

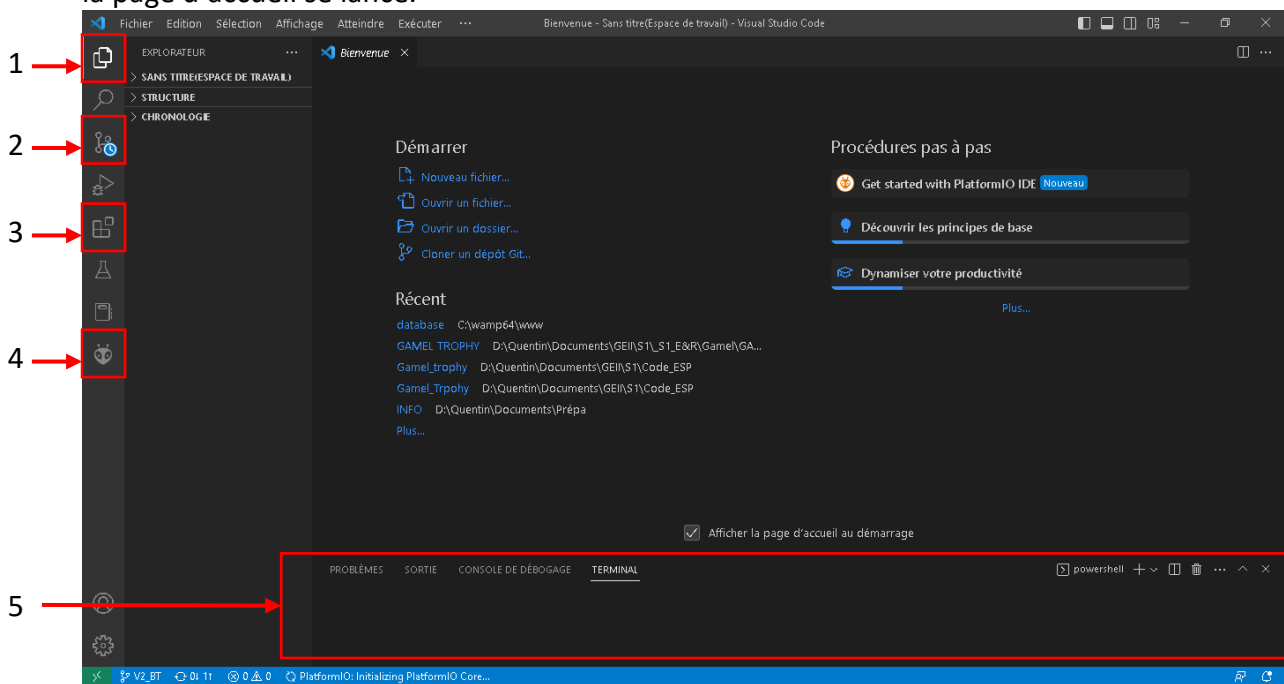


L'Arduino est un langage spécifique au microcontrôleur Arduino (et à plein d'autres), il est basé sur le langage C/C++.

Un IDE (environnement de développement) existe et est spécifique à Arduino, cependant, on utilisera le logiciel **Visual Studio Code** avec l'extension **PlatformIO** pour la suite de ce tutoriel et pour autre projet Arduino.

Installation des logiciels et extensions

Comme indiqué ci-dessus, nous allons utiliser [Visual Studio Code](#), après installation du logiciel, la page d'accueil se lance.

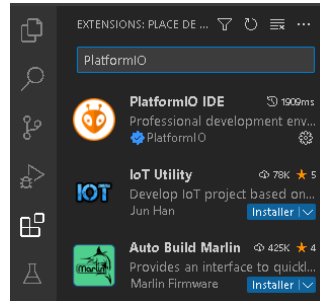


Voici les différentes parties que l'on va principalement avoir besoin par la suite.

