Instalación de STM32CubeIDE

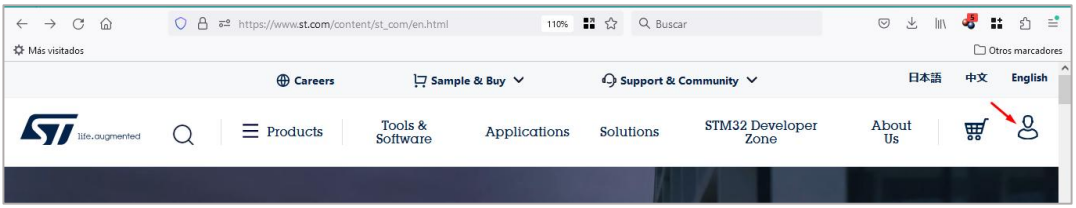
STM32CubeIDE es un IDE de la empresa ST para sus microcontroladores ARM de 32 bits, que integra el IDE Eclipse, el configurador gráfico *STMCubeMX* y diversas herramientas propias y de terceros, como depuradores, grabadores, *middlewares*, herramientas de test etc. La gran potencia de este entorno reside en el amplio desarrollo de drivers y ejemplos para toda su gama de microcontroladores, desde los más simples *Cortex* M0 hasta los de doble núcleo con Linux embebido, y en la posibilidad de configurar y habilitar sus periféricos de una forma gráfica, para luego utilizarlos a través de funciones específicas.

Los pasos para poder utilizar esta herramienta son los siguientes: *(La mayoría están descriptos en el video https://www.youtube.com/watch?v=bQIvecq3m7Q)*

1. Registrarse

Nota: En el momento de la descarga se le solicitará estar registrado con email y contraseña. El registro es gratuito. Puede hacerlo directamente con el enlace del paso 2 al momento de intentar la descarga.

Ir a https://www.st.com, y seleccionar el icono de la derecha para realizar el registro.



2. Descargar STM32CubeIDE

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

A mitad de página, en Get Software, elegir el sistema operativo en el que se instalará (para Windows es la última opción), y la versión.

Para avanzar con la descarga se le solicita acceder a términos y condiciones, y también estar registrado.

3. Instalar STM32CubeIDE

De la forma usual de cualquier software. Puede escoger carpeta de instalación. Aceptar todos los drivers que quiera instalar (ST-Link-, Segger j-link etc).

4. Descargar las bibliotecas de funciones

Para cada una de las familias de microcontroladores, (STM32F0xx, STM32G0xx, STM32F1xx, STM32F3xx, STM32F4xx, STM32F7xx, STM32Hxx etc.) ST ha desarrollado un conjunto de funciones denominada HAL (biblioteca de abstracción de hardware) que permite manejar sus periféricos de una forma relativamente simple y consistente. Cuando se quiere hacer uso de la herramienta de configuración gráfica STM32CubeMX integrada en el IDE, es necesario descargar el conjunto de funciones de la familia. Puede dejarse que el IDE lo realice automáticamente, o bajar e instalar manualmente la biblioteca que se necesita. **Como en clase no disponemos de ancho de banda suficiente para que todos bajen estas bibliotecas al mismo tiempo, vamos a realizar la instalación manual de las mismas**.

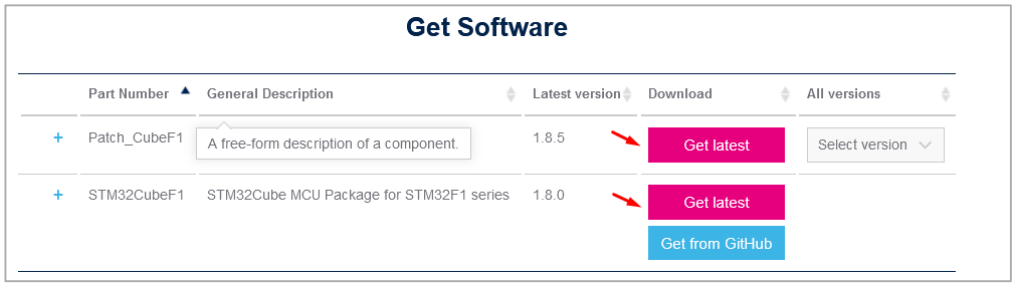
En nuestras clases utilizaremos la placa STM32F407 *Discovery*, desarrollada por la propia empresa ST, que utiliza un STM32F407VGT6 (Cortex M4 hasta 168MHz) y que denominaremos ***la discovery.***

