



STMicroelectronics Rennes R&D Projet pyKare Kit de démarrage

Equipe Kare

Septembre 2021



Le projet Kare

- Ce projet est né en aout 2018 suite une participation à un fab lab (« l'Atelier Partagé » à Betton) où un projet de boîte connectée à base d'Arduino regroupait des personnes de 10 à 70 ans
- Devant l'intérêt et la facilité des plus jeunes, l'idée est venue de faire contribuer nos stagiaires d'observation de 3ème à la conception d'une telle boîte, à base de composants STMicroelectronics
- Une équipe multi-métiers de 15 personnes s'est formée sur le site fin aout 2018 pour arriver au premier prototype testé par les premiers stagiaires en décembre 2018
- En 2021, changement de version avec le microPython: le projet piKare arrive !
- « Kare » c'est « aimer, désirer » en breton, « être ensemble dans un groupe » dans d'autres langues... il résume donc notre ambition!

Ton objectif...

- Créer un objet similaire à celui vendu chez IKEA... mais fait par toi-même...

The image shows a screenshot of the IKEA France website. At the top, there is a navigation bar with the IKEA logo, a search bar, and links for "Mon panier" (Cart) with a count of 0, "Nos magasins" (Stores), "IKEA FAMILY", and "IKEA BUSINESS". Below the navigation bar is a secondary menu with categories like "Nos produits", "Nouveautés", "Chambre", "Salle de bain", etc., and a "Toute la maison" link. The main content area displays a white digital clock with a small screen showing the date and time, and various icons for alarm, month/date, and timer. To the right of the product image, the product name "KLOCKIS" is displayed in bold, followed by a description: "Horloge/thermomètre/réveil/minuteur, blanc". The price is listed as "4,99 €" with a note about "Eco-part. DFFF 0,10 €". Below the price, the article reference "Référence de l'article : 802.770.04" is shown. A rating section indicates "4.19 (91) Avis". A detailed description follows: "Cette horloge occupe très peu de place, est facile à utiliser et chacun de ses côté a une fonction différente : heure/date, thermomètre et alarme. Il suffit de retourner l'horloge pour changer de fonction." At the bottom, there is a "Quantité:" input field set to "1", a blue "Acheter en ligne" (Buy online) button, and a grey "Ajouter à la liste d'achats magasin" (Add to shopping list) button.

- Le projet est découpé en ateliers progressifs
- La vérification du matériel
- L'installation de l'environnement sur un PC
- La prise en main de la carte STM32 et du moniteur textuel
- La prise en main de la carte des capteurs et du moniteur graphique
- La mise en œuvre de l'affichage
- La mise en œuvre du bouton molette
- La mise en œuvre de toutes les fonctions

Atelier 1 - La vérification du matériel

Préparation mécanique préliminaire

- A faire par les tuteurs mais en dehors du temps de travail, l'ébavurage du boîtier 3D, qui peut nécessiter d'utiliser des outils tranchants.



Le matériel

- Matériel remis au stagiaire :

Boitier imprimé en 3D



Capteur de température

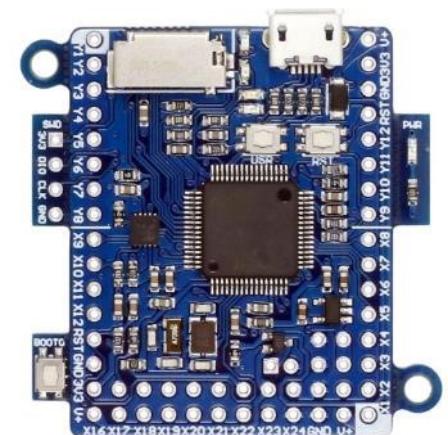


Câbles Dupont

Ecran LCD 128x128



Roue codeuse



pyBoard

Support pour stages d'observation - ST - Septembre 2021

Préparation informatique préliminaire

- A apporter par le stagiaire
 - Un PC laptop sous Windows 10
 - Avec l'environnement Thonny installé au préalable comme indiqué dans cette présentation
 - Ceci permettra au stagiaire de repartir chez lui à la fin de la période avec son environnement qui lui permettra de poursuivre le projet ou de continuer à utiliser des produits ST
 - Un câble mini-USB

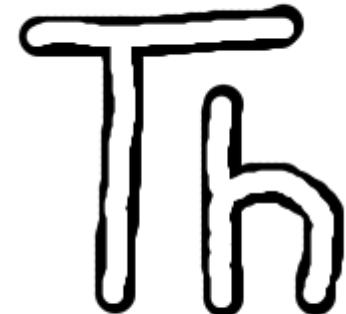


- Le tuteur aura à faire la soudure des connecteurs sur les différents composants.



Atelier 2 – Environnement sur PC

- Nous avons choisi d'effectuer le projet dans l'environnement Thonny qui est plus simple pour démarrer
- Thonny est un environnement de prototypage Python.
- Ce qui va nous intéresser
 - Des cartes matérielles STMicroelectronics
 - Un Environnement de Développement Intégré (IDE) permettant d'écrire du code programme et le charger sur le matériel
 - Des bibliothèques logicielles qui permettent de faire fonctionner les cartes matérielles

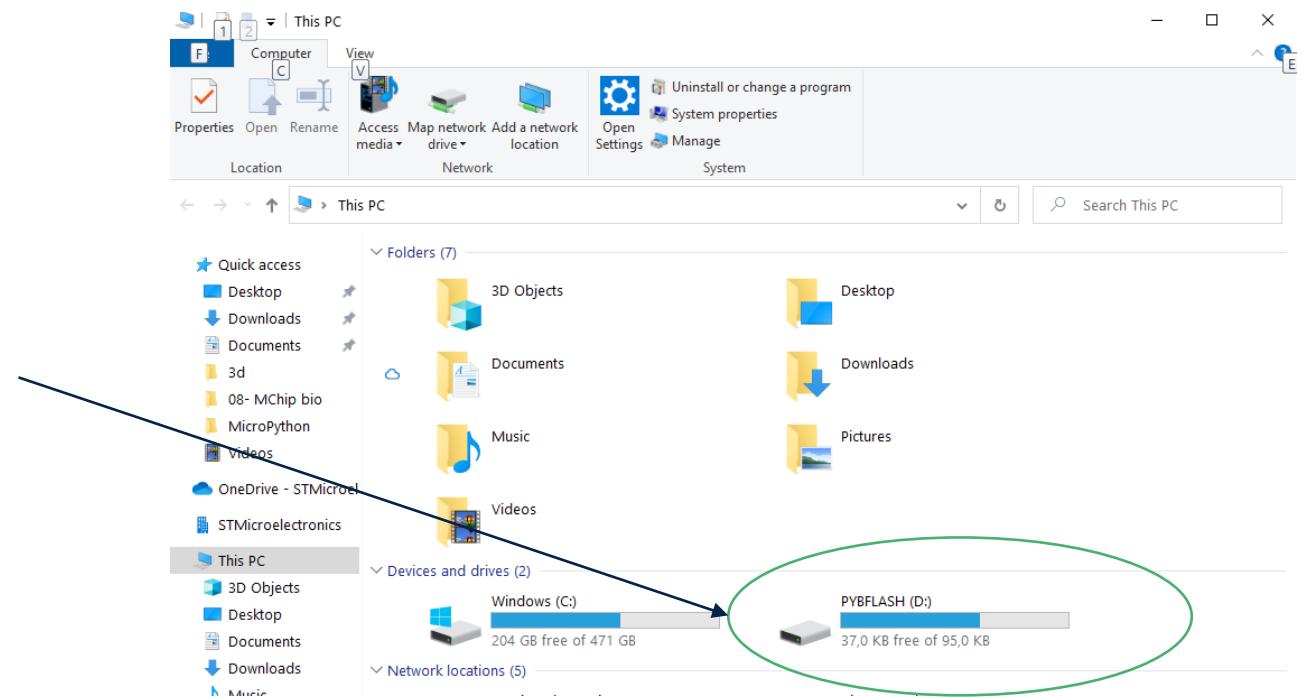


Etape 0: installation de la pyboard

- Connecter la pyBoard à un port USB du PC avec le câble micro-USB
- Dans l'explorateur de fichier, un nouveau volume doit apparaître:
 - PYFLASH

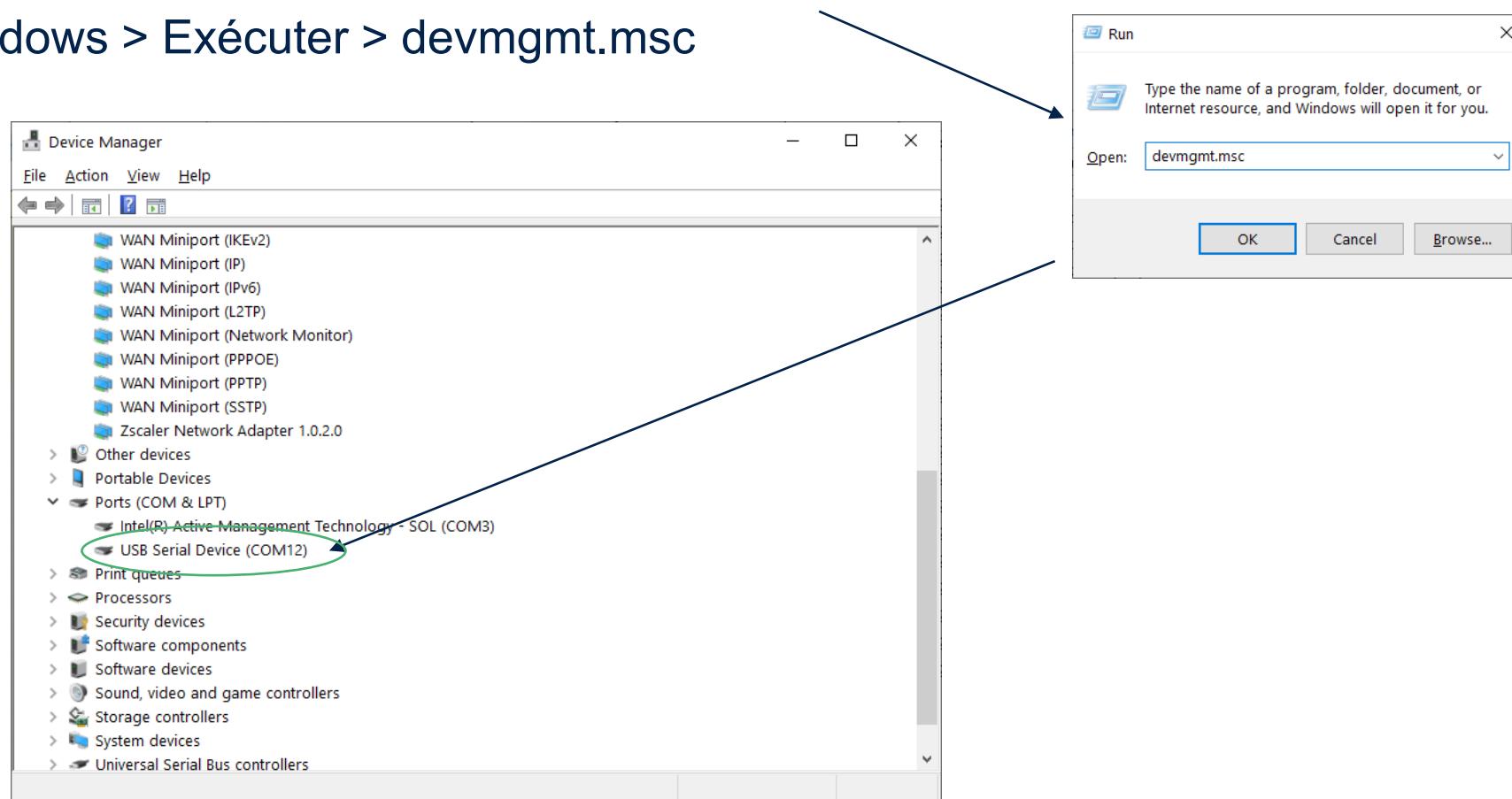


Attention: certains câbles USB ne font que l'alimentation. Le câble doit permettre le transfert de données, sinon la carte n'est pas reconnue sur le PC



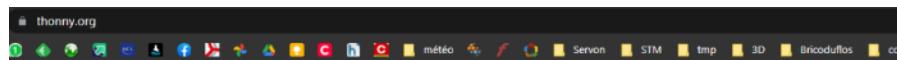
Etape 0: installation de la pyboard

- Dans le gestionnaire de périphérique, un nouveau port com doit apparaître:
 - Windows > Exécuter > devmgmt.msc



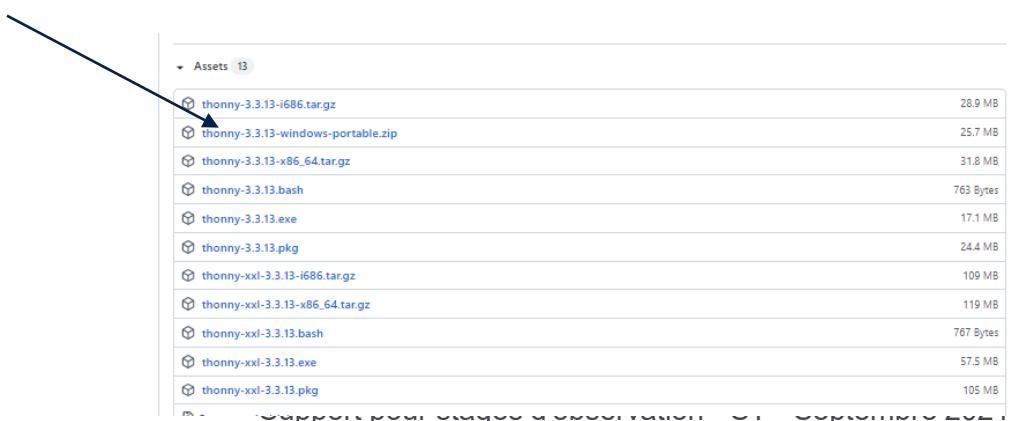
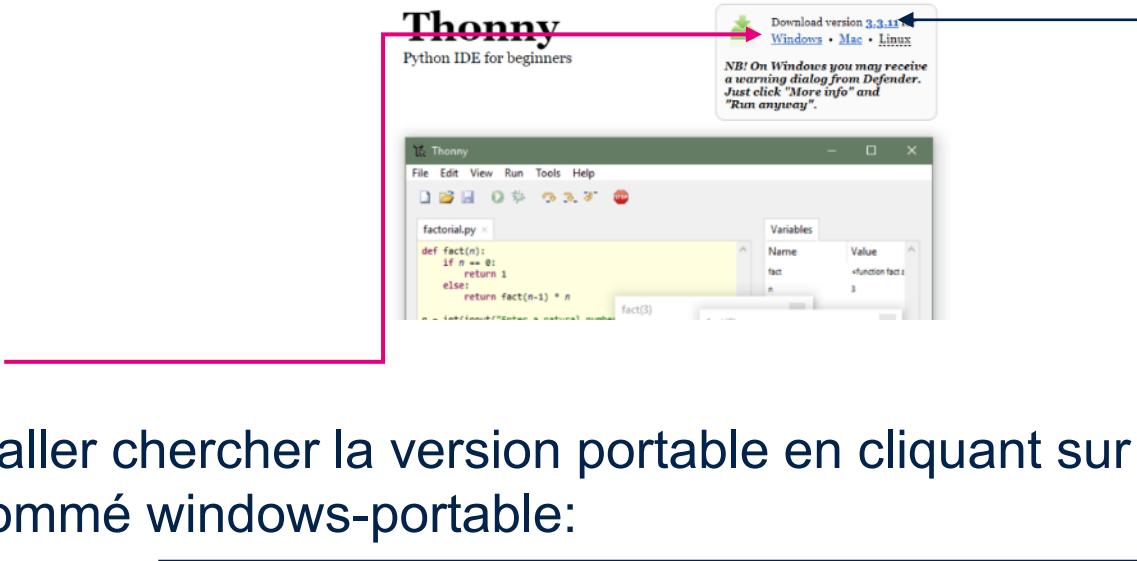
Etape 1: récupération de thonny

- Aller sur <https://thonny.org/>



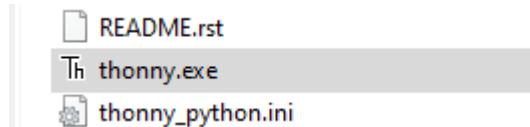
- Cliquer sur Downloads » Windows

- Ou si pas administrateur sur la machine, aller chercher la version portable en cliquant sur la version (ici 3.3.11) et télécharger le zip nommé windows-portable:
 - Par exemple: <https://github.com/thonny/thonny/releases/download/v3.3.13/thonny-3.3.13-windows-portable.zip>



Etape 2: installation de l'IDE

- Sous Windows10 l'installation se lance d'elle-même:
 - Les options par défaut sont les bonnes
- Sous un autre OS, double-cliquer sur le fichier d'installation qui a été téléchargé
 - Si pas admin, il suffit d'extraire le zip dans le répertoire de travail :
 - C:\temp, Mes documents...
- Lancer Thonny.exe



Name	Date	Size	Last modified	Content	Advanced	Attributes	Permsiged	Comment	CRC	Method	Characteris	Host OS	Version	Volume index
Thonny-3.3.3-windows-pestauchi	2021-07-25 18:07	4 300 356	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	D	-	-	AC114056	Raw	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin	2021-07-25 18:07	751 776	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	D	-	-	77420117	Raw	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny	2021-07-25 18:07	59 561 777	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	D	-	-	TC430701	Raw	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources	2021-07-25 18:07	58 535 031	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	D	-	-	5079111A	Raw	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons	2021-07-25 18:07	62 101	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	D	-	-	20080904	Raw	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\16x16	2021-07-25 18:07	5 35	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	D	-	-	CE925001	Raw	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\32x32	2021-07-25 18:07	1 007 317	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	D	-	-	7056191E	Raw	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\48x48	2021-07-25 18:07	11 615	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	39757C4C	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\64x64	2021-07-25 18:07	11 819	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	0F1C2353	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\80x80	2021-07-25 18:07	11 745	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	24240B00	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\128x128	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	94020202	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\160x160	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	FE43446B	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\256x256	2021-07-25 18:07	11 615	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	47450355	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\512x512	2021-07-25 18:07	11 615	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	3C7D1010	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\1024x1024	2021-07-25 18:07	12 142	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	104A1043	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\2048x2048	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	393A449F	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\4096x4096	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	7E9B0462	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\8192x8192	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	B0B33372	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\16384x16384	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	CE925002	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\32768x32768	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	2A37373C	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\65536x65536	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	2E14C485	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\131072x131072	2021-07-25 18:07	12 128	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	20561620	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\262144x262144	2021-07-25 18:07	12 140	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	CA5C5418	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\524288x524288	2021-07-25 18:07	12 171	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	7056191B	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\1048576x1048576	2021-07-25 18:07	12 171	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	1E3320E7	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\2097152x2097152	2021-07-25 18:07	12 171	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	4A3C224D	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\4194304x4194304	2021-07-25 18:07	12 171	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	AD483D47	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\8388608x8388608	2021-07-25 18:07	12 171	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	ED4F5388	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\16777216x16777216	2021-07-25 18:07	12 171	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	2F071010	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\33554432x33554432	2021-07-25 18:07	12 171	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	2A0431A1	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\67108864x67108864	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	05356885	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\134217728x134217728	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	BB422025	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\268435456x268435456	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	48422026	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\536864912x536864912	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	870700A0	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\1073729824x1073729824	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	103A5981	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\2147459648x2147459648	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	088E1C18	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\4294919360x4294919360	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	AB4C5402	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\8589838640x8589838640	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	89947477	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\1717967680x1717967680	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	073C2027	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\3435935200x3435935200	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	02C935A7	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\6871870400x6871870400	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	9E843518	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\13743740800x13743740800	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	273C4027	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\27487481600x27487481600	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	573C4028	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\54974963200x54974963200	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	873C4029	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\109949926400x109949926400	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	02C935A8	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\219899852800x219899852800	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	9E843519	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\439799705600x439799705600	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	273C402A	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\879599411200x879599411200	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	573C402B	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\1759198822400x1759198822400	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	873C402C	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\3518397644800x3518397644800	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	02C935A9	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\7036795289600x7036795289600	2021-07-25 18:07	12 172	2021-07-25 18:07	2021-07-25 18:07	2021-07-25 18:07	A	-	-	9E84351A	Deflate	NFTS	FAT	20	0
Thonny-3.3.3-windows-pestauchi\bin\Thonny\resources\icons\14073590579200x14073590579200	20													

