PRACTICA RASPBERRY PI PICO

Ángel Gabriel Escalante Alfaro[1], Ana Elizabeth Tovar Figueroa[2]
[1]Ingeniería en Comunicaciones y Eléctrica, ``Cómputo'', 8vo semestre
[2]Ingeniería en Comunicaciones y Eléctrica, ``Control'', 8vo semestre
Instituto Politécnico Nacional
Escuela Superior de Ingeniería Mecánica y Eléctrica ESIME, Zacatenco
aescalantea2001@alumno.ipn.mx, atovarf1900@alumno.ipn.mx

Boletín No. 102, 1o. de mayo de 2024

Abstract

In this work, a step-by-step explanation will be made about the installation of the program to make the proper use of the Raspberry Pi Pico, as well as a brief explanation about the microcontroller to help make circuits later having the knowledge of the pins it has; for future use with the microcontroller; As part of the practice, there are exercises to have a small guide on how to carry out its use, they are provided with the basic codes to perform the activity plus schemes to connect the devices necessary to perform the exercise such as the pins to use, the exercises to be performed are the basic ones such as turning on the blink, but they are of great help when starting to use the microcontroller, therefore the practice can be used by those who like to know a new microcontroller and perhaps later continue using it.

1. Introducción

El uso de los microcontroladores ha estado presente recientemente para el uso de diferentes aparatos domésticos, porque ayuda a tener un dispositivo el código para las diferentes funcionalidades, y con este uso se ahorran gran parte de componentes electrónicos. Por lo tanto, el conocimiento del uso de estos microcontroladores abre un gran camino en el ámbito profesional, por lo cual en este trabajo se dan los pasos necesarios para conocer un poco el uso de la Raspberry Pi Pico, realizando distintos ejercicios para ir conociendo las funciones de cada pin que se tiene, apoyándose en algunos esquemas.

2. Objetivos

- Introducción a la arquitectura de la tarjeta Raspberry-Pi-Pico.
- Diseño con tablas de decodificación.
- Instrucción delay para diseños con temporización.
- Diseño de semáforos y contadores básicos.

3. Generalidades del Raspberry Pi-Pico

La Raspberry Pi Pico H es una placa de desarrollo de microcontroladores de bajo costo y alto rendimiento con interfaces digitales flexibles. Presenta el RP2040, que marca el primer microcontrolador de Raspberry Pi diseñado internamente.

3.1 Especificaciones

- Microcontrolador RP2040 diseñado por Raspberry Pi en el Reino Unido.
- Procesador de doble núcleo ARM Cortex M0+, reloj flexible que funciona hasta 133 MHz.
- 264kB de SRAM.
- 2MB de memoria flash integrada.
- Módulo con carcasa que permite soldar directamente a placas de soporte.

- Soporte de dispositivo y host USB 1.1.
- Convertidor analógico-digital (ADC) de 3 canales de 12 bits.
- 26 pines GPIO multifunción.
- Interfaces SPI, I2C y UART.
- Puerto de depuración de cable serie (SWD).
- Dimensiones: 21 mm $\times$ 51 mm.

3.2 Características y Aplicaciones

Sus principales **características** son: Bajo costo, alto rendimiento, interfaces digitales flexibles, fácil de usar y amplia comunidad de soporte.

En cuanto a sus **aplicaciones**, destacan:

- Prototipado electrónico.
- Dispositivos portátiles.
- Internet de las cosas (IoT).
- Domótica.
- Robótica.
- Educación.

Disponibilidad: La Raspberry Pi Pico H está disponible para su compra en la tienda Raspberry Pi y en una variedad de distribuidores.

4. Diagrama de distribución de pines

Se muestra en la Figura 1, el número de pines, nombre y funciones para realizar las prácticas.

