Sexto proyecto
Elevador

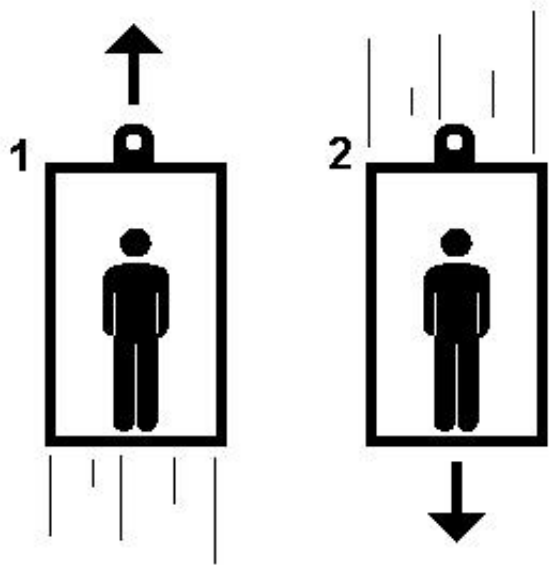
Alejandro Ramírez Jaramillo

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

Problema

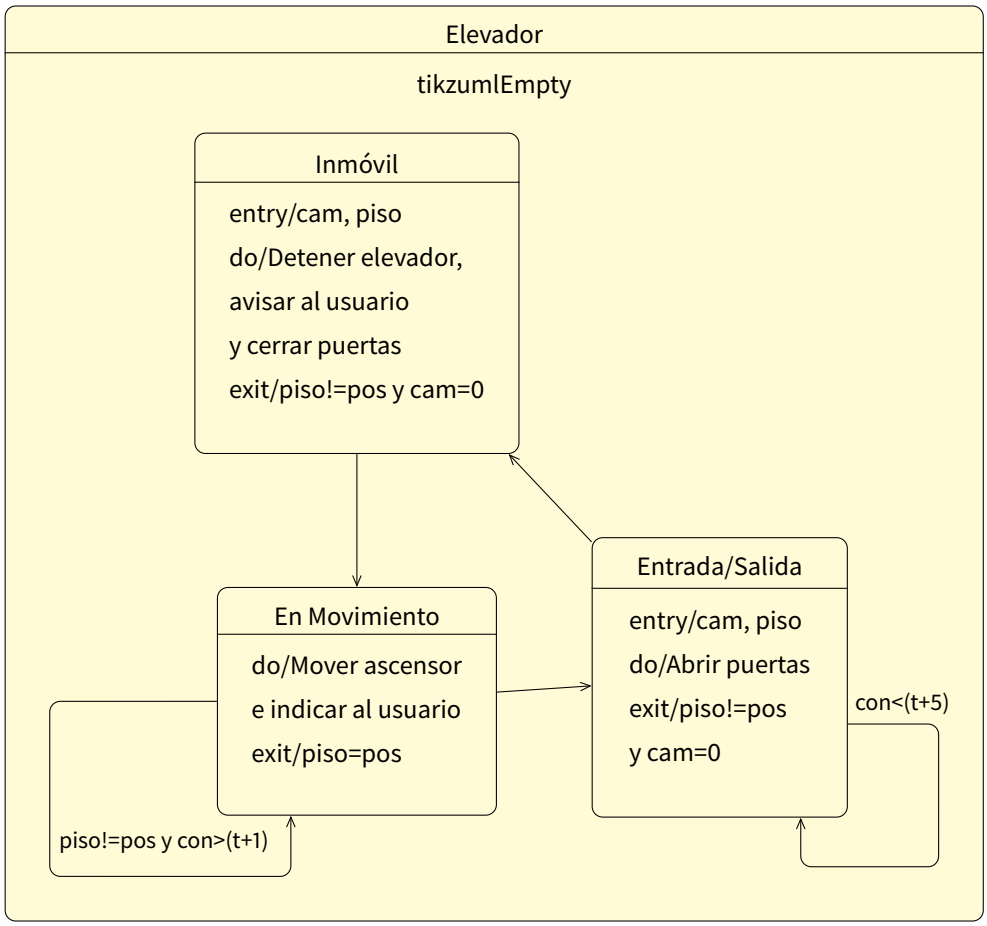
Un centro comercial requiere instalar un elevador para que sus clientes se muevan entre los tres pisos del mismo. Para indicar el piso al que debe llegar el elevador, el usuario debe presionar un botón tantas veces como el número del piso destino (por ejemplo si lo presiona 3 veces, irá al piso 3), una vez que elevador haya llegado a ese piso se quedará ahí hasta que sea llamado para ir a otro. Debido a la diferencia de altura entre ambos pisos, al elevador le toma aproximadamente 1 segundo moverse de un piso a otro (ya sea bajando o subiendo).