https://www.st.com/en/evaluation-tools/stm32f4discovery.html

Y la placa "*bluepill*", que es open hardware, muy accesible en tiendas de electrónica del país, que utiliza un microcontrolador stm32F103C8T6 (Cortex M3 hasta 72MHz), y que denominaremos ***la bluepill***.

Para la discovery ir al enlace:
https://www.st.com/en/embedded-software/stm32cubef4.html
Y bajar los dos paquetes

Para la bluepill, ir a
https://www.st.com/en/embedded-software/stm32cubef1.html
Y bajar los dos paquetes.

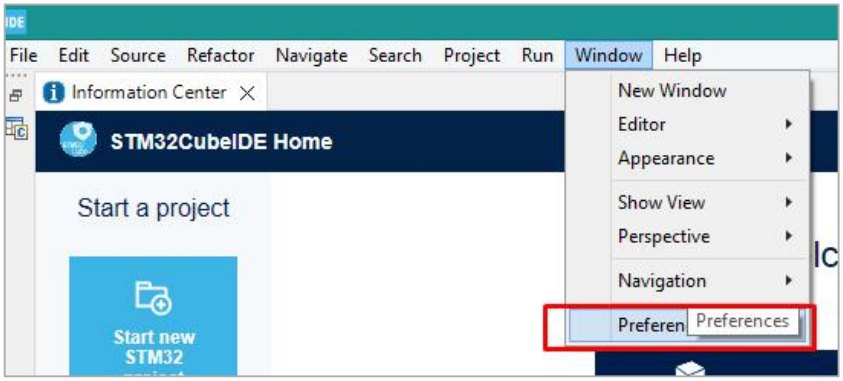


Todos estos paquetes se deberán descomprimir en una carpeta que luego deberá ser indicada dentro del STM32CubeIDE.
Por ejemplo, en una carpeta ***"C:\ST\repositorio"***
Cada paquete descomprimido genera su propia carpeta y un árbol de directorios en los que estarán los *drivers*, *middleware* de terceros y proyectos de ejemplo para distintas placas de la familia.

5. Configurar STM32CubeIDE para encontrar las bibliotecas de funciones

Iniciar el STMCube32IDE, escoger una carpeta de "*workspace*" cualquiera (sitio en el que se guardarán los proyectos).

Ir a "*Window\Preferences\STM32cube\Firmware Updater*" y escoger la carpeta donde están los paquetes descomprimidos (en nuestro ejemplo ***"C:\ST\repositorio"***).



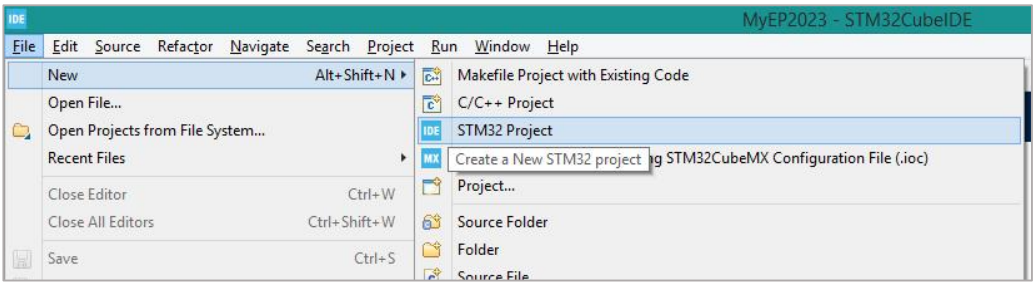
Para evitar que el IDE intente conectarse automáticamente al servidor buscando actualizaciones puede tildar la casilla *Off Line Mode*.

