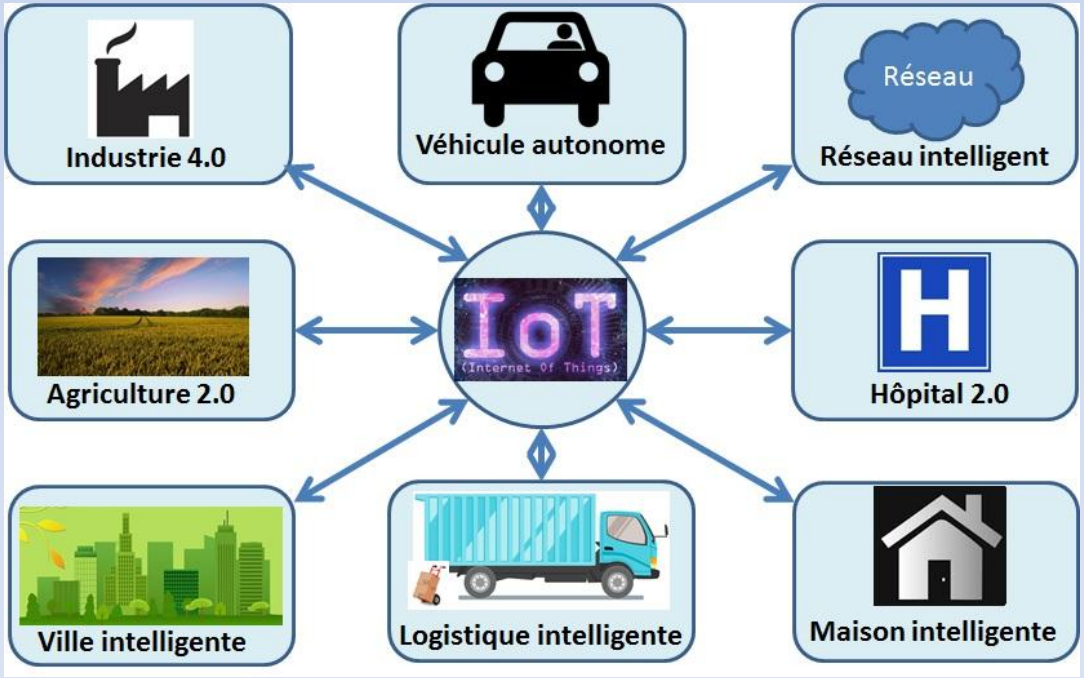
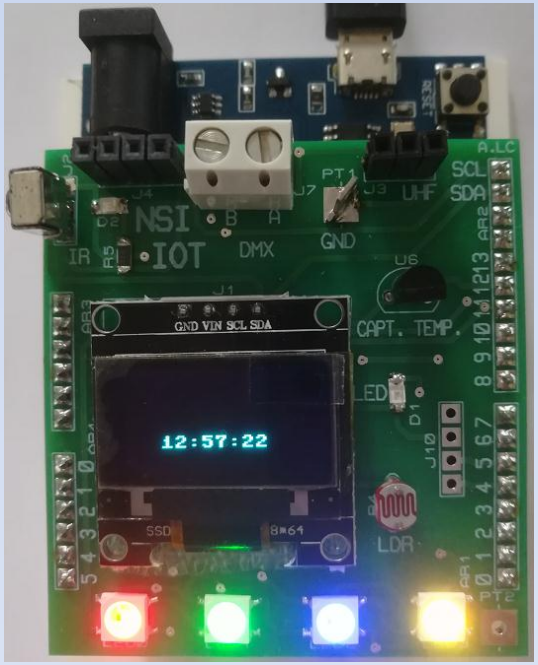
Les objets connectés
(IoT : Internet of Things)





nsi.touchard@gmail.com

A Le Cren

Sommaire

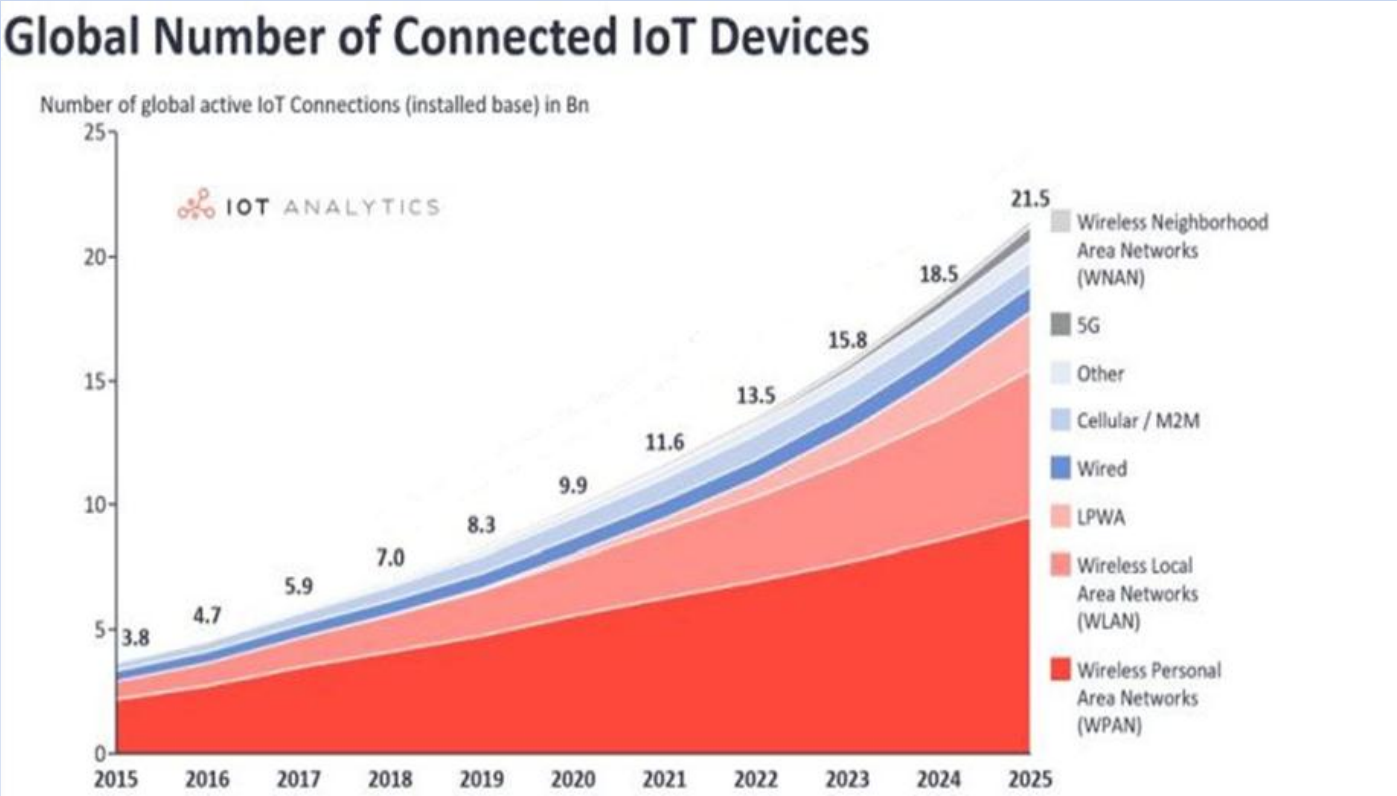
- Présentation des IOT
- Carte IOT
- Thèmes
- Technique
- Les TP proposés

https://github.com/f4goh/Carte-shield-IOT

IOT

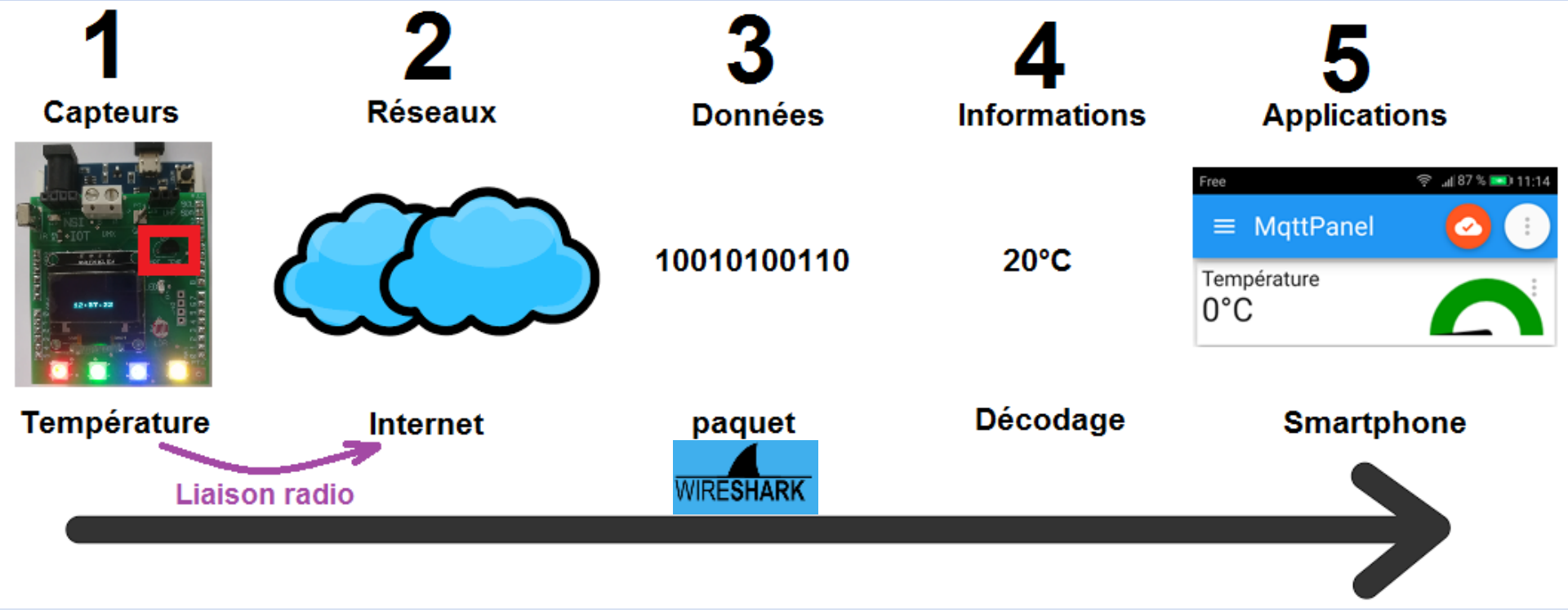
- *Une infrastructure mondiale pour la société de l'information*
- *Services évolués*
- *Interconnexion des objets*
- *Réseaux existants*

