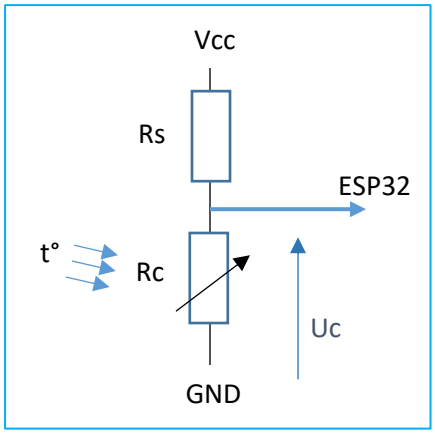
# Mesure de température CTN par un ESP32 avec Arduino

V1. oct 2024




*La température t° fait varier la résistance Rc d'un thermistor (CTN).*

*<u>Question</u> : comment déterminer t° dans l'ESP32 ?*

## A. Approche conventionnelle avec un micro-contrôleur : entrée analogique et Convertisseur Analog-Digital (ADC)

L'ESP32 dispose d'entrées analogiques avec ADC.

C'est un système à 3 étages :

1) la température fait varier la résistance Rc du thermistor,
2) la tension Uc varie selon Rc par le pont diviseur
3) l'ADC convertit Uc en une valeur numérique

A chaque étage il y a conversion et une marge d'erreur :

- Pour le thermistor, la formule de variation de sa résistance Rc avec la température est donnée par le constructeur mais elle n'est pas exactement conforme et pas identique pour chaque exemplaire.

- Uc dépend du pont diviseur Rs/Rc qui lui-même dépend de Vcc et la stabilité. Par exemple, si pour Vcc on utilise le 3V3 de l'ESP32, ce dernier est connu pour beaucoup varier.

- Enfin, le convertisseur ADC de l'ESP32. Son comportement est différent selon les générations d'ESP32 (ESP32, ESP3- S3, ESP32-C3, etc). L'Adc est paramétrable. Par défaut, sur l'ESP32, il restitue une valeur sur 12bits, de 0 à 4095, quand Uc varie de 0 à 3100mV. Un Adc est censé être linéaire, mais celui de l'ESP32 est connu pour ne pas l'être. C'est la plus grande source d'erreur dans la chaine des conversions ci-dessus. On verra qu'on peut limiter ses erreurs par quelques techniques et une calibration.

Dans l'ESP3 on connait adc. On note f(), g() et h() les fonctions de chaque étage :

1. la relation entre adc et Ut est déterminée par le convertisseur ADC du µC. `Uc= f(adc)`
2. la relation entre Uc et Rt est déterminée par le pont diviseur. `Rc= g(Uc)`
3. la relation entre t° et Rt est déterminée par la CTN. `t°= h(Rc)`

Répondre à la question initiale revient à remonter la chaine depuis la valeur Adc jusqu'à t.

connaissant la valeur adc, on aura t° = h( g (f( adc ) ) )

## 1 – Thermistor CTN et température

La résistance Rc de la CTN décroit avec la température selon une courbe donnée par l'équation théorique :

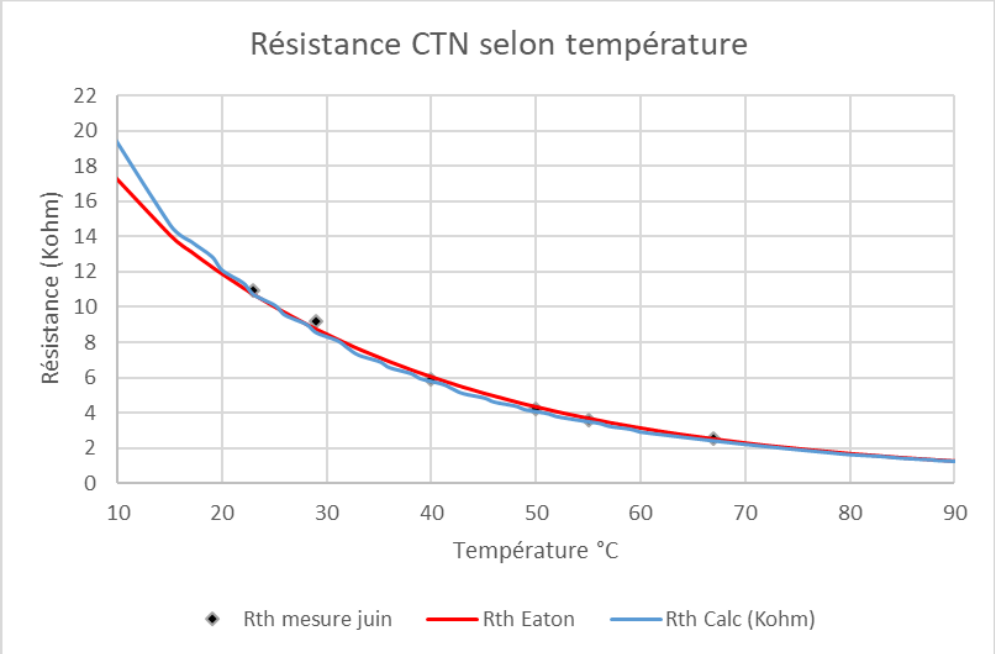
$$Rc = Ro\, e^{beta\left(\frac{1}{T} - \frac{1}{T0}\right)}$$

où $T$ la température en °Kelvin, $Ro$ la résistance à $To$ (25°C càd 298°K) et $beta$ le coefficient de la CTN

