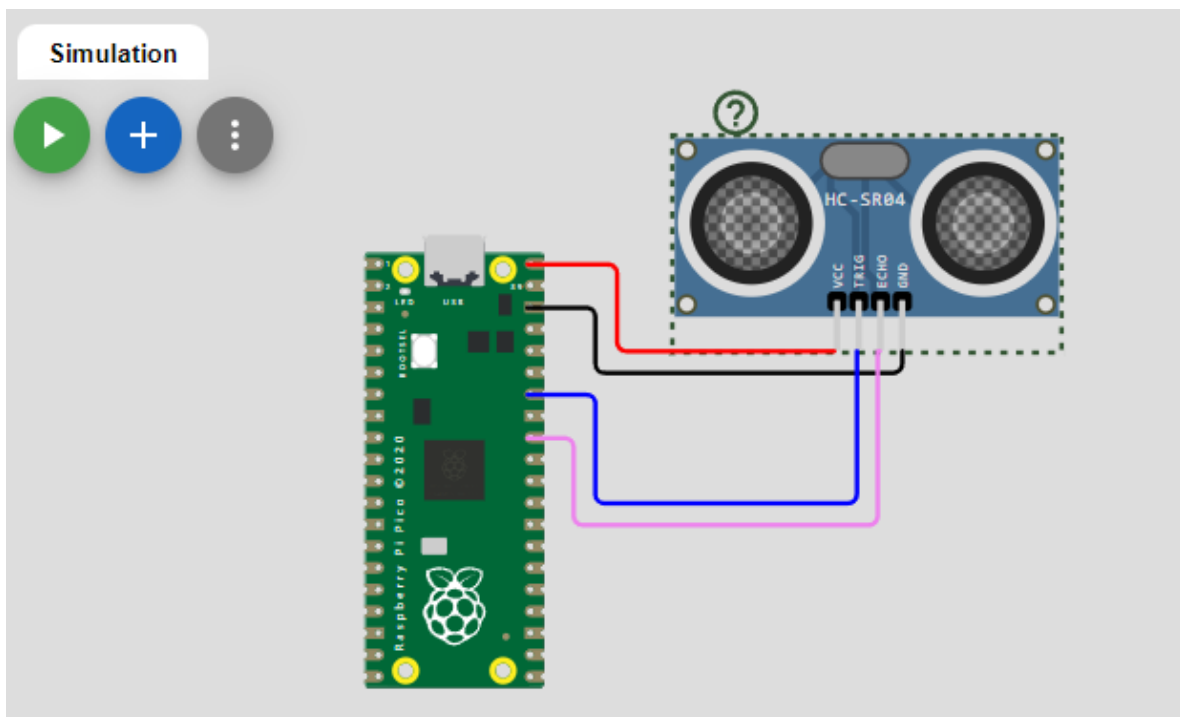


Pasos para la conexión de la Raspberry Pi PICO (Circuito) y Sensor de Distancia ultrasónico HC-SR04

Implementos.

- Placa Raspberry Pi Pico.
 - Sensor de Distancia ultrasónico HC-SR04
1. Conectar Vcc a VBUS “Salida de voltaje Positivo”
 2. Conectar GND de tu sensor a GND de la placa Raspberry Pi PICO.
 3. Conectar TRIG (disparador: **La longitud del pulso es proporcional al tiempo que tardó en detectarse la señal transmitida**) a GP28.
 4. Conectar ECHO (El pin Eco produce un pulso cuando se recibe la señal reflejada) a GP27.



PASOS PARA LA SIMULACIÓN DE TU SENSOR

Importación de módulos: Importamos las clases y funciones necesarias desde los módulos **machine** y **utime**. **machine** se utiliza para interactuar con los pines GPIO del Raspberry Pi Pico, mientras que **utime** se usa para manejar el tiempo en microsegundos.

Definición de pines: Se definen los pines **trig_pin** y **echo_pin** que están conectados al sensor HC-SR04. El pin **Trig** se configura como salida (**OUT**) para enviar el pulso, y el pin Echo se configura como entrada (**IN**) para recibir la señal de eco del pulso.

Realizado por: Geraldine Rodríguez.

```

1  from machine import Pin
2  import utime
3
4  # Definir los pines Trig y Echo
5  trig_pin = Pin(28, Pin.OUT)
6  echo_pin = Pin(27, Pin.IN)
7

```

Función medir_distancia(): Esta función realiza la medición de distancia utilizando el sensor HC-SR04. Se generan pulsos en el pin **Trig** para iniciar la medición y se calcula la duración del pulso de eco en el pin **Echo** para determinar la distancia.

```

# Función para medir la distancia
def medir_distancia():
    # Generar un pulso corto en el pin Trig
    trig_pin.low()
    utime.sleep_us(2)
    trig_pin.high()
    utime.sleep_us(5)
    trig_pin.low()

    # Medir el tiempo de respuesta en el pin Echo
    while echo_pin.value() == 0:
        pulso_inicio = utime.ticks_us()
    while echo_pin.value() == 1:
        pulso_fin = utime.ticks_us()

    # Calcular la duración del pulso
    duracion_pulso = utime.ticks_diff(pulso_fin, pulso_inicio)

    # Calcular la distancia en centímetros
    distancia_cm = duracion_pulso * 0.034 / 2

    return distancia_cm

```

Bucle principal while True: En este bucle, se llama repetidamente a la función **medir_distancia()** para obtener la distancia medida por el sensor. La distancia se imprime en la consola con dos decimales y luego el programa espera 1 segundo antes de realizar la próxima medición.

Manejo de interrupción (KeyboardInterrupt): Se utiliza un bloque **try-except** para manejar la interrupción. Cuando se detiene el programa manualmente, se imprime un mensaje indicando que el programa ha sido detenido por el usuario.

```
30
31 try:
32     while True:
33         distancia = medir_distancia()
34         print("Distancia: {:.2f} cm".format(distancia))
35         utime.sleep(1) # Esperar 1 segundo antes de la próxima medición
36
37 except KeyboardInterrupt:
38     print("\nPrograma detenido por el usuario")
```

¡Éxitos!