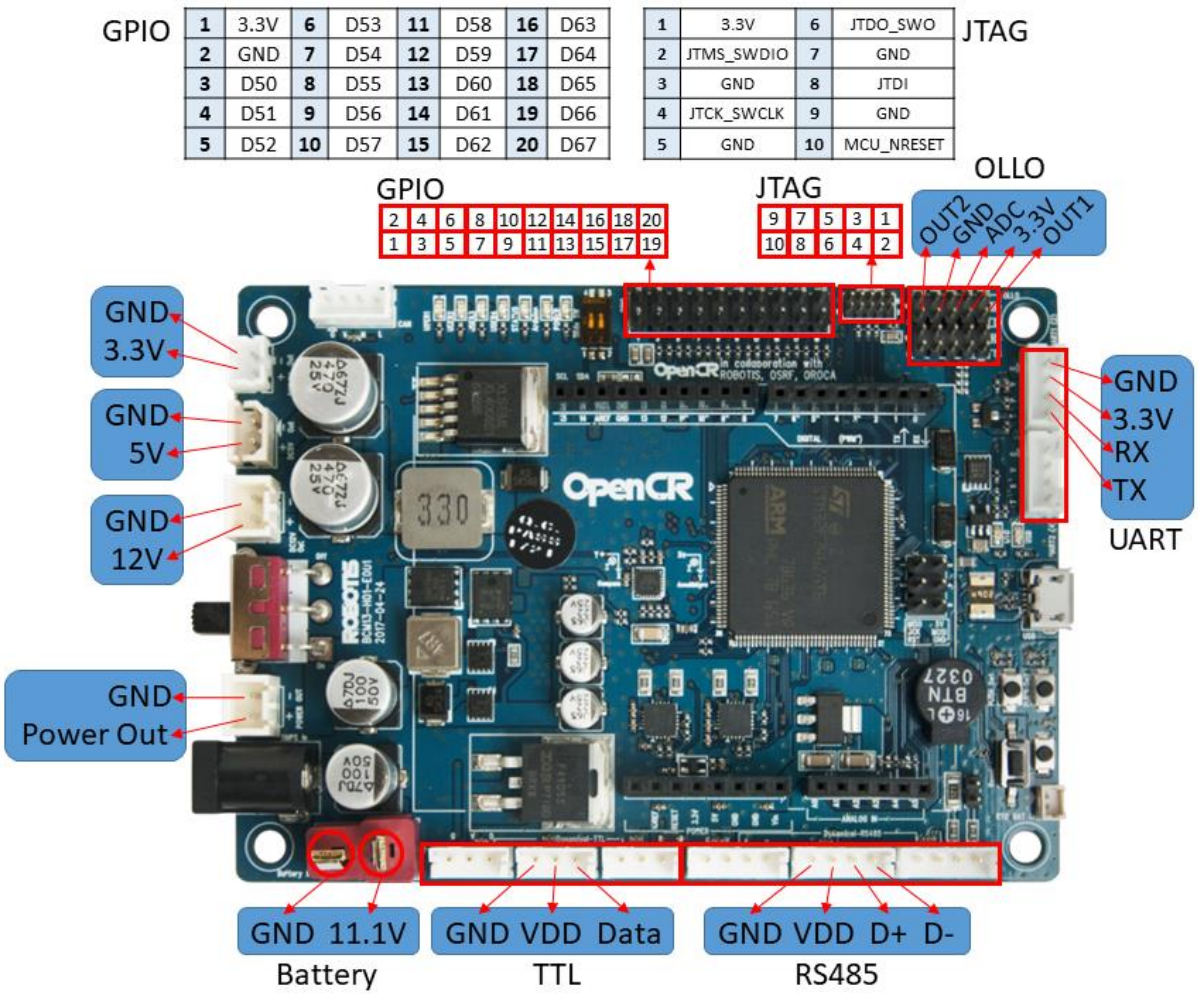
Doc Utilisation Moteur Saé

Table des matières

Pour une utilisation avec la carte STM32, vous pouvez aller directement à : Utilisation Pour STM32 Initialisation des moteurs



[Pin Définition]

Tout d'abord initialiser les moteurs pour faire la programmation de bas niveaux

Ne pas tenir compte sir la carte Open CR n'est pas utilisé :

[Open CR Test]

D'abords faire l'initialisation de l'open CR et lancer le programme pour les moteurs et leurs initialisations

[TurtleBot3 DYNAMIXEL setup instructions]

Programmation de bas niveaux (directement sur l'open CR)

Avec Arduino :

