Configuración de GPIO en STM32CubeIDE

En este manual se mostrará como configurar los pines de entrada y salida de la tarjeta de desarrollo STM32F401RETx, realizando un ejemplo que consiste en presionar el botón de usuario (azul) para que el LED de usuario (verde) prenda por un momento definido, y luego se apague.

Primeramente, se debe crear un proyecto en STM32CubeIDE, abriendo el software, dando clic en “Start new STM32project”, para después seleccionar la tarjeta STM32F401RETx, y así llegar a la vista de STM32CubeMX.

Figura 1: Vista de STM32CubeMX

Ahora, nos dirigimos a la datasheet para encontrar los pines del LED y el botón integrados en la tarjeta, como lo vemos en las imágenes 2 y 3.

User LD2: the green LED is a user LED connected to Arduino signal D13 corresponding to STM32 I/O PA5 (pin 21) or PB13 (pin 34) depending on the STM32 target. Refer to *Table 11* to *Table 23* when:

- the I/O is HIGH value, the LED is on
- the I/O is LOW, the LED is off

Figura 2: Posición del Pin del LED

B1 USER: the user button is connected to the I/O PC13 (pin 2) of the STM32 microcontroller.

Figura 3: Ubicación del Pin del botón azul de usuario

Ya que se conocen los pines, se procede a configurarlos en el software:

Primero se debe encontrar el Pin que corresponda a PC13, se hace clic en el y se selecciona “GPIO_Input” ya que es el pin del botón, y será usado como entrada. Para fácil ubicación, se le pone la etiqueta “Bazul”, por Botón Azul.

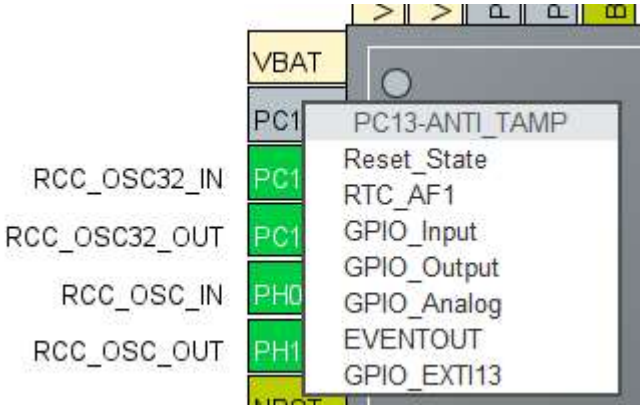

Figura 4: Opciones para el Pin

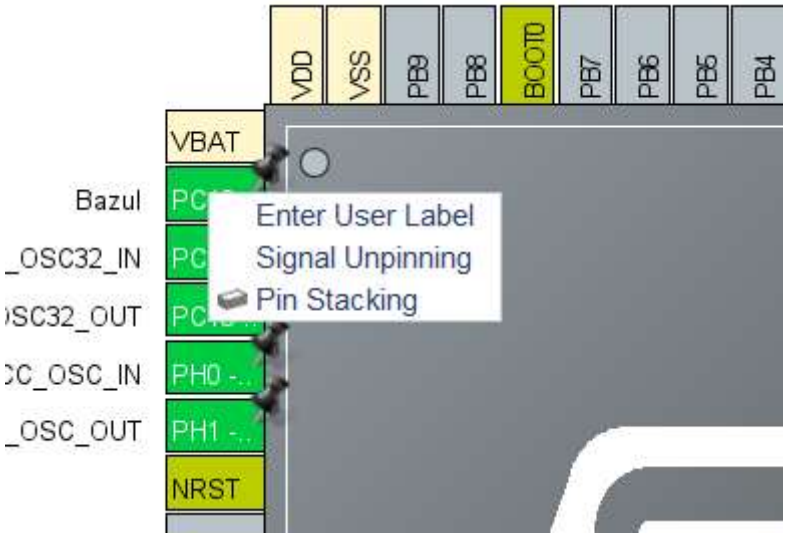

Figura 5: Etiquetando el Pin configurado como entrada

Ya que ha sido seleccionado el botón, se procede a elegir el pin digital de salida: el LED verde de usuario integrado en la tarjeta STM32F4. Para esto, hay que buscar el Pin PA5, que es el que está conectado al led de la tarjeta. Damos clic en el Pin, seleccionamos la opción “GPIO_Output” y le agregamos la etiqueta LDV, de led verde, de la misma forma que se hizo con el botón.

