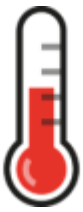
Capteurs - Température

[Mise à jour le 18/8/2023]

1. Généralités



La température est une grandeur physique mesurée à l'aide d'un thermomètre et étudiée en thermométrie. Dans la vie courante, elle est reliée aux sensations de froid et de chaud, provenant du transfert thermique entre le corps humain et son environnement.

En physique, elle se définit de plusieurs manières : comme fonction croissante du degré d'agitation thermique des particules (en théorie cinétique des gaz), par l'équilibre des transferts thermiques entre plusieurs systèmes ou à partir de l'entropie (en thermodynamique et en physique statistique).

La température est une variable importante dans d'autres disciplines : météorologie et climatologie, médecine, et chimie.

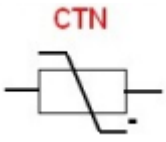
- *Ressource à consulter sur **Wikiversité*** : Capteur de température
- *Tableau comparatif des capteurs **Groove***
  - Fichier **Excel** à télécharger ici

---

2. Capteurs analogiques

2.1 Thermistance

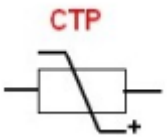
- **Généralités**
  Résistance électrique dont la valeur varie rapidement en fonction de la température.



- **CTN**
  Les CTN (Coefficient de Température Négatif, en anglais NTC, Negative Temperature Coefficient) sont des thermistances dont la résistance diminue de façon uniforme quand la température augmente. Leur modèle est donné ci-dessous.

$$R_{(T)} = R_0 . e^{B\left[\frac{1}{T}-\frac{1}{T_0}\right]}$$

R: Resistance in ambient temperature T (K)
(K: absolute temperature)
R_0: Resistance in ambient temperature T_0 (K)
B: B-Constant of Thermistor



- **CTP**
  Les CTP (Coefficient de Température Positif, en anglais PTC, Positive Temperature Coefficient) sont des thermistances dont la résistance augmente avec la température. On distingue les thermorésistances (augmentation continue et régulière de la résistance avec la température, voir ci-dessus) des CTP dont la valeur augmente fortement avec la température dans une plage de température limitée (typiquement entre 0 °C et 100 °C).

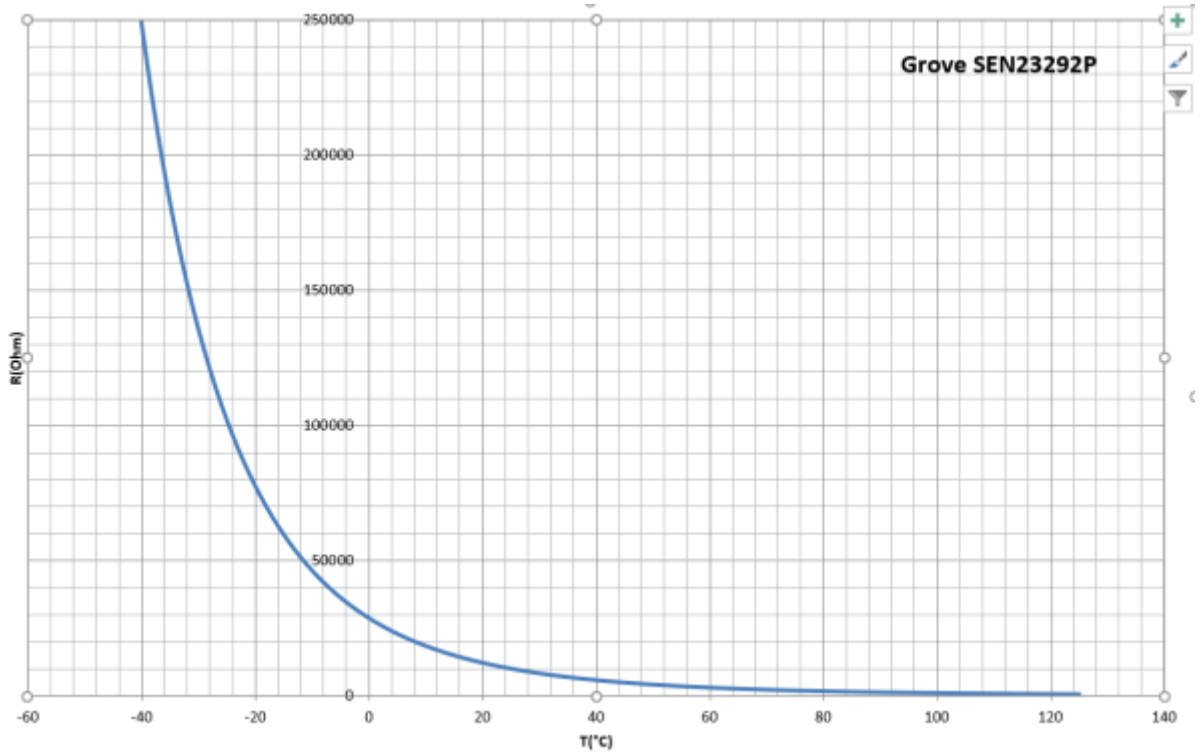
2.2 Module SEN23292P

- *Source* : wiki seeed studio

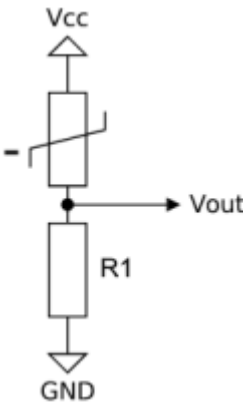
> Ce capteur de température compatible Grove à CTN **NCP18WF104F03RC** délivre un signal analogique de 0 à 5 Vcc en fonction de la température mesurée.

