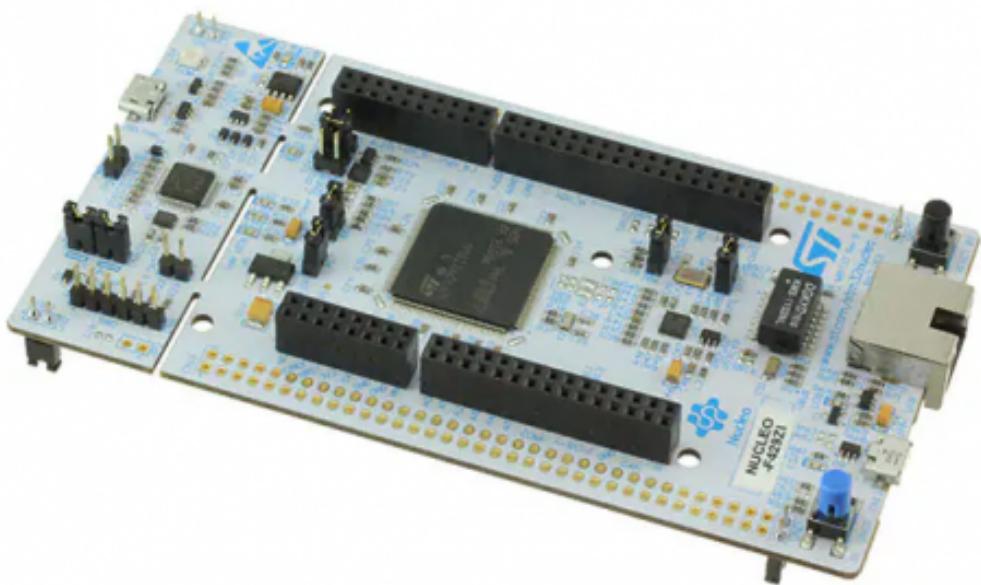


Desarrollo y Configuración de un Proyecto con STM32-F429ZI



Curso: Técnicas Digitales II

Profesor: Ing. Rubén Darío Mansilla

Autores: Vera Monasterio Candela, Barrientos Lucas, Cuellar Agustín

Fecha de entrega: Agosto 2024

Introducción

STM32CubeIDE es un entorno de desarrollo integrado (IDE) para el desarrollo de aplicaciones en microcontroladores STM32. Combina funciones de edición de código, compilación y depuración, y proporciona herramientas de configuración y generación de código basadas en STM32CubeMX.

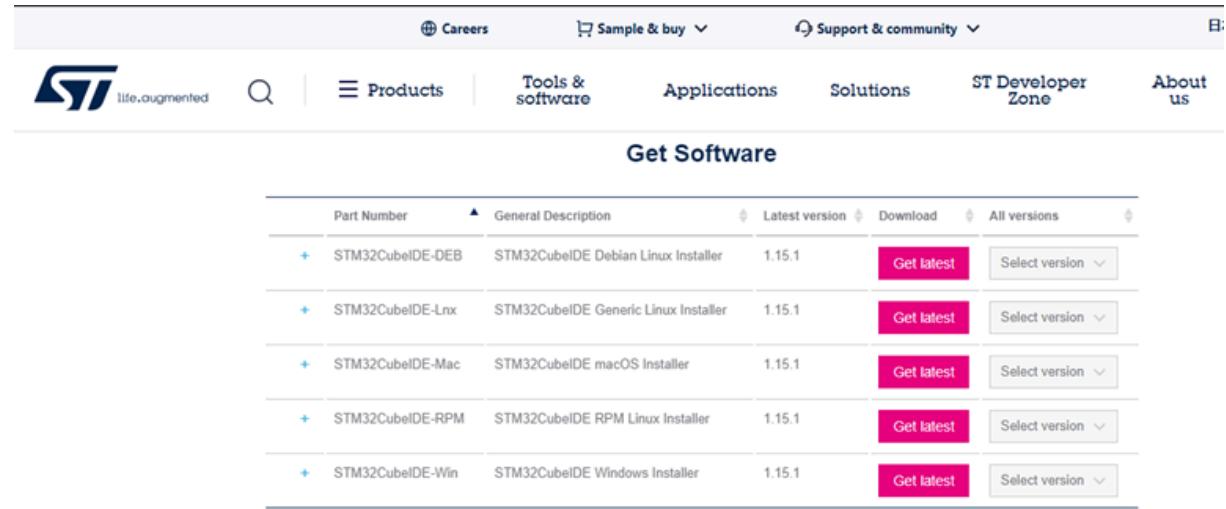
En primer lugar, ingresamos al link de descarga de la ide:

The screenshot shows the official STMicroelectronics website at st.com/en/development-tools/stm32cubeide.html. The page is titled "STM32CubeIDE ACTIVE Integrated Development Environment for STM32". It features two prominent pink buttons: "Get Software" and "Download databrief". Below these buttons is a navigation bar with tabs for "Overview", "Documentation", and "Tools & Software". The "Overview" tab is currently selected. At the bottom of the page, there is a dark banner with text about AI development and a yellow button to "Go to the ST Edge AI Suite".

Luego seleccionamos donde dice **get software**:

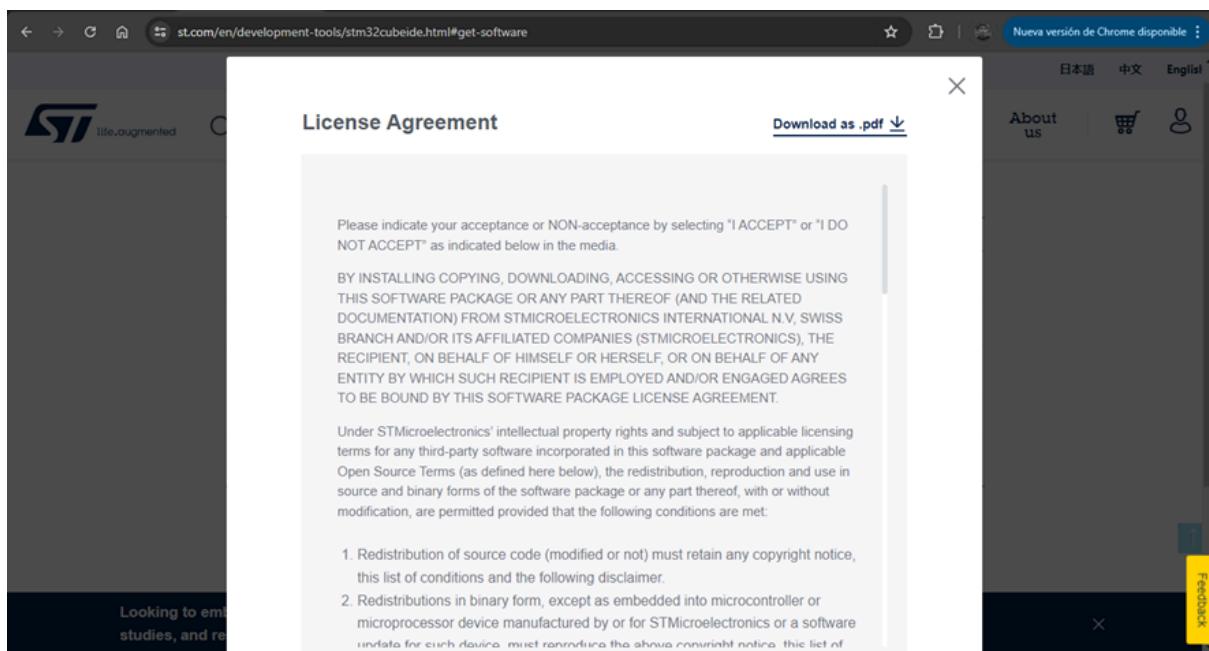
This screenshot shows the same STM32CubeIDE page as above, but with the "Get Software" button highlighted in pink, indicating it has been selected. The rest of the interface remains the same, including the navigation tabs and the AI development banner at the bottom.

A continuación, al ingresar buscaremos la versión adecuada



Part Number	General Description	Latest version	Download	All versions
STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	1.15.1	Get latest	Select version
STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	1.15.1	Get latest	Select version
STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	1.15.1	Get latest	Select version
STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	1.15.1	Get latest	Select version
STM32CubeIDE-Win	STM32CubeIDE Windows Installer	1.15.1	Get latest	Select version

Una vez seleccionada la opción de Windows, nos saldrá la licencia de Agreement, le damos aceptar y continuamos.

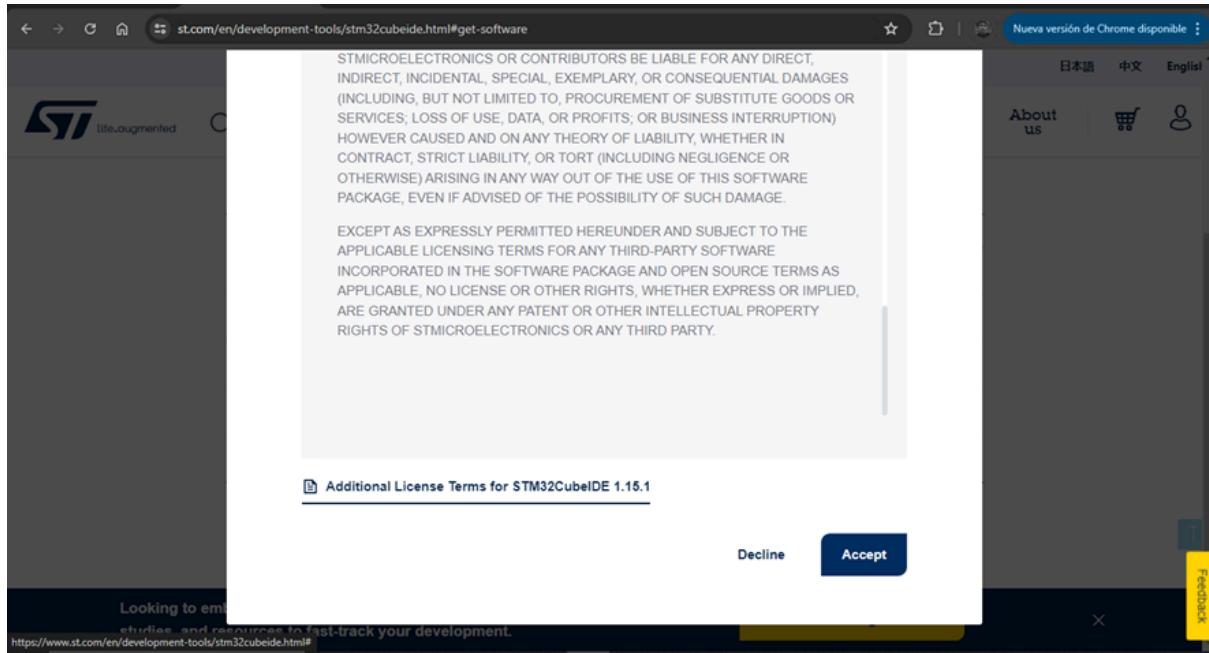


Please indicate your acceptance or NON-acceptance by selecting "I ACCEPT" or "I DO NOT ACCEPT" as indicated below in the media.

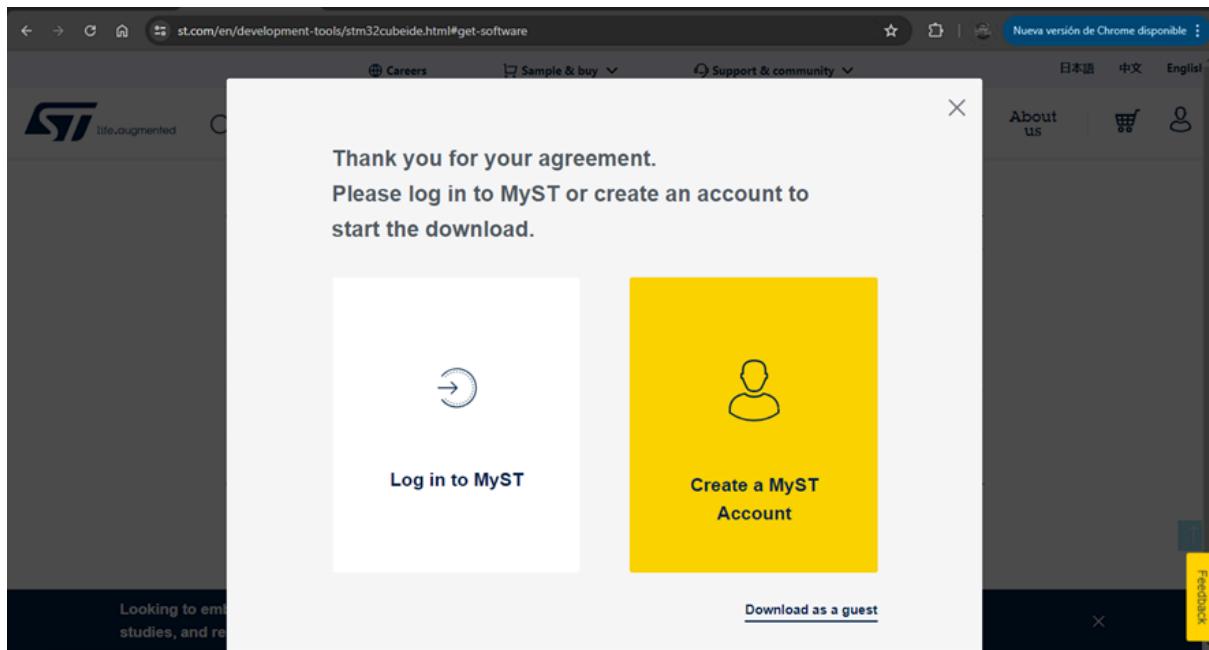
BY INSTALLING COPYING, DOWNLOADING, ACCESSING OR OTHERWISE USING THIS SOFTWARE PACKAGE OR ANY PART THEREOF (AND THE RELATED DOCUMENTATION) FROM STMICROELECTRONICS INTERNATIONAL N.V. SWISS BRANCH AND/OR ITS AFFILIATED COMPANIES (STMICROELECTRONICS), THE RECIPIENT, ON BEHALF OF HIMSELF OR HERSELF, OR ON BEHALF OF ANY ENTITY BY WHICH SUCH RECIPIENT IS EMPLOYED AND/OR ENGAGED AGREES TO BE BOUND BY THIS SOFTWARE PACKAGE LICENSE AGREEMENT.

