ELE3312
Microcontrôleurs et applications
Laboratoire 2

Auteur : Jean Pierre David

Introduction

Les microcontrôleurs modernes sont des circuits de plus en plus complexes dans le sens où ils contiennent de nombreux périphériques, chacun nécessitant des dizaines, voire des centaines, de bits de configuration. Afin de faciliter leur configuration, les manufacturiers mettent à la disposition des concepteurs des environnements de développement pour leur permettre de spécifier leurs exigences à haut niveau et laisser les outils générer les bits de configuration automatiquement. En outre, les outils sont aujourd'hui capables de générer ce qu'on appelle le HAL (Hardware Abstraction Layer). C'est un ensemble de méthodes C qui permettent à l'utilisateur de programmer un microcontrôleur d'une manière similaire à la programmation C/C++ sur ordinateur. En tant qu'ingénieur, on s'attend à ce que vous soyez capable d'exploiter au maximum la programmation de haut niveau avec le HAL, tout en étant capable de le modifier ou de le contourner si besoin est. Cela exige de vous une bonne compréhension du fonctionnement du microcontrôleur, de son jeu d'instructions, du HAL, et de la programmation séquentielle en général. À ce stade, votre compréhension du fonctionnement du microcontrôleur et de son jeu d'instruction est probablement encore assez rudimentaire. Ce laboratoire vise essentiellement à vous introduire à l'outil STM32Cube, qui vous permettra de générer le HAL à partir d'une description de haut niveau, et au logiciel Keil, qui vous permettra de compiler, simuler et exécuter les applications sur votre carte de prototypage.

Objectifs

1. Installer et configurer votre environnement de développement STM32Cube et Keil sur votre ordinateur personnel
2. Configurer le HAL pour votre carte microcontrôleur avec des périphériques de base
3. Utiliser Keil pour compiler et exécuter des applications sur votre carte microcontrôleur
4. Découvrir quelques fonctions utiles du HAL
5. Gérer les ports d'entrée/sortie

Prérequis

À partir de ce laboratoire, vous devrez toujours écrire, compiler, exécuter et déboguer du code C (en plus du code assembleur dans les prochains laboratoires). La programmation C est un prérequis indispensable à ce cours. Elle est vue au cours de première année INF1005C (programmation procédurale). Si vous connaissez la programmation procédurale mais pas la programmation C, il vous appartient de l'apprendre très rapidement. Les diapositives du cours INF1005C sont disponibles sur le site du cours ELE3312. Vous avez aussi toutes les ressources nécessaires sur le web, en particulier, le livre de référence par l'auteur du C lui-même :

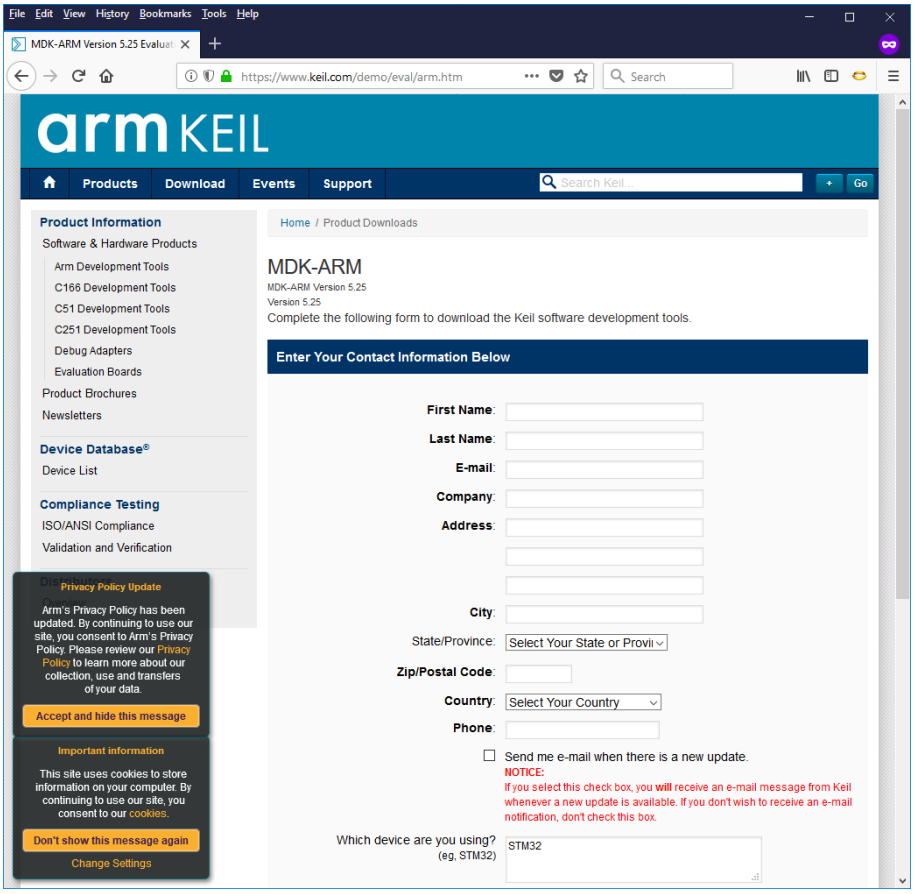
> ***Le Langage C, norme ANSI****, Brian W Kernighan et Dennis M Ritchie*

1ère partie : préparation à la maison

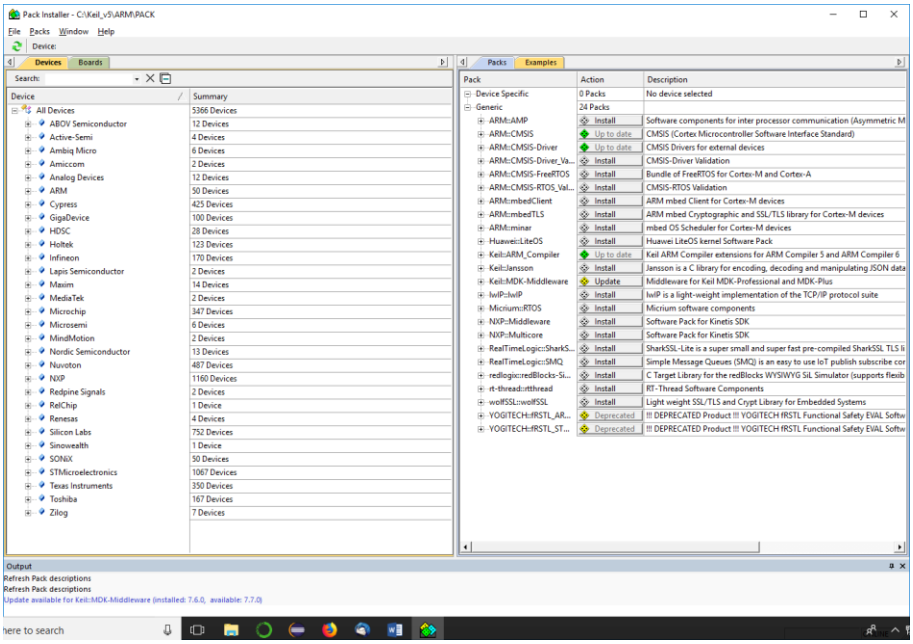
Mode opératoire

Installation de l'environnement *Keil*

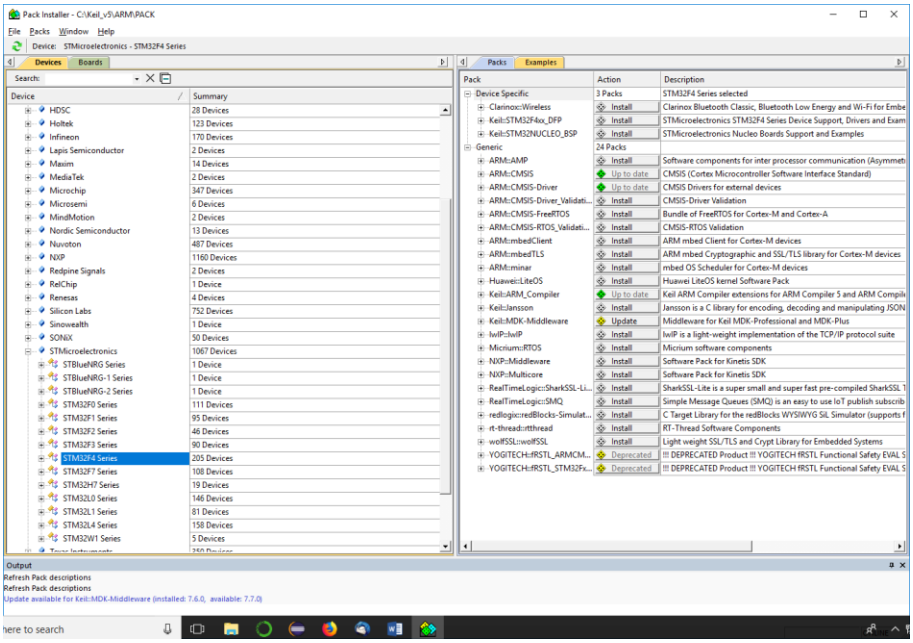
1. Allez à l'adresse suivante : https://www.keil.com/demo/eval/arm.htm
2. Complétez le formulaire pour vous donner accès à la version d'évaluation



