



## **Actividad de Formación Práctica 1**

*Entorno de Desarrollo Integrado STM32CubeIDE.*

*Programación de microcontroladores.*

### ***Grupo 3***

#### ***Integrantes:***

*Castro Oscar Martín*

*Décima Enrique Emanuel*

*Ortiz Nicolás Agustín*

*Tejerina Dámaris Carla Marianela*

[Contenido](#)

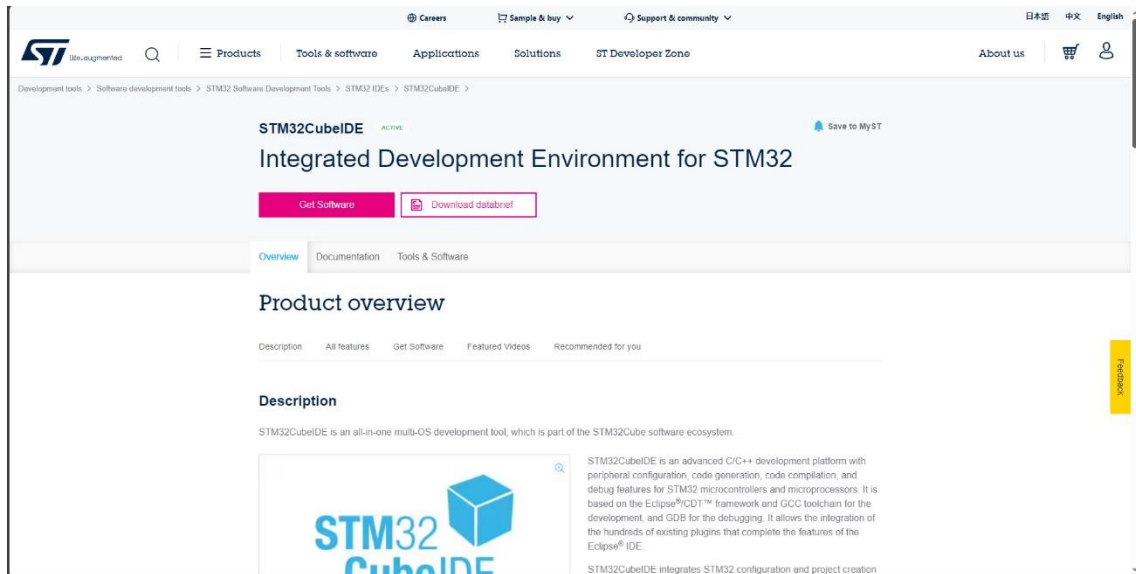
Descarga.....	2
Instalación.....	3
Creación de un Nuevo Proyecto.....	6
Proyecto de ejemplo: “Parpadeo de un LED” .....	12
Escritura del Código.....	13
Compilación del proyecto.....	14
Debug .....	15
Enlaces .....	19

## Descarga

Para descargar el STM32CubeIDE nos dirigimos a la página oficial de STMicroelectronics:

<https://www.st.com/en/development-tools/stm32cubeide.html> En

donde observaremos lo siguiente:

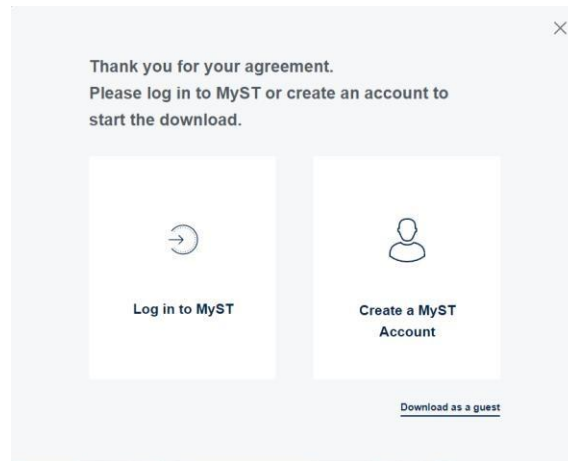


Debemos deslizar hasta la sección “Get Software”, en donde encontraremos las opciones de descarga, dependiendo de nuestro sistema operativo y la versión del software que deseamos descargar elegimos una opción de descarga. Si deseamos descargar la última versión disponible presionamos en “Get latest”.

### Get Software

Part Number	General Description	Latest version	Download	All versions
<a href="#">+ STM32CubeIDE-DEB</a>	STM32CubeIDE Debian Linux Installer	1.16.0	<a href="#">Get latest</a>	<a href="#">Select version</a>
<a href="#">+ STM32CubeIDE-Lnx</a>	STM32CubeIDE Generic Linux Installer	1.16.0	<a href="#">Get latest</a>	<a href="#">Select version</a>
<a href="#">+ STM32CubeIDE-Mac</a>	STM32CubeIDE macOS Installer	1.16.0	<a href="#">Get latest</a>	<a href="#">Select version</a>
<a href="#">+ STM32CubeIDE-RPM</a>	STM32CubeIDE RPM Linux Installer	1.16.0	<a href="#">Get latest</a>	<a href="#">Select version</a>
<a href="#">+ STM32CubeIDE-Win</a>	STM32CubeIDE Windows Installer	1.16.0	<a href="#">Get latest</a>	<a href="#">Select version</a>

Para permitirnos descargar el IDE, la pagina nos solicitará que creamos una cuenta, o en el caso de que ya tengamos una cuenta creada, que ingresemos. Esto es obligatorio para poder descargar el IDE.



Una vez creada la cuenta, nos permitirá descargar el instalador.

## Instalación

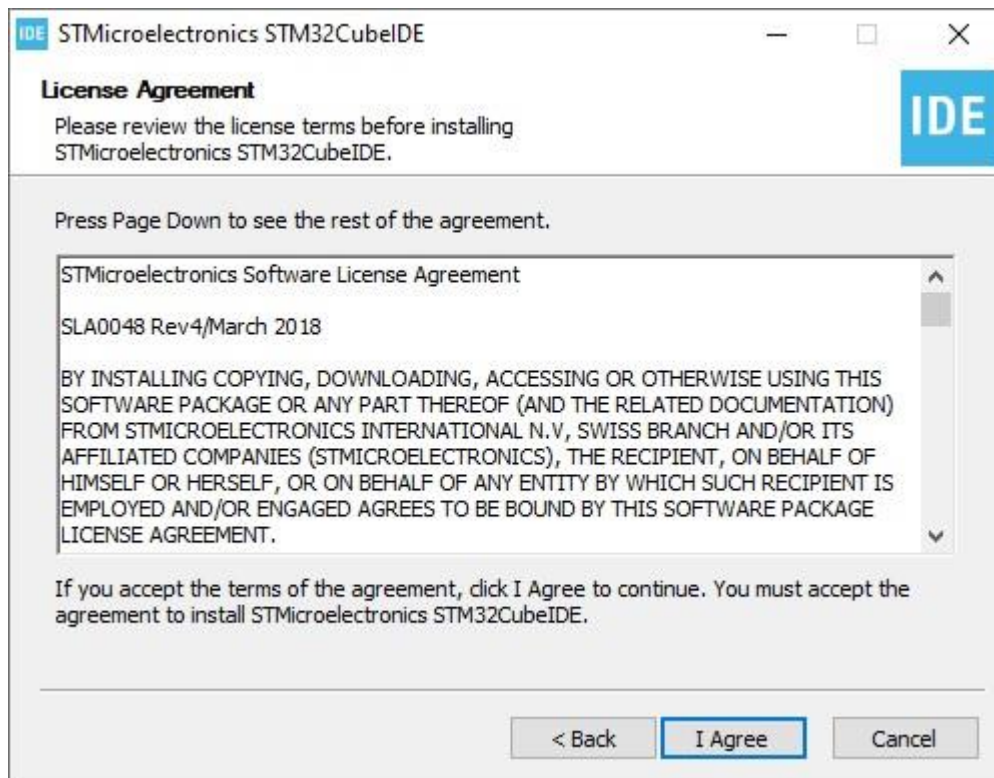
Comenzamos la instalación ejecutando el instalador ubicado en nuestra carpeta de Descargas:

Nombre	Fecha de modificación	Tipo	Tamaño
 st-stm32cubeide_1.15.1_21094_20240412_1041_x86_64	18/4/2024 11:48	Aplicación	1.037.069 KB