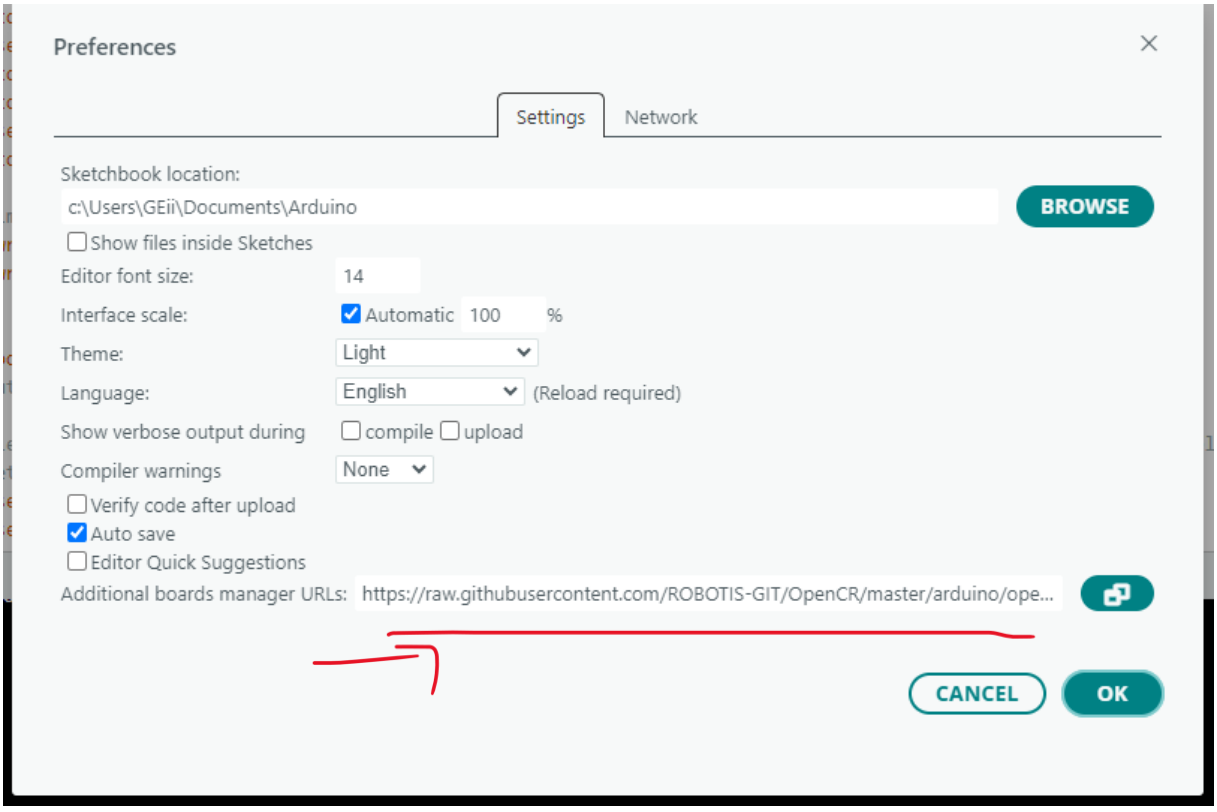
Installer Arduino IDE

https://www.arduino.cc/en/Main/Software

Ajouter la carte Open CR

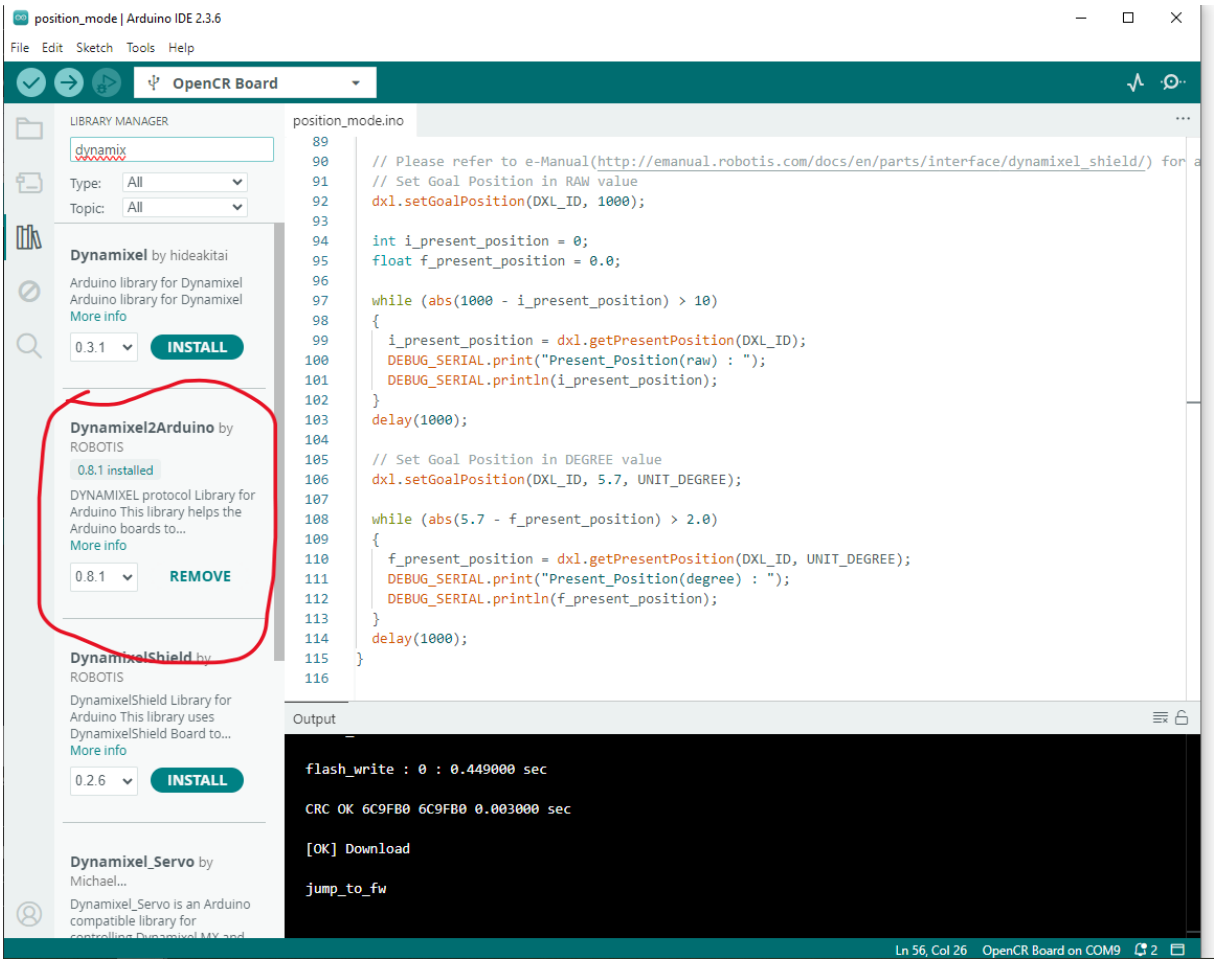
Dans File → Preferences, ajouter l'URL suivante dans *the Additional Boards Manager URLs :*

https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/arduino/opencr_release/package_opencr_index.json