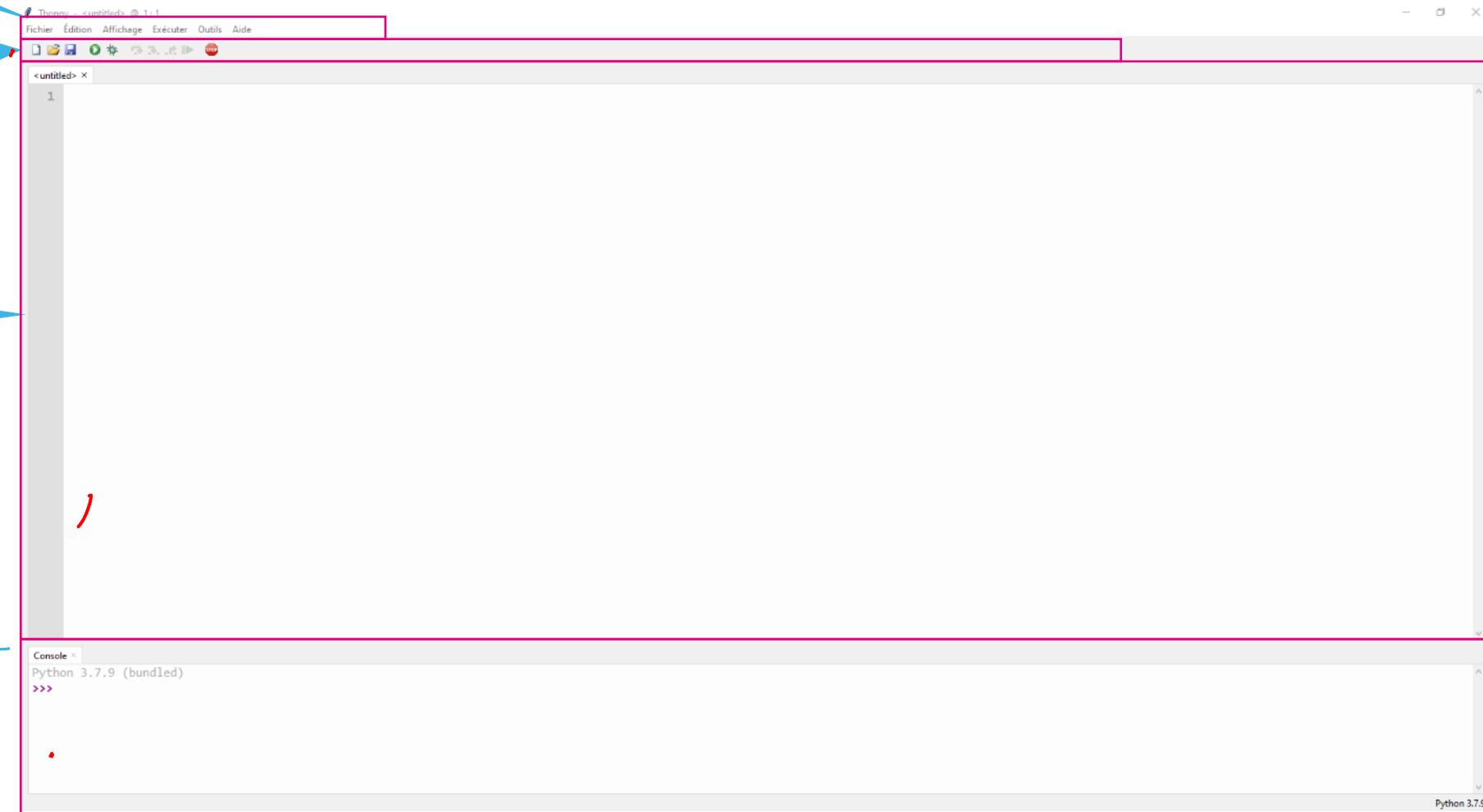
Menu principal

Icônes pour les opérations fréquentes

Editeur de programme

Console

Etape 3: interface utilisateur à l'installation



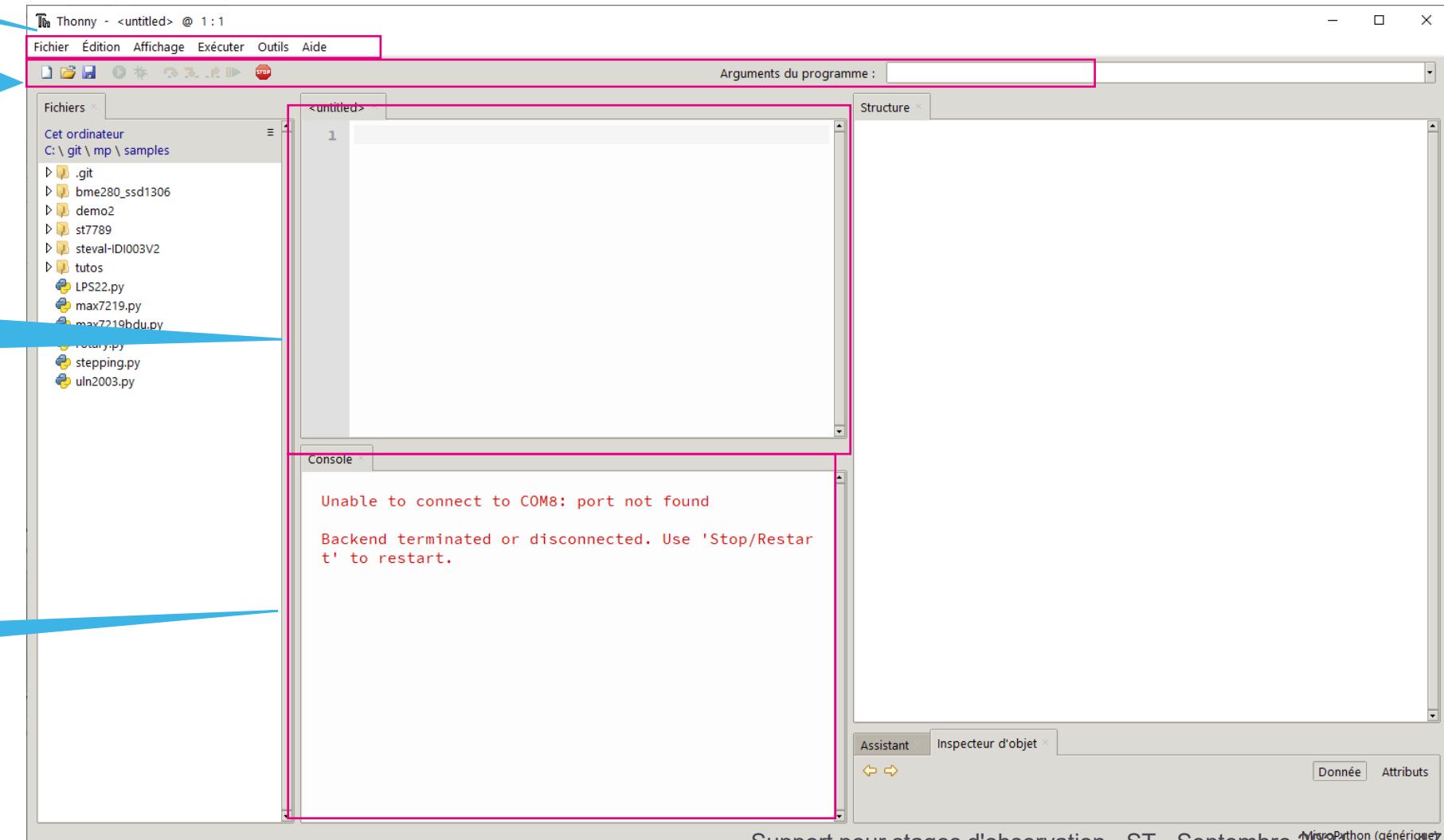
Menu principal

Icônes pour les opérations fréquentes

Editeur de programme

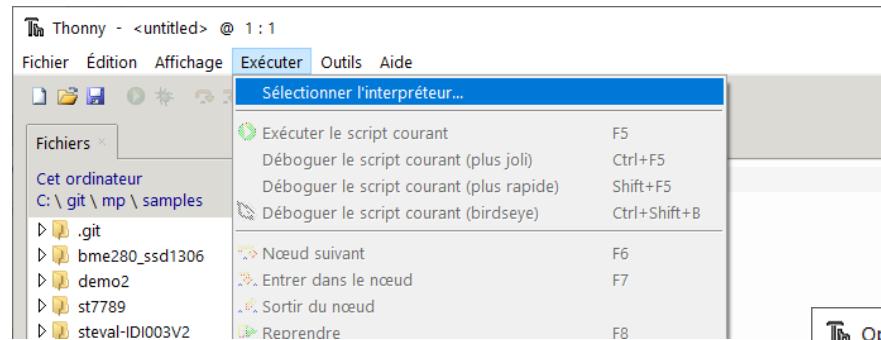
Console

Etape 3: interface utilisateur avancée



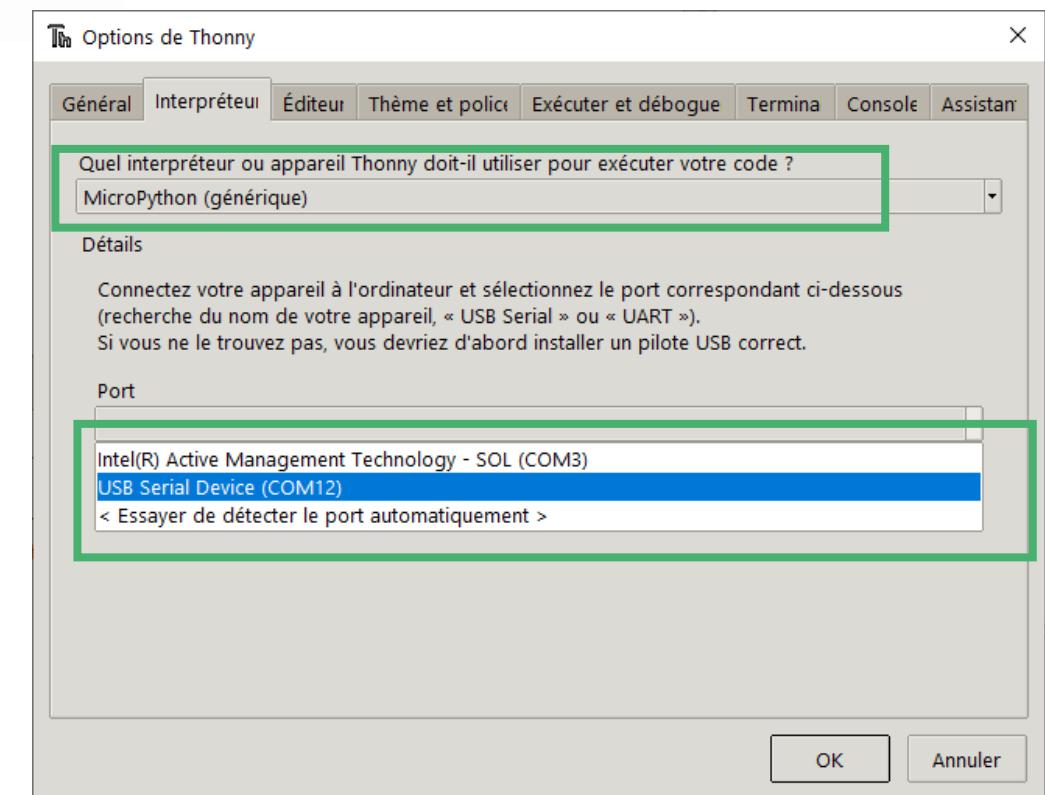
Etape 4: réglages pour la pyboard

- Dans Thonny:



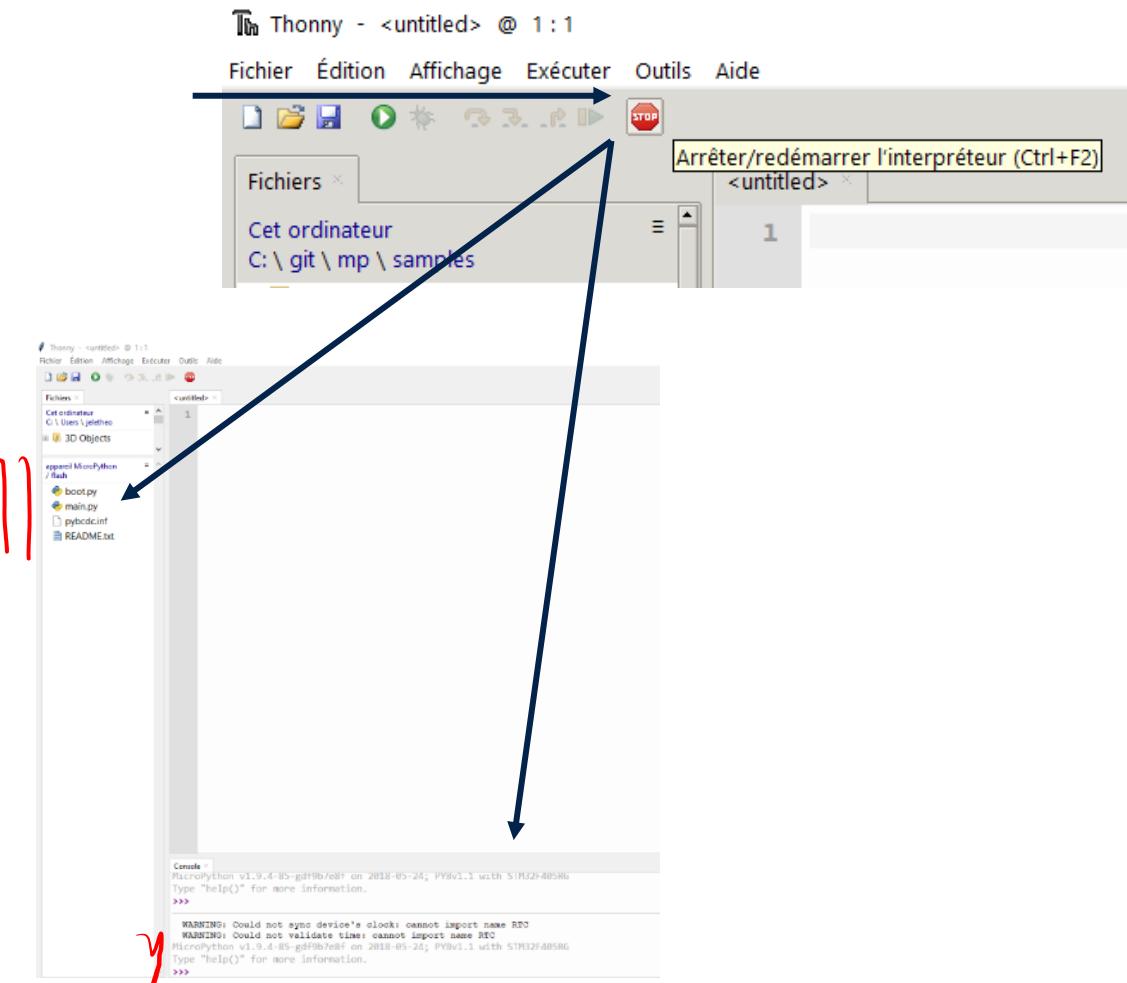
- Sélectionner « Micropython (générique) »
- Sélectionner le port COM de l'installation
(ou <Essayer de détecter le port automatiquement>)

```
Console >>> import time, RTC
WARNING: Could not sync device's clock: cannot import name RTC
WARNING: Could not validate time: cannot import name RTC
MicroPython v1.9.4-85-gdf9b7e8f on 2018-05-24; PYBv1.1 with STM32F405RG
Type "help()" for more information.
>>>
```



Etape 4: réglages pour la pyboard

- Presser Stop pour synchroniser la board
- Sélectionner « Affichage/Fichiers »
- Le contenu de la pyboard doit apparaître en bas à gauche:
 - Seul un main.py et un boot.py sont présents
- En bas à droite apparaît la version de la pyboard ainsi que le « prompt » (l'invite) python
 - >>>



Etape 4 : Les fichiers

- On peut éditer/exécuter les programmes soit à partir du PC (privilégié) soit à partir de la pyBoard (quand c'est au point)

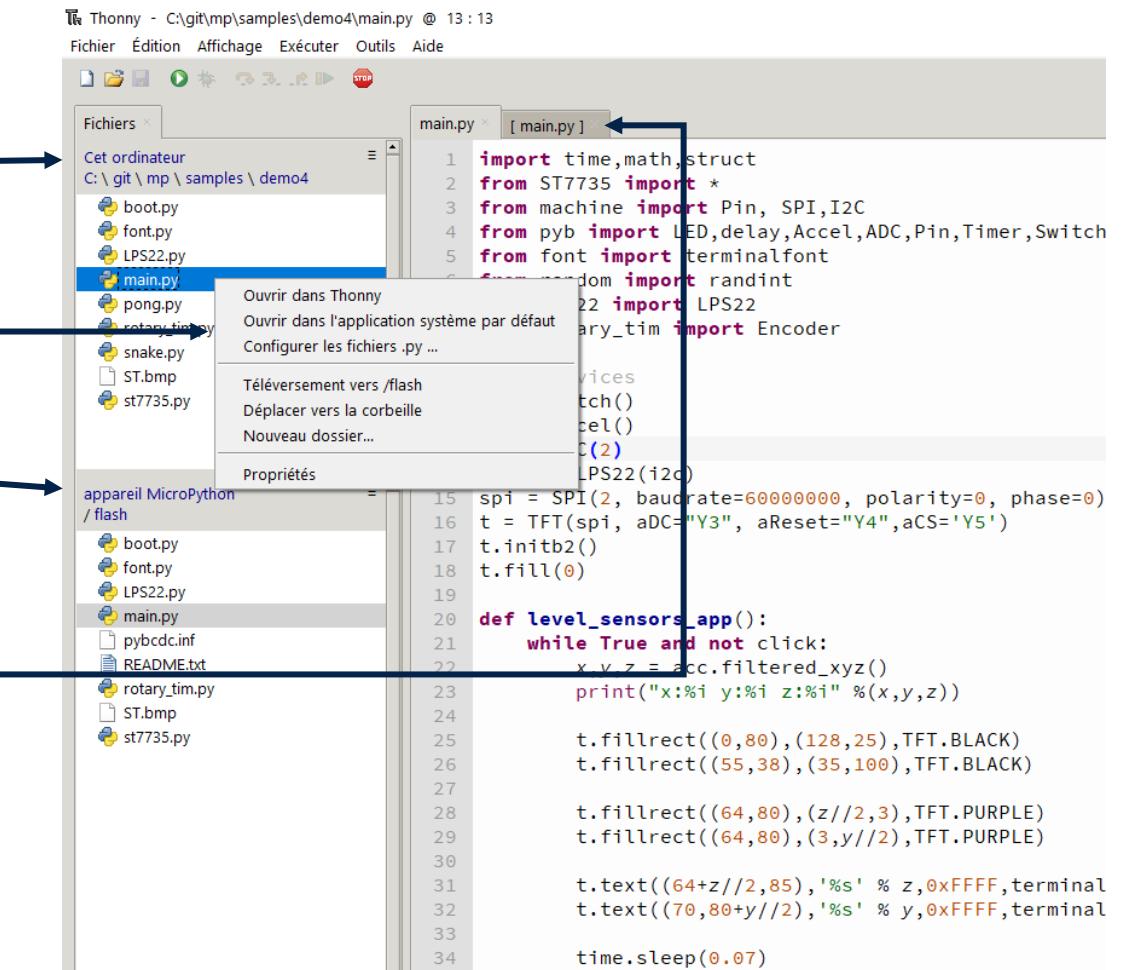
- **Fichiers sur le PC**

- Menu pour transferer des fichiers
 - Click-droit > Téléversement vers /flash

- **Fichiers sur la pyBoard**

- **Édition:**

- L'édition sur la carte est indiquée avec des crochets : [xxx.py]
 - L'édition sur la PC est indiquée sans crochets: xxx.py



The screenshot shows the Thonny IDE interface. On the left, there are two file browser panes. The top pane shows files on the local machine: boot.py, font.py, LPS22.py, main.py, pong.py, rotary_tim.py, snake.py, ST.bmp, and st7735.py. The bottom pane shows files on the microcontroller: boot.py, font.py, LPS22.py, main.py (selected), pybcd.inf, README.txt, rotary_tim.py, ST.bmp, and st7735.py. A context menu is open over the 'main.py' file in the bottom pane, with options like 'Ouvrir dans Thonny', 'Téléversement vers /flash', and 'Déplacer vers la corbeille'. The code editor on the right contains Python code for a TFT display driver.

```
import time,math,struct
from ST7735 import *
from machine import Pin, SPI,I2C
from pyb import LED,delay,Accel,ADC,Pin,Timer,Switch
from font import terminalfont
from random import randint
import LPS22
from rotary_tim import Encoder

def level_sensors_app():
    while True and not click:
        x,y,z = acc.filtered_xyz()
        print("x:%i y:%i z:%i" %(x,y,z))

        t.fillrect((0,80),(128,25),TFT.BLACK)
        t.fillrect((55,38),(35,100),TFT.BLACK)

        t.fillrect((64,80),(z//2,3),TFT.PURPLE)
        t.fillrect((64,80),(3,y//2),TFT.PURPLE)

        t.text((64+z//2,85),'%s' % z,0xFFFF,terminalfont)
        t.text((70,80+y//2),'%s' % y,0xFFFF,terminalfont)

        time.sleep(0.07)
```

! Attention bien appuyer sur STOP avant de téléverser des fichier vers la pyBoard

Etape 5: premiers tests

- Dans la console taper :
 - print("hello world")
 - <enter>
 - La carte pyboard répond...
- Taper les commandes suivantes:
 - Une led s'allume sur la board...

```
MicroPython v1.15-136-gea81bcf1c on 2021-05-21; PYBv1.1
with STM32F405RG
Type "help()" for more information.
>>> print("hello world")
· hello world
>>> |
```

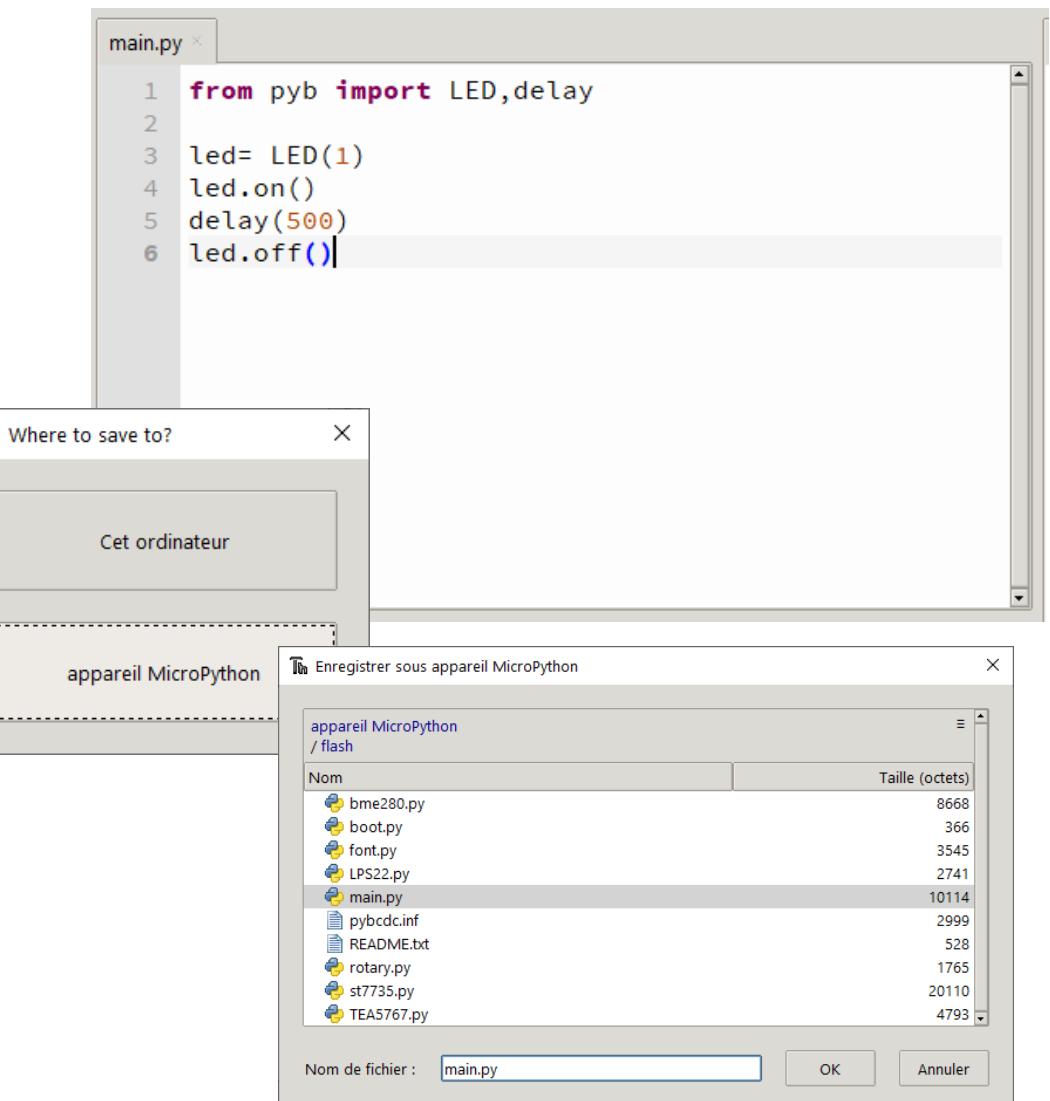
```
>>> from pyb import LED
>>> led = LED(1)
>>> led.on()
>>> |
```

Etape 6: premier programme

- Voici un premier programme:

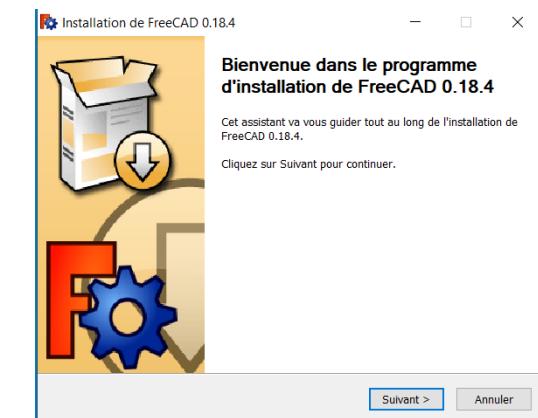
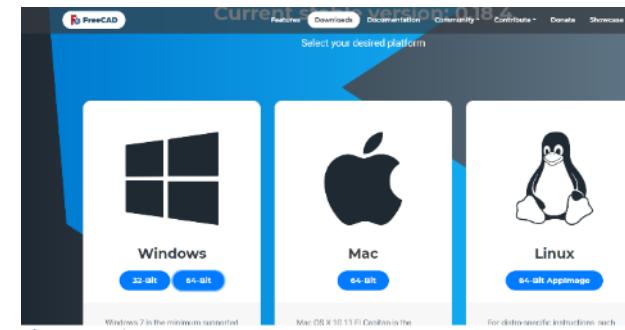
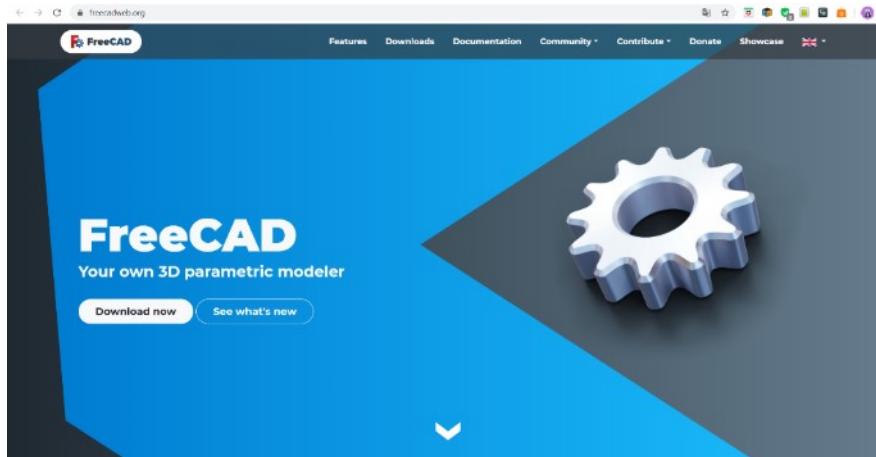
- Cliquer sur
- Entrer le programme →  main.py

- L'enregistrer sur “appareil Micropython”
 - main.py
- Lancer le programme avec « Exécuter/exécuter le script courant »
- Au lieu de seulement exécuter le programme sur la carte pyBoard à partir du PC, comme on a enregistré le programme sur la pyBoard, à chaque démarrage de la carte, la LED rouge va s'allumer 500ms puis s'éteindre, sans même connecter le PC: le système est autonome !



