

# Activité de rentrée 2023

– Partie programmation –



## Le mot du président

Bonjour à toi qui lit ce petit livret ! Cette activité de rentrée a pour but de te faire découvrir un peu les bases de la programmation sur microcontrôleur avec Mbed, afin de te familiariser un peu avec quelques concepts que tu pourrais rencontrer au cours de ta scolarité ou tes projets personnels. En fin de livret se trouve une introduction à STM32 Cube IDE, le logiciel que l'on utilise en cours à la place de ce qu'on va utiliser aujourd'hui, vous pourrez y jeter un œil après la séance si l'envie vous prend. Et ce n'est pas dramatique si vous ne comprenez pas tout ce qui y est expliqué, on n'est qu'au semestre 5 après tout, et vous n'allez le prendre en main qu'au semestre 6 ! L'ambition de ce livret est également de te faire découvrir petit à petit le local d'Ares et ce qu'on peut y faire en son sein, alors n'hésite pas à en faire le tour pendant ou après l'activité.

**Amusez-vous bien !**



# Au sommaire

Programmation Mbed – Prérequis & Conseils

4

5

Niveau 0 – Aide-mémoire

Niveau 1 – Clignotant

7

9

Niveau 2 – Tricolore

Niveau 3 – Jour, Nuit, Jour...

10

11

Niveau 4 – Jour, Nuit, version soft

Niveau 5 – 🎵 Joyeux anniversaire 🎵

12

14

Niveau 6 – A vous de jouer !

# Programmation Mbed – Prérequis & Conseils

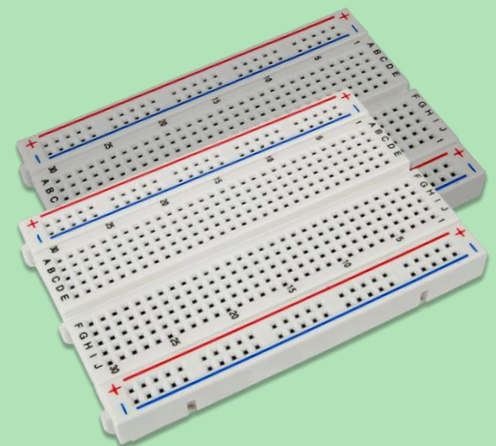
**ARM**  
mbed

Tout d'abord, rendez-vous sur le logiciel en ligne **Arm Keil Studio** [[studio.keil.arm.com](http://studio.keil.arm.com)]. Connectez-vous avec les identifiants suivants :  
Adresse email : ares.X@outlook.fr *[Remplacer le « X » par votre couleur de clan, par exemple « rouge », « noir », etc.]*  
Mot de passe : Aresensea95!

> Cliquez ici pour accéder à l'IDE <

## Déroulé de la séance de programmation

Chaque clan disposera d'une zone avec des composants et deux breadboards (plaques de prototypage). Vérifiez que vous êtes bien avec des membres de votre clan. On avisera si les groupes sont déséquilibrés en nombre. Vous pourrez faire la programmation sur votre ordinateur personnel ou sur ceux qu'on propose dans le local.



## Conseils [qui vont servir même plus tard !]

- Ne pas oublier les « ; » à la fin de chaque instruction
- Faire attention aux minuscules/MAJUSCULES dans les variables : rouge ≠ Rouge
- Vérifiez si le programme est bon avec la coloration syntaxique
- Regarder si la carte est bien détectée par le programme
- ~~- Ne pas frapper sa machine (ou son voisin) si ça ne fonctionne pas !~~

Et surtout, n'hésitez pas à demander de l'aide à un membre d'Ares ou votre voisin si vous êtes coincés ! Bonne séance !

