



# Capteurs - Gaz

[Mise à jour le 26/6/2024]

## 1. Capteurs de CO

### 1.1 Généralités



Le **monoxyde de carbone (CO)** est un gaz très dangereux qui est inodore, incolore et sans goût. Les symptômes les plus courants de l'intoxication au CO sont les maux de tête, les nausées, les vomissements, les vertiges, la fatigue et une sensation de faiblesse. Les signes neurologiques comprennent la confusion, la désorientation, les troubles visuels, la syncope et les convulsions.

Le monoxyde de carbone est produit par l'oxydation partielle de composés contenant du carbone; il se forme lors du fonctionnement d'un poêle ou d'un moteur à combustion interne dans un espace clos. [Wikipédia](#)

### Ordres de grandeur

	ppm
Atmosphère naturelle	0,1
Maison	0,5 à 5
Près d'une gazinière	5 à 15
Feu de bois domestique	5000

### 1.2 MQ7 - CO



### 1.2.1 Présentation

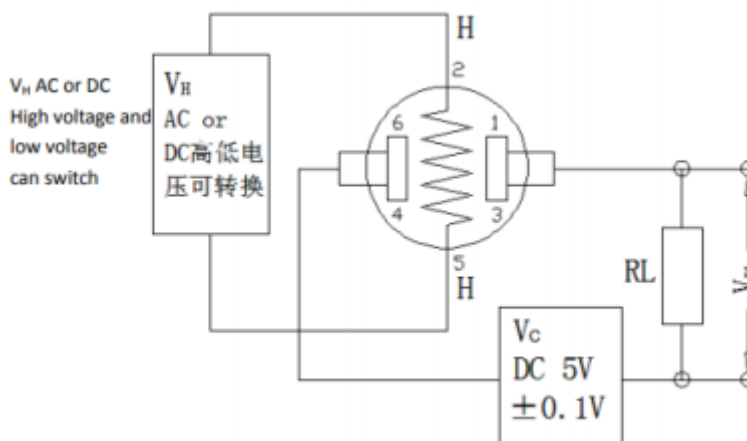
- **Source** : [wiki](#)

Module basé sur le capteur MQ7 permettant de détecter la présence de monoxyde de carbone CO de 20 à 2000 ppm. Haute sensibilité et temps de réponse rapide. La sensibilité est réglable par potentiomètre.

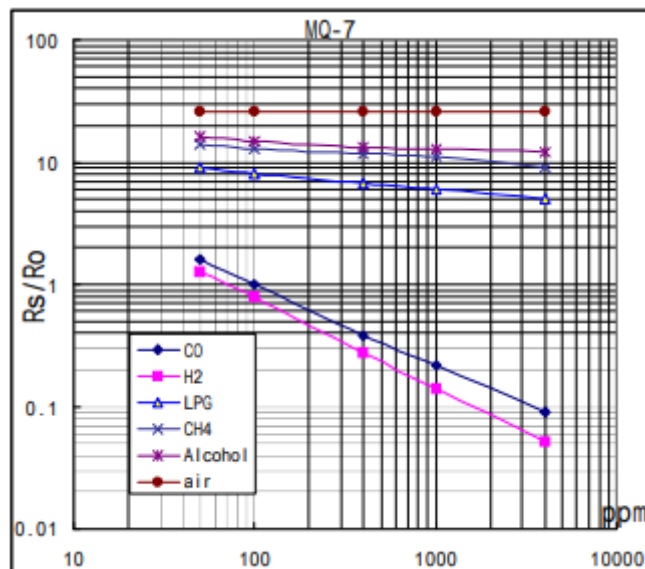
- **Distributeur** : [Gotronic](#)
- **Caractéristiques**
  - Alimentation: 5 Vcc
  - Sortie analogique
  - Temps de réponse rapide et haute sensibilité
  - Longue durée de vie et bonne stabilité
  - Dimensions: 37 x 27 x 14 mm




- **Documentation**
  - PDF à télécharger [ici](#)
  - [Schéma du module](#)



