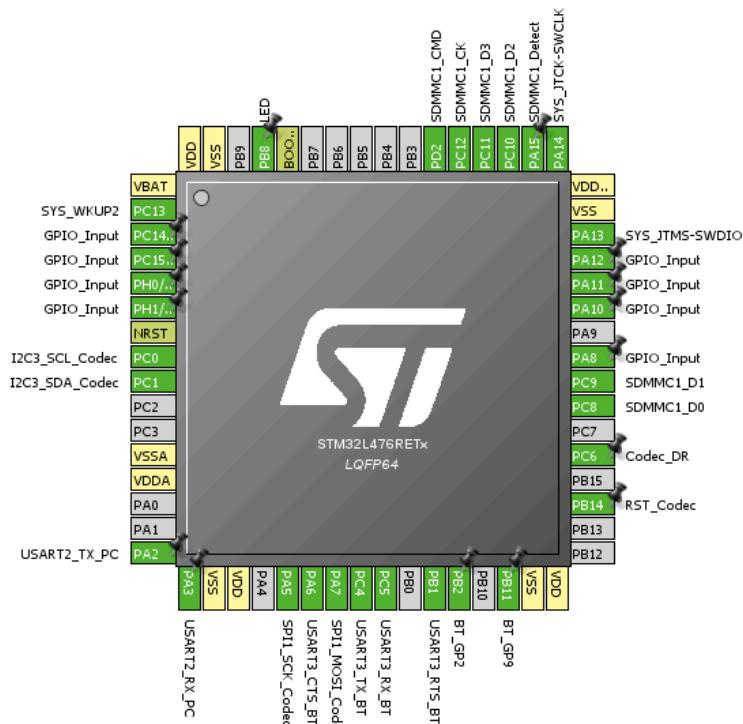


GPIO

Pines de entrada/salida STM32L476



El pin de entrada / salida de uso general del microcontrolador STM32 (GPIO) proporciona muchas formas de interactuar con circuitos externos dentro de un marco de aplicación.

En este documento daremos una breve explicación de su uso y configuración.

Glosario:

Esta sección define las principales siglas y abreviaturas utilizadas para la configuración de las entradas/salidas GPIO.

AMR: Puntaje máximo absoluto

GPIO: Entrada/salida de uso general

PP: Push Pull

PU: Pull Up

PD: Pull Down

OD: Drenaje abierto

AF funciona alternativa

VIH: El nivel mínimo de voltaje que se interpreta como un 1 lógico por una entrada digital
VIL: El nivel de voltaje máximo que se interpreta como un 0 lógico por una entrada digital
VOH: El nivel de voltaje mínimo garantizado que proporciona una salida digital establecida en el valor lógico 1

VOL: El nivel de voltaje máximo garantizado que proporciona una salida digital establecida en el valor lógico 0

VDD: fuente de alimentación externa para las I/Os

VDDIO2: fuente de alimentación externa para las I/Os, independiente de voltaje VDD
VDDA: fuente de alimentación externa para analógico

VSS: tierra

IIH: corriente de entrada cuando la entrada es 1

ILH: corriente de entrada cuando la entrada es 0

IOH: corriente de salida cuando la salida es 1

IOL: corriente de salida cuando la salida es 0

IIG: corriente de fuga

IINJ: corriente inyectada

Abreviación de los registros:

GPIOx_MODER:	Registro de modo de puerto GPIO
GPIOx_OTYPER:	Registro de tipo de salida GPIO
GPIOx_OSPEEDR:	Registro de velocidad de salida GPIO
GPIOx_PUPDR:	Registro pull-up/pull-down GPIO
GPIOx_IDR:	Registro de datos de entrada del puerto GPIO
GPIOx_ODR:	Registro de datos de salida del puerto GPIO
GPIOx_BSRR:	Registro Set / Reset GPIO
GPIOx_LCKR:	Registro de configuración de bloqueo GPIO
GPIOx_AFRL:	Registro bajo de función alternativa
GPIOx_AFRH:	Registro alto de función alternativa
GPIOx_ASCR:	Registro de control de interruptor analógico de puerto GPIO

