kbytes): ");
volumesize /= 1024;
Serial.println(volumesize);
Serial.print("Volume size (Mbytes): ");
volumesize /= 1024;
Serial.println(volumesize);

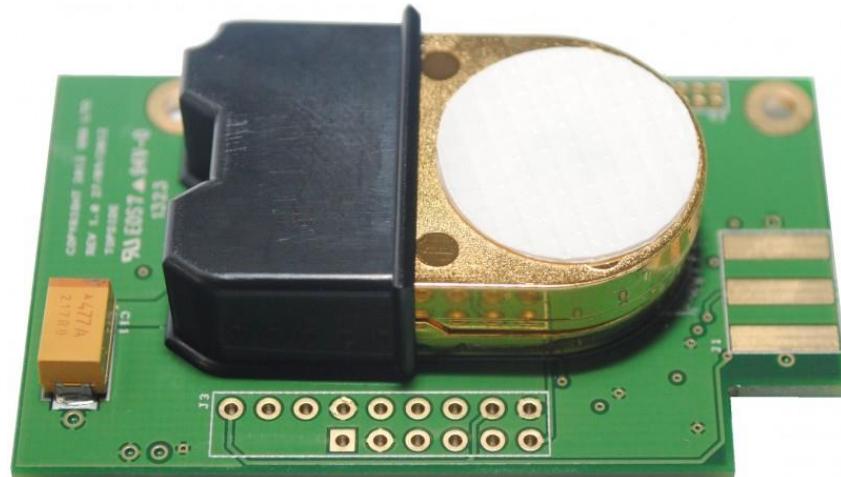
Serial.println("\nFiles found on the card (name, date and size in bytes): ");
root.openRoot(volume);

// list all files in the card with date and size
root.ls(LS_R | LS_DATE | LS_SIZE);
}

void loop(void) {
}

```

Capteur de CO₂ MISIR-5000



Specifications

- Low Power 20mW
- 0-5,000 ppm Measurement range
- Low noise measurement (<10ppm)
- 3.3V to 5V supply
- Analog Output
- Automatic Calibration

General Performance

- Warm-up Time: < 20s
- Operating Conditions: 0°C to 50°C, 0-95% HR non-condensing

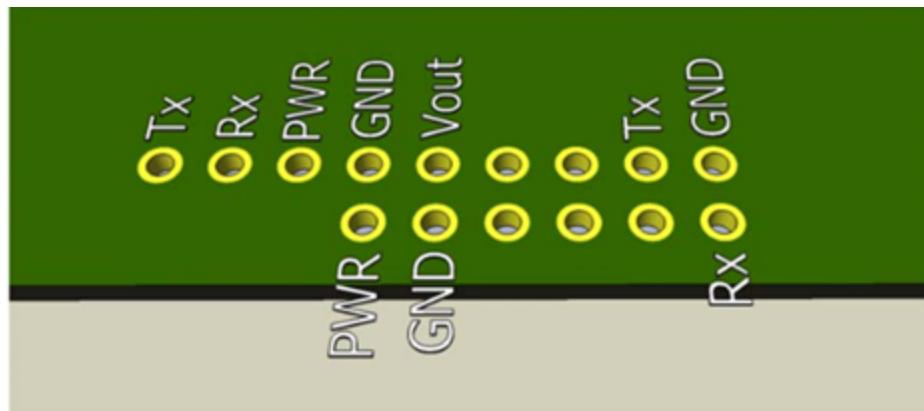
CO₂ Measurement

- Sensing Method: NDIR with Gold-plated optics
- Sample Method: Diffusion
- Measurement Range: 0-5,000ppm
- Accuracy: ±50 ppm ± 3% of reading
- Pressure Dependence: 0.13% of reading per mm Hg
- Calibration: Automatic Baseline (ABC) Calibration
- Response Time: 2 Minutes

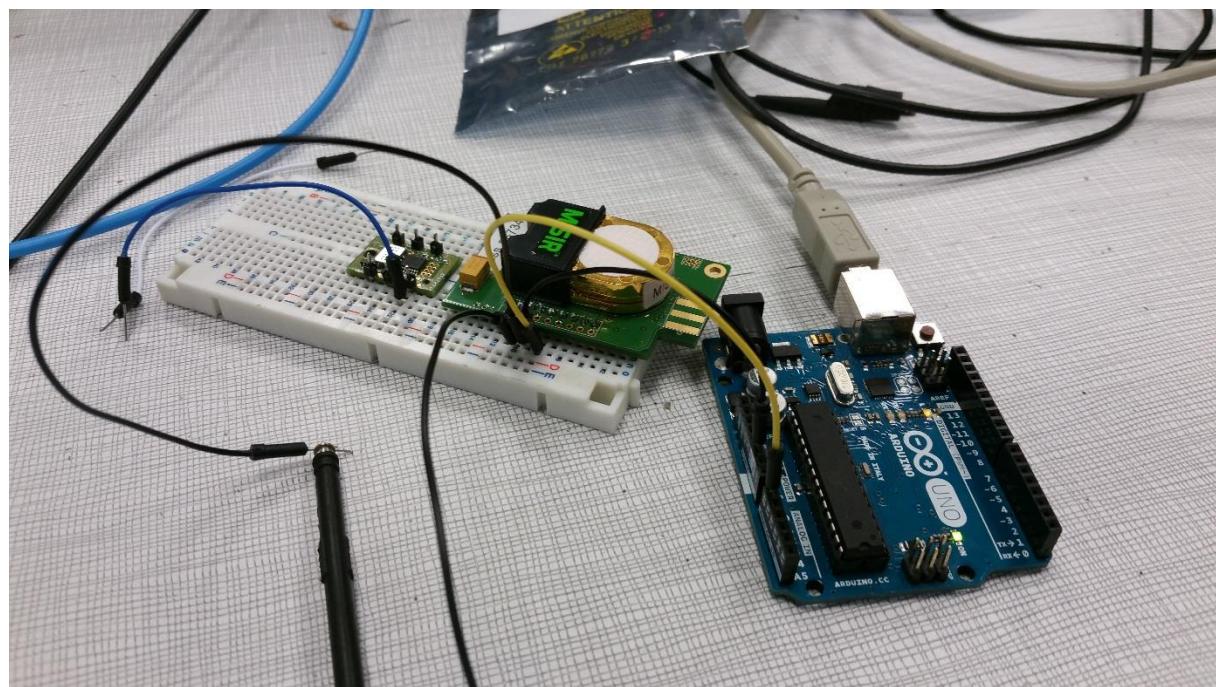
Electrical/Mechanical

- Power Input: 3.25-4.25V (3.3V recommended)
- Peak Current: 150mA
- Average Current: 6mA

Branchements du capteur sur la carte Arduino



Pin	Description	Comment
PWR	Vsupply	3.3V to 5.25V
GND	0V	Only one GND should be connected
Rx	Sensor receive line (UART)	5V Tolerant
Tx	Sensor transmit line (UART)	Vout = Vsupply
Vout	Voltage output	Vout proportional to CO2 concentration Full scale = Vsupply



Code Arduino pour tester le capteur



The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Tools, Sketch, Help, and a language selection dropdown. Below the menu is a toolbar with icons for Save, Run, Stop, Upload, and Download. The main workspace has a teal header bar with the text "misir §". The code area contains the following Arduino sketch:

```
// the setup routine runs once when you press reset:  
void setup() {  
    // initialize serial communication at 9600 bits per second:  
    Serial.begin(9600);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
    // read the input on analog pin 0:  
    int sensorValue = analogRead(A0);  
    // print out the value you read:  
    Serial.println("-----");  
    Serial.println(sensorValue);  
    Serial.println("-----");  
    delay(1000);          // delay in between reads for stability  
}
```

MiCS-VZ-89

Site détaillé:

<http://www.cdiweb.com/datasheets/e2v/MiCS-VZ-86%20and%20VZ-89%20rev%204.pdf>



Performance

Detection Method	Semiconductor gas sensor, detecting a wide range of VOCs
Monitoring Range	400-2000 ppm equivalent CO ₂ 0-1000 ppb isobutylene equivalent tVOCs
PWM Output (VZ-86)	Pin 1 : TTL output 30Hz, Range 5...95%, duty cycle 5V
I2C Output (VZ-89)	Pin 2 and 4 ; see VZ I2C SPEC rev A for details of operation
Response Time	Equivalent to conventional NDIR-CO ₂ sensors < 5 seconds for tVOC
Refresh Output Frequency	1 Hz

Operation

Supply Voltage	5V DC, regulated +/- 0.25V for F version 3.3V DC regulated +/- 0.25V for T version
Operating Power	150 mW
Warm-up Time	15 min
