Températures : Mesurer & Exploiter

Comme la vitesse de propagation des ondes sonores dans l'air dépend de la température, nous allons à présent utiliser notre Arduino pour mesurer des températures.

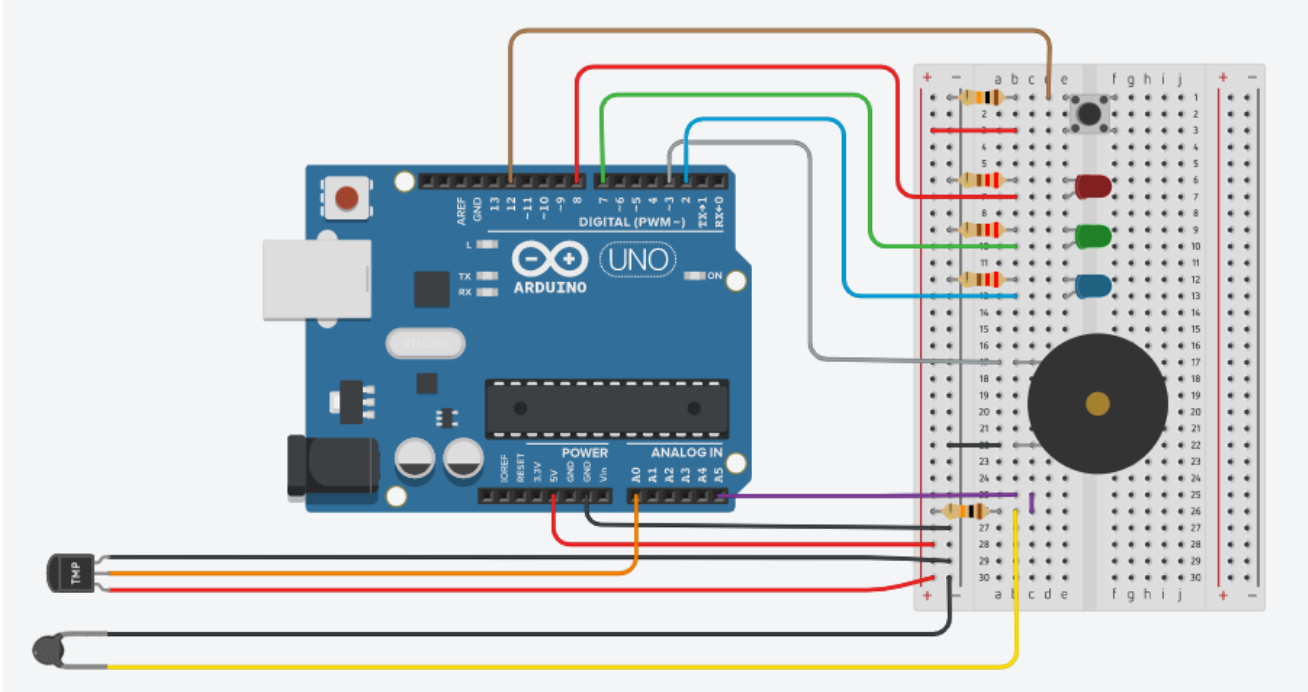
Les capteurs de température généralement utilisés avec un Arduino sont de deux types différents :

- les capteurs dont la tension de sortie est linéairement proportionnelle à la température (les plus courant : **TMP 36** et **LM 35**)

- les thermistances **CTN** (coefficient de température négatif) qui sont des capteurs de température résistifs, dont la résistance diminue de façon non linéaire quand la température augmente.

Avec un Arduino, on utilise ces capteurs pour réguler la température d'une pièce, surveiller la température de fonctionnement d'un matériel, dans les stations météo…

Toutes les activités de mesure de température et d'exploitation seront réalisées avec le circuit suivant :



. Liste des composants :

. 1 capteur de température (**TMP 36** ou **LM 35**)

. 1 thermistance

. 1 DEL rouge

. 1 DEL verte

. 1 DEL bleue

. 3 résistances de 220 Ω (résistance de protection des DELs)

. 2 résistances de 10 kΩ (résistance du bouton poussoir et de la CTN)

. 1 bouton poussoir

. 1 haut-parleur (ou piezzo)

Activité 1 : Mesure de températures avec un capteur TMP 36 ou LM 35

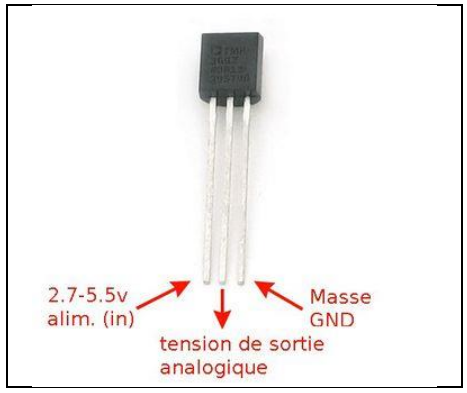
Le principe de fonctionnement de ce capteur repose sur la dépendance à la température de la caractéristique courant-tension des diodes à jonction au silicium qui sont un des composants du capteur.

Pour un courant fixe, la tension aux bornes d'une diode au silicium varie en fonction de la température.

En amplifiant le changement de tension, ces capteurs génèrent un signal analogique linéairement proportionnel à la température.

. Capteur de température TMP 36

Le capteur dispose de 3 broches :



Caractéristiques :

. tension de fonctionnement : + 2,7 V à + 5,5 V

. sortie calibrée en °C (**10 mV / °C**)

. plage de mesure : −40 ° C à + 125 ° C (fonctionne jusqu'à 150 °C)

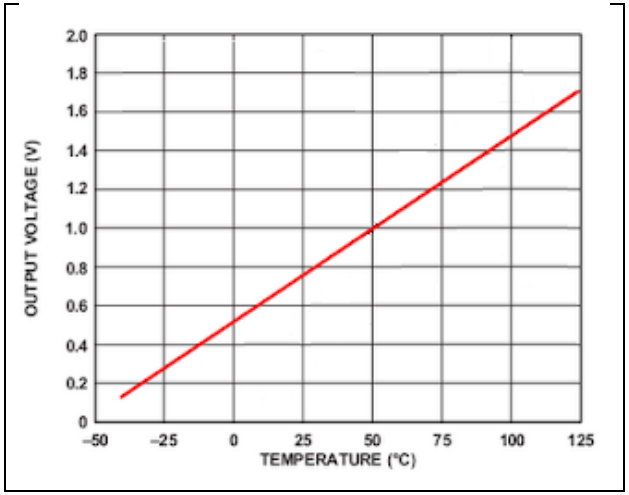
. précision : ±1 ° C à + 25 ° C (± 2 ° C pour des températures en dehors de la plage −40 ° C à + 125 ° C)

. Tension de sortie: 0.1V (-40°C) à 2.0V (150°C)

. Courant de charge: 0.05 mA

Le TMP36 permet de mesurer des températures négatives.

Le 0°C est placé à un offset de 500 mV :



Ainsi, toute mesure inférieure à 500 mv correspondra à une température négative.

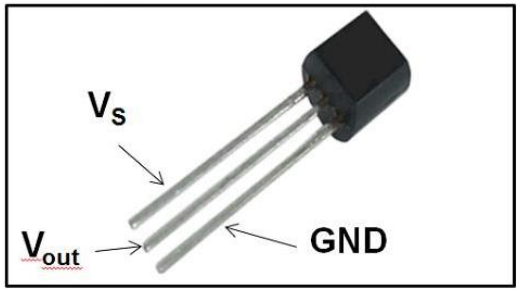
La conversion de la tension de sortie du capteur en température en °C est alors :

T (en °C) = (Tension de sortie (en mV) - 500) / 10

Soit : **T (en °C) = (Tension de sortie (en V) - 0,5) * 100**

. Capteur de température LM 35

Le capteur LM 35, fabriqué par Texas Instrument, dispose également de 3 broches :



Caractéristiques :

. Tension de fonctionnement : + 4 V à + 30 V

. Sortie calibrée en °C (**10 mV / °C**)

. Plage de mesure : −55 ° C à + 150 ° C

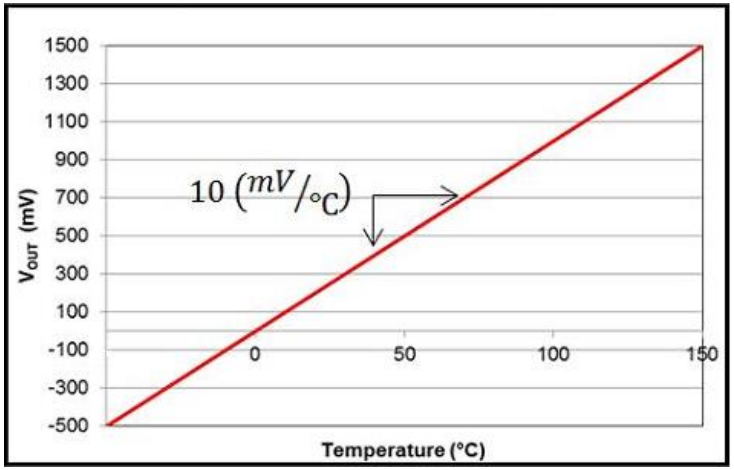
. Précision : +/-0.5°C à 25°C et +/-1°C à -55°C ou +150°C

. Tension de sortie : -0.55V (-55°C) à 1,5V (150°C)

. Courant de charge : 0.06 mA

Avec le capteur LM35, dans le cas de températures en dessous de 0°C, la tension de sortie est négative (-10°C équivaut à -0,1 V).

De fait, avec un Arduino, il n'est possible de mesurer que des températures positives car la tension appliquée sur une entrée analogique du microcontrôleur doit absolument être entre 0 et 5V.



La conversion de la tension de sortie du capteur en température en °C est alors :

T (en °C) = Tension de sortie (en mV) / 10

Soit : **T (en °C) = Tension de sortie (en V) * 100**

. Le programme

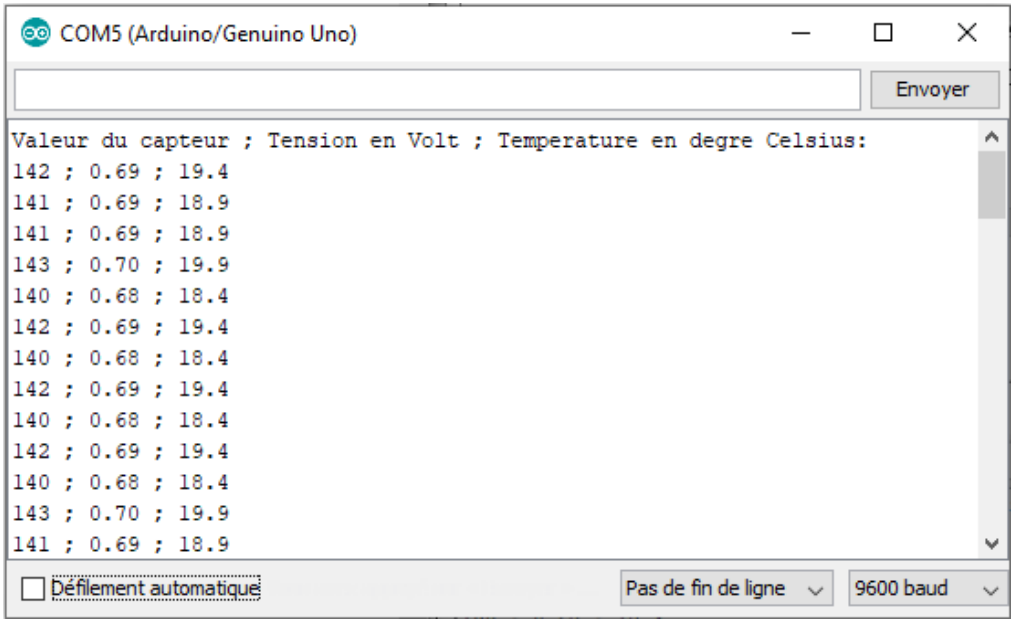
Le code de l'activité (**"Activity1.ino" dans le dossier "Codes /Températures"**) lit la valeur de la broche A0 (nombre entier entre 0 et 1023), convertit cette valeur en tension en V (nombre décimal entre 0 et 5 V) et affiche la température correspondante dans le moniteur série.

