Capteurs - Environnement

[Mise à jour le 24/3/2024]

1. Généralités sur les grandeurs physiques

1.1 Température

- **Ressource** : Wikipédia

1.2 Humidité

- **Ressource** : Wikipédia

1.3 Pression

- **Ressource** : Wikipédia

---

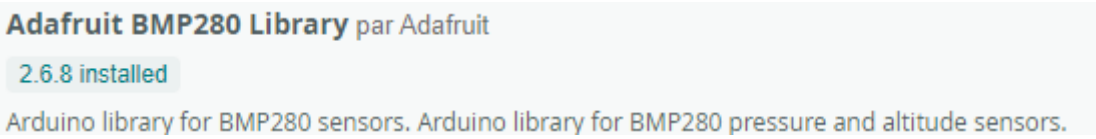
2. Capteurs de température et de pression

2.1 BMP280

> Ce capteur est basé sur le circuit BMP280 et mesure la pression atmosphérique, la température et l'altitude. Il communique avec un microcontrôleur via le bus I2C ou SPI.

- *Distributeur* : Gotronic
- *Caractéristiques*
  - Alimentation: 3,3 à 5 Vcc
  - Interface I2C:
    - sur connecteur Qwiic ou Stemma QT
    - sur pastilles femelles au pas de 2,54 mm
  - Interface SPI:
    - sur pastilles femelles au pas de 2,54 mm
  - Plages de mesure:
    - température: -40°C à 85°C
    - pression: 30 à 110 kPa
    - altitude: en fonction de la pression
  - Précision:
    - température: ±1°C
    - pression: ±1 hPa
    - altitude: ±1 m
  - Sortie 3,3 Vcc/100 mA maxi
  - Dimensions: 19,2 x 17,9 x 2,9 mm

- *Documentation*
  - PDF à télécharger **ici**
- ***Programmation d'une carte Arduino***
  - *Bibliothèques à installer dans l'IDE (I2C)*



3. Capteurs de température et d'humidité

3.1 HYT-221

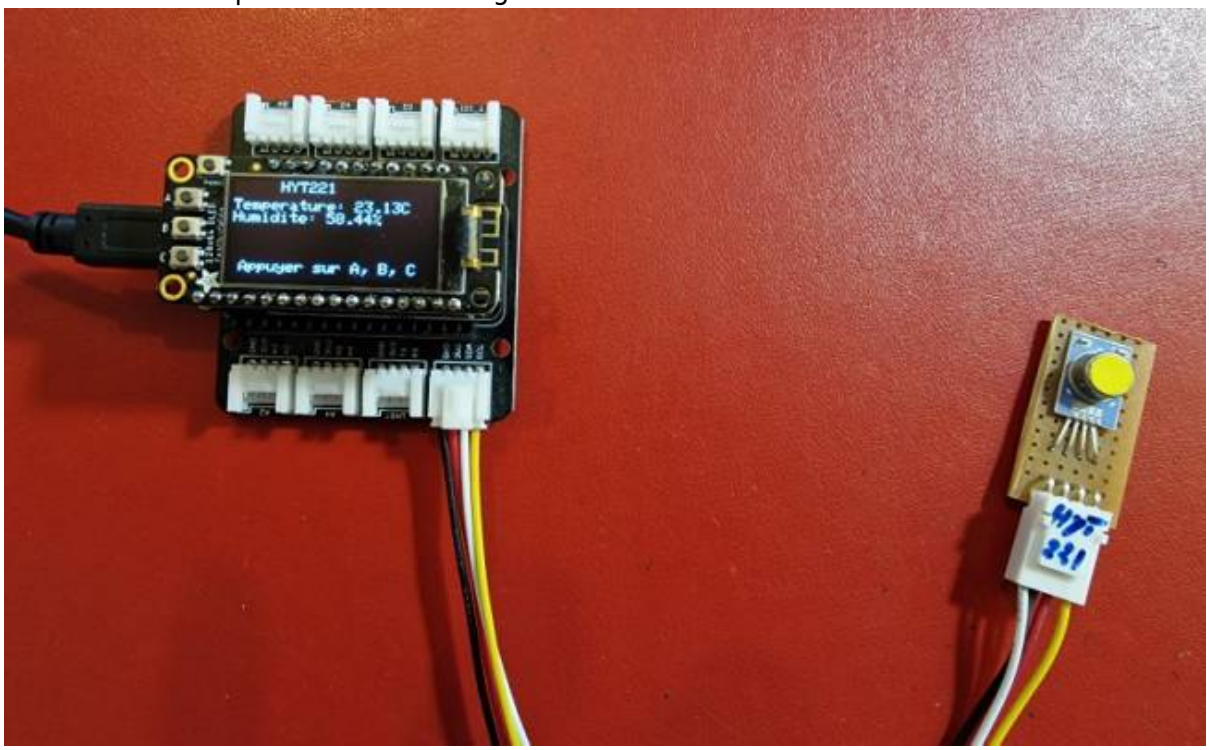
- *Source* : GitHub

> Capteur capacitif **numérique d'humidité et de température** relative présentant une précision de base de ±1,8% HR, calibré et compensé en température. Communication via le **bus I²C** (**adresse 0x28** par défaut).

