

## SoproLab va vous accompagner dans la découverte de nouvelles activités basées sur l'utilisation de microcontrôleurs.

Tout d'abord, merci de la confiance que vous nous avez accordé lors de l'achat de cette carte.

A nous maintenant de vous montrer que vous avez fait le bon choix !



### Préambule

« Quel peut être l'intérêt d'utiliser des microcontrôleurs ou d'étalonner une sonde CTN (capteur de température résistif) alors qu'on a des thermomètres numériques précis dans le labo ? »

C'est à priori un point de vue qui peut être partagé dans la mesure où cette pratique est à la frontière de trois mondes complémentaires qui nécessitent des compétences multiples :

#### Conceptualiser / Expérimenter / Mesurer

Pour ce qui est de ce premier point je ne pense pas qu'il soit nécessaire de le développer ;-)

#### Programmer

L'activité de programmation est la suite logique d'une activité commencée dès le collège en mathématiques (scratch) et en technologie (systèmes automatisés, capteurs, actionneurs). Cette pratique développe des compétences transférables dans d'autres domaines : l'abstraction, la modélisation, la planification, l'investigation ...

Elle est de plus transversale avec les mathématiques et les sciences numériques et technologie en seconde (SNT). En ce sens, c'est montrer aux élèves que les connaissances acquises dans d'autres matières (*le langage Python, l'algorithmie*), peuvent être mises à profit dans le domaine expérimental. Pour certains d'entre eux, cela leur permettra d'aborder les sciences via une autre approche. La programmation devient alors une activité concrète « Je peux allumer une LED grâce à Python ! »

Elle permet de diversifier les activités pédagogiques proposées. On peut alors associer un signal lumineux à une valeur mesurée selon un seuil fixé par

programmation (*ex : allumer la LED verte si LDR.valeur est inférieure à 2000 sinon allumer la LED rouge*).

Dans le domaine scientifique et technique, tous nos élèves ne feront pas de la recherche fondamentale, mais beaucoup d'entre eux seront sans doute un jour amenés à réfléchir sur les moyens de simplifier une tâche en utilisant ou modifiant des outils de programmation (*feuille de calculs, systèmes automatisés, objets connectés, ...*). L'évolution impressionnante des sites internet où des adolescents témoignent de leur pratique avec des cartes Arduino est une bonne illustration de ces propos. Pour certains d'entre eux d'ailleurs, leur projet n'aurait sans doute pas pu aboutir sans de bonnes connaissances en sciences physiques.

De plus, les microcontrôleurs facilitent grandement le traitement, le stockage puis le transfert de données mesurées vers un ordinateur pour leur exploitation par la suite (*feuille de calculs, compte rendus, ...*)

#### Utiliser des composants électroniques

Il y a encore quelques années, mettre en œuvre des composants électroniques nécessitait de sérieuses compétences. Le développement important du numérique et l'accès à des composants sous forme de modules pré-montés à prix réduit sur des plateformes de vente en ligne, a permis de vulgariser sa pratique. Une recherche avec comme mot clefs « Arduino Capteurs » permet de s'en convaincre !

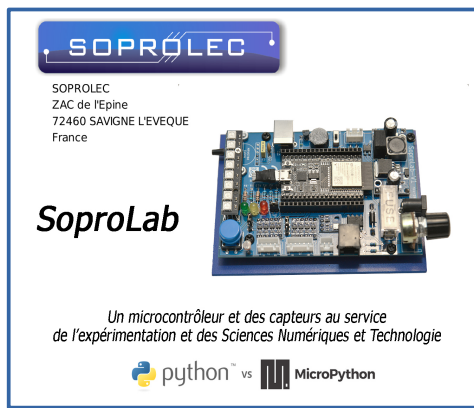
Aujourd'hui l'électronique est « à porté de main » de celles et ceux qui veulent développer leur créativité technique.

### En conclusion

L'usage des microcontrôleurs impose d'interpréter les résultats obtenus ... en un mot se poser la question de savoir ce que l'on mesure, et quelle est la signification du résultat obtenu au regard des conditions expérimentales (intervalles de validité, précision, interprétation, programmation des calculs, ...). C'est donc remettre en cause un résultat et s'interroger sur sa signification dans le contexte de l'expérience.