- *Distributeur* : Gotronic
- *Caractéristiques*
  - Alimentation: 5 Vcc
  - Plage de mesure: -40 à +125 °C
  - Précision: 1,5 °C
  - Dimensions: 20 x 20 x 13 mm

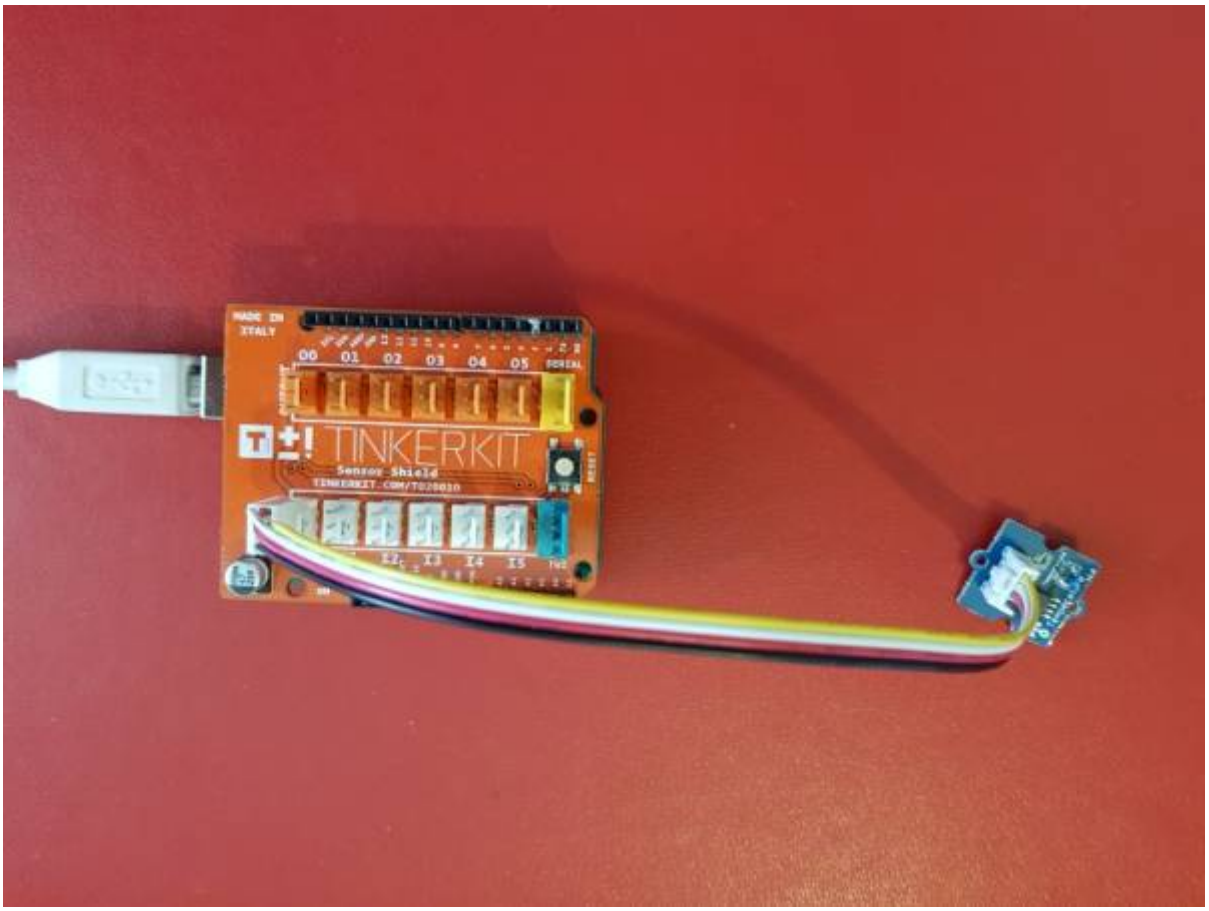
- *Modèle*
  - PDF à télécharger **ici**



- *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**



- *Connexion à un shield* Tinkerkit v2 monté sur une Arduino Uno.



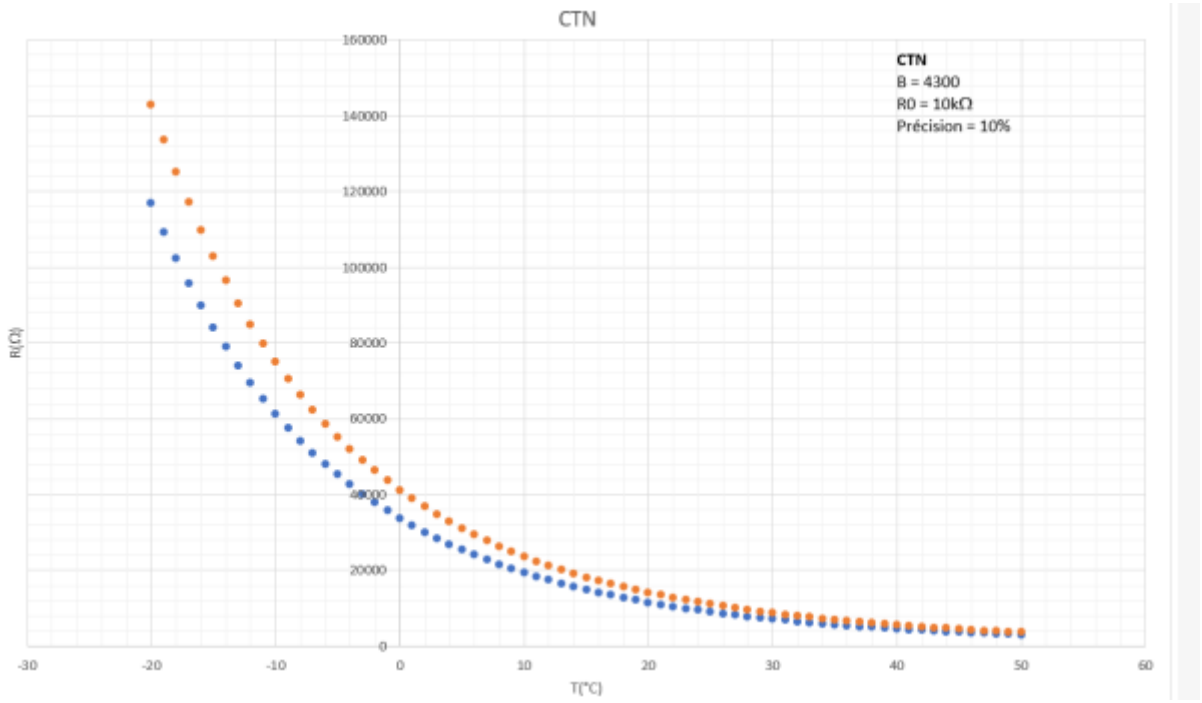
- *Un premier exemple pour tester le capteur*
  - Traitement à réaliser (**T=f(N)**) téléchargeable **ici**

