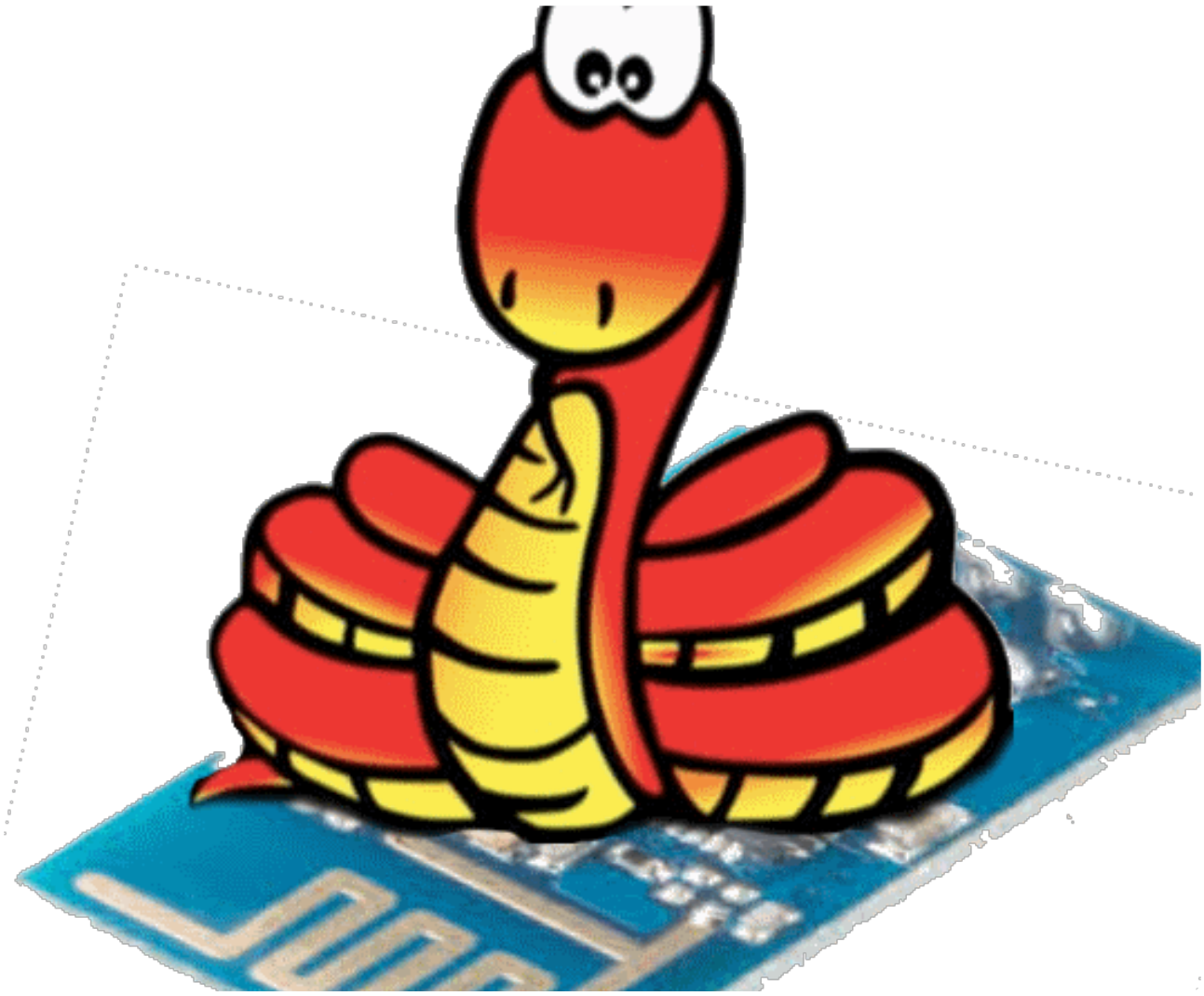


Python sur des Micro-contrôleurs ESP8266

Auteur : Fabien Roca · **Publié le** 22/10/2019 ·

Ce projet montre comment préparer son ordinateur pour travailler avec les microcontrôleurs de type ESP8266, ces derniers étant compatibles avec les langages C pour Arduino, blocs mais aussi le Python.

Les microcontrôleurs usuels du type Arduino (Atmega) sont souvent utilisés en bluetooth avec des applications générées via App Inventor. Le gros défaut est qu'il n'est pas possible de créer une application sur-mesure pour les téléphones à la pomme ou tout simplement certains ordinateurs dépourvus de ce type de connexion. L'avantage de l'ESP est la liaison wifi, il peut être tout simplement client ou serveur.

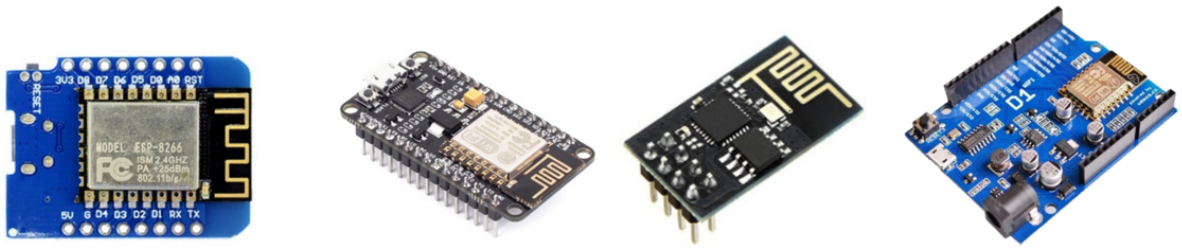
> **Télécharger tout d'abord le fichier microPython.7z fourni dans cette page.**

Étapes du projet

ÉTAPE 1

Présentation des ESP8266

Bien qu'il en existe différentes versions, vous pourrez utiliser la carte D1 car elle est au format Arduino UNO. Elle permet de brancher son matériel Arduino classique (Shield Grove, modules...) comme sur n'importe quelle carte UNO tout en bénéficiant de la puissance web d'une ESP8266. Elle est également compatible Arduino augmenté ou easycoding pour la programmation en blocs, avec l'environnement logiciel Arduino mais surtout ce qui va nous intéresser ici : avec Python.