# Niveau 0 – Aide-mémoire

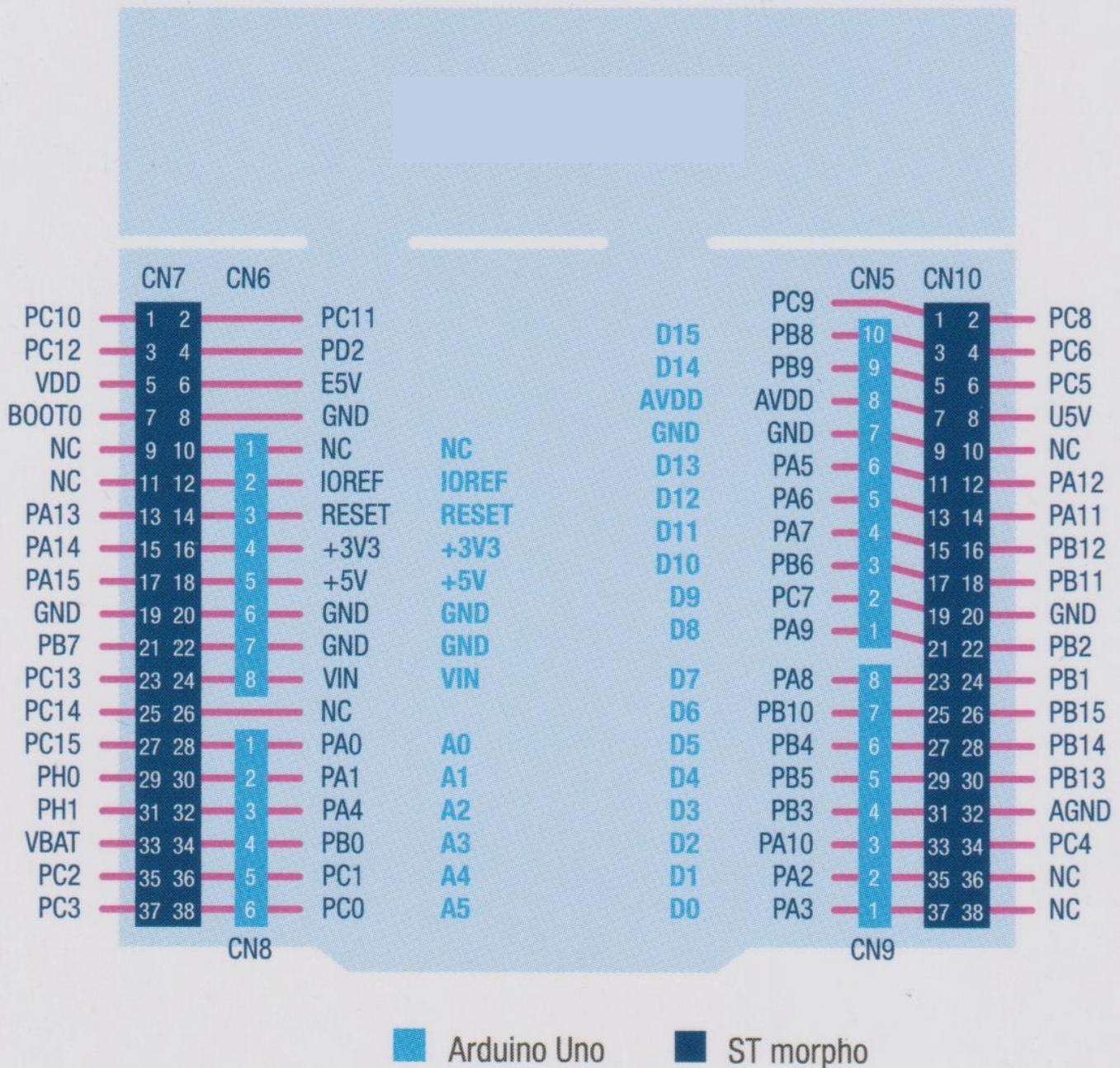
## Langage C++ et fonctions Mbed

Les variables		Les opérateurs	
<b>int</b> : entier naturel codage sur 4 octets [ $-2^{31}; 2^{31} - 1$ ]	<b>float</b> : nombre à virgule codage sur 4 octets	Arithmétiques : +, -, *, /, % Affectation : =, +=, -=, %=,  =, ^=, <=<=, >=>= Logique : &&,   , <, >, <=, >=, ==, !=	
<b>char</b> : caractère codage sur 1 octet [-128;127]	<b>double</b> : nombre à virgule codage sur 8 octets		
Condition		Boucles	
<pre>if (x &lt; 3) {     printf("x est inférieur à 3"); } else {     printf("x est supérieur ou égal à 3"); }</pre>		<pre>int x = 0; while (x &lt; 5) {     printf("coucou\n");     x++; }</pre>	<pre>int somme = 0; for (int i = 0; i &lt; 100; i++) {     somme += i; }</pre>
Directives de préprocesseur		Les tableaux	
<pre>//Inclusion d'une bibliothèque, ici mbed #include "mbed.h"  //Lors de la compilation, tous les N du code seront remplacés par des 7 #define N 7</pre>		<pre>#define SIZE 5 #define SIZE2 3  //Exemples de tableaux à 1 dimension int tab[SIZE] = {1, 9, 0, 1}; char tabchar[SIZE2] = {"a", "r", "e", "s"};  //Initialisation du premier tableau For (int i = 0; i &lt; SIZE; i++) {     tab[i] = 0; }</pre>	

Fonctions Mbed	
DigitalOut myDigitalInput(pin);	Définir le type d'une broche numérique (DigitalIn = entrée, DigitalOut = sortie).
myDigitalInput = 1;	Définir l'état d'une broche numérique (HIGH = 1, LOW = 0)
int state = myDigitalInput.read();	Lire l'état d'une broche.
ThisThread::sleep_for(1000ms);	Création d'une temporisation de 1000 ms = 1 seconde.
AnalogIn myAnalogInput(pin);	Définit le type d'entrée d'une broche analogique (A0, A1, ...)
PwmOut myPwmOutput(pin); myPwmOutput.write(dutyCycle);	Crée un signal PWM de rapport cyclique dutyCycle ∈ [0,1] (Pour info, PWM ≡ Modulation de Largeur d'Impulsion)
void tone(int frequency) { myPwmOutput.period(1.0 / frequency); myPwmOutput.write(0.5); }	Fonction permettant de jouer une fréquence « frequency » avec un buzzer. Après l'avoir écrite, on peut l'appeler en écrivant par exemple pour jouer la note DO : tone(262);
myPwmOutput.write(0.0);	Éteint un signal PWM (Pratique pour éteindre un buzzer !).
InterruptIn signal(pin);	Création d'une interruption.
signal.rise(&fonction);	Exécute la fonction « fonction » à la montée du signal (remplacer rise par fall pour la descente du signal).