Tenga en cuenta que el movimiento del ascensor hace necesario que para ciertos casos este pase por otros pisos antes de llegar a su destino, así que si se encuentra en el piso 1 y debe llegar al piso 3, deberá pasar por el piso 2. Este comportamiento debe verse reflejado en la programación. Como una advertencia para los usuarios se prepara una señalización para indicar cuando el elevador esta en movimiento y cuando esta quieto (solo cuando este quieto se podrá ingresar un nuevo destino).



1 Solución

1.1 Máquina de estados



- Inmóvil: Estado en el que comienza el sistema, el elevador esta inmóvil en un piso y con las puertas cerradas, esperando a que el botón sea presionado: Si es para un piso diferente pasará al estado "En Movimiento" y si es para el mismo piso entonces pasará a "entrada salida".
- En Movimiento: El elevador se mueve de un piso a otro, ya sea subiendo o bajando. El elevador se moverá un piso por segundo hasta llegar a su destino, entonces pasará al estado "entrada salida".
- Entrada/Salida: En este estado el elevador ha llegado al piso destino y abre sus puertas por 5 segundos, después las cierra y pasa al estado "Inmóvil".

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 2 segundos (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.

1.2.1 Variables globales

```
int piso=0;

int pos=1;
int con=0;
int lock=1;
int cam=1;
/**********************************************
* function declarations
**********************************************/
int main(void);
void GPIO_init(void);
void Ext_init(void);
void TIMER_init(void);
void elevator(void);
```

- piso: Indica el piso destino, aumenta en uno cada vez que se presiona el botón.
- pos: Posición actual del elevador. Se ve reflejado en los primeros 3 pines del puerto A (el pin 0 corresponde al piso 1, el pin 1 al piso 2 y así sucesivamente).
- con: Base de tiempo del sistema, marca en segundos el tiempo que se ha estado ejecutando el código, se usará determinar cuando debe cambiar el estado. Esta base de tiempo nunca debe ser alterada, para que pueda ser usada en distintas máquinas de estado.
- 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 o mover el ascensor, esta bandera se pone en 0 cuando hay un cambio de estado, después de leer por primera vez el contador en el nuevo estado se limpia con un uno.
- cam: Esta variable se pone en 0 cuando el botón es presionado, se limpia con 1 cuando el ascensor ya se movió al piso deseado.
- t (se inicializa dentro de la función del elevador): 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 o mover el elevador un piso.
- 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.
- elevator(): Máquina de estados que cumplirá la función del elevador.

1.2.2 Base de tiempo (TIM2_IRQHandler)

El TIMER aumentará en 1 el contador "con" cada 1 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;
}
```

1.2.3 Botón del ascensor (EXTI15_10_IRQHandler)

Aumenta en uno la variable piso cada vez que se presiona el botón y se pone la bandera "cam" en 0 para indicar que se va a comenzar un nuevo desplazamiento.

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

1.2.4 Main

```
int main(void)
{
        //Se inicializa las GPIO que se van a usar (Puerto A y C)
        GPIO_init();
        // Se configura la interrupci n externa
        Ext_init();
        //Inicializamos el TIMER que marcara el tiempo de la secuencia
        TIMER_init();
        //Inicializamos la m quina de estados
        elevator();
        while(1)
        {