Operating Temperature	0°C to 50°C
Operating Humidity	0%RH to 95%RH (non condensing)
Storage Temperature	-40°C to 80°C
Storage Humidity	0%RH to 95%RH (non condensing)

MiCS-VZ-89 Output

During “Functional Test Mode” only “Raw sensor” and “VOC_short” data are available. “VOC_short” is an image of sensor reactivity and can then be used for functional test.

Out of this initial period, the device will have the I2C data CO₂ equivalent [ppm] and tVOC equivalent referred to the isobutylene sensitivity unit [ppb].

D1:Data_byte_1: CO₂_equ: [13...242] -> CO₂_equ [ppm] = (D1 -13) * (1600/229) + 400

D2: Data_byte_2: VOC_short [13...242]

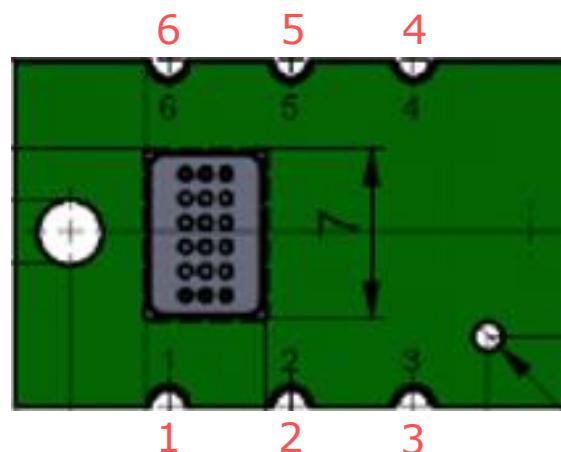
D3: Data_byte_3: tVOC: [13...242] -> tVOC [ppb] = (D3 -13) * (1000/229)

D4: Data_byte_4: Raw sensor first byte (LSB)

D5: Data_byte_5: Raw sensor second byte

D6: Data_byte_6: Raw sensor third byte (MSB) -> Resistor value [Ω] = 10*(D4 + (256*D5) + (65536*D6))

Branchements du capteur sur la carte Arduino



Pin Connection VZ-89

6:+ 5V/3.3V for T version	5: NC	4: SDA
1: NC	2: SCL	3: GND

Code Arduino pour tester le capteur

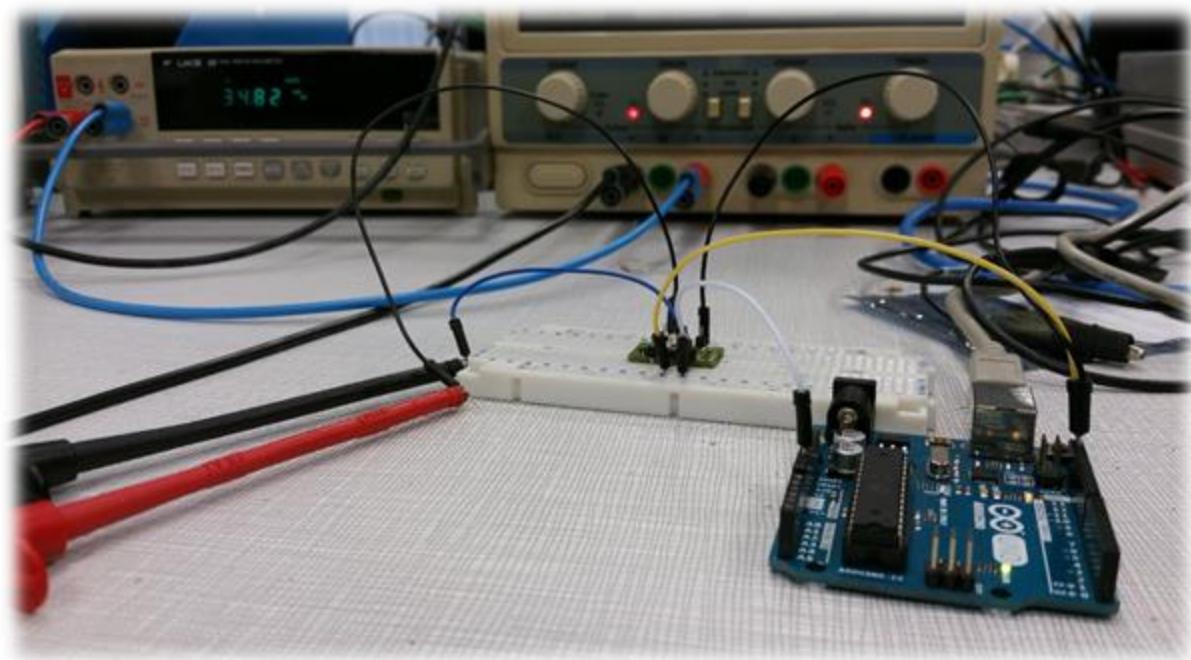
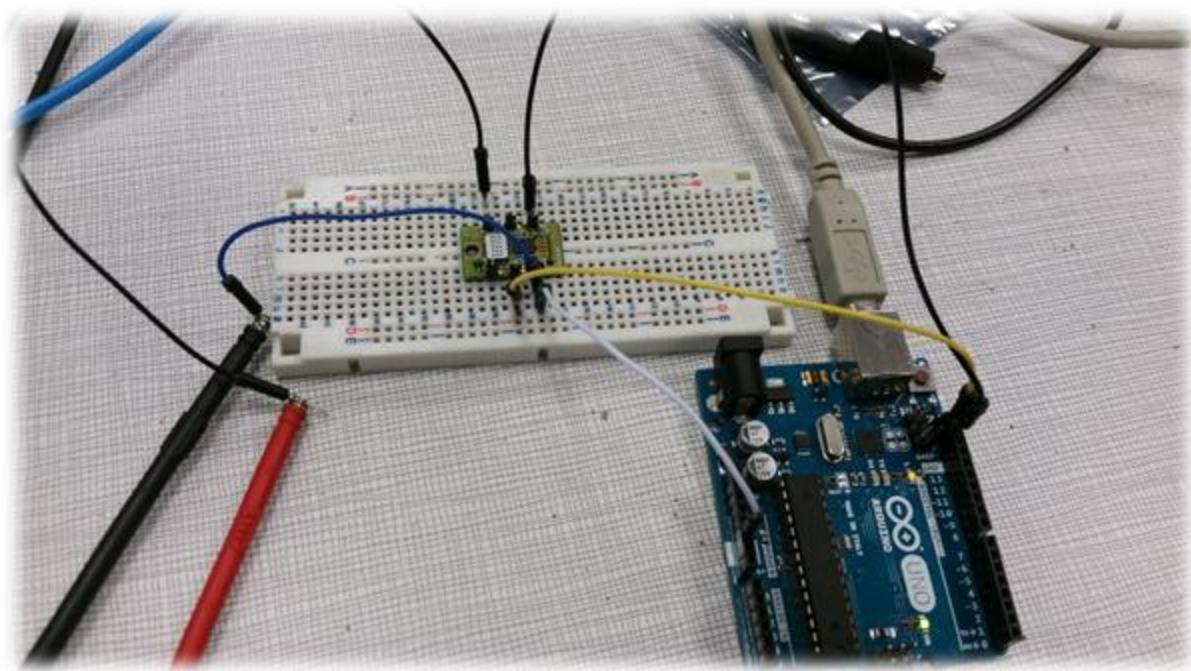
```
#include <Wire.h>