2.3 CTN 10k

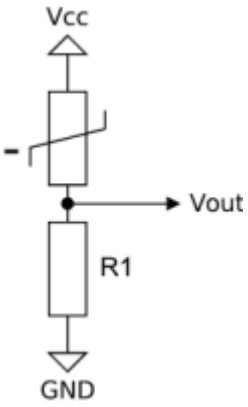
- *Distributeur* : Gotronic
- *Caractéristiques*
  - Résistance à 25°C : 10 kΩ
  - Puissance: 0.25 W.
  - Tolérance: ±10%.
  - B=4300.

- *Documentation*
  - Fichier Acrobat Reader à télécharger **ici**

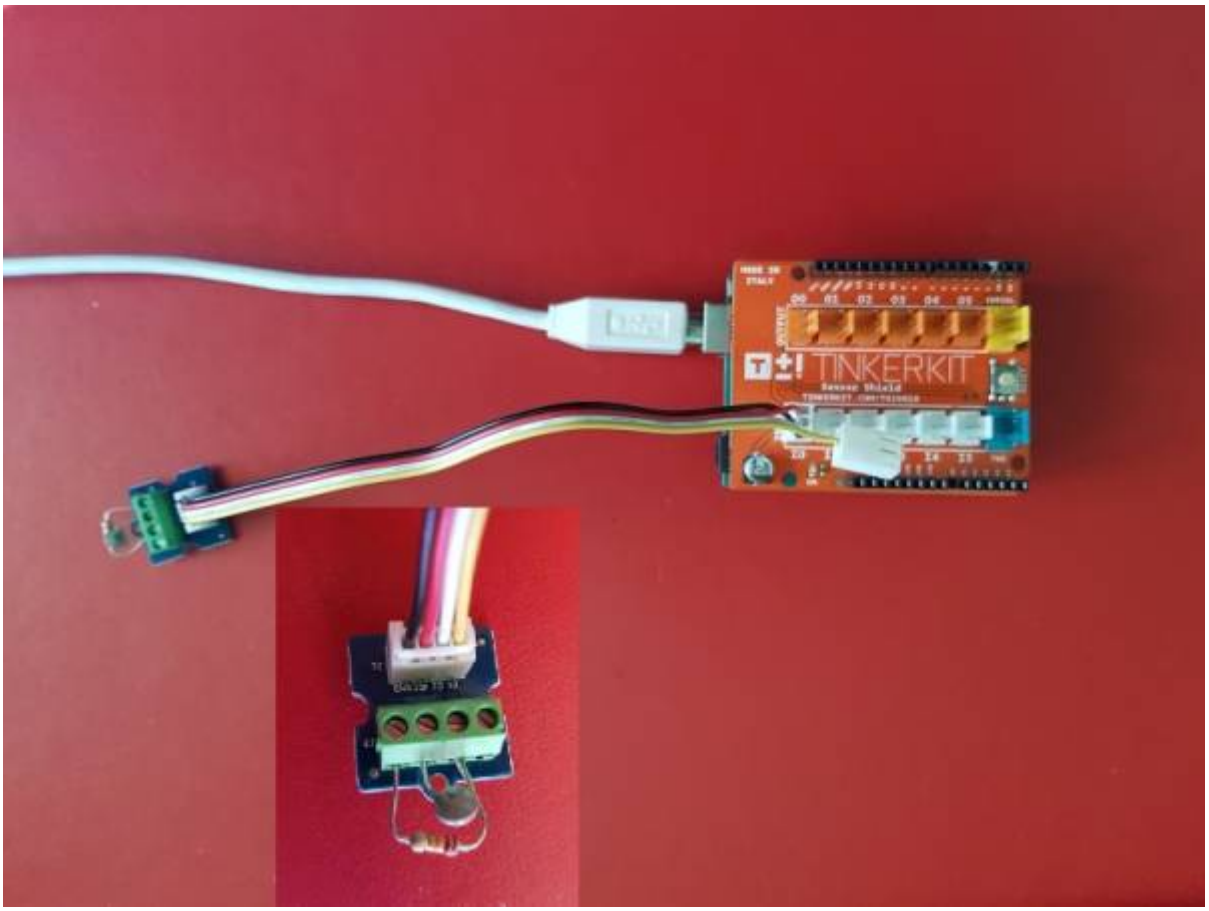
- *Modèle*
  - Résistance à 25°C : 10 kΩ
  - B=4300.
  - Tolérance: ±10%.



- *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**



- *Connexion à un shield* Tinkerkit v2 monté sur une Arduino Uno.