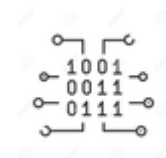
- **Modèle**



- **Aide** pour la **simulation de la chaîne de mesure**

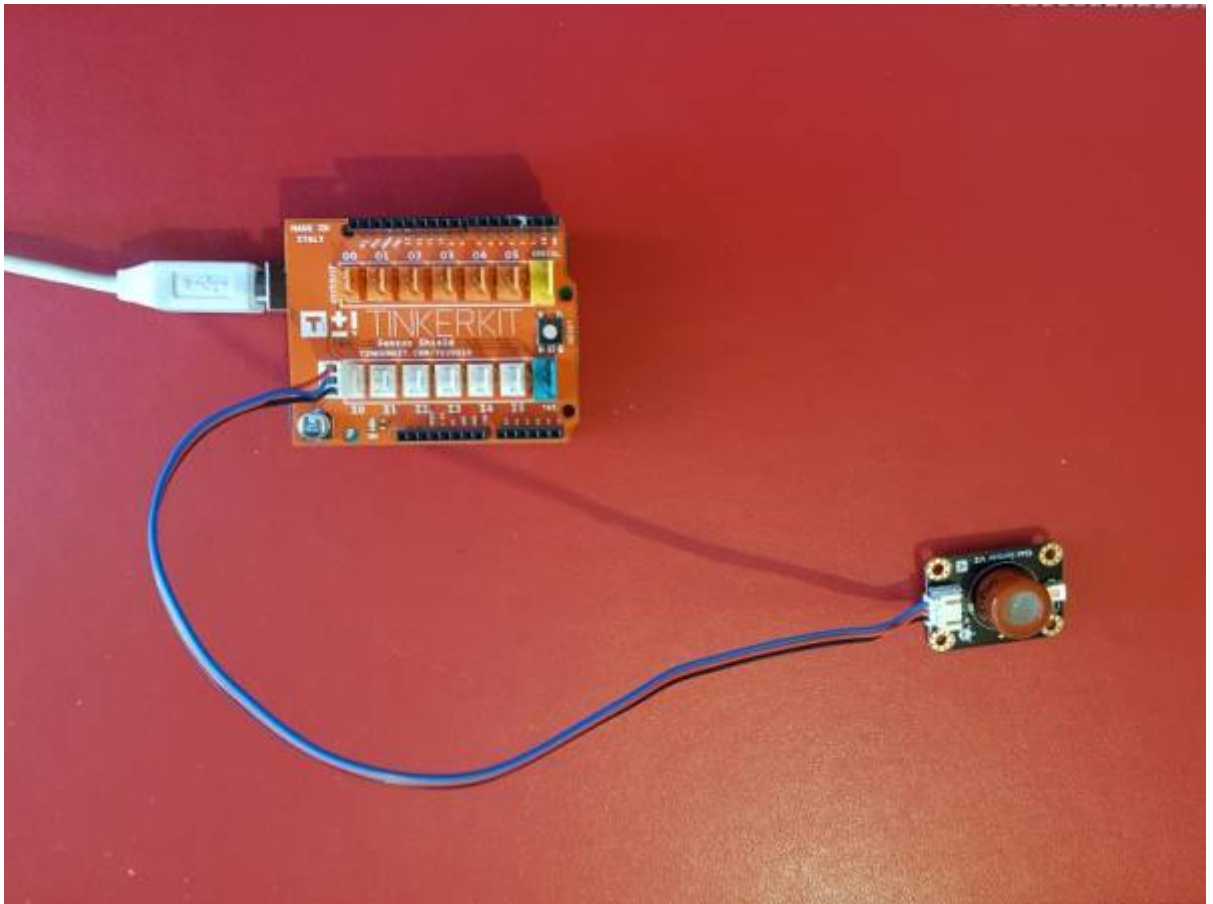
- Les équations de la chaîne de mesure sont téléchargeables [ici](#) 
- Le modèle à simuler est téléchargeable [ici](#)

- L'**Algorithme** du traitement à réaliser est téléchargeable [ici](#)



### 1.2.2 Exemples de code

- [Arduino UNO](#)
- [ESP \(Arduino\)](#)
- [Rpi Pico \(µPython\)](#)
- **Programmation** d'une carte Arduino Uno R3
  - Connexion à un shield [Tinkerkit v2](#).



- **Exemple** pour tester le capteur

\*.cpp

```
// Arduino Sample Code
// Non conforme : A modifier
void setup()
{
  Serial.begin(9600); // Set serial baud rate to 9600 bps
}
void loop()
{
  int val;
  val=analogRead(0); // Read Gas value from analog 0
  Serial.println(val,DEC);// Print the value to serial port
  delay(100);
}
```

A venir  
A venir

## 2. Capteurs de CO2

- **Ressources**

- [Why Monitor CO2 Levels in Classrooms ?](#)

### 2.1 Généralités

Le **dioxyde de carbone**, aussi appelé gaz carbonique ou anhydride carbonique, est un composé inorganique dont la formule chimique est CO<sub>2</sub>, la molécule ayant une structure linéaire de la forme O=C=O. Il se présente, sous les conditions normales de température et de pression, comme un gaz incolore, inodore, à la saveur piquante. Le CO<sub>2</sub> est utilisé par l'anabolisme des végétaux pour produire de la biomasse à travers la photosynthèse, processus qui consiste à réduire le dioxyde de carbone par l'eau, grâce à l'énergie lumineuse reçue du soleil et captée par la chlorophylle, en libérant de l'oxygène pour produire des oses, et en premier lieu du glucose par le cycle de Calvin.



