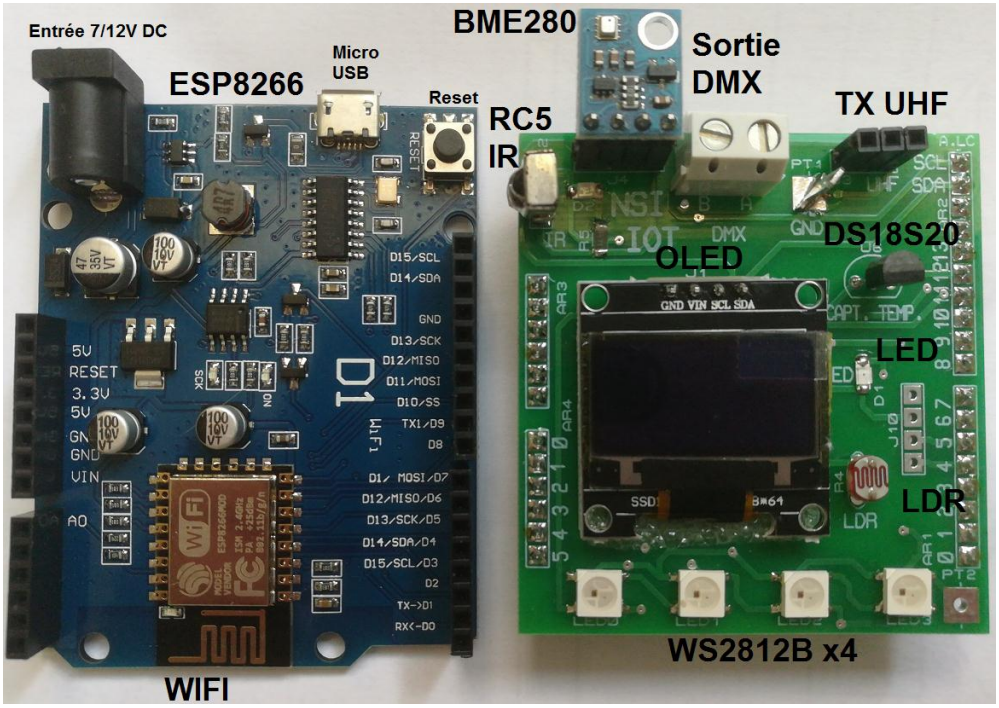
Carte IOT
(Description technique)

LE CREN Anthony



Description des périphériques montés sur la carte Shield

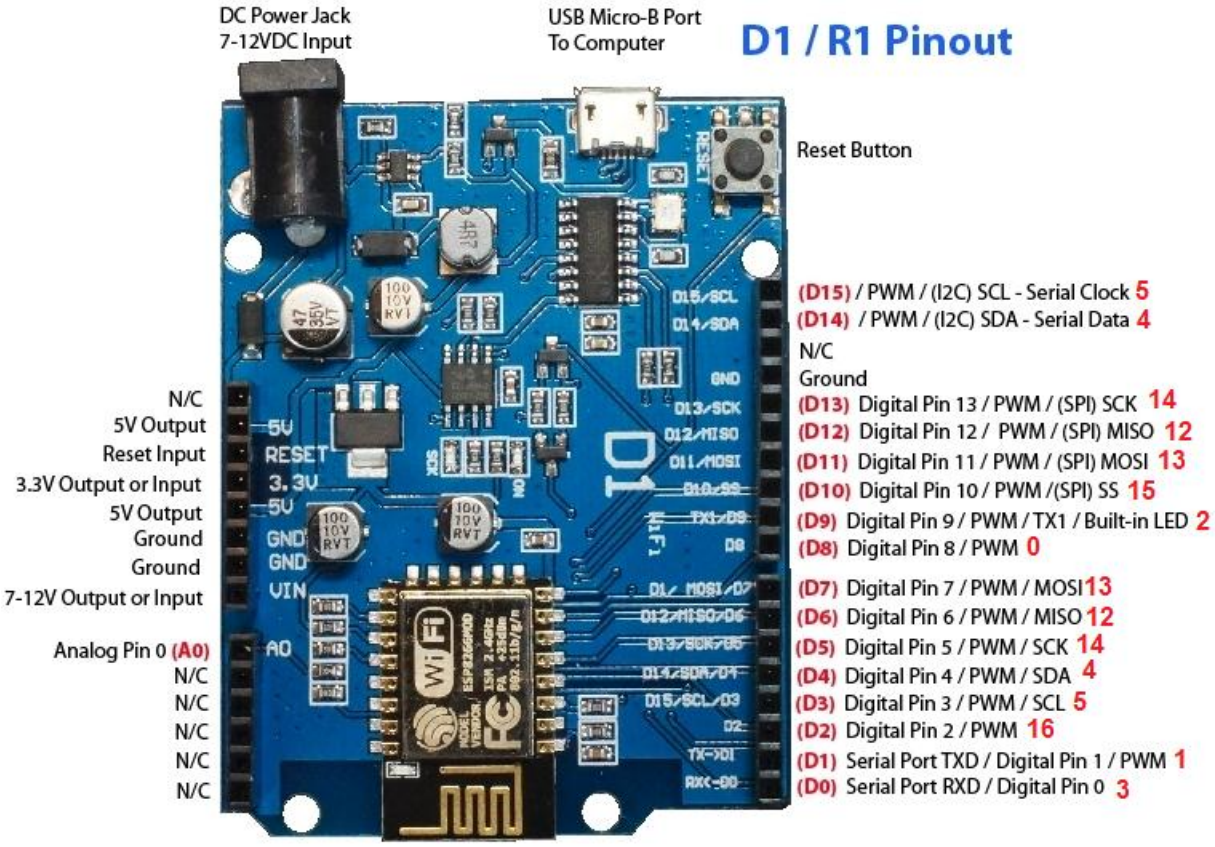
ESP32	ESP8266		Composant	Rôle
12	0	Logique (OUT)	Led rouge	Led de test
17	2	Logique (OUT)	Sortie RS485	Commande de spots DMX
21	4	Logique	SDA : Afficheur OLED	Afficheur OLED et capteur I2C divers
22	5	Logique	SCL : Afficheur OLED	
19	12	Logique	DS18S20	Capteur de température
23	13	Logique (IN)	VS1838B	Capteur infrarouge pour télécommande
18	14	Logique (OUT)	WS2812B	Ruban de 4 leds couleurs
5	15	Logique (OUT)	Emetteur UHF	Passerelle vers un réseau LPWAN
36	A0	Analogique (IN)	Capteur LDR	Capteur de lumière