En inversant cette formule, on obtient la fonction **h()** qui donne $t$ (en °C) en fonction de $Rc$

$$t° = \frac{1}{\frac{1}{beta}\ln\left(\frac{Rc}{R0}\right) + \frac{1}{T0}} - 273$$

Par comparaison avec les données de la datasheet EATON et avec 6 points de mesure réels disponibles (Rth mesures juin), cette formule théorique s'avère très proche de la réalité.




## 2 – Pont diviseur Rs - Rc

Le pont diviseur nous donne $Uc$ selon $Rs$ et $Rc$ (l'impédance d'entrée de l'ESP32 très élevée est ignorée)

$$Uc = Vcc\,\frac{Rc}{Rc + Rs}$$

On en déduit la fonction **g()** qui nous donne $Rc$ selon $Uc$ : $Rc = \frac{Rs}{\frac{Vcc}{Uc} - 1}$

## 3 – Convertisseur ADC de l'ESP32

Avec l'atténuation par défaut à 11dB, la valeur $adc$ est un entier sur 12bits, qui varie de 0 à 4095 quand $Uc$ varie de 0 à 3100mv. La conversion est supposée linéaire.

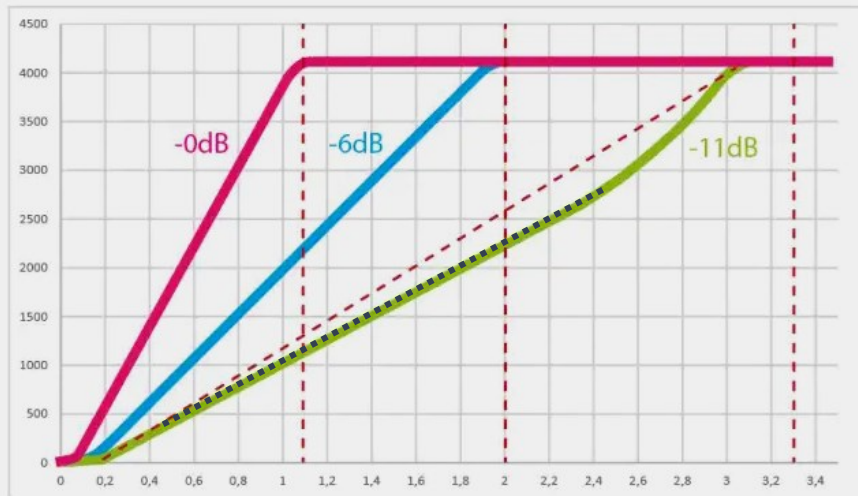
La fonction théorique de conversion est (Uc en Volts) :

$$adc = 4095 \frac{Uc}{3{,}1} \quad \text{càd} \quad Uc = 3.1 \frac{adc}{4095}$$

En réalité, selon Espressif, la conversion ADC de l'ESP32 n'est pas linéaire et dépend même de chaque exemplaire. Espressif explique que l'ADC utilise une tension de référence interne Vref qui varie pour chaque exemplaire. https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/peripherals/adc.html

https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html#adc-calibration

Des utilisateurs confirment. Par exemple : https://www.luisllamas.es/en/esp32-adc/




**Approche retenue**

On limite la plage de validité à [0.5 , 2.5V]. Sur cette plage la conversion est linéaire : c'est la ligne nore en pointillés serrés sur le graphe ci-dessus. Son équation est du type $adc = a\,.\,Uc + b$

Les coefficients a et b sont propres à chaque exemplaire d'ESP32. Pour déterminer a et b, càd calibrer l'ADC, quelques points de mesure suffisent. Par exemple avec un potentiomètre ou des résistances de valeur connue et un multimètre. Aussi, l'oversampling de la lecture de la pin par l'ADC est nécessaire pour limiter le bruit des mesures.
Exemple : J'ai déterminé pour un ESP32, a = 1277 et b = -181, pour un autre a = 1250 et b = - 125

On en déduit la fonction **f()** qui donne sur la plage [0.5V , 2.5V] $Uc$ en volts selon $adc$ :

$$Uc = A\,.\,adc + B \quad avec \;\; A = \frac{1}{a} \quad et \;\; B = -\frac{b}{a}$$

**Conclusion**

Sur l'application servant d'exemple, il est possible à mesurer la température avec une marge d'erreur inférieure à 2° C sur une plage de 19° à 90°.

