Institut d'Optique Graduate School
Interfaçage Numérique

Interfaçage Numérique

Travaux Pratiques

Semestre 6

MBED et STM32

---

1 séance

Ce sujet est disponible au format électronique sur le site du LEnsE - https ://lense.institutoptique.fr/ dans la rubrique Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Intro MBED et STM32.

Séance 1 / MBED et Nucléo-STM32 (sans maquette ! !)

Objectifs de la séance

Cette première séance est consacrée à la découverte de la **programmation de systèmes embarqués** (ici des cartes **Nucléo** de *STMicroelectronics*, basées sur des microcontroleurs *STM32*) et la prise en main de l'interface de développement **Keil Studio Cloud** (et les bibliothèques MBED).

Etape 0 - 30 min Créer un compte MBED et tester un premier programme

Etape 1 - 45 min Piloter des sorties numériques - LED

Etape 2 - 45 min Acquérir des données numériques - Bouton-poussoirs

Etape 3 - 45 min Mettre en œuvre des interruptions sur des événements externes

Etape 4 - 45 min Utiliser des sorties modulées en largeur d'impulsion (PWM) - LEDs

Etape 5 - 60 min Acquérir des données analogiques - Potentiomètre

IDE Keil Studio Cloud et MBED

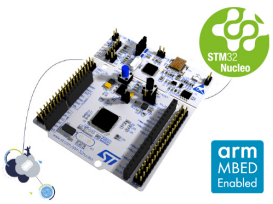
Mbed-OS est un système d'exploitation (*Operating System*) pour des **systèmes embarqués**, conçu pour des **microcontroleurs de type ARM**. Cet ensemble de bibliothèques permet une abstraction des couches matérielles de bas niveau du microcontroleur. https://os.mbed.com/

La division **Keil** de Arm fourni un environnement de développement en ligne, nommé **Keil Studio Cloud**, permettant l'utilisation de l'OS pour des projets basés sur des microcontroleurs STM32

https://www.keil.arm.com/

Carte Nucleo-STM32

Les cartes Nucleo sont des **plateformes de développement** basées sur les **microcontrôleurs STM32** de *STMicroelectronics*. Elles sont conçues pour faciliter le prototypage et le développement de projets embarqués, similaires aux cartes Arduino, mais elles sont souvent utilisées pour des applications plus complexes et performantes.

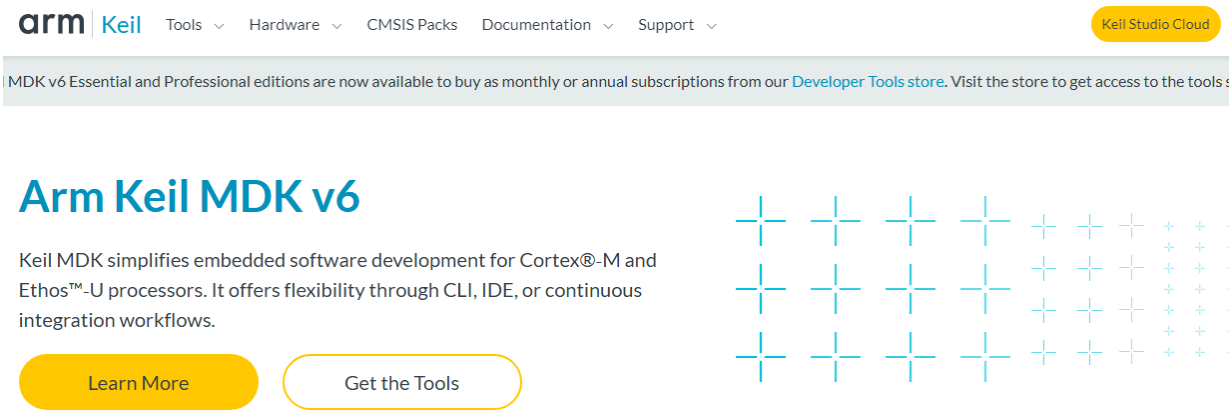


Elles sont équipées d'un débogueur ST-LINK intégré, ce qui permet de programmer et de déboguer le microcontrôleur directement sans matériel additionnel.