Par défaut, le code est prévu pour un capteur TMP 36, mais le calcul de la température avec un LM 35 étant en commentaire, il suffit de modifier le code pour mesurer une température avec ce capteur.

Activity1

```
// Déclaration des constantes et variables

int sensorVal = 0;
float tension = 0.0;
float temperature = 0.0;


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  Serial.print("Valeur du capteur");
  Serial.print(" ; Tension en Volt");
  Serial.println(" ; Temperature en degre Celsius:");
}


// Fonction principale en boucle

void loop() {
  sensorVal = analogRead(A0);
  tension = (sensorVal/1023.0)*5.0;

  // Capteur TMP 36
  temperature = (tension - 0.5) * 100;

  // Capteur LM 35
  //temperature = tension * 100;

  Serial.print(sensorVal);
  Serial.print(" ; ");
  Serial.print(tension,2);
  Serial.print(" ; ");
  Serial.println(temperature,1);
  delay(100);
}
```

Résultats dans le moniteur série :



Déroulement du programme :

- Déclaration des constantes et variables :

  . **int sensorVal = 0** (variable pour stocker la valeur de la broche A0)

  . **float tension = 0.0** (variable pour stocker la valeur en V de la tension correspondante à la valeur de la broche A0)

  . **float temperature = 0.0** (variable pour stocker le résultat du calcul de la température)

- Initialisation des entrées et sorties :

  . La liaison série est établie à un débit de 9600 bauds,

  . L'entête des mesures effectuées est affiché dans le moniteur série.

- Fonction principale en boucle :

. Lecture de la valeur de la broche A0 : **sensorVal = analogRead(A0)**

. Calcul de la tension en V correspondante :

tension = (sensorVal/1023.0)*5.0

. Calcul de la température en °C correspondante :

- Pour un TMP 36 : **temperature = (tension - 0.5) * 100**

- Pour un LM 35 : **temperature = tension * 100**

. Affichage des variables sensorVal, tension et temperature dans le moniteur série.

Remarque :

En plus du premier paramètre correspondant à la variable à afficher dans le moniteur série, la fonction **"Serial.print()"** admet un 2ème paramètre optionnel pour les nombres décimaux qui précise le nombre de décimales après la virgule :

. Serial.print(1.23456, 0) donne "1"

. Serial.print(1.23456, 2) donne "1.23"

. Serial.print(1.23456, 4) donne "1.2346"

Remarques :

La lecture d'une entrée analogique revoie une valeur entre 0 et 1023 pour une tension variant entre 0 et 5v.

La précision de la mesure est donc de 5 / 1024, soit 0.0048 V (soit environ 5 mV).

Donc, le signal de sortie du capteur TMP36 étant de 10 mv par degré, la mesure via l'entré analogique de l'Arduino est donc précise à +/- 0,5 °c.

Il est possible d'augmenter la précision de l'entrée analogique :

-En alimentant le TMP36 en 3,3 volts (disponible sur l'Arduino).

- Et en utilisant cette tension de 3,3v comme référence (sur la broche ARef) pour les lectures analogiques.

En effet, dans ce cas, la valeur lue sur l'entrée analogique évoluera de 0 à 1023 pour une tension entre 0 et ARef (3.3 V).

Soit une précision de 3.3 / 1024 = 0,0032 V (3,2 mV).

L'erreur est ici réduite à +/- 0.32°C.

Pour utiliser une tension de référence externe sur la broche ARef, pour la conversion analogique numérique des entrées analogiques, il faut l'initialiser dans la fonction setup() à l'aide de l'instruction :

analogReference(EXTERNAL)

Activité 2 : Alarme sonore par dépassement de température

L'objectif de cette activité est de réaliser une alarme sonore et visuelle qui se déclenchera lorsque la température mesurée par un capteur TMP 36 ou LM 35 est supérieure à une valeur seuil à définir, permettant, par exemple, de prévenir un utilisateur du dépassement de la température d'utilisation d'un matériel.

Il suffit pour cela de reprendre le programme de l'activité précédente, auquel nous allons ajouter le code de l'alarme sonore et visuelle déjà vu.

. Le programme

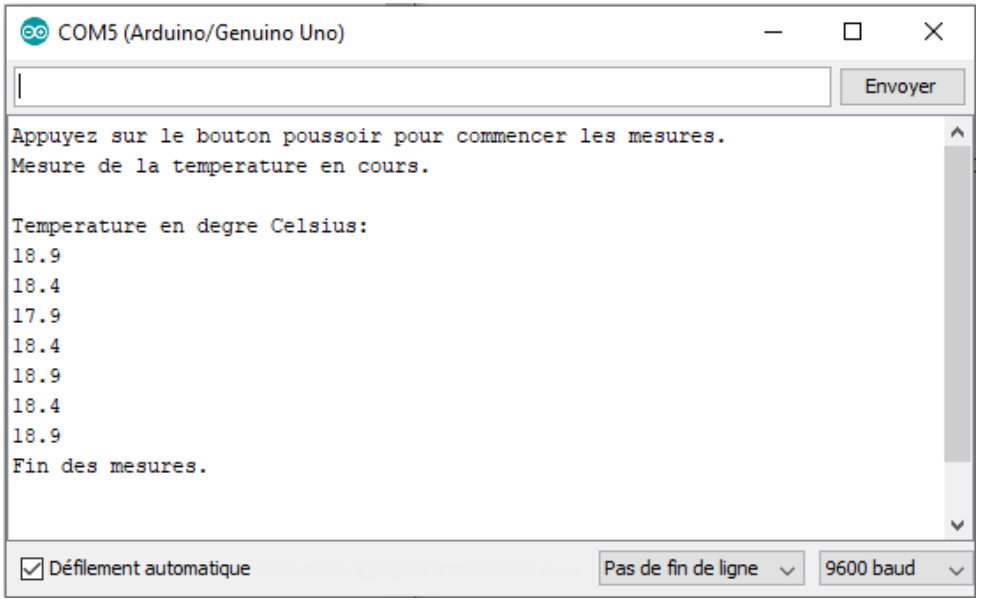
Le code (**"Activity2.ino" dans le dossier "Codes/Température"**) pourra être modifié pour voir l'influence des variables (seuil de température, fréquence de l'onde sonore, durée d'émission, durée de silence).

Activity2

```
// Déclaration des constantes et variables

const int PinSensor=0;
const int PinButton= 12;
const int PinLed = 8;
const int PinTone = 3;

int ValSensor = 0;
float tension = 0.0;
float Temp = 0.0;
float OldTemp = 0.0;
float TempAlarme = 25.0;

int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  pinMode(PinLed,OUTPUT);
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}


// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
delay(10);
```

```
if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
OldValButton = ValButton;

if (State==1)
  {
  if (OldState == 0)
  {
   Serial.println("Mesure de la temperature en cours.");
   Serial.println("");
   Serial.println ("Temperature en degre Celsius:");
   OldState=1;
  }
  ValSensor = analogRead(A0);
  tension = (ValSensor/1023.0)*5.0;

  // Capteur TMP 36
  Temp = (tension - 0.5) * 100;

  // Capteur LM 35
  //Temp = tension * 100;

  if (OldTemp != Temp)
  {
    Serial.println(Temp,1);
    OldTemp = Temp;
  }
  if (Temp > TempAlarme)
   {
      digitalWrite(PinLed, HIGH);
      tone(PinTone,440);
      delay(100);
      digitalWrite(PinLed, LOW);
      noTone(PinTone);
      delay(100);
   }
   else
   {
      digitalWrite(PinLed, LOW);
      noTone(PinTone);
   }
  delay(100);
  }
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    OldState = 0;}
  }
}
```

Résultats dans le moniteur série :



Déroulement du programme :

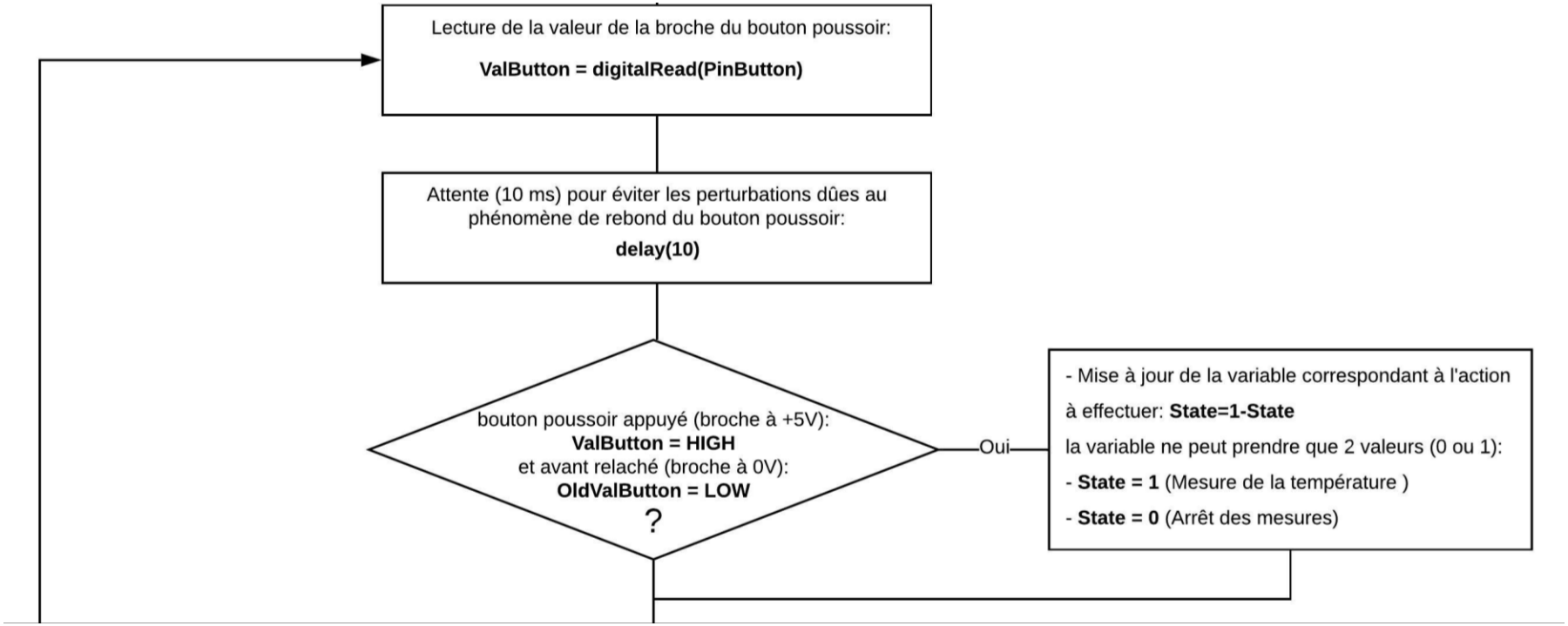
- 1. Déclaration des constantes et variables :