- *Distributeur* : Gotronic
- *Caractéristiques*
  - Alimentation: 2,7 à 5,5 Vcc
  - Consommation: <22 µA à 1 Hz (850 µA maxi)
  - Consommation en veille: <1 µA
  - Plage de mesure:
    1. 0 à 100% HR
    2. -40°C à 125°C
  - Précision:
    1. ±1,8% HR
    2. ±0,2°C
  - Hystérésis: < ±1% HR
  - Interface: I²C
  - Dimensions: 16 x 10 x 6 mm

- *Documentation*
  - PDF à télécharger **ici**

- **Télécharger** un exemple pour tester le capteur.
- **Mise en oeuvre du capteur avec un afficheur OLED**
  - *Description* : mesure de la température et de l'humidité à l'aide d'un capteur **HYT221**, test des boutons-poussoirs et affichage sur un écran Oled Adafruit SH1107.



  - *Matériels*

- Carte à microcontrôleur : Adafruit Feather Huzzah ESP8266 + Support Particle
- Afficheur : Adafruit OLED SH1107

◦ *Code Arduino*

*.cpp

```
// Matériels : Adafruit Feather Huzzah ESP8266  + Support Particle,
Adafruit OLED SH1107, HYT221, câble Qwiic
// Logiciel : Arduino

// A ajouter
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>

// Adresse I2C par défaut de HYT 221, 271, 371
#define HYT_ADDR 0x28

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

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

void setup()
{
  // Bus I2C
  Wire.begin();
  Wire.setClock(400000);
  display.begin(0x3C, true); // L'addresse 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
  // Connexion des boutons-poussoirs
  pinMode(BUTTON_A, INPUT_PULLUP);
  pinMode(BUTTON_B, INPUT_PULLUP);
  pinMode(BUTTON_C, INPUT_PULLUP);
}
```

```
void loop()
{
  double humidity;
  double temperature;

  // 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(30, 0);
  display.println("HYT221");

  Wire.beginTransmission(HYT_ADDR); // Début de la transmission avec le
capteur HYT221
  Wire.requestFrom(HYT_ADDR, 4);    // Nécessite 4 octets

  // Read the bytes if they are available
  // Les deux premiers octets sont l'humidité, les deux suivants la
température
  if (Wire.available() == 4)
  {
    int b1 = Wire.read();
    int b2 = Wire.read();
    int b3 = Wire.read();
    int b4 = Wire.read();

    Wire.endTransmission(); // Fin de la transmission avec le capteur
HYT221

    // Calcul de l'humidité
    int rawHumidity = b1 << 8 | b2;
    rawHumidity = (rawHumidity &= 0x3FFF);
    humidity = 100.0 / pow(2, 14) * rawHumidity;

    // Calcul de la température
    b4 = (b4 >> 2);
    int rawTemperature = b3 << 6 | b4;
    temperature = 165.0 / pow(2, 14) * rawTemperature - 40;

    // Affichage
```

```
    display.setCursor(0, 12);
    display.print("Temperature: ");
    display.print(temperature);
    display.println("C ");
    display.print("Humidite: ");
    display.print(humidity);
    display.println("% ");

    // Infos
    display.setCursor(5, 52);
    display.print("Appuyer sur A, B, C");
    display.display();
  }
  else
  {
    display.println("Pas de mesure");
  }
}
```

Télécharger le projet PlatformIO pour VSCode.

3.2 DHT22

Ce capteur de température et d'humidité (version pro DHT22) compatible Grove utilise une thermistance CTN et un capteur capacitif et délivre une sortie digitale.

- *Distributeur* : Gotronic
- *Caractéristiques*
  - Interface: compatible Grove
  - Alimentation: 3,3 à 6 Vcc

    - Consommation: 1,5 mA
    - Plage de mesure:
        - température: -40°C à 80°C (±0,5°C)
        - humidité: 5 à 99% HR (±2%)
    - Temps de réponse: 6 à 20 secondes
    - Interface : signal TOR (protocol spécifique 1 fil)
    - Dimensions: 40 x 20 x 11 mm

- *Wiki Seeed studio*
    - Site à consulter **ici**

4. Capteurs atmosphériques

4.1 BME280, BME680

4.1.1 Présentation

- ***Sources*** : site sparkfun

> Capteur environnemental mesurant la **température, la pression barométrique et l'humidité** ! Ce capteur est idéal pour toutes sortes de capteurs météorologiques / environnementaux et peut être utilisé à la fois en **I²C** et en SPI.

