UNIVERSIDAD NACIONAL DE COLOMBIA

Quinto proyecto
Semáforo

Alejandro Ramírez Jaramillo

Universidad Nacional de Colombia sede Manizales Email: alramirezja@unal.edu.co

Problema

La universidad realiza la instalación de un semáforo vehicular y peatonal. Lo contactan a usted para realizar la programación de dicho semáforo que deben contar con las siguientes condiciones:
El semáforo vehicular funcione con la secuencia que se puede observar en la figura 1.
El semáforo peatonal solo va ha estar en verde cuando el semáforo vehicular esté únicamente en estado 1 y el semáforo peatonal va a estar en rojo cuando el semáforo vehicular esté en los estados 2, 3 y 4.
La universidad dando prioridad a la seguridad de los peatones decide que se instale un botón donde el peatón una vez lo presione, el semáforo peatonal pasará a verde y el semáforo vehicular pasará a rojo, el botón solo funcionará cuando el semáforo vehicular esté en verde. El sistema debe tener un tic de 1 segundo.

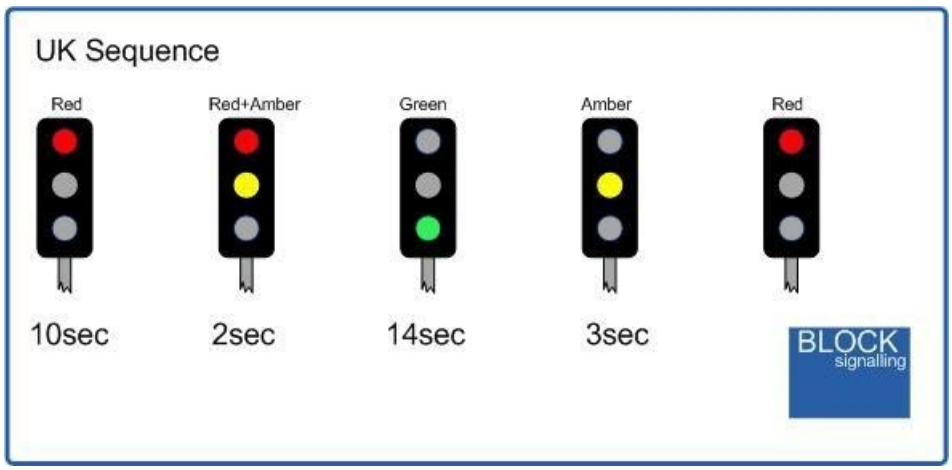


Figure 1. Secuencia Semáforo

1 Solución

1.1 Máquina de estados

Una máquina de estados nos permite entender un proceso desde sus posibles estados de manera secuencial y conectada. Para este ejercicio se preparó un ejemplo que abarca todo el sistema (semáforo vehicular y peatonal), en este se ven que luces de cada semáforo están encendidas en cada estado y cuales estados se ven afectados por entradas externas.

