Practica 2 - Instalación de Micropython en el microcontrolador RP2040
Sistemas Embebidos

Torres Anguiano Azael Arturo

2025-09-02

1 Introducción

El RP2040 tiene un conjunto de elementos importantes a la hora de entender los sistemas embebidos, primeramente, su uso especifico, pero de fácil implementación y ligero costo nos aporta un componente importante a implementar en decenas de proyectos. Durante este trabajo se inicializará e implementara micropython dentro de un RP2040, y conociendo sus características se manipulará como primer acercamiento para entenderlo a profundidad, manipulando la intensidad de su led integrado y probando su convertidor analógico digital para leer su temperatura interna.

2 Antecedentes

2.1 Microcontrolador RP2040

El microcontrolador RP2040 es un componente derivado de la Raspberry Pi, a diferencia de una computadora de uso general o la Raspberry Pi, este componente es de bajo costo, tiene un alto rendimiento y un fácil uso en su implementación como controlador. Posee un procesador ARM cortex-M0+, una SRAM de 264kB, soporte de 16MB para memoria flash fuera del chip en el bus QSPI, controlador DMA, 30 pines GPIO, 2 UARTs, 2 controladores SPI, 2 de tipo i2c, 16 canales PWM, un controlador de USB de 1.1 y 8 maquinas de estado de tipo PIO. Por todos estos elementos ayudan a que cualquiera que sea la aplicación de microcontrolador que se dese utilizar, desde de machin learning, control de motores, agricultura de audio, el RP2040 tiene el soporte y funciones necesarias para su funcionamiento.

2.2 Micropython

Micropython es un ligero y eficiente implementación del lenguaje de programación Python 3 que incluye un pequeño set de las librerías estándares de Python, pero optimizado para microcontroladores. Este software se utiliza principalmente para aprendizaje, ya que, aunque esta optimizado para microcontroladores, el intérprete (como el Python original) utiliza muchos recursos ya que no tiene compilador, es decir, las instrucciones se fabrican y operan a la marcha sin antes cargar y analizar el programa, como si pasa con lenguajes como C o hasta en Java

2.3 Primeros Pasos

El “hola mundo” es y ha sido el primer programa que se ve a la hora de implementar o aprender un nuevo lenguaje, circuito o elemento embebido, el “hola mundo” siempre tiende a ser un programa que imprima una

cadena de caracteres con esta frase, para entender las variables iniciales, las salidas y la estructura básica del programa, para el caso del RP2040 y otros microcomponentes similares utilizamos la “Atenuación del Led” es decir, cambiamos la intensidad del led en el microcomponente.

2.4 Loop Infinito

A diferencia de otros programas en computadora, los programas en sistemas embebidos siempre deben tener un componente de loop infinito, si bien en programas comunes podemos lograrlo, en los sistemas embebidos debemos tener bien en claro que una vez inician no debemos detenerlos, ya que son los que hacen que nuestro hardware funcione, de lo contrario serán piezas inútiles sin funciones.

3 Desarrollo

Como primer ejercicio a desarrollar se cambiarán los dos programas del parpadeo del led, estos programas simplemente encienden el led, por lo que en cada uno repetiremos ciertas líneas de código, estas líneas corresponden al encendido y apagado del led, intentaremos encender y apagar el led en un corto periodo de tiempo para que de la apariencia de tener una iluminación baja. Para el primer programa cambiaremos la función led.toggle por las led.on y led.off para controlar en encendido y apagado, y colocaremos funciones sleep con valores pequeños para que el programa pase de un estado a otro y de la ilusión de la baja intensidad.

```
    sleep_ms(1)
    led.off()
    sleep_ms(20)            # Wait for 500ms
    led.on()
```

Figure 1: Modificación del primer programa. Agregamos función off y on

Para el segundo programa se repite la función timer dos veces y se iguala la frecuencia, esto provocara que entre el tiempo de encendido y apagado no haya diferencia (y sea mínima) y de la ilusión del ejercicio pasado.

```
timer = Timer()             # Create the Timer object
timer.init(freq=50,         # Timer frequency set to 2.5Hz
    mode=Timer.PERIODIC,    # Timer will run endlessly (not one-shot)
    callback=blink          # Set callback function: blink
)

led = Pin(25, Pin.OUT)      # Setup pin 25 (sentinel LED) as output

timer = Timer()             # Create the Timer object
timer.init(freq=50,         # Timer frequency set to 2.5Hz
    mode=Timer.PERIODIC,    # Timer will run endlessly (not one-shot)
    callback=blink          # Set callback function: blink
)
```

Figure 2: Modificación del segundo programa, repetición del timer

Para el tercer ejercicio simplemente aplicaremos la fórmula de conversión de Celsius a Fahrenheit y usamos la función de round propia de Python para redondear los valores.

```
x = adc.read_u16()       # Read ADC
temp = x * K + 437.23    # Convert to celcius
tempF= (1.8*temp) + 32 ;#Convert to Far
R1 = round (temp,2)
R2 = round (tempF,2)
print(f'Temp:  {R1}°C') # Print temperature
print(f'TempF: {R2} °F')
```

Figure 3: Programa calculo y salida de temperatura

Por último, modificaremos el primer código y eliminaremos los retardos para tener un led atenuado, esto es sencillo si se revisa la documentación de PWM, en ella se ve que para inicializar un led en PWM se usa la función PWM(Led) y para cambiar la intensidad se usa el método duty_16(bits) donde el valor de bit puede ir desde 0 a 65545 para cambiar la intensidad del led

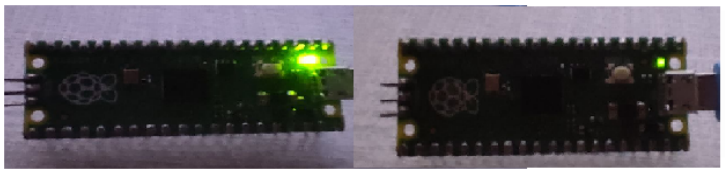
```
from machine import Pin,PWM

led = Pin(25, Pin.OUT)       #
pwm = PWM(led)               #
pwm.duty_u16(400)            #
```

Figure 4: Uso del PWM para atenuar el Led

4 Resultados Experimentales

Finalmente se consiguen los siguientes resultados, la atenuación del led y el calculo de la temperatura en los dos sistemas solicitados



```
MPY: soft reboot
Temp:  25.18°C
TempF: 77.32 °F
Temp:  24.24°C
TempF: 75.63 °F
Temp:  24.24°C
TempF: 75.63 °F
```

Figure 5: Comparación de la intensidad y lectura de la temperatura

5 Conclusiones

Los microcontroladores como el RP2040 nos ayudan a implementar funciones, programas o lecturas de elementos externos (temperatura, sonido, entradas y salidas de otros dispositivos) con bastante facilidad y de forma optima dentro del ambiente de los sistemas embebidos

6 Bibliografia:

1. Barr,M.,et. al,.(2007).Embedded Systems with C and GNU Development Tools. Sebastopol: O'Reilly Madia, Segunda Edición
2. RP2040 Datasheet: A microcontroller by Raspberry Pi.(2023) Raspberry Pi. Recuperado de: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf
3. MicroPython(2014) MicroPython.org. Recuperado de: https://micropython.org