À partir d'une certaine concentration dans l'air, ce gaz s'avère dangereux, voire mortel. La valeur limite d'exposition est de **3 % sur une durée de 15 minutes**. Cette valeur ne doit jamais être dépassée. Au-delà, les effets sur la santé sont d'autant plus graves que la teneur en CO<sub>2</sub> augmente.

Près de 87% des émissions de dioxyde de carbone attribuables à l'homme proviennent de la combustion de combustibles fossiles, tels que le charbon, le gaz naturel et le pétrole. [Wikipédia](#)

Les niveaux de CO<sub>2</sub> dans l'air et les problèmes de santé potentiels sont :

- **380 - 480 ppm** : taux normal de l'atmosphère
- **600 - 800 ppm** : taux correct en lieux fermés
- **1000 - 1100 ppm** : taux tolérable en lieux fermés
- **1100 - 1400 ppm** : il faut aérer pour assurer une bonne qualité de l'air, si possible.
- **2 000 à 5 000 ppm** : niveau associé aux maux de tête, à la somnolence et à l'air stagnant, vicié et étouffant. Une mauvaise concentration, une perte d'attention, une accélération du rythme cardiaque et de légères nausées peuvent également être présentes.
- **5000 ppm** : cela indique des conditions atmosphériques inhabituelles où des niveaux élevés d'autres gaz peuvent également être présents. Une toxicité ou une privation d'oxygène pourrait se produire. Il s'agit de la limite d'exposition admissible pour les expositions quotidiennes sur le lieu de travail.
- **40 000 ppm** : ce niveau est immédiatement nocif par manque d'oxygène.

### 2.2 MG811



### 2.2.1 Présentation

- **Source** : [wiki DFROBOT](#)

Module basé sur le capteur de gaz MG-811 permettant de détecter la présence de CO2. Un booster 6V met le capteur à température permettant une mesure précise. Haute sensibilité et temps de réponse rapide. Le module possède une sortie analogique et une sortie digitale ON/OFF (seuil réglable par potentiomètre).

- **Distributeur** : [Gotronic](#)

- **Caractéristiques**

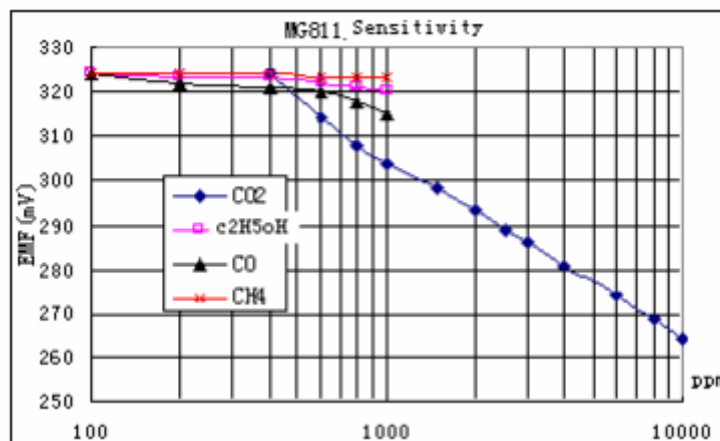
- Alimentation: 5 Vcc
- Sorties: une analogique et une digitale
- Dimensions: 42 x 32 x 30 mm



- **Documentation**

- PDF à télécharger [ici](#)
- [schéma du module](#)

- **Modèle**





- **Aide pour la *simulation de la chaîne de mesure***

- Les équations de la chaîne de mesure sont téléchargeables [ici](#)
- Le modèle à simuler est téléchargeable [ici](#)



- L'**Algorithme** du traitement à réaliser est téléchargeable [ici](#)



