

Étalonnage d'un capteur

1. Principe et application

Comment faire si l'on possède un capteur mais que l'on a pas sa caractéristique exacte et faire correspondre une valeur analogique lue avec une donnée physique réelle ?

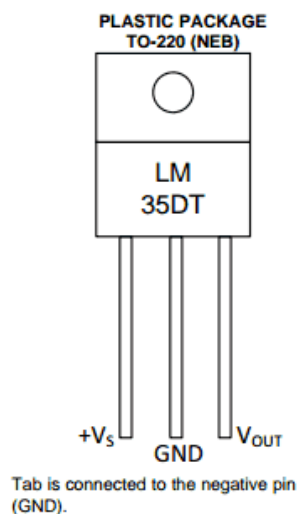
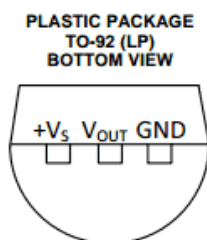
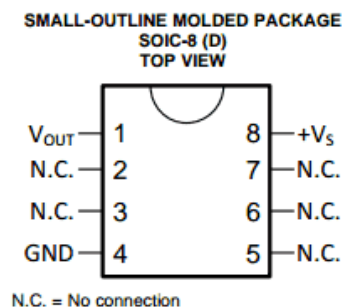
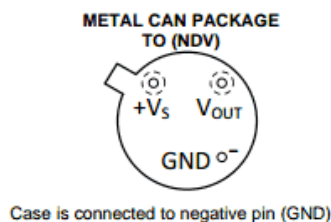
Par exemple, un capteur de température alimenté en 5V sur telle et telle broches, le signal de sortie est une tension en fonction de la température mais on est incapable de dire quelle en est la caractéristique (la courbe de tension en fonction de la température)".

Nous allons maintenant voir comment résoudre ce problème en prenant connaissance d'une méthode pour étalonner un capteur. Nous allons ainsi nous-mêmes déterminer la courbe caractéristique du capteur et déterminer son coefficient liant la température et la tension.

2. Le capteur utilisé

Pour mettre en œuvre la méthode, nous allons utiliser un capteur de température assez répandu qui se nomme "LM35". Il existe dans différents boîtiers que voici :

CONNECTION DIAGRAMS



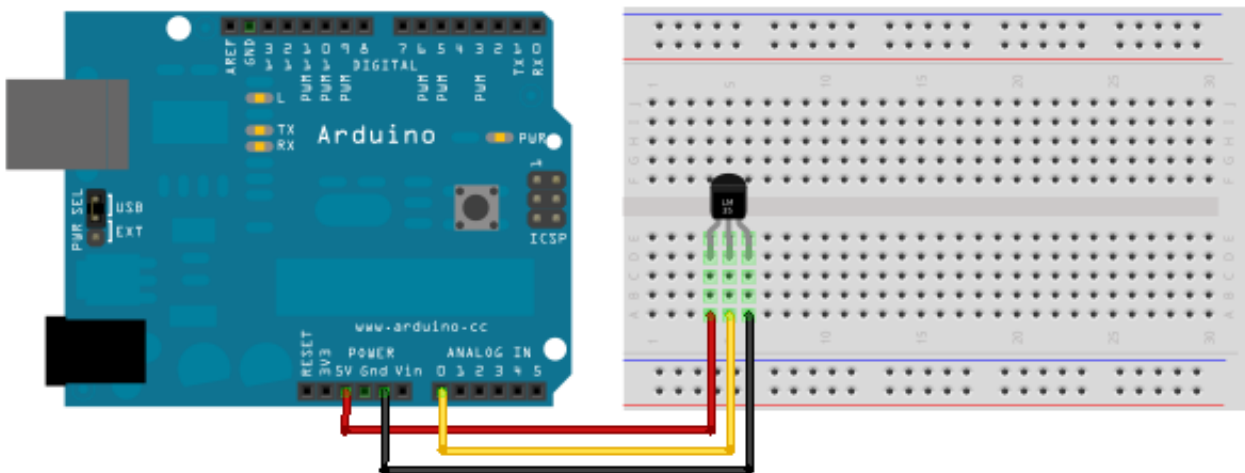
Le branchement, il est assez simple. Il suffit de relier +VS au 5V et GND à la masse. Le signal sera ensuite lu sur la broche Vout.

