

# Proyecto 2: Manejo de dimmer con PWM

## Alumnos:

- Garré, Sebastián Nehuen
- Mosler, Santiago Agustín
- Mozo, Demian Nehuel
- Trigo, Nicolás Daniel

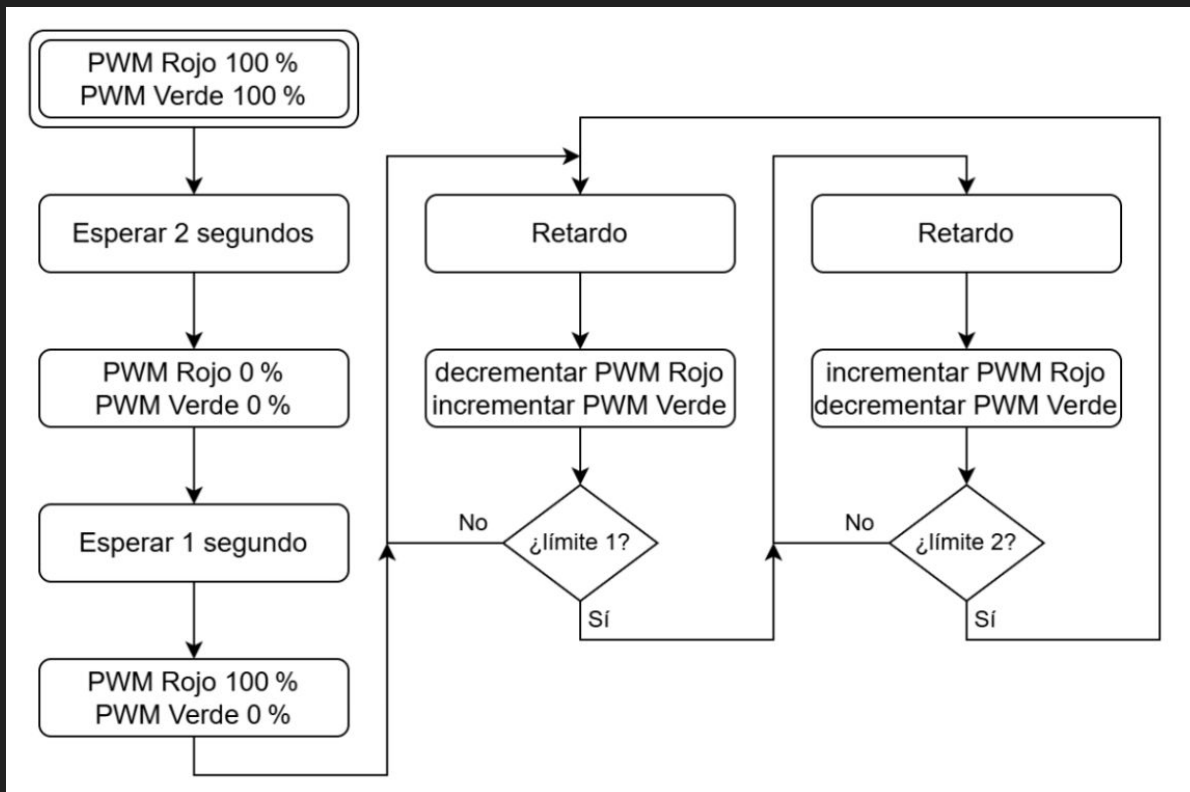
# Introducción

El objetivo es **aprender PWM (modulación por ancho de pulso) y manejo de retardos temporales** mediante un ejemplo práctico. Se requiere configurar jumpers, pines y programar el comportamiento deseado.

