



**Facultad Regional Tucumán**

**Departamento de Electrónica**

## Técnicas Digitales II

# Actividad de Formación Práctica 1

Tema: Entorno de desarrollo integrado STM32CubeIDE  
Programación de microcontroladores

Profesor:  
Ing. Rubén Darío Mansilla

ATTP:  
Ing. Lucas Abdala

Grupo 8:  
Gao Luciano, Mitre Emilio, Jiménez Emanuel, Jorrat Tomas.

Vencimiento:  
9 de agosto de 2024

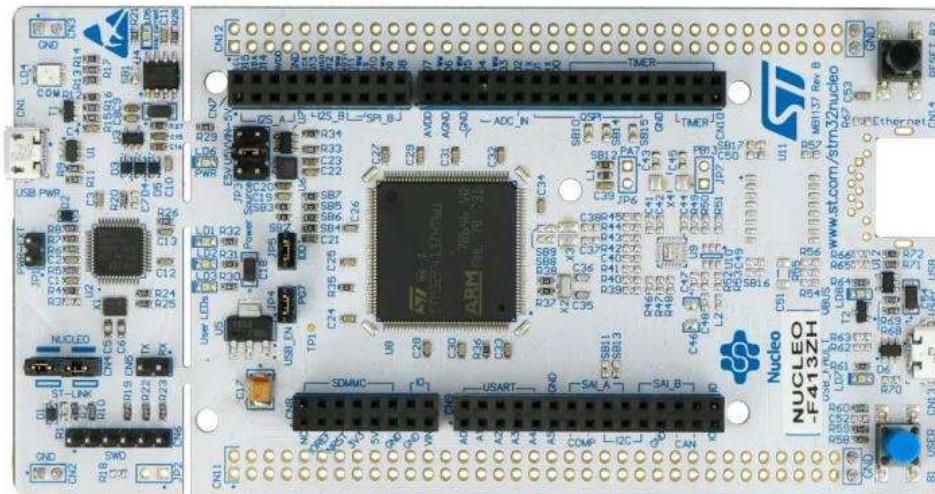
**Registros de cambios**

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	19 de junio de 2024

## Introducción:

Dentro de este curso se trabaja con el entorno de desarrollo STM32CubeIDE, el cual nos permitirá trabajar desarrollando programas y aplicaciones para microcontroladores STM32 de arquitectura ARM, en particular trabajaremos con las placas de desarrollo NUCLEO-F4XX.

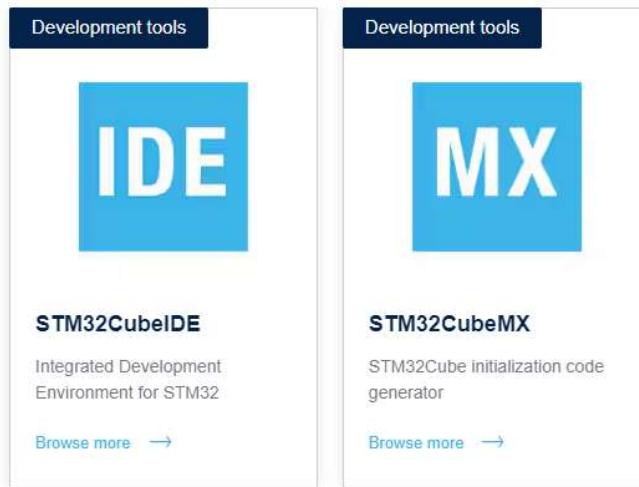
El objetivo de este trabajo es realizar una guía de inicialización en el entorno de desarrollo, tanto en el proceso de instalación como en la creación de un proyecto. La placa que usaremos como ejemplo es el modelo F413ZH.



## Guía de instalación:

Lo primero que tendremos que hacer es descargar el IDE (Integrated Development Environment), para ello debemos ingresar a la página web oficial de STMicroelectronics en la sección de herramientas y software donde encontraremos la opción para descargar el programa STM32CubeIDE y STM32CubeMX, el cual es un generador de código que nos ayudará a inicializar los proyectos en base a la placa de desarrollo que seleccionemos.

Ingresando al siguiente enlace ([www.st.com](http://www.st.com)), encontraremos dos recuadros como se observa en la imagen donde deberemos seleccionar primero la opción del IDE para descargarlo.

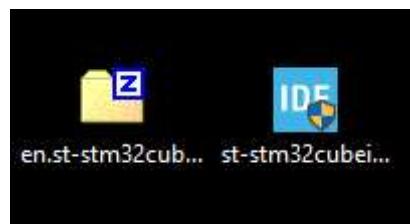


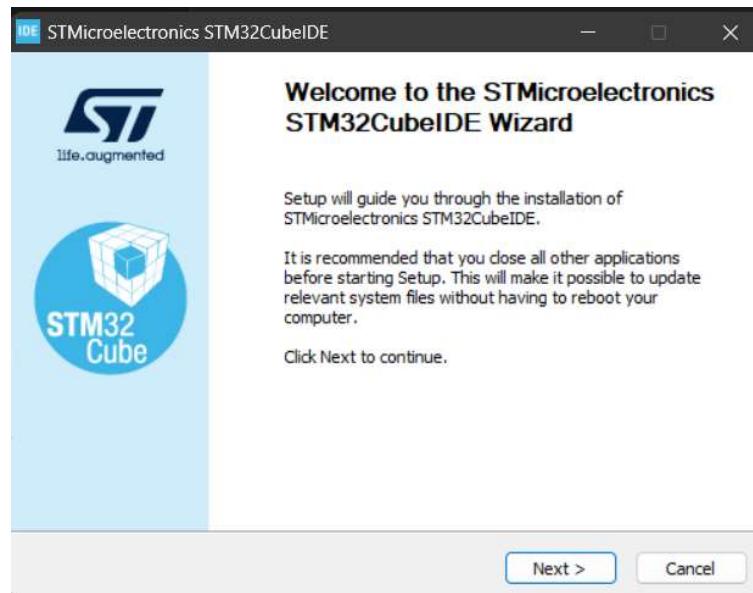
Una vez seleccionado el cuadro nos llevará a otra sección donde podremos encontrar un listado con las diferentes versiones del software según qué sistema operativo utilicemos, en este caso la explicación será para Windows por lo que deberemos elegir la última opción de la lista y seleccionar "Get latest". A partir de este paso nos pedirá aceptar los términos de uso y luego iniciar sesión o crear una cuenta en la página web para finalmente poder descargar el programa.

### Get Software

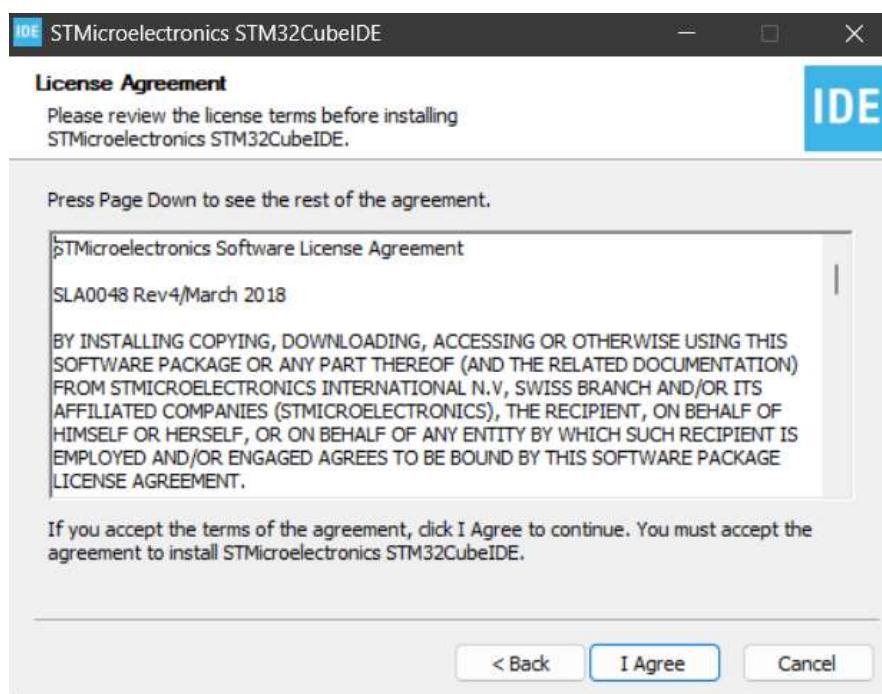
Part Number	General Description	Latest version	Download	All versions
+ STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	1.16.0	<a href="#">Get latest</a>	<a href="#">Select version</a>
+ STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	1.16.0	<a href="#">Get latest</a>	<a href="#">Select version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

Una vez que descargado el archivo este se encontrara comprimido en formato .zip el cual deberemos extraer utilizando 7zip, WinRAR o la herramienta integrada de Windows, luego deberemos inicializar el ejecutable del instalador con el icono de IDE.