ÉTAPE 2

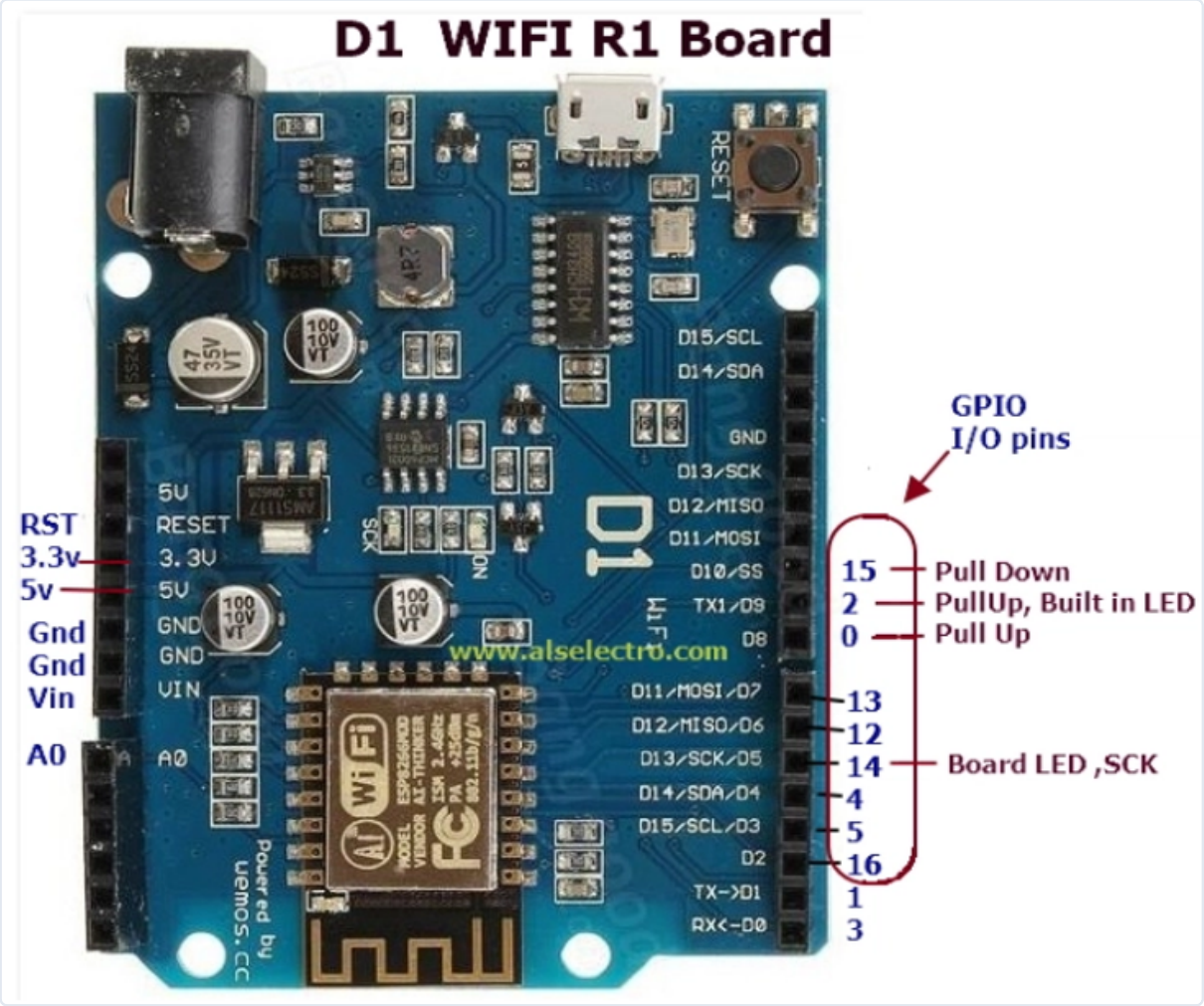
Avantages et inconvénients

Les cartes ESP8266 ressemblent énormément aux habituels Arduinos mais elles ne présentent pas les mêmes pins (ports ou GPIO). Un inconvénient est, sur la plupart, la présence d'un seul port analogique.

En revanche, elles possèdent plusieurs ports séries et tous les ports numériques (digitaux) permettent la PWM (Pulse Width Modulation: nous verrons dans un autre projet son utilisation).

Elles permettent aussi d'être des clients wifi (se connecter à des boxes, des partages de connexion...) mais aussi des serveurs et donc accueillir par exemple des petites pages web.

Pour finir ces cartes possèdent de l'espace de stockage pour les programmes et les données. Appelé SPIFFS, cette mémoire varie suivant les modèles de 512 Ko à 4Mo.



ÉTAPE 3

Préparation matérielle et logicielle

Préparation des ordinateurs :

- Installer les pilotes et les bibliothèques pour les cartes ESP sur l'ordinateur Pilote CH340 :

https://sparks.gogo.co.nz/assets/_site_/downloads/CH34x_Install_Windows_v3_4.zip

ou https://wiki.wemos.cc/downloads

- Télécharger le logiciel uPyCraft (fourni dans le kit ici en version 1.1):

https://github.com/DFRobot/uPyCraft

ou http://docs.dfrobot.com/upycraft

Préférez télécharger le pack complet en début de projet.

- Télécharger et installer le logiciel Python 3.8 si ce n'est pas déjà fait :

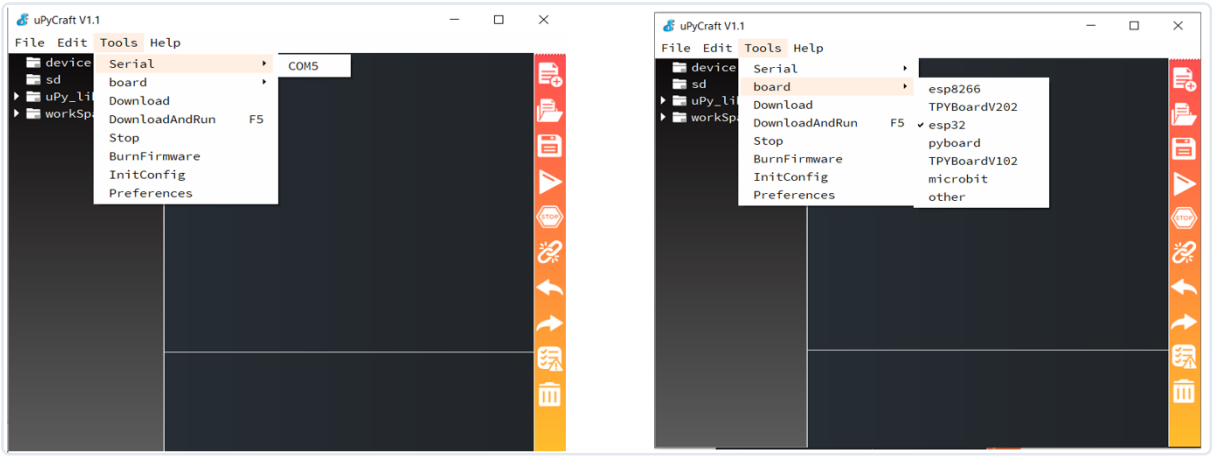
https://www.python.org/downloads/

Préparation du micro-contrôleur :
Si celui-ci n'est pas prévu pour python initialement, il faudra installer le firmware téléchargeable à l'adresse :
http://micropython.org/download

Il s'agit d'un fichier binaire de la forme : esp8266-20190125-v1.10.bin

Remarque: si vous utilisez un autre modèle, il vous faudra récupérer le firmware adapté.

Lancer uPyCraft.exe puis vérifier la présence du port série ainsi que le bon type de carte microcontrôleur.