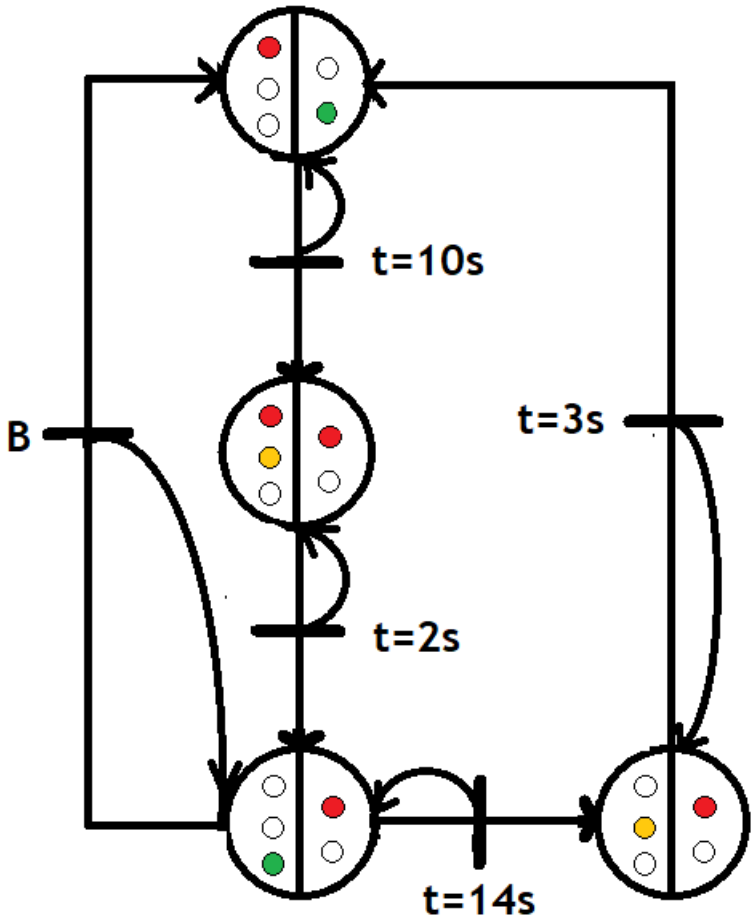


Figure 2. Máquina de estados Semáforo

El primer paso para el diseño de la máquina de estados es identificar la sucesión de estados a partir de un estado inicial determinado. Como trabajamos con un semáforo el cambio de estados ocurre cada cierto período de tiempo sin necesidad de una señal externa.
El semáforo de tres luces es el vehicular y el de dos luces es el peatonal. Como se puede apreciar en imagen de arriba, el sistema tiene 4 estados distintos y trabajan de forma secuencial: El primer estado corresponde a la luz roja del semáforo vehicular y verde del peatonal, entonces la secuencia normal sigue la dirección de las flechas.
Después de hacer esto procedemos a agregar los cambios en la secuencia causados por una señal externa (el botón). Analizando de nuevo el ejercicio encontramos lo siguiente: Si el semáforo vehicular esta en verde y el botón es presionado, entonces el semáforo peatonal pasará a verde y el vehicular pasará a rojo. Se identifica el estado que corresponde a la condición y el que corresponde

al nuevo estado al que se va a cambiar, estos se conectan con una flecha ligada a el botón (lo que representa que esa será la secuencia si B es)

1.2 Código

Para el ejercicio se usará: El timer, la interrupción externa y las GPIO. Hay tres secciones de código que darán forma a la maquina de estados:
En primer lugar el manejador del Timer, el cual esta configurado para ejecutarse cada 1 segundo (factor común de los tiempos entre cambios de estado) y aumentar en 1 el valor de un contador con el cual podremos llevar registro del tiempo. Para comenzar el código, se inicializan las variables y funciones que se van a usar:

```
int boton=1;
int con=0;
int t;
/**********************************************
* function declarations
**********************************************/
int main(void);
void GPIO_init(void);
void Ext_init(void);
void TIMER_init(void);
void semaforo(void);
/**********************************************
```

- botón: Esta variable se pone en 0 cuando el botón es presionado y la luz roja del semáforo vehicular esta encendida. Se limpia con 1.
- lock: Bandera que indica cuando se debe leer el valor del contador actual. Al cambiar de estado es necesario conocer el tiempo transcurrido, para calcular cuando hacer el siguiente cambio de estado, esta bandera se pone en 0 cuando hay un cambio de estado, despues de leer por primera vez el contador en el nuevo estado se limpia con un uno.
- con: Base de tiempo del sistema, marca en segundos el tiempo que se ha estado ejecutando el código, se usará para determinar cuando debe cambiar el estado. Esta base de tiempo nunca debe ser alterada, para que pueda ser usada en distintas máquinas de estado.
- t: Variable que almacenará el tiempo actual en segundos al momento de hacer un cambio de estados, se comparará con la base de tiempo para saber cuando hacer un cambio de estado.
- GPIO_init(): Inicializa las entradas y salidas necesarias, entre ellas el pin 13 del puerto C que esta conectado al botón.
- Ext_init(): Inicializa la interrupción externa, para más información remitirse a la guía de interrupción externa.
- TIMER_init(): Inicializa el Timer que se va a usar como base de tiempo, contando los segundos y almacenando el tiempo en la variable con. Para más información remitirse a la guía de TIMER.
- semaforo(): Máquina de estados que cumplirá la función del semáforo.