**Ce qui peut nous paraître évident ne l'est pas pour ceux qui le découvrent.**

**Avec les microcontrôleurs, en programmant une séquence d'instructions, l'élève acquiert de nouvelles compétences. De plus, il a le sentiment de « créer » son expérience. Cette étape me semble importante dans le processus d'appropriation d'une démarche expérimentale. Elle donne sens à l'activité.**



## Quelques étapes importantes avant de rédiger le premier programme.

Vous venez de recevoir vos cartes !

Dans 5 minutes, vous aurez réalisé votre première programmation d'un microcontrôleur en langage Python ...

*si tout se passe bien ;-)*



## Installation de la configuration

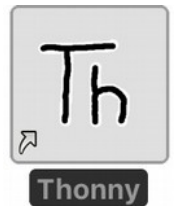
La première étape est indispensable : établir la connexion puis la communication entre la carte SoproLab et l'environnement Python.

*Les liens présentés ci-dessous sont accessibles depuis l'adresse [www.j-chouteau.org](http://www.j-chouteau.org) rubrique [ Carte SoproLab ]*

### 1) Installer un éditeur de code Python

Il va nous permettre de communiquer avec la carte. Plusieurs solutions s'offrent à vous : uPiCraft, Thonny ... chacun d'eux a des avantages et des inconvénients.

Je vous propose dans un premier temps d'utiliser celui utilisé dans les vidéos de présentation sur l'utilisation de la carte SoproLab [ [www.j-chouteau.org](http://www.j-chouteau.org) ] : THONNY



Il est en libre téléchargement sur le site : [www.thonny.org](http://www.thonny.org)

Cet éditeur contient d'ailleurs les bibliothèques utiles pour les activités de programmation en Python pour les sciences physiques (matplotlib, numpy, pyplot, ...) *exemples : tracé de vecteurs, de courbes, ...* Ce que vous faites avec eduPython ou pyzo peut être réalisé avec le logiciel Thonny.

### 2) Brancher la carte SoproLab à un port USB



### 3) Télécharger puis installer les pilotes ou drivers

Il est fort probable que votre ordinateur ne reconnaisse pas le périphérique branché et qu'il vous faille installer les pilotes ou drivers nécessaires. Selon votre version de Windows, Linux ou MacOS, ou bien de la configuration de votre ordinateur, les pilotes ou drivers qui vont permettre d'établir la connexion entre la carte SoproLab et votre ordinateur ne sont pas les mêmes.

Silicon Labs, l'entreprise qui a développé le composant qui permet la communication entre l'ordinateur et le microcontrôleur, met à disposition les pilotes nécessaires à l'adresse suivante :

<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

*Sinon, vous pouvez faire une recherche sur internet avec comme mots clefs : [ Silicon Labs CP210x USB-to-UART ]*

Si tout se passe correctement c'est plutôt bon signe, le plus dur est derrière vous ...  
Sinon, essayez avec une autre version de driver ou demandez de l'aide à l'informaticien de votre établissement. Windows est parfois un peu capricieux ...

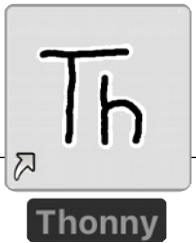
*En cas de problèmes, si cela peut vous aider, une documentation est disponible à cette adresse :*

[https://www.xilinx.com/support/documentation/boards\\_and\\_kits/install/ug1033-cp210x-usb-uart-install.pdf](https://www.xilinx.com/support/documentation/boards_and_kits/install/ug1033-cp210x-usb-uart-install.pdf)