Une évolution fulgurante



- En 2025
- 21 milliards d'objets
- $1500 milliards de CA

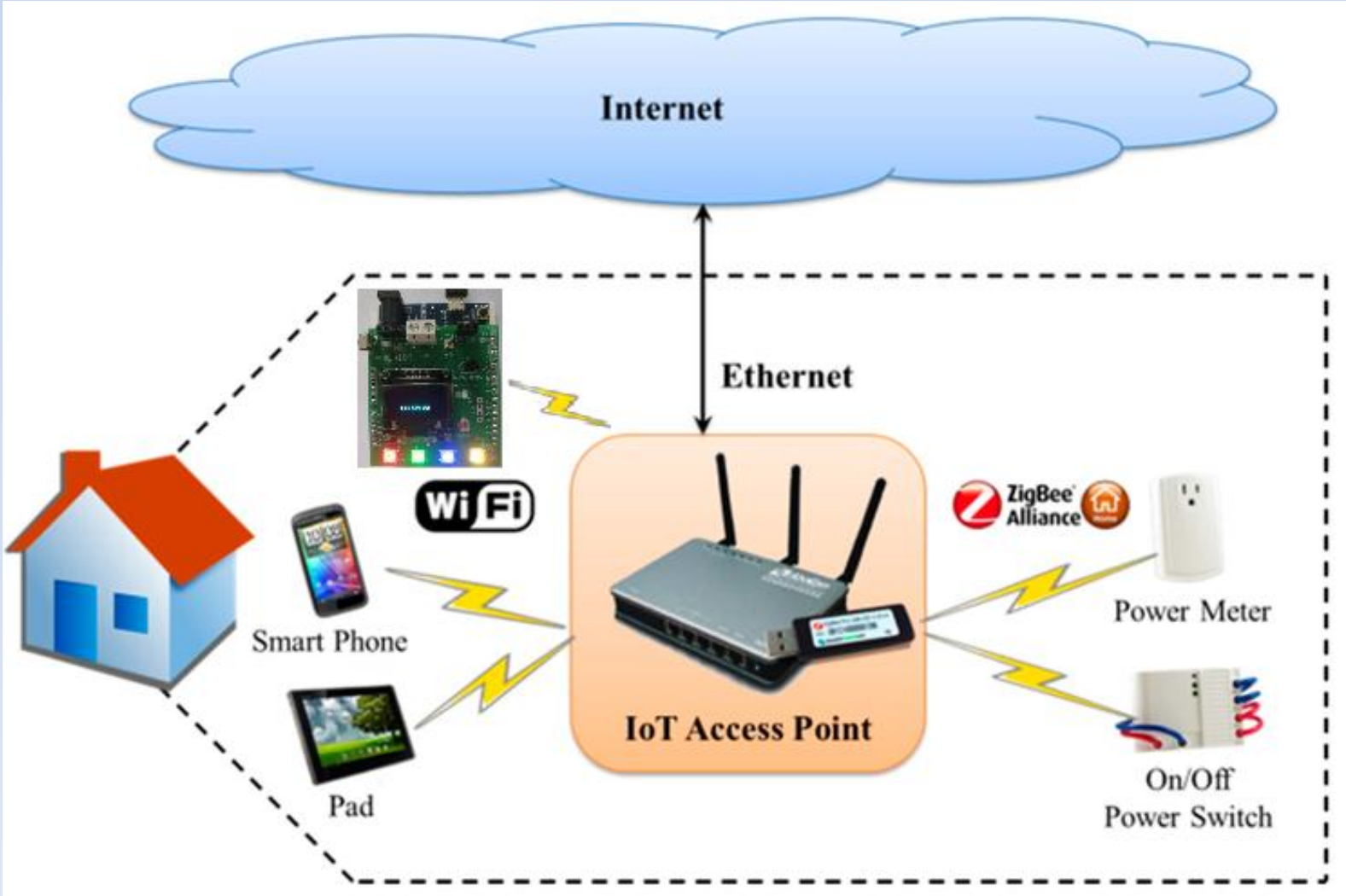
Les 5 composantes de l'IoT



IOT domestique (1/2)

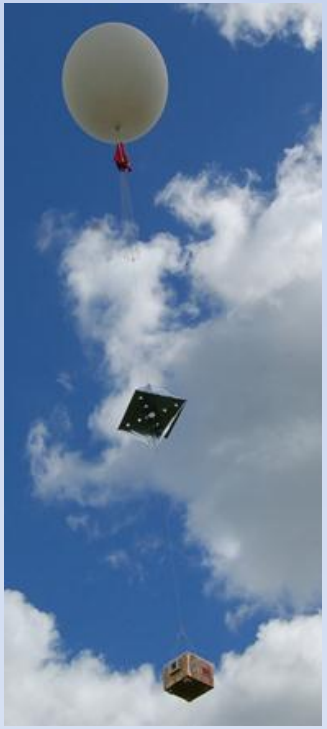
- Santé
- Confort
- Energie
- Sécurité
- Loisir

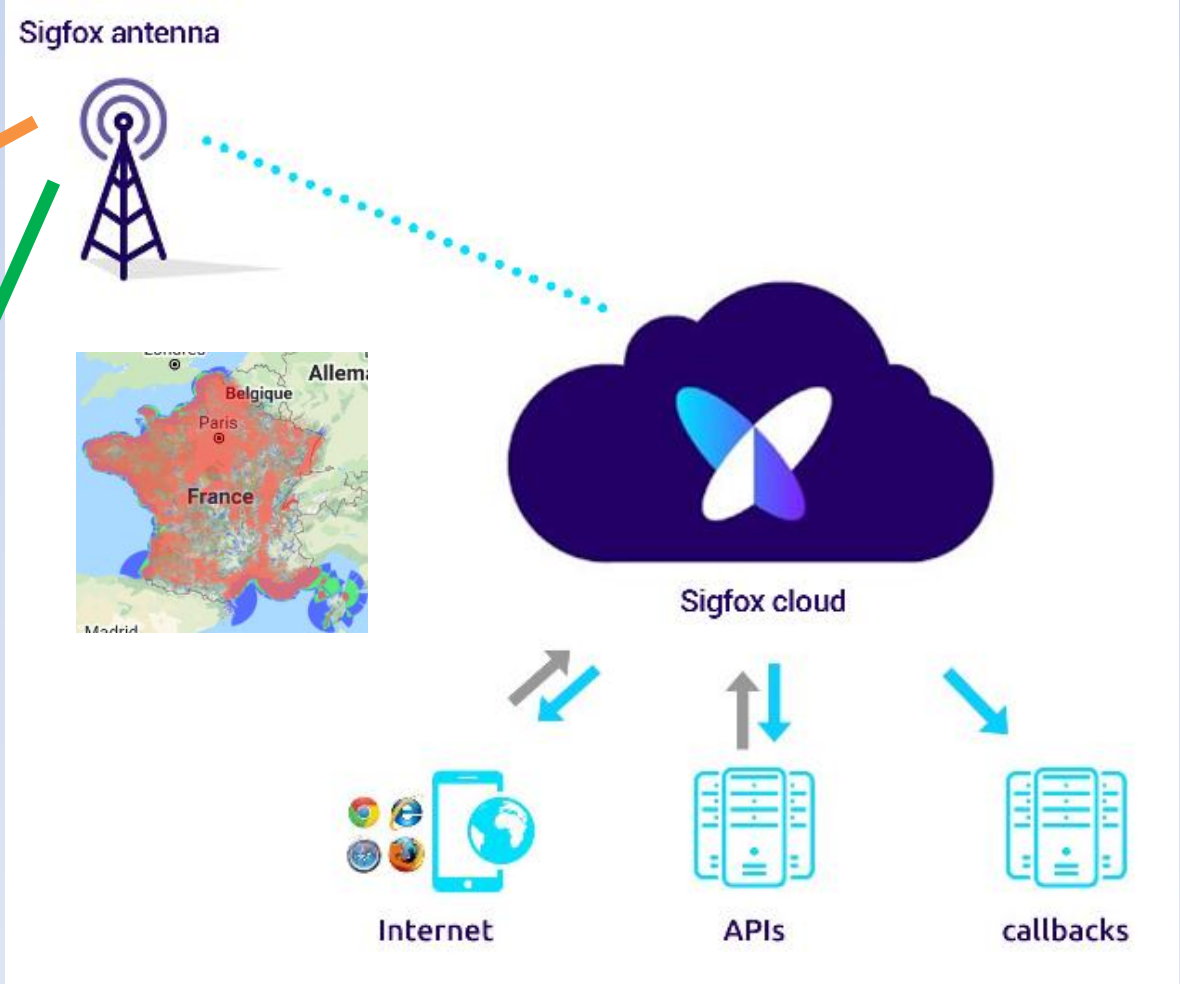
IOT domestique (2/2)



