UTN Universidad Tecnológica Nacional

Facultad Regional Tucumán

Departamento Electrónica

Técnicas Digitales II

Actividad de Formación Práctica 2

Tema: Entorno de Desarrollo Integrado STM32CubeIDE, programación de microcontorladores.

Profesor:

Ing. Rubén Darío Mansilla

ATTP:

Ing. Lucas Abdala

vencimiento:

30 de mayo de 2025

Año: 2025

1. Objetivos

- Que el alumno logre familiarizarse con el Entorno de Desarrollo Integrado (IDE) STM32CubeIDE, que usaremos como plataforma para desarrollar aplicaciones para sistemas embebidos basados en microcontroladores de 32 bits.
- Que el alumno logre experimentar un primer contacto con las herramientas de software, que se lanzan desde el IDE para el desarrollo de una aplicación, partiendo desde su edición, continuando con su compilación, depuración y terminando con su programación del *target* de la placa de desarrollo usada.
- Que el alumno logre experimentar la creación de nuevos proyectos basado en un proyecto ya existente y lo utilice como una plantilla para afrontar el desarrollo de un nuevo proyecto.
- Que el alumno se ejercite en las particularidades que presenta la programación de microcontroladores.
- Que el alumno acceda a la documentación que ofrece el sitio de desarrolladores de STM32CubeIDE y estudie los conceptos básicos necesarios para utilizar de la manera más eficiente posible la gran cantidad de herramientas, módulos y librerías que ofrece este paquete de software que se encuentra en permanente evolución.

2. Generalidades

Esta actividad de formación práctica es del tipo **práctica de laboratorio**.

Cómo desarrollar esta actividad:

- Elaborar un informe en el que desarrollará el tutorial que se indica en el punto 1 del capítulo 3: Consignas para desarrollar.
- Desarrollar las aplicaciones que se indican en el punto 2 del capítulo 3. Cada aplicación debe estar desarrollada en un proyecto y subida a un repositorio de GitHub.
- Insertar el link al repositorio de GitHub en el desarrollo del informe a continuación del tutorial siguiendo el orden indicado en cada punto de las consignas de esta actividad de formación práctica.
- Presentar, de manera grupal, las aplicaciones funcionando sin errores y explicar cómo se desarrolló y cómo trabaja el software.

Nota 1: Para el desarrollo de las aplicaciones de esta actividad de formación práctica se utilizará el entorno de desarrollo integrado STM32CubeIDE para aplicar sobre una placa de desarrollo STM32-NUCLEO-F4XX-YY.

Nota 2: Las placas de desarrollo sugeridas para las prácticas son: STM32-NUCLEO-F401-RE, STM32-NUCLEO-F412-ZG, STM32-NUCLEO-F413-ZH o STM32-NUCLEO-F429-ZI.

Nota 3: El desarrollo de las actividades de formación práctica se realizará y presentará de manera grupal.

3. Consignas para desarrollar

1. Elabore un tutorial en soporte digital, que explique cómo se aborda el uso del STM32CubeIDE, que contenga lo que se indica a continuación:
   - Instalación del software.
   - Creación de un proyecto nuevo desde cero que utilice la plantilla general que ofrece este IDE. Indique las configuraciones básicas para el uso del módulo GPIO correspondientes a los leds y el pulsador de usuario integrados en la placa de desarrollo que utilizaremos en este curso.
   - Descripción de la estructura del árbol de archivos de un proyecto en STM32CubeIDE y qué contiene cada carpeta de la misma.
   - Desarrollo de un proyecto que utilice uno de los leds integrados en la placa y lo haga parpadear cada 250 ms.
   - Explicación de cómo compilar el proyecto, cuál es el archivo que se genera (código objeto) y dónde se ubica en el árbol de archivos del mismo.
   - Explicación sobre cómo configurar el *debugger* para utilizarlo y cómo se baja el código objeto a la placa *target*.
   - El archivo que contiene el desarrollo de la actividad de formación teórica debe ser de tipo pdf y su nombre debe respetar el siguiente formato:
     AFP_1_Grupo_X_TDII_2025.pdf
2. Desarrolle las siguientes aplicaciones:
   2.1. **App 1.1:** Desarrolle una aplicación que encienda y apague de manera secuencial los tres leds *onboard* de la placa de desarrollo. La secuencia debe encender 200 ms y apagar 200 ms cada led comenzando por el LED1 (Green), continuando con el LED2 (Blue) y luego el LED3 (Red) para volver a iniciar con el LED1. La aplicación debe ser de carácter general, de manera que pueda extenderse a una cantidad mayor de leds con mínimas modificaciones, por este motivo se sugiere que use un vector para el manejo de los Leds.
   2.2. **App 1.2:** Desarrolle una aplicación que utilice el pulsador *onboard* de la placa de desarrollo para alternar entre dos secuencias diferentes. La aplicación inicia con la secuencia de la App 1.1 y, cuando se presione el pulsador, dicha secuencia debe invertirse y continuar, de manera que, cada vez que se presione el pulsador la secuencia actual se invierta. La aplicación debe ser de carácter general, por lo que aplica la misma recomendación para el punto anterior.
   2.3. **App 1.3:** Desarrolle una aplicación que utilice el pulsador *onboard* de la placa de desarrollo para alternar entre cuatro secuencias diferentes. La app iniciará con la secuencia 1 hasta que se presione el pulsador y pase a la secuencia 2, luego de presionar de nuevo el pulsador pasará a la secuencia 3 y así, sucesivamente, hasta la secuencia 4 para volver a comenzar con la secuencia 1. Descripción de las secuencias:
      - **Secuencia 1:** ídem App 1.1 con una alternancia de 150 ms.
      - **Secuencia 2:** hace parpadear los tres leds simultáneamente con una alternancia de 300 ms.
      - **Secuencia 3:** hace parpadear el LED1 con una alternancia de 100 ms, el LED2 con una alternancia de 300 ms y el led3 con una alternancia de 600 ms.
      - **Secuencia 4:** Hace parpadear simultáneamente LED1 y LED3, mientras que LED2 lo hará de manera inversa, con una alternancia de 150 ms. Mientras LED1 y LED3 están encendidos, LED2 estará apagado y luego a la inversa.