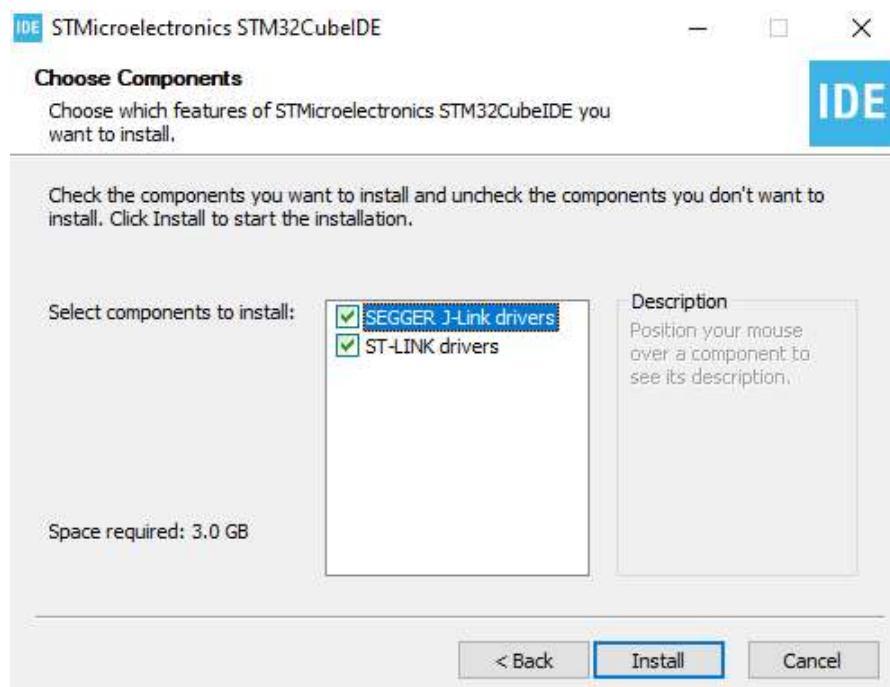
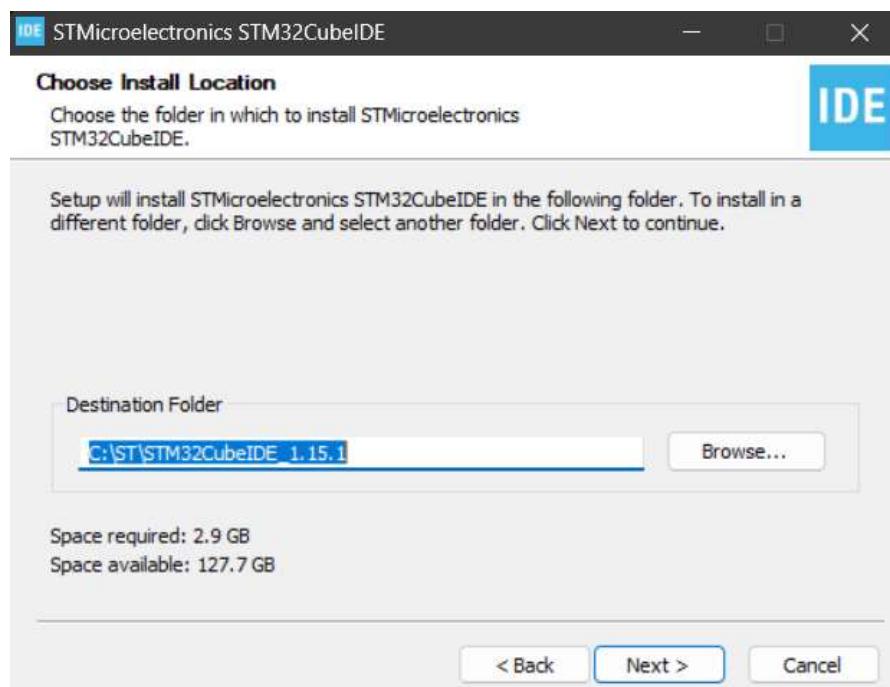




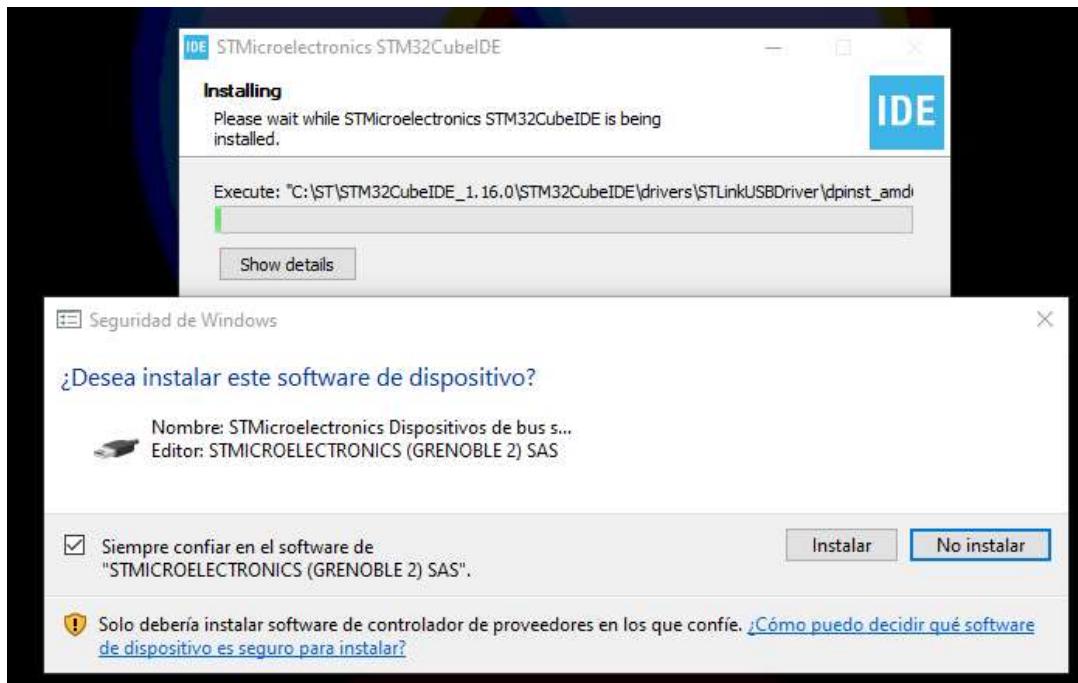
En la primera pantalla de bienvenida al instalador deberemos seleccionar Next y a continuación nos pedirá aceptar los términos y condiciones de uso.



Luego debemos seleccionar la dirección en nuestro disco donde deseemos instalar el programa y luego elegir si deseamos instalar otros drivers que nos pueden servir a futuro.



Al seleccionar la opción **Install** comenzara el proceso de instalación y posiblemente nos encontremos otro recuadro donde también deberemos seleccionar instalar.



Una vez completado el proceso de instalación seleccionamos la opción Next y finalmente nos dejara elegir si deseamos crear un acceso directo o no en el escritorio.



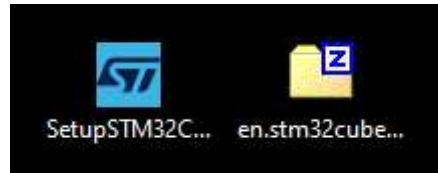
De esta manera tendremos instalado nuestro IDE. A continuación, antes de iniciar con nuestro primer proyecto deberemos instalar el otro programa mencionado al principio, el [STM32CubeMX](#).

Nos dirigimos al enlace donde entramos para descargar el IDE, pero seleccionamos el recuadro con el ícono MX. Bajamos hasta encontrar el listado de opciones para descargar y seleccionamos la última opción.

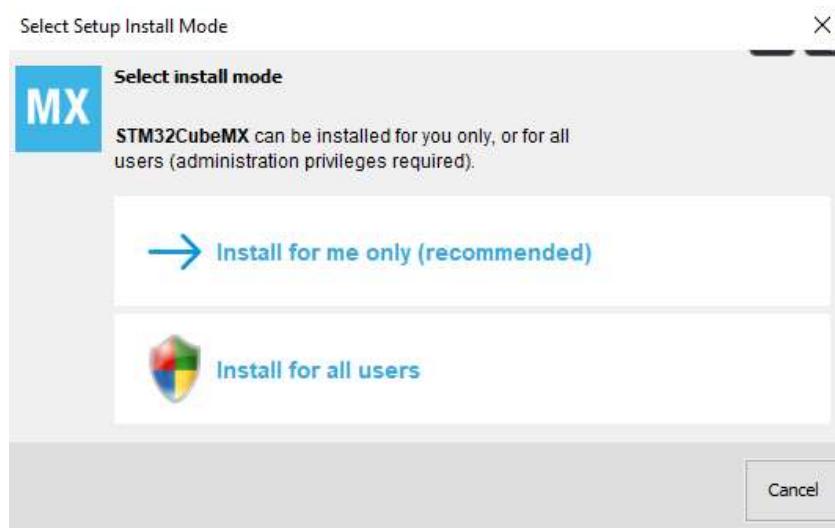
### Get Software

Part Number	General Description	Latest version	Download	All versions
+ STM32CubeMX-Lin	STM32Cube init code generator for Linux	6.12.0	<a href="#">Get latest</a>	<a href="#">Select version</a>
+ STM32CubeMX-Mac	STM32Cube init code generator for macOS	6.12.0	<a href="#">Get latest</a>	<a href="#">Select version</a>
+ STM32CubeMX-Win	STM32Cube init code generator for Windows	6.12.0	<a href="#">Get latest</a>	<a href="#">Select version</a>