3. La méthode

La méthode pour caractériser un capteur est assez simple. À l'aide d'une multitude de mesures et d'un appareil témoin, nous allons créer un tableau qui nous servira à calculer la courbe liant la tension à la donnée mesurée (à l'aide d'un [tableur](#)). Pour cela, en plus de votre capteur vous aurez besoin d'un appareil de mesure "témoin" qui vous servira de référence. Par exemple le bon vieux thermomètre qui traîne accroché à votre fenêtre fera parfaitement l'affaire !

3.1. Prise de mesures

Reliez le capteur à l'Arduino et l'Arduino à l'ordinateur, de la manière la plus simple possible, comme ceci par exemple :



Ensuite, nous devons récupérer les données envoyées par le capteur de manière régulière (ou rajouter un bouton et faite des envois lors de l'appui). Pour cela, voici un petit programme qui enverra les valeurs brutes ou converties en volts toutes les demi-secondes :

```
const int capteur = 0; //capteur branché sur la pin analogique 0

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

void loop()
{
  int valeur = analogRead(capteur);
  float tension = (valeur*5.0)/1024;

  Serial.print("Tension : ");
  Serial.print(tension);
  Serial.println(" V");
  Serial.print("Valeur : ");
  Serial.println(valeur);
  Serial.println("-----");

  delay(500);
}
```

Maintenant que tout est prêt, il nous faut un banc de test. Pour cela, préparez une casserole avec de l'eau contenant plein de glaçons (l'eau doit être la plus froide possible). Faites une première mesure avec votre capteur plongé dedans (attention, les broches doivent être isolées électriquement ou alors mettez l'ensemble dans un petit sac plastique pour éviter que l'eau n'aille faire un court-circuit). Faites en même temps une mesure de la température réelle observée à l'aide du thermomètre. Une fois cela fait, relevez ces mesures dans un tableau qui possédera les colonnes suivantes :

- Température réelle (en °C)
- Tension selon Arduino (en V)
- Valeur brute selon Arduino

Quand la première mesure est faite, commencez à faire réchauffer l'eau (en la plaçant sur une plaque de cuisson par exemple). Continuez à faire des mesures à intervalle régulier (tous les 5 degrés voire moins par exemple). Plus vous faites de mesure, plus l'élaboration de la courbe finale sera précise. Voici à titre d'exemple le tableau obtenu :