char tabValeur[6];
int resultat = 0;
int j = 0;

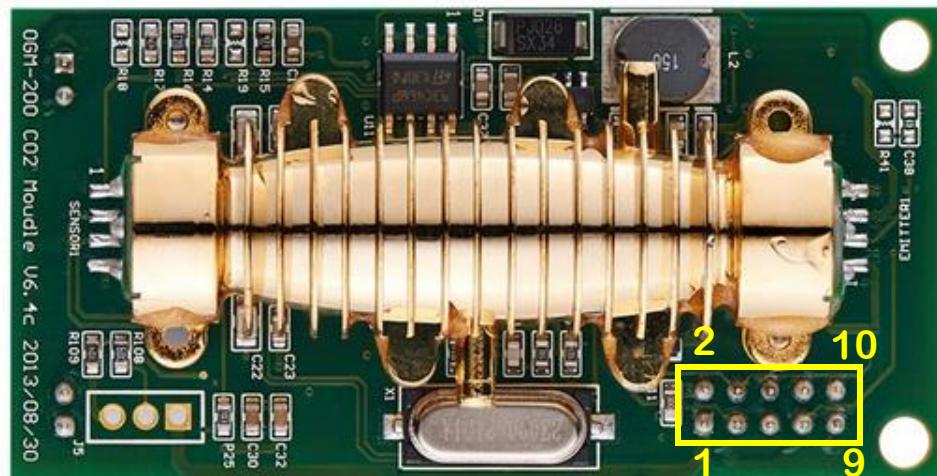
void setup() {
    // put your setup code here, to run once:
    Wire.begin();
    Serial.begin(9600);
    Wire.beginTransmission(0x70);
    Wire.write(9);
    Wire.endTransmission();
}

void loop() {
    // put your main code here, to run repeatedly:
    j = 0;
    Wire.requestFrom(0x70, 6);
    while(Wire.available())
    {
        int i = Wire.read();
        Serial.println(i);
        tabValeur[j] = i;
        j++;
    }
    Serial.println("-----");
    delay(1000);
    valeurCapteur(tabValeur);
}

