OZANAM

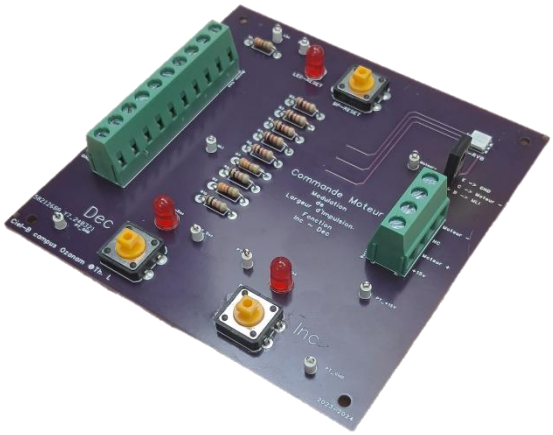
Rapport mini-projet :
Commande Moteur MLI

Benoit BOREK
BTS CIEL 2B
Campus Ozanam

Avant de commencer à câbler ou à créer le programme j'ai dû lister les constituants nécessaires pour mener à bien le mini-projet.

<u>Constituants nécessaires :</u>

- **Une carte PCB avec :**
  - Un bornier avec 10 bornes
  - Un bornier avec 4 bornes
  - 3 résistances 330 Ohms
  - 7 résistances 1k Ohms
  - 3 LED rouge
  - 1 LED RBG
  - 1 transistor BD681
  - 3 boutons poussoirs
  - 9 borniers test
- **Un Arduino mega 3560**
- **Un moteur**
- **Une alimentation externe**



Une fois que j'ai regrouper tous les constituants, il m'a fallu souder les résistances qui n'étaient pas soudés nativement. Pour cela j'ai installé un poste de soudure avec une troisième main et j'ai souder les résistances en respectant le schéma électrique du PCB.

J'ai ensuite procédé au câblage de mes constituants, le choix des pins Arduino étant libre, j'ai pu avoir un meilleur confort pour écrire le code par la suite.

Câblage :

Bornier à 10 ports :

Bornier	Pin Arduino
+5V	+5V
/RESET	Reset
VERT	7 (PMW)
ROUGE	6 (PMW)
BLEU	8 (PMW)
MLI	4 (PWM)
/INC	3 (PMW)
/DEC	2 (PMW)
NC	Pas utilisé
GND	GND

Bornier à 4 ports :

Bornier	
Moteur -	Au GND du moteur
NC	Pas utilisé
Moteur +	Au VCC du moteur
+15V	Au VCC de l'alimentation externe

A noter : Le moteur n'a pas de sens de branchement (hormis le fait que le moteur va tourner dans un sens différent) ce qui veut dire qu'il n'a pas de GND fixe ou de VCC fixe.

L'alimentation externe doit être régler à +15V et 150 mA pas plus au risque de détruire le moteur.

Le GND de l'alimentation externe doit être brancher au GND du bornier à 10 ports ou au GND de l'Arduino.

Une fois le câblage des composants fait il faut coder le programme qui permettra de mettre en relation les boutons, les LEDs et le transistor qui contrôlera indirectement la vitesse du moteur.

Le programme :

Donc au début du programme il m'a fallut déclarer des entiers qui enregistrerons le pin de l'Arduino que je souhaite. Je déclare aussi un entier « curseur » qui servira d'enregistrer la valeur du palier.

J'ai volontairement mis 5 paliers :

Pallier 1	0% vitesse max du moteur
Pallier 2	25% vitesse max du moteur
Pallier 3	50% vitesse max du moteur
Pallier 4	75 % vitesse max du moteur
Pallier 5	100% vitesse max du moteur

Ensuite dans le « void setup() », j'ai déclaré les pins de l'Arduino si étant INPUT ou OUTPUT.

```
void setup() {
// Déclaration des variables de la LED RGB en tant que PIN de l'Arduino.
// Les PINS seront en OUTPUT pour dire que l'Arduino sera prêt à envoyer du courant dans ces PINS.
pinMode(red, OUTPUT);
pinMode(green, OUTPUT);
pinMode(blue, OUTPUT);

// Je fait de même avec la variable mli.
pinMode(mli, OUTPUT);

// Déclaration des variables d'incrémentation et de décrémentation en tant que PIN de l'Arduino.
// Ces PINS sont en INPUT ce qui veut dire que l'Arduino sera prêt à recevoir du courant des ces PINS.
pinMode(inc, INPUT);
pinMode(dec, INPUT);

 // Serial.begin(9600); // Initialisation de la liaison Arduino <----> PC.
}
```