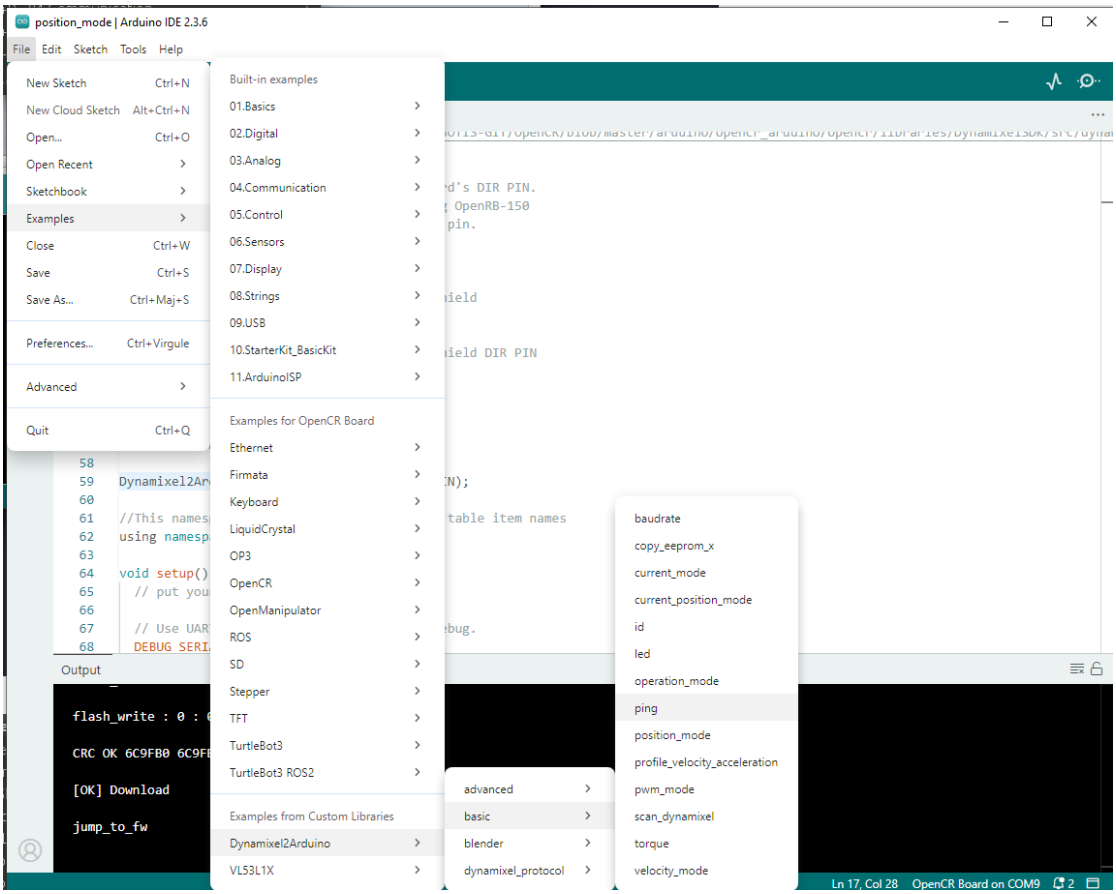


Utilisation de la bibliothèque Dynamixel2Arduino, à faire avec l'installation des moteurs.

A installer dans *librairy manager*



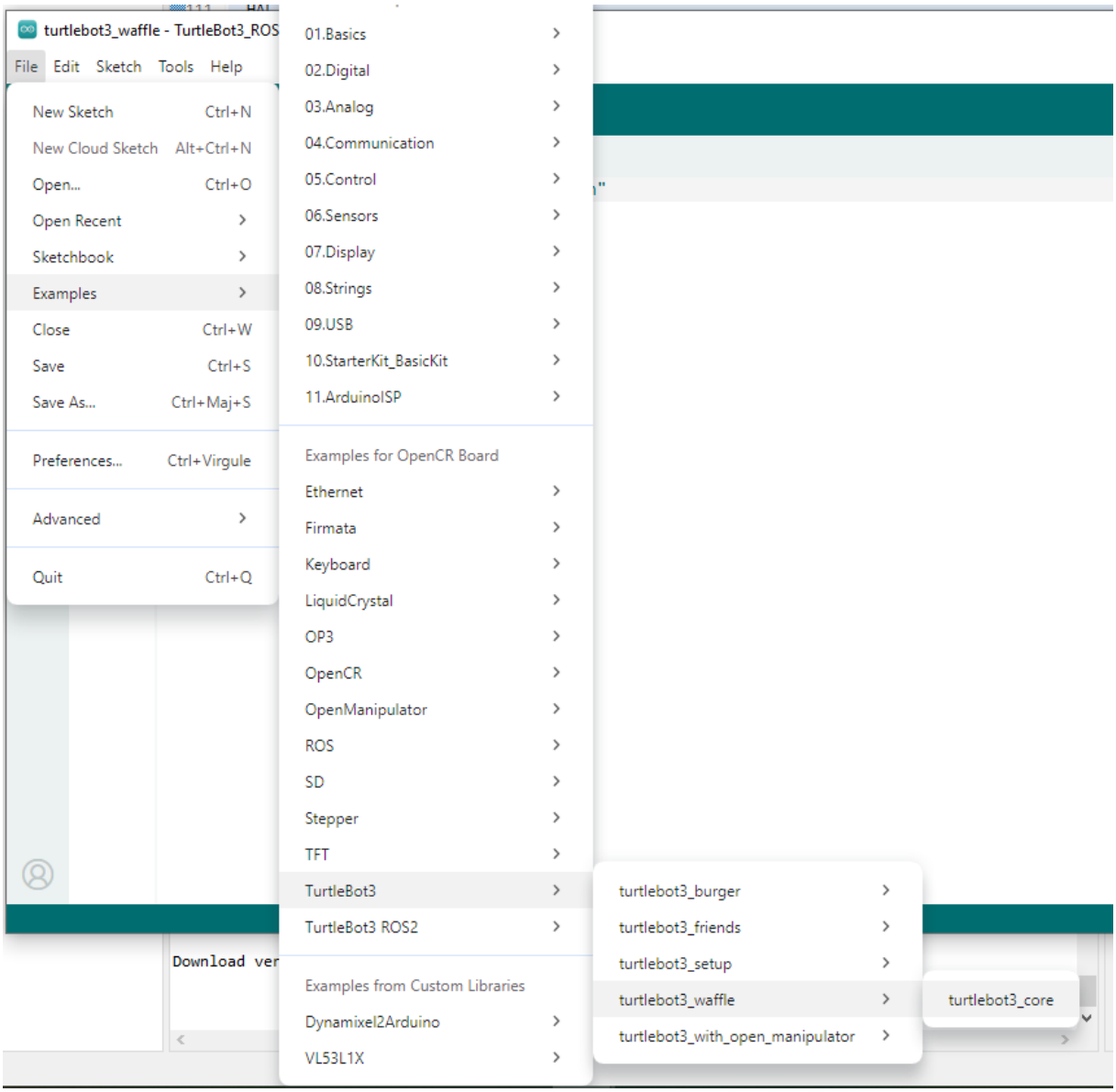
Utilisation des programmes exemples, ping ou autre pour vérifier fonctionnement des moteurs (bien pensé à changer le baud pour les moteurs Dynamixel)