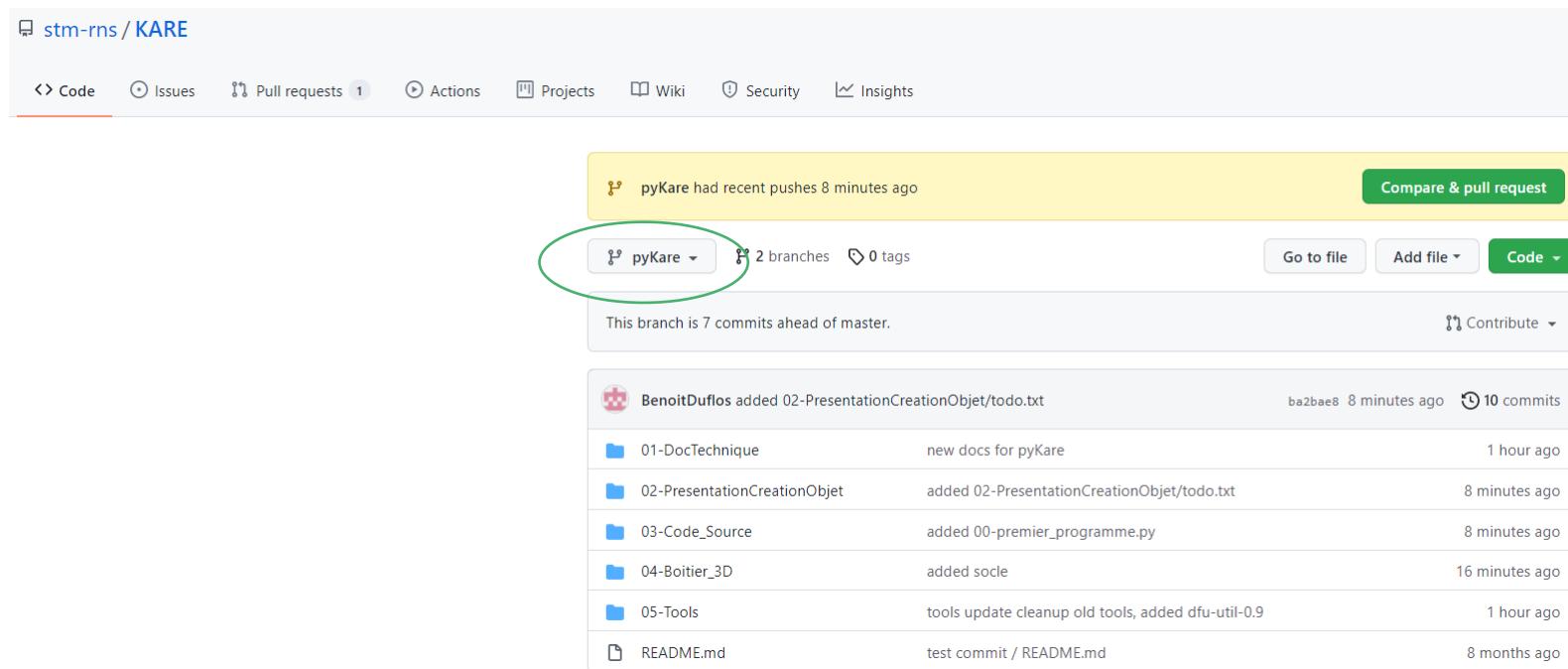
Etape 6: installation de freecad

- Si la création d'une partie de la boite en 3D fait partie du projet, installer FreeCAD sur le PC: <https://www.freecadweb.org/>



Etape 7: la bibliothèque des exemples

- Les fichiers sources des différents exemples à suivre sont disponibles sur GitHub:
<https://github.com/stm-rns/KARE/tree/pyKare>



pyKare had recent pushes 8 minutes ago

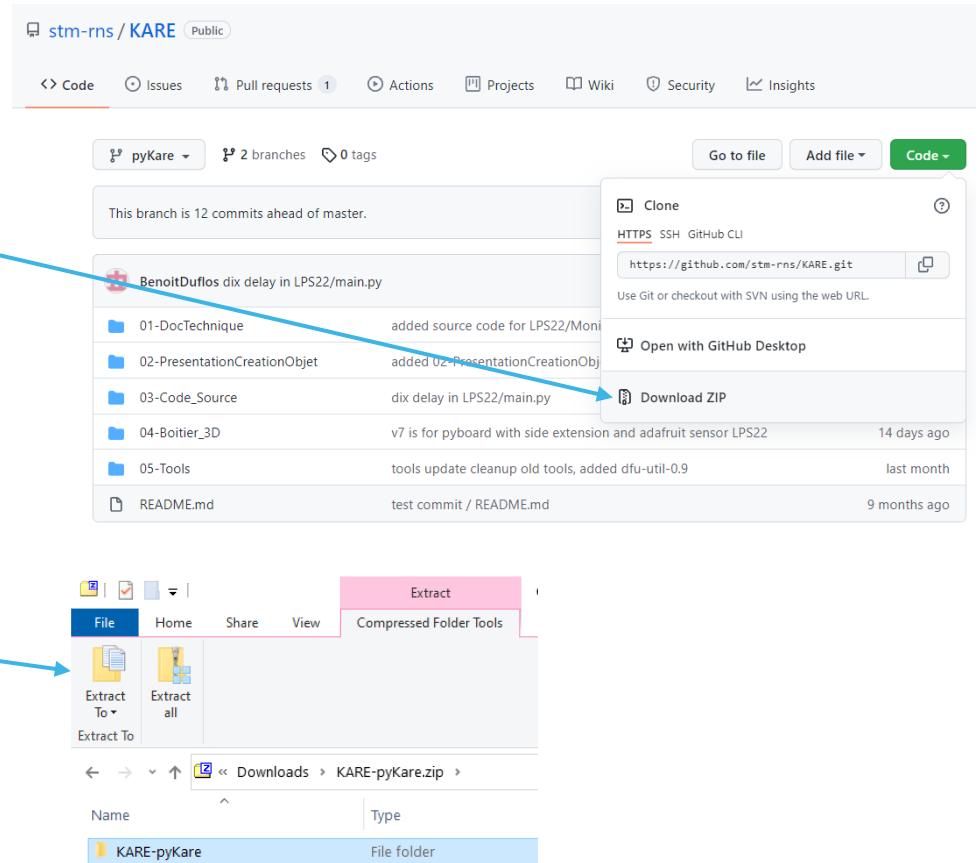
pyKare ▾ 2 branches 0 tags

This branch is 7 commits ahead of master.

Author	Commit Message	Date	Commits
BenoitDuflos	added 02-PresentationCreationObjet/todo.txt	ba2bae8 8 minutes ago	10 commits
	new docs for pyKare	1 hour ago	
	added 02-PresentationCreationObjet/todo.txt	8 minutes ago	
	added 00-premier_programme.py	8 minutes ago	
	added socle	16 minutes ago	
	tools update cleanup old tools, added dfu-util-0.9	1 hour ago	
	test commit / README.md	8 months ago	

Où trouver les sources des programmes

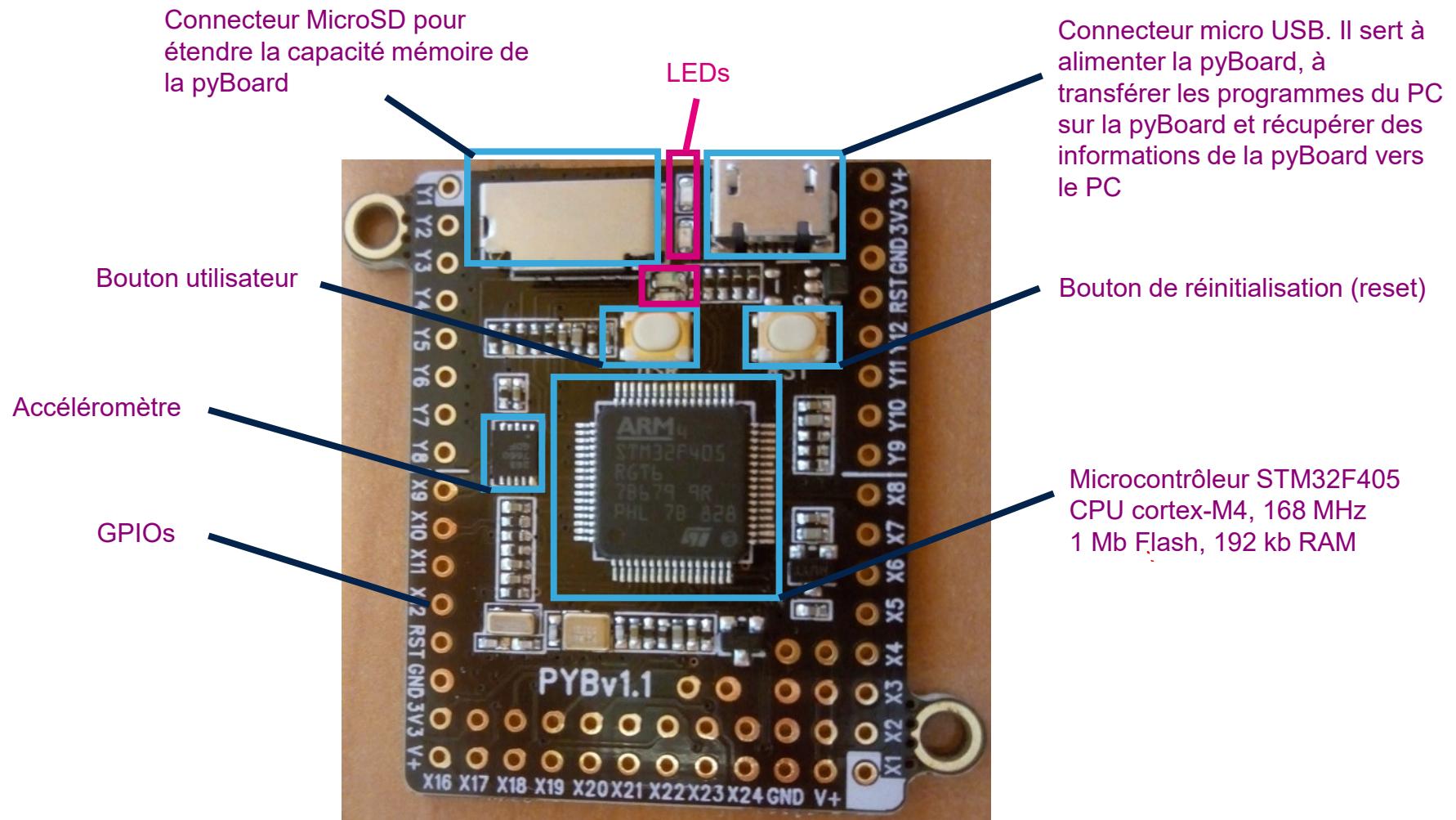
- Télécharger tous les documents en local :
 - Code > Download ZIP
- Sur votre PC:
 - Explorateur de fichier > Téléchargement
 - Selectionner/Ouvrir : Kare-pyKare.zip
 - Extraire vers > choisir son repertoire de travail:
 - C:\Data\



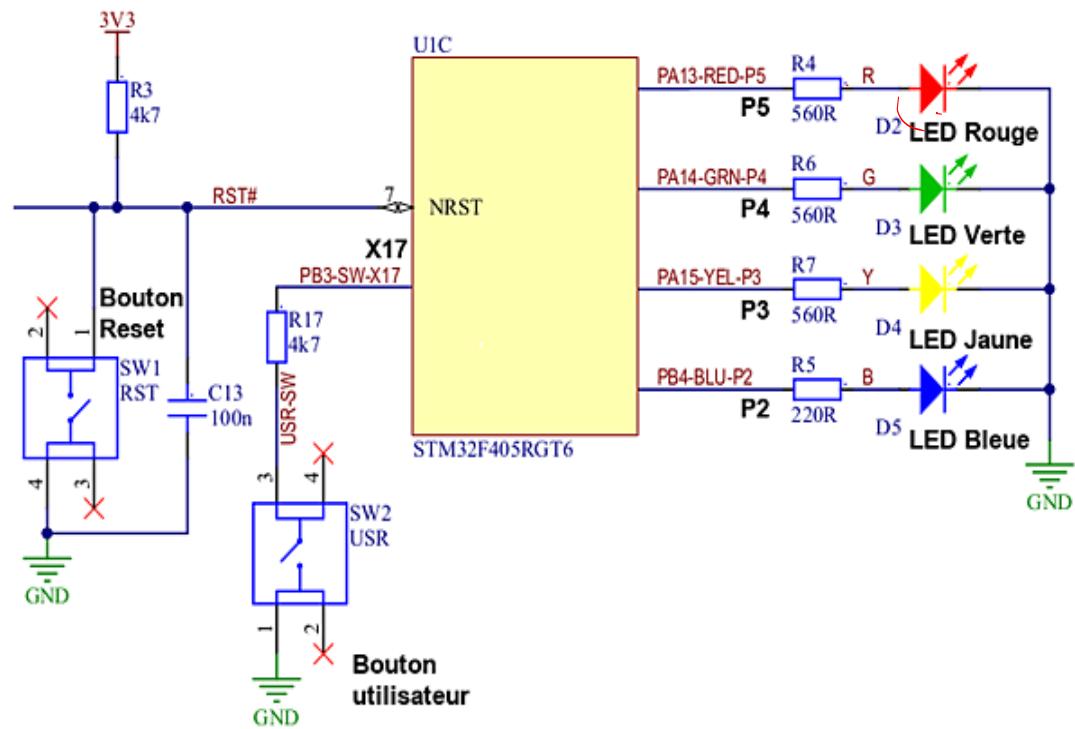
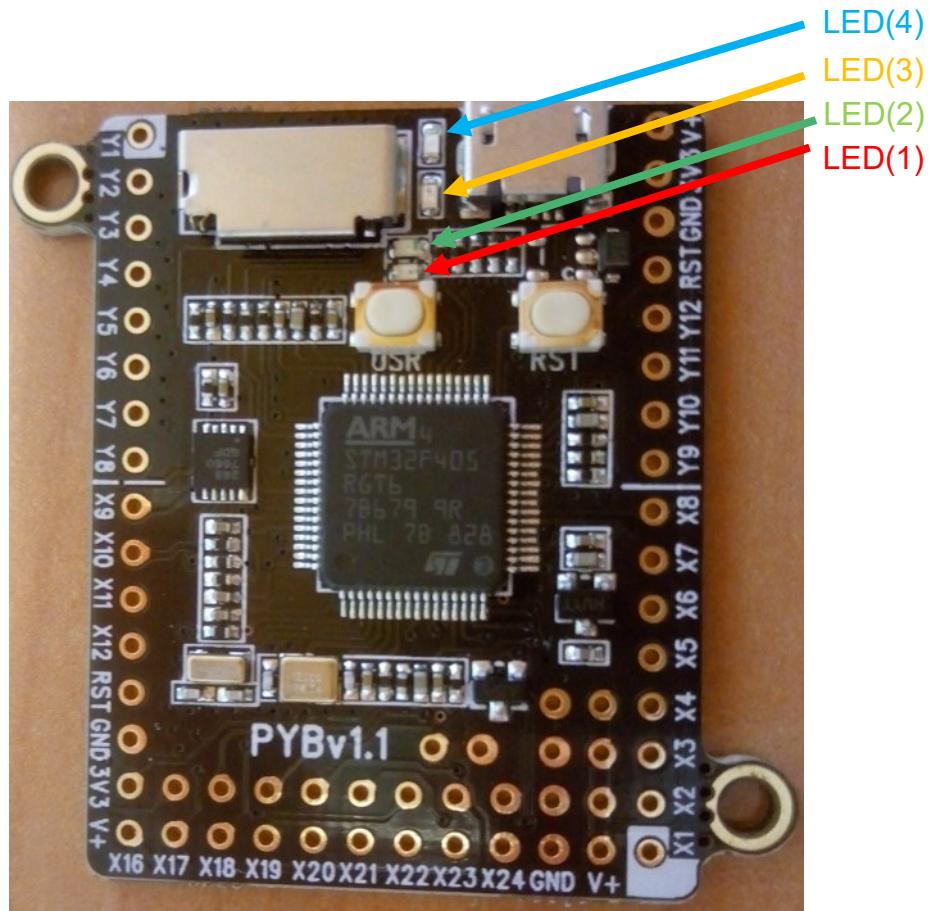
Atelier 3 – Prise en main de la pyBoard

- Le but de cet atelier est d'explorer le contenu de la pyBoard et de faire fonctionner des éléments présents sur la carte
 - Etape 1: allumer et éteindre les 4 LEDs
 - Etape 2: faire clignoter les 4 LEDs dans un ordre aléatoire
 - Etape 3: faire afficher des données du déroulement du programme sur l'écran du PC

Contenu de la pyBoard



Contenu de la pyBoard: les LEDs



Etape 1: allumer et éteindre les LEDs

- Reprendre le premier programme
- Enregistrer sur “appareil Micropython” dans un nouveau fichier:
`led1.py`
- Modifier le programme pour allumer puis éteindre après 2s dans l’ordre la LED rouge, puis la verte, la jaune et enfin la bleue
- Enregistrer le programme et lancer l’exécution

Etape 1: allumer et éteindre les LEDs

- Exemple de programme:

```
from pyb import LED, delay

led_red = LED(1)
led_green = LED(2)
led_orange = LED(3)
led_blue = LED(4)

leds = [led_red,led_green,led_orange,led_blue]

for led in leds:
    led.on()
    delay(2000)
    led.off()
```

Import des classes LED et delay du module pyb cf.
<https://docs.micropython.org/en/latest/pyboard/quickref.html>

Création de 4 objets LED

Création d'une liste contenant les 4 objets LED

Pour chaque LED:

- Allumage de la LED
- Attente de 2000ms
- Extinction de la LED

Etape 2: clignotement aléatoire

- Reprendre le programme précédent
- Enregistrer sur “appareil Micropython” dans un nouveau fichier:
`led2.py`
- Modifier le programme pour:
 1. Faire une boucle de 1000 iterations. Dans cette boucle:
 1. Générer un nombre aléatoire (random) entre 1 et 4 qui servira de numéro de LED
 2. Faire “toggler” la LED dont le numéro a été tiré
 3. Attendre 500ms
 2. Eteindre toutes les LEDs
- Enregistrer le programme et lancer l’exécution

Etape 2: clignotement aléatoire

- Import d'un nouveau module nécessaire pour la génération de nombres aléatoires:

```
from pyb import LED, delay  
from random import randint
```

- Utilisation de la fonction de génération de nombres aléatoires:

```
random.randint(a, b)  
Return a random integer  $N$  such that  $a \leq N \leq b$ .  
Alias for randrange( $a, b+1$ ).
```

- Utilisation de la fonction de toggling des LEDs:

```
led.toggle()
```

<https://docs.python.org/3/library/random.html>

<https://docs.micropython.org/en/latest/library/pyb.LED.html#pyb-led>

Etape 2: clignotement aléatoire

- Exemple de programme:

The screenshot shows the Thonny IDE interface. The code editor window contains the following Python script:

```
1 from pyb import LED, delay
2 from random import randint
3
4 led_red = LED(1)
5 led_green = LED(2)
6 led_orange = LED(3)
7 led_blue = LED(4)
8
9 leds = [led_red, led_green, led_orange, led_blue]
10
11 for i in range(1000):
12     num = randint(1,4)
13     leds[num].toggle() ← Index out of range
14     delay(1000)
15
16 [led.off() for led in leds]
17
```

The line `leds[num].toggle()` is highlighted with a red arrow pointing to the error message in the shell window.

The shell window displays the following traceback:

```
Traceback (most recent call last):
  File "<stdin>", line 15, in <module>
NameError: name 'led' is not defined
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 13, in <module>
IndexError: list index out of range
>>>
```

Index out of range

Le programme ne s'exécute pas, un problème a été détecté par l'interpréteur python



Etape 3: debug du programme

- Pour mettre un programme au point, il est utile de visualiser des informations lors de son déroulement. C'est «débugger » le programme.
- Il est possible de rajouter des messages envoyés par la pyBoard à la console de Thonny via l'interface série grâce à la commande python print():
 - `print("text")` affiche text sur la console
 - `print(a,b,c)` affiche le contenu des variables a, b et c les unes à la suite des autres sur une même ligne
 - `print("a: " + str(a) + "\n")` affiche le texte "a: " , puis le contenu de la variable a convertie en string, puis un retour à la ligne

Etape 3: debug du programme

```
Thonny - C:\Users\faarrive\test.py @ 13:19
File Edit View Run Tools Help
D test.py <untitled>
1 from pyb import LED, delay
2 from random import randint
3
4 led_red = LED(1)
5 led_green = LED(2)
6 led_orange = LED(3)
7 led_blue = LED(4)
8
9 leds = [led_red,led_green,led_orange,led_blue]
10
11 for i in range(1000):
12     num = randint(1,4)
13     print("num: "+str(num)+"\n")
14     leds[num].toggle()
15     delay(1000)
16
17 [led.off() for led in leds]
18
```

Shell >>> %Run -c \$EDITOR_CONTENT

```
Traceback (most recent call last):
  File "<stdin>", line 13, in <module>
NameError: name 'tostr' is not defined
```

>>> %Run -c \$EDITOR_CONTENT

```
num: 2
num: 1
num: 4
```

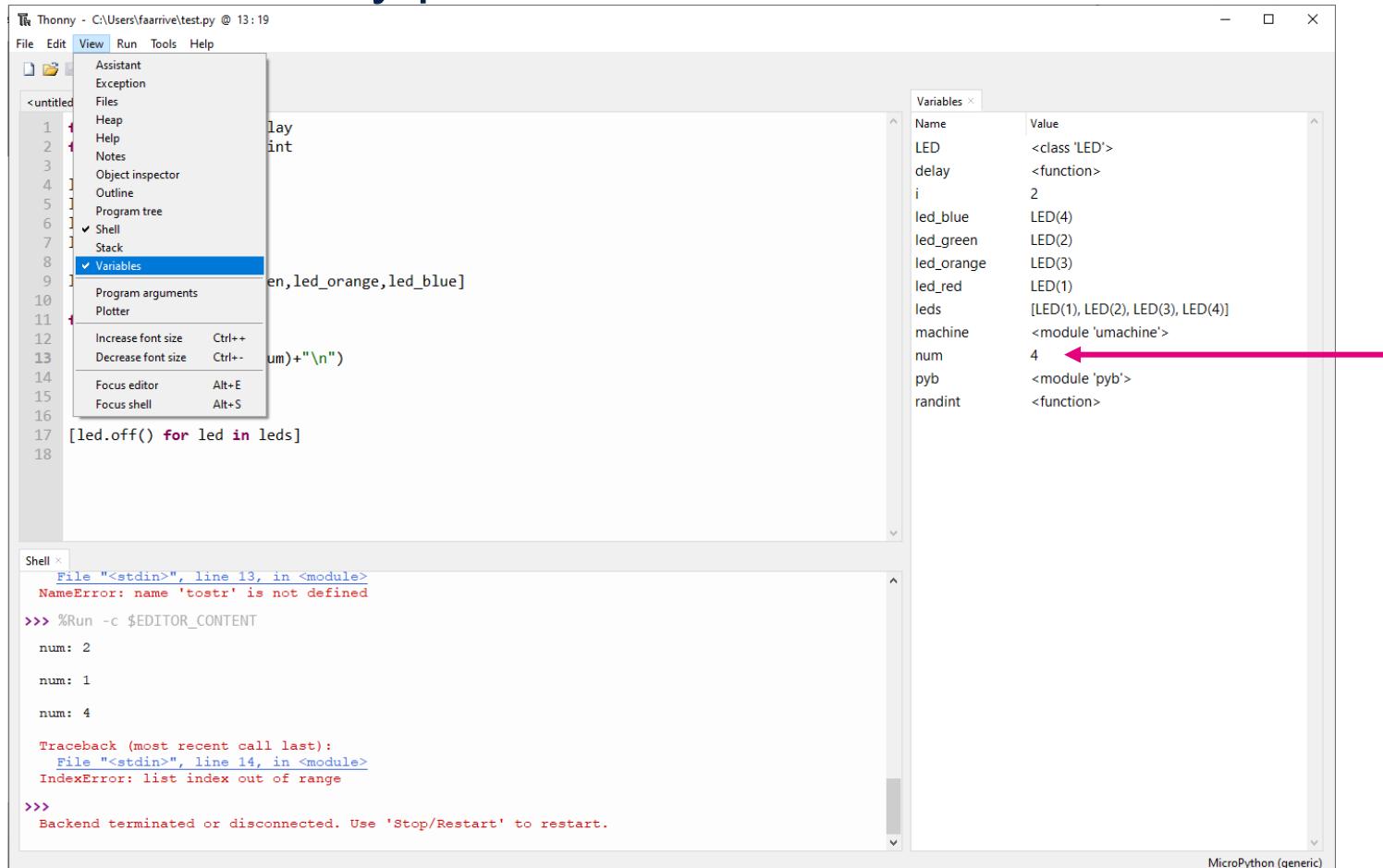
```
Traceback (most recent call last):
  File "<stdin>", line 14, in <module>
IndexError: list index out of range
```

>>>

MicroPython (generic)

Etape 3: debug du programme

- Il est également possible d'ajouter la fenêtre « variables » à l'environnement Thonny pour visualiser leur contenu



Atelier 4 – Le moniteur graphique

- Le but est d'apprendre à utiliser le moniteur graphique, une manière pratique de visualiser l'évolution d'une donnée au cours du temps.

L'accéléromètre intégré

La PyBoard intègre un accéléromètre: puisque ce composant est déjà soudé sur la carte et que l'environnement logiciel le gère déjà, il est prêt à être utilisé.

Répertoire de sources: `demo_moniteur_graphique`

```
from pyb import delay, Accel

# declaration d'un objet accelerometre
acc = Accel()

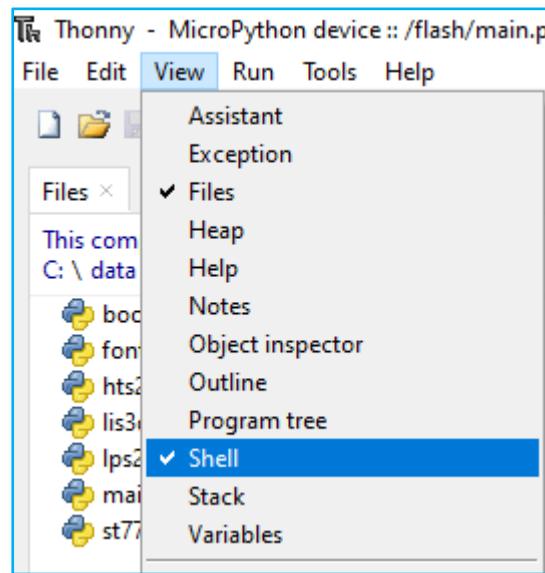
while True:
    # acquisition des valeurs sur les 3 axes
    x,y,z = acc.filtered_xyz()

    # affichage des valeurs
    print("x:%i y:%i z:%i" %(x,y,z))

    delay(200)
```

Moniteur textuel

Comme vu précédemment, la fonction *print* imprime dans la fenêtre *Shell*(moniteur textuel) les valeurs des axes de l'accéléromètre. Ces valeurs sont affichées 5 fois par seconde ce qui permet de voir que ces valeurs changent lorsque l'on incline la PyBoard.