Liaison courte portée

IOT scientifiques ou environnementaux



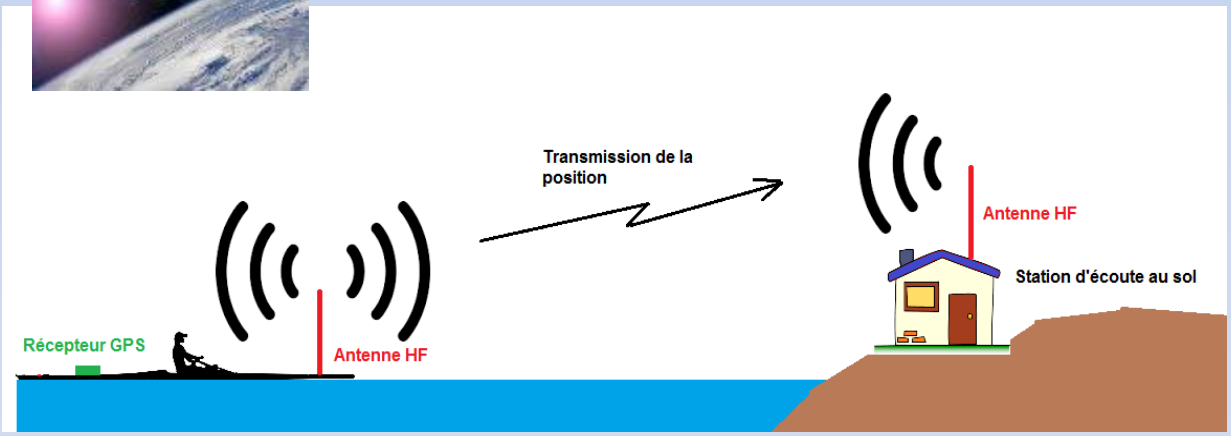


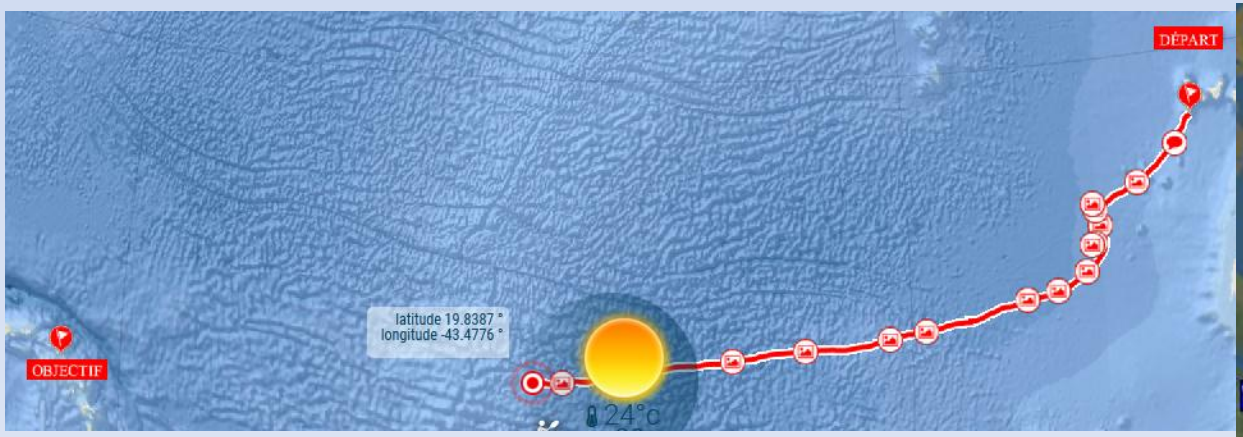
sigfox contrat 1 an

LoRa

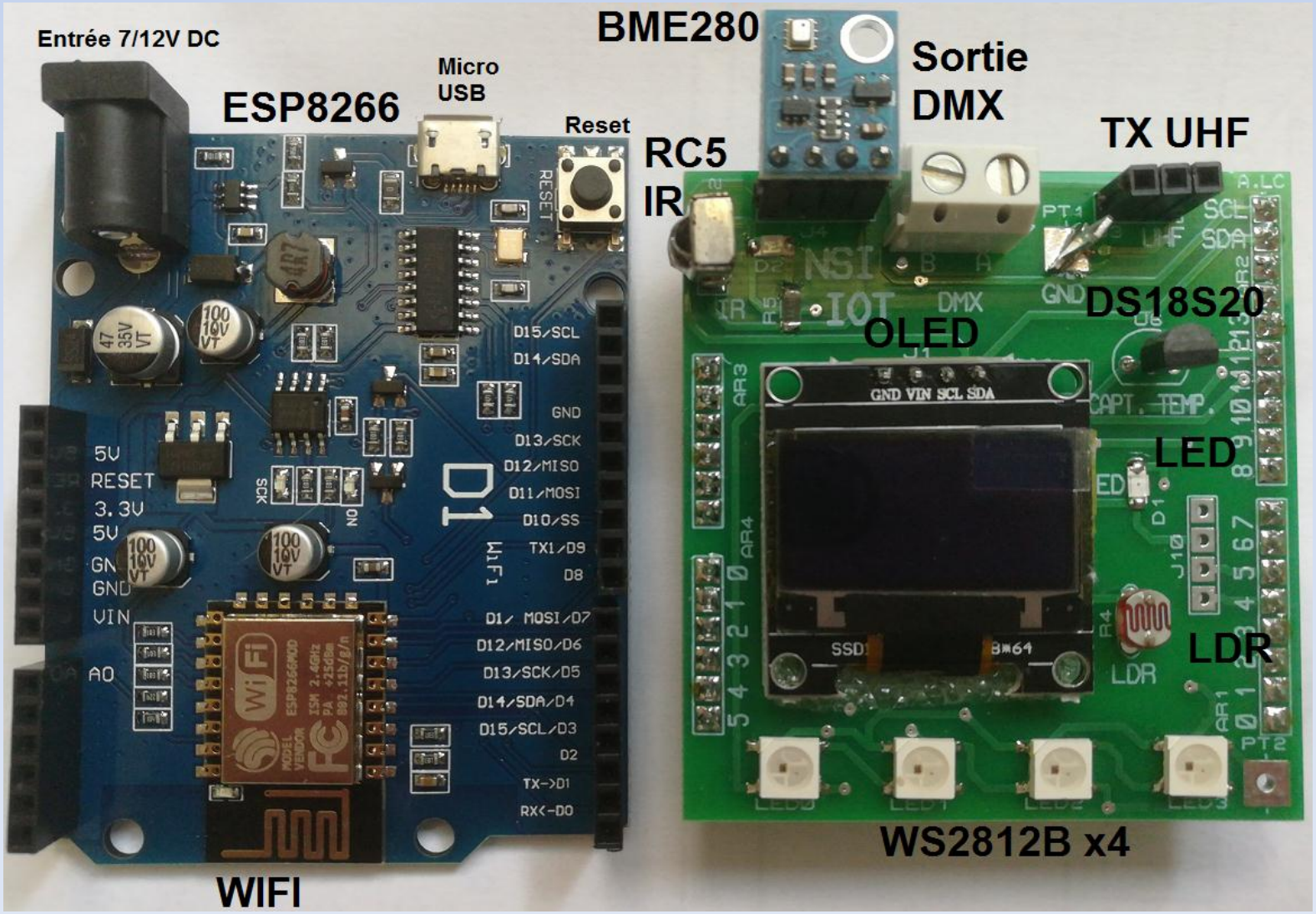
5G IOT

IOT géolocalisation



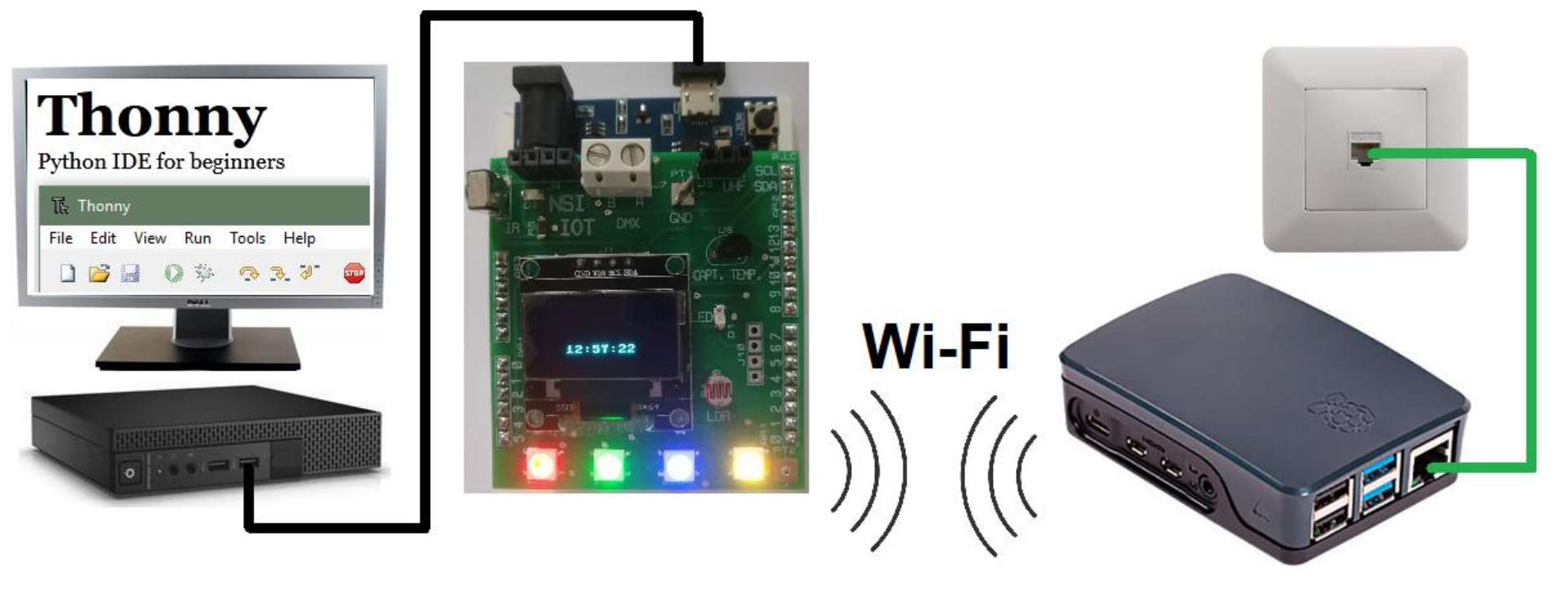


1 Découverte de la carte IOT



Carte sheild : Pas de fils

2 Connexion Wi-Fi



3 Protocole MQTT

Esp8266 UNO

Publish

Envoi de la température toutes les 10 secondes.

