UNIVERSIDAD NACIONAL DE COLOMBIA

Cuarto proyecto
PWM

Alejandro Ramírez Jaramillo

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

Introducción

PWM (Pulse width modulation) es un tipo de modulación que consiste en cambiar el ciclo útil de una onda cuadrada para controlar la energía que se va a entregar a una carga, también puede usarse para transmitir información, pero esta aplicación no es muy popular. La tarjeta STM32L476 permite generar un PWM internamente usando los TIMERS de la tarjeta, de los cuales podremos controlar características como la frecuencia y el ciclo útil usando los registros correspondientes y produciendo esta señal en los pines de la tarjeta.

Este ejemplo consiste en aumentar el ciclo útil del PWM cada vez que se presione el botón de la tarjeta hasta llegar a 100, entonces disminuirlo cada vez que se presione el mismo botón hasta que el ciclo útil se igual a cero y comenzar el proceso de nuevo. Este ejemplo fue diseñado para una tarjeta STM32L476 y se hará en el lenguaje de programación C.

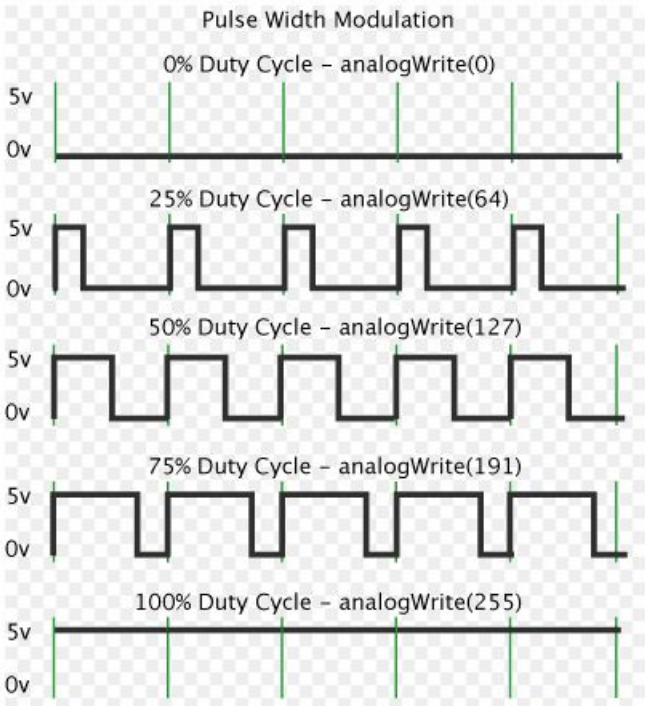

Figure 1. PWM con distintos valores de ciclo útil

Contenido

1 Registros

1.1 Inicialización de los relojes de los periféricos

1.1.1 RCC→AHB2ENR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG EN	HASHE N	AESEN (1)
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMIE N	ADCEN	OTGFS EN	Res.	Res.	Res.	GPIOIE N	GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 2. Distribución bits AHB2ENR, RM0351 Reference Manual, Pag. 251

Para el ejercicio el usuario usará el led 1 que se encuentra en la tarjeta, que está conectado al pin 5 del puerto A.

1.1.2 RCC→APB1ENR1

Registro que habilita o deshabilita el reloj para ciertos periféricos, con este registro se habilitara el reloj del TIMER 2 que se va a usar en el ejercicio.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 EN	OPAMP EN	DAC1 EN	PWR EN	Res.	CAN2 EN	CAN1 EN	CRSEN	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN(1)	USART3 EN	USART2 EN	Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Res.	Res.	WWD GEN	RTCA PBEN	LCD EN	Res.	Res.	Res.	TIM7 EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2 EN
rw	rw			rs	rw	rw				rw	rw	rw	rw	rw	rw

1. Available on STM32L45xxx and STM32L46xxx devices only.

Figure 3. Distribución bits APB1ENR1, RM0351 Reference Manual, Pag. 253

1.1.3 RCC→APB2ENR

Registro usado para activar el reloj de periféricos distintos a los puertos.En este ejercicio se hará uso de SYSCFG por lo cual se debe activar el reloj para este con el bit SYSCFGEN.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFSD M1 EN	Res.	SAI2 EN	SAI1 EN(1)	Res.	Res.	TIM 17EN	TIM16 EN	TIM15 EN
							rw		rw	rw			rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART 1 EN	TIM8 EN	SPI1 EN	TIM1 EN	SD MMC1 EN	Res.	Res.	FW EN	Res.	Res.	Res.	Res.	Res.	Res.	SYS CFGEN
	rw	rw	rw	rw	rw			rs							rw