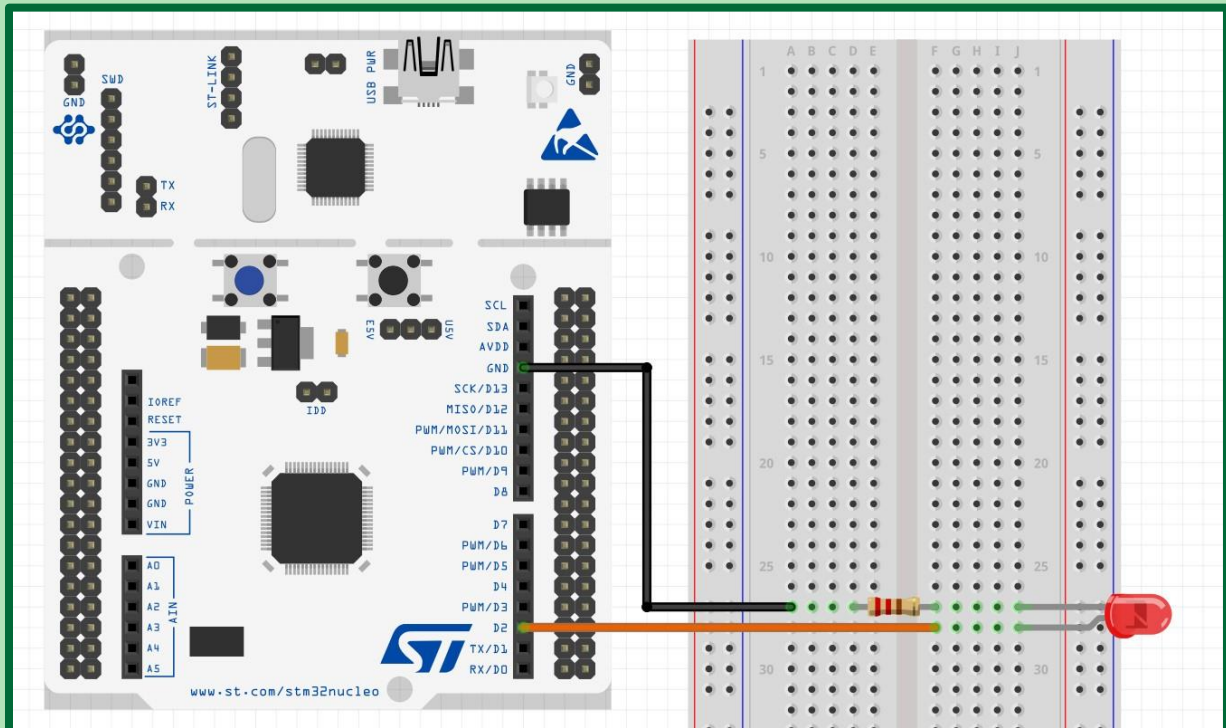


# Niveau 0 – Aide-mémoire : Pinout Configuration



# Niveau 1 – Clignotant

Les montages seront donnés pour tous les niveaux suivants sous cette forme :



Et comme à Ares nous sommes généreux, voilà en page suivante un code qui vous permettra d'allumer la LED et son explication :

La LED étant une sortie numérique, on définit la broche ledPin en tant que telle en précisant son type « DigitalOut ».

Après téléversement du code sur votre carte, tout le code compris en dehors de la boucle while [1] { } sera lancé une fois.

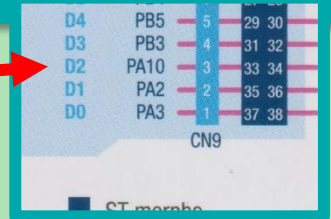
Après téléversement du code sur votre carte, tout le code compris dans le while [1] { } s'exécutera à l'infini !

```

1  #include "mbed.h"
2
3  DigitalOut ledPin(PA_10);
4
5  int main()
6  {
7      while (1) {
8          ledPin = 1;
9      }
10 }
11
12

```

La broche à laquelle est connectée la LED est « D2 ». Dans le pinout configuration de l'aide-mémoire, on voit que cela correspond à PA10.



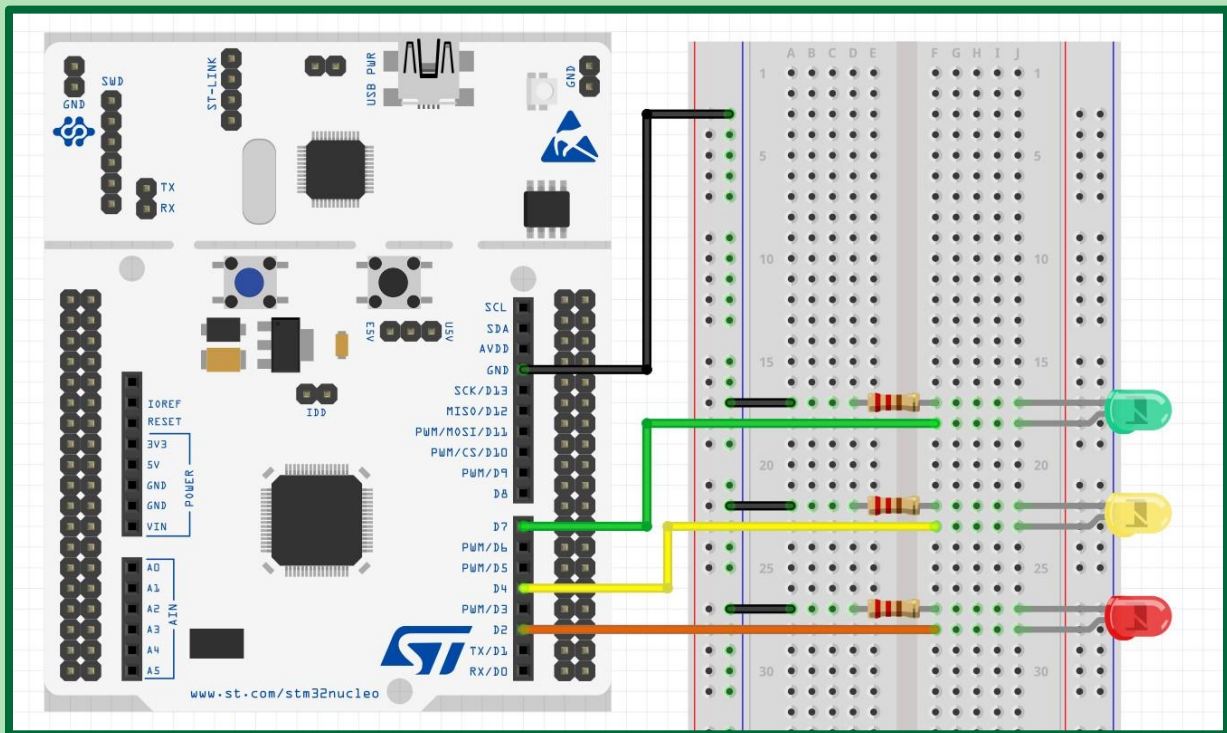
Dans la boucle infinie, on écrit ledPin = 1 pour la mettre à l'état haut et ainsi l'allumer !