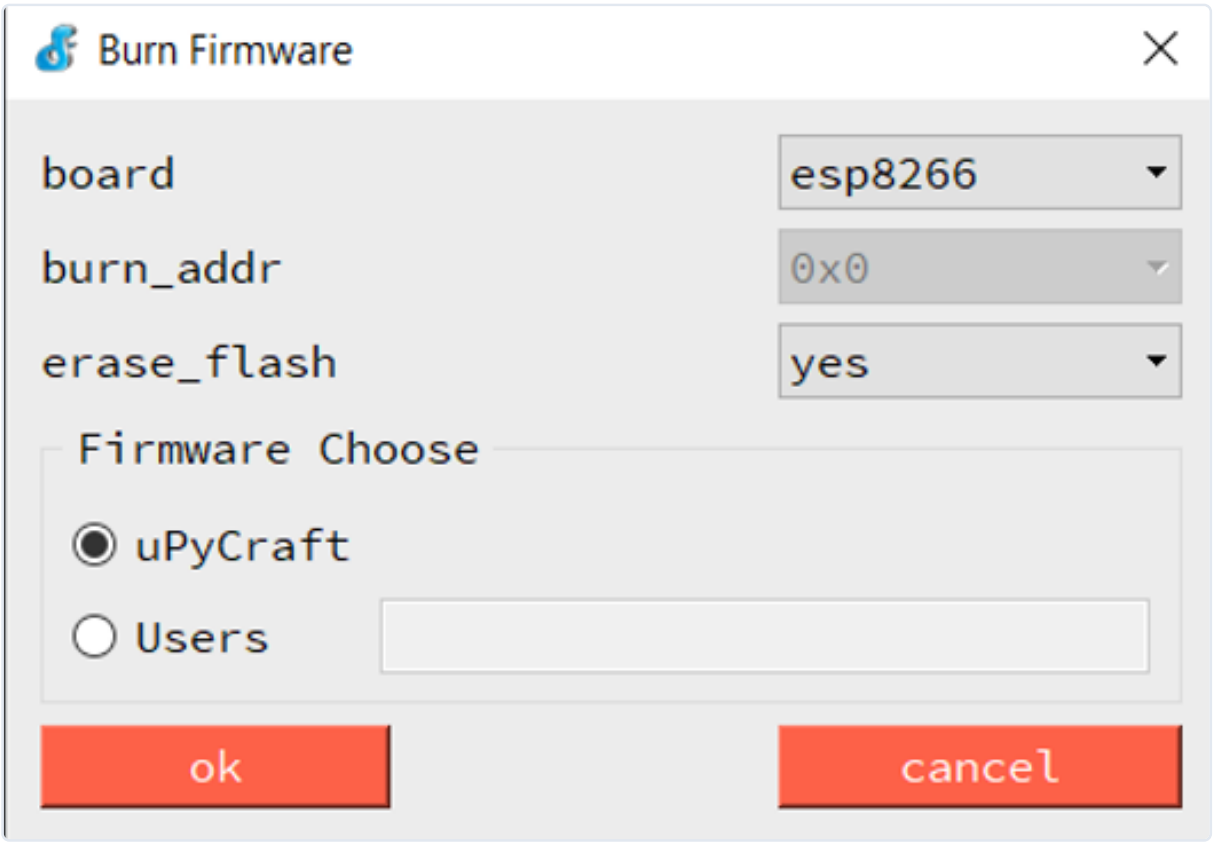

uPyCraft V1.1
File Edit Tools Help
device
sd
uPy_li
workSp
Serial
board
Download
DownloadAndRun F5
Stop
BurnFirmware
InitConfig
Preferences
COM5
esp8266
TPYBoardV202
esp32
pyboard
TPYBoardV102
microbit
other

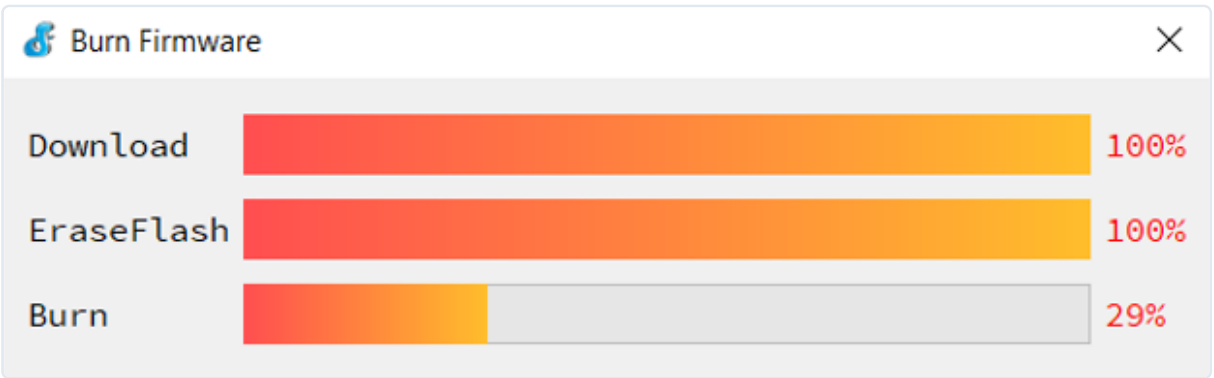
ÉTAPE 4

Flash de l'ESP

Choisissez alors les maillons de chaîne dans les boutons de droite afin de vous connecter à votre carte. Si jamais le firmware présent sur celle-ci n’est pas correct, une nouvelle fenêtre apparaît à l’écran et vous demande à nouveau de choisir le type de carte et le firmware à installer.

Commencez par choisir **uPyCraft** puis **Ok.** Si une erreur vous est signalée, choisissez **Users** et allez chercher le firmware que vous avez téléchargé précédemment. Encore une fois, pas de panique, si vous ne choisissez pas le bon firmware une erreur vous sera signalée, sinon tout se lance et se "Burn".





