Internet des objets et Middleware

email : kadionik@enseirb-matmeca.fr
web : kadionik.enseirb-matmeca.fr

Patrice KADIONIK
ENSEIRB-MATMECA

HISTORIQUE

- V 1.0 05/22 : Création du document depuis IT361.
- V 1.1 01/23 : Changement de cible : Raspberry Pi Pico W.

CHAPITRE 1 :
L'INTERNET DES OBJETS
RAPPELS

INTERNET DES OBJETS

- L'IoT correspond à une opportunité répondant à un nouveau besoin du moment :
  - La mise en œuvre d'un ensemble de technologies déjà éprouvées : connectivité IP, Linux embarqué, Java embarqué, *middlewares*, *frameworks*…
  - Un saut technologique : réseaux sans fil LPWAN (LoRa, Sigfox).
  - Un nouveau besoin qui met en œuvre ces technologies.

- Il s'était déjà produit la même chose avec le Web début années 1990. On peut situer le démarrage de l'IoT dans les années 2005-2010…

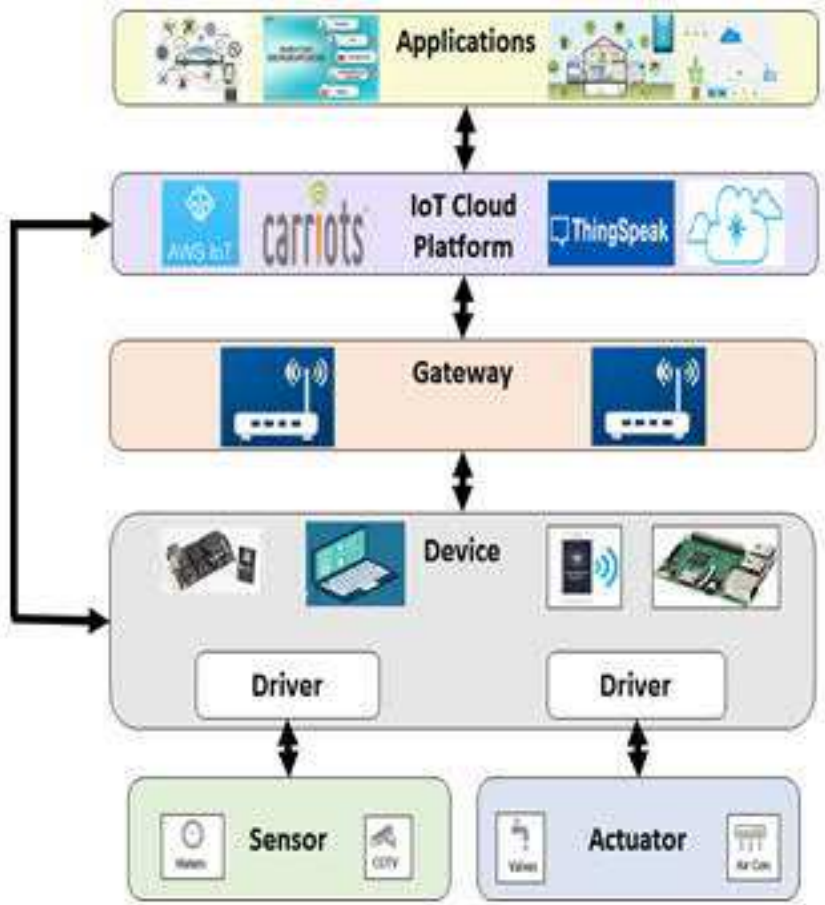
IoT=Internet of Things
LPWAN=Low Power Wide Area Network

INTERNET DES OBJETS

- Depuis quelques années (2010-2015), le développement des objets connectés s'est grandement simplifié car :
  - Il fait appel à des réseaux et bus standardisés (LoRa, ZigBee (IEEE 802.15.4), BLE…).
  - La connectivité à Internet est directement intégrée dans les objets connectés.
  - Il fait appel à des logiciels libres.
  - Il met en œuvre des *middlewares* pour cacher les spécificités matérielles et logicielles des objets connectés.
  - Il met en œuvre des canevas de programmation ou *frameworks* pour faciliter le développement logiciel et le déploiement des objets connectés.

BLE=Bluetooth Low Energy

INTERNET DES OBJETS



Architecture de l'IoT

INTERNET DES OBJETS

- Les capteurs (*sensor*) et actionneurs (*actuator*) permettent d'interagir avec le monde réel : acquisition de grandeurs physiques en entrée (volume, vitesse, température…) et pilotage de grandeurs en sortie.

- L'objet (*device*) est l'équipement HW/SW qui intègre les capteurs et actionneurs. C'est l'intelligence locale (*edge computing*) de contrôle. L'objet s'interface à une passerelle vers Internet par une connectivité Internet ou non.

- La passerelle (*gateway*) permet de se connecter à Internet.

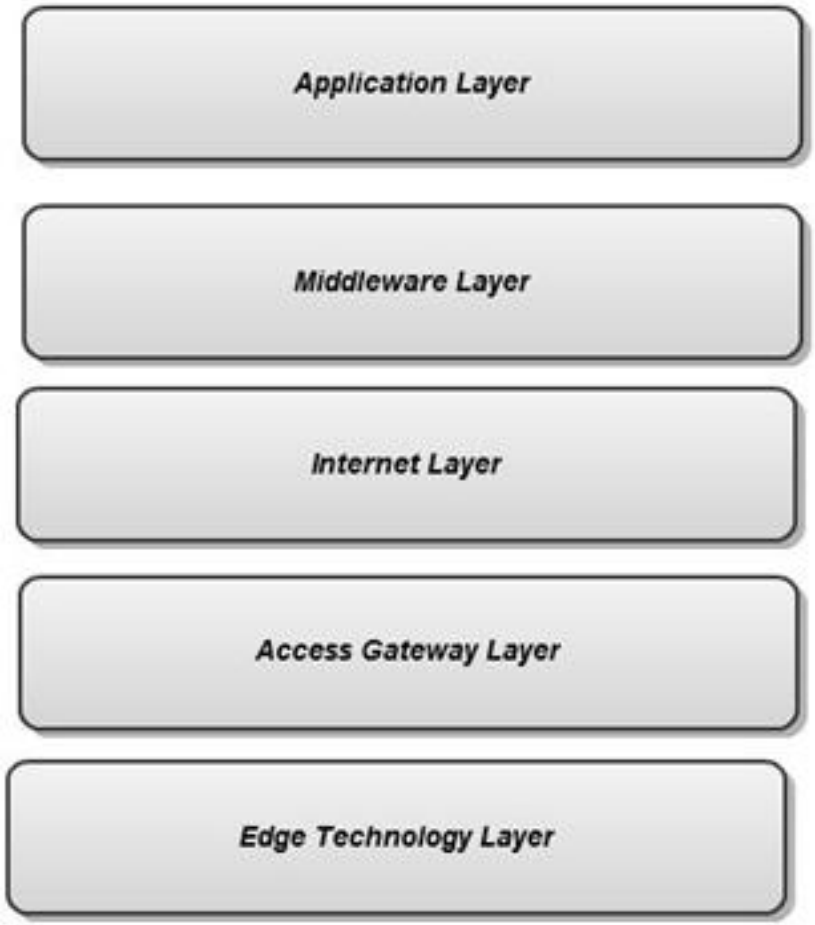
INTERNET DES OBJETS

- Le *cloud* IoT va collecter les données des différents objets et les contrôler. Il fournit les données aux applications IoT. Cela s'intègre dans une plateforme de développement (*framework*) fournissant un *middleware* (*middleware layer*).