# B. Approche spécifique ESP32

L'ESP32 a une fonction qui permet de mesurer directement sur ses entrées analogiques la tension $Uc$ en millivolts et ce de manière plutôt précise sans calibration.

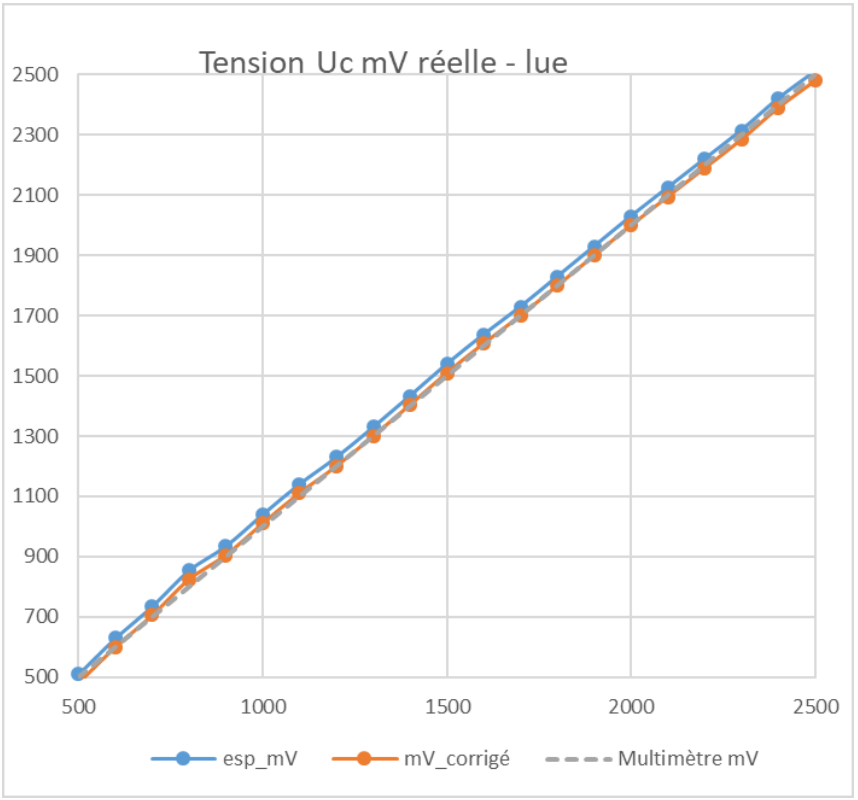
En Arduino c'est la fonction analogReadMilliVolts(pin).

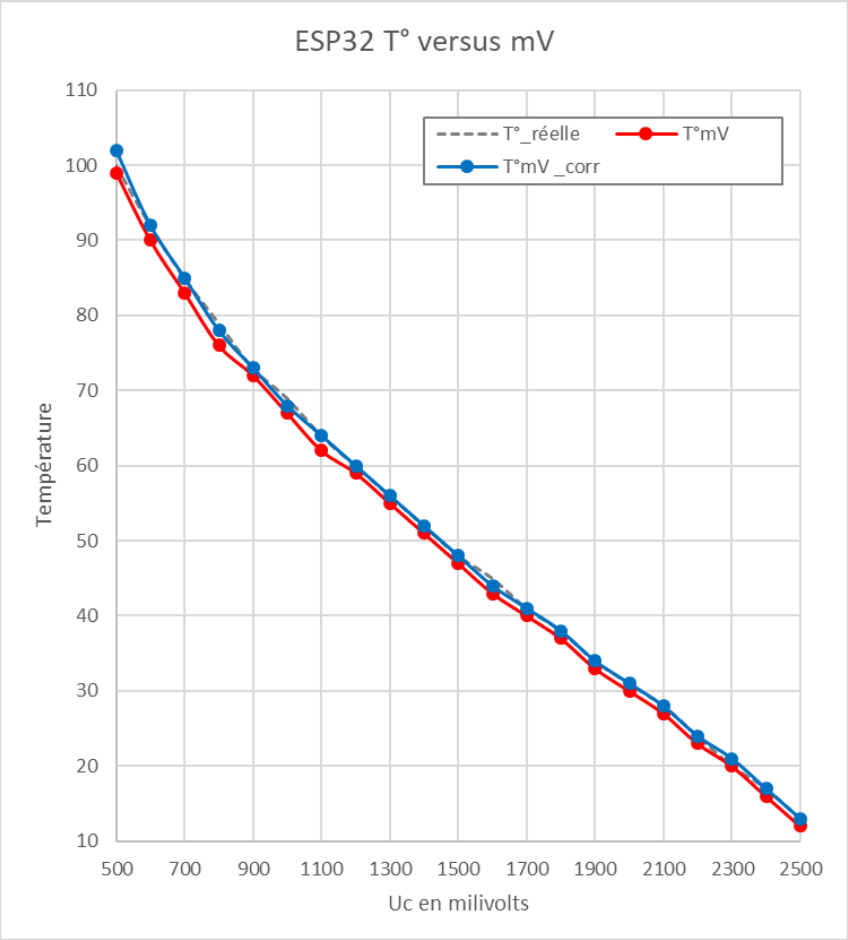
Cela fait une étape de moins et on a directement $t° = h(g(Uc))$ avec $Uc$ en Volts lu par l'ESP32.