2.4. **App 1.4:** Desarrolle una aplicación que haga parpadear simultáneamente los 3 leds *onboard* de la placa y que use el pulsador *onboard* para cambiar la frecuencia de parpadeo de manera secuencial entre 4 frecuencias predefinidas por los siguientes tiempos de alternancia:

- **Tiempo 1:** 100 ms.
- **Tiempo 2:** 250 ms.
- **Tiempo 3:** 500 ms.
- **Tiempo 4:** 1000 ms.

La aplicación inicia con la alternancia determinada por el tiempo 1, al presionar el pulsador pasa al tiempo 2 y así sucesivamente hasta llegar al tiempo 4, para reiniciar con el tiempo 1.

3. Utilice el repositorio en GitHub:
   - Cree un repositorio en GitHub para su grupo cuyo nombre respete el siguiente formato: **Grupo_ X_ TDII_ 2025**.
   - Suba a este repositorio las aplicaciones desarrolladas para esta actividad de formación práctica respetando el siguiente formato de nombre: **App_ 1_Y_Grupo_X_2025**.
   - Pegue el link al repositorio para cada App en el informe que presente para esta actividad de formación práctica. Indique claramente a que App corresponde cada link.

   Nota sobre el punto 3: X es el número de su grupo y **Y** es el número de aplicación como figura en el enunciado.

4. Realice la presentación de las aplicaciones, en el laboratorio de Técnicas Digitales, funcionando según lo que pide la consigna, con su correspondiente defensa de lo desarrollado. Fecha a acordar.

4. Bibliografía y documentación

- Documentación accesible desde el IDE en:

  Help → Information Center → STM32CubeIDE Manuals

ESTUDIANTE: Roqué Luciano J..... Luciano J. Roqué

ESTUDIANTE: Cruz Rodolfo Martín .....

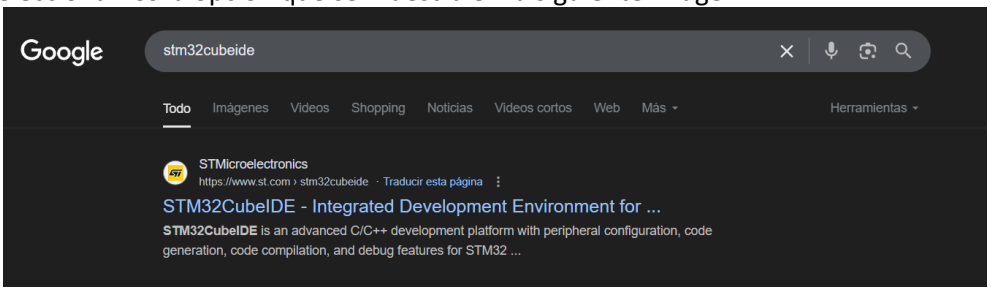
FECHA DE INICIO: __/__/__ FECHA DE PRESENTACIÓN: 3 0 /5 /25

CONFORMIDAD DEL DOCENTE:.....

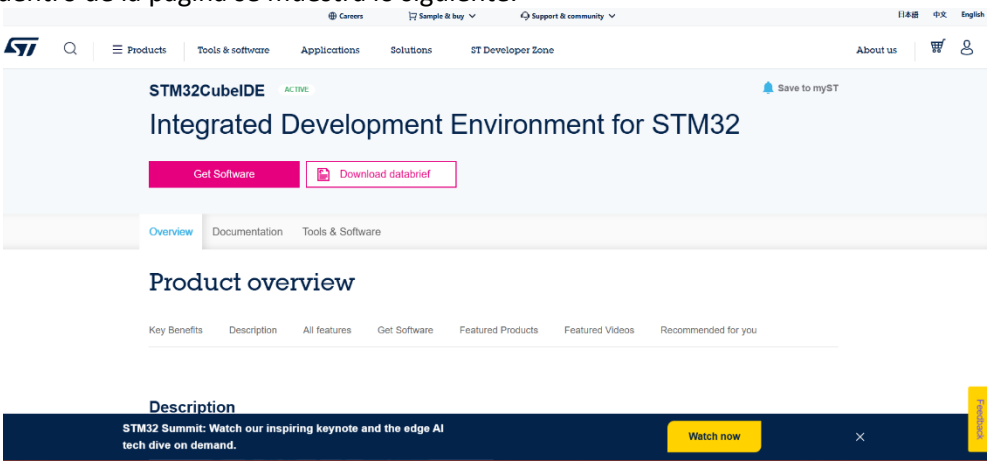
Tutorial

Instalación del software

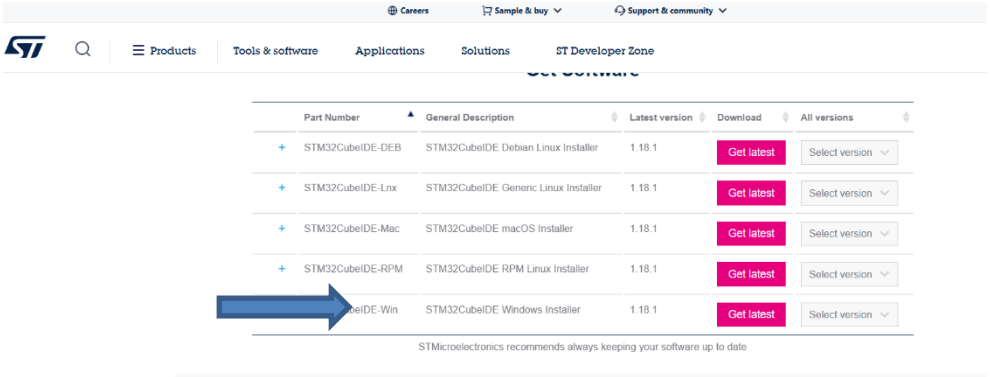
1º ingresamos al navegador, luego se procede escribir **stm32Cubeide**, al cargar los resultados de la búsqueda seleccionamos la opción que se muestra en la siguiente imagen**.**



