

# Práctica 5. Temporizadores y PWM

## (Duración: 2 sesiones)

### Introducción

En esta práctica, que está dividida en dos partes con trabajos previos independientes, adquirirá las nociones necesarias para gestionar el tiempo correctamente en un sistema electrónico. Aprenderá a generar señales de frecuencias y factores de servicio variables no solo para hacer parpadear diodos LED, sino también para controlar actuadores como, por ejemplo, un servomotor.

Durante la primera de las dos sesiones realizará ejercicios para manejar la temporización por *polling*. Esto significa que se pausará la ejecución del programa mientras se espera a que transcurra el tiempo. Durante la segunda sesión se registrarán *callbacks* para que se ejecuten automáticamente ciertas acciones cuando haya transcurrido un tiempo determinado mientras se realizan otras tareas de forma continua en el bucle de *scan*.

### Objetivos

Al finalizar la práctica, el alumno será capaz de:

- Resolver problemas sencillos en los que sea necesario generar señales a través de los pines GPIO con una frecuencia y factor de servicio determinados.
- Desarrollar sistemas electrónicos capaces de contar tiempo y generar señales de PWM tanto por *polling* como sin bloquear la ejecución del programa principal registrando *callbacks* para controlar actuadores.


### Evaluación

Aparte del trabajo previo, que el profesor revisará al principio de cada sesión de laboratorio, una semana después de la finalización de la práctica cada grupo entregará un **informe** que deberá contener las respuestas a las cuestiones que se plantean en el enunciado así como el código desarrollado siguiendo las directrices que se indican en la normativa de laboratorio.

Para obtener la puntuación máxima **no es necesario** completar los apartados opcionales. Se incluyen en el enunciado por si quiere profundizar más en la materia aunque, evidentemente, se valorará su esfuerzo si los hace y los incluye en el informe.

## Parte I. Ejercicios para afianzar conceptos

### Trabajo previo

Lea con atención el enunciado de la práctica y tome nota de las dudas que le surjan para preguntar al profesor. Intente también preparar el código de los apartados señalados con el símbolo . No hace falta que responda a las preguntas planteadas aún.

## 1. Entendiendo la temporización

Usando exclusivamente la función `sleep` de la librería `time` de Python y las funciones básicas para encender y apagar diodos de la clase `LED` de `gpiozero`, desarrolle un programa que haga parpadear el diodo conectado a `GPIO23` a una frecuencia de 2,5 Hz, es decir, que se ilumine durante 200 ms y esté apagado otros 200 ms. Compruebe con el **osciloscopio** que la señal es correcta.

## 2. Ejercitando la capacidad de creación

De nuevo solo con `sleep`, encienda el LED de `GPIO20` y haga que se vaya desplazando a la derecha cada segundo hasta llegar a `GPIO27`. Tras esperar un segundo, la secuencia volverá a empezar desde `GPIO20` al doble de velocidad, y así sucesivamente mientras que el tiempo de espera entre desplazamientos del LED sea superior a 1 ms. A partir de ese momento, la velocidad de cambio permanecerá constante. Llegados a este punto observará un efecto extraño en los diodos de la iMAT HAT. ¿Qué está pasando?

## 3. Aprendiendo a resolver problemas

Todavía sin utilizar funciones avanzadas de la librería `gpiozero` para que entienda cómo se pueden generar señales cuadradas con código de bajo nivel —solo puede usar los métodos `on()`, `off()` y `toggle()` de `LED` y la clase `Button` al completo—, se quiere conseguir que dos diodos LED parpadeen de forma simultánea con frecuencias diferentes y factores de servicio<sup>1</sup> distintos.