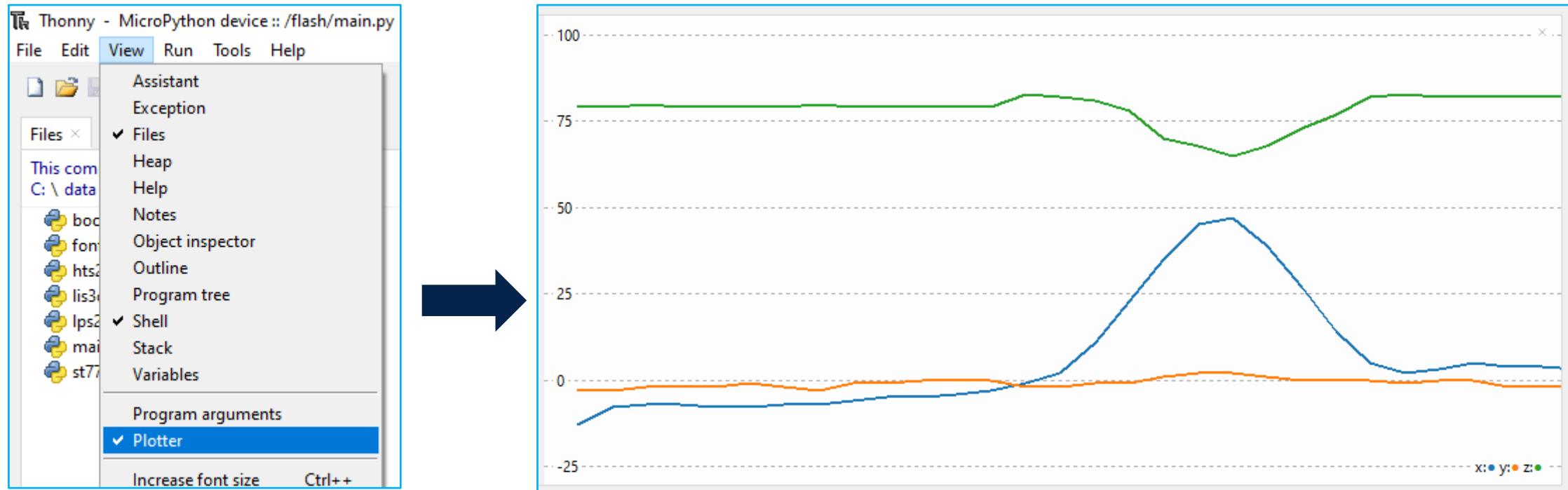
A screenshot of the Thonny IDE Shell window titled "Shell". It displays MicroPython v1.17 running on a STM32F405RG board. The shell prompt shows ">>> %Run -c \$EDITOR_CONTENT". Below the prompt, numerous lines of text are printed, representing the values of the x, y, and z axes of an accelerometer. The data shows periodic fluctuations, typical of an object being tilted.

```
MicroPython v1.17 on 2021-09-02; PYBv1.1 with STM32F405RG
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

x:-13 y:-3 z:79
x:-8 y:-3 z:79
x:-7 y:-2 z:80
x:-7 y:-2 z:79
x:-8 y:-2 z:79
x:-8 y:-1 z:79
x:-7 y:-2 z:79
x:-7 y:-3 z:80
x:-6 y:-1 z:79
x:-5 y:-1 z:79
x:-5 y:0 z:79
x:-4 y:0 z:79
x:-3 y:0 z:79
x:-1 y:-2 z:83
x:2 y:-2 z:82
x:11 y:-1 z:81
x:23 y:-1 z:78
x:35 y:1 z:70
x:45 y:2 z:68
```

Moniteur graphique

Thonny intègre également un moniteur graphique (*plotter*), qui permet de générer une ou plusieurs courbes en utilisant les données du moniteur textuel: la visualisation est beaucoup plus simple qu'avec des valeurs numériques.

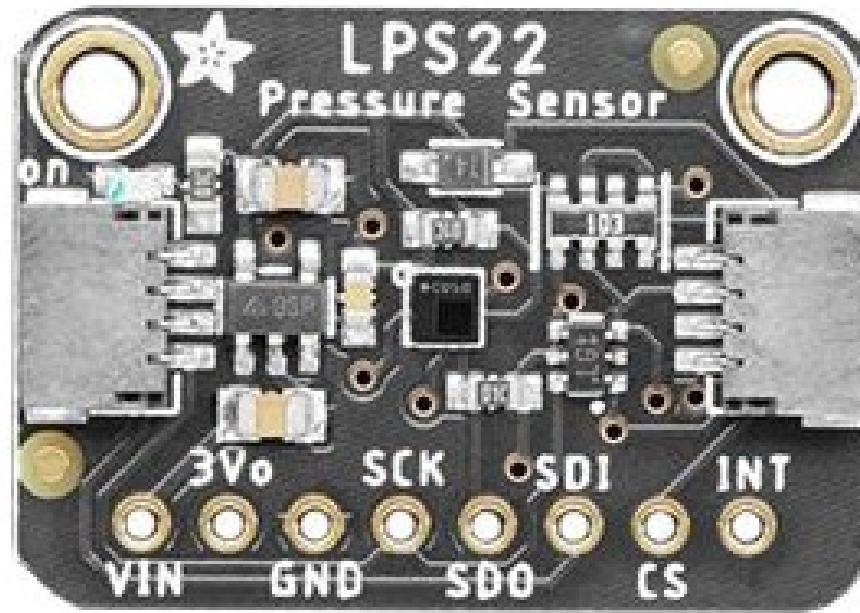


Atelier 5 – Le capteur pression atmosphérique LPS22

- Le but est de connecter la carte des capteurs (AdaFruit LPS22) à la PyBoard et d'afficher sur l'écran du PC la température et/ou pression
 - Etape 1: connecter physiquement les deux cartes PyBoard et LPS22
 - Etape 2: lire les informations du capteur
 - Etape 3: visualiser les données

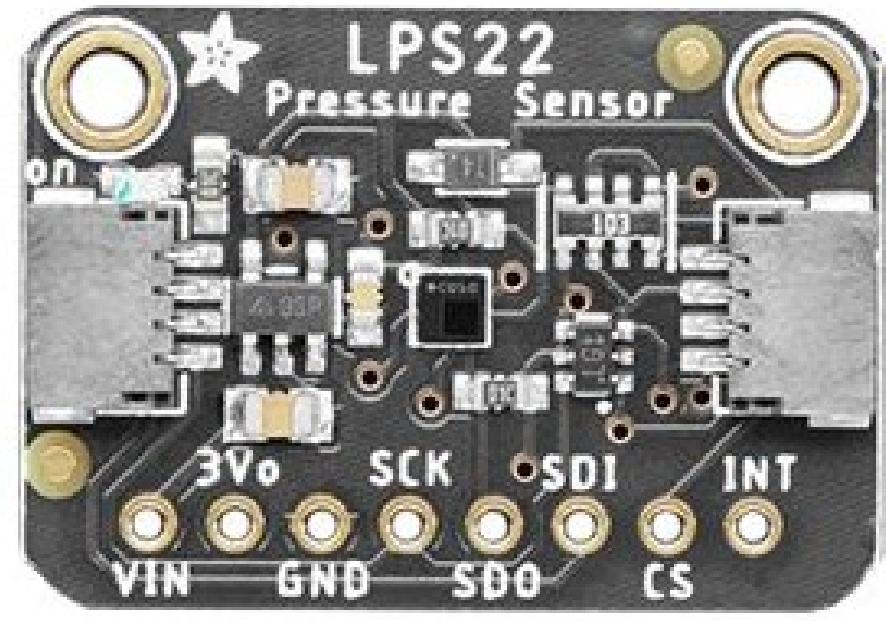
Etape 1: Connecter les cartes

La page principale du produit est disponible ici: <https://learn.adafruit.com/adafruit-lps25-pressure-sensor>.

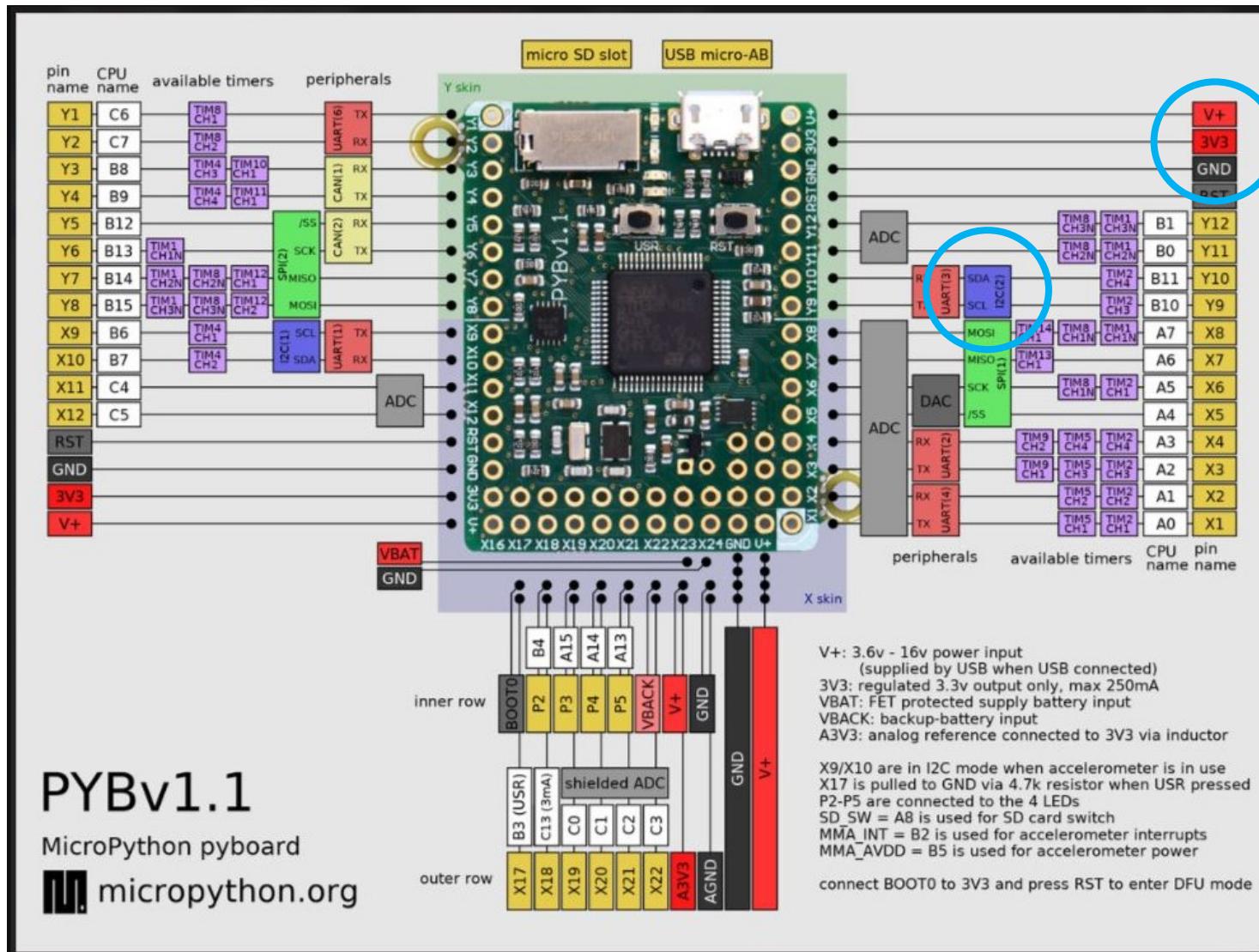


Etape 1: Connecter les cartes

- La description complète des plots de connexion est ici:
<https://learn.adafruit.com/adafruit-lps25-pressure-sensor/pinouts>.
- L'alimentation électrique se fait grâce aux plots VIN et GND: grâce au régulateur intégré, nous pouvons utiliser du 5V ou du 3.3V.
- La carte peut se connecter soit en SPI soit en I2C; pour ce projet, nous utiliserons l'I2C (plots SCK(horloge) et SDI(données)) en conservant l'adresse par défaut 0x5D (plot SDO non connecté)
- Les autres plots (3Vo, CS et INT) n'ont pas besoin d'être connectés.



Etape 1: Connecter les cartes

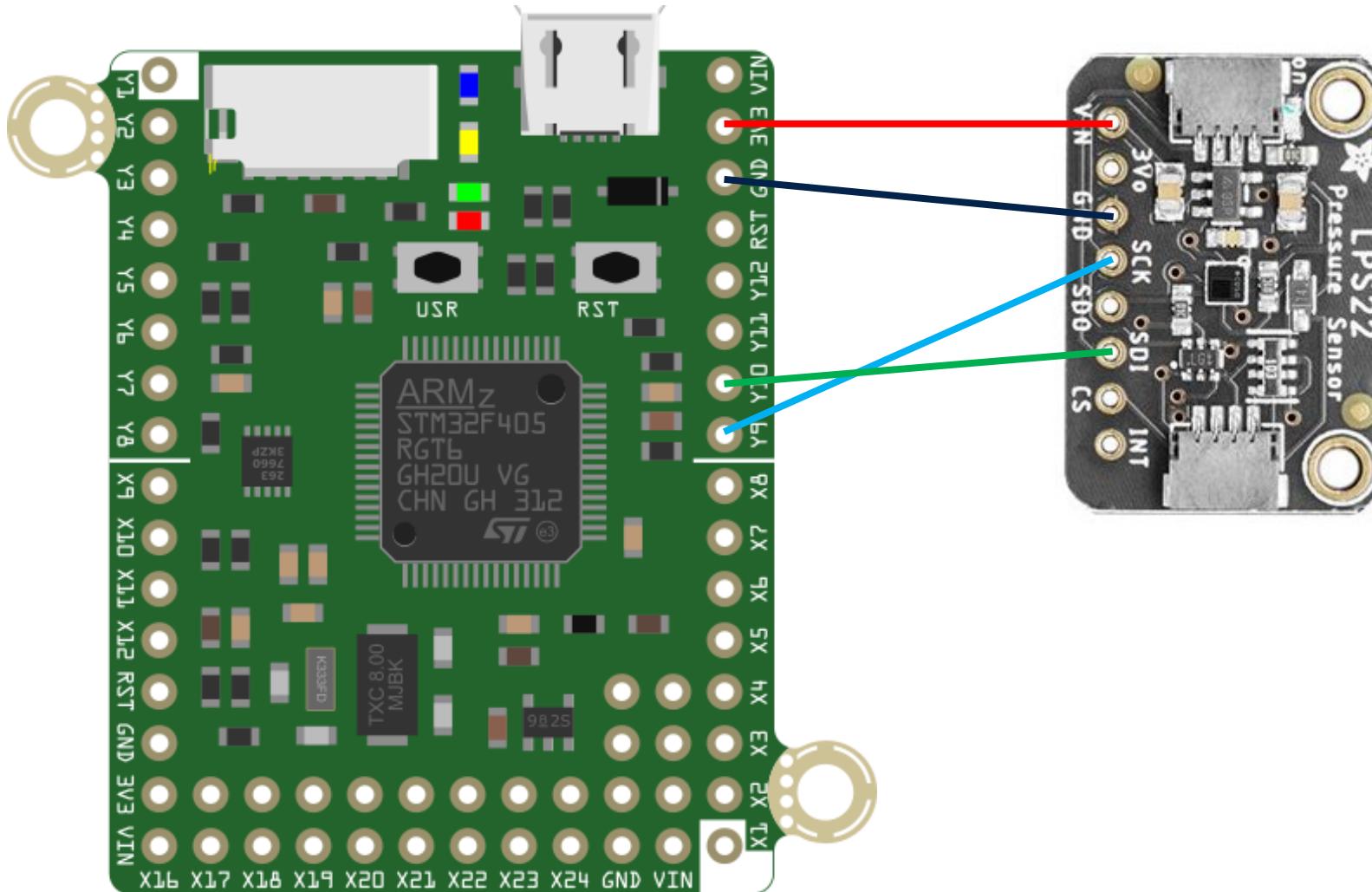


2 ports I2C sont disponible sur la PyBoard, nous utiliserons le second I2C(2) pour connecter la carte capteur LPS22:

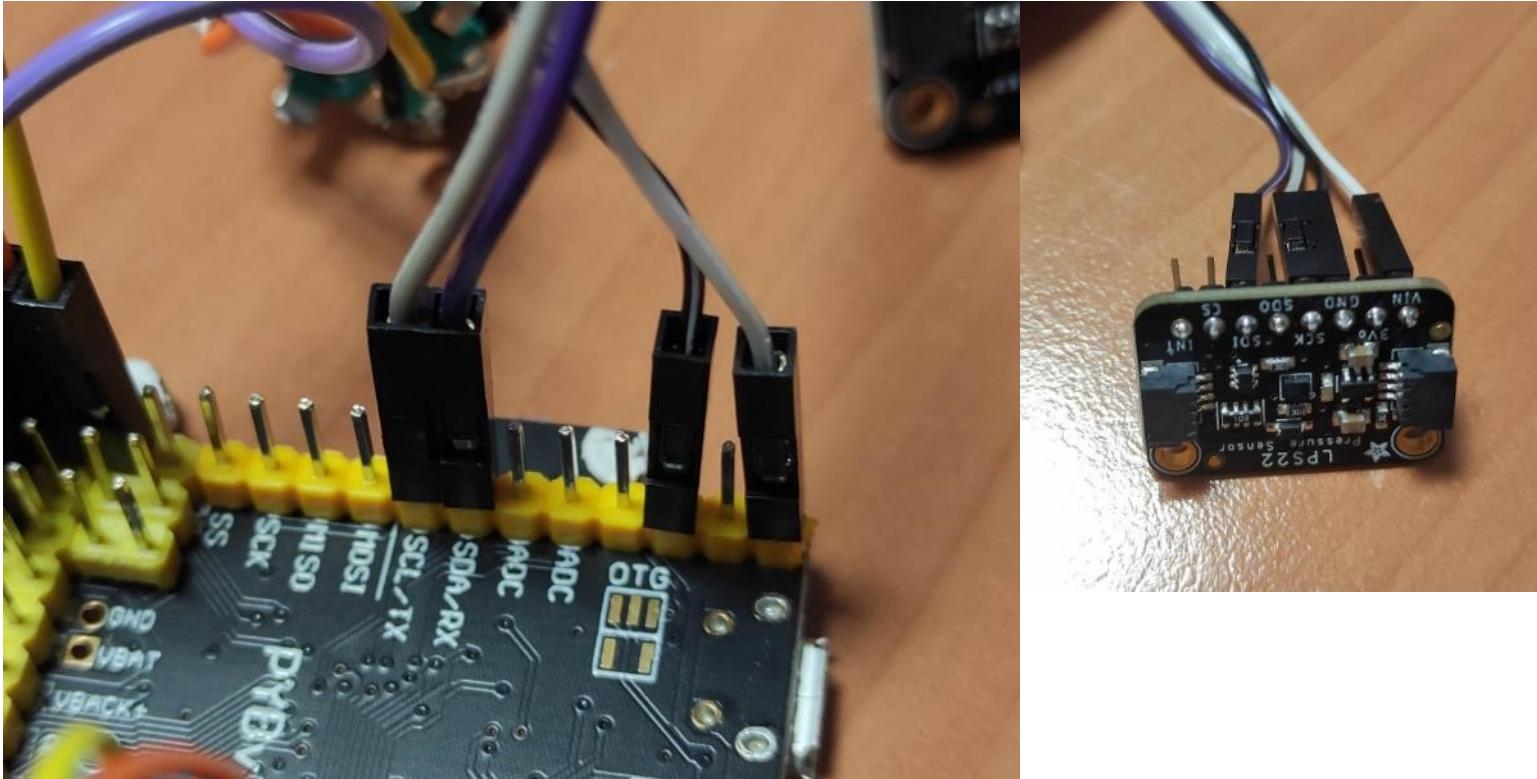
- I2C(2) / SDA (données) => plot Y10 → SDI
 - I2C(2) / SCL (horloge) => plot Y9 → SCK

Pour des raisons pratiques nous utiliserons les plots d'alimentation les plus proches: 3V3 et GND

Etape 1: Connecter les cartes



Etape 1: Connecter les cartes



Etape 2: Lire les informations du capteur

Il existe déjà un module python (LPS22.py) pour communiquer avec le capteur LPS22: ce module utilise le protocole I2C et tente de communiquer avec le capteur à l'adresse 0x5D (adresse par défaut du capteur).

Extrait du module LPS22:

```
class LPS22():
    def __init__(self, i2c, addr = 0x5D):
        self.i2c = i2c
        self.addr = addr
```

Etape 2: Lire les informations du capteur

Répertoire de sources: demo_LPS22

```
from machine import I2C
from pyb import delay
import LPS22

# Initialisation du port I2C(2)
i2c = I2C(2)

# Initialisation du capteur LPS22 via l'I2C
lps = LPS22.LPS22(i2c)

# affiche indefiniement la température et la pression toute les
# demi-secondes
while(True):
    # la fonction get du module lps récupère la température et
    # la pression
    temp, press = lps.get()
    print("temperature: {}, pression: {}".format(temp, press))
    delay(500)
```

Puisque nous avons connecté la carte capteur sur le second port I2C, nous devons l'indiquer à au module LPS22:

- créer un objet I2C (sur le port 2)
- Donner en paramètre cet objet I2C pour la création de l'objet LPS22

Le module LPS22 propose plein de fonctions (aller voir le source LPS22.py), nous utiliserons la fonction *get* qui permet de récupérer la température et la pression.

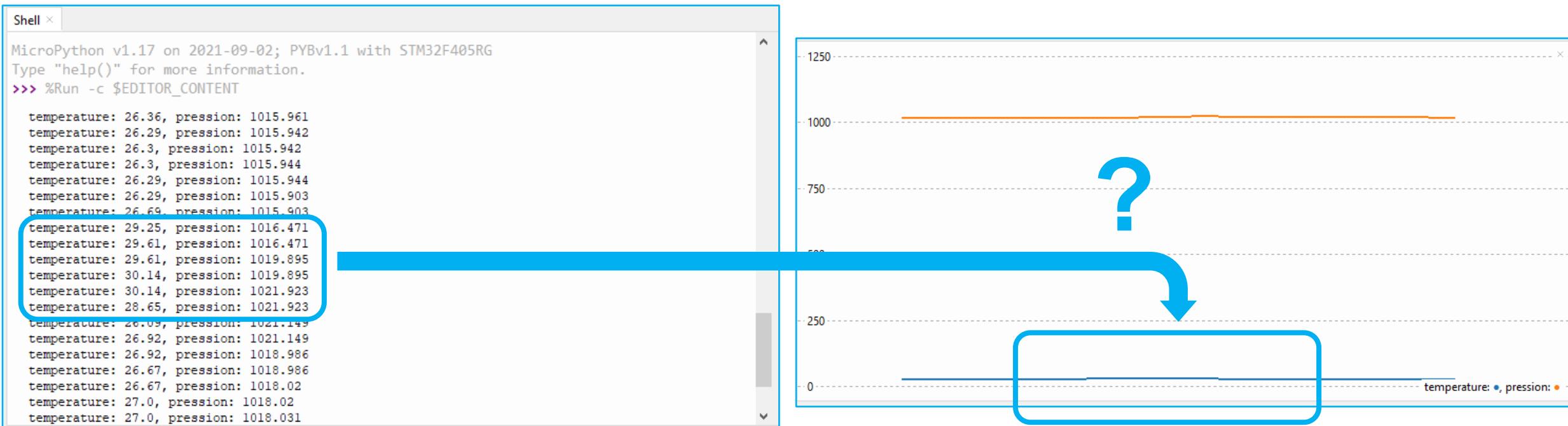
Etape 3: visualiser les données

Voici les données observées avec les moniteurs textuels et graphiques:



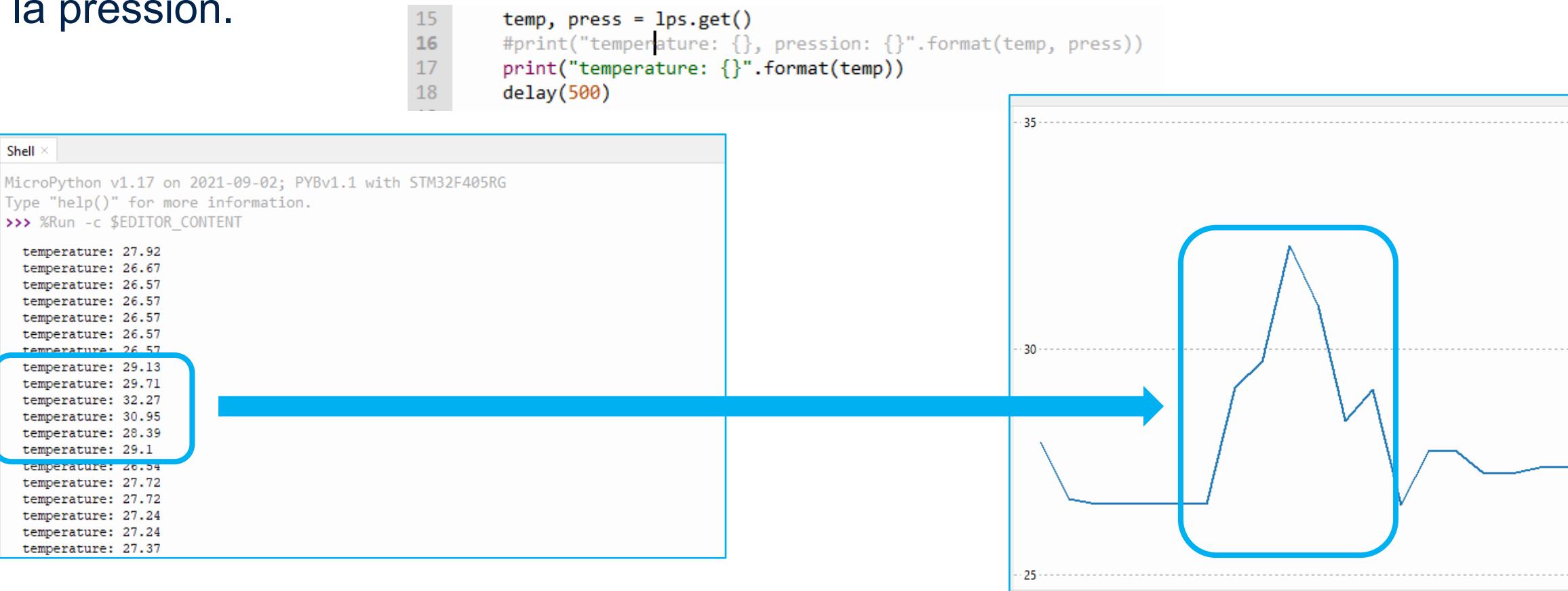
Etape 3: visualiser les données

Ici, le moniteur graphique pose problème car les valeurs de température et de pression étant très éloignées l'une de l'autre ($\sim 25 \leftrightarrow \sim 1000$), l'évolution de la température n'est pas visible (l'échelle imposée par la pression est trop grande).