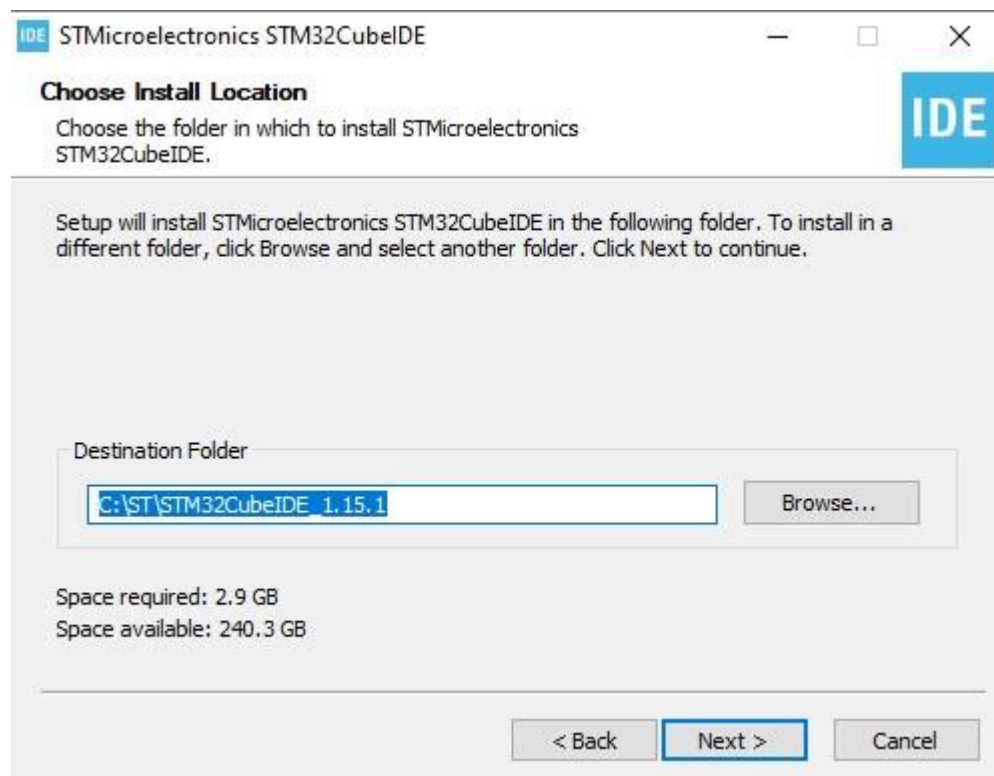
Y nos abrirá la siguiente ventana:



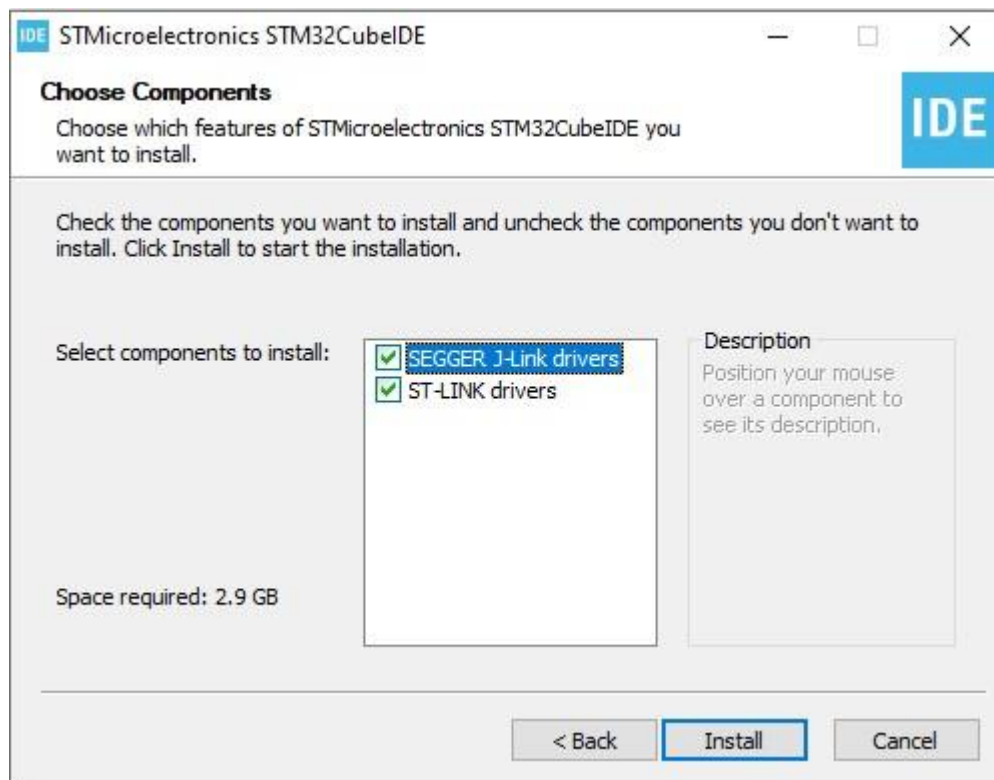
Presionamos "Next >" y nos aparecerá lo siguiente.



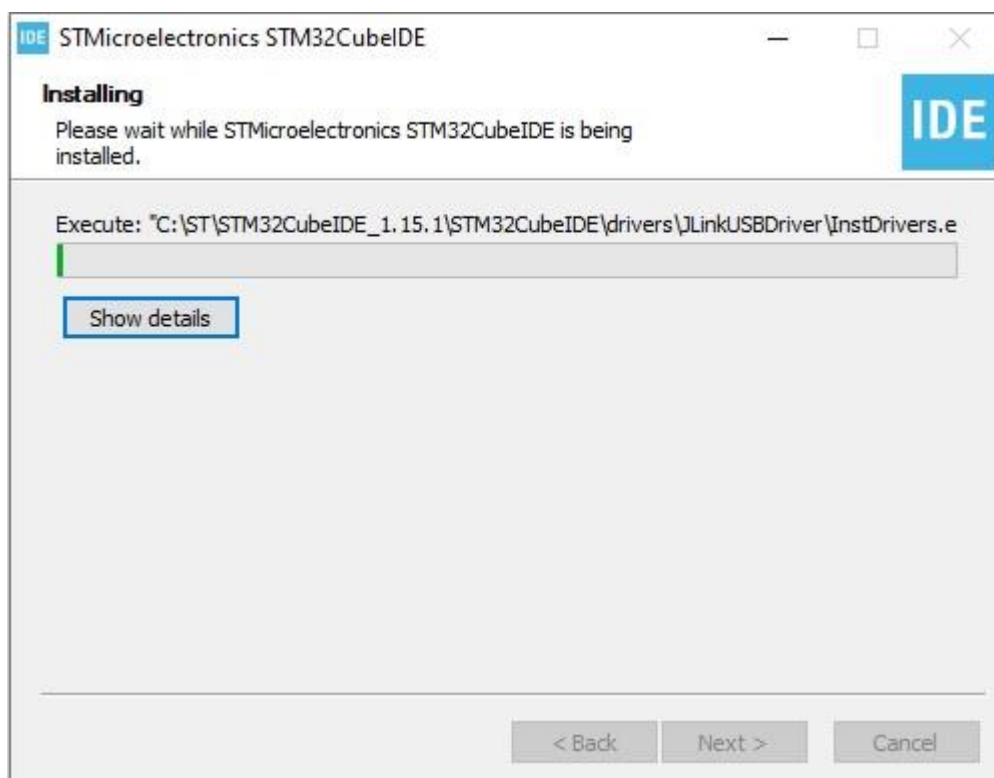
Aceptamos los términos presionando en “I Agree” y nos aparecerá el siguiente menú en donde podemos elegir la carpeta destino en donde se instalara el software.



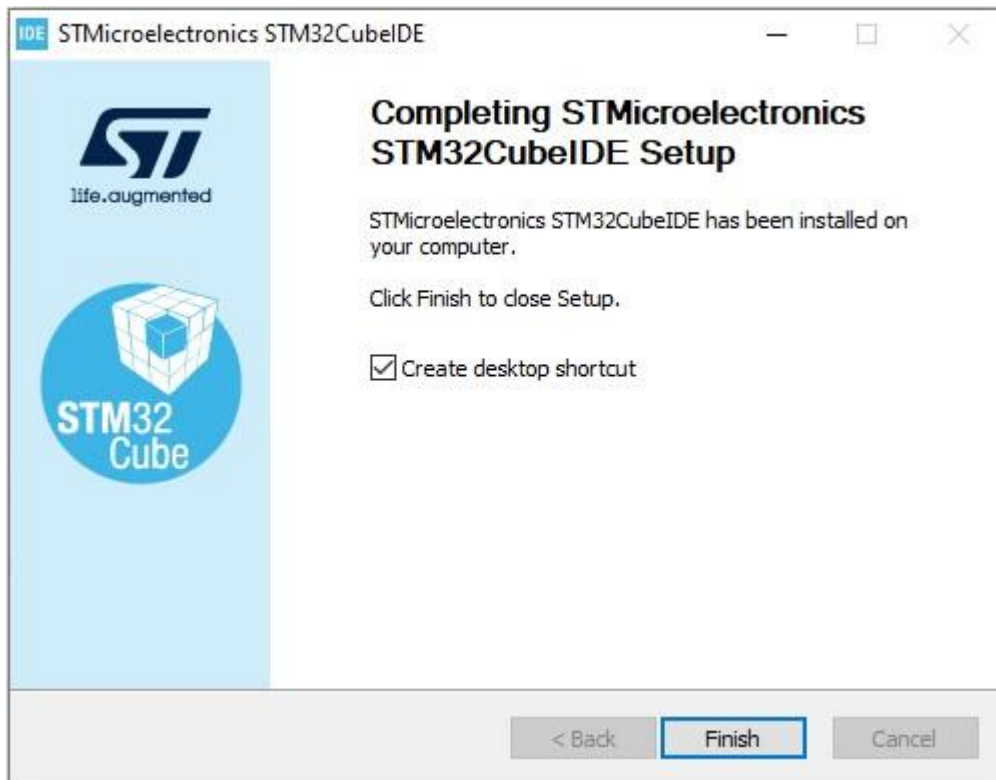
Presionamos en “Next >” y nos aparecerá el siguiente menú en donde nos permite instalar algunos complementos.



