Laboratorio 7

Escalonamiento de Tareas, Prioridades y Sincronización usando *Semaphores*

Objetivos

- Usar *semaphores* para sincronización de tareas.
- Observar efectos que pueden surgir al usar distintas prioridades en los hilos.
- Aprender sobre problemas y limitaciones de algunos algoritmos de escalonamiento

Procedimiento

En este laboratorio se implementará un semáforo de tránsito usando la Raspberry Pi. Usarán tres luces (LEDs) que representarán las señales para una dirección del tránsito, la otra dirección del tránsito, y una señal para peatones (vean la Figura 1). La luz encendida representa "luz verde" (carros o peatones pueden avanzar), y la luz apagada representa "luz roja" (carros o peatones deben detenerse). La luz peatonal sólo se activará si un botón (push button) se ha presionado previamente (el botón es para indicar el deseo de los peatones de cruzar la calle).

Para las luces se usarán tres puertos GPIO. Asegúrense de configurar dichos puertos como salidas. El botón será conectado a un cuarto puerto GPIO. Este puerto debe configurarse como entrada. Conecten el botón de tal forma que, al estar presionado, en el puerto se lea el valor de 1, y cuando no esté presionado, en el puerto se lea el valor de 0 (consideren un resistor de pull down). Pueden verificar que su conexión es correcta usando los comandos para una terminal de la herramienta gpio.

Nota: si el puerto no está configurado y la conexión del botón no está como se describe arriba, las funciones descritas abajo no funcionarán.

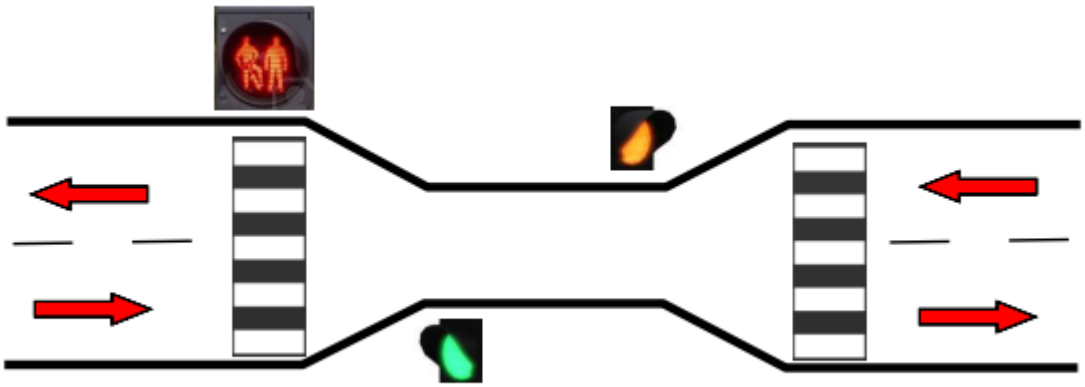

Figura 1. Esquema de las señales de tránsito y peatones.

Clases en C++

En su programa, para verificar si el botón peatonal ha sido presionado o no, deberán utilizar una clase llamada *"ButtonHandler"*. La cual se encuentra a continuación:

```
class ButtonHandler { //Se crea una clase para el control del botón
public:
    ButtonHandler() {
        pinMode(BTN1, INPUT);
        pullUpDnControl(BTN1, PUD_DOWN);
    }

    bool isButtonPressed() {
        return digitalRead(BTN1) == HIGH; // Devuelve true si el botón está
presionado
    }

    void waitForButtonRelease() {
        while (isButtonPressed()) {
            // Se espera a que el botón se suelte
        }
    }
};
```

El lenguaje de programación en C++ permite crear clases las cuales ofrecen soluciones más sencillas en cuanto a la programación orientada a objetos. Esto también ahorra el tener que utilizar librerías estáticas como se realizaba en C.

Parte 1: Escalonamiento por Poleo (*Polled Scheduling*)

En esta parte del laboratorio, crearán un programa con un hilo que actuará como planificador. El hilo deberá tener un ciclo infinito en el que se encienda la luz correspondiente a una dirección, luego se encienda la luz correspondiente a la otra dirección, y luego se revise el estatus del botón, para determinar si se debe encender o no la luz peatonal. En ningún momento debe haber más de una luz encendida. El tiempo en que permanezcan encendidas las luces puede ser de aproximadamente 1 segundo. Usen la función **std::this_thread::sleep_for()** (NO usen *timers*).

¿Qué limitaciones tiene este método?

Parte 2: Escalonamiento por Prioridades y Sincronización por *Semaphores*

En esta parte crearán un programa con tres hilos. Cada hilo será responsable de encender y apagar una de las luces. Para evitar que haya más de una luz encendida a la vez, deberán implementar un mecanismo de sincronización para los hilos. Esto se hará por medio de semáforos (*semaphores*). Deberán ajustar la prioridad y la política de escalonamiento de los hilos. El programa debe permitir ingresar como argumentos las prioridades de los tres hilos. Si se corre el

programa sin argumentos, las prioridades por defecto deben ser todas iguales a 1. Además de la implementación de los hilos deberán crear una clase que se encargue de controlar los LEDs es decir que tenga la capacidad de apagar y encender las luces.

El instructor discutirá la secuencia de instrucciones que los hilos deben ejecutar.

Deberán probar las siguientes configuraciones de prioridades de los hilos. Para cada caso, prueben de primero presionar el botón esporádicamente. Luego, prueben presionar el botón constante y rápidamente. Deberán reportar lo que observen.

Caso No.	Configuración de Prioridades	Observaciones
1	PL1 = PL2 = PLP	
2	PL1 = PL2 > PLP	
3	PL1 = PL2 < PLP	
4	PL1 > PL2 > PLP	
5	PL2 < PL1 < PLP	
6	PL1 < PL2 = PLP	
7	PL1 > PL2 = PLP	

Nota: PL1, PL2 y PLP son las prioridades de la luz 1, luz 2 y luz peatonal, respectivamente.

a) ¿Qué configuración(es) lleva(n) a un esquema de Round Robin?
b) ¿Hay configuración(es) que cause(n) que un hilo no llegue a ejecutar (*starvation*)?
c) ¿Se toparon con problemas al implementar sus programas? ¿Cómo resolvieron esos problemas?