DMX : ESP8266 : UART1 ou ESP32 : UART2

Configuration du sheild en fonction de la carte microcontrôleur :

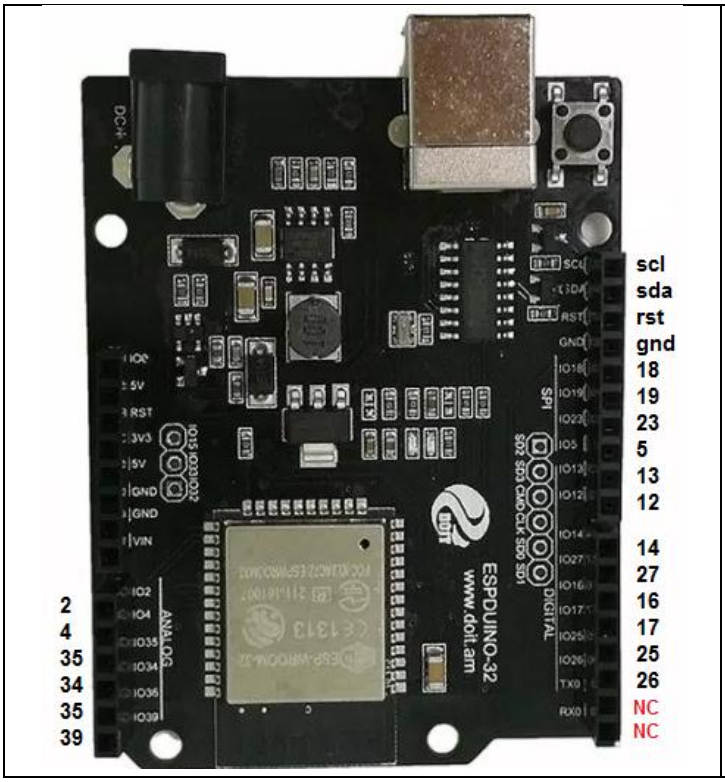
	J5	J6	J8	J9
ESP8266	Ouvert	Fermé	Ouvert	Fermé
ESP32	Fermé	Ouvert	Fermé	Ouvert

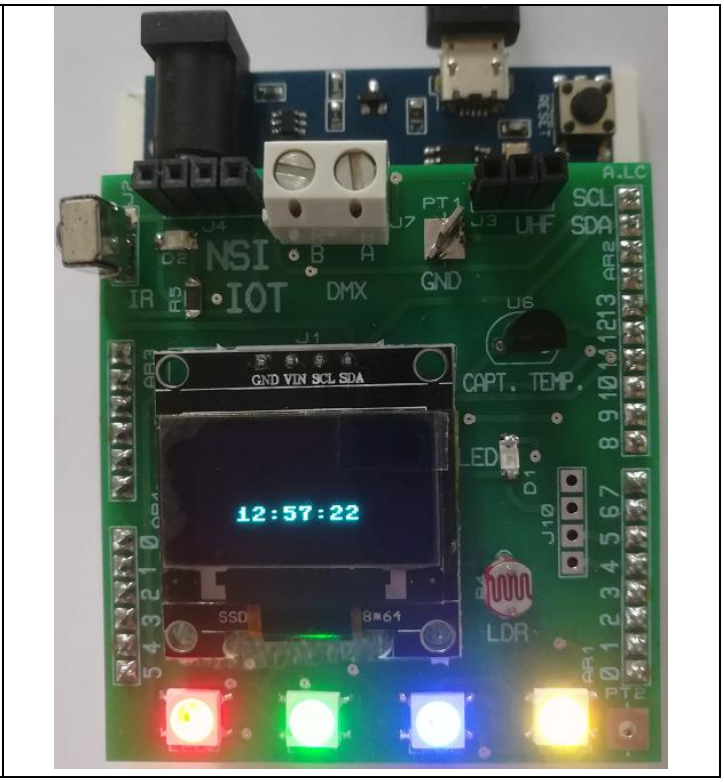
ESP8266



only pins 0, 2, 4, 5, 12, 13, 14, 15, and 16 can be used.