- Les applications IoT implémentent les services IoT (*smart city*, *smart industry*…).

INTERNET DES OBJETS

- On peut aussi raisonner en couches dans l'architecture IoT :



INTERNET DES OBJETS

- Couche *edge technology* : c'est la couche matérielle qui traite les entrées/sorties et gère les communications.
- Couche *access gateway* : c'est la couche qui gère le transport des données et leur routage vers Internet.
- Couche *middleware* : c'est la couche qui gère le traitement de données et fournit aux applications l'accès aux objets.
- Couche *application* : C'est la couche qui fournit différents services aux applications IoT.

CHAPITRE 2 :
MIDDLEWARE, VOUS AVEZ DIT MIDDLEWARES ?

MIDDLEWARES

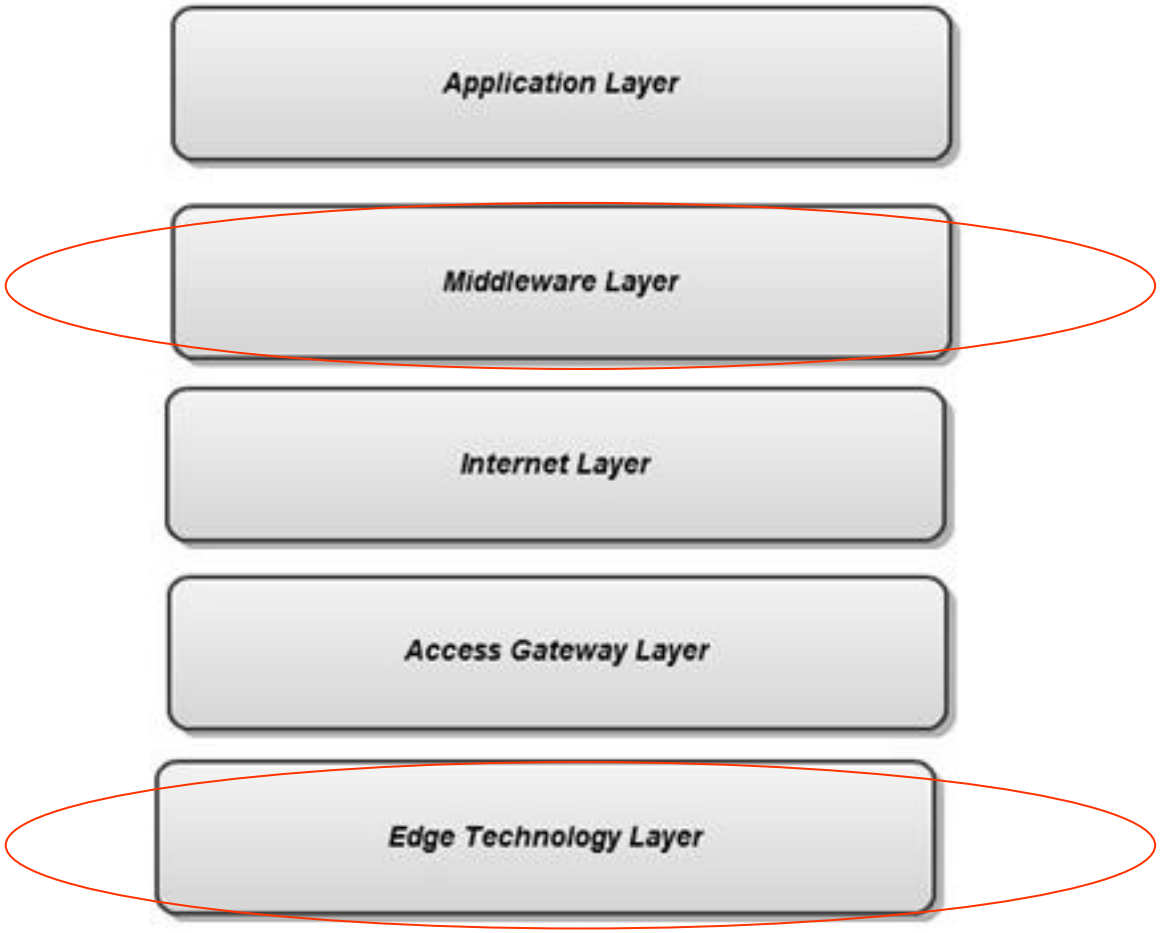
- L'intergiciel ou *middleware* dans l'IoT est un logiciel qui permet l'interface entre les différents composants.
- Le *middleware* rend homogène ce qui ne l'est pas et permet la communication entre des éléments qui ne le pourraient pas.
- Différents *middlewares* sont intégrés dans l'architecture IoT et fournissent une connectivité pour les capteurs mais aussi pour les couches applications.

MIDDLEWARES

- La couche *middleware* dans l'IoT est plutôt une couche haute dans l'architecture IoT.

- Néanmoins, il existe aussi du *middleware* plus « intimiste » dans la couche *edge technology* pour développer l'application embarquée dans l'objet IoT.

- On a donc :
  - Des *middlewares* haut niveau. Pour l'ingénieur en Génie Logiciel.
  - Des *middlewares* bas niveau. Pour l'ingénieur de l'embarqué.



middleware haut niveau

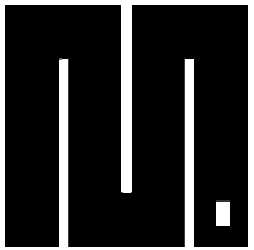
middleware bas niveau

MIDDLEWARES HAUT NIVEAU

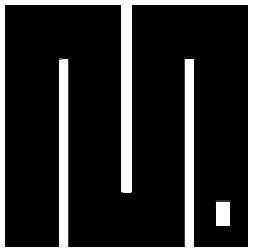
- On pourra citer comme *middlewares* de haut niveau :
  - AWS IoT Core : https://aws.amazon.com/fr/iot-core/
  - Microsoft Azure IoT Hub : https://azure.microsoft.com/fr-fr/services/iot-hub/
  - Google Cloud IoT Solutions : https://cloud.google.com/solutions/iot
  - Oracle IoT Platform : https://www.oracle.com/fr/internet-of-things/
  - IBM Watson IoT Platform : https://internetofthings.ibmcloud.com/
  - Mais aussi Kaa, ThingSpeak, Temboo, Samsara, Particle…
  - …

MIDDLEWARES BAS NIVEAU

- On pourra citer comme *middlewares* de bas niveau pour l'interaction avec un objet connecté :
  - TinyOS avec le langage C. TinyOS est un système d'exploitation de type événementiel qui a été développé pour déployer au départ des réseaux de capteurs sans fil.
    - http://www.tinyos.net/
  - MicroPython avec le langage Python. MicroPython est un interpréteur Python réservé aux objets connectés conçus autour d'un microcontrôleur.
    - https://micropython.org/



CHAPITRE 3 : MICROPYTHON



MICROPYTHON

- MicroPython est une implémentation de Python 3 adaptée aux objets connectés et aux petits systèmes embarqués.
- MicroPython inclut une bonne partie de Python mais pas tout à cause des limites de l'objet connecté généralement construit autour d'un microcontrôleur.
- MicroPython reste néanmoins compatible avec Python.
- Il tourne avec juste 256 Ko de RAM pour le code et 16 Ko de RAM pour les données.

