**Activité n°**⑤**: 3 capteurs de température**

L’idée est ici d’utiliser 3 capteurs de température différents : un capteur à brancher sur le breadbord(LM35), un capteur Grove et un capteur Grove étanche.

**Les buts des montage et objectifs :**

* Travail sur la notion de sortie numérique et analogique.
* Utiliser le moniteur et traceur série.
* Utiliser le Shield Grove.
* Utiliser différents capteurs de température.

**Matériel :**

* Arduino Uno
* Plaque d'essai (breadbord)
* Capteur de température LM 35
* Capteur de température Grove
* Sonde de température étanche
* Fils

|  |  |  |  |
| --- | --- | --- | --- |
|  | [LM35Z](#_I._Utilisation_du) | [Capteur de température Grove](#_II._Utilisation_du) | [DS18B20](#_III._Capteur_de) |
|  | Capteur de température LM35DZ - Boutique Semageek | Capteur de température Grove - Technologie Services | Capteur de température Grove 101990019 Seeed Studio - Capteurs de t° et  humidité | GO TRONIC |
| Plage de mesure | 0 -100 °C | -40 à +125 °C | -55°C à +125°C |
| Prix (€ TTC) | 2,60 | 3,25 | 7,60 |
| Doc. Technique | <https://www.gotronic.fr/art-lm35dz-10295.htm#complte_desc> | <https://www.gotronic.fr/art-capteur-de-temperature-grove-101020015-18965.htm> | <https://www.gotronic.fr/art-capteur-de-temperature-grove-101990019-23842.htm> |
| Avantages  Inconvénients | **Peu cher**  **Code modifiable facilement**  Montage peu pratique | Nécessite un Shield. | **Etanche.**  Code peu modifiable  Nécessite un Shield.  La partie cordon ne doit pas rester un long moment au-dessus de 70°C. |

# I. Utilisation du capteur LM35.

**Montage et branchements :**

|  |  |
| --- | --- |
| Capteur de température LM35DZ - Boutique Semageek |  |

**Document 1 : Caractéristiques du capteur LM35 : (extrait de données trouvées sur internet)**

Les caractéristiques techniques du module sont les suivantes :

* Calibrated Directly in Celsius (Centigrade)
* Linear + 10-mV/°C Scale Factor
* 0.5°C Ensured Accuracy (at 25°C)
* Rated for Full −55°C to 150°C Range

Ce composant présente l’avantage d’être linéaire sur la plage de mesure : c’est-à-dire que 1°C correspond à 10 mV.

|  |
| --- |
| **Document 2 : Code Arduino pour mesurer la température** |

void setup() {

Serial.begin(9600); ///initialisation de la communication

}

void loop() {

int valeur\_brute = analogRead(A0); ///on utilise la broche A0 pour lire les données analogiques

float temperature\_celcius = valeur\_brute \* (5.0 / 1023.0 \* 100.0);

Serial.println(temperature\_celcius);

delay(500);

}

**Document 3 : Fonctions utiles**

Attention à la ponctuation et à la casse.

|  |  |
| --- | --- |
| **Fonction** | **Rôle** |
| pinMode(**➀**, **➁**) ; | Permet de configurer le port de connexion de la carte en entrée ou en sortie.  **➀** : Ecrire le numéro du port de la carte à configurer (1, 2, 3, 4).  **➁** : Ecrire **OUTPUT** pour que le port soit une sortie.  Ecrire **INPUT** pour que le port soit une entrée. |
| digitalWrite(**➀**, **➁**) ; | Permet de fixer l’état du port de connexion.  **➀** : Ecrire le numéro du port concerné.  **➁** : Ecrire **HIGH** pour un état haut (1 logique soit 5V électrique).  Ecrire **LOW** pour un état bas (0 logique ou 0V électrique). |
| delay(**➀**) ; | Permet de programmer une temporisation.  **➀** : Ecrire la durée en millisecondes.  delay(1000) correspondra à 1000 ms d’attente. |

**Questions :**

1. Réaliser le montage à l’aide du *« breadboard* ».
2. Compléter la ligne 6.
3. Ecrire le calcul à faire pour calculer la température en degrés Celsius.
4. Compléter alors la ligne 9 pour calculer la température. (Voir aide éventuelle)
5. Réaliser le programme et le téléverser sur Arduino® et le tester.

Rappel : La tension mesurée sur la broche A0 est de 5 V maximum et on utilise 1023 bits pour coder l’information.

On peut mesurer une température qui varie entre 0°C et 100°C.