1. Not available on STM32L41xxx/42xxx devices.

Figure 4. Distribución bits APB2ENR, RM0351 Reference Manual, Pag. 258

1.2 Configuración de entradas y salidas

1.2.1 GPIOx→MODER

Con este registro establezca el pin 5 del puerto A para que funcione como "Alternate function", para que pueda usarse como salida de PWM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 5. Distribución bits MODER, RM0351 Reference Manual, Pag. 303

1.2.2 GPIOx→AFR

GPIOx alternate function selection, se escoge la función dependiendo de TIMER usado. Este registro se divide en dos: Low y High, lo cuales dependen de los pines que se vayan a seleccionar (en este caso es el pin 5 por lo que se usa Low).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 6. Distribución bits AFR, RM0351 Reference Manual, Pag. 309

Bits 31:0 **AFSEL[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Figure 7. Distribución bits AFR, RM0351 Reference Manual, Pag. 310

Para saber cual es la función correspondiente para el TIMER 2 en el pin 5 se acude a la tabla de funciones alternativas que se encuentra en la datasheet de la tarjeta (en las referencias se encuentra el link para descargarla).

Table 17. Alternate function AF0 to AF7[1]

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
		SYS_AF	TIM1/TIM2/ TIM5/TIM8/ LPTIM1	TIM1/TIM2/ TIM3/TIM4/ TIM5	TIM8	I2C1/I2C2/I2C3	SPI1/SPI2	SPI3/DFSDM	USART1/ USART2/ USART3
Port A	PA0	-	TIM2_CH1	TIM5_CH1	TIM8_ETR	-	-	-	USART2_CTS
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_RTS_ DE
	PA2	-	TIM2_CH3	TIM5_CH3	-	-	-	-	USART2_TX
	PA3	-	TIM2_CH4	TIM5_CH4	-	-	-	-	USART2_RX
	PA4	-	-	-	-	-	SPI1_NSS	SPI3_NSS	USART2_CK
	PA5	-	TIM2_CH1	TIM2_ETR	TIM8_CH1N	-	SPI1_SCK	-	-
	PA6	-	TIM1_BKIN	TIM3_CH1	TIM8_BKIN	-	SPI1_MISO	-	USART3_CTS
	PA7	-	TIM1_CH1N	TIM3_CH2	TIM8_CH1N	-	SPI1_MOSI	-	-
	PA8	MCO	TIM1_CH1	-	-	-	-	-	USART1_CK

Figure 8. Funciones alternativas, Datasheet STM32L476xx, Pag. 92

1.3 Configuración del TIMER

1.3.1 TIMx→PSC

$$CK\_CNT = \frac{f_{CK\_PSC}}{PSC[15:0]+1}$$

Donde:

- f_{CK_PSC} es la frecuencia de entrada al prescaler (Clock Prescaler)
- CK_CNT es la frecuencia de salida del prescaler (Clock Counter)
- PSC[15:0] es el valor contenido en el registro PSC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 9. Distribución bits TIMx→PSC, RM0351 Reference Manual, Pag. 1074

1.3.2 TIMx→ARR

TIM x auto-reload register, almacena el valor hasta el cual va a contar el TIMER, funciona distinto si el contador trabaja de forma ascendente o descendente .

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 10. Distribución bits TIMx→ARR, RM0351 Reference Manual, Pag. 1074

1.3.3 TIMx→CR1

TIM control register 1, este registro se usará para habilitar el contador de el TIMER.

Figure 11. Distribución bits TIMx→CR1, RM0351 Reference Manual, Pag. 966

1.3.4 TIMx→CCR1

TIMx capture/compare register 1, si el canal esta configurado como salida el valor en este registro se cargara en el registro de captura/comparación (reemplazando el valor precargado). Este valor es el que se comparara con el valor en el contador TIMx_CNT y determinará el ciclo útil.

Figure 12. Distribución bits TIMx→CCR1, RM0351 Reference Manual, Pag. 988

1.3.5 TIMx→CCMR1

TIMx capture/compare mode register 1 [alternate], este registro se usará para configurar el modo de funcionamiento de la comparación de la salida (PWM) con los bits [6:4] y habilitar la precarga del registro TIMx_CCMR1 con el bit 3.