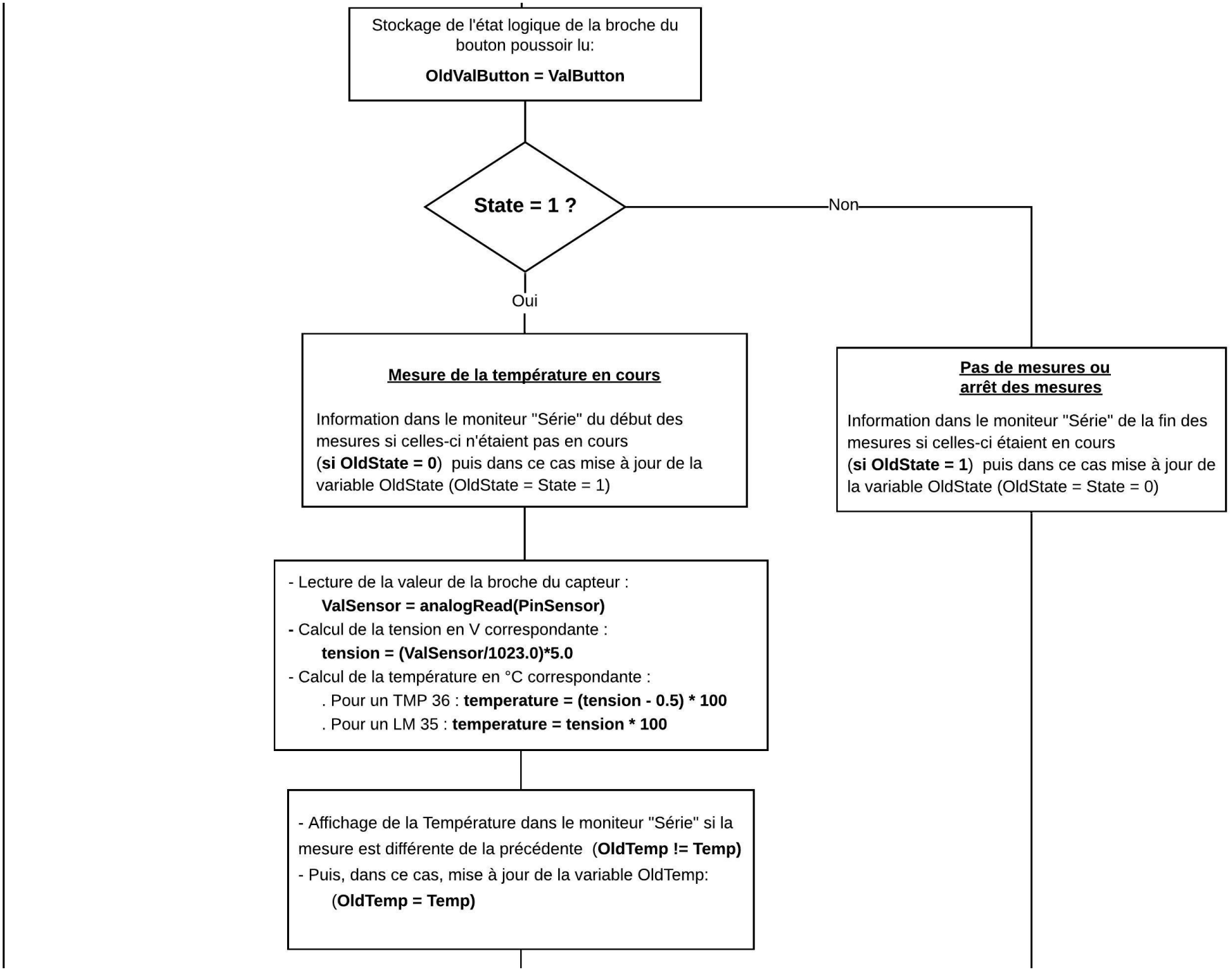
- N° de la broche correspondant au bouton poussoir: **const int PinButton = 12**
- N° de la broche correspondant à la DEL : **const int PinLed = 8**
- N° de la broche correspondant au piezo : **const int PinTone = 3**
- N° de la broche correspondant au capteur de température : **const int PinSensor = 0**
- Variable pour stocker la valeur de la broche du bouton poussoir: **int ValButton = 0**
- Variable pour stocker l'ancienne valeur de la broche du bouton poussoir: **int OldValButton = 0**
- Variable correspondant à l'action à effectuer: **int State = 0**
- Variable pour stocker l'ancienne valeur de la variable correspondant à l'action à effectuer: **int OldState = 0**
- Variable pour stocker la valeur de la broche du capteur de température: **int ValSensor = 0**
- Variable pour stocker la valeur en V de la tension correspondante à la valeur de la broche du capteur : **float tension = 0.0**
- Variable correspondant à la température calculée à partir de la valeur de la broche du capteur: **float Temp = 0.0**
- Variable correspondant à la température mesurée précédemment: **float OldTemp = 0.0**
- Variable correspondant à la température du seuil de déclenchement de l'alarme: **float TempAlarme = 25.0**

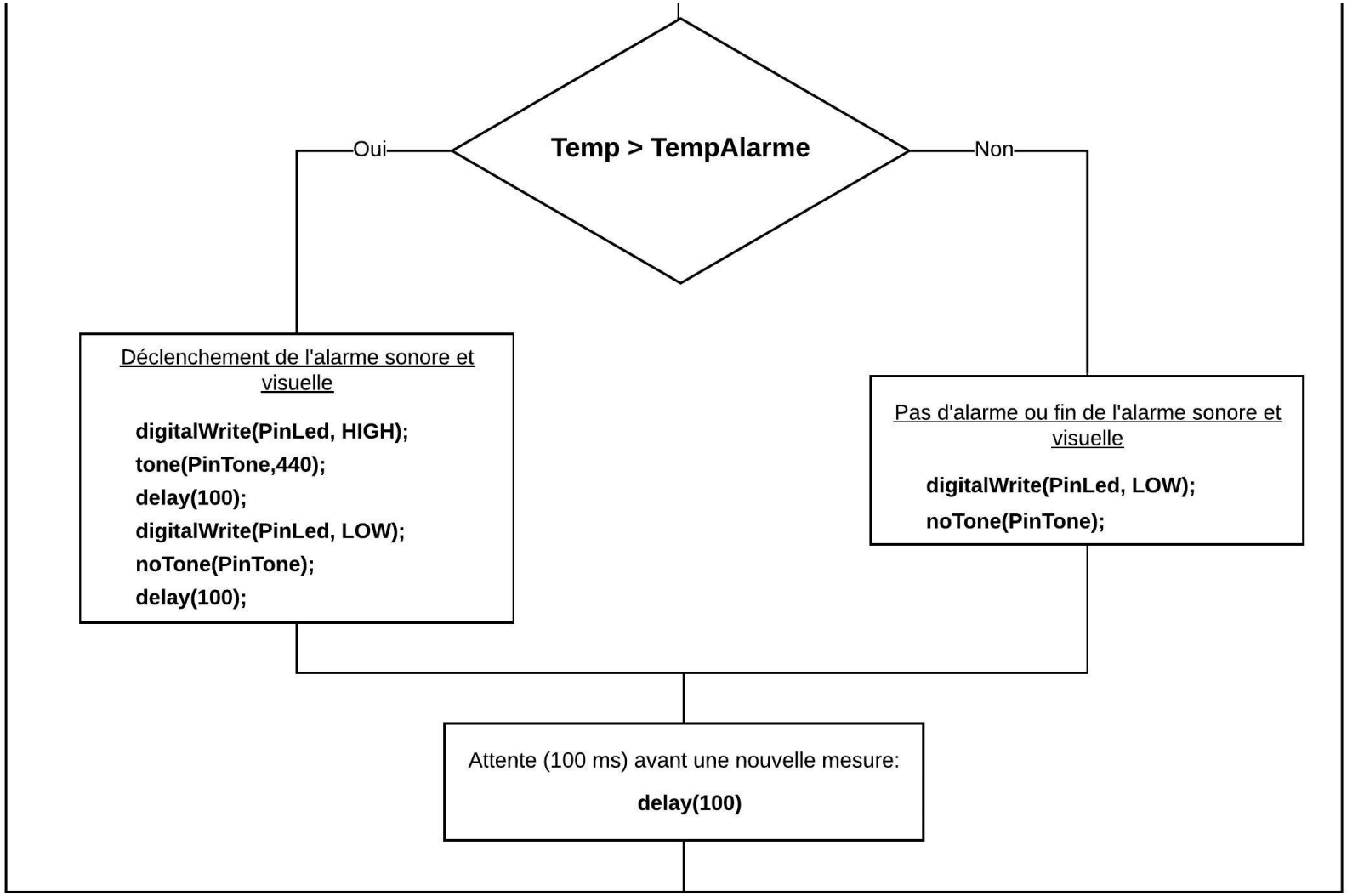
- 2. Initialisation des entrées et sorties :

- le débit de communication en nombre de caractères par seconde pour la communication série est fixé à 9600 bauds: **Serial.begin(9600)**
- La broche du bouton poussoir est initialisée comme une entrée digitale. Des données seront donc envoyées depuis cette broche vers le microcontrôleur: **pinMode (PinButton, INPUT)**
- La broche de la DEL rouge est initialisée comme une sortie digitale. Des données seront donc envoyées depuis le microcontrôleur vers cette broche : **pinMode (PinLed, OUTPUT)**

- 3. Fonction principale en boucle :




Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la température en cours
Information dans le moniteur "Série" du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information dans le moniteur "Série" de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
- Lecture de la valeur de la broche du capteur :
ValSensor = analogRead(PinSensor)
- Calcul de la tension en V correspondante :
tension = (ValSensor/1023.0)*5.0
- Calcul de la température en °C correspondante :
. Pour un TMP 36 : temperature = (tension - 0.5) * 100
. Pour un LM 35 : temperature = tension * 100
- Affichage de la Température dans le moniteur "Série" si la mesure est différente de la précédente (OldTemp != Temp)
- Puis, dans ce cas, mise à jour de la variable OldTemp:
(OldTemp = Temp)


Temp > TempAlarme
Oui
Non
Déclenchement de l'alarme sonore et visuelle
digitalWrite(PinLed, HIGH);
tone(PinTone,440);
delay(100);
digitalWrite(PinLed, LOW);
noTone(PinTone);
delay(100);
Pas d'alarme ou fin de l'alarme sonore et visuelle
digitalWrite(PinLed, LOW);
noTone(PinTone);
Attente (100 ms) avant une nouvelle mesure:
delay(100)

Activité 3 : Thermomètre à diodes électroluminescentes

Dans cette activité, nous allons utiliser les DELS rouge, verte et bleue du circuit d'étude afin de visualiser la zone dans laquelle se situe la température mesurée ($T_{mesurée}$) par un capteur TMP 36 ou LM 35 par rapport à une valeur de référence (T_{ref}) et un écart de température (ΔT) à définir :

- si $T_{mesurée} < T_{ref} - \Delta T$: La DEL bleue est allumée,
- si $T_{mesurée} > T_{ref} + \Delta T$: La DEL rouge est allumée,
- $T_{ref} - \Delta T < T_{mesurée} < T_{ref} + \Delta T$: La DEL verte est allumée.