- ***Distributeurs*** : Gotronic
- **Caractéristiques**
    - Alimentation: 3,3 à 5 Vcc
    - Plages de mesure:
        - température: -40°C à 85°C
        - humidité: 0 à 100% HR
        - pression: 300 à 1100 hPa
    - Précision:
        - température: ±1°C (±0,5°C pour le BME680)
        - humidité: ±3%
        - pression: ±1 hPa (0,12hPa pour le BME680)
    - Interfaces:
        - I2C: sur connecteur Qwiic de Sparkfun ou Stemma QT d'Adafruit. **Adresse I2C: 0x77** (0x76 via cavalier à connecter entre SDO et GND)

    - SPI: sur pastilles femelles au pas de 2,54 mm (connecteurs mâles à souder inclus)

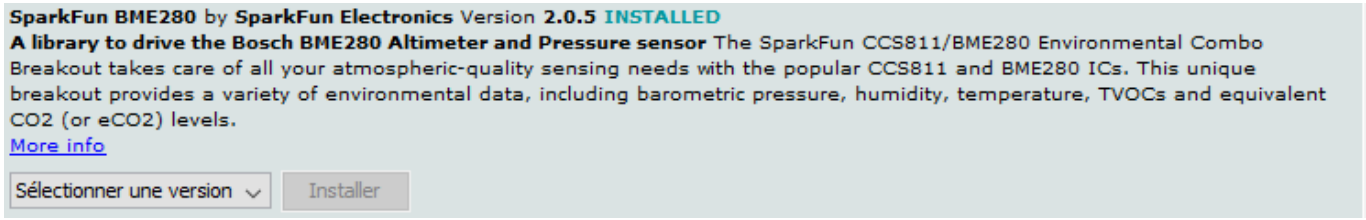
- ***Documentation***
    - PDF à télécharger **ici**

4.1.2 Bibliothèques

- RPi Pico(MicroPython)
- ESP32 (C++)

- *A installer dans le Raspberry Pi Pico*
    - Télécharger le code de la **bibliothèque BME280** sur Github, le copier dans un fichier *BME280.py* et l'installer dans le dossier **/lib** sur le raspberry Pi Pico. Modifier éventuellement l'adresse du composant dans le code de la bibliothèque (**0x76** par défaut), ou **0x77** (par ex: sparkfun).

- *A installer dans l'IDE*



- *Un premier exemple pour tester le capteur*
  → Fichier → Exemples → SparkFun BME280 → **Example1_BasicReadings.ino**

4.1.3 Exemples de code

- RPi Pico(MicroPython)
- ESP32 (C++)

- **Ressource**
    - MicroPython: BME280 with ESP32 and ESP8266 (Pressure, Temperature, Humidity) sur Random Nerd Tutorials

