Micropython

Programación fácil y para todos de microcontroladores.

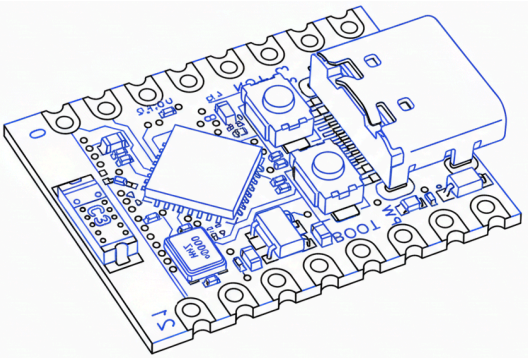


Tabla de contenido

Manual de apoyo taller Micropython

Muchas gracias por ofrecer tu ayuda para el taller *"Programación fácil y para todos de microcontroladores"*. Debido a la naturaleza de este taller, pueden darse múltiples eventualidades, las cuales requieren de la colaboración de voluntarios como es tu caso.

En el taller pueden haber dos tipos de personas que requieran ayuda durante las prácticas:

- Aquellas que están realizando prácticas usando el simulador en línea.
- Las que están realizando circuitos usando los componentes físicos provistos.

Como es previsible, estos segundos serán los que más ayuda puedan necesitar, y más dificultades puedan tener, por lo que este manual se centra principalmente en ellos.

Consejos para prácticas con microcontroladores

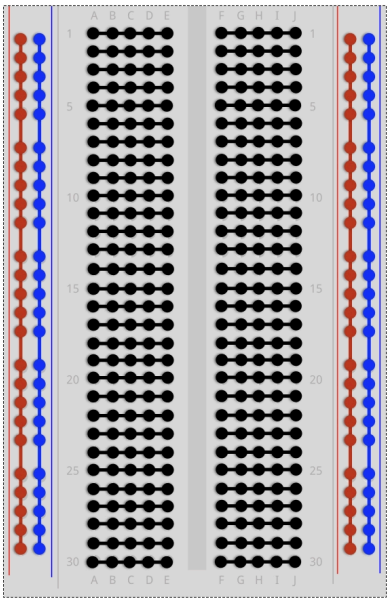
Es esencial que se sigan las siguientes recomendaciones, no sólo para completar las prácticas, sino para evitar que puedan producirse daños en los microcontroladores, los componentes, o incluso el equipo del asistente.

- Ante todo, evitar los cortocircuitos. Esto significa, conectar dos pines sin que haya un componente que actúe de resistencia entre ellos.
- Todos los componentes usados, salvo las resistencias, tienen polaridad o posición. Por tanto, se debe comprobar qué es cada pin del componente.
- Se debe ante todo tener cuidado con el pin de 5V. Éste está conectado directamente al USB del ordenador, y tiene un voltaje superior al admitido por la placa (3.3V). Su uso indebido puede provocar daños. No se requiere su uso hasta las últimas prácticas.
- Los pines de 5V y tierra están juntos, en las filas 1 y 2. Evitar confundirlos, y sobre todo unirlos, para evitar daños en el ordenador.
- Hay números en la protoboard indicando la posición del pin en filas, yendo éstas del 1 al 30. Fijarse en éstas y compararlas con el simulador, antes de conectar cables.
- No debe moverse el microcontrolador de la protoboard. Su posición es la ideal para las prácticas, y sólo necesitan usarse los pines correspondientes a las columnas *f, g, h, i, j* de la protoboard.
- No poner los pines de un mismo componente en la misma fila de la protoboard. Por ejemplo, ambas patillas de un LED en las columnas *g* y *h.* Al estar toda la fila conectada entre sí, la electricidad no pasará por el LED.
- Los LED requieren el uso de una resistencia, para evitar quemarlos. No es necesario su uso con el botón, bien utilizada la resistencia pull-up interna del microcontrolador.

Uso de la protoboard

La protoboard, es una placa de pruebas utilizada para probar circuitos de una forma sencilla gracias a sus filas y columnas conectadas, lo cual reduce la cantidad de cables necesarios. No obstante, es importante entender bien cómo están conectadas entre sí.

- El cuerpo interior, con columnas *a-j* y filas *1-30*, es donde pondremos los componentes.
- En las columnas de los lados, identificadas como positivo (+) y negativo (-), es donde pondremos el voltaje y la tierra respectivamente.
- Las filas del cuerpo interior donde ponemos los componentes, están unidas entre sí.
- En cambio, en la parte exterior, es toda la columna la que está unida.
- Ambas mitades de la protoboard, *a-e* y *f-j*, están separadas entre sí y no tienen continuidad.
- Para usar la tierra del ESP32 en la columna de tierra (línea azul), conecta de la fila *2*, columnas *g-j* un cable a la columna exterior.
- Por seguridad, no conectar los 5V hasta más adelante en las prácticas.



