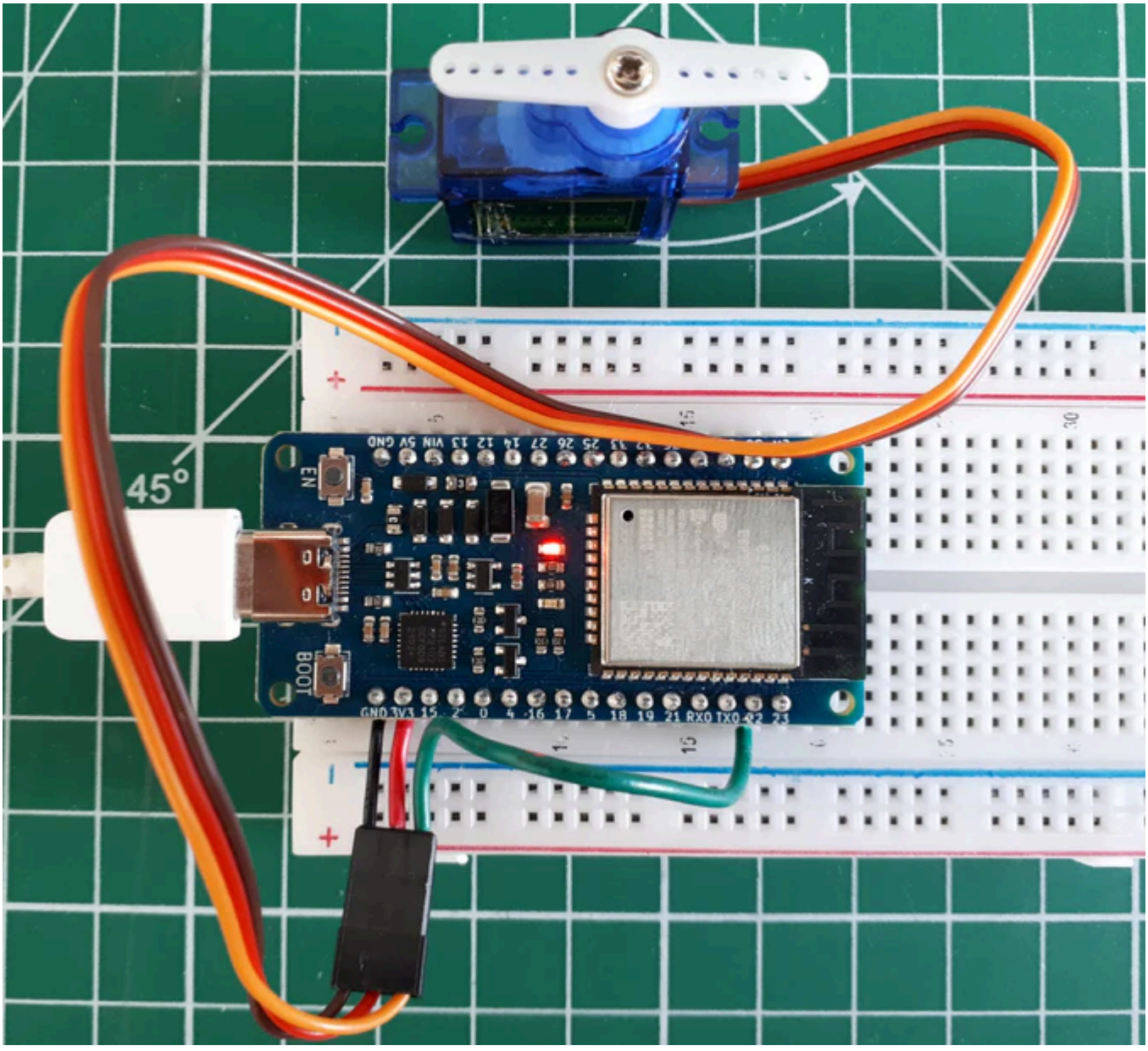
Utiliser un servomoteur avec une carte ESP32 en MicroPython

(Mis à jour le 23/01/2023)



Les servomoteurs, fréquemment raccourcis à « servo », sont une forme particulière de moteurs qui peuvent être fixés à une position spécifique avec une grande précision. Cette position est maintenue pour jusqu'à ce qu'une nouvelle instruction soit donnée.

