Atelier numérique
de Bures sur Yvette

microPython sur ESP

Chris - Arnaud

Atelier numérique
de Bures sur Yvette

Pourquoi microPython sur ESP

Facilité de prototypage

ESP répond sur une console arduino en ligne de commande

Disponibilité d'un environnement IA pour K210 (MAIXPY)

Python utilisé par les élèves et les étudiants

Installation sur ESP8266 (1/3)

- Il faut d'abord installer un nouveau firmware
  - https://docs.micropython.org/en/latest/esp8266/tutorial/intro.html
  - https://micropython.org/download/?port=esp8266

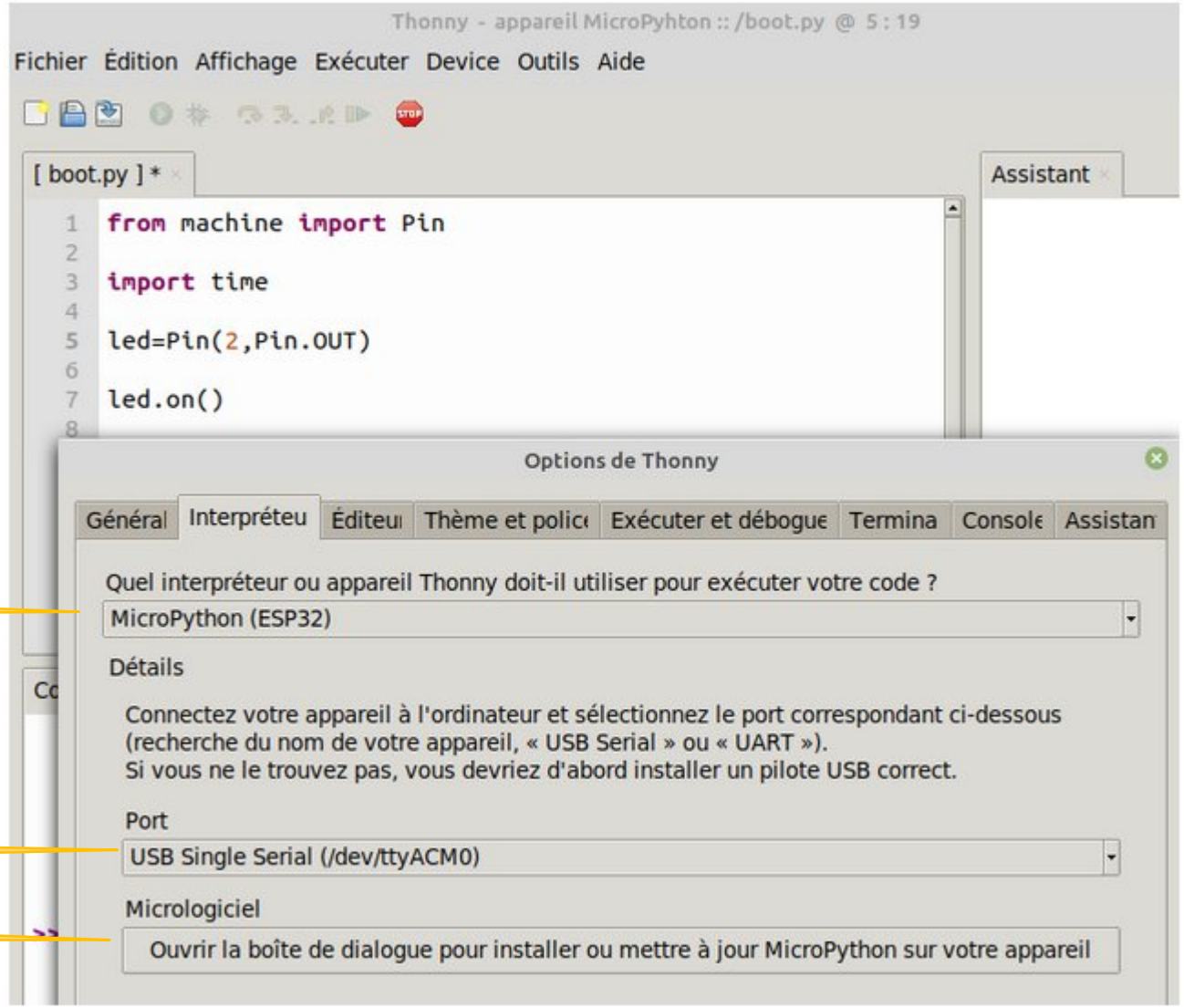
- Ensuite on déploie ce firmware (par exemple à partir d'un Raspberry)

```
> pip install esptool
> esptool.py --port /dev/ttyUSB0 erase_flash
> esptool.py --port /dev/ttyUSB0 --baud 115200 write_flash --flash_size=detect 0 esp8266-20170108-v1.8.7.bin
```