Under STMicroelectronics' intellectual property rights and subject to applicable licensing terms for any third-party software incorporated in this software package and applicable Open Source Terms (as defined here below), the redistribution, reproduction and use in source and binary forms of the software package or any part thereof, with or without modification, are permitted provided that the following conditions are met:

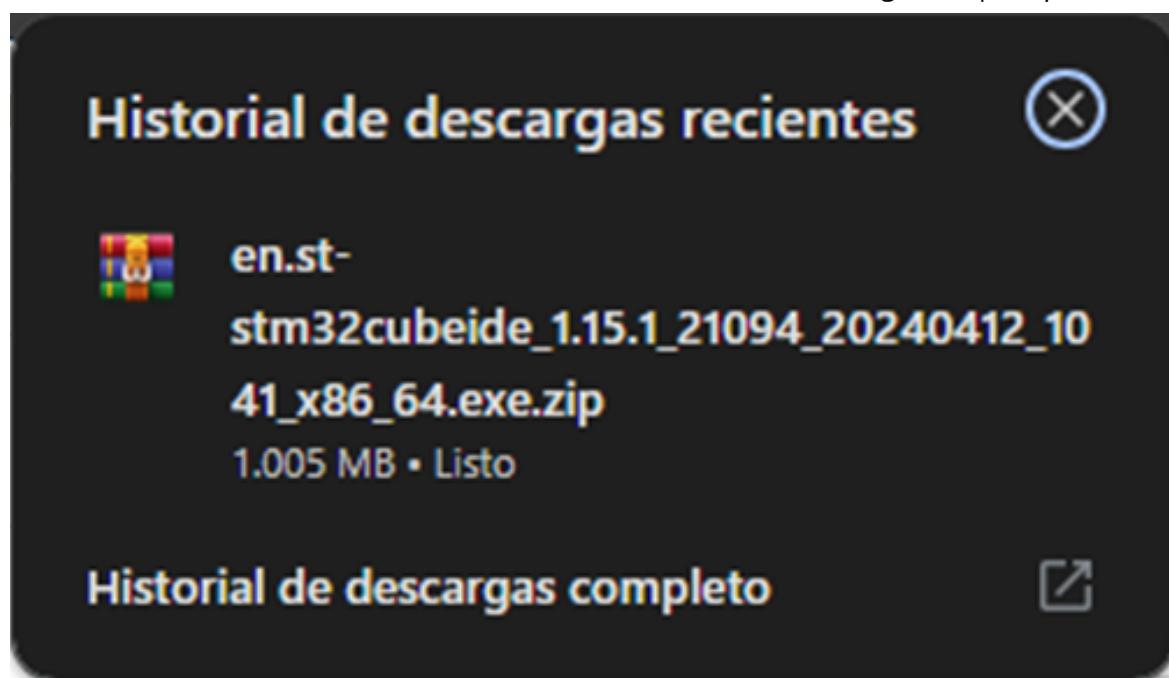
1. Redistribution of source code (modified or not) must retain any copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form, except as embedded into microcontroller or microprocessor device manufactured by or for STMicroelectronics or a software module for such device, must contain the above copyright notice, this list of



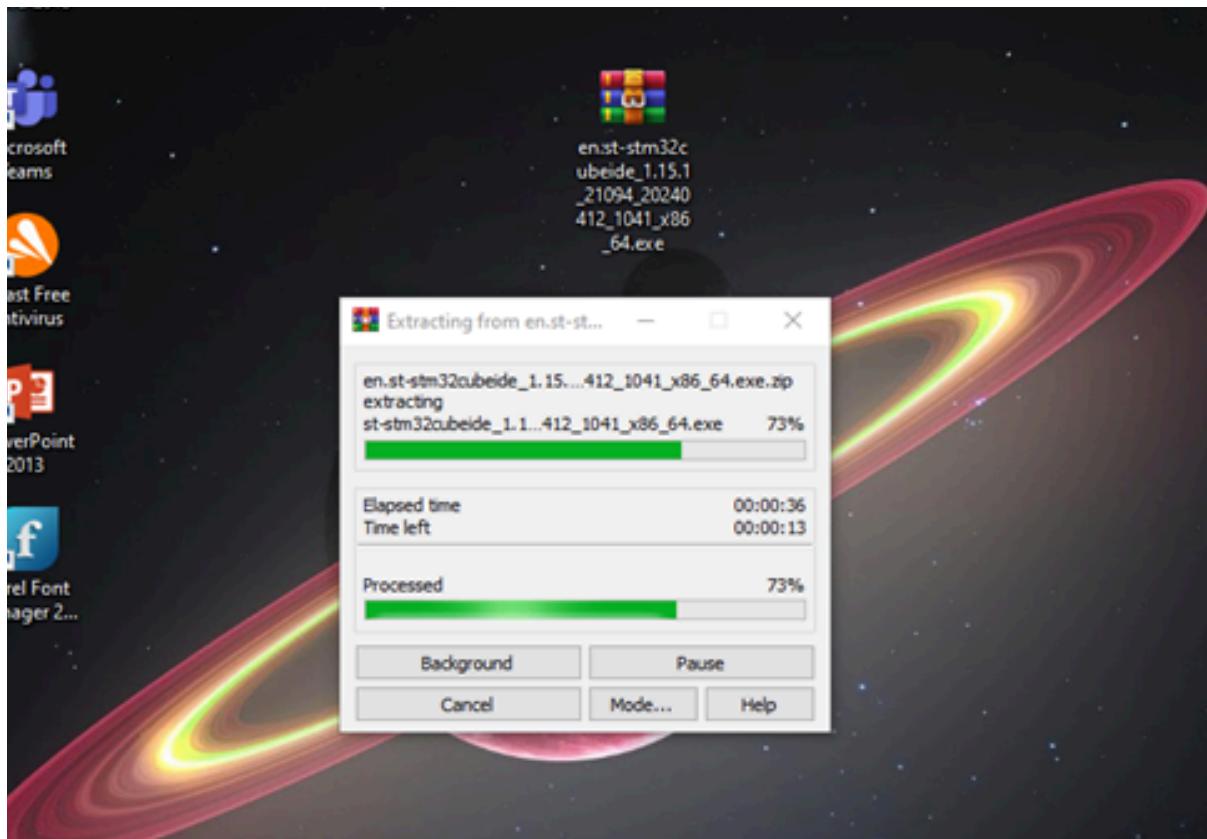
En el caso de no contar con una cuenta, nos va a pedir que creemos una para poder iniciar la descarga.



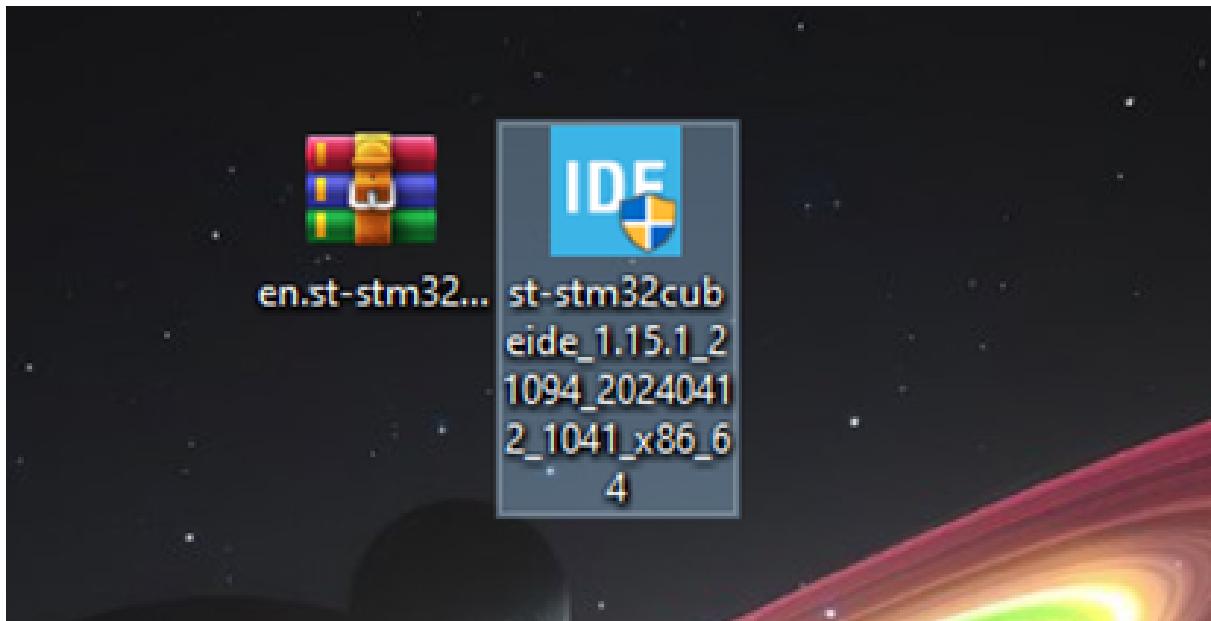
En este caso luego de crear la cuenta se tuvo que volver a cargar la página y nuevamente seleccionar para descargar la versión, una vez logueado se empieza a descargar el archivo.



Al finalizar la descarga, nos dirigimos a la carpeta de “descargas” de la pc y lo llevamos al escritorio para descomprimir el archivo.