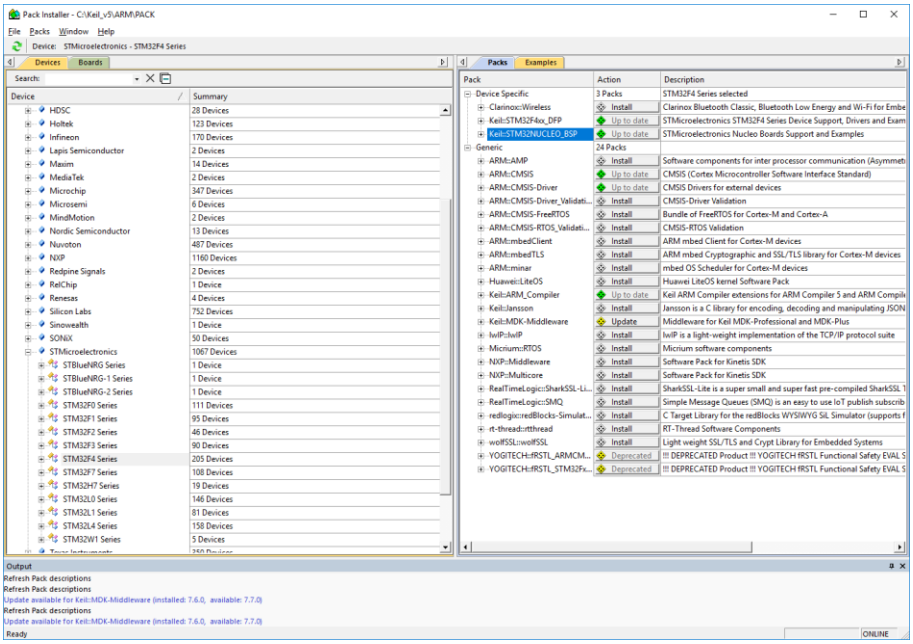
3. Téléchargez MDK-ARM
4. Installez l'environnement MDK-ARM en acceptant toutes les options par défaut
5. À la fin de l'installation, le Pack installer va vous permettre de télécharger les librairies dont vous aurez besoin :



6. Sélectionnez STMicroelectronics/STM32F4 Series (volet de gauche). Cela fera apparaître Keil::STM32F4xx_DFP et Keil::STM32NUCLEO_BSP en haut du volet de droite :



7. Cliquez sur le bouton Install à droite de Keil::STM32F4xx_DFP
8. Cliquez sur le bouton Install à droite de Keil::STM32NUCLEO_BSP
9. Une fois que les librairies sont installées, vous devriez obtenir l'écran suivant :

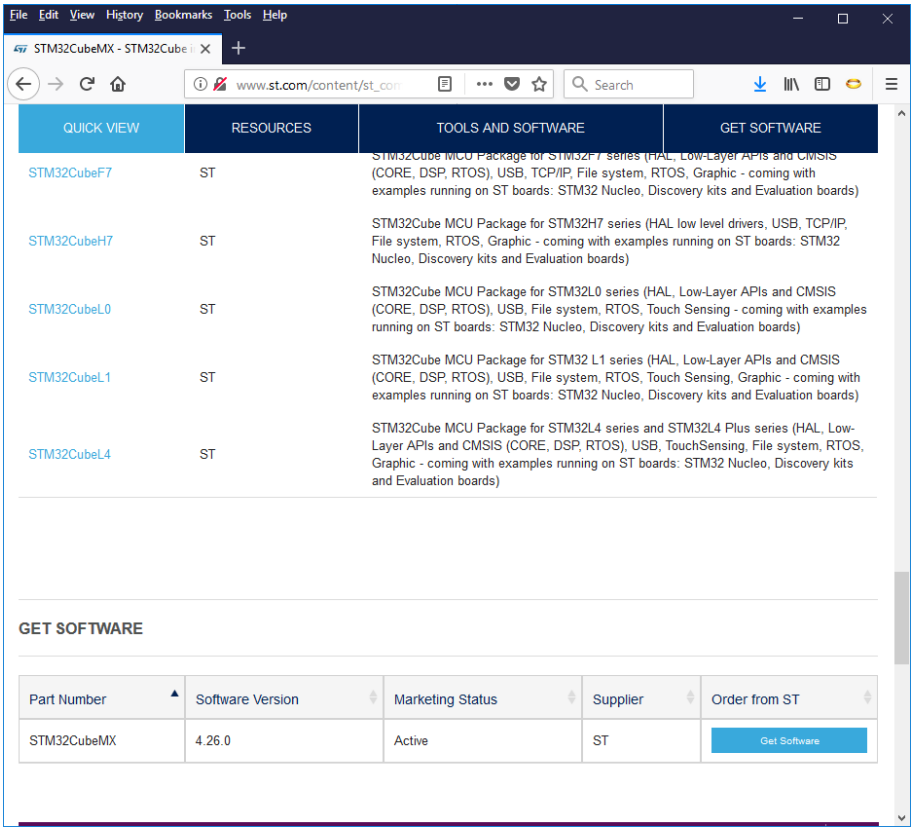


10. Fermer la fenêtre

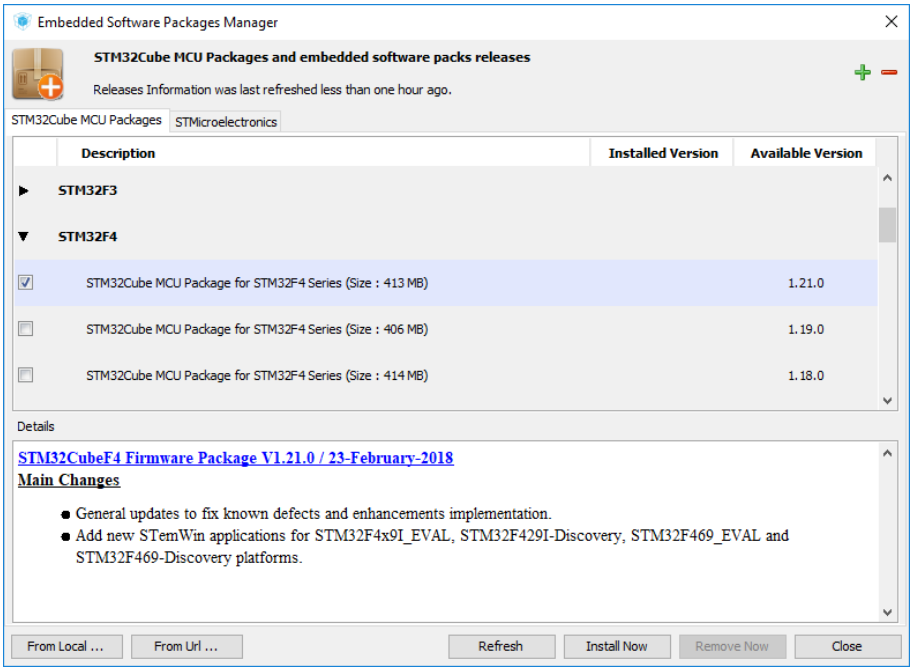
Installation de STM32Cube

Pour générer le HAL pour votre carte de développement de type STM32 (typiquement la carte NUCLEO-F446RE), vous aurez besoin de l'utilitaire STM32Cube, développé par le manufacturier.

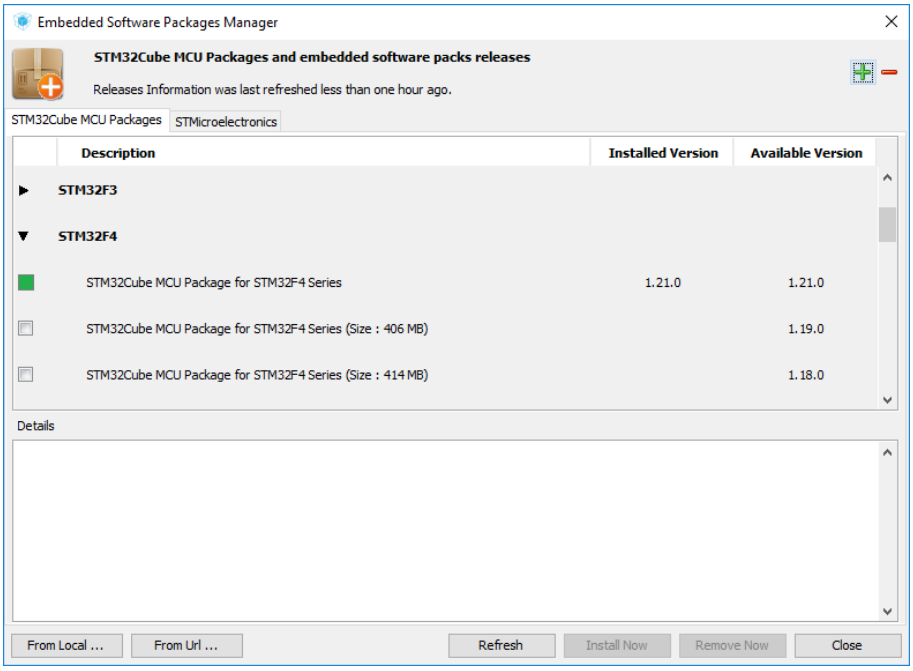
1. Allez à l'adresse suivante : http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html
2. Téléchargez l'utilitaire STM32CubeMX au bas de la page (bouton « Get Software »). Si ce n'est déjà fait, vous devrez vous enregistrer sur le site du manufacturier.