- *Un premier exemple pour tester le capteur*
  - Traitement à réaliser : T=f(N)

ctn.cpp

```
/*
Mesure de la température ambiante avec une CTN
Bibliothèque math.h : https://www.arduino.cc/en/math/h
*/
// Constantes
//--------------------------------------------------------
// CTN
const int Beta = 4300;   // Kelvin
const float T0 = 298.15; // Kelvin (25°C)
const int R0 = 10000;    // Résistance du capteur à 25°C
// Diviseur de tension
const int Vcc = 5;    // Volt
const int R1 = 12000; // Ohm
// CAN
const int n = 10;
const int VPE = 5;

// Variables
```

```
//-------------------------------------------------------
int CTN = A0;              // La CTN et son conditionneur sont connectés
sur la broche A0
int N = 0;                 // Image de la température, sortie du CAN
double temperature = 0.0; // Résultat du calcul de la température :
temperature=f(N)
// Coefficient du CAN
float kcan = pow(2, n) / VPE;
// Coefficients utilisés pour simplifier le calcul de la température
// k0, a
double k0 = kcan * Vcc * R1;
double a = R0 / exp(Beta / T0);
// k1, k2
double k1 = k0 / a;
double k2 = R1 / a;
//-------------------------------------------------------
void setup()
{
    Serial.begin(9600); // Fenêtre "serial" pour la mise au point
}
//-------------------------------------------------------
void loop()
{
    N = analogRead(CTN);
    temperature = Beta / log((k1 / N) - k2) - 273.15;
    Serial.println(temperature);
}
```

Le projet pour l'IDE **VSCode** de l'exemple ci-dessus est téléchargeable **ici**

3. Capteurs intégrés

3.1 LM35

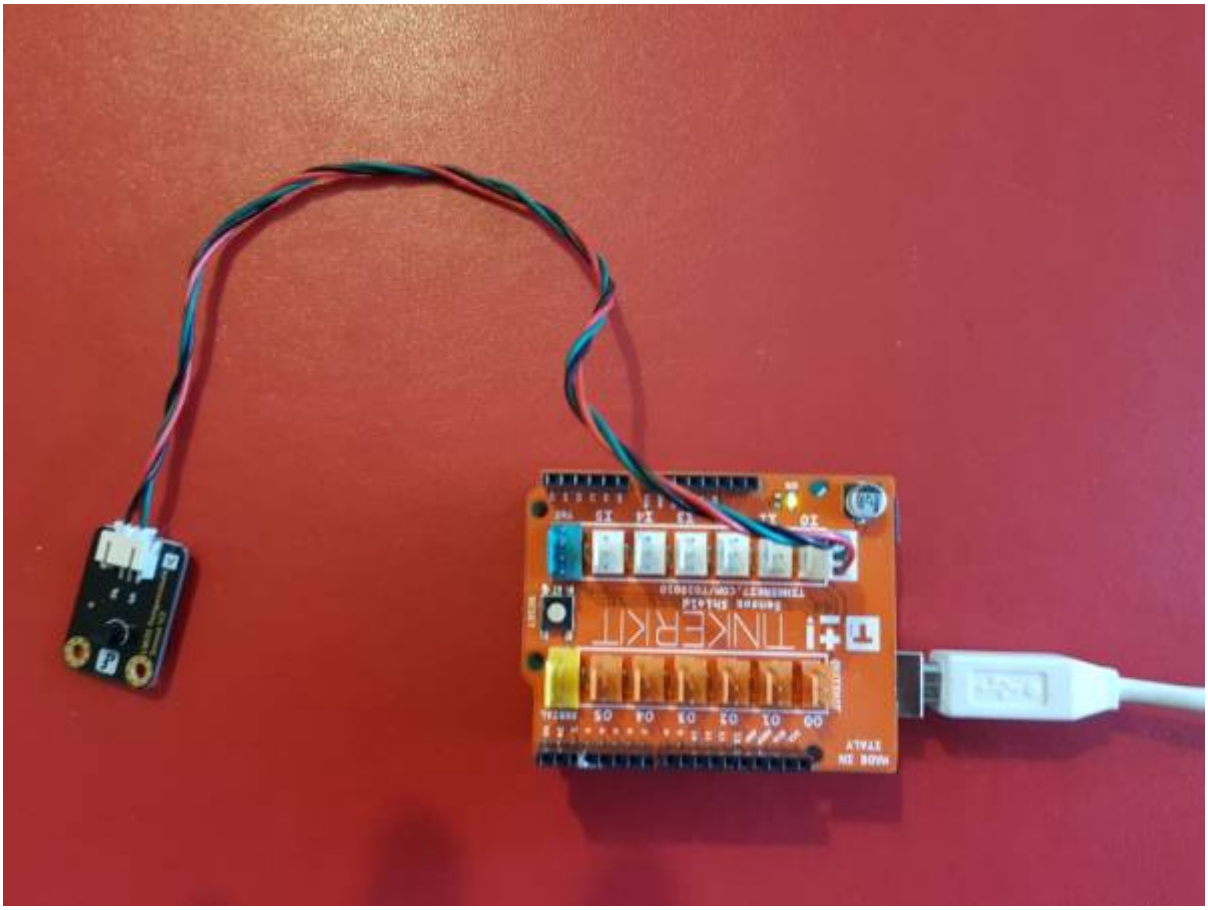
- *Source* : wiki

> Capteur de température analogique intégré.

- *Distributeur* : Gotronic
- *Caractéristiques*
  - Alimentation : 4 à 30 V
  - Plage de mesure : -55 / +150°C
  - Sensibilité : 10mV/°C
  - Précision : +/-0,5°C (à 25°C)
  - Boîtier : TO92

- *Documentation*
  - PDF à télécharger **ici**
- *Modèle*
  - Sensibilité : 10mV/°C

- *Aide pour la **simulation de la chaîne de mesure***
  - Le modèle à simuler est téléchargeable **ici**
- *Connexion à un shield* Tinkerkit v2 monté sur une Arduino Uno



- *Un premier exemple pour tester le capteur*

lm35.cpp

```
void setup()
{
    Serial.begin(9600); // Débit binaire : 9600 bps
}

void loop()
{
    uint16_t N;
    double temperature;
    // Lecture
    N=analogRead(A0);//LM35 connecté à Analog 0
    // Traitement
    temperature = (double) N * (5/10.24);
    // Ecriture
    Serial.print("Température:"); // Affiche la température sur le
moniteur
    Serial.print(temperature);
    Serial.println("C");
    delay(1000);
```

```
}
```

Le projet pour l'IDE **VSCode** de l'exemple ci-dessus est téléchargeable **ici**

Pour aller plus loin

Mesurer une température négative (**Télécharger**)

3.2 MM111

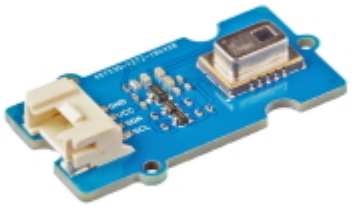
Capteur de température Velleman basé sur un amplificateur **MCP6L01T-E / LT** permettant de mesurer la **température** entre -50 et 150 °C. Il communique avec un microcontrôleur type Arduino ou compatible via une liaison analogique.