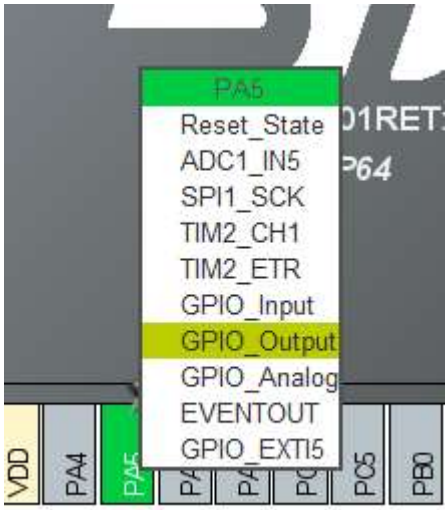

Figura 6: configurando el Pin PA5 como salida

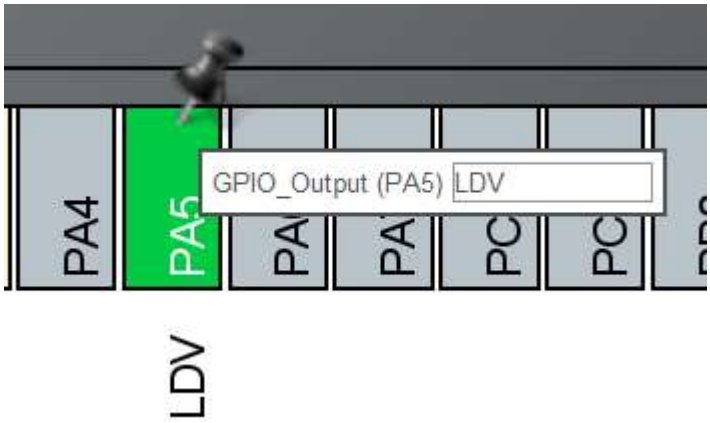

Figura 7: Nombrando el Pin del LED

Una vez asignados los Pins, se extiende el menú de “System Core” y se da clic a GPIO, lo cual mostrará los GPIOs configurados hasta el momento y sus opciones(Figura 8), donde se realizara la configuración correcta para cada uno de los pines utilizados, cambiando las resistencias Pull-up o Pull-down, definiendo su estado inicial, entre otras cosas.