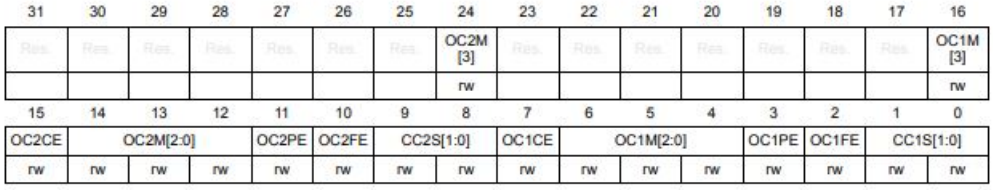

Figure 13. Distribución bits TIMx→CCMR1, RM0351 Reference Manual, Pag. 977

1.3.6 TIMx→CCER1

TIM1/TIM8 capture/compare enable register, se usa para habilitar un canal especifico de captura y comparación. Para el ejercicio se usa el canal CC1 como salida, el cual se habilita con el bit 0 de este registro.

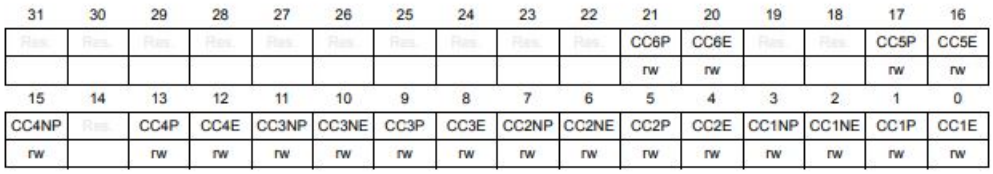

Figure 14. Distribución bits TIMx→CCER1, RM0351 Reference Manual, Pag. 983

1.3.7 SYSCFG→EXTICR[3]

External interrupt configuration register 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 15. Distribución bits SYSCFG_EXTICR4, RM0351 Reference Manual, Pag. 321

EXTIY (Y=15,14,13,12):
+0000: PA[Y] pin.
+0001: PB[Y] pin.
+0010: PC[Y] pin.
+0011: PD[Y] pin.
+0100: PE[Y] pin.
+0101: PF[Y] pin.
+0110: PG[Y] pin.

1.3.8 EXTI→IMR1

Interrup mask register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 16. Distribución bits EXTI_IMR1, RM0351 Reference Manual, Pag. 405

+ 0: Enmascarar la solicitud de interrupción.
+ 1: No enmascarar la solicitud de interrupción.

1.3.9 EXTI→RTSR1

Rising trigger selection register 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	RT18	Res.	RT16
									rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 17. Distribución bits EXTI_RTSR1, RM0351 Reference Manual, Pag. 405

+ Bits 31:23 y bit 17 reservados, no deben ser modificados.
+ Bits 22:18 y 16:0 pueden ser sobrescritos.
- 0: Solicitud de interrupción por flanco de subida deshabilitado.
- 1: Solicitud de interrupción por flanco de subida habilitado.

1.3.10 NVIC→IP[]

Interrupt priority, corresponde a los campos de prioridad que conforman NVIC→IP y se usa para establecer la prioridad de una interrupción.

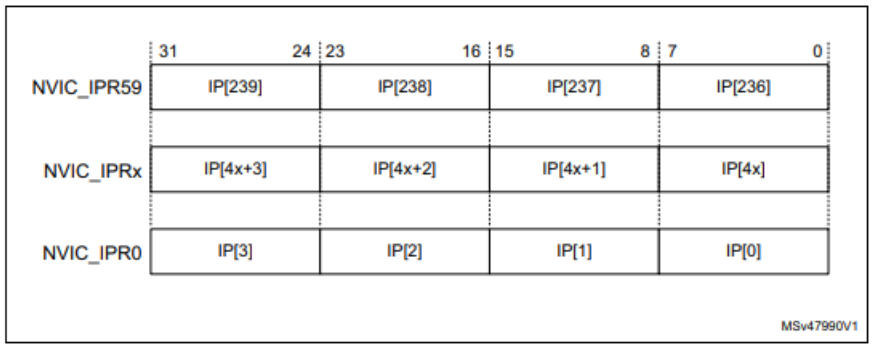

Figure 18. Distribución bits NVIC_IPR1, PM0214 Programming Manual, Pag. 215

1.3.11 NVIC_EnableIRQ(EXTI15_10_IRQn)

Habilita la interrupción externa, para que se ejecute una sección de código especifica cada vez que se oprima el botón.