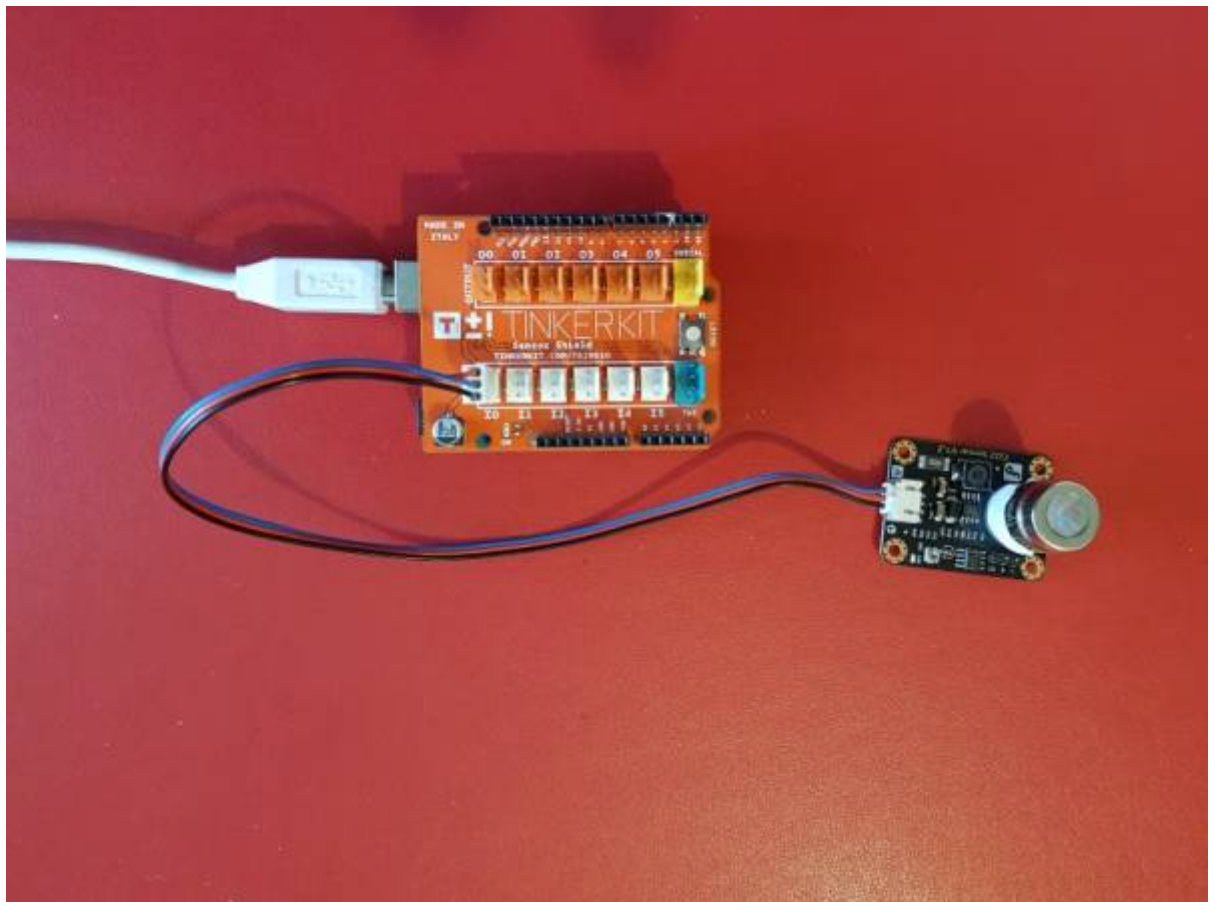
## 2.2.2 Exemples de code

- [Arduino UNO](#)
- [ESP \(Arduino\)](#)
- [Rpi Pico \(μPython\)](#)

- **Ressource** : [wiki](#) DFROBOT

- **Programmation d'une carte Arduino Uno R3**

- **Connexion** à un shield [Tinkerkit v2](#).





- **Exemple** pour tester le capteur

\*.cpp

```
/******Demo for MG-811 Gas Sensor Module
V1.1*****
Author: Tiequan Shao: tiequan.shao@sandboxelectronics.com
       Peng Wei:      peng.wei@sandboxelectronics.com

Licence: Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA
3.0)

Note: This piece of source code is supposed to be used as a
demostration ONLY. More
       sophisticated calibration is required for industrial field
application.

                                                                 Sandbox Electronics
2012-05-31
*****
*****/

/******Hardware Related
Macros*****/
#define          MG_PIN                (A0)      //define which
analog input channel you are going to use
#define          BOOL_PIN              (2)
#define          DC_GAIN               (8.5)    //define the DC
gain of amplifier

/******Software Related
Macros*****/
#define          READ_SAMPLE_INTERVAL  (50)    //define how many
samples you are going to take in normal operation
#define          READ_SAMPLE_TIMES     (5)     //define the time
interval(in milisecond) between each samples in
                                                    //normal operation

/******Application Related
Macros*****/
//These two values differ from sensor to sensor. user should derermine
this value.
#define          ZERO_POINT_VOLTAGE    (0.220) //define the
output of the sensor in volts when the concentration of CO2 is 400PPM
#define          REACTION_VOLTGAE     (0.030) //define the
voltage drop of the sensor when move the sensor from air into 1000ppm
CO2
```



```

/*****Globals*****/
float      C02Curve[3] =
{2.602,ZERO_POINT_VOLTAGE, (REACTION_VOLTAGE/(2.602-3))};
//two points are
taken from the curve.
//with these two
points, a line is formed which is
//"approximately
equivalent" to the original curve.
//data format:{ x,
y, slope}; point1: (lg400, 0.324), point2: (lg4000, 0.280)
//slope = (
reaction voltage ) / (log400 -log1000)

void setup()
{
  Serial.begin(9600); //UART setup,
  baudrate = 9600bps
  pinMode(BOOL_PIN, INPUT); //set pin to input
  digitalWrite(BOOL_PIN, HIGH); //turn on pullup
  resistors

  Serial.print("MG-811 Demonstration\n");
}

void loop()
{
  int percentage;
  float volts;

  volts = MGRead(MG_PIN);
  Serial.print( "SEN0159:" );
  Serial.print(volts);
  Serial.print( "V          " );

  percentage = MGGetPercentage(volts,C02Curve);
  Serial.print("C02:");
  if (percentage == -1) {
    Serial.print( "<400" );
  } else {
    Serial.print(percentage);
  }

  Serial.print( "ppm" );
  Serial.print("\n");

  if (digitalRead(BOOL_PIN) ){
    Serial.print( "====BOOL is HIGH====" );
  } else {