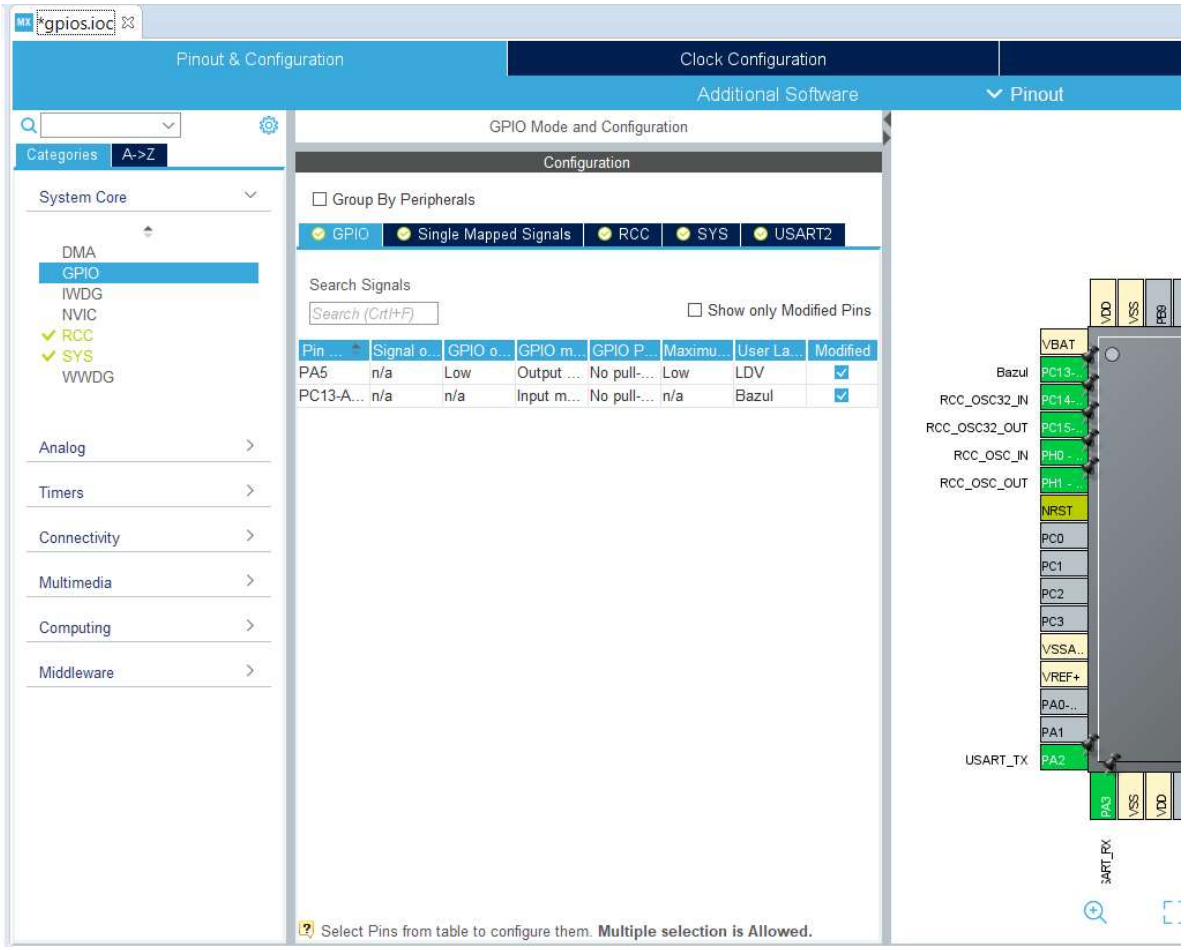

Figura 8: configuración de GPIO

Se selecciona PA5 y aparecerá un menú debajo con las configuraciones del botón, hay que asegurarse que (Figura 9):

- “GPIO mode” este en “Input mode”, ya que el botón funcionará como entrada.
- “GPIO Pull-up/Pull-down” se selecciona “Pull-up”, para habilitar la resistencia conectada del pin a Vcc, ya que el botón es conectado a GND cuando es presionado, y cuando no esta siendo presionado se mantiene en un estado flotante.
- “User Label”, por su parte, ya fue generado de forma automática debido a que se le asigno la etiqueta desde la pestaña de pinout.

