

## **MICROCONTROLADORES**

### **Práctica No. 1. Salidas Digitales.**

#### **1. Objetivo**

- Entender el funcionamiento de los puertos de Entrada/Salida.
- Crear un proyecto en el STM32CubeIDE
- Configurar la patita PC13 como salida y realizar un parpadeo del led cada 500 ms.

#### **2. Material y Equipo.**

- Computador o laptop.
- STM32CubeIDE.
- Un led de 5mm o 3mm.
- Una resistencia de 330Ω o 220Ω.

#### **3. Marco de Referencia**

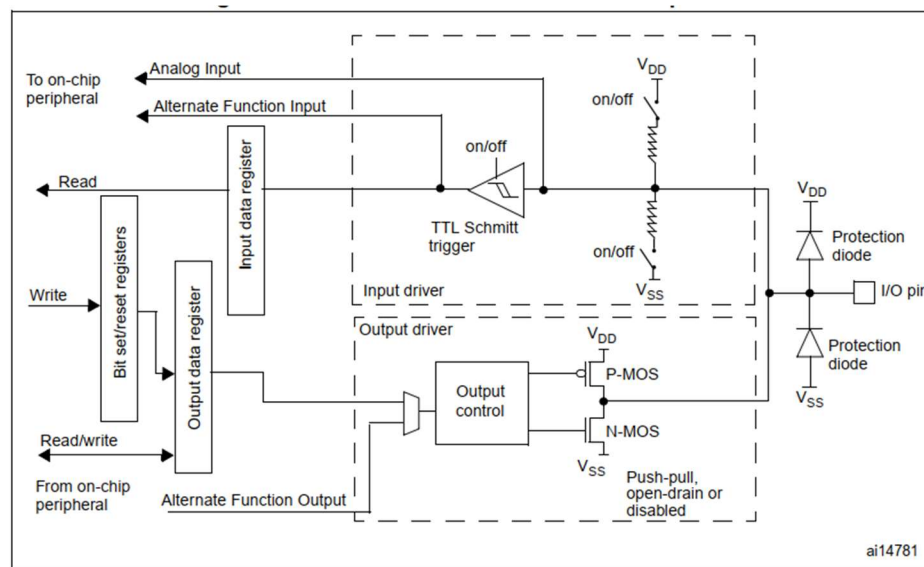
Cada puerto de E/S de propósito general tiene asociados dos registros de configuración de 32-bits (GPIOx\_CRL, GPIOx\_CRH), dos registros de datos de 32-bits (GPIOx\_ODR, GPIOx\_IDR), un registro set/reset de 32-bits (GPIOx\_BSRR), un registro reset de 16-bits (GPIOx\_BRR) y un registro de bloqueo de 32-bits (GPIOx\_LCKR).

Cada pin del microcontrolador puede ser configurado de la siguiente forma:

- Entrada Flotante (Input Floating).
- Entrada con Pull-Up (Input Pull-Up).
- Entrada con Pull-Down (Input Pull-Down).
- Analógica (Analog).

- Salida Open-Drain (Output Open-Drain).
- Salida Push-Pull (Output Push-Pull).
- Función Alterna Push-Pull (Alternate Function Push-Pull).
- Función Alterna Open-Drain (Alternate Function Open-Drain).

### ESTRUCTURA BÁSICA DE UN PUERTO DE E/S ESTÁNDAR.



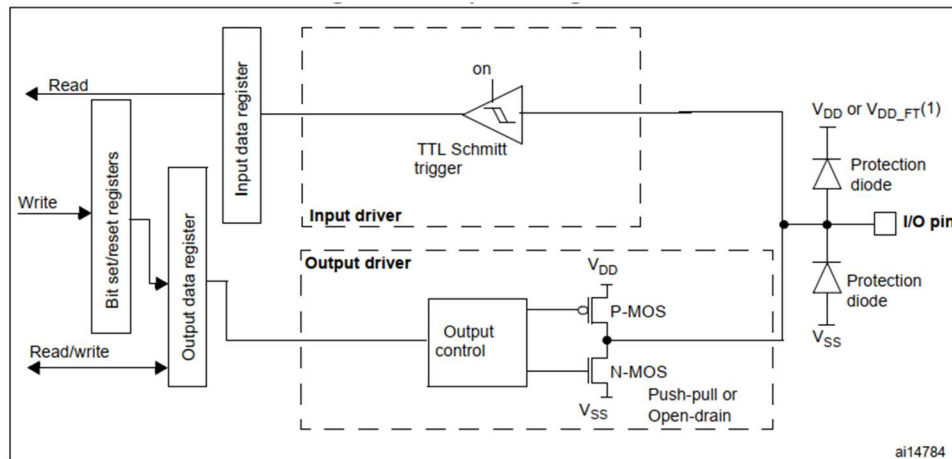
### Configuración de Salida.

Cuando el puerto es configurado como salida:

- El buffer de salida es habilitado.
  - Modo Open-Drain: Un "0" en el registro de salida activa el transistor N-MOS, mientras que un "1" en el registro de salida deja al puerto en modo de alta impedancia (Hi-Z, el transistor P-MOS nunca es habilitado).

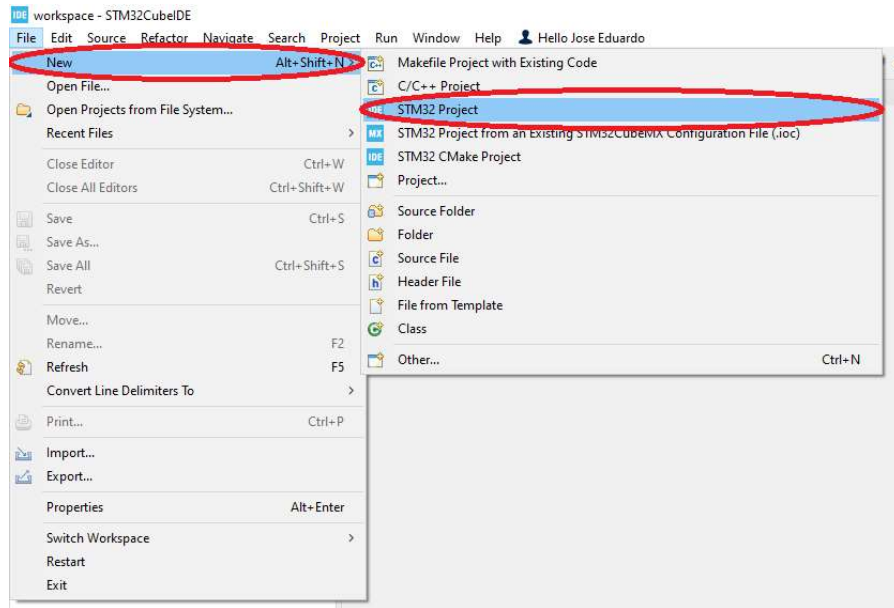
- Modo Push-Pull: Un “0” en el registro de salida activa el transistor N-MOS, mientras que un “1” activa el transistor P-MOS.
- La entrada Schmitt-Trigger es activada.
- Las resistencias Pull-Up y Pull-Down son deshabilitadas.
- Los datos presentes en los pines de E/S son muestreados en el registro de datos de entrada cada ciclo de reloj del APB2.
- Una lectura al registro de entrada de datos obtiene el estado de las E/S en modo Open-Drain.
- Una lectura al registro de salida de datos obtiene el ultimo valor escrito en modo Push-Pull.