2 PWM

Controlar la energía que se le entrega a una carga sirve para controlar su funcionamiento según sea necesario. Para ilustrar mejor la utilidad de un PWM tomemos como ejemplo un led cualquiera, los leds requieren de un cierto nivel de voltaje para alcanzar la corriente suficiente y funcionar, por lo tanto hay rangos de voltajes para los cuales el led no enciende lo que limita el control sobre el brillo .

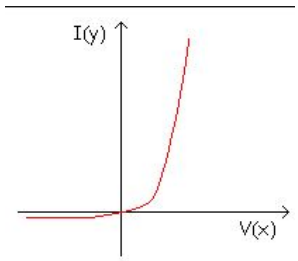

Figure 19. Curva característica diodo

La solución para este problema es alimentar el led con una señal de PWM cuyo voltaje máximo sea el que haga brillar todo lo posible en DC. Entonces para ajustar el brillo del led solo se necesita aumentar o disminuir el ciclo útil, esto es el porcentaje del período de la señal en que hay un voltaje distinto de cero.

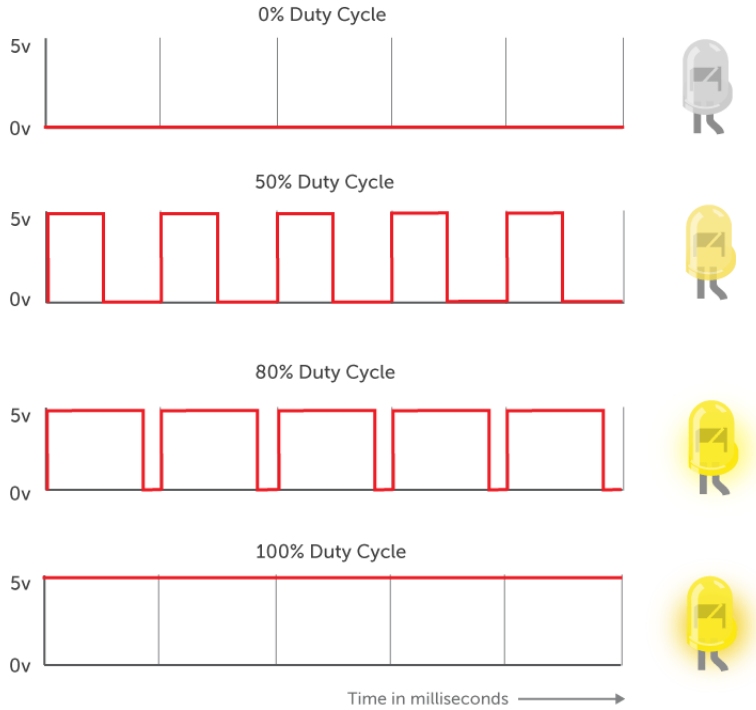

Figure 20. Brillo led para distintos valores de ciclo útil

Como se ve en la imagen anterior al aumentar el ciclo útil el led ilumina con

mayor intensidad, cuando el ciclo útil es del 100% el led llega a su máximo brillo y en 0% permanece apagado. Entonces ¿Por qué sucede esto? Los sistemas se resisten a los cambios repentinos, de forma que cuando una señal tiene una frecuencia muy alta el sistema no puede reaccionar inmediatamente a esta y lo hace de manera gradual.
Si la frecuencia del PWM fuera muy baja simplemente se vería el led parpadeando, cuando la frecuencia aumente el parpadeo se hará tan rápido que sera imperceptible y si aumenta aun más el led no alcanzara a apagarse antes de volver a recibir un voltaje que lo haga encenderse y así sucesivamente, lo que provoca que no brille al máximo ni este apagado, si no que tome un valor intermedio que dependerá de cuanto tiempo se le entrega energía y cuanto no.

3 Código PWM

El ejercicio consiste en aumentar el ciclo útil del PWM cada vez que se presione el botón de la tarjeta, cuando este llegue a 100 se cambiará la operación, en vez de aumentarlo, se disminuirá el ciclo útil del PWM hasta llegar a cero y se repetirá el proceso.