Con esto ya está listo el entorno para comenzar a desarrollar proyectos.

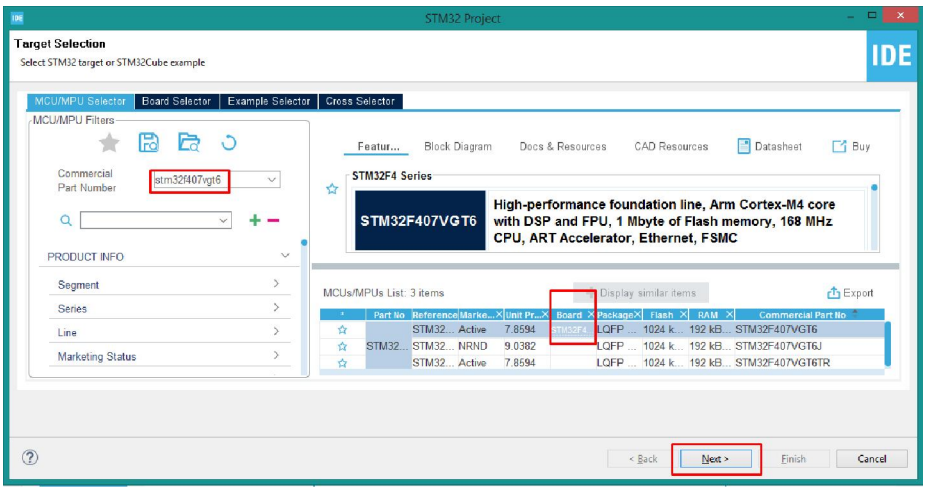
Creación de proyecto en STM32CubeIDE

6. Crear un proyecto auxiliado con la herramienta STM32CubeMX

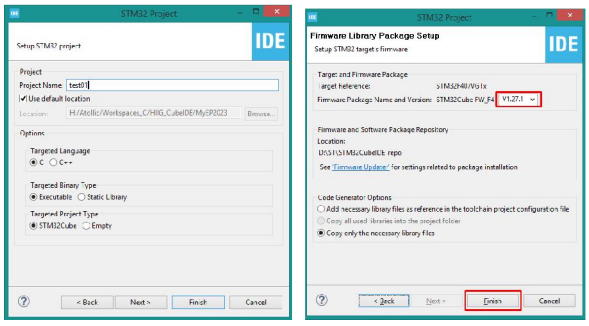
Para esto ir a *File\New\STM32 Project*



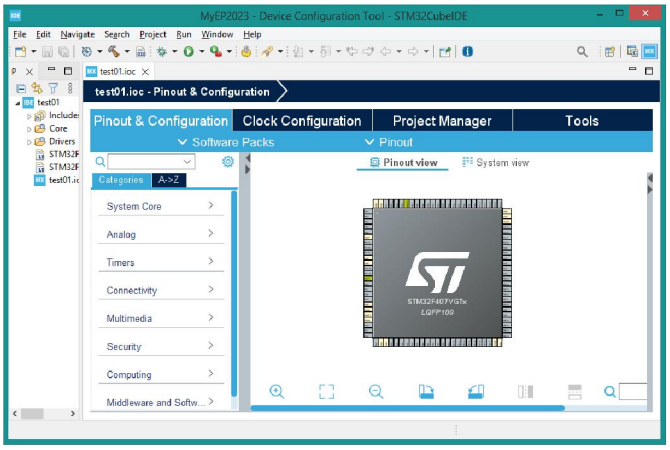
Aparecerá un formulario con varias pestañas.
El IDE tiene la opción de configurar microcontroladores solos o las placas oficiales de ST, que pueden traer algunos periféricos. La primera pestaña (MCU/MPU Selector) es para microcontroladores sueltos, la segunda (*Board Selector*) es para las placas oficiales.
Escribiendo en el campo de arriba a la izquierda el nombre del microcontrolador de desplegará una lista (de forma similar a la de Microchip Studio). También aquí puede verse en la columna *Board* las placas que llevan el microcontrolador seleccionado.



