

## 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

## 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.

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

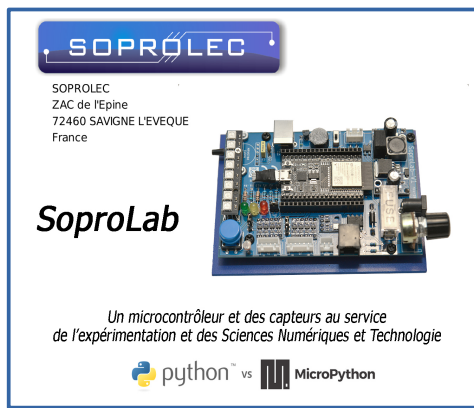
Dans le domaine scientifique et technique, tous nos élèves ne feront pas de la recherche fondamentale, mais beaucoup d'entre eux seront sans doute un jour amenés à réfléchir sur les moyens de simplifier une tâche en utilisant ou modifiant des outils de programmation (*feuille de calculs, systèmes automatisés, 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.



## 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

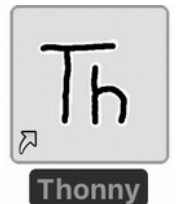
La première étape est indispensable : établir la connexion puis la communication entre la carte SoproLab et l'environnement Python.

*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 ...

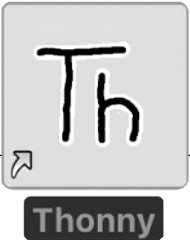
*En cas de problèmes, si cela peut vous aider, une documentation est disponible à cette adresse :*


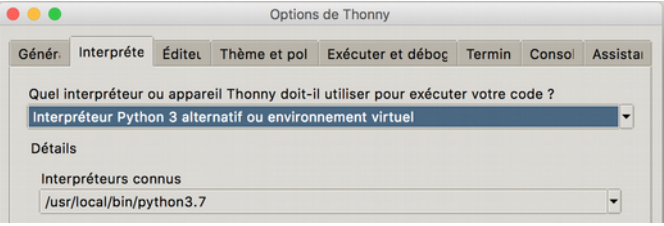
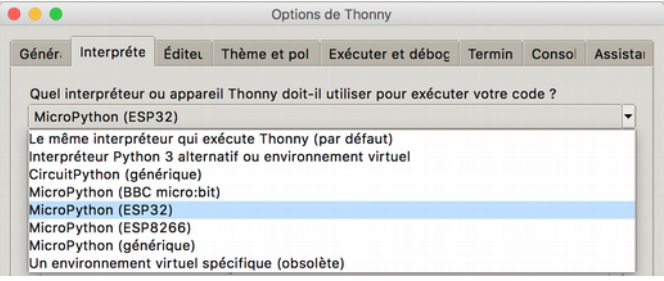
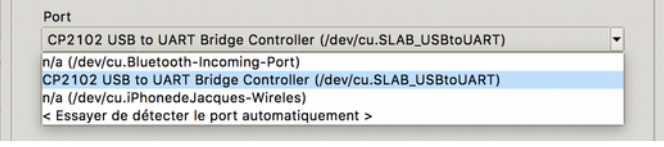
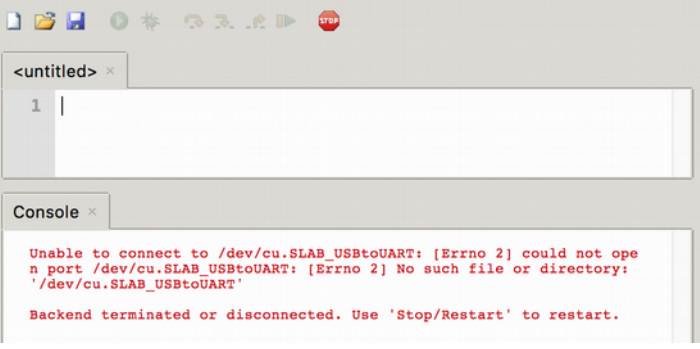
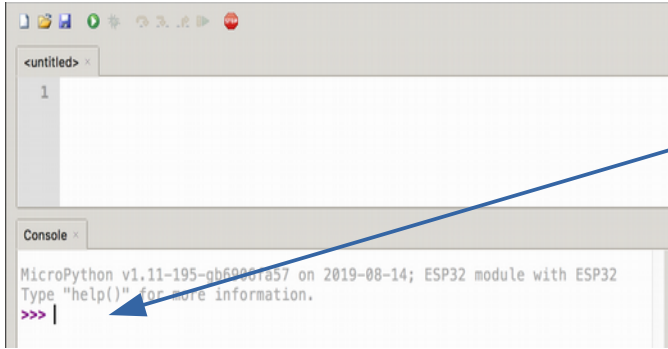
[https://www.xilinx.com/support/documentation/boards\\_and\\_kits/install/ug1033-cp210x-usb-uart-install.pdf](https://www.xilinx.com/support/documentation/boards_and_kits/install/ug1033-cp210x-usb-uart-install.pdf)