void valeurCapteur(char *tabValeurs)
{
    resultat = (tabValeurs[0] - 13)*(1600/229) + 400;
    Serial.println(resultat);
    Serial.println("-----");
}
```



OGM 200



Item		Specification
General	Operating Temperature	5°C ~ 50°C ^{*2} (non condensing)
	Operating Humidity	0 ~ 95% RH(Non-condensing)
	Operating Environment	Residential, Commercial spaces
	Storage Temperature	-20°C ~ 70°C (non condensing)
CO ₂ Measurement	Sensing Method	NDIR(Non-dispersive Infrared) dual channels
	Measurement Range	0 to 2,000 ppm (standard), or 0 to 5,000 ppm, other range please contact OST
	Precision	±(50ppm +3% of measured value) ^{*1}
	Warm-up Time	< 3 minutes
	Response Time (63%)	< 30 seconds (diffusion)
	Sampling Interval	~ 3 seconds
Electrical data	Power Input	5~6 VDC @500mA peak, 39mA average, minimum input 4.5V, Max 6.5V DC
	Output connector	10 pins
Output interface	Digital or analog Output	RS485 or UART, PWM analog output, Open drain threshold output (20V, 1A)
LED Interface	Two color LED drivers	Open drain LED drivers, need external current limit resistor, I _{LED} <20ma. (Common anode bi-color LEDs)
Altitude Compensation	Compensate the pressure difference due to altitude of CO ₂ module	Height in meter or feet, set from PC interface by using OST installer software.

Branchements du capteur sur la carte Arduino

Pin 1: +5~6V DC	Pin 2: GND
Pin 3: /RST(low active, 3.3V normal)	Pin 4: Reserved (Factory use)
Pin 5: Reserved (Factory use)	Pin 6: PWM analog output
Pin 7: THSOUT (open drain)	Pin 8: Reserved (Factory use)
Pin 9: RS485+ / TXD (3.3V logic)	Pin 10: RS485- / RXD (3.3V logic)

Normalement, la pin J1-6 est une PIN PWM Digitale de sortie qui sert également de sortie Analogue.

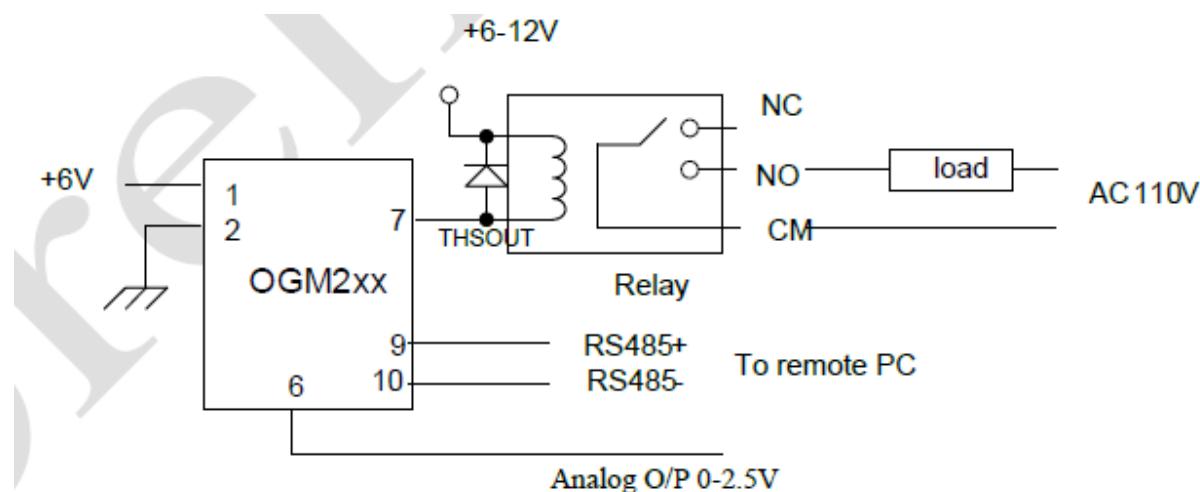
La sortie PWM a 12 bits de résolution avec une puissance maximale de 2.5V. Par défaut, la sortie analogue est mise à 0.5V pour 500ppm, et 2.0V pour 2000ppm.

Pour le modèle 5000ppm, la sortie analogue est à 0.5V pour 1000ppm et 2.5V pour 5000ppm.

Pour les modèles de plus haute concentration, tel que le 7S (7000ppm) ou 9S (9000ppm) et les modèles 5%, la sortie analogue est mise à 2.5V pour l'échelle plaine.

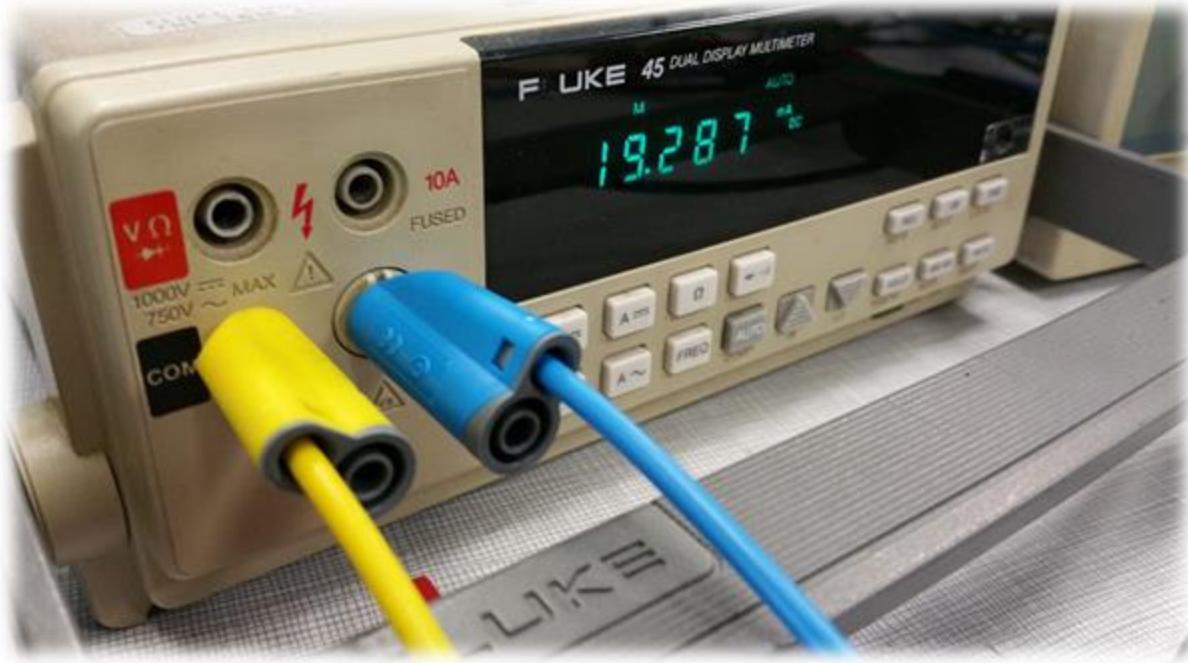
L'utilisateur peut programmer l'offset et l'échelle de la sortie analogue en utilisant l'OST donné avec le programme d'installation.

L'offset est utilisé pour interface 4-20mA.



Comparaison consommation des 3 capteurs CO2

OGM 200: 19mA



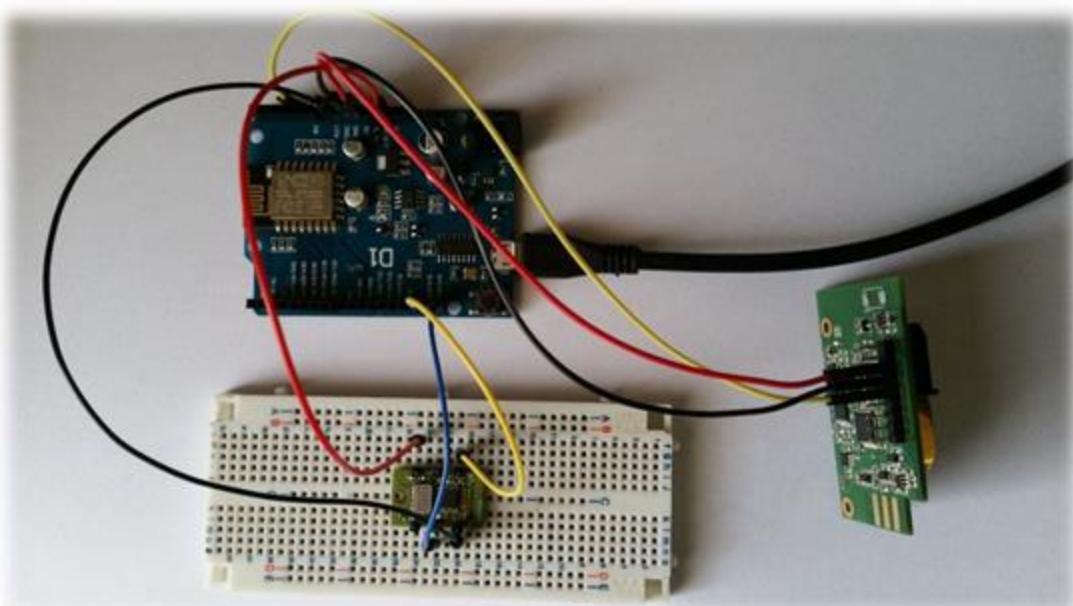