Una vez seleccionado el micro o la placa oprimir *Next*, escribir nombre de proyecto, *Next* nuevamente, luego escoger (si corresponde) el paquete de firmware a utilizar, y *Finish*.



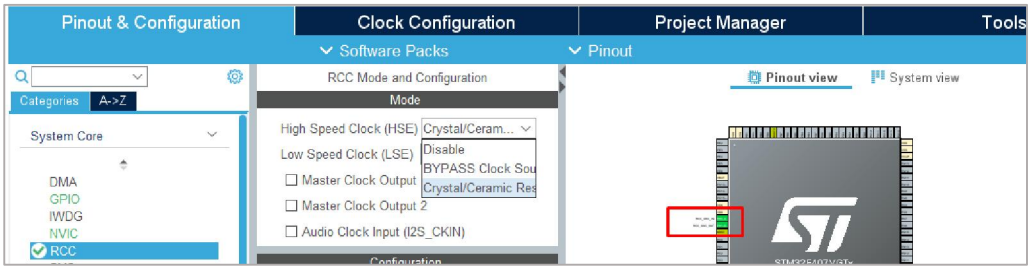
Con esto se llega a la herramienta de configuración gráfica *STMCubeMX* integrada en el IDE.



7. Configurar sistema y periféricos

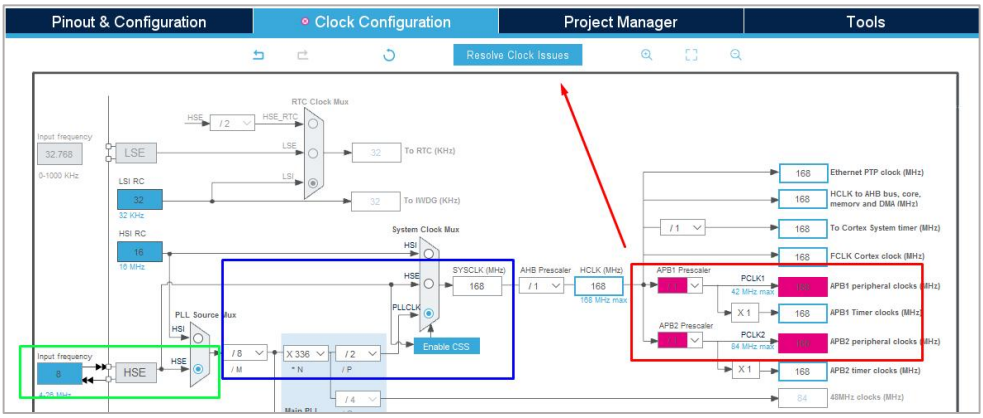
a. Configurar *Clock*

En la primera pestaña (*Pinout* & *Configuration*) se encuentran a la izquierda todos los periféricos, agrupados por categoría o alfabéticamente.



En la categoría *System Core**RCC*, escoger el clock de sistema. Si se utilizará clock interno dejar en estado *Disable*. Tanto la *discovery* como la *bluepill* tienen un cristal de 8MHz como reloj principal HSE (*high speed external*). Si se lo quiere utilizar debe seleccionarse la opción *Crystal/Ceramic Resonator*. Esta opción utiliza dos pines (*RCC_OSC_IN* y *RCC_OSC_OUT*).