Le brochage de la carte Nucleo L476RG est fournie en annexe à ce document : Brochage Nucléo L476RG

Etape 0a / Création d'un compte MBED

L'interface de développement **Keil Studio Cloud**, ainsi que les bibliothèques associées, est accessible depuis le site https://www.keil.arm.com/ moyennant une inscription préalable (gratuite).



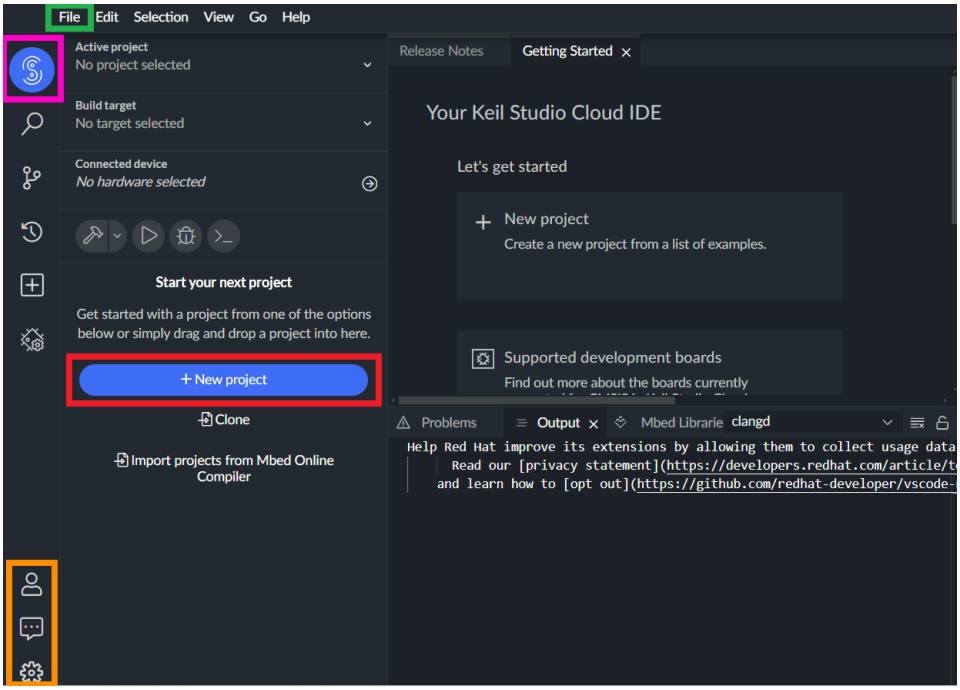
Sur la page principal du site de Keil.Arm, cliquer sur l'icône **KEIL STUDIO CLOUD** (en haut à droite, sur fond jaune).

Dans la nouvelle fenêtre *Log In | Arm Keil Studio*, cliquer sur **SIGN UP**.

Sur la page suivante *Local Registration*, cliquer sur **REGISTER** et saisir une adresse email valide.

Suivre les différentes étapes pour valider votre inscription. Un **email de confirmation** vous sera envoyé sur l'adresse indiquée dans le formulaire.

Après validation de votre adresse email, vous pourrez retourner sur la page de KEIL STUDIO CLOUD et saisir votre identifiant et votre mot de passe pour **accéder à l'interface du logiciel de programmation**.



*Vous trouverez également des ressources concernant les **microcontrôleurs** et les **systèmes embarqués** à l'adresse suivante :*