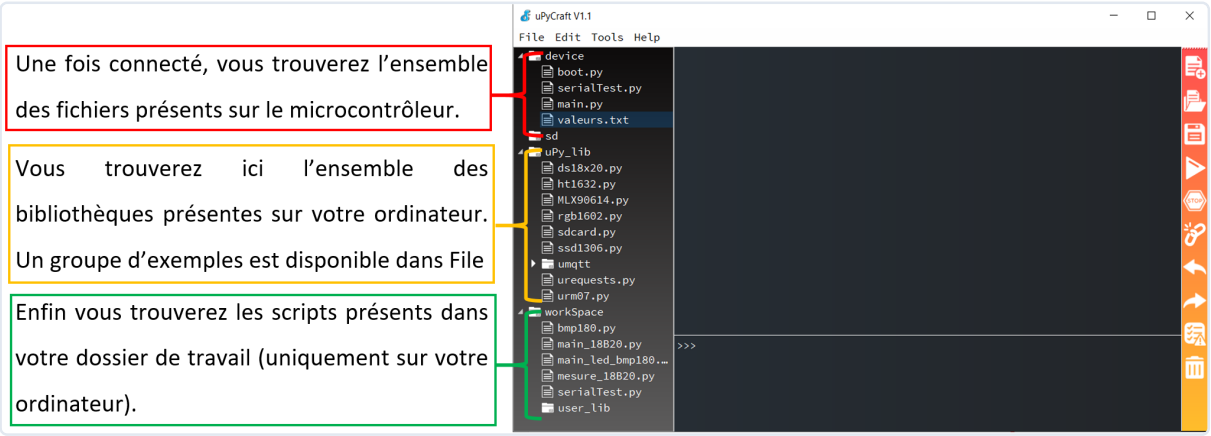
ÉTAPE 5

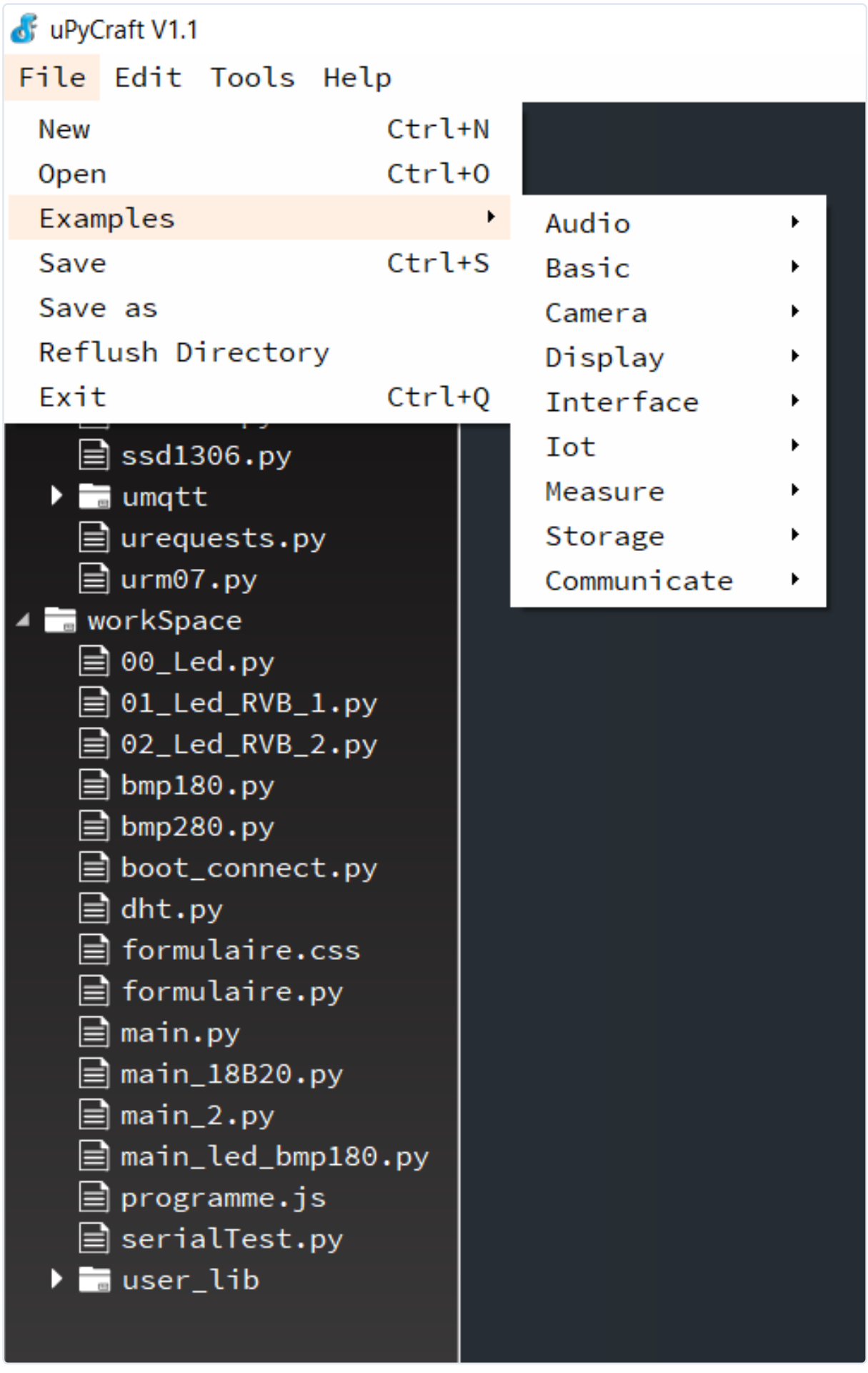
Présentation de uPyCraft

Le logiciel uPyCraft ne demande pas d'installation, il vous suffira de décompresser le dossier et de lancer l'exécutable.

Il est composé de plusieurs parties permettant de programmer en python mais aussi d'interagir avec le microcontroleur. Il est important de noter que la version fournie dans ce sujet possède déjà des bibliothèque de capteurs mais aussi des exemples qui vous permettront d'avancer plus vite.

Remarque : L'ESP, lorsqu'il est juste alimenté électriquement, lit en premier le fichier **boot.py** puis le fichier **main.py** si celui-ci existe. Si vous souhaitez donc réaliser un capteur embarqué, il est vous faudra nommer votre script en conséquence.




uPyCraft V1.1
File Edit Tools Help
New Ctrl+N
Open Ctrl+O
Examples
Save Ctrl+S
Save as
Reflush Directory
Exit Ctrl+Q
Audio
Basic
Camera
Display
Interface
Iot
Measure
Storage
Communicate
ssd1306.py
umqtt
urequests.py
urm07.py
workSpace
00_Led.py
01_Led_RVB_1.py
02_Led_RVB_2.py
bmp180.py
bmp280.py
boot_connect.py
dht.py
formulaire.css
formulaire.py
main.py
main_18B20.py
main_2.py
main_led_bmp180.py
programme.js
serialTest.py
user_lib

ÉTAPE 6

La base : allumer des Led

Il faut tout d'abord trouver le nom des pins du microcontrôleur utilisé (NodeMCU, D1 Mini...) et préparer le câblage de chacune des "pattes" de la Led RVB. Il s'agit ici du seul cas où la broche la plus longue correspond à la masse. Ensuite chacune correspond à une couleur.

Remarque: attention, il est possible de faire "griller" une seule couleur (souvent le rouge) si vous lui fournissez une trop haute tension et/ou un trop fort courant. Protégez vos Led avec des résistances entre 110 et 220 Ohms.

Nous branchons pour commencer la plus longue pin à la broche G de l'ESP puis une broche directement à la pin D1 (on évitera la D0 celle-ci ayant d'autres fonctions).

Attention: Certaines LED ne sont pas à anode commune mais à cathode, il ne faut donc pas relier la grande pin à la masse (G) mais au + (3,3V).

On voit que la D1 s'appelle GPIO5 donc on peut demander à uPyCraft de créer un nouveau programme tel que :

import time #on importe la fonction temps

from machine **import** Pin #on importe la possibilité de commander les pin de l'Esp

led=Pin(**5**,Pin.OUT) #On définit notre Led comme connectée à la pin D1 (GPIO5)

#On réalise une boucle **infinie**:

while True:

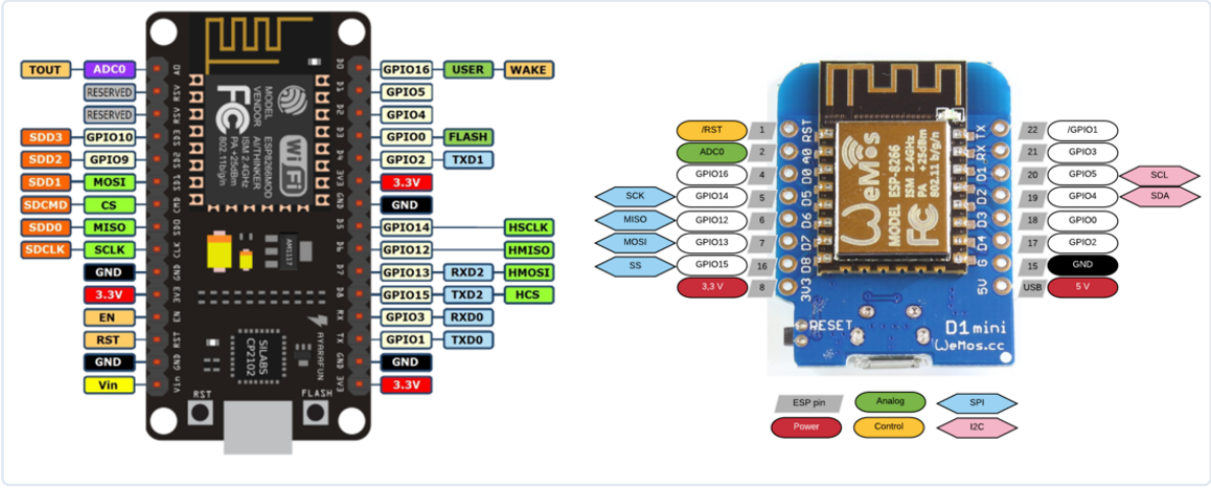
led.value(**1**) #On allume la Led

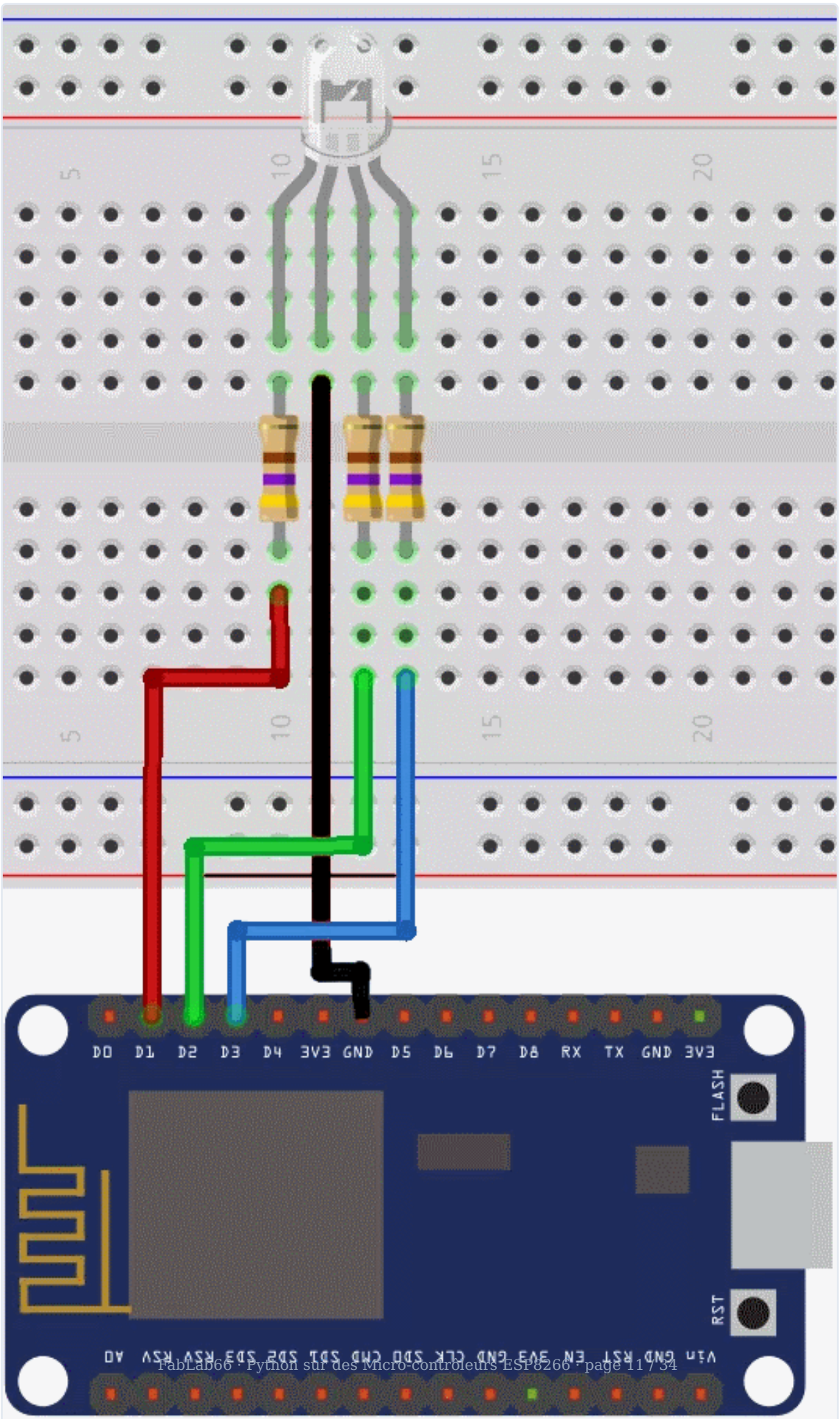
time.sleep(**0.5**) #On attend une demie seconde

led.value(**0**) #On éteint la Led

time.sleep(**0.5**) #On attend une demie seconde

#On recommence !


TOUT
ADC0
RESERVED
RESERVED
SDD3
GPIO10
SDD2
GPIO9
SDD1
MOSI
SDCMD
CS
SDD0
MISO
SDCLK
SCLK
GND
3.3V
EN
RST
GND
Vin
GPIO16
USER
WAKE
GPIO5
GPIO4
GPIO0
FLASH
GPIO2
TXD1
3.3V
GND
GPIO14
HSCLK
GPIO12
HMISO
GPIO13
RXD2
HMOSI
GPIO15
TXD2
HCS
GPIO3
RXD0
GPIO1
TXD0
GND
3.3V
/RST
ADC0
GPIO16
SCK
GPIO14
MISO
GPIO12
MOSI
GPIO13
SS
GPIO15
3.3 V
/GPIO1
GPIO3
GPIO5
SCL
GPIO4
SDA
GPIO0
GPIO2
GND
5 V
USB
RESET
D1 mini
WeMos.cc
ESP pin
Analog
SPI
Power
Control
I2C


D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
RST

```
1  import time
2  from machine import Pin
3  led=Pin(5,Pin.OUT)
4
5  while True:
6      led.value(1)
7      time.sleep(0.5)
8      led.value(0)
9      time.sleep(0.5)
```

ÉTAPE 7

La suite : allumage progressif

On a parlé au début du projet de **PWM** qui n'est rien d'autre qu'un signal alternant **LOW** et **HIGH** à haute fréquence. La LED est donc alternativement allumée et éteinte mais le cycle est tellement rapide que la persistance rétinienne nous donne l'illusion d'une LED allumée en permanence. Si la LED est allumée pendant 1ms et éteinte pendant 9ms, l'impression sera d'une luminosité de 10%. Le pourcentage de temps passé à l'état HIGH sur la période du signal est appelé le **rapport cyclique (duty)**. Le rapport cyclique n'est pas donné de 0 à 100 mais de 0 à 1023. Il sera nécessaire de faire un produit en croix...

1. Réalisons une boucle qui fera évoluer petit à petit ce rapport cyclique pour obtenir une luminosité maximale après environ 5 secondes

```
from machine import Pin, PWM

import time

pwmB = machine.PWM(machine.Pin(5))

pwmB.duty(0)

pwmR = machine.PWM(machine.Pin(4))

pwmR.duty(0)

pwmV = machine.PWM(machine.Pin(0))

pwmV.duty(0)


for i in range(1024):

pwmB.duty(i)

time.sleep(0.005)
```

2. Réalisons à présent une boucle infinie qui fera évoluer petit à petit ce rapport cyclique pour obtenir une luminosité maximale après environ 5 secondes puis une extinction progressive. Cela affectera chacune des couleurs les unes après les autres avant de recommencer.

```
from machine import Pin, PWM

import time
```

```
pwmB = machine.PWM(machine.Pin(5))
pwmB.duty(0)
pwmR = machine.PWM(machine.Pin(4))
pwmR.duty(0)
pwmV = machine.PWM(machine.Pin(0))
pwmV.duty(0)
while True:
for i in range(1024):
pwmR.duty(i)
time.sleep(0.005)
for i in range(1023, -1, -1):
pwmR.duty(i)
time.sleep(0.005)
for i in range(1024):
pwmV.duty(i)
time.sleep(0.005)
for i in range(1023, -1, -1):
pwmV.duty(i)
time.sleep(0.005)
for i in range(1024):
pwmB.duty(i)
time.sleep(0.005)
for i in range(1023, -1, -1):
pwmB.duty(i)
time.sleep(0.01)
```

```
from machine import Pin, PWM
import time
pwmB = machine.PWM(machine.Pin(5))
pwmB.duty(0)
pwmR = machine.PWM(machine.Pin(4))
pwmR.duty(0)
pwmV = machine.PWM(machine.Pin(0))
pwmV.duty(0)

for i in range(1024):
    pwmB.duty(i)
    time.sleep(0.005)
```