Vérifier que ce nom de fichier correspond (selon la version installée)

- Ou avec Thonny

Installation sur ESP (2/3)



Choix de la cible ESP

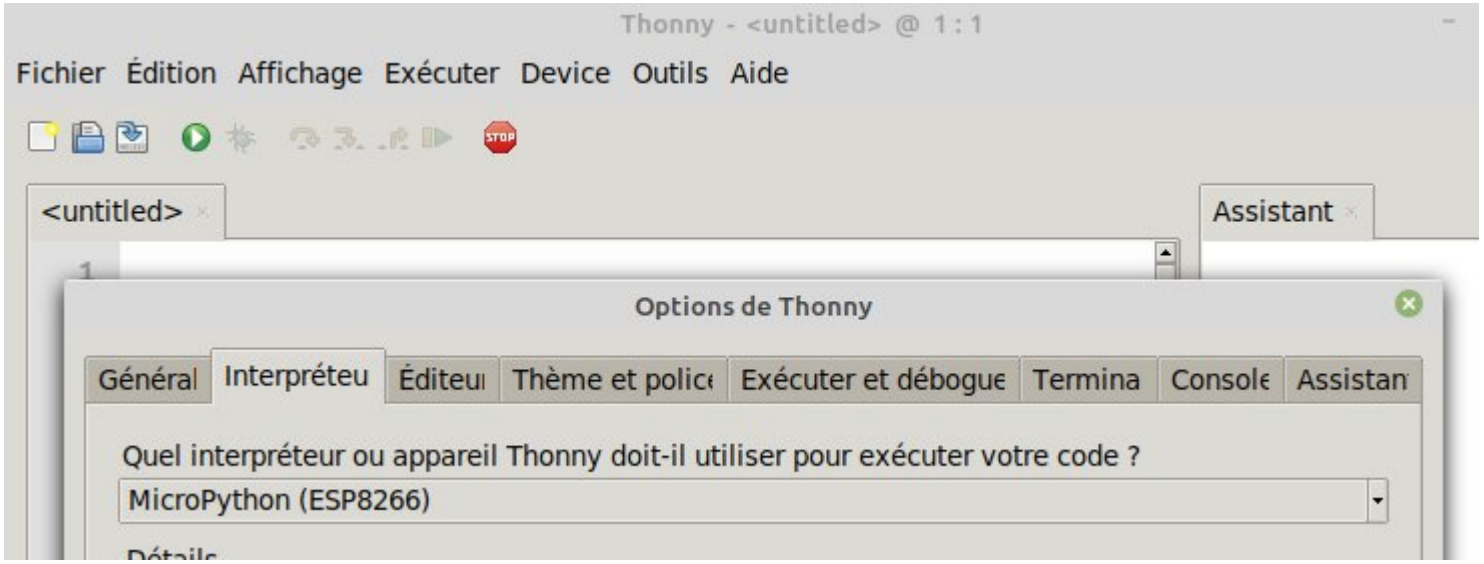
Vérifier le port USB

Enregistrement firmware ici

Installation sur ESP (3/3)

- NE PAS OUBLIER DE CHOISIR VOTRE ESP !