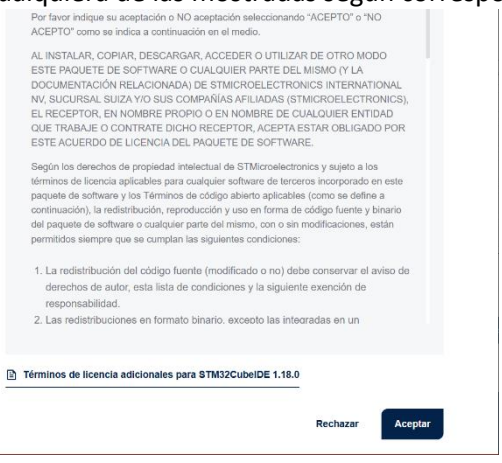
2º Una vez dentro de la página se muestra lo siguiente.

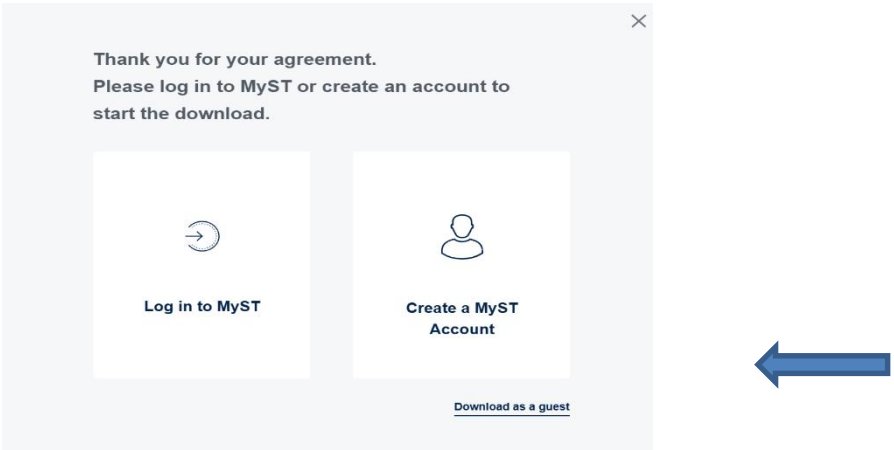


Al desplazarnos hacia abajo se verán las siguientes opciones, las misma depende del sistema operativo del ordenador. En este caso (particular) se opta por Windows.



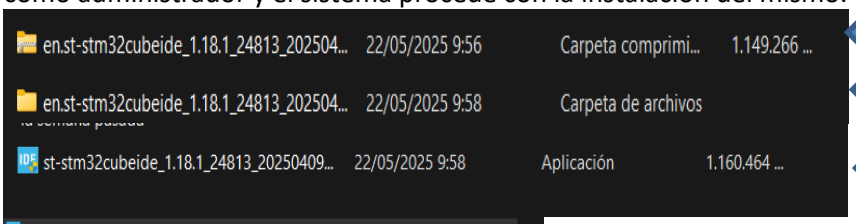
3º Una vez seleccionado esta opción (o cualquiera de las mostradas según corresponda), la pagina muestra lo siguiente.





En la primera imagen aceptamos los términos de licencia, esto nos llevara a otra ventana donde habrá tres opciones, en ese caso lo mejor sería optar por registrarse (en caso de que no lo estén) para evitar futuros problemas con el programa en caso de elegir la opción de invitado además si uno opta por registrase la página brinda ciertos beneficios como conferencia, libros e información sobre nuevos proyectos. Se debe tener en cuenta que, una vez realizado este paso, la página nos envía un correo (en algunos casos esta demora en llegar o bien el correo que nos llega es tomado como spam y eso no nos notifica el Gmail por eso es importante verificar ese detalle por cuenta propia) para terminar el proceso de registro.

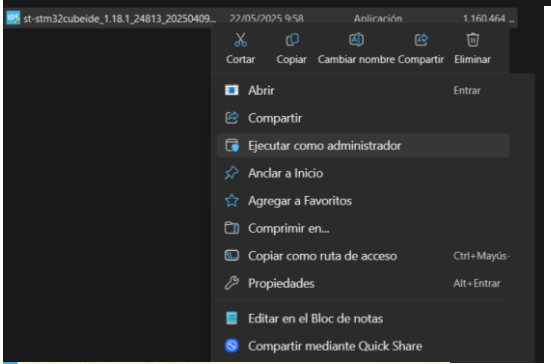
4º luego de terminar los pasos anteriores, podremos descargar el instalador, la cual se encuentra en un formato zip. La descomprimimos y esto generara una carpeta que contiene un archivo exe, la ejecutamos como administrador y el sistema procede con la instalación del mismo.