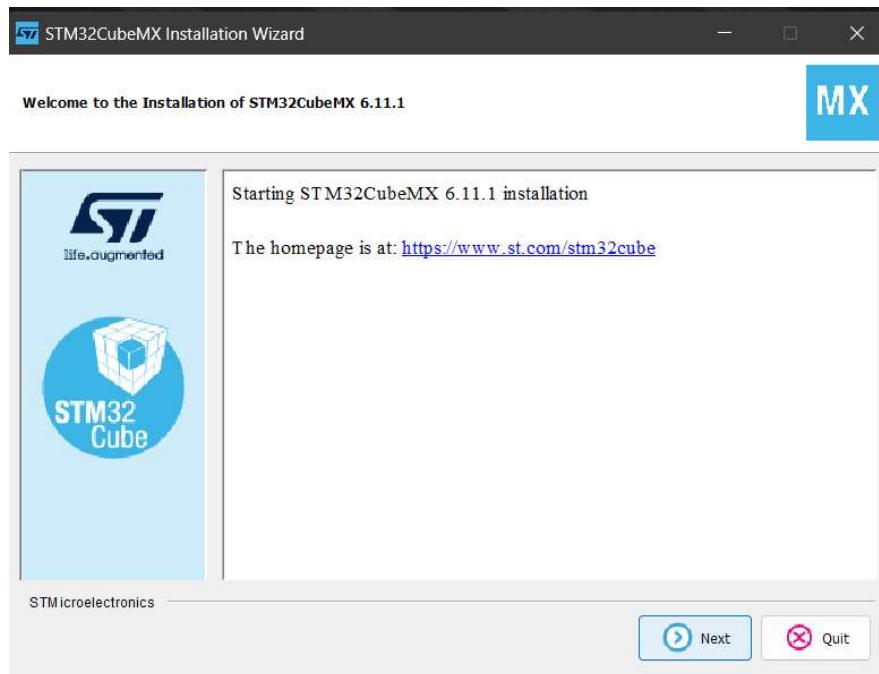
Nuevamente se descargará un archivo .zip el cual debemos descomprimir e inicializar el ejecutable con el ícono MX.

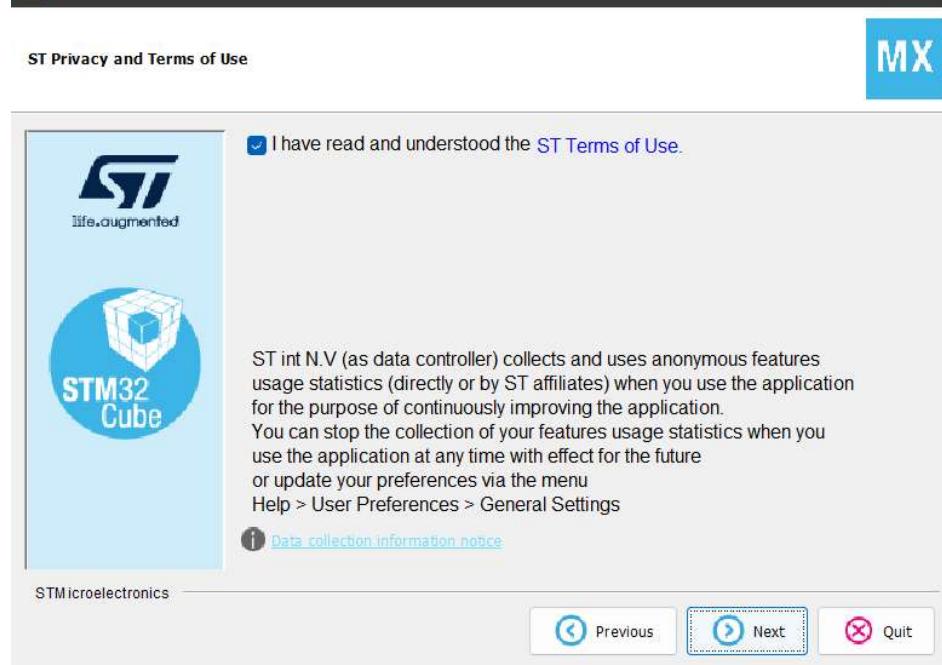
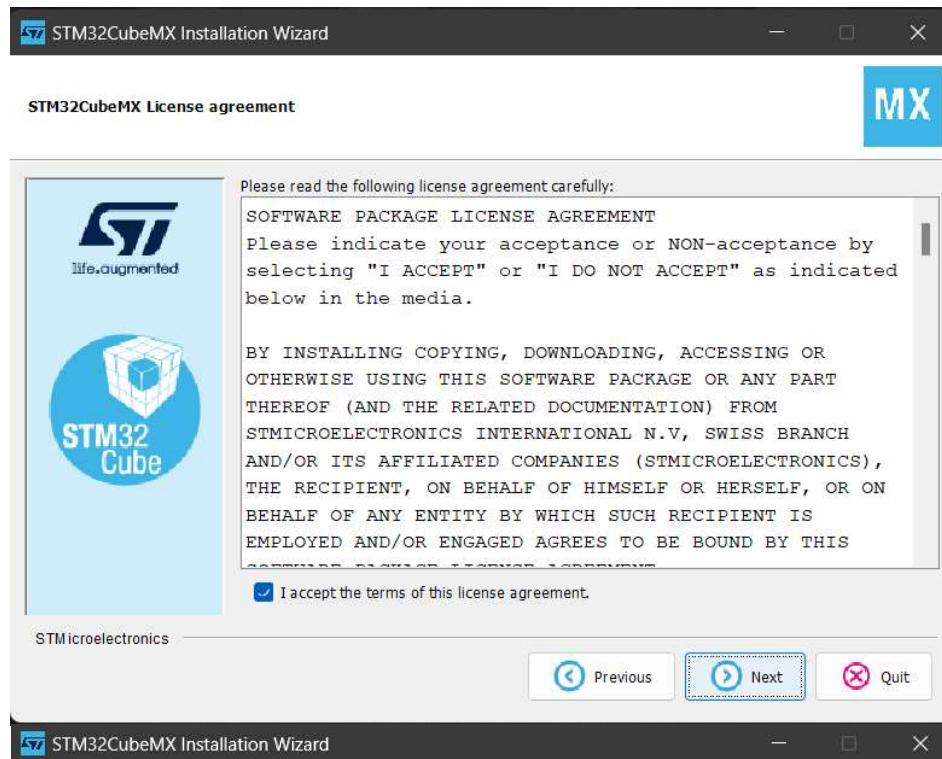


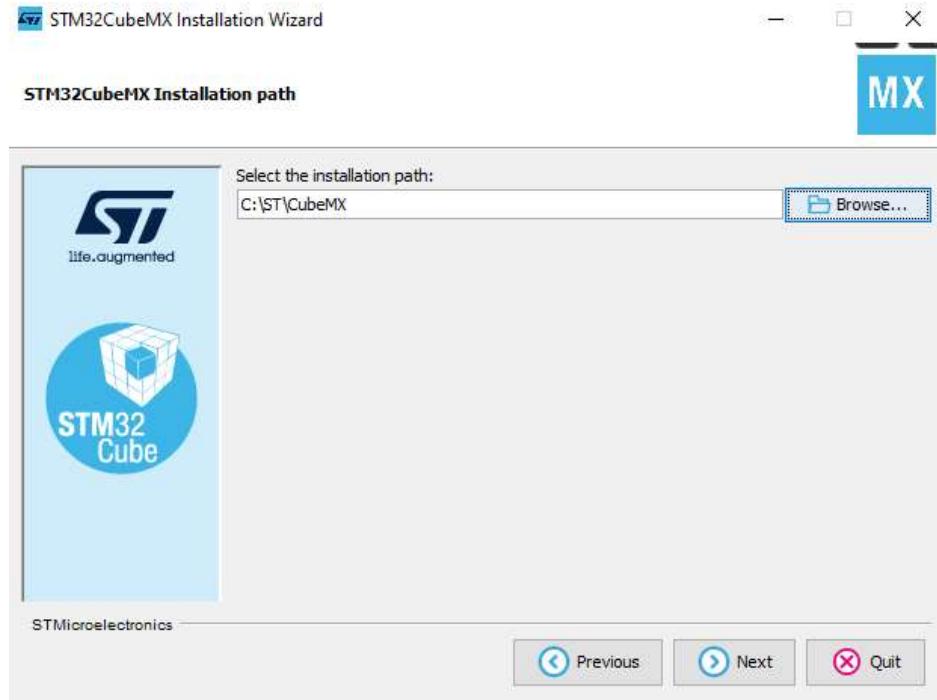
Podemos elegir si instalar el programa para un usuario o todos los usuarios del sistema en caso de tener más de uno.