. Le programme

Le code (**"Activity3.ino" dans le dossier "Codes/Température"**) pourra être modifié pour voir l'influence des variables (température de référence, écart de température).

Activity3.ino

```
// Déclaration des constantes et variables

const int PinSensor=0;
const int PinButton= 12;
const int PinLedR = 8;
const int PinLedV = 7;
const int PinLedB = 2;
const float TempRef = 20.0;
const float DT = 1.0;

int ValSensor = 0;
float tension = 0.0;
float Temp = 0.0;
float OldTemp = 0.0;

int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  pinMode(PinLedR,OUTPUT);
  pinMode(PinLedV,OUTPUT);
  pinMode(PinLedB,OUTPUT);
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}
```

```
// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
  delay(10);

  if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
  OldValButton = ValButton;

  if (State==1)
  {
  if (OldState == 0)
  {
   Serial.println("Mesure de la temperature en cours.");
   Serial.println("");
   Serial.println ("Temperature en degre Celsius:");
   OldState=1;
  }
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;

  // Capteur TMP 36
  Temp = (tension - 0.5) * 100;

  // Capteur LM 35
  //Temp = tension * 100;

  if (OldTemp != Temp)
  {
    Serial.println(Temp,1);
    OldTemp = Temp;
  }
  if ((Temp > TempRef - DT) && (Temp < TempRef + DT)) {
    digitalWrite(PinLedR, LOW);
    digitalWrite(PinLedV, HIGH);
    digitalWrite(PinLedB, LOW);
  }
  if (Temp > TempRef + DT)  {
    digitalWrite(PinLedR, HIGH);
    digitalWrite(PinLedV, LOW);
    digitalWrite(PinLedB, LOW);
  }

  if (Temp < TempRef - DT)  {
    digitalWrite(PinLedR, LOW);
    digitalWrite(PinLedV, LOW);
    digitalWrite(PinLedB, HIGH);
  }
  delay(100);
  }
  else
  {
   if (OldState == 1){
   Serial.println("Fin des mesures.");
   digitalWrite(PinLedB, LOW);
   digitalWrite(PinLedR, LOW);
   digitalWrite(PinLedV, LOW);
   OldState = 0;}
  }
}
```

Déroulement du programme :

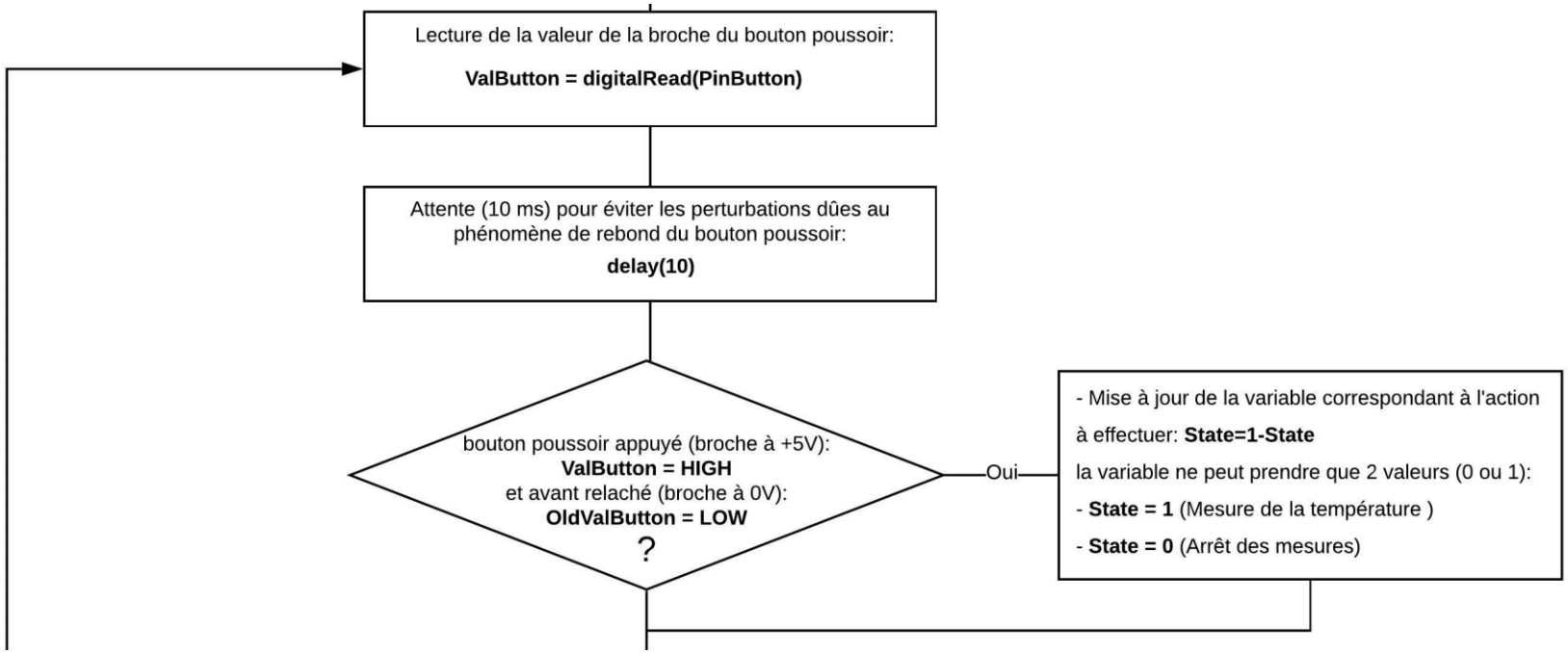
- 1. Déclaration des constantes et variables :

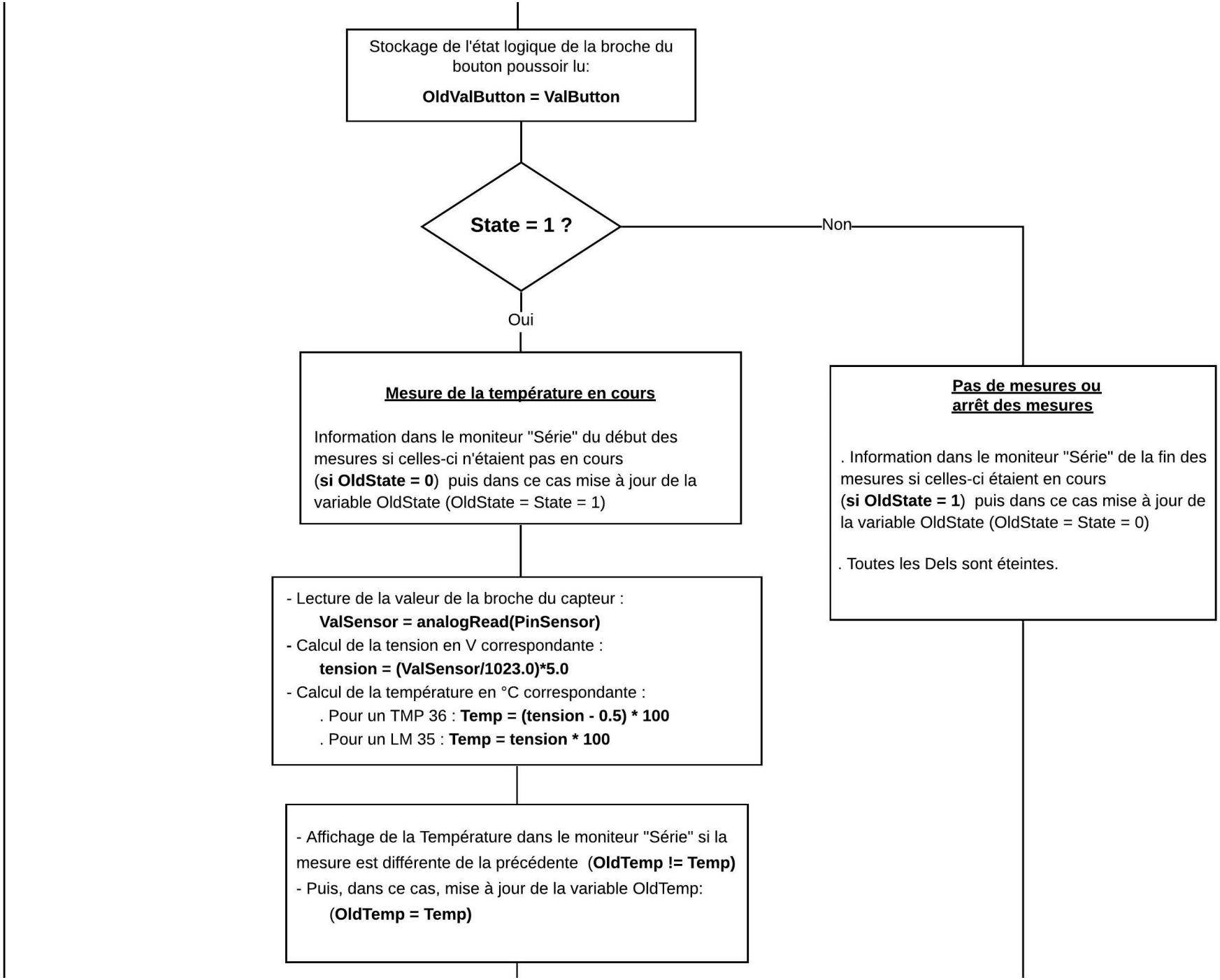
- N° de la broche correspondant au bouton poussoir: **const int PinButton = 12**
- N° de la broche correspondant à la DEL rouge : **const int PinLedR = 8**
- N° de la broche correspondant à la DEL verte : **const int PinLedV = 7**
- N° de la broche correspondant à la DEL bleue : **const int PinLedV = 2**
- N° de la broche correspondant au capteur de température : **const int PinSensor = 0**
- Variable pour stocker la valeur de la broche du bouton poussoir: **int ValButton = 0**
- Variable pour stocker l'ancienne valeur de la broche du bouton poussoir: **int OldValButton = 0**
- Variable correspondant à l'action à effectuer: **int State = 0**
- Variable pour stocker l'ancienne valeur de la variable correspondant à l'action à effectuer: **int OldState = 0**
- Variable pour stocker la valeur de la broche du capteur de température: **int ValSensor = 0**
- Variable pour stocker la valeur en V de la tension correspondante à la valeur de la broche du capteur : **float tension = 0.0**
- Variable correspondant à la température calculée à partir de la valeur de la broche du capteur: **float Temp = 0.0**
- Variable correspondant à la température mesurée précédemment: **float OldTemp = 0.0**
- Variable correspondant à la température de référence: **float TempRef = 20.0**
- Variable correspondant à l'écart de température par rapport à la température de référence: **float DT = 1.0**