#### CONFIGURACION DE SALIDA.

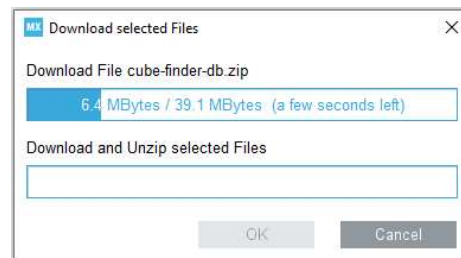


## 4. Desarrollo y Procedimiento.

1. Abrimos el STM32CubeIDE y seleccionamos el menú File>New>STM32 Project.

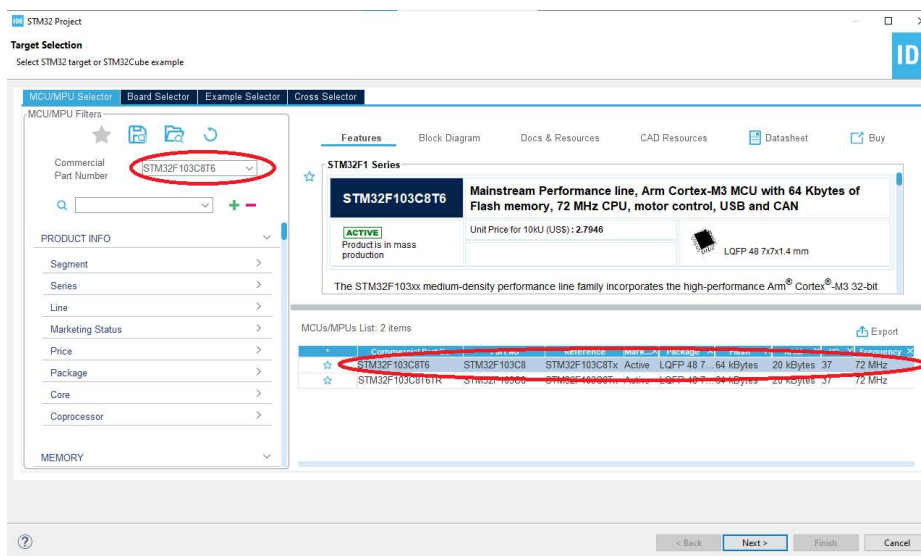


2. Al crear el proyecto el STM32CubeIDE puede actualizar algunas de sus características esto lo hace automáticamente al tener conectada la computadora a internet y aparecerá una ventana como la que se muestra a continuación.

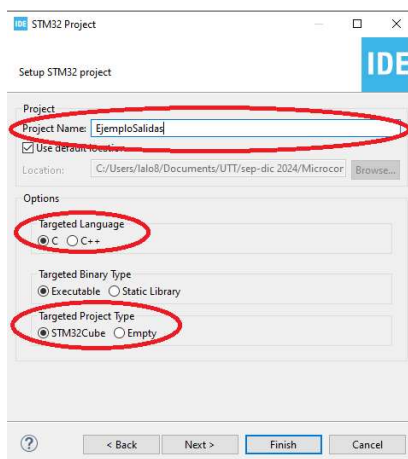


3. Después de actualizar nos mostrara una ventana donde debemos seleccionar el dispositivo que vamos a utilizar en el proyecto, el

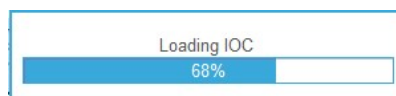
dispositivo a seleccionar es el STM32F103C8T6 como se muestra a continuación y damos click en “Next”.



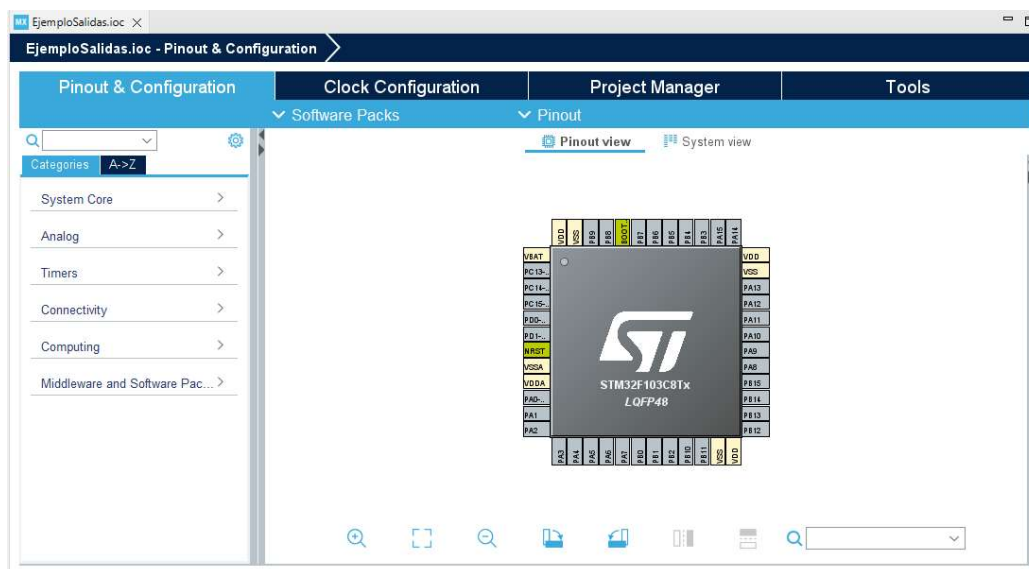
- En la siguiente ventana nos pedirá el nombre del proyecto y el tipo de proyecto que se va a crear. En “Project Name” escribimos el nombre del proyecto el cual será “EjemploSalidas” después nos aseguramos que en “Targeted Language” este seleccionado “C”, en “Targeted Binary Type” este seleccionado “Executable” y en “Targeted Project Type” seleccionamos “STM32Cube” y damos click en Finish.



