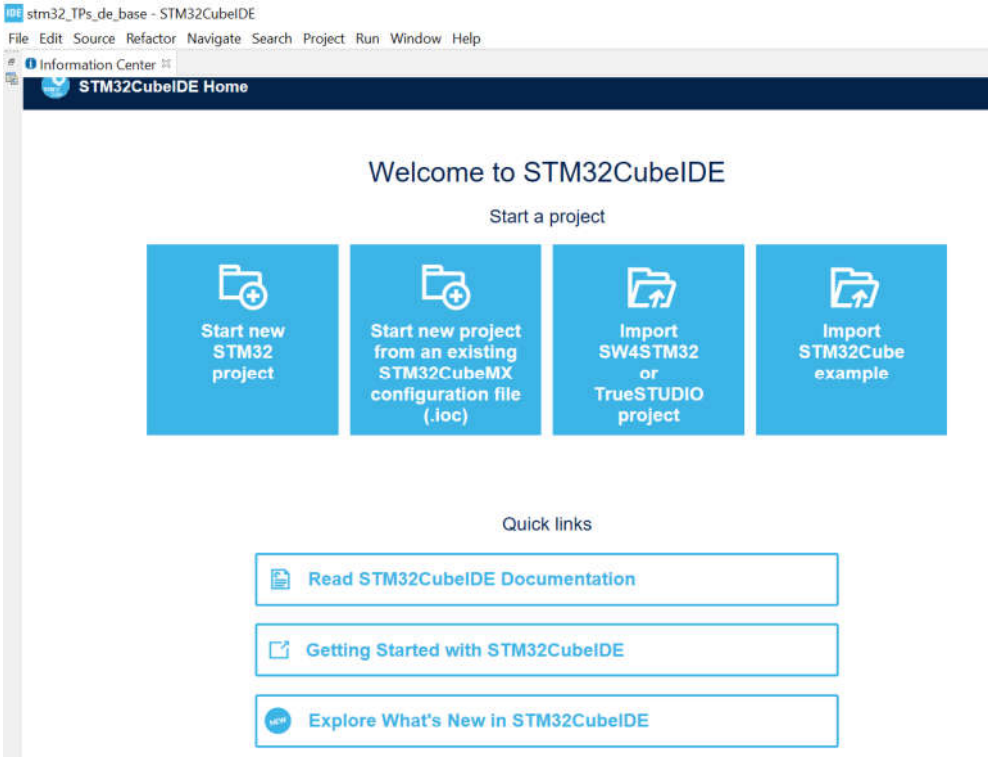
- **5 - Exemples de projets simples utilisant l'EDI « CubeIDE » :**
  - **5.1** – Premier projet sous STM32CubeIDE : ***« LED »***

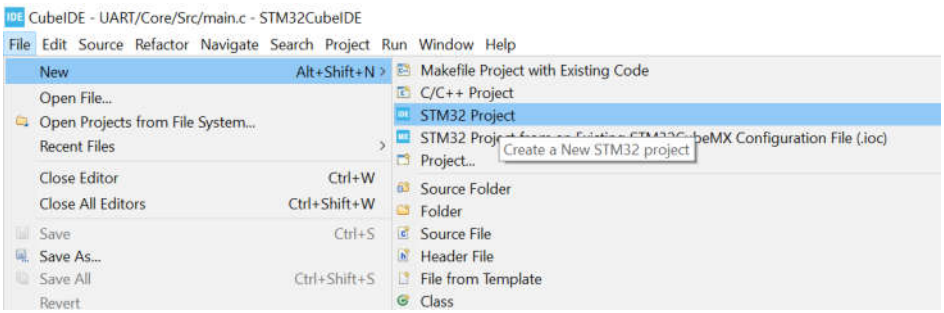
➢ **Lancement de STM32CubeIDE**



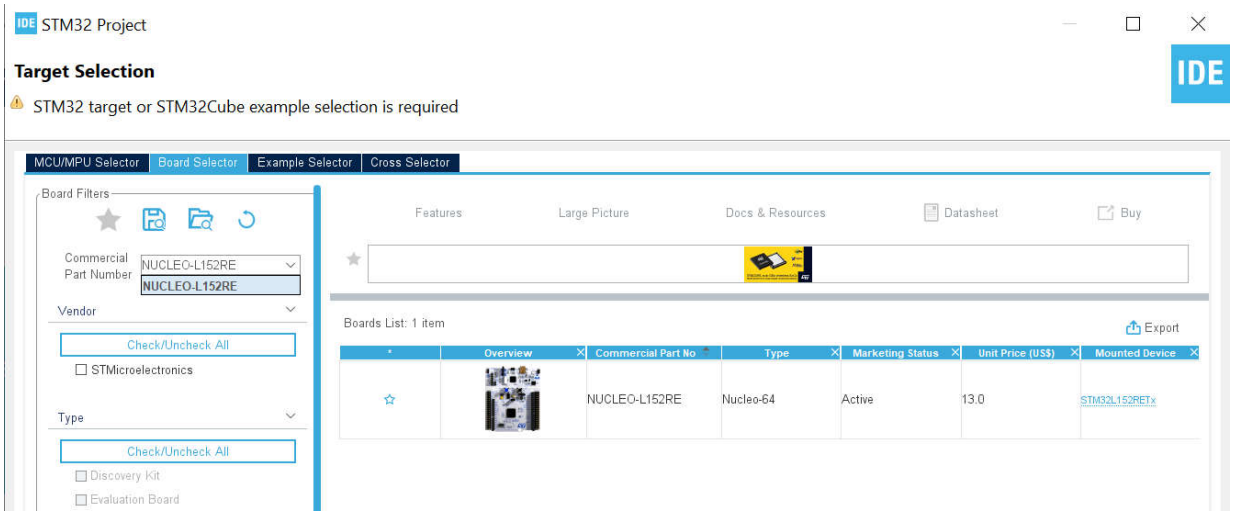
➢ **Création d'un nouveau projet sur STM32CubeIDE**