ESP32

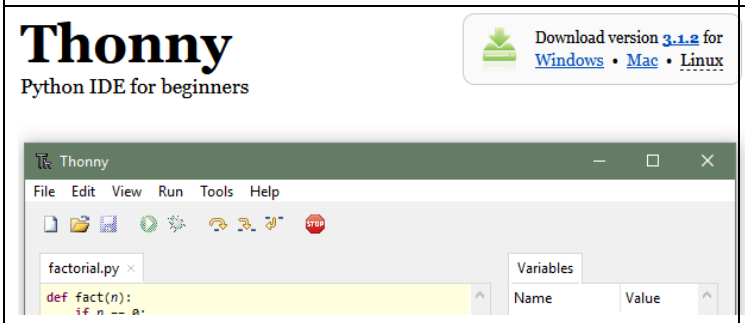


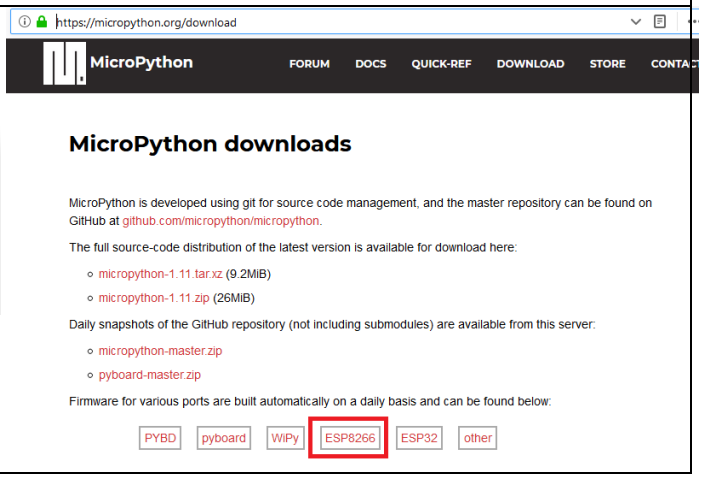


Annexe 1 : Implantation du firmware Micropython dans l'ESP8266

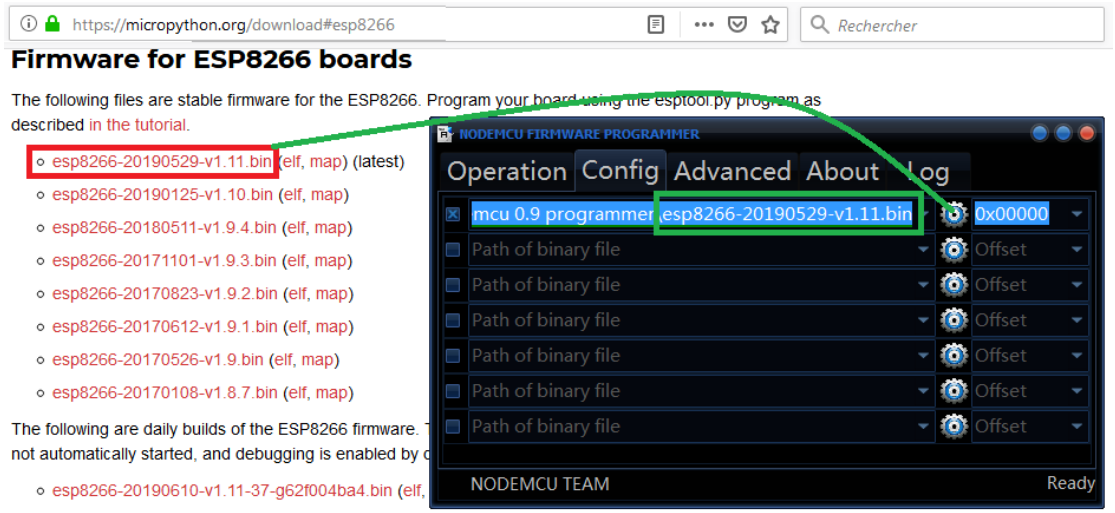
https://buildmedia.readthedocs.org/media/pdf/pycopy/latest/pycopy.pdf

Editeur	Firmware
https://thonny.org/ (Windows, linux installation via pip) ou https://codewith.mu/ (Windows, esp8266 non supporté sous linux)	https://micropython.org/download

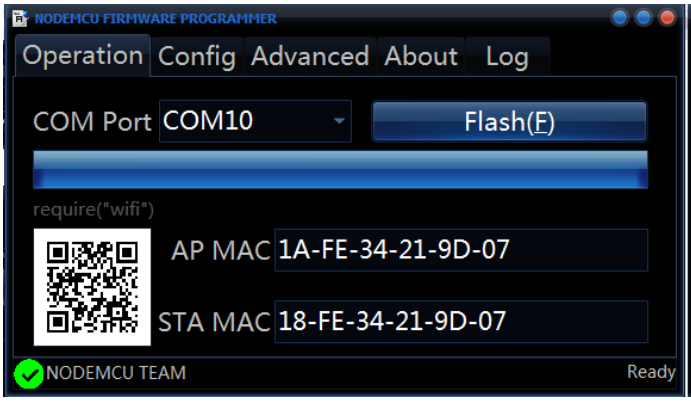


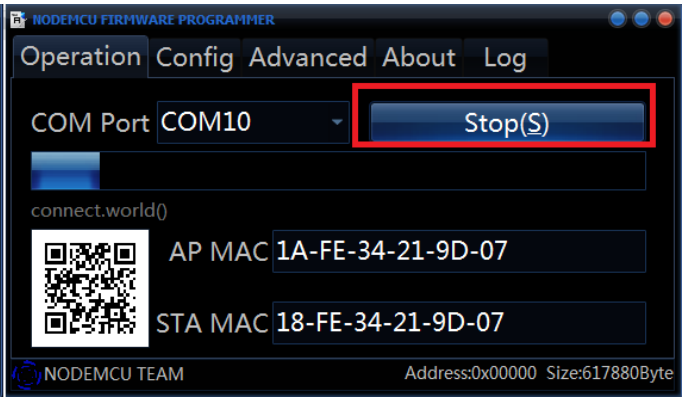


Flasher le firmware dans le 8266 (https://micropython.org/download#esp8266)
Avec nodemcu firmware programmer disponible ici : https://github.com/f4goh/NSI