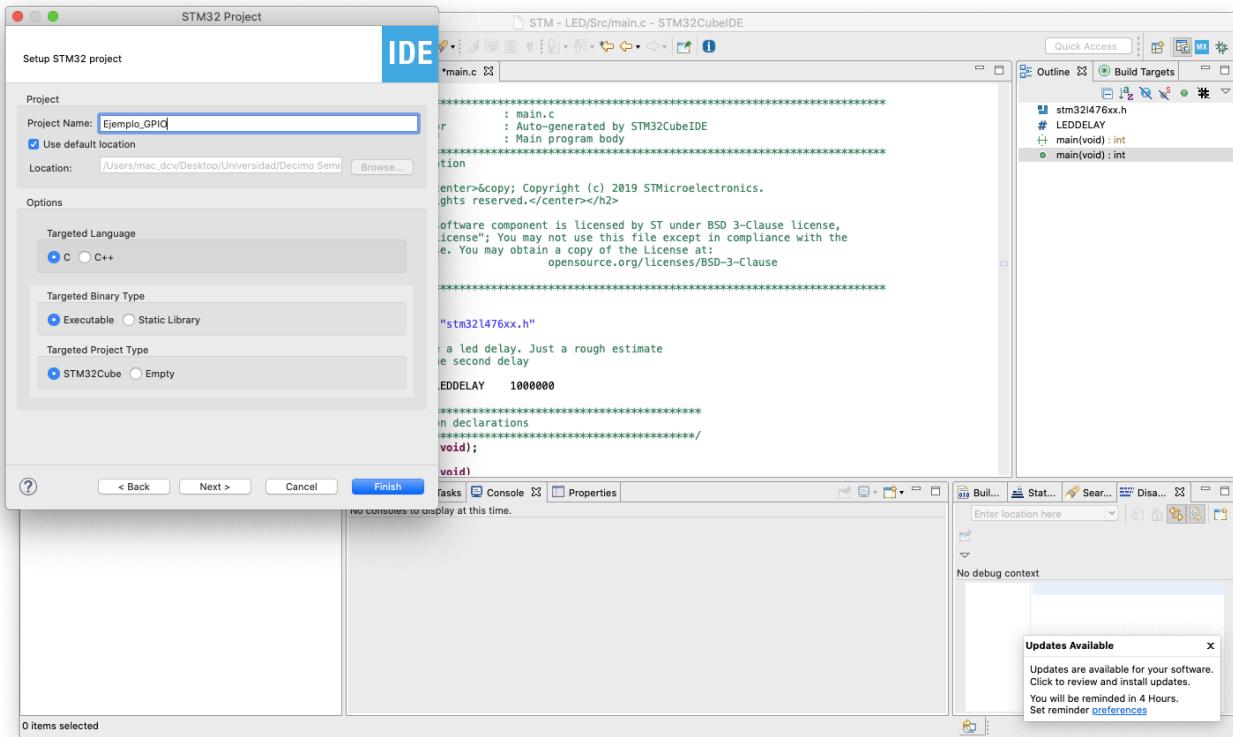
Pasos para configurar un GPIO:

- Seleccionar el número del pin de un que desea utilizar, este debe ser entre 0 y 15.
- Seleccionar el modo de operación; entrada, salida, analógico, interrupción, etc.
- Seleccionar la activación o desactivación de las resistencias de pull.
- Seleccionar la velocidad a la que trabajara dicho periférico.
- Especificar si el periférico está asociado a una función alternativa.

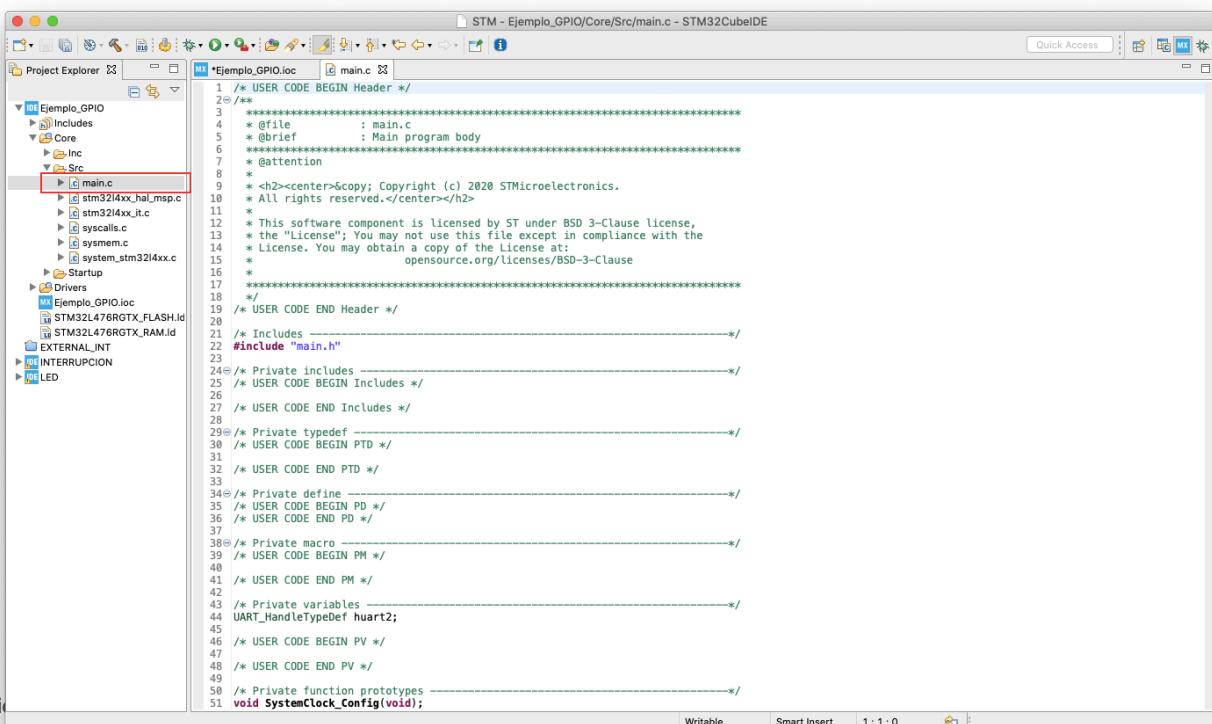
A continuación se muestra un ejemplo de configuración y uso de un puerto de una tarjeta SMT34L476 de la marca STM, con la ayuda del STM32CubeIDE.