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

1 – Vous pouvez lancer le logiciel THONNY



	Dans le menu : - Sélectionner l'onglet [ Exécuter ] puis [ Sélectionner l'interpréteur ...]
	[ Interpréteur Python 3 ] si vous souhaitez développer une application destinée à être utilisée sur l'ordinateur. <i>Ex : tracé de vecteurs, graphiques, traitement de données, ...</i>
	[ MicroPython ESP32 ] ou [ MicroPython générique ] pour une application destinée à être utilisée sur la carte SOPROLAB.  [ MicroPython ESP8266 ] pour une application destinée à être utilisée avec le kit [Objets Connectés] en mode serveur.
	Choisir le port USB où est connecté la carte SOPROLAB ou le module Objet Connecté : Silicon Labs CP2102 USB <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>
	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. 1 – Vérifier que le câble USB est bien branché à chaque extrémité (carte et ordinateur). 2– Vérifier que vous avez choisi [ MicroPython (ESP32) ou (générique) ] comme interpréteur (étape N°3) 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. 4 – Essayer de relancer la connexion en cliquant sur l'icône rouge [ Stop ] dans la barre de menu
	<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>

## 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:
>>> |
```

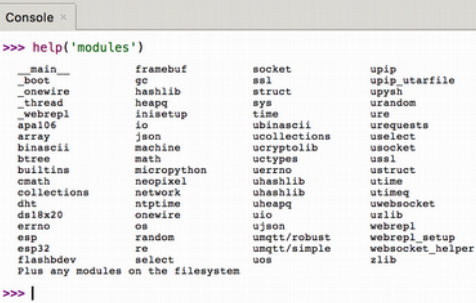
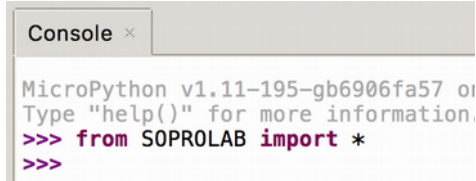
### Réinitialiser le microcontrôleur ou Interrompre un programme

Cliquer dans la fenêtre [ Console ] puis :

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

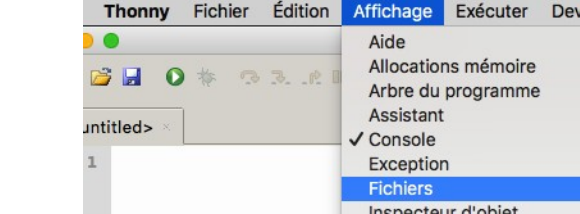
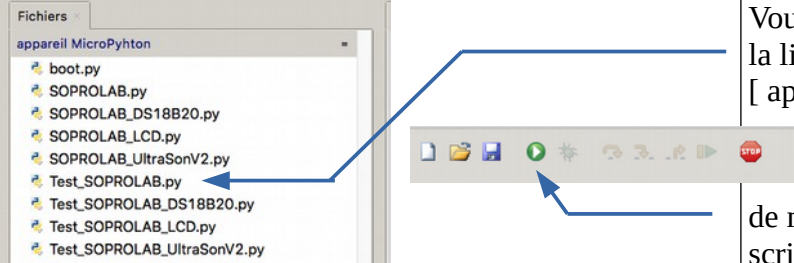
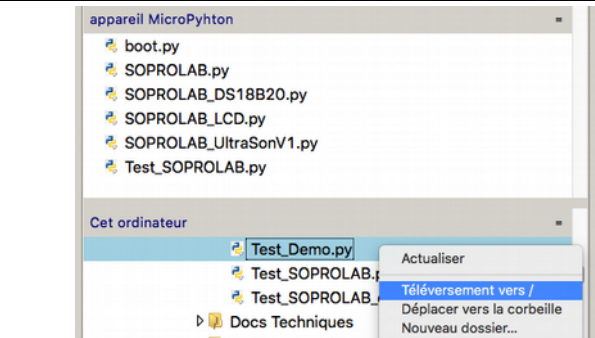
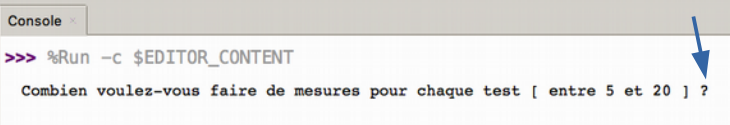
Ctrl + C → interrompre l'exécution d'un programme qui boucle ...



 <pre>&gt;&gt;&gt; help('modules') __main__      framebuf      socket          upip __boot        gc             ssl             upip.utarfile __onewire     hashlib        struct          upysh __thread       heapq          sys             urandom __webrepl     inisetup       time            ure apal06         io             uhashlib        urequests array          json           ucollections    uselect binascii       machine        ucryptolib      uselect btree          math           ctypes          usocket builtins       micropython    uerrno          ustruct cmath          neopixel       uhashlib        utime collections    network        uhashlib        utimeq dht            ntptime       uheapq          uwebsocket ds18b20        onewire        uio             uzlib errno          os             ujson           webrepl esp            random         umqtt/robust    webrepl_setup esp32          re             umqtt/simple    websocket_helper flashbdev     select         uos             zlib Plus any modules on the filesystem  &gt;&gt;&gt;  </pre>	<p>[ &gt;&gt;&gt; help('modules') ]</p> <p>Cette commande vous permet d’afficher toutes les bibliothèques disponibles avec microPython dans la mémoire programme du microcontrôleur.</p> <p>Chaque bibliothèque contient des fonctions spécifiques à importer avant de les utiliser : exemple la fonction [ <code>sleep(n)</code> ] pour effectuer des temporisations de [ <code>n</code> ] secondes. Elle est dans la bibliothèque [ <code>time</code> ] :</p> <ul style="list-style-type: none"><li>- soit vous importez toutes les fonctions de la bibliothèque time [ <code>import time</code> ] et vous utilisez ensuite la fonction sleep [ <code>time.sleep ( n )</code> ]</li><li>- soit vous n’importez que la fonction sleep : [ <code>from time import sleep</code> ] pour utiliser cette fonction ensuite : [ <code>sleep ( n )</code> ].</li></ul>
 <pre>MicroPython v1.11-195-gb6906fa57 on Type "help()" for more information. &gt;&gt;&gt; from SOPROLAB import * &gt;&gt;&gt;</pre>	<p>Si vous entrez l’instruction :</p> <p>[ &gt;&gt;&gt; from SOPROLAB import * ]</p> <p>vous avez l’impression qu’il ne s’est rien passé !</p> <p>Et pourtant vous venez de charger en mémoire toute la configuration de la carte pour accéder aux ressources le plus simplement possible.</p>

## 2- Ouvrir un programme de test comme exemple

Des programmes de test de la carte SoproLab sont déjà enregistrés dans la mémoire flash. Je vous présente ici le programme **Test\_SOPROLAB.py** afin d’explorer les fonctionnalités de la carte.

	<p>Pour afficher les bibliothèques et programmes d’exemples contenus dans la mémoire flash de la carte SoproLab, ajoutez la fenêtre [ FICHIER ] à l’écran :</p> <p>Vous pourrez alors visualiser, transférer, ...</p> <ul style="list-style-type: none"><li>- de nouvelles bibliothèques,</li><li>- visualiser le code python des programmes d’exemples.</li></ul>