Presionamos en “Install” y comenzara el proceso de instalación.



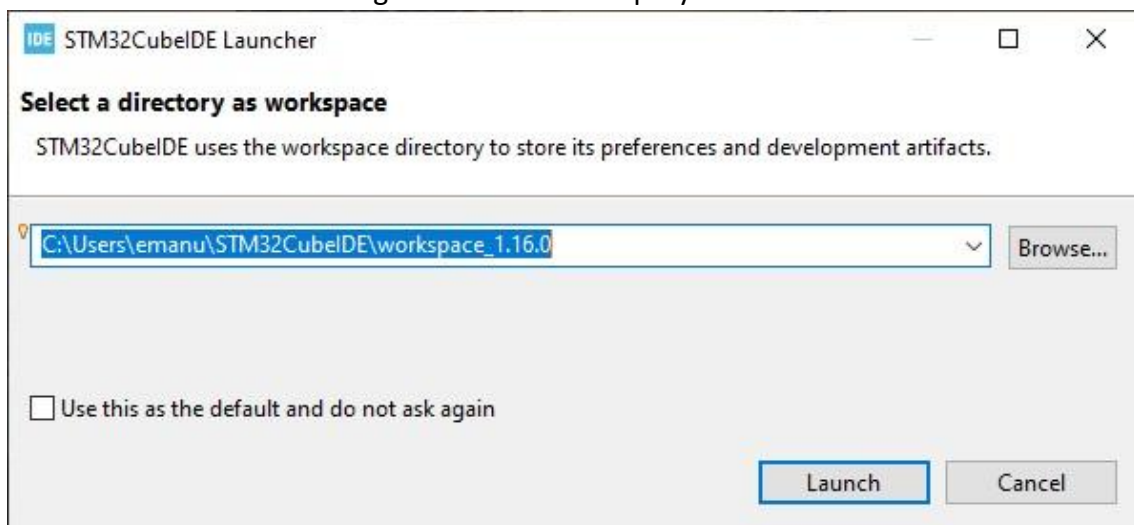
Concluida la instalación del IDE, nos aparece lo siguiente:



Tildamos la casilla "Create desktop shortcut" si deseamos que nos cree un acceso directo en el escritorio para abrir el IDE, luego presionamos en "Finish" concretando el proceso de instalación.

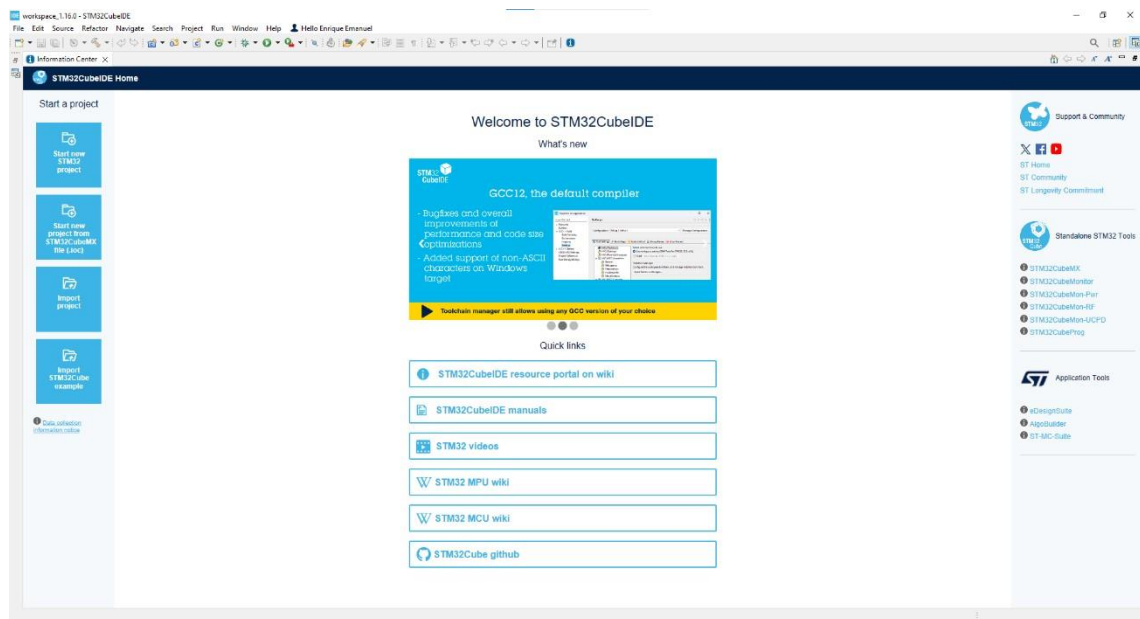
## Creación de un Nuevo Proyecto

Al abrir el IDE, debemos especificar el "workspace" (espacio de trabajo) en donde se guardarán nuestros proyectos.



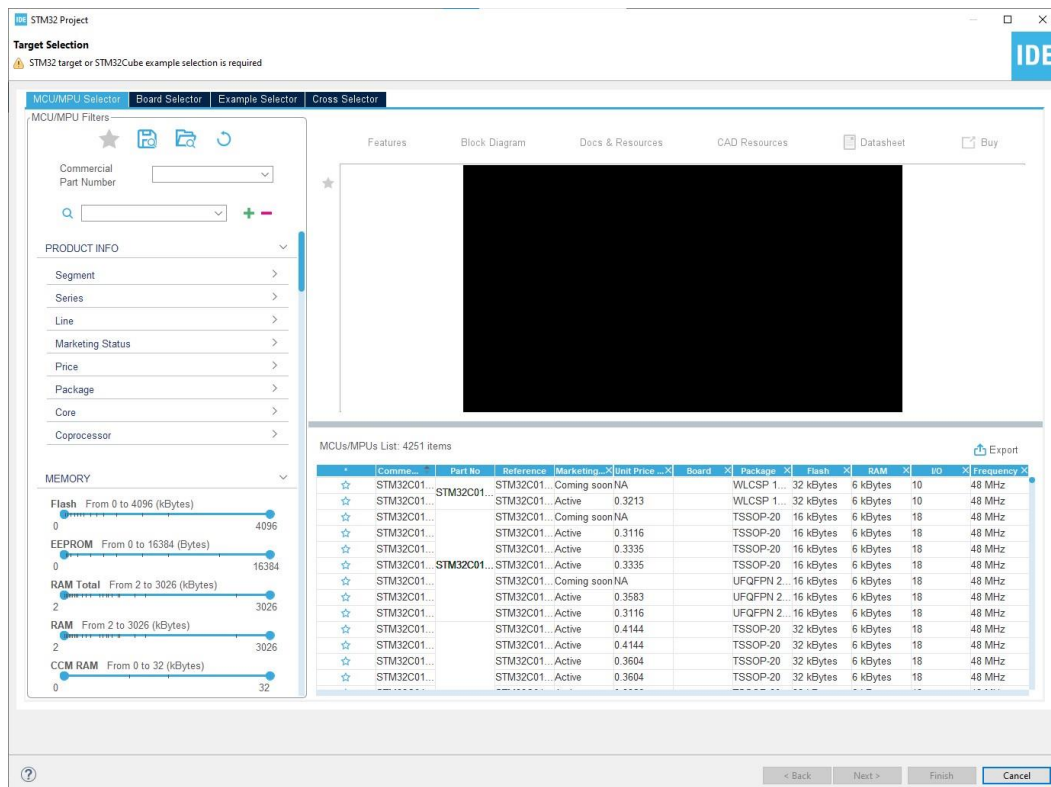
Seleccionamos una carpeta y presionamos en "Launch".