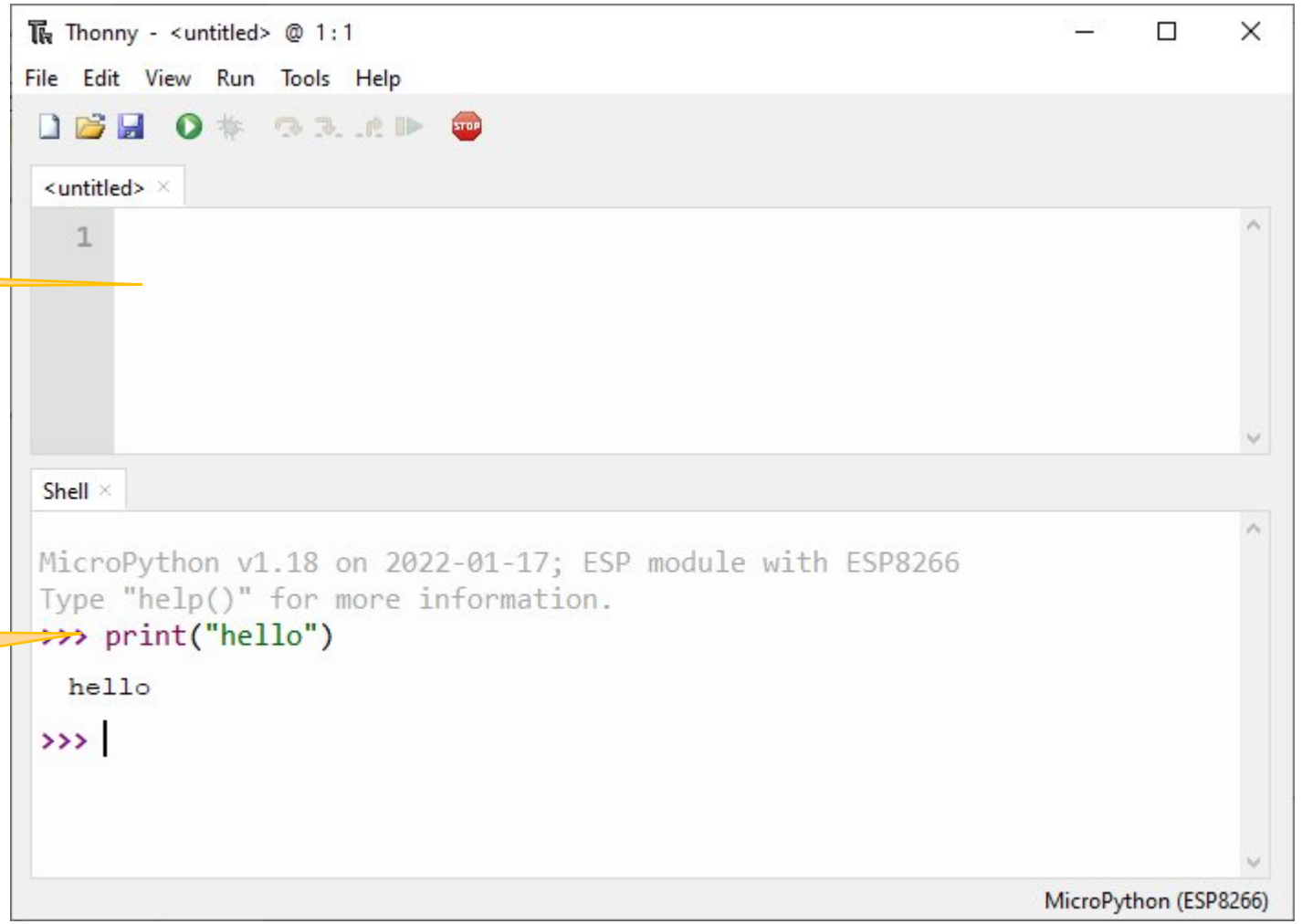
Choix ESP 8266



Utilisation avec l'éditeur Thonny

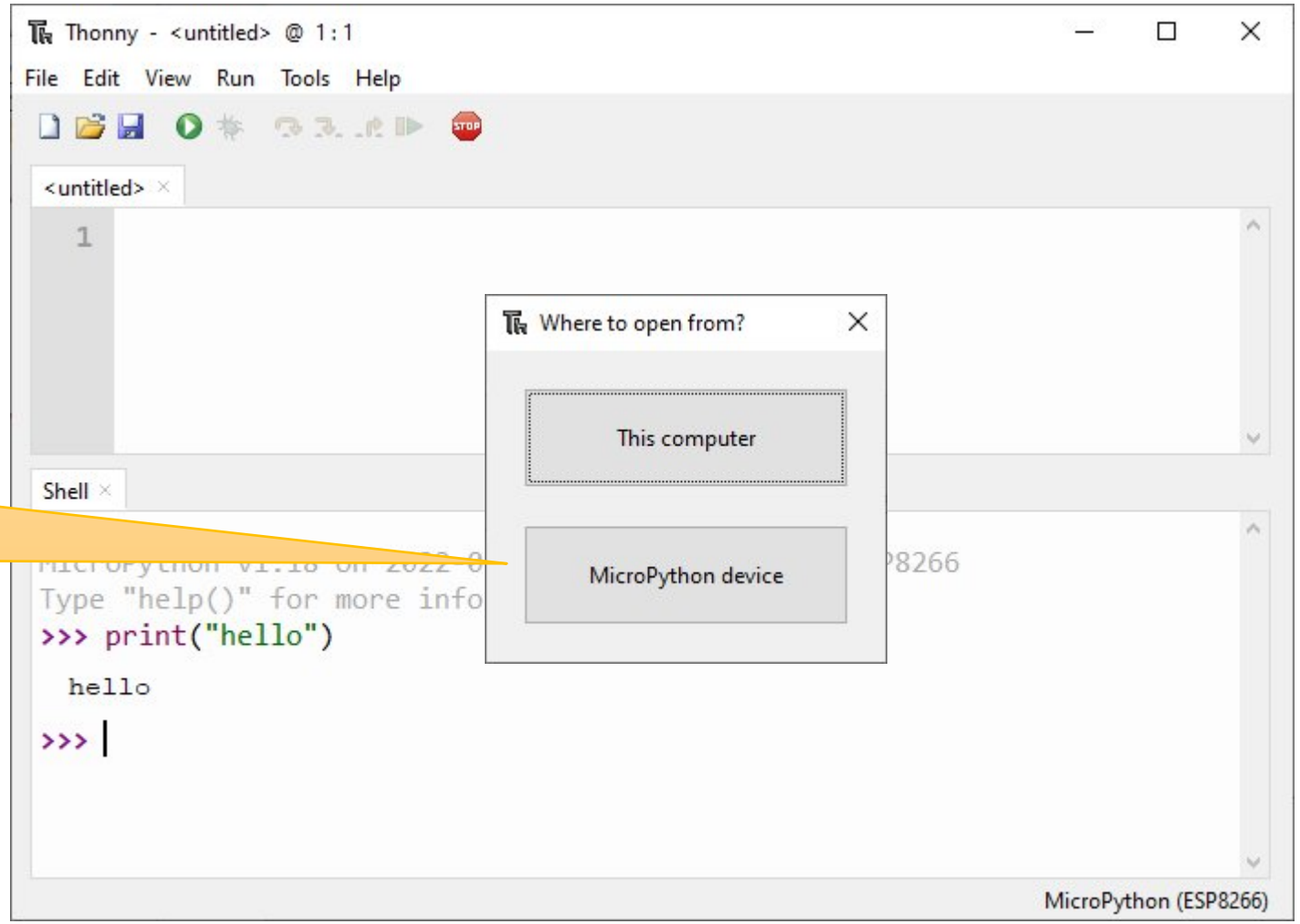