Misir-5000: 5mA



MiCS-VZ-89: 35mA



Datalogger sur WeMos D1



Pour récupérer les informations provenant des capteurs de CO₂ :

Ces informations indiquent le taux de CO₂ en PPM (partie par million) mesuré sur une longue période.

Afin de déterminer l'efficacité de chaque capteur, toutes les valeurs relevées du taux de CO₂ sont reportées sur un graphique. Pour cela un datalogger (enregistreur de données) est utilisé.

Un datalogger est un dispositif électronique, en général programmable qui enregistre des valeurs de mesures individuelles et des séries de mesures sur une longue période (pouvant couvrir plusieurs mois). Les valeurs sont automatiquement mesurées et enregistrées sur un serveur qui les affiche sous la forme d'un graphe.

Il est utilisé comme datalogger, une carte programmable WeMos D1 qui possède un ESP8266 et qui permet la connexion Wifi. Sur cette carte, il est possible de brancher plusieurs capteurs, pour notre projet ce sont les capteurs de taux de CO₂. Puis, ces données sont traitées si besoin avant de les envoyer sur le serveur Thingspeak lequel va permettre d'afficher l'évolution du taux de CO₂ en fonction du temps sous la forme d'un graphe.

Pour utiliser Thingspeak, il faut obligatoirement se rendre sur le site <https://thingspeak.com/>, et créer un compte.

Pour visualiser les résultats sous forme de graphe un « Channel » doit être créé.

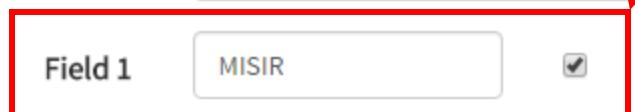
Dans un « Channel », il est possible de donner des noms aux différents graphes ex : « Field1 s'appelle MISIR »

New Channel

Name

Description

Field 1



On peut modifier les paramètres d'affichage (nombre de jours, nombre de résultats, couleur de fond, etc....) en cliquant sur le stylo en haut à droite de chaque graphe.



Field 1 Chart Options

Title: <input type="text"/>	Timescale: <input type="text"/>
X-Axis: <input type="text"/>	Average: <input type="text"/>
Y-Axis: <input type="text"/>	Median: <input type="text"/>
Color: #d62020	Sum: <input type="text"/>
Background: #ffffff	Rounding: <input type="text"/>
Type: line <input type="button" value="▼"/>	Data Min: <input type="text"/>
Dynamic?: true <input type="button" value="▼"/>	Data Max: <input type="text"/>
Days: 1 <input type="text"/>	Y-Axis Min: <input type="text"/>
Results: 960 <input type="text"/>	Y-Axis Max: <input type="text"/>

Save

Cancel

Le programme contenu dans la carte permet d'utiliser l'ESP8266 et donc d'avoir une connexion Wifi.