https://iogs-lense-training.github.io/nucleo-basics/contents/general.html

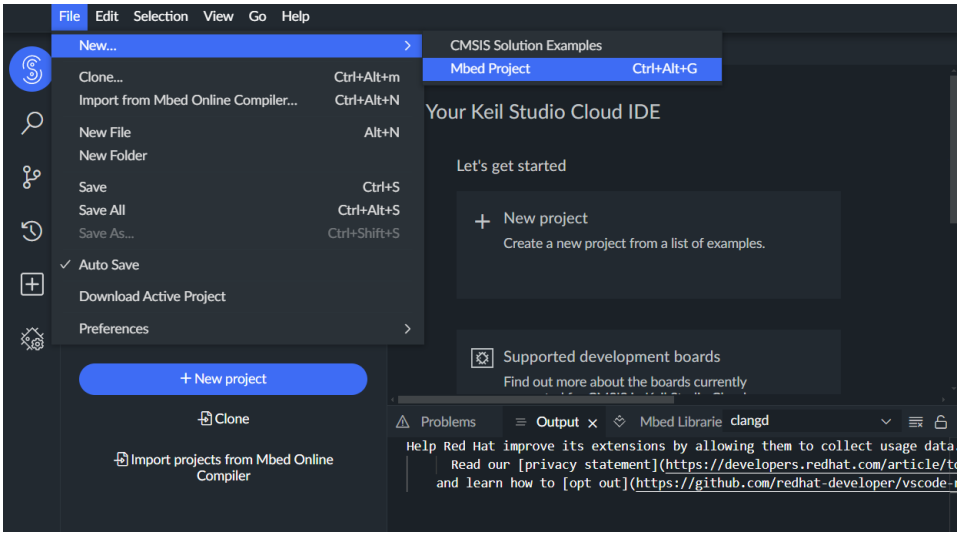
Etape 0b / Tester un premier programme

Afin de vérifier que toute la chaîne de prototypage est opérationnelle, nous allons nous intéresser à un **programme de base** permettant de faire **clignoter une LED** présente par défaut sur la carte Nucléo.

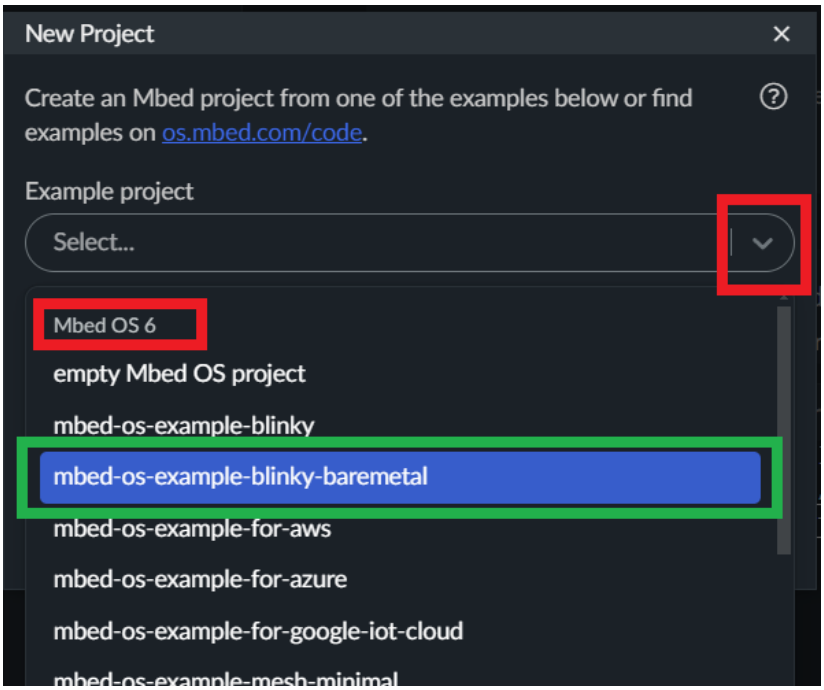
Créer un nouveau projet

Avant de pouvoir programmer le microcontroleur présent sur les cartes Nucléo que nous allons utiliser, il faut au préalable **créer un projet** dans l'interface *Keil Studio Cloud*.

Pour créer un nouveau projet, cliquer sur **File -> New... -> MBED Project**.



Sélectionner ensuite le projet ***mbed-os-example-blinky-baremetal*** dans MBED OS 6.



Donner un nom à votre projet dans la section ***Program Name***.
Décocher la case *Initialize this project as a Git repository* et cliquer sur **Add project**.

Programme Blinky (baremetal)

Le code principal se trouver dans le fichier ***main.cpp***, que vous pouvez retrouver dans la zone de droite de l'interface.

Le programme ***main.cpp*** ressemble à celui-ci :

```
1  #include "mbed.h"
2
3  #define WAIT_TIME_MS 500
4  DigitalOut led1(LED1);
5
6  int main()
7  {
8      printf("This␣is␣the␣bare␣metal␣blinky␣example␣running␣on␣Mbed␣OS␣%d.%d.%d.\n", MBE
9
10     while (true)
11     {
12         led1 = !led1;
13         thread_sleep_for(WAIT_TIME_MS);
14     }
15 }
```

Sur la carte Nucléo la sortie LED1 correspond à la LED nommée LD2.