Lancer le programme et vous devez voir normalement les deux moteurs avec les deux ID 1 et ID 2.

Lancer le programme *position mode*, pour avoir un aperçu simple de comment utiliser les moteurs simplement.

Pensez à bien remettre le programme de base TurtleBOT pour avoir la bonne alimentation sur les broches de l'OPENCR :



Pour faire avancer les moteurs à une certaine position et vitesse par défaut

Chaque roue a un codeur incrémental interne (≈ 4096 ticks/tour).
Tu peux calculer la distance :

- **Distance (m)** = (Δticks / 4096) × (périmètre roue).
- Périmètre roue ≈ 0.207 m (diamètre 66 mm).

Exemple lecture :

int32_t pos_left = dxl.getPresentPosition(DXL_ID_LEFT);

int32_t pos_right = dxl.getPresentPosition(DXL_ID_RIGHT);

Pour plus de précision sur les fonctions, une vidéo est disponible :

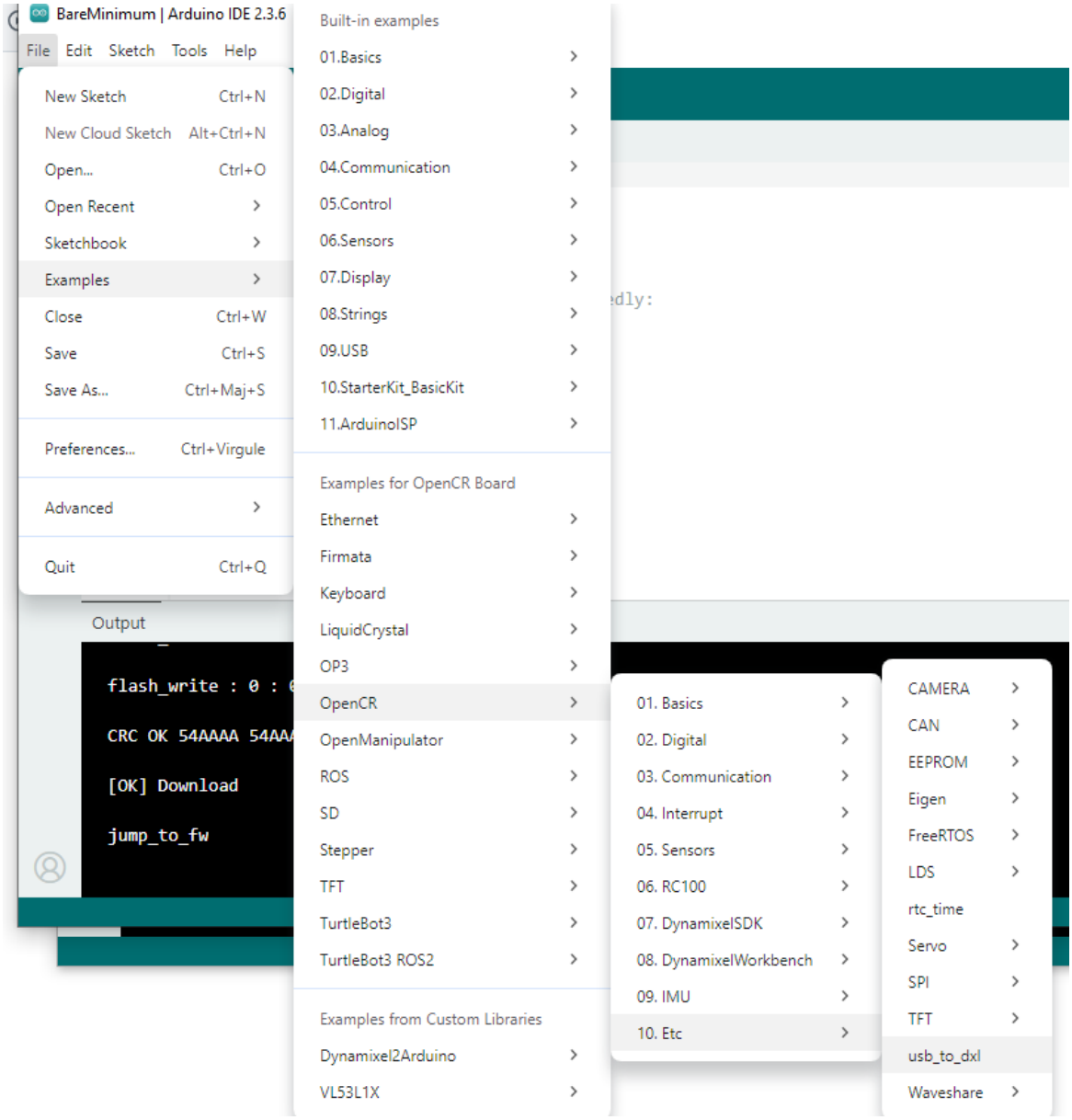
https://www.youtube.com/watch?v=iOBm5MJOLRo