En la pagina de inicio encontramos distintas opciones como comenzar e importar un proyecto. Tambien algunos links a videos, manuales y documentacion sobre el IDE.

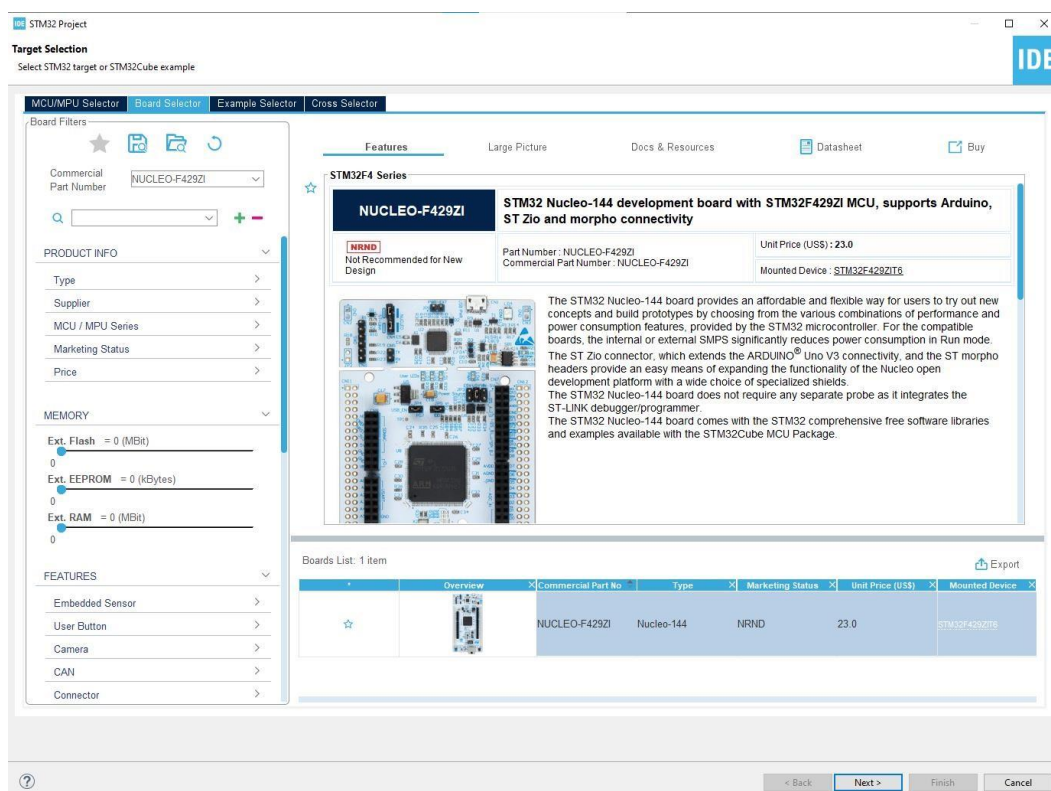


Para iniciar un nuevo proyecto presionamos “Start new STM32 project”.

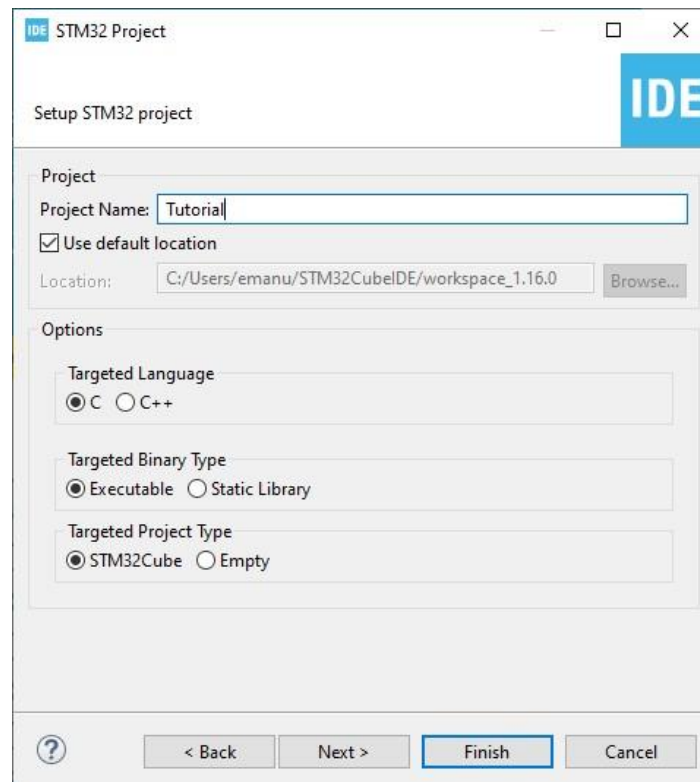




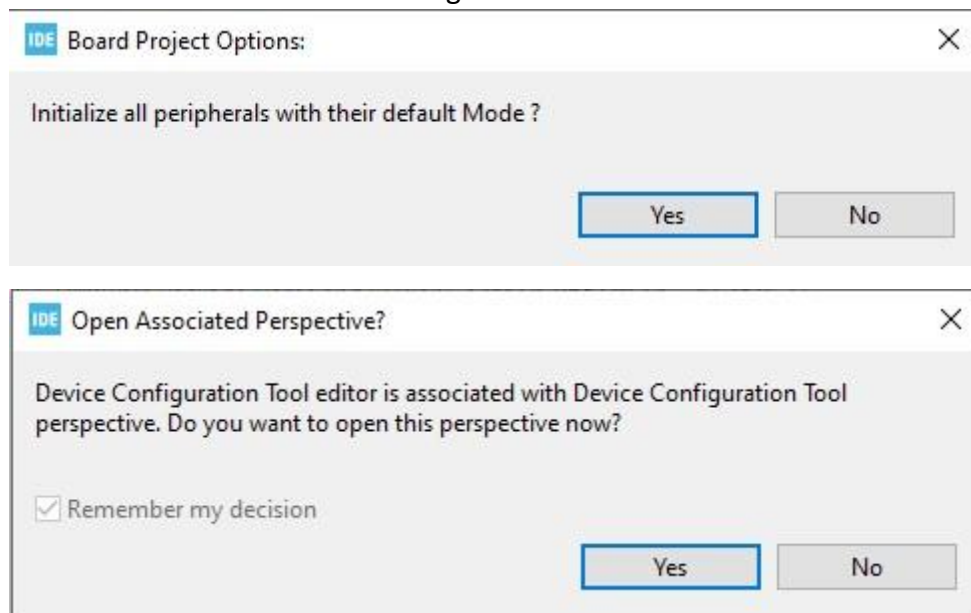
En esta ventana debemos seleccionar el microcontrolador o placa de desarrollo que vamos a utilizar en el proyecto.



Una vez que elegimos la tarjeta de desarrollo, pasamos a darle un nombre al proyecto.



Presionamos "Finish". Nos saltaran las siguiente ventana:

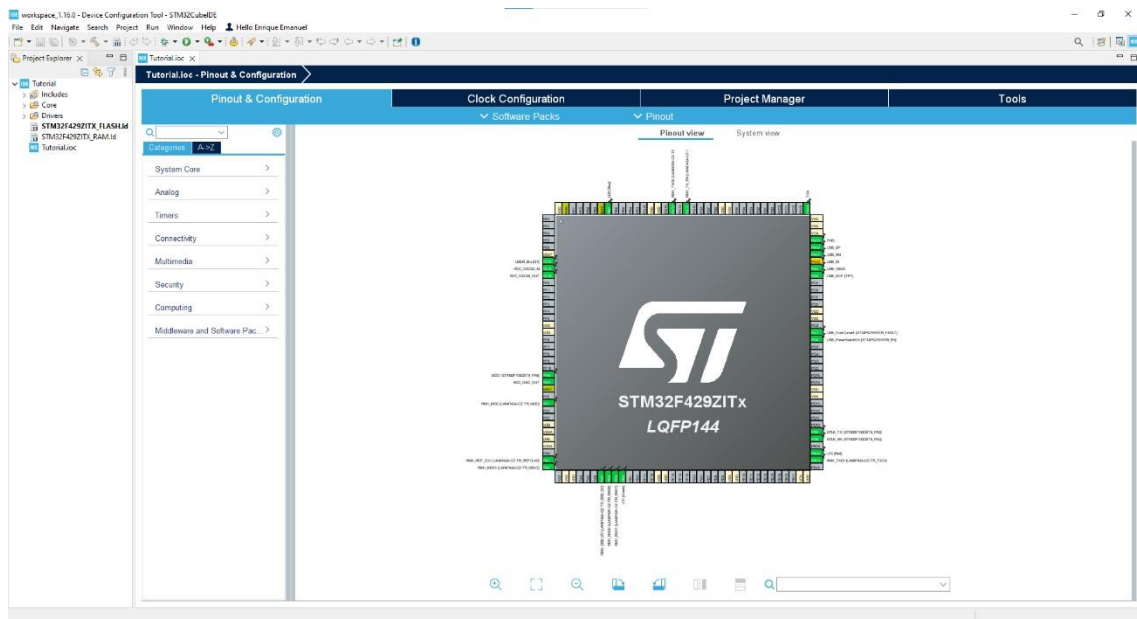


Confirmamos ambas ventanas y se procede a abrir el archivo de extensión "ioc".