En la segunda pestaña se configura la cadena de clocks para la CPU (SYSCLK) y los diversos periféricos. A la izquierda se selecciona *Input frequency*, y en el PLL *Source Mux* si se utiliza HSI o HSE. A continuación, hay una cadena de divisores (M, P, Q …) y multiplicadores de frecuencia (N), con los que se puede obtener casi cualquier frecuencia dentro de los rangos admisibles de la CPU y periféricos. En caso de no encontrar los valores correctos, se puede utilizar el botón *Resolve Clock Issues*. Por ejemplo, escribiendo en el casillero HCLK (MHz) la frecuencia deseada el *clock solver* puede encontrar los valores de M, P, Q, N etc.



b. Configurar pines del *debugger*

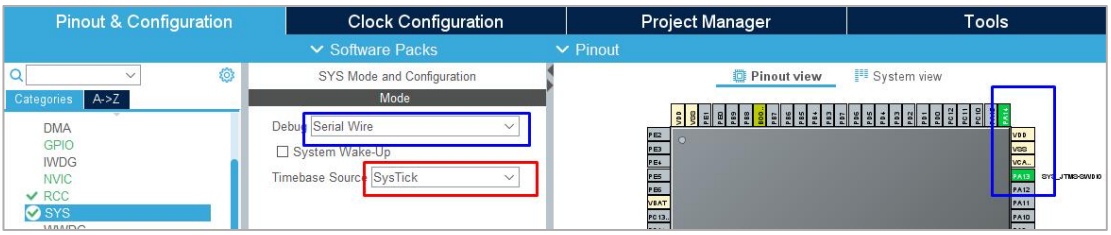
Nota: En las placas que hemos utilizado hasta ahora (*Arduino Nano*, *Arduino Uno*) hemos cargado los programas en los micros directamente a través del puerto serie, haciendo uso del propio chip USB/serie que viene en la placa y del programa de *bootloader* que fue grabado en el micro de la placa en el momento de la fabricación. Este bootloader se comunica con el *AVRDude*, lee las tramas de bytes a grabar y las va escribiendo en la zona de FLASH de programa. En el Atmega328P el *bootloader* se ubica en la región alta de la memoria FLASH (*boot* Flash *section*, 0x7000 a 0x7FFF), dejando la región de 0x0000 a 0x6FFF para programa de usuario.

En los micros que vamos a utilizar ahora también se los puede grabar por puerto serie u otro puerto de comunicaciones (SPI, I2C, CAN etc) haciendo uso de un programa de *bootloader* que ya viene grabado de fábrica en estos micros (en la nota de aplicación AN2606 de ST está extensamente tratado para las distintas gamas de microcontroladores).

En aplicaciones finales el *bootloader* permite realizar actualizaciones del firmware sin necesidad de conectar un programador. Pero durante las etapas de desarrollo es muy recomendable utilizar un programador/depurador (*programmer/debugger*), que va a permitir seguir el programa paso a paso y ver el estado de las variables y registros del microcontrolador para corregir errores.

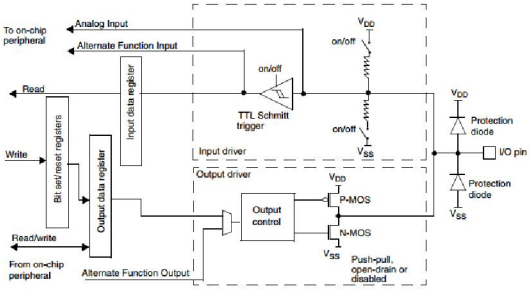
En la *discovery* viene incluido el programador/depurador *ST-Link*, por lo que vamos a utilizarlo. Para la *bluepill* es necesario utilizar un *ST-Link* externo.

Volviendo a la pestaña *Pinout & Configuration*, en la categoría *System Core**SYS*, seleccionar el *debugger Serial Wire*, y como base de tiempo el *timer SysTick*. El *Serial Wire* es necesario para poder grabar y depurar el microcontrolador a través del *ST-Link*.