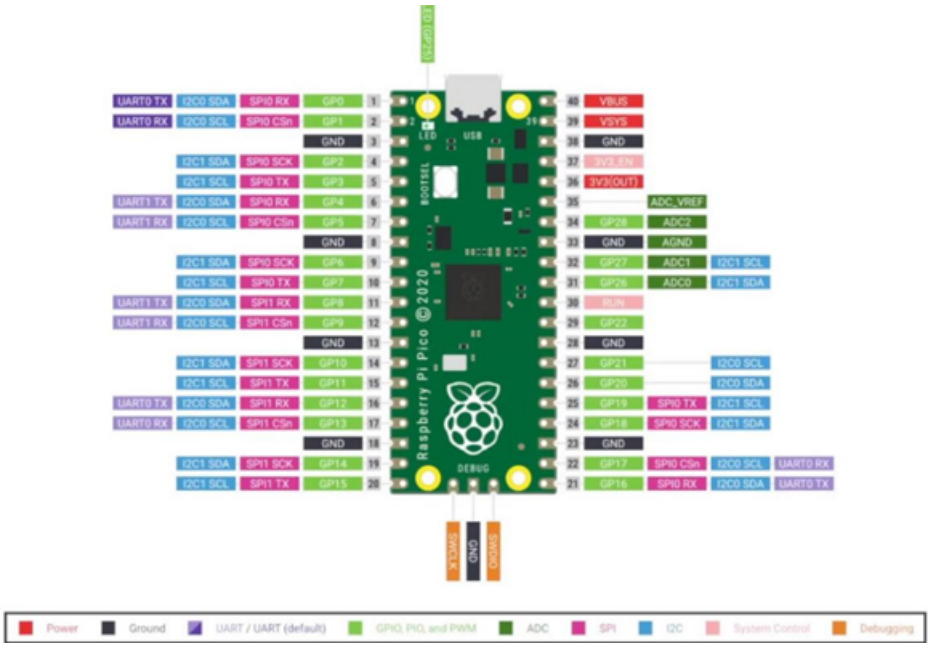

Figura 1 Diagrama de Raspberry Pi Pico H 2021.

5. Pasos para actualizar el firmware

1. Para poder utilizarlo tenemos que actualizar el firmware del Raspberry y lo primero es entrar en modo actualización, para ello mantenemos presionado el botón **BOOTSEL** de la Raspberry antes de conectarlo al PC y una vez conectado al PC soltamos el botón (ver Figura 2).

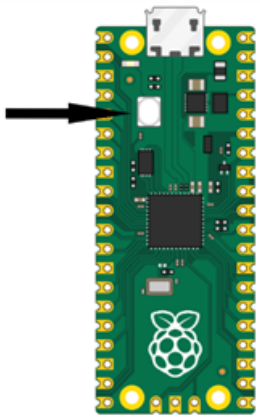

Figura 2 Botón BOOTSEL.

2. En el PC nos aparecerá un dispositivo de almacenamiento externo.
3. Entraremos al sitio web (Enlace 1), en el cual descargaremos el último firmware que nos aparezca (Figura 3).

Figura 3 Vista de la versión del Firmware que se usa.

4. Una vez descargado el archivo, lo localizaremos y lo copiaremos dentro del dispositivo de almacenamiento de la Raspberry.
5. Una vez copiado, la placa se va a reiniciar y estará lista para la programación.

6. Descarga del programador Thonny

1. Para comenzar a programar debemos ir a la página oficial de Thonny y descargar la versión más reciente para el sistema operativo (Enlace 2).

Figura 4 Vista de la página de descargas de Thonny.

2. Una vez descargado e instalado dejaremos la configuración inicial. En la Figura 5, se muestra la selección del idioma deseado para trabajar.

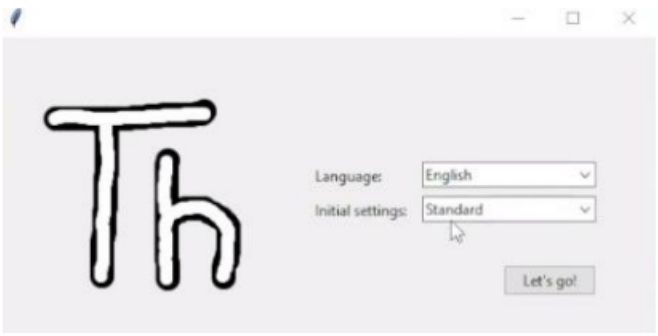

Figura 5 Vista de elección de idioma.

3. El programa viene con Python, y necesitamos programar con **MicroPython**. Para configurarlo tenemos que ir a la parte inferior derecha del compilador y seleccionar nuestra placa.