3. Décompressez l'archive et lancez l'application *SetupSTM32CubeMX-…exe*
4. Suivez les instructions pour installer le logiciel sur votre ordinateur et acceptez toutes les options par défaut.
5. Démarrez l'application STM32CubeMX
6. Allez dans le menu Help/Manage embedded software packages.
7. Cliquez sur STM32F4 et sélectionnez la version la plus récente :



8. Cliquez sur le bouton *Install Now.* À la fin de l'installation, la fenêtre devrait être similaire à la suivante :



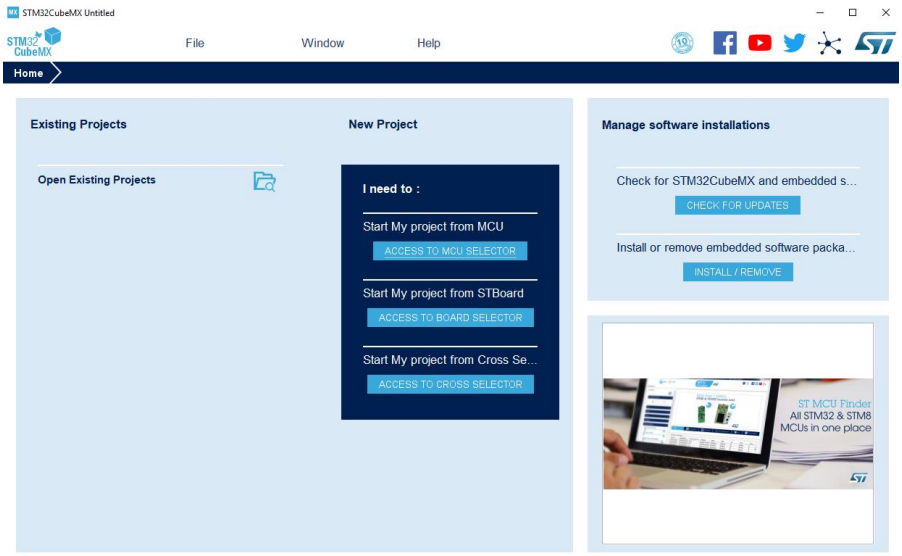
9. Fermez la fenêtre. Votre environnement de conception pour la série STM32F4 est maintenant prêt à être utilisé

Premier projet avec STM32Cube et Keil

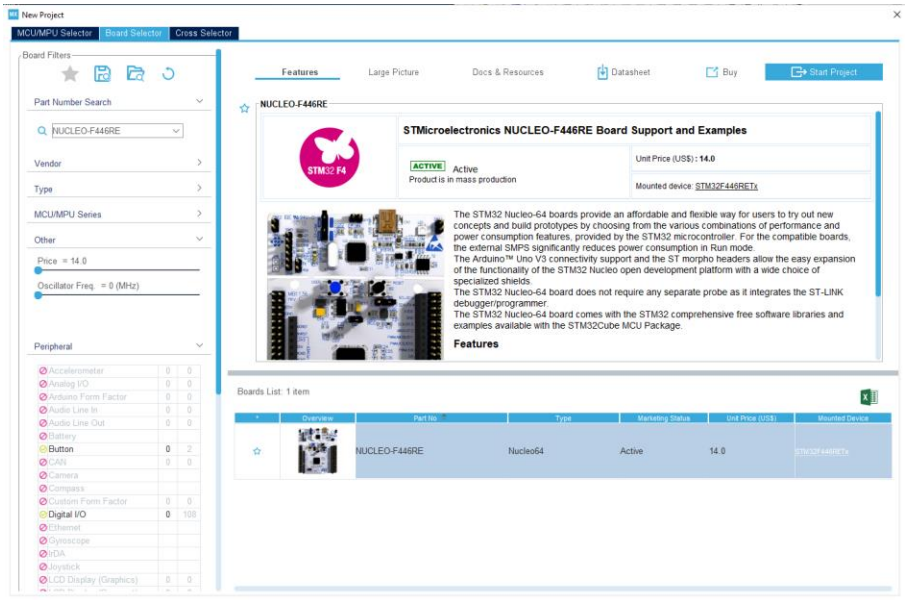
Avertissement important : de manière générale, n'utilisez JAMAIS un chemin de répertoire ou un nom de fichier contenant un espace ou des caractères spéciaux (accents etc.)

Génération du HAL avec STM32Cube

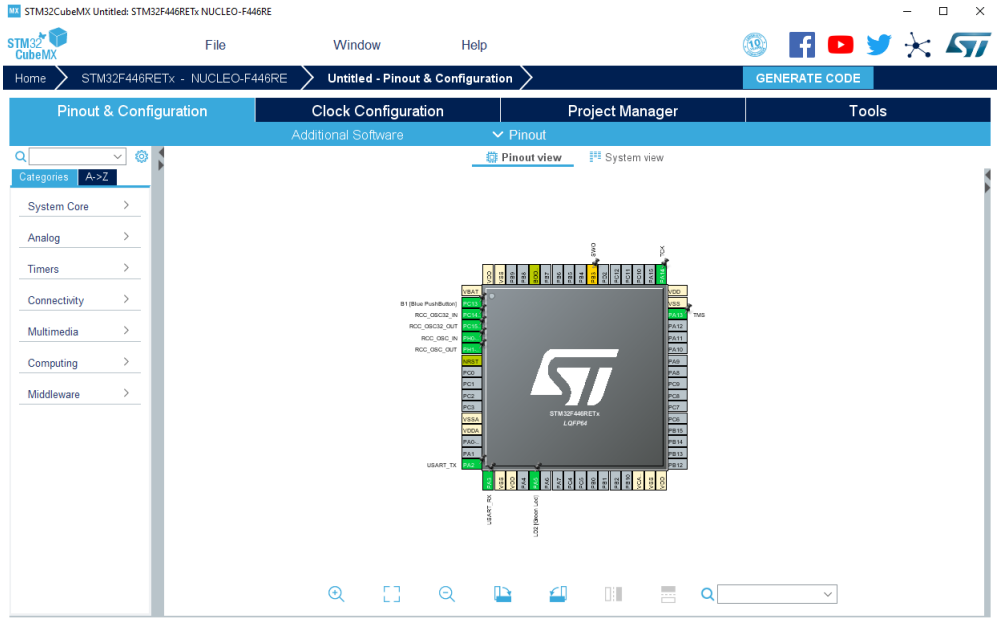
1. Dans la fenêtre d'accueil, cliquez sur *File* -> *New Project*



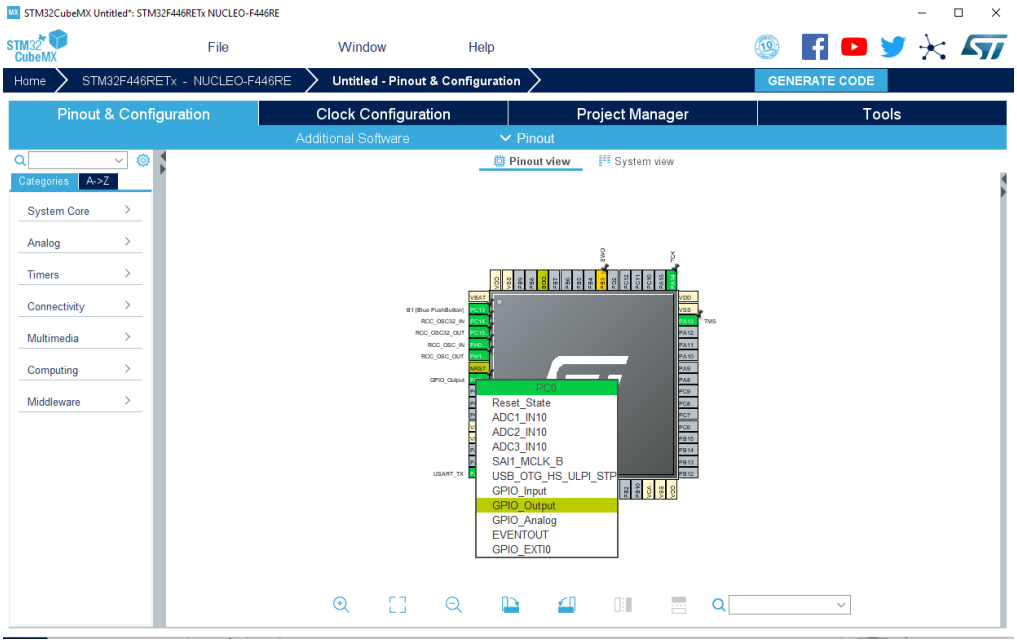
2. Cliquez sur le menu *Board Selector*
3. En utilisant les critères de sélection, allez chercher la carte que vous utilisez (en principe NUCLEO-F446RE) :



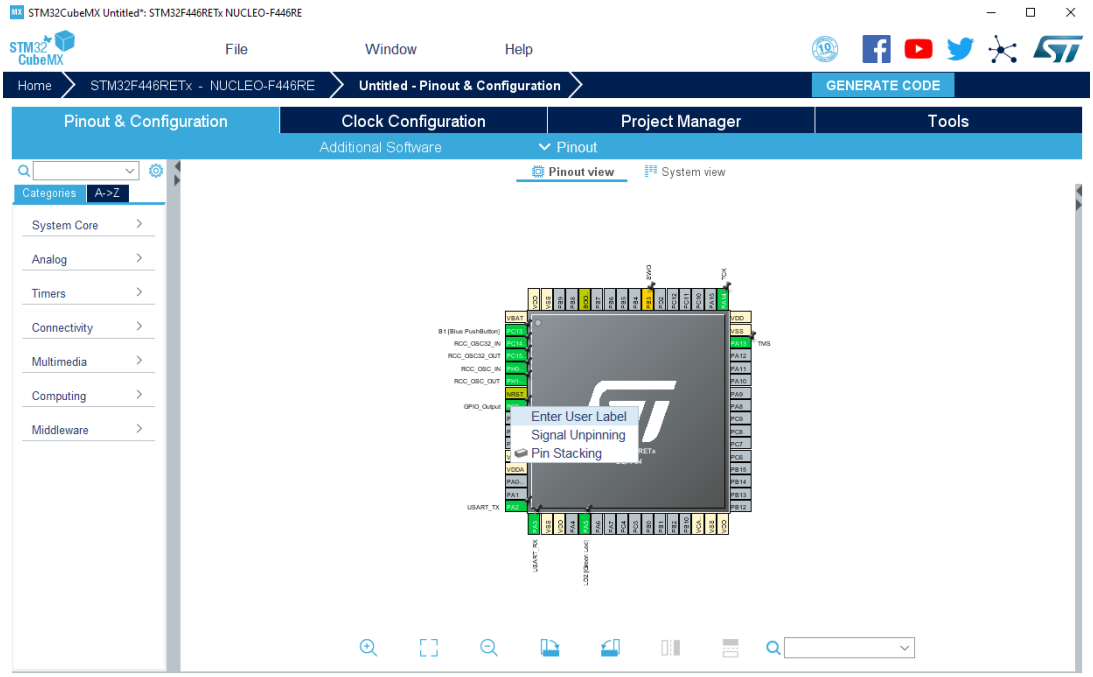
4. Cliquez sur le bouton Start Project (en haut à droite) et choisissez d'initialiser les périphériques avec leur mode par défaut. Vous verrez apparaitre une image correspondant au microcontrôleur que vous avez sur votre carte de développement :