MICROPYTHON

- MicroPython permet d'unifier les plateformes matérielles hétérogènes.
- C'est la même philosophie que le langage Java mais adapté à l'embarqué.
- L'objet embarqué l'interpréteur MicroPython appelé REPL (*Read Evaluate Print Loop*).
- REPL permet d'exécuter une instruction Python ou bien un fichier source Python. Par la suite, on fera la confusion MicroPython et Python par soucis de simplification…

MICROPYTHON

- Globalement, MicroPython fournit des bibliothèques qui reprennent les fonctionnalités générales des bibliothèques Python (os, time…).

- Mais généralement, pour réduite l'empreinte mémoire, les bibliothèques MicroPython implémentent en réalité un sous-ensemble des bibliothèques Python.

- Suivant le portage spécifique pour un processeur donné (cible), ce sous-ensemble est plus ou moins important.

BIBLIOTHEQUES MICROPYTHON

- MicroPython supporte les bibliothèques **standards** suivantes :
  - array – arrays of numeric data
  - binascii – binary/ASCII conversions
  - builtins – builtin functions and exceptions
  - cmath – mathematical functions for complex numbers
  - collections – collection and container types
  - errno – system error codes
  - gc – control the garbage collector
  - hashlib – hashing algorithms
  - heapq – heap queue algorithm
  - io – input/output streams
  - json – JSON encoding and decoding
  - math – mathematical functions

BIBLIOTHEQUES MICROPYTHON

- MicroPython supporte les bibliothèques **standards** suivantes :
  - os – basic “operating system” services
  - random – generate random numbers
  - re – simple regular expressions
  - select – wait for events on a set of streams
  - socket – socket module
  - ssl – SSL/TLS module
  - struct – pack and unpack primitive data types
  - sys – system specific functions
  - time – time related functions
  - uasyncio — asynchronous I/O scheduler
  - zlib – zlib decompression
  - _thread – multithreading support

BIBLIOTHEQUES MICROPYTHON

- MicroPython supporte les bibliothèques **spécifiques** suivantes :
  - bluetooth — low-level Bluetooth
  - btree – simple BTree database
  - cryptolib – cryptographic ciphers
  - framebuf — frame buffer manipulation
  - machine — functions related to the hardware
  - micropython – access and control MicroPython internals
  - neopixel — control of WS2812 / NeoPixel LEDs
  - network — network configuration
  - uctypes – access binary data in a structured way

BIBLIOTHEQUES MICROPYTHON

- MicroPython supporte enfin des bibliothèques **liées au portage** sur une cible donnée :
  - pyb : spécifique à la carte Pyboard
  - pycom : spécifique à la carte Wipy
  - esp : spécifique à une cible ESP8266 and ESP32
  - esp32 : spécifique à une cible ESP32
  - rp2 : spécifique à une cible RPi-Pico

BIBLIOTHEQUES MICROPYTHON

- Les bibliothèques MicroPython sont documentées et décrites à l'adresse :
  - https://docs.micropython.org/en/latest/library/index.html
- Il n'y a pas de difficultés de programmation en MicroPython si l'on sait programmer en Python.
- L'intérêt est ailleurs :
  - Unifier les cibles matérielles.
  - Accéder de façon très simple au matériel et aux E/S.
  - En conséquence, tout le monde peut piloter de l'électronique si l'on possède une cible MicroPython.

HELLO WORLD MICROPYTHON

- Le « Hello World » en MicroPython s'écrit… :

```
print('Hello World!')
```

- Pas de différences avec Python !

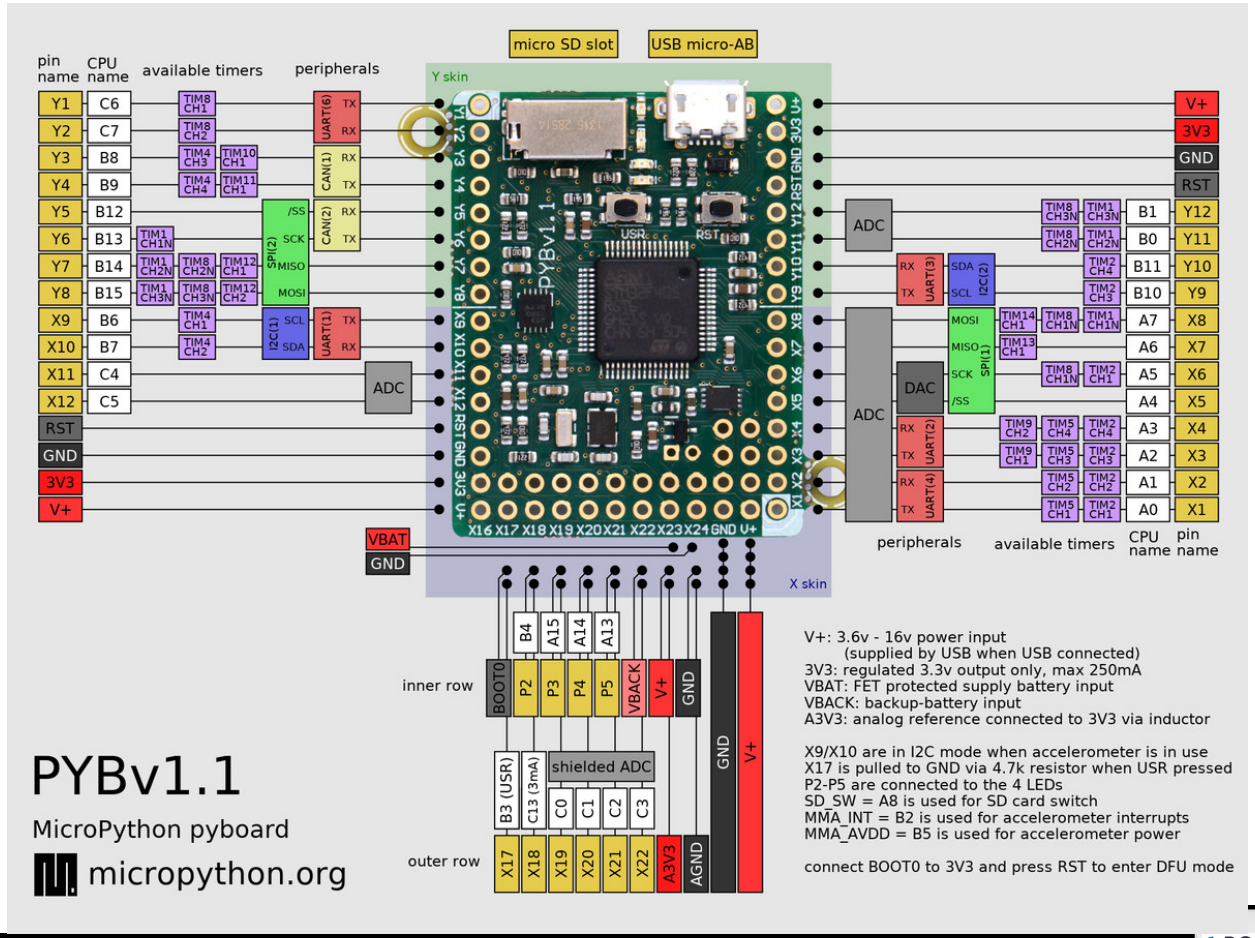
CIBLES MICROPYTHON

- Il existe des plateformes matérielles (libres) pour faire du MicroPython.