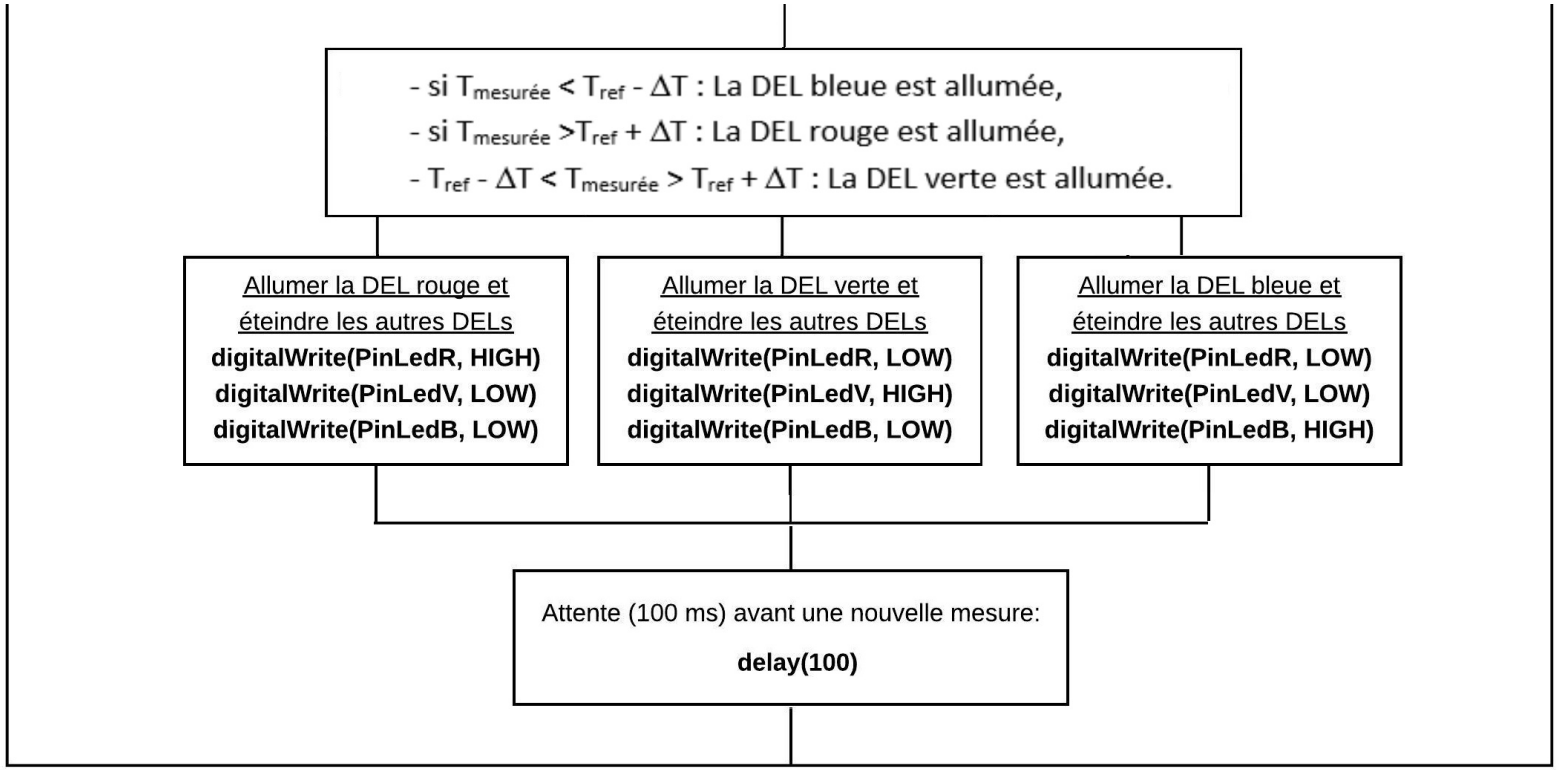
- 2. Initialisation des entrées et sorties :

- le débit de communication en nombre de caractères par seconde pour la communication série est fixé à 9600 bauds: **Serial.begin(9600)**
- La broche du bouton poussoir est initialisée comme une entrée digitale. Des données seront donc envoyées depuis cette broche vers le microcontrôleur: **pinMode (PinButton, INPUT)**
- Les broches des DELs rouge, verte et bleue sont initialisées comme des sorties digitales. Des données seront donc envoyées depuis le microcontrôleur vers ces broches.

- 3. Fonction principale en boucle :




Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la température en cours
Information dans le moniteur "Série" du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
. Information dans le moniteur "Série" de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
. Toutes les Dels sont éteintes.
- Lecture de la valeur de la broche du capteur :
ValSensor = analogRead(PinSensor)
- Calcul de la tension en V correspondante :
tension = (ValSensor/1023.0)*5.0
- Calcul de la température en °C correspondante :
. Pour un TMP 36 : Temp = (tension - 0.5) * 100
. Pour un LM 35 : Temp = tension * 100
- Affichage de la Température dans le moniteur "Série" si la mesure est différente de la précédente (OldTemp != Temp)
- Puis, dans ce cas, mise à jour de la variable OldTemp:
(OldTemp = Temp)


- si $T_{mesurée} < T_{ref} - \Delta T$: La DEL bleue est allumée,
- si $T_{mesurée} > T_{ref} + \Delta T$: La DEL rouge est allumée,
- $T_{ref} - \Delta T < T_{mesurée} > T_{ref} + \Delta T$: La DEL verte est allumée.
Allumer la DEL rouge et éteindre les autres DELs
digitalWrite(PinLedR, HIGH)
digitalWrite(PinLedV, LOW)
digitalWrite(PinLedB, LOW)
Allumer la DEL verte et éteindre les autres DELs
digitalWrite(PinLedR, LOW)
digitalWrite(PinLedV, HIGH)
digitalWrite(PinLedB, LOW)
Allumer la DEL bleue et éteindre les autres DELs
digitalWrite(PinLedR, LOW)
digitalWrite(PinLedV, LOW)
digitalWrite(PinLedB, HIGH)
Attente (100 ms) avant une nouvelle mesure:
delay(100)

Activité 4 : Etalonnage d'une thermistance (CTN)

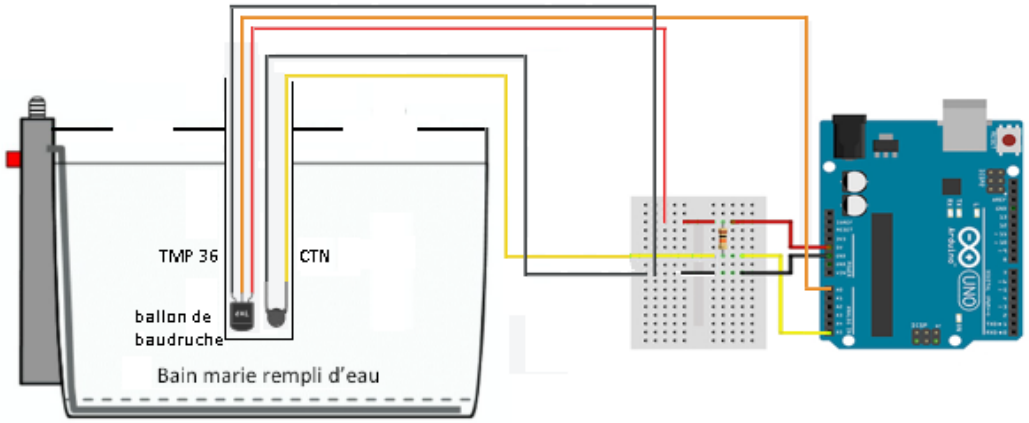
Dans notre laboratoire, nous avons trouvé une thermistance ne portant aucune indication et dont nous ne disposons pas de la fiche technique (datasheet) du fabricant.

L'objectif de cette activité est d'étalonner cette thermistance à l'aide d'un capteur de température TMP 36 ou LM 35.

Pour cela, nous allons faire varier la température et on va demander à l'Arduino de mesurer la résistance de la thermistance en fonction de la température qui sera mesurée en parallèle par le capteur.

On pourra alors tracer la caractéristique **résistance/température** qui permettra après de mesurer n'importe quelle température avec cette thermistance grâce à la mesure de sa résistance.

. Le montage

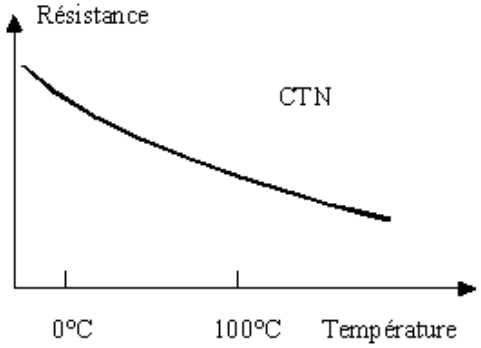


. Les thermistances

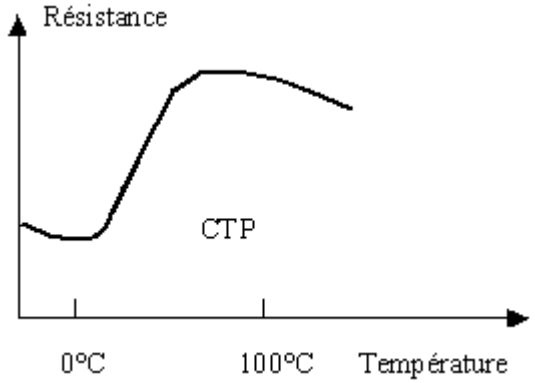
Les thermistances, composées d'un matériau semi-conducteur, combinaison de métaux et de matériaux à base d'oxyde métallique (oxyde de manganèse, cobalt, cuivre et nickel), sont des capteurs de température résistifs, dont la résistance varie de façon importante et non linéaire quand la température change même faiblement.

On distingue deux types de thermistances : les CTN et les CTP.

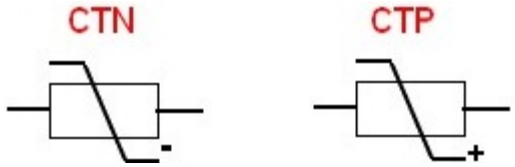
- Les CTN (Coefficient de Température Négatif, en anglais NTC, Negative Temperature Coefficient) sont des thermistances dont la résistance diminue quand la température augmente :



- Les CTP (Coefficient de Température Positif, en anglais PTC, Positive Temperature Coefficient) sont des thermistances, fabriquées à base de titanate de baryum, dont la résistance augmente fortement avec la température dans une plage de température limitée (typiquement entre 0 °C et 100 °C), mais diminue en dehors de cette zone :



Le symbole des thermistances est basé sur celui d'une résistance variable :



Même si la variation de résistance en fonction de la température d'une thermistance n'est pas linéaire, il est tout à fait possible de l'utiliser pour faire des mesures de température avec une grande précision si la loi de variation est connue, notamment avec une CTN.

En effet, Quand l'effet Joule (échauffement dû au passage du courant) est négligeable, on peut exprimer une relation entre la résistance de la CTN et sa température par ***la relation de Steinhart-Hart*** :

$$\frac{1}{T} = A + B\ln(R_T) + C(\ln(R_T))^3$$

- R_T est ***la résistance*** (en ohms) du capteur à la température T (en kelvins).
- A, B et C sont ***les coefficients de Steinhart–Hart*** (donnés par le constructeur ou obtenus expérimentalement avec trois mesures de référence) qui sont des constantes caractéristiques du composant valides à toute température.