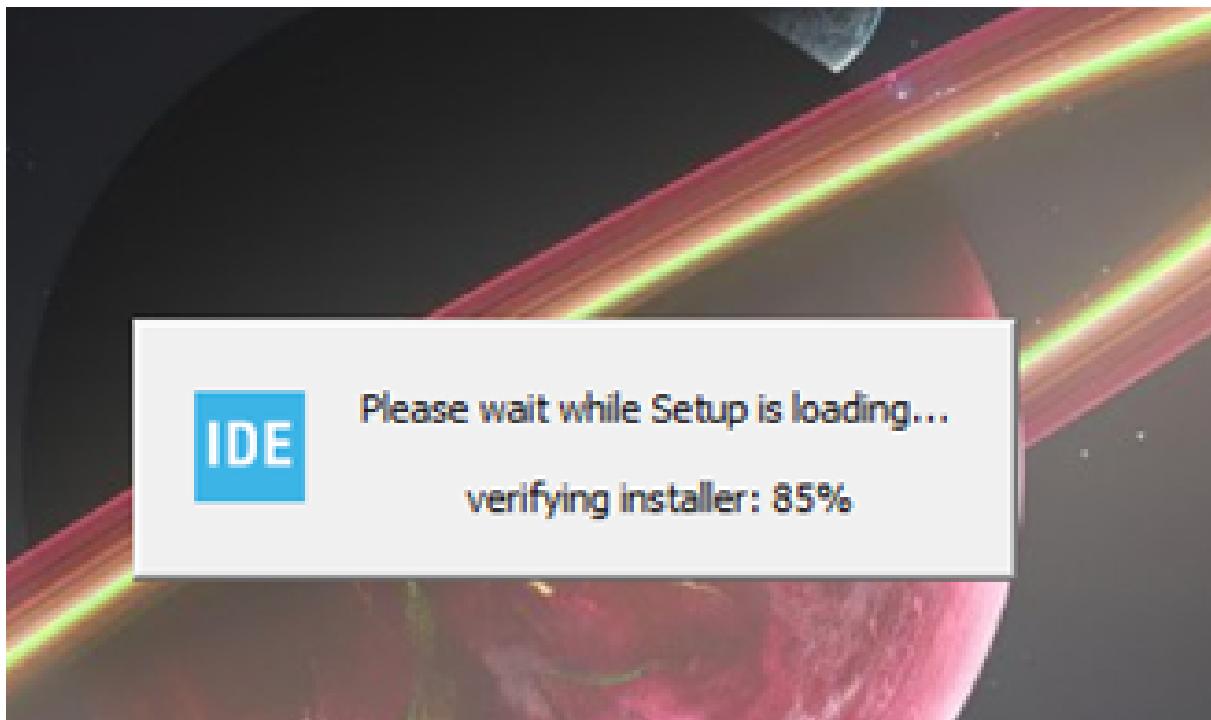


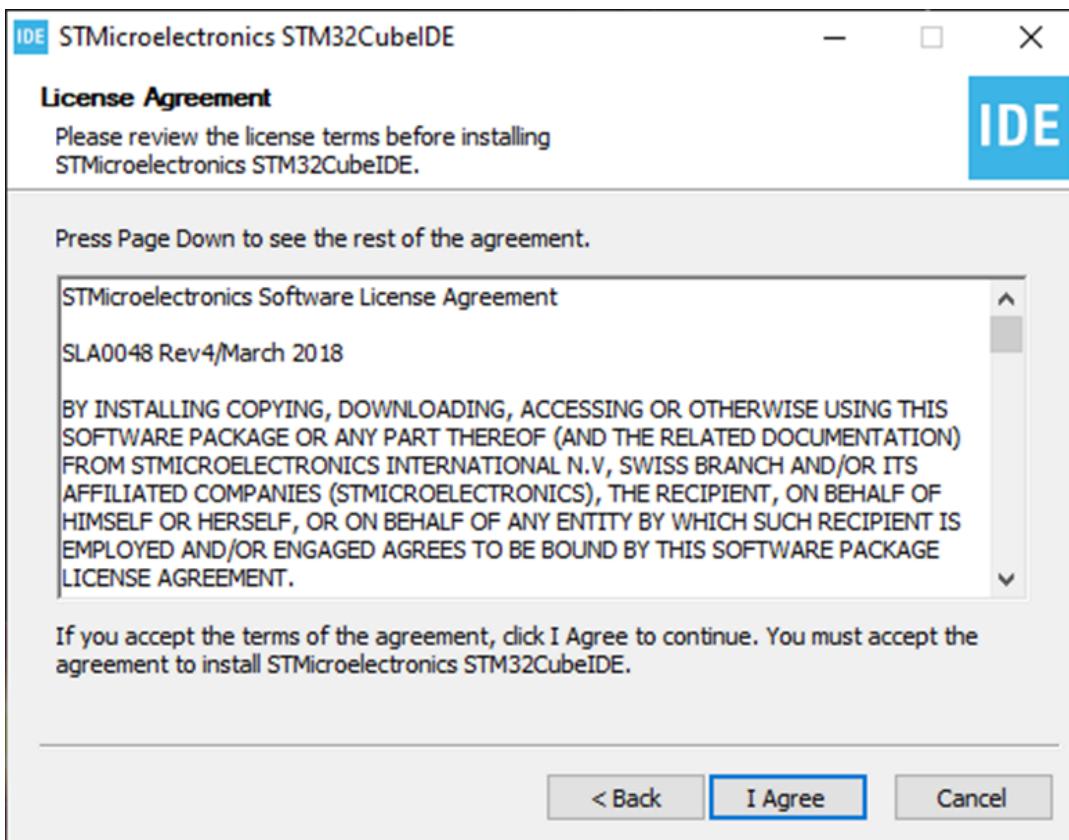
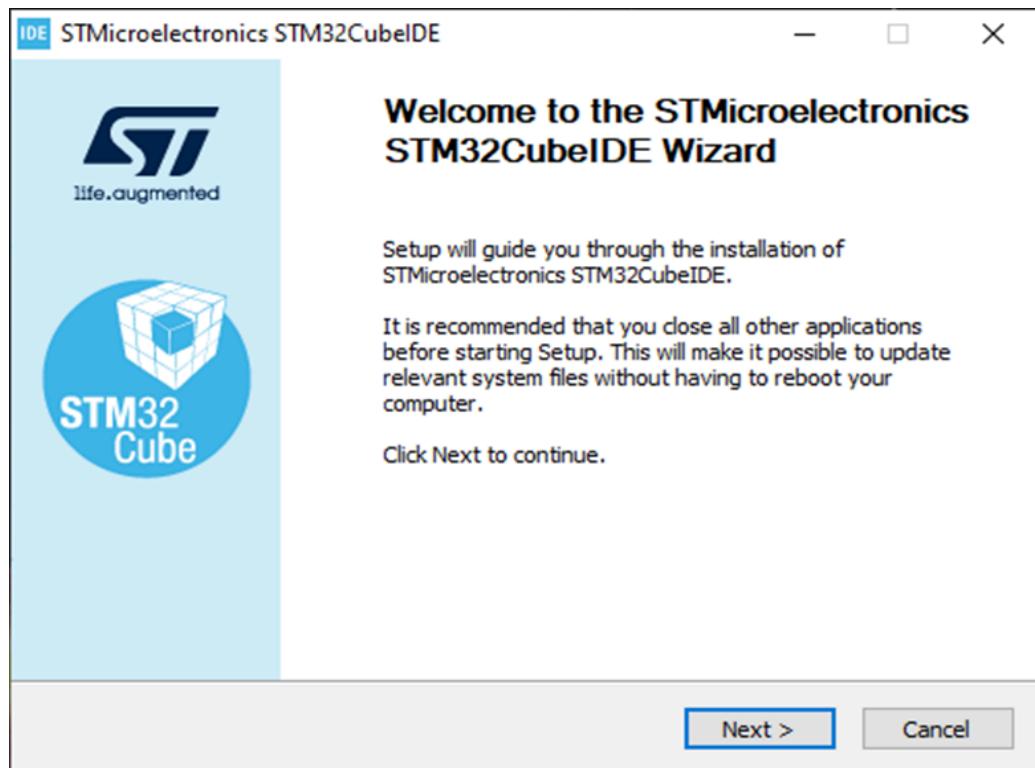
Luego de descomprimir obtendremos el archivo del instalador



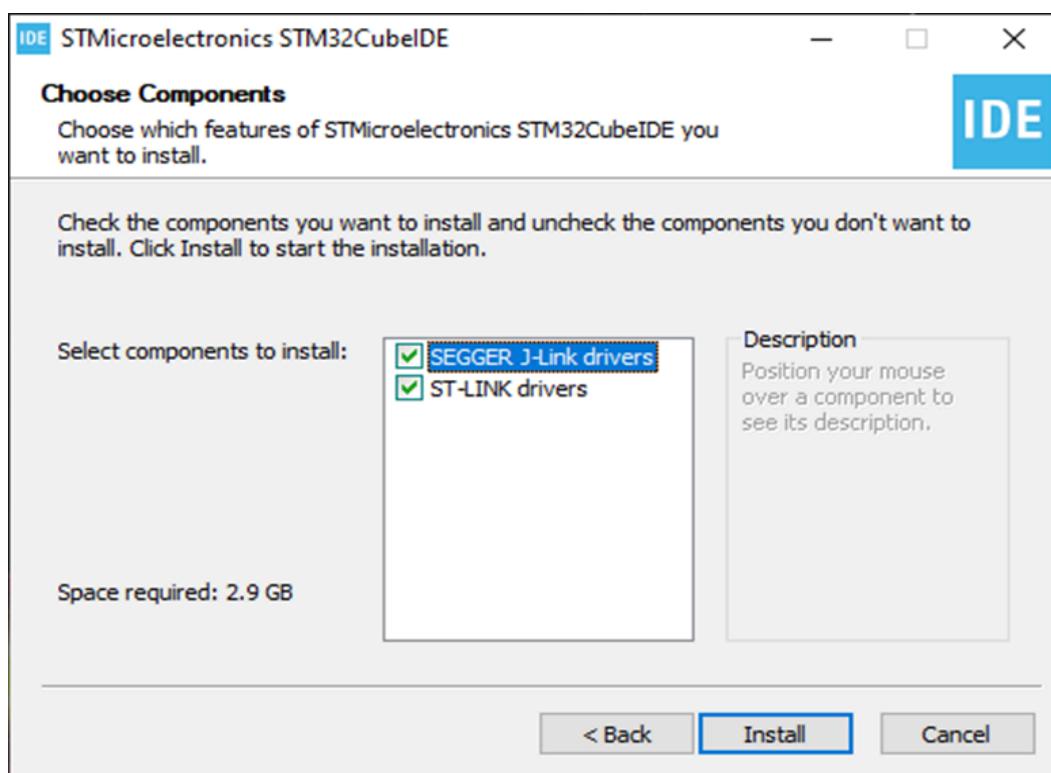
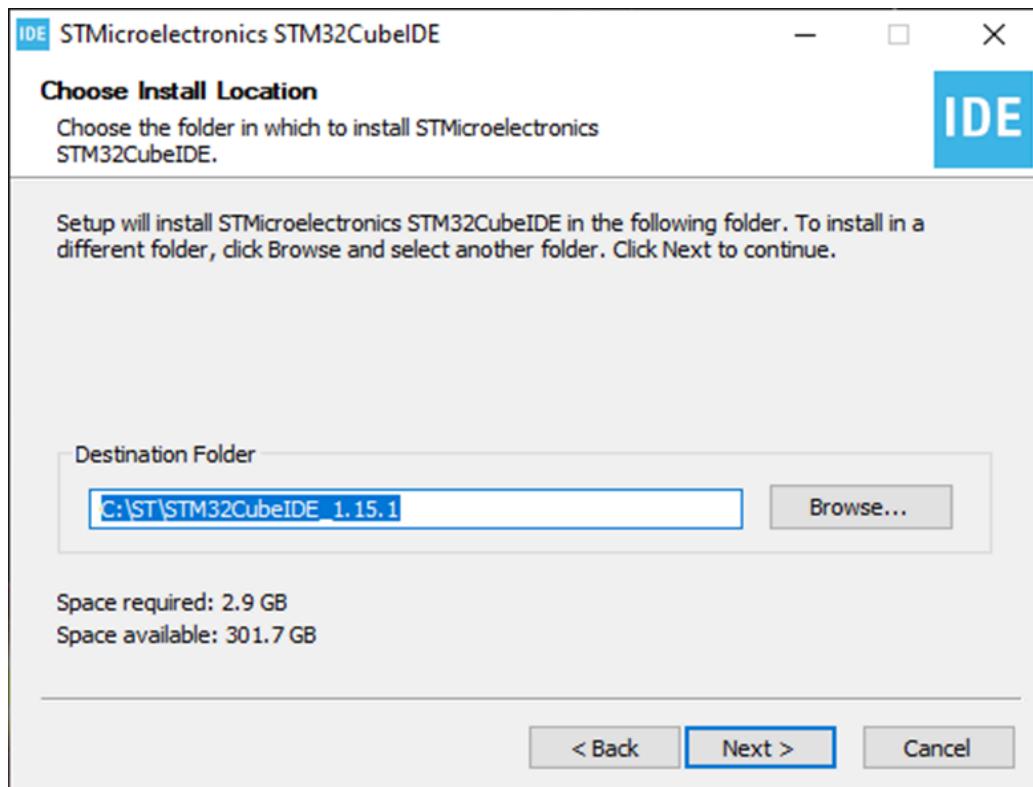
A continuación ejecutamos el programa como admins luego de aceptar cuando nos pregunte si deseamos ejecutar como tal.

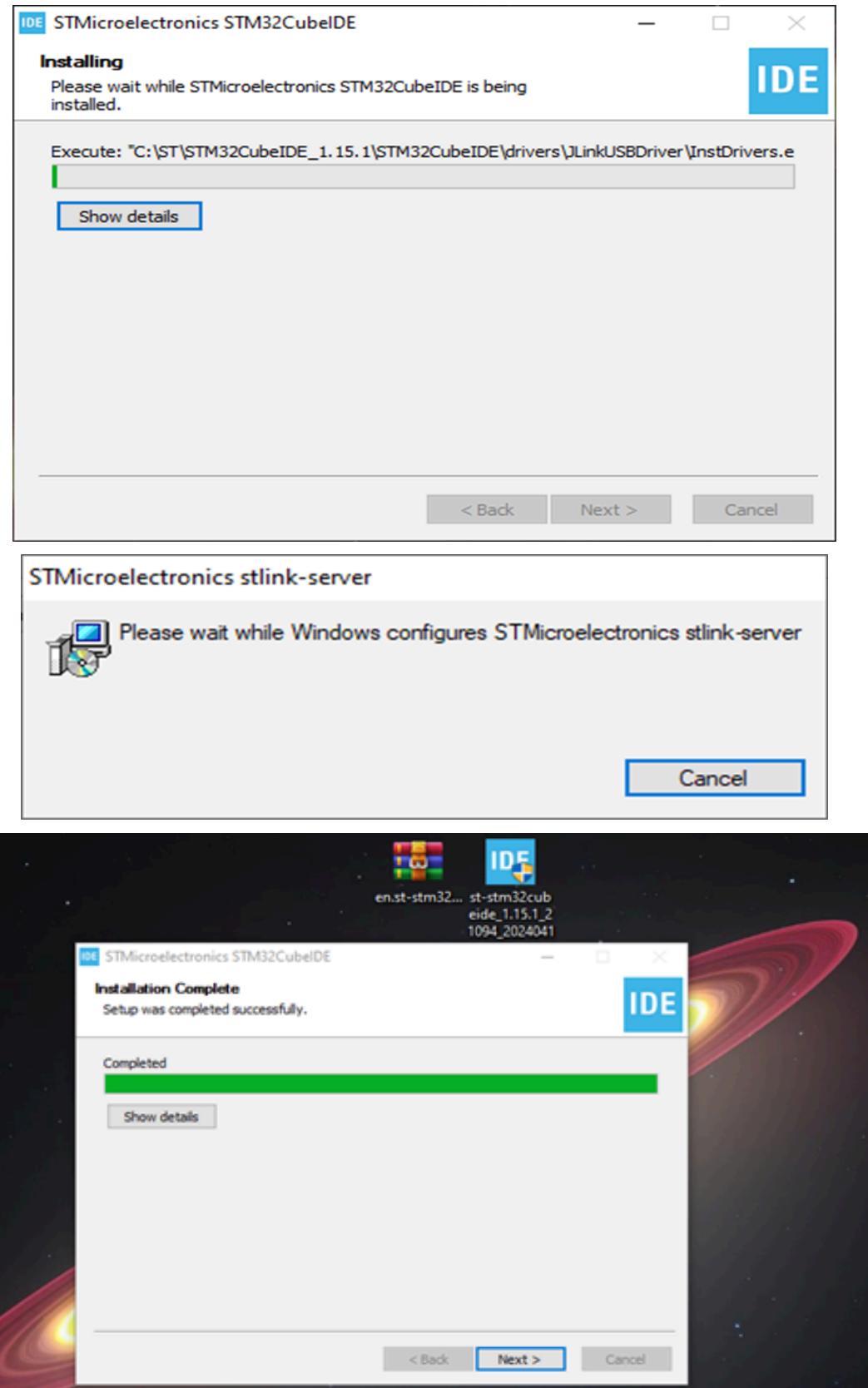
Una vez realizado este paso, nos aparecerá un cartel de carga.