Le saviez-vous ? : Le langage de programmation utilisé pour coder sur Mbed est le C++. Il est souvent utilisé pour coder des applications mobiles, des jeux vidéo, des logiciels de bureautique, en robotique...

**Mission : Faire clignoter la LED en changeant son état toutes les secondes**

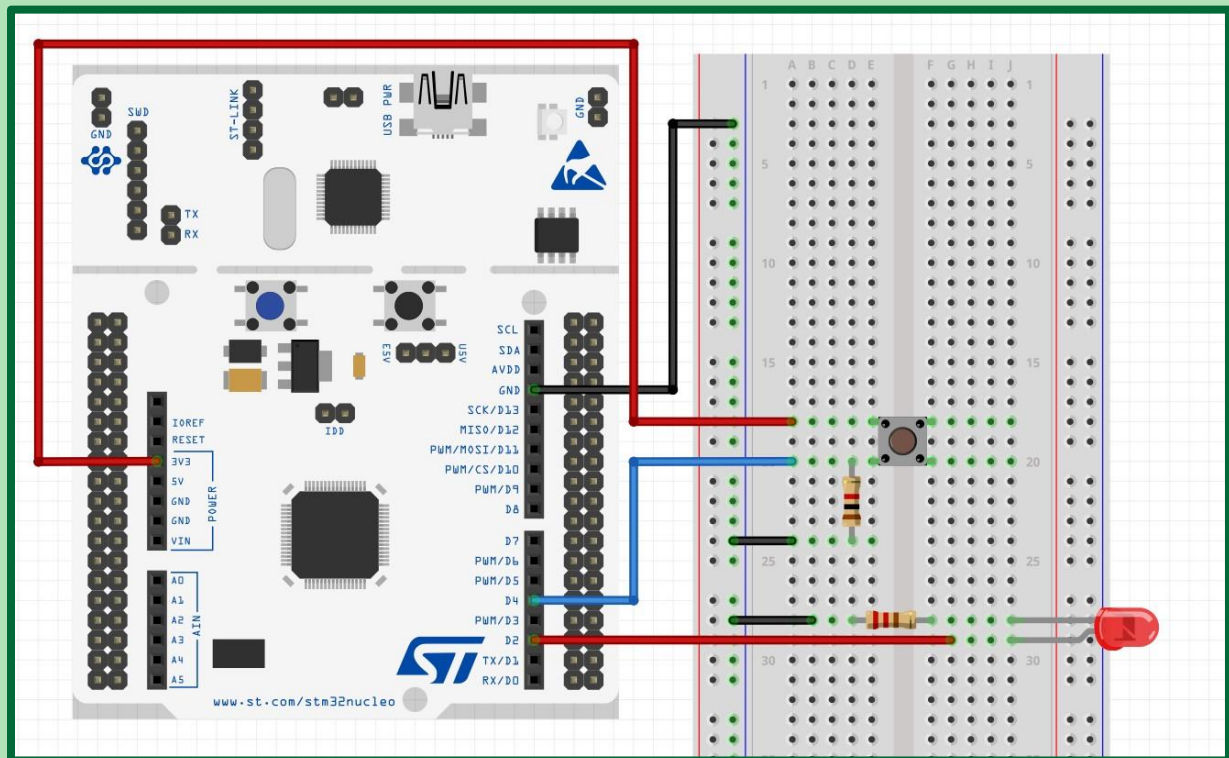


## Niveau 2 – Tricolore



**Mission : Changer de manière successive l'état des 3 LEDs**

# Niveau 3 – Jour, Nuit, Jour...

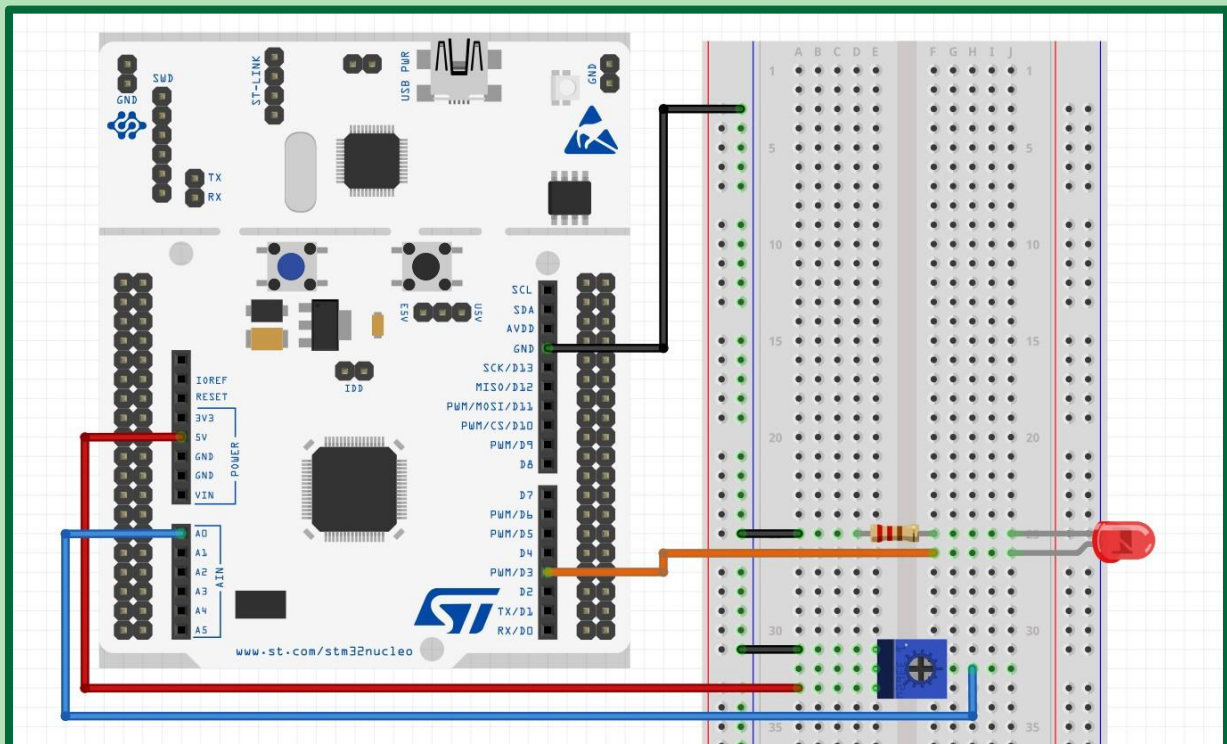


**Mission : Commander l'allumage de la LED grâce à un bouton**

Peut-être que pour connaître l'état du bouton, une fonction existe pour lire l'état d'une broche numérique... Peut-être même qu'elle est dans l'aide-mémoire. La vie est bien faite non ?



# Niveau 4 – Jour, Nuit, version soft

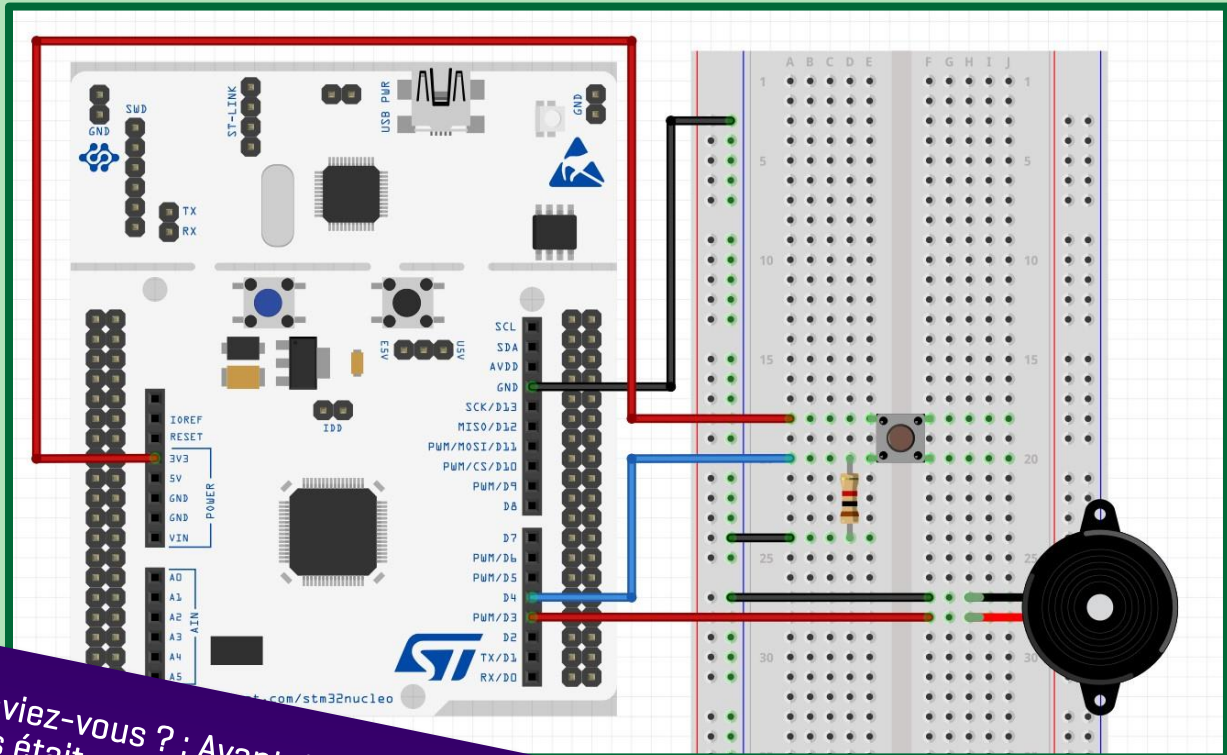


**Mission : Faire varier l'intensité de la LED émise à l'aide d'un potentiomètre**

La broche du potentiomètre est connectée à un port analogique. Hmm... Analogique, analogique, analog... Et la LED est branchée à une broche PWM... Hmm...



# Niveau 5 – Joyeux anniversaire



Le saviez-vous ? : Avant d'être un pôle du BDTech, Ares était une association de robotique à part-entière fondée le 27 septembre 1996. C'est bientôt son anniversaire !

Fréquences des notes et partition simplifiée de « Joyeux anniversaire » :

Notes	Do	Do#	Ré	Ré#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si	Do2
$f$ (Hz)	262	277	294	311	330	349	370	391	415	440	466	493	523



Il y'a 25 notes dans jouées « Joyeux anniversaire ». On peut utiliser la méthode bourrine qui serait d'écrire 25 fois `tone()` pour chaque note, mais je suis sûr qu'il y'a une solution plus rapide ! Une couleur de note correspond à une durée différente, pensez à varier les délais entre celles-ci. Et par pitié, n'oubliez pas d'éteindre le buzzer après un certain temps dans votre code avec `.write(0.0)` !