Código de colores y pines

Para facilitar el desarrollo de las las prácticas, emplearemos un mismo código común tanto en las simulaciones como en las prácticas reales. Recomendamos usar este código:

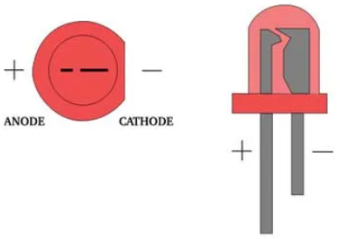
- **5 voltios:** cable **rojo** *(corresponde a fila 1)*. Se recomienda conectar sólo en los últimos ejercicios, ya que puede ser peligroso. Llegado el momento, conectar a la columna exterior positiva (línea roja).
- **Tierra:** cable **negro** *(corresponde fila 2)*. Se requiere en todos los ejercicios, por lo que se recomienda conectarlo a la columna exterior negativa (línea azul).
- **Pin 1**: cable **amarillo** *(corresponde a fila 7)*. Se usará en todas las prácticas.
- **Pin 2:** cable **verde** *(corresponde a fila 6)*. Se emplea en las prácticas 4, 5 y 6.
- **Pin 3:** cable **azul** *(corresponde a fila 5)*. Sólo se usa en la práctica 4 (led RGB).

Los kits provistos tienen más cables rojos y negros con este código en mente, al requerirse un mayor número por su uso.

Componentes electrónicos

A continuación se listan los diferentes componentes que se emplearán durante las prácticas, indicando su correcto uso.

LED

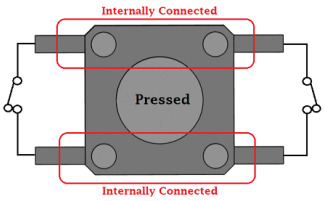


Requiere usar una resistencia.
La patilla más larga es la positiva.
Tiene un lado plano alrededor de la base que permite identificar la patilla negativa.

Resistencia

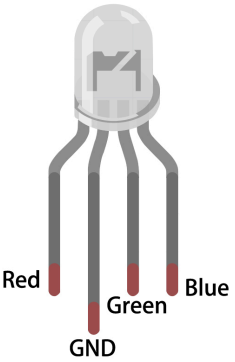
No tiene polaridad. Se recomienda su uso con todos los LED de las prácticas. Se incluyen 3 valores: 220Ω, 330Ω y 10KΩ.
Los valores se indican mediante franjas de colores. Puede usarse una calculadora online para ver el valor.

Interruptor



Requiere una resistencia, ya sea física o pull-up del microcontrolador, en modo input.
Tiene 4 pines, estando unidos a pares. En el momento de pulsarse, los 4 se unen.
Los dos lados más alejados entre sí, son los que están unidos internamente. Los más juntos, se unen al pulsar el botón.

LED RGB



Requiere una resistencia, debiéndose usar una por cada pin de color.
No confundir el LED RGB del APA106. El LED RGB tiene una cabeza más grande. Sólo se proporciona uno, a diferencia del APA106.
El pin más largo es tierra, el que se encuentra solo a su lado corresponde al rojo, y los dos a su otro lado verde y azul, estando éste más lejos.
Se recomienda usar los pines 1 para rojo, 2 verde y 3 azul. El cable amarillo corresponde para rojo. El cable rojo será 5V más adelante.

LED APA106

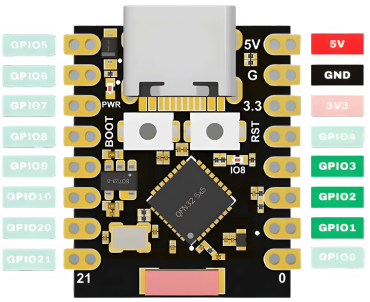
Requiere una única resistencia en su pin de datos.
No confundir con LED RGB. Tiene una cabeza más pequeña, pines de diferentes tamaños y se proporcionan 3.
El pin más corto es tierra, al igual que el led RGB, el más corto la entrada de datos. El que se encuentra al lado de datos, es el pin de 5V. El restante y que está más lejos de la entrada de datos, es el pin de salida de datos.
Se recomienda usar los pines 1 para entrada de datos (cable amarillo). Colocándose los 3 intercalados,con entrada y salida en la misma columna, no son necesarios cables de datos adicionales.

Fotoresistencia (LDR)

Requiere una resistencia de 10KΩ para que la variación de voltaje sea lineal en el rango medio, ya que el LDR varía su resistencia entre 100Ω y 1MΩ, variando ésta demasiado bruscamente por los cambios de luz.

Las patillas no están polarizadas, ya que es una resistencia. Dependiendo de la luz ambiental, la resistencia puede ser demasiado grande o insuficiente, pero debería poder realizarse el ejercicio variando el punto a partir del cual se considera que no hay luz.

ESP32 C3 SuperMini



Se alimenta de 5V a través del conector USB-C, pero opera a 3.3V. Tiene 16 pines en total, pero sólo usaremos 5, indicándose su color y posición en la protoboard en la sección *"códigos de colores y pines"*. Éstos pueden verse en el diagrama del margen.
Para evitar doblar las patillas y facilitar las prácticas, el microcontrolador no debe separarse de la protoboard.