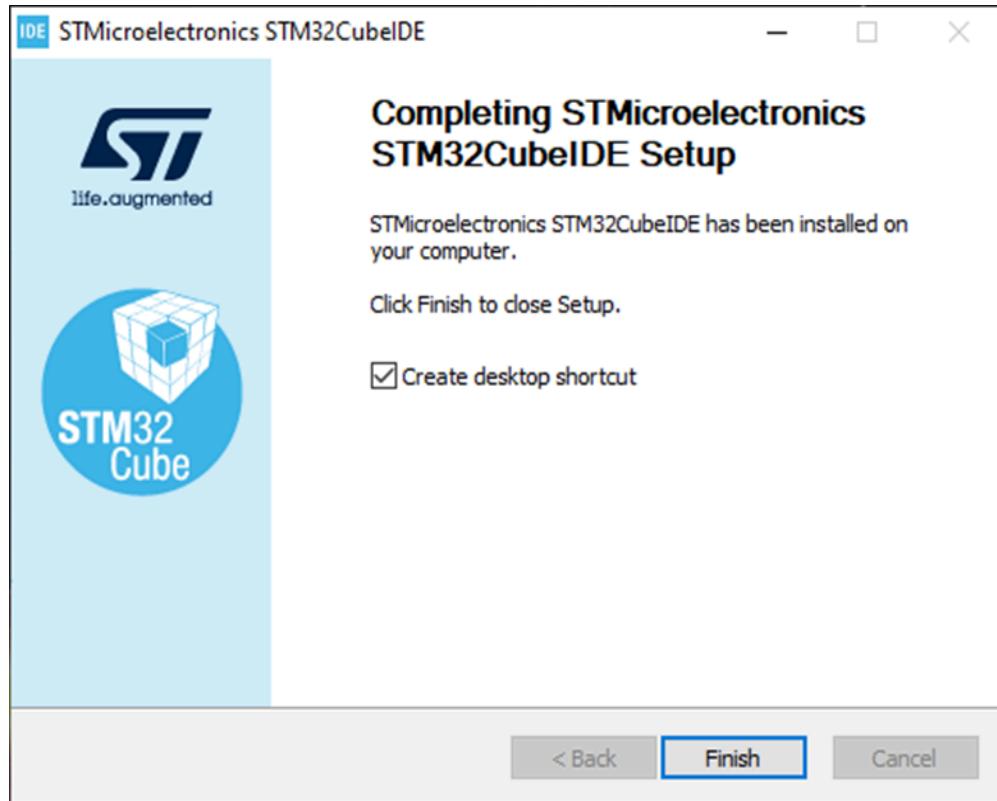


Por defecto viene seleccionada la dirección donde se va a instalar el programa:





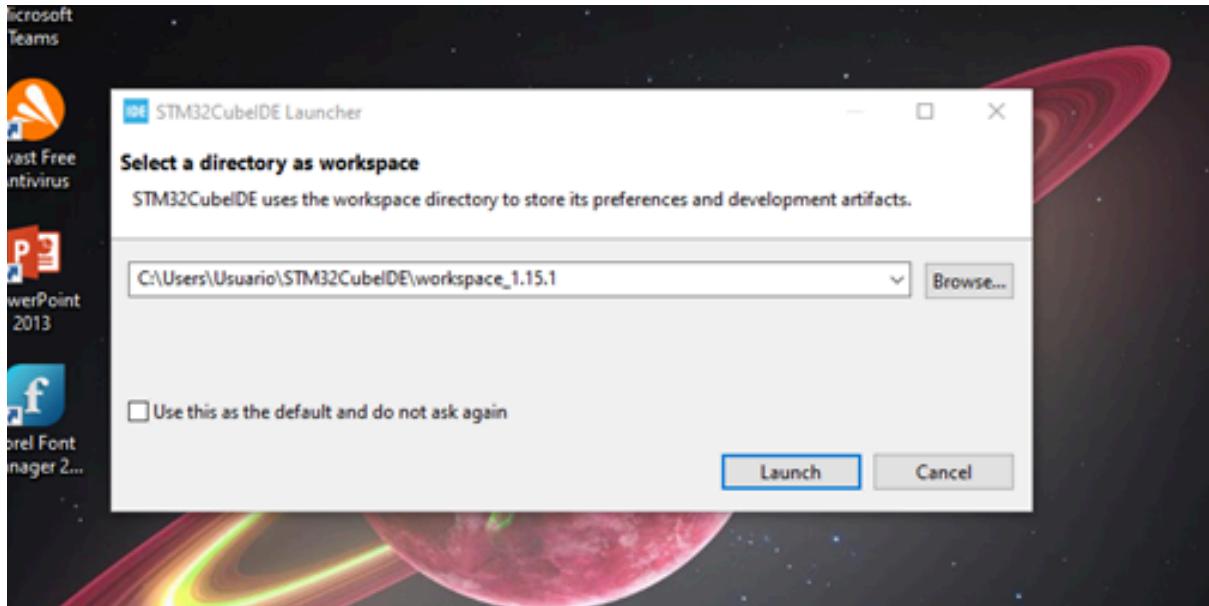
Al terminar la instalación damos click en next, donde tendremos la opción de crear un acceso directo seleccionando dicha opción, luego de eso damos click en finish.



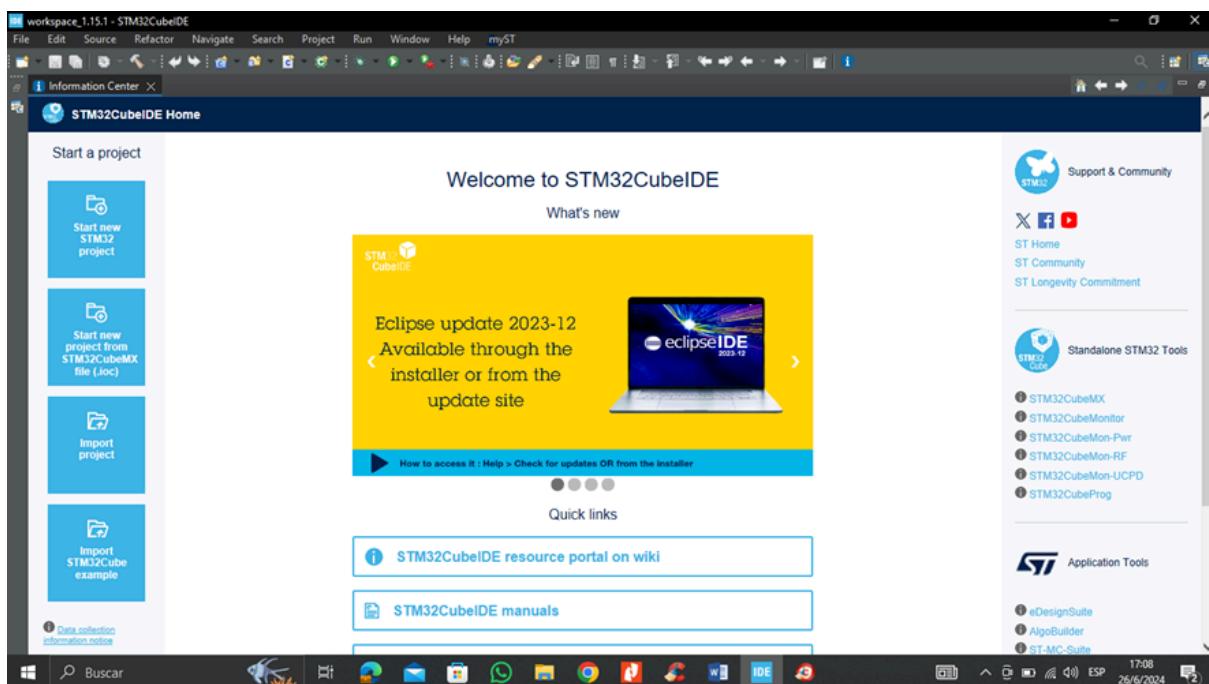
Luego de eso se creará el acceso directo:



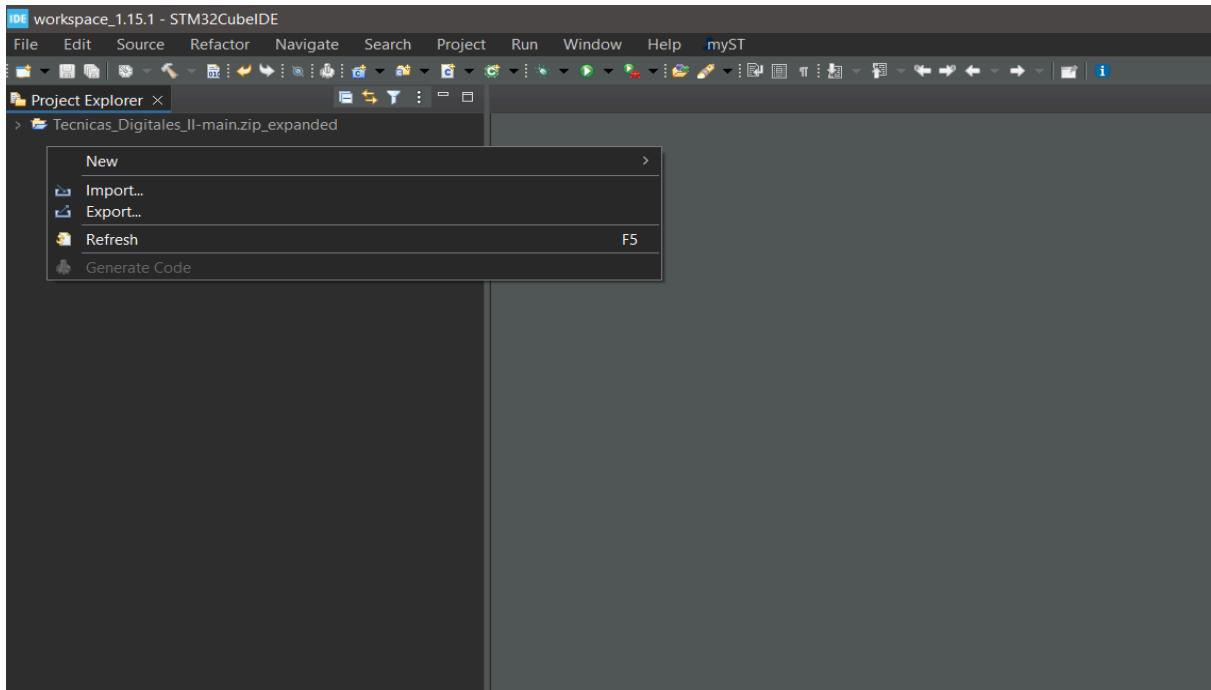
Como siguiente paso, abrimos el programa en el cual nos pregunta donde queremos guardar nuestro workspace, pero que por defecto ya tiene configurado una dirección en el disco c.



