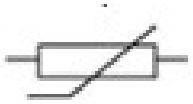
Le but de la séance est de fabriquer un indicateur lumineux du niveau de température.

Caractéristique d'une thermistance.

Document 1 : Un capteur électrique, la thermistance

Un capteur est composant électronique permettant de faire le lien entre une grandeur physique que l'on souhaite mesurer (ici la température) et une tension électrique.
Une thermistance est l'un des principaux capteurs de température utilisé en électronique et basé sur la variation de la résistance électrique en fonction de la température.
Dans le cadre de ce TP, une thermistance de type CTN (Coefficient de Température Négatif) est utilisée : sa résistance diminue lorsque la température augmente et inversement.
Son symbole est le suivant :

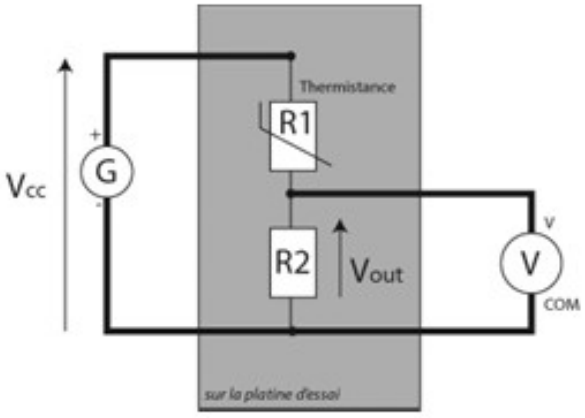


Document 2 : Tracé de la courbe d'étalonnage d'une thermistance

Chaque capteur possède ses propres caractéristiques : la courbe d'étalonnage d'une thermistance sert à établir une relation mathématique entre une tension mesurée en sortie d'un montage électrique contenant la thermistance et la température du milieu.
Pour obtenir une telle courbe il faut faire varier la température du milieu, Θ (en °C) et mesurer la tension de sortie, Vout (en V).
Remarque : la tension mesurée n'est pas directement celle aux bornes de la thermistance, on utilise un pont diviseur de tension (PDDT) présenté dans le document 3.
Les valeurs de V_{out} et de Θ sont relevées directement dans un tableur, tout au long de l'expérimentation.
La courbe d'étalonnage correspond à la modélisation de la représentation graphique Vout = f(Θ) obtenue, par le modèle mathématique le plus adéquat.

Document 3 : Le pont diviseur de tension

Le montage électrique à réaliser utilise un pont diviseur de tension*, une résistance, R = 1,0 kΩ, une thermistance CTN de résistance variable R1, un générateur de tension continu VCC = 5V et un voltmètre permettant de mesurer la tension V_{out} qualifiée de tension de « sortie » du montage.



* Un pont diviseur de tension est un montage qui permet de diviser la tension d'alimentation (U_{AC}). Ainsi la tension de « sortie » va pouvoir varier de 0 à 5V.

Formule du pont diviseur de tension : $V_{out} = \frac{R2}{R1+R2} \times Vcc$

Question préliminaire

- A partir des documents 1 et 3, compléter le texte suivant :
- La tension d'alimentation V_{CC} est constante : V_{CC} = …… .
- D'après la loi des mailles, si U_{R1} diminue alors V_{out} …… et inversement.
- Lorsque θ augmente alors R_1 …… et U_{R1} …… (d'après la loi d'Ohm).
- Ainsi, lorsque θ augmente alors V_{out} (tension mesurée) …… .

Travail à réaliser

- À l'aide des documents 1 à 3, proposer une démarche expérimentale permettant d'utiliser une thermistance comme capteur de température ?
- Une fois validé par le professeur, mettre en œuvre le protocole expérimental.

Quelques conseils :

- Utiliser une gamme de température comprise entre la température ambiante et 50°C.
- Pour éviter un court-circuit, les soudures de la thermistance avec les fils monobrins peuvent être protégées avec une gaine thermo-rétractable.
- Dans le cas d'une manipulation en binôme, relever directement les valeurs expérimentales dans un tableur-grapheur.
- À partir des fonctionnalités du tableur-grapheur, modéliser les points expérimentaux par la fonction mathématique qui vous semble la plus adéquate.
- Relever l'équation obtenue : $V_{out} = f(\Theta)$.
- À partir de l'équation de la courbe d'étalonnage précédente, exprimer Θ en fonction de V_{out}.