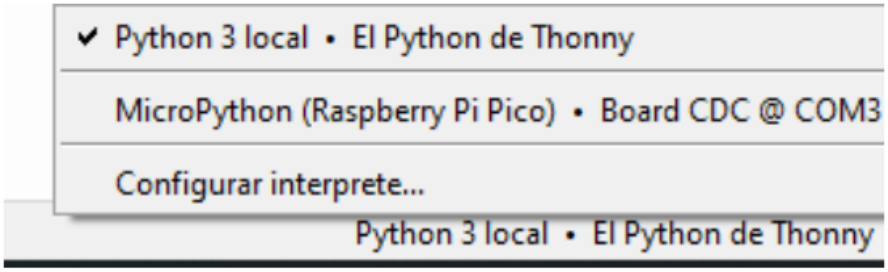

Figura 6 Configuración para MicroPython.

4. Por último, vamos a poder guardar los programas en el mismo Raspberry o en la PC. Es recomendado guardarlos en la misma tarjeta una vez que hayamos hecho un programa. Se debe nombrar con extensión `.py` (Figuras 7 y 8).

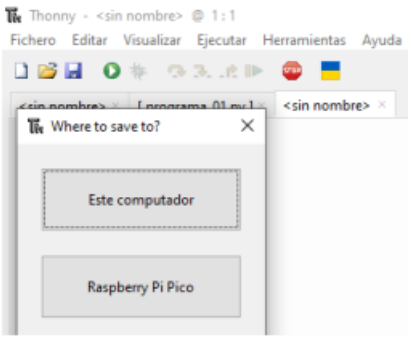

Figura 7 Elección de destino de guardado.

Figura 8 Guardar el proyecto con extensión .py.

7. Programas para iniciar

Programa 1. Blink LED en la tarjeta.

```
from machine import Pin
from utime import sleep

led = Pin(25, Pin.OUT)

while True:
    led.on()
    sleep(0.5)
    led.off()
    sleep(0.5)
```

Programa 2. Blink utilizando un LED externo a la tarjeta.

Considera el pin 24 (GPIO 18) para conectar el LED. Revisa las conexiones en la Figura 9.

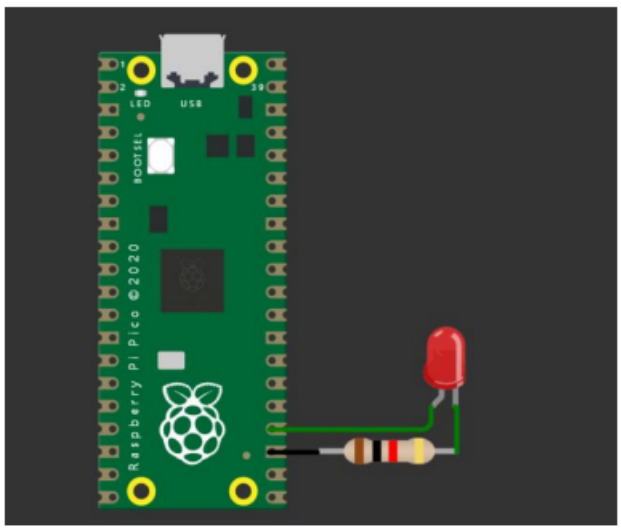

Figura 9 Diagrama de conexión para un LED externo.

```
from machine import Pin
from utime import sleep

led = Pin(18, Pin.OUT)
```

```
while True:
    led.on()
    sleep(0.5)
    led.off()
    sleep(0.5)
```

Programa 3. Utilizar un pushbutton para controlar el encendido del LED.

Considera el pin 19 (GPIO 14) para conectar el LED y el pin 20 (GPIO 15) para el pushbutton. Utiliza resistencias de 330 Ω para el LED y de 10 KΩ para el pushbutton (Figura 10).