Figure 1 Code du "void setup()"

Pour finir dans le « void loop() » j'ai fait en sorte que quand le pin du bouton d'incrémentation/décrémentation change d'état (état bas) alors il va incrémenter ou décrémenter le curseur.

```arduino
if (digitalRead(inc) == LOW) // Condition si j'appuie sur le bouton incrémentation alors le curseur va s'incrémenter de 1.
{
  curseur = curseur + 1; // Incrémentation du curseur.

  if(curseur > 4) { // Condition de sécurité, si j'incrémente et que le curseur est supérieur à 4 (valeur max).
                    // Alors je remet la valeur du curseur à 4.
    curseur = 4;
  }
}
```

Figure 2 Code pour le bouton d'incrémentation

J'ai aussi mis dans le code une sécurité qui permet à ce que le curseur ne va pas au-dessus de 4. Il y avait une autre question que je me suis poser c'est pourquoi j'ai dû mettre la condition en LOW et pas en HIGH ? Pour moi ça serait plus logique. Pour répondre à cette question j'ai utilisé un Pico Scope pour relever le chronogramme du bouton.

Figure 3 Chronogramme d'un bouton d'incrémentation

C'est avec ce chronogramme ci-dessus que j'ai compris que le bouton laissait passer de base la tension et que c'est en appuyant sur celui-ci qu'il va couper la tension. C'est pourquoi dans le

code j'ai dû mettre un LOW dans la condition car si c'était un HIGH alors la variable curseur va s'incrémenter tout seul jusqu'à que j'appuie sur le bouton ce qui n'est pas recherché.

Ensuite dans le code j'ai codé chaque pallier avec la couleur correspondant à celui-ci ainsi qu'à la tension de sortie du MLI.

```
if(curseur == 1) { // Condition si le curseur est égal à 1 alors il va changer la couleur de la LED RGB et la puissance du moteur (25%)

 // La LED RGB éclairera en orange.
  analogWrite(green, 127);
  analogWrite(red, 128);
  analogWrite(blue, 0);

// Je change la sortie mli à 64 donc le moteur tourne à 25% de sa vitesse maximale.
  analogWrite(mli, 64);
}
```

Figure 4 Exemple du code au pallier 2

La variable MLI correspond à la tension de sortie du port MLI (port 4 de l'Arduino) variant de 0V à +5V. Sur les Arduino Méga il y a plusieurs ports qui sont notés « PWM », ils servent à envoyer ou recevoir une tension analogique, ce qui n'est pas le cas des ports qui ne le sont pas. De base les ports de l'Arduino peuvent soit envoyé 0V soit +5V mais pas +2,3V par exemple.

Dans le code Arduino on n'entre pas directement la valeur de la tension de sortie mais à la fréquence ou elle va sortir du +5V et 0V. J'ai pu relever cette fréquence avec un Pico Scope qui m'a permis de visualiser comment une sortie « PWM » fonctionnait.