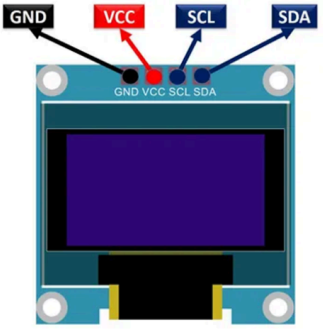
La práctica con la fotoresistencia es un ejercicio adicional, no estando disponible en las diapositivas, ni contando con simulación. Aquellos que hayan terminado los ejercicios previos, pueden encontrarla en el repositorio de Github, como *"extra1"*.

Componentes adicionales

Los siguientes componentes se proporcionan por separado, al haberse completado las prácticas anteriores. Sólo se llegan a proporcionar 5 de cada, por limitaciones en su número. Preferentemente, se entregará primero el kit de pantalla OLED y sensor de temperatura y humedad.

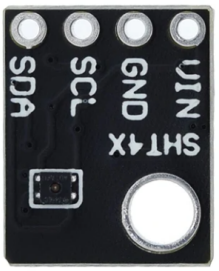
Los pines del mismo tipo (tierra, 5V, SCL y SDA) de distintos dispositivos I2C pueden unirse entre sí, para permitir que éstos se comuniquen con el microcontrolador.

OLED SSD1306



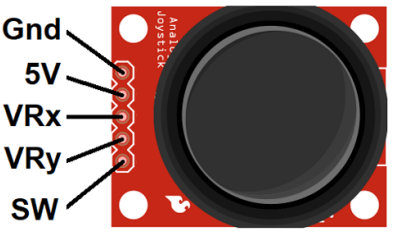
No es necesario utilizar una resistencia con este componente.
La comunicación es vía I2C, con dirección 0x44.
La pantalla tiene una resolución de 128 x 64 píxeles, a un solo color.
Se recomienda pin 1 (cable amarillo) para SCL y pin 2 (cable verde) para SDA.
Los pines se encuentran indicados en el propio dispositivo, (siendo VCC los 5V) mirando hacia arriba.

Sensor de temperatura y humedad SHT41



No es necesario utilizar una resistencia con este componente.
La comunicación es vía I2C, con dirección 0x3C.
Ofrece una gran fiabilidad: ±0,2 °C y ±1,8%RH.
Se recomienda pin 1 (cable amarillo) para SCL y pin 2 (cable verde) para SDA.
Los pines se encuentran indicados en el propio dispositivo (siendo VIN los 5V), mirando hacia arriba.

Joystick



No requiere resistencia en sus pines 5V, VRx ni VRy, pero sí SW (poner resistencia o usar la resistencia internal pull-up en modo entrada).
Los pines VRx y VRy dan respectivamente las posiciones del eje X e Y del joystick. SW es igual que un interruptor, accionándose cuando se pulsa el joystick.
El joystick funciona como un potenciómetro, dando valores según se mueve la palanca.
Los pines se encuentran indicados en el propio dispositivo, mirando hacia arriba.

Altavoz pasivo (buzzer)

Opera entre 3.3V y 5V. Se emplea el modo PWM (modo analógico) de Micropython para variar la frecuencia, y con ello conseguir diferentes tonos o notas, pudiendo generar melodías. No requiere una resistencia.
Está polarizado, correspondiendo el cable rojo al positivo (que irá conectado al a un pin GPIO) y el negro a tierra. Pueden conectarse directamente cables a sus extremos, metiéndolos dentro del capuchón negro.

Las prácticas del joystick y del altavoz pasivo son complementarias, y no se encuentran en las diapositivas. Aquellos que no cuenten con un kit para la práctica I2C, pueden recibir cualquiera de ambos kits para los ejercicios extra 2 y extra 3, estando éstos documentados en la carpeta demos del repositorio de Github. El ejercicio extra 3 debe realizarse en parejas.

Una vez que hayan terminado la práctica del ejercicio 6 (pantalla OLED y sensor de temperatura y humedad), tomarlos para que otras personas puedan realizar dicha práctica.

Ordenador de los asistentes

Todos los ordenadores, indistintamente de si usarán el simulador o un kit para prácticas reales, deberán tener conexión a Internet y la presentación abierta, la cual está disponible en: https://github.com/Nekmo/micropython-workshop

En el caso de los ordenadores usados para prácticas reales, deberán tener un conector USB-A (el rectangular grande) ya sea en el propio equipo o a través de un adaptador, o un conector USB-C y usar su propio cable. En lo que se requiere al software, sólo se soporta como navegadores Chrome (y la mayoría de sus derivados), quedando excluidos Firefox y Safari. Pueden ser necesarios otros cambios según el sistema operativo.

GNU/Linux

Lo más sencillo es usar una regla de udev. Ten en cuenta que esta regla puede ser insegura, por lo que es mejor modificarla después:

```
echo 'KERNEL=="ttyACM[0-9]*",MODE="0666"'| sudo tee /etc/udev/rules.d/50-myusb.rules
sudo udevadm control --reload-rules
sudo udevadm trigger
```

También puede intentarse añadir el usuario a los grupos dialout o/y uucp:

```
sudo usermod -aG dialout <user>
sudo usermod -aG uucp <user>
newgrp
```

Puede ser necesario cerrar sesión o reiniciar el equipo, si no funcionan los comandos que recargan los grupos y las reglas de udev.