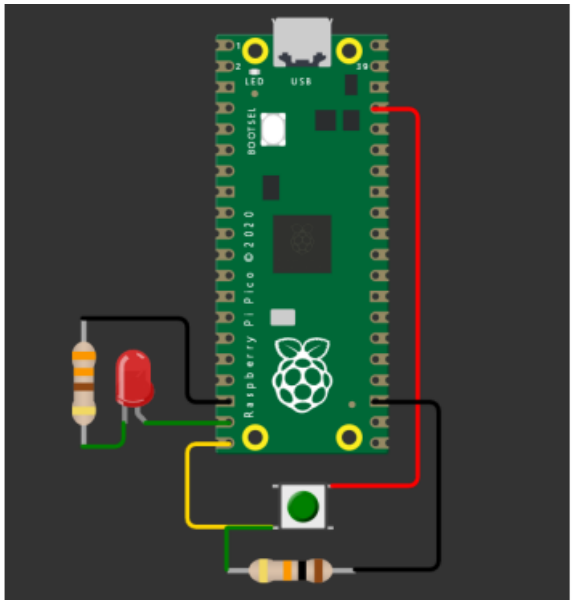

Figura 10 Diagrama del circuito con pushbutton y LED.

```
import machine
import utime

# Configura los pines GPIO
led_pin = machine.Pin(14, machine.Pin.OUT)
button_pin = machine.Pin(15, machine.Pin.IN, machine.Pin.PULL_DOWN)

# Función para verificar el estado del botón
def check_button():
    return button_pin.value()

# Bucle principal
while True:
    # Verifica el estado del botón
    if check_button():
        led_pin.value(1) # Si se presiona el botón, enciende el LED
    else:
        led_pin.value(0) # Si no se presiona el botón, apaga el LED

    # Espera un breve período para evitar rebotes
    utime.sleep_ms(20)
```

7.1 Diseño de semáforos

Programa 4. Semáforo simple.

El siguiente programa implementa un semáforo simple utilizando la instrucción `sleep` para temporizar (Figura 11).

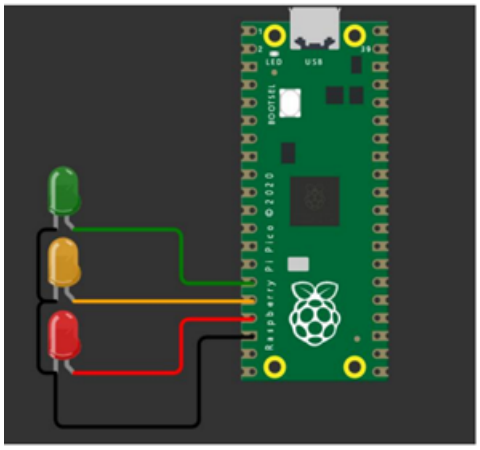

Figura 11 Diagrama de conexión de los LEDs para semáforo.

```
import machine
import utime

# Configurar pines GPIO
verde_pin = machine.Pin(11, machine.Pin.OUT)
amarillo_pin = machine.Pin(12, machine.Pin.OUT)
rojo_pin = machine.Pin(13, machine.Pin.OUT)

def encender_verde():
    verde_pin.on()
    amarillo_pin.off()
    rojo_pin.off()

def encender_amarillo():
    verde_pin.off()
    amarillo_pin.on()
    rojo_pin.off()

def encender_rojo():
    verde_pin.off()
    amarillo_pin.off()
    rojo_pin.on()

while True:
    encender_verde()
    utime.sleep(5)
    encender_amarillo()
    utime.sleep(2)
    encender_rojo()
    utime.sleep(5)
```

Programa 5. Semáforo simple con parpadeo.

Para incluir un parpadeo en el LED de color verde durante el estado ámbar:

```
import machine
import utime

verde_pin = machine.Pin(11, machine.Pin.OUT)
amarillo_pin = machine.Pin(12, machine.Pin.OUT)
rojo_pin = machine.Pin(13, machine.Pin.OUT)

def encender_verde():
    verde_pin.on(); amarillo_pin.off(); rojo_pin.off()

def encender_amarillo():
    verde_pin.off(); amarillo_pin.on(); rojo_pin.off()

def encender_rojo():
    verde_pin.off(); amarillo_pin.off(); rojo_pin.on()

def parpadear(luz, duracion):
    for _ in range(duracion):
        luz.off()
        utime.sleep(0.5)
        luz.on()
        utime.sleep(0.5)

while True:
    encender_verde()
    utime.sleep(10)

    parpadear(amarillo_pin, 3)
    utime.sleep(1)

    encender_rojo()
    utime.sleep(10)

    encender_amarillo()
    utime.sleep(5)
```

Programa 6. Semáforo simple con parpadeo utilizando una tabla de decodificación.