```

```
Serial.print( "=====BOOL is LOW=====" );
}

Serial.print("\n");

delay(500);
}

/***** MGRead *****/
*****
Input:  mg_pin - analog channel
Output: output of SEN-000007
Remarks: This function reads the output of SEN-000007
*****
*****/
float MGRead(int mg_pin)
{
    int i;
    float v=0;

    for (i=0;i<READ_SAMPLE_TIMES;i++) {
        v += analogRead(mg_pin);
        delay(READ_SAMPLE_INTERVAL);
    }
    v = (v/READ_SAMPLE_TIMES) *5/1024 ;
    return v;
}

/***** MQGetPercentage *****/
*****
Input:  volts - SEN-000007 output measured in volts
        pcurve - pointer to the curve of the target gas
Output: ppm of the target gas
Remarks: By using the slope and a point of the line. The x(logarithmic
value of ppm)
        of the line could be derived if y(MG-811 output) is provided.
As it is a
        logarithmic coordinate, power of 10 is used to convert the
result to non-logarithmic
        value.
*****
*****/
int MQGetPercentage(float volts, float *pcurve)
{
    if ((volts/DC_GAIN )>=ZERO_POINT_VOLTAGE) {
        return -1;
    } else {
        return pow(10, ((volts/DC_GAIN)-pcurve[1])/pcurve[2]+pcurve[0]);
    }
}
```

}

- A venir
- A venir

## 2.3 SCD4X - CO2, T, (P), RH



### 2.3.1 Présentation

- **Sources** : [wiki](#) SCD40 M5STACK | [Mouser](#) SCD41

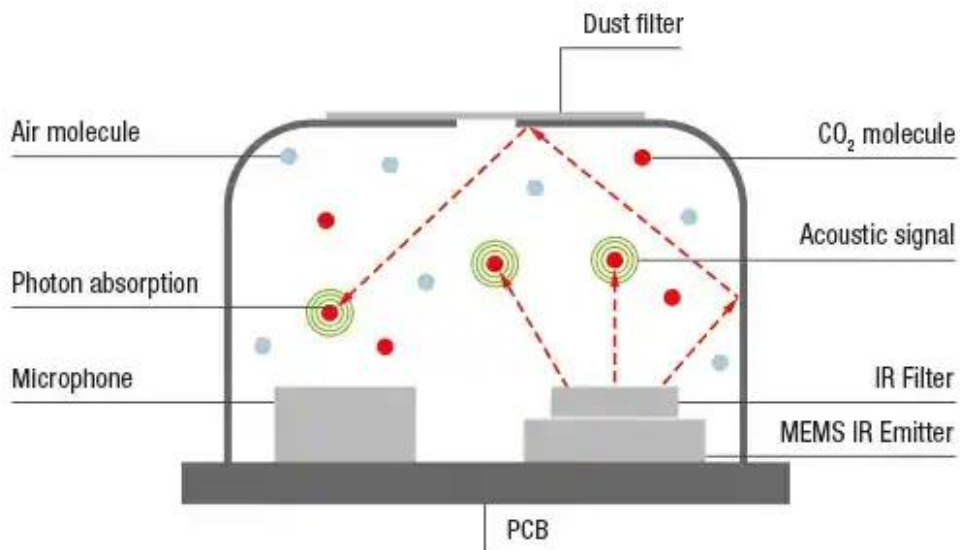
Module [M5STACK](#) combinant des capteurs de CO2 (photoacoustique), de température et d'humidité à connecter à une carte à microcontrôleur via un bus I2C.