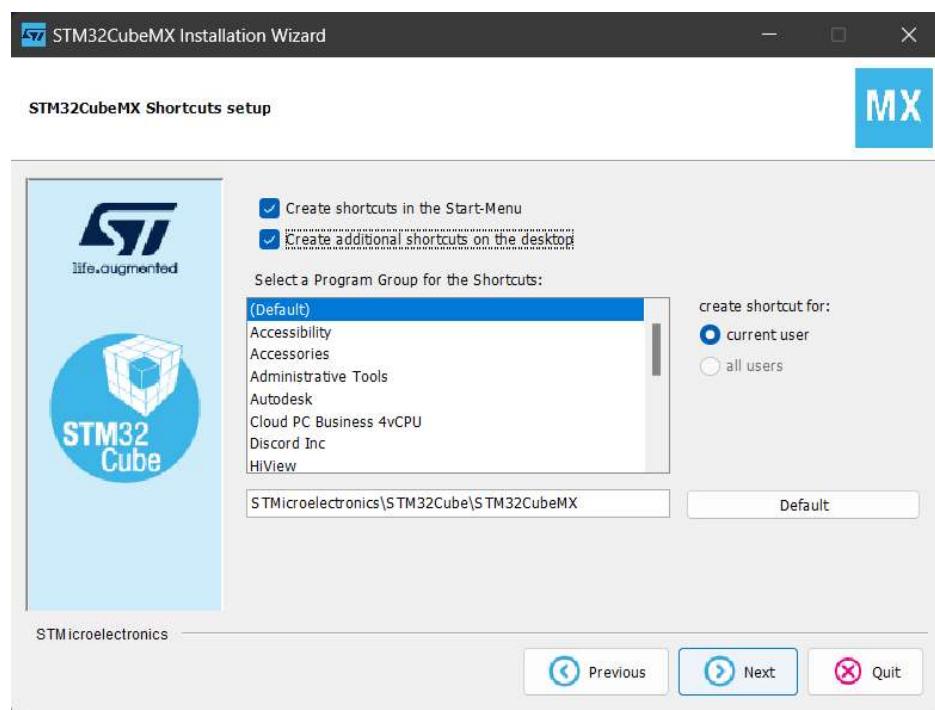
El proceso de instalación es básicamente el mismo donde deberemos aceptar los términos y condiciones de uso, seleccionar la ubicación donde instalaremos el programa (Debe ser diferente a donde instalamos el IDE).

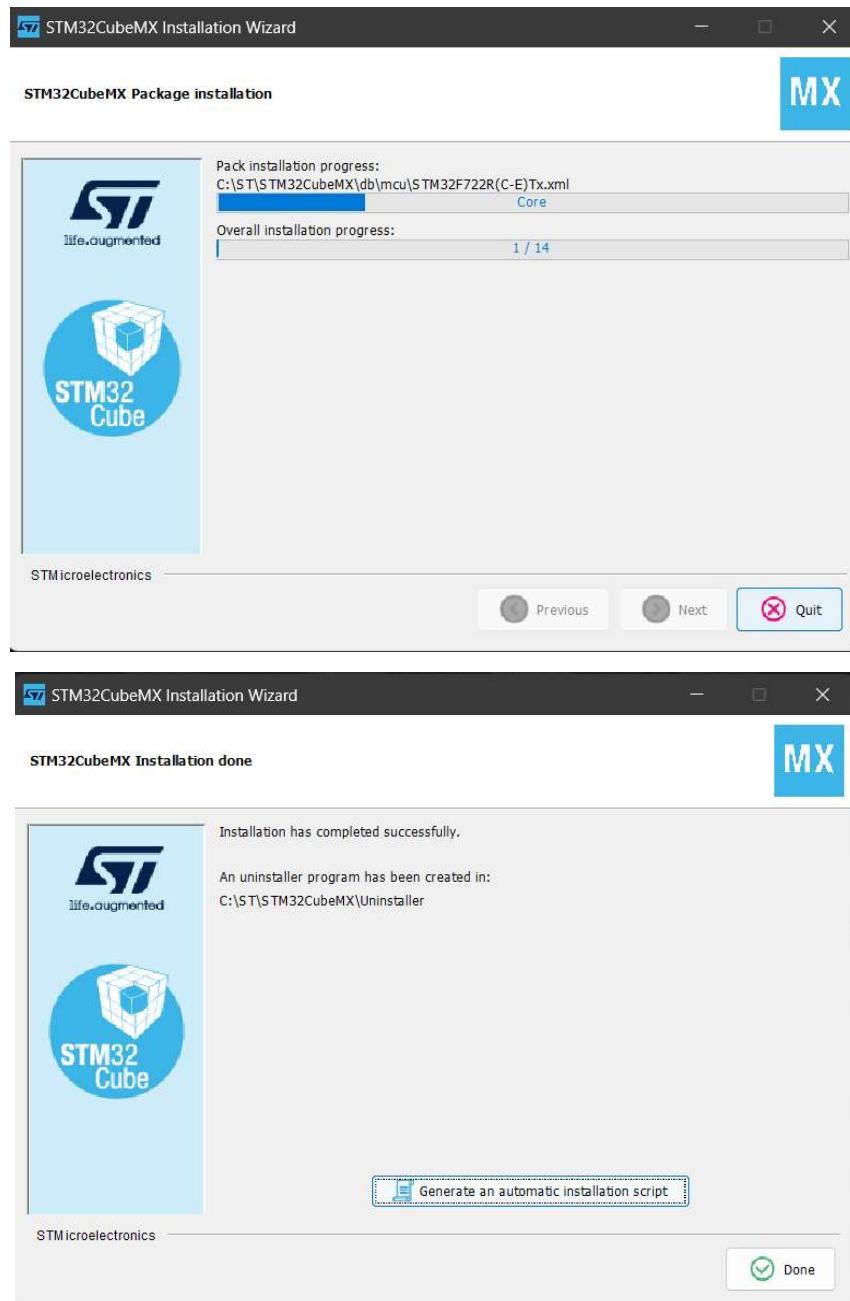




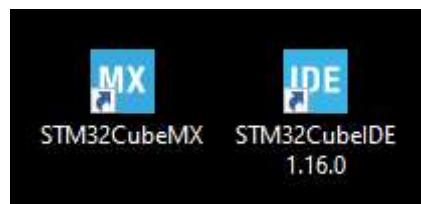


Nos pide seleccionar si deseamos crear accesos directos en el escritorio y el menú de inicio, luego al seleccionar Next comenzara el proceso de instalación.



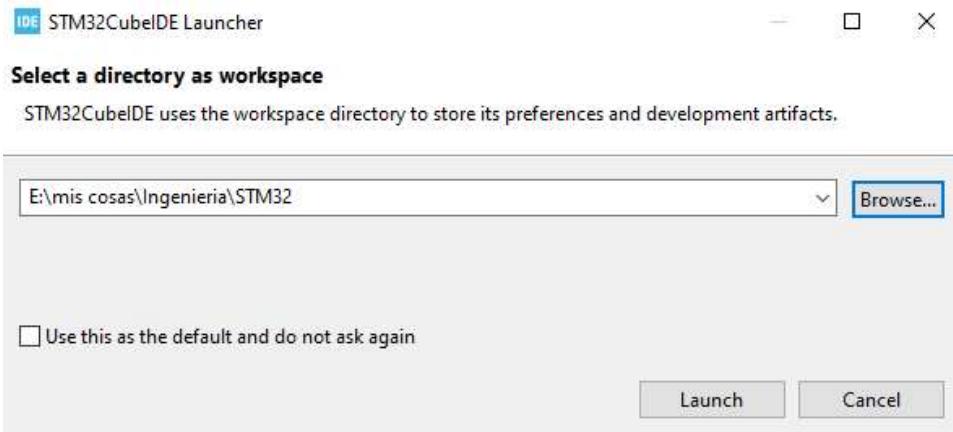


Una vez completada la instalación presionamos “Done” y ahora podremos encontrar en el escritorio el acceso directo a ambos programas.



### Creación de un proyecto:

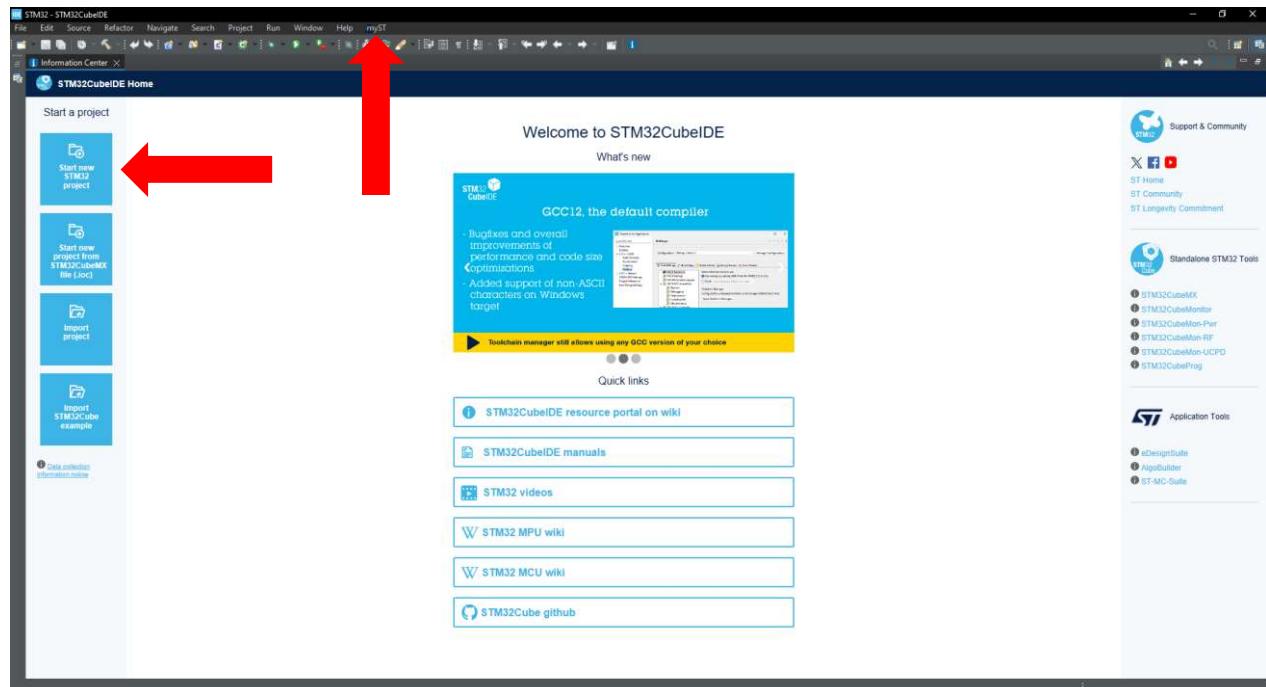
Para crear nuestro primer proyecto debemos iniciar el programa STM32CubeIDE y lo primero que nos pedirá es seleccionar una dirección para crear un Workspace el cual será el espacio donde crearemos nuestros programas y se generan los archivos necesarios para el desarrollo.