Windows

En las pruebas realizadas con Windows 10, tanto en Google Chrome como en Microsoft Edge, no ha sido necesario instalar drivers adicionales. Puede ser necesario no obstante, instalar los drivers CP210x o CH340. Reiniciar el equipo tras ello.

Mac OS

La situación en Mac OS es la misma que en Windows, pero en este caso Safari no tiene soporte. Debe usarse Google Chrome. Si fuese necesario instalar los drivers CP210x o CH340 específicos de este sistema. Reiniciar el equipo tras ello.

Kit de pruebas y reseteo

Ante la dificultad para conseguir resultados favorables con un kit de prácticas, compuesto por microcontrolador, protoboard y cable, es necesario descartar lo antes posible que el problema se encuentra en estos, y no en el ordenador del asistente. Para ello, se dispone de un kit de pruebas y reseteo, el cual es capaz no sólo de verificar su correcto funcionamiento, sino además restaurarlo en caso de ser necesario.

El kit se compone de una Raspberry PI, una batería externa, un circuito y 8 cables unidos en un único conector. No se debe desconectar el cable del circuito, para evitar doblar pines.

Al conectarse la corriente de la Raspberry PI, se escuchará una melodía, indicando que se encuentra preparada para su uso. Tras ello, puede enchufarse el kit de prácticas, dándose los siguientes pasos.

1. Detección del dispositivo USB. Se produce un primer pitido corto agudo.
2. Comprobación de ficheros para las prácticas, y copiado si faltasen. Tanto si falla como si se realiza correctamente, continúa el programa.
3. Ejecución de código de demo, para su verificación junto con el circuito del kit de pruebas.
    a. Si se ejecuta correctamente, sonarán dos pitidos cortos agudos. Finaliza el programa.
    b. Si falla, sonará un pitido grave prolongado.
4. Sólo en caso de error, se intentará resetear el microcontrolador y volver a escribir una imagen de Micropython en el mismo.
    a. Si se ejecuta correctamente, sonarán tres pitidos cortos agudos. Desconecta el microcontrolador, y vuelve a conectarlo para verificar su correcto funcionamiento y que copie los ficheros faltantes.
    b. Si falla, sonará de nuevo un pitido grave prolongado. Desconecta el microcontrolador, y vuelve a conectarlo, sólo esta vez manteniendo el botón de boot (el izquierdo) del microcontrolador a la vez que lo conectas.

El circuito para pruebas, permite verificar el correcto funcionamiento de todos los pines y la recepción de corriente. Puede conectarse antes o después de empezar la prueba. Conectarlo desde la fila *1* a la *8*, en las columnas *g-j* de la protoboard del asistente. Los cables rojo, negro y rojo deben quedar próximos al cable USB (una pegatina indica el sentido). No forzar si entra muy duro en la protoboard, con que entre un poco es suficiente.

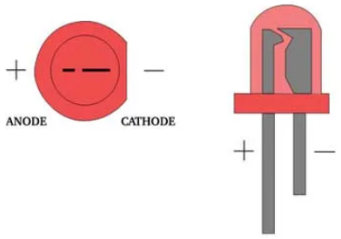
- **5V (led 1):** se recibe corriente a través del ordenador a través del cable USB.
- **3.3V (led 2):** la transformación de corriente funciona correctamente.
- **GPIO 4 (led 3):** funciona este pin de datos. Debe parpadear con fuerza.
- **GPIO 3 (led 4):** igual que el anterior.
- **GPIO 2 (led 5):** igual que el anterior.
- **GPIO 1 (led 6):** igual que el anterior.
- **GPIO 0 (led 7):** igual que el anterior.

Prácticas del taller

A continuación se indica la resolución y consejos para las diferentes prácticas realizadas durante el taller, tanto en su modalidad usando el simulador como creando los circuitos físicos.

Ejercicio 1

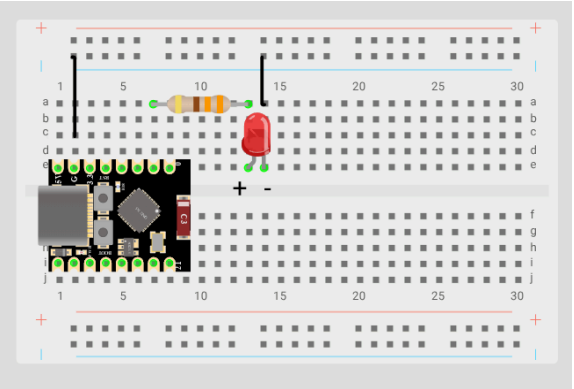
Se guiará a los asistentes durante la realización de este ejercicio, por lo que no debería ser necesaria ayuda.



La mayor dificultad durante este ejercicio, siendo el primero, radica en la comprensión de cómo funciona la protoboard, la posición del LED (patilla larga positivo, corta negativo) y la incorrecta colocación de los cables, sobre todo en la protoboard (fila 2 tierra, fila 7 GPIO1).

En el código fuente, hay que entender que Pin(1) corresponde a la fila 7 de la protoboard, siendo la correspondiente al GPIO1. Ver diagrama del ESP32 C3 en caso de dudas.

