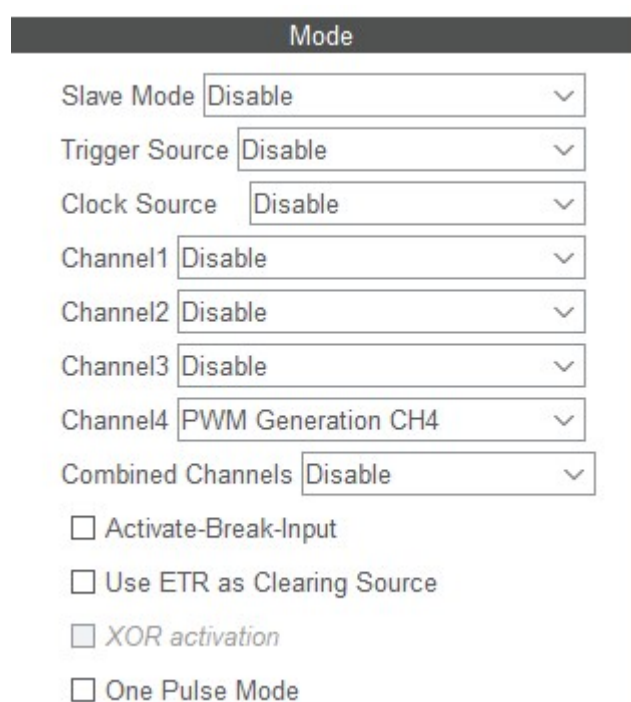


Configuración de PWM en STM32CubeIDE

En este manual se utilizará un PWM a partir de un timer para cambiar el brillo de un led. La STM32F401RE cuenta con 11 timers, 4 son de 16 bits y el resto son variables entre 16 y 32 bits. En este manual se configurará el timer 1 en el canal 4 como PWM.

Una vez que se crea un proyecto en STM32CubeIDE, se selecciona la tarjeta apropiada, y se llega a STM32CubeMX, hay que seleccionar el Timer 1, y habilitar el canal 4 como PWM Generation CH4.



The image shows the 'Mode' configuration window for a timer in STM32CubeMX. It contains several dropdown menus and checkboxes. The 'Channel4' dropdown is set to 'PWM Generation CH4', while all other dropdowns are set to 'Disable'. The checkboxes for 'Activate-Break-Input', 'Use ETR as Clearing Source', 'XOR activation', and 'One Pulse Mode' are all unchecked.

Parameter	Value
Slave Mode	Disable
Trigger Source	Disable
Clock Source	Disable
Channel1	Disable
Channel2	Disable
Channel3	Disable
Channel4	PWM Generation CH4
Combined Channels	Disable
Activate-Break-Input	<input type="checkbox"/>
Use ETR as Clearing Source	<input type="checkbox"/>
XOR activation	<input type="checkbox"/>
One Pulse Mode	<input type="checkbox"/>

Figura 1: Configuración de Timer 1

Esto hará que el PWM sea por medio de hardware, es decir, se tiene una función que puede colocar el valor del ciclo de trabajo sin la necesidad de calcular el tiempo en alto y bajo para el valor, por lo que se debe asignar un GPIO del microcontrolador para hacer uso de el.

En la parte de configuración del timer, se puede ajustar la frecuencia y el ciclo de trabajo con los que operará el PWM.



Figura 2: configuración del timer 1

Aquí es necesario usar un par de formulas para calcular el prescaler y el periodo de cuenta. Estas son:

$$F_t = (F_c)/(P+1) \quad 2. T = (1/F_t)(C_p+1)$$

Dónde:

F_t = Frecuencia de operación del Timer

P = Prescaler

T = Tiempo

C_p = Periodo de Cuenta (Preiod Counter)

Proponiendo un Prescaler de 15 y una frecuencia de trabajo de 10KHz tenemos una frecuencia máxima de 1MHz y un Counter Period de 99, es decir, la cantidad de pasos que le tomará al PWM llegar al 100% de su ciclo de trabajo:

Ft = 1MHz 2. Cp = 99

Después de colocar los valores en su respectiva casilla, ya esta todo listo para generar el código y comenzar a hacer el programa, pasando a la vista C/C++ y al archivo main.c. Hay que buscar cerca de la línea 97 donde dice “USER CODE BEGIN 2”, y entonces inicializar el PWM con la función HAL_TIM_PWM_Start(), que requiere de dos parámetros: el timer a usar, y el canal del timer que controlará el GPIO.

```
97  /* USER CODE BEGIN 2 */
98  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
99  __HAL_TIM_SetCompare(&htim1, TIM_CHANNEL_4, 0);
100
101  /* USER CODE END 2 */
```

Figura 3: Parámetros de las funciones.

La función a cargo de controlar el GPIO que se mapeo por el microcontrolador es __HAL_TIM_SetCompare(); la cual lleva 3 parámetros: el timer que será usado, el canal y el numero de pasos. Esto último es el valor del Counter Period del Timer1, por lo que solo puede tomar valores entre 0 y 99, donde 99 es el 100% del ciclo de trabajo. Si quisiéramos hacer un ciclo a la mitad, deberíamos redondear a 51.

Para apreciar el efecto del PWM en el GPIO se colocan dos ciclos de repetición FOR. El primero irá incrementando el valor del ciclo de trabajo mientras que el segundo lo hará decrecer.

```
105  while (1)
106  {
107      /* USER CODE END WHILE */
108
109      /* USER CODE BEGIN 3 */
110      for(uint8_t duty=0; duty<99; duty++){
111          __HAL_TIM_SetCompare(&htim1, TIM_CHANNEL_4, duty);
112          HAL_Delay(20);
113      }
114      for(uint8_t duty=99; duty>0; duty--){
115          __HAL_TIM_SetCompare(&htim1, TIM_CHANNEL_4, duty);
116          HAL_Delay(20);
117      }
118      HAL_Delay(200);
119  }
120  /* USER CODE END 3 */
121 }
```

Figura 4: control del PWM

Una vez terminado, hay que compilar el código para comprobar que no haya errores, para entonces bajarlo a la tarjeta y entrar al modo de depuración para hacer pruebas.

Se utilizaron un LED verde, una resistencia de 330 ohms, y dos jumpers macho-hembra para realizar la prueba, donde la salida del PWM, en el pin PA11 se conectó al cátodo del LED, después el ánodo a la resistencia, y esta conectada al GDN de la tarjeta.

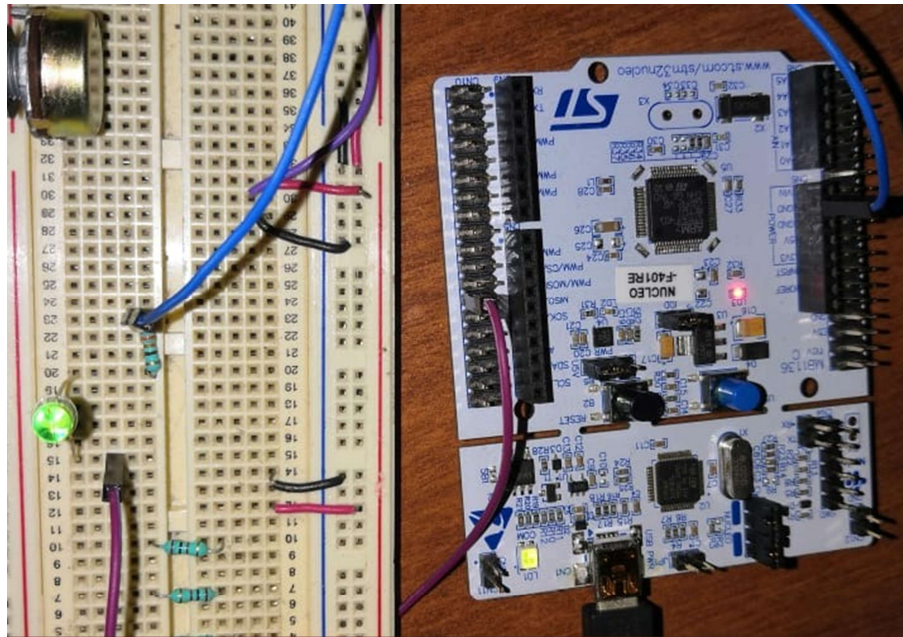


Figura 5: ejemplo de implementación.