- *Distributeur* : Gotronic
- *Caractéristiques*
  - Alimentation: 5 Vcc
  - Plage de mesure: -50 à +150 °C
  - Sensibilité: 21 mV/°C
  - Précision: 0,4 °C
  - Dimensions: 22 x 22 x 5 mm

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

4. Capteurs numériques

4.1 Matrice de capteurs de température infrarouge (AMG8833) Grove



- *Source* : wiki seed studio

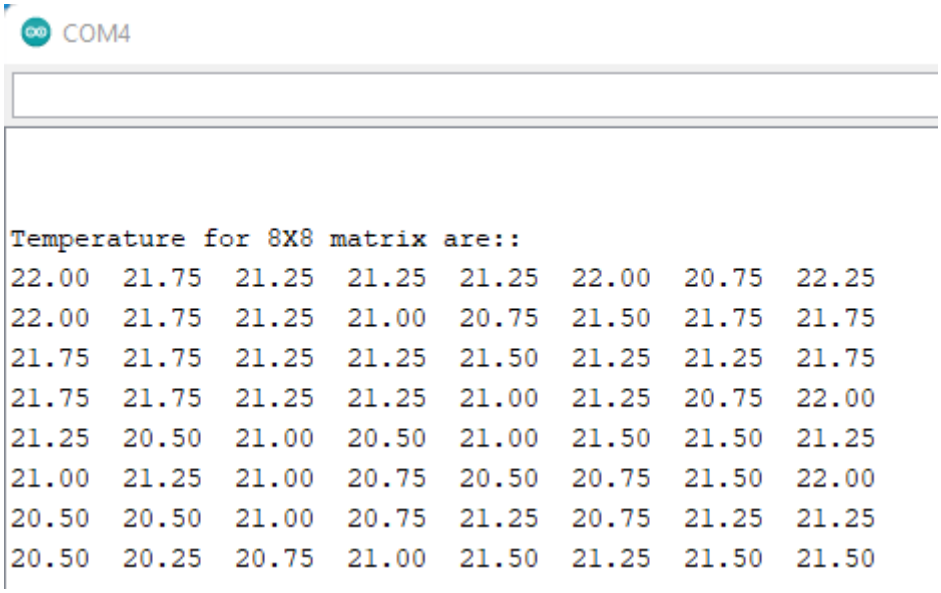
> Le réseau de capteurs de température infrarouge Grove (AMG8833) est un capteur de haute précision basé sur la technologie MEMS avancée. Il peut prendre en charge la détection de température d'une zone bidimensionnelle : 8 × 8 (64 pixels) et une distance de détection maximale de 7 mètres.

- *Distributeur* : Gotronic
- *Caractéristiques*
  - Alimentation: 3,3 ou 5 Vcc
  - Résolution: 8 x 8 pixels
  - Plage de mesure: 0 à 80 °C
  - Précision: ± 2,5 °C
  - Portée: ± 7 m
  - Angle de vision: 60 °
  - Interface: I2C
  - Adresses I2C: 0x68 par défaut (0x69 via un pont à souder)

- *Documentation*
  - PDF du datasheet **AMG8833** à télécharger **ici**
- *Bibliothèque à télécharger à partir de GitHub et à installer dans l'IDE* : Seeed_AMG8833

> Pour utiliser la bibliothèque *Seeed_AMG8833* avec un ESP, mettre la déclaration des types de données en commentaire.

- **Un premier exemple pour tester le capteur avec l'IDE Arduino**
  → Fichier → Exemples → Grove IR Matrix Temperature sensor AMG8833 → basic_demo.ino
  - *Résultat attendu*



- **Mise en oeuvre du capteur avec un afficheur OLED**
  - *Description* :
  - *Matériels*
    - Carte à microcontrôleur : Adafruit Feather Huzzah ESP8266 + Support Particle

Télécharger le projet PlatformIO pour VSCode.