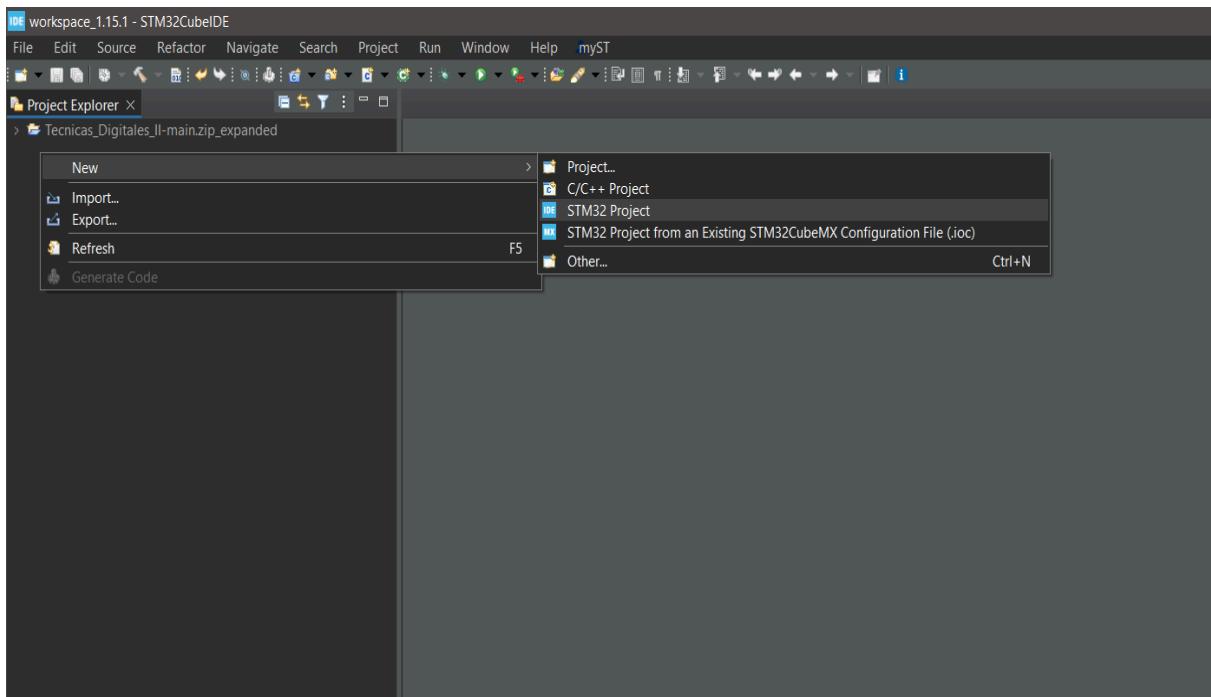
Damos a launch y se abrirá el programa



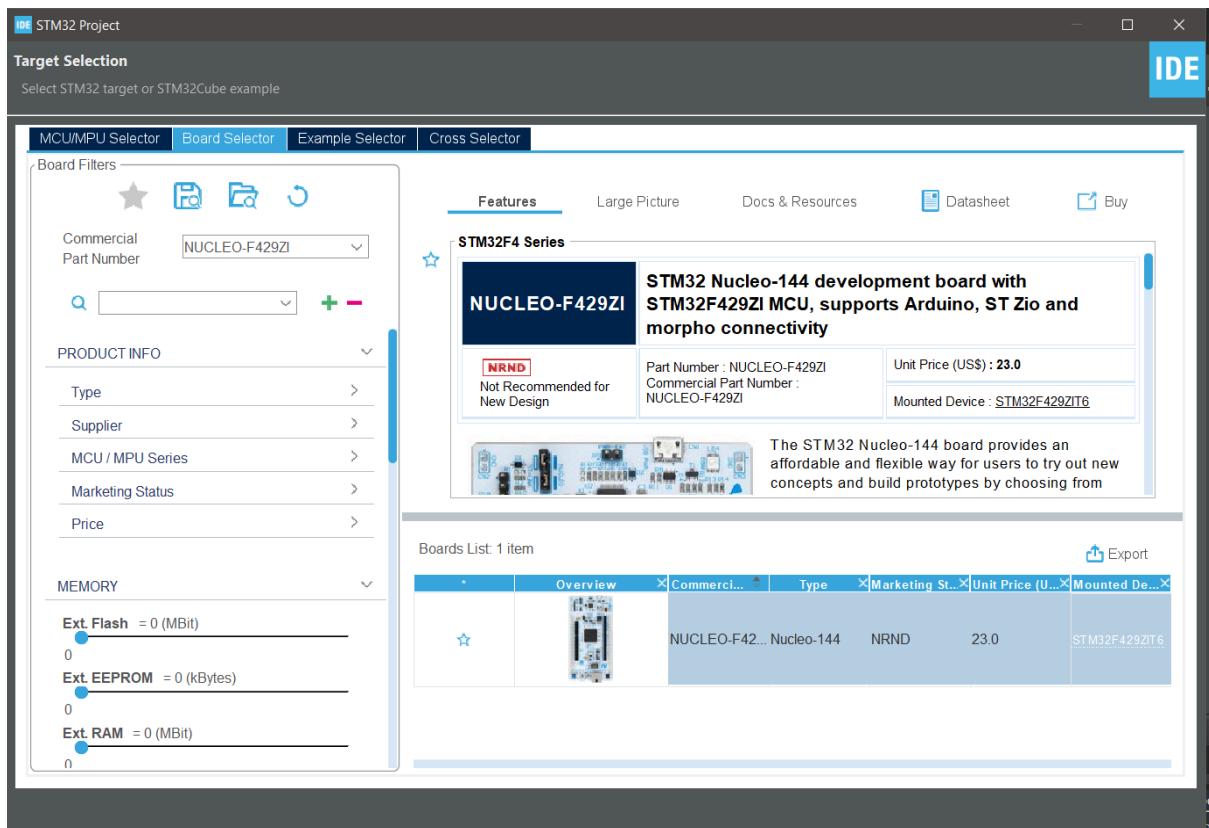
Una vez que el programa STM32CubeIDE se abre, en el project explorer, en el espacio vacío hago click derecho donde pongo el new.



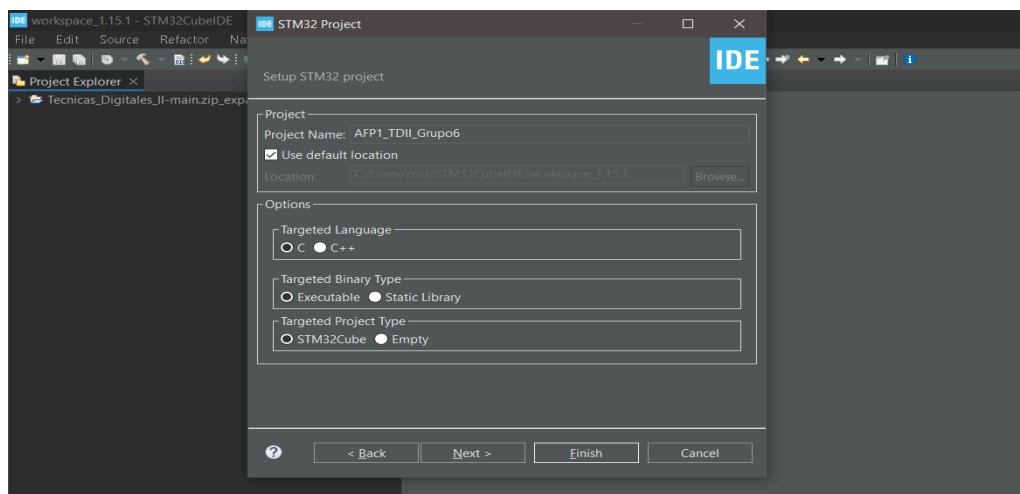
Luego elijo el STM32 project donde a continuación se abrirá una ventana



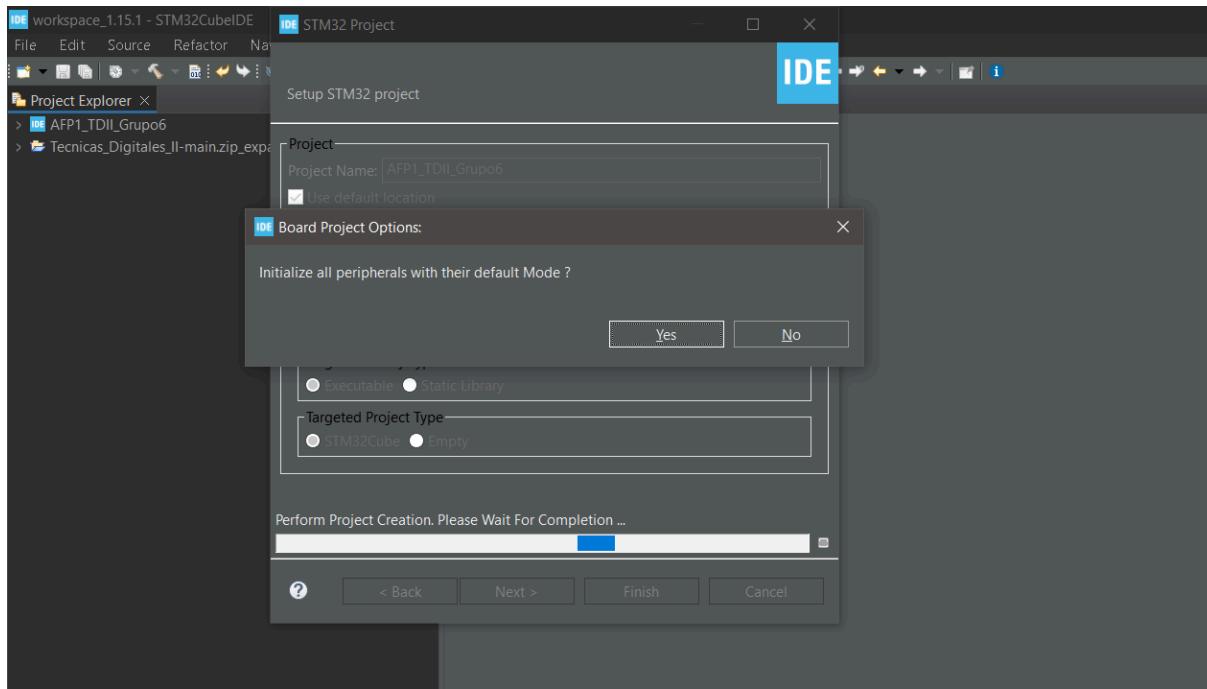
Al abrirse esta ventana (Target Selection), por defecto se pone en MCU/MPU Selector, lo que debemos hacer es cambiarlo a **Board Selector** y luego en donde dice Comercial Part Number escribiremos el nombre de la placa con la cual vamos a trabajar; en este caso es la **NUCLEO-F429ZI**, la seleccionamos y por último hacemos click en next.



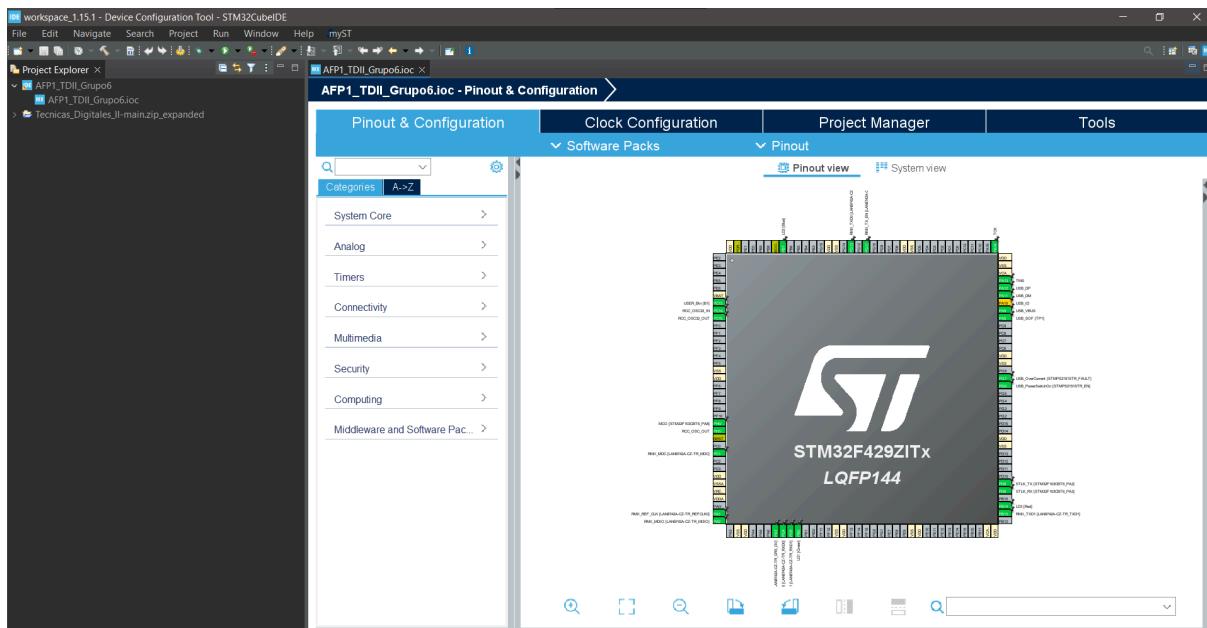
Se nos abre esta ventana mas pequeña, en la cual asignamos un nombre al proyecto y configuraremos el programa de la siguiente manera:



Como último paso, al hacer click en **Finish**, nos aparecerá un cartel preguntándonos si queremos inicializar con los periféricos que vienen del modo default, en el cual debemos hacer click en **Yes**.