Cabe aclarar que los pasos de instalación, configuración y preparación de la interfaz para la creación de un nuevo proyecto no será un tema a tratar en este documento. Se parte de la base de que estos conceptos ya están cubiertos.

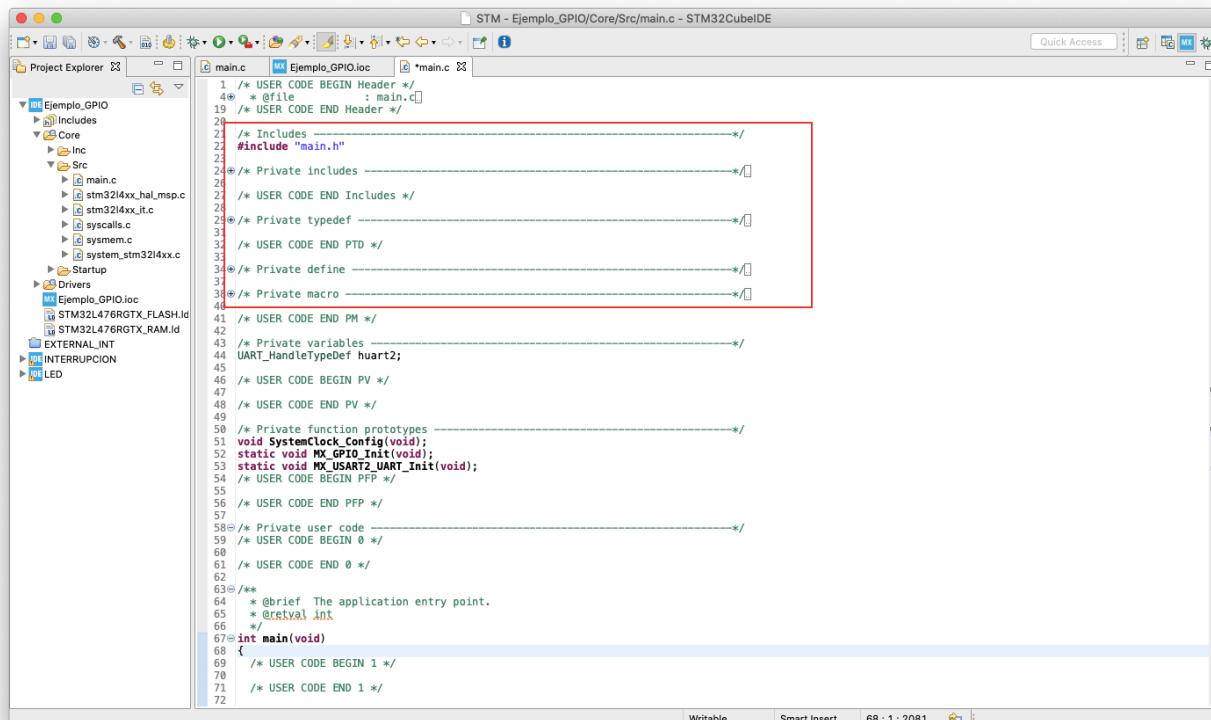
Siendo así partimos en el STM32CubeIDE desde la creación de un nuevo proyecto:



Luego de la creación limpia y nueva del proyecto, nos dirigimos a la carpeta Core/Src y abrimos el archivo main.c que es donde escribiremos todo el código.