Editeur de fichier

Commandes interactives qui vont tourner sur l'ESP8266

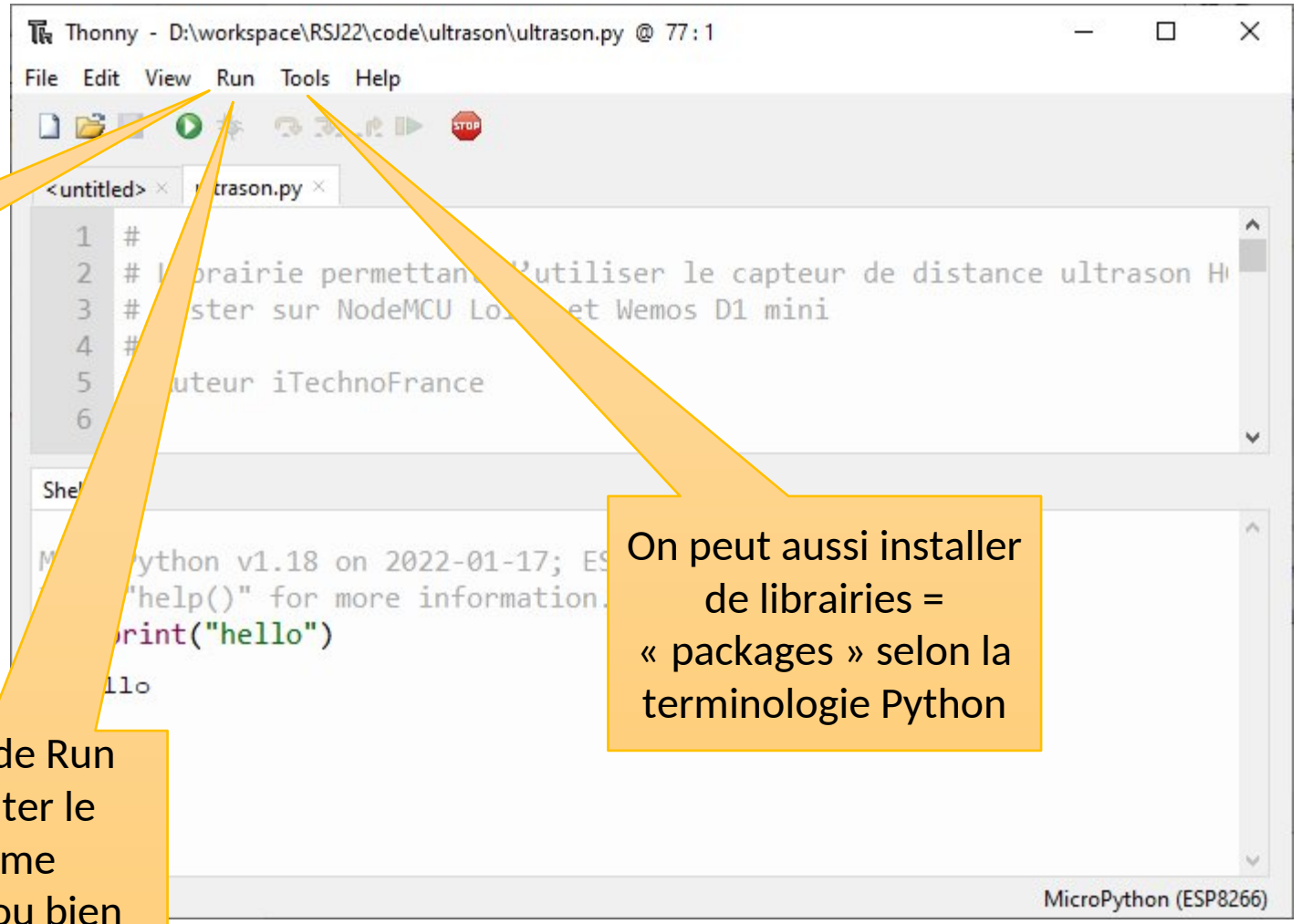


Travail à partir d'un fichier

On peut travailler avec des fichiers de la machine hôte ou bien directement avec des fichiers sur le ESP8266