Etape 3: visualiser les données

Afin de pouvoir visualiser le changement de température, nous n'allons plus afficher la pression.



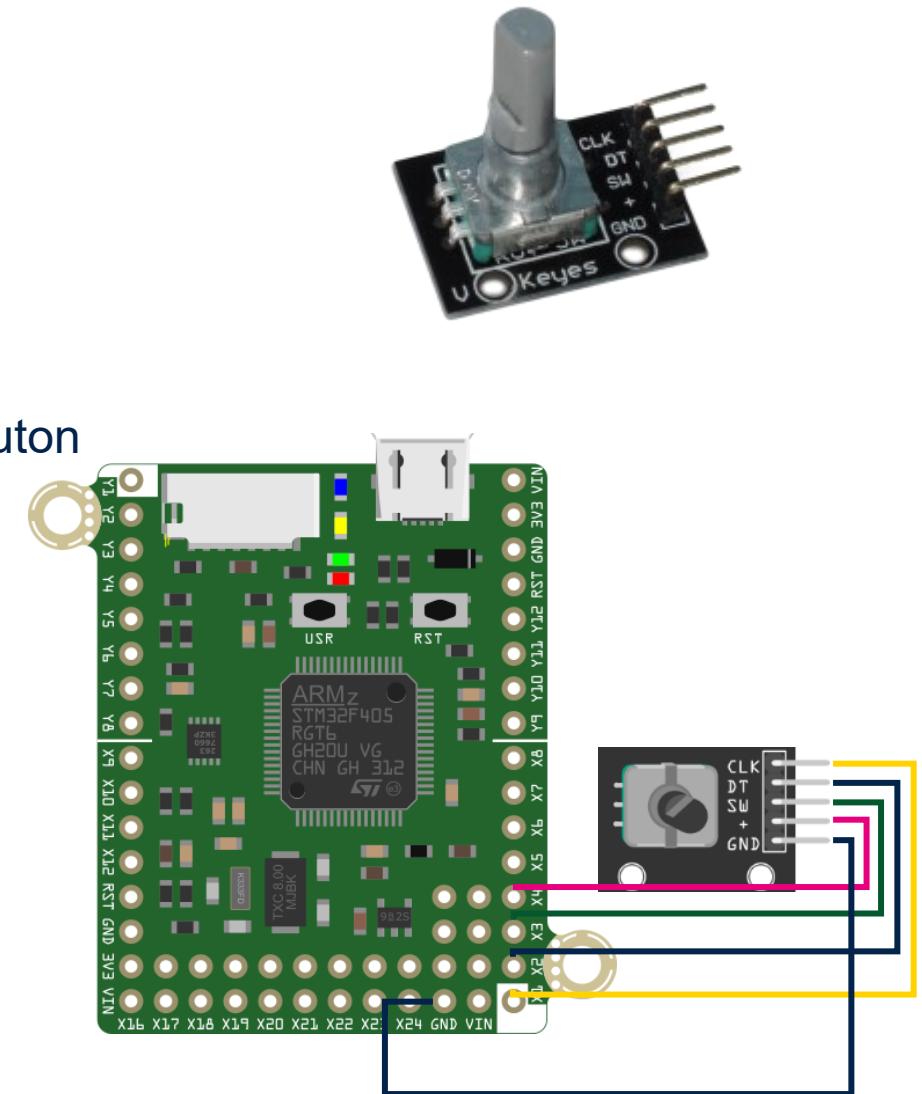
Atelier 6 – Le bouton molette

- Le but est de connecter le bouton molette à la carte pyboard et observer l'effet de l'action du bouton

Etape 1: connecter le bouton molette

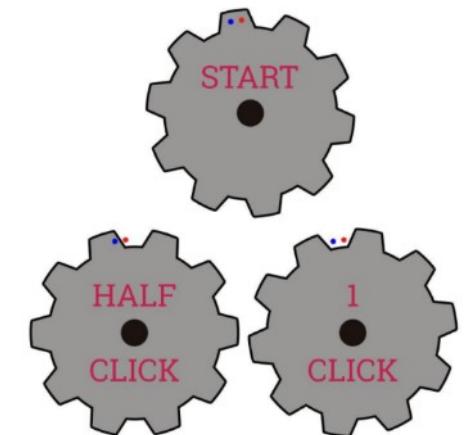
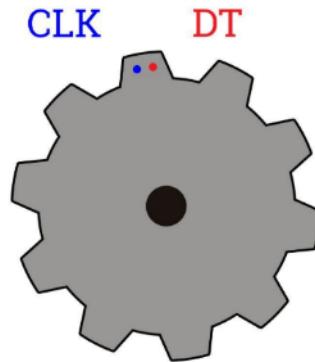
- Connexion de la pyboard avec le bouton molette

- Le bouton molette comporte 5 connections
 - CLK (Encoder Pin X): détection de la rotation
 - DT (Encoder Pin Y): détection de la rotation
 - SW (Pushbutton Switch): détection de la pression sur le bouton
 - +(Supply): alimentation (5V ou 3V3)
 - GND: alimentation (masse)
- Ces connections sont à relier à la carte comme suit:
 - GND <=> GND
 - VIN X4 <=> + 
 - X1 <=> CLK
 - X2 <=> DT
 - X3 <=> SW

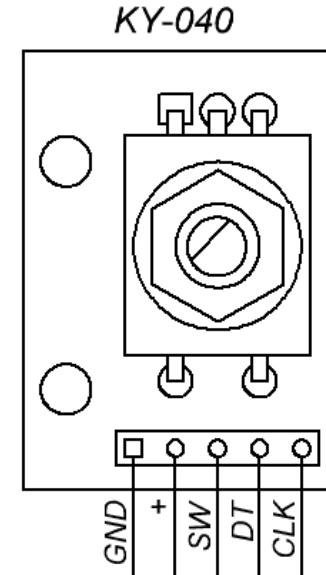
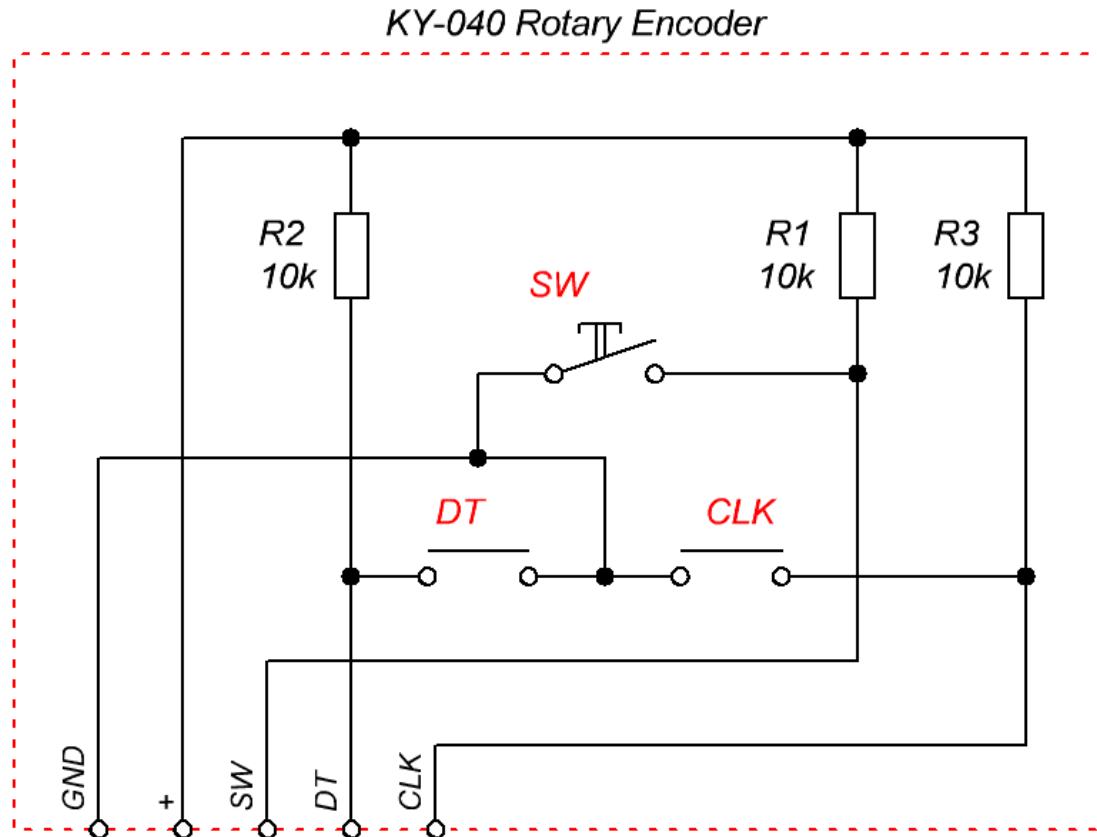


Principe de fonctionnement

- Deux signaux (CLK et DT ou X et Y) détectent si ils sont chacun devant un creux (valeur 0) ou une dent (valeur 1) de la molette
- Au démarrage, ils sont tous deux par exemple devant une dent: $X=1, Y=1$
- Quand on tourne, un des signaux arrivera en premier devant le creux et passera à 0. Selon que c'est X ou Y qui passe devant le creux en premier, on saura dans quel sens on tourne
- Quand on continue à tourner, le second signal passe à son tour devant le creux. Sa valeur passe à 0
- Puis, le premier signal passe devant la prochaine dent (1) et enfin le second signal également (1)

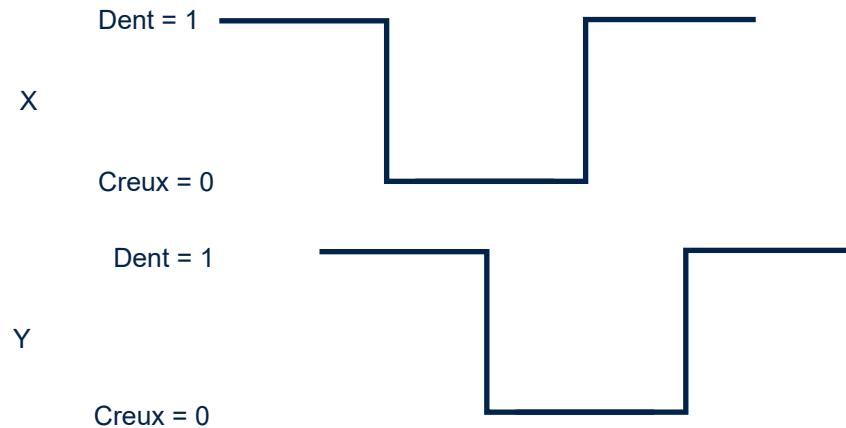


Roue codeuse: schéma électrique

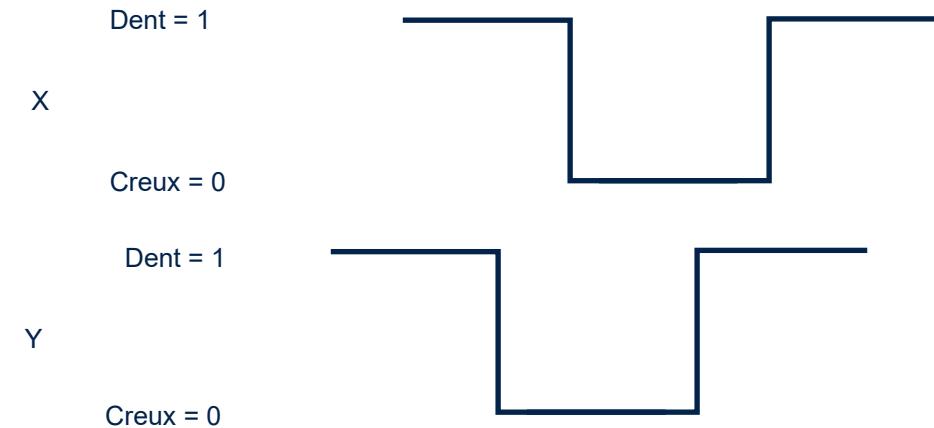


Principe de fonctionnement

- Le but est de passer par toutes les montées et descentes de X et Y pour compter un cran



Dans un sens

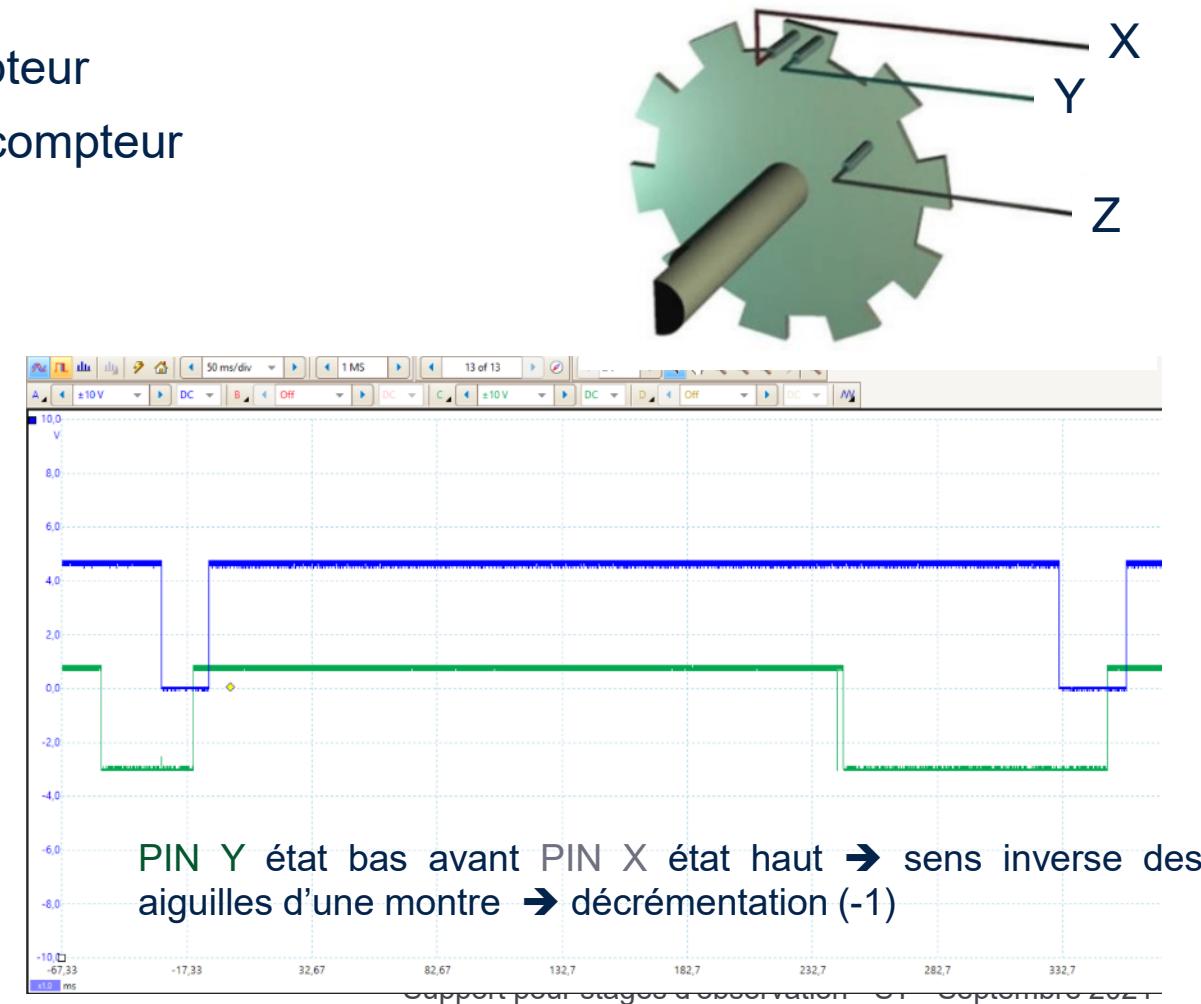


Dans l'autre sens

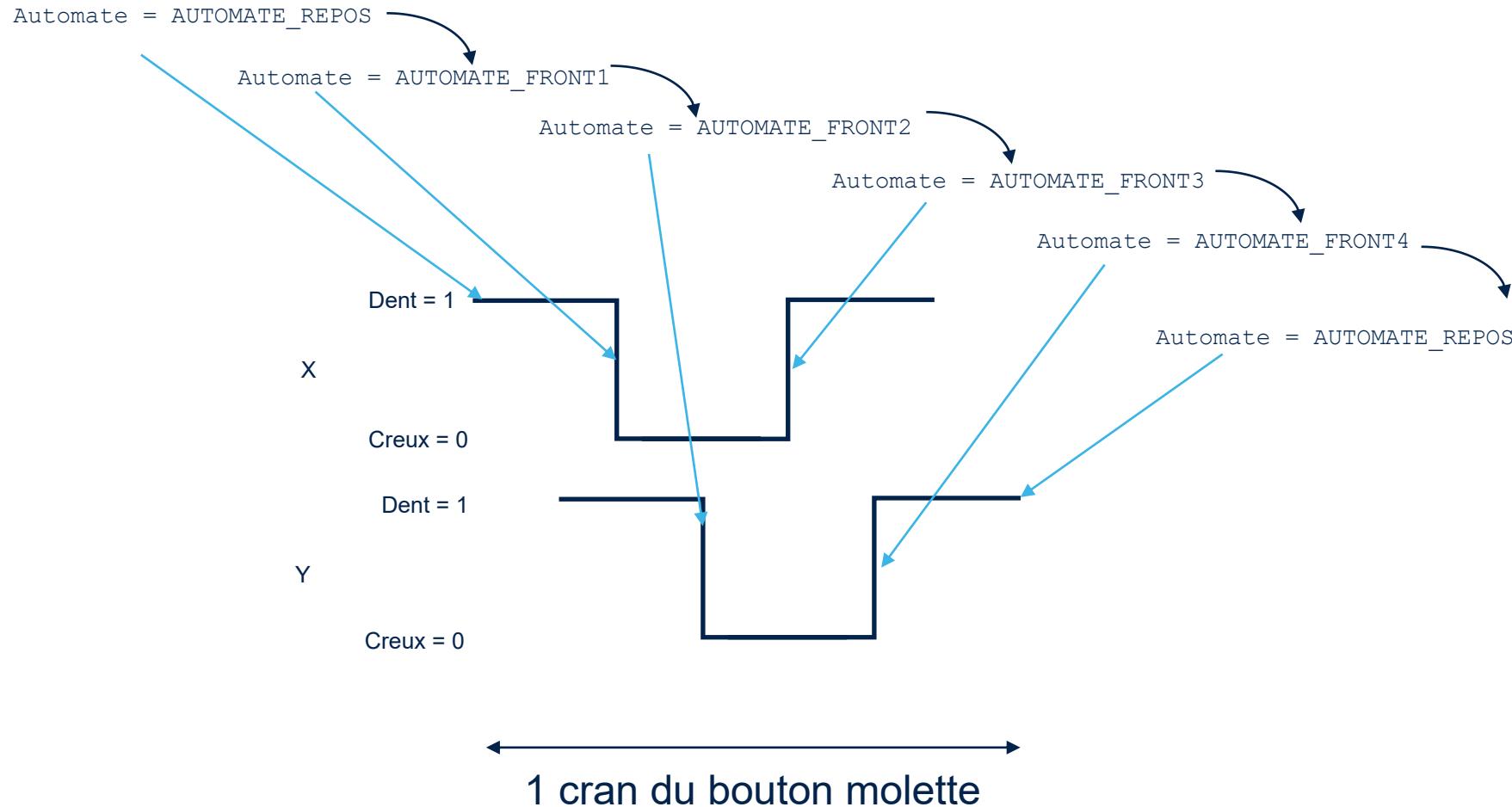
Principe de fonctionnement

- Deux signaux X et Y utilisés

- Une rotation dans un sens incrémente un compteur
- Une rotation dans l'autre sens décrémente un compteur
- Visualisation sur l'oscilloscope:



Principe du programme (version simple)



Programme d'exemple

```
# main.py -- put your code here!
from machine import Pin
from pyb import delay

pin_vpp = Pin('X4',Pin.OUT)
pin_vpp.value(1)
pin_x = Pin('X1')
pin_y = Pin('X2')
pin_key = Pin('X3')

cur_x = last_x = pin_x.value()
cur_y = last_y = pin_y.value()

# Machine à 5 états (0, 1, 2, 3, 4) pour passer sur les 4 fronts descendants/montants de X et Y
automaton = 0

#X --- --3---
# 0 |_1_____|

#Y ----- --4--
#          |_2_____|

#X ----- --14--
#          |_12_____|

#Y --- -13---
# 0 |_11_____|

# Indication de sens (1, -1) ou 0 : pas démarré
forward = 0
# Position courante
position = 0

while True:

    cur_x = pin_x.value()
    cur_y = pin_y.value()

    if (last_x != cur_x):

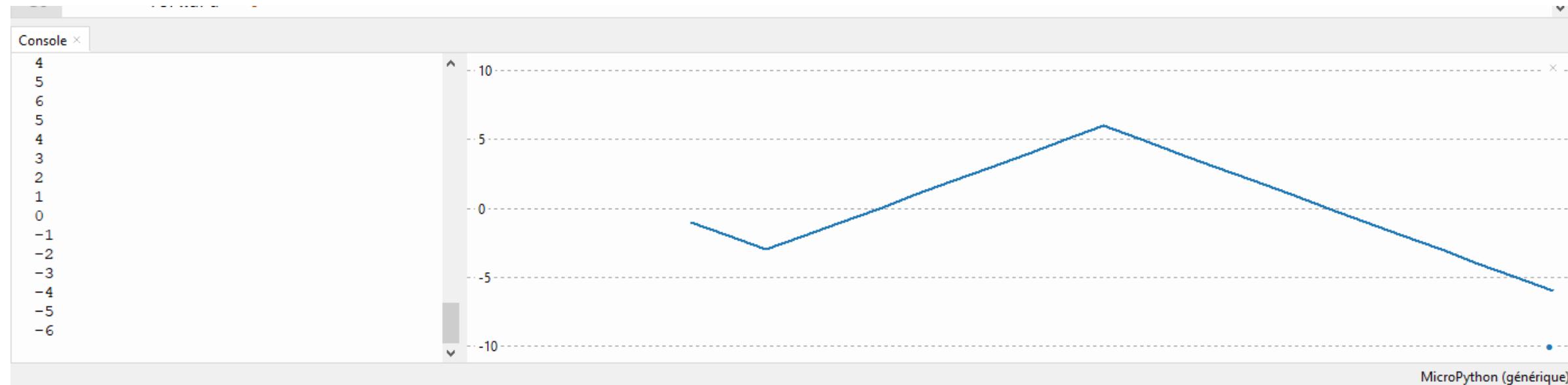
        # x a changé
        last_x = cur_x
        if cur_x == 0:
            # Front descendant de x
            if automaton == 0:
                # On n'avait pas encore démarré: on démarre dans le sens positif
                forward = 1
                automaton = 1
            elif automaton == 11:
                automaton = 12
        else:
            # Front montant de x
            if automaton == 2:
                automaton = 3
            elif automaton == 13:
                automaton = 14

        if (last_y != cur_y):
            # y a changé
            last_y = cur_y
            if cur_y == 0:
                # Front descendant de y
                if automaton == 0:
                    # On n'avait pas encore démarré: on démarre dans le sens négatif
                    forward = -1
                    automaton = 11
                elif automaton == 1:
                    automaton = 2
            else:
                # Front montant de y
                if automaton == 3:
                    automaton = 4
                elif automaton == 12:
                    automaton = 13

        if automaton == 4 or automaton == 14:
            position += forward
            print(position)
            automaton = 0
            forward = 0
```

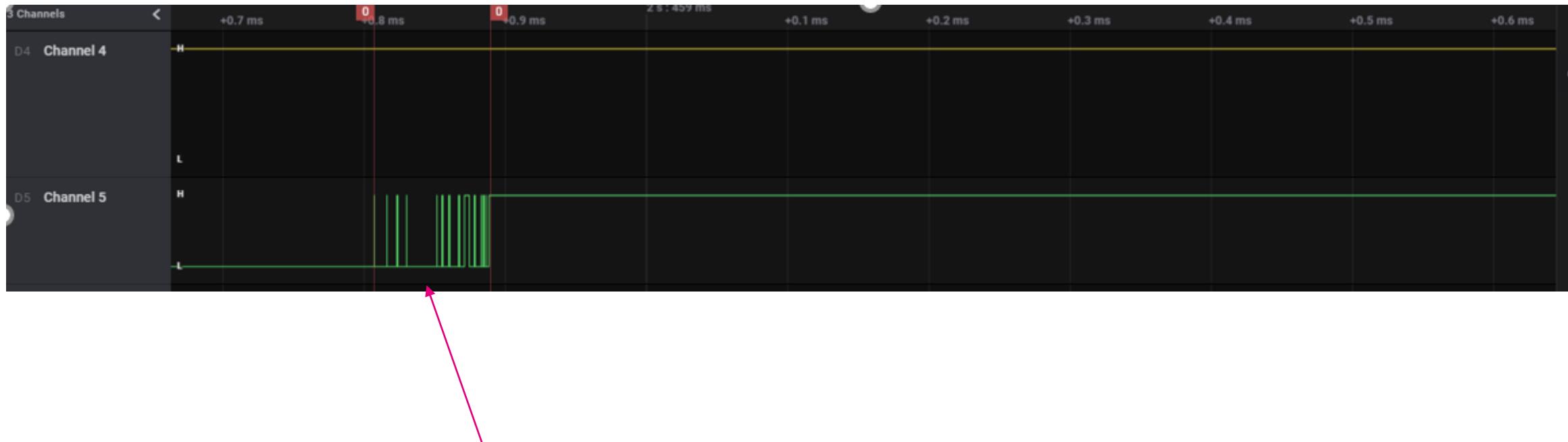
Vérification du fonctionnement

- Lancer le programme d'exemple avec « Exécuter le script courant »
- La *console* et le *grapheur* affichent la position de la molette à chaque changement de position



Problème des rebonds

- De temps en temps on peut voir des ratés sur les incrémentations ou décrémentations du compteur. C'est à cause des rebonds des contacts mécaniques.