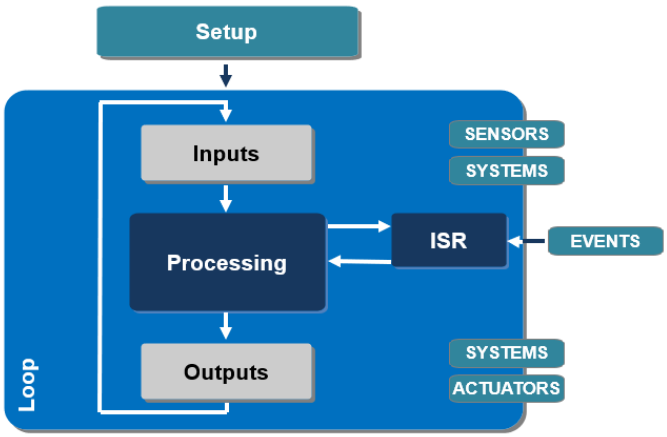
Le langage utilisé par l'environnement MBED est un langage proche du C++. Le programme ainsi écrit doit nécessairement **être compilé** avant de pouvoir **être téléversé** sur la carte où il sera ensuite **exécuté**.

Structure du code

Un programme embarqué est constitué de **deux étapes principales** :

— une **initialisation** : exécutée une fois à la mise sous tension de la carte ou lors de l'appui sur le bouton Reset.

— une **boucle infinie** : exécutée de manière infinie. Cette boucle a pour principale mission, sur un système embarqué, de récupérer les valeurs des entrées, de calculer les valeurs des sorties et de mettre à jour les sorties.

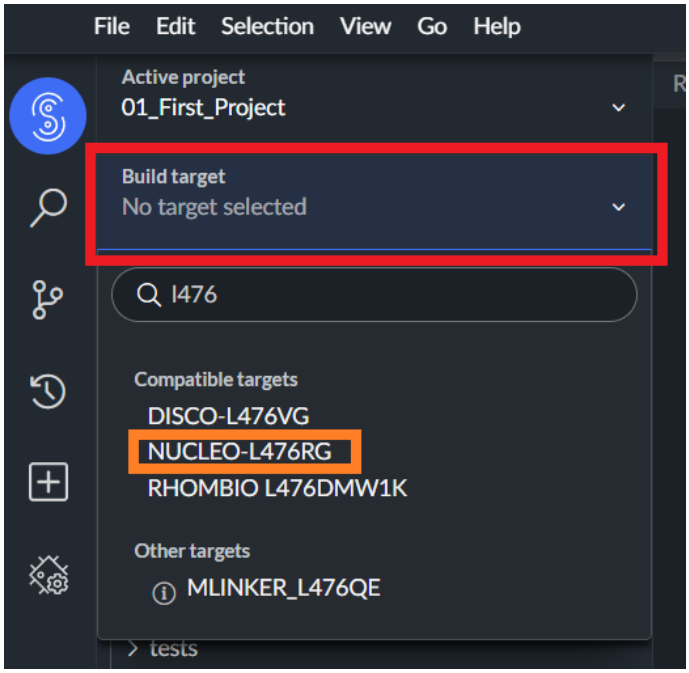


D'autres étapes sont possibles lorsqu'on autorise le fonctionnement par interruption (voir dans la suite de ce document).

Choix d'une carte

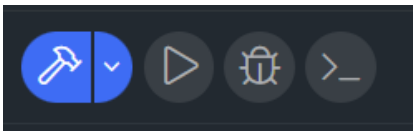
La compilation d'un tel programme se fait pour une cible particulière. Avant de pouvoir compiler, il est donc nécessaire de préciser sur quel microcontrôleur (ou quelle carte de prototypage) ce code sera exécuté.

Sélectionner la carte **NUCLEO-L476RG** dans le gestionnaire de projet, sous-partie *Build target*.



Compilation

Une fois la cible choisie, il est maintenant possible de compiler le programme. Pour cela, cliquer sur la première icône de la barre d'action :



A la fin de cette étape, si la compilation s'est terminée avec succès, un fichier binaire est téléchargé.