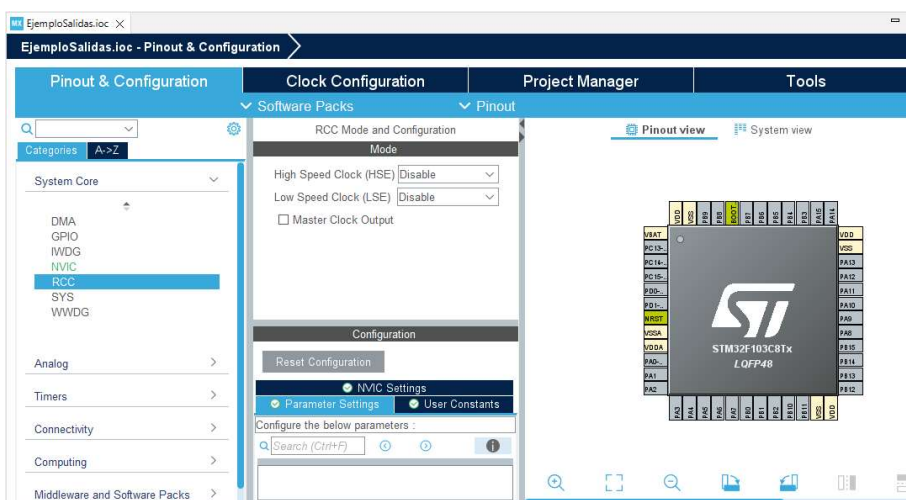
5. Esperamos a que inicie la interfaz del STM32Cube nos aparecerá una barra de progreso como la que se muestra a continuación.



6. Al terminar de cargar la interfaz del CubeMX nos aparecerá una ventana dentro del STM32CubeIDE con cuatro pestañas arriba al centro que son “Pinout & Configuration”, “Clock Configuration”, “Project Manager” y “Tools”. La pestaña seleccionada por default es “Pinout & Configuration” y nos muestra el empaquetado del microcontrolador y su “pinout” (lado derecho) y del lado izquierdo una columna varias opciones de configuración que utilizaremos para habilitar ciertas características del MCU tales como habilitar el oscilador externo y los periféricos que vayamos a usar.

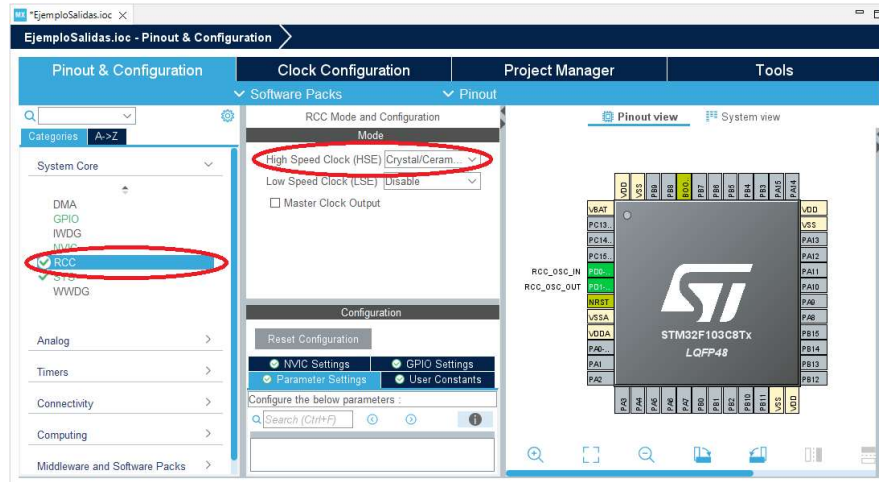


7. Lo primero que haremos es configurar el reloj del MCU para que opere a su máxima frecuencia que son 72MHz. Lo primero que tenemos que hacer es asegurarnos de estar en la pestaña “Pinout & Configuration” y en la columna de la izquierda seleccionamos la opción “System Core” y la desplegamos con el símbolo “>” que se encuentra a la derecha y seleccionamos la opción “RCC” y nos mostrara otra ventana intermedia como se muestra a continuación.

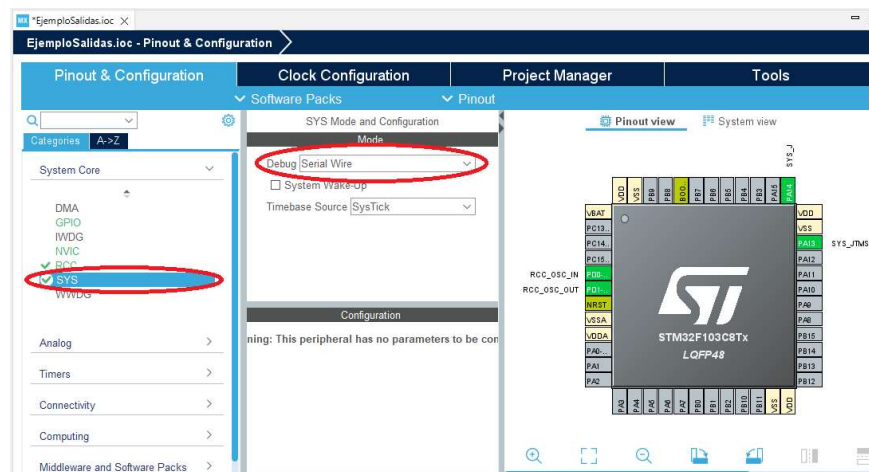




8. En la ventana “RCC Mode and Configuration” vamos a la opción “High Speed Clock (HSE)” y seleccionamos “Crystal/Ceramic Resonator” automáticamente los pines PD0 y PD1 se pondrán de color verde como se muestra en la siguiente figura.