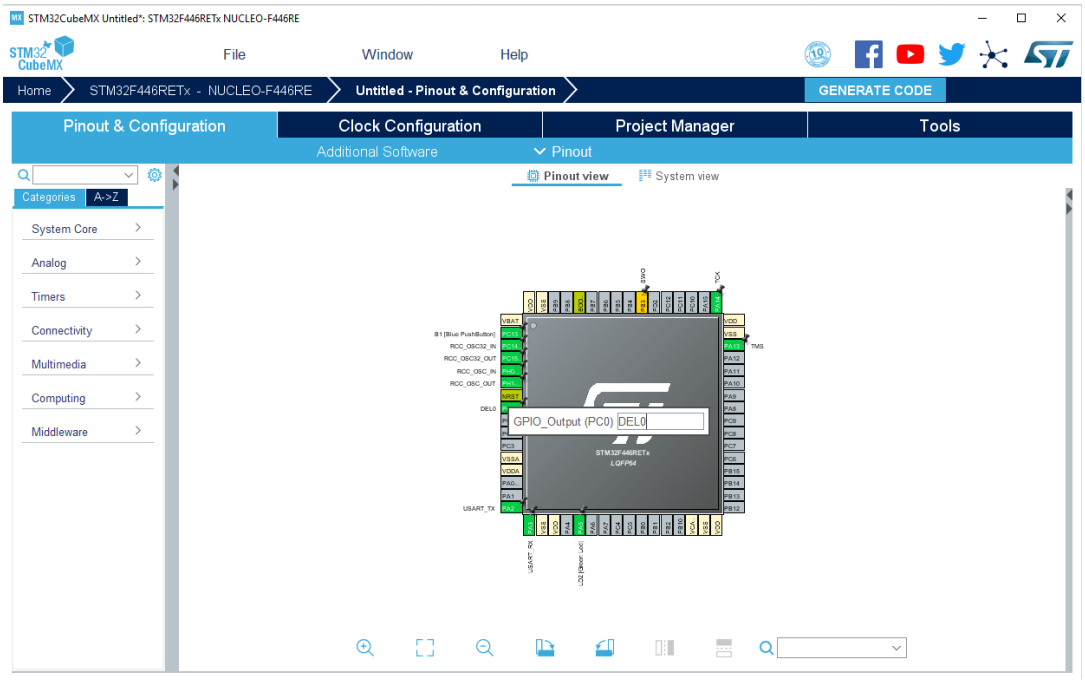
5. Dans un premier temps, nous allons simplement configurer les broches PC0 à PC7 en mode output et nous allons leur donner un nom significatif. Cliquez sur la broche PC0 et sélectionnez le mode GPIO_OUTPUT :



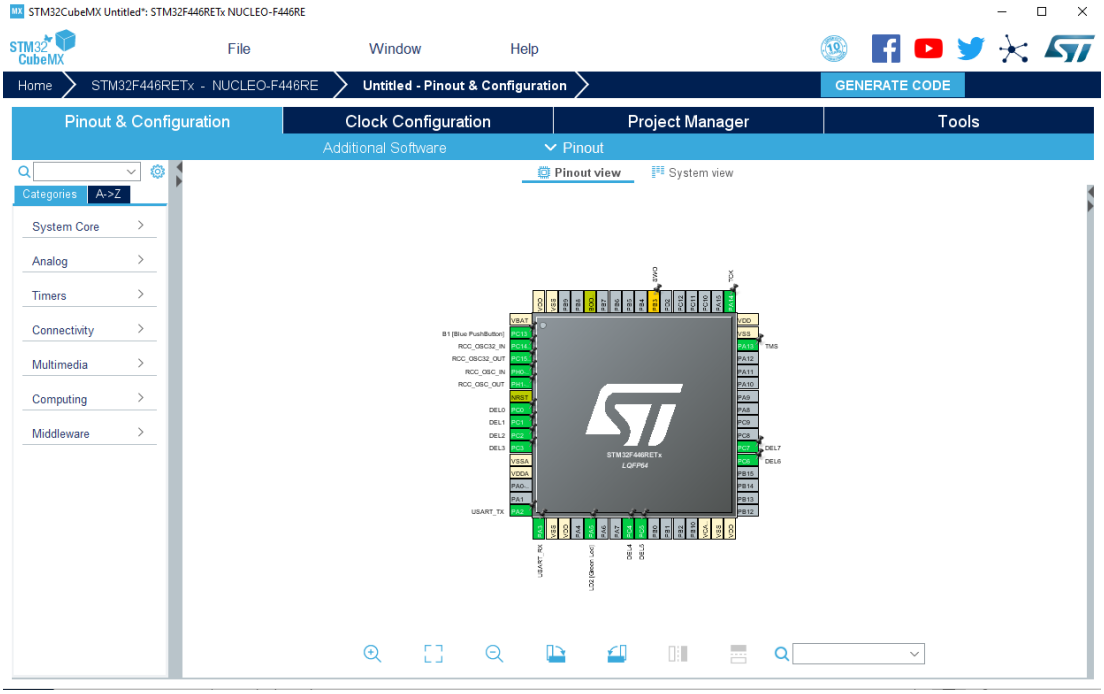
6. Cliquez avec le bouton droit de la souris sur PC0 et sélectionnez *Enter User Label* :



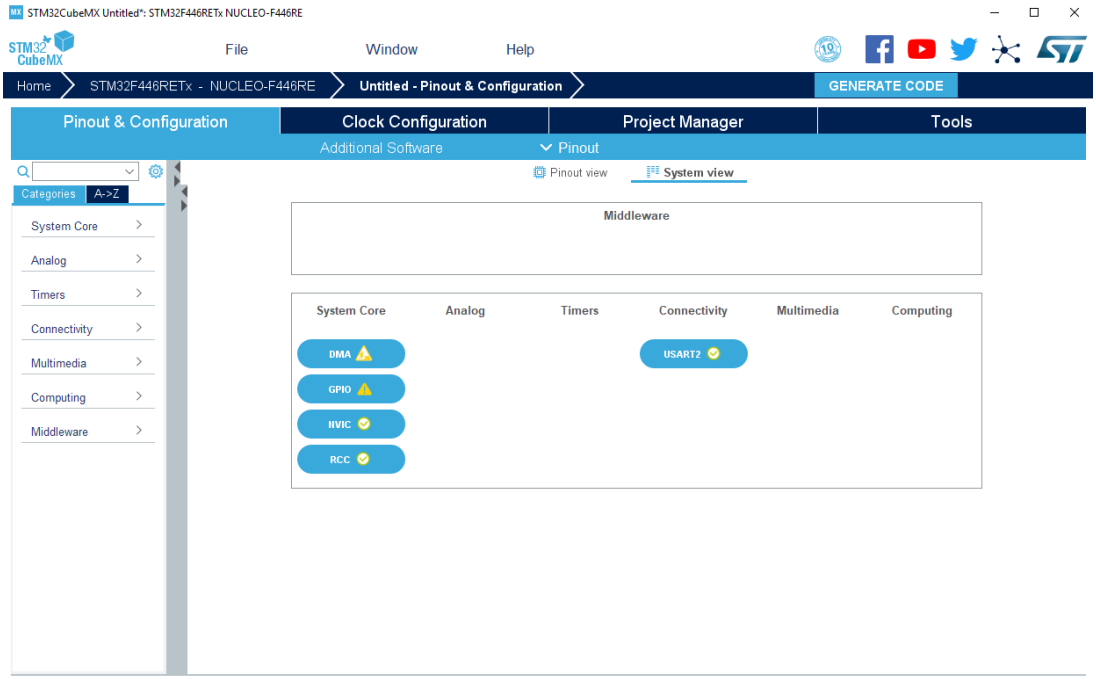
7. Nommez la broche DEL0 :



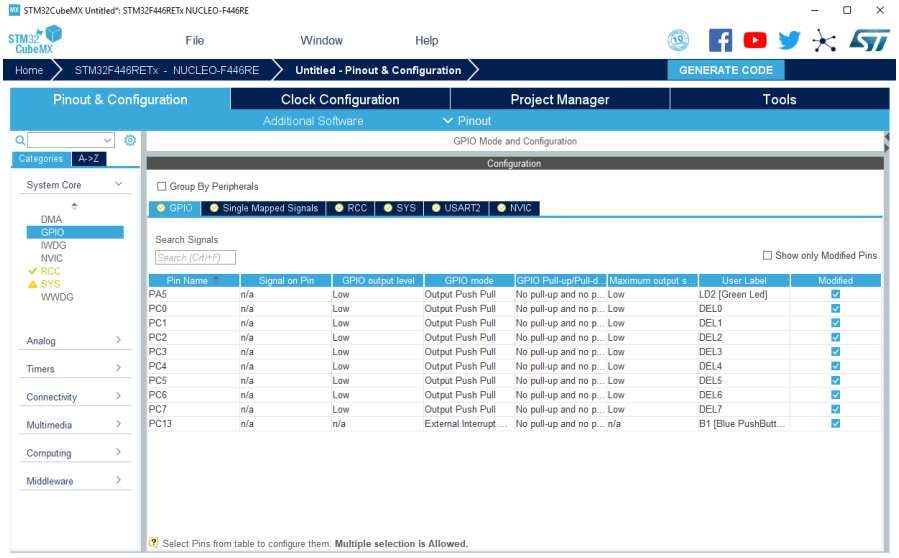
8. Répétez la procédure pour les broches PC1 à PC7 :