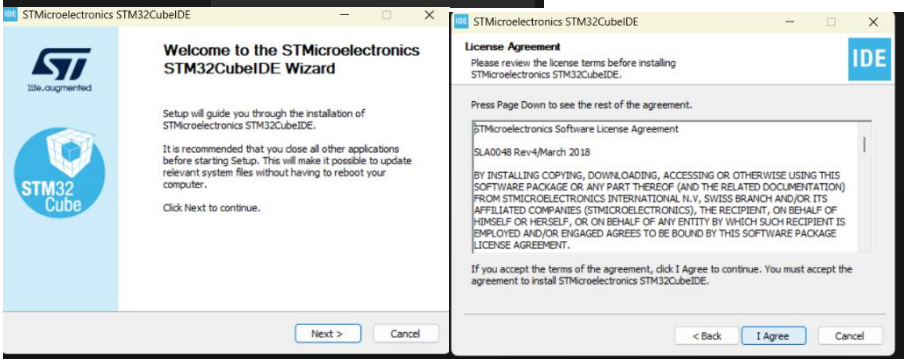


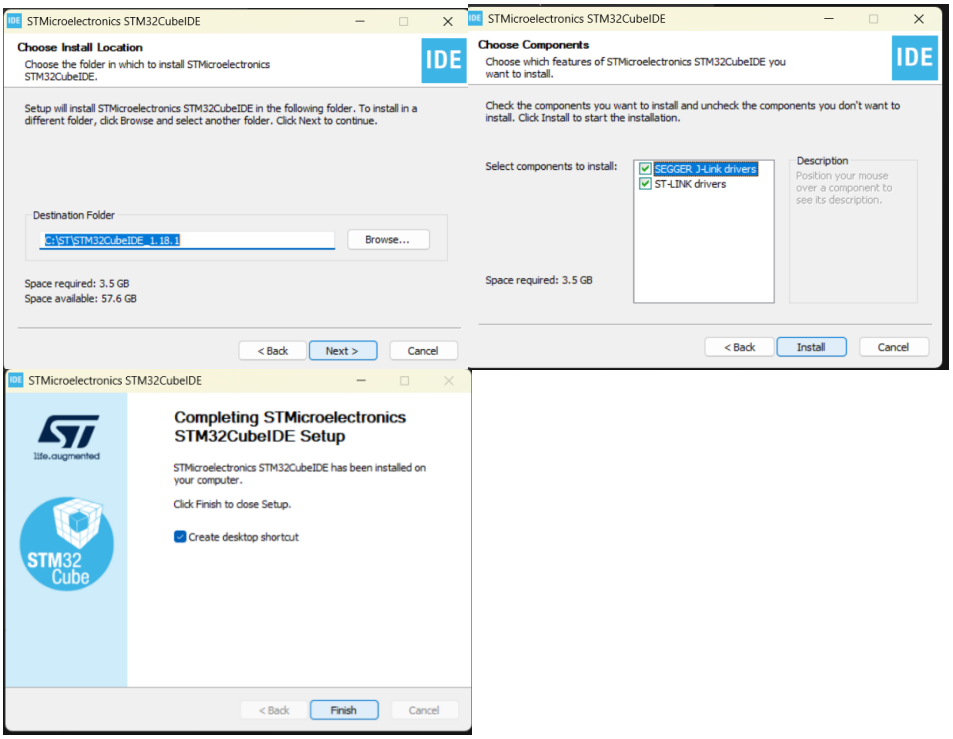
Archivo zip.

Carpeta generada.

Instalador.



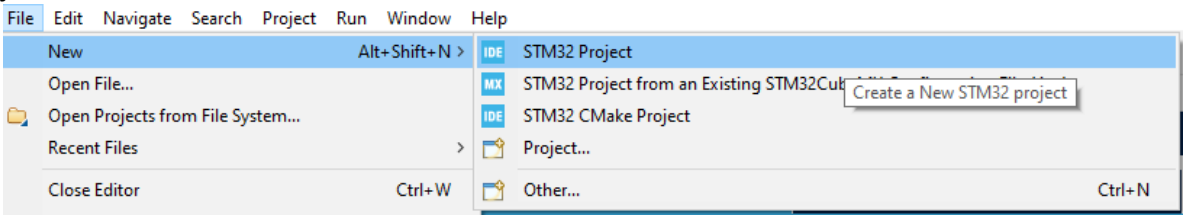




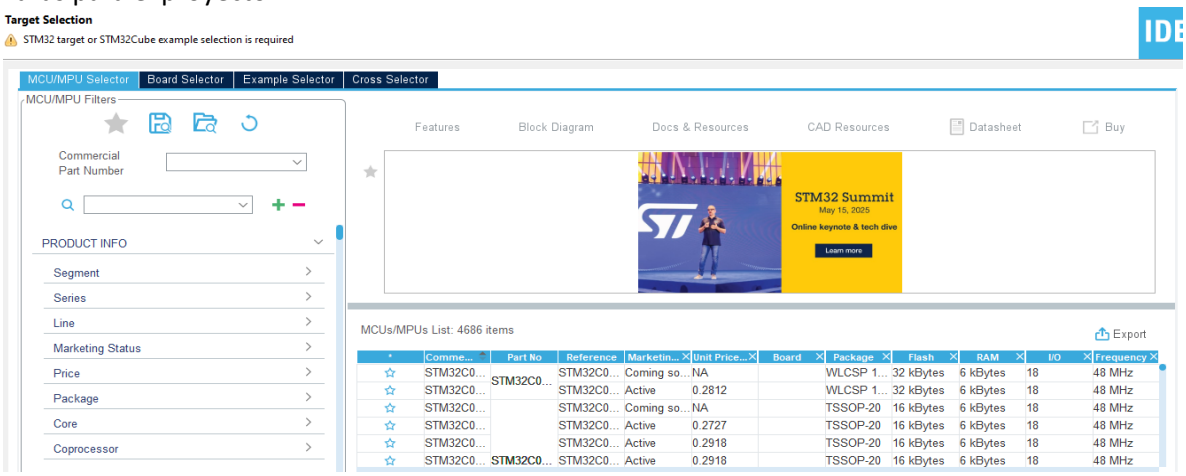
De este modo finaliza la instalacion del programa.