**Aide question 4**

① valeur\*(5.0/1023.0\*100)

② valeur\*(1023.0/5.0\*100)

③ valeur/(5.0/1023.0\*100)

④ valeur\*(5/1023\*100)

**Corrigé**

La proposition n°1 est la bonne.

La proposition n°4 est juste aussi mais elle posera problème car on utilise des valeurs entières. Le résultat du calcul sera donc une valeur entière. Ecrire 5.0 1023.0 permet de passe avec des valeurs décimales (float)

Amélioration de la précision (<https://www.carnetdumaker.net/articles/la-conversion-analogique-numerique-avec-arduino-genuino/>)

Pour qu'un convertisseur analogique / numérique puisse faire son travail, il a besoin d'une tension maximum de référence. Cette tension s'appelle AREF par convention.

Pouvoir choisir cette tension est très intéressant, car cela permet de réduire la plage de tension de mesure et donc d'améliorer drastiquement la précision des mesures.

Si on mesure un signal analogique toujours compris entre 0 volt et 1,5 volt, comme c'est le cas avec un capteur LM35 par exemple, il est dommage d'avoir une référence à 5V, car plus de la moitié des points de mesure ne sont jamais utilisés.

Dans le cas d'un LM35 de gammes "grand publique" ou "industrielle", la tension maximum est même de 1.1v (= 110°C), avoir une tension de référence à 5V est donc dramatique pour la précision de la mesure. Pour rappel, 5 volts / 1024 points (10 bits) = 4.89mV par point.

Les microcontrôleurs, comme ceux utilisés dans les cartes Arduino, disposent généralement d'une broche spéciale nommée AREF sur laquelle on peut venir injecter sa propre tension de référence. Il est aussi possible d'utiliser une ou plusieurs tensions de référence interne en fonction du microcontrôleur.

Avec une carte Arduino classique (Uno, Mega2560, etc.), il est possible de choisir la référence de tension en utilisant la fonction [analogReference()](https://www.arduino.cc/en/Reference/AnalogReference).

Dans notre cas, plusieurs choix sont possibles :

* utiliser le mode EXTERNAL en reliant la broche AREF à la sortie +3V de la carte Arduino,
* utiliser le mode INTERNAL (Arduino UNO) ou INTERNAL1V1 (Arduino Mega2560).

L'utilisation du mode EXTERNAL et de la broche AREF a l'avantage d'être générique. Cela fonctionne sur n'importe quelle carte Arduino classique. Cependant, la seule tension autre que 5 volts disponible sur une carte Arduino est de 3.3 volts. Or, 3.3 volts / 1024 points (10 bits) = 3.23mV par point, c'est mieux, mais encore loin d'être parfait.

En utilisant le code suivant avec le montage du chapitre précédent, vous devriez constater une amélioration de la précision de la mesure :

*//Code d'exemple pour le capteur LM35 (2°C ~ +110°C) avec une meilleure précision.*

*// Fonction setup(), appelée au démarrage de la carte Arduino*

void setup() {

*// Initialise la communication avec le PC*

Serial.begin(9600);

*// Améliore la précision de la mesure en réduisant la plage de mesure*

analogReference(INTERNAL); *// Pour Arduino UNO*

*// analogReference(INTERNAL1V1); // Pour Arduino Mega2560*

}

*// Fonction loop(), appelée continuellement en boucle tant que la carte Arduino est alimentée*

void loop() {

*// Mesure la tension sur la broche A0*

int valeur\_brute = analogRead(A0);

*// Transforme la mesure (nombre entier) en température via un produit en croix*

float temperature\_celcius = valeur\_brute \* (1.1 / 1023.0 \* 100.0);

*// Envoi la mesure au PC pour affichage et attends 250ms*

Serial.println(temperature\_celcius);

delay(250);

L'extrait de code ci-dessus est disponible en téléchargement sur [cette page](https://www.carnetdumaker.net/snippets/15/) (le lien de téléchargement en .zip contient le projet Arduino prêt à l'emploi).

**N.B. L'appel à la fonction analogReference() doit être fait au démarrage avant de faire le moindre appel à analogRead().**

Pour mesure des températures négatives, c’est plus complexe puisqu’il faut utiliser une alimentation en tension négative.

# II. Utilisation du capteur grove.

**Montage et branchements :**

On utilise maintenant le Shield et le capteur de température Grove.

On donne le code Arduino permettant de mesurer la température avec ce capteur.