1.2.1 Base de tiempo (TIM2_IRQHandler)

El TIMER aumentará en 1 el contador "con" cada segundo, esto se usará para conocer el tiempo de ejecución del programa con el cual se decidirá cuando cambiar de estados. Esta base de tiempo no debe ser alterada para poder usarla en múltiples maquinas de estado a la vez sin la necesidad de crear un nuevo contador para cada una.

```
void TIM2_IRQHandler(void)
{

    // clear interrupt status
    if (TIM2->DIER & 0x01) {
        if (TIM2->SR & 0x01) {
            TIM2->SR &= ~(1U << 0);
        }
    }
                con +=1;
}
```

1.2.2 Máquina de estados

Conociendo cuanto tiempo ha transcurrido se decidirá que luces encender, a través de un switch case que comparará el contador "con" con la variable "t", para conocer cuanto tiempo ha transcurrido desde que se cambió de estado. Cada estado representa una combinación de leds encendidos del semáforo y cada vez que se cambia el estado se lee el tiempo actual para saber cuando hacer el siguiente cambio de estado.

```
void semaforo(void){
        enum states {STATE0, STATE1, STATE2, STATE3} current_state;
        int lock=0;
        current_state = STATE0; //set the initial state

        while(1){
                switch(current_state){
                        case STATE0:
                                if (lock==0){
                                        t=con;
                                        lock=1;
                                        GPIOA->ODR &= 0x00;
                                        GPIOA->ODR |= (1 << 2);
                                }
                                if(con>=(t+10)){
                                                current_state = STATE1;
                                                lock=0;
                                }else{
                                                current_state = STATE0;

                                                }
                                break;
                        case STATE1:
                                if (lock==0){
                                        t=con;
                                        lock=1;
                                        GPIOA->ODR |= (1 << 1);
                                }
                                if(con>=(t+2)){
                                                current_state = STATE2;
                                                lock=0;
                                }else{
                                                current_state = STATE1;

                        }
                                break;
                        case STATE2:
                                if (lock==0){
                                        boton=1;
                                        t=con;
                                        lock=1;
                                        GPIOA->ODR &= 0x00;
                                        GPIOA->ODR |= (1 << 0);
                                }
                                if(con>=(t+14)){
                                                current_state = STATE3;
                                                lock=0;
                                }else if(boton==0){
                                        current_state = STATE3;
                                        boton=1;
                                        lock=0;
```

```
                        }else{
                                current_state = STATE2;

                }
                        break;
                case STATE3:
                        if (lock==0){
                                t=con;
                                lock=1;
                                GPIOA->ODR &= 0x00;
                                GPIOA->ODR |= (1 << 1);
                        }
                        if(con>=(t+3)){
                                current_state = STATE0;
                                lock=0;
                        }else{
                                current_state = STATE3;

                }
                        break;
                }// switch(current_state)
        } //while(true)


}
```

La máquina de estados esta encerrada en un ciclo while que hará que se ejecute permanentemente al llamar la función, con la excepción de las interrupciones del sistema.