Cette formule, valable à toutes les températures, peut être simplifiée sur une plage limitée de températures. La formule devient :

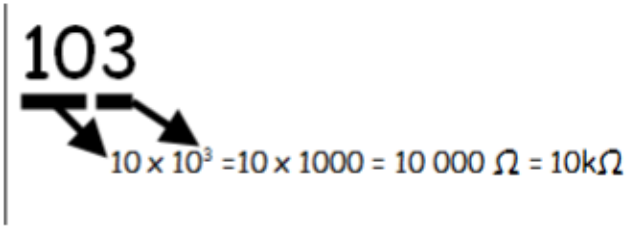
$$\frac{R_T}{R_0} = \exp\left(\beta \times \left(\frac{1}{T} - \frac{1}{T_0}\right)\right)$$

- R_T est ***la résistance*** (en ohms) du capteur à la température T (en kelvins).
- R_0 est la résistance annoncée à une température de référence T_0 (souvent 25 °C).
- β (en kelvins) est un coefficient considéré constant par approximation dont l'usage est limité à une plage de température [T1;T2] :

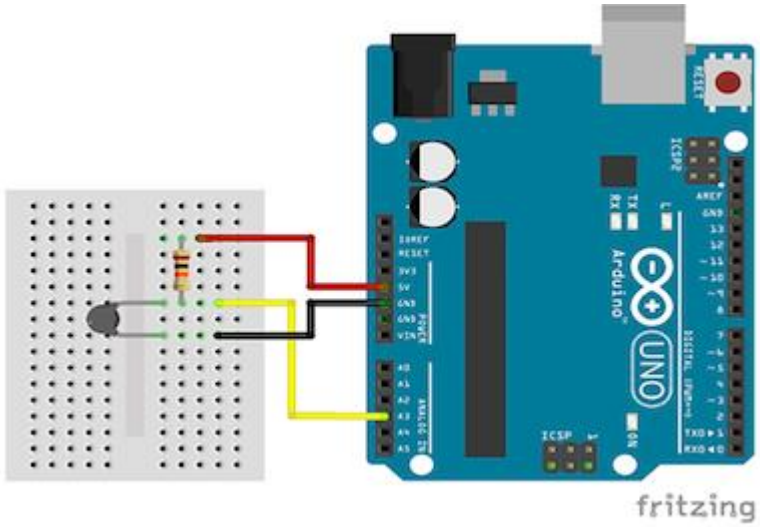
$$\beta = \frac{T_1.T_2}{T_2 - T_1} \times \ln\left(\frac{R_1}{R_2}\right)$$

La principale caractéristique d'une CTN ou d'une CTP est la valeur de la résistance R_0 (en Ω) à la température de 25°C.

Le plus souvent, le marquage du composant indique la valeur de R_0. Il s'agit d'un nombre à 3 chiffres, les 2 premiers chiffres représente la valeur, le 3ème chiffre le coefficient multiplicateur en puissance de 10 :

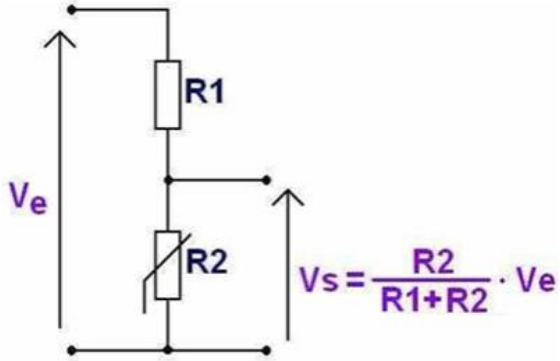


Avec un Arduino, On utilise la thermistance dans un montage (Cf. ci-dessous), avec une résistance fixe de 10 kΩ, qu'on appelle un diviseur de tension.



La thermistance est alimentée en 5V depuis l'Arduino. Le point entre les deux résistances est relié à une broche analogique de l'Arduino et on mesure la tension de cette broche par la fonction **"analogRead (broche) "**.

Diviseur de tension :



En fonction de la température (T), la valeur de la thermistance R2 varie, ce qui modifie le courant qui circule dans le circuit et donc la tension Vs aux bornes de R2.

La résistance R1 de 10kΩ, elle a pour rôle de limiter le courant circulant dans le montage.

La tension Ve est fournie par l'ARDUINO et vaut 5V.

Vs est la tension mesurée par le microcontrôleur. On peut alors calculée R2 :

$$\mathbf{R2 = \frac{Vs\ (R1+R2)}{Ve}} \quad \textbf{donc : R2 (Ve - VS) = R1.Vs}$$

$$\textbf{Soit : } \mathbf{R2 = \frac{R1.Vs}{(Ve - Vs)}}$$

Et tracer la caractéristique **R2 = f(T)**.

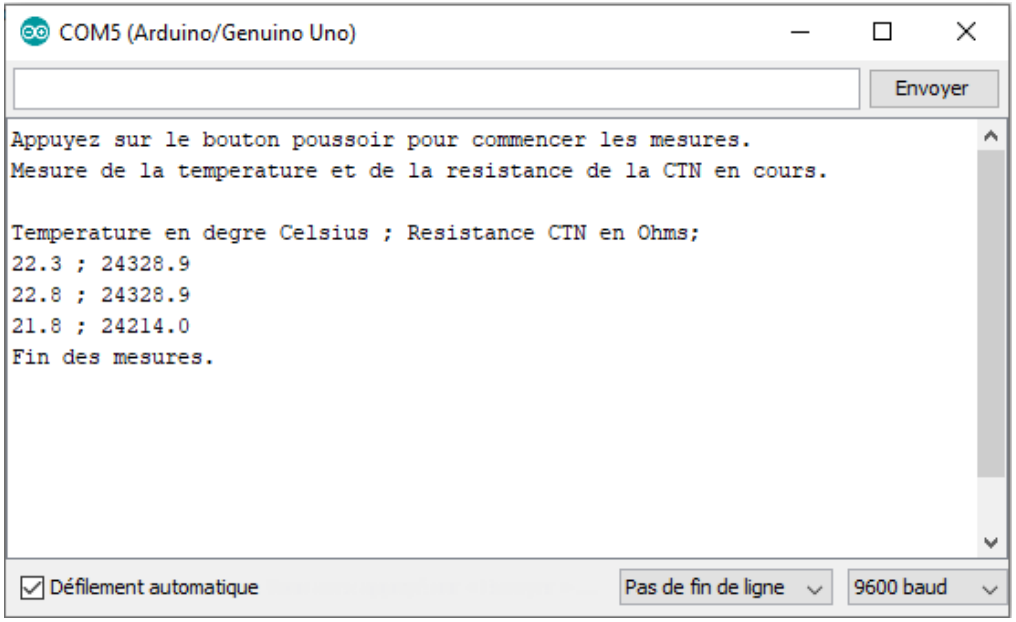
. Le programme

Voici le code de l'activité (**"Activity4.ino" dans le dossier "Codes/Température"**) :

Activity4

```
// Déclaration des constantes et variables

const int PinTMP = 0;
const int PinCTN = 5;
const int PinButton = 12;


int ValTMP = 0;
int ValCTN = 0;
float TempTMP = 0.0;
float OldTempTMP = 0.0;
float Rctn = 0.0;
float Vctn = 0.0;


int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}


// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
  delay(10);
  if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
  OldValButton = ValButton;

  if (State==1)
  {
    if (OldState == 0)
    {
    Serial.println("Mesure de la temperature et de la resistance de la CTN en cours.");
    Serial.println("");
    Serial.println ("Temperature en degre Celsius ; Resistance CTN en Ohms;");
    OldState=1;
    }
    ValTMP = analogRead(PinTMP);
    // Capteur TMP 36
    TempTMP = ((ValTMP/1023.0)*5.0 - 0.5) * 100;
    // Capteur LM 35
    //TempTMP = (ValTMP/1023.0)*5.0 * 100;

    ValCTN = analogRead(PinCTN);
    Vctn = (ValCTN/1023.0)*5.0;
    Rctn = 10000*Vctn/(5-Vctn);

    if (OldTempTMP != TempTMP)
    {
      Serial.print(TempTMP,1);
      Serial.print(" ; ");
      Serial.println(Rctn,1);
      OldTempTMP = TempTMP;
    }

  delay(100);
  }
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    OldState = 0;}
  }
}
```

Résultats dans le moniteur série :



Déroulement du programme :

- 1. Déclaration des constantes et variables :

. **const int PinTMP = 0** (broche du capteur de température TMP 36)
. const int PinCTN = 5 (broche de la CTN)
. const int PinButton = 12 (Broche du bouton poussoir)

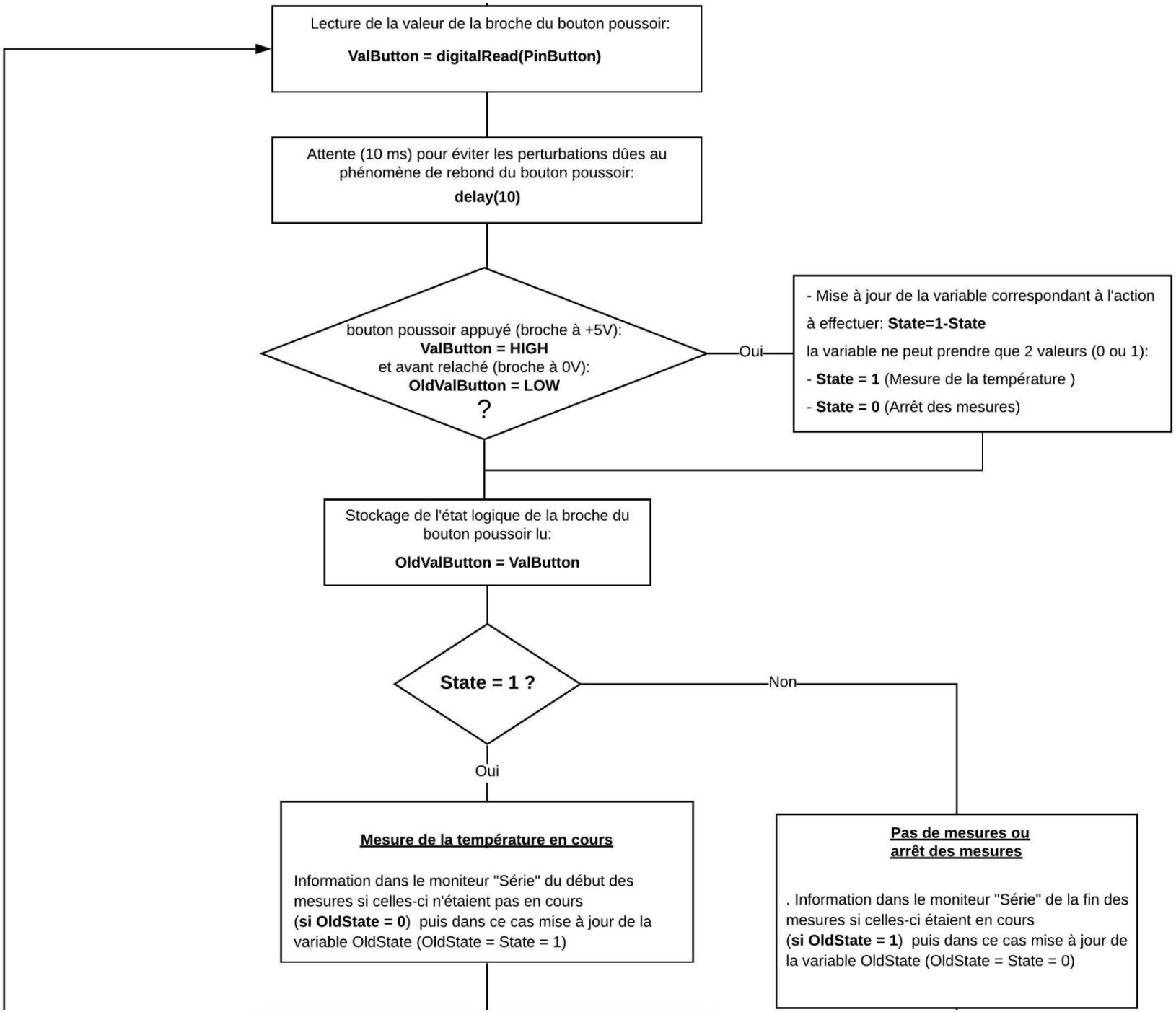
. **int ValTMP = 0** (variable nombre entier valeur broche du TMP 36)
. int ValCTN = 0 (variable nombre entier valeur broche de la CTN)
. float TempTMP = 0.0 (variable nombre décimal calcul température)
. float OldTempTMP = 0.0 (variable nombre décimal ancien calcul température)
. float Rctn = 0.0 (variable nombre décimal calcul Résistance CTN)
. float Vctn = 0.0 (variable nombre décimal calcul tension broche CTN)