Broker (Serveur)

https://shiftr.io/

Subscribers

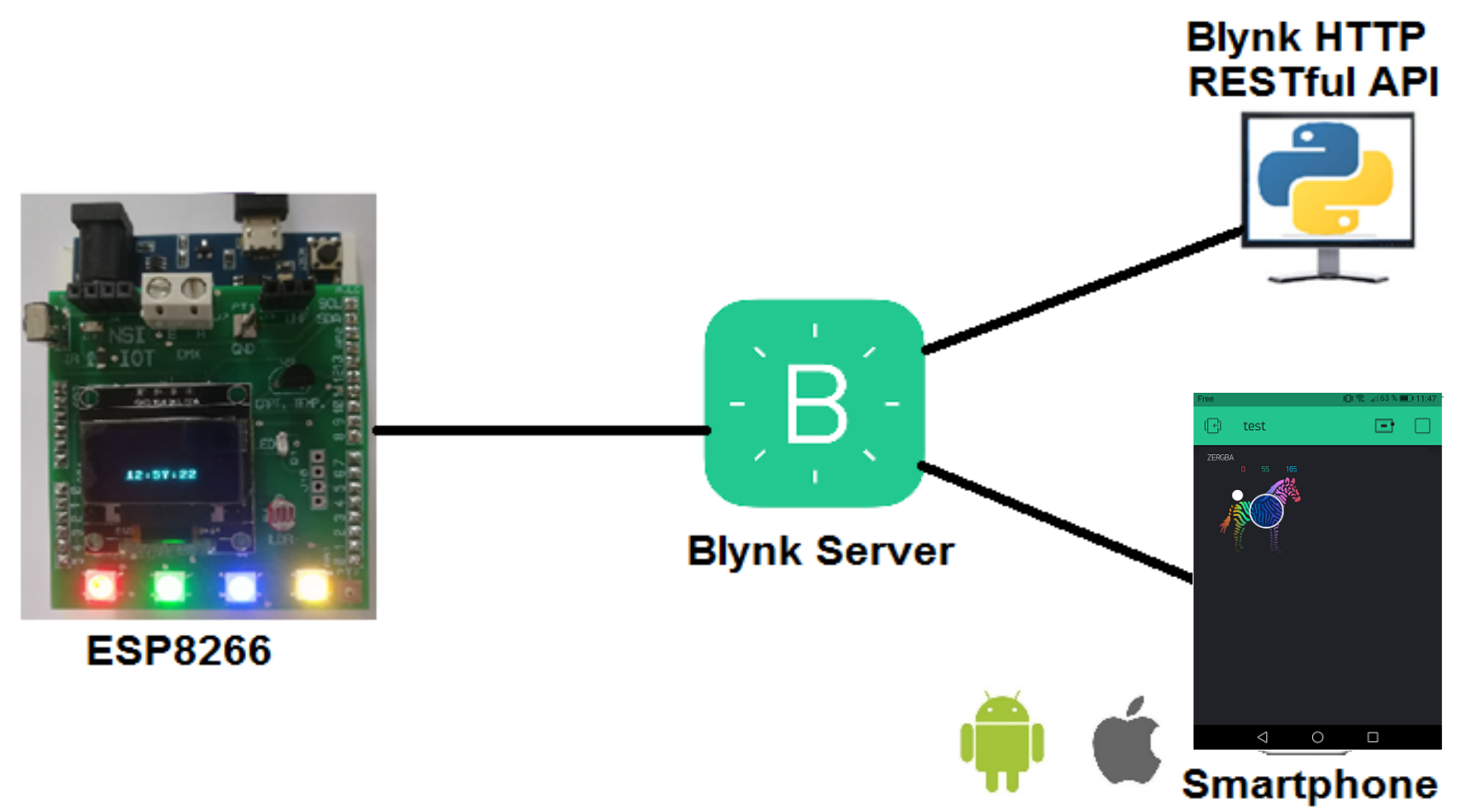
Client

Abonné N°1
(visualisation de la température avec **python sur PC**

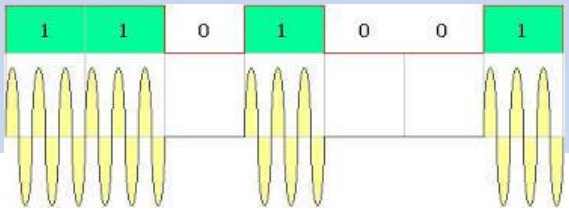
Client

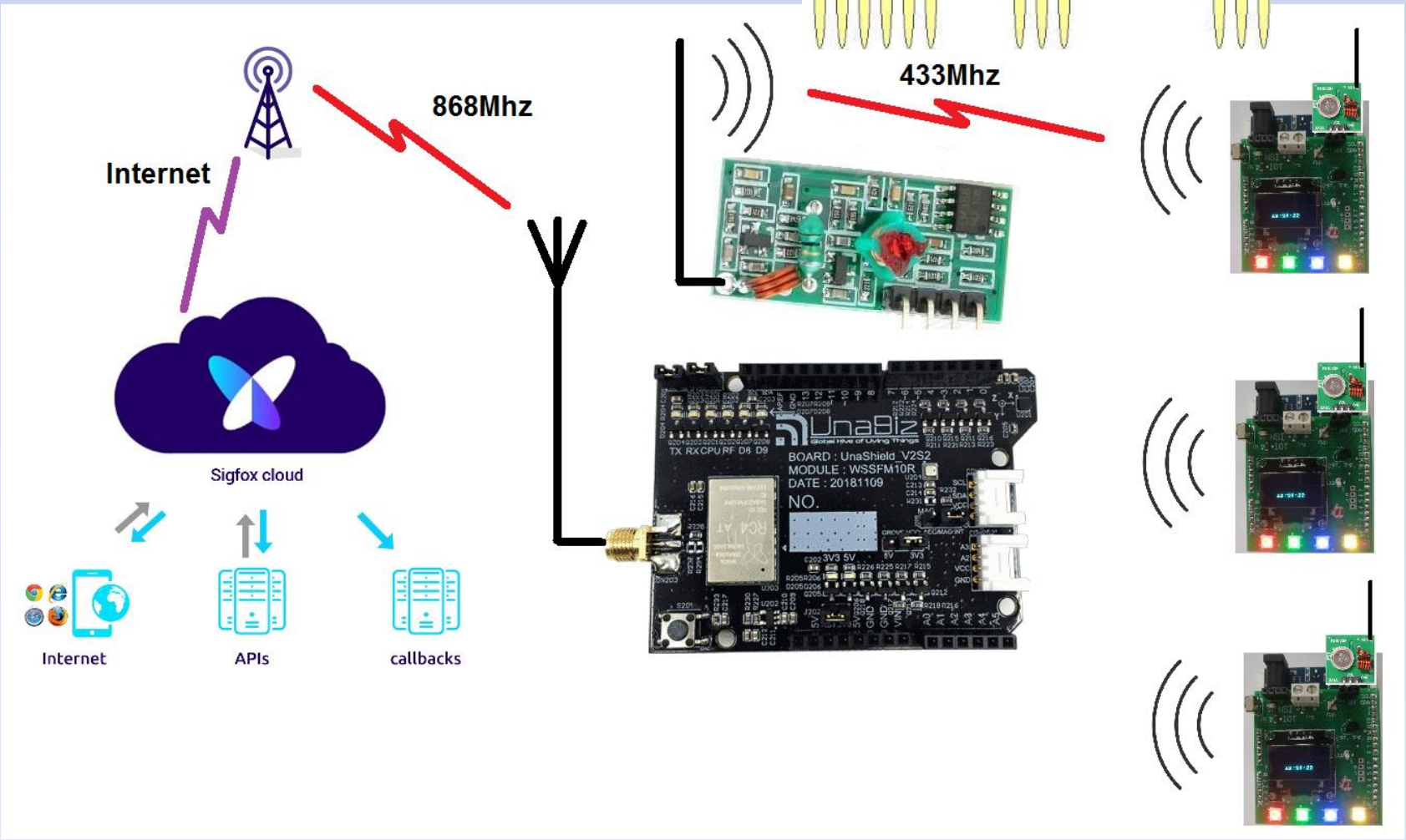
Abonné N°2
(visualisation de la température avec une tablette :**MQTT Dash**)

