Référence/type[1]	LOO C++ TD4		☒ TD ☐ AP
Thématique principale	Développement C++ embarqué avec MBED-OS		
Thématique(s) secondaire(s)			
Durée	2h	***Niveau***[2]	A
Prérequis	✓ Cours et activités précédentes de C++		
Compétences opérationnelles et niveau cible[3]	✓ Mettre en place un projet C++ avec un environnement de développement MBED-OS, qu'il soit local ou en ligne (***M***) ✓ Construire une application exploitant quelques fonctionnalités du RTOS MBED-OS (***A***)		
Modalités et critères d'évaluation	Auto-évaluation		
Matériel(s), équipement(s), composant(s) nécessaires	Carte cible compatible MBED (LPC1768, Nucleo…) Grove-LED Button		
Logiciel(s) nécessaire(s)[4]	Environnement de développement MBED (en ligne ou local)		
Ressources	https://os.mbed.com/		

Avant la séance…

Prendre connaissance (lire avec attention) le diaporama « o6 - Intro MBED », préparer d'éventuelles questions sur le sujet.

Mettre en place un environnement de développement MBED-OS (https://os.mbed.com/mbed-os/).

Remarque : compte-tenu de la fin annoncée du projet MBED par ARM, il est conseillé de travailler plutôt avec l'IDE en ligne (ARM Keil Studio Cloud, https://studio.keil.arm.com/auth/login) plutôt que d'installer en local MBED-Studio.

[1] TD : ½ groupe, apports de cours, mise en pratique guidée – TP : ½ groupe, travail autonome majoritaire.

[2] Le niveau se rapporte à la thématique principale. Il peut ici être entendu comme un indicateur de la difficulté du travail (N-Facile, A-Sans grande difficulté, M-Quelques points complexes, MA-Difficile)

[3] Les niveaux cible correspondent pour chaque compétence opérationnelle au niveau d'acquisition de la celle-ci à l'issue du travail dans son intégralité (en incluant les phases préparatoires et de synthèse).

[4] En plus d'un environnement de développement C++

Travail encadré en séance

Apports de cours

- ✓ Diaporama « 06 - Intro MBED » - Revue rapide, réponse aux questions.

Activités

Activité 1 – Prise en main de l'environnement et des objets associés aux GPIO

- ➢ Récupérer et connecter à la machine de développement une carte cible. Celle-ci peut-être, au choix :
  - Une carte MBED-LPC1768
  - Une carte Nucleo-L073RZ
  - Une carte Nucleo-L476RG
- ➢ Démarrer MBED-Studio en local ou se connecter à Keil Studio Cloud en ligne. Réaliser les procédures d'identification. Vérifier que la cible soit correctement détectée par l'IDE.
- ➢ Créer un nouveau projet de type « Mbed Project ». Choisir comme exemple « mbed-os-example-blinky ».
- ➢ Produire l'exécutable pour la cible choisie, téléverser le binaire dans le MCU. Vérifier le bon fonctionnement (une LED de la cible doit clignoter toutes les 500ms).
  - Identifier les différents moyens mis à disposition par la plateforme MBED pour permettre la programmation d'une cible (téléversement du binaire).
- ➢ Analyser le code de l'exemple. Identifier les classes/objets utilisés.
- ➢ A l'aide de la documentation de MBED-OS (https://os.mbed.com/docs/mbed-os/v6.16/introduction/index.html) identifier les méthodes les plus intéressantes de l'API de la classe *DigitalOut*.
- ➢ Identifier les autres classes MBED permettant de gérer les GPIO. Indiquer les principaux cas d'utilisation de celles-ci.

Activité 2 – Identification des ressources prises en charge et du nommage des broches

Il est toujours, quel que soit le RTOS utilisé, d'identifier les ressources effectivement prises en charge par ce dernier ainsi que la convention de nommage des broches. Outre la recherche dans les diverses pages de documentation, pas toujours couronnée de succès, il existe une solution toujours fonctionnelle, bien que parfois peu ergonomique : aller trouver cette information dans le code.