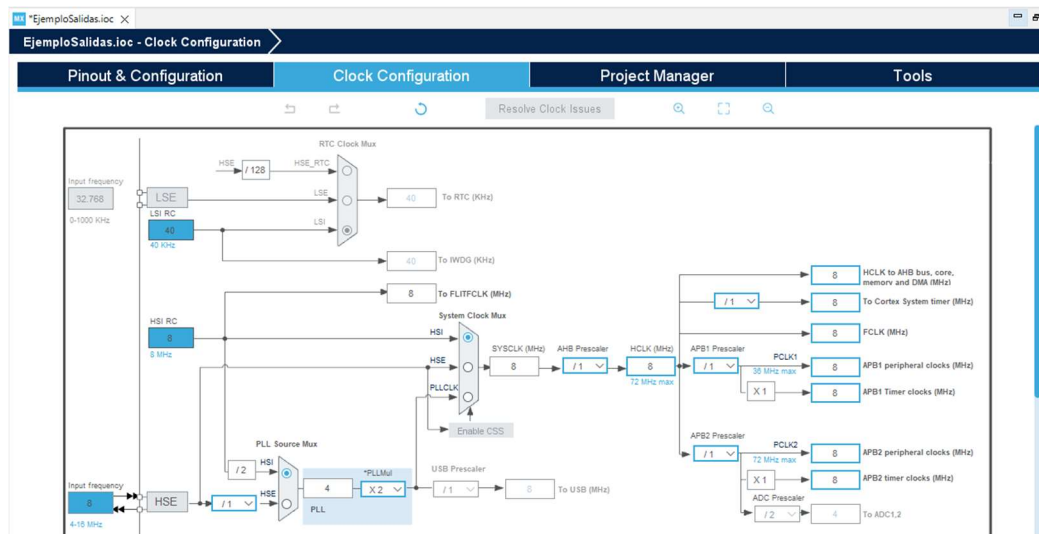


9. Lo siguiente es configurar el modo depurador para eso en la columna de la izquierda dentro de “System Core” vamos a la opción “SYS” y nos aparece nuevas opciones en la columna central y vamos a la opción “Debug2” y seleccionamos “Serial Wire” con lo cual los pines PA13 y PA14 se pondrán de color verde como se muestra en la siguiente figura.

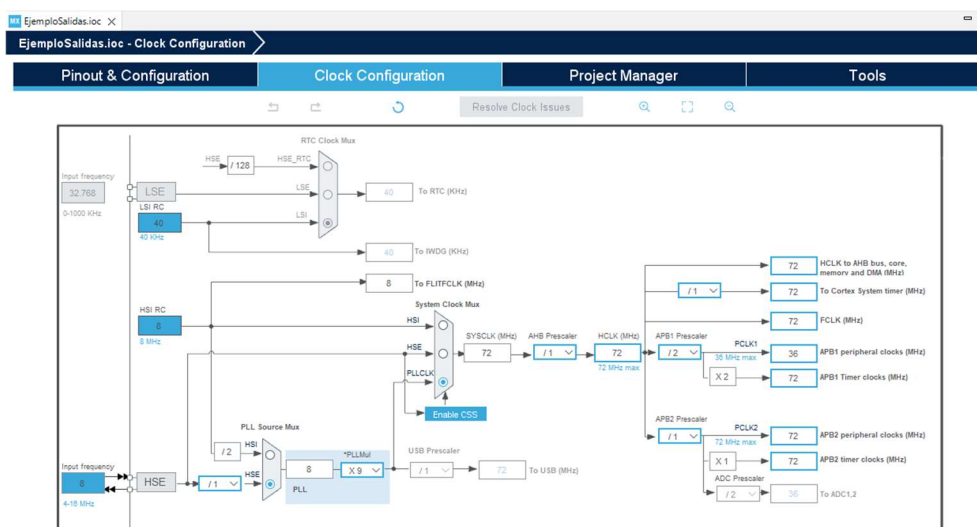




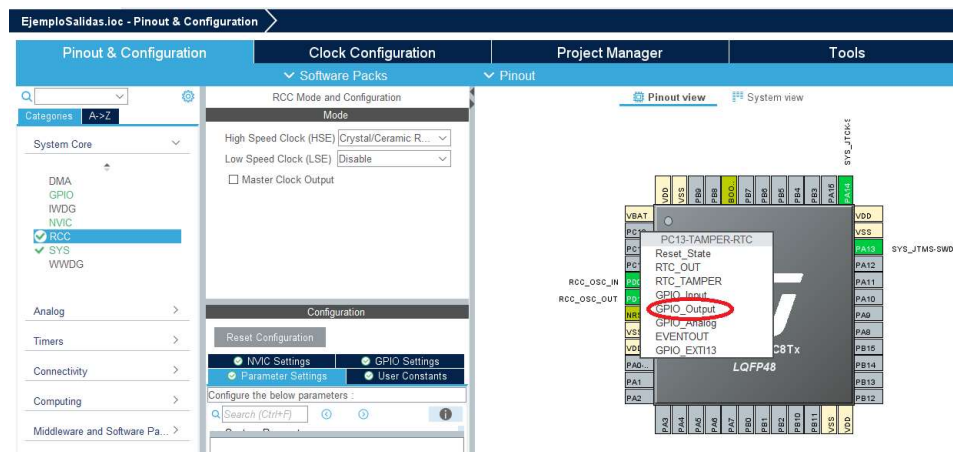
10. Lo siguiente es configurar el oscilador para que el microcontrolador tenga una señal de reloj interna de 72MHz, para eso nos vamos a la pestaña “Clock Configuration” se nos despliega la estructura interna del oscilador para el STM32F103C8T6 como se muestra a continuación.



11. Los pasos para configurar la frecuencia de 72MHz son los siguientes. Primero nos aseguramos que en el multiplexor “PLL Source Mux” seleccionamos “HSE” después en el multiplexor principal “System Clock Mux” seleccionamos “PLLCLK” con eso en el recuadro “SYSCLK” y “HCLK” aparecerá una frecuencia de 16 MHz u 8 MHz entonces en el recuadro “HCLK” cambiamos el 16 u 8 por 72 y damos “Enter” automáticamente el CubeMX hará el cálculo de los divisores y multiplicadores para alcanzar dicha frecuencia dejando la siguiente configuración.



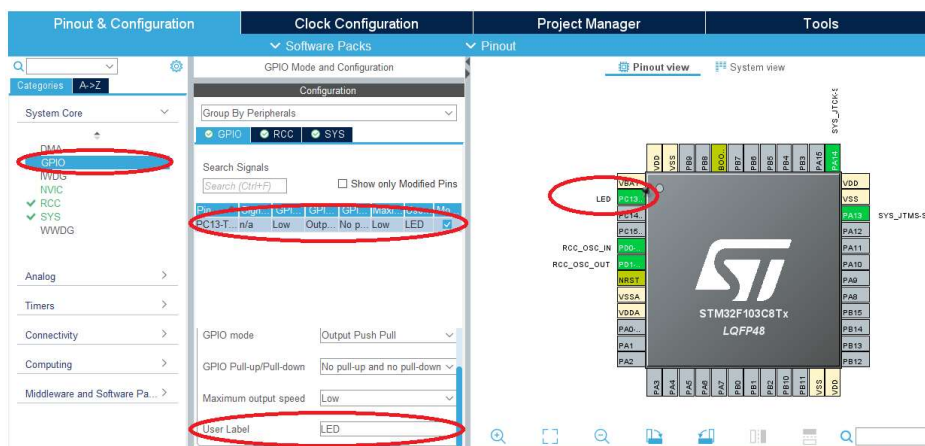
12. Después de cambiar la frecuencia regresamos a la pestaña “Pinout & Configuration” vamos a la parte del empaquetado del MCU también llamado “Pinout View” buscamos el pin PC13 y lo configuramos como salida como se muestra a continuación.