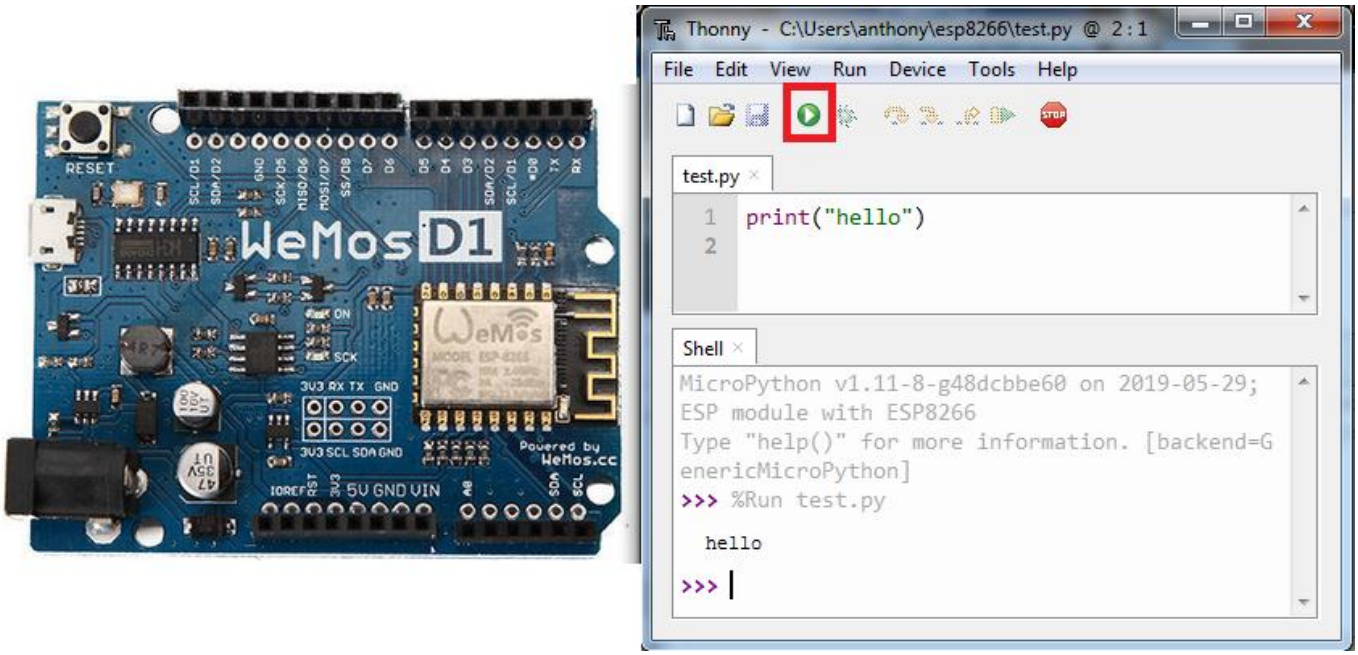


Puis bouton Flash.