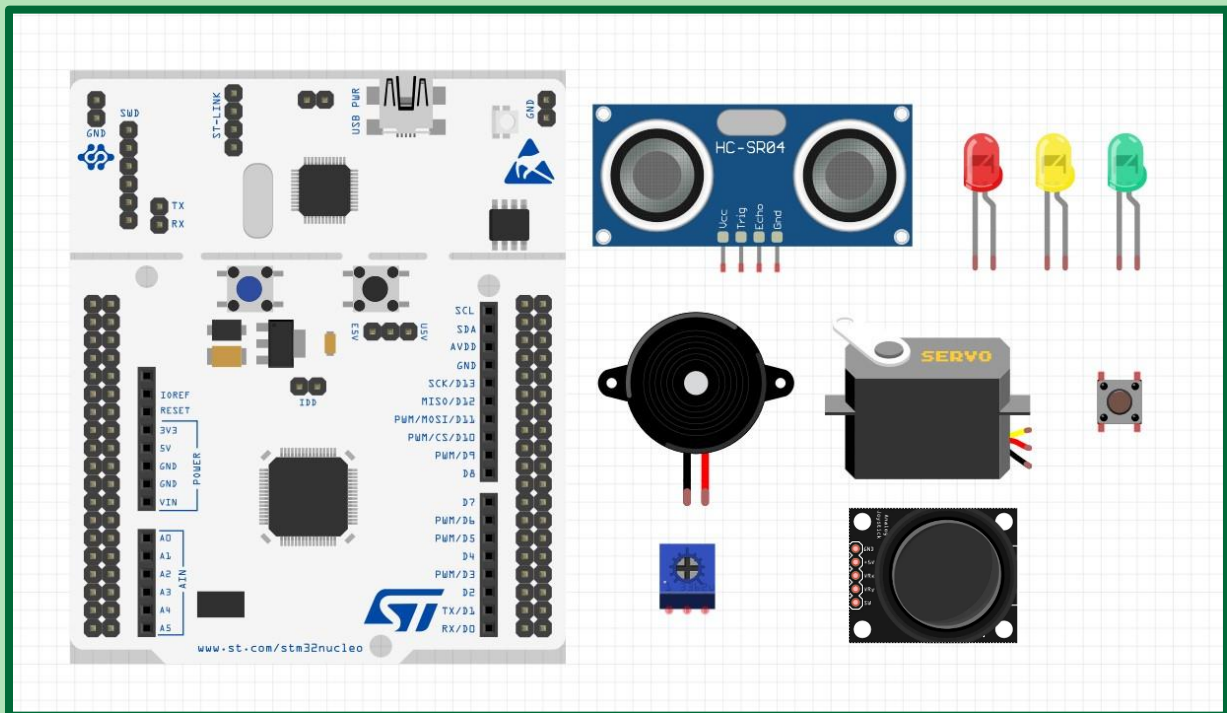


**Mission : Jouer « Joyeux anniversaire » lorsque le bouton est appuyé**

**Mission bonus : Points supplémentaires pour votre clan si vous arrivez à jouer une autre mélodie !**



# Niveau 6 – À vous de jouer !



**Mission : Avec le matériel proposé et les autres membres de votre clan, réalisez quelque chose d'original !**

N'oubliez pas de brancher des résistances aux composants qui en ont besoin. Ils vous remercieront en ne produisant pas de fumée !



# FIN.

# Micro-introduction à STM32



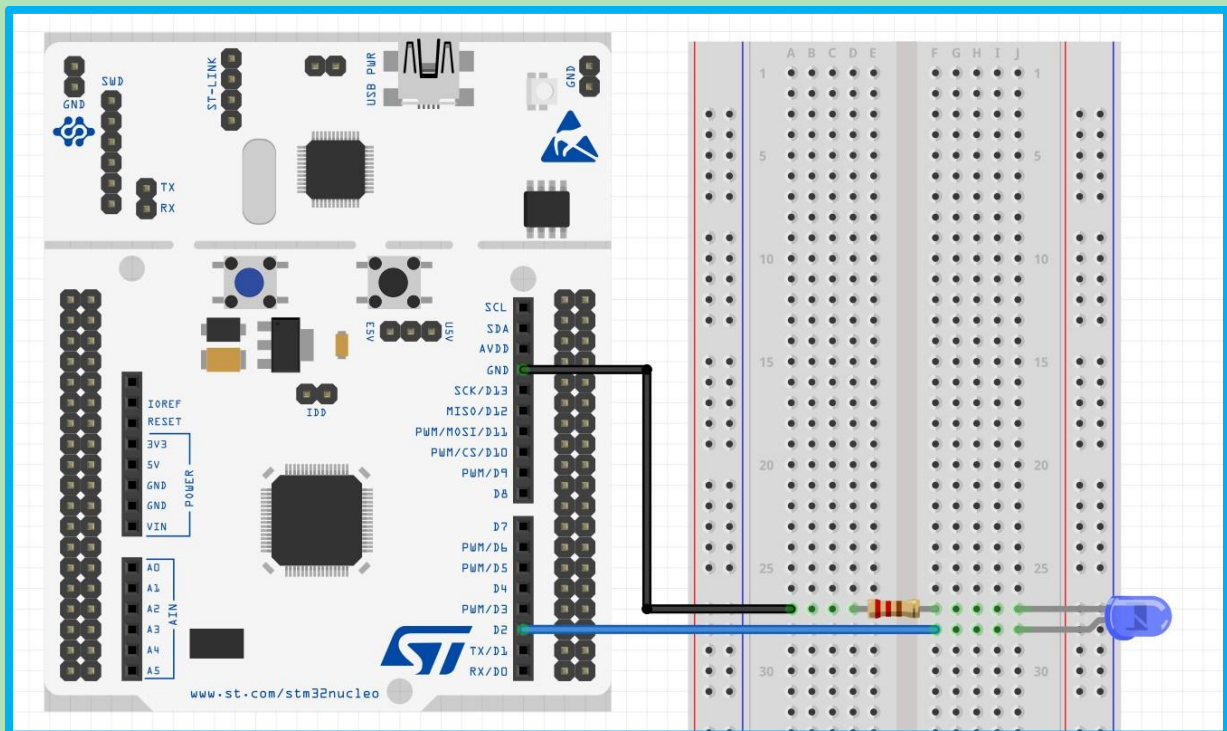
Ici à l'ENSEA, l'école nous apprend à utiliser STM32 Cube IDE, un logiciel un peu plus complexe, mais très important car il est utilisé au sein de beaucoup d'entreprises.