1.2.3 Botón semáforo peatonal

A continuación se adicionará el botón que altera el estado de los semáforos, el cual sera el mismo botón que esta conectado a la tarjeta. Para que sea detectado en cualquier momento de la ejecución del código se diseño como una interrupción externa que al ser ejecutada revisa en que momento del tiempo se encuentra la secuencia para identificar si esta en el estado adecuado (determinado como una de las condiciones en el ejercicio), hacer la bandera botón igual a cero y que se haga el cambio de estado.

```
void EXTI15_10_IRQHandler(void)
{
        //Check if the interrupt came from exti13
        if(EXTI->PR1 & (1 <<13)) {
                        boton=0;
                // Clear pending bit
                EXTI->PR1 = 0x00002000;
        }
}
```

1.2.4 GPIO_init()

Se inicializan las GPIO, se usarán los primeros 3 pines del puerto A como una salida para el semáforo vehicular y el pin 13 del puerto C como una entrada para la interrupción externa.

```
void GPIO_init(void){
        RCC->AHB2ENR |= 0x00000005;
        // Enable GPIOA and GPIOC Peripheral Clock (bit 0 and 2 in AHB2ENR register)
        // Make GPIOA Pin5 as output pin (bits 1:0 in MODER register)
        GPIOA->MODER &= 0xABFFFFFF;             // Clear bits 11, 10 for P5
        GPIOA->MODER &= 0xFFFFF755;             // Write 01 to bits 11, 10 for P5
        GPIOA->ODR &=0x0000;
        // Make GPIOD Pin13 as input pin (bits 27:26 in MODER register)
        GPIOC->MODER &= 0xFFFFFFFF;             // Clear bits 27, 26 for P13
        GPIOC->MODER &= 0xF3FFFFFF;             // Write 00 to bits 27, 26 for P13
}
```

1.2.5 Ext_init()

Configura la interrupción externa, escogiendo el pin 13 del puerto C como la entrada de la señal de interrupción y haciendo que detecte los filos de subida.

```
void Ext_init(void){
    // enable SYSCFG clock
        RCC->APB2ENR |= 0x1;
        // Writing a 0b0010 to pin13 location ties PC13 to EXT4
        SYSCFG->EXTICR[3] |= 0x20; // Write 0002 to map PC13 to EXTI4
        // Choose either rising edge trigger (RTSR1) or falling edge trigger (FTSR1)
        EXTI->RTSR1 |= 0x2000;   // Enable rising edge trigger on EXTI4
        // Mask the used external interrupt numbers.
        EXTI->IMR1 |= 0x2000;    // Mask EXTI4
        // Set Priority for each interrupt request
        NVIC->IP[EXTI15_10_IRQn] = 0x10; // Priority level 1
        // enable EXT0 IRQ from NVIC
        NVIC_EnableIRQ(EXTI15_10_IRQn);
}
```