## 4) Configurer THONNY pour l'adapter à la carte SoproLab

1 – Vous pouvez lancer le logiciel THONNY



	<p>Dans le menu :</p> <ul style="list-style-type: none"> <li>- Sélectionner l'onglet [ Exécuter ]</li> <li>puis [ Sélectionner l'interpréteur ...]</li> </ul>
	<p>[ Interpréteur Python 3 ]</p> <p>si vous souhaitez développer une application destinée à être utilisée sur l'ordinateur.</p> <p><i>Ex : tracé de vecteurs, graphiques, traitement de données, ...</i></p>
	<p>[ MicroPython ESP32 ] ou [ MicroPython générique ] pour une application destinée à être utilisée sur la carte SOPROLAB.</p> <p>[ MicroPython ESP8266 ] pour une application destinée à être utilisée avec le kit [Objets Connectés] en mode serveur.</p>
	<p>Choisir le port USB où est connecté la carte SOPROLAB ou le module Objet Connecté : Silicon Labs CP2102 USB</p> <p><i>(ici /dev/cu.SLAB... correspond à une connexion sur un ordinateur Linux ou MacOS, sur un PC vous devriez voir SLAB... sur com3 / com4 ou com... ).</i></p>
	<p>Si Thonny vous informe qu'il n'a pas réussi à établir une connexion avec la carte à microcontrôleur c'est que le driver installé ou que l'interpréteur choisit n'est pas adapté à la carte SoproLab.</p> <ol style="list-style-type: none"> <li>1 – Vérifier que le câble USB est bien branché à chaque extrémité (carte et ordinateur).</li> <li>2– Vérifier que vous avez choisi [ MicroPython (ESP32) ou (générique) ] comme interpréteur (étape N°3)</li> <li>3 - Vérifier dans le panneau de configuration de windows s'il n'y a pas un problème dans l'installation du pilote Silicon Labs sur le port USB où est branché la carte SoproLab. Auquel cas essayez d'installer un autre pilote adhoc.</li> <li>4 – Essayer de relancer la connexion en cliquant sur l'icône rouge [ Stop ] dans la barre de menu</li> </ol>
	<p><b>Tant que vous N'obtenez PAS ce message avec les trois chevrons [ &gt;&gt;&gt; _ ], c'est que la connexion n'a pas été établie.</b></p>

## 5) En avant pour le premier programme ...



Les trois chevrons [ >>> \_ ] s'affichent !

TOUT EST EN PLACE pour de nouvelles aventures ...



La fenêtre [ Console ] : C'est l'interface de communication avec l'interpréteur microPython.

On peut y entrer des instructions en ligne de commande :  
[ >>> from machine import Pin ]

C'est dans cette fenêtre aussi que s'affiche le résultat d l'instruction [ print(...) ] ou que l'utilisateur peut entrer des données via l'instruction :  
[ reponse = input( ... ) ].

Les instructions sont exécutées au fur et à mesure qu'elles sont saisies. Les instructions précédentes peuvent être rappelées en utilisant la flèche vers le haut.

## 1- Quelques commandes Python pour prendre le contrôle de la carte SoproLab

Avec l'utilisation du logiciel Thonny, nous allons découvrir la carte grâce à quelques commandes de base en Python :



Pour effacer le contenu de la console, un clic droit sur la souris fait apparaître un menu ...

De même vous pouvez faire apparaître un traceur de courbe (Grapheur) qui permet de visualiser l'évolution d'une variable dans le temps lorsque vous affichez sa valeur avec l'instruction [ print ( val ) ]

```
>>> x=1234,56789
>>> x
(1234, 56789)
>>> x=1234.56789
>>> x
1234.568
>>> |
```

### Quelques calculs ...

[ >>> x = 1234,56789 ]

Comme vous pouvez le constater 1234,56789 n'est pas un nombre ! C'est une liste composée de deux nombres puisque le séparateur décimal est le point « . » et non la virgule !

```
Console x
>>> x = 1234.5678
>>> print('x a pour valeur {:.2f}'.format(x))
x a pour valeur 1234.57
>>> |
```

### Mettre en forme l'affichage du contenu d'une variable :

[ >>> x = 1234.5678  
>>> print ('x a pour valeur {:.2f}'.format(x)) ]  
permet de n'afficher que deux décimales pour une valeur de type [ f : float ].  
*Note : le type « float » correspond aux nombres décimaux.*

```
>>> n=int(input('Combien d'itérations ? '))
Traceback (most recent call last):
  File "<stdin>", line 1
SyntaxError: invalid syntax
>>> n=int(input("Combien d'itérations ? "))
Combien d'itérations ? 20
>>> n
20
>>> |
```

Note : Avant de saisir la réponse à la question, il faut cliquer dans la fenêtre [ Console ] devant le point d'interrogation.