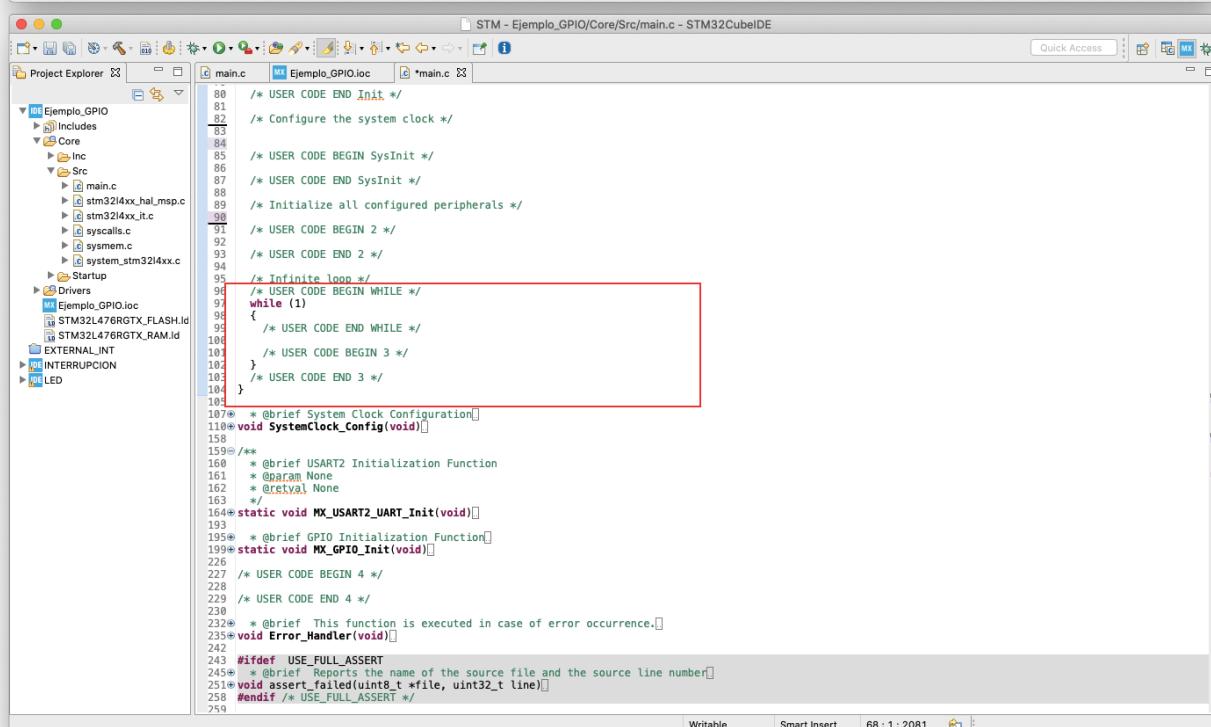
Inicialmente nos encontraremos con un código que la interfaz de desarrollo genera automáticamente con algunas funciones útiles pero que no usaremos en este ejemplo en particular, por lo que simplemente se pueden ignorar. En este punto nos fijaremos solo en dos partes claves del código, que serán la parte inicial donde declararemos las librerías a usar y nuestras variables y el ciclo while infinito que ejecutara el microcontrolador.



```

STM - Ejemplo_GPIO/Core/Src/main.c - STM32CubeIDE

main.c [ Ejemplo_GPIO.ioc ] *main.c [ ]
1  /* USER CODE BEGIN Header */
2  * @file   main.c
3  /* USER CODE END Header */
4
5  /* Includes _____*/
6  #include "main.h"
7
8  /* Private includes _____*/
9
10 /* USER CODE END Includes */
11
12 /* Private typedef _____*/
13
14 /* USER CODE END PTD */
15
16 /* Private define _____*/
17
18 /* USER CODE END Define */
19
20 /* Private macro _____*/
21
22 /* USER CODE END PM */
23
24 /* Private variables _____*/
25 UART_HandleTypeDef huart2;
26
27 /* USER CODE BEGIN PV */
28
29 /* USER CODE END PV */
30
31 /* Private function prototypes _____*/
32 void SystemClock_Config(void);
33 static void MX_GPIO_Init(void);
34 static void MX_USART2_UART_Init(void);
35
36 /* USER CODE BEGIN PFP */
37
38 /* USER CODE END PFP */
39
40 /* Private user code _____*/
41
42 /* USER CODE BEGIN 0 */
43
44 /* USER CODE END 0 */
45
46 /* USER CODE END */
47
48 /* _____@*/
49 /* @brief  The application entry point.
50 * @retval int
51 */
52 int main(void)
53 {
54     /* USER CODE BEGIN 1 */
55
56     /* USER CODE END 1 */
57
58     /* Infinite loop */
59
60     /* _____@*/
61     /* @brief System Clock Configuration
62     */
63     /* _____@*/
64     /* @brief USART2 Initialization Function
65     */
66     /* _____@*/
67     /* @brief GPIO Initialization Function
68     */
69     /* _____@*/
70     /* @brief Reports the name of the source file and the source line number
71     */
72 }
```



```

STM - Ejemplo_GPIO/Core/Src/main.c - STM32CubeIDE

main.c [ Ejemplo_GPIO.ioc ] *main.c [ ]
80 /* USER CODE END Init */
81
82 /* Configure the system clock */
83
84 /* USER CODE BEGIN SysInit */
85
86 /* USER CODE END SysInit */
87
88 /* Initialize all configured peripherals */
89
90 /* USER CODE BEGIN 2 */
91
92 /* USER CODE END 2 */
93
94 /* Infinite loop */
95
96 /* USER CODE BEGIN WHILE */
97 while (1)
98
99 /* USER CODE END WHILE */
100
101 /* USER CODE BEGIN 3 */
102
103 /* USER CODE END 3 */
104 }
105
106 /* @brief System Clock Configuration
107 */
108 void SystemClock_Config(void)
109
110
111 /* @brief USART2 Initialization Function
112 */
113 static void MX_USART2_UART_Init(void)
114
115
116 /* @brief GPIO Initialization Function
117 */
118 static void MX_GPIO_Init(void)
119
120
121 /* USER CODE BEGIN 4 */
122
123 /* @brief This function is executed in case of error occurrence.
124 */
125 void Error_Handler(void)
126
127
128 /* @brief USE_FULL_ASSERT
129 */
130 void assert_failed(uint8_t *file, uint32_t line)
131
132 /* _____@*/
133 #endif /* USE_FULL_ASSERT */
134 }
```

Todo lo demás para este ejemplo en particular lo podemos eliminar.