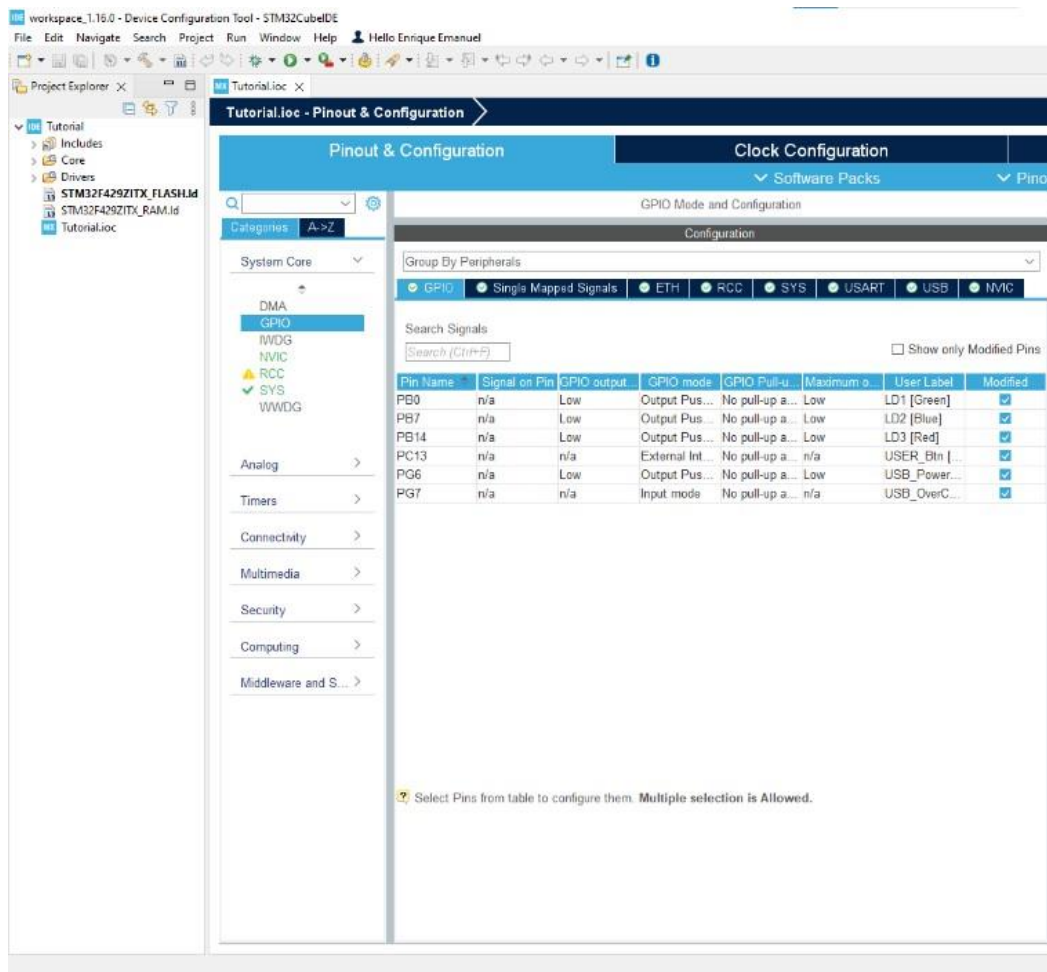
En el STM32CubeIDE, un archivo con extensión "ioc" (que significa "Input Output Configuration") es un archivo de configuración que define los periféricos, pines y configuraciones de hardware para un proyecto específico de STM32. Este archivo es generado por STM32CubeMX, que es una herramienta de configuración gráfica integrada en el STM32CubeIDE.

El IOC permite al usuario configurar los pines del microcontrolador, habilitar y configurar periféricos como UART, SPI, I2C, ADC, etc. También incluye la configuración

del sistema de reloj del microcontrolador, como la selección de la fuente del reloj y las divisiones para obtener las frecuencias deseadas.



En panel lateral podemos encontrar todas estas configuraciones.

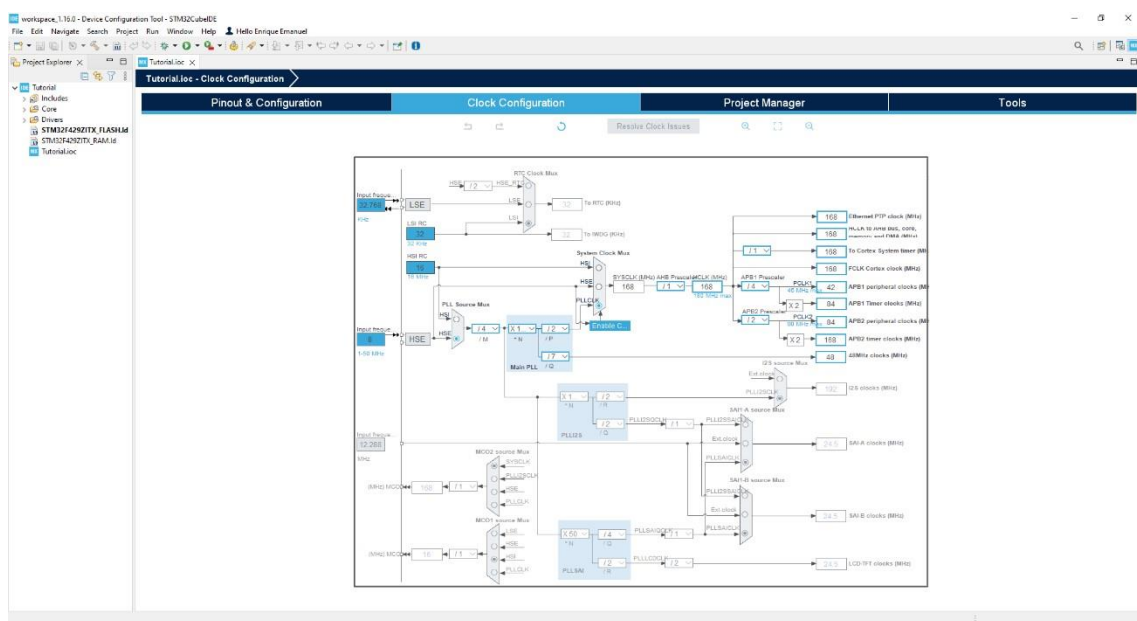


Por defecto al seleccionar la tarjeta de desarrollo adecuadamente, permite que el software por defecto seleccione automáticamente los pines conectados a los leds y a

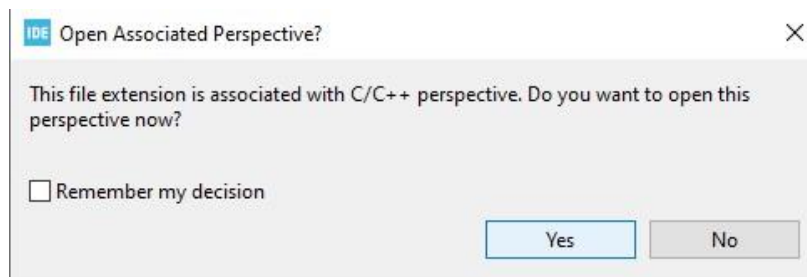
los botones de la placa. En caso de no ser así, o si en nuestro proyecto necesitamos configurar algo que no se encuentra configurado por defecto, se lo realiza en este panel.

Es recomendable realizar todas las configuraciones antes de empezar a escribir el código y no cambiarlo en todo el proceso. Esto debido a que una vez que la configuración está completa, el archivo ".ioc" procede a generar el código inicial de configuración y los archivos de cabecera necesarios para el proyecto. Esto incluye inicializar los periféricos y configurar los pines según lo especificado en el archivo.

Como siguiente paso se debe configurar la frecuencia del reloj. Al igual que en el menú anterior, si la tarjeta de desarrollo se selecciona adecuadamente, esto se configura correctamente por defecto.