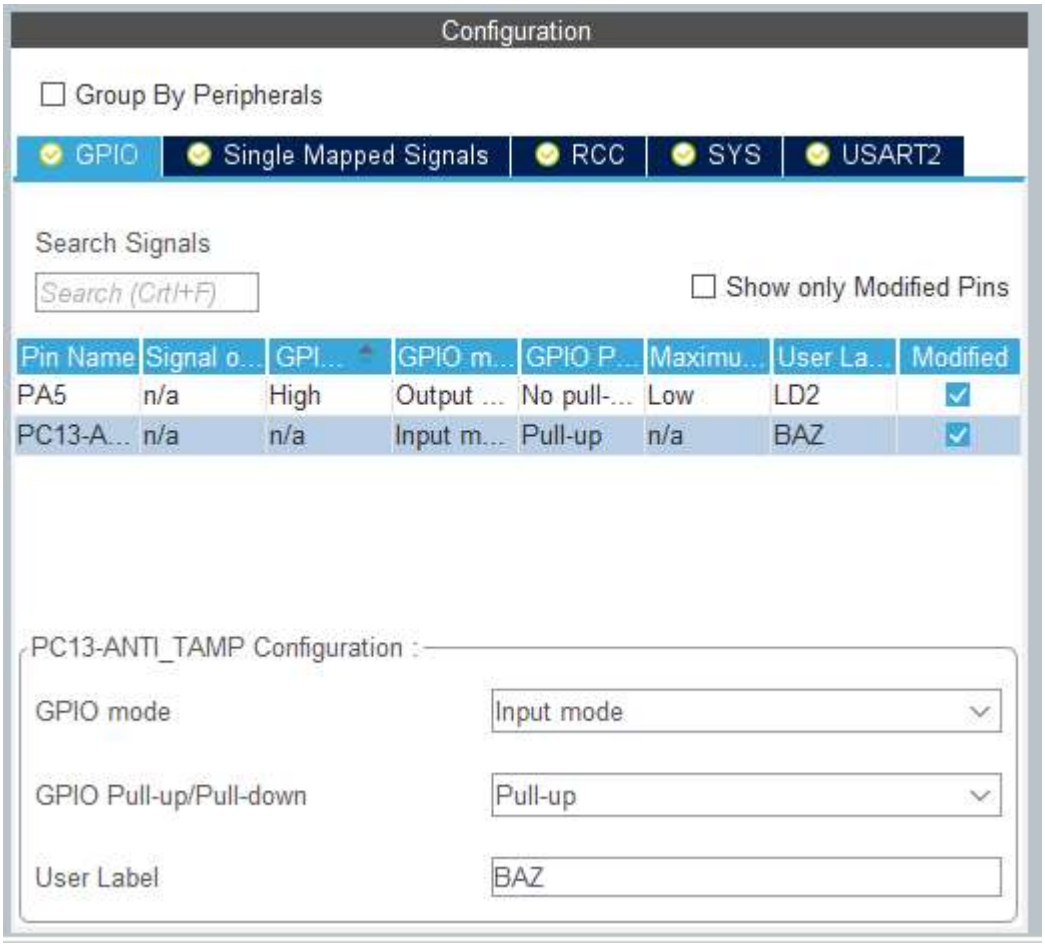

Figura 9: menú y configuración de GPIO de botón

Para PA5, el puerto del LED Verde (Figura 10):

- “GPIO output level” debe estar en “HIGH”, ya que el LED se encuentra conectado como ánodo común, por lo que un estado “HIGH” apagaría el LED, mientras que un estado “LOW” lo encendería. Debido a esto, se deja en “HIGH” para que este apagado hasta que el botón sea presionado.
- “GPIO mode” debe estar en “Output Open Drain”.
- “GPIO Pull-up/Pull-down” debe ponerse en “No Pull-up and no pull-down”, ya que no se requiere ninguna de estas resistencias del microcontrolador, el pin se usará como salida.
- “User Label” será generado de forma automática ya que fue etiquetado anteriormente.