A continuación, nos encontraremos con la pantalla de inicio del IDE, donde arriba a la izquierda veremos un recuadro con la opción “Start New STM32 Project”, para poder comenzar con la creación y configuración básica para nuestra primera aplicación.

Como se trata la primera vez que iniciamos el programa es posible que descargue algunos archivos extras durante el proceso e incluso nos pida iniciar sesión con la cuenta que creamos anteriormente.

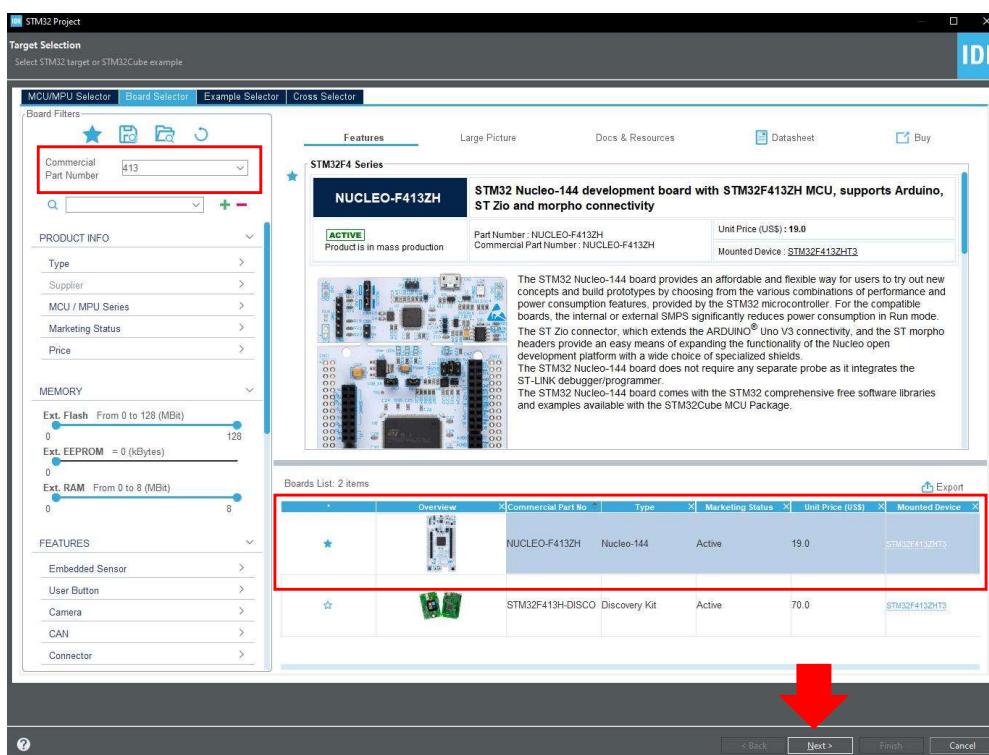
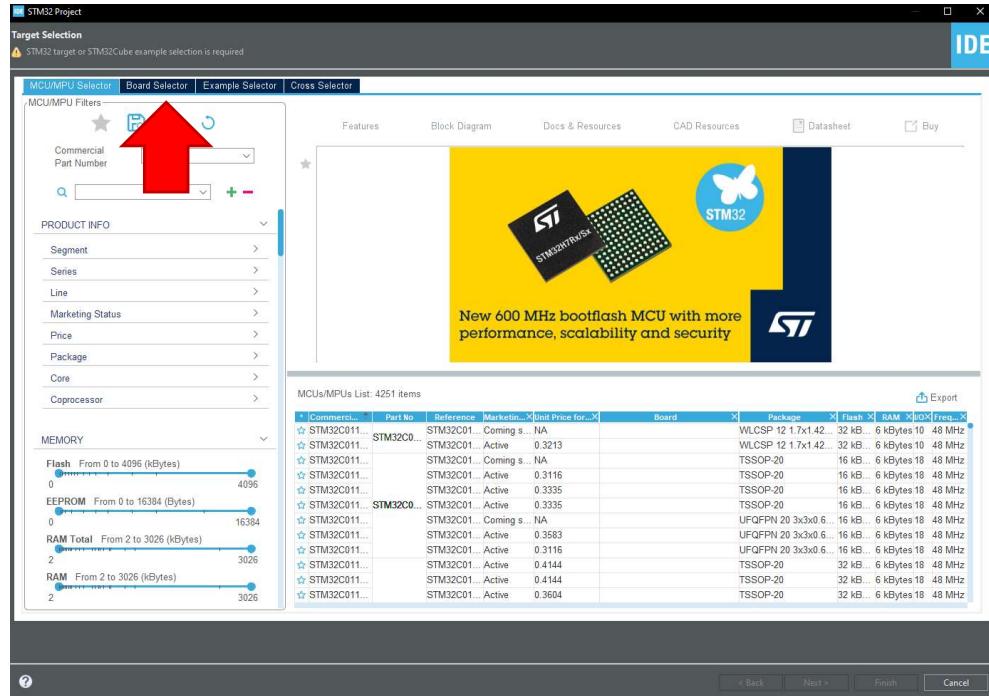
Para iniciar sesión debemos buscar en la barra superior de opciones donde dice myST e ingresar con nuestros datos.



Una vez que seleccionemos iniciar proyecto nuevo nos encontraremos con una ventana donde deberemos seleccionar el modelo de placa con el que trabajaremos, debemos ir a la pestaña en la

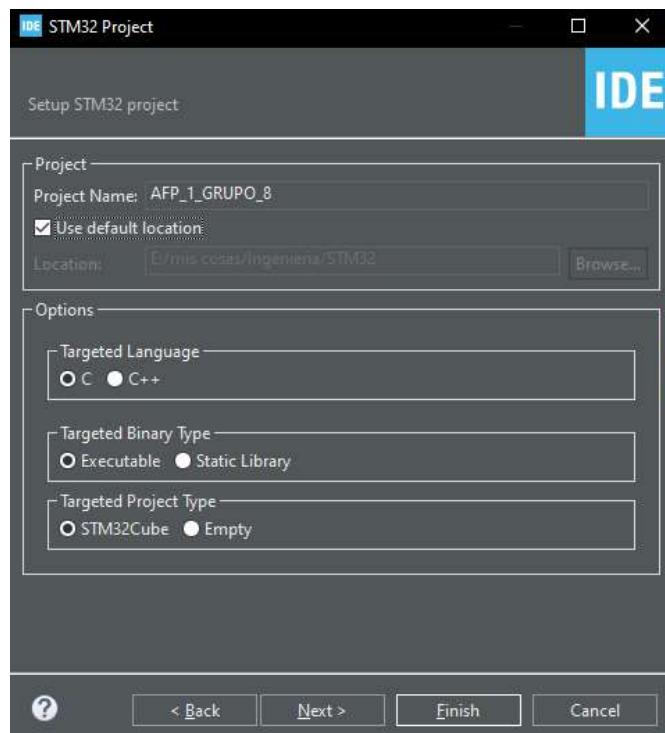
parte superior donde dice "Board Selector" y en el sector superior izquierdo hay un recuadro de búsqueda donde debemos escribir el modelo de nuestra placa, que este caso utilizaremos el modelo Nucleo-F413ZH.

Una vez seleccionado el modelo de la placa presionamos el botón Next en la parte inferior derecha.

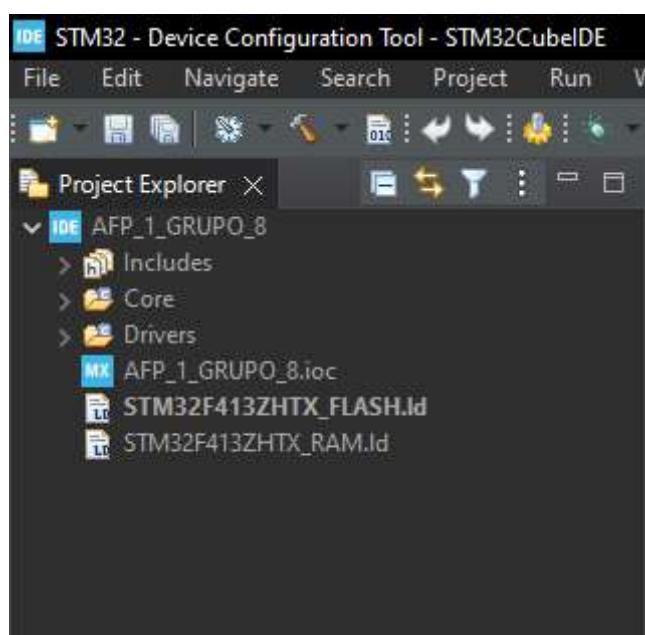


Luego de seleccionar la placa nos encontraremos con una ventana donde deberemos escribir el nombre de nuestro proyecto, cambiar la ubicación donde se almacenará si lo deseamos y podemos elegir otras opciones, pero en este caso utilizaremos las predeterminadas.