Rebonds sur le signal Y lors du passage de 0 à 1

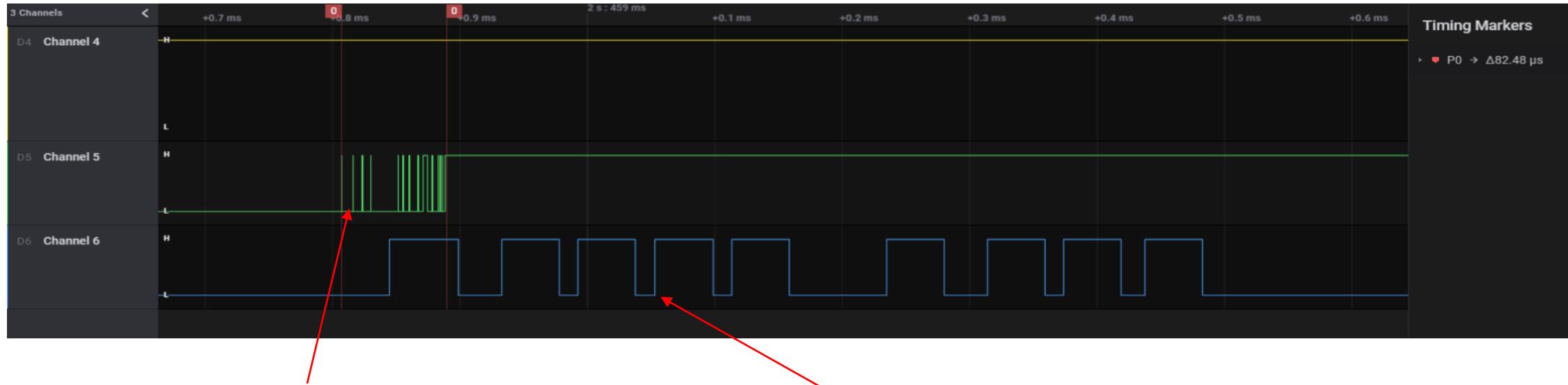
Problème de scrutation

- Avec le programme actuel, les niveaux doivent constamment être scrutés pour permettre l'avancement de l'automate.
- Si d'autres activités doivent se faire en parallèle, on risque de rater des états. La solution est d'utiliser une gestion par interruptions. L'interruption interrompt le programme courant suite à un évènement comme le front montant ou descendant d'un signal sur une entrée du microprocesseur. Sur la pyboard, l'interruption appelle une fonction « callback » associée à l'interruption et retourne au programme courant.
- `Pin('X1').irq(trigger=Pin.IRQ_FALLING|Pin.IRQ_RISING, handler=Callback)`

Déclenchement sur front montant et descendant

Callback qui sera appelée à chaque trigger

- La callback n'est pas appelée sous interruption mais plus tard

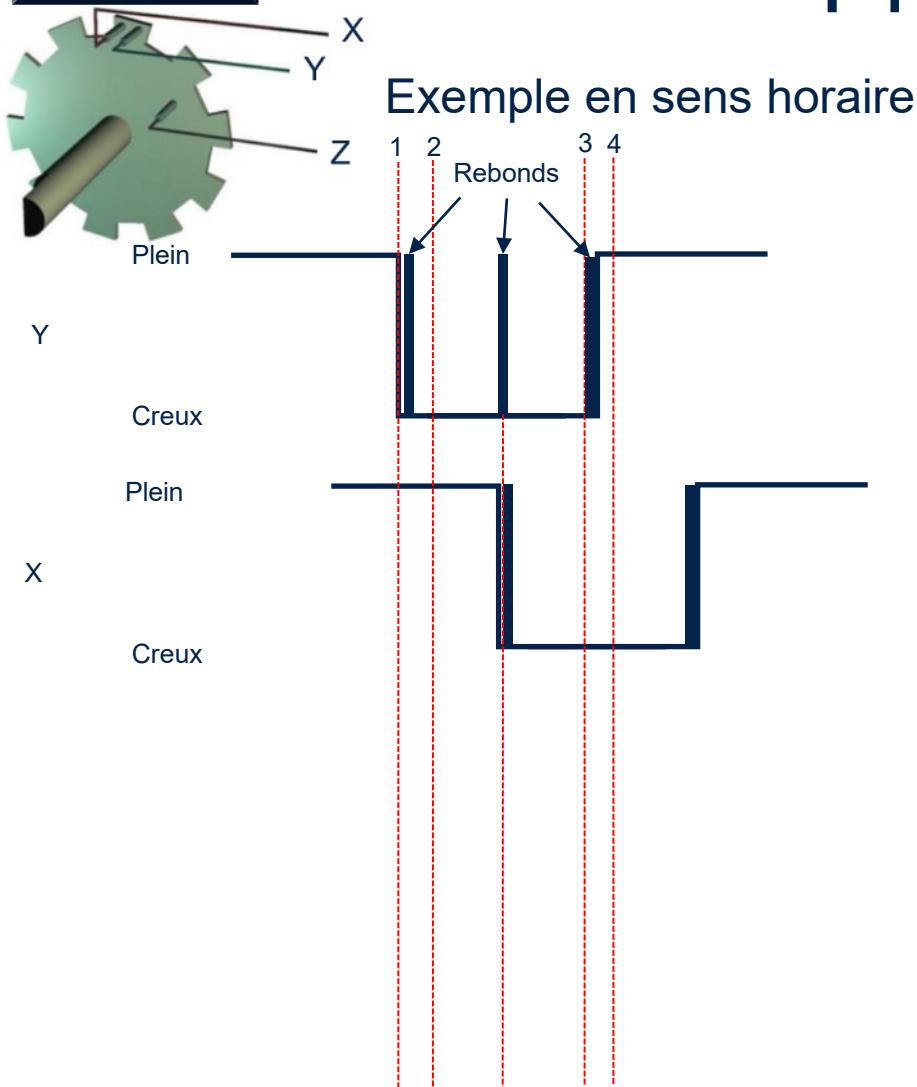


Interruption générée sur
chaque front montant ou
descendant

Callback associée à
chaque interruption

- Les printf sont très utiles pour le debug d'un programme
- Pour ne pas interférer sur le temps réel, on peut utiliser des GPIOs pour indiquer par exemple l'entrée ou la sortie d'une callback
- `debug_c = Pin(Pin('X4'), Pin.OUT, Pin.PULL_DOWN)` ← GPIO X4 déclaré en sortie avec pulldown
- `debug_c(1)` ← Passage de la pin à l'état 1
- `debug_c(0)` ← Passage de la pin à l'état 0
- Le signal bleu de page précédente a permis de tracer l'entrée et sortie de la callback grâce à l'ajout de `debug_c(1)` au début de la callback et de `debug_c(0)` à la fin de la callback

Principe (version 1 avec interruptions)



- 1: front descendant sur Y: `y_callback` est appelée. On memorise le niveau de X (ici 1) et on arme un timer pour filtrer les rebonds sur Y
- 2 : expiration du timer avant de revenir à l'état initial
- 3 : front montant sur Y: `y_callback` est appelée. On mémorise le niveau de X (ici 0) et on arme un timer pour filtrer les rebonds sur Y
- 4 : expiration du timer avant de revenir à l'état initial

```
def timer_callback(self, timer):
    # On a terminé le filtrage. Arret du timer
    timer.deinit()
    self.state = 0

    # Si Y est à l'état bas et que le dernier niveau de Y etait à 1
    if self.pin_y.value() == 0:

        if self.lastlevel == 1:
            # L'état de X quand Y a changé de valeur permet de déterminer le sens positif ou négatif
            self.forward = self.x_level ^ 1
            # On indique qu'on vient de franchir un cran
            self.position += 1 if self.forward else -1

            self.lastlevel = 0
        else:
            self.lastlevel = 1

def y_callback(self, line):
    # On vient de recevoir un front descendant ou montant sur le signal Y
    # Si on n'a pas démarré le timer de filtrage
    if self.state == 0:
        # On mémorise l'état du signal X
        self.x_level = self.pin_x.value()
        # lancement du timer de 5ms pour filtrer les rebonds
        Timer(4, period=5, callback=self.timer_callback)
        # On indique qu'on a démarré le timer pour filtrer tous les prochains fronts
        self.state = 1
```

Programme d'exemple

```

from machine import Pin
from pyb import Timer

class Encoder:
    def __init__(self, pin_x, pin_y, pin_key=None, key_cb=None):
        self.pin_x = pin_x
        self.pin_y = pin_y
        self.position = 0
        self.key_cb = key_cb
        self.state = 0
        self.x_level = 0
        self.lastlevel = 1

        # Interruption Y: sur front montant et descendant
        self.y_interrupt = pin_y.irq(trigger=Pin.IRQ_FALLING | Pin.IRQ_RISING,
                                     handler=self.y_callback)

        # Interruption sur l'appui du bouton
        if pin_key and key_cb:
            self.k_interrupt = pin_key.irq(trigger = Pin.IRQ_FALLING, handler=self.k_callback)

    def timer_callback(self, timer):
        # On a terminé le filtrage. Arret du timer
        timer.deinit()
        self.state = 0

        # Si Y est à l'état bas et que le dernier niveau de Y etait à 1
        if self.pin_y.value() == 0:
            if self.lastlevel == 1:
                # L'etat de X quand Y a changé de valeur permet de determiner le sens positif ou
                négatif
                self.position += 1 if self.x_level == 0 else -1

                self.lastlevel = 0
            else:
                self.lastlevel = 1

    def y_callback(self, line):
        # On vient de recevoir un front descendant ou montant sur le signal Y
        # Si on n'a pas démarré le timer de filtrage
        if self.state == 0:
            # On mémorise l'état du signal X
            self.x_level = self.pin_x.value()
            # lancement du timer de 5ms pour filtrer les rebonds
            Timer(4, period=5, callback=self.timer_callback)
            # On indique qu'on a démarré le timer pour filtrer tous les prochains fronts
            self.state = 1

            def k_callback(self, line):
                if self.key_cb:
                    self.key_cb(line)

            def myCallBack(p):
                s.position = 0

                pin_vpp = Pin('X4',Pin.OUT)
                pin_vpp.value(1)

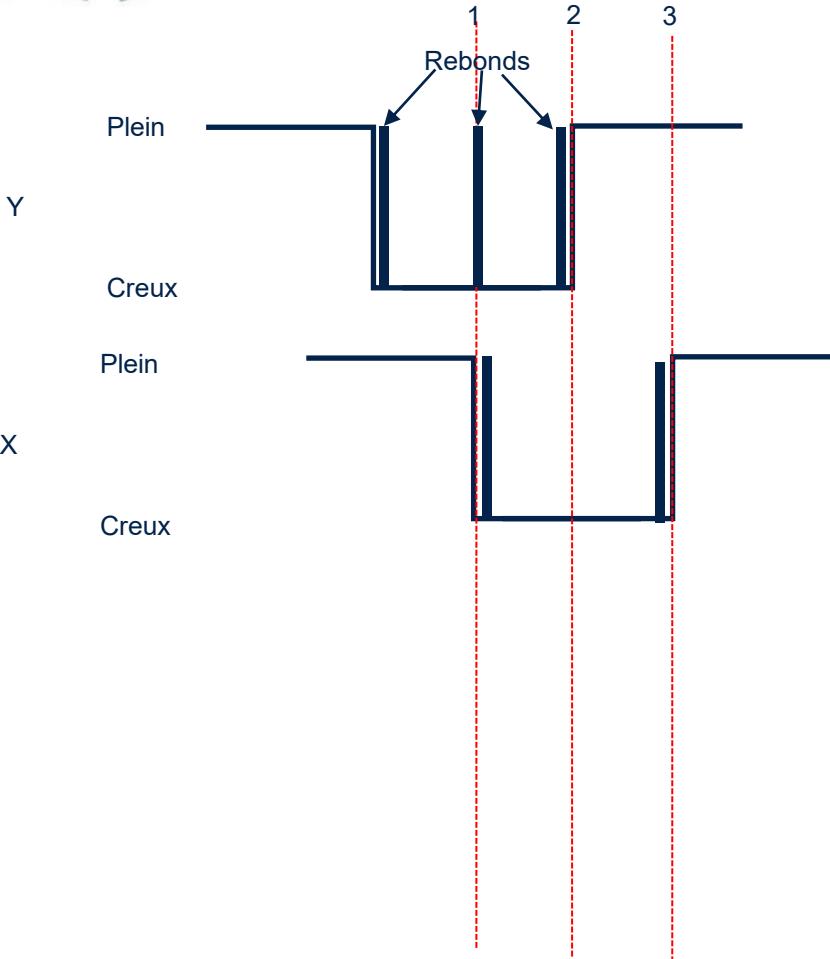
                s = Encoder(Pin('X1'),Pin('X2'), # ==> signaux roue codeuse
                            Pin('X3'),myCallBack) # ==> signal bouton et sa fonction
                pos = -1
                while True:
                    if (s.position != pos):
                        pos = s.position
                        print(s.position)

```



Principe (version 2 avec interruptions)

Exemple en sens horaire



- 1: front descendant sur X: `x_callback` est appelée. Le niveau de Y détermine l'état suivant 1 ou 2 (ici l'état suivant sera 2)
- 2: front montant sur Y: `y_callback` est appelée. Si Y vaut 1 et X vaut 0 on passe à l'état 3
- 3: front montant sur X: `x_callback` est appelée. Si on est dans l'état 3 on met à jour le compteur et on reinitialise l'état à 0

```

def x_Callback(self, line):
    # Démarrage de la machine d'état
    if self.state == 0:
        if self.pin_x.value() == 0:
            # y = 0 => next state = 2
            # y = 1 => next state = 1
            self.state = 2 - self.pin_y.value()
    # Fin de la machine d'état pour le cas incrément (falling edge x puis rising edge y et rising edge x)
    elif self.state == 3:
        if self.pin_x.value() == 1 and self.pin_y.value() == 1:
            self.cnt_update(-1)

def y_Callback(self, line):
    # Fin de la machine d'état pour le cas décrément (falling edge x puis rising edge x et rising edge y)
    if self.state == 1:
        if self.pin_x.value() == 1 and self.pin_y.value() == 1:
            self.cnt_update(+1)
    # Front montant de y pour le cas incrément
    elif self.state == 2:
        if self.pin_x.value() == 0 and self.pin_y.value() == 1:
            self.state = 3

```

Programme d'exemple

```

from machine import Pin

class Encoder:
    def __init__(self, pin_x, pin_y, user_cb=None, pin_key=None, key_cb=None):
        self.pin_x = pin_x
        self.pin_y = pin_y
        self.user_cb = user_cb
        self.key_cb = key_cb
        self.state = 0
        self.value = 0
        self.x_Irq = pin_x.irq(trigger=Pin.IRQ_FALLING|Pin.IRQ_RISING, handler=self.x_Callback)
        self.y_Irq = pin_y.irq(trigger=Pin.IRQ_RISING, handler=self.y_Callback)

        if pin_key and key_cb:
            self.k_interrupt = pin_key.irq(trigger = Pin.IRQ_FALLING, handler=self.k_callback)

    # Mise à jour du compteur en fin de machine d'état + reset de la machine d'état
    def cnt_update(self,inc):
        self.state = 0
        self.value += inc
        if self.user_cb: self.user_cb(self.value)

    def x_Callback(self, line):
        # Démarrage de la machine d'état
        if self.state == 0:
            if self.pin_x.value() == 0:
                # y = 0 => next state = 2
                # y = 1 => next state = 1
                self.state = 2 - self.pin_y.value()
        # Fin de la machine d'état pour le cas incrément (falling edge x puis rising edge y et
        # rising edge x)
        elif self.state == 3:
            if self.pin_x.value() == 1 and self.pin_y.value() == 1:
                self.cnt_update(-1)

    def y_Callback(self, line):
        # Fin de la machine d'état pour le cas décrément (falling edge x puis rising edge x et
        # rising edge y)
        if self.state == 1:
            if self.pin_x.value() == 1 and self.pin_y.value() == 1:
                self.cnt_update(+1)

    # Front montant de y pour le cas incrément
    elif self.state == 2:
        if self.pin_x.value() == 0 and self.pin_y.value() == 1:
            self.state = 3

    def k_callback(self, line):
        if self.key_cb:
            self.key_cb(line)

    def resetCallBack(p):
        s.value = 0

    def printCallBack(p):
        print(s.value)

    if 1:
        pin_vpp = Pin('X4',Pin.OUT)
        pin_vpp.value(1)

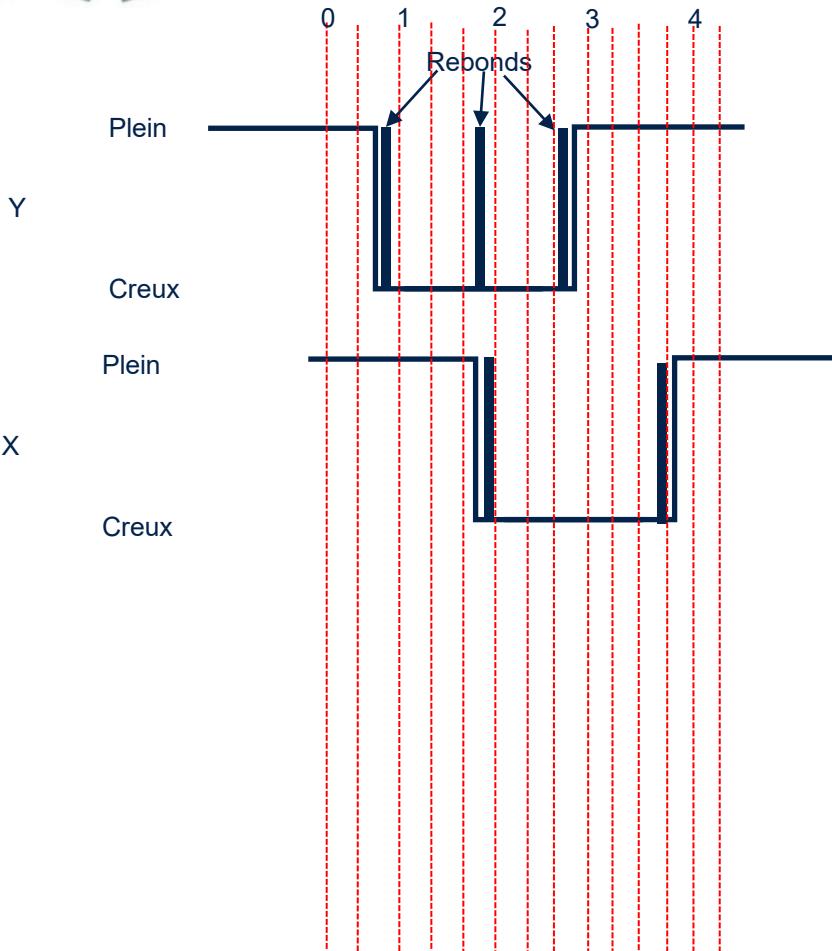
        s = Encoder(Pin('X1'),Pin('X2'),printCallBack, # ==> signaux roue codeuse
                    Pin('X3'),resetCallBack)           # ==> signal bouton et sa fonction

```



Principe (version 3 avec interruptions)

Exemple en sens horaire



- Une interruption timer est générée toutes les 2ms
- Les états de Y et X sont lus à chaque interruption pour progresser dans les différents états

```
def tim_callback(self,line):
    x = self.pin_x.value()
    y = self.pin_y.value()
    step = (x << 4) | y
    if step == self.step:
        return
    elif self.step == 0x11: # etat initial
        if step == 0x01: # etat 1 --> forward
            self.forward = True
        elif step == 0x10: # etat 1 --> backward
            self.forward = False
        self.step = step
    elif self.step == 0x10 or self.step == 0x01: # etat 1
        if step == 0x00: # --> etat 2
            self.step = step
    elif step == 0x11: # etat final
        self.step = step
        if self.forward == True:
            self.position += 1
        else:
            self.position -= 1
    elif self.step == 0x00: # etat 2
        if step == 0x01 or step == 0x10 :
            self.step = step # --> etat 3
```

Programme d'exemple

```

from machine import Pin
from pyb import Timer

pin_vpp = Pin('X4',Pin.OUT)
pin_vpp.value(1)

class Encoder:
    def __init__(self, pin_x, pin_y,pin_key=None,key_cb=None):
        self.forward = True
        self.pin_x = pin_x
        self.pin_y = pin_y
        self.position = 0
        self.key_cb = key_cb
        self.step = 0x11
        self.tim = Timer(4, period=2, callback=self.tim_callback)

        if pin_key and key_cb:
            self.k_interrupt = pin_key.irq(trigger = Pin.IRQ_FALLING, handler=self.k_callback)

    def tim_callback(self,line):
        x = self.pin_x.value()
        y = self.pin_y.value()
        step = (x << 4)| y
        if step == self.step:
            return
        elif self.step == 0x11: # etat initial
            if step == 0x01: # etat 1 --> forward
                self.forward = True
            elif step == 0x10: # etat 1 --> backward
                self.forward = False
            self.step = step
        elif self.step == 0x10 or self.step == 0x01: # etat 1
            if step == 0x00: # --> etat 2
                self.step = step
            elif step == 0x11: # etat final
                self.step = step
                if self.forward == True:
                    self.position += 1
                else:
                    self.position -= 1
            elif self.step == 0x00: # etat 2
                if step == 0x01 or step == 0x10 :
                    self.step = step # --> etat 3
                    if step == 0x01 or step == 0x10 :
                        self.step = step # --> etat 3

    def k_callback(self, line):
        if self.key_cb:
            self.key_cb(line)

    def myCallBack(p):
        s.position = 0

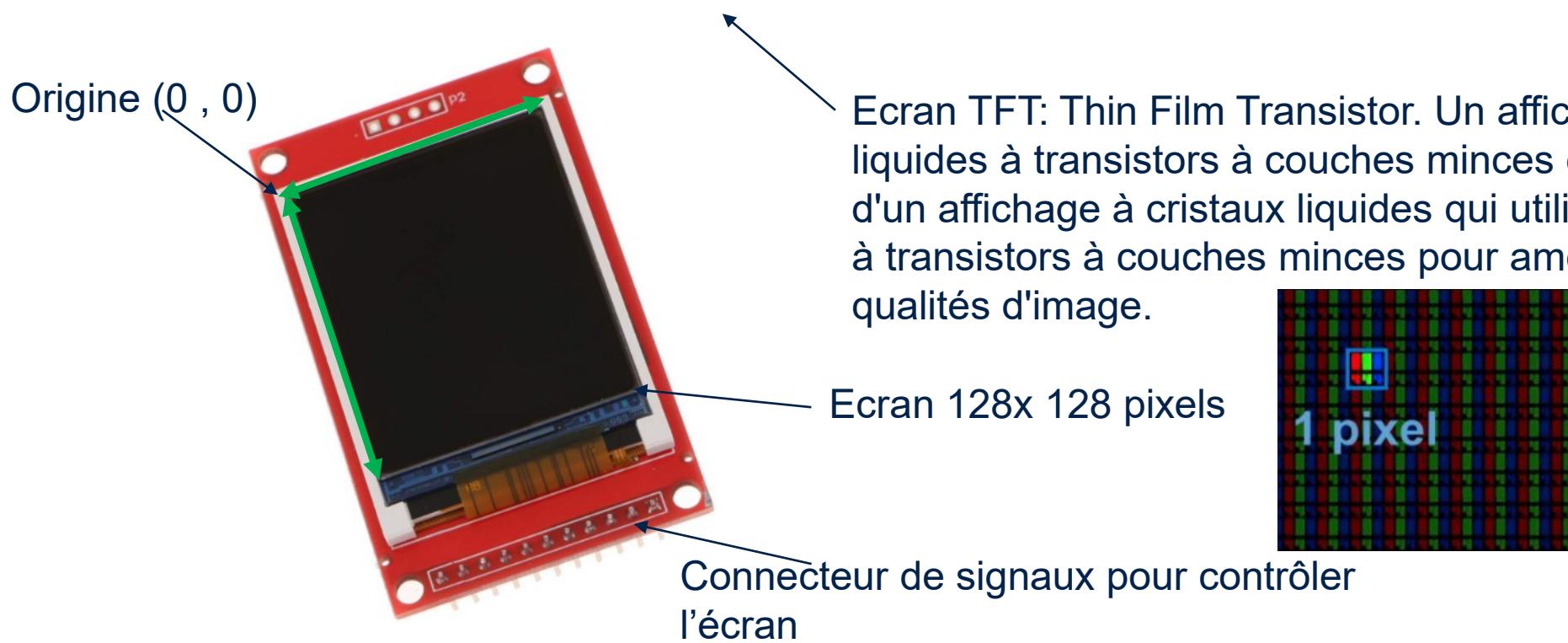
        s = Encoder(Pin('X1'),Pin('X2'), # ==> signaux roue codeuse
                    Pin('X3'),myCallBack) # ==> signal bouton et sa fonction
        pos = -1
        while True:
            if (s.position != pos):
                pos = s.position
                print(s.position)

```

Atelier 7 – L'écran LCD ST7735

Atelier 7: écran LCD TFT

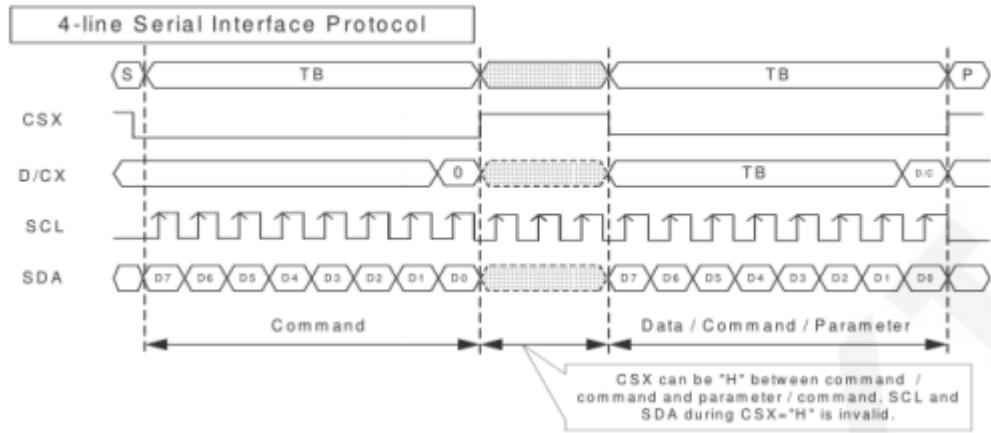
- Qu'est ce qu'un écran LCD TFT 128x128



- La datasheet permet de comprendre comment utiliser et communiquer avec l'écran:

https://github.com/stm-rns/KARE/blob/pyKare/01-DocTechnique/LCD_ST7735.pdf

- Qu'est ce qu'un interface SPI ?