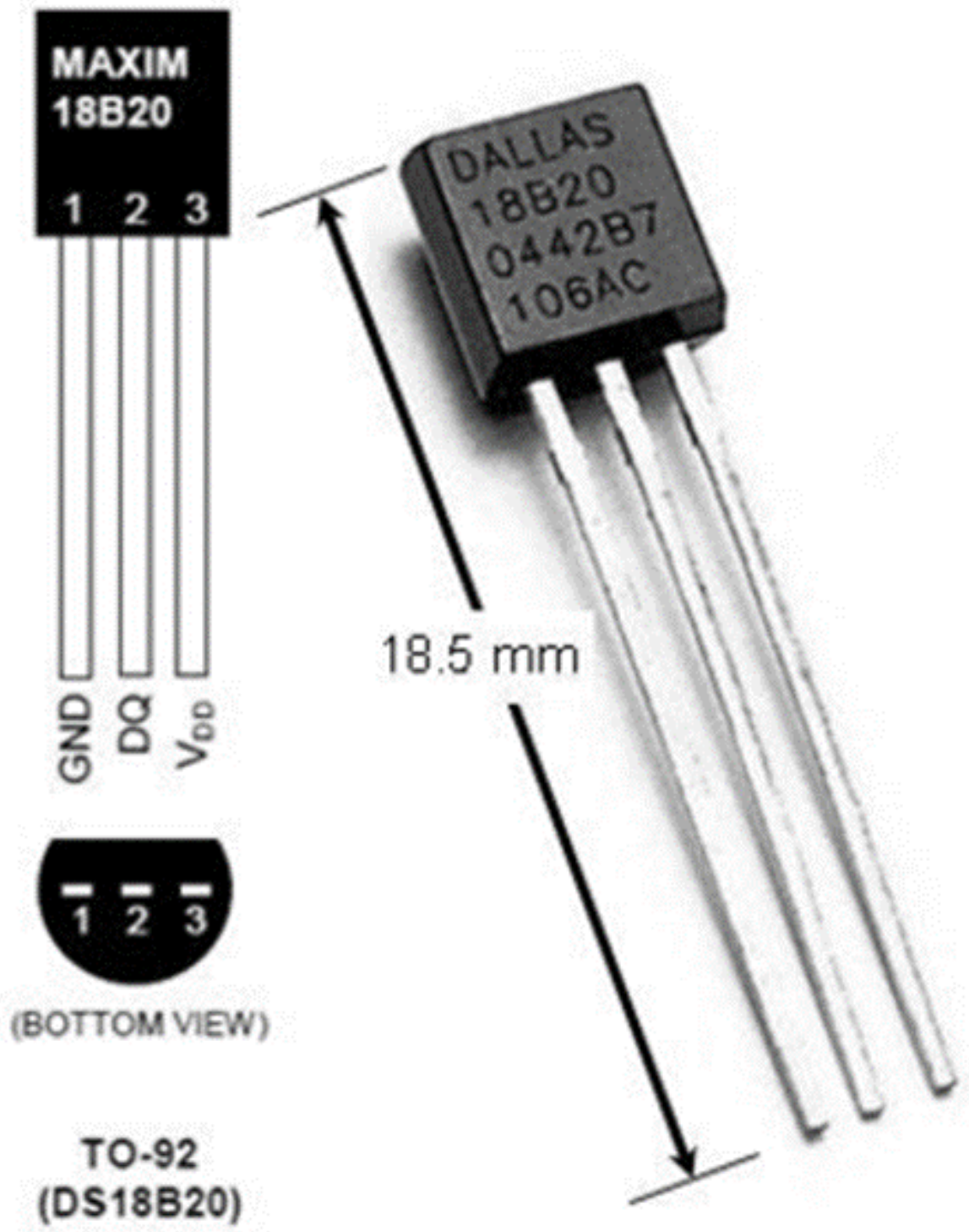
```
from machine import Pin, PWM
import time
pwmB = machine.PWM(machine.Pin(5))
pwmB.duty(0)
pwmR = machine.PWM(machine.Pin(4))
pwmR.duty(0)
pwmV = machine.PWM(machine.Pin(0))
pwmV.duty(0)
while True:
    for i in range(1024):
        pwmR.duty(i)
        time.sleep(0.005)
    for i in range(1023, -1, -1):
        pwmR.duty(i)
        time.sleep(0.005)
    for i in range(1024):
        pwmV.duty(i)
        time.sleep(0.005)
    for i in range(1023, -1, -1):
        pwmV.duty(i)
        time.sleep(0.005)
    for i in range(1024):
        pwmB.duty(i)
        time.sleep(0.005)
    for i in range(1023, -1, -1):
        pwmB.duty(i)
        time.sleep(0.01)
```

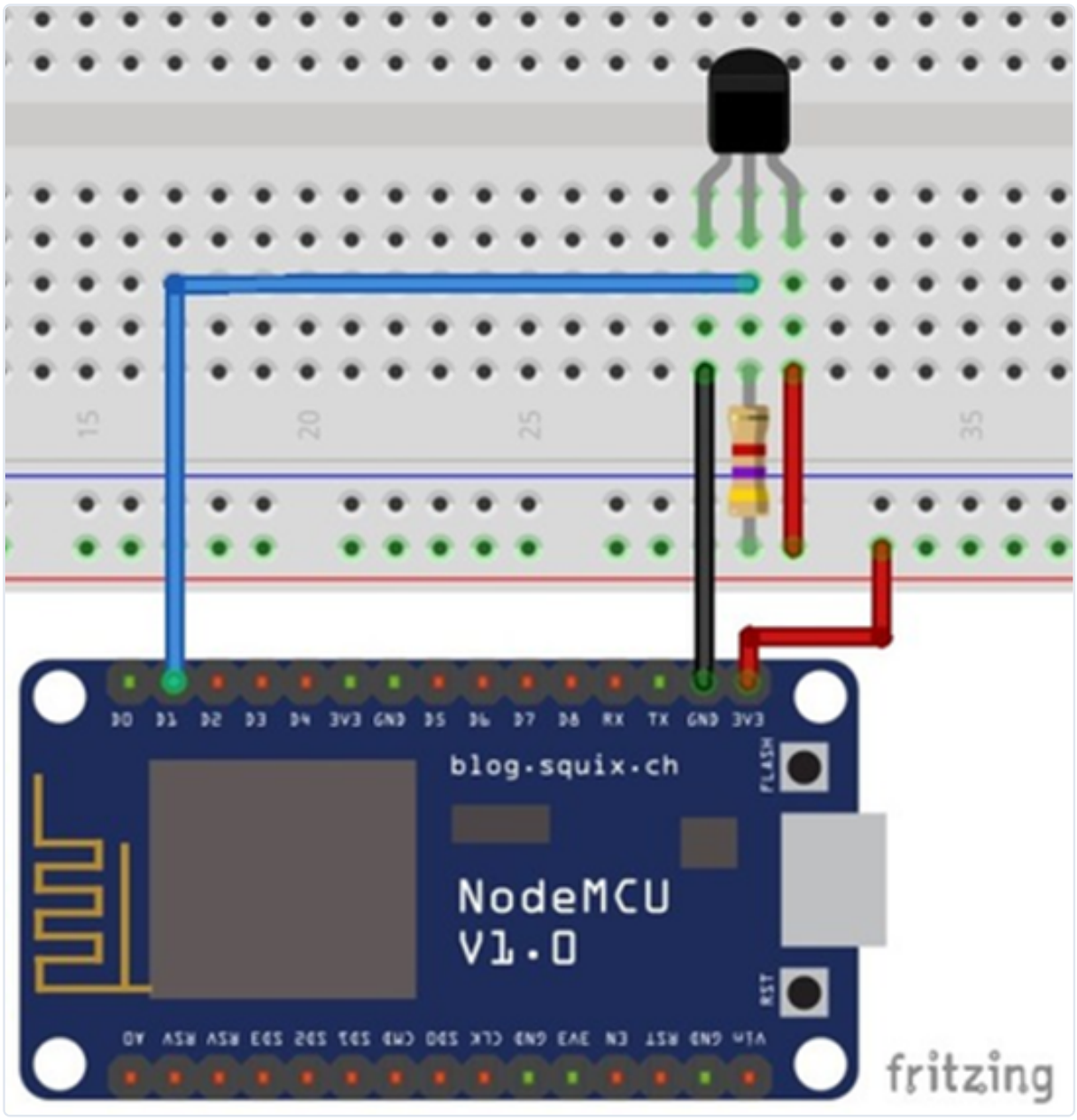
ÉTAPE 8

Mesure de température: Dallas 18B20

Comme pour beaucoup de matériel dans nos ordinateurs, des drivers sont nécessaires pour faire fonctionner les capteurs. En langage de programmation, ces drivers sont appelés librairies.