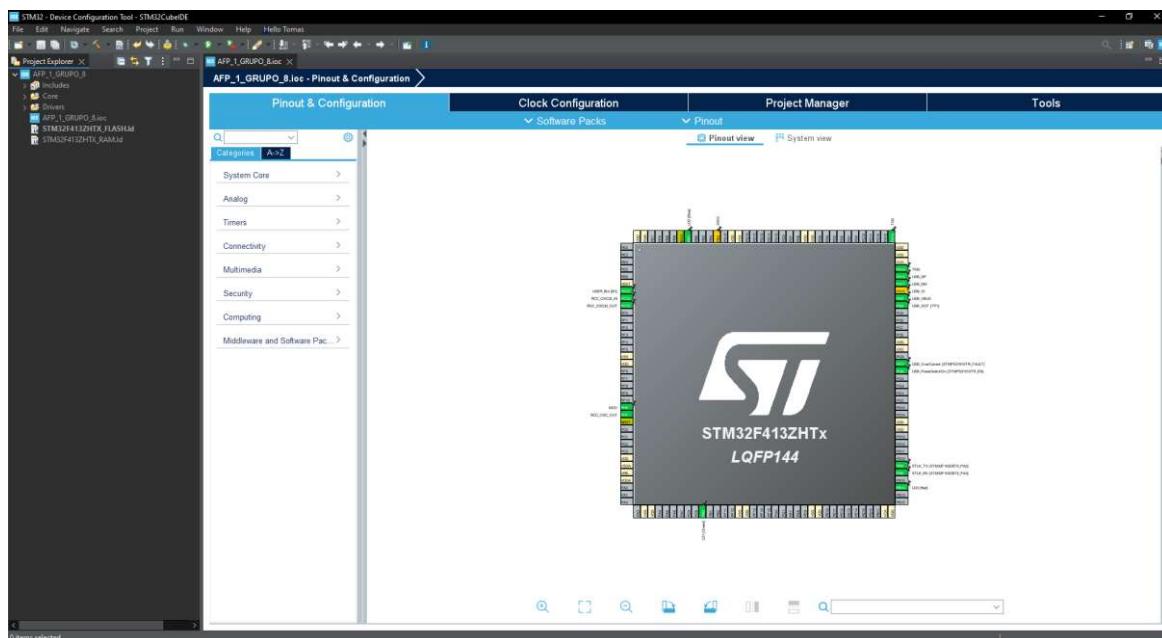
Seleccionamos el botón finish y se creara nuestro proyecto.



Ahora nos encontraremos en la ventana de trabajo, en la parte izquierda se encuentra el explorador de proyectos donde veremos todos los proyectos dentro un Workspace seleccionado, en este caso solo veríamos el que acabamos de crear.



En el sector derecho veremos una ventana que corresponde al archivo .ioc, donde encontraremos la configuración de la placa y el pinout. Donde podremos configurar según nuestras necesidades como se comportará cada pin del microcontrolador, los diferentes módulos de la placa e incluso hasta la frecuencia de reloj del microcontrolador.



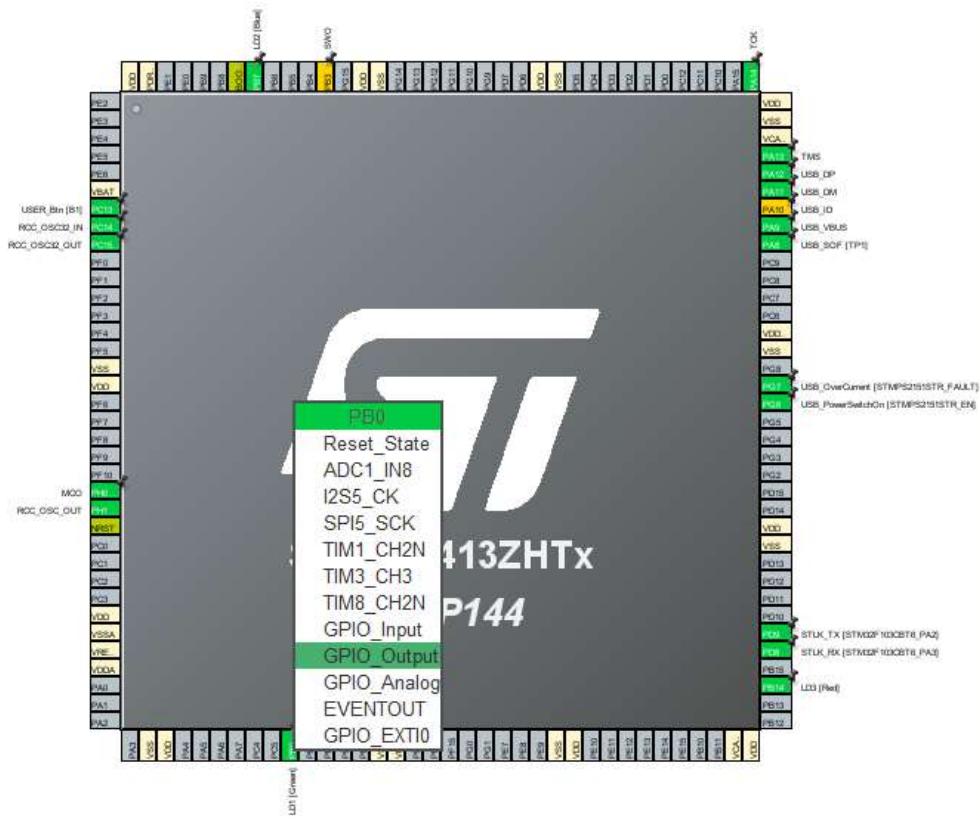
### Modulo GPIO:

El primer módulo que aprenderemos a configurar es el GPIO (General Purpose Input/Output), el cual se encarga de controlar los puertos o pines de entrada y salida del microcontrolador.

Pin Name	Signal on Pin	GPIO output I/O	GPIO mode	GPIO Pull-up/Pull-down	Maximum output current	User Label	Modified
PB0	n/a	Low	Output Push...	No pull-up an...	Low	LD1 [Green]	<input checked="" type="checkbox"/>
PB7	n/a	Low	Output Push...	No pull-up an...	Low	LD2 [Blue]	<input checked="" type="checkbox"/>
PB14	n/a	Low	Output Push...	No pull-up an...	Low	LD3 [Red]	<input checked="" type="checkbox"/>
PC13	n/a	n/a	External Inter...	No pull-up an...	n/a	USER_Btn [B1]	<input checked="" type="checkbox"/>
PG6	n/a	Low	Output Push...	No pull-up an...	Low	USB_PowerS...	<input checked="" type="checkbox"/>
PG7	n/a	n/a	Input mode	No pull-up an...	n/a	USB_OverCur...	<input checked="" type="checkbox"/>

Debemos abrir la pestaña System Core en el sector izquierdo y seleccionar la opción GPIO, en la parte central veremos una pestaña con un listado, cada fila corresponde a un pin del microcontrolador, con su nombre en la primera columna. Si seleccionamos uno este se marcará en el diagrama a la derecha con el chip.

En este diagrama podemos seleccionar cada pin y configurarlo para que se comporte como una de las opciones que nos da en la lista. En este caso configuramos el pin PB0, el cual corresponde a un led soldado en la placa, como GPIO\_Output.



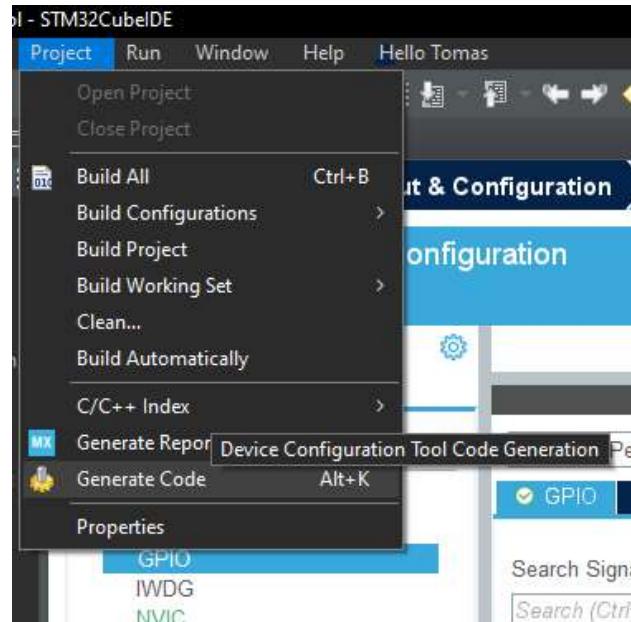
Como configuración predeterminada del programa veremos en el listado 3 leds que están en la placa configurados como salida y el botón de usuario como una entrada. Es posible asignarle una etiqueta a cada pin, por ejemplo LD1. Luego en nuestro programa podremos utilizar estos pines para lograr que los leds se enciendan y se apaguen. Además, si lo deseamos podemos configurar otros pines para agregar leds externos u otros botones.