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



Ejercicio 2

Para este ejercicio, no es necesario hacer ninguna modificación en cómo están instalados los componentes respecto al ejercicio anterior.

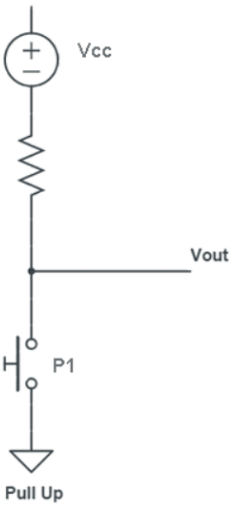
De este ejercicio, PWM hace que la instancia de Pin pueda manejarse de forma analógica, usándose el método duty() con un valor entre 0 y 1023.

El primer for aumenta la intensidad entre ambos valores, y el segundo la disminuye.

```
1   from machine import Pin, PWM
2   from time import sleep
3
4   led = Pin(1)
5   pwm = PWM(led)
6
7
8   while True:
9       for i in range(0, 1024, 10):
10          pwm.duty(i)
11          sleep(0.01)
12      for i in range(1023, -1, -10):
13          pwm.duty(i)
14          sleep(0.01)
```

Ejercicio 3

Este ejercicio, tiene como mayores dificultades, entender cómo se colocará el botón, y cómo funciona la resistencia Pull-up interna.

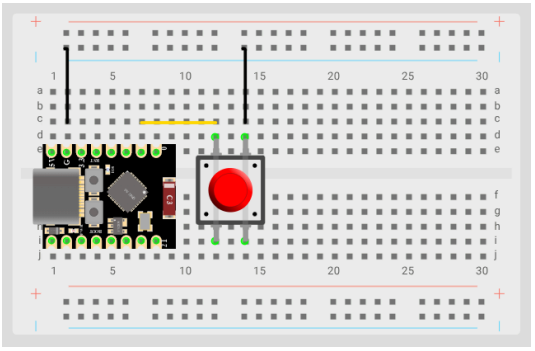


El botón debe colocarse en la mitad, entre ambos lados de la protoboard. Sólo hay una posición posible, sin doblar las patillas. Éstas, como ganchos, deben agarrar la protoboard. Las patillas más lejanas entre sí, deben quedar cada una en una mitad de la protoboard.

La resistencia interna pull-up, añade una resistencia entre la lectura interna del pin (Vout), y la salida del pin. Si el circuito de salida del pin está abierto, la corriente sólo sale hacia Vout, teniendo como valor 1. En caso de cerrarse (el botón se pulsa), la corriente irá directamente hacia tierra, cambiando el valor de lectura de Vout a 0.

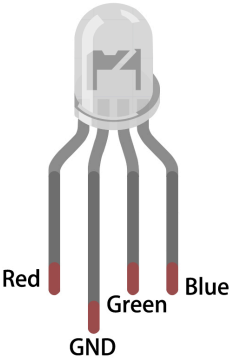
Es por ello, que debe configurarse la el modo Pull-up en el código. Es importante igualmente poner el modo entrada (IN) en vez de salida. En el programa, se comprobará el estado del pin cada 100ms.

```
1   from machine import Pin
2   from time import sleep
3
4   button = Pin(1, Pin.IN, Pin.PULL_UP)
5
6
7   while True:
8       state = not button.value()
9       print(int(state))
10      sleep(0.1)
```



Ejercicio 4

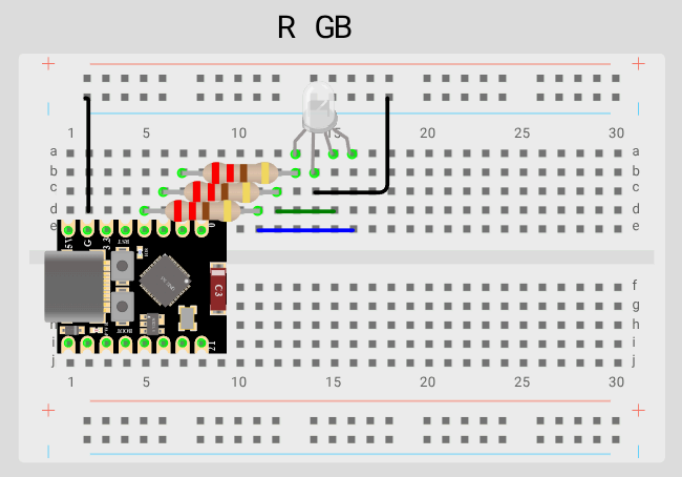
Este ejercicio es muy similar a los dos primeros. Es como si tuviésemos 3 diodos LED, pero en uno solo. La mayor dificultad radica en el número de cables, ya que es fácil equivocarse. Es por ello, que es muy importante seguir la siguiente regla:



- Pin 1: cable **amarillo**, correspondiente al color **rojo** del LED (el rojo se usará para 5V más adelante). Puede llegar a hacerse sin cable, sólo con la resistencia.
- Pin 2: cable **verde**, correspondiente a su mismo color.
- Pin 3: cable **azul**, siendo su mismo color del LED.