	<p>Vous devriez voir le fichier [Test_SOPROLAB.py ] dans la liste des programmes présents dans la fenêtre : [ appareil MicroPython ] ( <i>carte SoproLab</i> ).</p> <p>Après avoir ouvert ce fichier de test, vous pouvez l’exécuter en cliquant sur l’icône verte dans la barre de menu ou partir du menu : [ Exécuter / Exécuter le script courant ] ou bien encore la touche [ F5 ].</p>
	<p><b>Pour installer une nouvelle bibliothèque microPython ou téléverser un programme destiné aux élèves</b></p> <p>A partir de la fenêtre [ Fichier ], vous faites un clic à droite sur le fichier à téléverser pour afficher le menu ci-contre.</p> <p>[ Téléverser vers / ] permet de transférer le fichier vers la mémoire flash du microcontrôleur.</p>
 <pre>&gt;&gt;&gt; %Run -c \$EDITOR_CONTENT Combien voulez-vous faire de mesures pour chaque test [ entre 5 et 20 ] ?</pre>	<p>La première instruction vous demande le nombre de mesures à effectuer pour chaque test.</p> <p>Vous pouvez répondre à la question <b>après avoir cliqué dans la fenêtre [ Console ], à la suite du point d’interrogation.</b></p>

Note : il se peut qu’à la première utilisation de la carte aucune donnée n’ait été mémorisée dans la mémoire de données (eeprom). Lors du deuxième essais, vous pourrez vérifier que les mesures enregistrées lors du premier essais s’affichent bien à l’écran.

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

#### BUZ → Buzzer

BUZ.beep()  
Produit un son de 800 Hz pendant 500ms

BUZ.son ( F, ms )  
Produit un son de fréquence [ F ] pendant une durée de [ ms ] millisecondes.

**Note :** Le buzzer est connecté à un interrupteur. Lorsque celui-ci est du côté du microcontrôleur, le buzzer est en marche.

#### LED\_v / LED\_j / LED\_r

LED\_x.on ( )  
Allume la LED Verte, Jaune ou Rouge

LED\_x.off ( )  
Eteint la LED Verte, Jaune ou Rouge

#### NEOPIX ( Led multicolores )

NEOPIX[i] = ( r, v, b )  
Détermine les composantes en lumières Rouge, Verte et Bleue produite par la LED n° i, avec  $i \in [0, 7]$  (max 255).  
NEOPIX[0]=(255, 255, 0) → Jaune

NEOPIX.write ( )  
Mise à jour de l'état des LED pour en changer l'aspect.



La propriété [ .valeur ] d'un objet (ex. MPX.valeur ) est le résultat de la Conversion Analogique Numérique sur 12bits → [ 0 ; 4095 ] **donc elle n'a pas d'unité.**

Dans notre exemple, MPX.valeur n'est pas une valeur en hPa !

#### MPX → Capteur de pression

MPX.mesure ( )  
Effectue une mesure. La valeur est disponible ensuite dans la variable [ MPX.valeur ]

MPX.valeur [ 0 ; 4095 ]  
Valeur de la mesure de pression  
Ex : print ("Mesure : ", MPX.valeur)

MPX.pression ( hPa )  
Cette fonction n'est pas encore intégrée. Elle permettra d'afficher la pression en hPa avec la réserve toutefois que ce sera une valeur approximative compte tenu de l'étalonnage nécessaire selon le capteur !

#### BP → Bouton poussoir

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 ( )

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 ].

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.

#### MEMOIRE [i] → eeprom

Cf. Note spéciale sur l'utilisation de l'eeprom en fin de cette Notice

#### LDR → Capteur de lumière

LDR.mesure ( )  
Effectue une mesure. La valeur est disponible ensuite dans la variable [ LDR.valeur ]

LDR.valeur [ 0 ; 4095 ]  
Valeur de la mesure de lumière  
Ex : print ("Mesure : ", LDR.valeur)

#### BMP → T°C et Pression atm

BMP.mesure ( )  
Effectue une mesure. Les valeurs sont disponibles ensuite dans les variables ci-dessous :

BMP.pression [ hPa ]  
Pression atmosphérique en hPa.

BMP.temperature [ °C ]  
Température ambiante en °C.

#### POT → Potentiomètre

POT.mesure ( )  
Effectue une mesure. La valeur est disponible ensuite dans les variables :

POT.valeur [ 0 ; 4095 ]  
Valeur de la mesure de tension  
Ex : print ("Mesure : ", POT.valeur)

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 ("Tension : ", POT.tension)

#### CTN → capteur de température

(sonde à brancher sur le connecteur J2 )

CTN.mesure ( )  
Effectue une mesure. La valeur est disponible ensuite dans la variable [ CTN.valeur ]

CTN.valeur [ 0 ; 4095 ]  
Valeur de la mesure de température  
Ex : print ("Mesure : ", CTN.valeur)

La bibliothèque associée à chaque module optionnel supplémentaire est téléchargeable depuis la plateforme github :

<https://github.com/SoproLab/Soprolab>  
dossier [ Modules ]

Une fois téléversée dans la mémoire flash, puis importée :  
[ from bibliothèque import \* ] les objets supplémentaires sont alors disponibles :

**HCSR** : capteur de distance à Ultrasons

**PIV** : commande d'orientation du servo moteur ( version 2 )

**LCD** : Ecran à cristaux liq.

**DS** : capteur de température numérique.

**BUZZER / MICRO** : mesure de la célérité du son dans l'air.

...

Console x