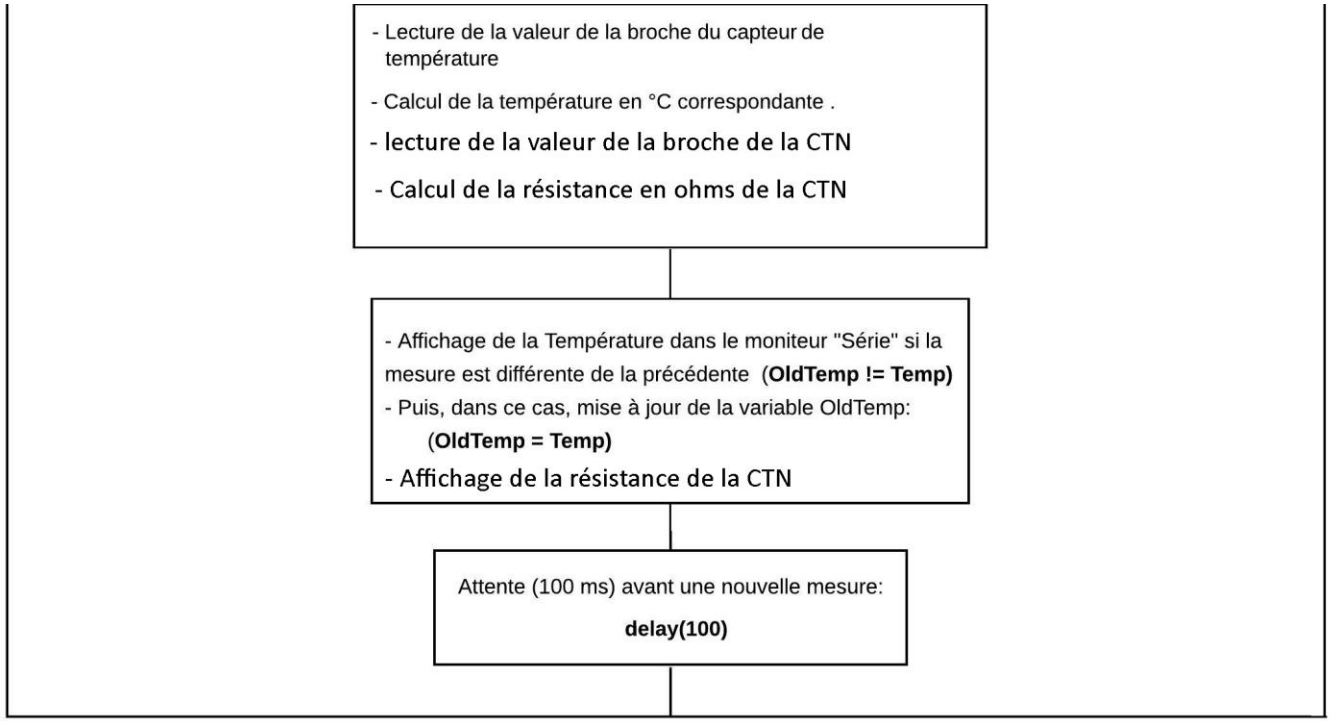
. int ValButton = 0 (variable nombre entier valeur broche bouton poussoir)
. int OldValButton = 0 (variable nbr entier ancienne valeur broche btn poussoir)
. int State = 0 (variable nombre entier action à effectuer)
. int OldState = 0 (variable nombre entier action effectuée précédemment)

- 2. Initialisation des entrées et sorties :

. Initialisation de la liaison série à un débit de 9600 bauds
. Initialisation de la broche du bouton poussoir en entrée

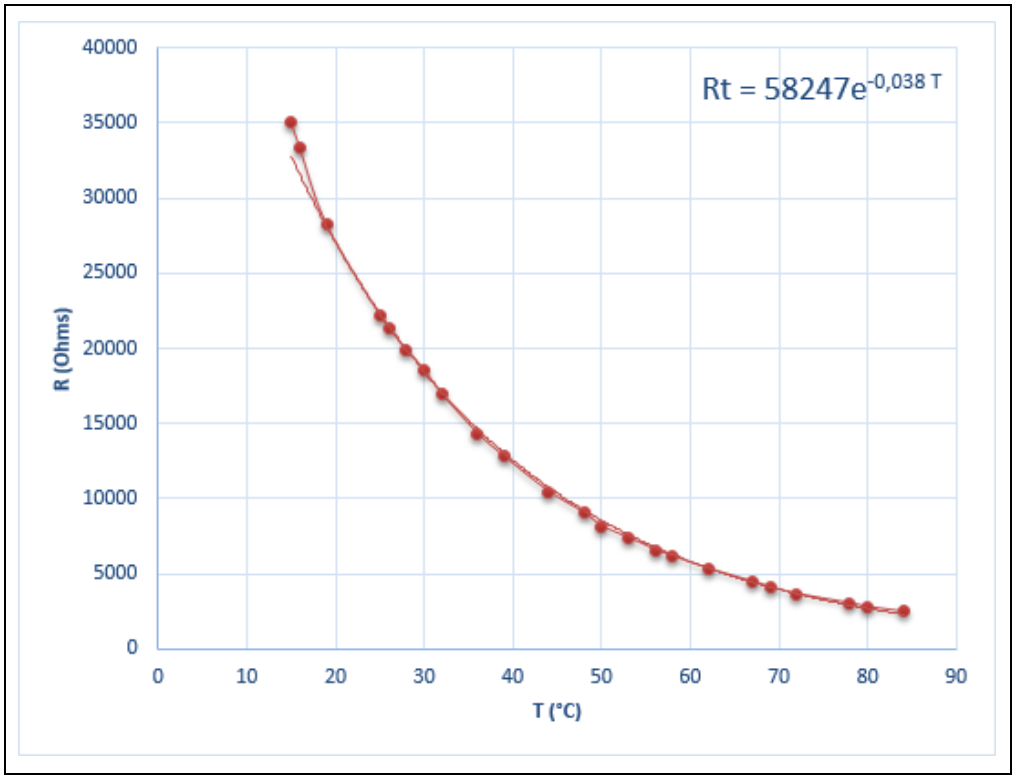
- 3. Fonction principale en boucle :


Lecture de la valeur de la broche du bouton poussoir:
ValButton = digitalRead(PinButton)
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
Oui
- Mise à jour de la variable correspondant à l'action à effectuer: State=1-State
la variable ne peut prendre que 2 valeurs (0 ou 1):
- State = 1 (Mesure de la température)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la température en cours
Information dans le moniteur "Série" du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
. Information dans le moniteur "Série" de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)


- Lecture de la valeur de la broche du capteur de température
- Calcul de la température en °C correspondante .
- lecture de la valeur de la broche de la CTN
- Calcul de la résistance en ohms de la CTN
- Affichage de la Température dans le moniteur "Série" si la mesure est différente de la précédente (OldTemp != Temp)
- Puis, dans ce cas, mise à jour de la variable OldTemp: (OldTemp = Temp)
- Affichage de la résistance de la CTN
Attente (100 ms) avant une nouvelle mesure:
delay(100)

. Exploitation des mesures

En chauffant progressivement le bain-marie, on relève la température (T) donnée par le capteur TMP 36 et la résistance (Rt) de la CTN. Puis on trace la caractéristique **Rt = f(T)** :



La modélisation donne : $\mathbf{Rt} = \mathbf{5,82.10^4}\, e^{(-\frac{T}{26,32})}$ (T en °C, R en Ω).

On en déduit alors l'expression de la température en fonction de la résistance de la CTN, dans la plage de températures de l'expérience :

$$e^{(-\frac{T}{26,32})} = \frac{Rt}{5,82.10^4}$$

$$-\frac{T}{26,32} = ln(\frac{Rt}{5,82.10^4})$$

$$T = 26,32\, ln(\frac{5,82.10^4}{Rt})$$

Ainsi, avec l'Arduino, pour n'importe quelle température dans la plage de l'expérience, par mesure de la résistance de la CTN, on pourra déterminer la température correspondante.

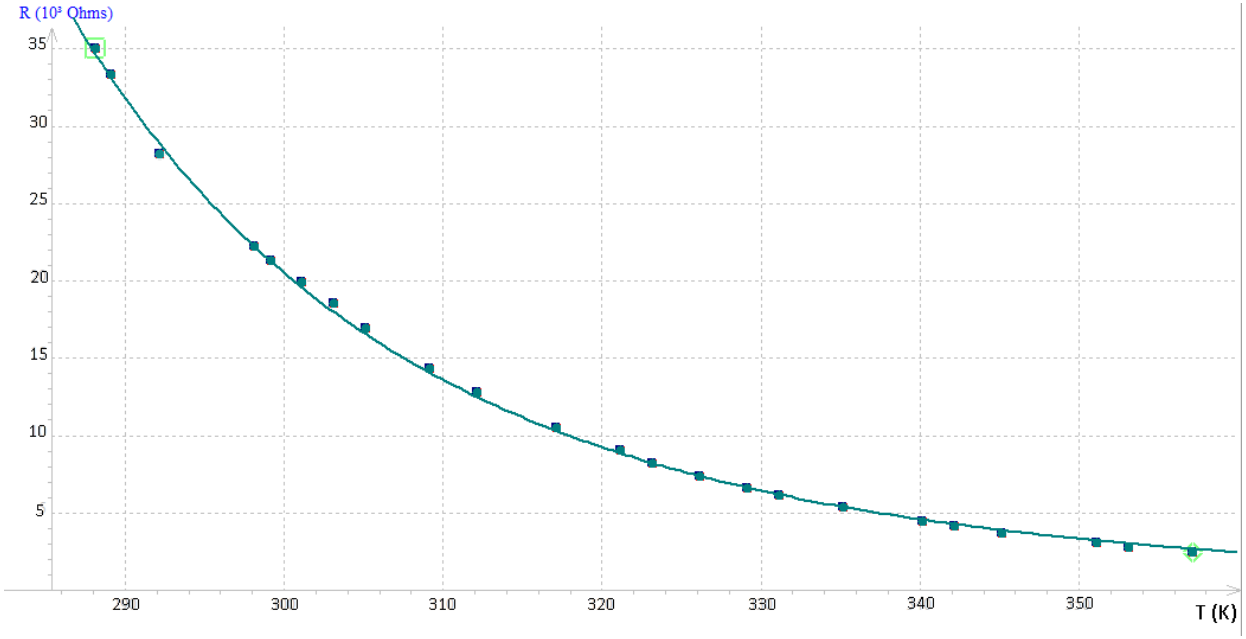
On peut également modéliser la caractéristique Rt = f(T) de notre CTN avec la relation simplifiée de ***Steinhart-Hart*** (T en K):

$$\frac{R_T}{R_0} = \exp\left(\beta \times \left(\frac{1}{T} - \frac{1}{T_0}\right)\right)$$

A partir du graphe, on détermine une des deux grandeurs caractéristiques de la CTN:

→ à T_0 = 298,15 K, R_0 = 22,2 kΩ

La modélisation suivant cette relation à l'aide du logiciel Régressi donne la valeur de β (en K) :



Expression du modèle

R(T)=22200*exp(b*((1/T)-(1/298.15)))

Résultats de la modélisation

Ecart expérience-modèle
1,7 % sur R(T)
Ecart quad. R=280,9 Ohms

b=(3,82 ±0,06) 10³

Soit : **β = 3820 K**

Dans la plage de températures de l'expérience, la relation simplifiée de ***Steinhart-Hart*** est vérifiée.

