Reporte. Triple Semáforo con Doble Núcleo

Unidad Académica Multidisciplinaria Mante (UAMM)

Arquitectura de Computadoras

Montoya Garza Luis Ángel

Hernández Ruiz Haydee Michelle

Rueda Martínez Alison Michelle

Silva Sánchez Yamilka Arely

7° “E”

M.C. López Piña Daniel

Ciudad Mante, Tamaulipas. Octubre del 2025

Introducción

El microcontrolador RP2040 se emplea en esta práctica para demostrar la ejecución concurrente mediante su arquitectura de doble núcleo. El objetivo es distribuir la carga de trabajo de un sistema de control de tráfico en MicroPython. Un conjunto de semáforos (1 y 2) operará con lógica reactiva y condicional (Núcleo 1), alternando entre modos normal y nocturno basado en la entrada de un pulsador. Simultáneamente, un semáforo 3 mantendrá un ciclo de tiempo fijo y autónomo (Núcleo 0). Esta estrategia de programación busca asegurar la independencia temporal de las funciones y evitar el bloqueo del sistema.

Desarrollo

El código presentado utiliza la capacidad de doble núcleo de la RP2040 para ejecutar dos sistemas de semáforo de manera concurrente en MicroPython. La principal característica es que la lógica del Núcleo 1 controla el estado de los semáforos principales basándose en la activación de un pulsador, alternando entre un ciclo normal y un modo nocturno, mientras que el Núcleo 0 opera un tercer semáforo de forma continua e independiente.

Se importan los módulos Pin, sleep_ms y _thread. Se define un pulsador en el pin 5 con pull-down. A continuación, se configuran las salidas para los LEDs: dos juegos de semáforos (Semaforo 1 y 2) gestionados por el Núcleo 1 (pines 16 al 22), y un semáforo individual (Semaforo 3) para el Núcleo 0 (pines 10 al 12). Al inicio, todos los LEDs de los semáforos 1 y 2 se apagan para asegurar un estado inicial conocido.

Funciones de Ciclo del Núcleo 1 (Modo Normal y Nocturno):

ciclo_doble_normal(): Implementa un ciclo de semáforo cruzado estándar. Este ciclo alterna el paso: primero el Semaforo 1 está en verde (5 segundos) mientras el Semaforo 2 está en rojo, luego el Semaforo 1 pasa a amarillo (2 segundos). Posteriormente, el Semaforo 1 pasa a rojo y el Semaforo 2 pasa a verde, repitiendo la secuencia en la otra dirección.

modo_nocturno(): Simula el modo nocturno al hacer parpadear rápidamente los LEDs rojo del Semaforo 1 y amarillo del Semaforo 2 (70 ms encendido, 70 ms apagado). Esta es una señal de precaución constante.

La función semaforo_noct_thread() es la rutina de trabajo del Núcleo 1. Esta función contiene el bucle infinito que gestiona la lógica principal de los semáforos 1 y 2. Si el pulsador está presionado (boton.value() == 1), se ejecuta la función modo_nocturno().Si el pulsador no está presionado, se ejecuta la función ciclo_doble_normal().

Esta función es lanzada al segundo núcleo mediante _thread.start_new_thread(semaforo_noct_thread, ()), asegurando que el control de los semáforos 1 y 2 se ejecute en paralelo con el código principal.

El bucle while True principal se ejecuta en el Núcleo 0 y controla el Semaforo 3 (pines 10,11,12) de forma completamente autónoma. Este semáforo sigue un ciclo simple y continuo: Verde por 5 segundos, amarillo por 2 segundos y Rojo por 7 segundos.

La separación de tareas es clara: el Núcleo 1 se encarga de la lógica condicional y reactiva (pulsador → modo), mientras que el Núcleo 0 mantiene un ciclo fijo, garantizando que el funcionamiento del Semaforo 3 no se vea afectado por las demoras o el modo activo de los otros dos semáforos.

Codigo:

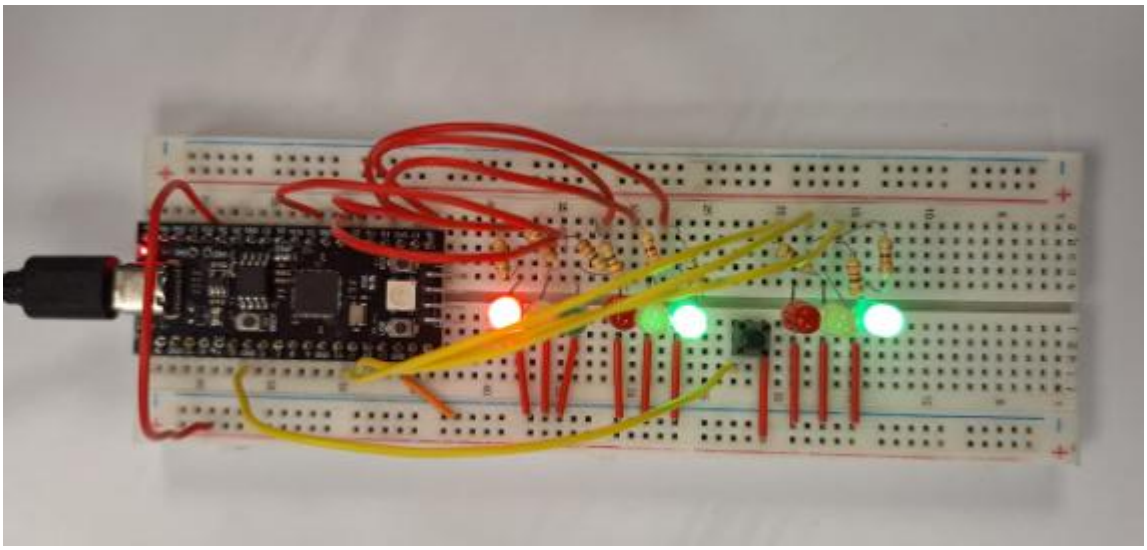
```
from machine import Pin
from time import sleep_ms
import _thread

boton = Pin(5, Pin.IN, Pin.PULL_DOWN)

# PRIMER NUCLEO SEMAFORO NOCTURNO
# SEMAFORO 1
n0_rojo1 = Pin(20, Pin.OUT)
n0_amarillo1 = Pin(21, Pin.OUT)
n0_verde1 = Pin(22, Pin.OUT)

# SEMAFORO 2
n0_rojo2 = Pin(16, Pin.OUT)
n0_amarillo2 = Pin(17, Pin.OUT)
n0_verde2 = Pin(18, Pin.OUT)

# SEGUNDO NUCLEO SEMAFORO NORMAL
n1_rojo = Pin(10, Pin.OUT)
n1_amarillo = Pin(11, Pin.OUT)
n1_verde = Pin(12, Pin.OUT)

n0_rojo1.value(0)
n0_amarillo1.value(0)
n0_verde1.value(0)
n0_rojo2.value(0)
n0_amarillo2.value(0)
n0_verde2.value(0)

def ciclo_doble_normal():
    # Estado: semáforo 1 en verde, semáforo 2 en rojo
    n0_verde1.value(1)
    n0_rojo2.value(1)
    sleep_ms(5000)

    # Estado: semáforo 1 en amarillo, semáforo 2 en rojo
    n0_verde1.value(0)
    n0_amarillo1.value(1)
    sleep_ms(2000)

    # S1 en rojo, S2 en verde
```

```
    n0_rojo2.value(0)
    n0_amarillo1.value(0)
    n0_rojo1.value(1)
    n0_verde2.value(1)
    sleep_ms(5000)

    # S1 en rojo, S2 en amarillo
    n0_verde2.value(0)
    n0_amarillo2.value(1)
    sleep_ms(2000)

    # Apagar rojo del S1 y amarillo del S2
    n0_rojo1.value(0)
    n0_amarillo2.value(0)

def modo_nocturno():
    n0_rojo1.value(1)
    n0_amarillo2.value(1)
    sleep_ms(70)
    n0_rojo1.value(0)
    n0_amarillo2.value(0)
    sleep_ms(70)

def semaforo_noct_thread():
    while True:
        if boton.value() == 1:
            modo_nocturno()
        else:
            ciclo_doble_normal()


# Iniciar hilo en segundo núcleo
_thread.start_new_thread(semaforo_noct_thread, ())

# Bucle principal en primer núcleo
while True:
    # Verde encendido
    n1_verde.on()
    n1_amarillo.off()
    n1_rojo.off()
    sleep_ms(5000)

    # Amarillo encendido
```

```
n1_verde.off()
n1_amarillo.on()
n1_rojo.off()
sleep_ms(2000)

# Rojo encendido
n1_verde.off()
n1_amarillo.off()
n1_rojo.on()
sleep_ms(7000)
```

Evidencia

https://github.com/luisangelmg11/Arquitectura-de-Computadoras/blob/main/1.6%20Practica%20Evidencia%20Video.mp4



Conclusión

La implementación validó la eficacia del modelo de programación dual-core en la RP2040. El Núcleo 1 gestionó con éxito la conmutación de estados de los semáforos 1 y 2 en respuesta al evento del pulsador. Paralelamente, el Núcleo 0 mantuvo la precisión y la ejecución ininterrumpida del ciclo del semáforo 3. Este enfoque de segregación de tareas es fundamental para construir sistemas donde la alta capacidad de respuesta a eventos externos (Núcleo 1) debe coexistir con la exactitud en la temporización de procesos de fondo (Núcleo 0).

Perfiles GitHub

https://github.com/luisangelmg11
https://github.com/HaydeesitaSJJH
https://github.com/alisonrueda
https://github.com/Yamixjk