Soit cliquez sur Start new STM32 project

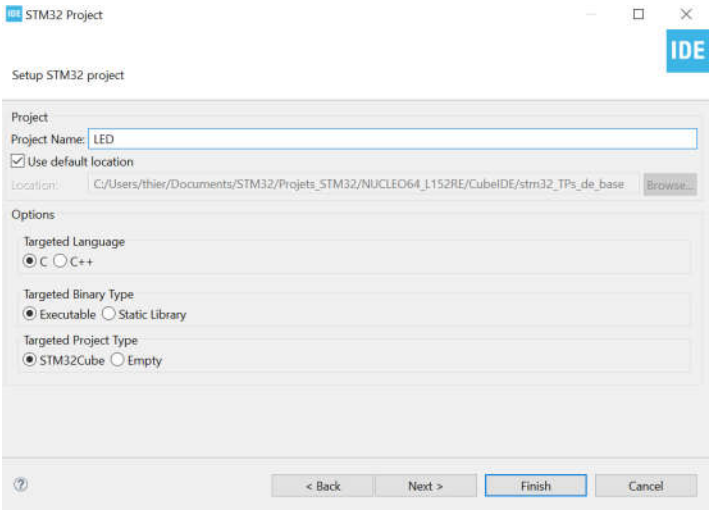
Soit File → New → STM32 Project



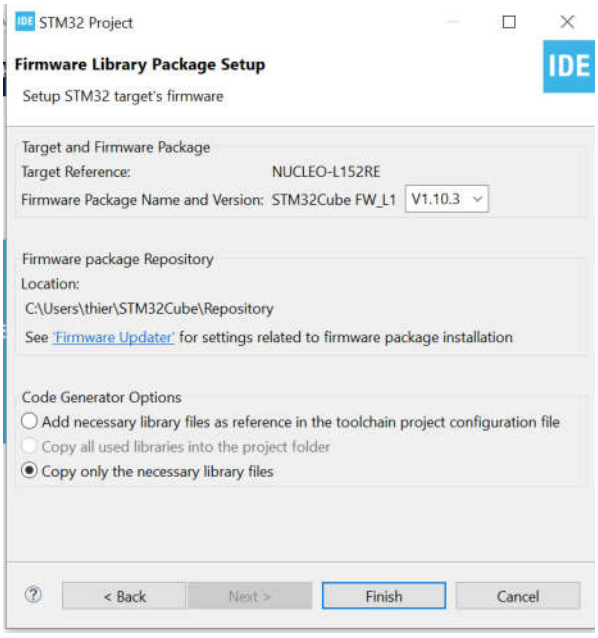
➢ **Choix de la carte NUCLEO utilisée (NUCLEO-L152RE)**



Cliquer sur Next et donner un nom au projet

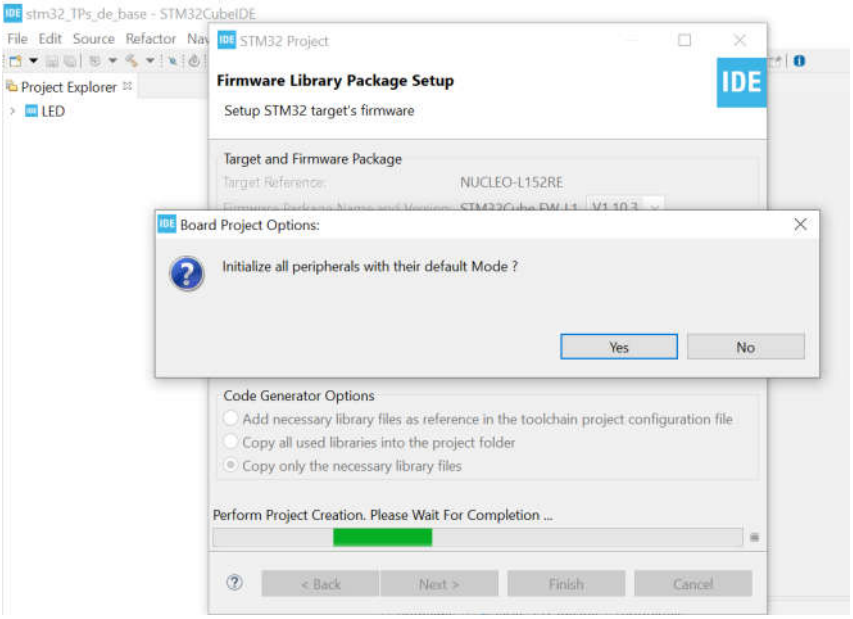


Cliquer sur Next puis sur Finish

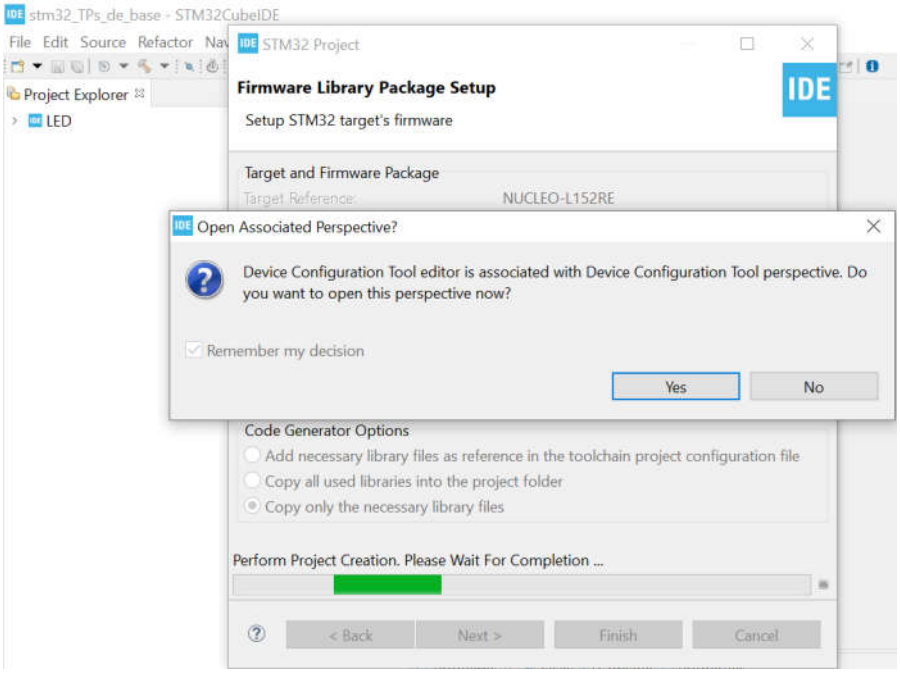


Cliquer sur Yes à la question

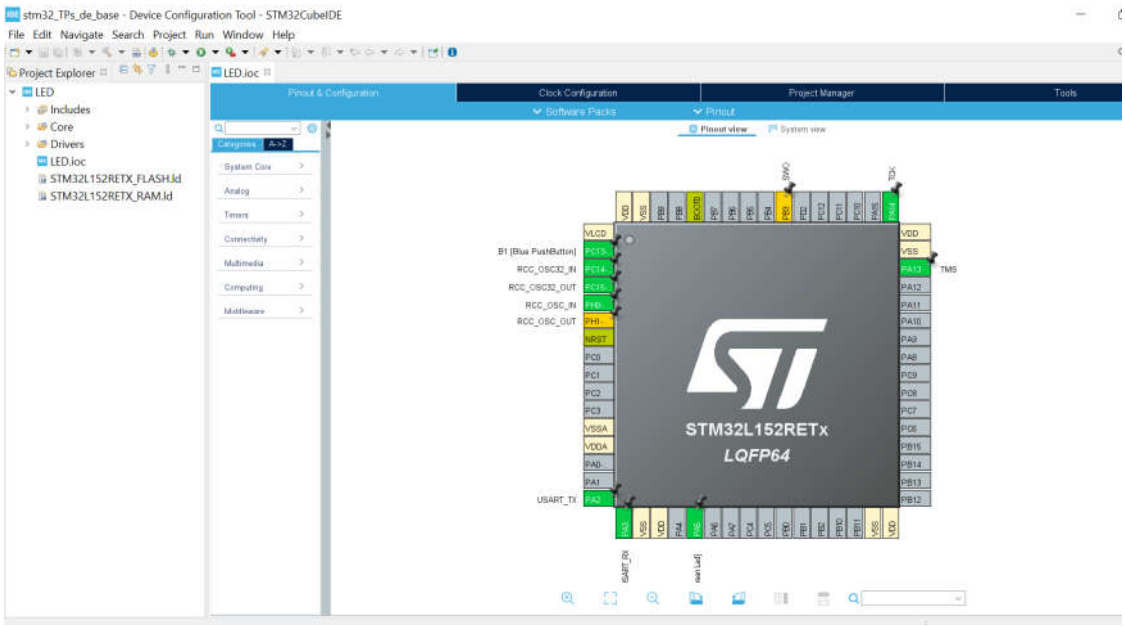
« **Initialize all peripherals with their default Mode ?** »