Utilisation Pour STM32 Initialisation des moteurs

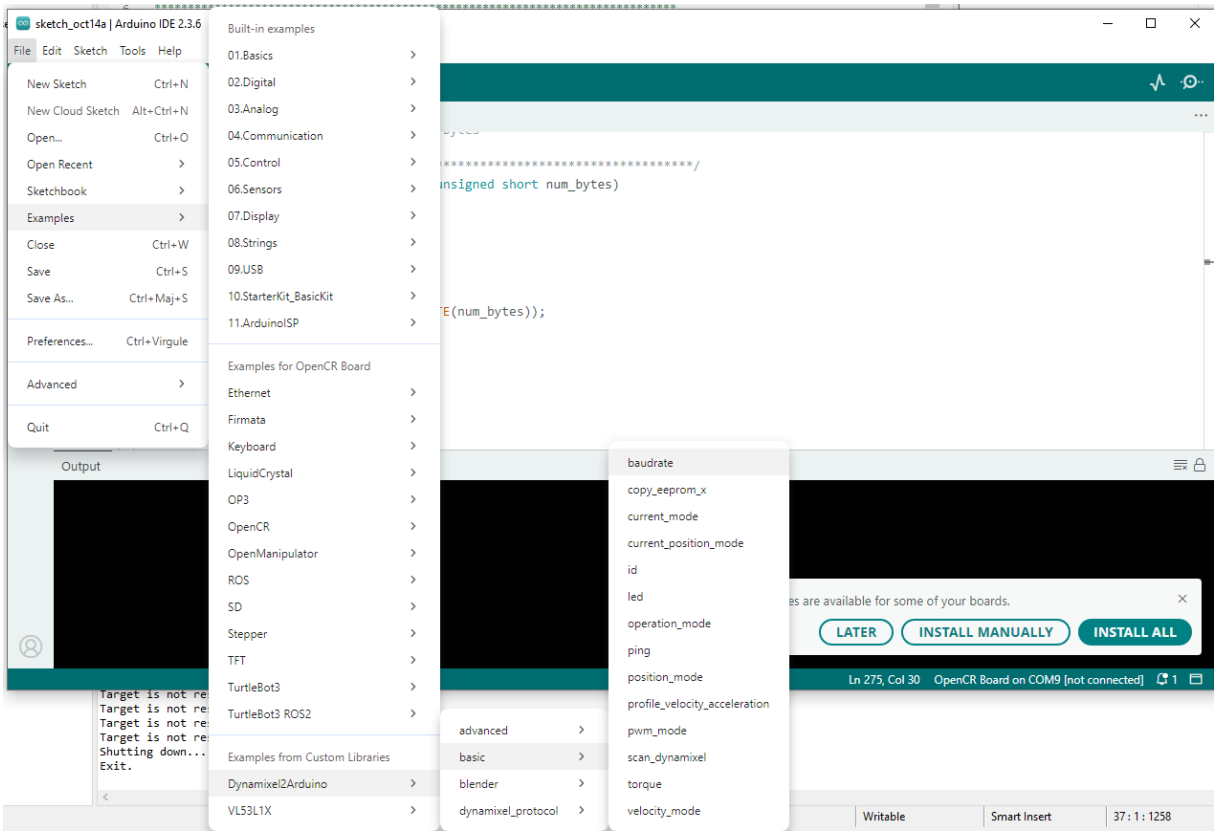
Si ce n'a pas été fait, brancher avec les moteurs avec la carte OPEN CR.

Autre méthode :

Installation de dynamixel Wizard 2 en USB soit avec un adapteur ou avec la carte OpenCR pour lier en USB et changer le baud rate des moteurs en 57600. Il est possible aussi d'exécuter le programme pour changer le BAUD. Si possible avec le logiciel dynamixel Wizards 2, mettre à jour les moteurs à la dernière version disponible.