The screenshot shows the STM32CubeIDE interface. The Project Explorer on the left lists several projects and their components. The main.c file is open in the center editor window. The code in main.c is as follows:

```

1  /* USER CODE BEGIN Header */
2  #include "main.h"
3
4  /* USER CODE END Header */
5
6  /* Includes _____*/
7  #include "stm32l476xx.h"
8
9  /* Function declarations _____*/
10
11 int main(void)
12 {
13     /* USER CODE BEGIN 1 */
14
15     while (1)
16     {
17         /* USER CODE END WHILE */
18
19         /* USER CODE BEGIN 3 */
20
21     }
22     /* USER CODE END 3 */
23
24
25 }
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

```

The status bar at the bottom indicates the file is Writable, has Smart Insert enabled, and shows page 40 of 1185.

El propósito principal de este ejercicio será configurar un puerto GPIO en modo salida para encender un led que en este caso se encuentra ya incorporado en la tarjeta, pero el proceso para encender un led externo sería exactamente igual. La idea es que el led parpadee para poder ver que si esta funcionando correctamente nuestro código.

Para esto comenzaremos incluyendo la librería de la tarjeta y declarando un delay que nos servirá para hacer parpadear el led mas adelante

```

STM - Ejemplo_GPIO/Core/Src/main.c - STM32CubeIDE

Project Explorer [x] [x] [x]
main.c [x] Ejemplo_GPIO.ioc [x] *main.c [x]
Ejemplo_GPIO
  Includes
    Core
      Inc
        main.c
        stm32xxx_hal_msp.c
        stm324xx_it.c
        syscalls.c
        sysmem.c
        system_stm324xx.c
      Startup
        Ejemplo_GPIO.ioc
        STM32L476RGTX_FLASH.Id
        STM32L476RGTX_RAM.Id
      EXTERNAL_INTERRUPT
      LED
        Binaries
        Includes
          Src
            main.c
            desktop.ini
        Startup
        Debug
        desktop.ini
        LED_ON.launch
        LED.launch
        STM32L476RGTX_FLASH.Id
        STM32L476RGTX_RAM.Id

main.c
1 /* USER CODE BEGIN Header */
48 * @file           : main.c
19 /* USER CODE END Header */
20
21 /* Includes */
22 #include "stm32l476xx.h"
23
24 // create a led delay. Just a rough estimate
25 // for one second delay
26 #define LEDDELAY 1000000
27
28
29 * ****
30 * function declarations
31 ****
32 int main(void);
33
34 int main(void)
35 {
36     while (1)
37     {
38         /* USER CODE END WHILE */
39
40         /* USER CODE BEGIN 3 */
41     }
42     /* USER CODE END 3 */
43
44 }
45

```

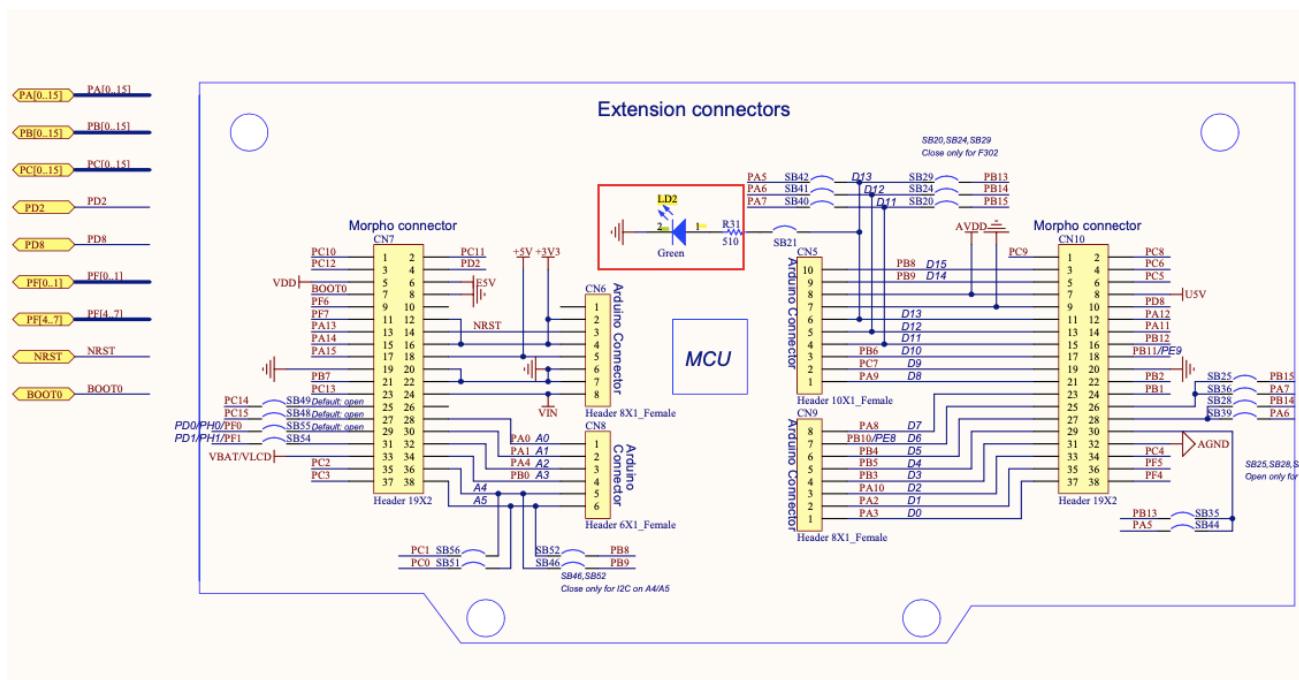
Ahora los pasos que debemos seguir se describen a continuación