- Ces cibles peuvent être à la base d'un objet connecté de bien plus petite dimension et puissance si l'on avait utilisé une carte Raspberry Pi.

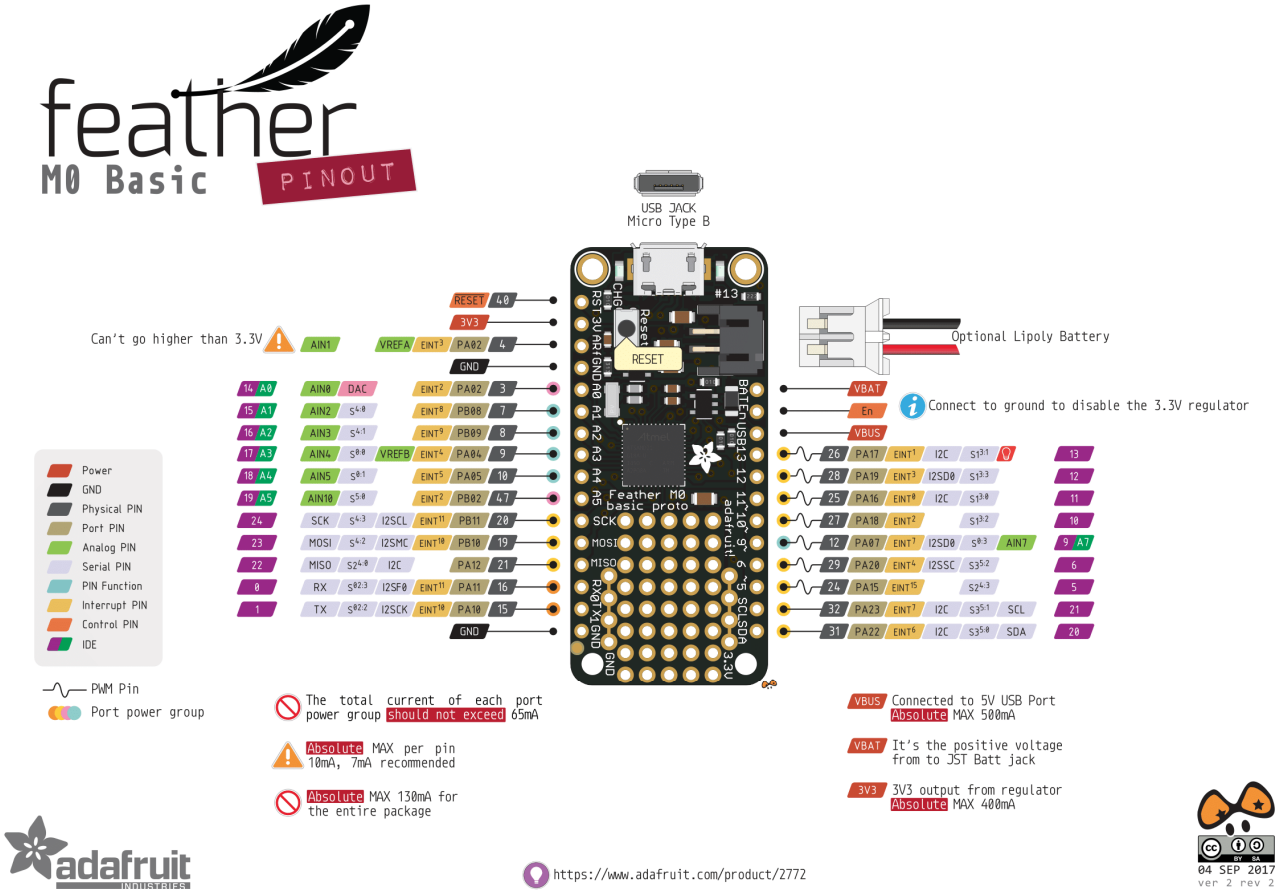
CIBLES MICROPYTHON

- Carte pyboard (première cible MicroPython) :
  - https://www.lextronic.fr/pyboard-micropython-10102



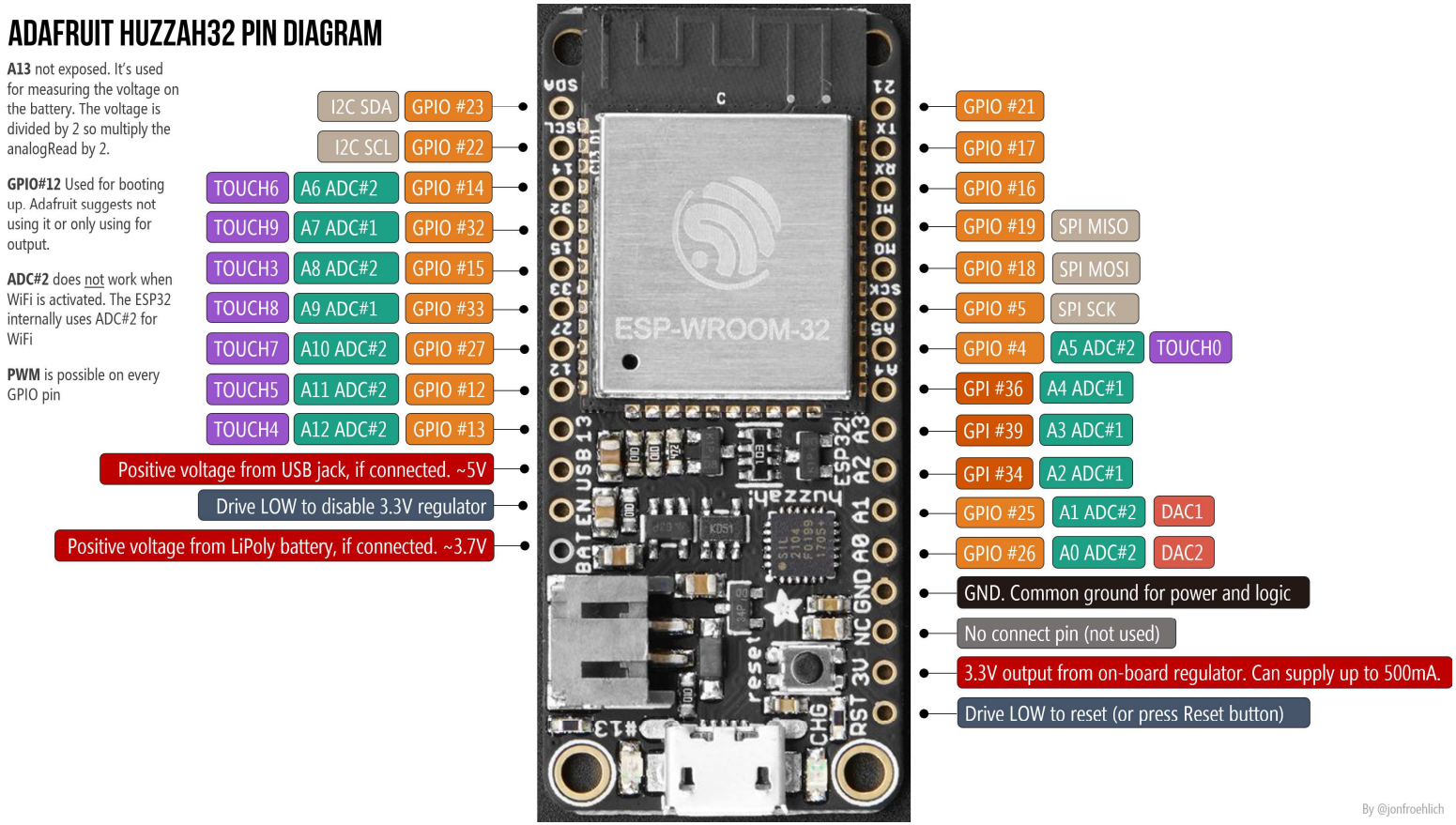
CIBLES MICROPYTHON

- Carte à base d'ESP8266 : carte Adafruit Feather :
  - https://www.lextronic.fr/cartes-adafruit-feather-9700