- **Distributeur** : [Gotronic](#)
- **Applications**
  - Détection environnementale
  - Qualité de l'air
  - Etudes de ventilation
- **Les capteurs Sensirion**
  - **SCD40** : CO2, température et humidité relative
  - **SCD41** : CO2, température, pression et humidité relative
- **Caractéristiques communes au SCD40 et SCD41**
  - Alimentation : 2,4 à 5,5V
  - Plages de mesure :
    1. CO2
      1. SCD40 : 400 à 2000 ppm ( $\pm 50$  ppm + 5 % de lecture)
      2. SCD41 : 400 à 5000 ppm ( $\pm 40$  ppm + 5 % de lecture)
    2. température : -10 à 60 °C (voir datasheet)
    3. humidité : 0 à 95 %RH (voir datasheet)
  - Interface série I2C : **SLA = 0x62**

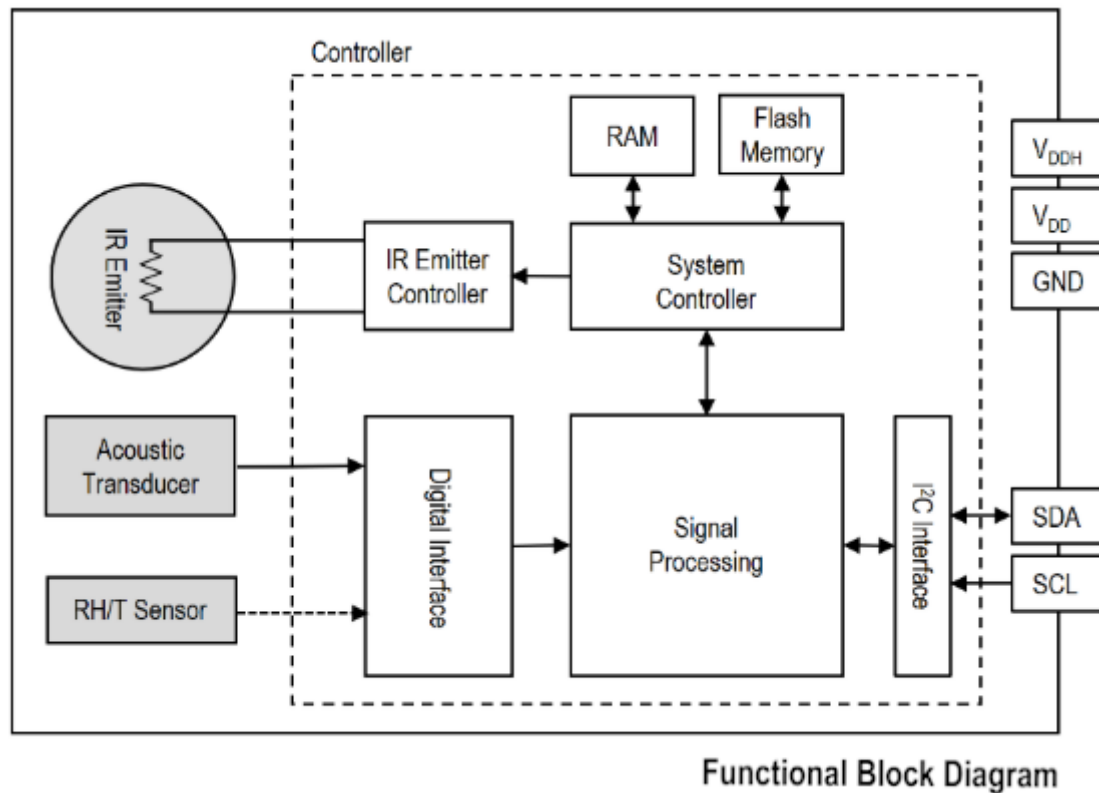
- Connecteur
  - M5STACK SCD40 : HY2.0-4P
- Dimensions
  - M5STACK SCD40 : 48 x 24 x 16 mm
- Poids
  - M5STACK SCD40 : 8 g

• **Principe** 

- What types of NDIR sensors exist and how do they work ? [[Télécharger](#)]



• **Schéma-blocs**



- **Documentation**

- PDF du SCD40 à télécharger [ici](#)
- Schéma à télécharger [ici](#)

### 2.3.2 Bibliothèques

- [Arduino UNO](#)
- [ESP \(Arduino\)](#)
- [Rpi Pico \(µPython\)](#)

- *A partir du gestionnaire de bibliothèques de l'IDE Arduino, installer :*

**Sensirion I2C SCD4x** par Sensirion

0.4.0 installed

Library for the SCD4X sensor family by Sensirion Enables you to use the SCD40 and SCD41 sensors via I2C.

- A venir
- A venir



### 2.3.3 Exemples de code

- [Arduino UNO](#)
- [ESP \(Arduino\)](#)
- [Rpi Pico \(µPython\)](#)
  
- **Ressources**
  - [Wiki M5STACK](#)



- **Exemple de l'IDE Arduino pour tester le capteur**  
Dans l'IDE Arduino, sélectionner : Fichier → Exemples → Sensirion I2C SCD4x → **exempleUsage**
  
- A venir
  
- A venir

## 2.5 SGP30 - Qualité de l'air intérieur



### 2.5.1 Présentation

- **Source** : [wiki](#) Seeed Studio

Le SG30 est capable de détecter une large gamme de composés organiques volatils (COV) et de H2. Il est destiné à la **surveillance de la qualité de l'air intérieur**.