Téléversement du programme

La carte doit être connectée à l'ordinateur via le port USB adéquat. Les cartes Nucléo sont reconnues comme des supports de stockage amovible.

Il suffit alors de déplacer le fichier binaire obtenu après la compilation dans le support de stockage.

Attention ! Le fichier va automatiquement disparaître du stockage. Il sera cependant bien téléverser dans la carte Nucléo (LED clignotante au niveau du port USB de la carte Nucléo).

Attention ! Il est parfois nécessaire de cliquer sur le bouton *Reset* (bouton noir) sur la carte Nucléo pour faire démarrer le programme. Et parfois débrancher et rebrancher le câble USB...

Etape 1 / Piloter des sorties numériques - LED

Afin de pouvoir interagir avec le monde extérieur, les microcontrôleurs disposent d'un ensemble d'**entrées** et de **sorties**.

Chacune de ces entrées-sorties portent un nom, au format PX_N, où X est le nom du port (A, B...) et N le numéro de la broche.

Exemple de la broche PA_7 (port A, broche 7)

Choix d'une broche

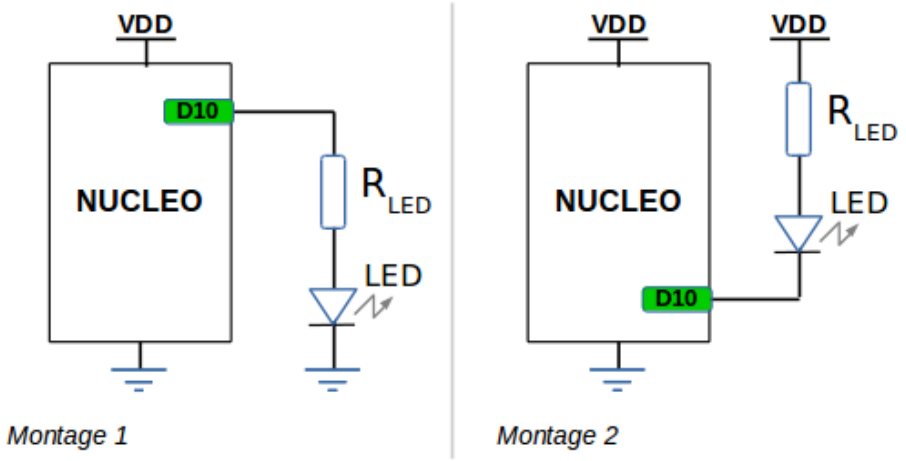
Toutes les broches peuvent être utilisées en **entrée** ou en **sortie numérique**, c'est à dire un type de signal qui ne peut prendre que **deux états** : haut ou bas (aussi appelés 1 ou 0, ou encore *HIGH* et *LOW* en Arduino).

Certaines broches ont également d'autres fonctionnalités : entrées analogiques, sorties modulées PWM, communication série...

Câblage d'une LED

Nous allons voir ici comment connecter une LED à la carte Nucléo sur la broche D10 par exemple de la carte (ou PB_6 - port B, broche 6).

Les schémas de câblage possibles sont les suivants :



Exemple de la broche D10 reliée à une LED (ou PB_6 - port B, broche 6)

Dans le cas des cartes Nucléo, la tension VDD est égale à $3.3\,\mathrm{V}$.

Il est indispensable d'associer la LED à une **résistance de protection**, permettant de limiter le courant :

$$R_{LED} > \frac{VDD - V_F}{I_{Fmax}}$$

Les valeurs V_F et I_{Fmax} dépendent de la LED utilisée et sont à chercher dans sa documentation technique.

Programme

Paramétrage

Pour **configurer une broche en sortie**, il faut ajouter l'instruction suivante **en dehors de toute fonction** (où *D10* est le nom d'une broche du composant) :

```
1  DigitalOut led2(PB_6);
```

led2 correspond au nom de la variable assignée à la broche D10.

Utilisation

Pour **affecter une valeur à une broche en sortie**, il faut utiliser une des deux instructions suivantes (où *led2* est le nom de la variable associée à la broche du composant) selon que l'on veut mettre la sortie à l'état bas (*0*) ou à l'état haut (*1*) :

```
1  led2 = 0;
2  led2 = 1;
```

Travail à réaliser

➜ **M** Créer un nouveau projet basé sur l'exemple ***mbed-os-example-blinky-baremetal*** dans MBED OS 6.