## Deux différences majeures avec Mbed

On peut programmer en langage C++, mais aussi en langage C. C'est ce dernier qui est enseigné à l'école.

Pour définir les entrées/sorties du microcontrôleur, on peut le coder à la main mais aussi le générer automatiquement grâce à l'interface proposée par STM32.

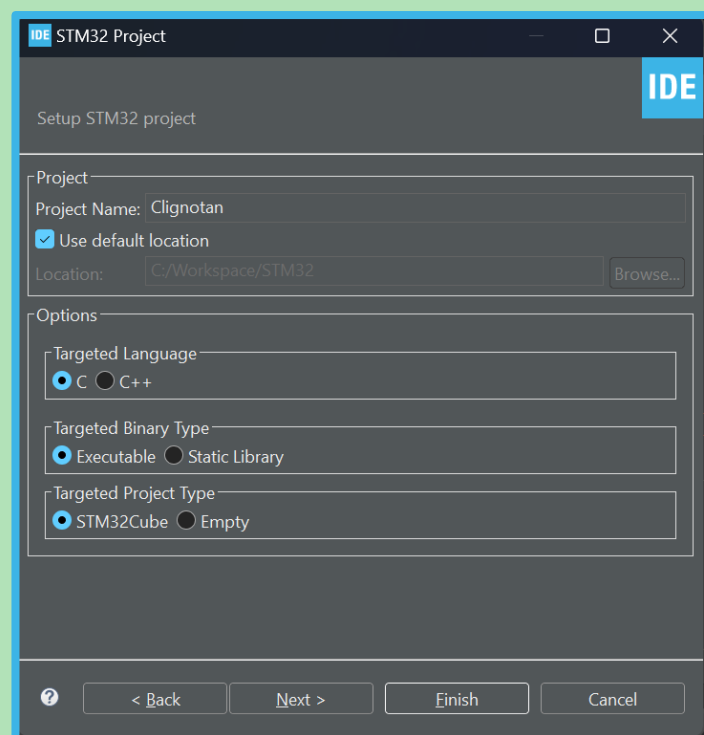
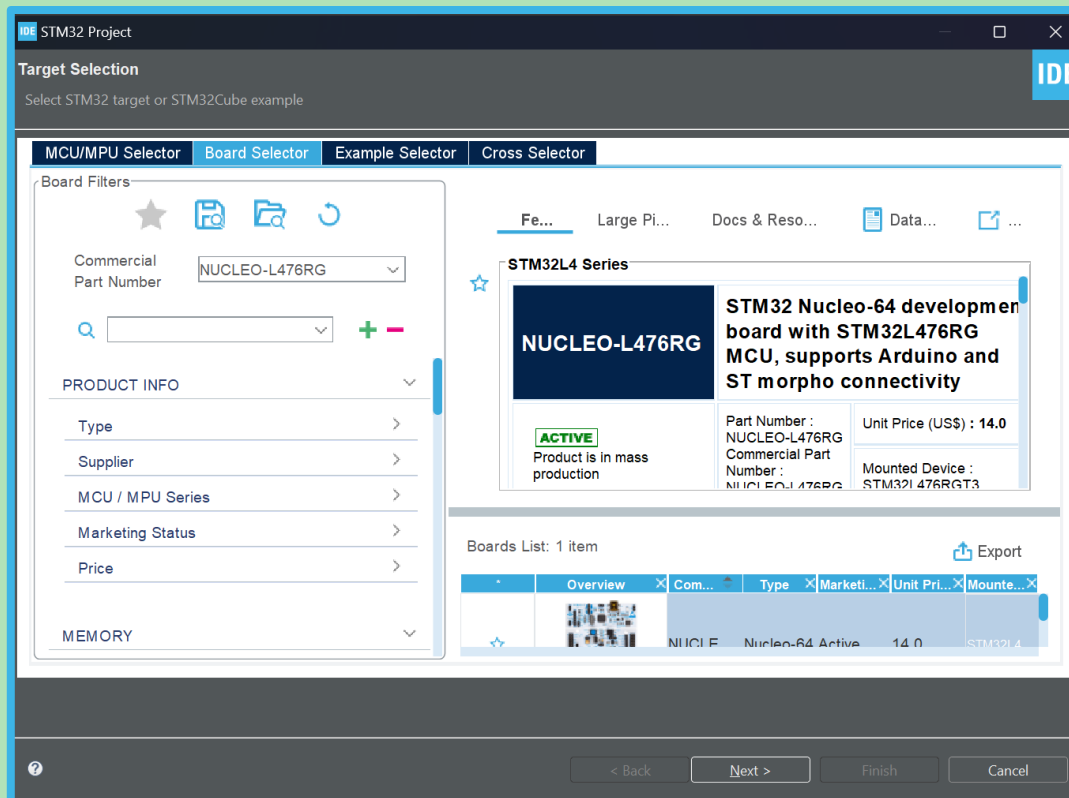
# Niveau 1 EX – Clignotant version STM32



Le montage est le même, que celui du niveau 1, mais pour faire comme si c'était nouveau, j'ai mis une LED bleue.