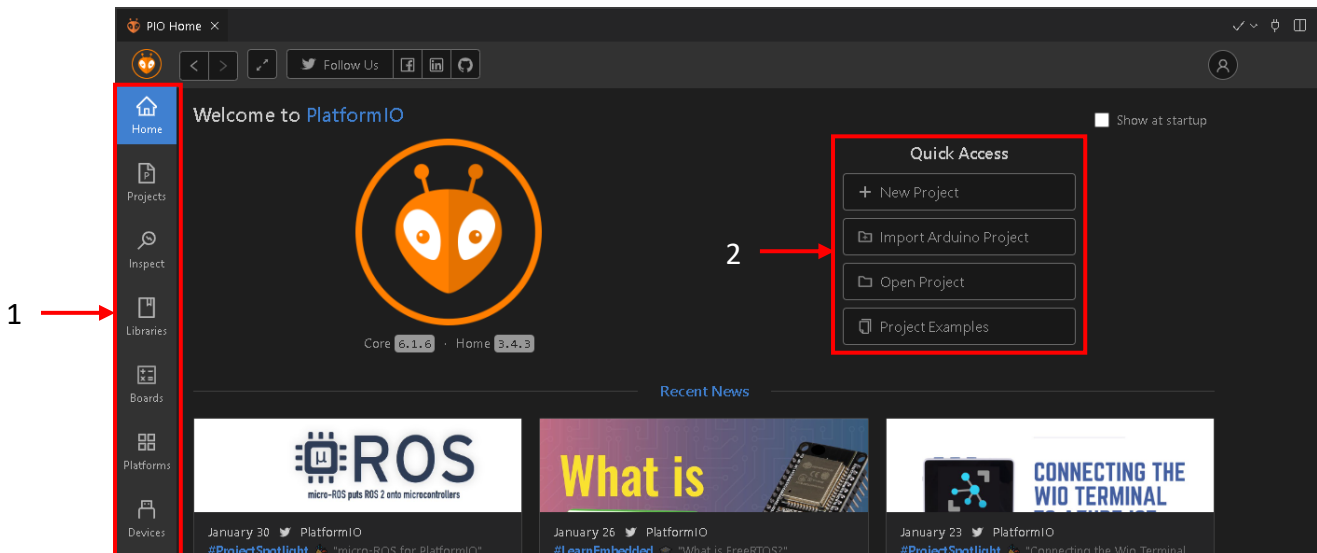
1. **L'onglet Explorateur** : permet de visualiser tous les fichiers du dossier du projet.
2. **L'onglet Contrôle de code source** : est la partie reliée à GitHub et permet de créer le versioning du code et le travail collaboratif.
3. **L'onglet Extension** : permet de rajouter des modules à VS code pour qu'il soit adapté à nos besoins.
4. **L'onglet PlatformIO** : permet de gérer tous les aspects du projet Arduino.
5. **L'onglet Terminal** : retourne toutes les informations liées au code, et sera utile pour visualiser les données de l'Arduino.

Ajout de PlatformIO

- Pour cela, rendez-vous dans l'onglet extension (3.), et entrez le nom de l'extension. Et installez l'extension.



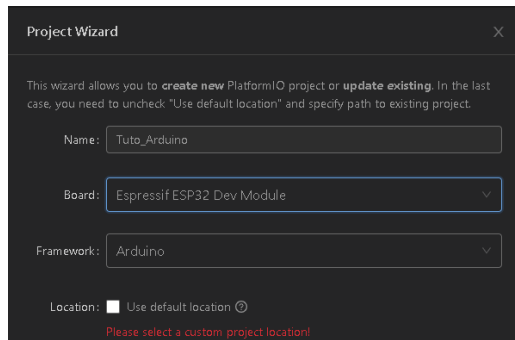
- Une fois l'extension installée et initialisée, la page d'accueil se lancera.



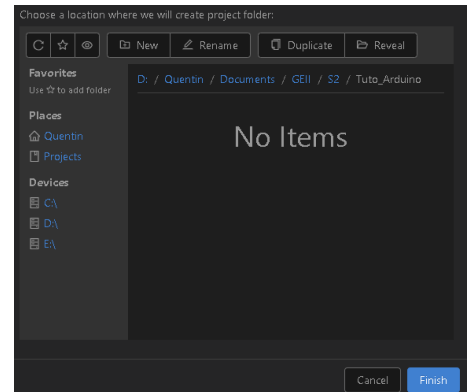
1. Ces onglets ne nous seront pas très utiles par la suite, mais ils sont pratiques pour gérer les différents paramètres des projets.
2. Cette partie est le commencement du projet. On va pouvoir y faire plusieurs actions, comme créer un projet, ouvrir un exemple ou encore ouvrir des projets existants.