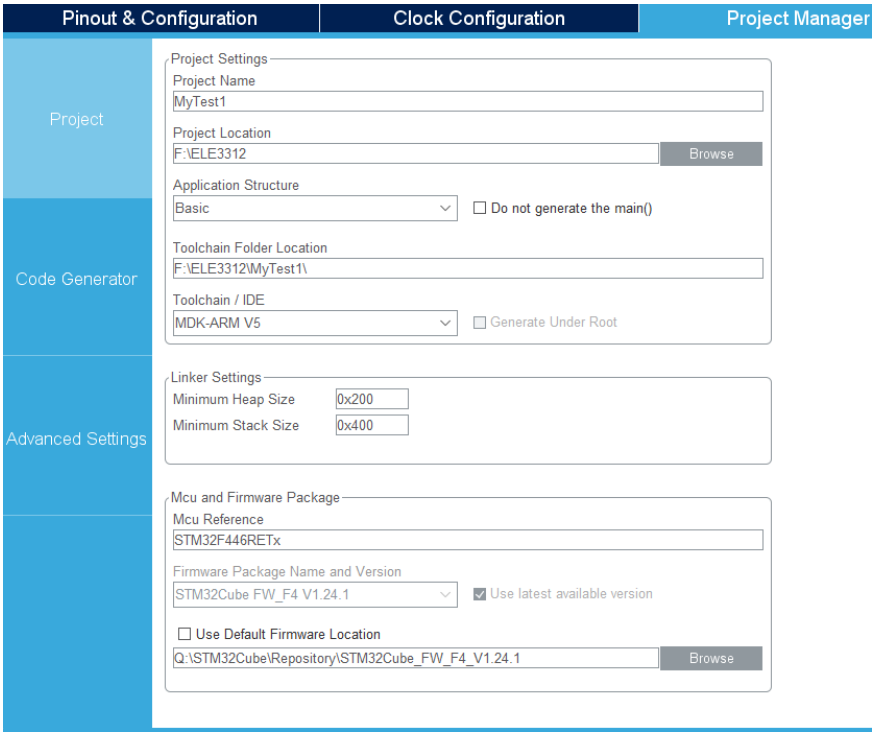
9. Cliquez sur l'onglet *System View* :



10. Cliquez sur le bouton GPIO pour visualiser vos configurations :

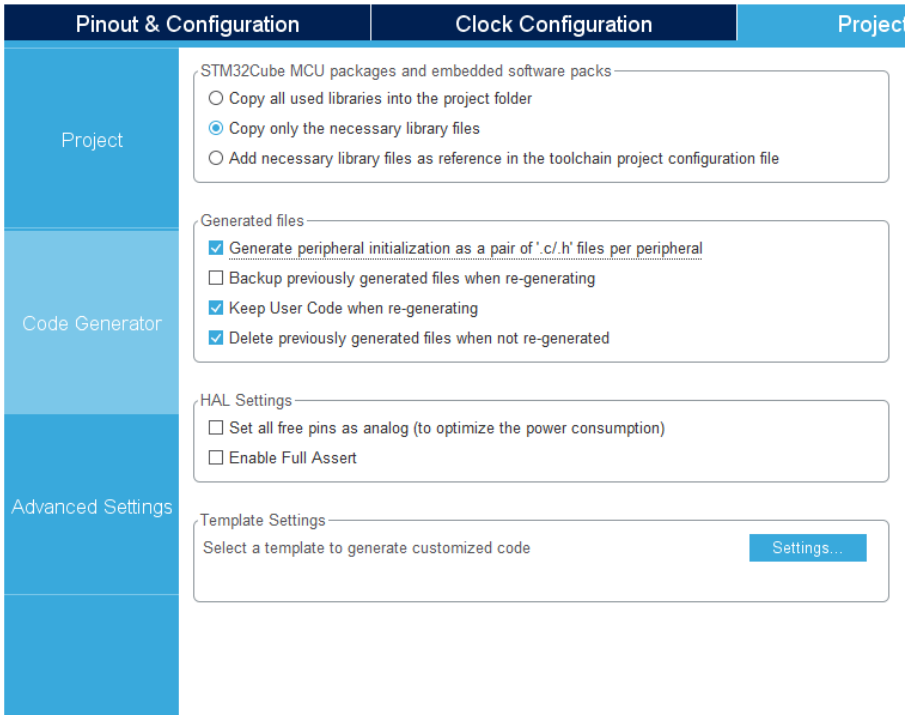


11. À ce stade, nous conservons les configurations par défaut (valeur de sortie initialisée à 0, mode push-pull, pas de pull-up et pas de pull-down, faible vitesse de changement.
12. Allez dans le menu Project Manager. Nous devons donner un nom au projet (par exemple MyTest1), un emplacement sur le disque (dans cet exemple nous utilisons F:\ELE3312 mais vous pouvez choisir votre propre emplacement sur votre disque dur), et configurer certains paramètres :

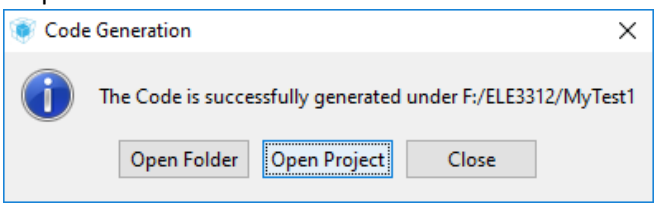


Emplacement du firmware au laboratoire (utilisez le firmware le plus récent):
`Q:\STM32Cube\Repository\STM32Cube_FW_F4_V1.25.0`
Note : Sur votre ordinateur personnel, choisissez l'option *Use Default Firmware Location.*

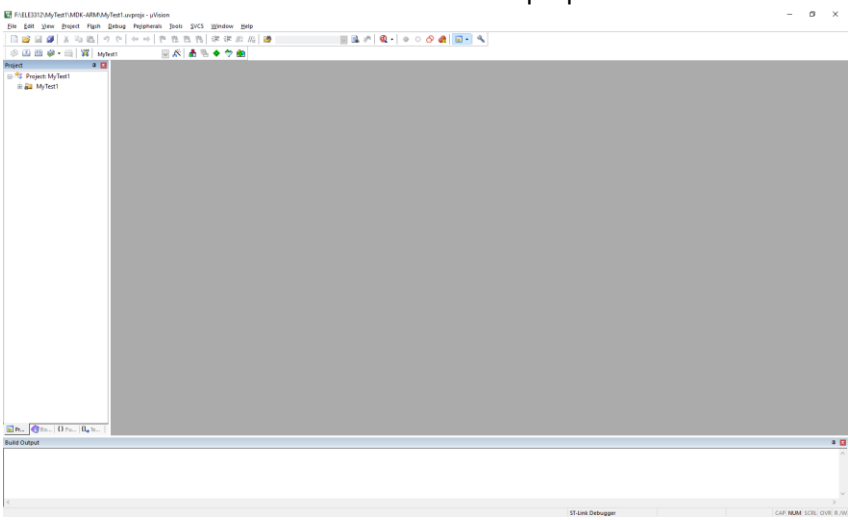
13. Dans l'onglet Code Generator, configurez les options comme suit :



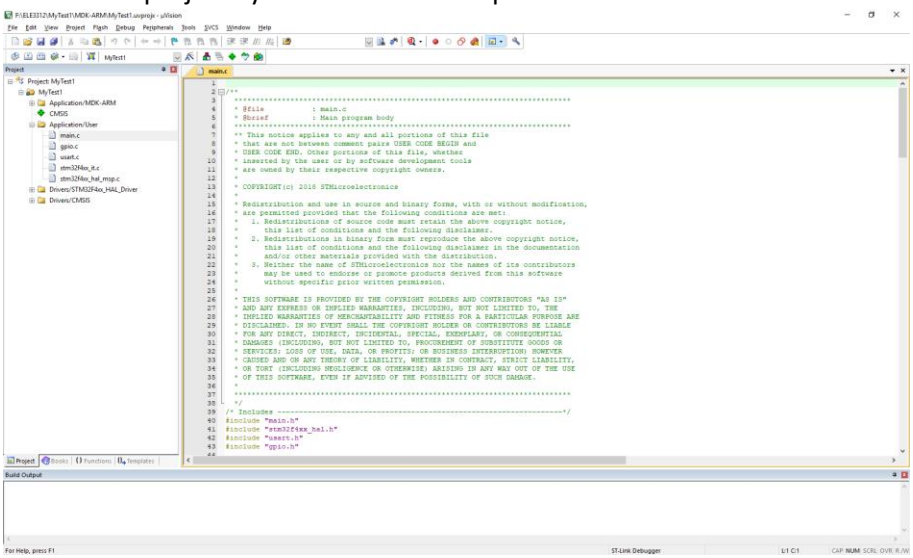
14. Cliquez sur le bouton *Generate Code* :