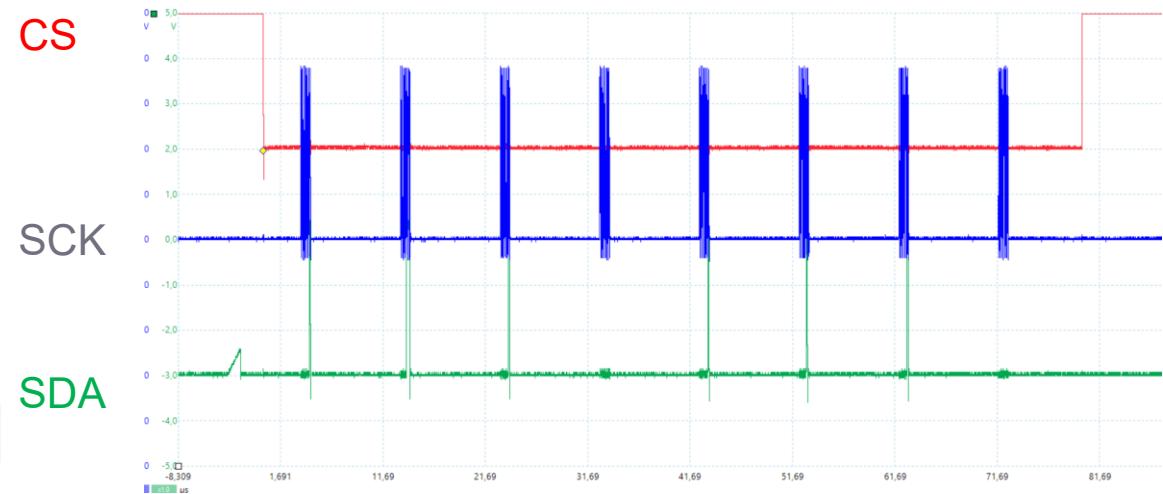


CSX is a slave chip select, and the chip is enabled only when CSX is low.

D/CX is the data/command control pin of the chip. When DCX is low, the command is written. When it is high, the data is written.

SCL is the SPI bus clock, and each rising edge transmits 1 bit of data;

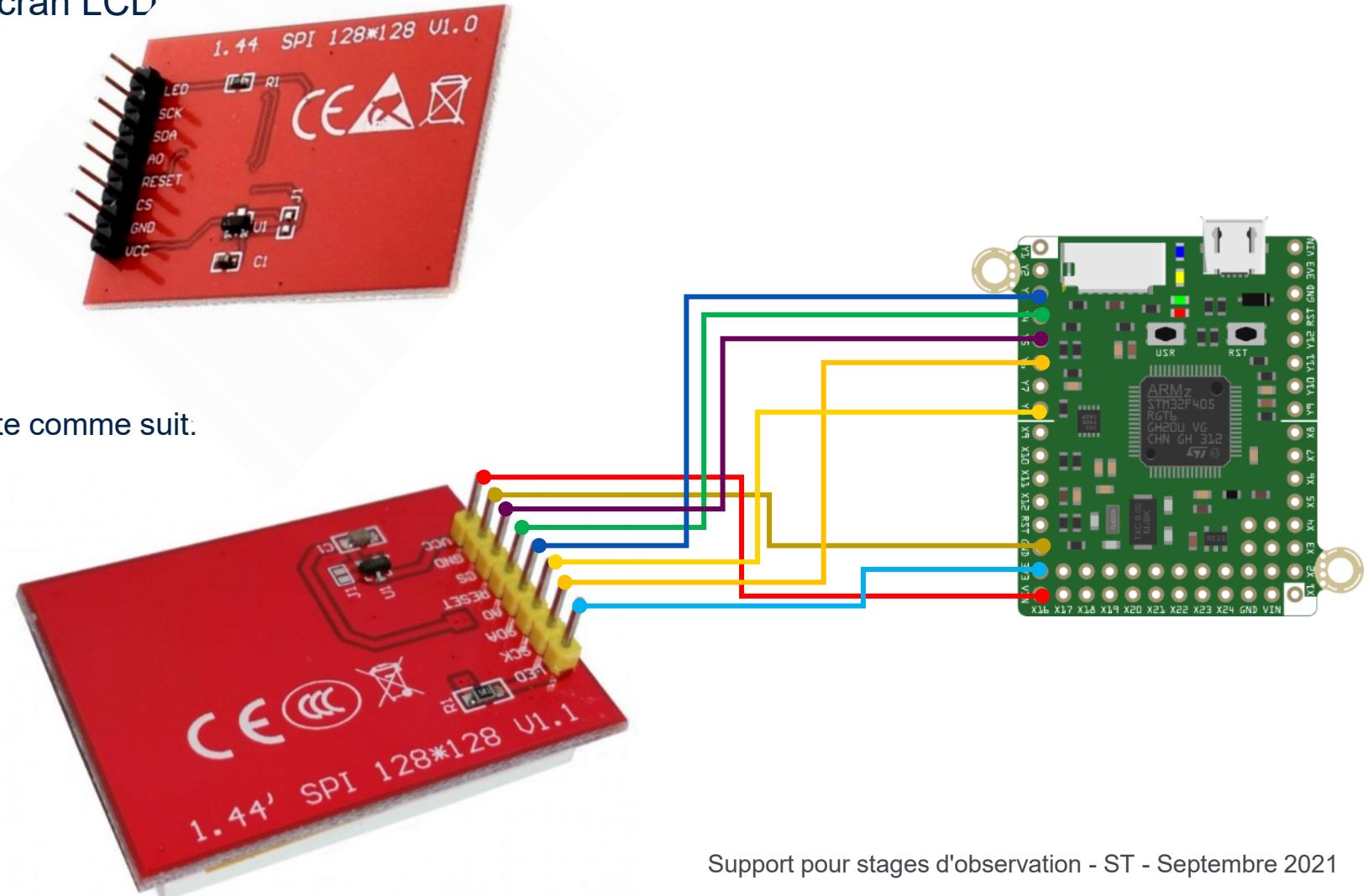
SDA is the data transmitted by SPI, and it transmits 8-bit data at a time. The data format



Atelier 7: connecter le LCD

- Connexion de la pyboard avec l'écran LCD

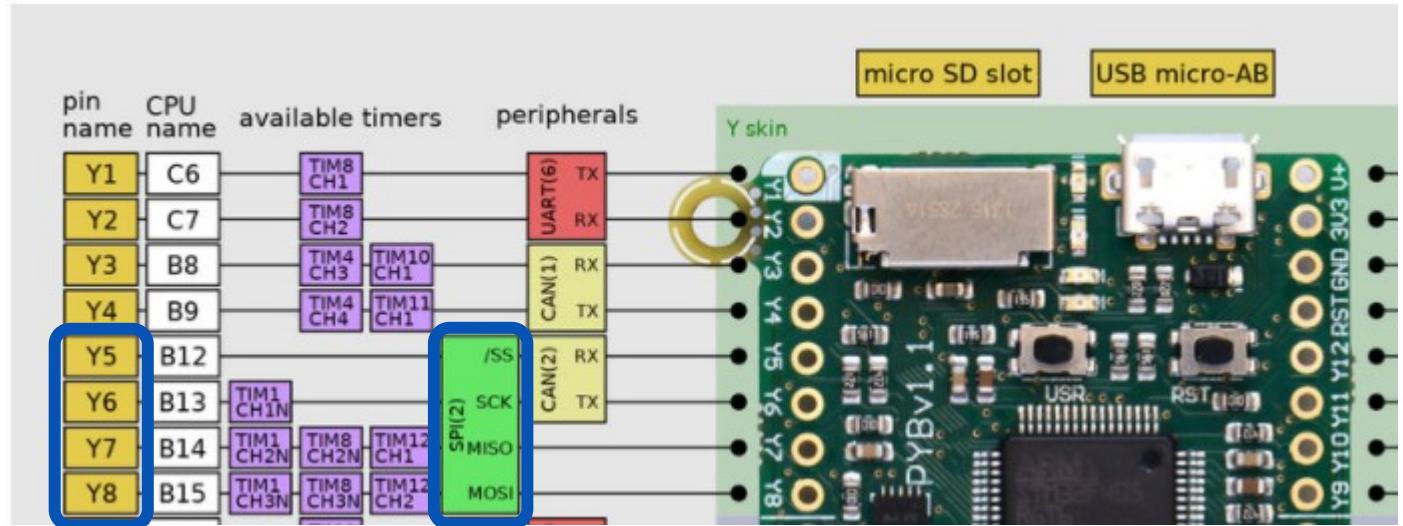
- Le LCD comporte 8 connections
 - LED: Backlight
 - SCK: Serial Clock
 - SDA: Serial Data
 - A0: DC-Data Command
 - RESET: Reset
 - CS: Chip Select
 - GND: alimentation (masse)
 - + Vcc: alimentation (5V ou 3V3)
- Ces connections sont à relier à la carte comme suit:
 - GND ↔ GND
 - Vcc ↔ V+
 - SCK ↔ Y6
 - SDA ↔ Y8
 - RST ↔ Y4
 - A0 ↔ Y3
 - CS ↔ Y5
 - LED: 3V3



Atelier 7: Pourquoi dois je câbler comme cela ?

- Le câblage n'est pas aléatoire, il est défini dans le code de mon programme main.py:
- Essayez de retrouver dans l'initialisation dans le programme main.py où se trouve cette information ?
 - Rechercher les fonctions spi et tft

- SCL \Leftrightarrow Y6 \Leftrightarrow SCK
- SDA \Leftrightarrow Y8 \Leftrightarrow MOSI
- RST \Leftrightarrow Y4
- A0 \Leftrightarrow Y3
- CS \Leftrightarrow Y5 \Leftrightarrow /SS (chipselect)



```
spi = SPI(2, baudrate=60000000, polarity=0, phase=0)
t = TFT(spi, aDC="Y3", aReset="Y4", aCS='Y5')
```

Atelier 7: Ecrire/ réaliser des figures sur le LCD

- Lancez le main.py suivant:

```
from ST7735 import *
from machine import Pin, SPI, I2C
from font import terminalfont
from random import randint
if 1:
    spi = SPI(2, baudrate=60000000, polarity=0, phase=0)
    t = TFT(spi, aDC="Y3", aReset="Y4", aCS='Y5')
    t.initb2()
    t.fill(TFT.FOREST)
    t.setvscroll(2,2)
    t.text((0,60),'ST Micro',TFT.BLUE,terminalfont,2)
    t.fillrect((25,10),(60,28),TFT.CYAN)
    t.circle((90,100), 8, TFT.GREEN)
```

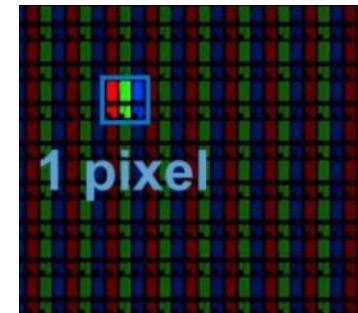
- Regardons en détail les fonctions « text, fillrect & circle » et leurs arguments. Ce sont des fonctions prédéfinies dans le driver du ST7735. Allons voir dans le fichier ST7735.py.

Atelier 7: Ecrire/ réaliser des figures sur le LCD

- La fonction rect est dans le fichier du driver ST7735.py:

```
def rect( self, aStart, aSize, aColor ) :  
    '''Draw a hollow rectangle.  aStart is the smallest coordinate corner  
    and aSize is a tuple indicating width, height.'''  
    self.hline(aStart, aSize[0], aColor)  
    self.hline((aStart[0], aStart[1] + aSize[1] - 1), aSize[0], aColor)  
    self.vline(aStart, aSize[1], aColor)  
    self.vline((aStart[0] + aSize[0] - 1, aStart[1]), aSize[1], aColor)
```

- En regardant la fonction fillrect ? Quelle est la différence ?
- Essayons de trouver dans ce fichier les différentes couleurs possibles, comment faire du NAVY en programmant les pixels?



Atelier 7: Ecrire/ réaliser des figures sur le LCD

- La fonction text est la suivante:

```
def text( self, aPos, aString, aColor, aFont, aSize = 1, nowrap = False ) :  
    '''Draw a text at the given position. If the string reaches the end of the  
    display it is wrapped to aPos[0] on the next line. aSize may be an integer  
    which will size the font uniformly on w,h or a or any type that may be  
    indexed with [0] or [1].'''
```

- Essayons de modifier le main.py pour écrire ton prénom, un rectangle aux contours verts, et un cercle plein de couleur rouge.

Atelier 7: Ecrire/ réaliser des figures sur le LCD

- Nous pouvons rajouter un effet scrolling vertical au LCD en rajoutant la fonction suivante:

```
while True:  
    for i in range(0,90,1):  
        t.vscroll(i)  
        time.sleep(0.015)  
    for i in range(90,0,-1):  
        t.vscroll(i)  
        time.sleep(0.015)
```

- Quels arguments permettent de modifier le scrolling vertical ?

Atelier 7: Afficher une image complète sur le LCD

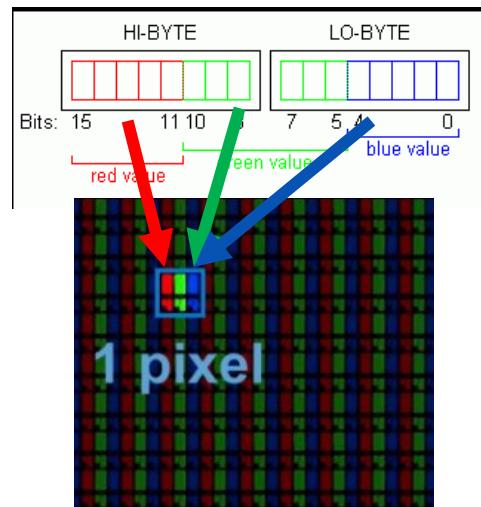
- Etape 1: Choisir une image sur internet de votre choix: Attention à la taille du fichier bmp. Il doit faire une taille de 128x128 en 24 bits. Si nécessaire, on peut la convertir grâce à paint (faire un resize en 128x128 pixel et enregistrer 24bits bitmap). Le format « bmp » permet d'allouer 1 octet bleu, 1 octet vert et 1 octet rouge par pixel.
- Etape 2: Pour que l'image soit « exploitable » par l'écran LCD, il faut qu'il soit dans un format binaire.
- Utilisons le site suivant <https://lvgl.io/tools/imageconverter> pour convertir le fichier en RGB565

- Qu'est ce que veut dire RGB565 pour un pixel ?
- On alloue 5 bits pour le rouge, 6 bits pour le vert et 5 bits pour le bleu.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
r7	r6	r5	r4	r3	r2	r1	r0	g7	g6	g5	g4	g3	g2	g1	g0	b7	b6	b5	b4	b3	b2	b1	b0	
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
r7	r6	r5	r4	r3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	g7	g6	g5	g4	g3	g2	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b7	b6	b5	b4	b3	0	0	
									r7	r6	r5	r4	r3	g7	g6	g5	g4	g3	g2	b3	b6	b5	b4	b3

Atelier 7: Afficher une image complète sur le LCD

- Que représente le format RGB565 pour un pixel ?



Atelier 7: Afficher une image complète sur le LCD

- Etape 3: utiliser le main.py suivant:

```
from ST7735 import *
from machine import Pin, SPI, I2C
from font import terminalfont
from random import randint

spi = SPI(2, baudrate=60000000, polarity=0, phase=0)
t = TFT(spi, aDC="Y3", aReset="Y4", aCS='Y5')
t.initb2()
t.fill(TFT.FOREST)

with open('output.bin', 'rb') as f:
    for y in range(128):
        for x in range(128):
            data = (f.read(2))
            d = (data[1]<<8)|(data[0])
            pos = [x,y]
            t.pixel(pos, d)
```

- Question: comment faire pour avoir l'image en Cyan ?

Atelier 8: Optimiser l'affichage d'image sur le LCD

- Etape 4: on peut optimiser l'affichage de l'image avec ce programme suivant:

```
from random import randint
from machine import SPI
from st7735 import *
import struct

spi = SPI(2, baudrate=60000000, polarity=0, phase=0)
t = TFT(spi, aDC="Y3", aReset="Y4", aCS="Y5")
t.initb2()
t.fill(TFT.FOREST)

def displayBMP24(t,bmp,x0=0,y0=0):
    bmp = open(bmp,'rb')
    data = bmp.read(54)
    bm,file_size,x1,x2,h54,h40,w,h,layers,bits_per_pixel,compression ,x3,res_h,res_v,x7,x8 = struct.unpack("<HLHHLLLLHHLLLLL", data[:54])
    assert bm == 0x4D42, "bad header 1"
    assert h54 == 54, "bad header 2"
    assert h40 == 40, "bad header 3"
    assert layers == 1, "bad header 4"
    #assert len(data) == file_size, "bad header 5"
    assert bits_per_pixel == 24, "bad header 6"
    assert compression == 0, "bad header 7"

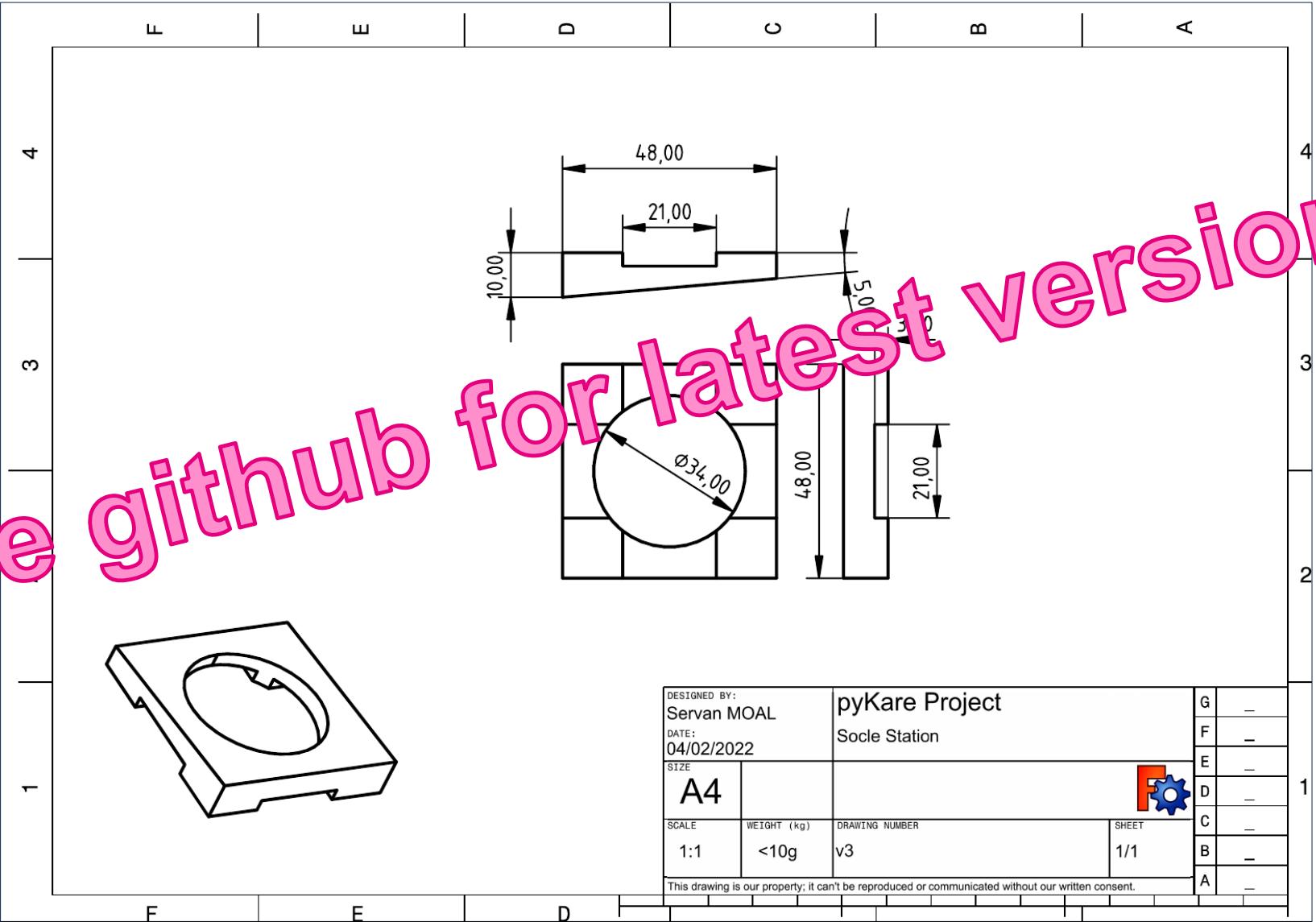
    line_size_in_bytes = bits_per_pixel * w//8
    struct_line = '>%dH' % (w)

    for y in range(h):
        line = bmp.read(line_size_in_bytes) # read bytes from file
        #pixels = [ TFTColor(line[i*3+2],line[i*3+1],line[i*3]) for i in range(w) ] # RGB24 --> RGB565
        pixels = [ ((line[i*3+2] & 0xF8) << 8) | ((line[i*3+1] & 0xFC) << 3) | (line[i*3] >> 3) for i in range(w) ] # RGB24 --> RGB565
        d = struct.pack(struct_line, *pixels) # pack bytearray
        t.image(x0, h-y+y0, w-1+x0, h-y+y0, d) # display

displayBMP24(t,'ST.bmp', 0,0)
```

Atelier 8 - FreeCAD

See github for latest version





Atelier 9 – Code Final

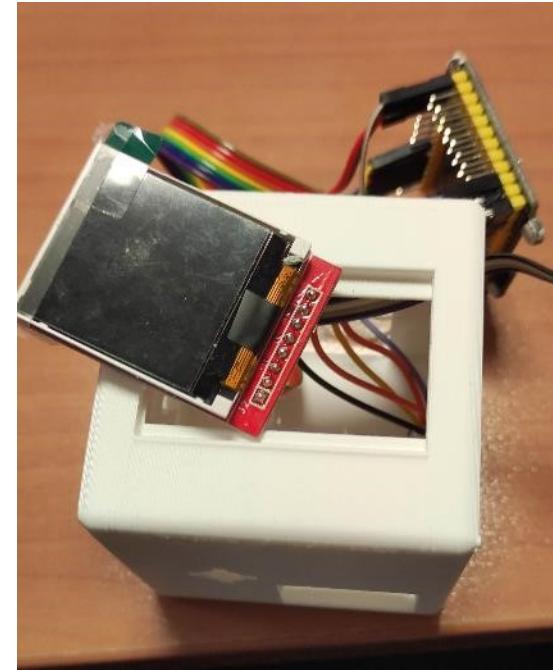
Atelier 9: Code final et intégration

- Extraire depuis votre répertoire de travail le fichier Final.zip
 - C:\.....\KARE\03-Code_Source\Final\Final.zip
 - Dans Thonny:
 - Télécharger l'ensemble des fichiers dans la racine de la board:
 - main.py
 - fnt.bin
 - ST_48.bmp
 - st7735.py
 - rotary_tim.py
 - boot.py
 - LPS22.py
 - Tester pong.py
 - Nous allons essayer d'intégrer le pong dans le programme complet

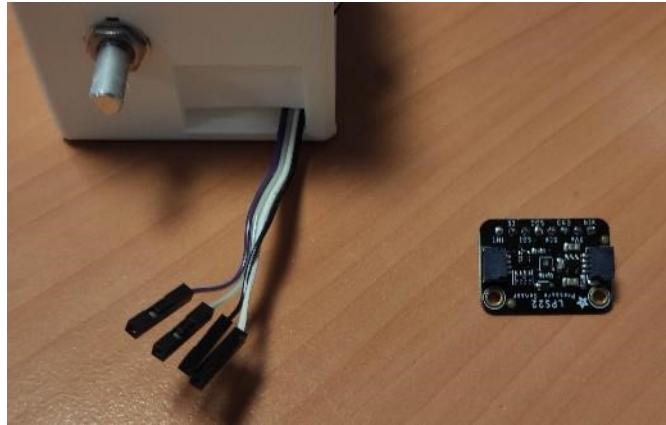
Atelier 9 - Assemblage



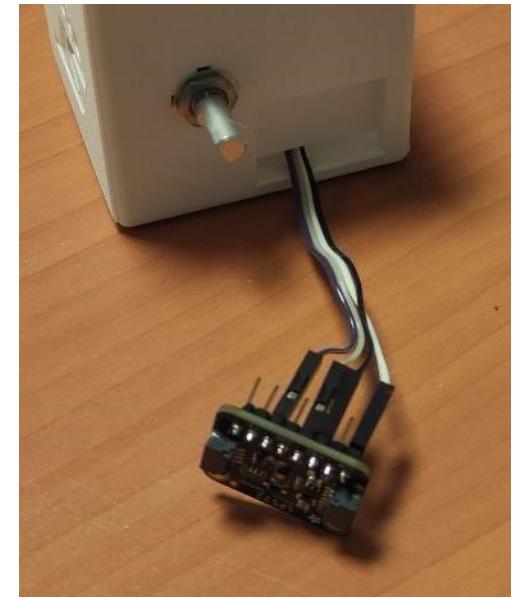
1. Faire passer l'ensemble câblé par le trou de la pyBoard
2. Ressortir l'écran et le positionner
3. Faire ressortir la molette et visser son écrou.