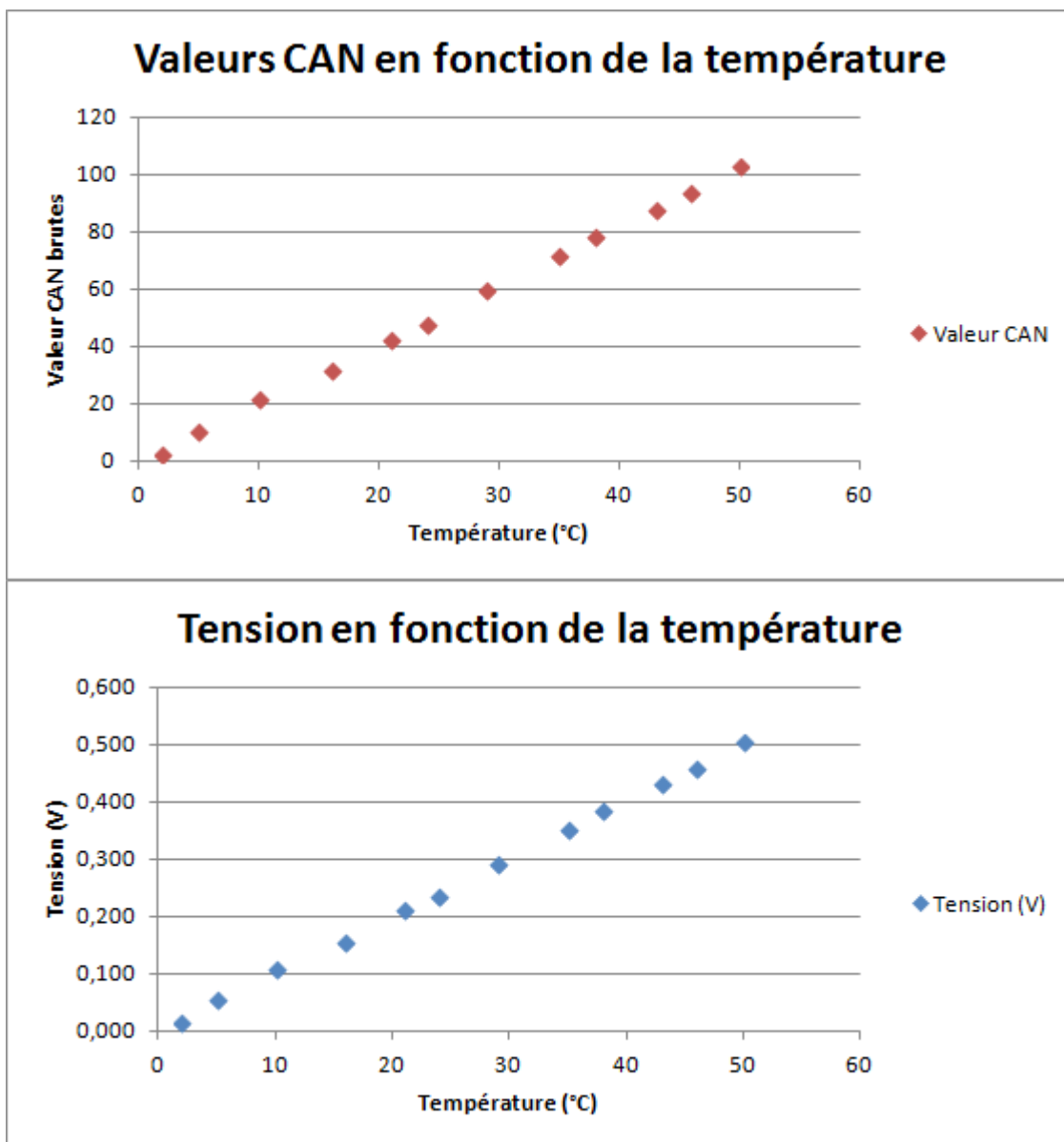
Température (°C)	Tension (V)	Valeur CAN
------------------	-------------	------------

2	0,015	3
5	0,054	11
10	0,107	22
16	0,156	32
21	0,210	43
24	0,234	48
29	0,293	60
35	0,352	72
38	0,386	79
43	0,430	88
46	0,459	94
50	0,503	103