Identificar desde el datasheet de la tarjeta en que pin esta ubicado el led interno que queremos encender

- Activar el reloj para dicho puerto.
- Configurar el pin x de GPIOx en modo salida.
- Colocar el pin x para encender el led
- Usar el delay
- Colocar el pin x en bajo para apagar el led

Nota: este ciclo se repetirá infinitas veces dado a que se programara dentro del ciclo while por defecto del programa

En este caso, observando el datasheet encontramos que el pin que deseamos configurar es el pin 5 del puerto A, que es donde se encuentra el led integrado de la tarjeta STM que vamos a usar



Sabiendo esto, pasamos a realizar los pasos anteriormente descritos

1. Activar el reloj del puerto GPIOA

```

STM - Ejemplo_GPIO/Core/Src/main.c - STM32CubeIDE

main.c
Ejemplo_GPIO.ioc
main.c

1 /* USER CODE BEGIN Header */
4 * @file   : main.c
19 /* USER CODE END Header */
20
21 /* Includes */
22 #include "stm32l476xx.h"
23
24 // create a led delay. Just a rough estimate
25 // for one second delay
26 #define LEDDELAY 1000000
27
28
29 //*****+-----+-----+-----+-----+-----+-----+-----+-----+
30 //+ function declarations
31 //*****+-----+-----+-----+-----+-----+-----+-----+-----+
32 int main(void);
33
34 int main(void)
35 {
36     // Enable GPIOA Peripheral Clock (bit 0 in AHB2ENR register)
37     RCC->AHB2ENR = 0x00000001;
38
39 }
40
41 while (1)
42 {
43     /* USER CODE END WHILE */
44
45     /* USER CODE BEGIN 3 */
46
47 }
48 /* USER CODE END 3 */
49
50

```

2. Configurar el pin 5 del GPIOA en modo salida

The screenshot shows the STM32CubeIDE interface with the project 'Ejemplo_GPIO' open. The Project Explorer on the left lists various source files and library components. The main editor window displays the 'main.c' file, which contains C code for a GPIO application. A red box highlights a specific section of code in the middle of the file:

```
1 /* USER CODE BEGIN Header */
2 #ifndef __MAIN_H
3 #define __MAIN_H
4
5 #endif /* End of header */
6
7 // Includes -----
8 #include "stm32l476xx.h"
9
10 // Create a led delay. Just a rough estimate
11 // for one second delay
12 #define LEDDELAY    1000000
13
14 // **** Create a led delay. Just a rough estimate
15 // for one second delay
16 // ****
17
18 int main(void);
19
20 int main(void)
21 {
22     // Enable GPIOA Peripheral Clock (bit 0 in AHB2ENR register)
23     RCC->AHB2ENR = 0x00000001;
24
25     // Make GPIOA Pin5 as output pin (bits 1:0 in MODER register)
26     GPIOA->MODER &= 0xFFFFF3FF;      // Clear bits 11, 10 for P5
27     GPIOA->MODER |= 0x00000400;    // Write 01 to bits 11, 10 for P5
28
29     while (1)
30     {
31         /* USER CODE END WHILE */
32
33         /* USER CODE BEGIN 3 */
34
35         /* USER CODE END 3 */
36
37 }
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 }
```

3. Colocar el pin 5 del puerto A en alto para encender el led

The screenshot shows the STM32CubeIDE interface with the following details:

- Project Explorer:** Shows the project structure with files like CODIGO_MORSE, Ejemplo_GPIO, and various source and header files.
- Code Editor:** Displays the main.c file containing C code for GPIO control. A red box highlights the line `GPIOA->ODR |= 0x0020;`. The code includes setting up the GPIOA Peripheral Clock and configuring Pin 5 as an output.
- Terminal:** Shows the build log:

```
terminated: Ejemplo_GPIO [STM32 Cortex-M C/C++ Application] ST-LINK (ST-LINK GDB server)
Time elapsed during download operation: 00:00:00.572
```
- Output:** Shows "Verifying ..." followed by "Download verified successfully".
- Messages:** A message box indicates "Debugger connection lost. Shutting down..."
- Right Panel:** Shows the Outline and Build Targets panes, with the main(void) function selected in the Build Targets pane.
- Bottom Right:** An "Updates Available" notification is shown, stating "Updates are available for your software. Click to review and install updates. You will be reminded in 4 Hours. Set reminder preferences".