```
//Connexion WIFI
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
///////////
}
```

Ce programme récupère et traite les informations provenant des différents capteurs (le code pour récupérer les valeurs venant du capteur MISIR et du capteur OGM sont identiques).

```
while (cpt < 30)
{
    //Récupération valeur capteur MISIR
    valeurMisir = analogRead(A0);
    resultatMisir = resultatMisir + valeurMisir;
    Serial.println(resultatMisir);
///////////

    //Récupération valeur capteur VZ
    int j = 0;
    Wire.beginTransmission(0x70);
    Wire.write(9);
    Wire.endTransmission();
    Wire.requestFrom(0x70, 6);
    while (Wire.available())
    {
        int i = Wire.read();
        tabValeur[j] = i;
        j++;
    }
    valeurVZ = (tabValeur[0] - 13) * (1600 / 229) + 400;
    resultatVZ = resultatVZ + valeurVZ;
    Serial.println(resultatVZ);
/////////
}
```

Les informations sont récupérées toutes les secondes durant 30 secondes et la moyenne de ces 30 valeurs est calculée.

```
VZ = resultatVZ / 30;
Serial.println(VZ);

misir = resultatMisir / 30;
Serial.println(misir);
```

Ces informations sont envoyées à Thingspeak

```
//Envoie des données au serveur toute les 30 secondes
if (client.connect(server, 80)) { // "184.106.153.149" ou api.thingspeak.com
    String postStr = apiKey;
    postStr += "&field1=";
    postStr += String(VZ);
    postStr += "&field2=";
    postStr += String(misir);
    postStr += "\r\n\r\n";

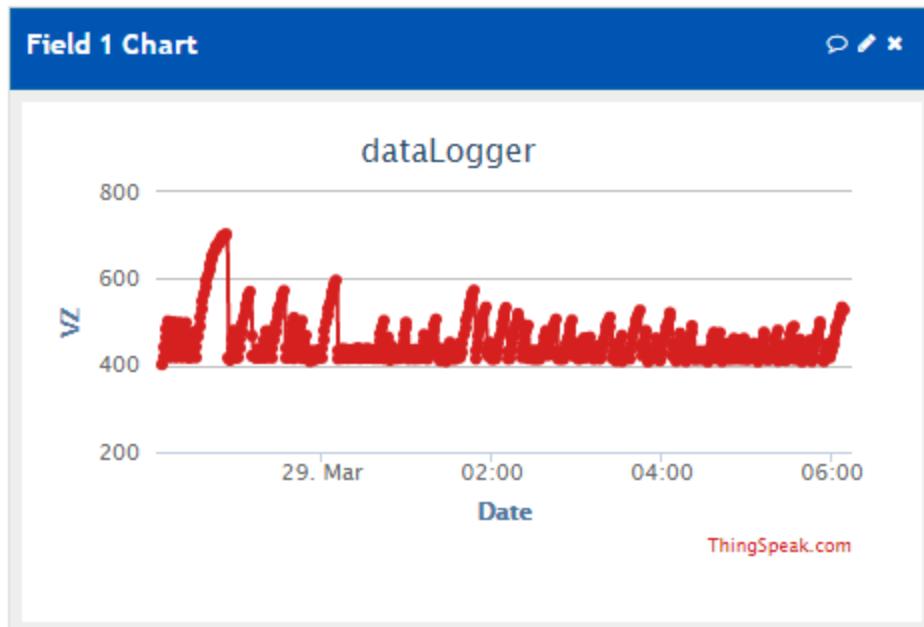
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
    Serial.println("OK");
}
client.stop();