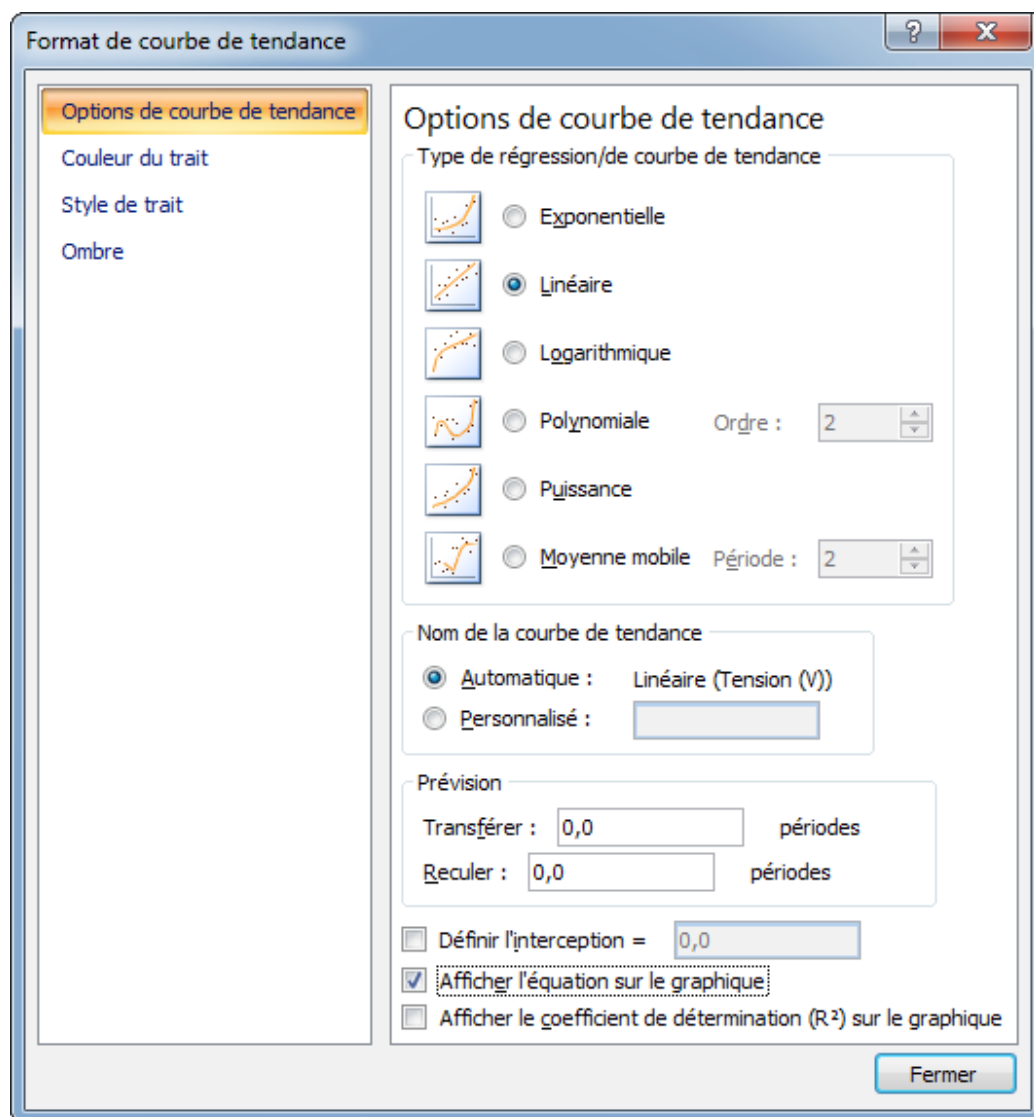
3.2. Tracé de la caractéristique

Lorsque vous avez fini de prendre toutes vos valeurs, vous allez pouvoir passer à l'étape suivante qui est : Calculer la caractéristique de votre courbe