Creación de un proyecto nuevo

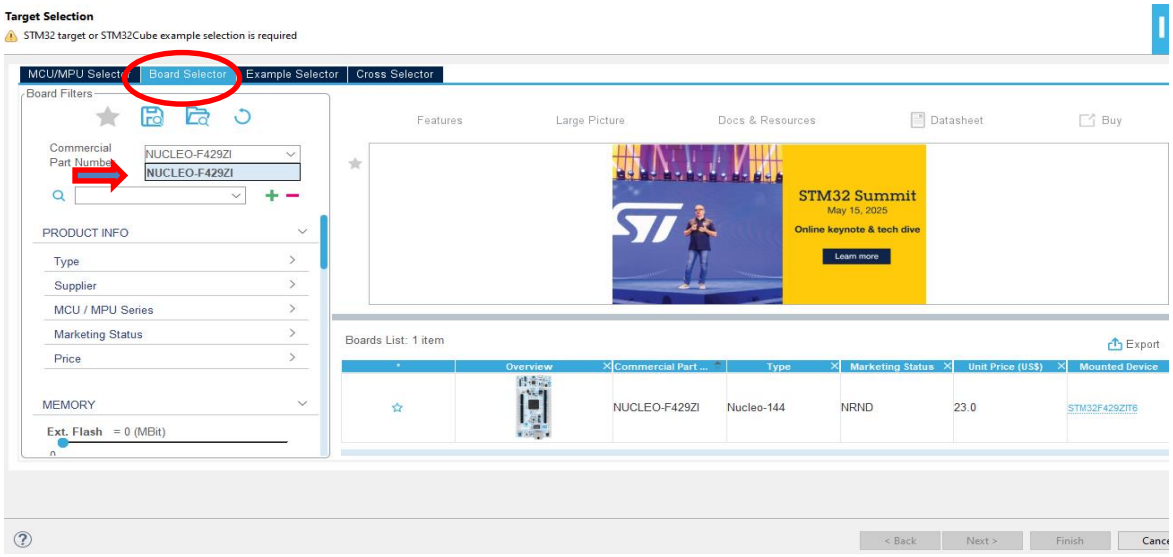
Inicializa el programa STM32CubeIDE. Luego para crear un proyecto, dirígete a **File → New → STM32 Project.**



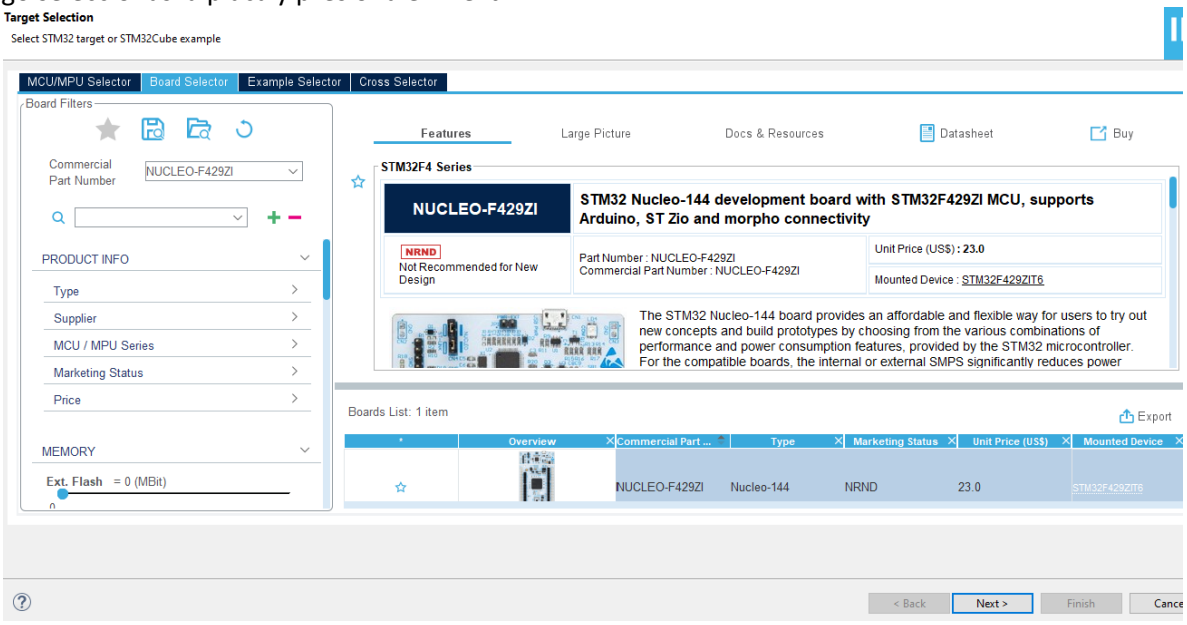
Se inicializará un menú como se observa en la imagen, en donde podrás buscar la placa de desarrollo que utilizarás para el proyecto.



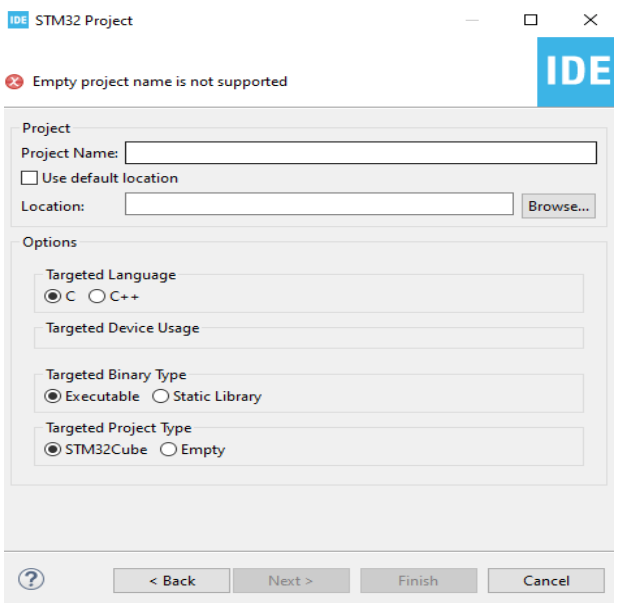
Para facilitar la búsqueda, selecciona la pestaña **Board Selector**, donde podrás filtrar por el modelo de tu placa. Luego en Comercial Part Number, colocas el modelo del núcleo de tu placa.