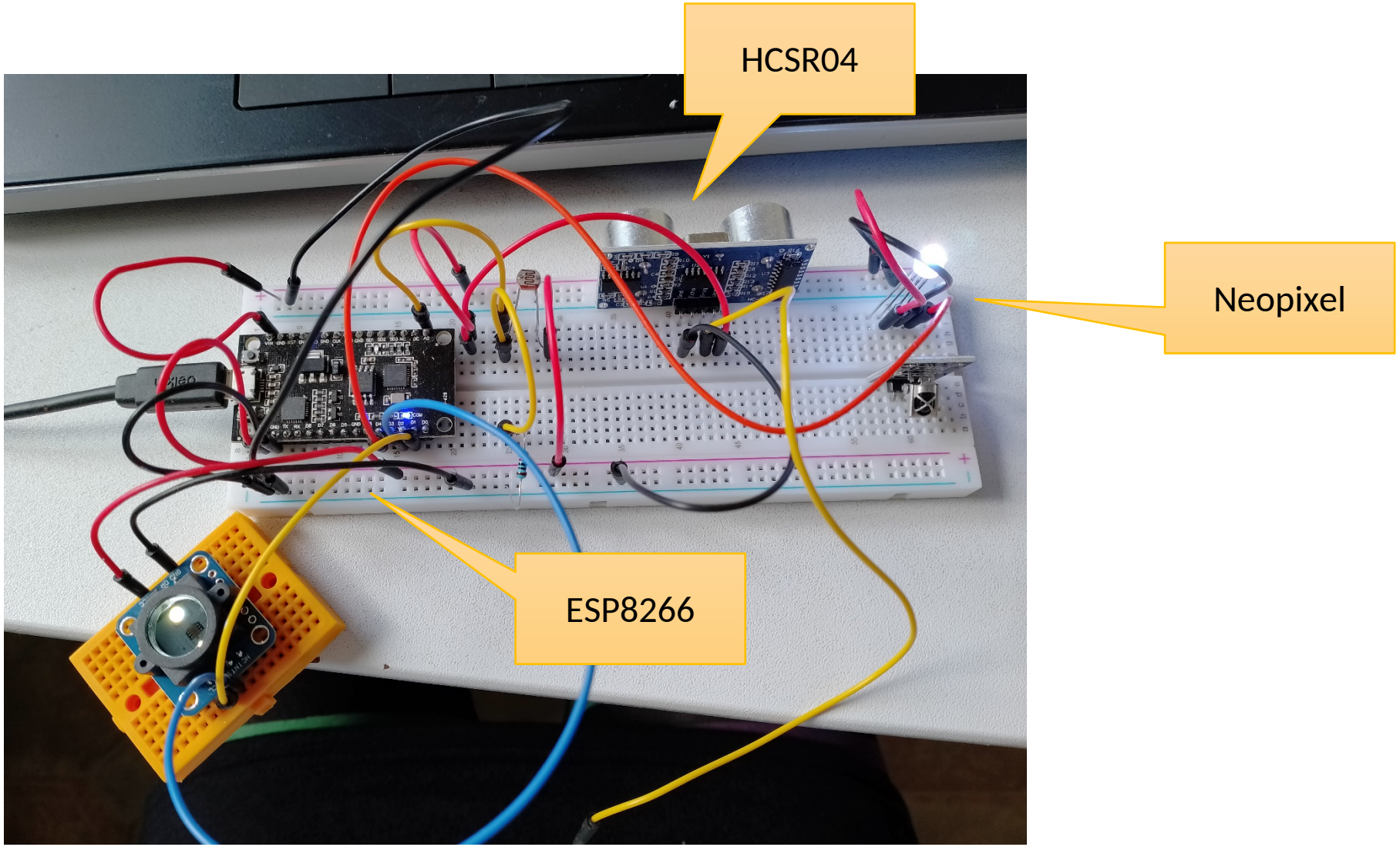
La commande Run va exécuter le programme écrit dans le fichier chargé



La commande Run peut exécuter le programme totalement ou bien en mode debug

On peut aussi installer de librairies = « packages » selon la terminologie Python

Le TP



Un programme typique

On définit les caractéristiques structurelles et fonctionnelles de notre module (ici le HCSR04) dans une classe

```
from machine import Pin, PWM
import machine, time

class HCSR04():
    def __init__(self, pin_echo, pin_trig):
        self.pin_echo = pin_echo
        self.pin_trig = Pin(pin_trig, Pin.OUT)  # definit la broche TRIGGER en
sortie
        self.pin_echo = Pin(pin_echo, Pin.IN)  # definit la broche ECHO en
entree
        self.pin_trig.off()  # broche TRIGGER niveau bas au repos
        # on definit un Timeout si le retour echo depasse 23.8 ms
        # 410 cm  * 2 (aller/retour) * 29.1 (vitesse du son 1cm = 29.1us)
        tmo = int(410 * 2 * 29.1)
        self.tmo = tmo

    def lecture_distance(self):
        self.pin_trig.off()  # broche TRIGGER niveau bas au repos
        time.sleep_us(2)  # on attend 20us
        self.pin_trig.on()  # broche TRIGGER niveau haut
        time.sleep_us(10)  # pendant 10us
        self.pin_trig.off()
        self.temps_distance = machine.time_pulse_us(self.pin_echo, 1, self.tmo)
        # print("temps_distance", self.temps_distance)
        if (self.temps_distance == -1):
            # Timeout, mesure distance > 4m
            self.calcul_distance_cm = -1
        else:
            # le temps que met l'echo est a diviser par 2 (aller-retour)
            # la vitesse du son est de 0.034320 cm/us (343.2 m/s)
            # pour 1 cm donc on obtient 29.1 us
            self.calcul_distance_cm = (self.temps_distance / 2) / 29.1
        return(self.calcul_distance_cm)
```

```
...
pin_echo = 4  # sortie D2 -> GPIO04
pin_trigger = 5  # sortie D1 -> GPIO05

hc_sr04 = HCSR04(pin_echo, pin_trigger)

while True:
    distance = hc_sr04.lecture_distance()
    if (distance == -1):
        print ("Mesure supérieure à 4 mètres")
    else:
        print ((distance), "cm")
    time.sleep(0.1)
```