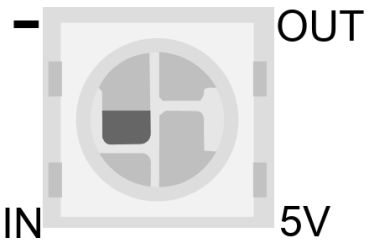
```
from machine import Pin, PWM
from time import sleep
import urandom


r = PWM(Pin(1), freq=1000)
g = PWM(Pin(2), freq=1000)
b = PWM(Pin(3), freq=1000)


def set_color(red, green, blue):
    r.duty(red)
    g.duty(green)
    b.duty(blue)


while True:
    red = urandom.getrandbits(10)
    green = urandom.getrandbits(10)
    blue = urandom.getrandbits(10)
    set_color(red, green, blue)
    sleep(1)
```

Debe usarse una resistencia por cada color. Con cada iteración del bucle, obtenemos un valor aleatorio con getrandbits para cada valor de RGB, dando un color diferente. En caso de duda, ver el ejercicio 2, en que se utiliza también PWM.



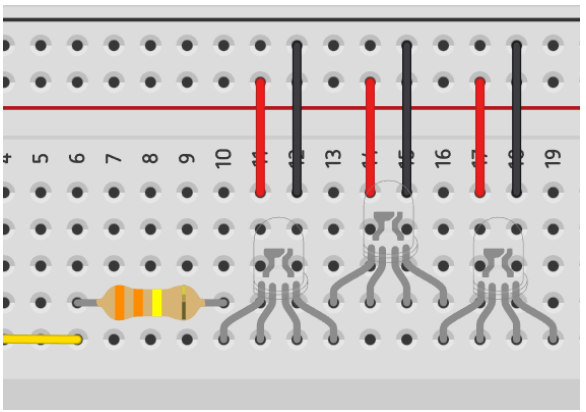
Ejercicio 5

Este ejercicio es el único de los ejercicios con diapositivas, que no cuenta con protoboard en la simulación, y el componente no es igual al utilizado en la práctica con los componentes físicos. Siendo 3 LED, y requiriendo cada uno de ellos tierra y 5V, también puede suponer un reto.



En el simulador, se recomienda pulsar en cada patilla, para saber a qué corresponde cada una de ellas. Sin girar el componente (lo cual no se recomienda, pues está bien orientado), cada pin corresponde a los de la imagen del margen.

Este es además el primer ejercicio que requiere de 5V. Es importante no confundirlo con tierra, y colocarlo en la línea roja de la protoboard. Al colocar los LED, se recomienda ponerlos uno tras otro, intercalando el pin de entrada con el de salida en una misma fila.



Es necesaria una resistencia, pero únicamente en el pin de datos. El pin de OUT del último LED se queda suelto, no conectado a nada.

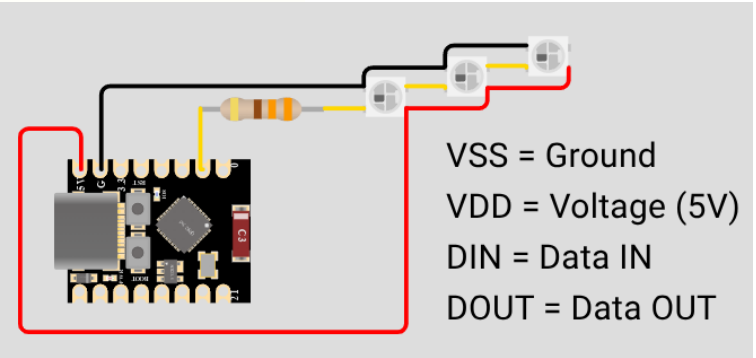
El pin de entrada de datos del primer LED irá conectado al GPIO 1 a través de la resistencia, y el pin de salida irá conectado al de entrada del siguiente LED.

El LED es parecido al RGB, pero puede verse que su cabeza es más pequeña. La patilla larga es igualmente la correspondiente a tierra, pero en este caso se alimenta de 5V,

En el programa, se usará el módulo incluido APA106, el cual requiere indica el pin LED a utilizar, y el número de LED que hay. Tenemos un listado de colores entre los que alternaremos (rojo, verde y azul) y offset indica a qué posición corresponde el inicio actual. Por ejemplo, si es cero, los colores serán rojo verde y azul, si es 1 serán verde, azul, rojo, etc.