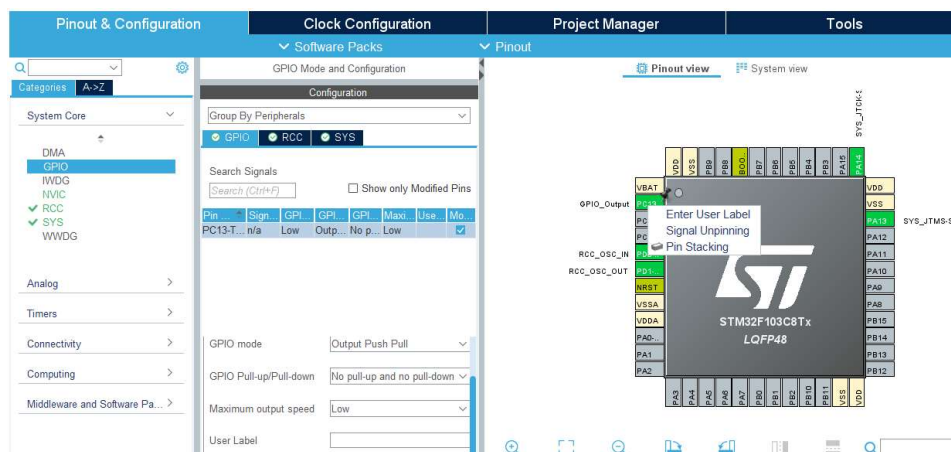
13. El pin PC13 cambiara a color verde y aparecerá con la leyenda “GPIO\_Output” para cambiar esa etiqueta hay dos formas la primera es ir a la columna de la izquierda en la opción “System Core” y después “GPIO” en la columna central “GPIO Mode and Configuration” seleccionamos PC13 en la parte inferior de esa misma columna nos desplegara las opciones de configuración del PIN las cuales son:

- GPIO output level: Low.
- GPIO mode: Output Push Pull.
- GPIO Pull-up/Pull-down: No pull-up and no pull-down.
- Maximum output speed: Low.
- User Label: LED.

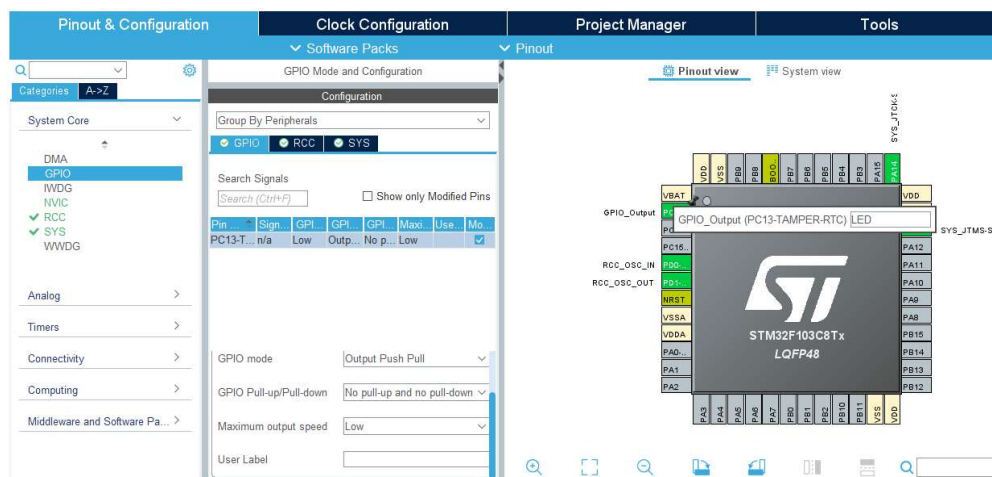
Las opciones se muestran a continuación.



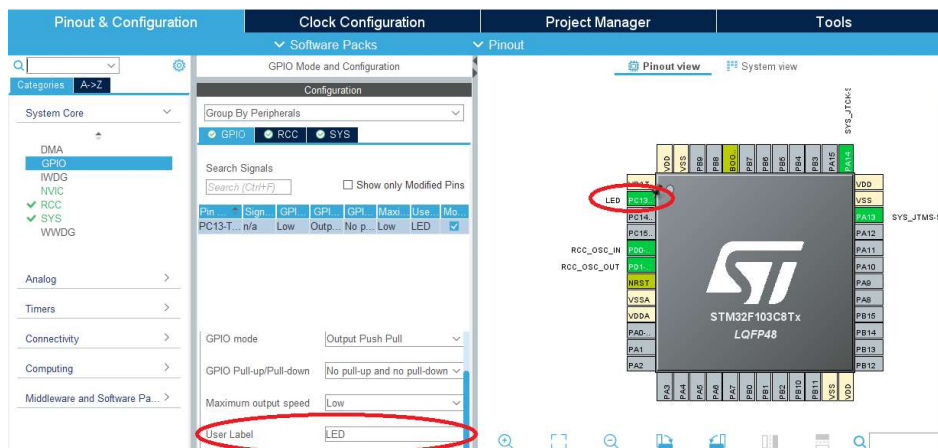
La segunda opción es mas directa y es haces click derecho sobre el pin y nos aparece un menú y seleccionamos la opción “Enter User Label” como se muestra a continuación.



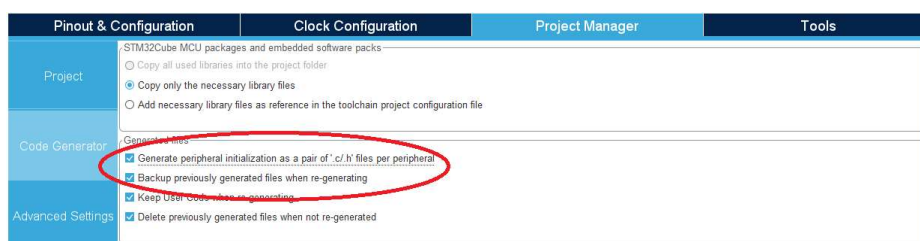
Al seleccionar la opción “Enter User Label” nos aparece un cuadro de texto donde ingresamos la etiqueta para ese pin como se muestra a continuación.