Ahora dentro del ciclo while escribiremos las líneas que harán parpadear el led

4. Usar el delay

```

STM - Ejemplo_GPIO/Core/Src/main.c - STM32CubeIDE

Project Explorer
  CODIGO_MORSE
    Ejemplo_GPIO
      Binaries
      Includes
      Core
      Drivers
      Debug
        Ejemplo_GPIO.ioc
        Ejemplo_GPIO.launch
        STM32L476RGTX_FLASH.Id
        STM32L476RGTX_RAM.Id
  EXTERNAL_INT
  INTERRUPTION
  LED
    LED_BLINK
      Binaries
      Includes
      Src
        main.c
    Startup
    Debug
      LED_BLINK.Debug.launch
      LED_BLINK.launch
      STM32L476RGTX_FLASH.Id
      STM32L476RGTX_RAM.Id

Ejemplo_GPIO.ioc
main.c

27
28
29 //*****
30 * function declarations
31 *****/
32 int main(void);
33 void delay(volatile uint32_t s);
34
35 int main(void)
36 {
37     // Enable GPIOA Peripheral Clock (bit 0 in AHB2ENR register)
38     RCC->AHB2ENR = 0x00000001;
39
40     // Make GPIOA Pin5 as output pin (bits 1:0 in MODER register)
41     GPIOA->MODER &= 0xFFFF33FF; // Clear bits 11, 10 for P5
42     GPIOA->MODER |= 0x00000400; // Write 01 to bits 11, 10 for P5
43
44     // Set GPIOA Pin5 to 1 (bit 5 in ODR register)
45     GPIOA->ODR |= 0x0020; // write 1 to pin 5
46
47     while (1)
48     {
49         // Set a Delay
50         delay(LEDDELAY);
51
52         // Set GPIOA Pin5 to 0 (bit 5 in ODR register)
53         GPIOA->ODR ^= (1 << 5); // write 0 to pin 5
54     }
55
56 }
57
58 // A simple and not accurate delay function

```

Verifying ...

Download verified successfully

Debugger connection lost.
Shutting down...

5. Colocar el pin 5 del puerto A en bajo para apagar el led

```

STM - Ejemplo_GPIO/Core/Src/main.c - STM32CubeIDE

Project Explorer
  CODIGO_MORSE
    Ejemplo_GPIO
      Binaries
      Includes
      Core
      Drivers
      Debug
        Ejemplo_GPIO.ioc
        Ejemplo_GPIO.launch
        STM32L476RGTX_FLASH.Id
        STM32L476RGTX_RAM.Id
  EXTERNAL_INT
  INTERRUPTION
  LED
    LED_BLINK
      Binaries
      Includes
      Src
        main.c
    Startup
    Debug
      LED_BLINK.Debug.launch
      LED_BLINK.launch
      STM32L476RGTX_FLASH.Id
      STM32L476RGTX_RAM.Id

Ejemplo_GPIO.ioc
main.c

27
28
29 //*****
30 * function declarations
31 *****/
32 int main(void);
33 void delay(volatile uint32_t s);
34
35 int main(void)
36 {
37     // Enable GPIOA Peripheral Clock (bit 0 in AHB2ENR register)
38     RCC->AHB2ENR = 0x00000001;
39
40     // Make GPIOA Pin5 as output pin (bits 1:0 in MODER register)
41     GPIOA->MODER &= 0xFFFF33FF; // Clear bits 11, 10 for P5
42     GPIOA->MODER |= 0x00000400; // Write 01 to bits 11, 10 for P5
43
44     // Set GPIOA Pin5 to 1 (bit 5 in ODR register)
45     GPIOA->ODR |= 0x0020; // write 1 to pin 5
46
47     while (1)
48     {
49         // Set a Delay
50         delay(LEDDELAY);
51
52         // Set GPIOA Pin5 to 0 (bit 5 in ODR register)
53         GPIOA->ODR ^= (1 << 5); // write 0 to pin 5
54     }
55
56 }
57
58 // A simple and not accurate delay function

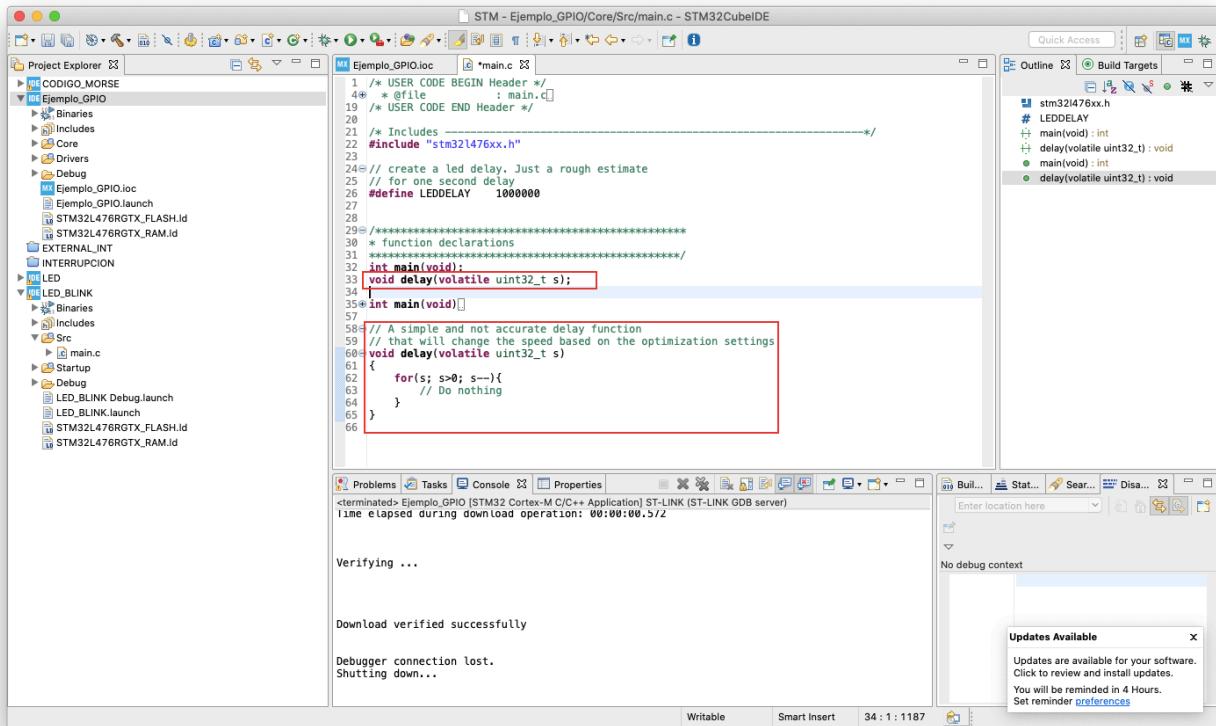
```