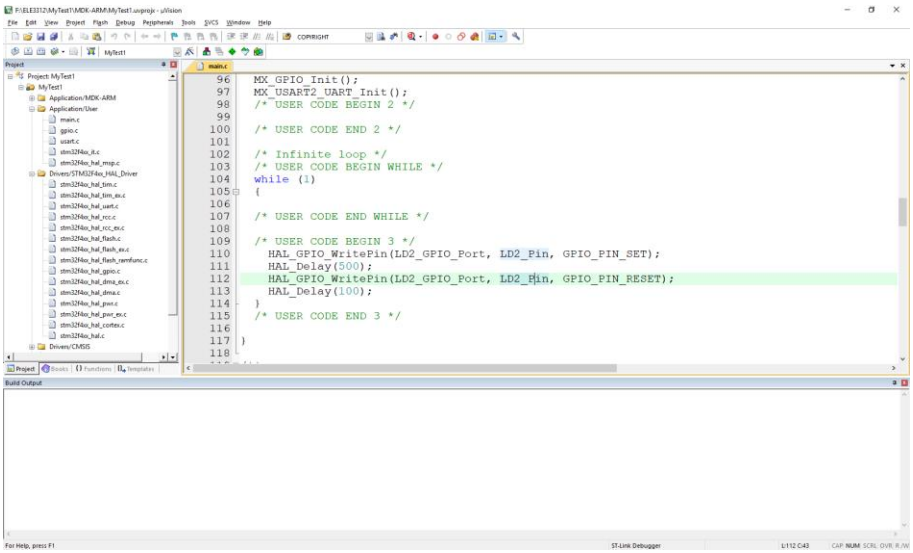
15. Cliquez sur Open Folder
16. Allez dans le répertoire MDK-ARM et double-cliquez sur le fichier MyTest1 avec l'icône du logiciel uVision (Keil) :
17. Keil s'ouvre avec votre environnement tout préparé :



18. Déroulez le projet MyTest1 et double-cliquez sur le fichier main.c :

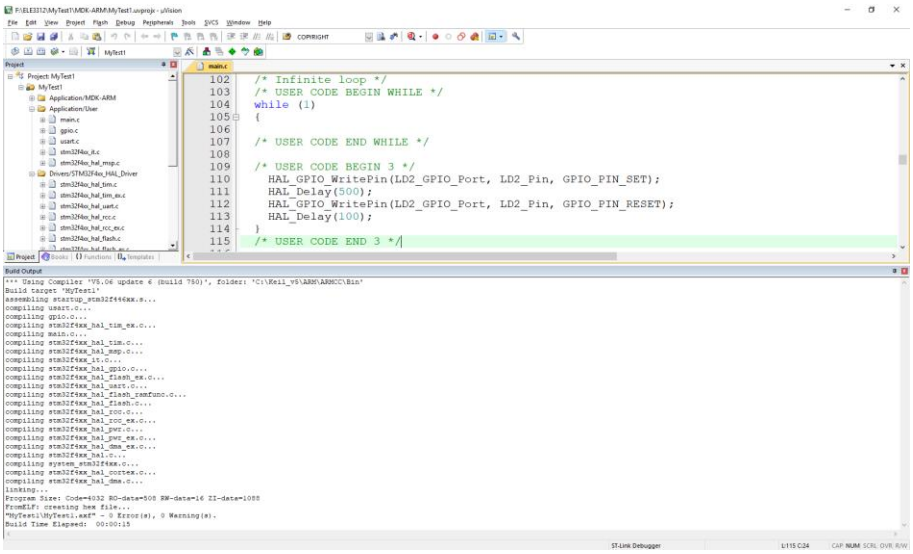


19. Éditez la fonction main comme suit :

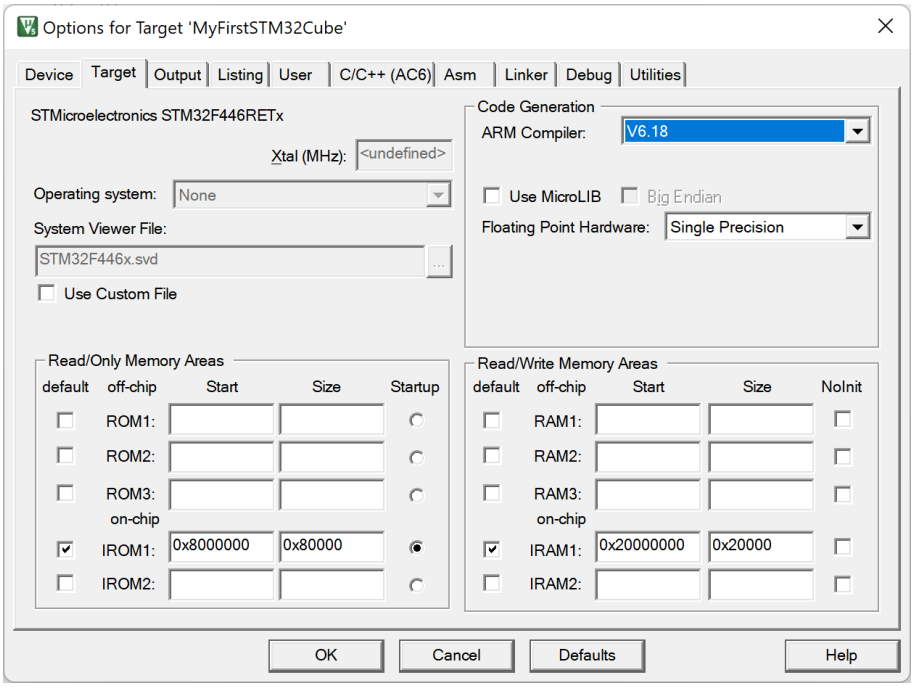


```
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
HAL_Delay(500);
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
HAL_Delay(100);
```

20. Appuyez sur F7 pour compiler votre projet :

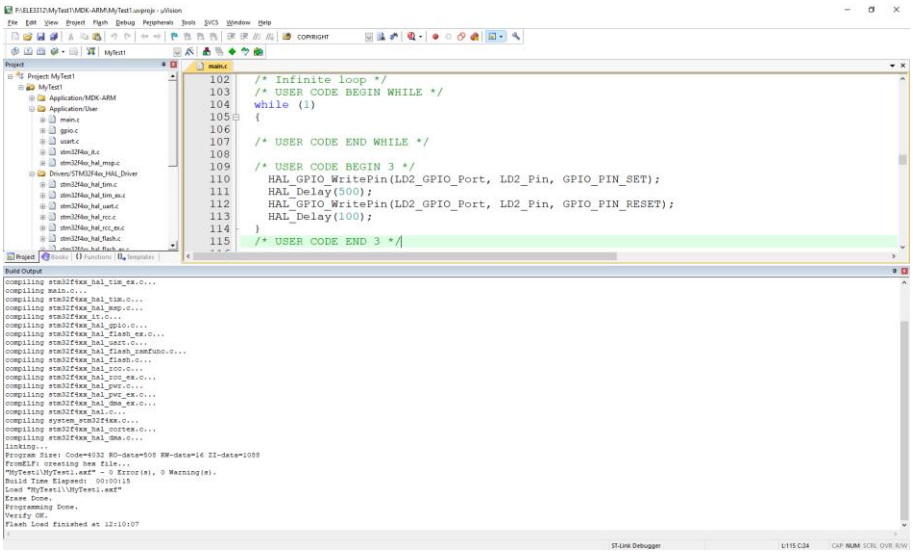


Si la version 5 du compilateur ARM n'est pas installée, allez dans Project->Options… Target



Choisissez la version 6.xx pour *ARM compiler* et recompilez (F7)

21. Connectez votre carte de développement à votre PC et appuyez sur F8 pour téléverser votre programme dans la carte :



22. Pour démarrer l'exécution, pesez sur le bouton noir (RESET) de la carte. Contemplez votre œuvre, la DEL verte clignote 500ms allumée, 100ms éteinte.
23. Essayez d'autres délais pour vérifier que vous maitrisez les fonctions de compilation (F7), télé-versement (F8) et exécution (bouton noir de RESET).

Premiers pas avec STM32Cube, le HAL, Keil et la carte NUCLEO

La documentation de STM32Cube se trouve ici (**UM1718**, 319 pages) :
http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html

La documentation du HAL se trouve ici (**UM1725**, 1838 pages) :
www.st.com/resource/en/user_manual/dm00105879.pdf

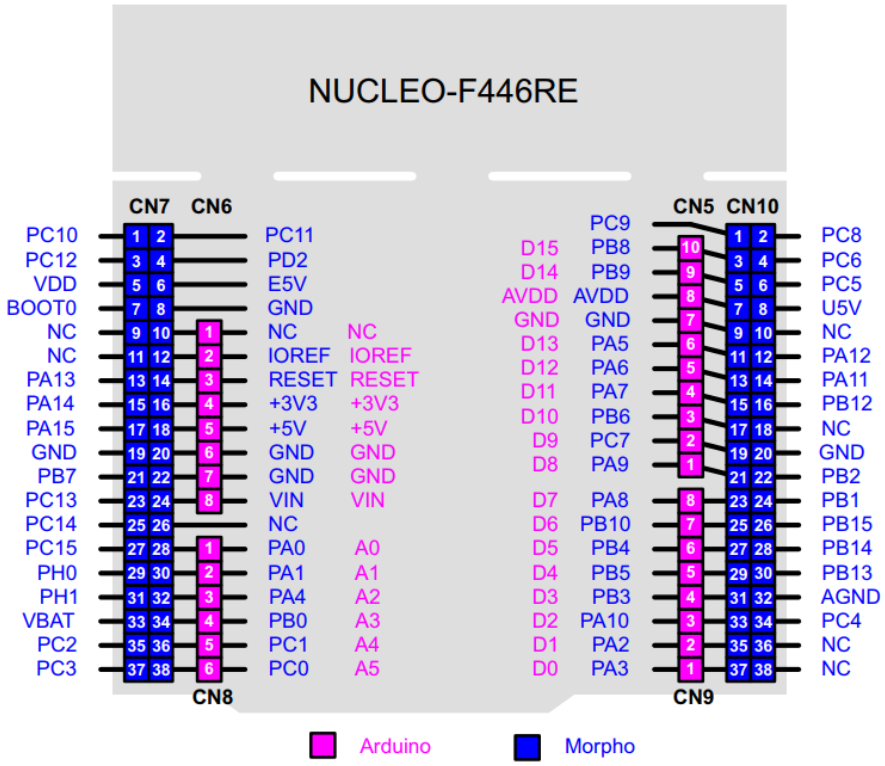
La documentation de Keil (en ligne, impossible de savoir combien de pages) se trouve ici :

http://www.keil.com/support/man/docs/uv4/

La documentation de la carte NUCLEO (**UM1724**, 69 pages) se trouve ici :

www.st.com/resource/en/user_manual/dm00105823.pdf

On y trouve notamment le schéma suivant avec la description de chacune des broches des connecteurs :