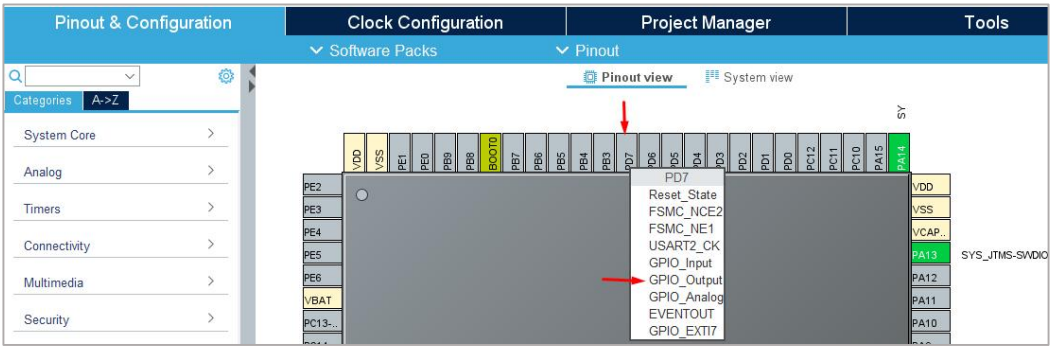
c. Configurar un pin GPIO como salida

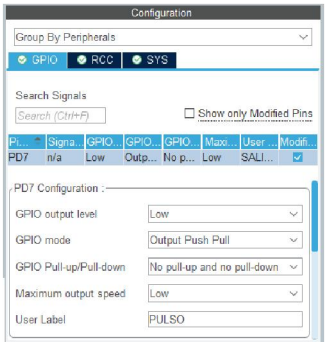
Vemos un ejemplo de configuración de un periférico básico. Comenzamos con un pin como GPIO.



Tanto en la *bluepill* como en la *dicovery* los pines como GPIO tienen diversas opciones de configuración de entrada y de salida. Se elige la direccionalidad (entrada/salida), tipo de salida (*push-pull/open drain*), resistencias (*pull-up/pull-down*), velocidad (control del *slew-rate* para minimizar EMI y oscilaciones de la salida en las conmutaciones).

Este periférico se configura directamente tocando el pin que se va a utilizar en la gráfica del chip.



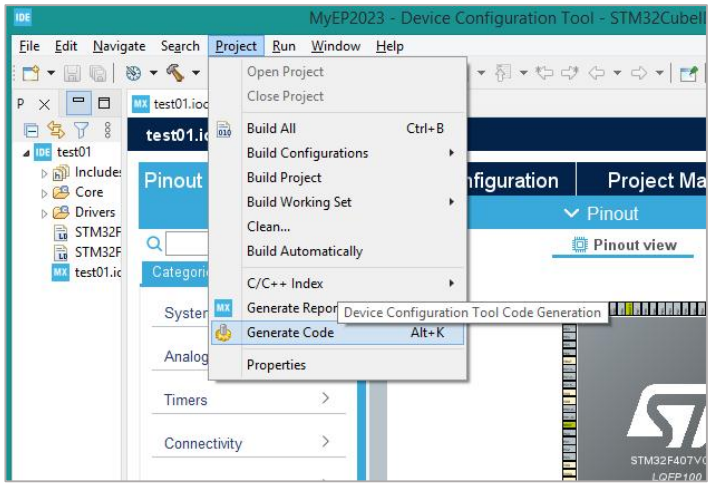


Al seleccionar GPIO Output, aparecen las opciones de configuración, para elegir estado inicial (*GPIO Output Level*), Modo (*Push-Pull*, *Open Drain*), velocidad, e incluso ponerle un rótulo. Este rótulo servirá luego para usar las funciones de escritura del pin sin especificar puerto o número de pin, haciendo el código más claro y portable.

En el ejemplo se utiliza el pin PD7, configurado como *Push-Pull*, a bajo *slew-rate* (para conmutaciones de hasta 10Mhz), con el nombre PULSO.