4 Plate-forme Blynk

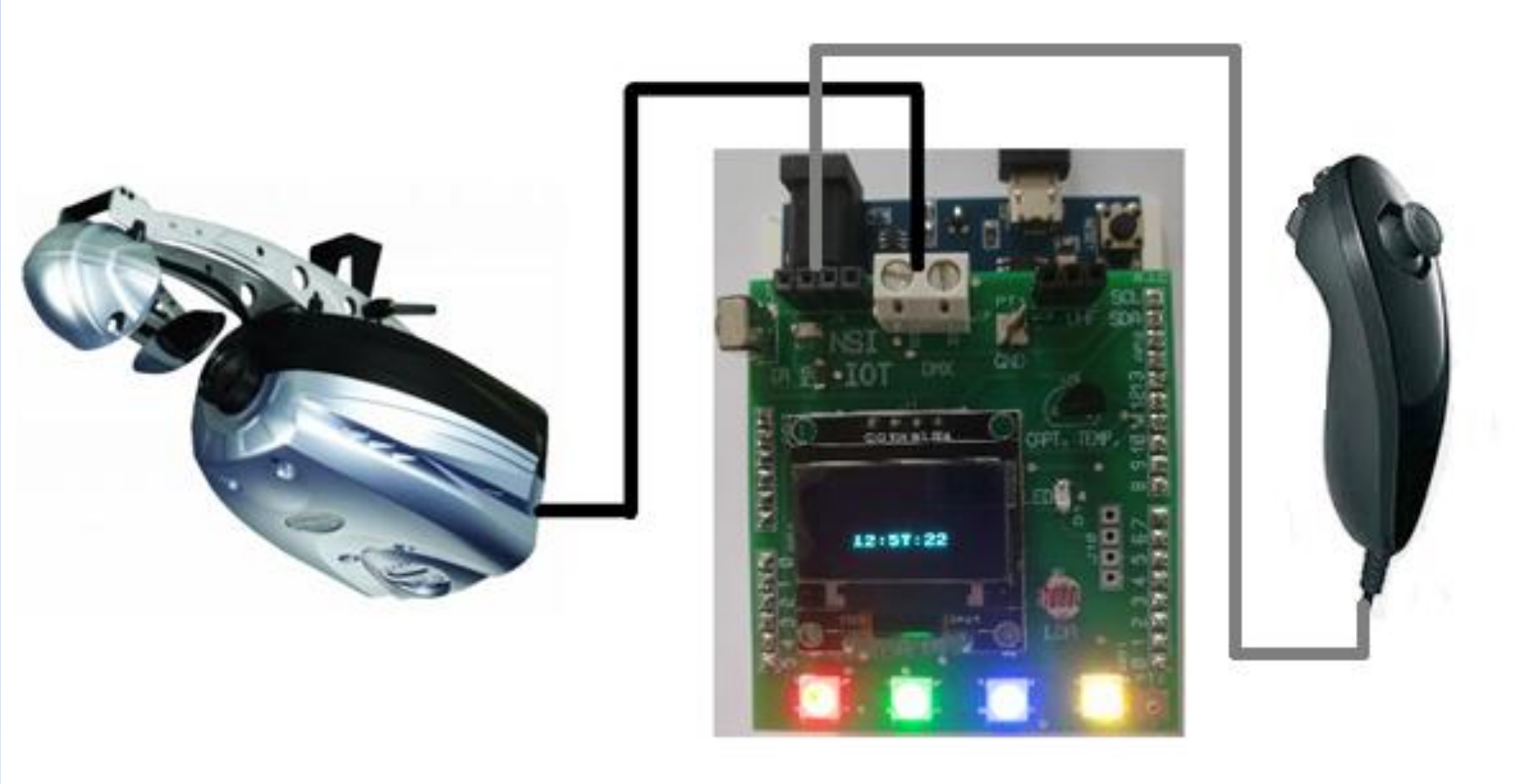


5 Extension Sigfox





6 Extension DMX



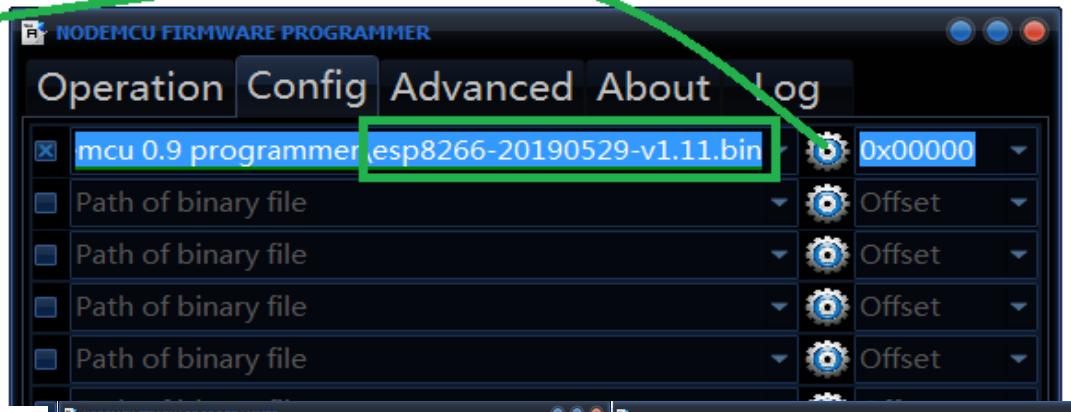
Programmer la carte ESP

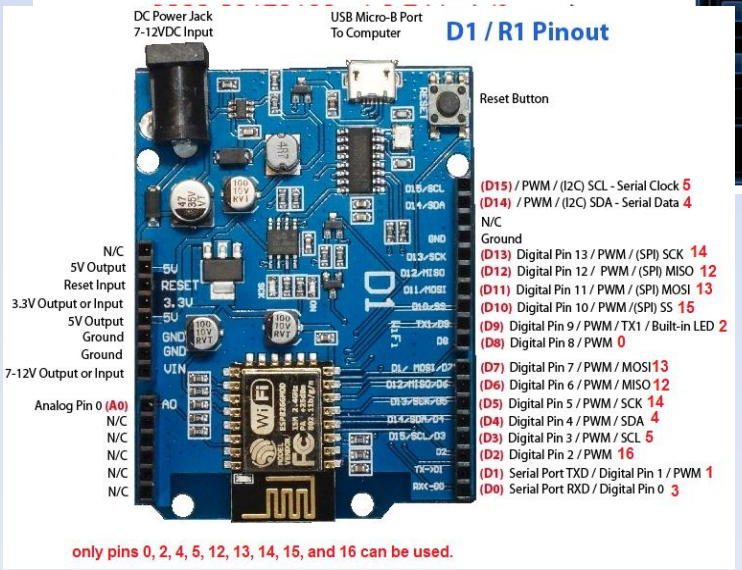
https://micropython.org/download#esp8266

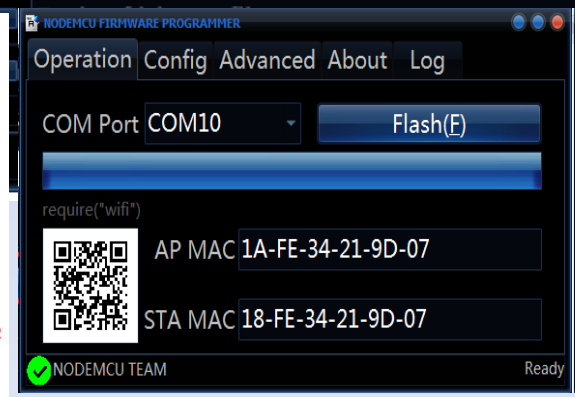
Firmware for ESP8266 boards

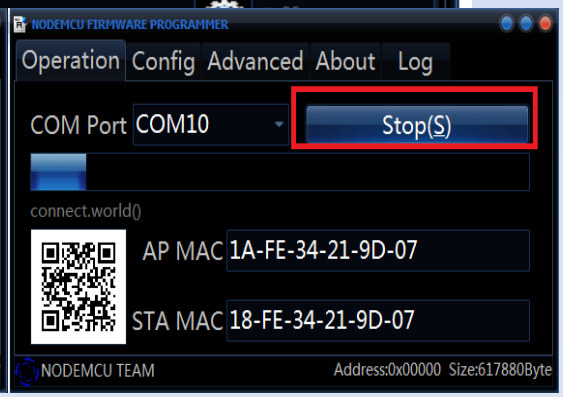
The following files are stable firmware for the ESP8266. Program your board using the esptool.py program as described in the tutorial.

- esp8266-20190529-v1.11.bin (elf, map) (latest)
- esp8266-20190125-v1.10.bin (elf, map)
- esp8266-20180511-v1.9.4.bin (elf, map)
- esp8266-20171101-v1.9.3.bin (elf, map)
- esp8266-20170823-v1.9.2.bin (elf, map)
- esp8266-20170612-v1.9.1.bin (elf, map)
- esp8266-20170526-v1.9.bin (elf, map)