Cliquer sur Yes

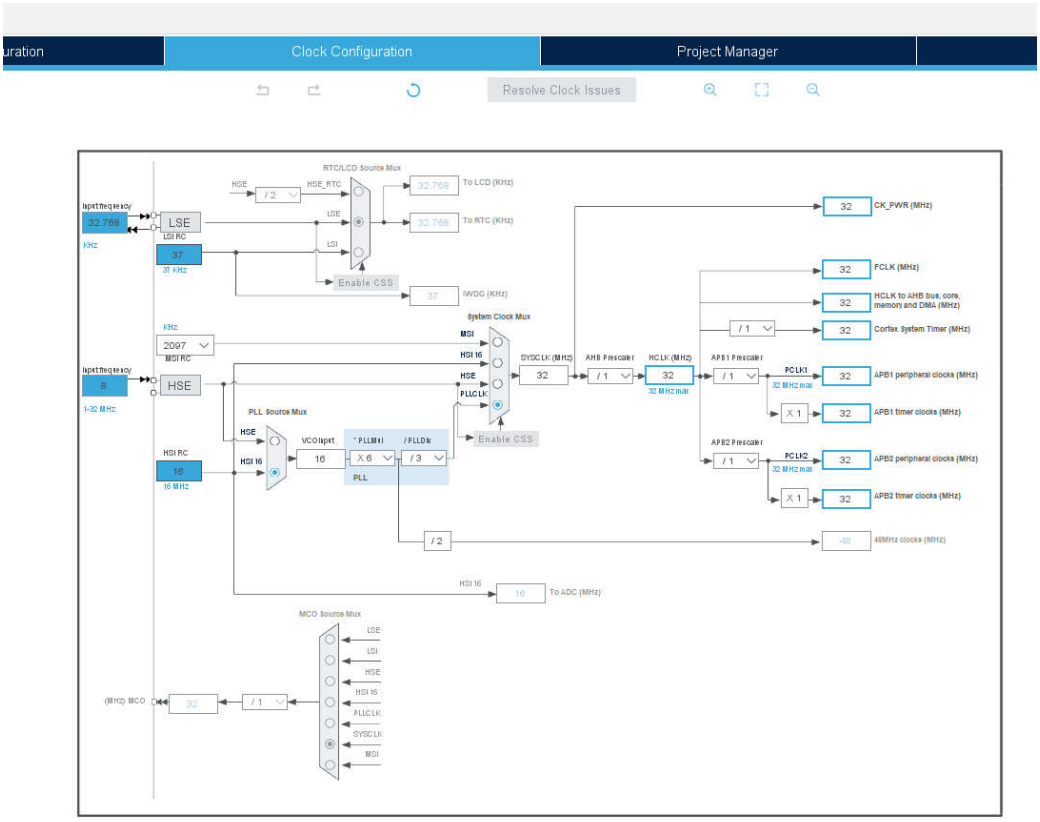


- **Création du projet LED sur STM32CubeIDE et ouverture du fichier LED.ioc (fichier STM32CubeMX)**



Le fait d'avoir coché la case **« Initialize all IP with their default mode »** conduit STM32CubeMX à initialiser toutes les ressources disponibles sur la carte (broches colorées en vert), dont celle qui va nous intéresser dans ce chapitre, LD2, qui est une LED de couleur verte connectée au GPIO du port PA5. (Et plus tard le bouton poussoir user PC13)

Hypothèse : l'onglet Clock Configuration est par défaut correctement configuré pour la carte sélectionnée.

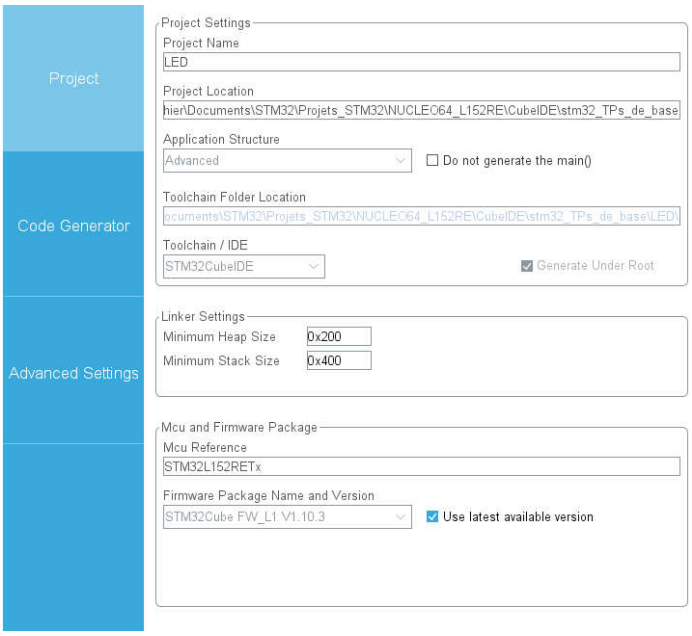


Sur l'onglet « Project Manager »

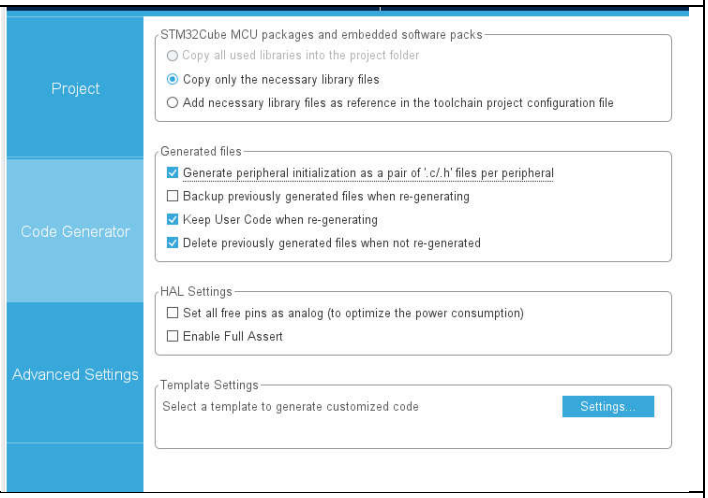
On renseignera les paramètres suivants au niveau de l'onglet **Project** :

■ le nom du projet dans **Project Name** : « LED » ;
■ l'emplacement **Project Location** de ce projet sur le disque de l'ordinateur.
■ l'EDI choisi pour générer le code C d'initialisation dans **Toolchain / IDE**. **STM32CubeIDE est automatiquement choisi**
Si CubeMX vous auriez le choix de l'IDE.