Capteur de température piloté par Arduino

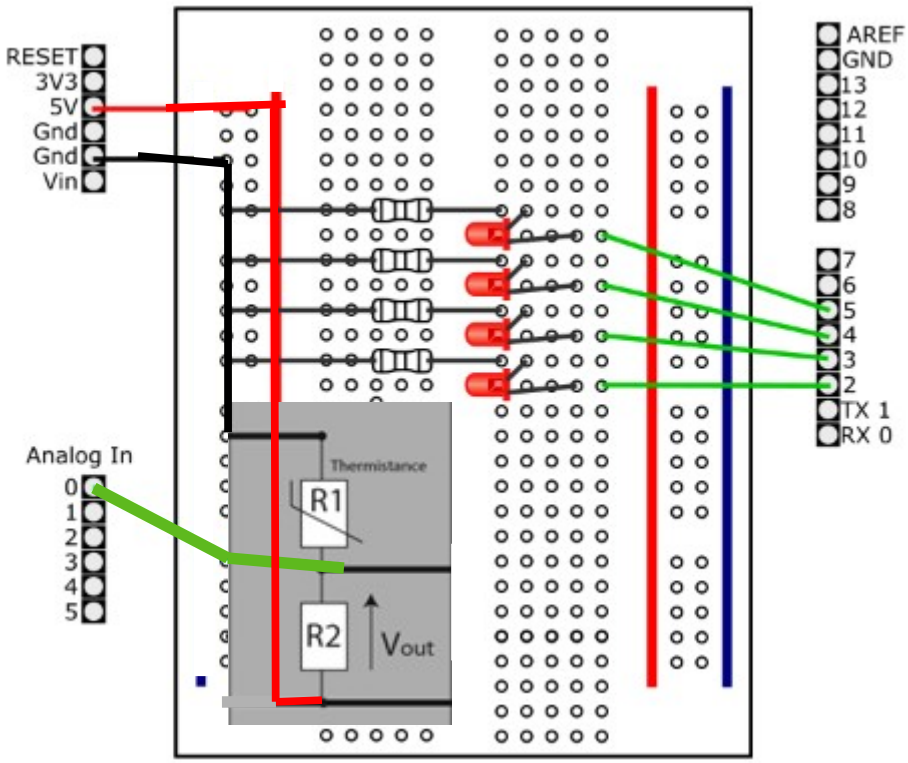
Le montage précédent est associé au microprocesseur ARDUINO. Voir schéma ci-dessous.

Compléter le programme permettant de mesurer avec la carte la température.

Quelle est la fonction de ce programme ?

Réaliser le montage.

```
const int analogPin = A0 ;  // Broche analogique A0
const float U ;
const float temp ;
const float tempamb =  10 ; //à compléter avec la mesure de la température seuil souhaitée
const int digitalPin1 = 2  ; // Broche digitale de sortie
const int digitalPin2 = 3  ; // Broche digitale de sortie                    ;
const int digitalPin3 = 4  ; // Broche digitale de sortie
const int digitalPin4 = 5  ; // Broche digitale de sortie

void setup() {
  pinMode(digitalPin1, OUTPUT); // Configure la broche digitale en sortie
  pinMode(digitalPin1, OUTPUT); // Configure la broche digitale en sortie
  pinMode(digitalPin1, OUTPUT); // Configure la broche digitale en sortie
  pinMode(digitalPin1, OUTPUT); // Configure la broche digitale en sortie

  Serial.begin (9600) ;// Initialise la communication série
  }
void loop() {
  int val = analogRead (analogPin) ; // Lit la valeur sur A0 (0-1023)
  float U = val*5/1023 ; // Convertit en tension (0-5V)
  float temp = 23*U-20  ; //à compléter avec l'étalonnage du capteur de température

// Affiche la température dans le moniteur série (optionnel)
  Serial.print ("temp = ") ;
  Serial.print (temp) ;
  Serial.print  ("°C ") ;

// Compare la valeur lue au seuil
  if (temp > tempamb + 5) {
    digitalWrite (2,HIGH) ; } // Met la sortie à 5 V
  else {
    digitalWrite (2, LOW) ; } // Met la
sortie à 0 V
  if (temp > tempamb + 10) {
    digitalWrite (3,HIGH) ; }
    else {
      digitalWrite (3, LOW) ; }
  if (temp > tempamb + 15) {
    digitalWrite (4,HIGH) ; }
  else {
    digitalWrite (4, LOW) ; }
  if (temp > tempamb + 20) {
    digitalWrite (5,HIGH) ; }
  else {
    digitalWrite (5, LOW) ;}

  delay (100) ;

}
```



Quelques explications du programme

Déclaration de variables

```
const int analogPin = A0 ;
```

signifie **déclarer une constante** nommée analogPin de type **int** (entier), et lui attribuer la valeur A0.