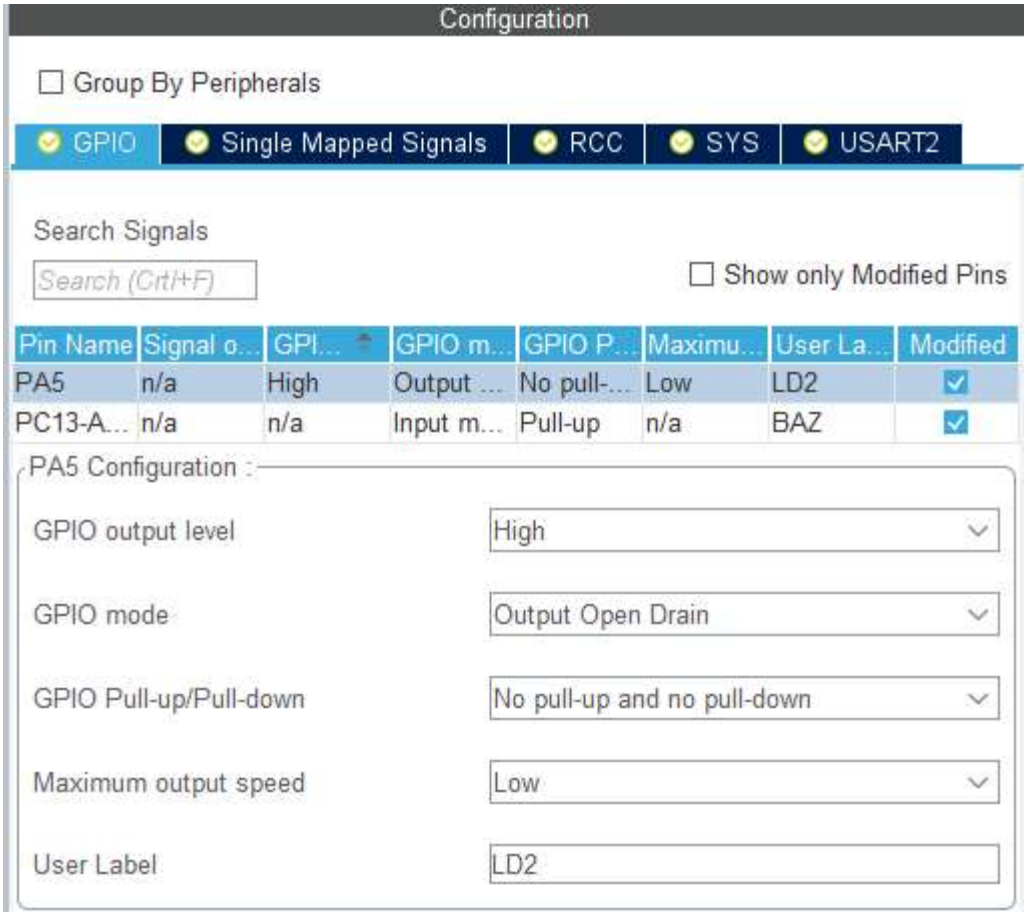

Figura 10: menú y configuración de GPIO de LED

Ya que estén configurados los pines, se debe revisar la configuración del proyecto (Figura 11 y 12), y entonces generar el código presionando en Project -> Generate Code, aparecerá un menú donde solamente hay que nombrar el proyecto y dar ok, si no cambia la vista automáticamente, se debe cambiar a C/C++ y posteriormente expander el folder src y abrir main.c.

