Práctica 2

Introducción a la electrónica GPIO con el LED RGB

Grado en Ingeniería en Robótica Software
GSyC, Universidad Rey Juan Carlos

1. Introducción

Una de las primeras prácticas fue sobre cómo manejar un LED (normal) mediante la técnica de PWM o modulación del ancho de pulso. Se puede considerar esta como el *Hola mundo* de la electrónica. Es fácil de instalar, se recuerdan conceptos de electricidad que ya tenías aprendido de antes y, además, el resultado es vistoso.

En esta ocasión, vamos a usar un LED más avanzado y, por tanto, más complejo de instalar y programar: el LED RGB. Evidentemente, las siglas RGB ya te resultarán familiares; son las siglas de *Red*, *Green*, *Blue*. Este *macro*LED contiene en una sola pieza esos tres LEDs[1] de colores. Además, y como ya sabrás, estos son los colores primarios de la luz, lo que significa que —modulándolos convenientemente— podrás conseguir cualquier color de luz.

Para manejar apropiadamente este LED, vamos a aprender en detalle cómo funcionan los puertos o pines GPIO, ya que son clave para conectar los sensores y actuadores que veremos durante el curso.

2. Los pines GPIO

Como ya hemos visto en la Práctica 1, además del interfaz para conexión de dispositivos estándar (HDMI, Ethernet, USB, Jack), la Raspberry Pi ofrece una serie de pines denominados GPIO (*General Purpose Input Output*). Estos pines se emplean comúnmente para conectar dispositivos electrónicos, como sensores y actuadores.

Ya hemos aprendido que estos pines son digitales (si queremos usarlos como analógicos hemos de acoplar un DAC, *Digital to Analog Converter*), y que pueden ser configurados de entrada o de salida. Al configurarlos de salida, como hicimos en la Práctica 1, pueden ser activados o desactivados por código; esto es, pueden ponerse a 1 (*HIGH*, o nivel alto de voltaje) o a 0 (*LOW*, o nivel bajo de voltaje). Por contra, si los configuramos de entrada, pueden leer datos binarios; es decir, unos (hay señal de voltaje) o ceros (no hay voltaje, o este es cercano a cero).

Es muy importante tener en cuenta que los puertos GPIO funcionan sobre niveles lógicos de $3{,}3V$: $0V$ significa el 0 lógico o *LOW* y $3{,}3V$ es la tensión equivalente al 1 lógico o *HIGH*. Por ello, hay que tener cuidado de no conectar nunca los pines de alimentación de $3{,}3V$ y $5V$ directamente entre sí, o un pin de $5V$ directamente a un pin GPIO.

[1] Realmente, deberíamos hablar de *ledes* (igual que el plural de *red* es *redes*), aunque coloquialmente está más extendido el uso —incorrecto— de *LEDs*.

Respecto al uso de un pin GPIO como pin de entrada, esta puede ser configurada de dos formas: como evento/interrupción para que, al producirse esta, genere alguna acción sobre el sistema; o, lo más común y el modo que emplearemos nosotros, que la acción sea realizada cuando uno o más pines sean activados por algún sensor.

Por último, y según las especificaciones del bus de puertos GPIO de la Raspberry Pi 3B+, los pines de $3{,}3V$ son capaces de proporcionar un máximo de $50mA$ al mismo tiempo. Aunque no existe un limitador, nos sirve para hacernos una idea de qué números estamos manejando. Dicho de otro modo, no ocurrirá nada porque nos pasemos ligeramente de ese valor en un momento dado. Por ejemplo, si pensamos en el caso de mayor demanda de intensidad, que sería cuando están encendidos los tres LEDs de color al mismo tiempo, la placa estará perfectamente protegida si nos mantenemos cercanos a esos $50mA$. Podemos establecer la intensidad máxima de cada LED del siguiente modo: $50mA/3 \approx 17mA \implies I_{max_R} = I_{max_G} = I_{max_B} = 17mA$.

3. El LED RGB

Antes de proceder con la instalación de este LED tan particular hemos de echar números. Veremos que cada color requiere de un valor diferente de tensión e intensidad: el rojo es el que menos tensión requiere, mientras que el azul es el que más tensión necesita.