Ou

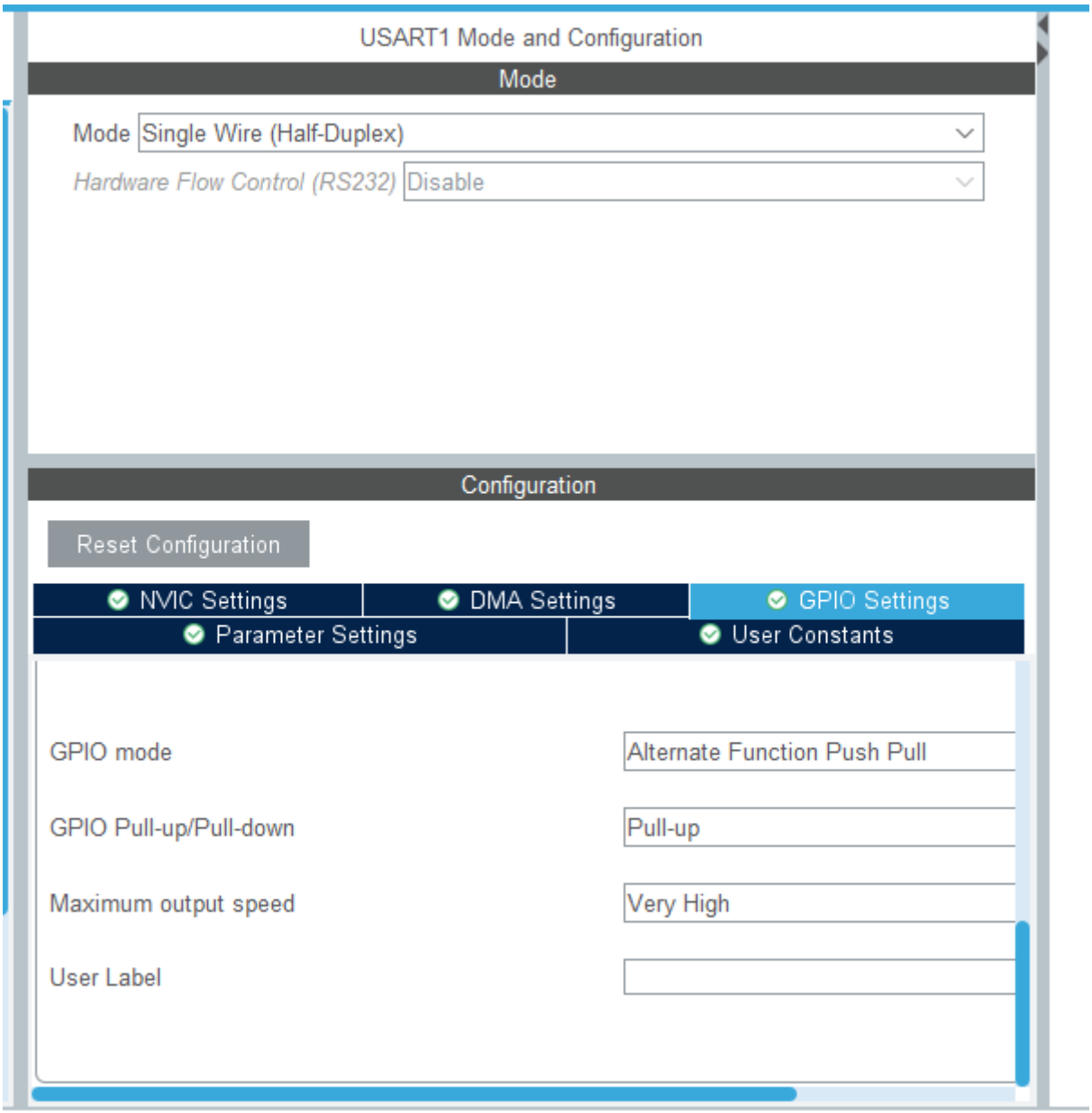


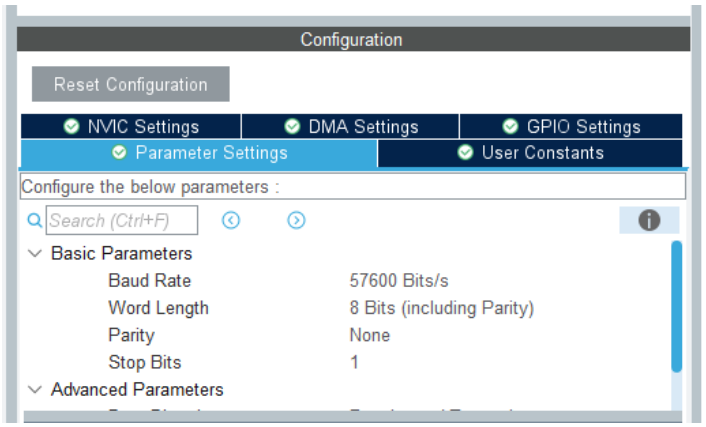
Pour éviter les problèmes de communication UART, mettre la vitesse BAUD des moteurs à 57600 !

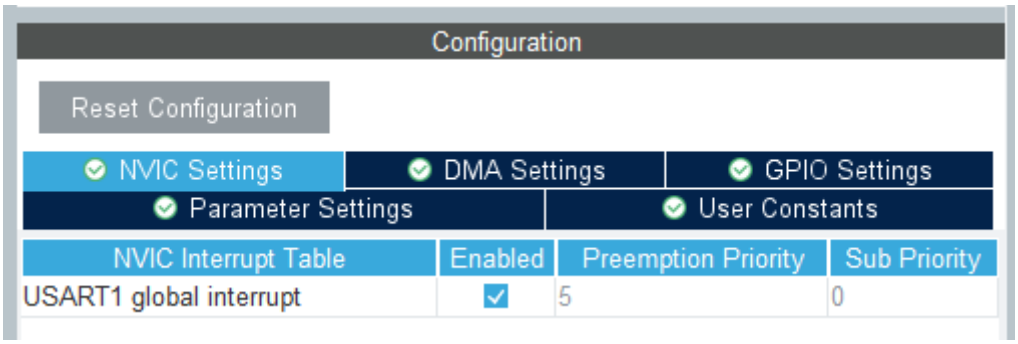
Une fois que les moteurs sont à la bonne vitesse, les erreurs de communication pourront être évitées avec la carte NUCLEO.

Utilisation avec une carte NUCLEO

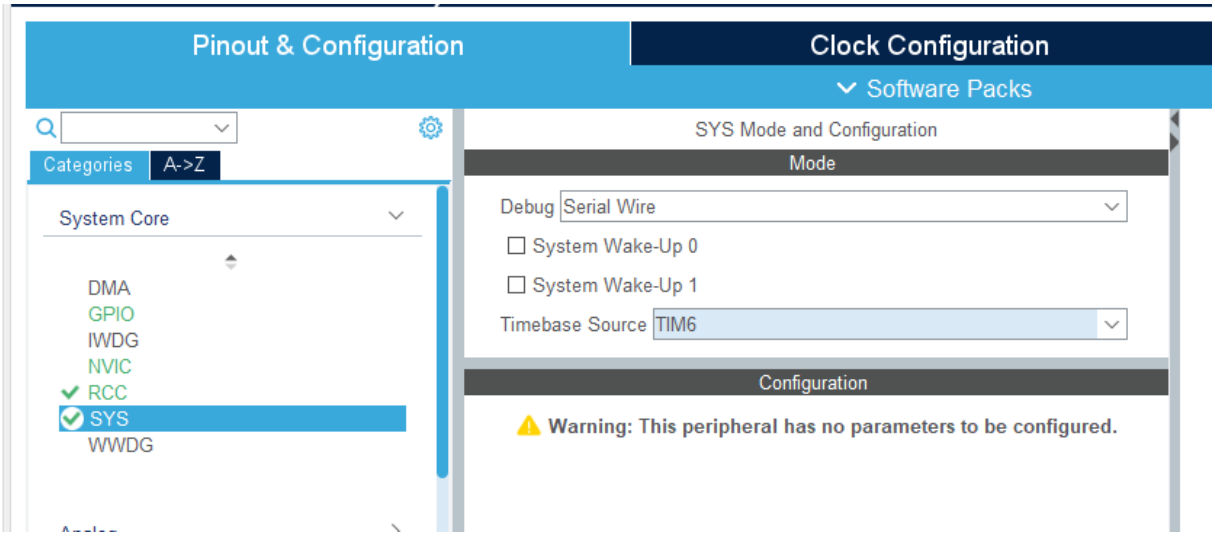
Créer un projet pour votre carte NUCLEO


USART1 Mode and Configuration
Mode
Mode
Single Wire (Half-Duplex)
Hardware Flow Control (RS232)
Disable
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
GPIO mode
Alternate Function Push Pull
GPIO Pull-up/Pull-down
Pull-up
Maximum output speed
Very High
User Label


Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Ctrl+F)
Basic Parameters
Baud Rate
57600 Bits/s
Word Length
8 Bits (including Parity)
Parity
None
Stop Bits
1
Advanced Parameters



Mettre la broche sur Interruption dans les paramètres et penser à bien utilisé la nouvelle horloge *quand on utilise le Middleware FreeRTOS* :

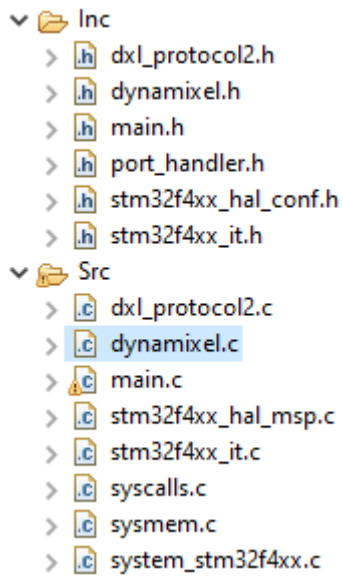


Voici les configurations du pin en TX pour USART 1 qui est relié au data du moteur

Mode Single Wire (Half-Duplex)

Baud rate : 57600 Bits/s

Voici l'arborescence du projet pour avoir la librairie pour le fonctionnement des moteurs



Pour installer cette librairie voici le lien :

https://github.com/Myrlon/STM32MotorDynamixelTurtleBot

Une fois la bibliothèque correctement installée prendre le fichier main.c exemple et upload sur la carte NUCLEO et tester le programme.

Pour les délais se trouvant entre chaque commande des deux moteurs dans le fichier main.c, il y'a l'air d'avoir besoin d'un temps pour faire fonctionner les moteurs « en même temps ». On met donc un délai qui peut possiblement être réduit, mais à 100ms cela semble fonctionner.

```
printf("Avancer\r\n");
Dxl_Move(1, FORWARD);
HAL_Delay(100);
Dxl_Move(2, FORWARD);
HAL_Delay(3000);
```

Exemple de fonctions pour faire avancer les moteurs

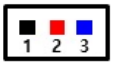
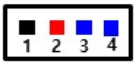
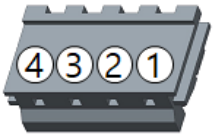
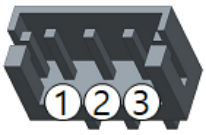
Montage des moteurs

Sachant que les moteurs se contrôlent en half-duplex, le pin doit être connecté au même moteurs, l'identification se fera dans la trame avec la sélection de l'ID des moteurs, un seul pin suffit donc.

Utilisé du 12V pour alimenter les moteurs, les alimentations de la carte nucléo ne suffiront pas.

Pensé bien à mettre la masse de la carte NUCLEO en commun avec la masse de l'alimentation 12V

4. 4. Connector Information

Item	TTL	RS-485
Pinout	1 GND 2 VDD 3 DATA	1 GND 2 VDD 3 DATA+ 4 DATA-
Diagram		
Housing	 JST EHR-03	 JST EHR-04
PCB Header	 JST B3B-EH-A	 JST B4B-EH-A
Crimp Terminal	JST SEH-001T-P0.6	JST SEH-001T-P0.6
Wire Gauge for DYNAMIXEL	21 AWG	21 AWG

Connector Information

Voici le connecteur du moteur qui est TTL, il y'a 3 broches GND, VDD (12V) et DATA (la broche de l'half-Duplex UART).

Bien vérifier le sens du connecteur avant de faire les branchements.

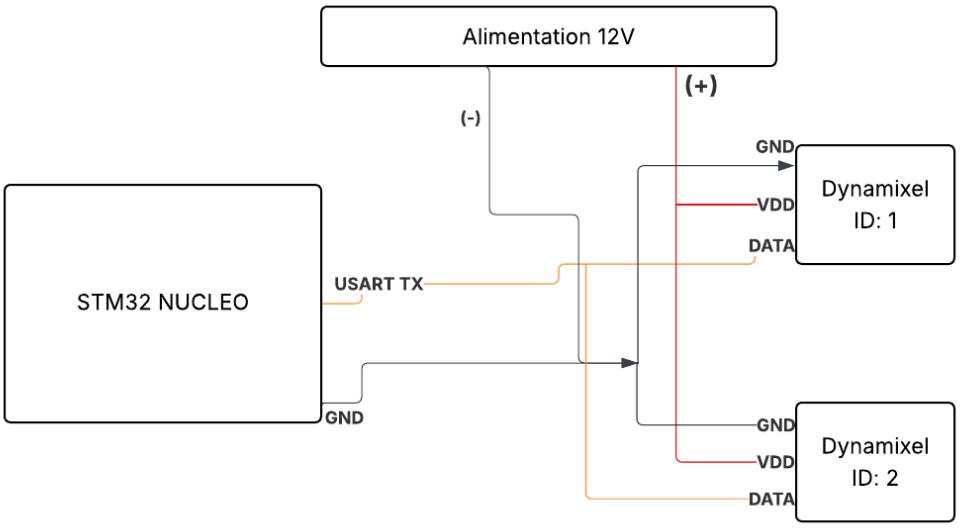


Schéma câblage du test des moteurs

Pour la connexion avec la batterie faire attention à ces étapes pour l'alimentation :

Ne pas brancher le câble USB avant d'avoir câbler l'alimentation externe et de l'avoir alimenté, et l'utilisé que pour debug et si pas beaucoup de courant est tirée.