- **Distributeur** : [Mouser](#)

- **Caractéristiques**
  - Alimentation : 3,3V , 5V
  - Consommation: < 1,5 mA
  - Interface : I2C
  - Connectique : Qwiic
  - Capteur de Gaz MOX
  - Plages de mesure
    - 400 à 60000ppm d'**eCO2** (dioxyde de carbone calculé équivalent, résolution: 11 ppm)
    - 0 à 60 000ppb de COVT (composé organique volatile total, résolution: 13 ppb)
    - éthanol: 0 à 1000 ppm (précision: 15%, résolution: 0,2 %)
    - H2: 0 à 1000 ppm (précision: 10%, résolution: résolution: 0,2 %)
  - Temps de chauffe: < 15 sec
  - Interface série I2C : **SLA = 0x58**



- **Documentation**
  - PDF à télécharger [ici](#)

## 2.5.2 Bibliothèques

- [Arduino UNO](#)
- [ESP \(Arduino\)](#)
- [Rpi Pico \(µPython\)](#)
- *A partir du gestionnaire de bibliothèques de l'**IDE Arduino**, installer :*

**Adafruit SGP30 Sensor** par Adafruit

This is an Arduino library for the Adafruit SGP30 Gas / Air Quality Sensor This is an Arduino library for the Adafruit SGP30 Gas / Air Quality Sensor

- Avenir
- Avenir



## 2.5.3 Exemples de code

- [Arduino UNO](#)
- [ESP \(Arduino\)](#)

- [Rpi Pico \(µPython\)](#)

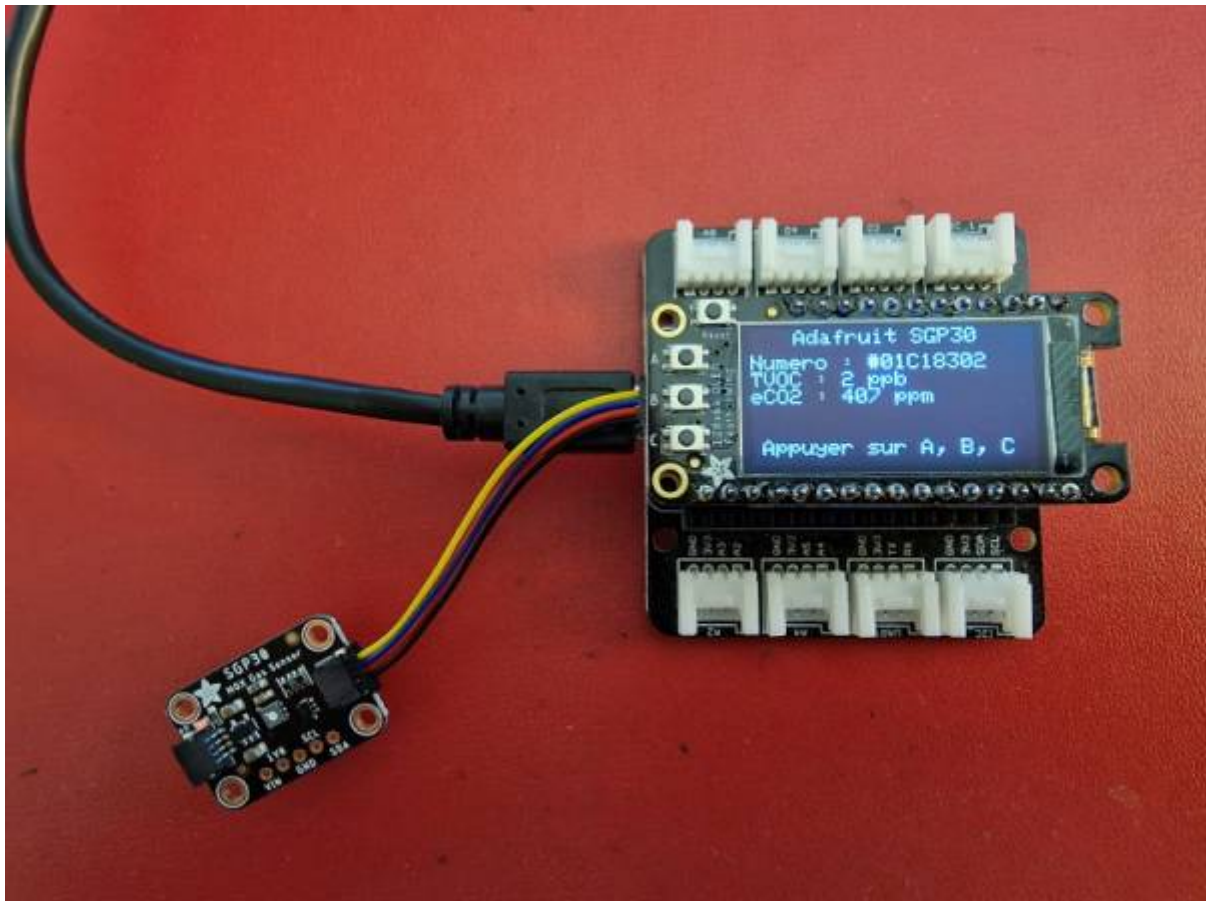
- **Exemple de l'IDE Arduino pour tester le capteur**