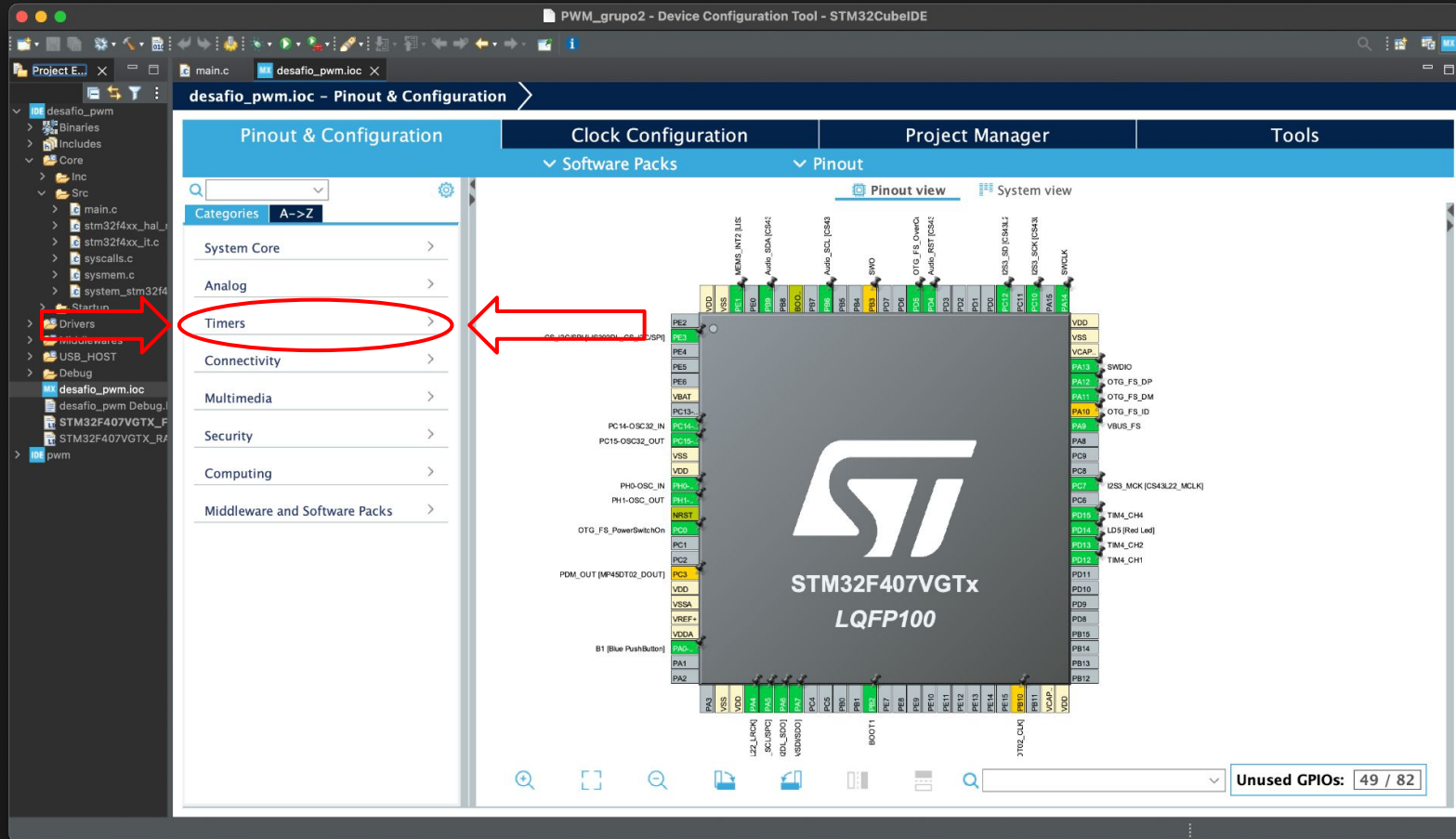
## Funcionamiento:

- 1) LEDs **rojo** y **verde** al 100% (2 seg).
- 2) Ambos apagados (1 seg).
- 3) LED **rojo** al 100%, **verde** apagado.
- 4) Transición:  
**Rojo** baja intensidad progresivamente .  
**Verde** sube intensidad progresivamente .
- 5) Se invierte el proceso.
- 6) El ciclo se repite indefinidamente.

# Diagrama de flujo



# Configuración del timer



# Configuración del timer

PWM\_grupo2 - Device Configuration Tool - STM32CubeIDE

Project Explorer: main.c, desafio\_pwm.ioc

desafio\_pwm.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout | System view

Categories: A~>Z

System Core

Analog

Timers

- RTC
- TIM1
- TIM2
- TIM3**
- TIM4**
- TIM5
- TIM6
- TIM7
- TIM8
- TIM9
- TIM10
- TIM11
- TIM12
- TIM13
- TIM14

Connectivity

Multimedia

Security

Computing

Middleware and Software Packs

Pinout view

STM32F407VGTx LQFP100

Unused GPIOs: 49 / 82

# Configuración del timer

PWM\_grupo2 - Device Configuration Tool - STM32CubeIDE

desafio\_pwm.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

TIM4 Mode and Configuration

Mode

Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock
Channel1	PWM Generation CH1
Channel2	PWM Generation CH2
Channel3	Disable
Channel4	PWM Generation CH4
Combined Channels	Disable
<input type="checkbox"/> Use ETR as Clearing Source	
<input type="checkbox"/> XOR activation	
<input type="checkbox"/> One Pulse Mode	