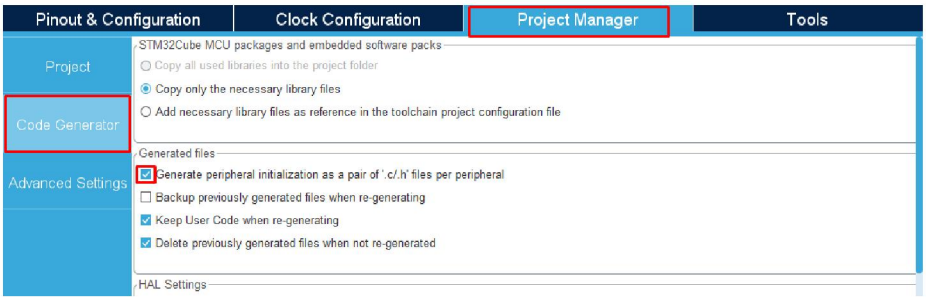
8. Generar código a partir de la configuración

Una vez que se han habilitado y configurado todos los periféricos y subsistemas del microcontrolador, ir a *Project**Generate Code*

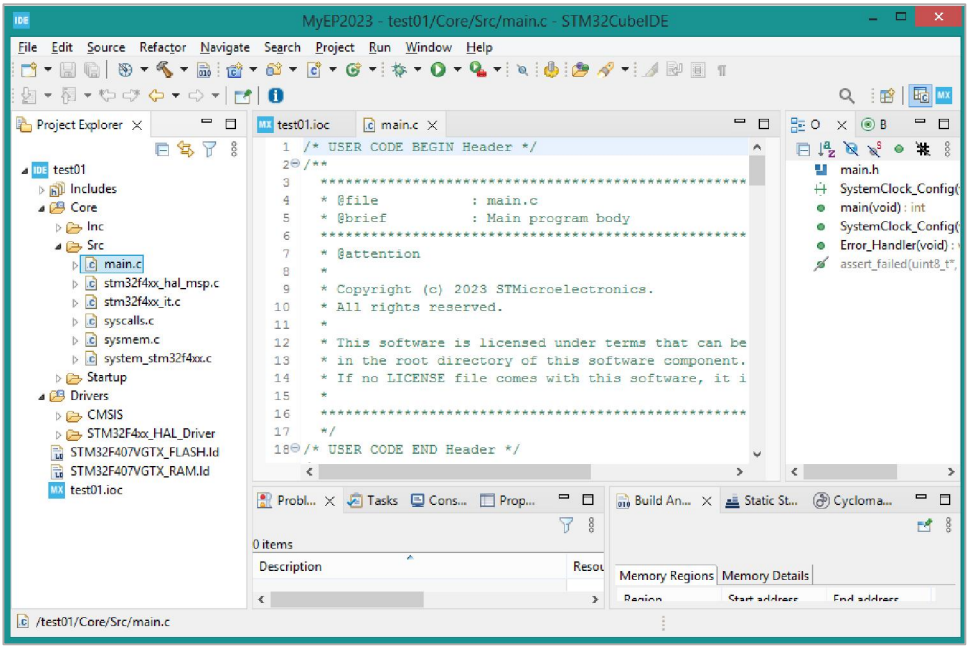


Esto generará un árbol de archivos con los *drivers* de los periféricos y la configuración inicial de la placa

Nota: Es recomendable que los drivers de periféricos estén en archivos separados. Para esto dentro del configurador gráfico ir a la pestaña *Project Manager*, *Code Generator* y tildar *Generate peripheral inictialization as pair of 'c/h' files per peripheral*.



. El archivo principal en el que el usuario puede escribir su código está localizado en la carpeta ***Core\Src\main.c***.