### Demander une valeur entière à l'utilisateur :

[ >>> n = int(input ( ' Combien d\'itérations ? ' ))  
combien d'itération ? 20 ]

Si vous utilisez des apostrophes en début et fin de chaîne de caractères, notez ici la nécessité d'ajouter [ \ ] devant une apostrophe dans le texte que vous souhaitez afficher. Un moyen d'éviter ce soucis est d'utiliser des guillemets : [ input ( ' ' Combien d'itérations ? ' ' ) ]

```
>>> from time import sleep
>>> while True:
    sleep ( 1 )

Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
KeyboardInterrupt:
>>> |
```

### Réinitialiser le microcontrôleur ou Interrompre un programme

Cliquer dans la fenêtre [ Console ] puis :

Ctrl + D → réinitialiser le microcontrôleur au cas où des problèmes de chargement de bibliothèque auraient eu lieu.


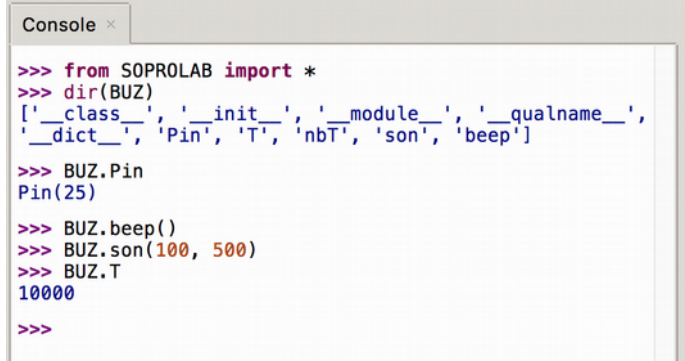
Ctrl + C → interrompre l'exécution d'un programme qui boucle ...





**Chaque ressource sur la carte (capteurs, indicateurs lumineux ou sonore) est considérée comme un « objet »** (une [ instance de classe ] dans le jargon informatique de la programmation orientée objet).