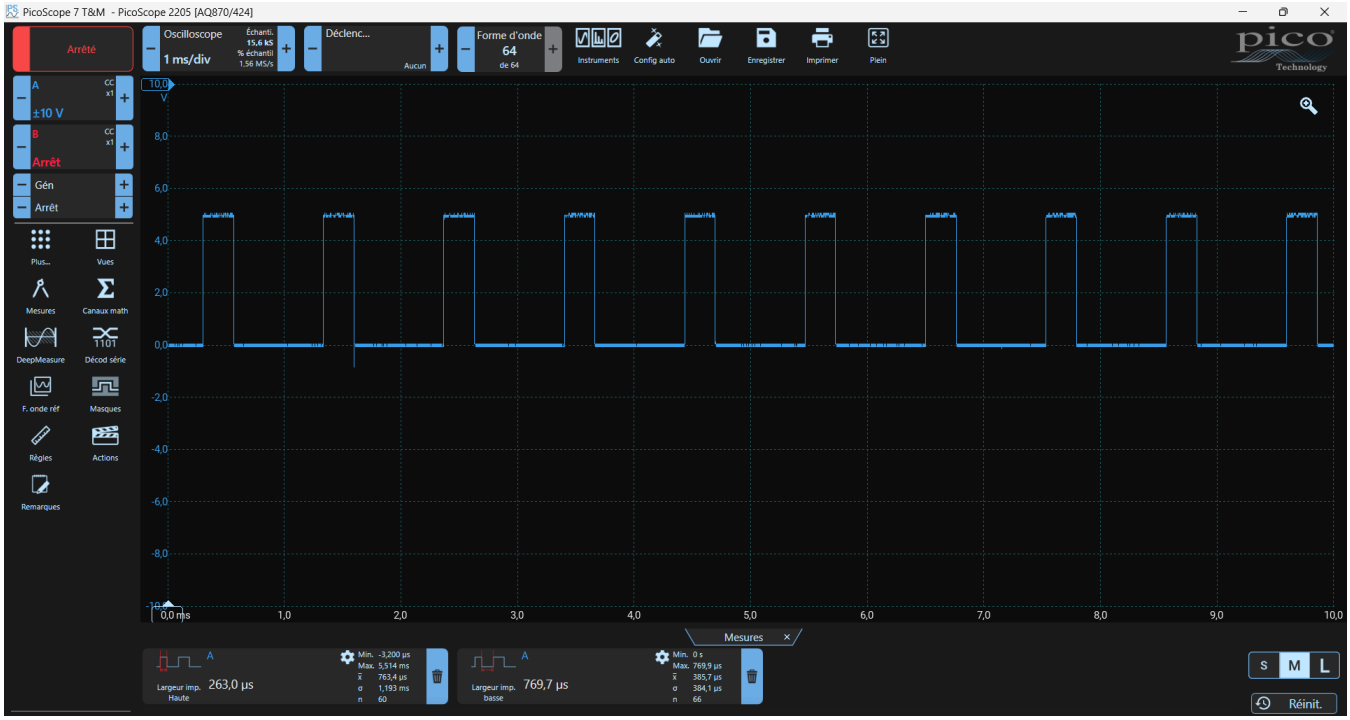


Figure 5 Chronogramme d'une sortie "PWM" à 25%

Sur le chronogramme ci-dessus j'ai pu visualiser cette fréquence d'altération entre 0V et 5V.

Une fois le codage fini j’ai flashé le code dans mon Arduino, j’ai testé les boutons d’incrémentation et de décrémentation et tout fonctionnait correctement.

Pour finir j’ai envoyé le code sur mon repository GitHub pour que n’importe qui puissent voir mon code et me proposer de l’améliorer.

Aussi il me permettra d’enregistrer en ligne mon projet et donc de pouvoir le récupéré sur n’importe quel appareil.

Téléchargement, installation et téléversement du code :

Téléchargement et installation du code :

GIT doit être installé sur votre ordinateur (Obligatoire pour l'installation automatique).
Lien de téléchargement de GIT : https://git-scm.com/downloads

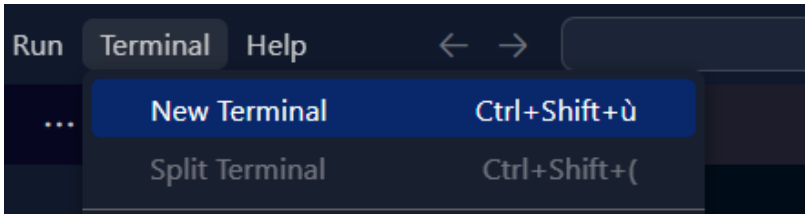
Le code doit être téléverser depuis l'IDE Visual Studio Code Ver 1.95.2.

L'extension Platform IO (Core : 6.1.16 Home : 3.4.4) est requise et doit être installer au préalable depuis Visual Studio Code.

Deux options s'offrent à vous :

Installation automatique :

1- Créez un dossier dans votre ordinateur et ouvrez le dans Visual Studio Code.
2- Cliquez sur Terminal puis New Terminal sur la barre d'action de Visual Studio Code.



3- Dans le Terminal écrivez :
« git clone https://github.com/BOREKBenoit/moteurIncDec.git »

4- Pressez la touche « Entrer », le programme git va récupérer tous les fichiers sur mon repository contenant le code.