**Document 1 : Code Arduino pour mesurer la température**

const int B =; // B value of the thermistor

const int R0 =; // R0 = 100k

const int pinTempSensor = A0; // Grove - Temperature Sensor connect to A0

#if defined(ARDUINO\_ARCH\_AVR)

#define debug Serial

#elif defined(ARDUINO\_ARCH\_SAMD) || defined(ARDUINO\_ARCH\_SAM)

#define debug SerialUSB

#else

#define debug Serial

#endif

void setup()

{

Serial.begin(9600);

}

void loop()

{

int a = analogRead(pinTempSensor);

float R = 1023.0/a-1.0;

R = R0\*R;

float temperature = 1.0/(log(R/R0)/B+1/298.15)-273.15;

Serial.print("temperature = ");

Serial.println(temperature);

delay(1000);

**Document 2 : Caractéristiques du capteur Grove : (extrait de données trouvées sur internet)**

Les caractéristiques techniques du module sont les suivantes :

* Voltage: 3.3 ~ 5V
* Zero power resistance: 100 KΩ
* Resistance Tolerance: ±1%
* Operating temperature range: -40 ~ +125 ℃
* Nominal B-Constant： 4250 ~ 4299K

Source : <https://wiki.seeedstudio.com/Grove-Temperature_Sensor_V1.2/>

**Questions :**

La formule permettant de calculer la température en Kelvin est

1. Modifier cette formule pour obtenir la température en °C.
2. Ecrire cette formule en utilisant la syntaxe Arduino (sur une seule ligne)
3. Si la valeur de B pour les mesures que l’on souhaite faire est de 4275, compléter la ligne adéquate.
4. Compléter également la ligne adéquate pour la valeur de la résistance de référence.
5. Sur quelle broche est branché le capteur de température ?

Corrigé :

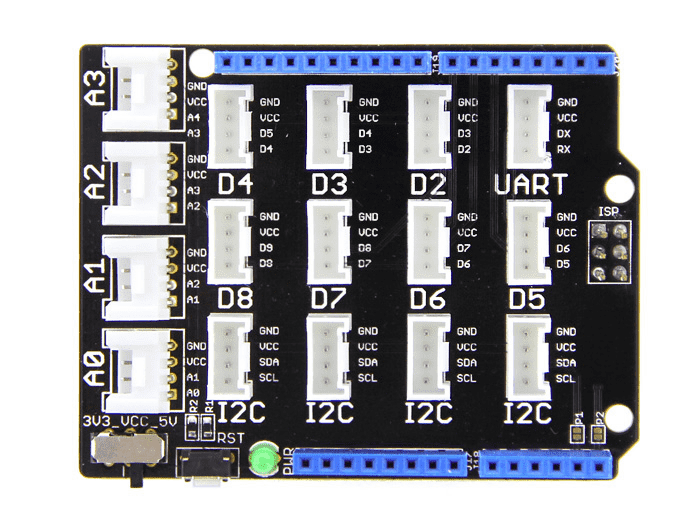
1.

2. 1.0/(log(R/R0)/B+1/298.15)-273.15;

3. ligne 1 B = 4275

4. Ligne 2 R0 = 100000 (100 kOhm)

5. Sur la broche A0 (ligne 3)



# III. Capteur de sonde étanche

**Montage et branchements :**

On utilise le capteur étanche.

**Document 1 : Code Arduino pour mesurer la température**

// Ajout des bibliothques

#include <OneWire.h>

#include <DallasTemperature.h>

// Choix du port sur le shield Arduino

#define ONE\_WIRE\_BUS 2

OneWire oneWire(ONE\_WIRE\_BUS);

void setup(void)

{

// Démarrage du port série

Serial.begin(115200);

// Démarrage des bibliothèques

sensors.begin();

}

void loop(void)

{

float tempC = sensors.getTempCByIndex(0);

// Vérification de la lecture des données

if(tempC != DEVICE\_DISCONNECTED\_C)

{

Serial.println(tempC);

Delay(1000) ;

}

else

{ Serial.println("Erreur: impossible de lire les données. Vérifiez la connectique");

**Questions :**

1. Réaliser le montage électrique et téléverser le programme Arduino.
2. Sur quel port allez-vous brancher la sonde de température ?
3. Vérifier que la température s’affiche bien (en faisant outils 🡪 moniteur série)
4. Modifier la ligne delay(1000); et remplacer là par delay(3000); Téléverser de nouveau le programme. Que pouvez-vous en conclure sur l’effet de cette ligne. Quelle valeur aller vous garder ? Quelle est l’unité de la grandeur indiquée en parenthèse
5. Dans quelle variable est stockée l’information de la température ?
6. Quel est l’effet de l’instruction Serial.println()? (Pour tester remplacer la ligne Serial.println(tempC); par Serial.print(tempC); ). N’oubliez pas de téléverser le programme.