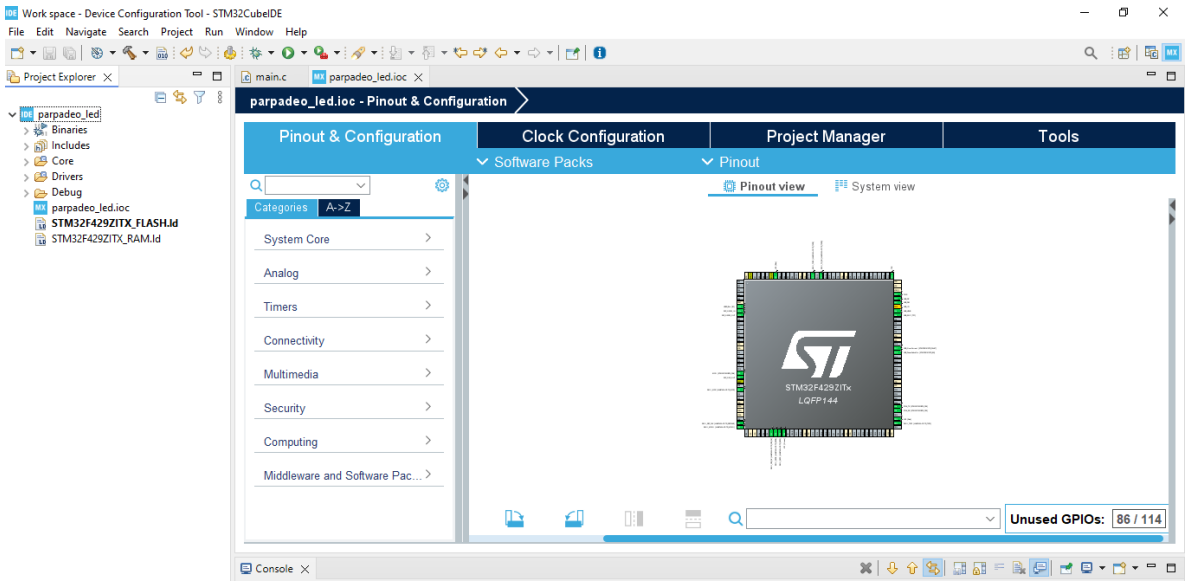
Luego seleccionas la placa y presiona en Next.



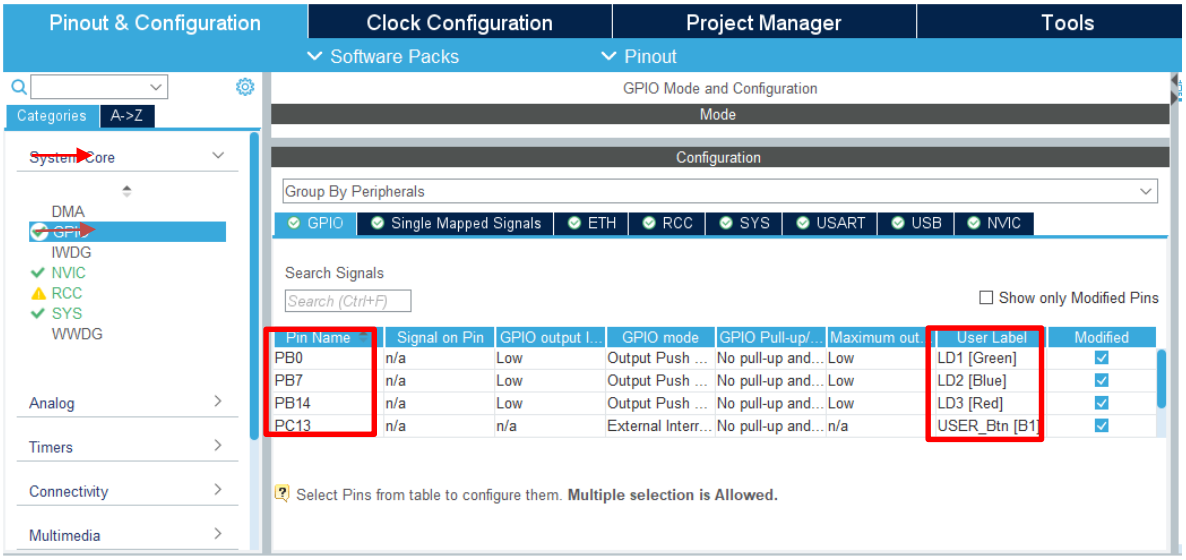
A continuación, aparecerá una ventana donde podrás ingresar el **nombre del proyecto** y seleccionar la **ruta de guardado**. Una vez hecho esto presiona en Finish.



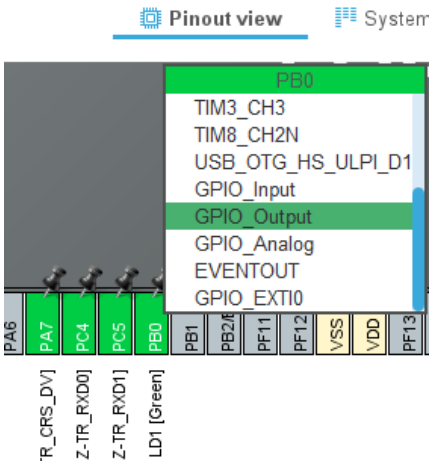
Una vez creado el proyecto, se abrirá automáticamente la vista general de **Pinout & Configuration,** donde podrás comenzar a configurar el microcontrolador según los requerimientos de tu aplicación.



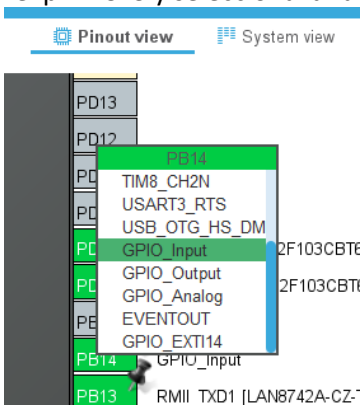
Creado el proyecto y seleccionada la placa, se debe configurar el módulo GPIO para poder utilizar los recursos integrados de la placa, como los LEDs y el pulsador de usuario. Para identificar a qué pines del microcontrolador están conectados los LEDs integrados en la placa, utilizamos la herramienta CubeMX. En la pestaña **System Core → GPIO**, podemos ver un listado detallado de los pines configurados. En la columna **User Label**, se indica a qué LED corresponde cada uno. También podremos ver el pin del pulsador de usuario.