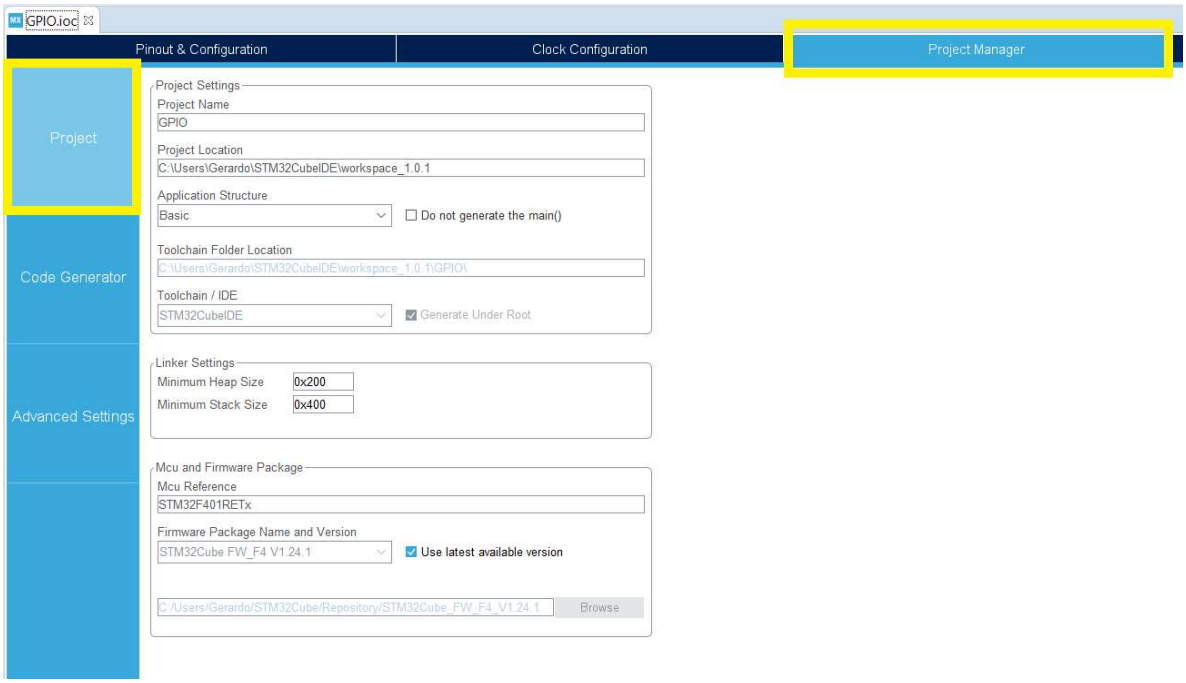

Figura 11: Pestaña de configuración de proyecto

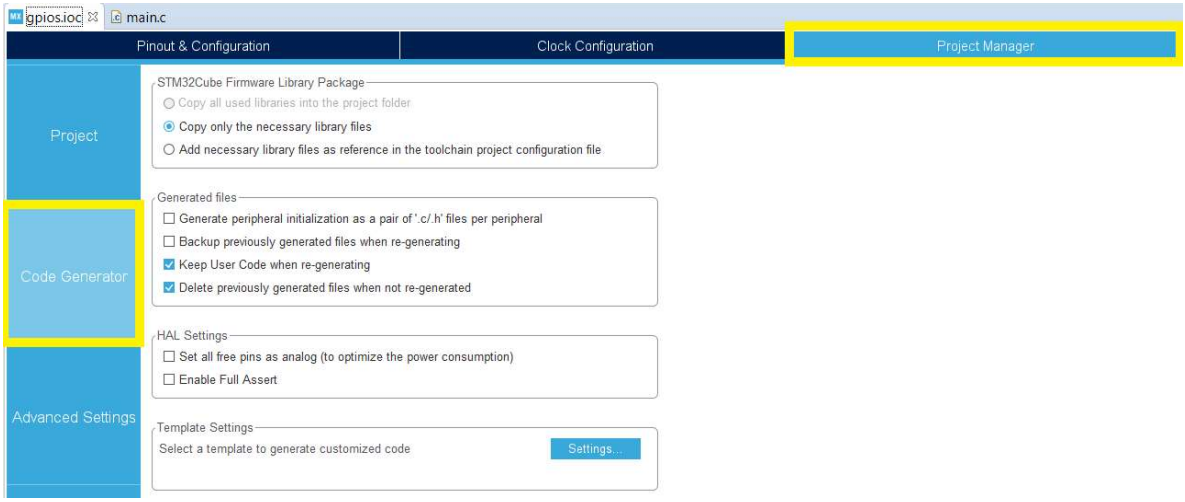

Figura 12: Pestaña de generación de código

Primero, después de un poco de información de uso de software, se encuentra en la línea 22 el inicio de los archivos cabecera con las bibliotecas necesarias para nuestro microcontrolador, después, en la línea 52 se encuentran funciones declaradas de forma automática de acuerdo a las configuraciones hechas en STM32CubeMX, en donde:

- *Void SystemClock_Config(void)* es una función que contiene las configuraciones del reloj con el que se va a trabajar.

- *static void MX_GPIO_Init(void)* es la función que va a contener las configuraciones realizadas con anterioridad a los pines de entrada y salida del microcontrolador. Estas configuraciones aparecen como función porque no se selecciono la opción de “Generate peripheral initialization as a pair of ‘.c/.h’ files per peripheral” (Figura 12).
- *“static void MX_USART2_UART_Init(void)”* no se usará en este ejemplo, sin embargo, es generado ya que la configuración predeterminada de la tarjeta en STM32CubeIDE trae el USART2 configurado.

```
21 /* Includes ------------------------------------------------------------------*/
22 #include "main.h"

51 /* Private function prototypes -----------------------------------------------*/
52 void SystemClock_Config(void);
53 static void MX_GPIO_Init(void);
54 static void MX_USART2_UART_Init(void);
```

Figura 13: Bibliotecas y prototipos de función

Después, en la línea 68 se encuentra la función principal, donde se comenzará a generar el código. Se puede observar que dentro de la función main se llama a las funciones de inicialización:

- *HAL_Init()* resetea e inicializa la interfaz Flash.
- *SystemClock_Config()* inicia las configuraciones del reloj.
- *MX_GPIO_Init()* configura los pines de entrada-salida.

```
68 int main(void)
69 {
70   /* USER CODE BEGIN 1 */
71
72   /* USER CODE END 1 */
73
74
75   /* MCU Configuration--------------------------------------------------------
76
77   /* Reset of all peripherals, Initializes the Flash interface and the Systic
78   HAL_Init();
79
80   /* USER CODE BEGIN Init */
81
82   /* USER CODE END Init */
83
84   /* Configure the system clock */
85   SystemClock_Config();
86
87   /* USER CODE BEGIN SysInit */
88
89   /* USER CODE END SysInit */
90
91   /* Initialize all configured peripherals */
92   MX_GPIO_Init();
93   MX_USART2_UART_Init();
94   /* USER CODE BEGIN 2 */
```

Figura 14: Configuración inicial

Después en la línea 100 se encuentra un “while infinito”, que ejecutará todo lo que tenga dentro todo el tiempo. El código de la aplicación de este manual será escrito dentro del while de la siguiente manera:

Primero se define una variable entera llamada ***m***, a la que se le asignará el valor del pin de entrada, en este caso el estado del botón, por medio de la siguiente función:

HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)

La cual lee el estado del pin indicado y regresa un 1 si se encuentra en “HIGH” o “SET”, y devuelve un 0 si el estado es “LOW” o “RESET”.