Serial.println("Waiting...");
//Thingspeak à besoin de minimum 15s de delais entre chaque updates
delay(20000);
///////////////////////////////
```

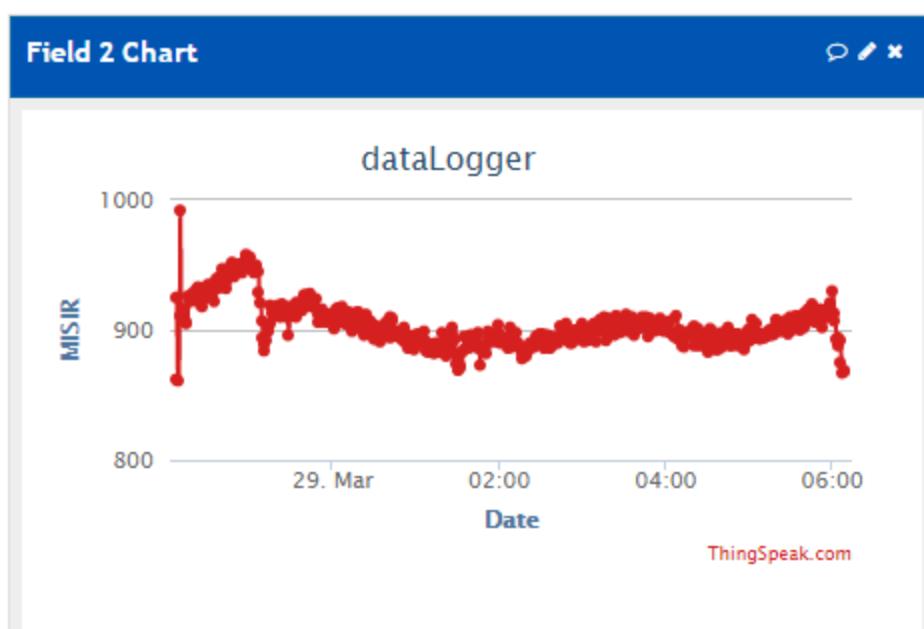
Relevé des mesures de chaque capteur

Les mesures ont été réalisées durant toute une nuit dans une chambre occupée par 1 personne.

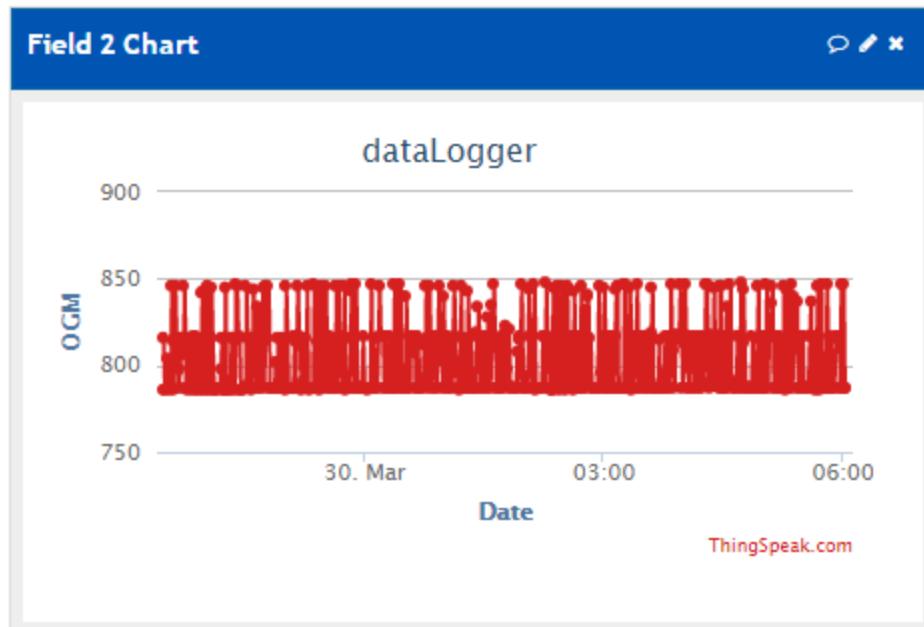
Capteur MISC VZ-89:



Capteur MISIR 5000:



Capteur OGM 200:



En comparant les différents graphes, on constate que le capteur MISIR 5000 est le plus précis.

Le capteur MISIR 5000 donne les valeurs les plus réalistes

Conclusion

Après avoir réussi à faire fonctionner et à tester chacun des éléments de notre projet nous pouvons conclure dans un premier temps que chaque capteur est fonctionnel, certains plus ou moins précis que d'autres du même domaine.

Les capteurs qui nous ont donné le plus de difficultés sont les capteurs de qualité d'air, de gaz CO2.

Lors de ce projet nous avons néanmoins rencontré des difficultés :

- L'outil GitHub est un outil pratique pour coder mais possède des inconvénients tels que l'organisation du projet car il est difficile de supprimer ou mettre des fichiers de façon simple.
- Les différents capteurs de CO2 qui parfois étaient assez compliqué à faire fonctionner, les résultats affichés étaient au départ peu fiable.
- Des difficultés de se connecter à internet chez soi avec les ESP qui nous ont été prêté en cours.

Dans l'ensemble le projet s'est déroulé sans trop de difficulté et nous a laissé travailler en autonomie tout en acquérant de nouvelles connaissances.