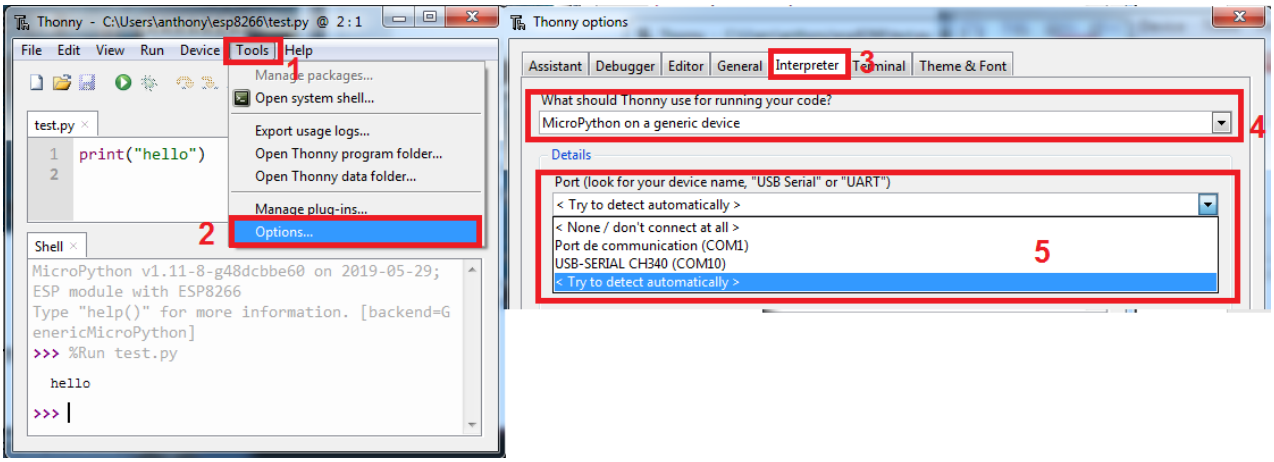




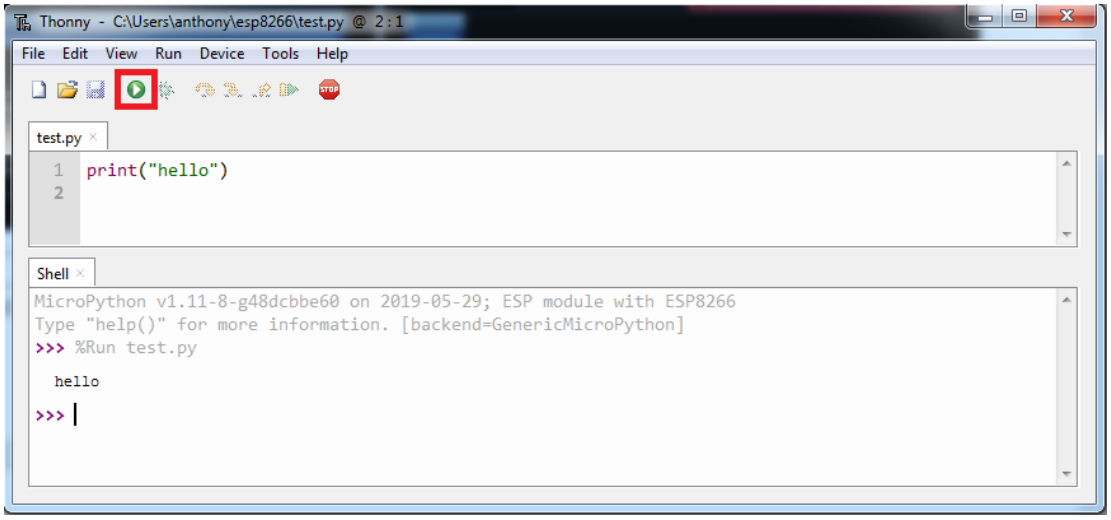
Faire un reset de la carte esp8266 **avant** de lancer Thonny



Configuration en mode esp8266



Finir par tester le bon fonctionnent



Annexe 2 ESP32 et Micropython

Pour programmer le firmware de l'esp32, il faudra installer esptool sous linux en ligne de commande

https://github.com/espressif/esptool

```
sudo apt install python-pip
sudo pip install --upgrade pip
sudo pip install esptool
pip install pyserial
sudo pip install pyserial
```

Repérer le port série de la carte esp32

```
nsi@nsi-LIFEBOOK-A555:~$ ls /dev/tty*
/dev/tty17  /dev/tty32  /dev/tty48  /dev/tty63      /dev/ttyS2   /dev/ttyS7
/dev/tty18  /dev/tty33  /dev/tty49  /dev/tty7       /dev/ttyS20  /dev/ttyS8
/dev/tty19  /dev/tty34  /dev/tty5   /dev/tty8       /dev/ttyS21  /dev/ttyS9
/dev/tty2   /dev/tty35  /dev/tty50  /dev/tty9       /dev/ttyS22  /dev/ttyUSB0
/dev/tty20  /dev/tty36  /dev/tty51  /dev/ttyprintk  /dev/ttyS23
/dev/tty21  /dev/tty37  /dev/tty52  /dev/ttyS0      /dev/ttyS24
/dev/tty22  /dev/tty38
 /dev/tty53  /dev/ttyS1      /dev/ttyS25
```

Télécharger le firmware pour l'esp32 (https://micropython.org/download#esp32)

Renommer le fichier par esp32.bin

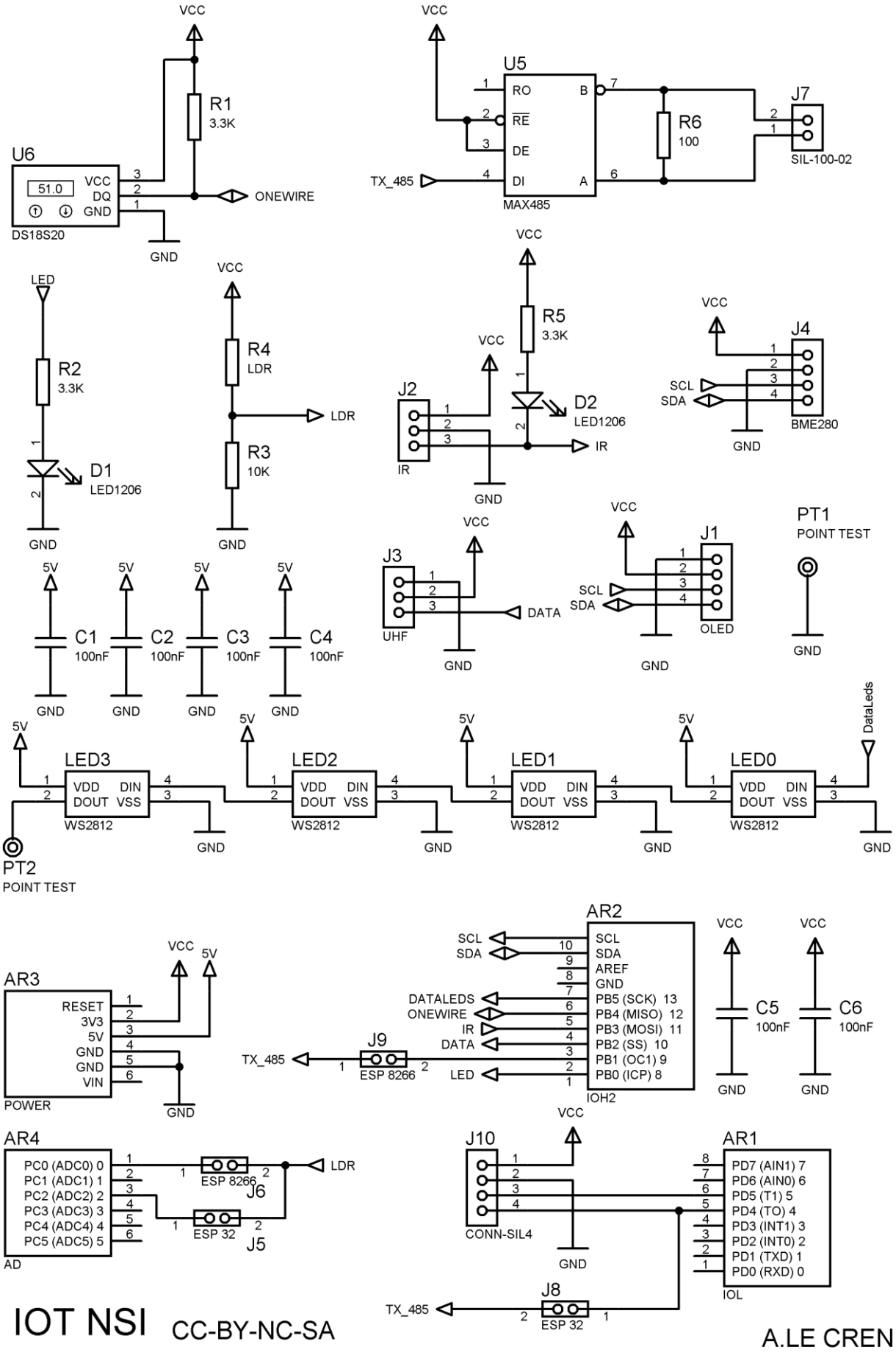
Exécuter les 2 commandes suivantes :

```
python esptool.py --port /dev/ttyUSB0 erase_flash
python esptool.py --chip esp32 --port /dev/ttyUSB0 write_flash -z 0x1000
esp32.bin
```