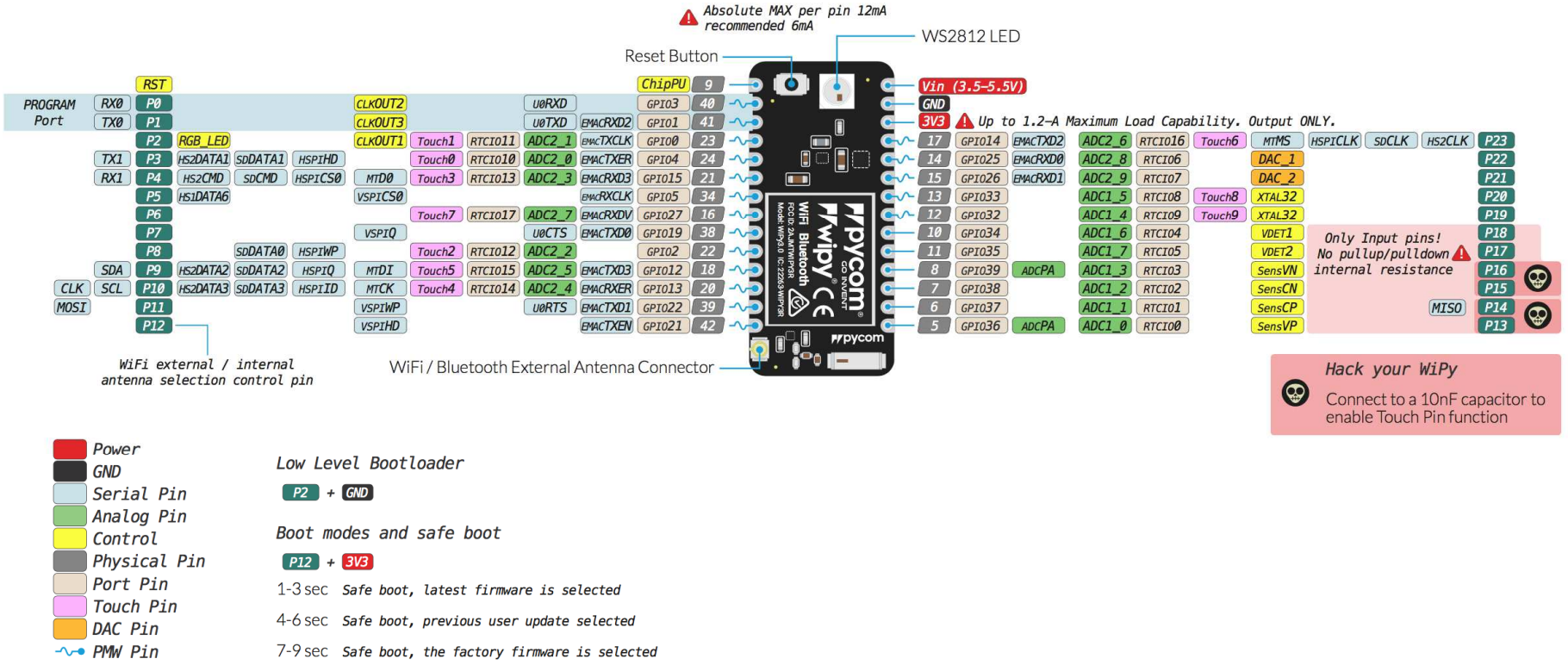
CIBLES MICROPYTHON

- Carte à base d'ESP32 : carte Adafruit Huzzah 32 :
  - https://www.lextronic.fr/module-feather-huzzah32-esp32-31584.html