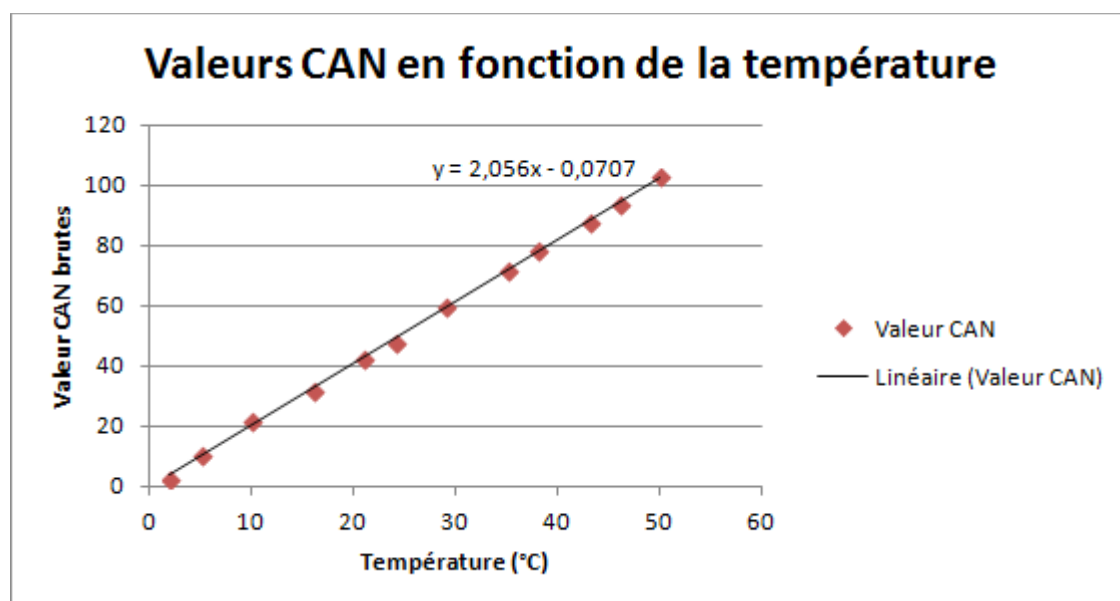
On va en faire deux, une symbolisant les valeurs brutes de la conversion du CAN (entre 0 et 1023) en rouge et l'autre qui sera l'image de la tension en fonction de la température en bleu. Nous pourrons alors déterminer deux caractéristiques, selon ce qui vous arrange le plus.

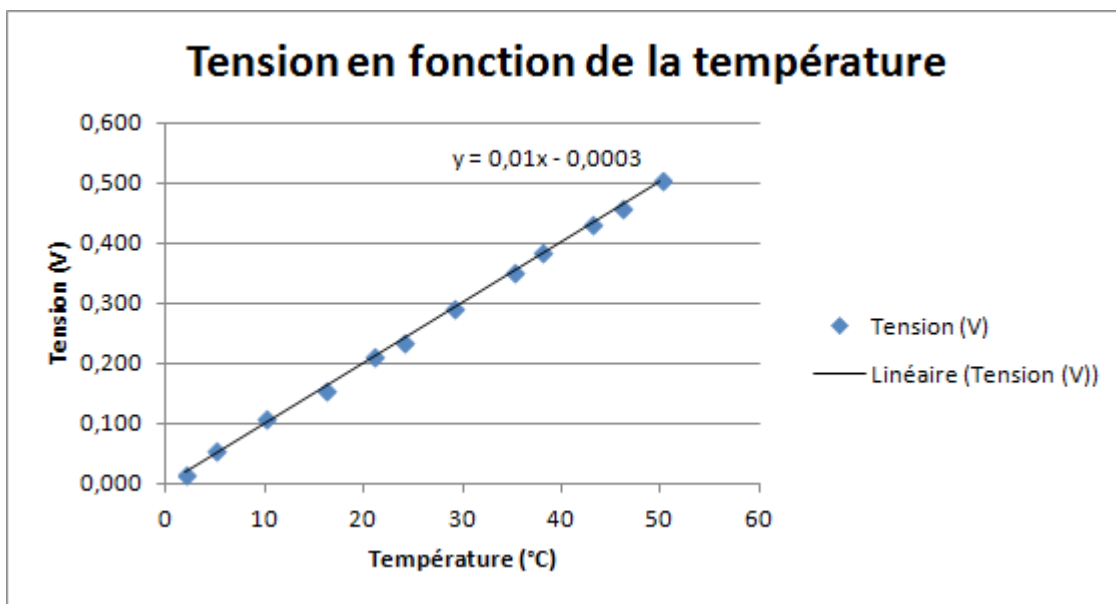


Une fois cela fait, il ne reste plus qu'à demander au logiciel de graphique de nous donner la **courbe de tendance** réalisée par ces points. Sous LibreOffice calc, il suffit de cliquer sur un des points du graphique et choisir ensuite l'option "Ajouter une courbe de tendance...". Vous aurez alors le choix entre différents types de courbe (linéaire, exponentielle...). Ici, on voit que les points sont alignés, il s'agit donc d'une équation de courbe linéaire, de type $y = a.x + b$. Cochez la case "Afficher l'équation sur le graphique" pour pouvoir voir et exploiter cette dernière ensuite.