■ *Si vous êtes sur CubeMX Cochez **Use Default Firmware Location**. Ainsi, STM32CubeMX ira directement rechercher le package STM32CubeL1 dans le répertoire dans lequel il a été installé.*



Au niveau de l'onglet **Code Generator**, laissez les paramètres par défaut et cochez en plus
« Generate peripheral initialization as a pair of '.c/.h' files per IP », ce qui fournit, lors de la génération du code C d'initialisation, deux fichiers XX.c et XX.h pour chaque périphérique et non une initialisation regroupée dans le main.c, qui deviendrait rapidement illisible.

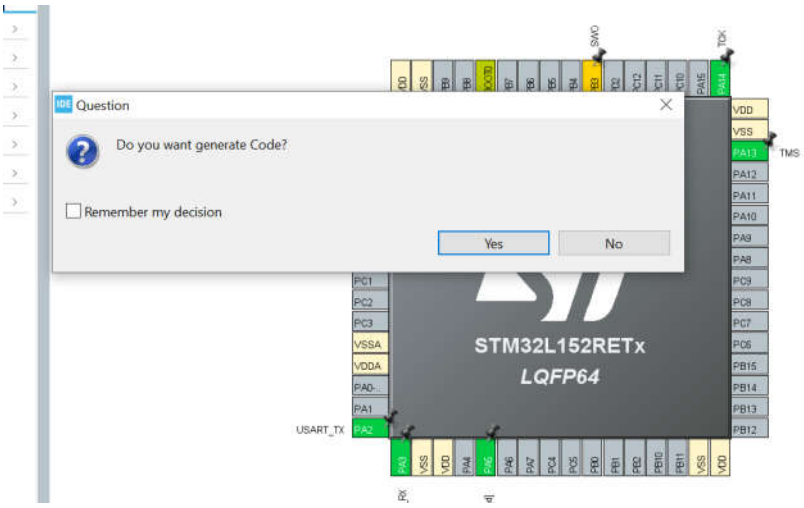


Le troisième onglet, **Advanced Settings**, ne doit pas être modifié car il correspond à des fonctions avancées que nous n'utiliserons pas dans ce projet.

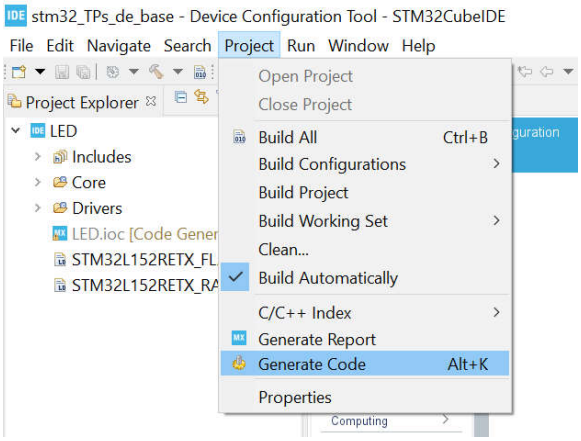
➢ Génération du code C d'initialisation

Merci de sauver le projet « LED »

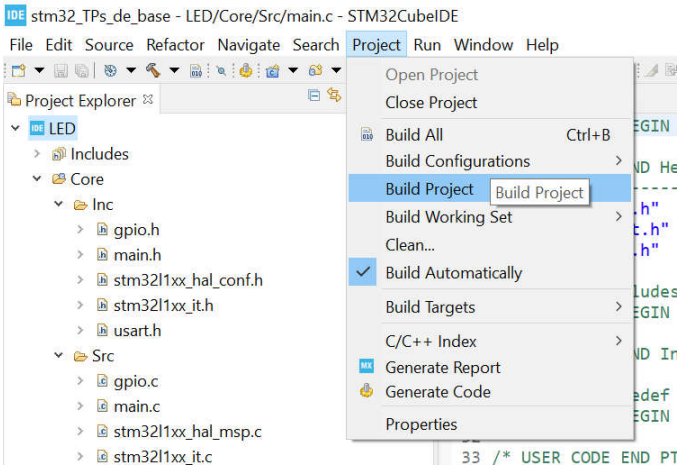
Une fenêtre s'ouvre pour savoir si vous voulez générer le code C « cliquer sur yes »



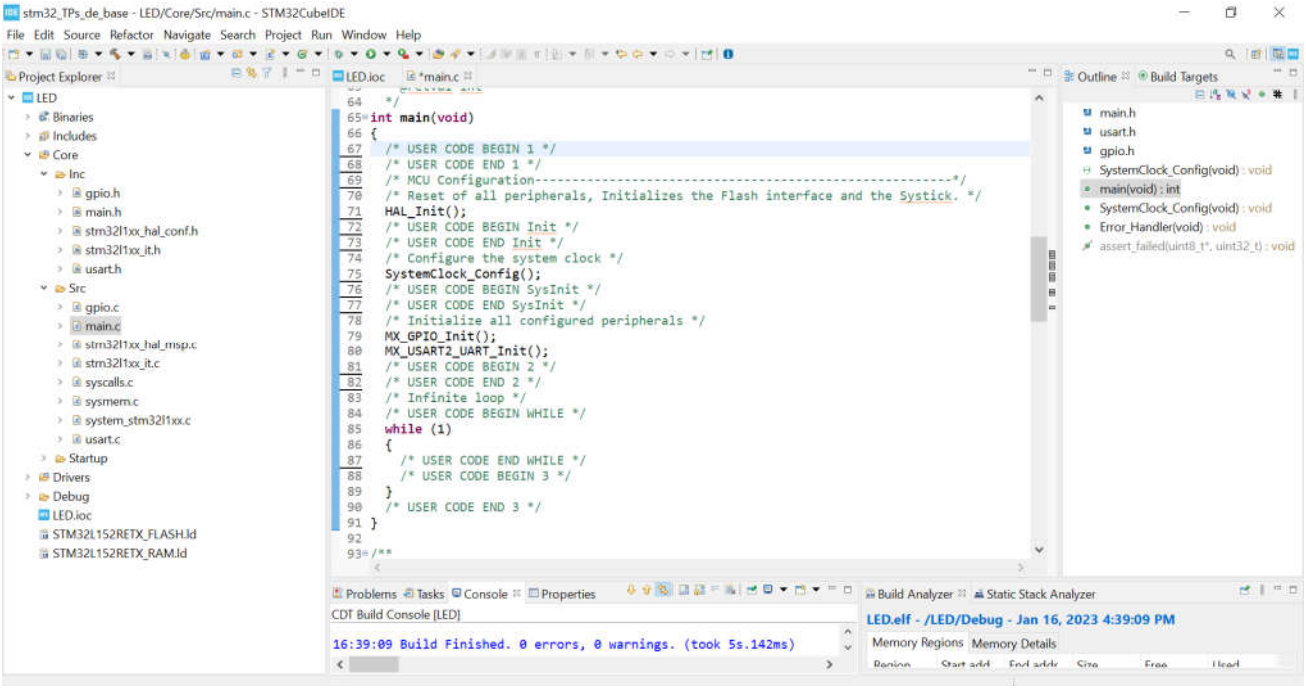
Ou alors aller dans l'onglet « Projet » et « Generate Code »