Les datasheets du processeur STM32F446RE (202 pages) sont ici :

http://www.st.com/resource/en/datasheet/stm32f446re.pdf

Le manuel de référence du processeur STM32F446RE (**RM0390**, 1328 pages) est ici :

http://www.st.com/resource/en/reference_manual/dm00135183.pdf

Le guide de programmation (**PM0214**, 260 pages) du Cortex M4 (le cœur du circuit STM32F446RE), se trouve ici :

www.st.com/resource/en/programming_manual/dm00046982.pdf

Le manuel de référence de l'architecture (916 pages) ARMv7-M (utilisée par le Cortex M4) se trouve ici :

https://static.docs.arm.com/ddi0403/eb/DDI0403E_B_armv7m_arm.pdf

On s'attend à ce que vous maitrisiez tous ces documents dans les plus brefs délais … Enfin, non, ce n'est pas possible (ouf !) La pédagogie que nous allons utiliser dans ce cours consiste à vous guider pas à pas au moyen d'exemples. Quand vous aurez besoin d'information technique, on vous donnera des indications pour trouver l'information dans les documents sans que cela prenne trop de temps. Toutefois, **trouver l'information recherchée parmi les documents disponibles est une compétence essentielle à ce cours et elle sera évaluée à l'examen.** Assurez-vous donc de vous pratiquer personnellement à cet exercice (plutôt que de faire confiance à un partenaire ou à une autre équipe). **Prenez donc le temps dès maintenant de télécharger et de feuilleter rapidement chaque document, avec une attention particulière sur la table des matières.**

Expérience 1

Identifiez les broches des connecteurs correspondant aux broches DEL0...3 Connectez-y des DELs et réalisez un jeu de lumières dans lequel la lumière passe d'une DEL à la suivante dans une rotation sans fin.

Vous avez assigné les signaux DEL0...3 aux ports PC0...3 En regardant le diagramme de l'assignation des broches présenté un peu plus haut vous pouvez trouver tous les ports et connecter chacun d'eux à l'anode d'une diode. La cathode sera reliée à la masse à travers une résistance de 330R pour chaque DEL. Ensuite, essayer le code suivant dans la boucle infinie :

```
HAL_GPIO_WritePin(DEL0_GPIO_Port, DEL0_Pin, GPIO_PIN_SET);
HAL_Delay(500);
HAL_GPIO_WritePin(DEL0_GPIO_Port, DEL0_Pin, GPIO_PIN_RESET);

HAL_GPIO_WritePin(DEL1_GPIO_Port, DEL1_Pin, GPIO_PIN_SET);
HAL_Delay(500);
HAL_GPIO_WritePin(DEL1_GPIO_Port, DEL1_Pin, GPIO_PIN_RESET);

HAL_GPIO_WritePin(DEL2_GPIO_Port, DEL2_Pin, GPIO_PIN_SET);
HAL_Delay(500);
HAL_GPIO_WritePin(DEL2_GPIO_Port, DEL2_Pin, GPIO_PIN_RESET);

HAL_GPIO_WritePin(DEL3_GPIO_Port, DEL3_Pin, GPIO_PIN_SET);
HAL_Delay(500);
HAL_GPIO_WritePin(DEL3_GPIO_Port, DEL3_Pin, GPIO_PIN_RESET);
```

Si vous connaissez un peu la structure de votre microcontrôleur, vous savez que les bits du port C sont regroupés en mots de 16 bits et sont accessibles au moyen du pointeur *GPIOC*. Ce pointeur pointe vers une région mémoire constituée de plusieurs champs, dont le champ *ODR*, qui contient l'état de sortie de chaque bit. Essayez le code suivant dans la boucle infinie :

```
GPIOC->ODR = 1;  HAL_Delay(500);
GPIOC->ODR = 2;  HAL_Delay(500);
GPIOC->ODR = 4;  HAL_Delay(500);
GPIOC->ODR = 8;  HAL_Delay(500);
```

Le comportement est le même que le code précédent mais il est plus simple à écrire, à comprendre et à exécuter. Vous aurez remarqué que les constantes 1, 2, 4 et 8 correspondent aux valeurs binaires 0b0001, 0b0010, 0b0100, 0b1000, de sorte que les DEL 0 à 3 s'allument encore en séquence.

Expérience 2

Réaliser un compteur 4 bits qui s'incrémente chaque seconde.

Essayez le code suivant pour la boucle infinie :

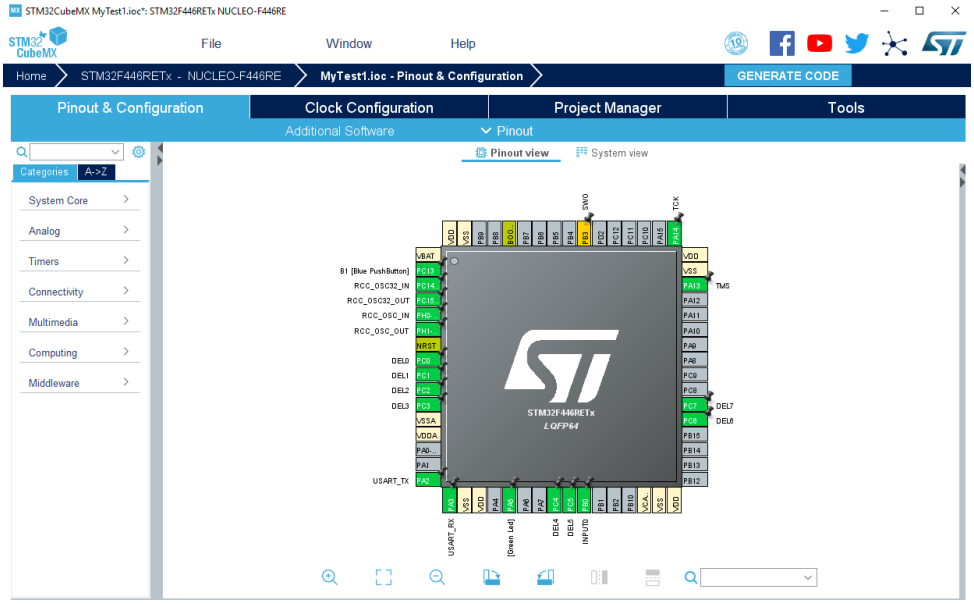
```
int count = 0;

while (1)
{
	GPIOC->ODR = count;
	count = count+1;
	if (count == 16) count = 0;
	HAL_Delay(1000);
}
```

Que se passe-t-il si vous supprimez la ligne `if (count == 16) count = 0;` ? Expliquez !

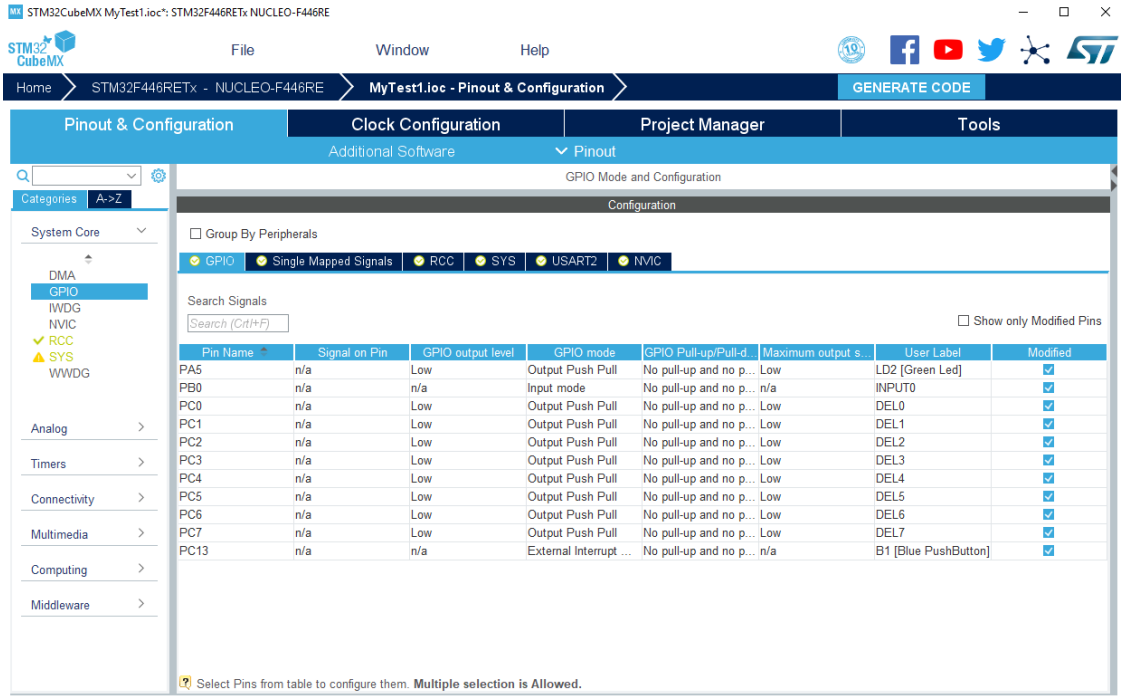
Expérience 3

Utiliser le port PB0 comme entrée pour contrôler le compteur binaire. Le compteur ne doit compter que lorsque PB0 est vrai.