Ya con todos estos pasos realizados, lo siguiente es que se nos abrirá la ventana de “Pinout & Configuration”

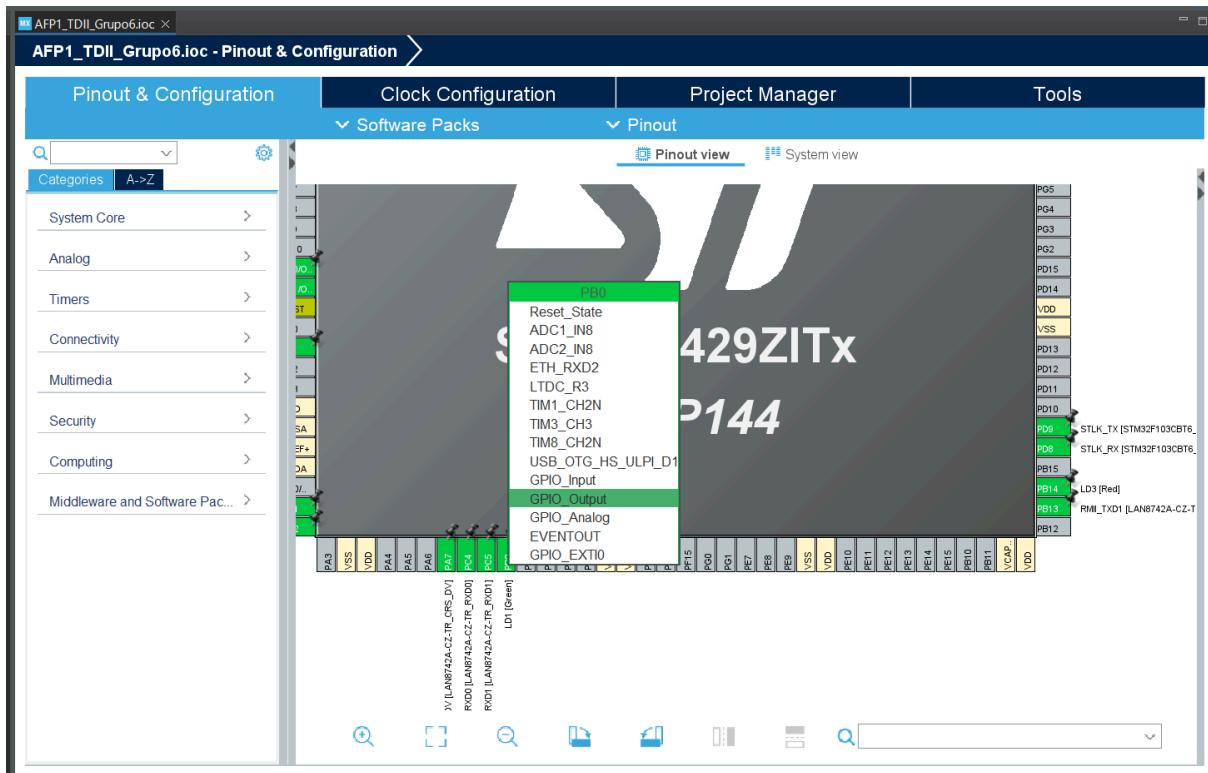


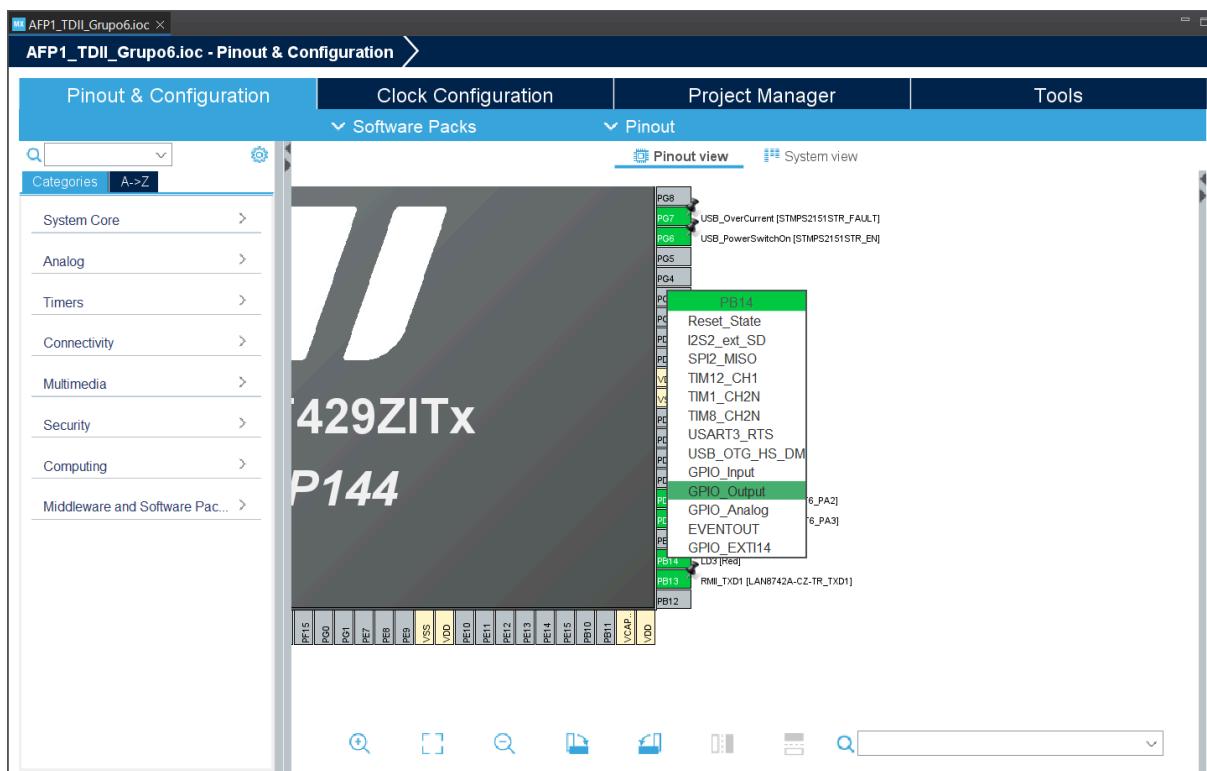
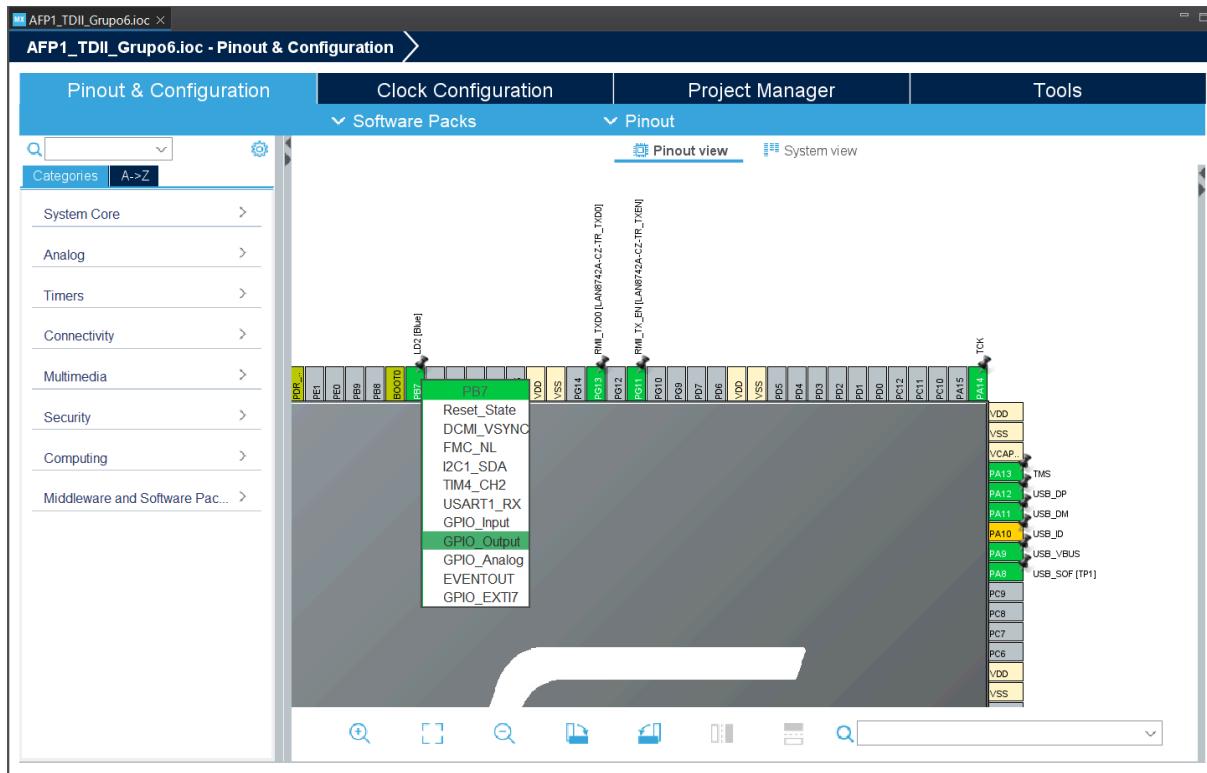
Configuración del Módulo GPIO

En el configurador de pines, selecciona los pines correspondientes a los LEDs y al pulsador de usuario. La placa NUCLEO-F429ZI tiene LEDs integrados en los siguientes pines:

- LED1 (LD1): Pin PB0
- LED2 (LD2): Pin PB7
- LED3 (LD3): Pin PB14

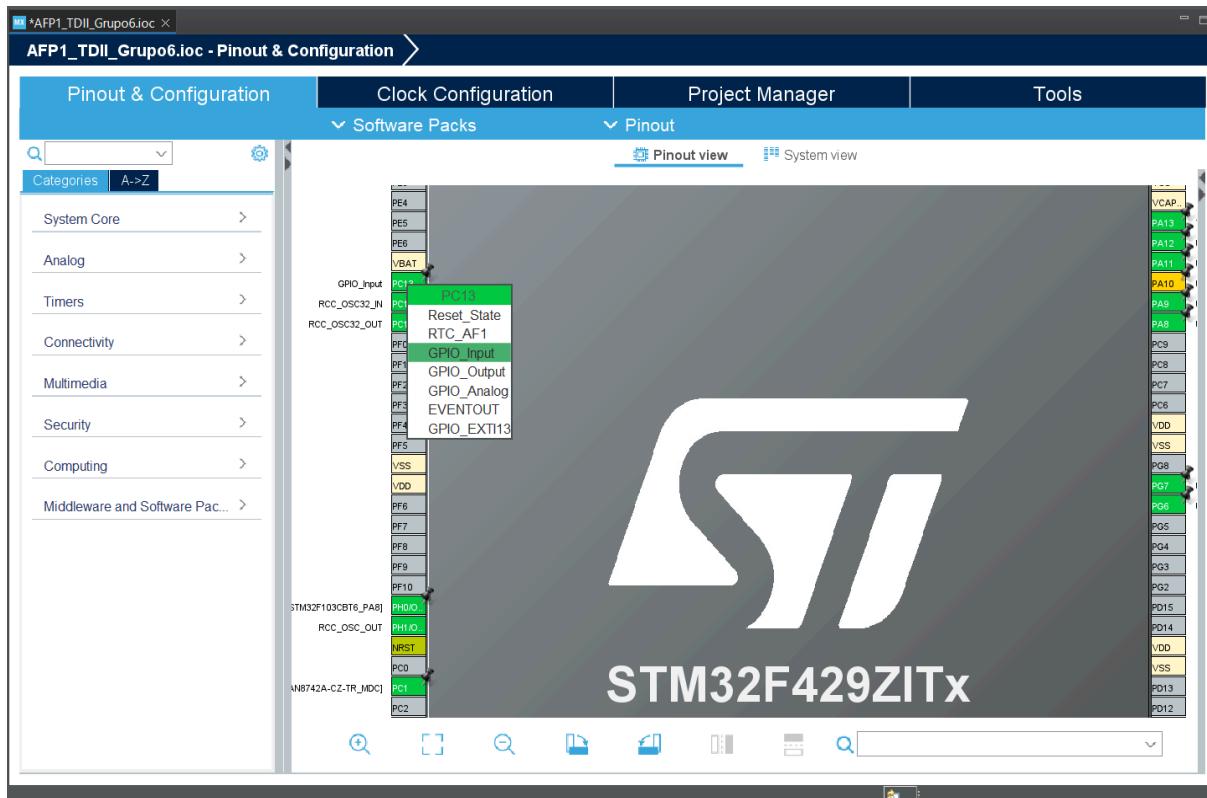
En el configurador de pines, hacemos clic en los pines PB0, PB7 y PB14 y seleccionamos **GPIO_Output**.