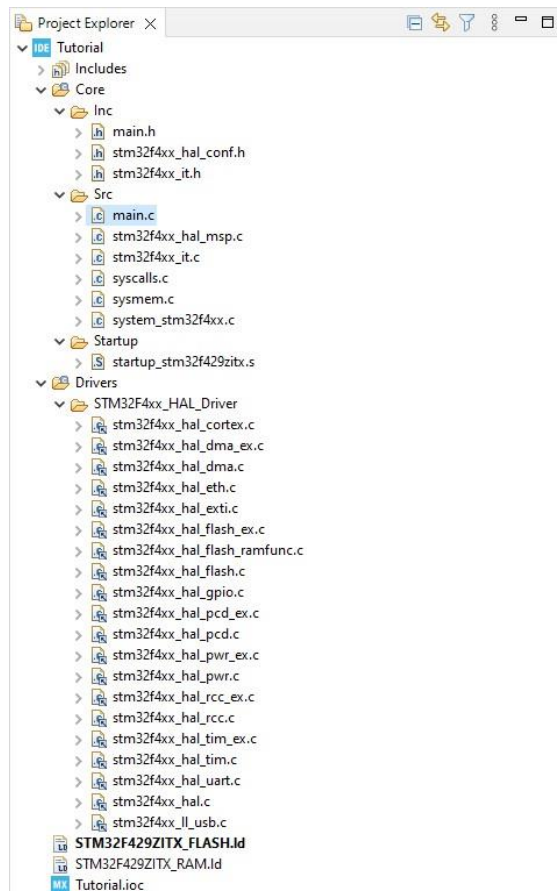
Al guardar la configuración del IOC, nos saltará la siguiente ventana:



Al confirmar esta ventana se genera el código correspondiente.

También se generan distintas carpetas:

- **Includes:** Es una carpeta que se suele configurar como ruta de inclusión para los archivos de cabecera (.h).
- **Core:** Esta carpeta contiene archivos esenciales para el núcleo del proyecto, que son fundamentales para el arranque y funcionamiento del microcontrolador.
  - ✚ **Src:** En esta carpeta se encuentran los archivos fuente (.c) que incluyen el código de inicialización del sistema, el archivo “main.c” y otros archivos de código que gestionan el núcleo del sistema.
  - ✚ **Inc:** En esta carpeta se encuentran los archivos de cabecera (.h) que definen funciones, macros y tipos de datos esenciales para la configuración del sistema y el arranque del microcontrolador. Incluye el archivo “main.h” en donde (entre otras cosas) se encuentran configuradas las etiquetas de los pines configurados por el “.ioc”.
- **Drivers:** Contiene los controladores de hardware necesarios para interactuar con los periféricos del microcontrolador.
  - ✚ **HAL:** Esta carpeta incluye los controladores HAL generados para los periféricos configurados en el archivo “.ioc”. La HAL (Hardware Abstraction Layer) es una capa de abstracción de hardware que proporciona funciones para interactuar con los periféricos de manera independiente del hardware específico, esto permite más portabilidad al código.



## Proyecto de ejemplo: “Parpadeo de un LED”

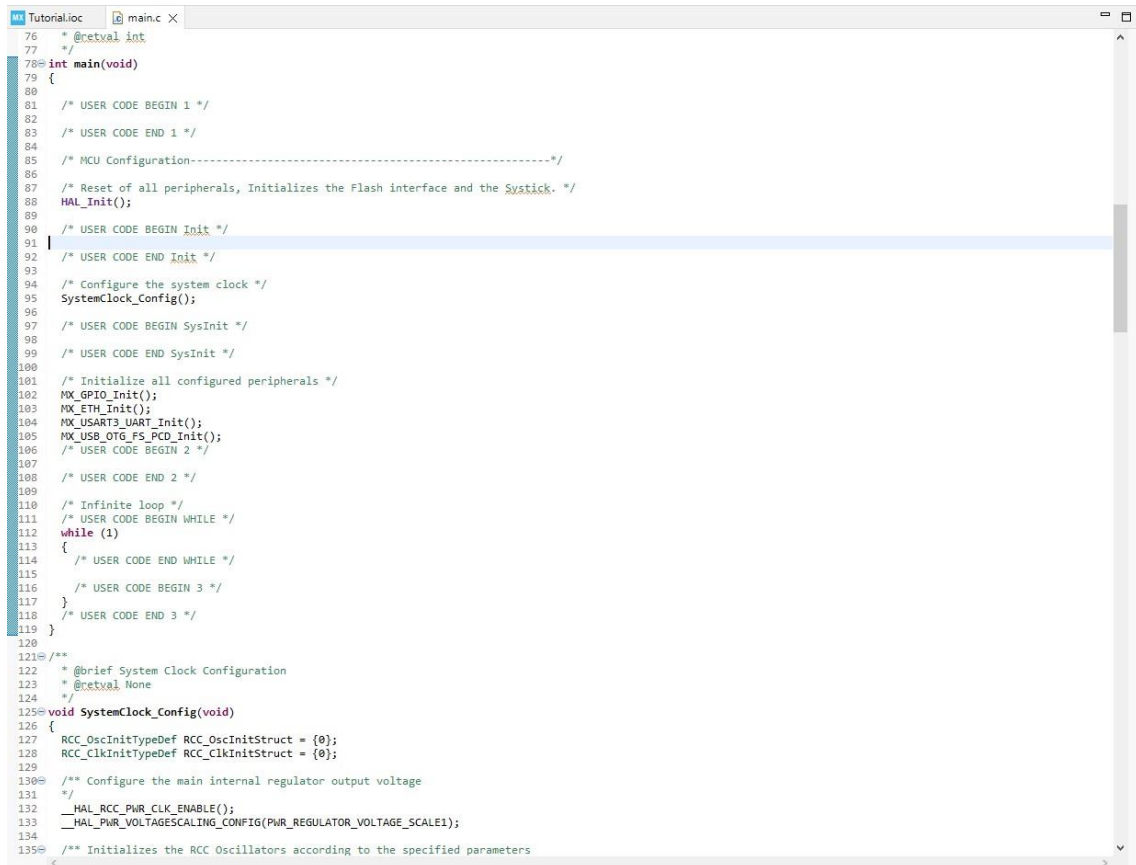
Ahora vamos a desarrollar un ejemplo de proyecto:

Una aplicación que haga parpadear cada 250 milisegundos un led integrado en la tarjeta de desarrollo.

## Escritura del Código

Luego de haber configurado correctamente el “.ioc” y habiendo colocado una etiqueta al pin en donde se encuentra el led integrado en la placa, se genera el código.

Abrimos el archivo “main.c” ubicado en la ruta “/Core/Src/main.c”. Encontraremos el código generado de la siguiente forma:



```
76  * @retval int
77  */
78  int main(void)
79  {
80
81  /* USER CODE BEGIN 1 */
82
83  /* USER CODE END 1 */
84
85  /* MCU Configuration-----*/
86
87  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
88  HAL_Init();
89
90  /* USER CODE BEGIN Init */
91
92  /* USER CODE END Init */
93
94  /* Configure the system clock */
95  SystemClock_Config();
96
97  /* USER CODE BEGIN SysInit */
98
99  /* USER CODE END SysInit */
100
101  /* Initialize all configured peripherals */
102  MX_GPIO_Init();
103  MX_ETH_Init();
104  MX_USART3_UART_Init();
105  MX_USB_OTG_FS_PCD_Init();
106  /* USER CODE BEGIN 2 */
107
108  /* USER CODE END 2 */
109
110  /* Infinite loop */
111  /* USER CODE BEGIN WHILE */
112  while (1)
113  {
114      /* USER CODE END WHILE */
115
116      /* USER CODE BEGIN 3 */
117  }
118  /* USER CODE END 3 */
119  }
120
121  /**
122   * @brief System Clock Configuration
123   * @retval None
124   */
125  void SystemClock_Config(void)
126  {
127      RCC_OscInitTypeDef RCC_OscInitStruct = {0};
128      RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
129
130      /** Configure the main internal regulator output voltage
131       */
132      __HAL_RCC_PWR_CLK_ENABLE();
133      __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
134
135      /** Initializes the RCC Oscillators according to the specified parameters
```

La aplicación empieza por la función “main”, dentro de esta función se encuentra un bucle infinito:

```
110  /* Infinite loop */
111  /* USER CODE BEGIN WHILE */
112  while (1)
113  {
114      /* USER CODE END WHILE */
115
116      /* USER CODE BEGIN 3 */
117  }
118  /* USER CODE END 3 */
```

Lo que se encuentre dentro de este bucle, se ejecutara infinitamente mientras no se genere una interrupción o cese la alimentación de la tarjeta de desarrollo.

Dentro de este bucle, es donde escribiremos el código ya que necesitamos que nuestra secuencia se ejecute de forma infinita.