1. El diodo conectado a `GPIO23` parpadeará a una frecuencia de 1 Hz, mientras que el diodo de `GPIO24` lo hará a 0,2 Hz y se iluminará solo el 25 % del periodo, en lugar del 50 % como ha hecho hasta ahora. Una vez programado, verifique las formas de onda con el **osciloscopio**.
2. Manteniendo el comportamiento de `GPIO24` del ejercicio anterior, haga que `GPIO23` parpadee a 0,2 Hz con las pulsaciones impares de `GPIO7`, y que vuelva a 1 Hz con las pares. Al modificar la frecuencia, las señales de ambos LED comenzarán desde el principio (encendidas) para que los parpadeos permanezcan sincronizados. ¿Se ha fijado en que si pulsa y suelta muy rápido, el sistema no siempre responde? ¿Sería capaz de explicar la razón? ¿Se le ocurre alguna solución?

## Parte II. Aplicación

### Trabajo previo

Como trabajo previo para la segunda sesión, prepare el código de la Sección 4 utilizando cualquier funcionalidad de la librerías `gpiozero`, `threading` y `sched` que considere oportunas. Anote las dudas que le surjan para preguntar al profesor.

## 4. Ejercitando la capacidad de creación

El ojo humano no es capaz de apreciar parpadeos de frecuencias superiores a unos 60 Hz<sup>2</sup>. En el caso de un LED, a partir de este punto lo que se percibe es que se ilumina en mayor o menor medida en función de cuanto tiempo esté encendido en relación al periodo. Por ejemplo, si el LED se controla con una señal de 100 Hz y está encendido el 80 % del periodo, parecerá que tiene mayor intensidad luminosa que si está encendido solo el 20 %.

<sup>1</sup>Porcentaje del tiempo que una señal está a nivel alto en relación al periodo.

<sup>2</sup>Esto explica por qué las frecuencias de refresco de las pantallas de los móviles empiezan en ese valor...

1. Aprovechando este hecho, se quiere hacer un programa para que todos los LED de la iMAT HAT se enciendan gradualmente hasta alcanzar su luminosidad máxima. Cada segundo, los diodos se iluminarán un 20 % más y, cuando llegue al final, la secuencia volverá a empezar. En otras palabras, los LED empezarán apagados; transcurrido un segundo, se encenderán al 20 %; tras otro segundo, se pondrán al 40 %; y así sucesivamente. La frecuencia de la señal será de 100 Hz.
2. Modifique el programa anterior para que los diodos conectados de GPIO20 a GPIO23 sigan incrementando su luminosidad cada segundo y los de GPIO24 a GPIO27 lo hagan cada medio segundo.

## 5. Aprendiendo a resolver problemas

Las señales cuadradas con factor de servicio variable se denominan PWM (de *Pulse Width Modulation*) y se utilizan con mucha frecuencia para enviar órdenes a múltiples actuadores. Usando gpiozero, en este ejercicio se quiere controlar la posición de un servomotor para abrir y cerrar una barrera de acceso de vehículos. Para ello se necesita que el servo alterne entre las posiciones de 45° (cerrada) y 135° (abierta) cada vez que se active el pulsador de GPIO19 —si considera que los ángulos van de -90° a 90° como sucede en la clase AngularServo de gpiozero, la posición de cerrado se corresponderá con -45° y la de abierto con 45°—. La barrera también se cerrará automáticamente si lleva más de 10 segundos abierta. Compruebe la influencia de usar pigpio frente a RPi.GPIO como factoría de pines.

**Observación:** Recuerde que para utilizar pigpio es necesario arrancar el servicio (demonio) escribiendo `sudo pigpiod` en el Terminal.

Conecte el servomotor que le proporcionará su profesor al conector triple GPIO14 de la iMAT HAT disponible en el bloque etiquetado como DIG (SERVO). **El pin de control es el interior;** el de 5 V, el central; y GND, el que está en el exterior de la tarjeta. Los servomotores tienen un cable de tres hilos, de los cuales el **blanco** (a veces amarillo), se corresponde con la señal de control de PWM, el rojo es la alimentación de 5 V, y el negro, la referencia de tensiones.

**Observación:** Los servos de posición disponibles en el laboratorio esperan señales de PWM de 0,5 ms (0°) a 2,5 ms (180°) a nivel alto sobre el periodo habitual de 20 ms (50 Hz).

