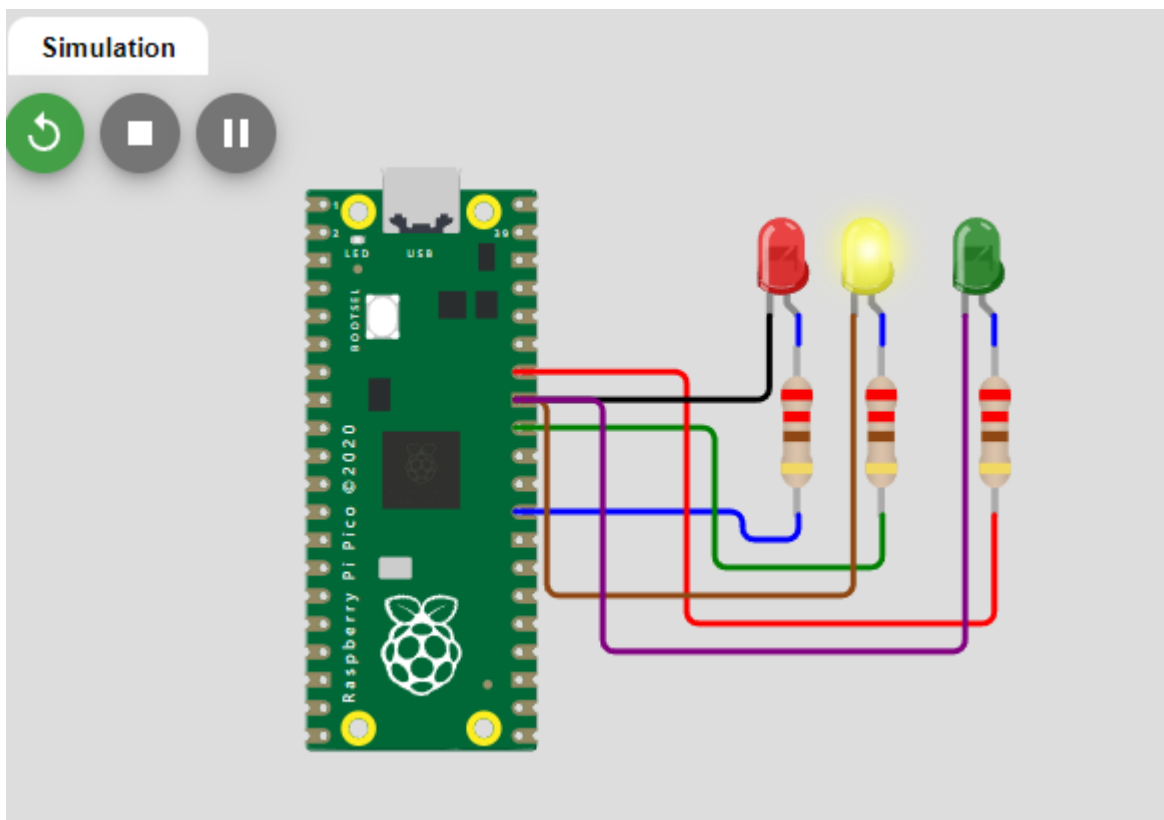


Pasos para la conexión de la Raspberry PI PICO (Circuito)

Implementos.

- Placa Raspberry Pi Pico.
 - LED de colores: rojo, amarillo y verde.
 - 3 resistencias de 220 ohmios.
1. Conectar el Led de color rojo a la salida de GP22.
 2. Conectar el Led de color Amarillo a la salida de GP27.
 3. Conectar el Led de color Verde a la salida de GP28.
 4. Recuerda que ANODO (+) del Led se conecta a la resistencia.
 5. Recuerda CATODO (-) del Led va a la conexión GDN o negativo.
 - 6.



PASOS PARA LA PROGRAMACIÓN DE TU SEMÁFORO.

Lo primero se importan las bibliotecas necesarias ("**Pin**" para controlar los pines GPIO y "**time**" para pausas). Luego se configuran los pines GPIO correspondientes a los LEDs rojo, amarillo y verde como salidas (**Pin.OUT**). La función **cambiar_semaforo()** se encarga de cambiar el estado de los LEDs según el patrón del semáforo (rojo -> amarillo -> verde ->).

```

1  from machine import Pin
2  import time
3
4  # Configurar los pines de salida para los LEDs
5  pin_rojo = Pin(22, Pin.OUT)
6  pin_amarillo = Pin(27, Pin.OUT)
7  pin_verde = Pin(28, Pin.OUT)

```

Cuando llamas al método `on()` en un objeto **Pin**, estás indicando que quieres establecer el pin en un estado de "alto" o "encendido", lo que significa que se aplicará un voltaje positivo al pin, lo que puede activar un LED u otro dispositivo conectado al pin.

Cuando llamas `off()` hace lo contrario al método `on()`, es decir, establece el pin en un estado de "bajo" o "apagado", lo que quita el voltaje del pin.

```

pin_verde = Pin(28, Pin.OUT)

# Función para cambiar el estado de los LEDs para el semáforo
def cambiar_semaforo():
    pin_rojo.on()
    time.sleep(2) # Tiempo en rojo
    pin_rojo.off()

```

Finalmente, se inicia un ciclo **while True** que ejecutará continuamente el cambio de estado del semáforo. Puedes ajustar los tiempos de cada color cambiando los valores dentro de `time.sleep()` según tus necesidades.

```

26  while True:
27      cambiar_semaforo()

```