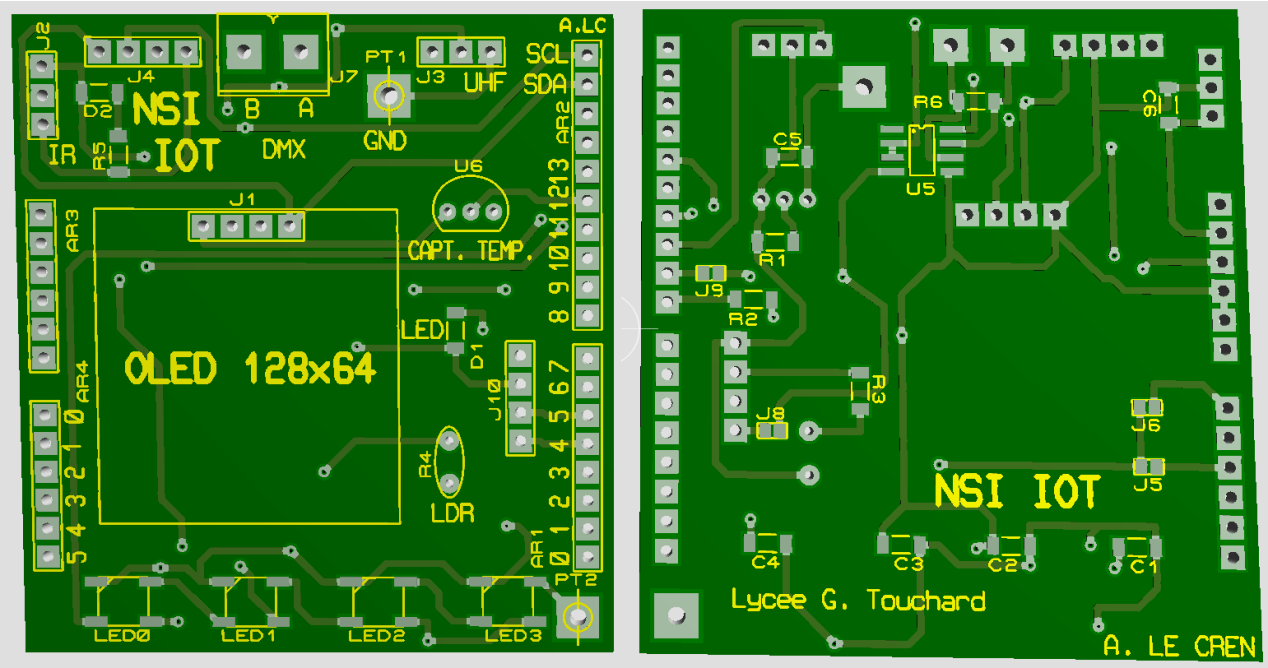
Commandes afin de changer l'adresse MAC de l'ESP32 si nécessaire

```
python espefuse.py --port /dev/ttyUSB0 summary
python espefuse.py --port /dev/ttyUSB0 dump
python espefuse.py --port /dev/ttyUSB0 mac
python espefuse.py --port /dev/ttyUSB0 get_custom_mac
python espefuse.py --port /dev/ttyUSB0 burn_custom_mac de:ad:be:ef:fe:00
python espefuse.py --port /dev/ttyUSB0 get_custom_mac
```

Annexe 3 : Schéma structurel



Circuit imprimé sheild IOT



Nomenclature

Résistances

3	R1,R2,R5	1K	cms1206
1	R3	10K	cms1206
1	R4	LDR	ldr classique trou traversant petit modèle
1	R6	100	cms1206

Condensateurs

6	C1-C6	100nF	cms1206

Circuits intégrés

https://www.ebay.fr/itm/100Pcs-MAX485-MAX485CSA-Txrx-RS485-RS422-Lowpwr-SOP-8-Date-CODE-12-wv/262963276497

1	U5	MAX485	sop-8

https://www.ebay.fr/itm/10PCS-IC-DS18S20-DS1820-Digital-Thermometer-IC-GOOD-QUALITY-Li2/132256267252