Ils ont un excellent rapport poids puissance. Le populaire servo bleu SG90 de TowerPro a un couple de 1,5 kg/cm pour seulement 9g. Son petit prix et sa facilité de commande à partir d'un ESP32 en font un choix populaire pour les makers !

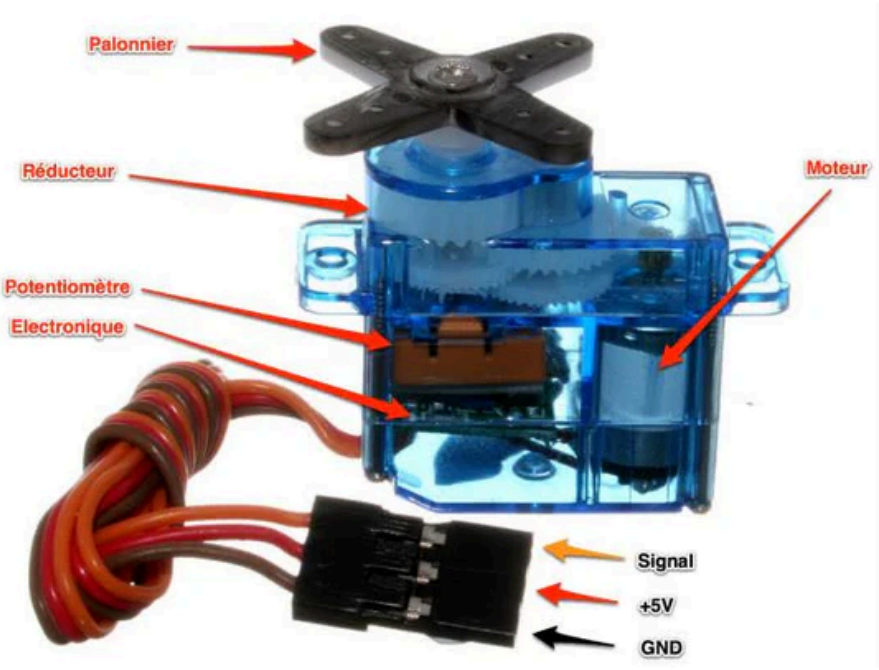
> **Note**
>
> Les servos sont très utilisés dans le modélisme (direction des roues des voitures télécommandées, commande des gouvernes de dérive et de profondeur sur les avions, etc.), mais aussi dans la robotique et l'industrie, par exemple pour réguler des flux de liquides dans des vannes.

Prendre en main le servomoteur SG90

La principale différence entre un servomoteur et un moteur classique est que la plupart des servomoteurs peuvent tourner uniquement entre 0 et + 180 degrés. Un servomoteur est composé d'un moteur courant continu (CC) classique, de différents engrenages pour augmenter le couple (et réduire la vitesse) ainsi que le système d'asservissement. La prouesse est d'avoir réussi à empaqueter tout ce mécanisme dans un boîtier aussi petit !

> **Avertissement**
>
> La butée en plastique du servomoteur qui limite la rotation est relativement fragile. Il faut éviter de tourner manuellement l'axe du servomoteur et de forcer en butée.