**Mission EX : Faire clignoter la LED en utilisant STM32.**

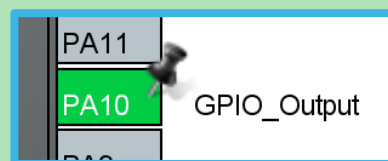
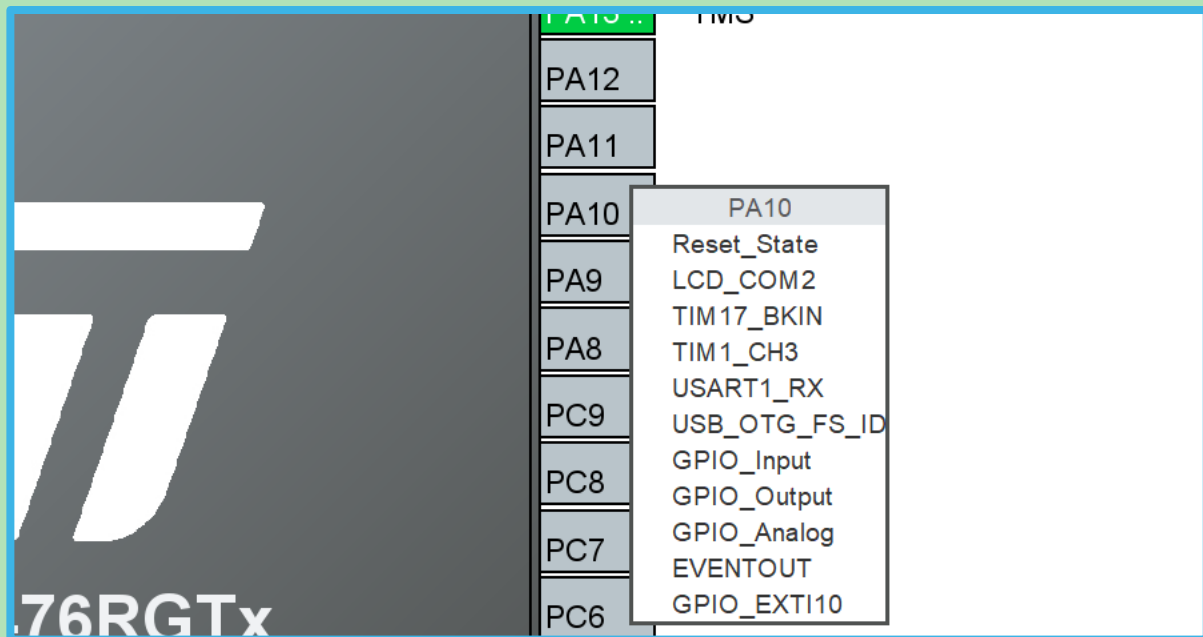


Pour vous expliquer brièvement ces deux images : Après avoir installé STM32, je crée un nouveau « projet STM32 » en sélectionnant la bonne carte et le bon langage cible (Langage C).










Comme vous avez bonne mémoire, vous savez que la broche « D2 » est celle qui correspond à « PA10 » sur le microcontrôleur, on la configure donc directement en sortie [GPIO\_Output].



Je n'oublie pas de générer le code associé en appuyant sur ce bouton , sinon la configuration ne sera pas prise en compte dans le code !

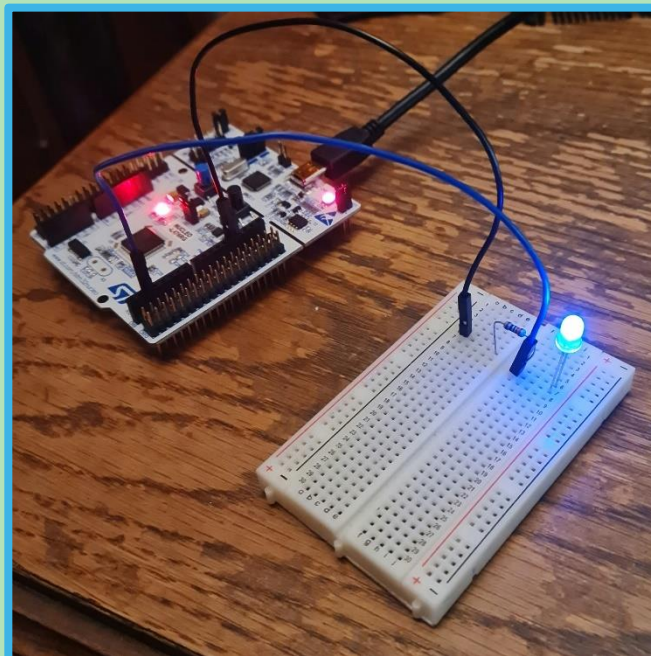
J'écris ensuite mon code dans le fichier « main.c » :

```
main.c x *Clignotant.ioc
86  /* USER CODE END Sysinit */
87
88  /* Initialize all configured peripherals */
89  MX_GPIO_Init();
90  MX_USART2_UART_Init();
91  /* USER CODE BEGIN 2 */
92
93  /* USER CODE END 2 */
94
95  /* Infinite loop */
96  /* USER CODE BEGIN WHILE */
97  while (1)
98  {
99      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_10);
100     HAL_Delay(1000);
101     /* USER CODE END WHILE */
102
103     /* USER CODE BEGIN 3 */
104 }
105 /* USER CODE END 3 */
106 }
```

J'utilise une des fonctions de la librairie HAL, déjà intégrée à STM32. `HAL_GPIO_TogglePin()` permet de changer l'état de la broche définie en paramètre, ici PA10.

`HAL_Delay` nous permet de faire l'attente de 1000ms soit 1 seconde.

On retrouve notre amie la boucle `while (1) { }` ! On peut aussi écrire `while (true) { }`.



Et voilà une belle LED qui clignote ! (Ce n'est qu'une image donc ça ne clignote pas, mais je vous promets que ça fonctionnait quand je l'ai fait !)