➢ **Compilation du projet « LED » : Onglet « Project » → « Build Project »**



L'arborescence ***LED/Core/Src*** de l'explorateur de projet liste les fichiers d'initialisation des différents périphériques de la carte, notamment ***gpio.c***, qui correspond au bouton poussoir utilisateur B1 et à la LED LD2 qui nous intéressent dans ce chapitre, ou encore le fichier ***main.c***, point d'entrée du programme que nous allons modifier pour faire clignoter LD2.
Lors de la génération du code C d'initialisation, STM32CubeIDE (ou STM32CubeMX) a créé le fichier ***main.c*** dans lequel se trouve la fonction principale du programme ***int main(void)***.



À l'intérieur de ce fichier ***main.c***, certaines autres fonctions indispensables sont appelées :

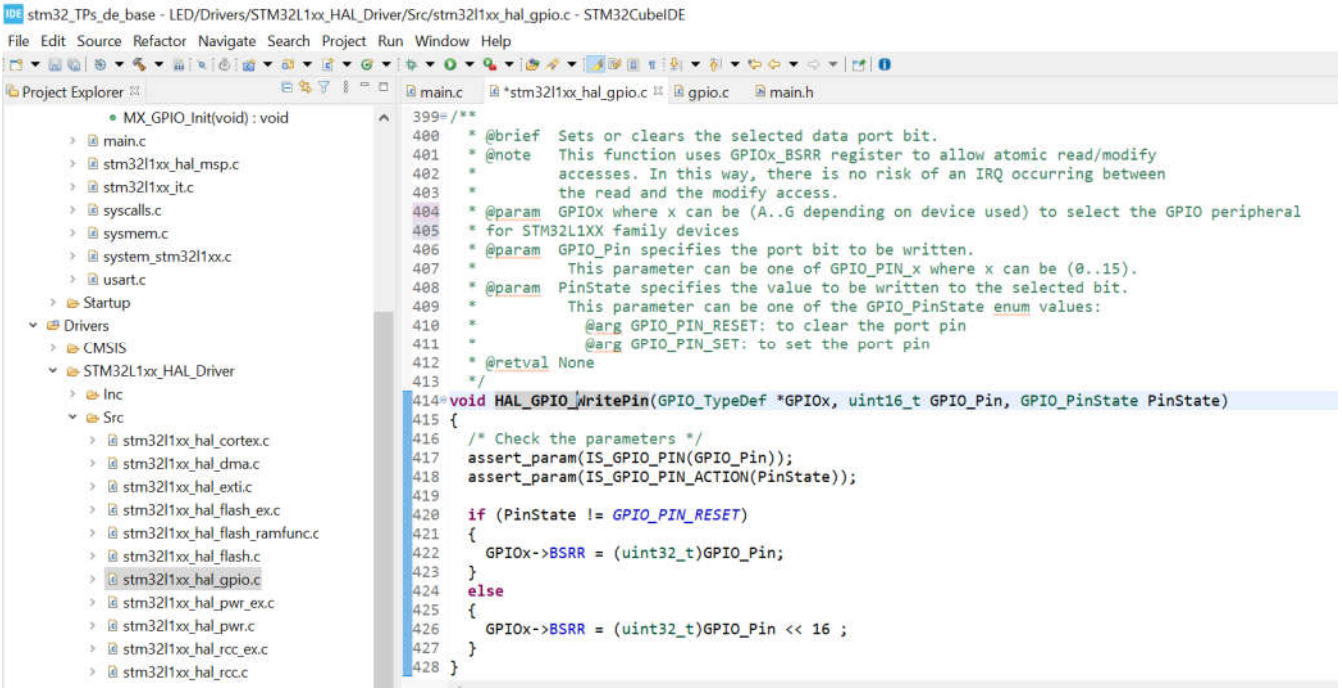
- ■ **HAL_Init()** initialise tout le matériel de la carte Nucleo-L152RE : mémoire, contrôleur d'interruption, base de temps du système et autres fonctions de bas niveau.
- ■ **SystemClock_Config()** initialise toutes les horloges du STM32L152RET6 conformément à ce qui a été établi dans l'onglet ***Clock Configuration*** de STM32CubeMX.
- ■ **MX_GPIO_Init()** initialise les GPIO : bouton-poussoir utilisateur B1 et LED LD2.
- ■ **MX_USART2_UART_Init()** initialise la communication vers le monde extérieur.

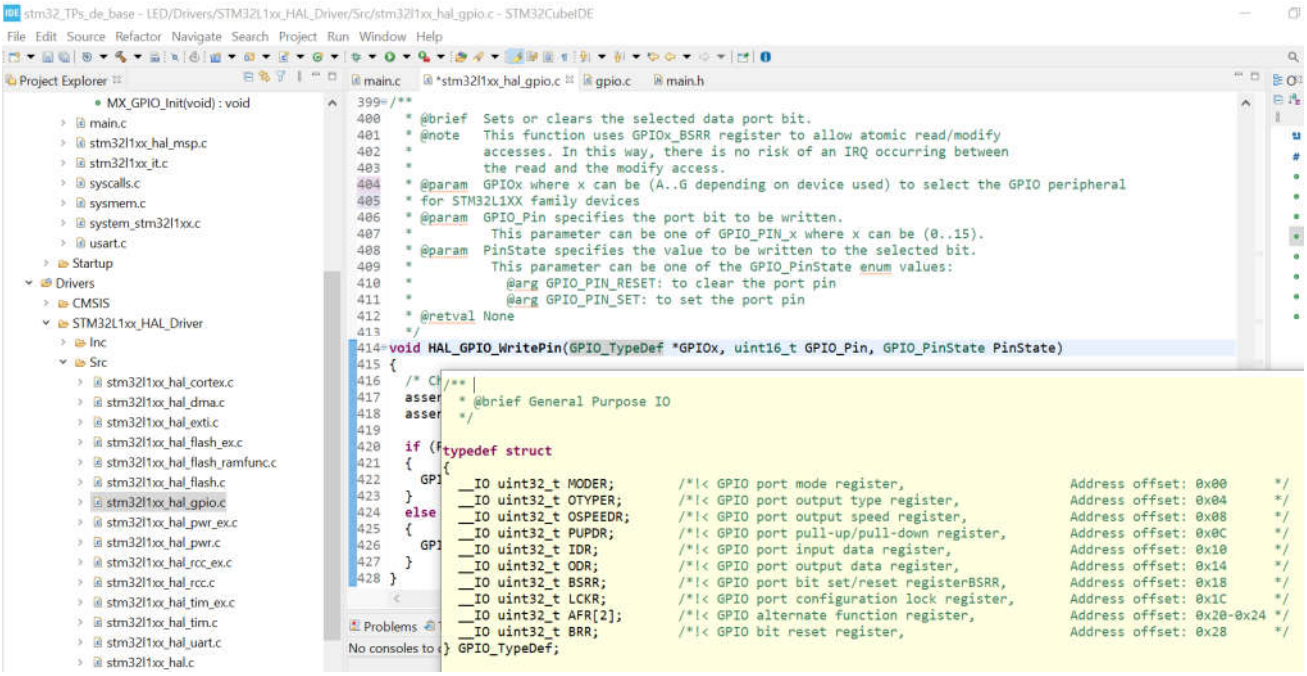
Tout ce qui se trouve entre /* et */ est un commentaire en langage C et n'est donc pas exécuté. Certains types de commentaires sont tout de même particuliers :
/* USER CODE BEGIN */
…
/* USER CODE END */ avec éventuellement des nombres ou des noms après BEGIN et END.