## 6. Para los más intrépidos (Opcional)

Como al laboratorio se viene a aprender, si le sobra tiempo y como buen ingeniero quiere experimentar y ensuciarse las manos, puede aprovechar para familiarizarse con el control de motores de corriente continua (DC). Asegúrese de que tiene conectado el chip L293D en el zócalo U2, un puente en H que permite que un motor gire en dos sentidos. Si olvidó colocarlo en la primera práctica, pídaselo a su profesor.

Las señales de control de los motores DC suelen ser de una frecuencia igual o superior a 20 kHz, el límite de audición humano, ya que de otro modo se percibiría un pitido muy molesto<sup>3</sup>. Desafortunadamente, el PWM que ha utilizado hasta ahora funciona por *software* y no es capaz de generar señales precisas de una frecuencia tan elevada. Por ese motivo, hay que utilizar el generador de PWM por *hardware* de la Raspberry Pi. Aunque dispone de las instrucciones para configurar el módulo de PWM en los apuntes de teoría, se reproducen los pasos a continuación:

<sup>3</sup>Los perros tendrían alguna queja más con la elección de este umbral...

1. Edite el archivo /boot/config.txt y añada **dtoverlay=pwm-2chan,pin2=13,func2=4** en una línea aparte al principio del archivo.
2. Reinicie la Raspberry Pi y después escriba `lsmod | grep pwm` en el terminal para verificar que todo se ha configurado correctamente. Debería aparecer `pwm_bcm2835` seguido de unos números.
3. Instale la librería `rpi-hardware-pwm` con `pip`.

Dado que aún no ha aprendido cómo controlar un motor en clase, el Código 1 presenta un ejemplo que hace girar el MOTOR 1 a la mitad de su velocidad máxima. Cuanto mayor sea el factor de servicio (*duty cycle*), mayor será la velocidad del motor. Con el 100 % girará a su velocidad máxima, mientras que si se establece un 0 %, se detendrá. Para seleccionar el sentido de giro, se utilizan los pines GPIO4 y GPIO5. Solo uno de ellos debe estar a nivel alto al mismo tiempo.

```
import RPi.GPIO as GPIO
from rpi_hardware_pwm import HardwarePWM

def main():
    pwm = HardwarePWM(pwm_channel=0, hz=20_000 * 0.933)

    # Pines de sentido de giro
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(4, GPIO.OUT)
    GPIO.setup(5, GPIO.OUT)

    # Seleccionar un sentido de giro. Intercambiar HIGH y LOW para escoger el contrario.
    GPIO.output(4, GPIO.HIGH)
    GPIO.output(5, GPIO.LOW)

    pwm.start(initial_duty_cycle=50)

    try:
        while True:
            # En el bucle de scan se pueden cambiar el factor de servicio y la frecuencia con:
            # pwm.change_duty_cycle(60.0)
            # pwm.change_frequency(25_000)
            pass
    except KeyboardInterrupt:
        pwm.stop()
        GPIO.cleanup()

if __name__ == "__main__":
    main()
```

**Código 1.** Ejemplo de uso del módulo de PWM por *hardware* para controlar el MOTOR 1. Fíjese en que la frecuencia deseada se ha multiplicado por el **número mágico 0,933** para corregir un error sistemático de la librería.

Pida un motor de corriente continua y conéctelo al enchufe para bloques terminales etiquetado como MOTOR 1. No importa qué cable conecte a 1Y y cuál a 2Y. A continuación, proporcione 9V a POWER (EXT) desde la fuente del laboratorio para alimentar el motor y pruebe el código de ejemplo. Familiarícese con el factor de servicio y el sentido de giro. Finalmente, modifique el programa para controlar simultáneamente el MOTOR 2. Tendrá que localizar los números de pines en el esquema de la iMAT HAT. Capture con el **osciloscopio** alguna de las señales generadas para asegurarse de que son correctas.