3.1 Inicialización

Se importan las librerías que se usarán y se definirá la variable PWMCOUNT y una variable booleana (verdadero o falso) que va a ser usadas después.

```
* Libraries include
**********************************************/
#include "stm32l476xx.h"
#include <stdbool.h>
/**********************************************
* definition variables
**********************************************/
#define PWMCOUNT 100

bool up = true;
```

3.2 Configuración GPIOs

Debido a que se hará uso del led que esta conectado a la tarjeta es necesario activar el reloj para el puerto A y configurar el pin 5 de este puerto (al cual esta conectado el led) con una función alternativa, además se usará el botón de la tarjeta entonces también se debe activar el reloj del puerto C y configurar el pin 13 del mismo como una entrada.

```
// enable GPIOA clock
RCC->AHB2ENR = 0x5;
// Make GPIOA Pin5 as Alternate pin (bits 1:0 in MODER register)
GPIOA->MODER &= 0xFFFFFBFF;
GPIOC->MODER &= 0xF3FFFFFF;
// Choose Timer2 as Alternative Function for pin 5 LED
GPIOA->AFR[0] |= (1 << 20);
```

3.3 Configuración TIMER 2

Habilitamos el reloj para el TIMER.

```
// enable TIM2 clock
RCC->APB1ENR1 |= (1 << 0);
```

Se establece el valor del prescalador y el contador (usando la variable PWM-COUNT), definiendo la frecuencia del PWM.

```
// fCK_PSC / (PSC[15:0] + 1)
// 4 MHz / n + 1 =  timer clock speed
```

```
TIM2->PSC = 0; //TIM Clock 4MHz

// set total count
TIM2->ARR = PWMCOUNT; // ARR = F_timer/F_PWM
```

Después se escoge el ciclo útil, se configura para que el canal 1 trabaje como PWM y se habilitan el canal.

```
// set duty cycle on channel 1
TIM2->CCR1 = 0;

// enable channel 1 in capture/compare register
// set oc1 mode as pwm (0b110 or 0x6 in bits 6-4)
TIM2->CCMR1 |= (0x6 << 4);
// enable oc1 preload bit 3
TIM2->CCMR1 |= (1 << 3);
// enable capture/compare ch1 output
TIM2->CCER |= (1 << 0);
```

Después de configurar el TIMER 2 y la interrupción externa han sido habilitados, se habilita el contador del TIMER lo que da comienzo al PWM.

```
// Enable Timer 2 module (CEN, bit0)
TIM2->CR1 |= (1 << 0);
```

3.4 Configuración Interrupción Externa

Para comenzar se debe activar el reloj de la interrupción (SYSCFG). Se debe tener en cuenta que el registro usado para activar este reloj es distinto del que se usó para activar el reloj de timer 2.

```
// enable SYSCFG clock
RCC->APB2ENR |= 0x1;
```

Después se selecciona el puerto y pin que accionara la interrupción, en este caso es el pin 13 del puerto C. También escogemos que la solicitud de interrupción se haga cuando en el respectivo pin se detecte un flanco de subida.

```
// 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
EXTI->RTSR1 |= 0x2000;    // Enable rising edge trigger on EXTI4
// Mask the used external interrupt numbers.
EXTI->IMR1 |= 0x2000;     // Mask EXTI4
```

Por último se le asigna una prioridad a la interrupción y se habilita.

```
// 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
EXTI->RTSR1 |= 0x2000;    // Enable rising edge trigger on EXTI4
// Mask the used external interrupt numbers.
EXTI->IMR1 |= 0x2000;     // Mask EXTI4
```

3.5 Manejador de la Interrupción Externa ($EXTI15_10_IRQHandler$)

En primer lugar se verifica que haya una solicitud de interrupción proveniente del pin 13 usando el registro PR1.

```
// Check if the interrupt came from exti0
if (EXTI->PR1 & (1 << 13)) {
```

Entonces, el ejercicio requiere hacer distintas operaciones en caso de que el ciclo útil llegue a 0 o a 100, para diferenciar ambas situaciones se usará la variable booleana "up". Cuando el ciclo útil llegue a 100 el valor de up sera 'falso' y si llega a 0 entonces será 'verdadero'.

```
if(TIM2->CCR1 == 100) {
        up = false;
} else if(TIM2->CCR1 == 0) {
        up = true;
}
```

Entonces se usa if/else para aumentar o disminuir el ciclo útil según el valor de "up".

```
if(up) {
        TIM2->CCR1 += 10;
} else {
        TIM2->CCR1 -= 10;
}
```

Para finalizar se limpia la bandera de la solicitud de interrupción.

```
// Clear pending bit
EXTI->PR1 = 0x00002000;
```

4 Ejercicios prácticos

1. Disminuya la frecuencia del PWM hasta unos pocos Hz y describa que sucede.

5 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.