Ce type de commentaire a une signification particulière pour STM32CubeMX, qui les interprète comme des tags (étiquettes ou balises). Il est conseillé de ne pas les supprimer, mais plutôt d'insérer notre code à l'intérieur de ces balises car, si nous devons générer de nouveau le code C d'initialisation avec STM32CubeMX après l'ajout d'un périphérique par exemple, le code placé entre ***BEGIN*** et ***END*** sera conservé. En revanche, si notre code a été ajouté à l'extérieur de ces balises, il sera tout simplement supprimé.
La fonction main générée contient l'instruction ***while (1)*** suivie d'une accolade ouvrante et d'une accolade fermante. C'est à l'intérieur de ces deux accolades, et si possible à l'intérieur de balises ***BEGIN*** et ***END***, que nous allons insérer notre code pour faire clignoter LD2.

/* USER CODE BEGIN 3*/
Le code à insérer entre les balises est représenté sur la figure ci-contre :
/* USER CODE END 3*/
	while (1)
	{
	/* USER CODE END WHILE */
	/* USER CODE BEGIN 3 */
	/* USER CODE BEGIN 3 */
	
	HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 1);
	HAL_Delay(500); /* Insert delay 500 ms */
	HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 0);
	HAL_Delay(200); /* Insert delay 500 ms */
	
	/* USER CODE END 3 */
	}
	/* USER CODE END 3 */
	}

Utilisation de la fonction **HAL_GPIO_WritePin()** disponible sur le fichier **stm32l1xx_hal_gpio.c** :
Drivers/STM32L1xx_HAL_Driver/Src/stm32l1xx_hal_gpio.c

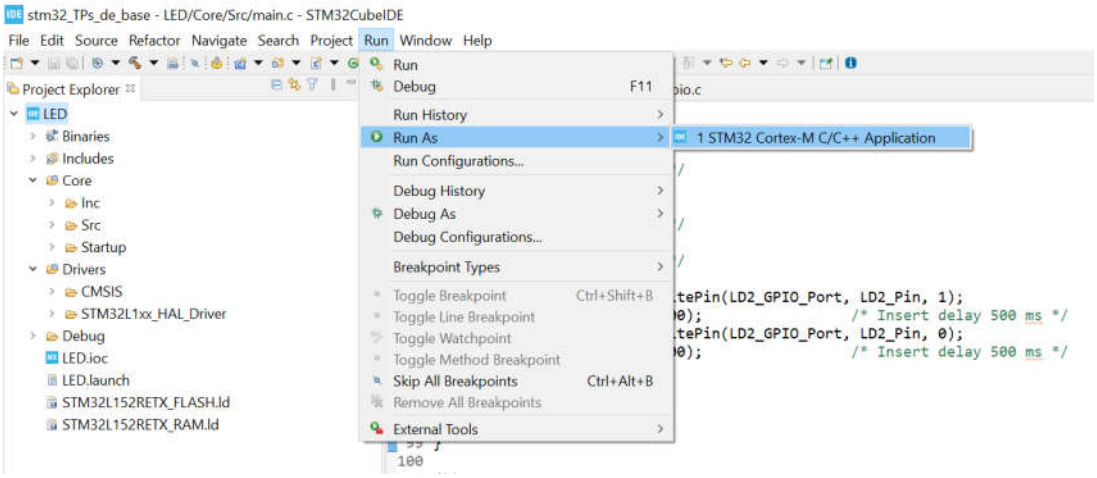




- **RE Compilation du projet « LED » : Onglet « Project » → « Build Project »**

Vérifier si O Errors et 0 Warnings

- **Programmation de la carte NUCLEO**



La LED LD1 clignote alternativement vert et rouge pendant la programmation, puis se stabilise au rouge une fois la programmation terminée.
Un appui sur le bouton-poussoir RESET B2 de la carte Nucleo-L152RE initialise le programme.