1	U6	DS18S20	to-92

Diodes : 2 D1,D2 LED cms1206

Divers :

1	AR1	IOL	barette male sécable 8pts
1	AR2	IOH2	barette male sécable 10pts
1	AR3	POWER	barette male sécable 6pts
1	AR4	AD	barette male sécable 6pts
1	J1	OLED	oled ssd1306 128x64 I2C
1	J2	IR	VS1838B recepteur IR
1	J3		barette femelle sécable 3pts
1	J4		barette femelle sécable 4pts
1	J7	SIL-100-02	bornier gris 2pts
1	J10		barette femelle sécable 4pts
4	LED0-LED3	WS2812b	led RGB boitier blanc cms a souder

https://www.ebay.fr/itm/0-96-I2C-IIC-SPI-Serial-128X64-OLED-LCD-LED-Display-Module-for-Arduino-SSD1306/223119333626
https://www.ebay.fr/itm/WS2812B-5050-SMD-Addressable-Digital-RGB-LED-4-pin-Chip-5V-Black-or-White/183460578312
https://www.ebay.fr/itm/20PCS-VS1838-TL1838-VS1838B-Universal-Infrared-Receiving-Head-For-Remote-control/201249361916

Ajouter une carte **esp8266 ou esp32**
https://www.ebay.fr/itm/OTA-WeMos-D1-CH340-WiFi-Development-Board-ESP8266-ESP-12E-For-Arduino-IDE-UNO-R3/163429353623

https://www.ebay.fr/itm/ESP32-UNO-R3-D1-R32-WIFI-Bluetooth-USB-B-CH340-Devolopment-Board-For-Arduino/264083453537

Sans oublier le cordon micro-usb

Annexe 4 : Programmation asynchrone avec librairie uasyncio

https://github.com/peterhinch/micropython-async/blob/master/TUTORIAL.md

Attention la dernière version d'uasyncio n'est pas dans le firmware de micropython. Surveillez les mises à jour.

```
import machine
import uasyncio as asyncio

led1 = machine.Pin(2, machine.Pin.OUT)
led2 = machine.Pin(15, machine.Pin.OUT)


async def blink(led, delay):  # coroutine
    while True:
        print(led, "on")
        led.on()
        await asyncio.sleep_ms(delay)
        print(led, "off")
        led.off()
        await asyncio.sleep_ms(delay)


# boucle d'événements
loop = asyncio.get_event_loop()
loop.create_task(blink(led1, 500))  # Schedule ASAP
loop.create_task(blink(led2, 1000))  # Schedule ASAP
loop.run_forever()
#ctrl+c pour stopper
```

```
Pin(2) on

Pin(15) on

Pin(2) off

Pin(15) off

Pin(2) on

Pin(2) off

Pin(15) on

Pin(2) on

Pin(2) off

Pin(15) off
```

```
Traceback (most recent call last):
  File "C:\Users\anthony\esp8266\test.py", line 21, in <module>
  File "uasyncio/core.py", line 173, in run_forever
  File "uasyncio/__init__.py", line 69, in wait
KeyboardInterrupt:
>>>
```

>>> import uasyncio
>>> dir(uasyncio)
['__class__', '__name__', '__path__', 'DEBUG', 'log', 'select', 'sleep', 'sleep_ms', 'time', 'ucollections', 'uerrno', 'utimeq', 'type_gen', 'set_debug', 'CancelledError', 'TimeoutError', 'EventLoop', 'SysCall', 'SysCall1', 'StopLoop', 'IORead', 'IOWrite', 'IOReadDone', 'IOWriteDone', 'get_event_loop', 'SleepMs', 'cancel', 'TimeoutObj', 'wait_for_ms', 'wait_for', 'coroutine', 'ensure_future', 'Task', '_socket', 'PollEventLoop', 'StreamReader', 'StreamWriter', 'open_connection', 'start_server', 'uasyncio', 'core']

Remarque :
Par défaut il n'y a pas le module uasyncio dans l'esp32, il faut l'installer manuellement après être connecté sur votre point d'accès wifi.

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

  Installing to: /lib/
  Warning: micropython.org SSL certificate is not validated
  Installing micropython-uasyncio 2.0 from https://micropython.org/pi/uasyncio/uasyncio-2.0.tar.gz
  Installing micropython-uasyncio.core 2.0 from https://micropython.org/pi/uasyncio.core/uasyncio.core-2.0.tar.gz
```

upip.install('micropython-ssd1306') # pour l'afficheur oled ssd1306

Annexe 5 : Exemples de programmes Micropython

Timers en micropython

La précision du signal généré sur la broche 13 n'a rien de comparable avec le même programme en langage C.

```
"""
timer irq
the ESP32 has 4 hardware timers. For this example, we will use timer 0.
"""

import machine

led = machine.Pin(16, machine.Pin.OUT)
timer = machine.Timer(0)


def handleInterrupt(timer):
    global ledState
    ledState ^= 1
    led.value(ledState)

ledState = 0

timer.init(freq=1000, mode=machine.Timer.PERIODIC, callback=handleInterrupt)
# timer.init(period=1000, mode=machine.Timer.PERIODIC, callback=handleInterrupt)

while True:
    pass
```

Scanner I2C

```
"""
scan i2c esp32
Scan i2c bus...
i2c devices found: 1
Decimal address:  60  | Hexa address:  0x3c
"""

import machine
i2c = machine.I2C(scl=machine.Pin(4), sda=machine.Pin(5))

print('Scan i2c bus...')
devices = i2c.scan()

if len(devices) == 0:
    print("No i2c device !")
else:
    print('i2c devices found:',len(devices))

for device in devices:
    print("Decimal address: ",device," | Hexa address: ",hex(device))
```

Afficheur OLED SSD1306