4.1 MLX90614

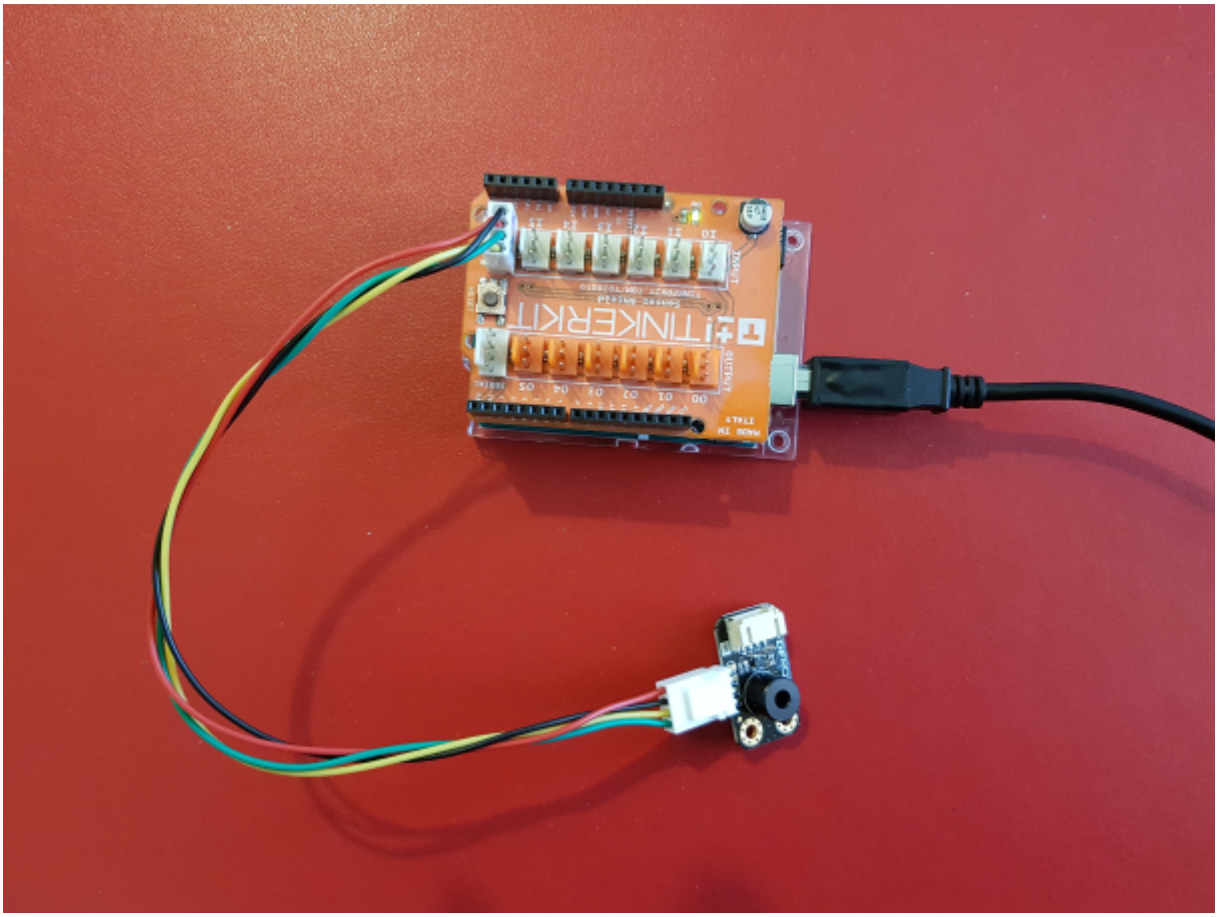
- *Source* : wiki

Ce module capteur de température IR sans contact est basé sur un MLX90614 et comporte un convertisseur analogique-numérique et un DSP (Digital Signal Processor) pour des résultats fiables et précis.

- *Distributeur* : Gotronic
- *Caractéristiques*
  - Alimentation: 3,3 à 5 Vcc
  - Consommation: 1,2 mA
  - Plage de mesure: -70 à 382 °C
  - Résolution: 0.01 °C
  - Interface: I2C
  - Dimensions: 32 x 18 mm

- *Documentation*
  - PDF à télécharger **ici**
- *Bibliothèques à installer dans l'IDE*

- *Connexion à un shield* Tinkerkit v2 monté sur une Arduino Uno



- *Un premier exemple pour tester le capteur*

mlx90614.cpp

```
/***************************************************
  This is a library example for the MLX90614 Temp Sensor

  Designed specifically to work with the MLX90614 sensors in the
  adafruit shop
  ----> https://www.adafruit.com/products/1748
  ----> https://www.adafruit.com/products/1749

  These sensors use I2C to communicate, 2 pins are required to
  interface
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ****************************************************/

#include <Wire.h>
#include <Adafruit_MLX90614.h>

Adafruit_MLX90614 mlx = Adafruit_MLX90614();

void setup() {
  Serial.begin(9600);

  Serial.println("Adafruit MLX90614 test");

  mlx.begin();
}

void loop() {
  Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempC());
  Serial.print("*C\tObject = "); Serial.print(mlx.readObjectTempC());
Serial.println("*C");
  Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempF());
  Serial.print("*F\tObject = "); Serial.print(mlx.readObjectTempF());
Serial.println("*F");

  Serial.println();
  delay(500);
}
```