        }

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

1.2.5 GPIO_init()

Se inicializan las GPIO, se usarán los primeros 3 pines del puerto A como una salida para indicar el piso en el que se encuentra el elevador 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
}
```

1.2.6 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);
}
```

1.2.7 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);
}
```

1.2.8 Máquina de estados

Se leerá el número de veces que se presionó el botón para trasladar el elevador al piso correspondiente. En caso de que sea el mismo piso no se hará nada. El elevado comenzará a moverse un piso por segundo hasta llegar a su destino, entonces se pasará al estado de Entrada/Salida, abriendo las puertas del elevador por 5 segundos (los que se podrá apreciar en el bit 4 del Puerto A, 1 como puertas abiertas y cero como cerradas), pasado este tiempo las puertas se cerrarán y el elevador se quedará en el mismo piso esperando volver a ser llamado.

```
void elevator(void){
        //Estados: En Movimiento, Inmovil y entrada salida
        enum states {En_Movimiento, Inmovil,entrada_salida} current_state;
        int lock=0;
        int t;
        current_state = Inmovil; //set the initial state
        GPIOA->ODR |=(1<<0);
        while(1){
//En caso de que se oprima m s de tres veces el bot n,
//Se interpretara como tres veces.
                if (piso>3){
                        piso=3;
                }
                switch(current_state){
                        case En_Movimiento:
//Guardamos el tiempo en el que se entr  al estado
                                if (lock==0){
                                        t=con;
                                        lock=1;
                                        GPIOA->ODR &= 0x00;
                                }
//Se ejecutar  cada segundo
                                if (con>(t+1)){
//Si el piso destino esta arriba del actual
                                        if (pos<piso){
                                                pos+=1;
                                                GPIOA->ODR &= 0x0;
                                                GPIOA->ODR |=(1<<(pos-1));
                                                t+=1;
                                        }
                                        else{
//Si el piso destino esta abajo del actual
                                                if(pos>piso){
                                                        pos-=1;
                                                        GPIOA->ODR &= 0x0;
                                                        GPIOA->ODR |=(1<<(pos-1));
                                                        t+=1;
```

```
                                        }
//Si se alcanza el piso destino cambiar de estado
//y borrar las variables del b ton
                                        if(pos==piso){
                                                piso=0;
                                                current_state=entrada_salida;
                                                cam=1;
                                                lock=0;
                                        }

                                        }
                                }
                                break;
                        case Inmovil:
//Si se presiono el bot n y el piso destino
//es distinto del actual pasa al estado en movimiento
                                if((piso!=pos)&(cam==0)){
                                        current_state = En_Movimiento;
                                        lock=0;
                                }else {
//Si el bot n fue preionado pero el piso destino es igual
//  al actual, entonces pasa al estado entrada salida
                                        if((cam==0)&(piso==pos)){
                                        current_state = entrada_salida;
                                        lock=0;
                                }
                                }

                                break;
                        case entrada_salida:
//Enciende el led que indica que el ascensor ya no se esta moviendo
                                GPIOA->ODR|=0x20;
//Abre las puertas del ascensor y el tiempo en que lo hace
                                if (lock==0){
                                        t=con;
                                        lock=1;
                                        GPIOA->ODR |= 0x10;
                                        cam=1;
                                }
//5 segundos despues de abrir la puerta
                                if (con>(t+5)){
// pasa al estado inmovil
                                        current_state=Inmovil;
//Y cierra la puerta
                                        GPIOA->ODR &= ~(1<<4);

                                }
                                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. El puerto A se usa de la siguiente manera:

- Pin 0: Piso 1, cuando hay un uno en este pin significa que el ascensor esta en el piso 1.
- Pin 1: Piso 2, cuando hay un uno en este pin significa que el ascensor esta en el piso 2.
- Pin 2: Piso 3, cuando hay un uno en este pin significa que el ascensor esta en el piso 3.
- Pin 4: Indica si las puertas del elevador están abiertas (1) o cerradas (0).
- Pin 5: Indica si el elevador ya llegó al piso destino (1) o sigue en movimiento (0).

2 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.