La plupart des RTOS travaillent selon une philosophie « board », c'est-à-dire que le point d'entrée, du point de vue logiciel est la carte sur laquelle est installée la cible MCU. Pour chacune des cartes cibles un ou plusieurs fichiers de définition (.h) existent, c'est dans ces fichiers qu'il faut aller chercher.

Depuis la page de démarrage (https://os.mbed.com/) il est possible d'accéder à la liste des cartes supportées (Hardware → Boards). Une fois la recherche de la carte effectuée, toutes les informations associées sont accessibles.

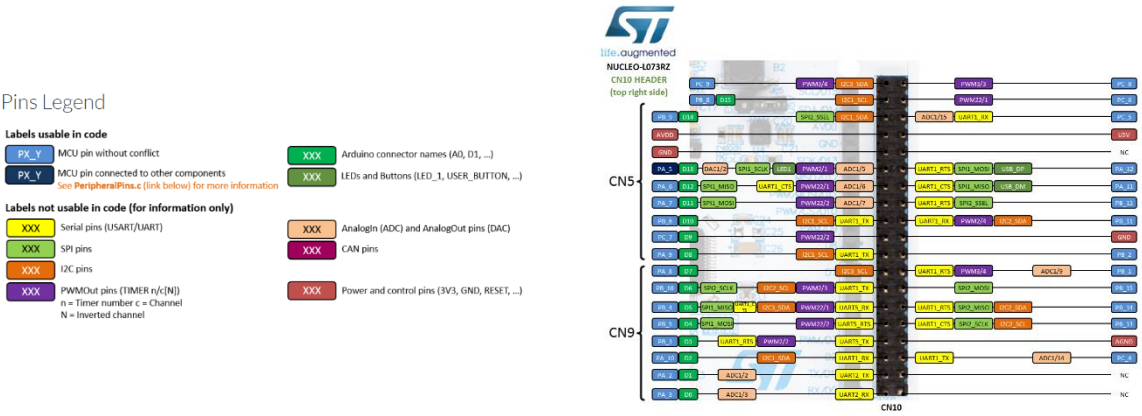
Cas de la carte Nucleo-L073RZ

Page d'entrée de la carte : https://os.mbed.com/platforms/ST-Nucleo-L073RZ/

On trouve sur cette page :