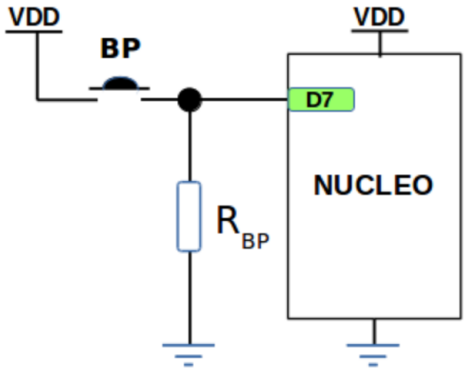
➜ **M** Réaliser un programme qui permet d'allumer deux LEDs rouges cablées sur les sorties D10 et D11. Les LEDs devront s'allumer de manière complémentaire chaque seconde.

Etape 2 / Acquérir des données numériques - Bouton-poussoirs

Câblage d'un bouton-poussoir

Nous allons voir ici comment brancher un bouton-poussoir à la carte Nucléo sur la broche D7 par exemple de la carte (ou PA8 - port A, broche 8).

Le schéma de câblage est le suivant :



Exemple de la broche D7 reliée à un bouton-poussoir (ou PA_8 - port A, broche 8)

Avant de câbler la sortie du bouton-poussoir à l'entrée de la carte, vérifier que vous avez bien une tension de $0\,\mathrm{V}$ (lorsque le bouton-poussoir n'est pas activé) et une tension de $3.3\,\mathrm{V}$ (dans l'autre cas).

Programme

Paramétrage

Pour **configurer une broche en entrée**, il faut ajouter l'instruction suivante **en dehors de toute fonction** :

```
1  DigitalIn  sw1(PA_8);
```

Utilisation

Pour **récupérer une valeur d'une entrée**, il faut utiliser l'instruction suivante :

```
1  int bpVal = sw1;
```

La variable *bpVal* peut valoir 1 ou 0 selon l'état du bouton-poussoir (ou de l'entrée).

Travail à réaliser

➜ **M** Créer un nouveau projet basé sur l'exemple ***mbed-os-example-blinky-baremetal*** dans MBED OS 6.

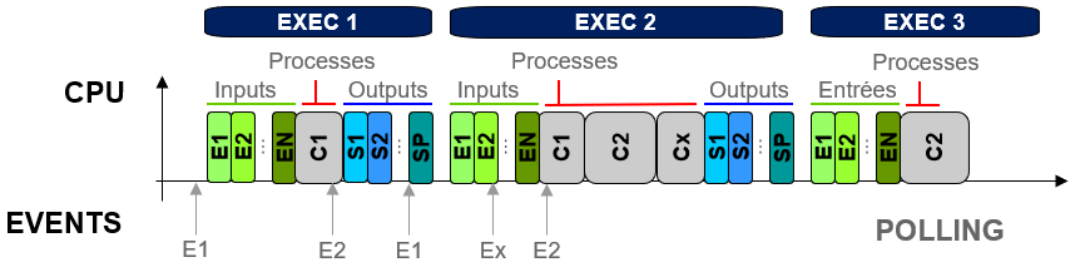
➜ **M** Réaliser un programme qui permet d'allumer chacune des LEDs rouges précédentes à l'aide de deux bouton-poussoirs cablés sur les entrées D6 et D7 (un bouton-poussoir par LED).

Etape 3 / Mettre en œuvre des interruptions sur des événements externes

Scrutation

La **scrutation** (ou *polling* en anglais) est une méthode de **vérification régulière de l'état des périphériques ou des capteurs** dans un système embarqué pour détecter si un événement spécifique s'est produit.