Hacé clic sobre cada uno de los pines como el `PB0`, en el diagrama del microcontrolador. Selecciona la función **GPIO_Output.**

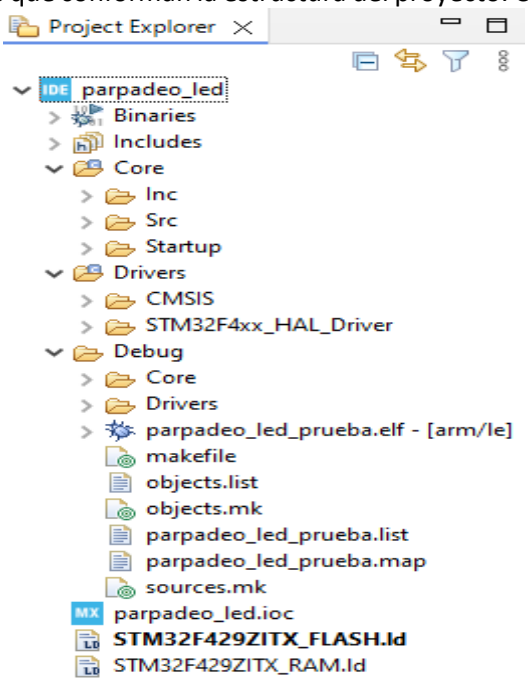


Para el pulsador se debe hacer click en el pin PC13 y selecciona la función GPIO_Input.



Descripción de la estructura del árbol de archivos

Una vez que generas tu proyecto en STM32CubeIDE, vas a ver en el panel izquierdo (vista *Project Explorer*) una serie de carpetas y archivos que conforman la estructura del proyecto. Cada uno tiene un rol específico.



Core: Contiene los archivos fuente y encabezados principales del usuario y del sistema.

- Inc: Carpeta que contiene los archivos `.h` del usuario. Incluye declaraciones de funciones, variables globales, macros y configuraciones que se utilizan en los archivos fuente. Dentro de esta carpeta está el archivo main.h.

- Main.h: Archivo de cabecera asociado a `main.c`, donde se definen prototipos, etiquetas de pines y variables globales relacionadas con la aplicación principal.

- Src: Carpeta que contiene los archivos fuente `.c` del usuario. Aquí se encuentra el código de aplicación principal.
  - Main.c: Archivo donde se escribe la lógica de la aplicación.
- Startup: Contiene el archivo de arranque en lenguaje ensamblador.

Drivers: Contiene las bibliotecas necesarias para acceder a los periféricos y al núcleo del microcontrolador.

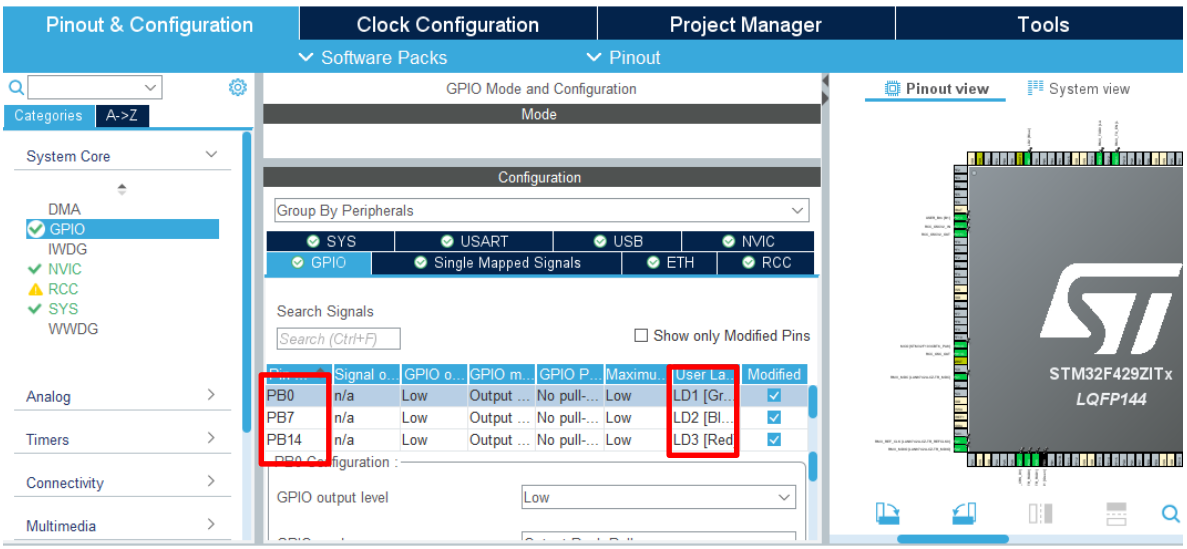
- CMSIS: es una capa de abstracción provista por ARM. Define registros del procesador, vectores de interrupción y tipos estándar. Facilita la portabilidad del código entre distintos microcontroladores ARM.
- STME32F4xx_HAL_Driver: Contiene la implementación de la HAL específica para la serie STM32F4 en este caso. Proporciona funciones de alto nivel para interactuar con periféricos como GPIO, UART, ADC, etc., sin necesidad de acceder directamente a los registros

Debug: Al compilar el proyecto, STM32CubeIDE crea esta carpeta donde coloca los archivos de salida y otros temporales.