Es recomendable escribir nuestro código entre los comentarios:



```
>> /* USER CODE BEGIN */
```

```
>> /* USER CODE END */
```

Ya que lo que se escriba dentro de estos comentarios se conservaran en el caso de tener que realizar un cambio en el archivo “.ioc”. Recordemos que al editar este archivo se genera un código nuevo.

Para cambiar el estado del LED, usaremos dos de las funciones de la HAL.

Una es la función “HAL\_GPIO\_TogglePin” del módulo GPIO (General Purpose Input Output), que lo que hace es invertir el estado actual de una salida digital, esto es, si la salida se encuentra en estado alto, con esta función pasaría a estar en estado bajo, y viceversa. Esta función toma como parámetros de entrada dos datos: uno es el puerto, otro es el pin.

La otra función es “HAL\_Delay”, la función “HAL\_Delay” se utiliza para introducir una pausa o retardo en la ejecución del programa. Esta función permite que el programa espere una cantidad específica de milisegundos antes de continuar con la siguiente instrucción. Esta función toma como parámetro de entrada la cantidad de milisegundos que dura el retardo.

```
110  /* Infinite loop */
111  /* USER CODE BEGIN WHILE */
112  while (1)
113  {
114      HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
115      HAL_Delay(250);
116
117      /* USER CODE END WHILE */
118
119      /* USER CODE BEGIN 3 */
120  }
121  /* USER CODE END 3 */
122 }
```

## Compilación del proyecto

Para compilar el proyecto debemos ir al icono del martillo ubicado en el menú superior, y desplegar las opciones.

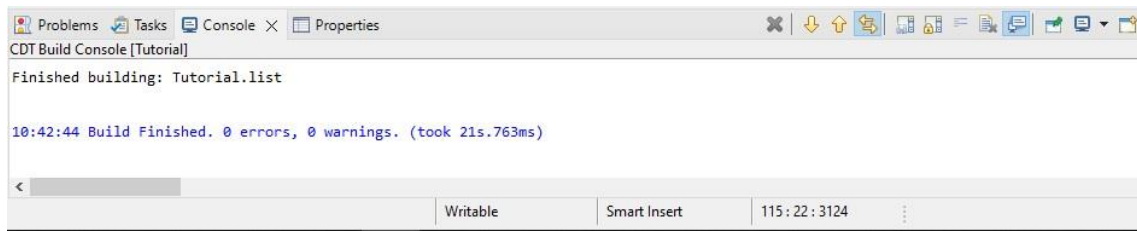


El modo “Debug” está diseñado para facilitar la depuración del programa.

El modo “Release” está diseñado para la ejecución final del programa en un entorno de producción.

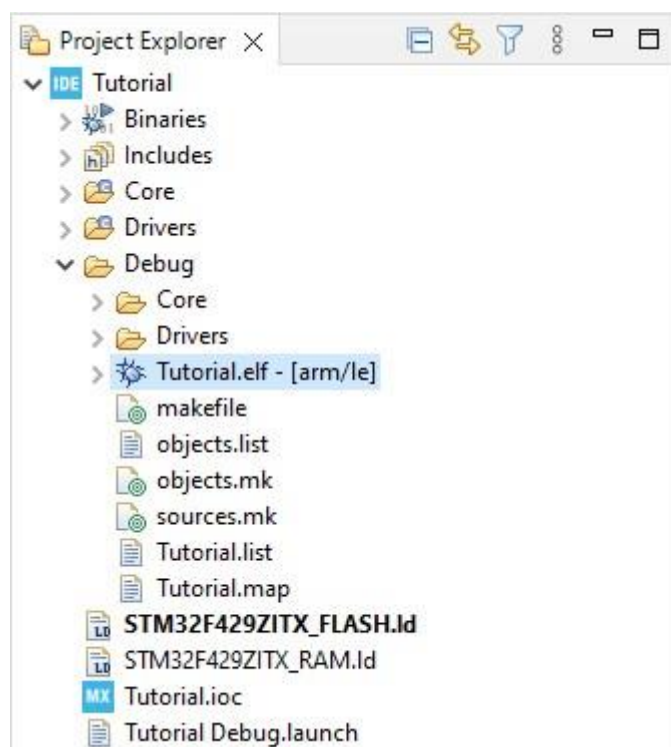
Al presionar en el icono del martillo comienza a compilarse el programa.

Al finalizar la compilación, en la parte inferior encontraremos la consola, donde nos aparecerá lo siguiente:



Si la compilación falla por algún error en nuestro código, esta consola nos indicará en que línea se encuentra el error y una descripción breve del error. Debemos corregir todos los errores para que nuestro código pueda compilarse. Los errores que se corrigen en la fase de compilación son errores de sintaxis.

Luego de compilar el programa, se crea un archivo de extensión “.elf” ubicado en la carpeta “Debug”.



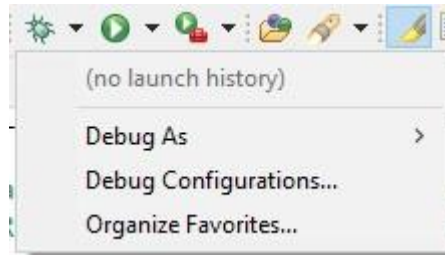
Este archivo contiene el código máquina que ejecutará el microcontrolador, junto con información adicional de depuración, secciones de datos y metadatos del programa.

## Debug

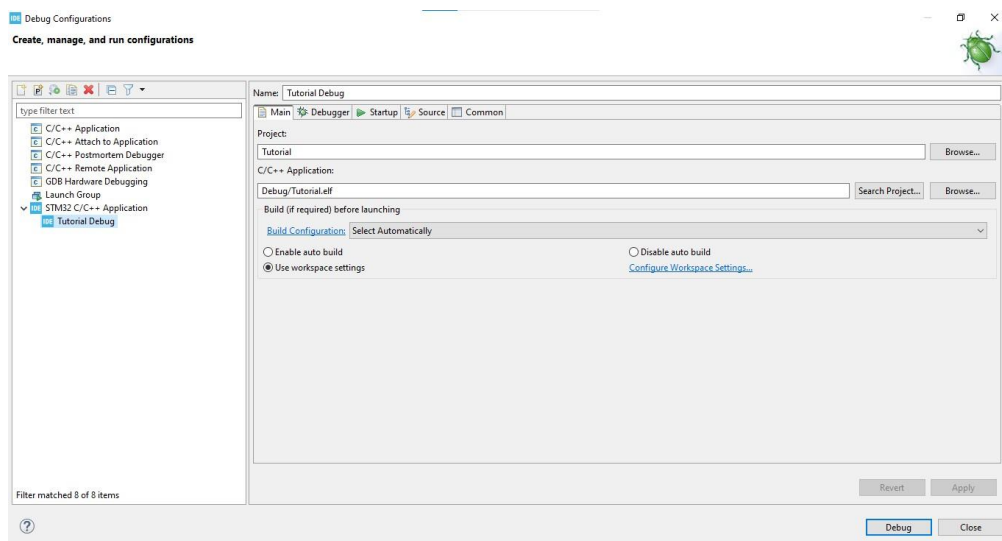
El objetivo principal de la depuración es garantizar que el programa funcione como se espera, eliminando o corrigiendo cualquier problema que pueda afectar su rendimiento o funcionalidad. En este proceso se analizan los posibles errores lógicos, fallos de ejecución y problemas de rendimiento.



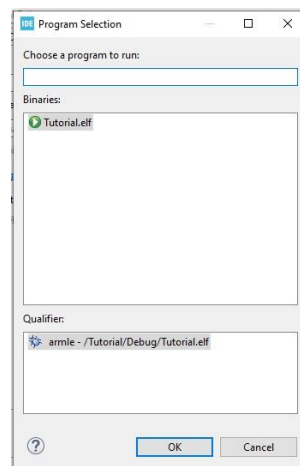
Para Depurar el programa nos dirigimos al icono “Debug”, desplegamos el menú, y vamos a “Debug Configurations...”.



Se nos muestra la siguiente ventana:

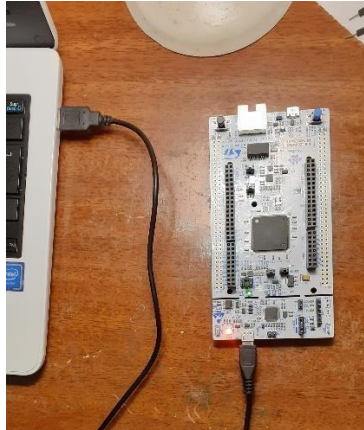


En esta ventana, presionamos en “STM32 C/C++ Application”, y seleccionamos la aplicación que acabamos de programar. Luego presionamos en “Search Project...” y seleccionamos el “.elf” que se creó en el proceso de compilación.