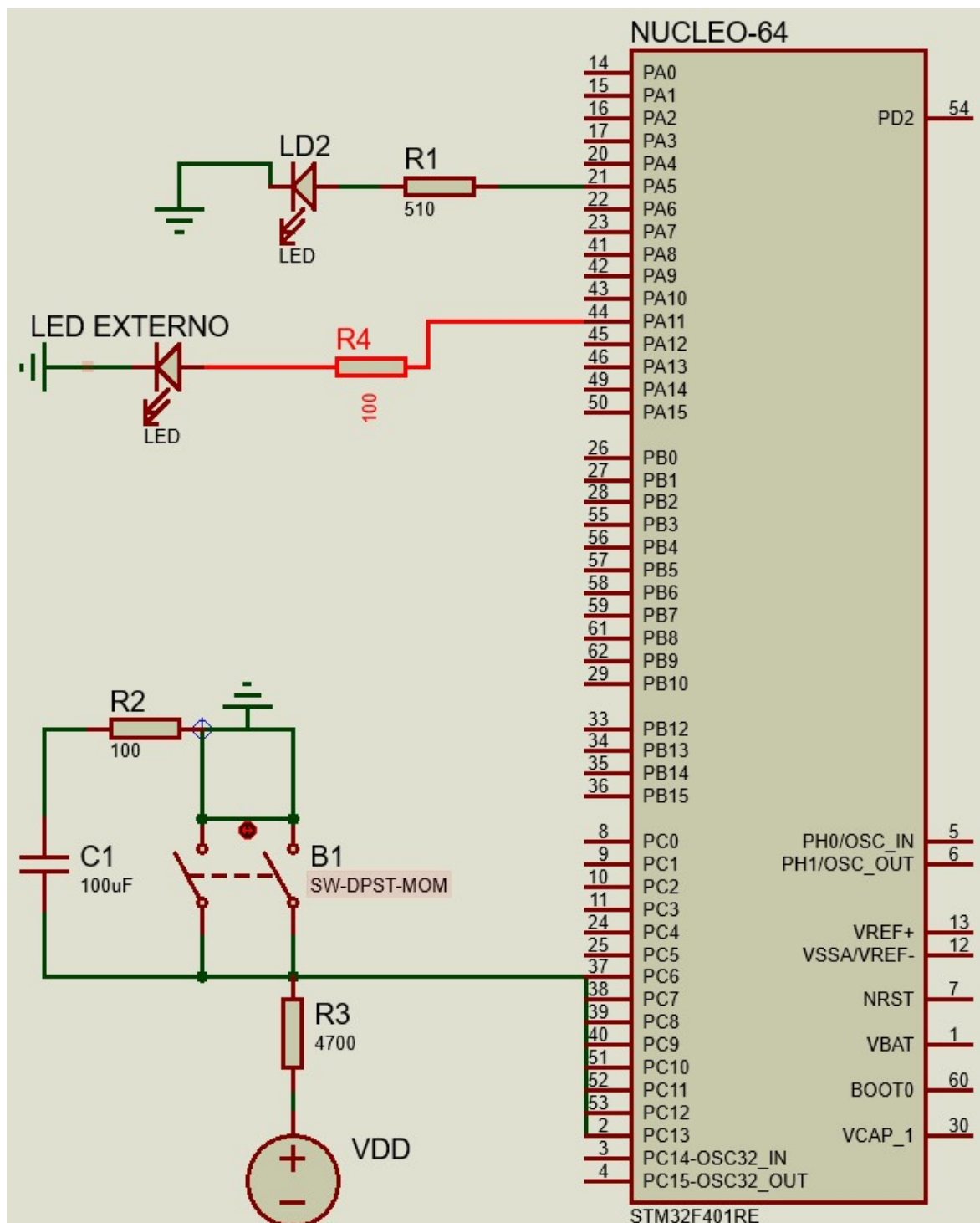
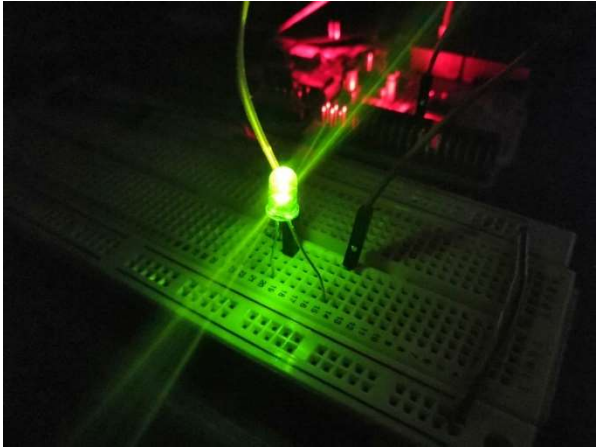
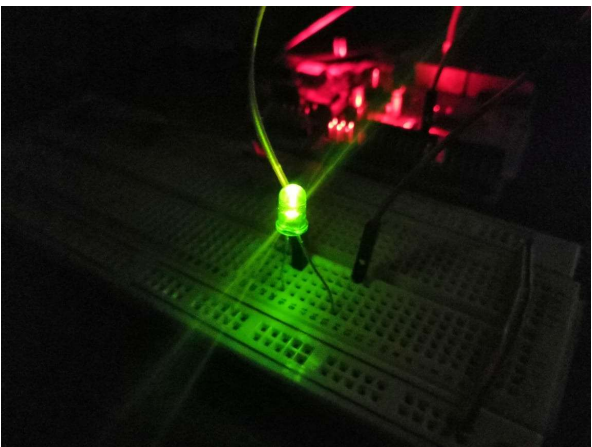
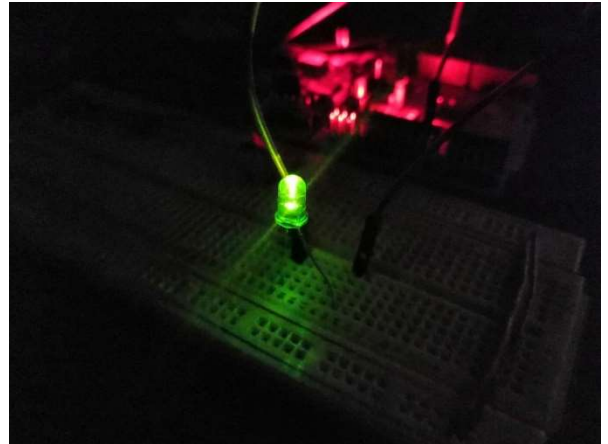
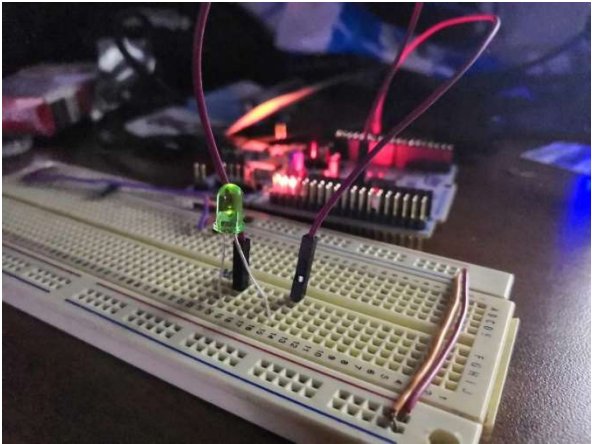


Figura 6: Esquemático de conexión



Figuras 7, 8, 9 y 10: apreciación del nivel de intensidad del LED