El bucle for se utiliza para dar un color a cada LED, según su posición, y el que corresponde según el offset actual. La instancia leds puede accederse como si fuese una lista. Es necesario ejecutar write para que aplique los cambios.

```
1   from machine import Pin
2   from apa106 import APA106
3   from time import sleep
4
5   NUM_LEDS = 3
6
7   pin = Pin(1)
8   leds = APA106(pin, NUM_LEDS)
9   colors = [
10      (255, 0, 0),
11      (0, 255, 0),
12      (0, 0, 255),
13  ]
14  offset = 0
15
16
17  while True:
18      for num_led in range(NUM_LEDS):
19          next_color = colors[(num_led + offset) % NUM_LEDS]
20          leds[num_led] = next_color
21      leds.write()
22      offset = (offset + 1) % NUM_LEDS
23      sleep(0.5)
```



Ejercicio 6

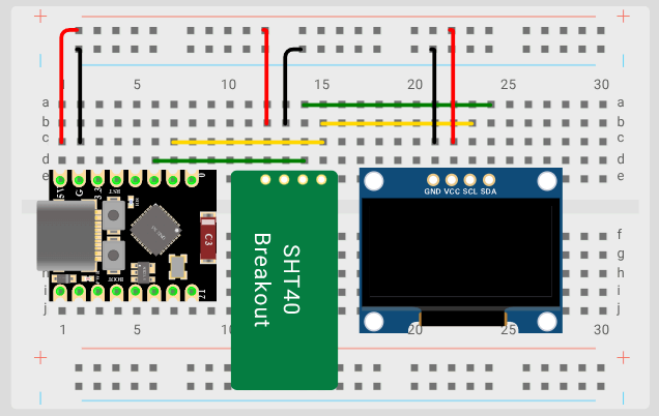
Para este ejercicio se usará la bolsa de componentes 5, la cual se encuentra fuera del kit. A medida que cada uno acabe la práctica, tomarlo para que otro pueda realizarla. Mientras, aquellos que no tengan kit, pueden realizar uno de los ejercicios extra.

Cada componente tiene indicado en sus pines a lo que corresponde y no se requieren resistencias. SCL corresponde al pin 1 (amarillo) y el pin 2 a SDA (verde). La mayor dificultad radica en entender que bajo un mismo cable, pueden ir 2 componentes a la vez tanto para SCL como SDA.

Este ejercicio requiere los módulos adicionales ssd1306 y sht4x, pero están de serie en los microcontroladores proporcionados. La instancia i2c, la cual requiere de ambos pines, permite conectar múltiples dispositivos. El método scan permite comprobar los identificadores de todos ellos.

Usando la instancia i2c, se instancia tanto la pantalla oled como el sensor, debiéndose definir la dirección de cada uno de ellos. En el bucle, al acceder a sensor.measurements se obtienen los valores actuales para la temperatura y humedad. oled.fill(0) vacía la pantalla. oled.text() pone el texto, siendo los dos últimos valores el margen izquierdo y superior de la pantalla. Es necesario ejecutar oled.show() para mostrar los textos a mostrar.

```
from machine import Pin, I2C
from time import sleep
from ssd1306 import SSD1306_I2C
from sht4x import SHT4X


scl = Pin(1)
sda = Pin(2)
i2c = I2C(0, scl=scl, sda=sda)
print(i2c.scan())

oled = SSD1306_I2C(128, 64, i2c, 0x3C)
sensor = SHT4X(i2c, 0x44)


while True:
    temperature, humidity = sensor.measurements
    oled.fill(0)
    temperature_text = "Temp: {:.1f} C".format(temperature)
    humidity_text = "Hum:  {:.1f} %".format(humidity)
    oled.text(temperature_text, 0, 10)
    oled.text(humidity_text, 0, 30)
    oled.show()
    sleep(2)
```



Extra 1

Esta práctica no cuenta con simulación, debiéndose hacer de forma física. El objetivo es utilizar una fotoresistencia para, en caso de reducirse lo suficiente la luz ambiental, que se encienda un LED. Para ello, debe hacerse un divisor de voltaje. Para ello, se proporciona una resistencia de 10KΩ. Para el LED que se enciende, puede usarse la resistencia y el LED del ejercicio 1.

Las instrucciones, el código fuente y el diagrama pueden encontrarse en la carpeta "demos" del repositorio de Github, como *"extra1"*. No hay ejercicio para el código fuente, sólo la resolución, pero es necesario modificar el valor de umbral según la luz ambiental.

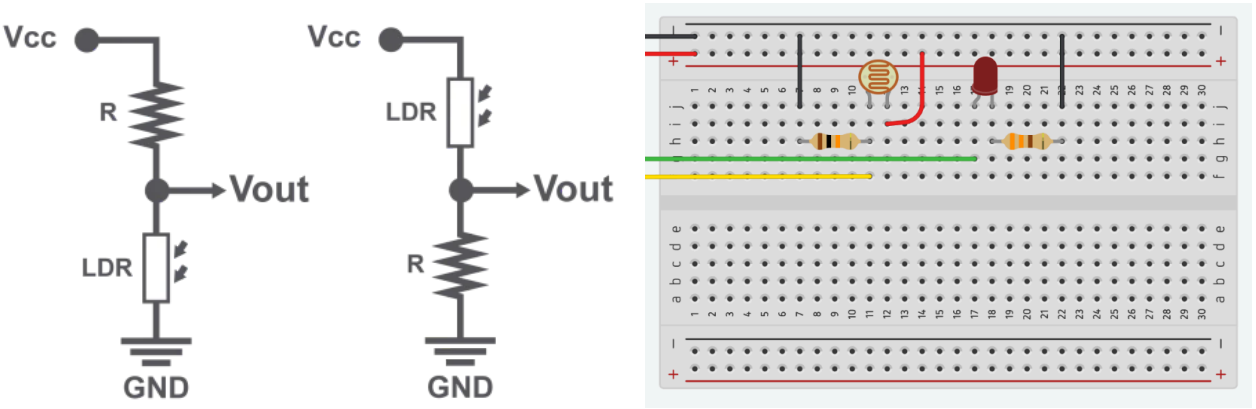
Para hacer el divisor de voltaje, conectamos el LDR al pin de 3.3V en una de sus patillas y en la otra al GPIO del microcontrolador. En dicha patilla, ponemos también una resistencia, que irá en su otro extremo a tierra. De esta forma, el pin GPIO medirá un valor de tensión que dependerá de la resistencia del LDR, y por tanto, de la luz que recibe.