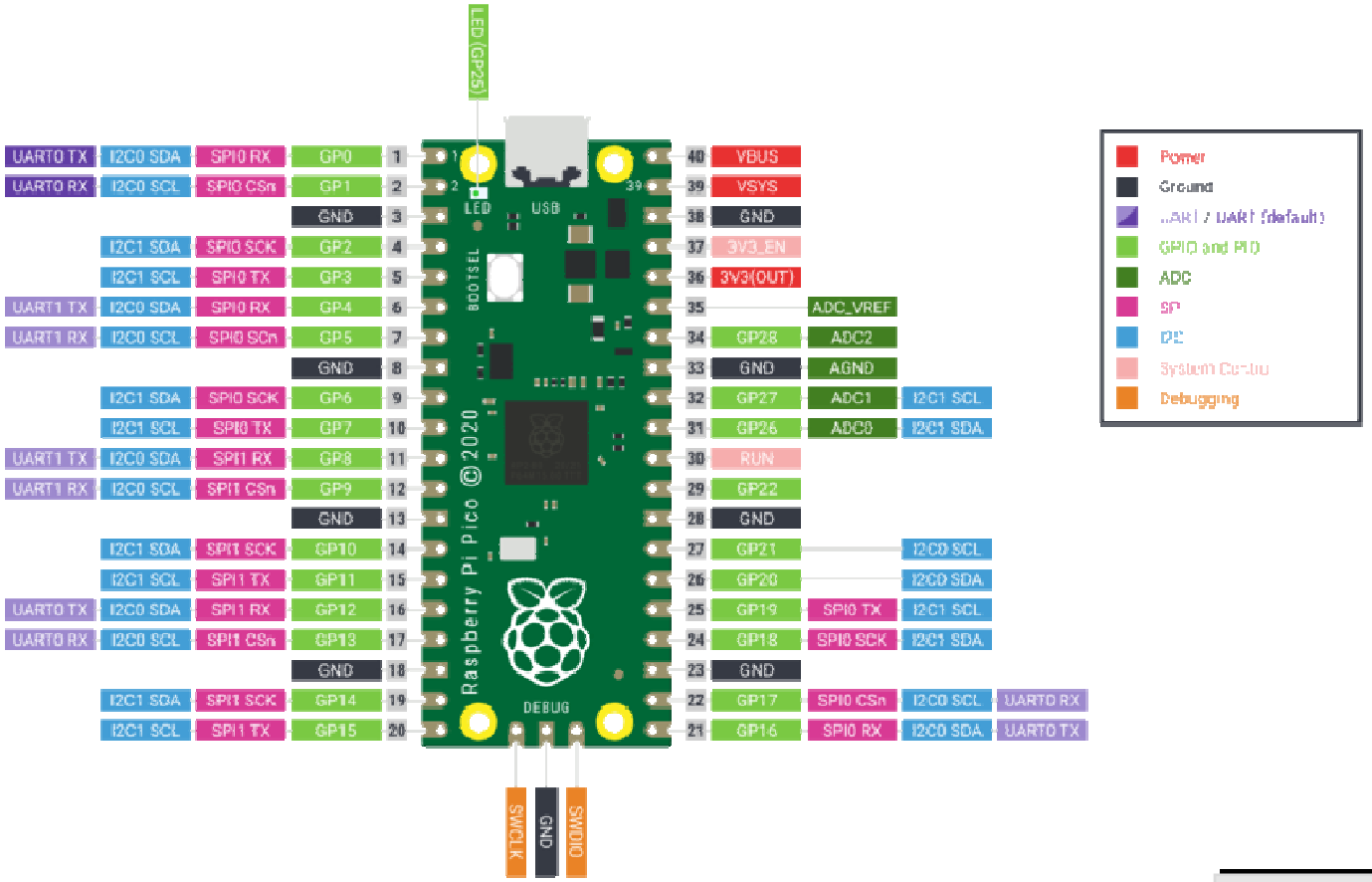
CIBLES MICROPYTHON

- Carte Wipy :
  - https://www.lextronic.fr/module-wipy-3-0-35061.html



CIBLES MICROPYTHON

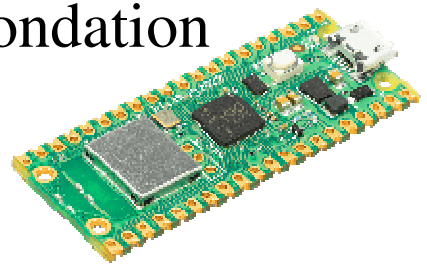
- Carte Raspberry Pi Pico :
  - https://www.lextronic.fr/carte-raspberry-pi-pico-61989.html



CARTE CIBLE RASPBERRY PI PICO

- La carte cible Raspberry Pi Pico W (RPi-Pico W) est la carte que nous utiliserons en TP pour réaliser un objet connecté mettant en œuvre le *middleware* bas niveau MicroPython :

- La carte RPi-Pico W a été développée par la fondation Raspberry :
  - https://www.raspberrypi.com



- La documentation de la carte est disponible à l'adresse :
  - https://www.raspberrypi.com/products/raspberry-pi-pico/
  - https://docs.micropython.org/en/latest/rp2/quickref.html

CARTE CIBLE RPI-PICO ET MICROPYTHON

- Machine :

```
import machine

help(machine)               # display all members from the machine module
machine.freq()              # get the CPU frequency
machine.freq(240000000)     # set the CPU frequency to 240 MHz
```

- Temps :

```
import time

time.sleep(1)           # sleep for 1 second
time.sleep_ms(500)      # sleep for 500 milliseconds
time.sleep_us(10)       # sleep for 10 microseconds
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- Pins et GPIO :

```
from machine import Pin

p0 = Pin(0, Pin.OUT)     # create output pin on GPIO0
p0.on()                  # set pin to "on" (high) level
p0.off()                 # set pin to "off" (low) level
p0.value(1)              # set pin to on/high

p2 = Pin(2, Pin.IN)      # create input pin on GPIO2
print(p2.value())        # get value, 0 or 1

p4 = Pin(4, Pin.IN, Pin.PULL_UP)   # enable internal pull-up resistor
p5 = Pin(5, Pin.OUT, value=1)      # set pin high on creation
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- ADC. Conversion Analogique/Numérique :

```
from machine import ADC, Pin

adc = ADC(Pin(26))      # create ADC object on ADC pin
adc.read_u16()          # read value, 0-65535 across voltage range 0.0v - 3.3v
```

- PWM :

```
from machine import Pin, PWM

pwm0 = PWM(Pin(0))       # create PWM object from a pin
pwm0.freq()              # get current frequency
pwm0.freq(1000)          # set frequency
pwm0.duty_u16()          # get current duty cycle, range 0-65535
pwm0.duty_u16(200)       # set duty cycle, range 0-65535
pwm0.deinit()            # turn off PWM on the pin
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- UART. Liaison série asynchrone RS232 Tx/Rx :

```
from machine import UART, Pin

uart1 = UART(1, baudrate=9600, tx=Pin(4), rx=Pin(5))
uart1.write('hello')  # write 5 bytes
uart1.read(5)         # read up to 5 bytes
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- Bus I2C. Liaison série synchrone SDA/SCL:

```
from machine import Pin, SoftI2C, I2C

i2c = SoftI2C(scl=Pin(5), sda=Pin(4), freq=100_000)          # Soft I2C

i2c = I2C(0)   # default assignment: scl=Pin(9), sda=Pin(8) # Hard I2C
i2c = I2C(1, scl=Pin(3), sda=Pin(2), freq=400_000)


i2c.scan()               # scan for devices

i2c.readfrom(0x3a, 4)    # read 4 bytes from device with address 0x3a
i2c.writeto(0x3a, '12') # write '12' to device with address 0x3a

buf = bytearray(10)      # create a buffer with 10 bytes
i2c.writeto(0x3a, buf)   # write the given buffer to the peripheral
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- RTC. Horloge Temps Réel :

```
from machine import RTC

rtc = RTC()
rtc.datetime((2017, 8, 23, 2, 12, 48, 0, 0)) # set a specific date and
                                             # time, eg. 2017/8/23 1:12:48
rtc.datetime() # get date and time
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- Bus 1 fil. Connexion d'un capteur de température DS1820 :

```
from machine import Pin
import onewire

ow = onewire.OneWire(Pin(12)) # create a OneWire bus on GPIO12
ow.scan()                     # return a list of devices on the bus
ow.reset()                    # reset the bus
ow.readbyte()                 # read a byte
ow.writebyte(0x12)            # write a byte on the bus

import time, ds18x20

ds = ds18x20.DS18X20(ow)
roms = ds.scan()
ds.convert_temp()
time.sleep_ms(750)
for rom in roms:
    print(ds.read_temp(rom))
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- Accès Wifi :

```
import machine
import network
import socket
import time

WIFI_SSID = "wifinetwork"
WIFI_PASSWORD = "wifipassword"

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(WIFI_SSID, WIFI_PASSWORD)
while wlan.isconnected() == False:
    print('Waiting for connection...')
    time.sleep(1)

print("Connected to Wifi " + WIFI_SSID)
print(wlan.ifconfig())
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- MQTT : https://docs.pycom.io/tutorials/networkprotocols/mqtt/

```
import machine
import network
import socket
import time
from mqtt import MQTTClient

WIFI_SSID = "wifinetwork"
WIFI_PASSWORD = "wifipassword"

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(WIFI_SSID, WIFI_PASSWORD)
while wlan.isconnected() == False:
    print('Waiting for connection...')
    time.sleep(1)
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

- MQTT : https://docs.pycom.io/tutorials/networkprotocols/mqtt/