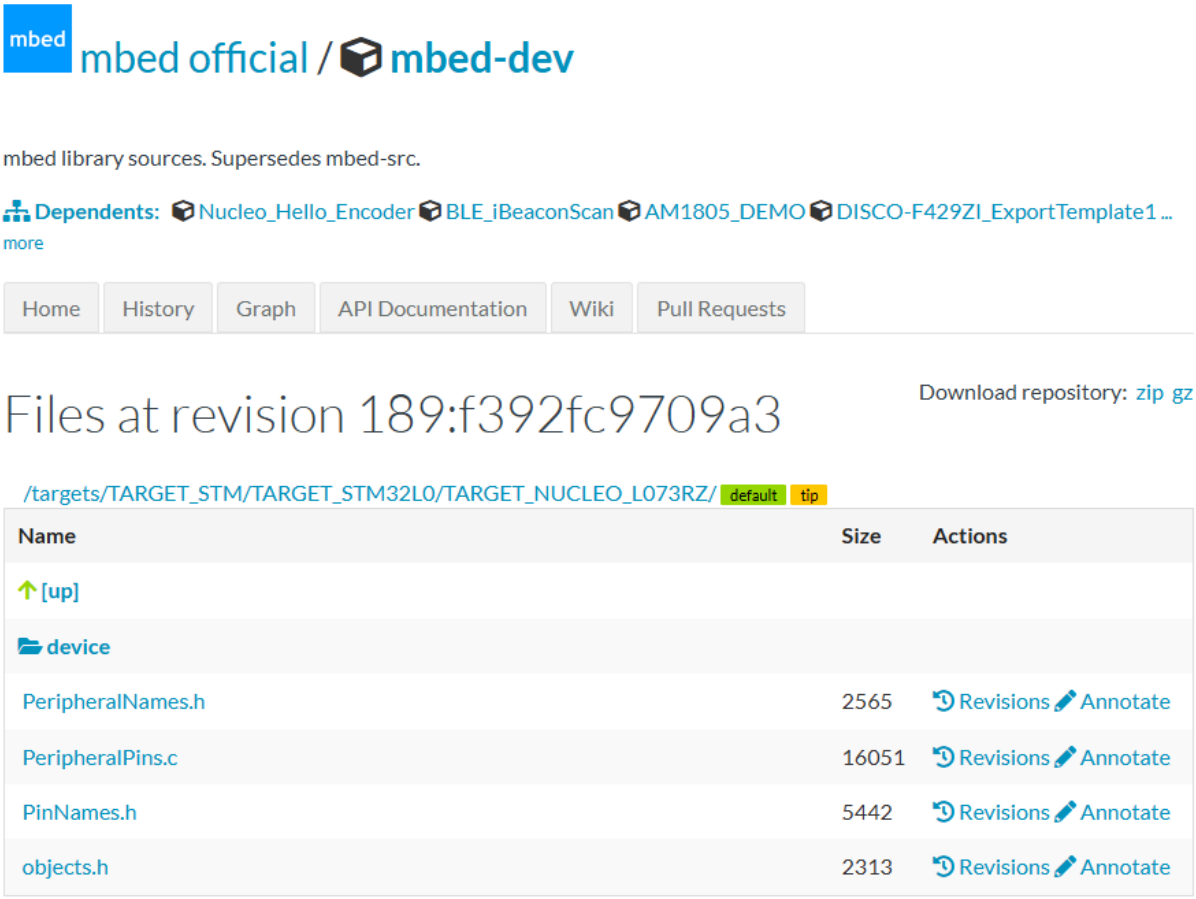
- ✓ Un aperçu de la carte (Overview)
- ✓ Le résumé des caractéristiques du MCU embarqué (Microcontroller features)
- ✓ Le résumé des caractéristiques de la carte (Nucleo features)

- ✓ Le câblage des connecteurs (Board pinout)
  - Avec un code couleur selon la ou les fonctions de la broche



On trouve aussi, et c'est cela qui va nous intéresser maintenant des liens vers la définition de la carte :

- ☞ https://developer.mbed.org/users/mbed_official/code/mbed-dev/file/default/targets/TARGET_STM/TARGET_STM32L0/TARGET_NUCLEO_L073RZ/



Les fichiers les plus intéressants sont ici :

- ✓ PeripheralNames.h
- ✓ PinNames.h

Dans ce second fichier on (PinNames.h) on trouve notamment :

- ✓ La ou les conventions de nommage des broches. On voit ici que les lignes sont identifiées P[*port*]_[*pin*], par exemple :
  - La ligne PAo est identifiée PA_o
  - La ligne PC6 est identifiée PC_6
- ✓ Mais aussi qu'il y a des alias définis pour la connectique Arduino :
  - Do = PA_3
  - …
- ✓ Enfin, d'autres alias pour des broches et/ou fonctions particulières :
  - LEDo = PA_5 : permet d'accéder directement avec le nom « LEDo » une led embarquée sur la carte
  - USER_BUTTON = PC13 : fait la même chose pour le BP
  - I2C_SCL, I2C_SDA identifient deux lignes pour un port I²C
  - …

En ce qui concerne le fichier « PeripheralNames.h », la principale information donnée concerne les ressources internes du MCU supportées par le RTOS. En ce qui concerne la carte Nucleo-L073RZ, les ressources supportées sont :

- ✓ Le convertisseur analogique numérique ADC1
- ✓ Le convertisseur numérique analogique DAC1
- ✓ Cinq UART : UART1, UART2, UART4, UART5 et LPUART1
- ✓ Deux contrôleurs SPI : SPI1 et SPI2
- ✓ Trois contrôleurs I²C : I2C1, I2C2 et I2C3
- ✓ Quatre sorties PWM : PWM2, PWM3, PWM21 et PWM22

Remarque : il faut bien intégrer le fait que toutes les ressources d'un MCU ne sont pas forcément supportées par un RTOS (ou une version du RTOS).