Para el diseño de decodificadores, la declaración de arreglos es básica.

```
import machine
import utime

tabla_decodificacion = [
    [1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 0, 0], # Verde
    [1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 0, 0],
    [1, 1, 0], [0, 1, 0], [1, 1, 0], [0, 1, 0], # Inicia parpadeo
    [1, 1, 0], [0, 1, 0],                       # Concluye parpadeo
    [0, 0, 1], [0, 0, 1], [0, 0, 1], [0, 0, 1], # Rojo
    [0, 0, 1], [0, 0, 1], [0, 0, 1], [0, 0, 1]
]

verde_pin = machine.Pin(11, machine.Pin.OUT)
```

```
amarillo_pin = machine.Pin(12, machine.Pin.OUT)
rojo_pin = machine.Pin(13, machine.Pin.OUT)

def actualizar_semaforo(estado):
    verde_pin.value(tabla_decodificacion[estado][0])
    amarillo_pin.value(tabla_decodificacion[estado][1])
    rojo_pin.value(tabla_decodificacion[estado][2])

estado_actual = 0
while True:
    actualizar_semaforo(estado_actual)
    estado_actual = (estado_actual + 1) % len(tabla_decodificacion)
    utime.sleep(1)
```

7.2 Diseño de Contadores con Decodificadores

Programa 7. Contador de 4 bits con display de siete segmentos.

Considérese el diseño de un contador de 4 bits, de carrera libre, que está conectado a un decodificador hexadecimal para un display de 7 segmentos de cátodo común. El diseño incluye un push button que funciona como Reset.

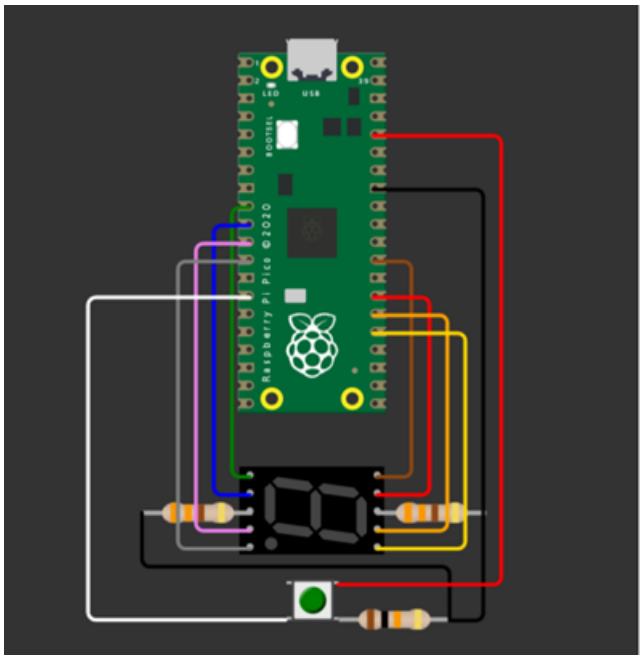

Figura 12 Diagrama de conexiones del circuito para contador y display.

```
import machine
import utime

segmentos_display = [
    machine.Pin(20, machine.Pin.OUT), # A
    machine.Pin(19, machine.Pin.OUT), # B
    machine.Pin(8,  machine.Pin.OUT), # C
    machine.Pin(7,  machine.Pin.OUT), # D
    machine.Pin(6,  machine.Pin.OUT), # E
    machine.Pin(21, machine.Pin.OUT), # F
    machine.Pin(22, machine.Pin.OUT)  # G
```

```
]

push_button = machine.Pin(10, machine.Pin.IN, machine.Pin.PULL_DOWN)

segmentos_tabla = [
    [1, 1, 1, 1, 1, 1, 0], # 0
    [0, 1, 1, 0, 0, 0, 0], # 1
    [1, 1, 0, 1, 1, 0, 1], # 2
    [1, 1, 1, 1, 0, 0, 1], # 3
    [0, 1, 1, 0, 0, 1, 1], # 4
    [1, 0, 1, 1, 0, 1, 1], # 5
    [1, 0, 1, 1, 1, 1, 1], # 6
    [1, 1, 1, 0, 0, 0, 0], # 7
    [1, 1, 1, 1, 1, 1, 1], # 8
    [1, 1, 1, 1, 0, 1, 1], # 9
    [1, 1, 1, 0, 1, 1, 1], # A
    [0, 0, 1, 1, 1, 1, 1], # B
    [1, 0, 0, 1, 1, 1, 0], # C
    [0, 1, 1, 1, 1, 0, 1], # D
    [1, 0, 0, 1, 1, 1, 1], # E
    [1, 0, 0, 0, 1, 1, 1]  # F
]

def decoder_7seg(n):
    for i, pin in enumerate(segmentos_display):
        pin.value(segmentos_tabla[n][i])

contador = 0
while True:
    if push_button.value() != 1:
        contador = (contador + 1) % 16
    decoder_7seg(contador)
    utime.sleep(1)
```