Atelier 9 - Assemblage



1. Noter l'ordre des fils du LPS22
2. Décâbler pour faire passer par la fente la nappe.
3. Recâbler



Atelier 9 - Assemblage



1. Placer la pyBoard dans son emplacement
2. Fermer avec le pied
3. Brancher
4. C'est fini!



Atelier 9: Code final et intégration

- Trouver la table des applications et ajouter pong_app:

```
APPS = [
    ("ST logo",bouncing_logo_app),
    ("Heure", reveil_app),
    ("Meteo",meteo_app),
    ("Niveau",level_sensors_app),
    ("Mandelbrot",mandel_app),
    ("Bouncing Balls",balls_app),
    ("Snake",snake_app),
    ("Dice",dice_app),
    ("Rect. Falls",rect_fall_app),
    ("Pong",pong_app),
]
```

EXPERT

Copier la fonction pong_app():

- Depuis pong.py dans main.py du PC et tester

```
def pong_app():
    # Dimensions et position de la raquette: Longueur, hauteur, position X,Y
    raquetteL = 4
    raquetteH = 10
    raquetteX = 1
    raquetteY = 60
    raquetteVitesse = 4
    last_posR = e.position

    # Dimensions et position de la balle: Longeur, hauteur, position X,Y
    balleR = 3
    balleX = randint(40,80)
    balleY = randint(40,80)

    # Zone de rebond de la balle
    zoneXLow = raquetteL+balleR
    zoneXHigh = 128 - balleR
    zoneYLow = balleR
    zoneYHigh = 128 - balleR

    # Sens de déplacement de la balle
    balleMoveX = balleMoveY = 1

    # Nombre de touches de balle
    scoreOK = scoreKO = 0

    raquetteCouleur = TFT.WHITE

    t.fill(0) # Ecran noir
    update_score = True
    while True:
        # Gestion de la raquette
        posR = e.position
        if posR != last_posR: # La raquette a bougé
            last_posR = posR
            # On efface toute la ligne de la raquette en couleur de fond
            t.fillrect((1,1),(10,128),TFT.BLACK)
            inc = raquetteVitesse if e.forward else -raquetteVitesse
            if 0 <= raquetteY+inc <= 128 - raquetteH: raquetteY += inc
        # Affichage de la raquette
```

Atelier 9: Code final et intégration

```
t.fillrect((raquetteX,raquetteY),(raquetteL,raquetteH),raquetteCouleur)

if update_score: # Affichage score ou autres données
    t.fillrect((100,28),(6,80),TFT.BLACK) # on efface
    t.text((100,28),'%d' % scoreOK,TFT.GREEN,terminalfont,1)
    t.text((100,100),'%d' % scoreKO,TFT.RED,terminalfont,1)
    update_score = False

# Gestion de la balle
t.fillcircle((balleX,balleY),balleR,TFT.BLACK) # On efface la balle d'avant
balleX = balleX + balleMoveX
balleY = balleY + balleMoveY
t.fillcircle((balleX,balleY),balleR,TFT.YELLOW)

# La balle passe sur le score
if 100 - balleR <= balleX <= 105 + balleR: update_score = True

if balleX >= zoneXHigh: # On atteint la limite supérieure en X: on change de sens
    balleMoveX = -balleMoveX
elif balleX <= zoneXLow:
    balleMoveX = randint(1,2) # nouvelle vitesse de balle
    update_score = True # le score va changer
    # On a atteint la bas de l'écran. Il faut donc voir si la raquette est là...
    if raquetteY - balleR <= balleY <= raquetteY + raquetteH + balleR:
        scoreOK += 1 # La balle est sur la raquette
        raquetteCouleur = TFT.GREEN
    else:
        scoreKO += 1 # La balle a raté la raquette
        raquetteCouleur = TFT.RED

# On atteint les limites Y: on change de sens
if balleY >= zoneYHigh or balleY <= zoneYLow:
    balleMoveY = -balleMoveY

time.sleep(0.005)
if sw() or click: break
```

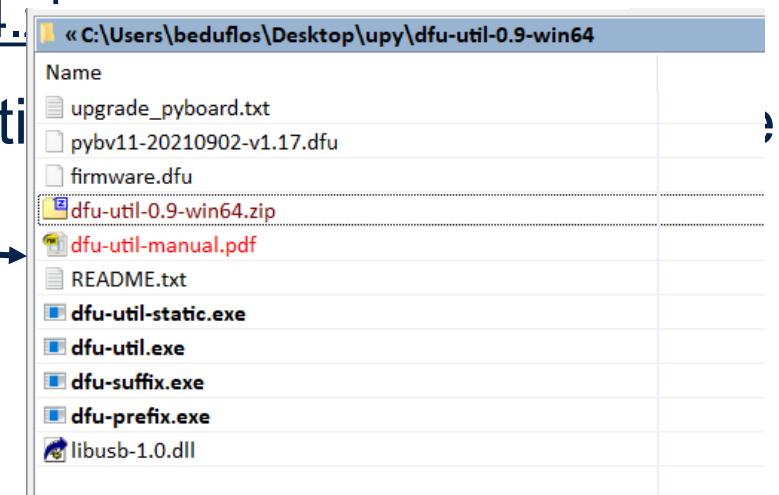
Pour aller plus loin...

Mettre à jour de sa pyboard

- Voici comment configurer votre carte PyBoard pour qu'elle passe en mode DFU (Device Firmware Update = mise-à-jour du micrc)
- Télécharger le dernier firmware micropython pour la PyBoard:
 - <https://micropython.org/download/pybv1/>
- Télécharger dfu-util 0.9:
 - <http://dfu-util.sourceforge.net/releases/dfu-util-0.9-win64.zip>
- Dézipper le firmware (fichier dfu) ainsi que dfu-util

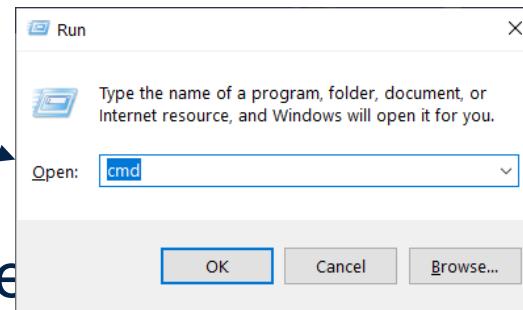
Firmware for PYBv1.1 boards

- standard: v1.17-8-gbbbdef4cc (latest) ; v1.17-7-gaf64c2ddb ; v1.17-6-g9792c9105 ; v1.17 ; v1.16-262-g89145c6aa ; v1.16 ; v1.15 ; v1.14 ; v1.13 ; v1.12 ; v1.11 ; v1.10 ; v1.9.4 ; v1.9.3 ; v1.9.2 ; v1.9.1 ; v1.9 ; v1.8.7 ;
- double FP: v1.17-8-gbbbdef4cc (latest) ; v1.17-7-gaf64c2ddb ; v1.17-6-g9792c9105 ; v1.17 ; v1.16-262-g89145c6aa ; v1.16 ; v1.15 ; v1.14 ; v1.13 ; v1.12 ; v1.11 ; v1.10 ; v1.9.4 ; v1.9.3 ; v1.9.2 ;
- threading: v1.17-8-gbbbdef4cc (latest) ; v1.17-7-gaf64c2ddb ; v1.17-6-g9792c9105 ; v1.17 ; v1.16-262-g89145c6aa ; v1.16 ; v1.15 ; v1.14 ; v1.13 ; v1.12 ; v1.11 ; v1.10 ; v1.9.4 ; v1.9.3 ; v1.9.2 ;
- double FP + threading: v1.17-8-gbbbdef4cc (latest) ; v1.17-7-gaf64c2ddb ; v1.17-6-g9792c9105 ; v1.17 ; v1.16-262-g89145c6aa ; v1.16 ; v1.15 ; v1.14 ; v1.13 ; v1.12 ; v1.11 ; v1.10 ; v1.9.4 ; v1.9.3 ; v1.9.2 ;
- network: v1.17-8-gbbbdef4cc (latest) ; v1.17-7-gaf64c2ddb ; v1.17-6-g9792c9105 ; v1.17 ; v1.16-262-g89145c6aa ; v1.16 ; v1.15 ; v1.14 ; v1.13 ; v1.12 ; v1.11 ; v1.10 ; v1.9.4 ; v1.9.3 ; v1.9.2 ; v1.9.1 ; v1.9 ; v1.8.7 ;



Mettre à jour sa pyboard

- Mettre la PyBoard hors tension et déconnecter tout
- Relier les deux broches en rouge comme sur l'image (BOOT0 et 3V)
- Ouvre une console, aller dans répertoire créé et exécuter la commande
 - Windows > Exécuter > cmd
 - Cd votre_répertoire
 - dfu-util --alt 0 -D pybv11_xxxxxx-vx.xxx.dfu
- Mettre la PyBoard hors tension et connecter toutes les broches
- Fini !



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1734]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\beduflos>cd C:\Users\beduflos\Desktop\upy\dfu-util-0.9-win64

C:\Users\beduflos\Desktop\upy\dfu-util-0.9-win64>dfu-util --alt 0 -D pybv11-20210902-v1.17.dfu
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

Match vendor ID from file: 0483
Match product ID from file: df11
Opening DFU capable USB device...
ID 0483:df11
Run-time device DFU version 011a
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuERROR, status = 10
dfuERROR, clearing status
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 2048
DfuSe interface name: "Internal Flash "
file contains 1 DFU images
parsing DFU image 1
image for alternate setting 0, (2 elements, total size = 367816)
parsing element 1, address = 0x08000000, size = 14592
Download [=====] 100% 14592 bytes
Download done.
parsing element 2, address = 0x08020000, size = 353208
Download [=====] 100% 353208 bytes
Download done.
done parsing DfuSe file
```

Le matériel

Commande du matériel

- Le matériel nécessaire se trouve facilement sur Internet
- Les prix varient beaucoup d'un fournisseur à l'autre pour des produits équivalents
- Il faut être prudent qu'une même référence peut amener à des cartes électroniques très différentes. Par exemple, pour la carte des capteurs, la référence est le composant principal LPS22... mais beaucoup de cartes différentes existent qui portent ce nom. Il faudra donc se référer aux photos d'illustration de ce fichier pour faire le bon choix...

Révision des commandes

Vendeur: YTF Technology

 Module de pression atmosphérique module de pression atmosphérique LPS22HB

€ 3,84

Livraison: € 2,01 via AliExpress Standard Shipping Temps estimé pour la livraison: 10 août >

+ Laisser un message

Sous-total	€ 76,71
Livraison	€ 2,01
TVA	€ 15,14
Coupons vendeur	
Total	€ 93,87

 Capteur de pression LPS22, stam...

€ 11,68

+ envoi: € 2,91

MBM-Chip Store

Carte microPython

AliExpress

ZTHOME Party Store 94.5% Évaluations positives

+ Suivre 1025 Abonnés

Je cherche... Sur AliExpress Dans ce

Accueil boutique produits ▾ Articles en Promos Meilleure vente Nouveautés réaction

Carte de développement de programmation Python MicroPython Pyboard V1.1, 4.2x 3.3cm, 1 pièce

★★★★★ 4.9 ▾ 7 Avis 16 Commandes

€ 10,06 € 14,37 -30%

TVA sera incl. au paiement

Quantité:

- 20 + Supplémentaire 1% (4 unités ou plus)
9956 unités disponibles

Expédition : € 5,40

Vers France via AliExpress Standard Shipping ▾

Délai de livraison estimé : 10 août ⓘ

Acheter maintenant Ajouter au panier 20

75 jours de protection de l'acheteur Remboursement garanti

Carte des capteurs



MBM-Chip Store ▾

97.5% Évaluations positives

+ Suivre

865 Abonnés

Je cherche...

Sur AliExpress

Accueil boutique

produits ▾

Articles en Promos

Meilleure vente

Nouveautés

réaction

Capteur de pression LPS22, stamma QT/Qwiic, LP, 4633

€ 11,68

Prix TVA incluse

Quantité:



1



999 unités disponibles

Expédition : € 2,91

Vers France via AliExpress Standard Shipping ▾

Délai de livraison estimé : 02 nov. ?

Acheter maintenant

Ajouter au panier



75 jours de protection de l'acheteur
Remboursement garanti

Ecran LCD

AliExpress

cuiisw module Store ▾
98.3% Évaluations positives

+ Suivre
11709 Abonnés

Je cherche... Sur AliExpress Dans ce magasin

Accueil boutique produits ▾ Articles en Promos ▾ Meilleure vente Module ▾ réaction

cuiisw



Ecran TFT en couleur, lecteur IC, module LCD de 0.96, 1.3, 1.44, 1.8 pouces, IPS, 7P, SPI, HD, 65 K, composant ST7735/ST7789, 80x160 ou 240x240 (pas OLED)

★★★★★ 4.9 ✓ 256 Avis 473 Commandes

€ 3,77

TVA sera incl. au paiement
Réduction instantanée: € 0,87 de réduction chaque € 33,84 dépensés ✓

Couleur: 1.44 TFT inch



Quantité:

— 1 + 5% (10 unités ou plus)
9227 unités disponibles

Livraison gratuite
Vers France via Cainiao Super Economy Global ✓
Délai de livraison estimé : 01 sep. ⓘ

Acheter maintenant Ajouter au panier

3391

Ensemble des cartes

- Pour référence la commande passée par ST chez Logitrade (pour AliExpress) en juillet 2021

Item	Article	Désignation Article	Quantité	Prix Net HT	Unité	Montant Net HT	Code Taxes	Délais (jours ouvrés)
00000001	PYBOARD-VII	CARTE MICROPYTHON PYBOARD V1.1	20,0000	13,5000	UN	270,00	20,00	0
00000002	CETFT144P	CARTE ECRAN TFT COULEUR 1.44 POUCES 128X128	20,0000	6,0000	UN	120,00	20,00	0
00000003	LBS22HB	MODULE DE PRESSION ATMOSPHERIQUE LPS22HB	20,0000	6,5000	UN	130,00	20,00	0

Nappes de liaison

- Attention à prendre des nappes à trous aux extrémités et pas à pics...

Farnell
AN AVNET COMPANY

Tous 2762507

[Tous les produits](#) [Fabricants](#) [Ressources](#) [Communauté](#)

MISES A JOUR IMPORTANTES:
Nous vous informons qu'en raison d'un grand nombre d'appels, des interlocuteurs anglois sont à votre disposition de 9h à 16h pour répondre à vos demandes
Rendez-vous sur la page "Informations de livraison" pour consulter les dernières informations sur notre service de livraison

Accueil > Outils et fournitures de production > Outils et cartes de prototypage > Cavaliers

BC-32629

Cavalier, Assortiment, Multicolor, 26 AWG, 40 Fils, Femelle/Femelle, 200 mm





Fabricant :	BUD INDUSTRIES
Réf. Fabricant:	BC-32629
Code Commande :	2762507
Fiche technique:	 BC-32629 - Datasheet

Découvrez tous les documents techniques

 Ajouter au comparateur

L'image a des fins d'illustration uniquement. Veuillez lire la description du produit.

amazon.fr Électro Prime

High-Tech ▾

Livré à LE Béton 35830

Parcourir les catégories ▾

Chez LE Ventes Flash Nouveau · Prix Mini Chèques cadeaux Vendre Aide

High-Tech Meilleures ventes Téléphone ▾ Photo & Caméscopes ▾ TV & Vidéo ▾ Audio & HiFi ▾ Objets connectés ▾ Accessoires High Tech ▾ GPS & Auto ▾ Informatique ▾ Bons plans ▾ Offres reconditionnées ▾

Bonjour LE Votre compte ▾ Testez Prime ▾ Vos Listes ▾ Panier

High-Tech > Consommables et accessoires > Accessoires Image et Son > Boîtiers Pi TV

Vous avez acheté cet article le 19 janvier 2019.

Taille: 40pcs 20cm | Couleur: Female to Female | Afficher cette commande

Ganvol 40 câbles DE 20 cm - Femelle vers Femelle - sans Soudure - Flexibles - pour Platine d'expérimentation Arduino, Raspberry Pi Modèle A/Modèle B 1 1 + 2 3/Module d'ordinateur/Zero de Ganvol

★★★★★ 15 commentaires client

Amazon's choice pour "cable dupont"

Prix : **EUR 3,95** Livraison gratuite dès EUR 25 d'achats en France métropolitaine. [Détails](#)

Tous les prix incluent la TVA.

Livraison GRATUITE (EUR 0,01 pour les livres) en point retrait. [Détails](#)

2 neufs à partir de **EUR 3,95**

Taille: 40pcs 20cm

40pcs 20cm 120pcs 20cm

Couleur: Female to Female

Ganvol Câbles de Dupont sont parfait pour une utilisation sur plaque à prototype. Utilisez régulièrement ces câbles sur des breadboards et des cartes Arduino pour faire câblages. Il permet une bonne connectivité entre les différents éléments Arduino! Câbles pour réaliser des branchements sur une carte Arduino.

Les fiches sont parfaitement calibrées, permis de brancher 40 de ces câbles sur un Raspberry Pi/ Arduino sans problèmes. Tous les connecteurs sont idéalement conçus pour GPIO ou pour plaquette d'essai avec connecteurs au pas de 2,54.

Connexion toutes les rames, on peut choisir de séparer les fils ou non, permet de faire des montages esthétiques, propres. le fait qu'il y est plusieurs types et plusieurs couleurs rend l'achat très utile.

Les câbles sont très facile à manier. Ils sont également "collés" entre eux en fonction de leur connecteur, mais détachable facilement.

Pas d'odeur particulière, sauf l'odeur de plastique habituelle quand on colle le nez dessus. Mais pour bricoler avec quelques câbles, l'odeur disparaît rapidement. Après une aération rien de problématique.

[Voir plus de détails](#)

EUR 3,95

Livraison gratuite dès EUR 25 d'achats en France métropolitaine. [Détails](#)

Voulez-vous le faire livrer le vendredi 3 mai? Commandez-le dans les 4 h et 10 mins et choisissez la Livraison en 1 jour ouvré au cours de votre commande. [En savoir plus.](#)

En stock.

Quantité : [Ajouter au panier](#)

[Acheter cet article](#)

Vendu par **GanvolTech** (incluant la TVA) et expédié par Amazon.

Livré en 1 jour ouvré

Profitez de tous les avantages de livraison en vous inscrivant à **Amazon Prime**

amazon prime

La commande 3. Click & Collect

Boutons

- Version Radiospares:

Encodeur rotatif mécanique, , Incrémental, 18 impulsions par tour, axe de 6 mm, Traversant

Code commande RS: 781-6827 | Référence fabricant: PEC11R-4120F-S0018 | Marque: Bourns



Documentation technique

[Datasheet](#)

En stock à partir du 26/10/2021, pour livraison dès le lendemain

1 Unité

[Commander](#)

Prix pour la pièce
1,74 €
HT 2,09 €
TTC

Unité	Prix par unité
1 - 9	1,74 €
10 - 24	1,40 €
25 - 99	1,39 €
100 - 249	1,28 €
250 +	1,22 €

Options de conditionnement :

- Conditionnement standard
- Conditionnement industriel standard

Version Amazon:



BIENVENUE ADRESSE ARTICLES EMBALLAGE LIVRAISON PAIEMENT VALIDATION

Vérification et validation de votre commande

En passant votre commande, vous acceptez les [Conditions générales de vente d'Amazon](#). Veuillez consulter notre [Notice Protection de vos informations personnelles](#), notre [Notice Cookies](#) et notre [Notice Annonces publicitaires basées sur vos centres d'intérêt](#).

Vous voulez gagner du temps lors de votre prochaine commande en allant directement à cette page lorsque vous payez?
 Vous souhaitez que les informations de paiement et de livraison ci-dessous deviennent vos coordonnées par défaut.

Adresse d'expédition [Modifier](#)

LE THEO Jean-Pierre STMicroelectronics

10 Rue Jouanet

STMicroelectronics

RENNES, 35700

France

Téléphone: 0678946071

[Ajouter des instructions de livraison](#)

[Envoyer à plusieurs adresses](#)

Livraison GRATUITE en point retrait

20 points sont près de cette adresse [Voir](#)

Mode de paiement [Modifier](#)

***-0075

Adresse de facturation [Modifier](#)

LE THEO Jean-Pierre

24 Allée Erik Satie

BETTON, 35830

France

Codes chèques-cadeaux et codes promotionnels

Saisissez le code

[Appliquer](#)

[Acheter](#)

Récapitulatif de Commande

Articles :	EUR 34,28
Livraison :	EUR 4,99
Total :	EUR 39,25
Bon de réduction:	-EUR 4,99

Montant total : EUR 34,26

Le total de la commande inclut la TVA.
[Voir les détails.](#)

Promotions appliquées :
+ Livraison Gratuite
[Comment sont calculés les frais de livraison ?](#)

amazon prime LE THEO Jean-Pierre, nous vous offrons un essai gratuit de 30 jours à Amazon Prime : recevez cette commande et les prochaines éligibles avec la livraison en 1 jour ouvré gratuite (ou 0,01€ par livre). [En savoir plus](#)

Date de livraison estimée : 3 mai 2019



Hikig 5 pcs Ky-040 Rotary Encoder module avec 15 x 16,5 mm avec capuchon de bouton pour Arduino (lot de 5) Hkt1062

EUR 11,42

Peut bénéficier d'Amazon Prime [Inscrivez-vous maintenant](#)

En stock

Quantité : 3 [Modifier](#)

Vendu par : Hikig Online EU

[Ajouter des options cadeau](#)

Choisissez votre mode de livraison:

Demain, 29 avril
GRATUIT (ou 0,01€ par livre) : Livraison en 1 jour ouvré avec l'essai gratuit de 30 jours [amazon prime](#))

vendredi 3 mai

Livraison Standard

Demain, 29 avril

Livraison en 1 jour ouvré

Livraison à la carte

Recevez-le mardi, le 30 avril entre 14h et 16h

Numéro de téléphone du destinataire : 0678946071

[Modifier le numéro de téléphone](#)

[En savoir plus](#)

Ruban d'impression 3D

Farnell_14_Wxx*** A compléter pour votre recherche

VERBATIM - 55251 - 3D PRINTING FILAMENT, PLA, 1.75MM, WHITE - 55251	1	2828655	1	33.60 €
---	---	---------	---	---------

BREXIT UPDATE Un changement apporté à la marque Farnell Offres Nous contacter Assistance Suivi des commandes

Tous 2828655 Connectez-vous Enregistrez-vous Mon compte 0 article: 0,00 €

Tous les produits Fabricants Ressources Communauté Favoris Outils

Accueil > Outils et fournitures de production > Imprimantes 3D & Accessoires > Filaments pour imprimantes 3D > 55251

55251 - Filament Imprimante 3D, 1.75mm, PLA, Blanc

Verbatim

Fabricant : VERBATIM
Réf. Fabricant: 55251
Code Commande : 2828655
Fiche technique: 55251 Datasheet
Découvrez tous les documents techniques

L'image a des fins d'illustration uniquement. Veuillez lire la description du produit.

Ajouter au comparateur Rédiger Un Avis

Informations produit

<input type="checkbox"/> Diamètre: 1.75mm	<input type="checkbox"/> Largeur Rouleau: 72mm
<input type="checkbox"/> Couleur Filament: Blanc	<input type="checkbox"/> Diamètre Moyeu du Rouleau: 102mm
<input type="checkbox"/> Matériau Filament: PLA (Polylactide)	<input type="checkbox"/> Gamme de produit: -
<input type="checkbox"/> Température, fusion: 220°C	

Filament d'imprimante 3D, Série PolyLite, Dia. 1,75mm, Noir, PLA, 150°C, 1 kg **1** 70525 **1** 35.09 €
Filament d'imprimante 3D, Dia. 1,75mm, Blanc, PLA, 146 à 150°C, 1 kg **1** 1103010001 **1** 29.99 €
Filament d'imprimante 3D, Dia. 1,75mm, Bleu, PLA, 146 à 150°C, 1 kg **1** 1103010101 **1** 29.99 €

19 En stock
19 en stock pour une livraison le jour ouvré suivant (UK stock)
Voir heures limites
Consulter le stock et les délais d'approvisionnement

38,18 €
Prix pour : Pièce
Multiple: 1 Minimum: 1

Quantité **Prix**
1+ 38,18 €

Demandez un devis pour de plus grandes quantités

Qté: Ajouter au panier

Ajouter Référence Interne / Note à la ligne Ajouter aux favoris

Documents techniques

Technical Data Sheet EN

Le matériel (versions alternatives)

Le matériel (première génération pour les tuteurs)

- Matériel remis au stagiaire :



Câbles Dupont

Boitier imprimé en 3D



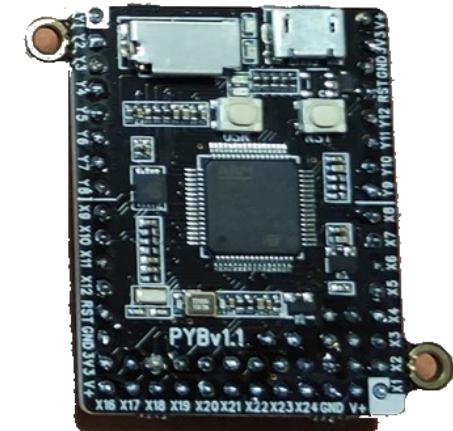
Ecran LCD 128x128



Capteur de température



Roue codeuse



pyBoard

- Project manager: benoit.duflos@st.com
- Sponsor: Jean-pierre.lettheo@st.com

Thank you

© STMicroelectronics - All rights reserved.

The STMicroelectronics corporate logo is a registered trademark of the STMicroelectronics group of companies. All other names are the property of their respective owners.