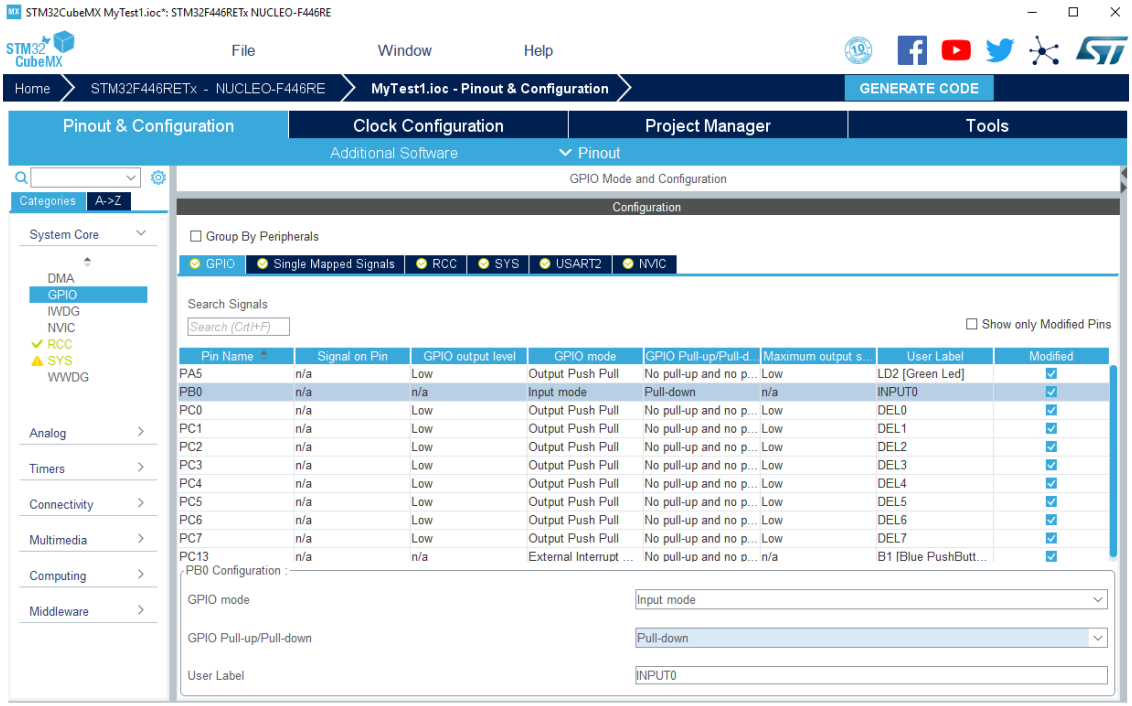
1. D'abord, il faut configurer le port PB0 en entrée. Pour y arriver, retournez dans STM32Cube, configurez le port PB0 en tant que GPIO_Input et renommez le INPUT0.



2. Allez dans l'onglet Configuration et cliquez sur le bouton GPIO :



3. Ajoutez un pull down sur l'entrée PB0. Cela ajoutera (à l'intérieur du microcontrôleur) une résistance entre PB0 et la masse. Donc, par défaut, cette entrée sera toujours à la masse.



4. Clique sur OK et régénérez le HAL dans le menu Project/Generate Code

5. Retournez dans Keil et essayez le code suivant pour la boucle infinie :

```
int count = 0;
while (1)
{
      while (HAL_GPIO_ReadPin(INPUT0_GPIO_Port, INPUT0_Pin)
            == GPIO_PIN_RESET);
      GPIOC->ODR = count;
      count = count+1;
      if (count == 16) count = 0;
      HAL_Delay(1000);
}
```

6. Appuyez sur F7 (compiler) ensuite F8 (téléverser). Appuyez sur le bouton noir (RESET) et observez … rien ne devrait se passer.
7. Connectez maintenant un fil entre le 3.3V et PB0 et observez si le compteur commence à compter.

Expérience 4

Utilisez le compteur pour compter le nombre de fois qu'on pèse sur un bouton.

Essayez le code suivant, qui attend d'abord qu'on pèse sur un bouton, incrémente, et attend ensuite qu'on arrête de peser sur le bouton. Comme dans l'exemple précédant, on remplacera le bouton par un simple fil qui fait un court-circuit entre le 3.3V et l'entrée PB0.

```
int count = 0;
while (1)
{
      while (HAL_GPIO_ReadPin(INPUT0_GPIO_Port, INPUT0_Pin)
            == GPIO_PIN_RESET);
      GPIOC->ODR = count;
      count = count+1;
      if (count == 16) count = 0;
      while (HAL_GPIO_ReadPin(INPUT0_GPIO_Port, INPUT0_Pin)
            == GPIO_PIN_SET);
}
```

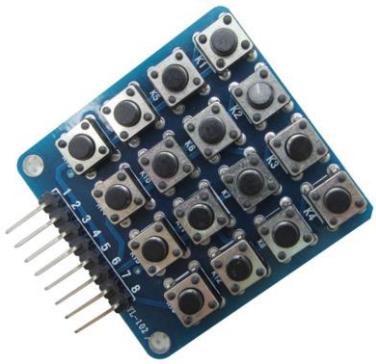
Vous devriez remarquer que le compteur semble sauter des états, voire faire n'importe quoi. En fait, lorsqu'on fait contact avec un fil ou un interrupteur, on a un phénomène de rebond qui fait que le contact se ferme et s'ouvre plusieurs fois sans qu'on s'en rende compte. Une manière simple de contourner le problème est d'attendre un petit délai après chaque début de transition. Essayez le code suivant :

```
int count = 0;
while (1)
{
      while (HAL_GPIO_ReadPin(INPUT0_GPIO_Port, INPUT0_Pin)
            == GPIO_PIN_RESET);
      HAL_Delay(20);
      GPIOC->ODR = count;
      count = count+1;
```

```
        if (count == 16) count = 0;
        while (HAL_GPIO_ReadPin(INPUT0_GPIO_Port, INPUT0_Pin)
              == GPIO_PIN_SET);
        HAL_Delay(20);
}
```

2ème partie : Laboratoire en salle à Polytechnique

Un énoncé parmi les suivants sera assigné aléatoirement à chaque équipe. Ces questions font appel au clavier :



Comme on peut l'apercevoir sur le circuit imprimé, les broches 5, 6, 7 et 8 correspondent aux colonnes (de gauche à droite) alors que les broches 1, 2, 3 et 4 correspondent aux rangées (de haut en bas). Lorsqu'un bouton est pesé, un court-circuit survient entre la colonne et la rangée concernées. Dans les énoncés ci-dessous, on va connecter la broche 1 au +3.3V et les broches 5 à 8 aux ports PC0…3 (qui eux seront configurés avec un pull-down). Donc, au repos, les ports PC0…3 retournent le signal '0' tandis que dès qu'on pèse sur une touche de la rangée 1, le port PCx correspondant vaut '1'.

Si vous n'avez pas de clavier, utilisez simplement 4 fils. Chaque fil aura une extrémité connectée à une broche d'entrée. L'autre extrémité sera flottante. Pour simuler une pression de touche, connecter l'extrémité flottante au +3.3V.

Énoncé 1

Écrivez un programme qui valide un code à 4 chiffres préenregistré (donné par le chargé de lab juste avant l'évaluation). Lorsque le code est correct, les quatre DELs connectées aux ports PC4…7 s'allument en séquence (changement de DEL chaque 500ms) pendant dix secondes. Si on attend plus que 5 secondes entre deux chiffres, le système revient à son point de départ.

Suggestion : vous pourriez avoir besoin de la fonction HAL_GetTick() (voir la documentation sur le HAL).

Énoncé 2

Écrivez un programme qui réalise un mini piano électronique. Lorsqu'on pèse sur une touche, le système envoie une onde carrée sur le port PC4 et son inverse sur le port PC5. Chaque touche aura une période différente (24 ms, 12 ms, 8 ms et 6 ms). Connectez ensuite les canaux gauche et droit du casque écouteur sur les sorties PC4 et PC5 (sans connecter la masse). Ensuite, raffinez le contrôle du temps pour obtenir des fréquences plus représentatives des notes de musique, par exemple : 293.665 Hz (ré), 329.628 Hz (mi), 369.994 Hz (fa#) et 440 Hz (la). Pour y parvenir, vous devrez inventer votre propre fonction de délai pour chaque note. Par exemple, vous pouvez utiliser une boucle *for* dont le nombre d'itérations correspond à une seconde. Ensuite, vous réduisez le nombre d'itérations pour obtenir les délais demandés.

Énoncé 3

Écrivez un programme qui envoie des messages en morse sur les 4 DELs en même temps. Un point dure 100ms. Une barre dure 500ms. Le délai entre deux symboles d'un même caractère est de 100ms. Le délai entre deux caractères est de 700ms. Quatre messages (un par bouton) sont pré-encodés : 'HELLO', 'BYE', 'WAIT' et 'SOS'. De l'information sur le Morse est disponible ici :

https://fr.wikipedia.org/wiki/Code_Morse_international