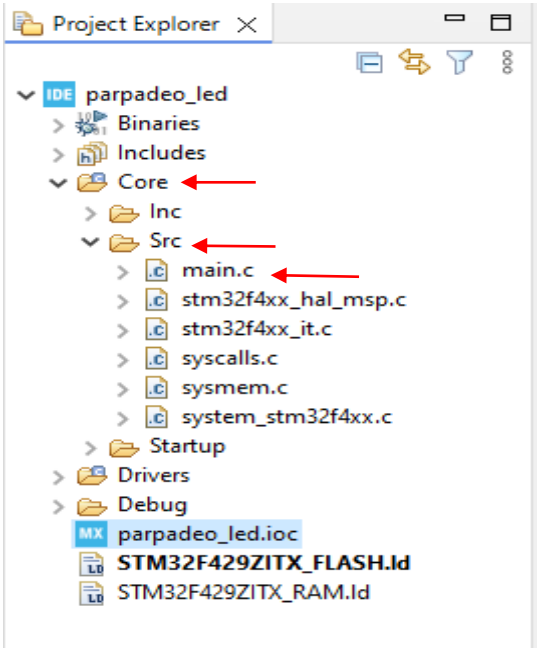
- Archivo.elf: Código objeto resultante de la compilación. Contiene el programa ejecutable junto con información de depuración. Es el archivo que se graba en la memoria del microcontrolador.
- Objects.list, archivo.map, archivo.list: son archivos auxiliares que describen el contenido del binario, símbolos utilizados, direcciones en memoria y dependencias entre módulos.
- Makefile, objects.mk, sources.mk: son archivos de construcción. Son utilizados por el sistema de compilación Make para gestionar la generación del binario. Se actualizan automáticamente.

Desarrollo de un proyecto

Para el caso de un proyecto que utilice uno de los leds integrados en la placa y lo haga parpadear cada 250 ms. El primer paso es identificar a qué pines del microcontrolador están conectados esos LEDs. En la pestaña **System Core → GPIO**, podremos ver que los LEDs verde, azul y rojo están conectados a los pines **PB0, PB7** y **PB14**, respectivamente.



Una vez verificada la configuración de los pines, vamos a programar el comportamiento del LED en el archivo **main.c**, ubicado en **Core → Src → main.c**.

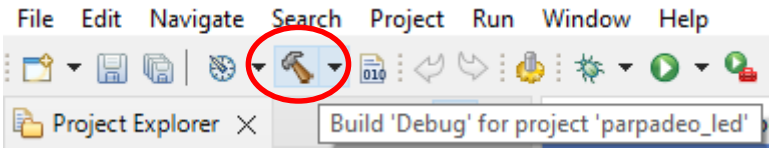


En ese archivo, dentro del bucle `while (1)`, escribiremos el código necesario para que el LED se encienda y apague con una pausa de 250 ms entre cada cambio.

```
  while (1)
  {
    /* USER CODE END WHILE */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
  HAL_Delay(250);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
  HAL_Delay(250);
    /* USER CODE BEGIN 3 */
  }
```

Compilación del proyecto

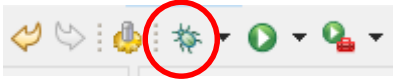
Una vez finalizada la escritura del código fuente, el proyecto debe ser **compilado** para generar un archivo ejecutable que pueda ser cargado en la placa. Hace clic en el botón **Build Project** (ícono del martillo en la barra superior).



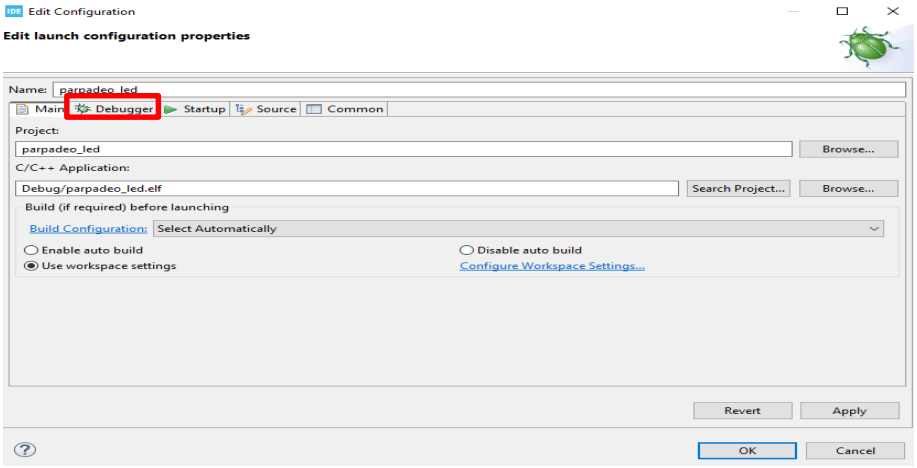
El IDE compilará automáticamente todo el código fuente, y si no hay errores, generará un mensaje de confirmación. Durante la compilación se crea un archivo ejecutable con extensión `.elf`. Este archivo incluye el código máquina listo para ejecutar en el microcontrolador e información de depuración (como nombres de funciones, variables, etc.). Se guarda automáticamente dentro de la carpeta `Debug`, la cual es generada al compilar.

Configuración del debugger

Al momento de ejecutar el programa en la placa, es necesario crear una configuración de depuración. Esto se hace haciendo click directamente en el botón debug (el icono verde con el escarabajo).



Luego se abrirá una ventana como la de la imagen.

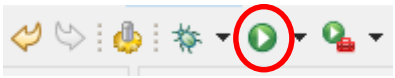


Verificá que en la pestaña Debugger diga:

- ST-Link: es el **programador y depurador oficial** de STMicroelectronics.
- Interface: SWD. es un protocolo de depuración creado por ARM.

Luego hacé clic en **Apply** y después en **Debug.**

Una vez configurado el debugger, al hacer clic en el botón **Debug**, el IDE utiliza el programador ST-Link para transferir el archivo `.elf` generado durante la compilación. Este archivo, ubicado en la carpeta `Debug`, contiene el código objeto que será almacenado en la **memoria Flash** del microcontrolador. Al iniciar la sesión de depuración, el programa se graba automáticamente y comienza a ejecutarse si se presiona el botón **Resume**.



El programa se ejecutará directamente sobre la placa.