Verifying ...

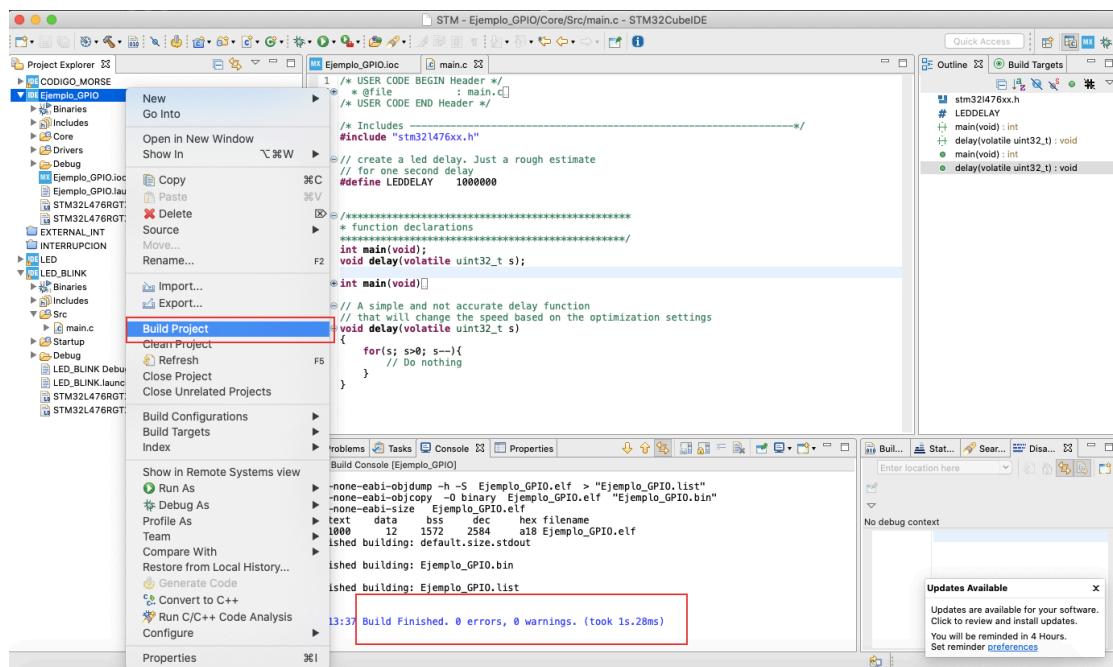
Download verified successfully

Debugger connection lost.
Shutting down...

Este ciclo se repetirá de manera infinita desde que carguemos y corramos el código en la tarjeta, sin embargo es importante declarar un par de cosas adicionales para que el delay funcione correctamente.



Luego para cargar el código a la tarjeta y comprobar su funcionamiento primero debemos copiarlo, la compilación debe mostrar al final cero errores.



Luego de compilar y no tener errores podemos proceder a conectar la tarjeta al ordenador por USB y cargar el código en ella.