```
client = MQTTClient("device_id", "io.adafruit.com",user="your_username",
   password="your_api_key", port=1883) # Ne pas changer "device_id"

client.connect()

while True:
    print("Sending ON")
    client.publish(topic="youraccount/feeds/lights", msg="ON")
    time.sleep(1)

    print("Sending OFF")
    client.publish(topic="youraccount/feeds/lights", msg="OFF")
    time.sleep(1)
```

CARTE CIBLE RPI-PICO ET MICROPYTHON

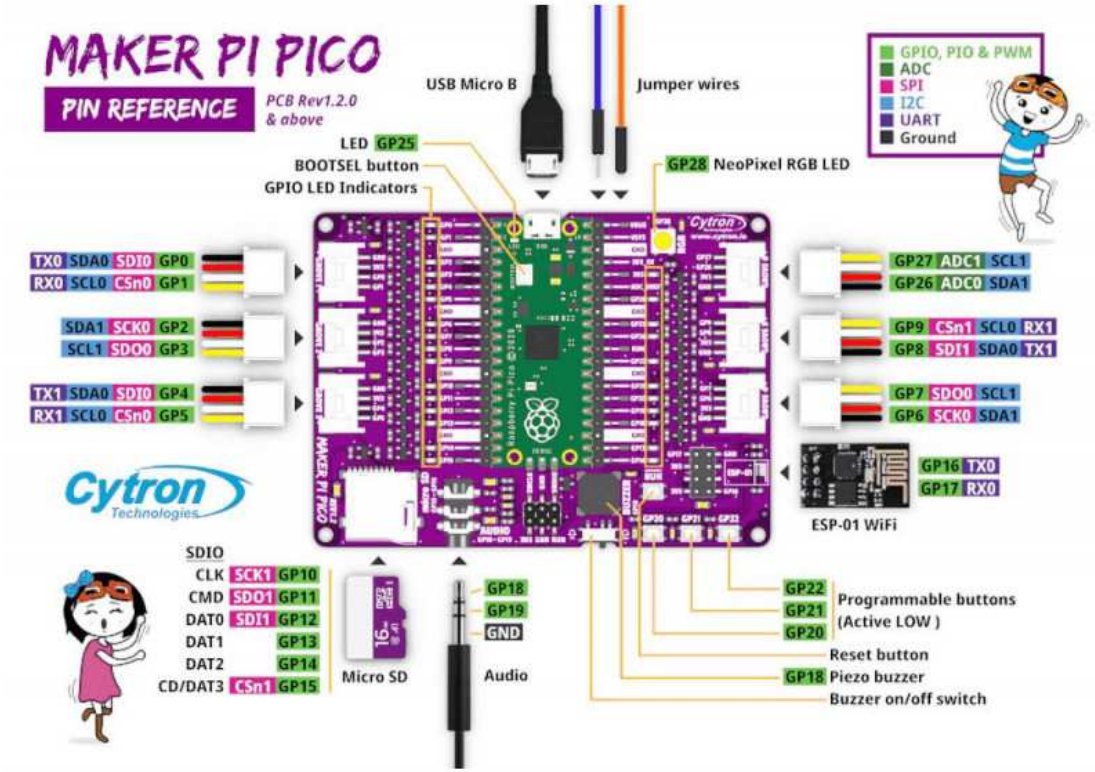
- MQTT : https://docs.pycom.io/tutorials/networkprotocols/mqtt/
- Que fait ce programme ?

CARTE CIBLE RPI-PICO ET MICROPYTHON

- La carte RPi-Pico W est installée sur une carte support : la carte *Maker Pi Pico*.

- La carte *Maker Pi Pico* permet d'avoir accès aux entrées/sorties et à quelques périphériques :
  - 3 boutons poussoirs (GP20-22).
  - 1 led RVB compatible Neopixel (GP28).
  - 1 *buzzer* Piezzo (GP18) désactivable à l'aide d'un mini interrupteur.
  - 1 jack audio femelle Stéréo 3,5 mm (GP18-19).
  - 1 slot pour ESP-01 (GP16, 17).
  - 1 slot microSD(GP10-15).
  - 6 connecteurs Grove.

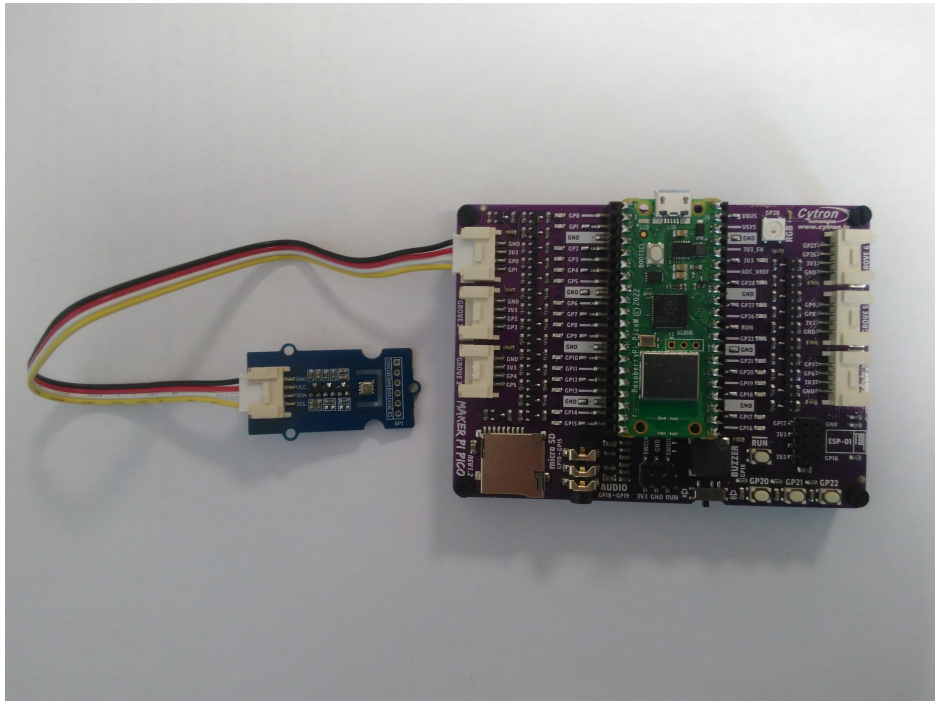
CARTE CIBLE RPI-PICO ET MICROPYTHON



CARTE CIBLE RPI-PICO ET MICROPYTHON

- On connectera alors par l'interface Grove un capteur universel I2C de température, pression, humidité et de gaz : le BME680.

- L'interface Grove est une interface universelle très utilisée par les *makers*.



CARTE CIBLE RPI-PICO ET MICROPYTHON

- La lecture de la température et de la pression de la carte Pysense se fait comme suit :

```
from machine import *
from time import *
import bme680

i2c = I2C(0, scl=Pin(1), sda=Pin(0), freq=400_000) # I2C #0
print("Adresses I2C utilisees : " + str(i2c.scan()) + "\n")

bme = bme680.BME680_I2C(i2c=i2c)