Adicionalmente, conectaremos un LED que se encenderá cuando la luminosidad ambiental sea baja, y se apagará cuando la luminosidad sea alta. Éste irá conectado a otro pin GPIO del microcontrolador, y emplearemos una resistencia de 330 ohmios en serie para limitar la corriente.

La mayor dificultad de esta práctica está en entender cómo realizar el divisor de voltaje y en qué consiste. Tenemos dos posibles formas:

Mayor luz, menor voltaje (izquierda):
Al haber más luz, la resistencia LDR disminuye, por lo que *"escapará"* más electricidad hacia tierra, llegando menos a Vout (el pin GPIO).

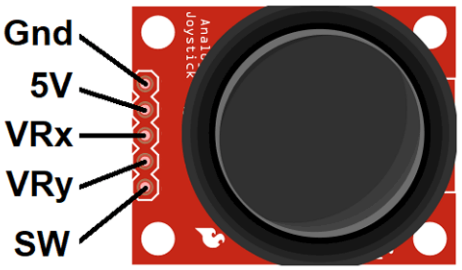
Mayor luz, más voltaje (derecha):
Al haber más luz, disminuye la resistencia LDR, y con ello llega más electricidad a Vout (pin GPIO).



En las instrucciones, se indica la predisposición segunda (derecha). La resistencia de 10KΩ además de proteger nuestro GPIO, ayuda a que la variación de voltaje sea lineal en el rango medio, ya que el LDR varía su resistencia entre 100Ω y 1MΩ, siendo demasiado brusco según los cambios de luz.

Extra 2

Este ejercicio cuenta con simulación, pero para ver los resultados, debe hacerse de forma física. El objetivo es emplear el joystick proporcionado, para obtener los valores de X, Y y cuando se pulsa el botón del joystick. Para ello, debe usarse el ADC (analógico a digital), el cual permite leer valores de entrada analógicamente.



Es importante remarcar, que la predisposición de los pines en el joystick del simulador y la de los joysticks suministrados no es la misma.

ADC no funciona con todos los pines del ESP32, pero sí con los usados durante las prácticas. Para usar ADC, debemos crear una instancia con el pin correspondiente, y aplicar una atenuación.

La atenuación es necesaria porque, para hacer las mediciones, el microcontrolador compara el voltaje recibido con un valor de referencia, siendo éste 1.1V. Para medir voltajes mayores se aplica una atenuación de -11dB a la señal de entrada, aumentando la capacidad de lectura hasta 3.9V teóricos, siendo seguro hasta 3.3V. Esta atenuación corresponde a ADC.ATTN_11DB.

Extra 3

Esta práctica no cuenta con simulación, debiéndose hacer de forma física. El objetivo de esta práctica es comprender cómo se realiza la comunicación WiFi con Micropython, haciendo uno de Access Point (servidor) y otro como cliente WiFi. El cliente dispone de un botón y un LED, encendiéndose este y mandando la señal al servidor indicando que se ha pulsado el botón. Cuando recibe la orden, el servidor emite un pitido por el altavoz, y cuando deja de pulsarse, el pitido cesa. El servidor cuenta igualmente con un LED. Esta práctica es similar a un telégrafo o un timbre.

El servidor inicia una red WiFi con el patrón morse-xxxx, la cual buscará el cliente. En caso de haber varias personas haciendo este ejercicio, se recomienda cambiar el patrón en cliente y servidor, para evitar conflictos. La red no tiene contraseña y está abierta. El PWM iniciado corresponde al altavoz, pudiéndose cambiar los parámetros PWM_FREQ y DUTY para cambiar el tono, volumen… El servidor socket.socket() es un servidor TCP/IP en el puerto 1234, aceptando conexiones dentro del bucle. Cuando reciba los mensajes ON o OFF, encenderá o apagará el altavoz y el LED.

El cliente busca redes WiFi abiertas con el patrón, y se conectará a la primera según la fuerza de la señal. Si no encuentra ninguna, terminará el programa, debiéndose reiniciar. Una vez conectado, inicia el socket TCP/IP con socket.socket() al servidor, el pin del botón y el LED. En el bucle va comprobando el valor del botón, y cuando se pulsa enciende el LED y manda los mensajes ON/OFF según corresponda.

La electrónica de esta práctica es muy sencilla, siendo la mayor dificultad el funcionamiento de la red WiFi. Si ambos microcontroladores están muy alejados, acercarlos. Si hay varios equipos haciendo la práctica, cambiar el patrón de la red.