4.2 TMP102

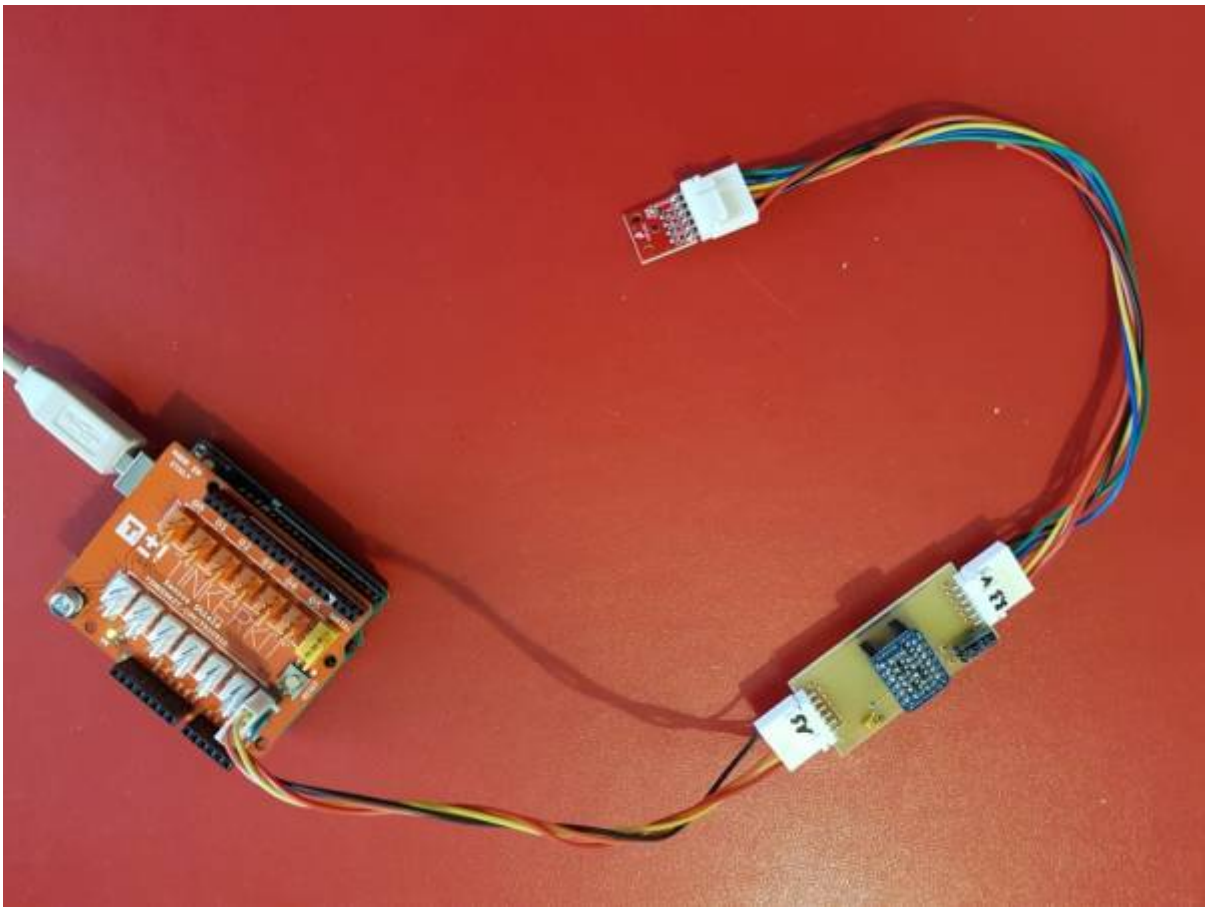
- *Source* : wiki

Le TMP102 est capable de lire des températures avec une résolution de 0,0625 °C et une précision allant jusqu'à 0,5 °C. La sortie possède des résistances intégrées de 4,7kΩ pour les communications I2C et fonctionne de 1,4V à 3,6V. La communication I2C utilise une signalisation à drain ouvert, il n'est donc pas nécessaire d'utiliser le décalage de niveau.

- *Distributeur* : Mouser
- *Caractéristiques*
  - Alimentation: 1,4 à 3,6 Vcc
  - Consommation: 10 µA maxi (1 µA en veille)
  - Plage de mesure: -40 °C à +125 °C
  - Précision: 0,5 °C (de -25 °C à +85 °C)
  - Résolution: 12bits, 0,0625 °C
  - Interface série 2 fils (I2C)
  - Dimensions: 16 x 16 mm

- *Documentation*
  - PDF à télécharger : Datasheet Résumé Datasheet
  - Schéma à télécharger **ici**
- *Bibliothèques à télécharger dans l'IDE*
  - Ressource : Description de la bibliothèque sparkfun et exemple de code

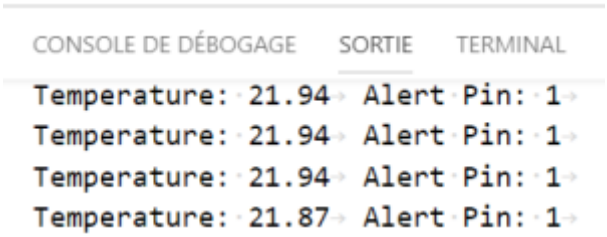
- *Connexion à un shield* Tinkerkit v2 monté sur une Arduino Uno.



- *Un premier exemple pour tester le capteur*

Arduino Examples → Examples from Custom Libraries → SparkFun_TMP102_Library → **SparkFun_TMP102_Breakout_Example.ino**

Exemple de résultat attendu



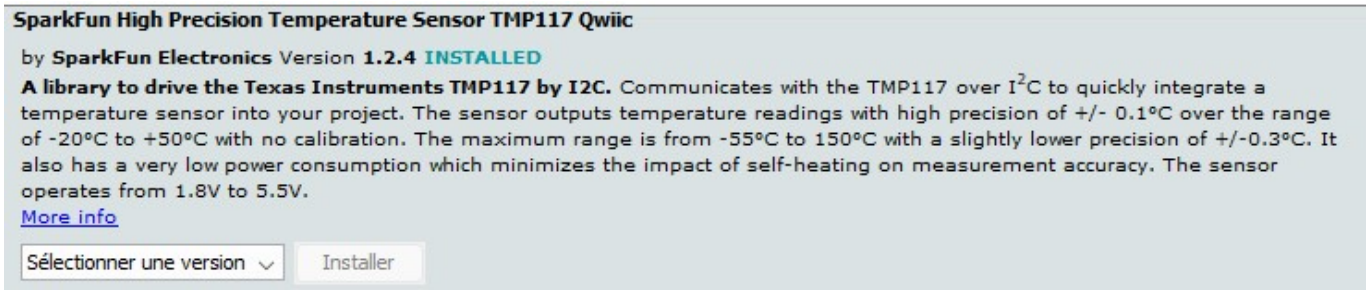
4.3 TMP117

- *Source* : wiki

> Le TMP117 fournit un résultat de **température 16 bits** avec une **résolution de 0,0078 °C** et une **précision allant jusqu'à ±0,1 °C** sur la plage de température de **-20 °C à 50 °C** sans étalonnage. Le TMP117 possède une interface compatible I2C et SMBus™, une fonctionnalité d'alerte programmable, et l'appareil peut prendre en charge jusqu'à quatre appareils sur un seul bus. Une EEPROM intégrée est incluse pour la programmation de l'appareil avec une mémoire supplémentaire de 48 bits disponible pour une utilisation générale.