while True:
    temp = str(round(bme.temperature, 1)) + ' C'
    pres = str(round(bme.pressure, 1)) + ' hPa'
    print('Temperature :', temp)
    print('Pression :', pres)
    sleep(5)
```

CHAPITRE 4 :
MISE EN ŒUVRE DE MICROPYTHON

MISE EN ŒUVRE DE MICROPYTHON

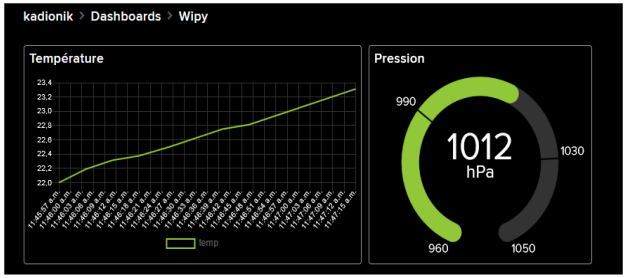
- La mise en œuvre de MicroPython sera abordée durant les travaux pratiques
- La mise en œuvre de MicroPython se fera avec la carte cible RPi-Pico W.
- La carte RPi-Pico W sera programmée dans l'environnement MicroPython.

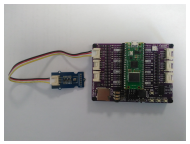
MISE EN ŒUVRE DE MICROPYTHON

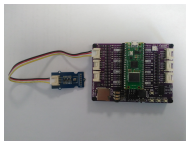
- La carte RPi-Pico W sera connectée à Internet via une base Wifi construite à l'aide d'une carte Raspberry Pi.
- Les données pression et température de la carte RPi-Pico W seront envoyées par MQTT vers le site io.adafruit.com qui fournit gratuitement un *broker* MQTT (l'usage doit être parcimonieux).
- Le site adafruit.com est un site de vente d'équipements électroniques pour les *makers*.
- Un *dashboard* sera construit pour afficher température et pression sous forme de *time series*.

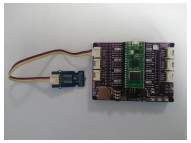
MISE EN ŒUVRE DE MICROPYTHON

Dashboard io.adafruit.com



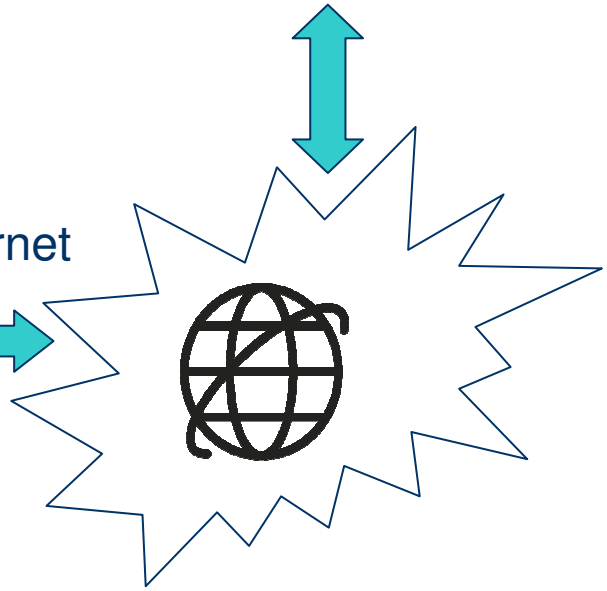






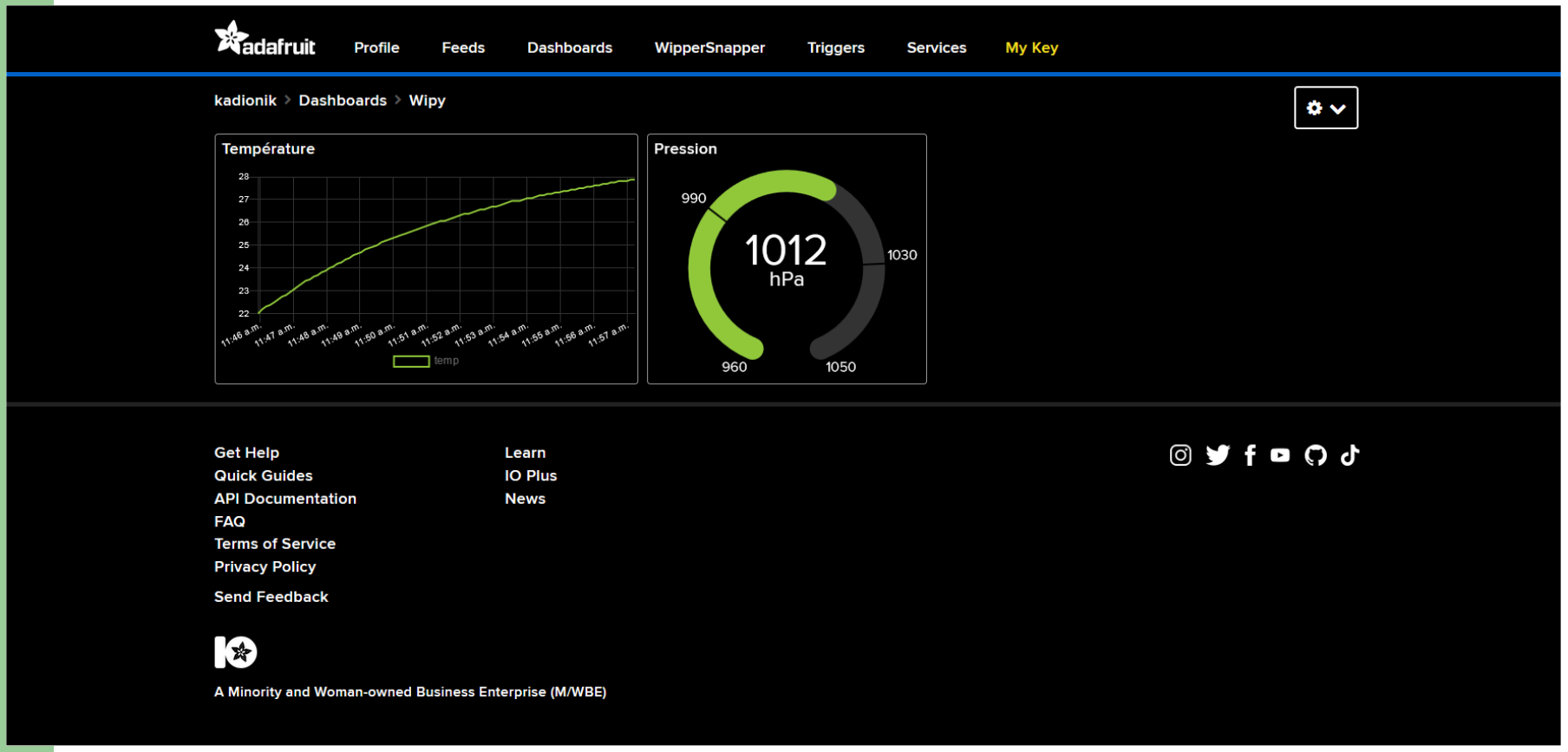
Wifi

Ethernet



Internet

MISE EN ŒUVRE DE MICROPYTHON



CONCLUSION

CONCLUSION

- Nous avons vu l'importance des *middlewares* dans l'IoT.

- Nous avons pu distinguer 2 types de *middlewares* :
  - *Middlewares* de haut niveau : dans l'infrastructure pour homogénéiser les flux et formats de données issus des objets. Indépendance vis-à-vis des données pour une meilleure interopérabilité.
  - *Middlewares* de bas niveau : dans l'objet pour homogénéiser le matériel et les E/S. Indépendance vis-à-vis du matériel pour une meilleure interopérabilité.

CONCLUSION

- Nous avons pu voir un *middleware* de bas niveau, MicroPython et sa mise en œuvre dans un objet connecté à base d'une carte cible RPi-Pico W.

- La mise en œuvre pratique sera abordée en TP…

BIBLIOGRAPHIE

REFERENCES BIBLIOGRAPHIQUES

- *Middleware for Internet of things*. Gaurav Bajpai. Ressource Internet
- *Investigating IoT Middleware Platforms for Smart Application Development*. Preeti Agarwal1, Mansaf Alam. Livre Smart Cities. Opportunities and Challenges pp. 231-244
- *Role of middleware for Internet of things: a study*. Soma Bandyopadhyay and al. International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.3, August 2011

- MicroPython : http://docs.micropython.org/en/latest/index.html
- Carte RPi-Pico W : https://www.raspberrypi.com/products/raspberry-pi-pico/