Cas de la carte mbed LPC1768

Page d'entrée de la carte : https://os.mbed.com/platforms/mbed-LPC1768/

On remarque que l'on retrouve sensiblement les mêmes informations que pour la cible précédente.

Il n'y a cependant pas de lien direct vers les fichiers de définition. Il faudra effectuer la recherche par nous même dans la codebase MBED. On peut pour cela utiliser, par exemple le dépôt github : https://github.com/ARMmbed/mbed-os

Après quelques fouilles, on comprend l'architecture du code et on arrive sur le répertoire associé à la cible :

- ☞ https://github.com/ARMmbed/mbed-os/tree/master/targets/TARGET_NXP/TARGET_LPC176X/TARGET_MBED_LPC1768

Le fichier « PinNames.h » existe, il nous indique que :

- ✓ La règle de nommage est inspirée du brochage du module : p5, p6…
- ✓ Les LEDs sont « aliasées » (LED1, LED2…)
- ✓ Quelques lignes I2C sont aussi « aliasées »

On remarque, dans le répertoire parent un autre fichier « PeripheralNames.h ». Celui-ci indique les ressources prises en charge par le RTOS sur cette cible.

Remarque : On s'aperçoit que selon la cible, les fichiers ne sont pas exactement nommés de manière identique. Il faut être capable de retrouver les informations utiles malgré ça.

Activité 3 – Multithreading avec MBED-OS

En s'appuyant sur les informations données sur cette page : https://os.mbed.com/docs/mbed-os/v6.16/apis/thread.html, mettre en place une application architecturée autour de deux threads :

- ✓ Un thread fait clignoter la (une) led selon une période passée en paramètre (ms).
- ✓ Un thread émet toutes les secondes, sur la console, le message « Message thread wake-up xx times. ». Avec la valeur « xx » s'incrémentant à chaque réveil.

Activité 4 – Mise en œuvre GPIO in & out

Un module breakout Grove-LED Button(https://wiki.seeedstudio.com/Grove-LED_Button/) est à mettre en œuvre.

- ➢ Choisir une ligne disponible sur la connectique adaptée au pilotage d'une LED. Câbler la led du module sur cette ligne. Valider le fonctionnement en réexploitant l'exemple « blinky ».
- ➢ Faire le même travail pour le BP intégré au module. Valider le câblage/fonctionnement en mettant en place une recopie permanente de l'état du BP sur la LED.
- ➢ Mettre en place une application qui à chaque appui sur le BP change l'état de la led. On exploitera obligatoirement ici les interruptions (événement front descendant issu du BP).
- ➢ Observer le comportement, proposer une idée (sans se préoccuper du « comment on fait ») pour résoudre ce problème. Préciser la stratégie, identifier les problèmes fondamentaux à résoudre.
- ➢ Rechercher les outils MBED susceptibles de nous aider. Modéliser et mettre en place la solution. Valider.
- ➢ Bien préciser le rôle de la « EventQueue » ainsi que les contextes dans lesquels cet outil est très intéressant.

Synthèse

A partir de la documentation disponible sur le site de MBED (https://os.mbed.com/docs/mbed-os/v6.16/apis/index.html) identifier les différentes classes/objets permettant de :

- ✓ Gérer la synchronisation entre threads (protection des ressources, transmission de données…).
- ✓ Mettre en œuvre des composants I²C et SPI.
- ✓ Réaliser l'acquisition d'un signal analogique, produire un signal analogique.
- ✓ Générer un signal de type PWM.
- ✓ …

Après la séance…

Prendre connaissance du sujet de l'AP7, commencer à réfléchir au dispositif à développer.