On utilisera à nouveau la pin D1 et on veillera à placer une résistance de 4,7kOhms entre cette broche et l'alimentation (3,3V).


MAXIM
18B20
1 2 3
GND
DQ
VDD
1 2 3
(BOTTOM VIEW)
TO-92
(DS18B20)
DALLAS
18B20
0442B7
106AC
18.5 mm


blog.squix.ch
NodeMCU
V1.0
FLASH
RST
fritzing

ÉTAPE 9

Mesure de température: BMP 180

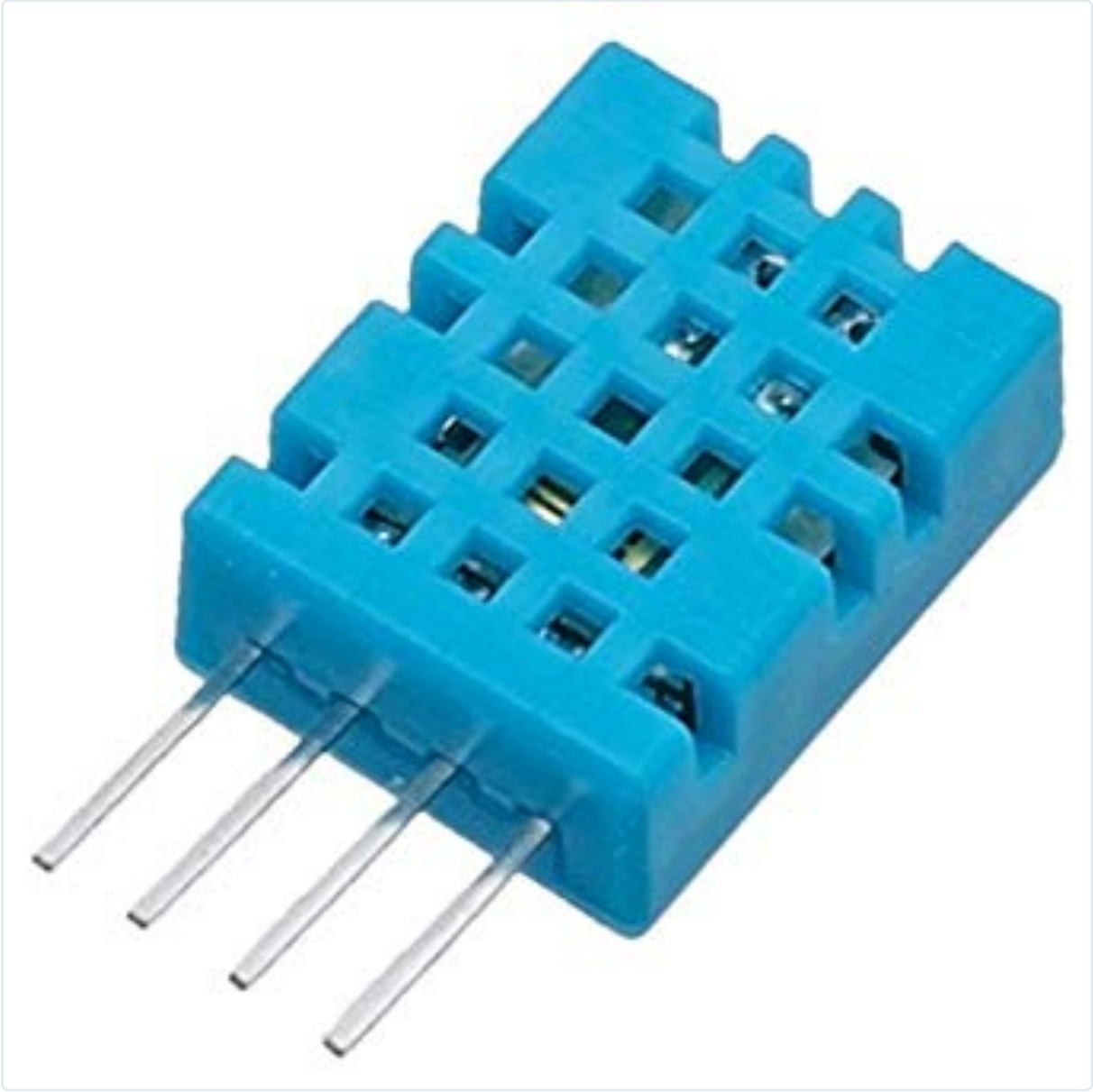
ÉTAPE 10

Mesure de température et Pression : BMP280

ÉTAPE 11

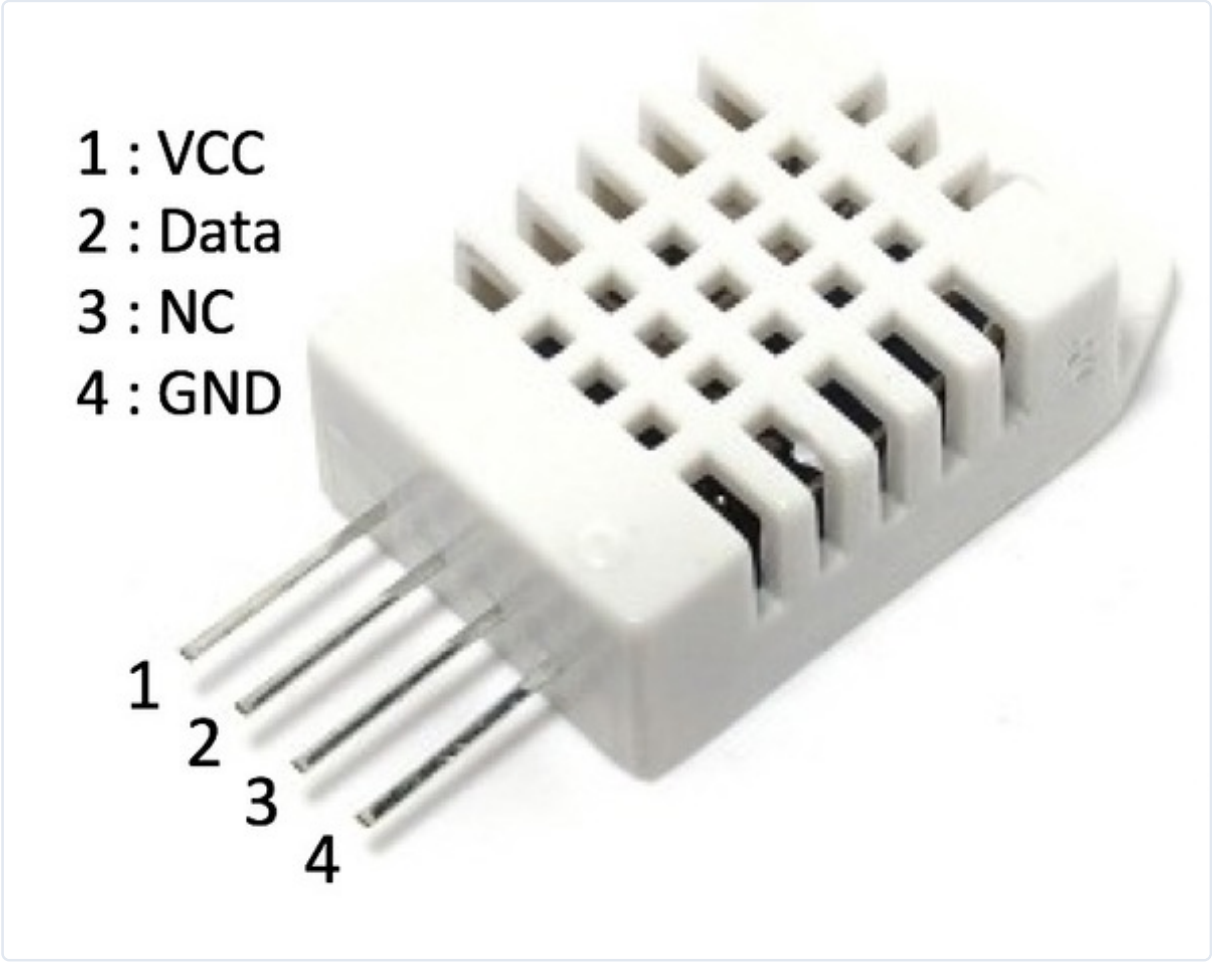
Mesure de Température et Humidité : DHT 11

```
import dht, time
import machine
d = dht.DHT11(machine.Pin(4))
while True:
  print(d.measure())
  print(d.temperature())
  print(d.humidity())
  time.sleep(3)
```