Programme principal

Allumer des diodes colorées Neopixel

```
import machine, neopixel
import time
import random

np = neopixel.NeoPixel(machine.Pin(4), 1)

dt = 20

np[0] = (0, 0, 0)
np.write()
time.sleep_ms(1000)

print("green")
for i in range(255):
    np[0] = (i, 0, 0)
    np.write()
    time.sleep_ms(dt)

print("red")
for i in range(255):
    np[0] = (0, i, 0)
    np.write()
    time.sleep_ms(dt)

print("blue")
for i in range(255):
    np[0] = (0, 0, i)
    np.write()
    time.sleep_ms(dt)
```

La librairie Neopixel fournit toutes les commandes pour piloter les diodes Neopixel

Ainsi il suffit d'instancier un objet neopixel qui saura adresser les diodes Neopixel

```
print("yellow")
for i in range(255):
    np[0] = (i, i, 0)
    np.write()
    time.sleep_ms(dt)

print("cyan")
for i in range(255):
    np[0] = (i, 0, i)
    np.write()
    time.sleep_ms(dt)

print("violet")
for i in range(255):
    np[0] = (0, i, i)
    np.write()
    time.sleep_ms(dt)

np[0] = (0, 0, 0)
np.write()
```

Serveur Web

```
import socket
import select
import network
import esp
from machine import Pin, ADC

esp.osdebug(None)

import gc
gc.collect()

def web_page(client, commande):
    html = """
<!DOCTYPE html> <html> <head> <style> ... </style> \n
  <center> <h1>Robot Service Jeunesse</h1>  ... </center> \n
</head> \n
</html>
"""


    while True:
        try:
            n = len(html)
            ns = client.write(html)
            client.close()
            print("sending answers", n, ns)
            break
        except:
            pass
```

```
pot = ADC(0)

def cps(pot):
    a = 40
    b = 430
    v = pot.read()
    return (v - a)/(b - a)

def module_a():
    print(" lecture du module...")

def do_something_else():
     print("doing somethig else")
     print("CPS = ", cps(pot))
     print("A= ", module_a())
```

Définition des actions spécifiques dans le serveur, pour contrôler des modules

Définition de la page Web pour le client

```
ssid = 'RCO_123'              #Set your own
password = '12345678'     #Set your own password

ap = network.WLAN(network.AP_IF)
ap.active(False)
ap.active(True)
ap.config(essid=ssid, password=password)

while ap.active() == False:
  pass

print('Connection is successful')
print(ap.ifconfig())

i = 0
while True:
    print('Creating socket')
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        print('Binding socket on port ', 80)
        s.bind(('', 80))
        break
    except:
        s.close()

    if i > 10:
        print("timeout")
        break
    i += 1

s.listen(5)
print('Listening socket')
```

Établissement de la connexion

Lecture non-bloquante des données en provenance du client

Décodage des commandes du client

Rafraichissement de la page Web du client en fonction des commandes reçues

Lecture non-bloquante du Web ET des différents modules

```
def client_handler(commande):
    print("client_handler", commande)

def read(s):
    r, w, err = select.select((s,), (), (), 1)
    if not r: return
    for readable in r:
        client, addr = s.accept()
        print('Got data from %s' % str(addr))
        request = client.recv(2048).decode()
        request = str(request)
        l = len(request)
        pos = request.find("/?LED")
        if pos > 0:
            request = request[pos+7:]
            pos = request.find("&")
            commande = request[0:pos]
            print('len={} Content = [{}] '.format(l, request))

            web_page(client, int(commande))


            client_handler(commande)
        else:
            web_page(client, 0)
            print("no commande")

while True:
    read(s)
    do_something_else()
```

Atelier numérique
de Bures sur Yvette

microPython sur github anumby et arnaudrco

Développement voiture Anumby

https://github.com/anumby-source/developpement-voiture/wiki#developpement-voiture

Environnement IA pour K210 (MAIXPY)

https://github.com/anumby-source/developpement-voiture/wiki/d%C3%A9veloppement-IA-en-python

TP CROUS sur esp

https://github.com/arnaudrco/CROUS-micro-python/wiki

TP sur RASPERRYPICO

https://github.com/anumby-source/raspberry-pico/wiki

[...]