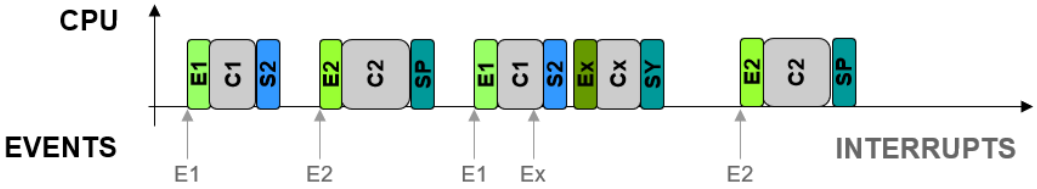
Dans ce contexte, le programme principal exécute une boucle continue (fonction *loop()* sous Arduino) où il interroge périodiquement chaque périphérique pour voir s'il nécessite une action.



La **scrutation** ne permet pas de gérer d'un événement rapidement et monopolise le microcontrôleur pour vérifier constamment l'état des capteurs ou des périphériques.

Interruptions

Une **interruption** est un signal envoyé au microcontrôleur pour lui demander d'**arrêter temporairement l'exécution de son programme principal** et de **s'occuper d'une tâche prioritaire spécifique**. Lorsque l'interruption est terminée, le microcontrôleur reprend son programme là où il s'était arrêté.



Cette méthode permet au système de **réagir rapidement à des événements externes** - changement de signal sur une broche, compteur qui atteint une certaine valeur - sans avoir besoin de vérifier constamment si l'événement a eu lieu, contrairement à la scrutation.

Pour en savoir plus sur les interruptions, vous pouvez consulter le document à l'adresse suivante :
https://iogs-lense-training.github.io/nucleo-basics/contents/polling_interrupts_rtos.html

Les interruptions sont également souvent utilisées dans les applications où un timing précis est nécessaire (par exemple, un compteur de temps).

Paramétrage

Pour **configurer une broche en entrée permettant une interruption**, il faut ajouter l'instruction suivante **en dehors de toute fonction** :

```
1  DigitalIn    sw1(PA_8);
```

Il existe 2 modes possibles pour les interruptions sur des signaux externes (entrées numériques) :
— **rise** - front montant d'un signal
— **fall** - front descendant d'un signal

Il faut ensuite paramétrer l'action à réaliser lors d'un événements sur cette entrée.
Pour cela, il faut ajouter l'instruction suivante au début de la fonction *main()* (phase d'initialisation) :

```
1  sw1.rise(&sw_ISR);
```

où *sw_ISR()* est une fonction, appelée **routine d'interruption**.

Routine d'interruption

Une **routine d'interruption**, aussi appelée **ISR** (*Interrupt Service Routine*), est la fonction qui s'exécute automatiquement en réponse à une interruption.

C'est une fonction classique, mais qui ne retourne pas de donnée.

```
1  void sw_ISR(void){
2    bool led_state = led1;
3    led1 = !led_state;
4  }
```

Dans cet exemple, à chaque front descendant sur l'entrée *PA_8*, la fonction *sw_ISR()* est appelée. Elle lit alors l'état de la sortie *led1* pour ensuite l'inverser.

Travail à réaliser

On se propose de tester le code *interrupt_on_input.cpp*. Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Exemples MBED pour STM32*.

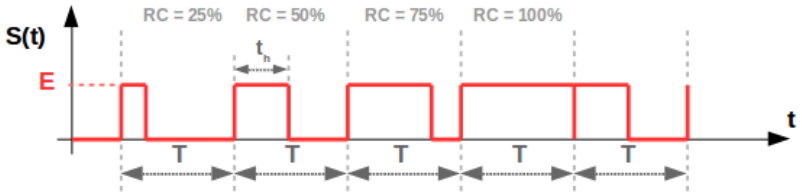
➜ **M** Tester le code fourni.

➜ **Q** A quel moment est appelée la fonction *sw_ISR()* ?

➜ **M** Réaliser un programme qui permet d'allumer à l'aide d'un premier interrupteur, câblé sur l'entrée D7, une LED câblée sur la sortie D10 et de l'éteindre à l'aide d'un second interrupteur câblé sur l'entrée D6. **Votre programme devra utiliser uniquement le principe d'interruption.**

Etape 4 / Utiliser des sorties modulées en largeur d'impulsion (PWM) - LEDs

La **modulation en largeur d'impulsions** (MLI ou *PWM – Pulsed Witdh Modulation* – en anglais) est une méthode permettant de générer un signal rectangulaire de **période** T **fixe** (ou de fréquence $1/T$ fixe) et dont le **rapport cyclique**, i.e. le rapport du temps haut t_h sur la période, noté $RC = \frac{t_h}{T}$, est variable.