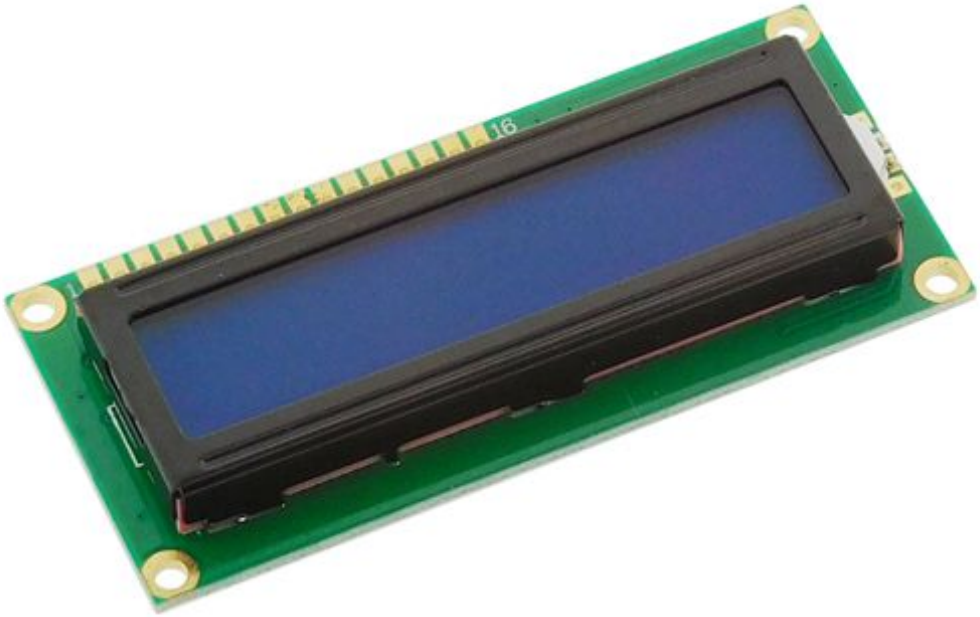
ÉTAPE 12

Mesure de Température et Humidité : DHT 22



ÉTAPE 13

Afficher des informations sur un afficheur LCD 16x2



ÉTAPE 14

[EN COURS] Capteur de distance HC-SR04

#
Mesure de distance
#
Programme permettant d’utiliser la librairie qui gere le capteur
de distance ultrason HC-SR04
Tester sur NodeMCU Lolin et Wemos D1 mini
#
Auteur iTechnoFrance
#

Connecter le VCC au Vin de l'ESP ainsi que le GND au GND de l'ESP, les pin Echo et Trigger doivent être connecter a des pin sur l'ESP (dans l'exemple GPIO04 et GPIO05).

```
import hcsr04, time
# pin_echo : pin pour mesurer la distance
# pin_trig : pin pour envoyer les impulsions
pin_echo = 4 &amp;nbsp;# sortie D2 —&amp;gt; GPIO04
pin_trigger = 5 &amp;nbsp;# sortie D1 —&amp;gt; GPIO05
# initialisation librairie HC-SR04
hc_sr04 = hcsr04.HCSR04(pin_echo, pin_trigger)
while True:
&amp;nbsp; &amp;nbsp; distance = hc_sr04.distance_cm()
&amp;nbsp; &amp;nbsp; if (distance == -1):
&amp;nbsp; &amp;nbsp; &amp;nbsp; &amp;nbsp; print ("Mesure superieure à 4 metres")
&amp;nbsp; &amp;nbsp; else:
&amp;nbsp; &amp;nbsp; &amp;nbsp; &amp;nbsp; print ((distance), " cm ")
&amp;nbsp; &amp;nbsp; time.sleep(1)
```


R
T
GND
Echo
Trig
VCC

ÉTAPE 15

Détecteur de collision

Sachant qu'il s'agit d'un capteur numérique (qui ne sort que 0 ou 1) sur la Pin OUT.

Pour transformer une pin en entrée (lire un capteur) il faut utiliser la commande :

Entree=Pin(5,Pin.IN)

Et pour lire sa valeur, il vous suffit de taper :

Entree.value()

Exemple de code:

```
import time
from machine import Pin
entree=Pin(5,Pin.IN)
while True:
  alume=entree.value()
  time.sleep(0.05)
  if alume==1:
    print("1")
  elif alume==0:
   print("0")
```

ÉTAPE 16

Détecter la pression sur un bouton poussoir

On peut s'inspirer du détecteur de collision pour lire l'entrée d'une pin reliée à un bouton poussoir ou un interrupteur.

ÉTAPE 17

Utiliser un moteur à courant continu (DC)

ÉTAPE 18

Utiliser un Servo Moteur SG90

```
from machine import Pin, PWM

import time

servo = machine.PWM(machine.Pin(5), freq=50)

while True:

  time.sleep(1)

  servo.duty(40)

  time.sleep(1)

  servo.duty(115)

  time.sleep(1)

  servo.duty(77)

  time.sleep(1)
```


(PWM) Signal line
(+5V)
(0 V) GND
TIANKONGRC™
Micro Servo 9g
SG90 ANALOG

ÉTAPE 19

Utiliser un moteur pas à pas avec driver

ÉTAPE 20

Utiliser un GPS Neo6M

ÉTAPE 21

Mini Serveur : créer son propre point d'accès wifi.

ÉTAPE 22

Rejoindre un point d'accès Wifi et héberger une page html

ÉTAPE 23

Sitographie

https://dfrobot.gitbooks.io/upycraft/content/

https://randomnerdtutorials.com/category/0-esp32-micropython/

https://itechnofrance.wordpress.com/2018/02/21/micropython-et-le-capteur-de-distance-ultrason-hc-sr04/