Damos Enter y el pin desplegará la etiqueta que le hemos puesto así como en la opción “User Label” en la columna central “GPIO Mode and Configuration”. La configuración debería quedar de la siguiente forma.



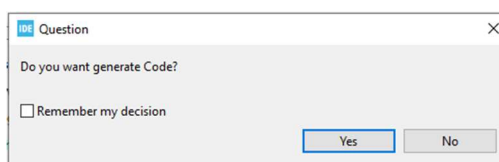
14. Lo siguiente es ir a la pestaña “Project Manager” y en la opción “Code Generator” palomeamos las dos primeras opciones de “Generated Files” como se muestra a continuación.



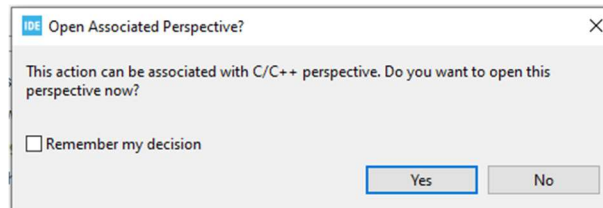
La primera opción “Generate peripheral initialization as a pair of ‘.c/.h’ files per peripheral” genera un archivo de código fuente .c y un archivo de cabecera .h por periférico habilitado.

La segunda opción “Backup previously generated files when re-generation” guarda un respaldo de los archivos modificados después de una regeneración de código al modificar algún parámetro dentro del CubeMX.

15. Por ultimo lo unico que nos falta es guardar el proyecto para que el CubeMX genere el codigo C correspondiente a todas las configuraciones hechas. Nos aparecera una venta preguntando si queremos generar el codigo a lo cual decimos que si (Yes).



Luego nos volverá a preguntar esta vez si queremos cambiar de perspectiva a la de C/C++ a lo cual decimos que si (Yes).



Nota: Si es la primera vez que se genera el código el CubeMX descargará los archivos y librerías necesarias para la familia de microcontroladores que estemos usando en este caso para el STM32F103C8T6 corresponde las librerías HAL para la familia F1.

Una vez terminada la generación de código volveremos a la perspectiva de C/C++ dependiendo si es la primera vez que se genera un programa el tiempo puede variar y debe hacerse esta configuración con una conexión a internet para descargar las librerías necesarias.

16. El código de la práctica es el siguiente.

```

1 #include "main.h"
2 #include "gpio.h"
3
4 void SystemClock_Config(void);
5
6 int main(void)
7 {
8     HAL_Init();
9
10    SystemClock_Config();
11
12    MX_GPIO_Init();
13
14    while (1)
15    {
16        HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
17        HAL_Delay(500);
18    }
19 }
20
21 void SystemClock_Config(void)
22 {
23     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
24     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
25
26     /** Initializes the RCC Oscillators according to the specified parameters
27      * in the RCC_OscInitTypeDef structure.
28      */
29     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
30     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
31     RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
32     RCC_OscInitStruct.HSIState = RCC_HSI_ON;
33     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;

```



```

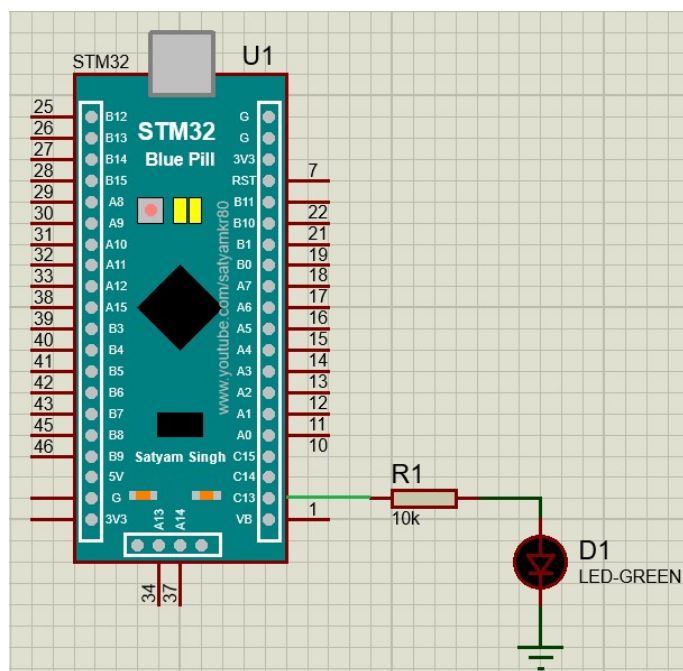
34 RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
35 RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
36 if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
37 {
38     Error_Handler();
39 }
40
41 // Initializes the CPU, AHB and APB buses clocks
42 //
43 RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
44 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
45 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
46 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
47 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
48 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
49
50 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
51 {
52     Error_Handler();
53 }
54 }
55
56 void Error_Handler(void)
57 {
58     /* USER CODE BEGIN Error_Handler_Debug */
59     /* User can add his own implementation to report the HAL error return state */
60     __disable_irq();
61     while (1)
62     {
63     }
64     /* USER CODE END Error_Handler_Debug */
65 }
66
67 #ifndef USE_FULL_ASSERT
68 /**
69  * @brief Reports the name of the source file and the source line number
70  * where the assert_param error has occurred.
71  * @param file: pointer to the source file name
72  * @param line: assert_param error line source number
73  * @retval None
74  */
75 void assert_failed(uint8_t *file, uint32_t line)
76 {
77     /* USER CODE BEGIN 6 */
78     /* User can add his own implementation to report the file name and line number,
79     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
80     /* USER CODE END 6 */
81 }
82 #endif /* USE_FULL_ASSERT */

```

Nota: Recuerde que todo el código debe ir dentro de los comentarios “USER CODE BEGIN” y “USER CODE END” de lo contrario al regenerar el código todo el código puesto fuera de estos comentarios será eliminado por lo tanto no borre los comentarios que el CubeMX agregue al código, los programas mostrados en este y demás practicas serán sin comentarios por efectos de espacio en el documento.