En los parámetros debe de indicarse el pin del puerto de entrada (GPIO_TypeDef* GPIOx), y justo después se escribe el pin que fue configurado como entrada (uint16_t GPIO_Pin), que en este ejemplo es llamado Bazul_Pin, por lo que la función debería quedar de la siguiente manera:

HAL_GPIO_ReadPin(Bazul_GPIO_Port, Bazul_Pin)

Debido a que el código se encuentra dentro del *while* infinito el estado del pin de entrada estará siendo leído todo el tiempo, así que se agregará un *if*, para que cuando el el botón entre en estado de “RESET” o bien cuando el botón sea presionado, el LED se encenderá medio segundo, para esto condicionamos la variable ***m*** igual a 0, así que si el botón se encuentra pulsado, el pin de salida (LDV) enciende por un segundo para luego apagarse.

Para colocar el estado a un pin de salida, se usará la función:

HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)

con la que se indica un estado “SET” o “RESET” a los pines de salida digitales, indicando un estado “ALTO” o “BAJO”, “ON” u “OFF”.

Los parámetros de la función son:

- GPIO_TypeDef* GPIOx, que se refiere al puerto donde el pin fue declarado una salida,
- uint16_t GPIO_Pin, el numero o nombre del pin declarado como salida digital,
- GPIO_PinState PinState, el estado requerido para el pin

De esta forma, para encender el pin se define la función como

HAL_GPIO_WritePin(LDV_GPIO_Port, LDV_Pin, GPIO_PIN_SET);

Y para que permanezca encendico se hace uso de la función *HAL_Delay()*, que es un retardo en milisegundos, por lo que si queremos que el LED prenda por medio segundo, debemos de escribir la función de retraso de la forma mostrada a

continuación: *HAL_Delay(500)* y posteriormente apagaremos el LED cambiando el estado de nuevo, de la siguiente manera:

HAL_GPIO_WritePin(LDV_GPIO_Port, LDV_Pin, GPIO_PIN_RESET);

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

    m = HAL_GPIO_ReadPin(Bazul_GPIO_Port, Bazul_Pin);
    if (m==0)
       {
                   HAL_GPIO_WritePin(LDV_GPIO_Port, LDV_Pin, GPIO_PIN_SET);
                   HAL_Delay(500);
                   HAL_GPIO_WritePin(LDV_GPIO_Port,LDV_Pin, GPIO_PIN_RESET);
        }
}
/* USER CODE END 3 */
```

Figura 15: Código completo

Si se va mas abajo se puede encontrar que en la línea 121 comienza la función que contiene la configuración del reloj, que termina en la 158; y a partir de la 198 y hasta la 224 se encuentran la configuración de los pines de entrada y salida como fueron configurados en STM32CubeMX.