**Chaque objet a donc des propriétés et des méthodes** (ou procédures) qui permettent d'interagir avec l'objet :

<p><b>BUZ → Buzzer</b></p> <p>BUZ.beep() Produit un son de 800 Hz pendant 500ms</p> <p>BUZ.son ( F, ms ) Produit un son de fréquence [ F ] pendant une durée de [ ms ] millisecondes.</p> <p><b>Note :</b> Le buzzer est connecté à un interrupteur. Lorsque celui-ci est du côté du microcontrôleur, le buzzer est en marche.</p> <p><b>LED_v / LED_j / LED_r</b></p> <p>LED_x.on ( ) Allume la LED Verte, Jaune ou Rouge</p> <p>LED_x.off ( ) Eteint la LED Verte, Jaune ou Rouge</p> <p><b>NEOPIX ( Led multicolores )</b> - mise à jour Version 2 -</p> <p>NEOPIX.LED[i] = ( r, v, b ) Détermine les composantes en lumières Rouge, Verte et Bleue produites par la LED n° i, avec i ∈ [ 0, 7 ] (max 255). NEOPIX.LED[0]=(255, 255, 0) → Jaune</p> <p>NEOPIX.LED.write ( ) Mise à jour de l'état des LED pour en changer l'aspect.</p> <p>NEOPIX.arc_en_ciel ( ) Permet d'obtenir un dégradé de couleurs bleu / cyan / vert / jaune / rouge / blanc.</p> <p>NEOPIX.eteindre ( ) Permet d'éteindre les huit LED NeoPixel avec une seule instruction.</p> <p>NEOPIX.bargraphe ( n ) Permet d'obtenir un dégradé de couleurs du vert (n=1) au rouge (n=8). Si n=0, les huit LED sont éteintes.</p>	<p><b>MPX → Capteur de pression</b></p> <p>MPX.valeur [ 0 ; 4095 ] Valeur de la mesure de pression Ex : print ("Mesure : ", MPX.valeur)</p> <p>MPX.pression ( hPa ) Étalonnage 'automatique' grâce au BMP180, capteur de pression &amp; T°C atmosphériques, embarqué sur la carte.</p> <p><b>BP → Bouton poussoir</b></p> <p>BP.valeur ( ) [ 0 ; 1 ] Evalue l'état du bouton poussoir et retourne la valeur False [0] = BP relâché ou True [1] = BP enfoncé Ex : etat = BP.valeur ( )</p> <p>BP.drapeau [ 0 ; 1 ] Cette variable prend la valeur [True] soit [1] lorsque l'utilisateur appuie sur le bouton poussoir puis reste à [1] jusqu'à ce qu'une instruction dans le programme la remette à 0 [ BP.drapeau = False ].</p> <p>BP.impulsion [ 0 ; 1 ] Cette variable prend la valeur [True] soit [1] lorsque l'utilisateur appuie sur le bouton poussoir mais elle ne reste à [1] que pendant 200ms. La valeur est automatiquement remise à 0 ensuite.</p> <p><b>LDR → Capteur de lumière</b></p> <p>LDR.valeur [ 0 ; 4095 ] Valeur de la mesure de lumière Ex : print ("Mesure : ", LDR.valeur)</p> <p><b>CTN → capteur de température</b> (sonde à brancher sur le connecteur J2 )</p> <p>CTN.valeur [ 0 ; 4095 ] Valeur de la mesure de température Ex : print ("Mesure : ", CTN.valeur)</p>	<p><b>BMP → T°C et Pression atm</b></p> <p>BMP.pression [ hPa ] Pression atmosphérique en hPa.</p> <p>BMP.temperature [ °C ] Température ambiante en °C.</p> <p><b>POT → Potentiomètre</b></p> <p>POT.valeur [ 0 ; 4095 ] Valeur de la mesure de tension Ex : print ("Mesure : ", POT.valeur)</p> <p>POT.tension [ 0 V ; 3,3 V ] Valeur décimale de la tension calculée par une règle de proportionnalité à partir de la valeur [ POT.valeur ] Ex : print ("U = ", POT.tension, " V ")</p> <p> <b>La propriété [ .valeur ]</b> d'un objet est le résultat de la Conversion Analogique Numérique sur 12bits soit [ 0 ; 4095 ] ce nombre n'a donc pas d'unité. Ex : POT.valeur ∈ [ 0 ; 4095 ] alors que POT.tension correspond à la tension mesurée en Volt : → [ 0 ; 3,3 V ]</p> <p><b>MEMOIRE → eeprom</b> - mise à jour Version 2 -</p> <p>Un module spécifique a été développé pour lire et afficher les données mémorisées : <b>P6_Recup_data.py</b> deux lignes de code suffisent !</p> <pre>from P6_Recup_data import * Recup_data()</pre> <p>Ces lignes peuvent être entrées dans la [ Console ] pour afficher les données.</p> <p>Pour un traitement sous LibreOffice, il suffit ensuite de faire :</p> <ul style="list-style-type: none"> <li>- Sélectionner / Copier</li> <li>- Collage spécial [ séparateur : virgule ]</li> <li>- Dans les cellules : rechercher / remplacer le séparateur décimal '.' par une virgule ','</li> </ul>	<p>La bibliothèque associée à chaque module supplémentaire est téléchargeable depuis la plateforme github : <a href="https://github.com/SoproLab/Soprolab">https://github.com/SoproLab/Soprolab</a> dossier [ Modules ]</p> <p>Une fois téléversée dans la mémoire flash, puis importée : [ from bibliothèque import * ] les objets supplémentaires sont alors disponibles :</p> <p><b>HCSR</b> : capteur de distance à Ultrasons</p> <p><b>PIV</b> : commande d'orientation du servo moteur ( HCSR version 2 )</p> <p><b>DS</b> : capteur de température numérique.</p> <p><b>BUZZER / MICRO</b> : mesure de la célérité du son dans l'air.</p>
<p><b>LCD → Ecran LCD</b></p> <p>Dans la version 2, le module LCD est considéré comme intégré à la carte.</p> <p></p> <p>Lorsqu'on appelle la fonction [ BUZ.son(100,500) ] pour produire un son de 100 Hz, pendant 500 ms, la valeur de BUZ.T (période) est automatiquement mise à jour, soit 10 000µs.</p>	<p>La bibliothèque de gestion du module LCD a donc été intégrée dans la bibliothèque principale SOPROLAB_V2.py</p>	<p>LCD.effacer ( )</p> <p>LCD.ecran_on ( ) / _off ( )</p> <p>LCD.backlight_on ( ) / _off ( )</p> <p>LCD.place curseur ( x, y )</p> <p>Exemple : l'objet [ BUZ ] correspond au BUZZER.</p> <p>[ &gt;&gt;&gt; dir(BUZ) ] permet d'afficher les propriétés et les méthodes d'un objet. Ici on retiendra la propriété T (période en µs) et les méthodes son() et beep().</p> <p>[ &gt;&gt;&gt;&gt; BUZ.Pin ] affiche le numéro de broche où est connecté le BUZZER.</p> <p>[ &gt;&gt;&gt;&gt; BUZ.beep() ] est une fonction qui produit un son de 800 Hz pendant 500 ms → « beep ».</p> <p>[ &gt;&gt;&gt;&gt; BUZ.son( F, t_ms ) ] produit un son de fréquence [ F ] pendant une durée de [ t_ms ] en ms.</p> <p>[ &gt;&gt;&gt;&gt; BUZ.T ] est la période du son en µs.</p>	<p>LCD.caractere ( char )</p> <p>LCD.texte ( texte )</p> <p>LCD.afficher ( x, y, texte ) 8 caractères spéciaux peuvent être définis</p>