Premier projet

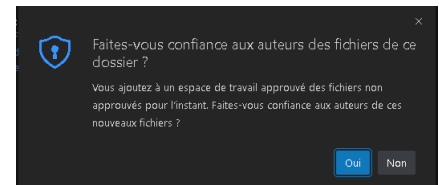
Maintenant, créons ensemble notre premier projet Arduino. Pour cela, cliquez sur « New Project ».



- Donner un nom au projet
- Entrer le nom de la carte utilisé (dans notre cas, ça sera toujours celle la)
- Laisser le Framework en Arduino
- Décider soit de laisser l'emplacement par défaut (pas conseillé), soit de choisir un dossier



Une fois l'emplacement choisi, cliquez sur « Finish ». Une fenêtre s'ouvrira pour demander si le dossier de travail est de confiance toujours dire « Oui ».



Après cela, les choses sérieuses commencent. Rendez-vous dans le dossier « src » et cliquez sur « main.cpp » (1.). Le fichier ouvert est le programme Arduino du projet, le fichier principal.



2 grandes parties sont visibles dans le code :

- La partie 2 -> est comme son nom l'indique le « setup », cette partie est exécuté qu'une seule fois au démarrage de l'Arduino. Elle correspond en C, à la zone entre le « int main » et le « while 1 ». Elle sert à initialiser tous les pins, les différentes bibliothèques et du paramétrage.
- La partie 3 -> correspond au « while 1 » du C, c'est une boucle infinie. Tout le code si trouvera.
- Toutes les variables et les prototypes de fonction se retrouveront avant le « void setup ».

Commande de base

Le langage Arduino simplifie énormément l'initialisation et la lecture / écriture des entrées/sorties. Cependant, il utilise la même syntaxe que le C, donc accolade et point-virgule. On y retrouve, dans la **partie setup** :

« **pinMode()** » pour fixer un pin sur Entrée ou Sortie

```
1 pinMode(numero_pin,type_de_pin); //Permet d'initialiser un pin
2 /*
3  Les types de pin :
4  - INPUT  -> pour un pin d'entrée
5  - OUTPUT -> pour un pin de sortie
6  */
```

« **Serial.begin()** » démarre la liaison série USB, les taux de transfert les plus fréquents sont « 9600 » et « 115 200 ».

```
1 Serial.begin(9600); //Démarre la liaison série de la carte
```

L'**assignement des variables** est exactement le même qu'en C. Deux exemples de constantes pratiques à utiliser :

« **const int** » est recommandé pour définir des variables avec des valeurs constantes comme les pins par exemple

```
1 const int bt_Jaune = 21;
```

« **unsigned long int** » est un entier de type long, et pas signé. Ce type de variables est utilisé pour sauvegarder le temps actuel.

```
1 unsigned long int temps = millis(); //millis() étant la fonction qui renvoie le temps écoulé en milliseconde
```

;

Les tableaux se déclarent de la même manière qu'en C

```
1 int tableau[4] = {0,0,0,0};
```

Les fonctions utiles dans le « void loop » :

« **digitalWrite()** » cette fonction prend en paramètre 1, le numéro de la pin à modifier et en 2, la valeur voulue.

```
1 digitalWrite(pin,valeur_de_sortie);//Permet de mettre à 1 ou 0 un pin
2 /*
3  Les valeurs de sortie:
4  - HIGH -> pour mettre le pin à 1
5  - LOW  -> pour mettre le pin à 0
6  */
```

« **digitalRead()** » cette fonction renvoie une valeur booléen (0 ou 1), elle peut être utilisée dans un « if » ou dans une assignation de variable.

```
1 digitalRead(pin); //Pour lire la valeur d'un pin (0 ou 1)
```