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	63999
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)

# Configuración del timer

The screenshot displays the STM32CubeMX Pinout & Configuration window for a project named 'desafio\_pwm.ioc'. The 'Pinout & Configuration' tab is active, showing the 'TIM4 Mode and Configuration' section. The 'Counter Settings' sub-section is highlighted with a red box, and its settings are detailed in the table below. The 'Counter Settings' table shows the following values: Prescaler (PSC - 16 bits value) set to 0, Counter Mode set to Up, Counter Period (AutoReload Register - 16 bits value) set to 63999, Internal Clock Division (CKD) set to No Division, and auto-reload preload set to Disable. The 'Trigger Output (TRGO) Parameters' section shows the Master/Slave Mode (MSM bit) set to Disable (Trigger input effect not delayed) and the Reset (UG bit from TIMx\_EGR) set to Reset (UG bit from TIMx\_EGR). The 'PWM Generation Channel 1' section shows the Mode set to PWM mode 1, Pulse (16 bits value) set to 0, Output compare preload set to Enable, Fast Mode set to Disable, and CH Polarity set to High. The 'PWM Generation Channel 2' section shows the Mode set to PWM mode 1, Pulse (16 bits value) set to 0, Output compare preload set to Enable, Fast Mode set to Disable, and CH Polarity set to High. The 'PWM Generation Channel 4' section shows the Mode set to PWM mode 1, Pulse (16 bits value) set to 0, Output compare preload set to Enable, Fast Mode set to Disable, and CH Polarity set to High. The 'Counter Settings' table is as follows:

Parameter	Value
Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	63999
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

The 'Trigger Output (TRGO) Parameters' section shows the following settings:

Parameter	Value
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)

The 'PWM Generation Channel 1' section shows the following settings:

Parameter	Value
Mode	PWM mode 1
Pulse (16 bits value)	0
Output compare preload	Enable
Fast Mode	Disable
CH Polarity	High

The 'PWM Generation Channel 2' section shows the following settings:

Parameter	Value
Mode	PWM mode 1
Pulse (16 bits value)	0
Output compare preload	Enable
Fast Mode	Disable
CH Polarity	High

The 'PWM Generation Channel 4' section shows the following settings:

Parameter	Value
Mode	PWM mode 1
Pulse (16 bits value)	0
Output compare preload	Enable
Fast Mode	Disable
CH Polarity	High

## Definiciones en código (en `main.c`)

```
/* Private define -----*/  
  
#define TENCENDIDO 2000 // Tiempo en ms luces encendidas al inicio  
#define TAPAGADO 1000 // Tiempo en ms luces apagadas antes de transición  
#define TCAMBIO 20 // Retardo entre cambios de PWM  
#define VALCAMBIO 640 // Paso de cambio de ciclo de trabajo (1% de 64000)
```



# Loop principal

```
/* USER CODE BEGIN 2 */
// CONDICIONES INICIALES
inicio();
/* USER CODE END 2 */

/* Infinite loop */
while (1) {
    /* USER CODE BEGIN 3 */
    transicion1(); // gradualmente disminuye rojo, aumenta verde
    transicion2(); // gradualmente disminuye verde, aumenta rojo
    }
    /* USER CODE END 3 */
}
```

# Función inicio

```
void inicio(void) {  
    // Obtenemos el valor máximo de ciclo de trabajo de la configuración del hw  
    uint16_t ciclomaximo = TIM4->ARR; // Valor máximo de ciclo de trabajo  
    // Configuramos el PWM para que inicie con los LED encendidos  
    TIM4->CCR1 = ciclomaximo; // LED verde al 100 %  
    TIM4->CCR2 = ciclomaximo; // LED rojo al 100 %  
    // Arrancamos el PWM  
    HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_1); // Inicio de la modulación PWM, LED verde  
    HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2); // Inicio de la modulación PWM, LED rojo  
    // Mantenemos los LED encendidos por un tiempo  
    HAL_Delay(TENCENDIDO); // Retardo de TENCENDIDO milisegundos  
    // Apagamos los LED  
    TIM4->CCR1 = 0; // LED verde al 0 %  
    TIM4->CCR2 = 0; // LED rojo al 0 %  
    // Mantenemos los LED apagados por un tiempo  
    HAL_Delay(TAPAGADO); // Retardo de TAPAGADO milisegundos  
    // Encendemos el LED rojo  
    TIM4->CCR2 = ciclomaximo; // LED rojo al 100 %  
}
```