Choix de la broche de sortie

Toutes les broches du microcontroleur ne sont pas utilisables comme sortie modulée en largeur d'impulsion. Cela nécessite l'utilisation de modules internes spécifiques (*timer* et comparateur notamment).

Il faut choisir une broche où le terme *PWM* est mentionné.

Exemple de la broche PA_7 (port A, broche 7), utilisable en PWM

Travail à réaliser

On se propose de tester le code *pwm_fade.cpp*. Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Exemples MBED pour STM32*.

➜ **M** Tester le code fourni.

➜ **M** Visualiser à l'aide d'un oscilloscope le signal électrique appliqué sur la LED.

➜ **Q** Expliquer le principe de fonctionnement de ce système.

➜ **M** Réaliser un programme qui permet de modifier la luminosité de la LED câblée sur la sortie D7 à l'aide des deux bouton-poussoirs, câblés sur D10 et D11 (un pour augmenter, l'autre pour diminuer la luminosité). **Votre programme devra utiliser uniquement le principe d'interruption.**

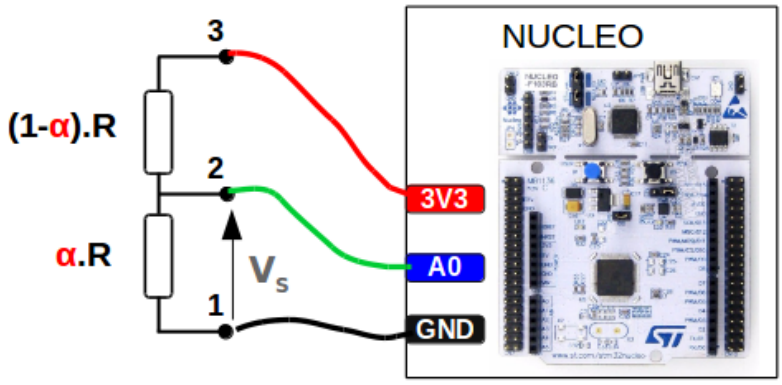
Etape 5 / Acquérir des données analogiques - Potentiomètre

Sur les systèmes numériques, et les microcontrôleurs en particulier, les broches sont naturellement des entrées/sorties numériques.

Or, la plupart des signaux qui nous entourent et que l'on cherche à mesurer (luminosité, température, vitesse...) sont des **grandeurs analogiques**. Pour palier à ce problème, la plupart des fabricants de microcontrôleurs ont intégré des **convertisseurs analogiques-numériques** (*ADC – Analog to Digital Converter*) à leur système, afin d'éviter de passer par des composants externes.

Câblage d'un potentiomètre

Afin de pouvoir générer une tension analogique dont la tension est variable, nous allons utiliser un potentiomètre.



Travail à réaliser

➜ **M** Câbler un potentiomètre de $10\,\mathrm{k\Omega}$ entre $0\,\mathrm{V}$ et $3.3\,\mathrm{V}$, sur ses broches 1 et 3. Vérifier que la broche 2 varie selon la position du curseur de $0\,\mathrm{V}$ à $3.3\,\mathrm{V}$.

On pourra utiliser la sortie $3.3\,\mathrm{V}$ *et la référence GND de la carte Nucléo pour alimenter le potentiomètre.*

On se propose de tester le code *analog_in.cpp*. Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Exemples MBED pour STM32*.

➜ **M** Tester le code fourni. Le potentiomètre devra être relié à l'entrée A0 de la carte Nucléo.

➜ **M** Utiliser le logiciel **TeraTerm** pour visualiser les messages transmis à la carte Nucléo via l'instruction *printf()*.

➜ **Q** Expliquer le fonctionnement de ce code.

➜ **M** Réaliser un programme qui permet de modifier la luminosité de la LED câblée sur la sortie D7 à l'aide de la valeur convertie du potentiomètre. Attention au type des variables... *int* ou *float* !

D'autres tutoriels sont disponibles à l'adresse suivante :
https ://lense.institutoptique.fr/nucleo/

Carte Nucléo-64 / STM32L476 / Broches d'entrées-sorties