« **analogRead()** » est une fonction qui renvoie une valeur flottante. Pour utiliser une entrée analogique, sur les cartes Arduino, les noms de ces ports sont « A0, A1... ». Sur une esp32 ce n'est pas le cas, juste le nom de l'entrée normal.

```
1 analogRead(pin); //Pour lire la valeur d'un pin (0 à (2**n)-1)
2 //n étant la résolution du CAN
```

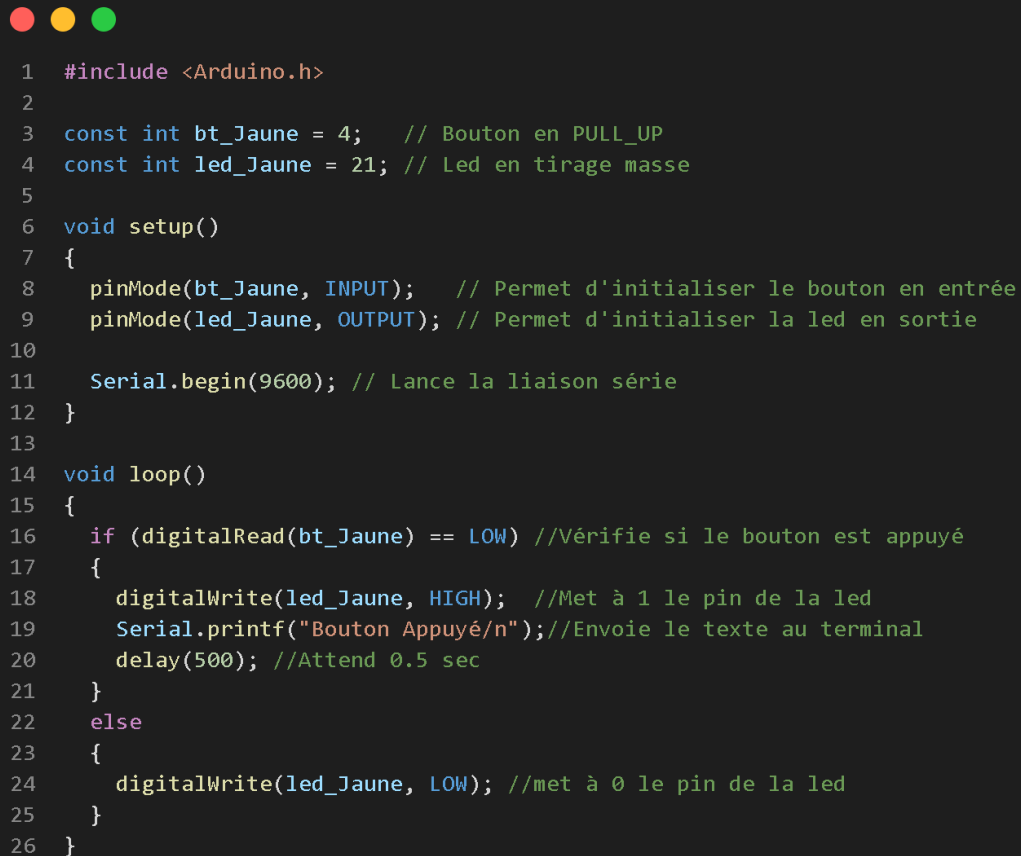
« **delay()** » cette fonction permet de faire une pause dans le programme pendant un temps donné. Son utilisation est assez problématique lors de communication, mais dans certains cas peut servir à ralentir le programme.

```
1 delay(temps); //Fait une pause d'un temps voulu en milliseconde
```

Le reste des principes du C s'applique dans le langage Arduino, tel que le « switch », « define » etc etc etc.

Exemple code simple

Ci-dessous, voici un code simple pour tester les acquis ci-dessus.