Vous pouvez parcourir les différents programmes de test contenus en mémoire ( ex : Test\_SOPROLAB\_V2.py ) afin d'en extraire des instructions en Python qui vous permettront de rédiger de nouvelles applications pour vos activités pédagogiques.

### 3- Ecrire un premier programme

Après cette introduction peut être un peu longue, mais cependant nécessaire pour prendre un bon départ, vous avez tous les éléments pour écrire votre premier programme ...

C'est parti ;-)



```
<untitled> *
1  """ Test_00 : Mon premier programme
2  Auteur : Jacques Chouteau - Janv 2020 -
3  Objectif : démonstration de l'utilisation de la carte Soprolab
4  avec le logiciel thonny.org """
5  from SOPROLAB import *
6
7  LED_v.on()
8
9  sleep ( 3 )
10
11 LED_v.off()
```

Console  
>>> |

**NOTE : Dans les exemples de code qui suivent, [ from SOPROLAB ... ] doit être remplacé par [ from SOPROLAB\_V2 ] - merci -**

Dans la fenêtre au dessus de la [ Console ], vous pouvez taper votre séquence d'instructions en microPython.

Ici le programme permet d'allumer la LED verte pendant 3 secondes.

Au moment d'exécuter votre programme, Thonny vous demandera alors dans quel espace de stockage vous souhaitez enregistrer votre programme avant de l'exécuter.

```
<untitled> * x
1  # test00 : Mon premier programme
2  # Auteur : Jacques Chouteau
3  # Objectif : démonstration de l'utilisation de la carte Soprolab
4
5  from SOPROLAB import *
6
7  LED_v.on()
8
9  sleep ( 3 )
10
11 LED_v.off()
```

Where to save to?  
( Votre ordinateur )  
appareil MicroPython

Deux choix s'offrent à vous :

- soit vous souhaitez enregistrer votre programme sur votre ordinateur. Une fois la carte débranchée, aucune trace de votre programme ne sera stockée dans la carte Soprolab.
- soit vous souhaitez stocker votre programme dans la carte [ appareil Micropython ] et il restera disponible par la suite, même si la carte a été débranchée.

### 4- « Trop facile ! »

Bon, soyons honnête... allumer une LED et l'éteindre ne va pas nous mener bien loin ... Je vous propose plutôt une situation plus concrète où l'utilisation du microcontrôleur prend tout son sens :



#### 1 – Initialisation du programme

Emettre un « beep » au démarrage du programme pour vérifier le bon fonctionnement du buzzer puis afficher l'information : [ Mesures en cours // Pour terminer appuyer sur le bouton poussoir ...].