Voici alors ce que l'on obtient lorsque l'on rajoute notre équation :





Grâce à l'équation, nous pouvons déterminer la relation liant la température et la tension (ou les valeurs du CAN). Ici nous obtenons :

- $y = 0,01x - 0,0003$ (pour la tension)
- $y = 2,056x - 0,0707$ (pour les valeurs du CAN)

Le coefficient constant (-0,003 ou -0,0707) peut ici être ignoré. En effet, il est faible (on dit **négligeable**) comparé aux valeurs étudiées. Dans les équations, x représente la température et y représente la tension ou les valeurs du CAN. On lit donc l'équation de la manière suivante : Tension en Volt égale 0,01 fois la température en degrés Celsius. Ce qui signifie que dorénavant, en ayant une mesure du CAN ou une mesure de tension, on est capable de déterminer la température en degrés Celsius.

Par exemple, si nous avons une tension de 300mV, avec la formule trouvée précédemment on déterminera que l'on a $0,3 = 0,01 \times \text{Temperature}$, ce qui équivaut à $\text{Temperature} = 0,3 / 0,01 = 30^\circ\text{C}$. On peut aisément le confirmer via le graphique.

Pour obtenir de l'aide sur le tableur



Pour obtenir de l'aide sur le grapheur



3.3. Adaptation du code

Puisque nous savons mesurer les valeurs de notre capteur et que nous avons une équation caractéristique, nous pouvons faire le lien en temps réel dans notre application pour faire une utilisation de la grandeur *physique* de notre mesure. Par exemple, s'il fait 50°C nous allumons le ventilateur. En effet, souvenez-vous, avant nous n'avions qu'une valeur entre 0 et 1023 qui ne signifiait physiquement pas grand chose. Maintenant nous sommes en mesure de faire la conversion. Il faudra pour commencer récupérer la valeur du signal.

Prenons l'exemple de la lecture d'une tension analogique du capteur précédent :

```
int valeur = analogRead(monCapteur); //lit la valeur
```

Nous avons ensuite deux choix, soit nous le transformons en tension puis ensuite en valeur physique grâce à la caractéristique du graphique bleu ci-dessus, soit nous transformons directement en valeur physique avec la caractéristique rouge. Prenons la dernière solution. Pour rappel, la formule obtenue était : $y = 2,056x - 0,0707$. Nous avons aussi dit que le facteur constant était négligeable, on a donc de manière simplifiée $y = 2,056x$ soit “la température est égale à la valeur lue divisée par 2,056 ($x = y/2,056$)). Nous n’avons plus qu’à faire la conversion dans notre programme.

```
float temperature = valeur/2.056;
```

Si l’on voulait écrire un programme plus complet, on aurait :

```
const int monCapteur = 0; //Capteur sur la broche A0;
```

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

void loop()
{
    int valeur = analogRead(monCapteur);
    float temperature = valeur/2.056;

    Serial.println(temperature);

    delay(500);
}
```

Et si jamais le coefficient n’est pas négligeable ?

Admettons qu’on obtienne la caractéristique suivante : $y = 10x + 22$ On pourrait lire ça comme “ma valeur lue par le CAN est égale à 10 fois la valeur physique plus 22”. Si on manipule l’équation pour avoir x en fonction de y, on aurait :

$$y = 10x + 22$$

$$x = (y - 22) / 10$$

Dans le code, cela donnerait :

```
void loop()
{
    int valeur = analogRead(monCapteur);
    float temperature = (valeur-22)/10;

    Serial.println(temperature);

    delay(500);
}
```