## Ejemplo función de transición

```
/*----- transición 1 -----*/
// Disminuye rojo, aumenta verde
void transicion1(void) {
    // Obtenemos valores de la configuración del hw
    uint16_t ciclomaximo = TIM4->ARR; // Valor máximo de ciclo de trabajo
    uint16_t cicloverde = TIM4->CCR1; // Valor actual de ciclo de trabajo en LED verde
    uint16_t ciclorojo = TIM4->CCR2; // Valor actual de ciclo de trabajo en LED rojo
    do {
        // Esperamos un tiempo antes de cambiar las intensidades de lso LED
        HAL_Delay(TCAMBIO); // Espera por TCAMBIO milisegndos
        // Disminuimos un poco la intensidad del LED rojo
        ciclorojo = (ciclorojo > VALCAMBIO) ? ciclorojo - VALCAMBIO
                                                : 0; // Decremento con saturación en 0 %
        TIM4->CCR2 = ciclorojo; // Actualiza el ciclo de trabajo en LED rojo
        // Aumentamos un poco la intensidad del LED verde
        cicloverde = (ciclomaximo - cicloverde > VALCAMBIO) ?
                                                                cicloverde + VALCAMBIO
                                                                : ciclomaximo; // Incremento con saturación en 100 %
        TIM4->CCR1 = cicloverde; // Actualiza el ciclo de trabajo en LED verde
        // Verificamos si repetimos o invertimos
    } while (ciclorojo > 0 && cicloverde < ciclomaximo); // Repetimos mientras no se alcance un límite
}
```

# Desafío

- 1) Inicia con LED **amarillo** al 100%
- 2) Aumenta **rojo** a 100% progresivamente
- 3) Disminuye **amarillo** a 0% progresivamente
- 4) Aumenta **verde** a 100% progresivamente
- 5) Disminuye **rojo** a 0% progresivamente
- 6) Aumenta **amarillo** a 100% progresivamente
- 7) Disminuye **verde** a 0% progresivamente
- 8) Repetir indefinidamente pasos 2–7.

# Loop principal

```
/* USER CODE BEGIN 2 */
// CONDICIONES INICIALES
inicio(); // (amarillo al 100%)
/* USER CODE END 2 */

/* Infinite loop */
while (1) {
    /* USER CODE BEGIN 3 */
    transicion1(); // Aumenta rojo a 100%
    transicion2(); // Disminuye amarillo a 0%
    transicion3(); // Aumenta verde a 100%
    transicion4(); // Disminuye rojo a 0%
    transicion5(); // Aumenta amarillo a 100%
    transicion6(); // Disminuye verde a 0%
    }
    /* USER CODE END 3 */
}
```

## Función `inicio()`

```
/*----- Inicio -----*/  
void inicio(void) {  
    // Obtenemos el valor máximo de ciclo de trabajo  
    uint16_t ciclomaximo = TIM4->ARR;  
  
    // Iniciamos con LED amarillo al 100%  
    TIM4->CCR4 = ciclomaximo; // LED amarillo al 100%  
  
    // Arrancamos los PWM  
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1); // Verde  
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2); // Rojo  
    HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4); // Amarillo  
}
```

# Ejemplo función de transición

```
/*----- Transición 1: Aumenta rojo a 100% -----*/  
void transicion1(void) {  
    uint16_t ciclomaximo = TIM4->ARR;  
    uint16_t ciclorojo = TIM4->CCR2;  
  
    do {  
        HAL_Delay(TCAMBIO);  
        ciclorojo = (ciclomaximo - ciclorojo > VALCAMBIO) ? ciclorojo + VALCAMBIO  
                                                             : ciclomaximo;  
  
        TIM4->CCR2 = ciclorojo;  
    } while (ciclorojo < ciclomaximo);  
}
```