*Exemple de code pour un **Raspberry Pi Pico***

*.py

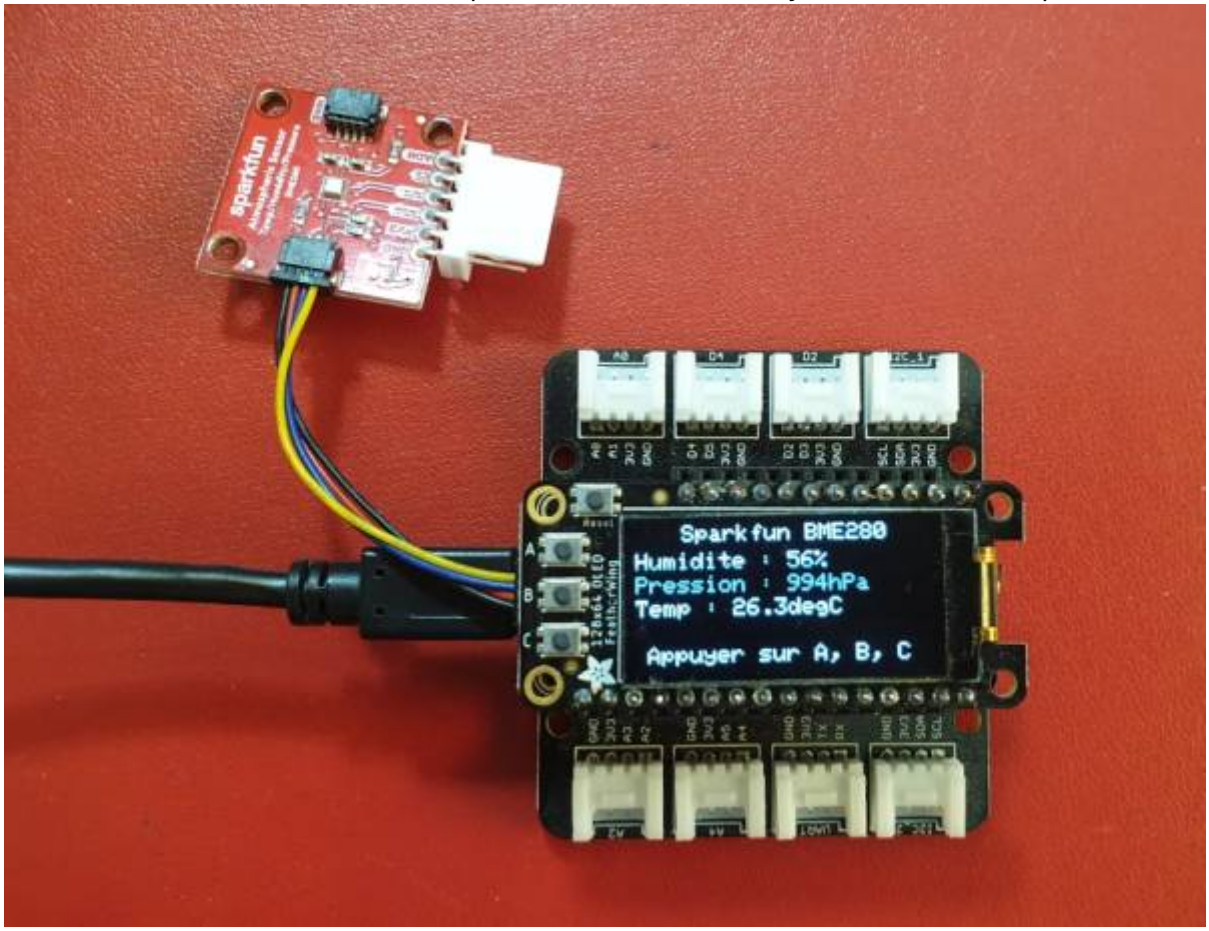
```
from machine import Pin, I2C
from time import sleep
import bme280 # bibliothèque du capteur (installée dans /lib

# RP2 - Pin assignment
i2c = I2C(1,scl=Pin(7), sda=Pin(6), freq=400_000)

while True:
  bme = bme280.BME280(i2c=i2c)
  temp = bme.temperature
  hum = bme.humidity
  pres = bme.pressure
  print('Temperature: ', temp)
  print('Humidity: ', hum)
  print('Pressure: ', pres)

  sleep(5)
```

- **Mise en oeuvre du capteur avec un afficheur OLED**
  - **Description** : mesure de de la température, de l'humidité et de la pression à l'aide d'un capteur **Sparkfun BME280**, 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 microcontrôleur : Adafruit Feather Huzzah ESP8266 + Support Particle

    - Afficheur : Adafruit OLED SH1107
  - ***Code Arduino***

*Exemple de code pour un **ESP32 Feather Huzzah***

*.cpp

```
// Matériels : Adafruit Feather Huzzah ESP8266  + Support Particle, Adafruit OLED SH1107, Sparkfun BME280, câble Qwiic
// Logiciel : Arduino

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>
#include "SparkFunBME280.h"

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

// Constructeurs
Adafruit_SH1107 display = Adafruit_SH1107(64, 128, &Wire);
BME280 bme_280; // L'adresse du circuit BME280 est 0x77 par défaut

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

  // Configuration de l'affichage
  display.setRotation(1); // Affichage horizontal
  display.setTextSize(1); // Horizontal
  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 (bme_280.beginI2C() == false)
  {
    display.println("DEFAUT(s)");
    display.println("1. Le capteur BME280 ne repond pas ! ");
    display.println();
    display.print("BLOCAGE du PROGRAMME");
    display.display(); // Transfert du buffer sur l'écran
```

```
    while (1)
      delay(10); // Blocage du programme
  }

  // Connexion des boutons-poussoir
  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(20, 0);
  display.println("Sparkfun BME280");

  // Humidité
  display.setCursor(0, 12);
  display.print("Humidite : ");
  display.print(bme_280.readFloatHumidity(), 0);
  display.println("%");

  // Pression en hPa
  display.setCursor(0, 22);
  display.print("Pression : ");
  display.print(bme_280.readFloatPressure() / 100, 0);
  display.println("hPa");

  // Température
  display.setCursor(0, 32);
  display.print("Temp : ");
  display.print(bme_280.readTempC(), 1);
  display.print("C");

  // Infos
  display.setCursor(5, 52);
  display.print("Appuyer sur A, B, C");
```

```
  // yield();
  display.display(); // Transfert du buffer sur l'écran
  delay(10);
}
```

Télécharger le projet PlatformIO pour VSCode.

4.2 SCD41

- **Capteur de CO^2, température et humidité**. Voir Capteurs - Gaz

4.3 SGP30

- **Capteur de qualité de l'air intérieur (CO^2, COV, éthanol, H2)**. Voir Capteurs - Gaz