```
"""
oled test
"""

import machine, ssd1306

i2c = machine.I2C(scl=machine.Pin(4), sda=machine.Pin(5))
oled = ssd1306.SSD1306_I2C(128, 64, i2c, 0x3c)
oled.fill(0)
oled.text("Hello f4goh", 0, 0)
oled.show()
```

Emission et réception sur l'UART 2 avec asyncio

```
"""
esp32 pinout
gpio16 rx
gpio17 tx
"""
import uasyncio as asyncio
from machine import UART
uart = UART(2, 9600)

async def sender():
    swriter = asyncio.StreamWriter(uart, {})
    while True:
        await swriter.awrite('Hello uart. \n')
        print('Wrote')
        await asyncio.sleep(2)

async def receiver():
    sreader = asyncio.StreamReader(uart)
    while True:
        res = await sreader.readline()
        print('Recieved', res)

loop = asyncio.get_event_loop()
loop.create_task(sender())
loop.create_task(receiver())
loop.run_forever()
```

Réception GPS sur l'UART 2 avec asyncio
A tester : https://github.com/alexmrqt/micropython-gps/blob/master/adafruit_gps.py

```
"""
esp32 pinout
gpio16 rx relié à la sortie GPS
gpio17 tx
"""
import uasyncio as asyncio
from machine import UART
uart = UART(2, 9600)

async def receiver():
    sreader = asyncio.StreamReader(uart)
    while True:
        res = await sreader.readline()
        nmea=res.split(b',')
        if nmea[0]==b'$GPGGA':
            #print('Recieved', nmea)
            print(int(float(nmea[1].decode())),end=',')
            print('latitude :',float(nmea[2].decode()),end=',')
            print(' longitude :',float(nmea[4].decode()))


loop = asyncio.get_event_loop()
loop.create_task(receiver())
loop.run_forever()
```

133405,latitude : 4753.415, longitude : 16.6092
133406,latitude : 4753.415, longitude : 16.6092
133407,latitude : 4753.415, longitude : 16.6092

Détecter un code infra-rouge RC5

https://www.st.com/content/ccc/resource/technical/document/application_note/c7/d1/63/f7/80/06/41/a4/CD00267896.pdf/files/CD00267896.pdf/jcr:content/translations/en.CD00267896.pdf

```
# ir 13

from machine import Pin
from time import sleep_us, ticks_us,
ticks_diff
from neopixel import NeoPixel

ir = Pin(13, Pin.IN)
led = Pin(0, Pin.OUT)
np = NeoPixel(Pin(14), 4)

time_start = ticks_us()
code = 2
actualBit = 0
lastBit = 1
bit = 1
trigger = 0


def callback(p):
    global time_start, code, actualBit,
lastBit, bit, trigger
    edge = ir.value()
    time_stop = ticks_us()
    delta = ticks_diff(time_stop, time_start)
    time_start = time_stop
    if delta > 2000:
        if bit == 13:
            print('-> ', code & 0x3f)
            trigger = code & 0x3f
        code = 2
        lastBit = 1
        actualBit = 0
        bit = 0
        led.value(led.value() ^ 1)
    else:
        long = 0
        valid = 0
        if delta > 1100:
            long = 1
        if edge == 1:
            if lastBit == 1:
                if long == 1:
                    actualBit = 0
                    lastBit = 1
                    valid = 1
            if lastBit == 0:
                if long == 0:
                    actualBit = 0
                    lastBit = 0
                    valid = 1
        if edge == 0:
            if lastBit == 1:
                if long == 0:
                    actualBit = 1
                    lastBit = 1
                    valid = 1
            if lastBit == 0:
                if long == 1:
                    actualBit = 1
                    lastBit = 0
                    valid = 1
        if valid == 1:
            bit += 1
            code |= actualBit
            if bit < 13:
                code <<= 1
            lastBit = actualBit


ir.irq(trigger=Pin.IRQ_RISING |
Pin.IRQ_FALLING, handler=callback)

couleur = (0, 0, 0)

while (True):
    if trigger == 1:
        couleur = (255, 0, 0)
    elif trigger == 2:
        couleur = (0, 255, 0)
    elif trigger == 3:
        couleur = (0, 0, 255)

    for n in range(0, 4):
        np[n] = couleur
    np.write()
```

Contrôler un périphérique DMX


```
"""
routines DMX configuré ici pour l'esp32
esp 8266 uart 1
esp 32   uart 2
        esp8266   esp32
broche   2(9)     17(4)
"""
from machine import UART
import machine, time
from array import array

dmx_message = array('B', [0] * 16) # 16 channels

def set_channels(message):
    for ch in message:
        dmx_message[ch] = message[ch]

def write_frame():
    #dmx_uart = machine.Pin(2,machine.Pin.OUT)
    dmx_uart = machine.Pin(17, machine.Pin.OUT)
    dmx_uart.value(0)
    time.sleep_us(74)
    dmx_uart.value(1)
    # Now turn into a UART port and send DMX data
    # dmx_uart = UART(1)
    dmx_uart = UART(2)
    dmx_uart.init(250000, bits=8, parity=None, stop=2)
    #send bytes
    dmx_uart.write(dmx_message)
    #Delete as its going to change anyway
    del(dmx_uart)

set_channels({1:80})
set_channels({2:255})
set_channels({3:174})
set_channels({4:255})

print(dmx_message)

write_frame()
```