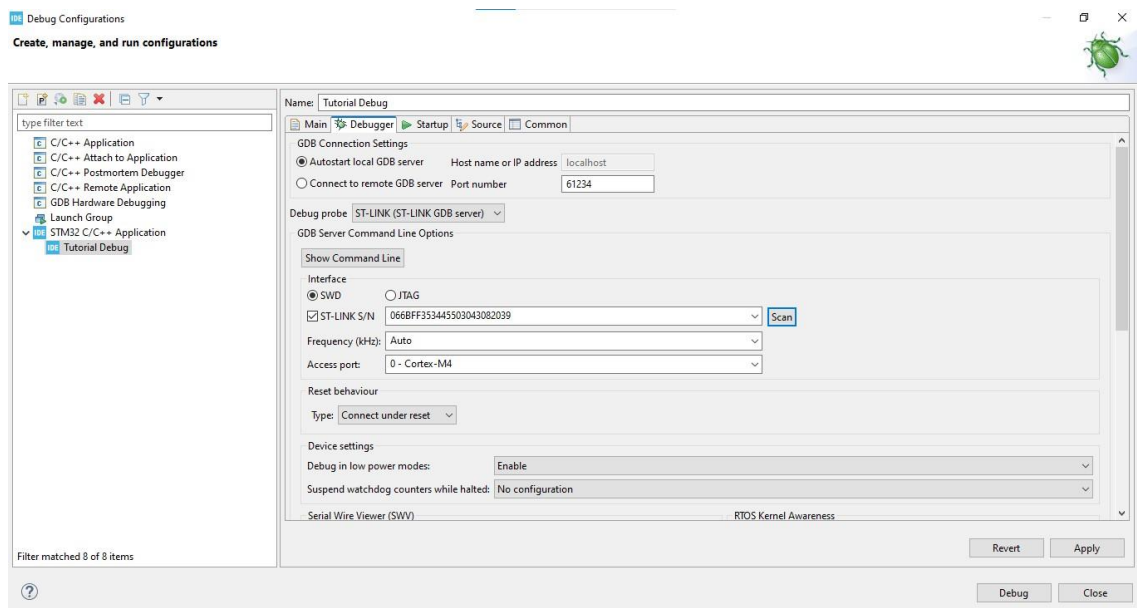


Nos vamos a la siguiente pestaña “Debugger”, donde configuraremos el ST-LINK, que es una herramienta de depuración y programación diseñada por STMicroelectronics para microcontroladores STM32. Su función principal es permitir a los desarrolladores cargar código en los microcontroladores STM32 y depurar aplicaciones durante el desarrollo.

Procedemos a conectar la tarjeta de desarrollo a nuestra PC.



Luego en la pestaña “Debugger” presionamos en el boton “Scan” para que se genere la conexión.

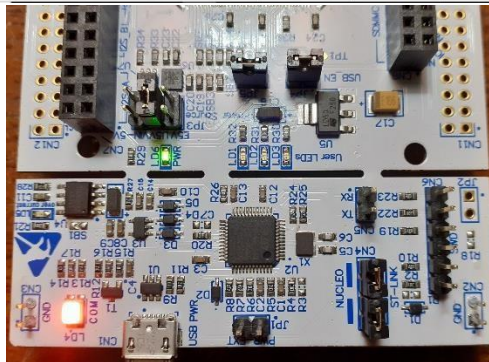
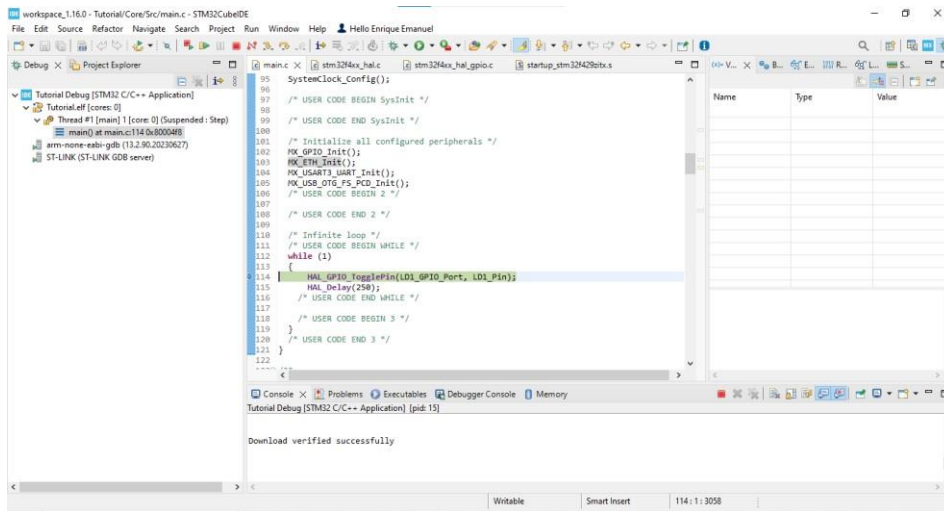


Una vez conectada la tarjeta presionamos en “Apply”, y luego en “Debug”, para comenzar la depuración del código.

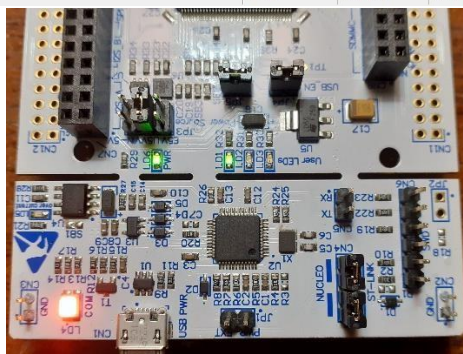
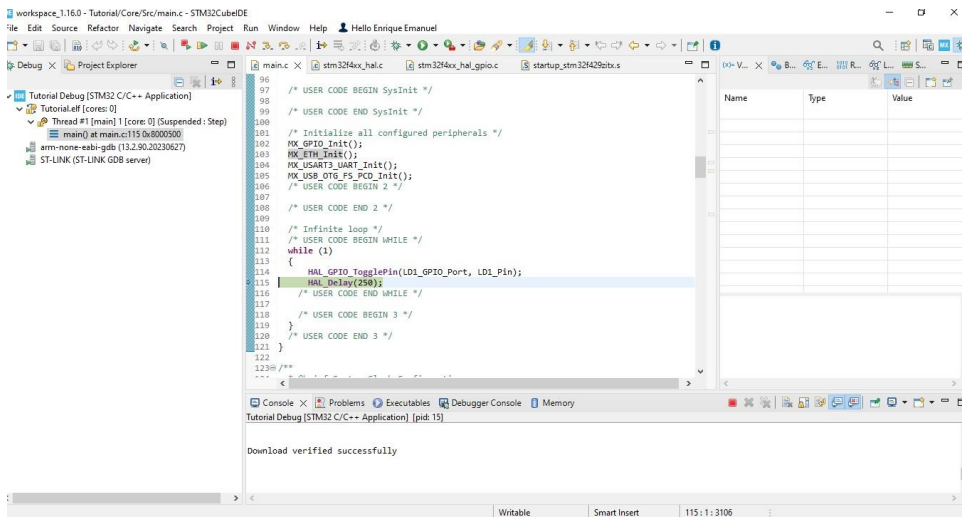
Durante la depuración del código nos aparecerá una ventana en donde podremos ver el valor de cada variable en cada momento del programa.

Para empezar a ejecutar las instrucciones paso a paso, presionamos F6.

Antes de ejecutar la instrucción “HAL\_GPIO\_TogglePin” por primera vez, el led se encuentra apagado.



Luego al ejecutar la instrucción el led en la tarjeta debería cambiar de estado a Encendido.



## Enlaces

Repositorio del Grupo 3:

[EmanuelDecima/Grupo 3 TDII 2024](#)

App 1.1:

[Grupo 3 TDII 2024/AFP 1 TDII 2024/AFP 1 Grupo 3 App 1.1 at main ·](#)

[EmanuelDecima/Grupo 3 TDII 2024](#)

App 1.2:

[Grupo 3 TDII 2024/AFP 1 TDII 2024/AFP 1 Grupo 3 App 1.2 at main ·](#)

[EmanuelDecima/Grupo 3 TDII 2024](#)

App 1.3:

[Grupo 3 TDII 2024/AFP 1 TDII 2024/AFP 1 Grupo 3 App 1.3 at main ·](#)

[EmanuelDecima/Grupo 3 TDII 2024](#)

App 1.4:

[Grupo 3 TDII 2024/AFP 1 TDII 2024/AFP 1 Grupo 3 App 1.4 at main ·](#)

[EmanuelDecima/Grupo 3 TDII 2024](#)