```
198 static void MX_GPIO_Init(void)
199 {
200   GPIO_InitTypeDef GPIO_InitStruct = {0};
201
202   /* GPIO Ports Clock Enable */
203   __HAL_RCC_GPIOC_CLK_ENABLE();
204   __HAL_RCC_GPIOH_CLK_ENABLE();
205   __HAL_RCC_GPIOA_CLK_ENABLE();
206   __HAL_RCC_GPIOB_CLK_ENABLE();
207
208   /*Configure GPIO pin Output Level */
209   HAL_GPIO_WritePin(LDV_GPIO_Port, LDV_Pin, GPIO_PIN_RESET);
210
211   /*Configure GPIO pin : Bazul_Pin */
212   GPIO_InitStruct.Pin = Bazul_Pin;
213   GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
214   GPIO_InitStruct.Pull = GPIO_NOPULL;
215   HAL_GPIO_Init(Bazul_GPIO_Port, &GPIO_InitStruct);
216
217   /*Configure GPIO pin : LDV_Pin */
218   GPIO_InitStruct.Pin = LDV_Pin;
219   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
220   GPIO_InitStruct.Pull = GPIO_NOPULL;
221   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
222   HAL_GPIO_Init(LDV_GPIO_Port, &GPIO_InitStruct);
223
224 }
```

Figura 16: Configuración en software de los puertos

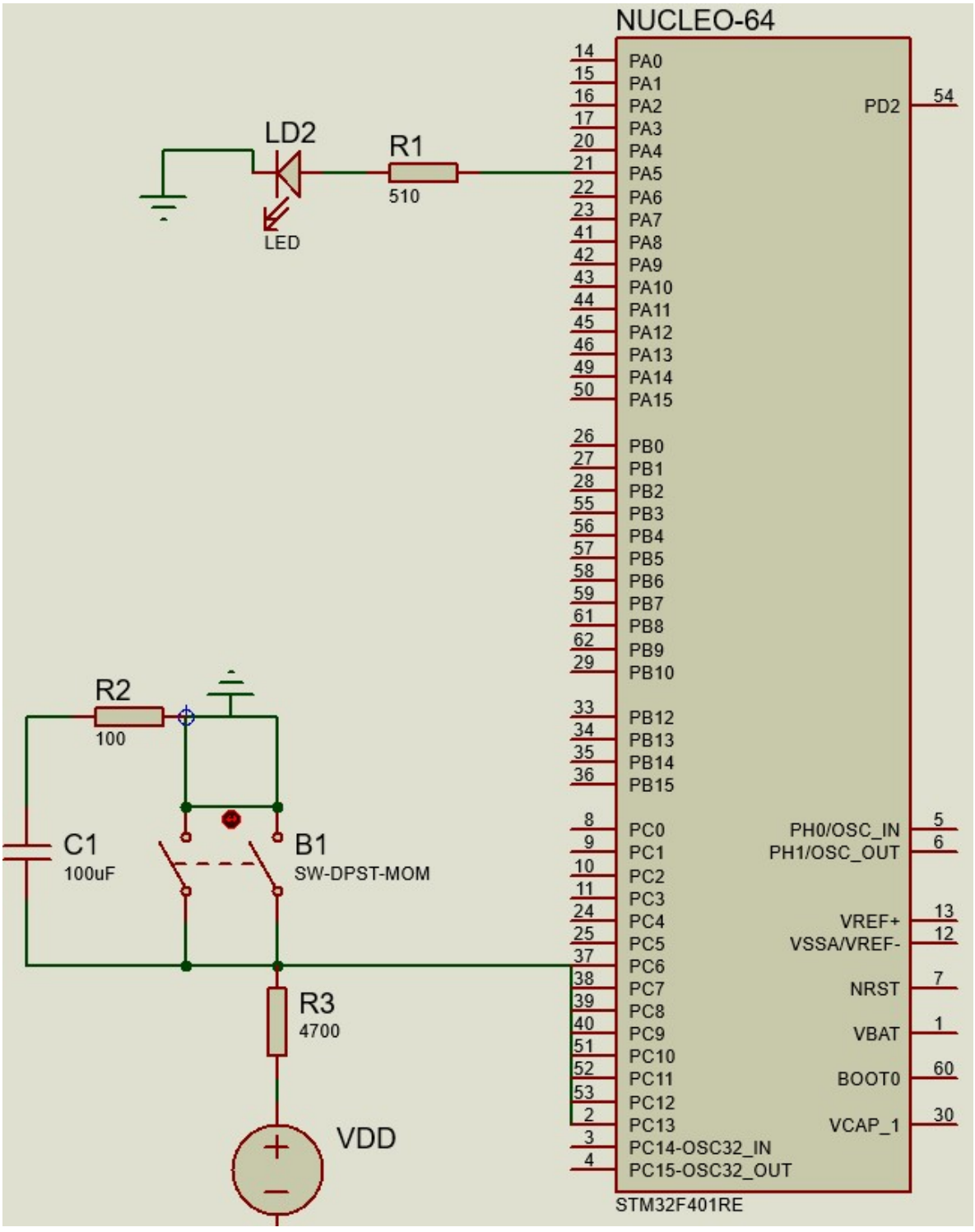

Figura 17: esquemático de la conexión del botón y el LED en el procesador.

Posteriormente se encuentra la función que se manda a llamar cuando hay algún error.

No es necesario preocuparse por estas funciones, no se les hará ningún cambio ya que fueron configuradas anteriormente desde STM32CubeMX.

Finalmente, hay que dirigirse al ícono del bicho, o presionar F11 para comenzar la sesión de depuración. Después de presionar “Resume”, al presionar el botón azul debería prender el LED verde por medio segundo, para después apagarse.