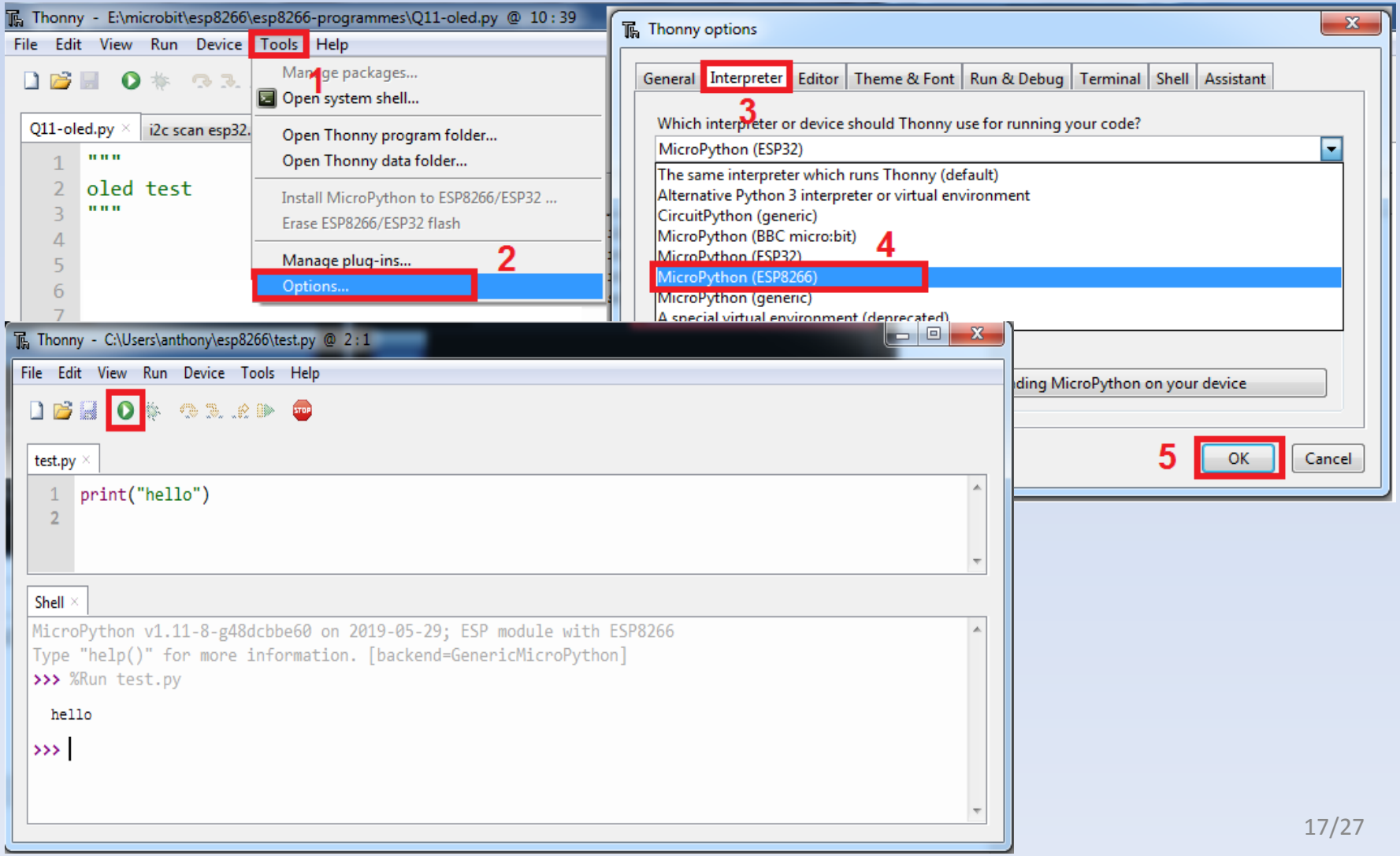








IDE Thonny


Thonny - E:\microbit\esp8266\esp8266-programmes\Q11-oled.py @ 10 : 39
File Edit View Run Device Tools Help
Manage packages...
Open system shell...
Open Thonny program folder...
Open Thonny data folder...
Install MicroPython to ESP8266/ESP32 ...
Erase ESP8266/ESP32 flash
Manage plug-ins...
Options...
1
2
Q11-oled.py
i2c scan esp32.
"""
oled test
"""
Thonny options
General Interpreter Editor Theme & Font Run & Debug Terminal Shell Assistant
3
Which interpreter or device should Thonny use for running your code?
MicroPython (ESP32)
The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment
CircuitPython (generic)
MicroPython (BBC micro:bit)
MicroPython (ESP32)
4
MicroPython (ESP8266)
MicroPython (generic)
A special virtual environment (deprecated)
ding MicroPython on your device
5
OK
Cancel
Thonny - C:\Users\anthony\esp8266\test.py @ 2 : 1
File Edit View Run Device Tools Help
test.py
print("hello")
Shell
MicroPython v1.11-8-g48dcbbe60 on 2019-05-29; ESP module with ESP8266
Type "help()" for more information. [backend=GenericMicroPython]
>>> %Run test.py
hello
>>>

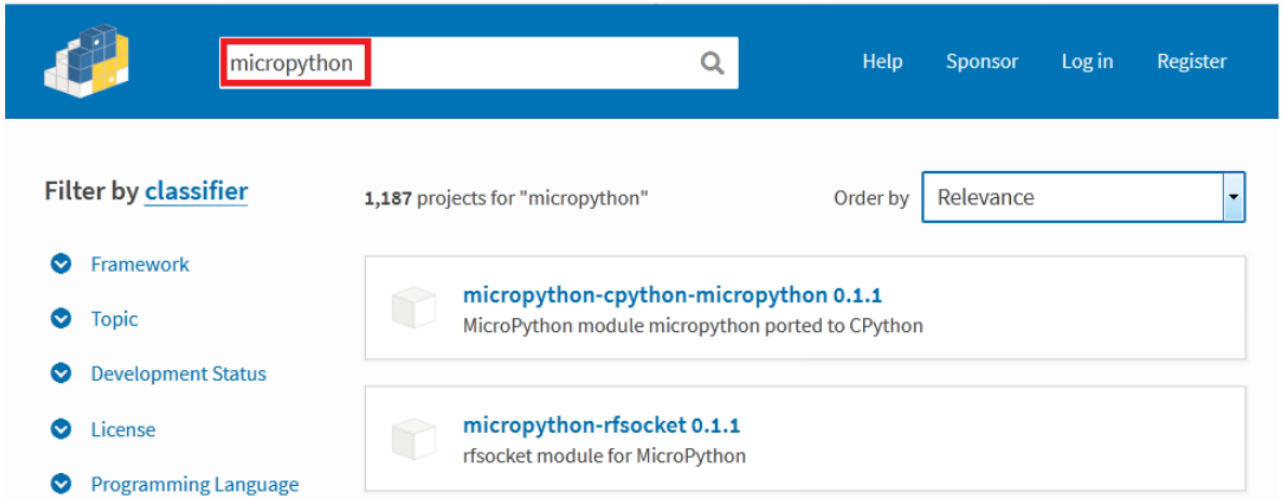
Les bibliothèques intégrées

Esp8266

```
MicroPython v1.12 on 2019-12-20; ESP module with ESP8266

Type "help()" for more information.
>>> help("modules")
__main__            http_client_ssl   uasyncio/__init__ upysh
_boot               http_server       uasyncio/core     urandom
_onewire            http_server_ssl   ubinascii         ure
_webrepl            inisetup          ucollections      urequests
apa102              lwip              ucryptolib        urllib/urequest
btree               machine           uctypes           uselect
builtins            math              uerrno            usocket
dht                 micropython       uhashlib          ussl
ds18x20             neopixel          uheapq            ustruct
esp                 network           uio               utime
example_pub_button                    ntptime           ujson
utimeq
example_sub_led     onewire           umqtt/robust      uwebsocket
flashbdev           port_diag         umqtt/simple      uzlib
framebuf            ssd1306           uos               webrepl
gc                  sys               upip              webrepl_setup
http_client         uarray            upip_utarfile     websocket_helper
Plus any modules on the filesystem
>>>
```

Ajout de bibliothèques

https://pypi.org/search/?q=micropython

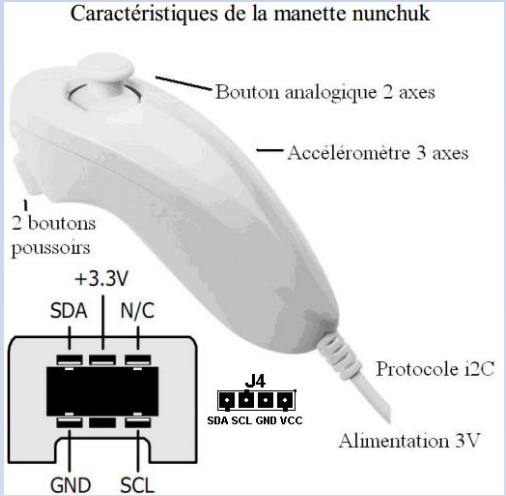


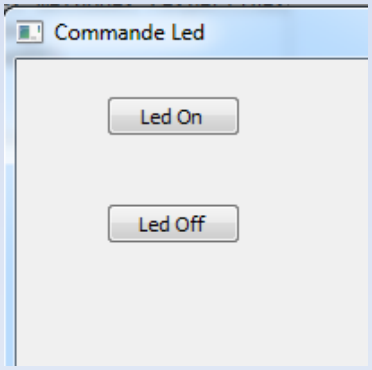

```
"""
wifi connexion
"""
import network
# user data
ssid = "Livebox-xxxx" # wifi router name
pw = "xxxxxxxxxxxxxxxxxx" # wifi router password

# wifi connection station mode
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(ssid, pw)

# wait for connection
while not wifi.isconnected():
    pass

# wifi connected
print(wifi.ifconfig())
```

```
>>> import upip
>>> upip.install('micropython-ssd1306')
```

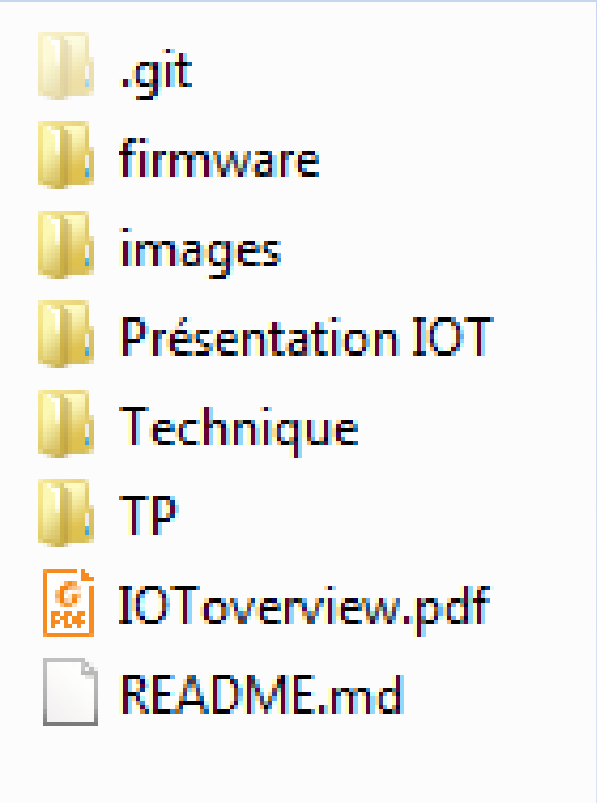
Programmes avancés

- Timers
- Liaison série UART
- Code IR RC5
- DMX
- Gestion des bibliothèques (POO)
- Nunchuk
- Bibliothèque uasyncio (gestion de tâches)
- PyQT



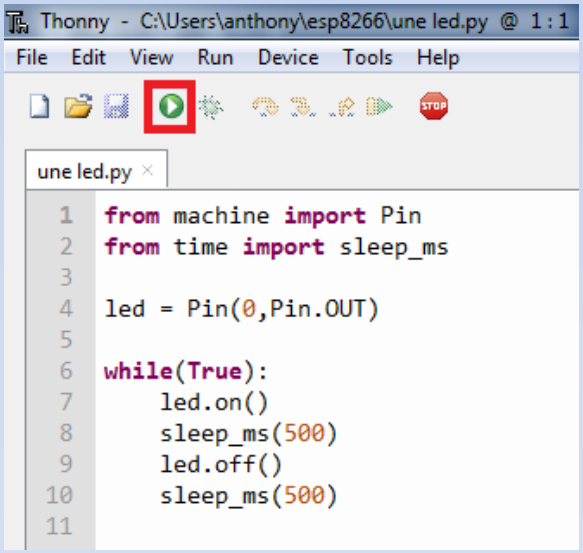


Exemples de TP (4 parties)