On en déduit alors l'expression de la température (en K) en fonction de la résistance de la CTN (en Ohms) :

$$R_T = R_0\, e^{\beta\left(\frac{1}{T}-\frac{1}{To}\right)} \quad \text{donc :} \quad ln(R_T) = ln(R_0\, e^{\beta\left(\frac{1}{T}-\frac{1}{To}\right)})$$

$$ln(R_T) = ln(R_0) + \beta\left(\frac{1}{T}-\frac{1}{To}\right)$$

$$\beta\left(\frac{1}{T}-\frac{1}{To}\right) = ln(R_T) - ln(R_0) = ln\left(\frac{R_T}{R_0}\right)$$

Soit :

$$\boxed{\frac{1}{T} = \frac{1}{\beta}\, ln\left(\frac{R_T}{R_0}\right) + \frac{1}{To}}$$

Activité 5 : Mesure de températures avec une thermistance CTN

Pour mesurer une température avec une thermistance CTN, il faut connaitre ses grandeurs caractéristiques. Le plus souvent, le constructeur fourni les valeurs suivantes :

- La valeur de sa résistance R_0 (résistance nominale en Ω) à la température de référence T_0 = 25 °C (298,15 K)

- La valeur de β (en K)

- La plage de température pour laquelle la relation entre la température T (en K) et R_T, la résistance (en ohms) de la CTN à cette température, est vérifiée :

$$\frac{1}{T} = \frac{1}{\beta} ln\left(\frac{R_T}{R_0}\right) + \frac{1}{To}$$

On en déduit :

$$T\ (en\ °C) = \frac{1}{\frac{1}{\beta} ln\left(\frac{R_T}{R_0}\right) + \frac{1}{To}} - 273{,}15$$

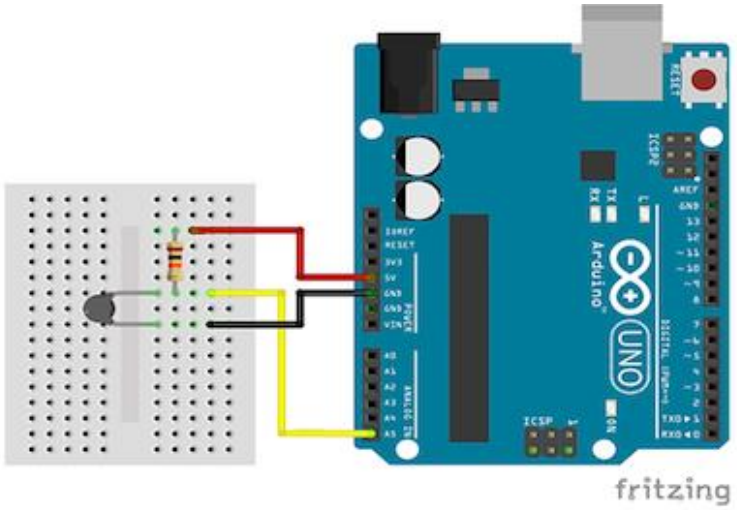
Par exemple, le capteur de température Groove est basé sur une thermistance CTN :

Le constructeur donne les informations suivantes :

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

En l'absence de ces grandeurs caractéristiques, il faut procéder à un étalonnage (cf. activité précédente), afin de les déterminer expérimentalement.

L'objectif de l'activité est d'écrire un programme généraliste permettant de mesurer une température avec une thermistance CTN quelconque, avec ce montage :



Le code demande, à l'initialisation du programme, de renseigner les valeurs de T_0, R_0 et β afin de pouvoir calculer la température à partir de la mesure de la résistance de la CTN.

. Le programme

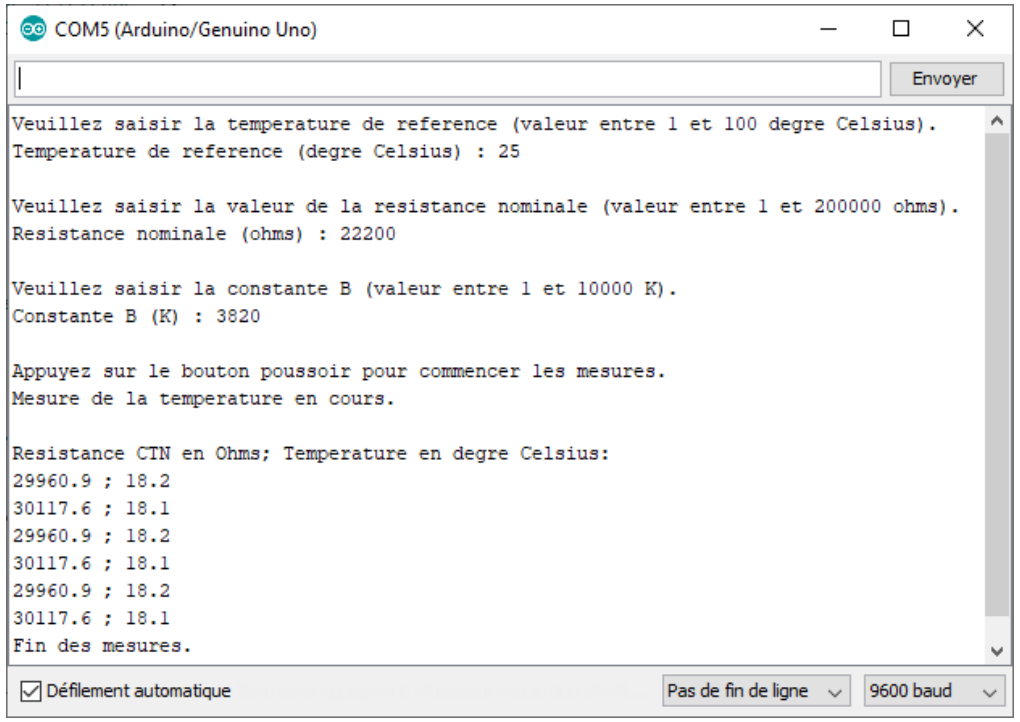
Voici le code de l'activité (**"Activity5.ino" dans le dossier "Codes/Température"**) :

Activity5

```
// Inclusion des librairies

#include <math.h>

// Déclaration des constantes et variables

const int PinCTN = 5;
const int PinButton = 12;

int ValCTN = 0;
float Temp = 0.0;
float OldTemp = 0.0;
float Rt = 0.0;
float Vctn = 0.0;
int TempRef = 0;
long Ro = 0;
int B = 0;

int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  Serial.print("Veuillez saisir la temperature de reference ");
  Serial.println("(valeur entre 1 et 100 degre Celsius).");
   while(TempRef<1 || TempRef>100)
  {
    TempRef=Serial.parseInt();
  }
  Serial.print("Temperature de reference (degre Celsius) : ");
  Serial.println(TempRef);
  Serial.println("");

  Serial.print("Veuillez saisir la valeur de la resistance nominale ");
  Serial.println("(valeur entre 1 et 200000 ohms).");
   while(Ro<1 || Ro>200000)
  {
    Ro=Serial.parseInt();
  }
  Serial.print("Resistance nominale (ohms) : "); Serial.println(Ro);
  Serial.println("");

  Serial.print("Veuillez saisir la constante B ");
  Serial.println("(valeur entre 1 et 10000 K).");
   while(B<1 || B>10000)
  {
    B=Serial.parseInt();
  }
  Serial.print("Constante B (K) : "); Serial.println(B);
  Serial.println("");

  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}


// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
  delay(10);

  if ((ValButton == HIGH)&&(OldValButton == LOW))
  {
  State=1-State;
  }
  OldValButton = ValButton;
```

```
  if (State==1)
  {
    if (OldState == 0)
    {
    Serial.println("Mesure de la temperature en cours.");
    Serial.println("");
    Serial.println ("Resistance CTN en Ohms; Temperature en degre Celsius:");
    OldState=1;
    }

    ValCTN = analogRead(PinCTN);
    Vctn = (ValCTN/1023.0)*5.0;
    Rt = 10000*Vctn/(5-Vctn);
    Temp = 1.0/(log(Rt/Ro)/B+1/(TempRef+273.15))-273.15;

    if (OldTemp != Temp)
    {
      Serial.print(Rt,1);
      Serial.print(" ; ");
      Serial.println(Temp,1);
      OldTemp = Temp;
    }

  delay(100);
  }
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    OldState = 0;}
  }
}
```

Résultats dans le moniteur série :



Déroulement du programme :

- 1. Inclusion des librairies

Pour effectuer les calculs de température, le code nécessite l'importation de la librairie "**math.h**".

- 2. Déclaration des constantes et variables :

. const int PinCTN = 5 (broche de la CTN)
. **const int PinButton = 12** (broche du bouton poussoir)

. **int ValCTN = 0** (variable nbr entier valeur de la broche de la CTN)

. **float Temp = 0.0** (variable nbr décimal valeur de la température)
. **float OldTemp = 0.0** (variable nbr décimal ancien calcul température)
. **float Rt = 0.0** (variable nbr décimal valeur résistance CTN)
. **float Vctn = 0.0** (variable nbr décimal valeur tension broche CTN)
. **int TempRef = 0** (variable nbr entier valeur température référence)
. **long Ro = 0** (variable nbr entier long valeur résistance CTN à Tref)
. **int B = 0** (variable nbr entier valeur constante β)

. int ValButton = 0 (variable nbr entier valeur broche bouton poussoir)
. int OldValButton = 0 (variable nbr entier ancienne valeur broche btn poussoir)
. int State = 0 (variable nombre entier action à effectuer)
. int OldState = 0 (variable nombre entier action effectuée précédemment)

- 3. Initialisation des entrées et sorties :

. Initialisation de la liaison série à un débit de 9600 bauds,

. Initialisation de la broche du bouton poussoir en entrée,

. Saisie des valeurs de T_0, R_0 et β afin de pouvoir calculer la température à partir de la mesure de la résistance de la CTN.

- Remarque :
La fonction **"parseInt()"** de la classe **"Serial"** retourne le premier entier long du tampon de la liaison série. Les caractères lettres ou le signe **"-"** sont ignorés. Au-delà, d'un certain temps (par défaut, 1 s), la fonction se termine et retourne "**0**" si le tampon est vide ou ne contient pas de nombre.

- 4. Fonction principale en boucle :

. Début des mesures en appuyant sur le bouton poussoir :

→ Lecture de la valeur de la broche de la CTN

→ Calcul de la résistance de la CTN

→ Calcul de la température

→ Affichage de la valeur de la résistance et de la température dans le moniteur série si la valeur de la température est différente de celle mesurée précédemment

. Fin des mesures en appuyant de nouveau sur le bouton poussoir