Figura 1: LED RGB

Por otro lado, como vemos, este LED tiene cuatro pines en lugar de dos, como puedes comprobar en la Figura 1. Esto es debido a que, como ya hemos mencionado previamente, un LED RGB incluye en realidad un circuito de tres LEDs diferentes. Comúnmente, el pin más largo (n.º 2) es el cátodo, mientras que los otros tres corresponden a los tres colores: rojo (1), verde (3) y azul (4).

Para salir de dudas y, como siempre, recuerda consultar la hoja de especificaciones de tu modelo concreto. Se adjunta la hoja de especificaciones de un LED RGB común. No obstante, el LED RGB que se incluye en el kit tiene la polaridad cambiada respecto a la norma común. Hay que tener esto en cuenta para la conexión del circuito (Sección 3.1).

3.1. Esquema de conexión

Para conectar el LED RGB y ajustarnos a las especificaciones de los pines GPIO de la placa, vamos a necesitar resistencias. El uso de estas es fundamental para conectar estos y otros dispositivos delicados a la placa, para proteger no solo a estos, sino —y lo que es más importante— a la placa. Ten en cuenta que, típicamente, estos LEDs soportan hasta $30mA$ de intensidad, pero nosotros, por seguridad, los vamos a mantener por debajo de $17mA$, como habíamos calculado.

Las resistencias que tendremos que usar dependerán de las especificaciones concretas del LED RGB. En cualquier caso, obviamente necesitarás tres resistencias; una por cada pin de color. Para saber leer el código de colores de una resistencia, recuerda tener en cuenta lo siguiente:

1. Nosotros usaremos resistencias de cuatro bandas, aunque también las hay de cinco.
2. Las resistencias son simétricas; para leerla correctamente fíjate en las dos bandas situadas en los extremos, verás que hay una que está más pegada al extremo que la otra; pues bien, esa debe quedar a la izquierda.
3. Por si no estás totalmente seguro con el paso anterior, otra forma para asegurarnos de que estamos leyendo la resistencia correctamente es observar que existe un extremo en el que existe mayor separación entre la última banda y la penúltima; pues bien, este extremo ha de quedar a la derecha.
4. Ya teniendo bien situada la resistencia, observamos la primera y segunda banda. Estas nos dan el valor. Cada una va de 0 a 9, obteniendo por tanto un valor total comprendido entre 00 y 99.
5. La tercera banda es el multiplicador; esto es, por cuánto hemos de multiplicar el valor obtenido en el anterior paso. Así, por ejemplo, si el valor obtenido de las bandas 1 y 2 es 10 y el multiplicador es 100, el valor de la resistencia será de 1000Ω.
6. La cuarta banda es la tolerancia; esto es, el margen de error entre el valor dado según el código de colores y el valor real de la resistencia. Un margen de error o tolerancia habitual es el 5 %, representado por el color dorado.

También existen calculadoras *on-line*[2] que nos vierten el valor de una resistencia en función de sus bandas de colores.

O también puedes hacer uso de alguna de las calculadoras *on-line* de resistencia específicamente diseñadas para cuando usemos LEDs[3][4], que seguro te resultarán de mucha ayuda. Estas emplean la famosa *Ley de Ohm*.

Para usar una de estas herramientas, implemente introduce la tensión de la fuente, $V_S = 3{,}3V$, el voltaje que queremos que llegue al LED, e.g. $V_{LED} = 1{,}5V$ (lógicamente, $V_{LED} <= V_S$), y la intensidad (I_{LED}) que deseemos aplicar a este (recuerda, $I_{LED} < 17mA$). Con estos datos, obtendremos la resistencia (R) que hemos de colocar, así como la potencia que tendrá el LED (P). Otra forma de trabajar, y siguiendo con la Ley de Ohm, según las necesidades, es *despejar* la V_{LED}; esto es, indicar la resistencia que vamos a aplicar, y en función de esta nos indicará qué tensión V_{LED} le llegará al LED.

Sea como sea, junto con las tres resistencias que finalmente insertemos y el cableado, resulta de especial importancia el uso de la *protoboard*. Con ayuda de esta, el esquema de conexión de un LED RGB común quedaría como vemos en la Figura 2.