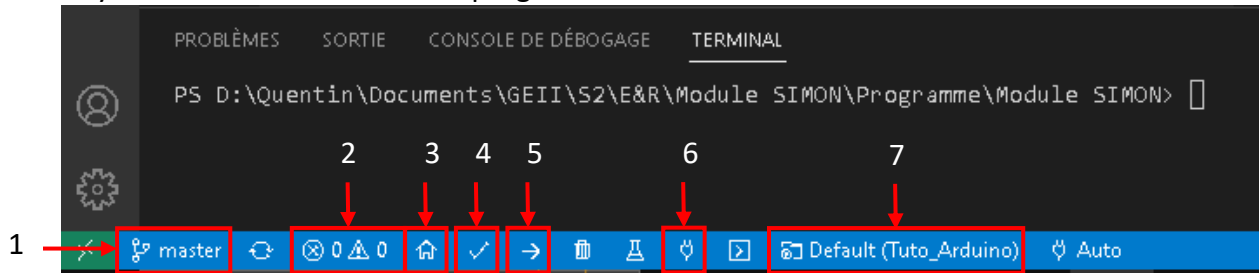
```
1  #include <Arduino.h>
2
3  const int bt_Jaune = 4;  // Bouton en PULL_UP
4  const int led_Jaune = 21; // Led en tirage masse
5
6  void setup()
7  {
8      pinMode(bt_Jaune, INPUT);  // Permet d'initialiser le bouton en entrée
9      pinMode(led_Jaune, OUTPUT); // Permet d'initialiser la led en sortie
10
11      Serial.begin(9600); // Lance la liaison série
12  }
13
14  void loop()
15  {
16      if (digitalRead(bt_Jaune) == LOW) //Vérifie si le bouton est appuyé
17      {
18          digitalWrite(led_Jaune, HIGH); //Met à 1 le pin de la led
19          Serial.printf("Bouton Appuyé/n");//Envoie le texte au terminal
20          delay(500); //Attend 0.5 sec
21      }
22      else
23      {
24          digitalWrite(led_Jaune, LOW); //met à 0 le pin de la led
25      }
26  }
```

Que fait ce code ?

1. Il inclut la bibliothèque Arduino
2. Déclare et définit les constantes des composants
3. Fixe le type des pins et démarre la liaison série
4. Ensuite vérifie si le bouton jaune est appuyé
5. Si oui, allume la led jaune, envoie « Bouton appuyé » et attend 0.5 sec
6. Sinon, éteint la led jaune
7. Recommence à la ligne 16

Wahooo c'est magique !

Voyons comment téléverser ce programme.



1. Indique dans qu'elle branche du code on travaille. (On verra plus tard avec GitHub)
2. Indique les erreurs présentes dans le code
3. Ouvre la page d'accueil de PlatformIO
4. Compile le code
5. Compile et téléverse le code dans la carte
6. Ouvre le moniteur série, cet onglet permet de discuter avec la carte
7. Indique quel projet va être compilé

Petit Exercice :

Maintenant, si on veut que lorsqu'on appui sur le bouton jaune, la led jaune s'allume, 0.3 sec après la led verte s'allume pendant 0.5 sec (les deux s'éteignent après le temps). Puis, si j'appuie sur le bouton Vert, toutes les led s'allument sans s'éteindre, et si on appuie sur le bouton noir tout s'éteint !?

Une fois le code fait et testé, voyez comment l'améliorer avec l'astuce ci-dessous.

Code BONUS

Exemple de code pour créer un timer sans utiliser delay().

```
1  #include <Arduino.h>
2
3  const int bt_Jaune = 4;  // Bouton en PULL_UP
4  const int led_Jaune = 21; // Led en tirage masse
5
6  unsigned long int temps = millis(); //millis() étant la fonction qui renvoie le temps écoulé en milliseconde
7
8  void setup()
9  {
10     pinMode(bt_Jaune, INPUT);  // Permet d'initialiser le bouton en entrée
11     pinMode(led_Jaune, OUTPUT); // Permet d'initialiser la led en sortie
12
13     Serial.begin(9600); // Lance la liaison série
14 }
15
16 void loop()
17 {
18     if (digitalRead(bt_Jaune) == LOW) //Vérifie si le bouton est appuyé
19     {
20         temps = millis();
21         Serial.printf("Bouton Appuyé\n");
22     }
23     if(millis() < temps + 500){
24         digitalWrite(led_Jaune,HIGH);
25     }else digitalWrite(led_Jaune,LOW);
26 }
27
```