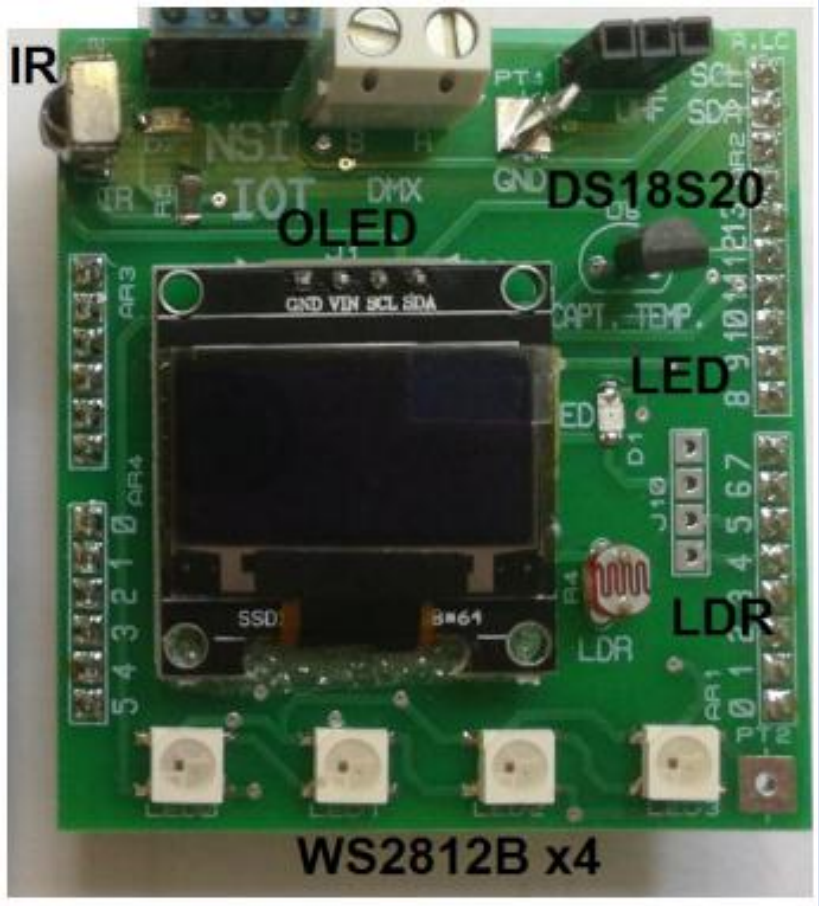
https://github.com/f4goh/Carte-shield-IOT

Partie 1 (Basic)




```python
from machine import Pin
from time import sleep_ms

led = Pin(0,Pin.OUT)

while(True):
    led.on()
    sleep_ms(500)
    led.off()
    sleep_ms(500)
```

```python
from machine import ADC
from time import sleep

adc = ADC(0)

while True:
    valeur=adc.read()
    print(valeur)
    sleep(1)
```



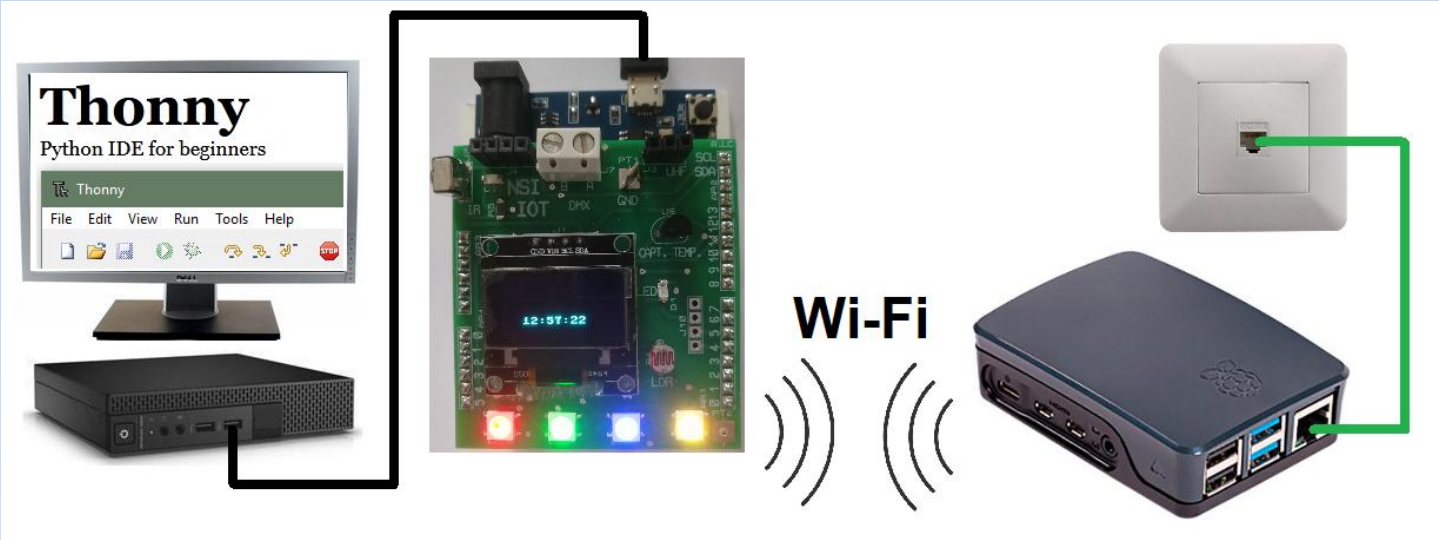
Partie 1 (Basic)

```python
from neopixel import NeoPixel
from machine import Pin

np = NeoPixel(Pin(14), 4)

np[0] = (0, 255, 255)
np[1] = (0, 128, 0)
np[2] = (0, 0, 64)
np[3] = (64, 0, 64)

np.write()
```

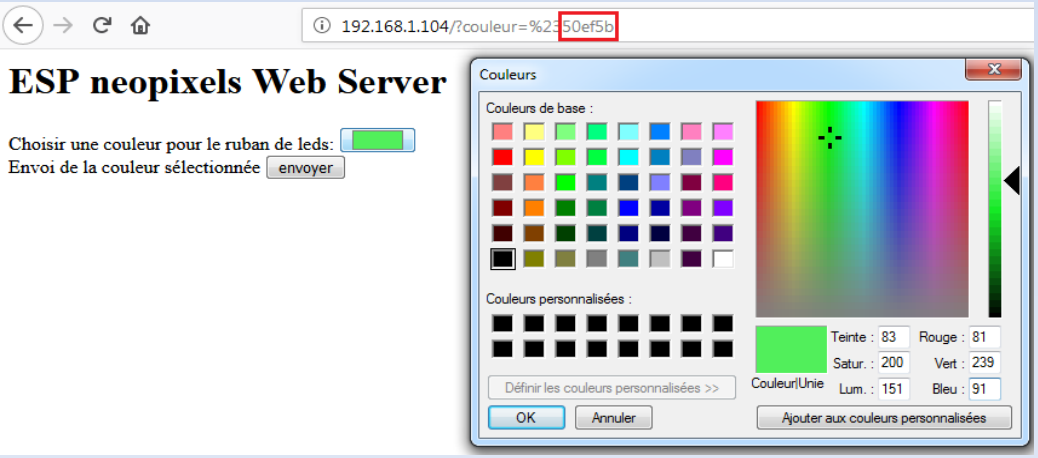
```python
from machine import Pin,I2C,RTC
from time import sleep

rtc = RTC()

#(year, month, day, weekday, hours, minutes, seconds, subseconds)
rtc.datetime((2019, 8, 2, 4,13,55, 0,0))

while True:
    horloge=rtc.datetime()
    print(horloge)
    sleep(1)
```

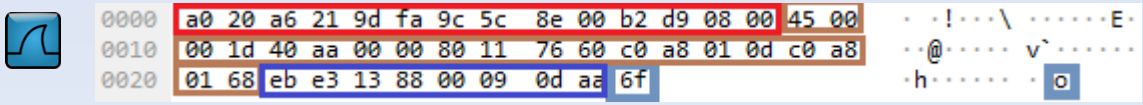
Mini projets

Partie 2 (Web)