Nótese que el esquema mostrado en la Figura 2 es el esquema de conexión general de un LED RGB común, en el que, como vemos, necesitamos conectar el cátodo común del LED RGB a un pin de toma de tierra (*ground*), y cada uno de los tres ánodos a un pin programable diferente, usando en total los pines: 9 (GND), 11, 13 y 15.

No obstante, y como ya se ha mencionado previamente, el LED RGB que se incluye en el kit tiene la polaridad cambiada. Es por ello que, siguiendo el esquema de conexión de la Figura 2, no funcionará. Simplemente, habría que conectar los cables al revés de lo mostrado en la Figura 2; esto es, considerando la patilla larga como ánodo. Considérese esto una excepción, pues la mayoría de estos LEDs RGB suelen estar configurados como se ha explicado de forma general en los párrafos anteriores.

[2] `https://www.digikey.es/es/resources/conversion-calculators/conversion-calculator-resistor-color-code`
[3] `https://ohmslawcalculator.com/led-resistor-calculator`
[4] `https://ledcalculator.net`

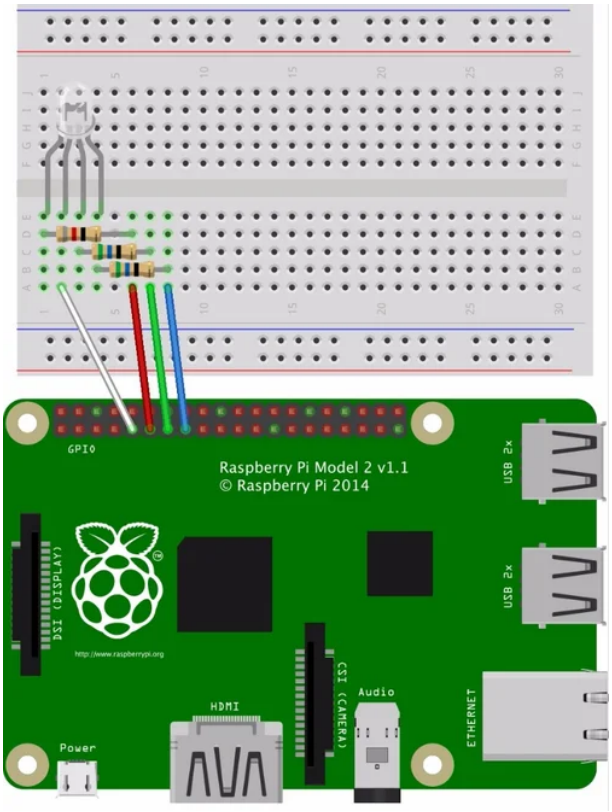

Figura 2: Esquema de conexión de un LED RGB común con protoboard

4. Programación del LED RGB

Una vez tenemos lo anterior, ya la programación es sencilla; basta con crearnos tres variables para guardar los pines correspondientes a cada uno de los tres colores y operar con ellos a nuestro antojo, como ya hicimos para el LED básico.

En el código que se facilita (`ledRGB.py`) hemos usado el modo `GPIO.BOARD`, por lo que hemos creado las siguientes tres variables:

```
rojoPin = 11
verdePin = 15
azulPin = 13
```

Se facilitan dos funciones para encender y apagar un pin determinado, así como una función concreta para encender el color rojo. Faltan, entre otras cosas, las instrucciones que ya hemos visto de *limpieza* de puertos GPIO (`GPIO.cleanup()`) de antes de finalizar la ejecución del programa.

5. Ejercicios

Ejercicio 1. Uso de colores

1. En primer lugar, modifica convenientemente el programa para que pueda gestionar el encendido y apagado de los tres colores primarios.

2. En segundo lugar, añade la funcionalidad necesaria para gestionar el encendido y apagado de, al menos, otros cinco colores, p. ej.: magenta, amarillo, cyan, blanco, etc.

Ejercicio 2. Interfaz de usuario

Partiendo de la funcionalidad implementada para el Ejercicio 1, añade nueva funcionalidad para que el programa pida por teclado al usuario las órdenes de qué quiere hacer. Por ejemplo:

```
encender rojo
apagar rojo
encender blanco
encender verde
salir
```