```
>>> from SOPROLAB import *
>>> dir(BUZ)
['_class_', '_init_', '_module_', '_qualname_',
 '_dict_', 'Pin', 'T', 'nbT', 'son', 'beep']
>>> BUZ.Pin
Pin(25)
>>> BUZ.beep()
>>> BUZ.son(100, 500)
>>> BUZ.T
10000
>>>
```

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.

Exemple : l'objet [ BUZ ] correspond au BUZZER.

[ >>> 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().

[ >>>> BUZ.Pin ] affiche le numéro de broche où est connecté le BUZZER.

[ >>>> BUZ.beep() ] est une fonction qui produit un son de 800 Hz pendant 500 ms → « beep ».

[ >>>> BUZ.son( F, t\_ms ) ] produit un son de fréquence [ F ] pendant une durée de [ t\_ms ] en ms.

[ >>>> BUZ.T ] est la période du son en µs.



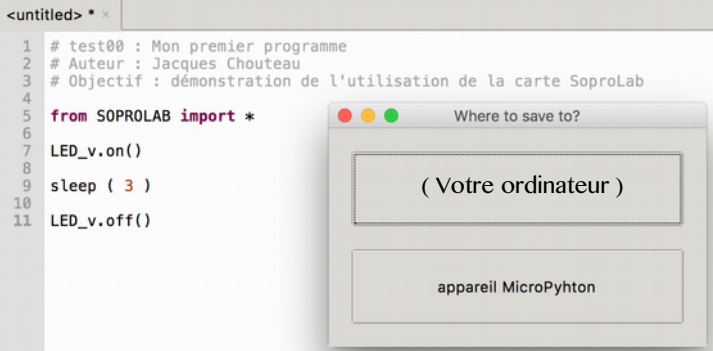
Vous pouvez parcourir les différents programmes de test contenus en mémoire ( ex : Test\_SOPROLAB.py ) afin d'en extraire des instructions en Python qui vous permettront de rédiger de nouvelles applications pour vos activités pédagogiques.

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 ;-)

	<p>Dans la fenêtre au dessus de la [ Console ], vous pouvez taper votre séquence d'instructions en microPython.</p> <p>Ici le programme permet d'allumer la LED verte pendant 3 secondes.</p>
	<p>Au moment d'exécuter votre programme, Thonny vous demandera alors dans quel espace de stockage vous souhaitez enregistrer votre programme avant de l'exécuter.</p>
	<p>Deux choix s'offrent à vous :</p> <ul style="list-style-type: none"><li>- 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.</li><li>- 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.</li></ul>

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 ]	[ T°C < 1500 ]	] 1500 – 2000 ]	] 2000 – 2500 ]	] 2500 < T °C ]
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 : 960 ↔ 3°C // 2200 ↔ 35°C // 3580 ↔ 88°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 :