- **Vérification sur la carte NUCLEO**

La LED LD2 clignote : LD2 ON pendant 500ms et LD2 OFF pendant 200ms

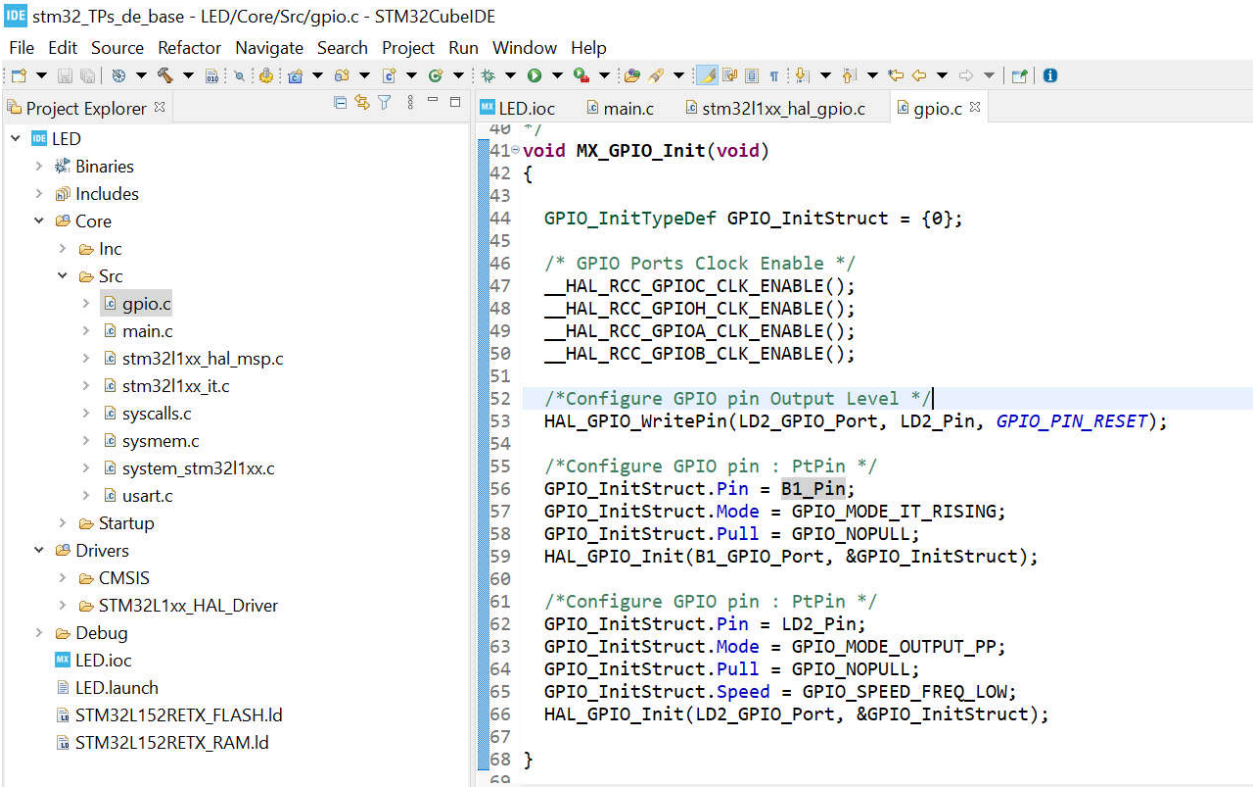
---

Détail sur la fonction d'initialisation des GPIO ***MX_GPIO_Init()***

Intéressons-nous à présent au code d'initialisation des GPIO.
Il suffit de se positionner sur la fonction ***MX_GPIO_Init()*** et d'appuyer sur F2.
Cette fonction est stockée dans le fichier **gpio.c**

```
main.c  stm32l1xx_hal_gpio.c
/* USER CODE END SysInit */
/* Initialize all configured peripherals */
MX_GPIO_Init();
/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : PtPin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : PtPin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

}
```

Le fichier qui contient la fonction ***MX_GPIO_Init()*** est le fichier ***gpio.c*** qui se trouve dans ***LED/Core/Src/gpio.c***.



Commentaires sur la fonction ***MX_GPIO_Init() :***

- ***Horloges :***

Nous constatons que les horloges des périphériques sont d'abord mises en route avec les fonctions ***__HAL_RCC_GPIOC_CLK_ENABLE()*** pour le port C, car le bouton-poussoir utilisateur B1 se trouve sur PC13, et ***__HAL_RCC_GPIOA_CLK_ENABLE()*** pour le port A, car la LED LD2 se trouve sur le PA5.

- ***Initialisation :***

La LED LD2 est éteinte en imposant unniveau bas sur PA5 avec la fonction suivante :

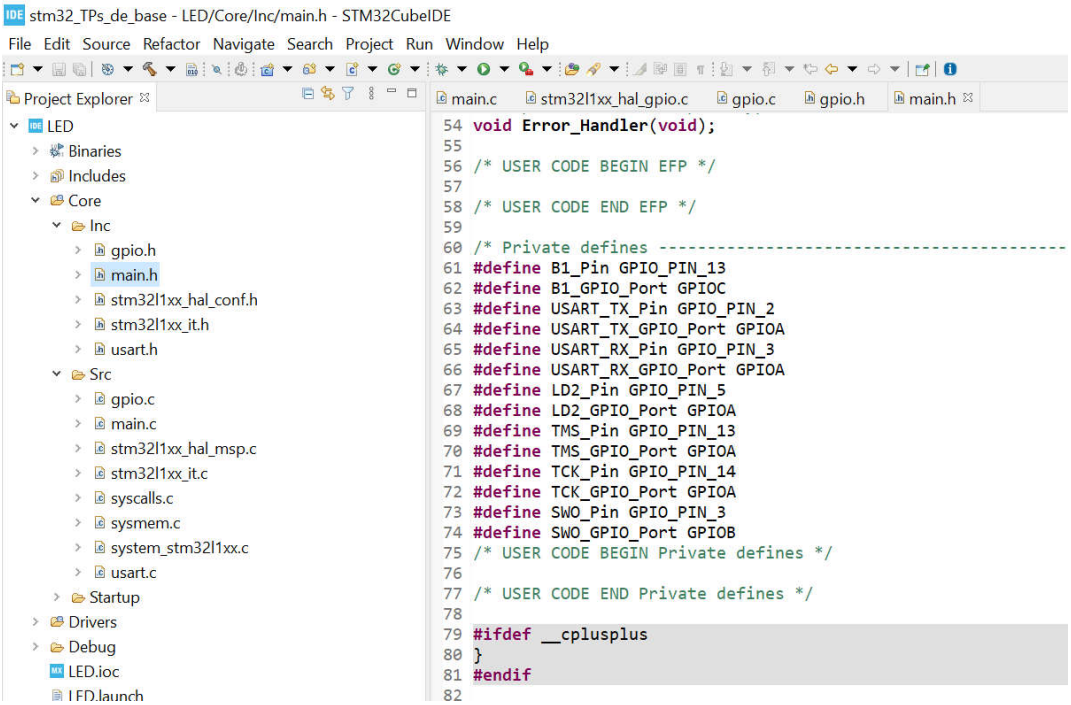
```
/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
```

- ***Configuration de B1 et de LD2 :***

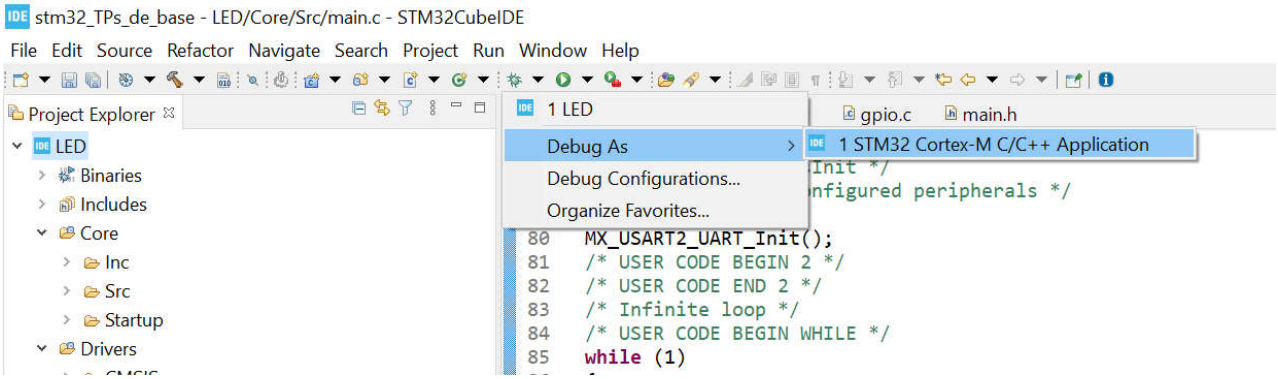
Ensuite, chaque GPIO sélectionné dans STM32CubeMX est configuré (dans cet exemple, B1 et LD2), avec son numéro **Pin**, son mode **Mode** et la configuration de sa broche **Pull**.