Este *main.c* tiene forma de plantilla, con algunas funciones declaradas e implementadas (las que corresponden a los subsistemas activados), y **zonas de usuario**, encerradas entre comentarios tipo "*USER CODE BEGIN n*"

```
/* USER CODE BEGIN 1 */
```

```
/* USER CODE END  1/
```

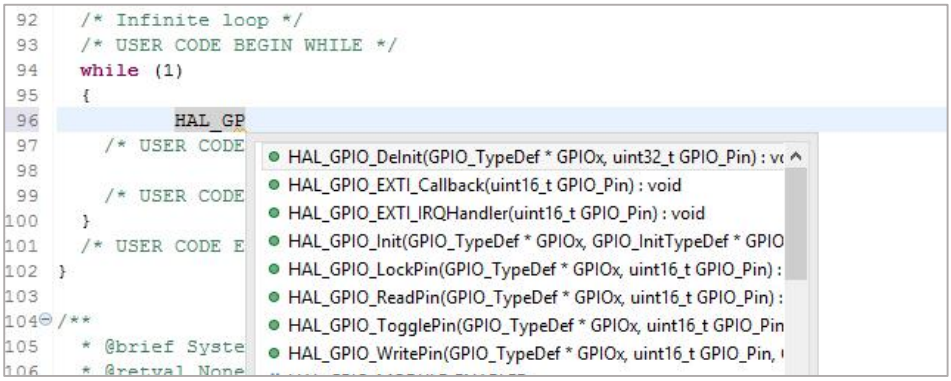
Todo lo que se escribe entre esos comentarios será preservado en caso de volver a generar código desde el asistente gráfico. Lo que esté fuera de los comentarios será reemplazado por el nuevo código generado.

9. Comenzar a escribir el código propio y utilizar la HAL

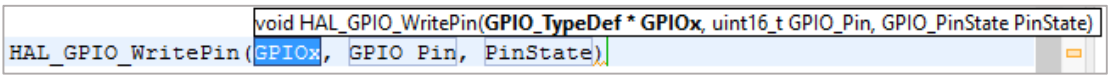
La mayoría de las funciones de la HAL tienen el formato:

HAL_*nombreperiferico_acción*(*handler periférico*, parámetros…)

Por ejemplo, para los GPIO es HAL_GPIO… etc. Se puede entonces invocar el asistente de autocompletado comenzando a escribir "HAL_GP" y oprimir la combinación de teclas CTRL+Espacio.



Por ejemplo, para escribir un '1' o un 0 en el pin 7 del puerto D, se utiliza la acción WritePin




```
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_7, SET);
```

En vez de un '1' se utiliza *SET*. La biblioteca HAL tiene definido un conjunto de tipos y estructuras de datos que permiten realizar un control de parámetros consistente.

Una rutina completa de *Blink* será, por ejemplo:

```
/* USER CODE BEGIN WHILE */
while (1)
{
      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_7, SET);
      HAL_Delay(100);
      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_7, RESET);
      HAL_Delay(100);
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Observe que se ha escrito entre los comentarios *USER CODE BEGIN WHILE* y *USER CODE END WHILE*. Si se hubiera escrito entre *USER CODE END WHILE* y *USER CODE BEGIN 3* se perdería en caso de volver a utilizar la operación *Project\Generate Code*. (si le ocurre por accidente, siempre puede utilizar Ctrl+Z).

La misma rutina de *Blink*, pero utilizando el rótulo del pin:

```
while (1)
{
      HAL_GPIO_WritePin(PULSO_GPIO_Port, PULSO_Pin, SET);
      HAL_Delay(100);
      HAL_GPIO_WritePin(PULSO_GPIO_Port, PULSO_Pin, RESET);
      HAL_Delay(100);
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
}
```

La definición de `PULSO_GPIO_Port` y `PULSO_Pin` las hizo automáticamente el STM32CubeMX, y están en el archivo main.h. Para no tener que recordar la notación, puede utilizarse el asistente de autocompletado, por ejemplo escribiendo *PUL* y apretando Ctrl+Espacio.

Los pasos siguientes los ensayaremos en clase.

Para una introducción a los ARM y primeros pasos en STM32CubeIDE pueden ver el video preparado por la cátedra:

https://www.youtube.com/watch?v=bQIvecq3m7Q

Si quiere ir adentrándose en el uso de este entorno recomendamos la página de Controllers Tech:

https://controllerstech.com/stm32-hal/

Tiene una buena cantidad de ejemplos con STM32CubeIDE, incluyendo el código y videos paso a paso.