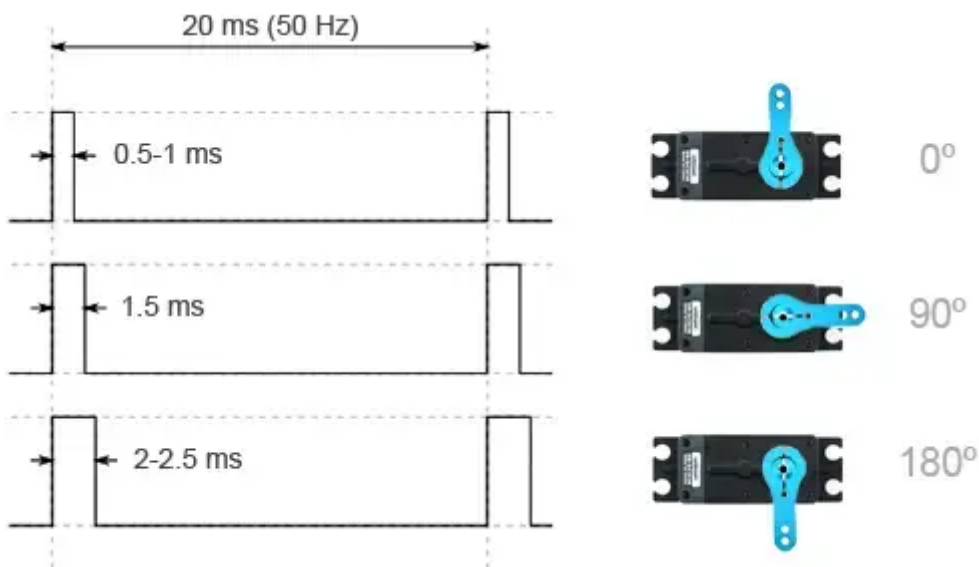


Servos

Hay varias opciones para manejar servos con micro:bit y python

Opción A: Lo más sencillo enviar un pulso adecuado

Los servos funcionan según la anchura del pulso que se envía, siendo los pulsos de 20mseg. Se explica mejor con una imagen :



Autor Luis Llamas CC-BY-SA <https://www.luisllamas.es/controlar-un-servo-con-arduino/>

1. Por lo tanto podríamos establecer primero pulsos de 20 mseg con la instrucción adecuada (por ejemplo en la puerta pin 8 sería **pin8.set_analog_period(20)**)
2. Enviar pulsos de forma adecuada. Ejemplo en puerta pin 8
 1. si queremos 0° enviamos pulsos de 1mseg que equivale a **pin8.write_analog(50)**
 2. si queremos 90° enviamos pulsos de 1.5mseg que equivale a **pin8.write_analog(75)**
 3. si queremos 180° enviamos pulsos de 2mseg que equivale a **pin8.write_analog(100)**

Los valores no responden a una regla de tres (en teoría 20mseg serían 255 en formato PWM) sino a **la experiencia-ensayo-prueba-error**. Hemos probado que para la puerta los valores 50-75-100 son correctos. Para la ventana que se ve bien la apertura y cierre los

valores son 30-60-100

Mas info en <https://support.microbit.org/support/solutions/articles/19000101864-using-a-servo-with-the-micro-bit>

Opción B Cargar una librería servo.py

1. Nos vamos a <https://github.com/microbit-playground/microbit-servo-class> y descargamos servo.py
2. Lo grabamos en la carpeta /mu_code/ donde se ha instalado el editor Mu
3. Utilizamos el código usando esta librería y poniendo los grados como grados

Por ejemplo para la puerta pin 8

```
sv1 = Servo(pin8)
sv1.write_angle(50) # turn servo to 50 degrees
```

Si quieres saber cómo se instala una librería, consulta la página LCD que ahí se ha instalado una librería <https://libros.catedu.es/books/smart-home-para-microbit/page/maqueta-lcd>

Opción C Crea tu una librería en tu programa

Esta opción esta extraída del tutorial del fabricante <https://docs.keyestudio.com/projects/KS4027-KS4028/en/latest/Python.html#project-6-servo>

No explicamos el código pues se extiende de los objetivos del curso

```
from microbit import *

class Servo:
    def __init__(self, pin, freq=50, min_us=600, max_us=2400, angle=180):
        self.min_us = min_us
        self.max_us = max_us
        self.us = 0
        self.freq = freq
        self.angle = angle
```

```
self.analog_period = 0
self.pin = pin
analog_period = round((1/self.freq) * 1000) # hertz to miliseconds
self.pin.set_analog_period(analog_period)
```

```
def write_us(self, us):
    us = min(self.max_us, max(self.min_us, us))
    duty = round(us * 1024 * self.freq // 1000000)
    self.pin.write_analog(duty)
    sleep(100)
    self.pin.write_analog(0)

def write_angle(self, degrees=None):
    if degrees is None:
        degrees = math.degrees(radians)
    degrees = degrees % 360
    total_range = self.max_us - self.min_us
    us = self.min_us + total_range * degrees // self.angle
    self.write_us(us)
```

```
Servo(pin8).write_angle(0)
display.show(Image.HAPPY)
```

```
while True:
    Servo(pin8).write_angle(0)
    sleep(1000)
    Servo(pin8).write_angle(45)
    sleep(1000)
    Servo(pin8).write_angle(90)
    sleep(1000)
    Servo(pin8).write_angle(135)
    sleep(1000)
    Servo(pin8).write_angle(180)
    sleep(1000)
```



Revision #10

Created 2025-11-06 08:40:41 CET by Juanjo Mejías

Updated 2026-10-30 14:13:14 CET by Juanjo Mejías