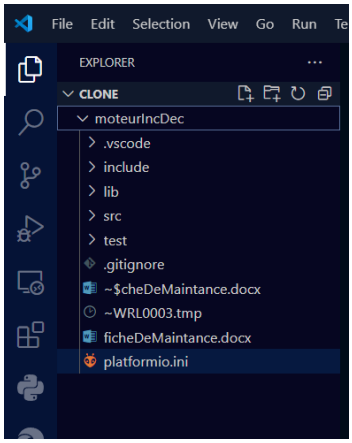


Figure 7 Dossier "CLONE" une fois la commande entrée.

```
PS                                > git clone https://github.com/BOREKBenoit/moteurIncDec.git
Cloning into 'moteurIncDec'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 30 (delta 5), reused 28 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (30/30), 591.31 KiB | 2.71 MiB/s, done.
Resolving deltas: 100% (5/5), done.
```

Figure 6 Terminal avec la commande écrite

Installation manuelle :

1- Rendez-vous sur le repository : https://github.com/BOREKBenoit/moteurIncDec
2- Cliquez sur « Code » un menu déroulant si présente et cliquez « DownloadZIP ».

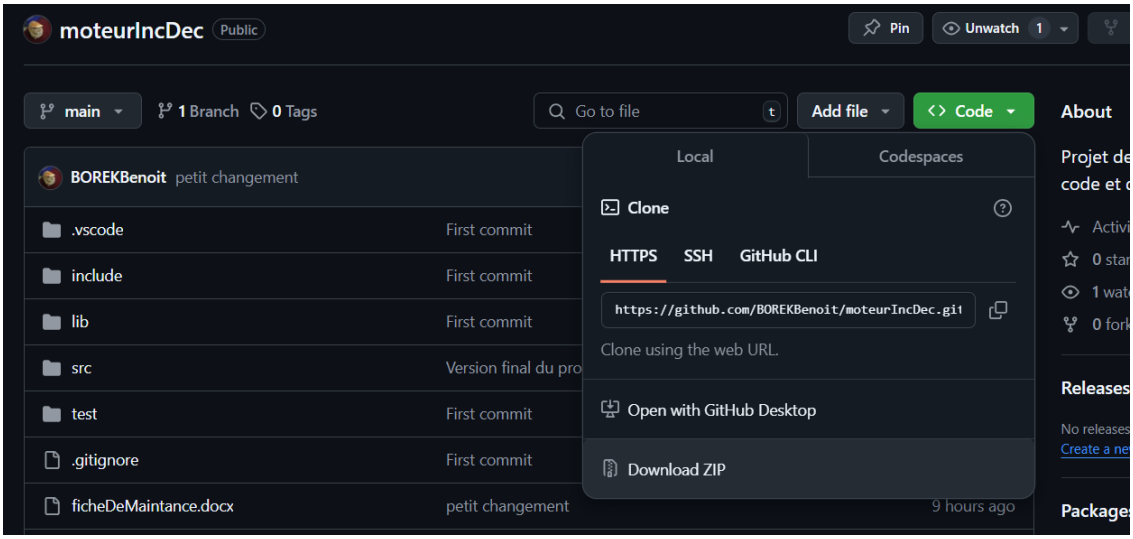


Figure 8 Capture pour illustrer la consigne numéro 2.

3- Une fois le .zip télécharger veuillez extraire le dossier et l'ouvrir dans Visual Studio Code.

Importation d'un projet dans l'extension Platform IO et téléversement du code :

Cliquez sur l'extension de Platform IO qui se trouve sur la barre latéral gauche de Visual Studio Code et ouvrez-le.