## 5. Diagrama Esquemático de la Practica.

1. El esquemático de la práctica es el siguiente.

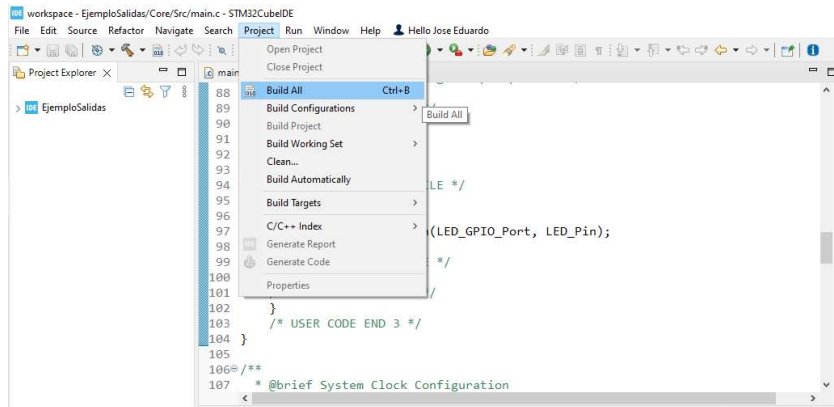



El led conectado a PC13 puede ser externo o se puede dejar sin conectar y usar el led que viene en la tarjeta.

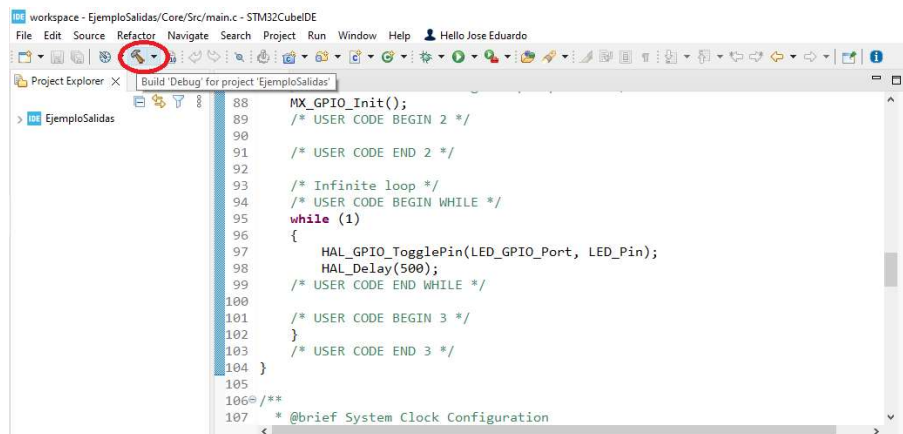
No se harán simulaciones en Proteus, pero se deja al alumno esta opción para que vea los resultados simulados, no se usará el simulador por que en su lugar se usará el modo de depuración con el programador ST-Link V2.

## 6. Compilación y Programación del STM32F103C8T6.

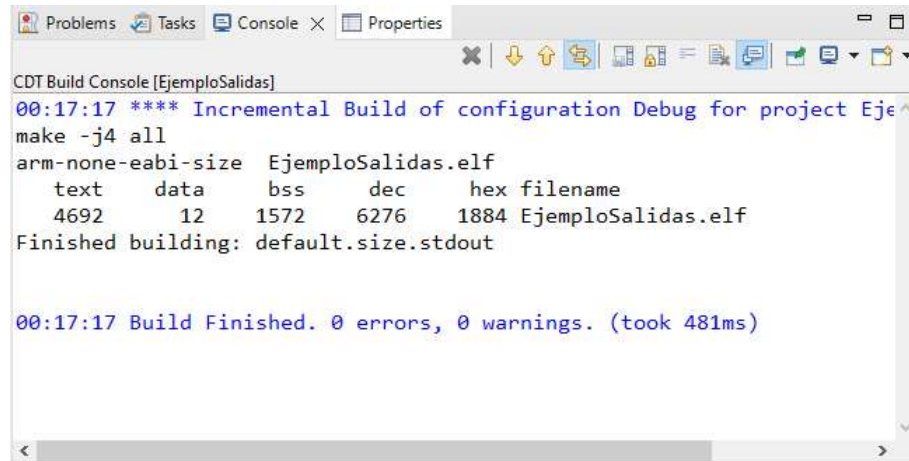
Para compilar el proyecto solo tenemos que ir al menú “Project>Build All” para compilar el proyecto como se muestra a continuación.



Otra forma de compilar el proyecto es a través de las opciones de la barra de herramientas en el icono del martillo  como se muestra a continuación.



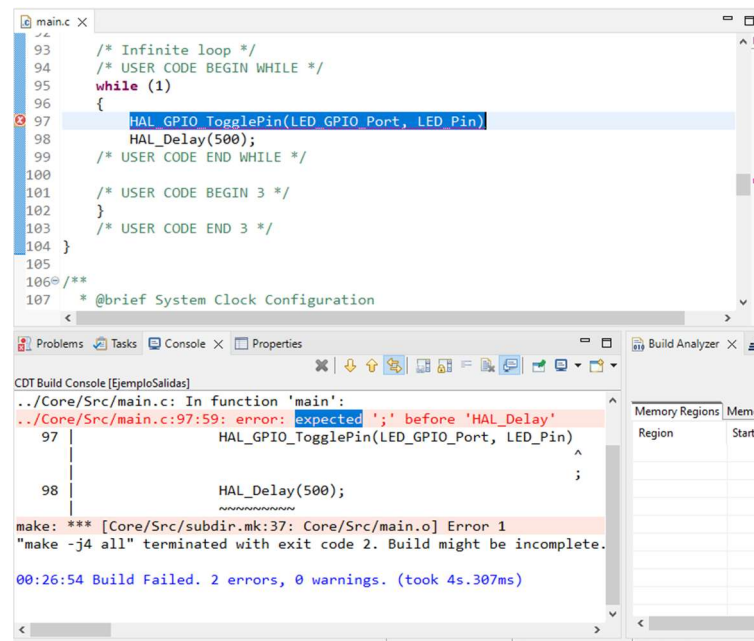
Al realizar la compilación del proyecto si todo ha ido correctamente y no se han encontrado errores en el programa en la ventana “Console” deberá aparecer lo siguiente.



```
CDT Build Console [EjemploSalidas]
00:17:17 **** Incremental Build of configuration Debug for project Eje
make -j4 all
arm-none-eabi-size  EjemploSalidas.elf
   text    data    bss     dec     hex filename
   4692     12   1572    6276    1884 EjemploSalidas.elf
Finished building: default.size.stdout

00:17:17 Build Finished. 0 errors, 0 warnings. (took 481ms)
```