```
Indication_Temperature.py x
1  """
2  Programme : INDICATION_TEMPERATURE.py
3  Auteur : Jacques Chouteau - Janv 2020 -
4  Objectif : Donner une indication de la température d'un liquide
5              par un signal lumineux, voire sonore.
6  Note : ce code pourrait être optimisé, mais l'objectif principal
7  est de le rendre facile à comprendre.
8  """
9  from SOPROLAB import *
10
11  print("=== Mesures en cours ===")
12  print("Pour terminer appuyez sur le bouton poussoir ...")
13
14  BP.drapeau = False
15
16  while BP.drapeau == False:
17
18      CTN.mesure() # Mesurer la température
19      Temp = CTN.valeur
20      print(Temp)
21
22      if Temp <= 1500:
23          LED_v.on()
24          LED_j.off()
25          LED_r.off()
26      elif Temp>1500 and Temp <=2000:
27          LED_v.on()
28          LED_j.on()
29          LED_r.off()
30      elif Temp>2000 and Temp <=2500:
31          LED_v.on()
32          LED_j.on()
33          LED_r.on()
34      else:
35          LED_v.on()
36          LED_j.on()
37          LED_r.on()
38          BUZ.son ( 1200, 1000 )
39      sleep ( 2 )
40  print("=== Fin du programme ===")
41
```

### Précaution : Fonction print et temporisation

Lors de l'utilisation de la fonction [ print ( ... ) ], il est conseillé d'effectuer une temporisation de quelques dizaines de millisecondes [ sleep\_ms(100) ] 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, on peut être confronté à des « plantages » du microcontrôleur alors qu'il n'y a aucune erreur de codage en Python.

### Note Spéciale à propos de l'utilisation de la mémoire de données : l'eprom

à faire : explication du stockage des informations octet par octet : MEMOIRE[i]

MEMOIRE[i] = val → écriture dans l'eprom à l'octet [ i ] de la valeur contenue dans la variable val.

val = MEMOIRE[i] → lecture de l'octet n°i contenu dans l'eprom puis stockage dans la variable val.