PB0 Configuration :	
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	LD1 [Green]

Dentro del listado si seleccionamos un pin, en la parte inferior encontramos una serie de opciones con las cuales podemos configurar el comportamiento del pin con el módulo GPIO. Para el caso de los leds en la placa la configuración será como en la imagen anterior.

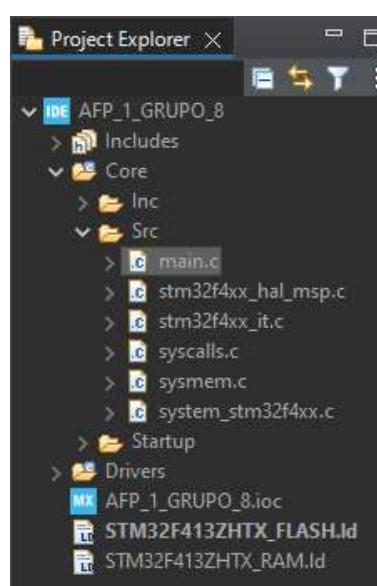
#### Generación del código:

Tras configurar los pines del microcontrolador generamos el código del programa, donde se crearán los archivos necesarios para escribir el código del programa y cargarlo en la placa de desarrollo.



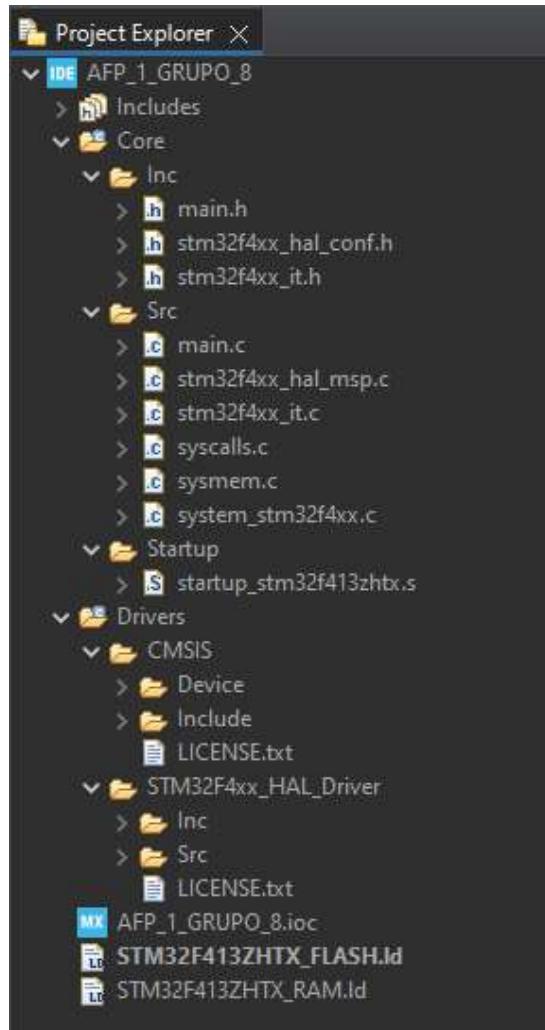
En la fila superior de opciones en la ventana del programa, seleccionamos la opción Project y luego Generate Code. Una vez que el IDE termine de generar los archivos se nos abrirá una ventana nueva con el nombre “main.c” o podemos navegar en el explorador de proyectos para encontrar el archivo.

Core > Src > main.c



#### Estructura del Árbol de Archivos en STM32CubeIDE:

En la ventana del explorador de proyecto tenemos un árbol con todos los archivos de nuestro programa, que contienen tanto el código del programa como la configuración y todos los drivers para que al compilar y cargarlo en nuestro microcontrolador este funcione correctamente.



### 1. Includes:

Esta carpeta suele contener los archivos de cabecera (headers) comunes a todo el proyecto

### 2. Core:

- **Inc.:**
  - main.h: Archivo de cabecera principal del proyecto. Suele contener definiciones globales, prototipos de funciones y declaraciones de variables.
  - stm32f4xx\_hal\_conf.h: Archivo de configuración de la HAL (Hardware Abstraction Layer) específico para la familia de microcontroladores STM32F4.
  - stm32f4xx\_it.h: Archivo de cabecera que declara los manejadores de interrupciones para el microcontrolador STM32F4.
- **Src :**

- main.c: Archivo fuente principal del proyecto. Contiene la función main y el flujo principal del programa.
- stm32f4xx\_hal\_msp.c: Archivo fuente que contiene funciones de inicialización del Sistema de Soporte del Microcontrolador (MSP).
- stm32f4xx\_it.c: Archivo fuente que implementa los manejadores de interrupciones declarados en stm32f4xx\_it.h.
- syscalls.c: Archivo fuente que implementa las llamadas al sistema (system calls) para la integración con las librerías estándar de C.
- sysmem.c: Archivo fuente que implementa la gestión de memoria para el proyecto.
- system\_stm32f4xx.c: Archivo fuente que inicializa el sistema y el reloj del microcontrolador.

### 3. Startup:

- startup\_stm32f429zitx.s: Archivo ensamblador que contiene el código de arranque (startup) para el microcontrolador STM32F429ZITx.

### 4. Drivers:

- CMSIS:
  - Esta carpeta contiene los archivos del CMSIS (Cortex Microcontroller Software Interface Standard), una librería estándar para microcontroladores Cortex-M.
- STM32F4xx\_HAL\_Driver:
  - Inc: Archivos de cabecera para los drivers HAL específicos para la serie STM32F4.
  - Src: Archivos fuente que implementan los drivers HAL específicos para la serie STM32F4.
  - LICENSE.txt: Archivo que contiene la licencia de uso de los drivers HAL.

### 5. AFP1\_TDII Grupo8.ioc:

- Archivo de configuración del proyecto generado por STM32CubeMX. Contiene toda la configuración de los periféricos y pines del microcontrolador

### 6. STM32F429ZITX\_FLASH.Id:

- Archivo de script de enlace (linker script) que define la organización de la memoria FLASH del microcontrolador.

### 7. STM32F429ZITX\_RAM.Id:

- Archivo de script de enlace (linker script) que define la organización de la memoria RAM del microcontrolador.

Esta estructura es típica en los proyectos de STM32CubeIDE, aunque algunas variaciones pueden ocurrir dependiendo de las opciones seleccionadas al crear el proyecto o de las particularidades de este.

### Desarrollo de un programa:

Como ejemplo crearemos un programa con la función de hacer parpadear uno de los leds integrados de la placa.

El desarrollo del código para nuestro programa debe hacerse en el archivo main.c, donde el mismo IDE genera las declaraciones necesarias para el microcontrolador.

```

MX_AF_1_GRUPO_8.ioc main.c

67  */
68  */
69 int main(void)
70 {
71
72     /* USER CODE BEGIN 1 */
73
74     /* USER CODE END 1 */
75
76     /* MCU Configuration-----*/
77
78     /* Reset of all peripherals, Initializes the Flash interface and the Systick */
79     HAL_Init();
80
81     /* USER CODE BEGIN Init */
82
83     /* USER CODE END Init */
84
85     /* Configure the system clock */
86     SystemClock_Config();
87
88     /* USER CODE BEGIN SysInit */
89
90     /* USER CODE END SysInit */
91
92     /* Initialize all configured peripherals */
93     MX_GPIO_Init();
94     MX_USART3_UART_Init();
95     MX_USB_OTG_FS_PCD_Init();
96     /* USER CODE BEGIN 2 */
97
98     /* USER CODE END 2 */
99
100    /* Infinite loop */
101    /* USER CODE BEGIN WHILE */
102    while (1)
103    {
104        /* USER CODE END WHILE */
105    }
106    /* USER CODE BEGIN 3 */
107
108    /* USER CODE END 3 */
109
110}
111 /**
112 * @brief System Clock Configuration
113 * @param None
114 */
115 void SystemClock_Config(void)
116 {

```

Como utilizaremos un led integrado en la placa, de los cuales configuraremos inicialmente al crear el proyecto, no es necesario declararlos variables.

Utilizaremos el led rojo que corresponde al pin PB14 y cómo podemos verificar en el archivo .ioc la etiqueta correspondiente es LD3 [Red].

Nuestro código debe escribirse dentro del loop while, en el segmento de main. Donde podemos guiarnos por los comentarios propios del IDE en que partes debe ir el código del usuario.

La función “HAL\_GPIO\_TogglePin()” no permite alternar el estado de un pin indicado, tal que para encender o apagar el led seria:

HAL\_GPIO\_TogglePin(GPIOB, LD3\_Pin)

GPIOB indica que se encuentra en la sección B de los pines.