Explications détaillées :

- const : Ce mot-clé indique que la variable est une **constante**, c'est-à-dire que sa valeur ne peut pas être modifiée après sa déclaration.
- int : C'est le type de la variable, ici un **entier** (nombre entier).
- analogPin : C'est le nom donné à cette constante. Elle représente la broche analogique que utilisé.
- A0 : C'est la valeur attribuée à la constante. A0 est le nom de la **première broche analogique** sur une carte Arduino.

Pourquoi utiliser const ?

- Cela rend le code plus clair et évite les erreurs : il ne pourra pas accidentellement être modifiée la valeur de analogPin ailleurs dans le programme.
- Cela permet aussi au compilateur d'optimiser le code.

Exemple d'utilisation :

Dans le programme, analogPin est utilisé pour indiquer à la fonction analogRead() quelle broche analogique lire :

```
int val = analogRead(analogPin);
```

Cela revient à écrire :

```
int valeurAnalogique = analogRead(A0);
```

Mais utiliser une constante comme analogPin rend le code plus lisible et plus facile à modifier (par exemple, pour veux changer de broche plus tard, il suffit de modifier la valeur de analogPin en un seul endroit).

La boucle loop()

La fonction loop() est une boucle infinie qui s'exécute en continu après la fonction setup(). Tout ce qui est écrit dans loop() est répété indéfiniment.

La fonction delay(100);

- delay(100); signifie que le programme **attend 100 millisecondes** avant de recommencer la boucle.
- Pendant ce délai, le microcontrôleur ne fait rien d'autre : il "dort" pendant 100 ms.
- Une fois ce délai écoulé, la boucle loop() recommence depuis le début.

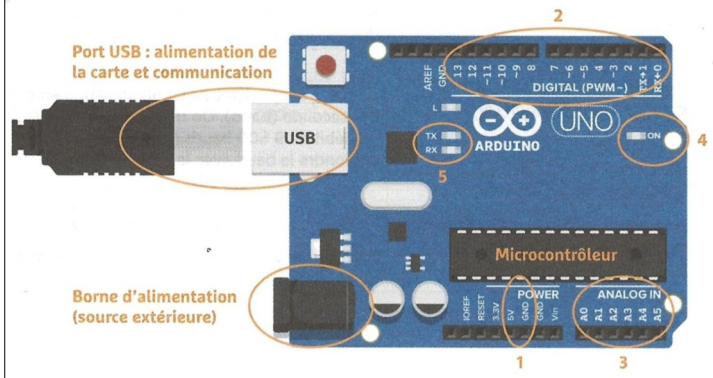
Conséquence sur la fréquence des mesures

- Le programme effectue une mesure, traite les données, puis attend 100 ms avant de refaire une mesure.
- Cela signifie qu'une nouvelle mesure est effectuée **toutes les 100 ms** (soit 10 fois par seconde).

Schéma du fonctionnement :

1. **Lire la tension** sur A0 avec analogRead(analogPin);
2. **Convertir la valeur** en tension (optionnel, pour l'affichage).
3. **Comparer la valeur** au seuil et agir sur la sortie digitale.
4. **Attendre 100 ms** avec delay(100);.
5. **Recommencer** depuis le début.

Carte Arduino®



L'une des cartes les plus utilisées est la carte Arduino® modèle Uno.

ARDUINO EN LIGNE

Le site **www.tinkercad.com** (dans la partie Circuits accessible une fois connecté) permet de simuler en ligne sans matériel un projet électronique avec un microcontrôleur de type Arduino® Uno.

- Les broches 5 V et GND (la masse) **(1)** permettent d'alimenter un circuit comme le ferait un générateur de tension continue de 5 V.

- La carte comporte **14 broches d'entrées/sorties numériques** (numérotées de 0 à 13), dites digitales **(2)**. Elles ne peuvent prendre que deux états : `HIGH` (haut, soit 5,0 V) ou `LOW` (bas, soit 0 V) par rapport à la masse (GND) qui est par définition égale à 0 V. On utilise les fonctions `pinMode()`, `digitalWrite()` et `digitalRead()` avec ces broches.