Fonctionnement d'un servomoteur

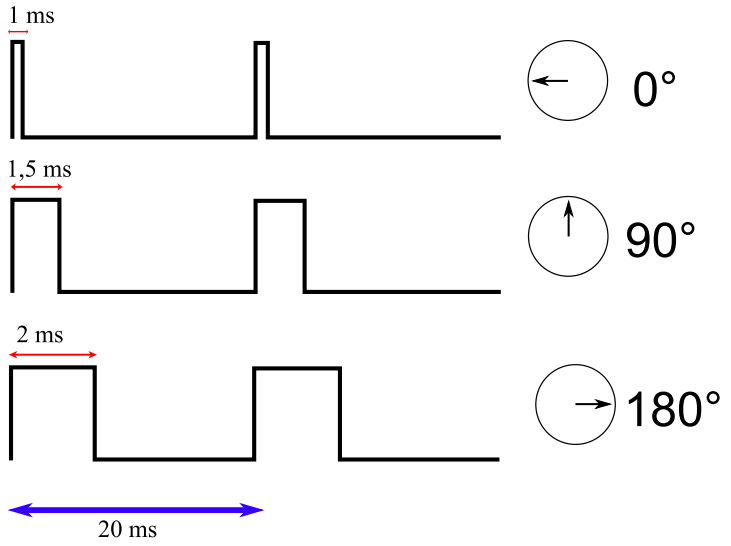


Un petit moteur CC est relié à un potentiomètre par l'intermédiaire d'un circuit électronique, ce qui permet de réguler finement la vitesse du moteur selon la position du potentiomètre. Une série d'engrenages est attachée à l'axe de sortie du moteur, afin de multiplier le couple tout en réduisant sa vitesse de rotation. Lorsque le moteur tourne, les engrenages entraînent le mouvement du bras qui à son tour actionne le potentiomètre. Si le mouvement s'arrête, le circuit électronique ajuste en continu la vitesse du moteur pour maintenir le potentiomètre et donc le bras à la même position.

Cette fonctionnalité est notamment très pratique pour les bras d'un robot qui ne retombe pas sous l'effet de son propre poids lorsque le mouvement s'arrête.

> **Note**
>
> C'est en quelque sorte un moteur intelligent 🙂

Ce petit servomoteur est commandé en utilisant un signal modulé en largeur d'impulsion (PWM) de 50 Hz de fréquence, soit une impulsion toute les 20ms. La position du servomoteur est déterminée par la durée des impulsions, généralement variant entre 1ms et 2ms.



Comment alimenter le servomoteur avec un ESP32

Dans la fiche technique du servo SG90 , l'alimentation optimale est de 5V. Ceci dit, il semblerait fonctionner également en 3.3V.

> **Note**
>
> Avec une tension de 5V, la rotation sera légèrement plus rapide !

Un servomoteur consomme beaucoup de courant, surtout quand il exerce un couple important. Puisque la broche 5V de la plupart des cartes ESP32 provient directement du bus USB, vous serez limité au maximum à 500mA. Avec 1 ou 2 servomoteurs de branché l'ESP32 devrait tenir la charge.

Au-delà de 2, utilisez plutôt une alimentation distincte. Dans ce cas, assurez-vous de connecter une broche `GND` de la carte à la borne négative de l'alimentation du servomoteur 😉.

> **Avertissement**
>
> Sur les cartes ESP32 d'uPesy, le fusible auto-réarmable se déclenchera peut-être si le courant est trop élevé.

Branchements du servomoteur SG90 sur l'ESP32 d'uPesy

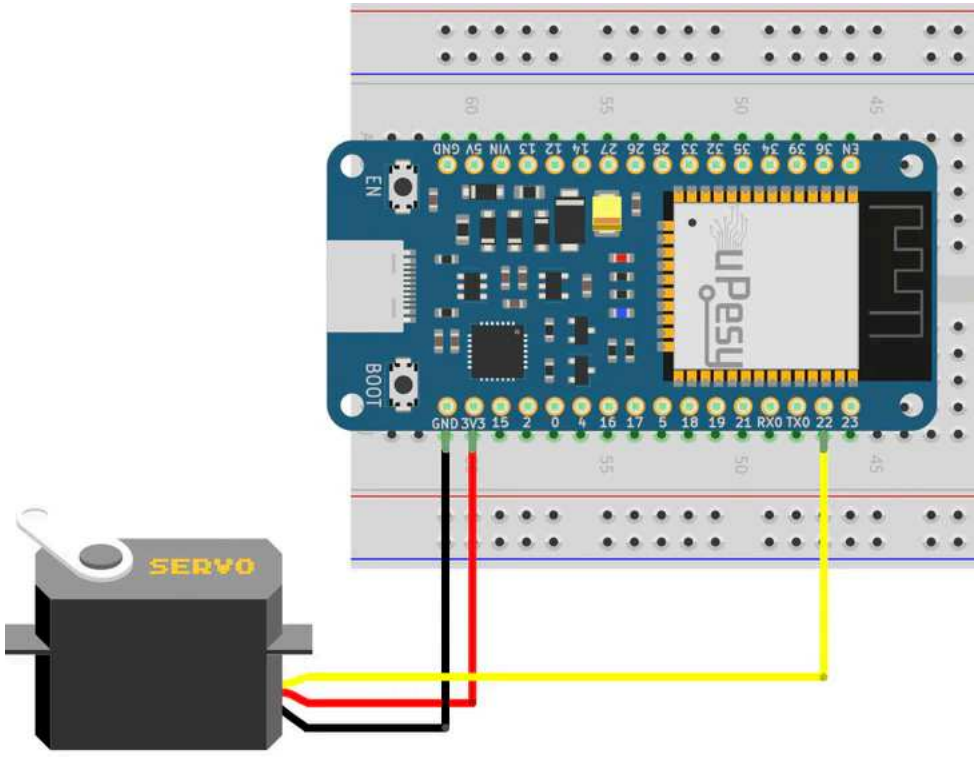
Un servomoteur SG90 contient 3 fils : 2 pour l'alimentation et 1 pour le signal de commande en PWM. Les couleurs des fils permettent de les différencier :