Luego si deseamos que haya una cierta demora entre que se enciende y se apaga el led para poder observar el parpadeo, utilizamos la función “HAL\_Delay()” donde entre paréntesis especificamos el tiempo en milisegundos, si queremos que parpadee cada 250 ms será:

HAL\_Delay(250)

```
while (1)
{
    /* USER CODE END WHILE */
    HAL_GPIO_TogglePin(GPIOB,LD3_Pin); //Alternar estado del pin PB14 o led rojo

    HAL_Delay(250); //Esperar 250 ms
    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Así quedaría el código para parpadear el led rojo integrado en la placa cada 250 ms. Luego se compila el código y carga en la placa.

## Compilación del Proyecto en STM32CUBEIDE

En STM32CubeIDE, compilar un proyecto implica transformar el código fuente escrito en lenguajes como C o C++ en código objeto y, finalmente, en un archivo ejecutable que puede ser cargado en el microcontrolador STM32. A continuación, te explico cómo se lleva a cabo el proceso de compilación, qué archivo se genera como resultado y dónde se encuentra en la estructura del árbol de archivos.

### Pasos para Compilar el Proyecto

#### 1.Abrir el Proyecto:

Abre STM32CubeIDE y carga tu proyecto. Asegúrate de que el proyecto esté configurado correctamente y que los archivos fuente y de encabezado estén en las carpetas correspondientes.

#### 2.Seleccionar el Modo de Compilación:

- **Debug:** Modo de depuración, donde el código se compila con información adicional para facilitar la depuración.
- **Release:** Modo de lanzamiento, donde el código se compila sin información de depuración y con optimizaciones para un rendimiento máximo.

Puedes seleccionar el modo deseado desde la configuración del proyecto o al crear una nueva configuración de compilación.

#### 3.Iniciar la Compilación:

- **Compilación Completa (Build All):** Compila todos los archivos del proyecto. Puedes hacer esto haciendo clic derecho en el proyecto en el Explorador de Proyectos y seleccionando Build Project, o desde el menú principal seleccionando Project > Build Project.
- **Compilación Incremental:** Si solo has cambiado algunos archivos, STM32CubeIDE recompilará solo esos archivos, en lugar de recompilar todo el proyecto.

#### 4.Verificación de Errores:

Durante la compilación, STM32CubeIDE mostrará mensajes en la consola. Si hay errores de compilación, se te notificará y se mostrarán los detalles para que puedas corregirlos.

### Archivos Generados Tras la Compilación

Cuando la compilación se complete con éxito, STM32CubeIDE genera varios archivos en el árbol de archivos del proyecto. Los más importantes son:

#### 1.Código Objeto (.o)

- **Qué es:** Son archivos intermedios que contienen el código máquina generado a partir de cada archivo fuente .c o .cpp. Cada archivo fuente tiene un archivo objeto correspondiente.
- **Ubicación:** Se encuentran en una subcarpeta dentro de Debug/ o Release/, dependiendo del modo de compilación. Generalmente, estos archivos no son visibles en el Explorador de Proyectos a menos que habilites la visualización de archivos ocultos.

#### 2.Archivo Ejecutable (.elf)

- **Qué es:** El archivo .elf es el principal archivo ejecutable que se carga en el microcontrolador. Este archivo incluye el código máquina final, así como información adicional como símbolos de depuración.
- **Ubicación:** Se encuentra en la carpeta Debug/ o Release/ bajo el directorio principal del proyecto. Por ejemplo, si tu proyecto se llama MyProject, el archivo se ubicará en MyProject/Debug/MyProject.elf.

#### 3.Archivo Binario (.bin)

- **Qué es:** Este es un archivo binario que contiene solo el código ejecutable en formato binario puro, sin información adicional.
- **Ubicación:** Al igual que el archivo .elf, se encuentra en Debug/ o Release/. El archivo puede llamarse MyProject.bin.

#### 4.Archivo Hexadecimal (.hex)

- **Qué es:** Es otro tipo de archivo ejecutable que contiene el código binario en formato hexadecimal. Lo usan algunos programadores para cargar el código en el microcontrolador.

- **Ubicación:** También está en la carpeta Debug/ o Release/, con un nombre similar a MyProject.hex.

## Ubicación de Archivos en el Árbol de Archivos

La ubicación de los archivos generados después de la compilación sigue una estructura específica dentro de tu proyecto:

- **MyProject/**
  - **Debug/ (o Release/)**
    - **MyProject.elf** (archivo ejecutable)
    - **MyProject.bin** (archivo binario)
    - **MyProject.hex** (archivo hexadecimal)
    - **src/**
      - **archivo1.o** (código objeto generado por archivo1.c)
      - **archivo2.o** (código objeto generado por archivo2.c)

## Cargar el Archivo en el Microcontrolador

Después de la compilación, el archivo MyProject.elf se utiliza generalmente para cargar el programa en el microcontrolador STM32 mediante una herramienta de depuración (como ST-Link) directamente desde STM32CubeIDE, utilizando la opción de Debug. También puedes utilizar el archivo .bin o .hex si prefieres otro método de programación.

Este flujo te permite compilar y ejecutar tu proyecto en un microcontrolador STM32 utilizando STM32CubeIDE de manera eficiente.

## Configuración del Debugger en STM32CubeIDE y Descarga del Código Objeto a la Placa Target

Configurar y utilizar el depurador (debugger) en STM32CubeIDE es un paso crucial para verificar y corregir el funcionamiento del código en un microcontrolador STM32. El depurador te permite cargar el código objeto en la placa, ejecutar el código paso a paso, y analizar el comportamiento del programa en tiempo real. A continuación, te explico cómo configurar el depurador y cómo descargar el código objeto a la placa.

### 1. Conexión Física

Primero, asegúrate de que tu placa STM32 está conectada físicamente a tu computadora. Normalmente, esto se realiza mediante un cable USB que conecta el puerto ST-Link (o similar) de la placa al puerto USB de tu PC.

### 2. Configuración del Debugger en STM32CubeIDE

■ **Crear Configuración de Depuración:**

1. **Abrir STM32CubeIDE:** Inicia STM32CubeIDE y abre tu proyecto.
2. **Acceder al Menú de Depuración:** Haz clic en la pequeña flecha al lado del ícono de depuración (un escarabajo verde) en la barra de herramientas y selecciona Debug Configurations....
3. **Crear una Nueva Configuración:** En la ventana de configuraciones de depuración, selecciona STM32 MCU C/C++ Application en el panel de la izquierda y haz clic en New Configuration (puedes hacer clic derecho y seleccionar New).
4. **Seleccionar el Proyecto y el Archivo ELF:** Asegúrate de que el proyecto correcto esté seleccionado en el campo Project, y que el archivo .elf generado tras la compilación esté seleccionado en el campo C/C++ Application.
5. **Configurar el Debugger:**
  - a. En la pestaña Debugger, selecciona ST-LINK (OpenOCD) como el hardware de depuración (este es el más común, pero puede variar según la placa).
  - b. En Port, deja la opción por defecto, que generalmente es USB.

■ **Configuración Avanzada (Opcional):**

- **Verificación de Memoria y Opciones de Programación:** En la sección Startup, puedes habilitar opciones como Reset and Run para que el microcontrolador se reinicie automáticamente al iniciar la depuración, y Load symbols para cargar los símbolos de depuración.
- **Velocidad del Debugger:** Si tienes problemas de conexión o estabilidad, podrías ajustar la velocidad de comunicación (en KHz) en la sección de configuración avanzada del ST-LINK.

■ **Guardar la Configuración:** Haz clic en Apply y luego en Close para guardar la configuración.

### 3. Iniciar la Sesión de Depuración y Descargar el Código

- **Iniciar la Depuración:** Una vez configurado el depurador, puedes iniciar la sesión de depuración haciendo clic en el ícono de depuración (escarabajo verde) o desde Run > Debug As > STM32 MCU C/C++ Application.

- **Descarga del Código Objeto:** STM32CubeIDE compilará el proyecto (si no está ya compilado) y luego descargará el archivo .elf a la memoria flash del microcontrolador. Esto se hace automáticamente durante el inicio de la sesión de depuración.
- **Verificación de Conexión:** Si todo está correctamente configurado y conectado, STM32CubeIDE debería iniciar la sesión de depuración. La placa se reiniciará y el código comenzará a ejecutarse bajo el control del depurador.

#### 4. Uso de las Herramientas de Depuración

Una vez que la depuración está en marcha, STM32CubeIDE te ofrece varias herramientas para analizar el comportamiento del programa:

- **Breakpoints:** Puedes establecer puntos de interrupción (breakpoints) en tu código. Esto hará que el microcontrolador se detenga cuando alcance esa línea, permitiéndote inspeccionar el estado del sistema.
- **Step Over / Step Into / Step Return:** Estas opciones te permiten ejecutar el código línea por línea. Step Over avanza al siguiente paso, Step Into entra en una función, y Step Return sale de la función actual.
- **Variables y Watchpoints:** Puedes inspeccionar y modificar variables en tiempo real mientras el código está detenido. También puedes monitorear variables específicas para detectar cambios en su valor.
- **Memoria y Registros:** Accede a los registros del microcontrolador y a las ubicaciones de memoria para ver el estado interno del dispositivo.
- **Consola y Traza:** La consola de depuración muestra información detallada sobre la ejecución del código y los mensajes de depuración que el microcontrolador envía de vuelta a la IDE.

#### 5. Finalizar la Depuración

Cuando termines de depurar, puedes detener la sesión haciendo clic en el botón Termíname (un cuadrado rojo) en la barra de herramientas de depuración. Esto detendrá el programa en el microcontrolador y finalizará la sesión de depuración en STM32CubeIDE.