Fichier → Exemples → Adafruit SGP30 Sensor → **sgp30test.ino**



- **Mise en oeuvre** du capteur avec un afficheur OLED

- *Description* : mesure de CO<sup>2</sup> et COVT à l'aide d'un capteur adafruit SGP30, test des boutons-poussoirs et affichage sur un écran Oled Adafruit SH1107. L'écran et le capteur sont reliés via le système Qwiic de Sparkfun.



- **Matériels**

- Carte à microcontrôle : [Adafruit Feather Huzzah ESP8266](#)
- Afficheur : [Adafruit OLED SH1107](#)

- **Code**



\*.cpp

```
// Matériels : Adafruit Feather Huzzah ESP8266 + Support Particle,  
Adafruit OLED SH1107, Adafruit SGP30, câble Qwiic.  
// Logiciel : Arduino  
  
// Bibliothèques  
#include <SPI.h>
```



```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>
#include "Adafruit_SGP30.h"

#define BUTTON_A 0
#define BUTTON_B 16
#define BUTTON_C 2

// Constructeurs
Adafruit_SH1107 display = Adafruit_SH1107(64, 128, &Wire);
Adafruit_SGP30 sgp;

void setup()
{
  // Bus I2C
  Wire.begin(); // Initialisation
  Wire.setClock(400000);
  display.begin(0x3C, true); // L'adresse de l'afficheur est 0x3C par
  défaut

  // Configuration de l'affichage
  display.setRotation(1); // Affichage horizontal
  display.setTextSize(1);
  display.setTextColor(SH110X_WHITE);
  display.clearDisplay(); // Pour ne pas afficher le logo Adafruit
  chargé
  // automatiquement à la mise sous tension

  // Test de la communication avec le capteur
  if (!sgp.begin())
  {
    display.println("Le capteur de repond pas");
    while (1)
      ;
  }
  display.print("Numero de serie #");
  display.print(sgp.serialnumber[0], HEX);
  display.print(sgp.serialnumber[1], HEX);
  display.println(sgp.serialnumber[2], HEX);
  display.display(); // Transfert du buffer sur l'écran
  delay(2000);

  // Connexion des boutons-poussoirs
  pinMode(BUTTON_A, INPUT_PULLUP);
  pinMode(BUTTON_B, INPUT_PULLUP);
  pinMode(BUTTON_C, INPUT_PULLUP);
}

void loop()
{
```

```
// Efface le buffer
display.clearDisplay();

// Test des boutons
display.setCursor(0, 0);

if (!digitalRead(BUTTON_A))
  display.print("[A]");
if (!digitalRead(BUTTON_B))
  display.print("[B]");
if (!digitalRead(BUTTON_C))
  display.print("[C]");

// Titre
display.setCursor(40, 0);
display.println("SGP30");

// Numéro de série
display.setCursor(0, 12);
display.print("Numero : #");
display.print(sgp.serialnumber[0], HEX);
display.print(sgp.serialnumber[1], HEX);
display.println(sgp.serialnumber[2], HEX);

// Mesures
if (!sgp.IAQmeasure())
{
  Serial.println("Measurement failed");
  return;
}

// Mesures et affichage
display.print("TVOC : ");
display.print(sgp.TVOC);
display.println(" ppb");
display.print("eCO2 : ");
display.print(sgp.eCO2);
display.println(" ppm");

// Infos
display.setCursor(5, 52);
display.print("Appuyer sur A, B, C");
display.display(); // Transfert du buffer sur l'écran

delay(500);
}
```



[Télécharger](#) le projet PlatformIO pour VSCode.

- A venir
- A venir

## Pour aller plus loin ...

- **Exemple de projet**  
[Surveillance des niveaux de CO2 et contrôle du débit d'air à l'aide du terminal Wio](#)

From:

<https://webge.fr/dokuwiki/> - **WEBGE Wikis**

Permanent link:

<https://webge.fr/dokuwiki/doku.php?id=materiels:capteurs:gaz:gaz&rev=1719414034>

Last update: **2024/06/26 17:00**