#### 2- Attendre 2 secondes entre chaque mesure et donner une indication de la température d'un liquide par un signal lumineux, voire sonore, selon les indications suivantes :

Capteur de température : valeur obtenue [0 ; 4095 ]	[ valeur < 1500 ]	] 1500 – 2000 ]	] 2000 – 2500 ]	] 2500 < valeur ]
LED Verte	On	On	On	On
LED Jaune	Off	On	On	On
LED Rouge	Off	Off	On	On
BUZZER	Off	Off	Off	1200 Hz pdt 1s

( Valeurs approximatives obtenues avec la CTN : 1000 ↔ 25°C // 2380 ↔ 50°C // 3000 ↔ 70°C )



#### 3 – Pour aller plus loin : Donner une indication plus précise de la température en utilisant les LED NeoPixel avec un dégradé de couleur du bleu ( froid ) vers le rouge ( très chaud ).



## Solution proposée :

```
Indicateur_temperature.py
1  """
2  Programme : Indicateur de température lumineux et sonore
3  Auteur Jacques Chouteau - Avril 2020
4  Objectif : Donner une indication de la température d'un liquide
5             avec un signal lumineux puis sonore si celle-ci dépasse un seuil fixé.
6  """
7  from SOPROLAB_V2 import * # Charger toutes les bibliothèques de la carte en mémoire
8
9  print("==== Mesures en cours ====")
10 print("Pour terminer, appuyez sur le bouton poussoir bleu ...")
11
12 BP.drapeau = False # Mettre à zéro l'indicateur d'appui
13
14 while BP.drapeau == False : # Tant que le bouton poussoir n'a pas été enfoncé ...
15     if CTN.valeur <= 1500 :
16         LED_v.on()
17         LED_j.off()
18         LED_r.off()
19     elif 1500 < CTN.valeur <= 2000 :
20         LED_v.on()
21         LED_j.on()
22         LED_r.off()
23     elif 2000 < CTN.valeur <= 2500 :
24         LED_v.on()
25         LED_j.on()
26         LED_r.on()
27     else :
28         BUZ.son ( 1200, 1000 )
29         sleep ( 200 )
30 print("\n==== Fin du programme ====")
```

## Alternative :

```
Indicateur_temperature.py
1  """
2  Programme : Indicateur de température lumineux et sonore
3  Auteur Jacques Chouteau - Avril 2020
4  Objectif : Donner une indication de la température d'un liquide
5             avec un signal lumineux puis sonore si celle-ci dépasse un seuil fixé.
6  Ici, on utilise le ruban de LED NeoPixel comme indicateur lumineux.
7  La fonction NEOPIX.bargraphe() accepte un argument compris entre 0 et 8.
8  On fait évoluer le bargraphe de 1 à 8 pour des valeurs de la CTN
9  comprises entre [ 500 et 2500 ]. ( CTN.valeur ~ 880 pour 6°C )
10 Autre variante : On utilise la fonction BP.impulsion
11
12 """
13 from SOPROLAB_V2 import * # Charger les bibliothèques de la carte en mémoire
14
15 print("==== Mesures en cours ====")
16 print("Pour terminer, appuyez sur le bouton poussoir bleu ...")
17
18 while not BP.impulsion : # Tant que le bouton poussoir n'a pas été enfoncé ...
19
20     val_CTN = ( CTN.valeur if CTN.valeur <= 3000 else 3000 ) - 500
21
22     NEOPIX.bargraphe ( 1 + int(7*val_CTN/2500) ) # Min : 1 / Max : 8
23
24     if val_CTN == 2500 :
25         BUZ.son ( 1200, 1000 )
26         sleep_ms ( 200 )
27
28 NEOPIX.eteindre( )
```

## Précaution avec la fonction [ print ( ) ] => prévoir une temporisation !

Lors de l'utilisation de la fonction [ print ( ... ) ], je vous conseille d'effectuer une temporisation de quelques dizaines de millisecondes [ sleep\_ms(20) ] afin de s'assurer que toutes les informations ont bien été transmises via le câble USB avant de reprendre la suite des instructions.

Sans cette précaution, vous pourriez être confronté à des « plantages » du microcontrôleur alors qu'il n'y a aucune erreur de codage en Python !

Pour des explications sur le fonctionnement des programmes lancés depuis le menu déroulant affiché sur l'écran LCD, rendez-vous sur le site [github](https://github.com) ou sur [j-chouteau.org](https://j-chouteau.org) ... ( Vidéos de démonstration , présentation du mode de fonctionnement en mini serveur WEB (Objet Connecté) , exemples de programmes , ... ).