Programa 8. Contador de números primos de 4 bits.

Escribir un programa como el anterior presenta una ventaja cuando se desea realizar un conteo de estados no consecutivos, como números primos (2, 3, 5, 7, 11 y 13).

```
import machine
import utime

# ... (Misma configuración de pines y tabla que el Programa 7) ...

def es_primo(numero):
    if numero <= 1:
        return False
    for i in range(2, int(numero**0.5) + 1):
        if numero % i == 0:
            return False
    return True

contador = 1
while True:
```

```
    if push_button.value() == 1:
        contador = 1
        utime.sleep_ms(100)

    if es_primo(contador):
        decoder_7seg(contador)
        utime.sleep(1)

    if contador == 13:
        contador = 0

    contador += 1
```

Programa 9. Contador ascendente-descendente de 4 bits.

Es posible controlar la dirección del conteo. Si `direccion` es 0, contará ascendentemente y si es 1 descendentemente.

```
import machine
import utime

# ... (Misma configuración de segmentos_display y segmentos_tabla) ...

# Pin para controlar la dirección
direccion_pin = machine.Pin(10, machine.Pin.IN, machine.Pin.PULL_DOWN)

contador = 0
direccion = 0

while True:
    if direccion_pin.value() == 1:
        direccion = 1
    else:
        direccion = 0

    decoder_7seg(contador)
    utime.sleep(1)

    if direccion == 0:
        contador += 1
        if contador > 15:
            contador = 0
    else:
        contador -= 1
        if contador < 0:
            contador = 15
```

8. Conclusiones

La práctica de Raspberry Pi Pico proporciona una introducción práctica y detallada al mundo de los microcontroladores, específicamente a través del uso del RP2040, el microcontrolador diseñado internamente por Raspberry Pi. Desde la actualización del firmware hasta la configuración del entorno de programación y la implementación de ejercicios básicos como el parpadeo de LEDs y el diseño de semáforos y contadores, esta práctica ofrece una guía completa para familiarizarse con el uso de Raspberry Pi Pico.

La versatilidad y el bajo costo de Raspberry Pi Pico lo convierten en una herramienta atractiva para una

variedad de aplicaciones, desde prototipado electrónico hasta proyectos de Internet de las cosas (IoT) y educación en ingeniería. Además, la amplia comunidad de soporte proporciona un entorno propicio para el aprendizaje continuo y el desarrollo de habilidades en el ámbito de la ingeniería.

Referencias y recursos electrónicos

[1] Raspberry Pi Pico Download. Disponible en: `https://micropython.org/download/RPI_PICO/`

[2] Sitio web de Raspberry Pi Pico. Disponible en: `https://www.raspberrypi.com/products/raspberry-pi-pico/`

[3] Guía de inicio rápido de Raspberry Pi Pico. Disponible en: `https://www.raspberrypi.com/documentation/pico/getting-started/`

[4] Hoja de datos de Raspberry Pi Pico. Disponible en: `https://datasheets.raspberrypi.com/pico/picodatasheet.pdf`

[5] Thonny Python IDE. Disponible en: `https://thonny.org/`

[6] Raspberry Pi (**2023, Febrero 13**). *Programa tu RPi Pico.* Recuperado de `https://raspberrypi.cl/2023/02/13/programa-tu-rpi-pico/`

Escalante Alfaro, Á. G., Tovar Figueroa, A. E. *(2026). PRACTICA RASPBERRY PI PICO. Boletín UPIITA. año XX, (NÚM) 2026.*