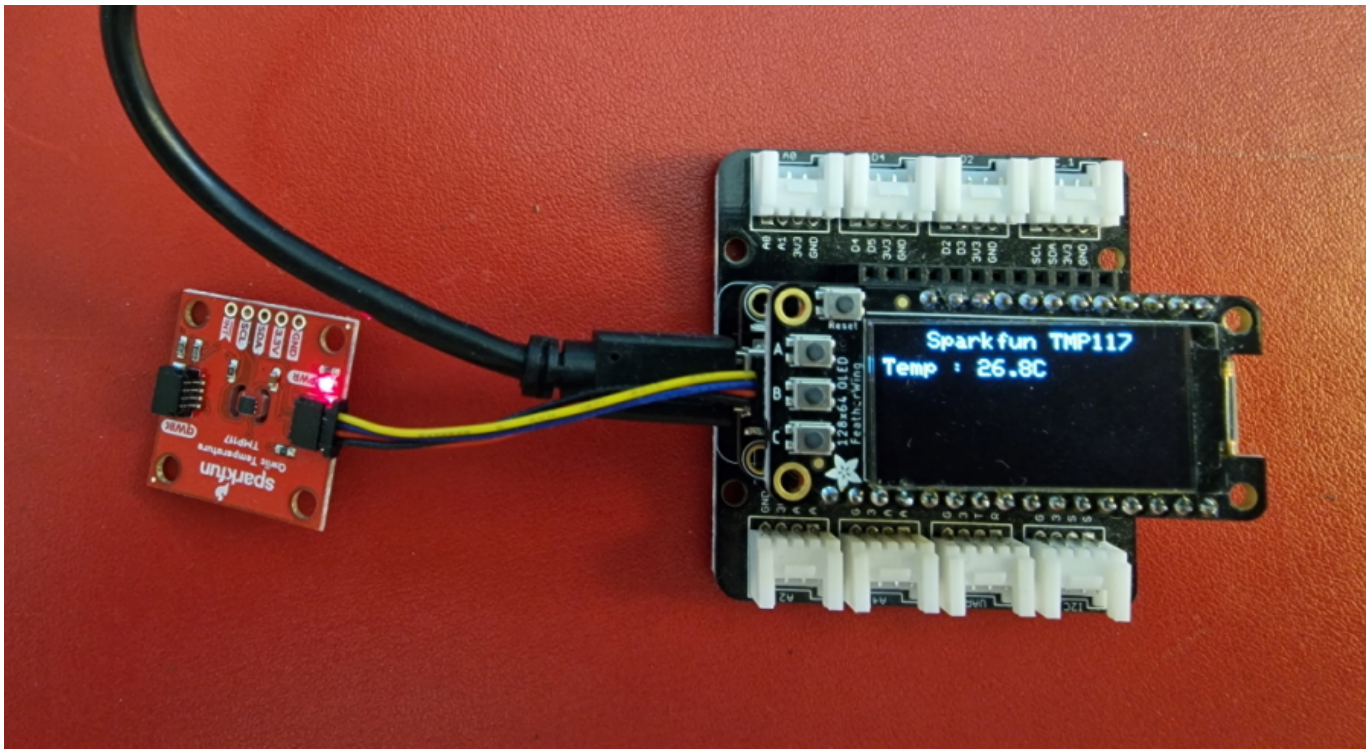
- *Distributeur* : Mouser
- *Caractéristiques*
  - Alimentation: 1,8 à 5,5 Vcc (3,3V avec câble Qwiic)
  - Consommation: 3,5 µA maxi (150 nA en veille)
  - Plage de mesure: -55 °C à +150 °C
  - Précision: 0,1 °C (de -20 °C à +50 °C)
  - Résolution: 16bits, 0,0078 °C
  - Interface série 2 fils (I2C)

- *Documentation*
  - PDF à télécharger : ici
  - Schéma à télécharger ici
- *Bibliothèques à télécharger dans l'IDE*
  - Ressource : Description de la bibliothèque sparkfun et exemple de code



- *GitHub*
  - SparkFun_TMP117_Arduino_Library

- **Un premier exemple pour tester le capteur avec l'IDE Arduino**
  → Fichier → Exemples → SparkFun_High_Precision_Temperature_Sensor_TMP117_Qwiic → **Example1_Basic_Readings.ino**
- **Mise en oeuvre avec un afficheur OLED**
  - *Description* : mesure de la température à l'aide d'un capteur **Sparkfun TMP117**, 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
  - Capteur de température : Sparkfun TMP117
- **Bibliothèques à installer dans l'IDE Arduino ou dans PlatformIO (VSCode)**
  - Adafruit GFX Library by Adafruit [GitHub]
  - Adafruit SH110X by Adafruit [GitHub]
  - SparkFun TMP117 by SparkFun [GitHub]
- *Code*

*.cpp

```
// Matériels : Adafruit Feather Huzzah ESP8266  + Support Particle,
Adafruit OLED SH1107, Sparkfun TMP117, câble Qwiic
// Logiciel : Arduino

// A ajouter
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>
#include <SparkFun_TMP117.h>

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

```
// Constructeurs
Adafruit_SH1107 display = Adafruit_SH1107(64, 128, &Wire);
TMP117 sensor; // L'adresse du circuit TMP117 est 0x48 = (GND) par défaut

void setup()
{
  // Bus I2C
  Wire.begin(); // Initialisation
  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
  // Test de la communication avec le capteur
  if (sensor.begin() == false)
  {
    display.println("DEFAUT(s)");
    display.print("1. Le capteur TMP117 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-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(20, 0);
  display.println("Sparkfun TMP117");

  // Mesure et affichage
  // Data Ready est un indicateur de mode de conversion - en conversion continue, l'indicateur dataReady doit toujours être haut
  if (sensor.dataReady() == true) // Affiche les valeurs de température que lorsque les données sont prêtes
  {
    display.setCursor(0, 12);
    display.print("Temp : ");
    display.print(sensor.readTempC(), 1);
    display.print("C");
    delay(500);
    display.display(); // Transfert du buffer sur l'écran
  }

  delay(10);
}
```

Télécharger le projet PlatformIO pour VSCode.