- On trouve **6 broches analogiques** (notées A0 à A5) **(3)**, dont la valeur est comprise entre 0 V et 5 V. On utilise les fonctions `analogWrite()` et `analogRead()` avec ces broches. Ces broches permettent de lire le signal d'un capteur analogique et de le convertir en une valeur numérique comprise entre 0 et 1 023 (0 correspond à 0 V et 1 023 correspond à 5 V).

- Le témoin d'alimentation **(4)** permet de vérifier que la carte est reliée à une alimentation électrique.

- Lorsque des données (visibles dans la fenêtre « **moniteur série** » de l'IDE) sont transmises entre l'ordinateur et la carte Arduino®, les LEDs RX et TX **(5)** clignotent.

REMARQUES

- Les broches analogiques peuvent aussi être utilisées en tant que broches numériques (elles seront numérotées dans ce cas de 14 à 19).
- Il ne faut pas brancher une tension supérieure à 5 V sur une broche de la carte Arduino® au risque de la détruire. Il est préférable de ne pas dépasser une consommation de 20 mA sur une broche programmée en sortie (et impératif de ne pas dépasser 40 mA).

Les fonctions

Une fonction est un bloc d'instructions que l'on peut appeler à tout endroit du sketch.

Fonction	Description	Syntaxe et paramètres
`pinMode()`	Configure la broche spécifiée pour qu'elle se comporte soit en entrée, soit en sortie.	`pinMode(broche, mode)` ***broche*** : n° de la broche de la carte Arduino®. ***mode*** : `INPUT` (entrée) ou `OUTPUT` (sortie).
`digitalWrite()`	Met un niveau logique `HIGH` (HAUT) ou `LOW` (BAS) sur une broche numérique. Si la broche a été configurée en SORTIE avec l'instruction `pinMode()`, sa tension est mise à la valeur correspondante : 5 V pour le niveau HAUT, 0 V (masse) pour le niveau BAS.	`digitalWrite(broche, valeur)` ***broche*** : n° de la broche de la carte Arduino®. ***valeur*** : `HIGH` ou `LOW`.
`digitalRead()`	Lit l'état d'une broche et renvoie la valeur `HIGH` (HAUT) ou `LOW` (BAS).	`digitalRead(broche)` ***broche*** : n° de la broche numérique que l'on veut lire.
`analogRead()`	Lit la valeur de la tension présente sur la broche spécifiée. La valeur numérique lue est comprise entre 0 et 1023, ce qui correspond à une tension comprise entre 0 et 5 V.	`analogRead(broche_analogique)` ***broche_analogique*** : n° de la broche analogique sur laquelle il faut convertir la tension analogique appliquée (comprise entre 0 et 5 V).

`tone()`	Génère une onde à la fréquence spécifiée sur une broche. La durée peut être précisée, sinon l'impulsion continue jusqu'à l'appel de l'instruction `noTone()`. La broche peut être connectée à un buzzer. Une seule note peut être produite à la fois.	`tone(broche, frequence)`
		ou
		`tone(broche, frequence, durée)`
		broche : n° de la broche sur laquelle la note est générée.
		frequence : fréquence de la note produite, en hertz (Hz).
		durée : durée de la note en milliseconde (optionnel).

`noTone()`	Stoppe la génération d'impulsions produite par l'instruction `tone()`.	`noTone(broche)` ***broche*** : n° de la broche sur laquelle il faut stopper la note.

Fonction	Description	Syntaxe et paramètres
`delay()`	Réalise une pause dans l'exécution du sketch pour la durée (en milliseconde) indiquée en paramètre.	`delay(ms)` ***ms*** **(*unsigned long*)** : nombre de millisecondes que dure la pause.
`Serial.begin()`	Fixe le débit de communication en nombre de caractères par seconde (l'unité est le baud) pour la communication série.	`Serial.begin(debit)` ***debit*** **(*int*)** : débit de communication en caractères par seconde (baud). On travaille souvent avec un débit de 9 600 bauds. Attention, il ne faut pas confondre le baud avec le bit par seconde.

`Serial.print()`
`Serial.println()`
	Affiche les données sur le port série.
	Affiche les données sur le port série, puis effectue un saut de ligne.
		`Serial.print(Val)`
		`Serial.println(Val)`
		Val : valeur à afficher pour n'importe quel type de données.