1.2.6 TIMER_init()

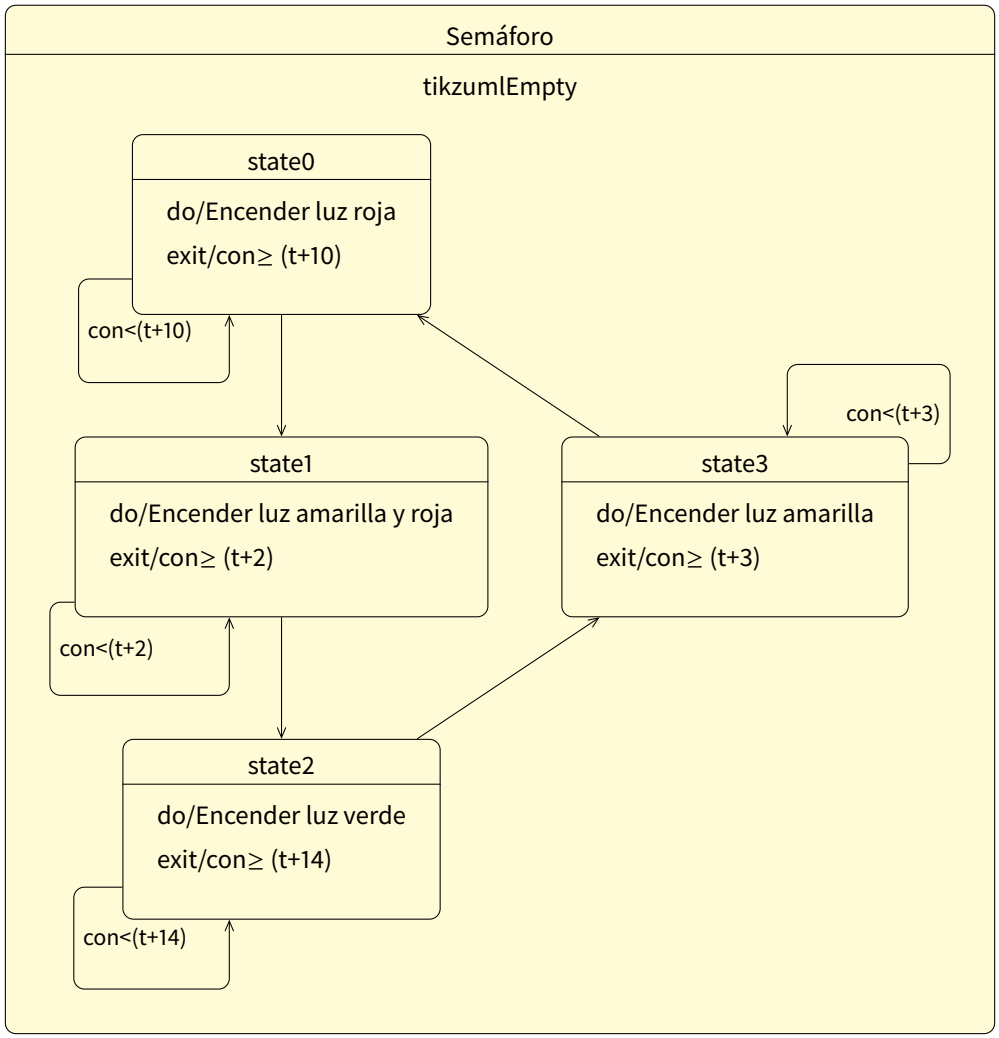
El Timer funcionará como una interrupción que se ejecutará cada 1 segundo, para esto se escoge el preescalador y el valor del tope (ARR). Este tiempo se almacenará en el contador "t".

```
void TIMER_init(void){
    // enable TIM2 clock (bit0)
    RCC->APB1ENR1 |= (1<<0);
    TIM2->PSC = 7999;
    TIM2->ARR = 400;
    // Update Interrupt Enable
    TIM2->DIER |= (1 << 0);
    NVIC_SetPriority(TIM2_IRQn, 2); // Priority level 2
    // enable TIM2 IRQ from NVIC
    NVIC_EnableIRQ(TIM2_IRQn);
    // Enable Timer 2 module (CEN, bit0)
    TIM2->CR1 |= (1 << 0);
}
```

1.2.7 Main

```
        // Se configura la interrupci n externa
        Ext_init();
        //Inicializamos el TIMER que marcara el tiempo de la secuencia
        TIMER_init();

        semaforo();
        while(1)
        {

        }

        __asm__("NOP"); // Assembly inline can be used if needed
        return 0;
}

void GPIO_init(void){
        RCC->AHB2ENR |= 0x00000005;
        // Enable GPIOA and GPIOC Peripheral Clock (bit 0 and 2 in AHB2ENR register)
```

2 Diagrama de estados (UML)



3 Referencias

+ PM0214 Programming manual, STM32 Cortex-M4 MCUs and MPUs programming manual.
+ RM0351 Reference manual, STM32L4x5 and STM32L4x6 advanced Arm-based 32-bit MCUs.
+ STM32l476rg datasheet.