Il s'avère que la tension lue par analogReadMilliVolts est systématiquement surestimée d'une valeur presque constante sur toute la plage (30mV pour un exemplaire, 52mV pour l'autre). C'est aussi rapporté par makerfabs (lien dans les références). On peut appliquer une correction de calibration de façon plutôt simple.

En appliquant la correction, la précision atteinte sur la température finale est excellente (la plupart du temps 0° et un maximum de 1°C) sur la plage 20 – 90°C.

Note : C'est dépendant de la stabilité de Vcc. Si Vcc fluctue, Uc change varie sans que la température ne change.






Références :

Espressif : https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/peripherals/adc.html

Espressif : https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html#adc-calibration

Espressif Adc Arduino : https://docs.espressif.com/projects/arduino-esp32/en/latest/api/adc.htm

https://www.luisllamas.es/en/esp32-adc/

https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/

https://deepbluembedded.com/esp32-adc-tutorial-read-analog-voltage-arduino/

Lecture avec analogReadMilliVolts() : https://www.makerfabs.com/blog/post/cautions-in-using-esp32-adc-makerfabs-2

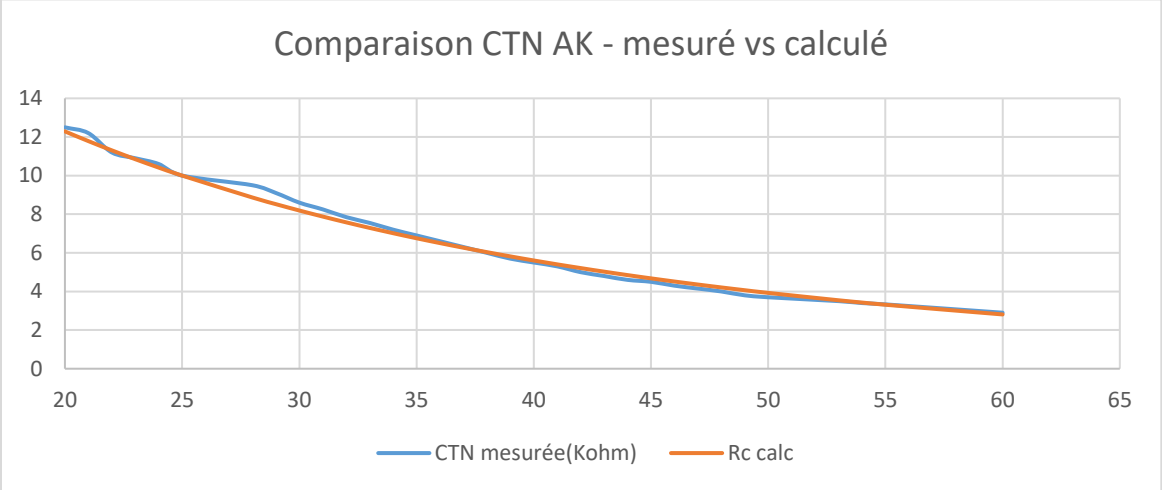
# Expérimentation ESP32-S3

Utilisation ESP32-S3 avec une CTN 10K d'origine inconnue.

**Calibrage CTN**

J'ai mesuré la résistance de ma CTN à différentes températures. Suffisamment de mesures pour valider la courbe et déterminer son beta.

La courbe calculée orange ci-dessous a été établie avec la formule pour un beta = 3600.




**Ajustements pour l'ADC de ESP32-S3**

J'ai mesuré sur mon ESP32-S3 avec un potentiomètre à la place de la CTN.

L'ESP32-S3 est supposé plus linéaire que l'ESP32 wroom, ce qui est confirmé par mes mesures. Cependant, la formule d'Espressif indiquée dans la datasheet s'éloigne de près de 5% des mesures.

J'ai adopté la formule ajustée $Uc = \frac{1}{1262} . adc$

Si on se réfère aux pages précédentes, cela revient à choisir $a=1262$ et $b=0$

Avec cette correction, la formule colle aux mesures avec une marge inférieure à 1%.