Servomoteur SG90	Couleur du fil	ESP32
`GND`	Marron	`GND`
`5V`	Rouge	`5V` ou `3V3`
Signal `PWM`	Orange	`GPIO22`

> **Note**
>
> Sur certains modèles de servo moteur, la couleur du fil pour le signal est jaune voire blanche au lieu d'une couleur orange.

On peut utiliser n'importe quelle broche de sortie de l'ESP32 pour contrôler le servomoteur car les broches de l'ESP32 sont toutes capables de produire une sortie PWM.

Circuit pour piloter un servomoteur avec une ESP32



Piloter un servomoteur depuis l'ESP32 avec un script Python

En fait, puisque c'est la largeur d'impulsion du signal PWM qui indique au servomoteur la position angulaire voulue, il n'y a pas vraiment besoin de librairie pour maîtriser la bête 😎. Mais cela peut venir rapidement fastidieux de calculer les bonnes valeurs, surtout quand on veut en piloter plusieurs en même temps. C'est pour cela que je recommanderai d'utiliser plutôt une librairie toute faite pour vous simplifier la vie dans nos projets DIY futurs.

Script Python basique pour piloter le servo avec ses propres calculs

Avec le servo SG90 de TowerPro, la position minimale correspond à une largeur impulsion de 0.5ms et la position maximale à une de 2.4ms. En faisant des calculs, on peut en déduire le duty-cycle du PWM, puis la valeur en bits de la largeur d'impulsion. Je ne vais pas détailler les calculs, car on va plutôt utiliser une libraire toute faite pour la suite.

> **Note**
>
> La difficulté est de trouver la bonne largeur d'impulsion PWM pour obtenir une position angulaire donnée.

Voici tout de même un script basique :

```
from machine import Pin,PWM
import time

sg90 = PWM(Pin(22, mode=Pin.OUT))
sg90.freq(50)

# 0.5ms/20ms = 0.025 = 2.5% duty cycle
# 2.4ms/20ms = 0.12 = 12% duty cycle

# 0.025*1024=25.6
# 0.12*1024=122.88

while True:
    sg90.duty(26)
    time.sleep(1)
    sg90.duty(123)
    time.sleep(1)
```

Je vous l'accorde ce n'est pas très facile à comprendre. C'est pour cela que nous allons utiliser une librairie !

Piloter un servo via un script Python avec la librairie `servo.py`

Je vous propose d'utiliser cette libraire MicroPython suivante pour facilement contrôler le servomoteur. Elle s'installe comme les autres librairies MicroPython : copier-coller le fichier sur votre carte ESP32 dans le gestionnaire de fichiers de MicroPython.