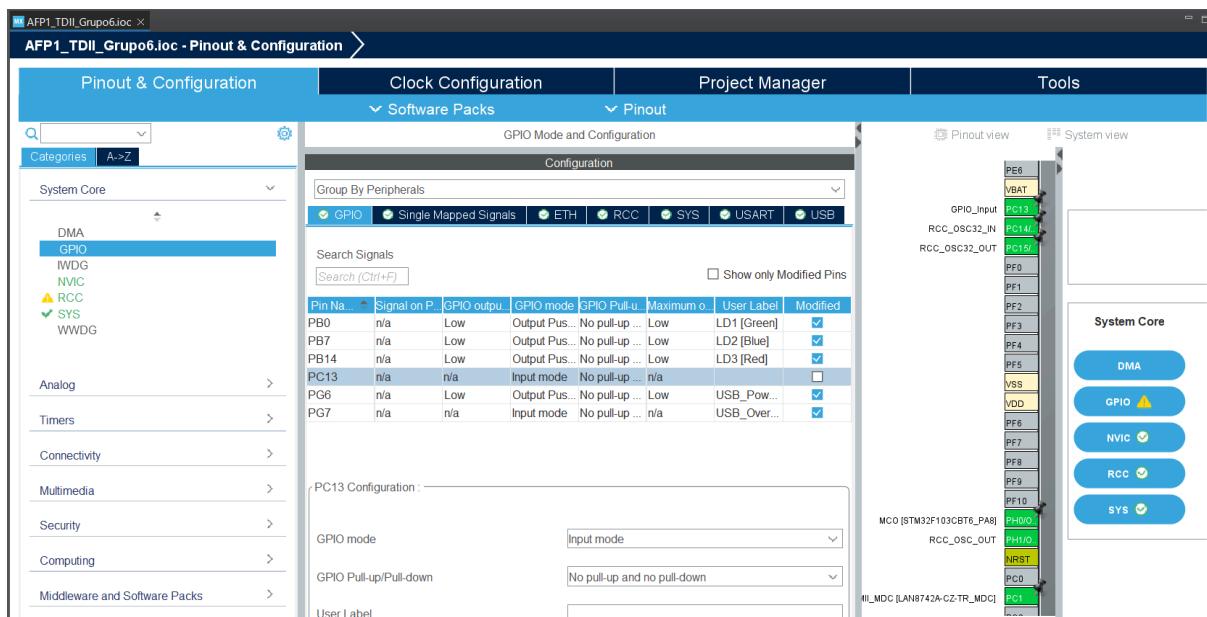
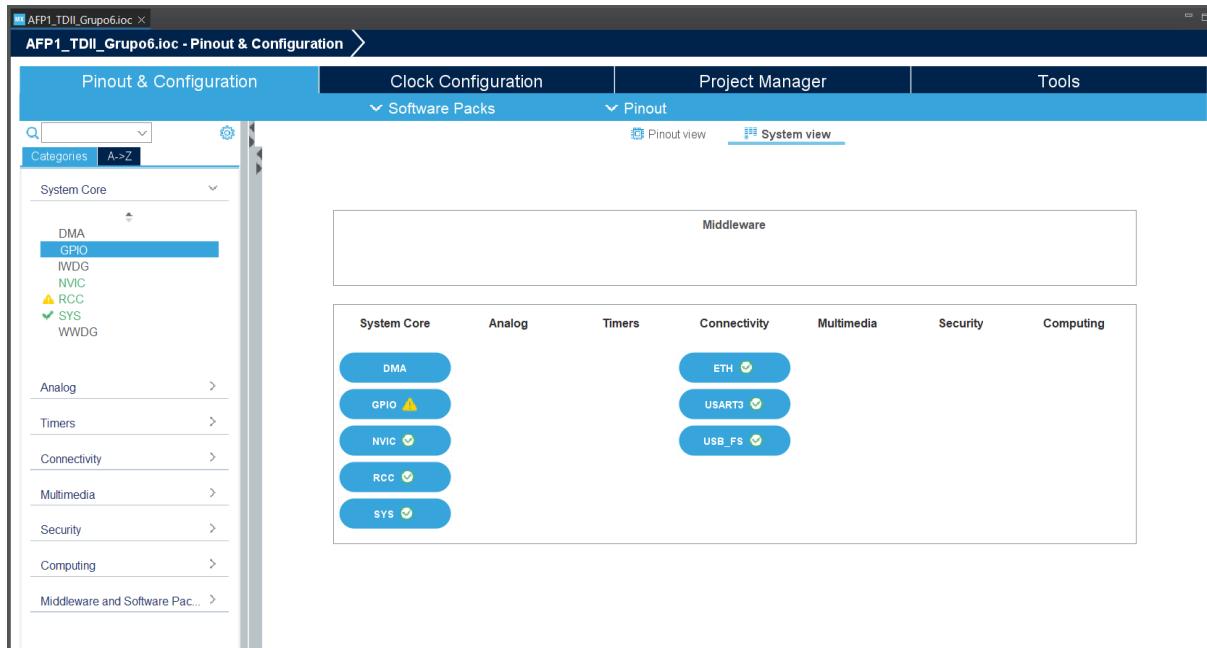
Configuración del Pin para el Pulsador de Usuario:

- El pulsador de usuario (B1) está conectado al pin PC13.
- En el configurador de pines, se hace clic en el pin PC13 y se selecciona **GPIO_Input**.



Configuración Adicional:

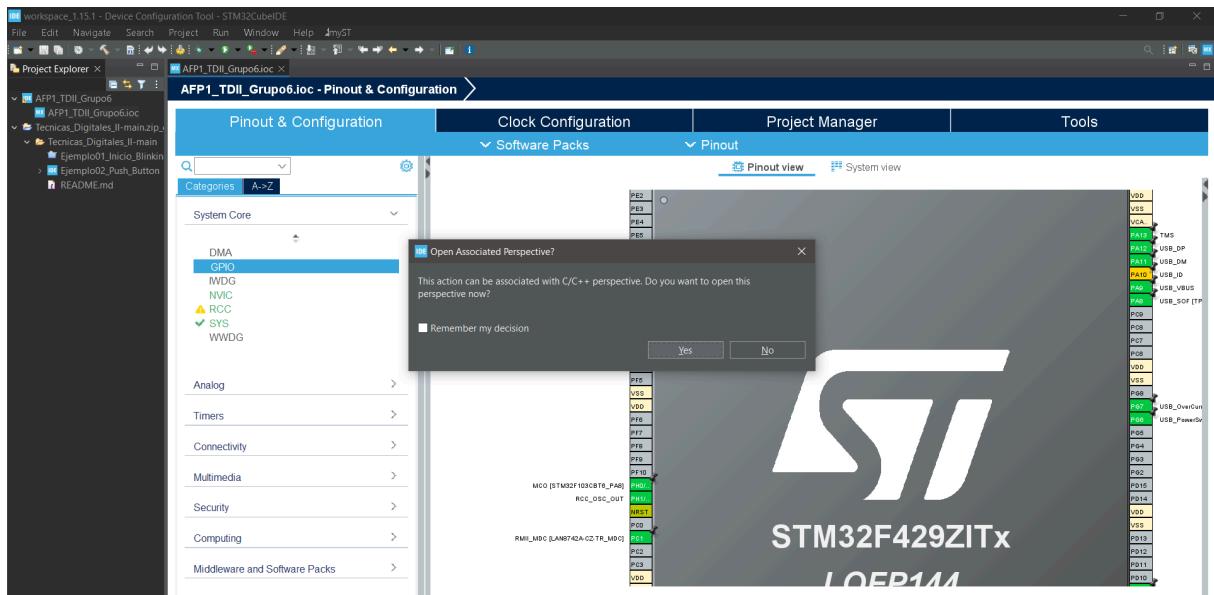
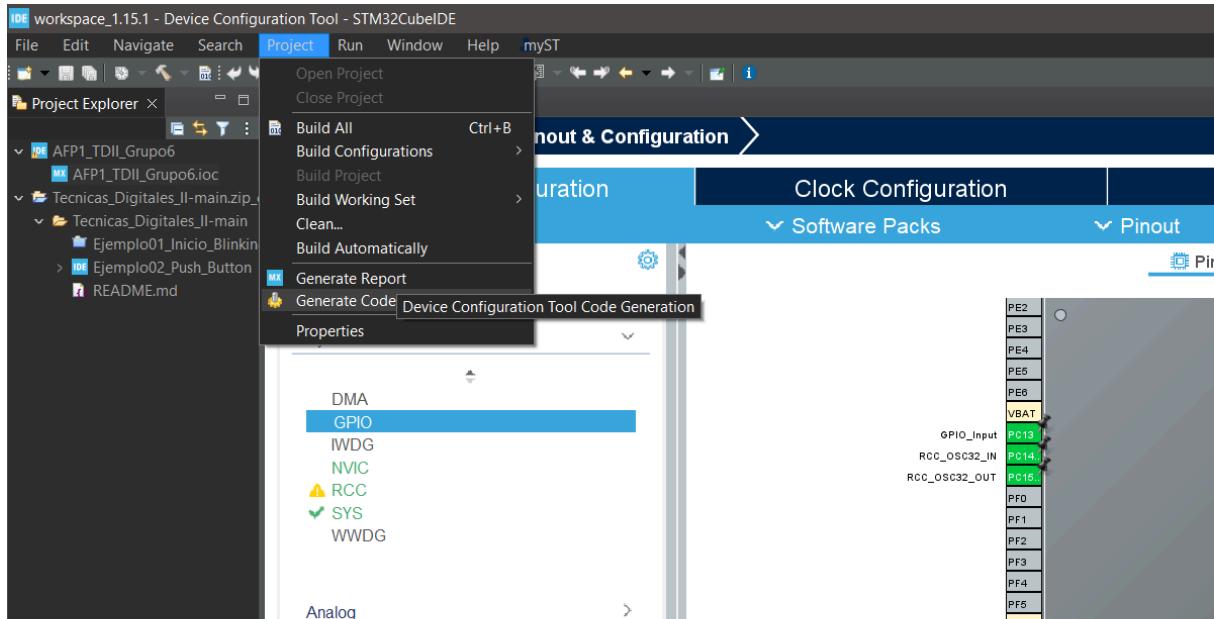
- En el panel de configuración de **GPIO**, nos aseguramos de que los modos de los pines estén configurados correctamente:
 - LEDs: **GPIO_Output** con **Push-Pull**.
 - Pulsador: **GPIO_Input**.



Generación del Código

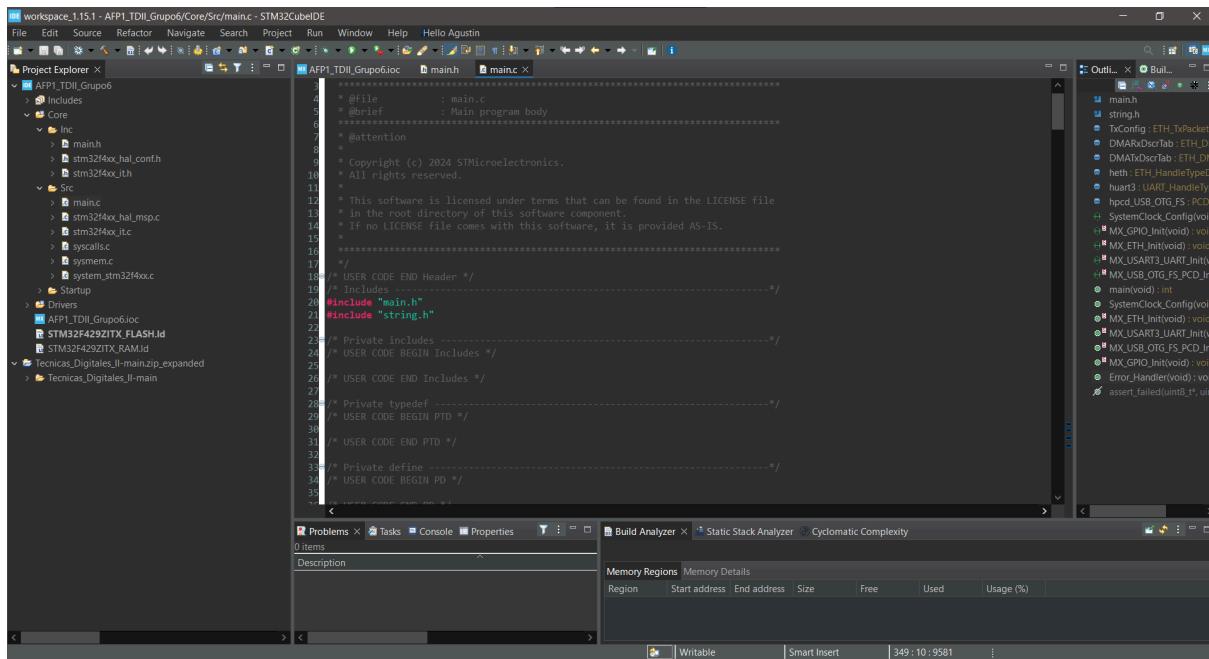
Generar el Código Inicial:

- Hacemos clic en el botón Project -> Generate Code.



Abrir el Archivo main.c:

- Navega hasta el archivo **main.c** en el árbol del proyecto.



The screenshot shows the STM32CubeIDE interface. The Project Explorer on the left lists the project structure, including the Core, Src, and Startup folders, along with various header and source files like main.h, main.c, and stm32f4xx_hal_conf.h. The code editor on the right displays the content of main.c, which includes copyright notices, includes for main.h and string.h, and defines for USER CODE BEGIN and END sections. The interface also includes toolbars, status bars, and other standard IDE components.