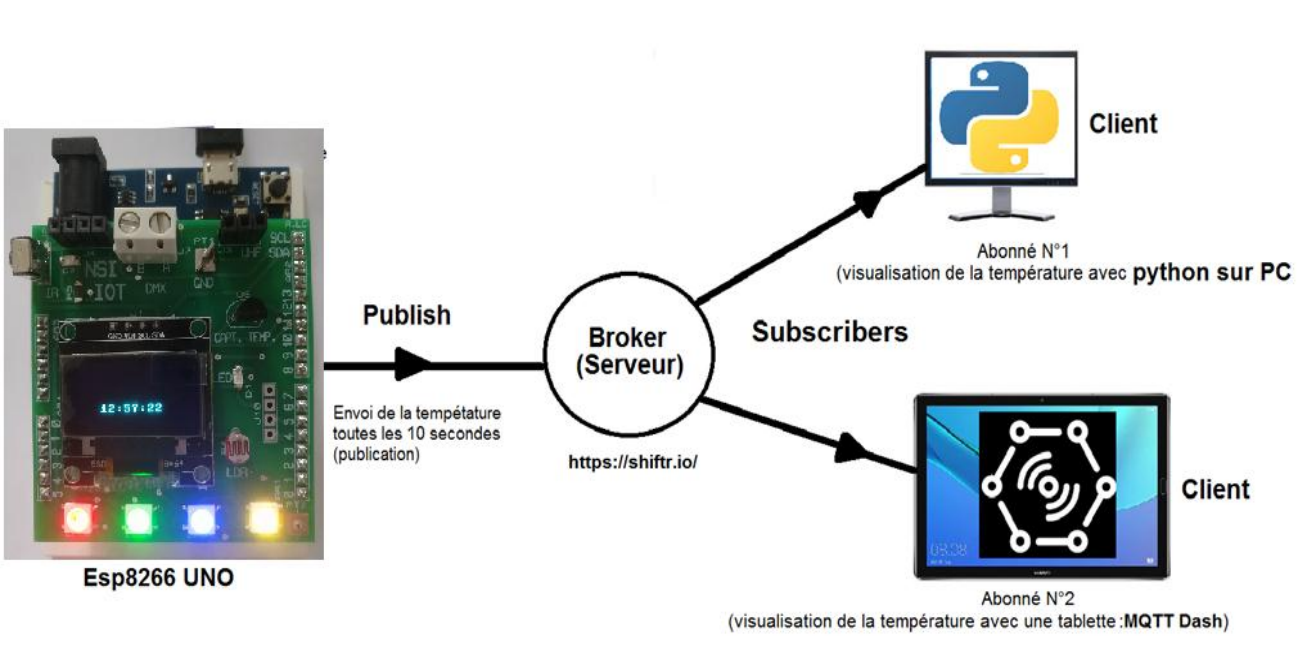

Thonny
Python IDE for beginners
Thonny
File Edit View Run Tools Help
Wi-Fi

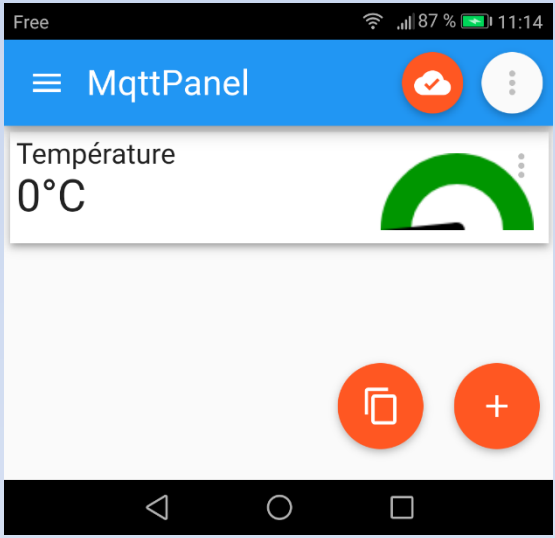

192.168.1.104/?LED=OFF
ESP Web Server
GPIO state: OFF
LED ON
LED OFF

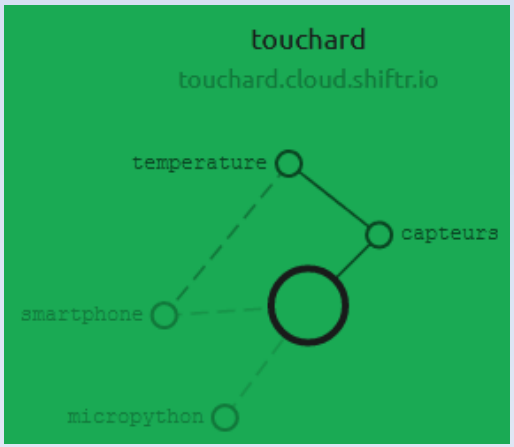

192.168.1.104/?couleur=%23 50ef5b
ESP neopixels Web Server
Choisir une couleur pour le ruban de leds:
Envoi de la couleur sélectionnée
envoyer
Couleurs
Couleurs de base :
Couleurs personnalisées :
Définir les couleurs personnalisées >>
OK
Annuler
Ajouter aux couleurs personnalisées
Teinte : 83
Satur. : 200
Lum. : 151
Rouge : 81
Vert : 239
Bleu : 91
Couleur|Unie


0000 a0 20 a6 21 9d fa 9c 5c 8e 00 b2 d9 08 00 45 00
0010 00 1d 40 aa 00 00 80 11 76 60 c0 a8 01 0d c0 a8
0020 01 68 eb e3 13 88 00 09 0d aa 6f

Partie 3 (MQTT)

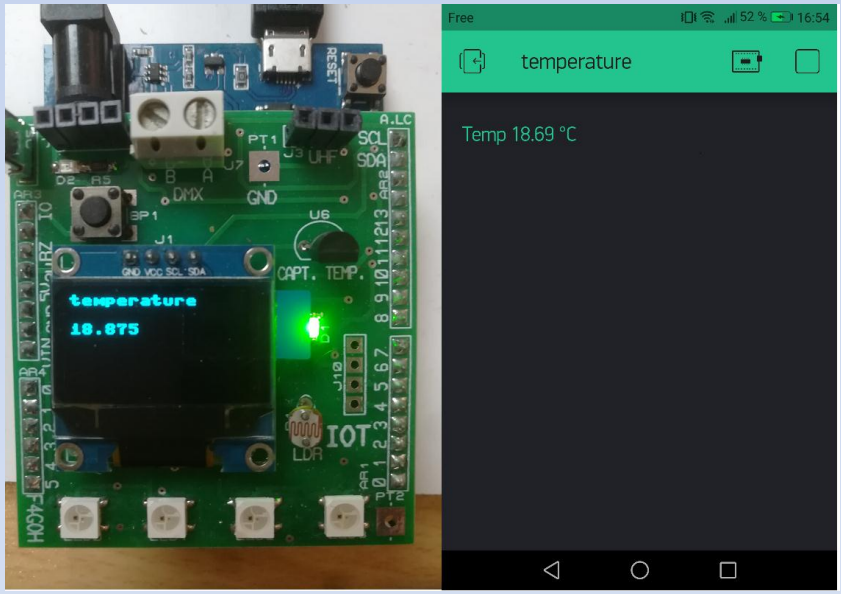


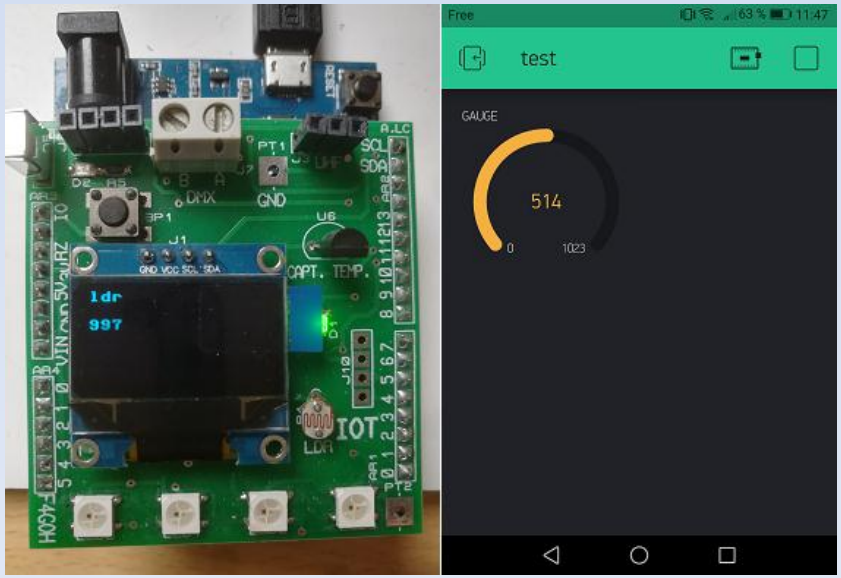


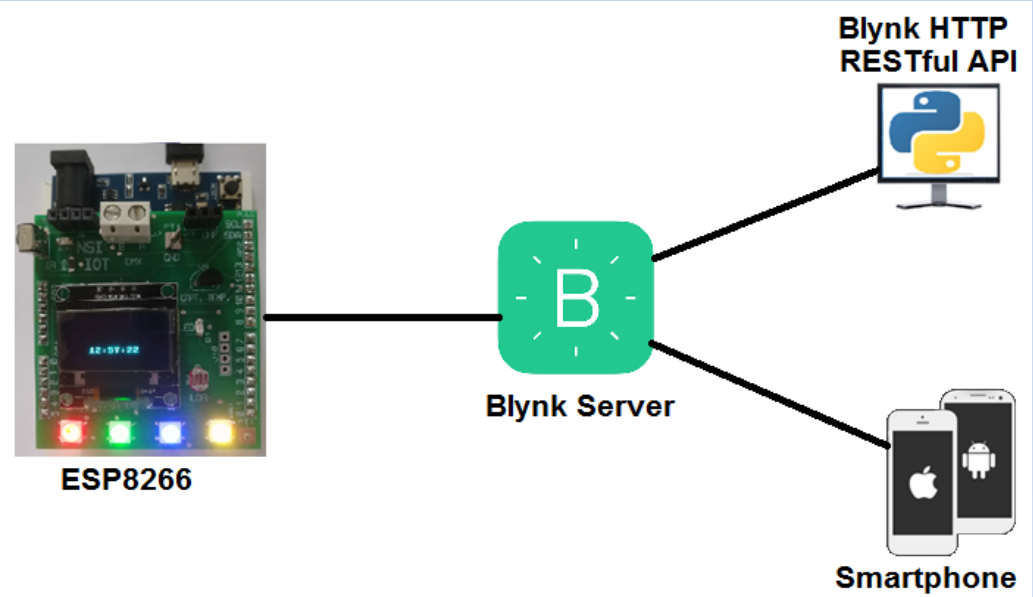



```python
while True:
    try:
        ds.convert_temp()
        sleep_ms( 750 )
        temp_celsius = ds.read_temp(capteur_temperature[0])
        print("Température : ",temp_celsius )
        client.publish("/capteurs/temperature", str(int(temp_celsius)))
        print("publish ok");
        time.sleep(PUBLISH_PERIOD_IN_SEC)
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()
        print("exit")
```

Partie 4 (Blynk)








```python
#Define 2 RGB colors
LOW_COLOR = '#f5b041'
HIGH_COLOR = '#85c1e9'

#This function is called at each smartphone request
@blynk.handle_event('read V{}'.format(LDR_VPIN))
def read_handler(vpin):
    ldr = adc.read()
    oled.fill(0)
    oled.text("ldr", 0, 0)
    oled.text(str(ldr), 0, 20)
    oled.show()
    print('ldr value=',ldr)
    if ldr<600:
        blynk.set_property(LDR_VPIN, 'color', LOW_COLOR)
    else:
        blynk.set_property(LDR_VPIN, 'color', HIGH_COLOR)
    blynk.virtual_write(vpin, ldr)


while True:
    blynk.run()
```

Bilan de la présentation

https://github.com/f4goh/Carte-shield-IOT