> **Avertissement**
>
> Dans la version actuelle de MicroPython pour l'ESP32 (v1.19), cette librairie n'est pas présente d'office. Seule la carte Pyboard, possède une librairie `servo` incluse de base dans MicroPython.

```
from machine import Pin, PWM

class Servo:
    # these defaults work for the standard TowerPro SG90
    __servo_pwm_freq = 50
    __min_u10_duty = 26 - 0 # offset for correction
```

```
    __max_u10_duty = 123- 0  # offset for correction
    min_angle = 0
    max_angle = 180
    current_angle = 0.001


    def __init__(self, pin):
        self.__initialise(pin)


    def update_settings(self, servo_pwm_freq, min_u10_duty, max_u10_duty, min_angle, max_angle, pi
        self.__servo_pwm_freq = servo_pwm_freq
        self.__min_u10_duty = min_u10_duty
        self.__max_u10_duty = max_u10_duty
        self.min_angle = min_angle
        self.max_angle = max_angle
        self.__initialise(pin)


    def move(self, angle):
        # round to 2 decimal places, so we have a chance of reducing unwanted servo adjustments
        angle = round(angle, 2)
        # do we need to move?
        if angle == self.current_angle:
            return
        self.current_angle = angle
        # calculate the new duty cycle and move the motor
        duty_u10 = self.__angle_to_u10_duty(angle)
        self.__motor.duty(duty_u10)

    def __angle_to_u10_duty(self, angle):
        return int((angle - self.min_angle) * self.__angle_conversion_factor) + self.__min_u10_dut


    def __initialise(self, pin):
        self.current_angle = -0.001
        self.__angle_conversion_factor = (self.__max_u10_duty - self.__min_u10_duty) / (self.max_a
        self.__motor = PWM(Pin(pin))
        self.__motor.freq(self.__servo_pwm_freq)
```

> **Avertissement**
>
> Le code de cette librairie fonctionne uniquement pour les cartes ESP32. (Le code est légèrement différent pour la Raspberry Pi Pico par exemple)

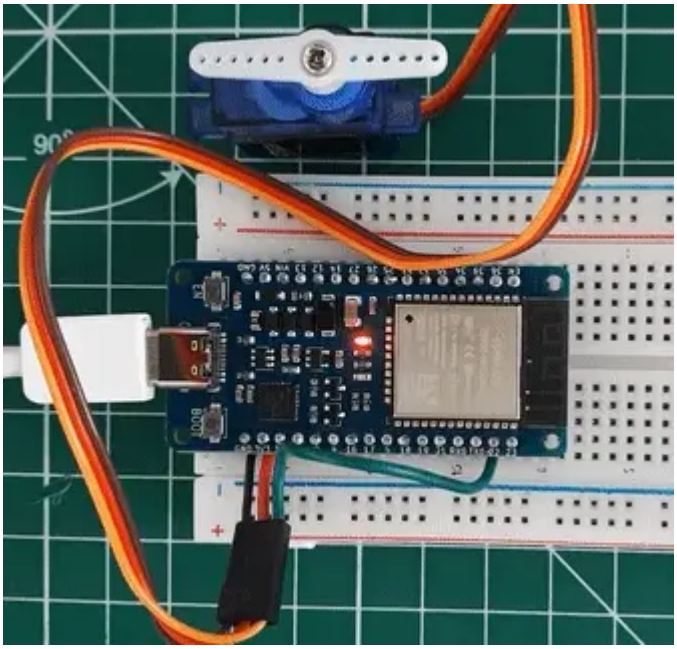
Pour utiliser la librairie, c'est extrême simple. Après avoir importé la classe `Servo` , on définit un objet `Servo` qui représente notre servomoteur bleu. On lui précise la broche utilisée pour le piloter dans les paramètres du constructeur.

Ensuite on indique la position angulaire voulue avec la fonction `.move(angle)` .

```
from servo import Servo
import time
```

```
motor=Servo(pin=22) # A changer selon la broche utilisée
motor.move(0) # tourne le servo à 0°
time.sleep(0.3)
motor.move(90) # tourne le servo à 90°
time.sleep(0.3)
motor.move(180) # tourne le servo à 180°
time.sleep(0.3)
motor.move(90) # tourne le servo à 90°
time.sleep(0.3)
motor.move(0) # tourne le servo à 0°
time.sleep(0.3)
```

Voici une vidéo de démonstration avec le code ci-dessus :



Contrôle du servo SG90 avec l'ESP32

Previous
Actionneurs

Next
Piloter des appareils électriques avec un relais et un ESP32 en MicroPython

Livraison - Retour

Soyez au courant des nouveautés d'uPesy

E-mail

En validant votre inscription, vous acceptez de recevoir par e-mail les actualités d'uPesy. Vous pouvez vous désabonner à tout moment.