Estructura del Árbol de Archivos en STM32CubeIDE

1. Includes

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

2. Core

o 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_Grupo6.ioc**

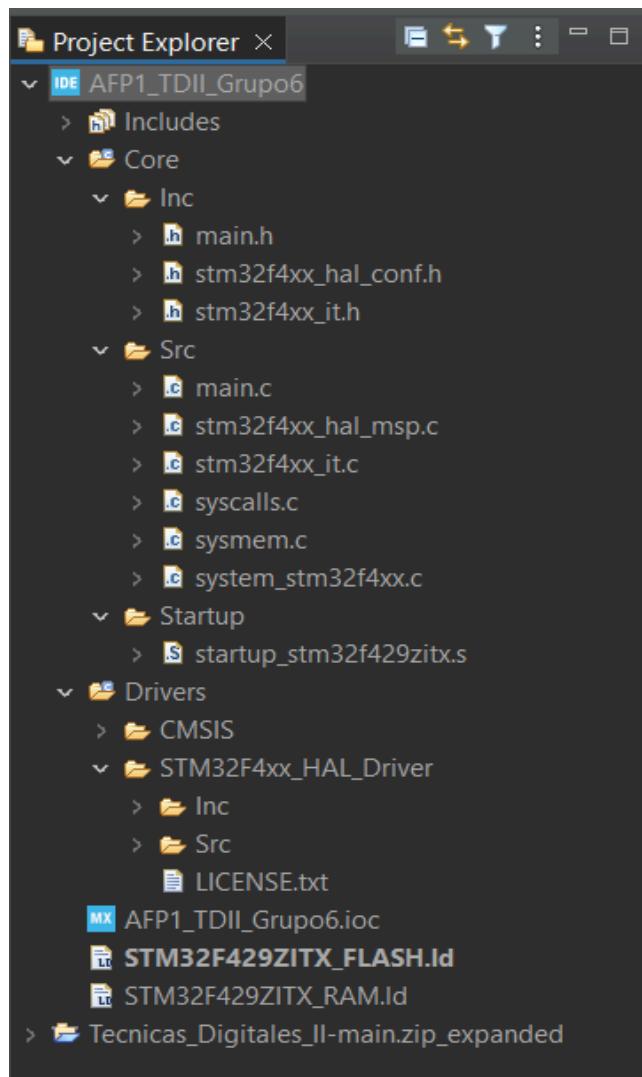
- Archivo de configuración del proyecto generado por STM32CubeMX. Contiene toda la configuración de los periféricos y pines del microcontrolador.

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.



Compilación del Proyecto en STM32CUBEIDE

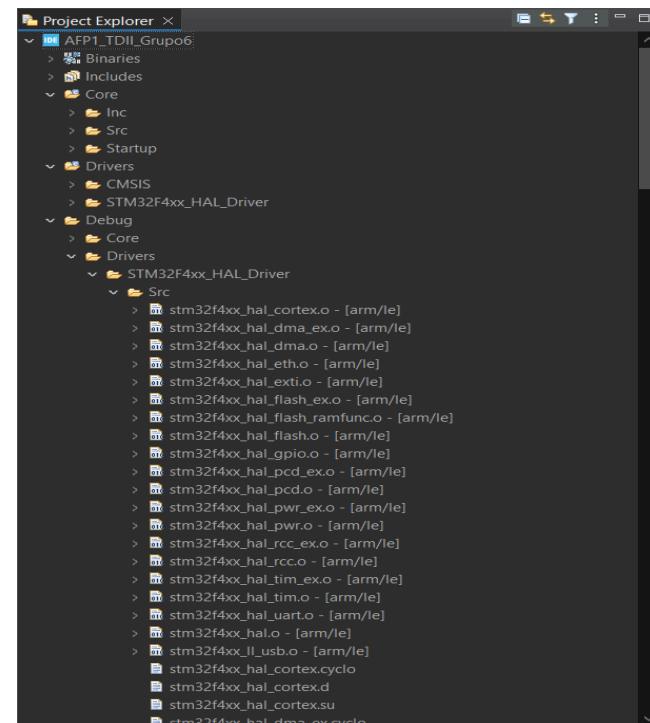
1. **Seleccionar el Proyecto:** En el panel de **Project Explorer**, seleccionar el proyecto que se desea compilar.
2. **Compilar el Proyecto:** Se puede compilar el proyecto de dos maneras:
 - Haciendo clic derecho en el proyecto y seleccionando **Build Project**.
 - O usando el atajo de teclado **Ctrl + B**.

Archivos Generados

Durante la compilación, STM32CUBEIDE genera varios archivos, incluyendo el archivo objeto y otros archivos de salida:

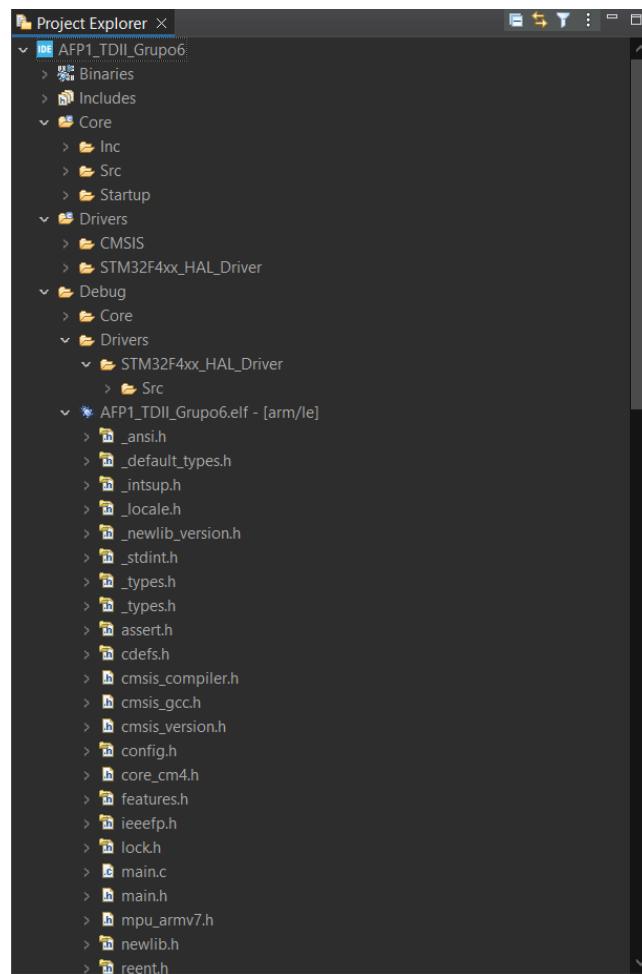
1. Código Objeto:

- **Archivo OBJ:** Los archivos objeto tienen la extensión **.o** (o **.obj** en algunos sistemas). Estos archivos contienen el código máquina generado a partir de tu código fuente y son usados por el enlazador para crear el archivo binario final.
- **Ubicación:** Los archivos objeto se encuentran en el directorio **Debug** o **Release**, dependiendo del modo de compilación. En la estructura de archivos de tu proyecto, la ruta suele ser algo como:



Archivo Binario Final:

- **Archivo ELF:** El archivo final es el archivo ejecutable en formato ELF ([.elf](#)). Este archivo contiene el código ejecutable y datos necesarios para la programación del microcontrolador.
- **Ubicación:** El archivo ELF se encuentra en el mismo directorio [Debug](#) o [Release](#).



Desarrollo de un programa que haga parpadear uno de los leds de la placa

Para hacer el desarrollo del código nos dirigimos hacia el archivo main.c en el cual escribiremos las líneas del correspondiente código.

Dentro del main.c tenemos que ir hacia el ciclo infinito o loop while(true) en donde escribiremos las funciones que hagan parpadear el led de la placa.

Primero debemos ubicar en el archivo .ioc cual es el pin de uno de los leds, en este caso utilizamos el led1 que está ubicado en el puerto B y su pin es el número 0, por lo tanto tendremos que PB0 es el led1 en este caso el led verde.

Mediante la función de HAL_GPIO_Toggle () hacemos que el led se prenda y se apague constantemente y con la función HAL_Delay() especificamos el tiempo de retraso o frecuencia de parpadeo del led, que en este caso son 250 ms.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
    HAL_Delay(250);
    /* USER CODE BEGIN B */
}
/* USER CODE END B */
```

Luego se compila para saber si hay algún error , y si todo está bien, se carga el programa en la placa una vez conectada ésta a la computadora, y se corre el programa para visualizar el parpadeo del led verde.

Link del proyecto que utiliza uno de los leds integrados en la placa y lo haga parpadear cada 250 milisegundos (GITHUB):

https://github.com/codecuellar/Grupo_6_TDII_2024/tree/master/Core/Src

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

1. Configuración del Debugger

Para configurar el debugger en STM32CubeIDE y depurar el código en la placa NUCLEO-F429ZI, hay que seguir estos pasos:

1. Conectar la Placa:

- Asegurarse de que la placa NUCLEO-F429ZI esté conectada a la computadora a través del cable micro USB. Verificar que la placa sea detectada por el sistema operativo.

2. Abrir el Proyecto:

- Abrir el proyecto en el que se está trabajando en STM32CubeIDE.

3. Configurar la Herramienta de Depuración:

- En el menú principal, seleccionar **Run > Debug Configurations**.
- En la ventana de configuraciones, seleccionar **STM32 Cortex-M C/C++ Application** en la lista de la izquierda y hacer clic en **New**.
- En la sección **Main**, asegurarse de que el nombre del proyecto y el archivo **.elf** estén correctamente seleccionados.
- Ir a la pestaña **Debugger** y seleccionar **ST-LINK (OpenOCD)** como interfaz de depuración. Verificar que la opción **Connect under reset** esté habilitada, lo cual es útil si la placa tiene algún problema al conectarse durante la depuración.
- En la sección **Startup**, se pueden configurar opciones como el retraso antes de la conexión o la opción de reiniciar la placa al final de la depuración, según sea necesario.

4. Iniciar la Depuración:

- Hacer clic en **Apply** para guardar la configuración y luego en **Debug** para iniciar la sesión de depuración.
- STM32CubeIDE cambiará automáticamente a la perspectiva de depuración, donde se podrá controlar la ejecución del programa, poner puntos de interrupción (breakpoints), y observar las variables en tiempo real.

2. Descarga del Código Objeto a la Placa Target

Una vez que se haya configurado el debugger, el siguiente paso es descargar el código objeto (archivo **.elf**) a la placa NUCLEO-F429ZI:

1. Compilación del Proyecto:

- Asegurarse de que el proyecto esté compilado correctamente. Para hacerlo, seleccionar **Project > Build Project** o presiona **Ctrl + B**.

2. Cargar el Código en la Placa:

- Con la placa conectada y detectada por STM32CubeIDE, seleccionar **Run > Debug** o presiona **F11**.
- STM32CubeIDE comenzará el proceso de descarga del código objeto a la placa target. Una vez finalizada la descarga, la ejecución del programa comenzará automáticamente en modo de depuración.

3. Monitoreo y Control:

- En la perspectiva de depuración, se puede monitorear la ejecución del código, detener la ejecución en los puntos de interrupción establecidos, y revisar el estado de las variables y los registros en la placa.

4. Finalizar la Depuración:

- Para finalizar la sesión de depuración, seleccionar **Terminate** en la barra de herramientas de depuración.

LINKS DE GITHUB

1. Link de repositorio: https://github.com/codecuellar/Grupo_6_TDII_2024/tree/master
2. Link de APP 1.1: https://github.com/codecuellar/Grupo_6_TDII_2024/tree/master/App_1_1_Grupo_6_2024
3. Link de APP 1.2: https://github.com/codecuellar/Grupo_6_TDII_2024/tree/master/App_1_2_Grupo_6_2024
4. Link de APP 1.3: https://github.com/codecuellar/Grupo_6_TDII_2024/tree/master/App_1_3_Grupo_6_2024
5. Link de APP 1.4: https://github.com/codecuellar/Grupo_6_TDII_2024/tree/master/App_1_4_Grupo_6_2024

Conclusión

El desarrollo de este proyecto práctico de laboratorio ha sido una buena experiencia que nos permitió profundizar en el uso del STM32CubeIDE y la programación en microcontroladores STM32. Durante el proceso, enfrentamos y superamos varios desafíos que nos ayudaron a consolidar nuestro conocimiento.

Uno de los primeros problemas surgió durante la configuración inicial del entorno de desarrollo STM32CubeIDE. Al instalar el software, encontramos dificultades para integrar las bibliotecas necesarias y configurar el IDE para trabajar correctamente con la placa NUCLEO-F429ZI. Sin embargo, resolvimos estos inconvenientes revisando detalladamente la documentación y recurriendo a foros técnicos.

Otro reto significativo fue al intentar configurar correctamente el debugger. Hubo problemas para establecer la conexión entre el IDE y la placa, lo que requirió ajustes en las opciones de depuración, como habilitar la conexión bajo reset. Este paso fue esencial para poder realizar un seguimiento eficiente del comportamiento del código durante su ejecución en la placa.

En cuanto al desarrollo de las aplicaciones solicitadas, uno de los principales desafíos fue implementar la lógica de alternancia entre secuencias en las aplicaciones que involucraban el uso del pulsador.

Finalmente, la creación del repositorio en GitHub y la correcta gestión de versiones también requirió atención para asegurar que todos los archivos se subieran correctamente y que el repositorio mantuviera un formato ordenado y coherente. Fue crucial para garantizar que las aplicaciones estuvieran disponibles y funcionaran correctamente en cualquier entorno.

En resumen, aunque enfrentamos diversas dificultades técnicas durante el desarrollo de este proyecto, logramos superarlas a través de un trabajo colaborativo y una investigación exhaustiva. Este proceso no solo nos permitió cumplir con los objetivos propuestos, sino que también nos dejó una valiosa experiencia en la resolución de problemas y la implementación de soluciones prácticas en un entorno de desarrollo real.