Si todo ha salido bien deberá aparecer 0 errors, 0 warnings. Si por el contrario hubiera algún error en esta misma ventana nos desplegar el tipo de error y al darle doble click al error nos llevara a la línea donde esta el error. La siguiente venta muestra un error típico cuando olvidamos un “;” al final de una instrucción.



```


93  /* Infinite loop */
94  /* USER CODE BEGIN WHILE */
95  while (1)
96  {
97    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin)
98    HAL_Delay(500);
99    /* USER CODE END WHILE */
100
101    /* USER CODE BEGIN 3 */
102  }
103    /* USER CODE END 3 */
104 }
105
106 /**
107  * @brief System Clock Configuration
  
```

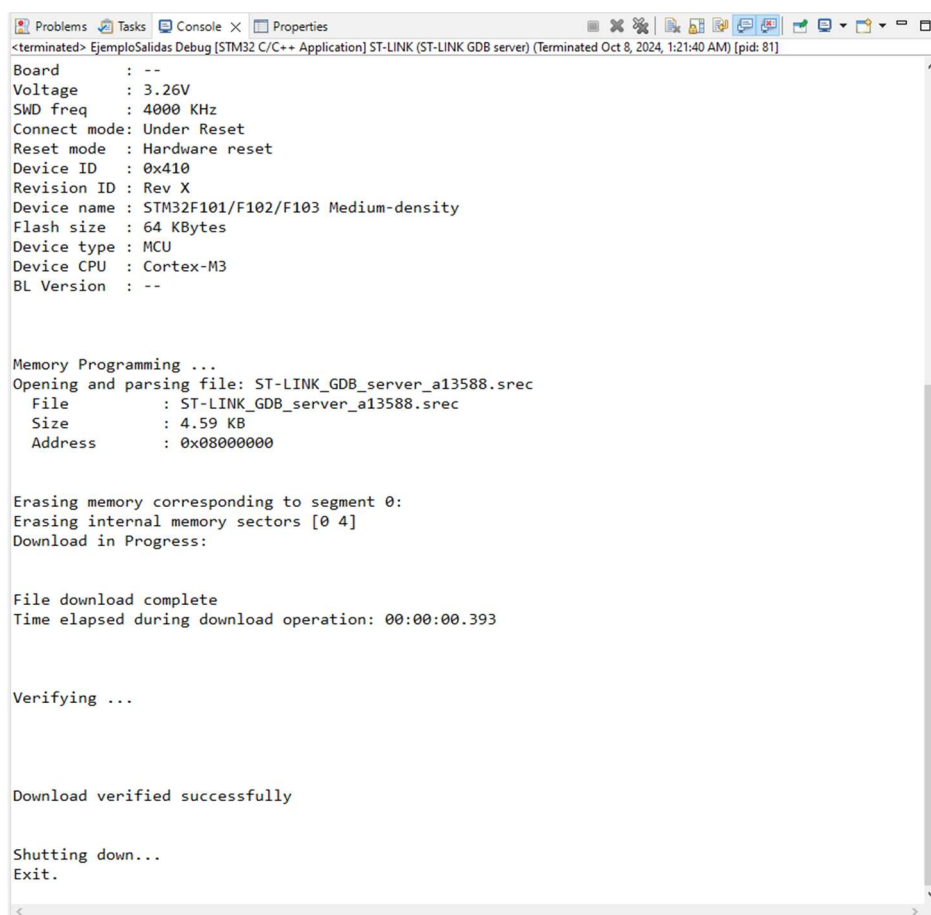
Problems: CDT Build Console [EjemploSalidas]

```

../Core/Src/main.c: In function 'main':
../Core/Src/main.c:97:59: error: expected ';' before 'HAL_Delay'
97 |     HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin)
    |                                         ^
98 |     HAL_Delay(500);
    |     ~~~~~
make: *** [Core/Src/subdir.mk:37: Core/Src/main.o] Error 1
"make -j4 all" terminated with exit code 2. Build might be incomplete.
00:26:54 Build Failed. 2 errors, 0 warnings. (took 4s.307ms)
  
```

Al darle doble click al error podemos ver que la línea donde esta se nos colorea de color azul y en la parte izquierda de la misma línea al lado del número de línea nos aparece una “X” indicando también el error. Corregimos el error agregando el punto y coma al final de la instrucción “HAL\_GPIO\_TogglePin” volvemos a compilar y ya no debería de aparecer el error.

El ultimo paso es grabar el programa en el microcontrolador para eso damos click en el botón  si no se ha compilado el programa se realizará una compilación después se empezará el proceso de grabación como se muestra en la siguiente figura.



```
<terminated> EjemploSalidas Debug [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Terminated Oct 8, 2024, 1:21:40 AM) [pid: 81]

Board      : --
Voltage    : 3.26V
SWD freq   : 4000 KHz
Connect mode: Under Reset
Reset mode : Hardware reset
Device ID  : 0x410
Revision ID: Rev X
Device name: STM32F101/F102/F103 Medium-density
Flash size: 64 KBytes
Device type: MCU
Device CPU : Cortex-M3
BL Version : --

Memory Programming ...
Opening and parsing file: ST-LINK_GDB_server_a13588.srec
File      : ST-LINK_GDB_server_a13588.srec
Size      : 4.59 KB
Address   : 0x08000000

Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 4]
Download in Progress:

File download complete
Time elapsed during download operation: 00:00:00.393

Verifying ...

Download verified successfully

Shutting down...
Exit.
```

Al final si no hay ningún problema con la tarjeta o el programador debería aparecer “Download verified succesfully” el MCU se reiniciará y nuestro programa empezara a funcionar y el led de la tarjeta conectado al pin PC13 empezara a parpadear con un intervalo de medio segundo con esto se finalizará la práctica.

## **7. Observaciones.**

Esta sección es para que el alumno anote sus observaciones.

## **8. Conclusiones.**

Esta sección es para que el alumno anote sus conclusiones.

## **9. Importante.**

Se deberá entregar un manual de prácticas al final del cuatrimestre que incluya todas las practicas que se realicen, este reporte debe contener lo siguiente.

- Portada, la portada debe incluir:
  - Escuela.
  - Materia.
  - Nombre del Alumno y Numero de Control.
  - Lugar y fecha de la entrega.

Después de la presentación las practicas deberán incluirse al reporte de prácticas una a una en orden empezando desde la practica 1 hasta la práctica que se llegue teniendo cada reporte de practica los siguientes puntos.

- Nombre y número de la práctica.
- Objetivo.
- Material usado.
- El fundamento teórico.
- El desarrollo y resultado (incluir el programa en imágenes, simulación y fotos de la practica funcionando en físico).
- Observaciones.
- Conclusiones.

Cada practica deberá ser validada en clase el alumno que no tenga validada todas las practicas no podrá entregar el manual de prácticas hasta que haya entregado las practicas faltantes.