- ***Liaison avec les définitions dans le main.h***

On peut aussi vérifier en sélectionnant ***B1_Pin*** ou ***B1_GPIO_Port***, puis F2, puis F3, que le GPIO correspondant au bouton-poussoir utilisateur B1 a été redéfini par STM32CubeMX dans le fichier ***main.h*** et correspond bien à PC13.
De même, en sélectionnant ***LD2_Pin*** ou ***LD2_GPIO_Port***, puis F2, puis F3, que le GPIO correspondant à la LED LD2 est bien PA5.



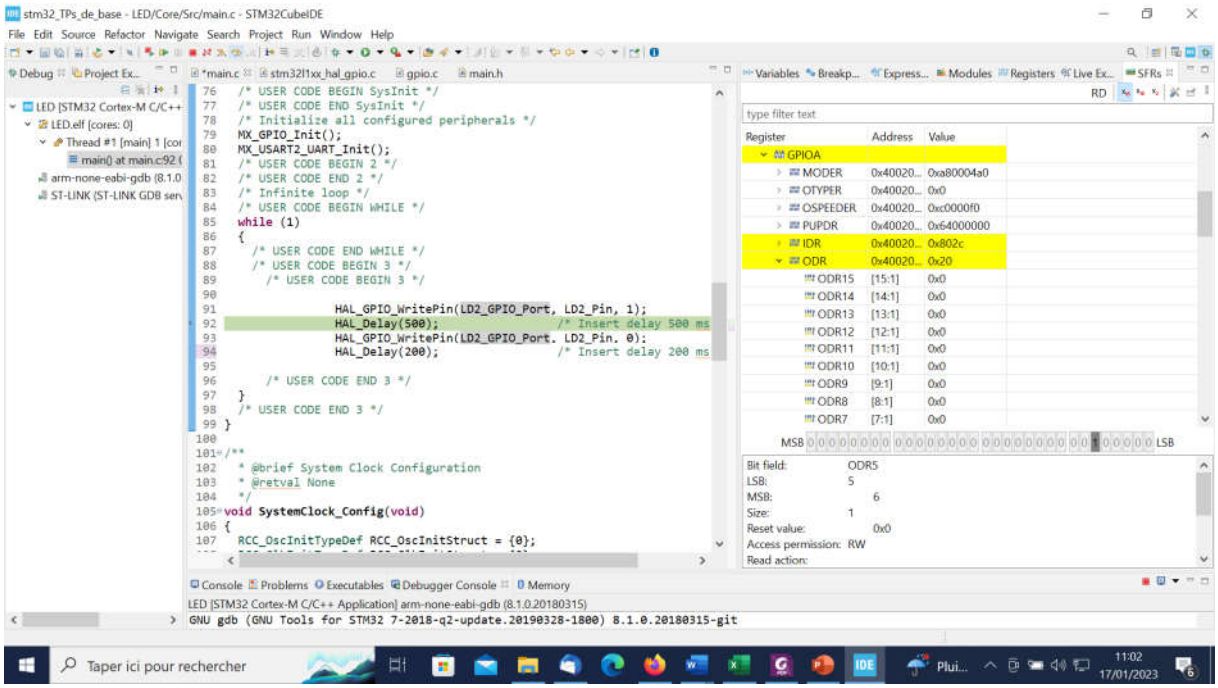
- **Programmation en mode Debug :**

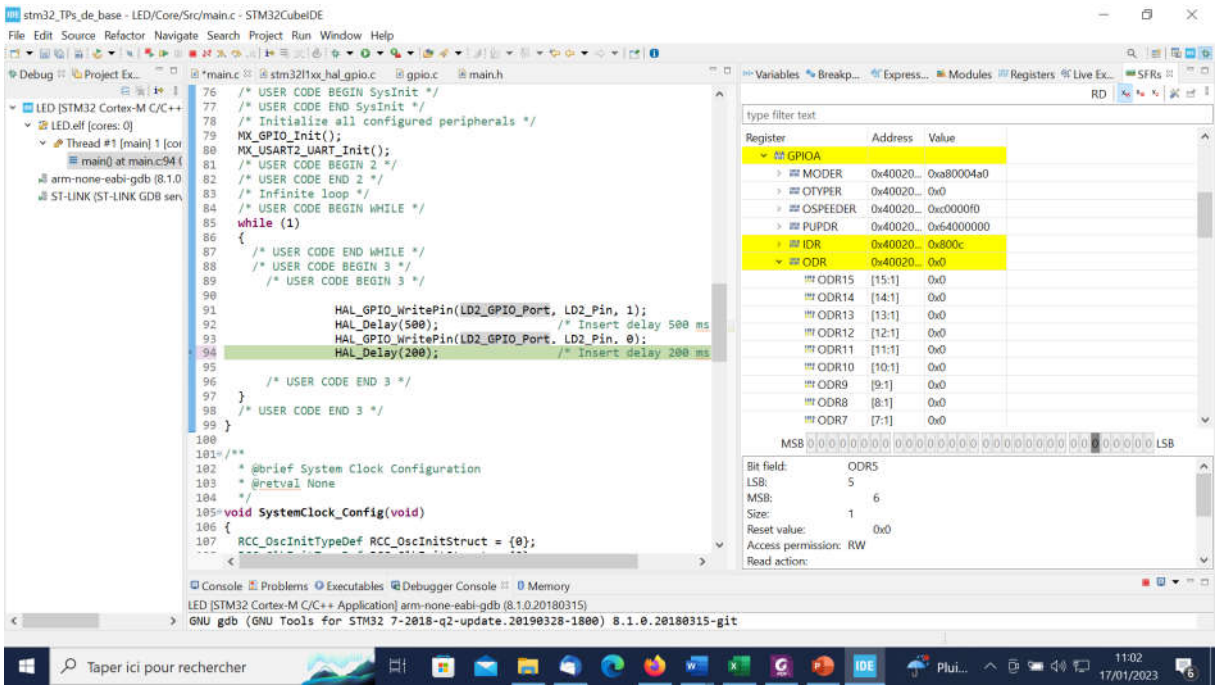


Le curseur se place au début du programme, vous pouvez avancer pas à pas en cliquant sur **Step Over**

Vérifier sur la carte NUCLEO au fur et à mesure de l'avancé du programme l'état de la LED

- **Visualisation du GPIOA et du registre ODR5 (LED → GPIOA PIN5)**


stm32_TPs_de_base - LED/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 1);
HAL_Delay(500); /* Insert delay 500 ms
HAL_GPIO_WritePin(LD2_GPIO_Port. LD2_Pin. 0);
HAL_Delay(200); /* Insert delay 200 ms
void SystemClock_Config(void)
GNU gdb (GNU Tools for STM32 7-2018-q2-update.20190328-1800) 8.1.0.20180315-git
Taper ici pour rechercher
11:02
17/01/2023


stm32_TPs_de_base - LED/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, 1);
HAL_Delay(500); /* Insert delay 500 ms
HAL_GPIO_WritePin(LD2_GPIO_Port. LD2_Pin. 0);
HAL_Delay(200); /* Insert delay 200 ms
void SystemClock_Config(void)
GNU gdb (GNU Tools for STM32 7-2018-q2-update.20190328-1800) 8.1.0.20180315-git
Taper ici pour rechercher
11:02
17/01/2023