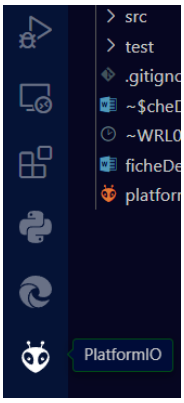


Figure 10 Extension se situant dans la barre latérale gauche

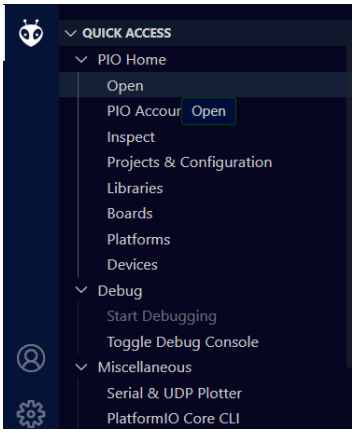
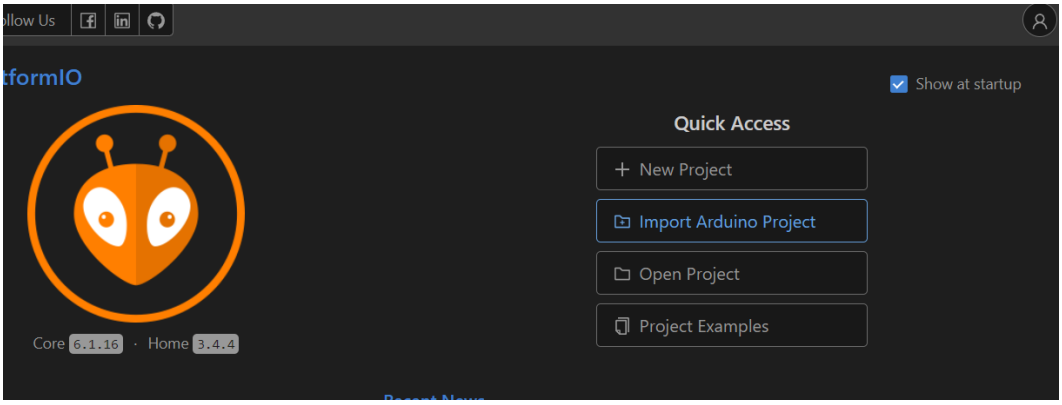


Figure 9 Ensuite cliquez sur "Open" pour ouvrir Platform IO

Une fenêtre va alors s'ouvrir et il faudra cliquer sur « importer un projet Arduino ».



Une fenêtre s'ouvre avec un menu déroulant « Select a board »,

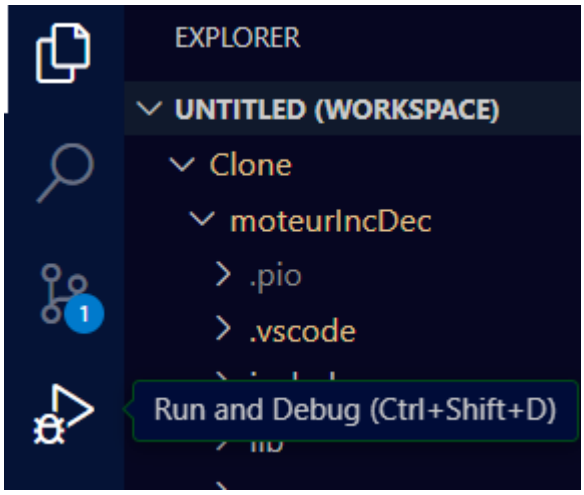
sélectionnez « Arduino Mega or Mega 2560 ATmega2560 (Mega 2560) ».

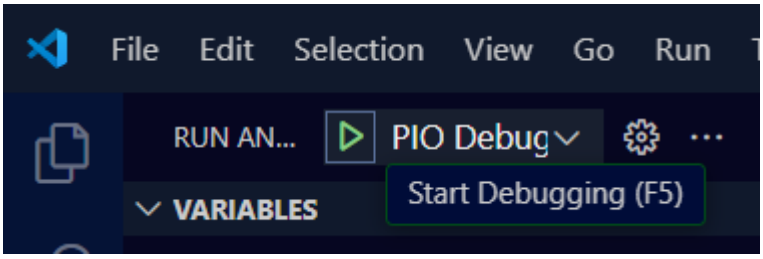
En-dessous une sorte de gestionnaire de fichier vous invite à ouvrir le dossier contenant tous les fichiers, puis cliquez sur « Import ».

L'extension va initialiser le projet en tant que projet Arduino.

Pour téléverser le code allez dans la barre latérale et cliquez sur « Run and Debug ».

Ensuite cliquez sur « Start debugging » ou F5.





Informations générales :

- Équipement : Commande Moteur – Modulation de Largeur d'impulsion. Fonction Inc – Dec.
- Date de maintenance : 20/11/2024.

Intervention spécifique :

- Symptômes constatés :
  - Les dix résistances ne sont pas soudées.
  - Le transistor BD681 a été soudé à l'envers.
- Réparation :
  - Actions réalisées : Dessoudage et soudage des composants.
  - Outils utilisés : Station de soudage avec fer à souder et pistolet à air chaud, 3ème main et une pince.
  - Temps d'intervention : environ 2 heures.

Pièces et consommables :

- Composants :
  - 3 résistances de 330 Ohm sur R1 ; R7 et R9.
  - 7 résistances de 1k Ohm sur R2 ; R3 ; R4 ; R5 ; R6 ; R8 et R10.
  - Transistor BD681 à tourner sur 180 degrés.
- Consommables :
  - Étain.

Vérification des soudures :

Figure 11 Résistance R4 soudé

Figure 12 Transistor BD681 soudé

Courbes relevées :