Connect the jumper between pin 2 and pin 3 of JP5

Check that JP1 is removed

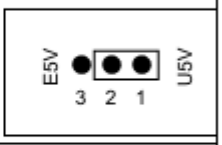
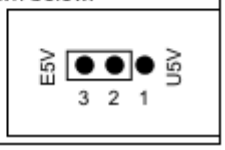
Connect the external power source to VIN or E5V

Power on the external power supply 7 V< VIN < 12 V to VIN, or 5 V for E5V

Check that LD3 is turned ON

Connect the PC to USB connector CN1

Table 8. Power-related jumper

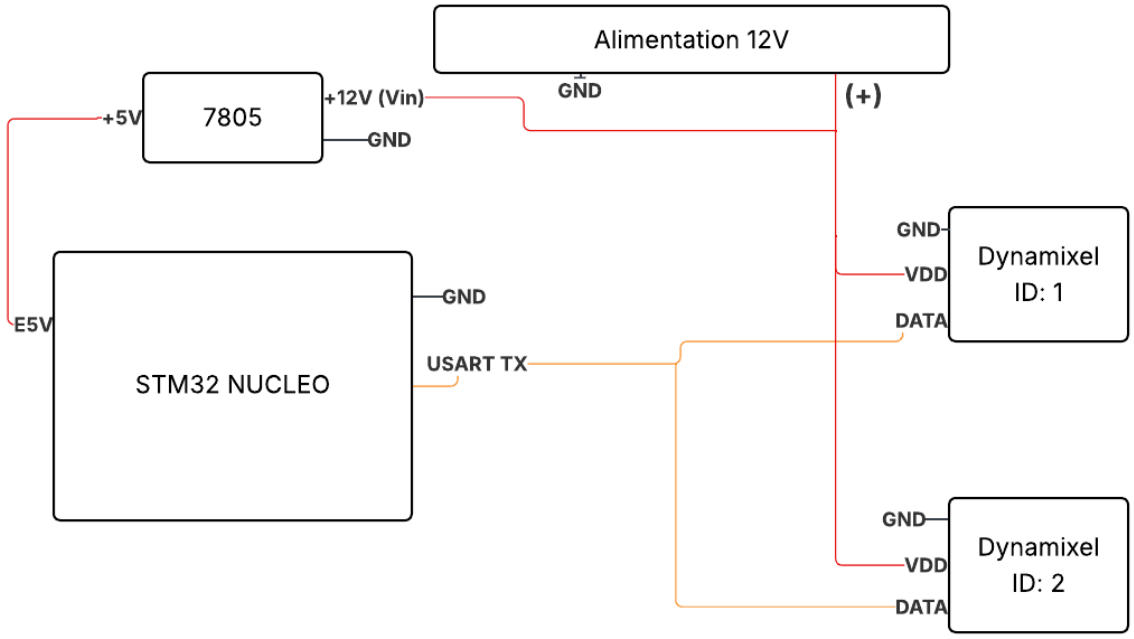
Jumper	Description
JP5	U5V (ST-LINK VBUS) is used as a power source when JP5 is set as shown below (Default setting) 
	VIN or E5V is used as a power source when JP5 is set as shown below. 

Pensez à vérifier sur la doc de la carte NUCLEO concernée

Pensez à remettre le Jumper à sa place quand l'alimentation externe n'est pas utilisée et si vous voulez Debug sans alimentations externe.

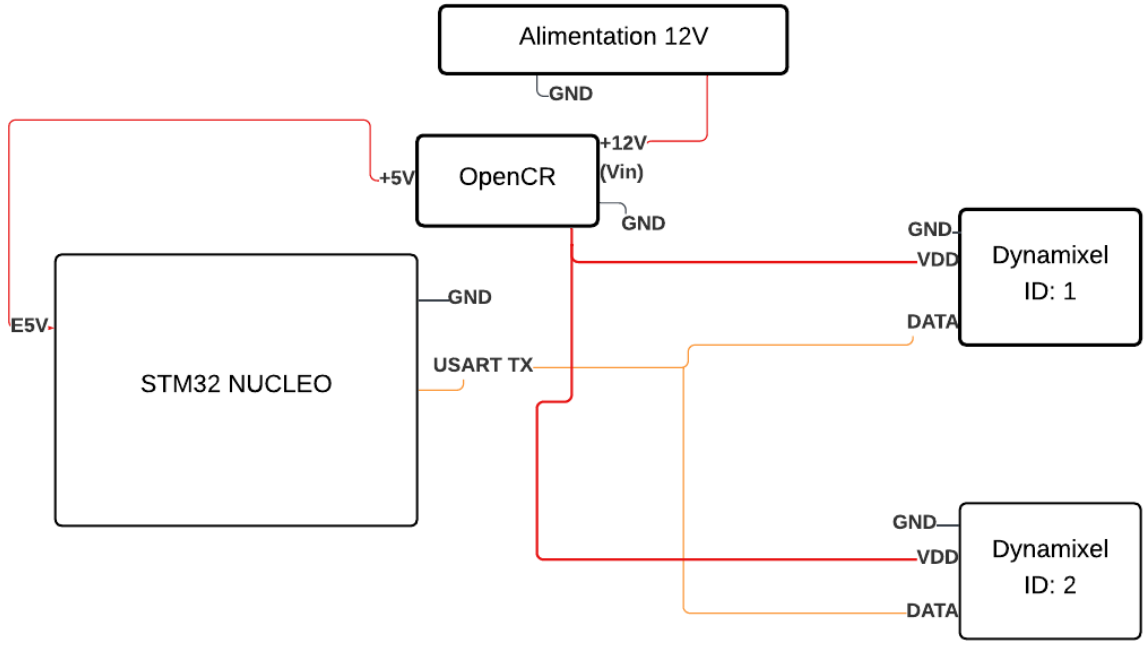
Montage avec batterie

Pour utilisation avec une batterie, c'est possible à condition de bien alimenter le montage, pour pas qu'il n'y a des pertes de puissance et un manque de courant à délivrer.
Pour cela en se référant à la doc de la carte NUCLEO il faut donc utiliser du 5V sur le pin E5V pour avoir assez de courant pour contrôler les moteurs en alimentant la carte NUCLEO avec la batterie. On a une batterie de 12V il faut donc un régulateur de tension, on peut utiliser le régulateur 7805 pour avoir notre tension de 5V.



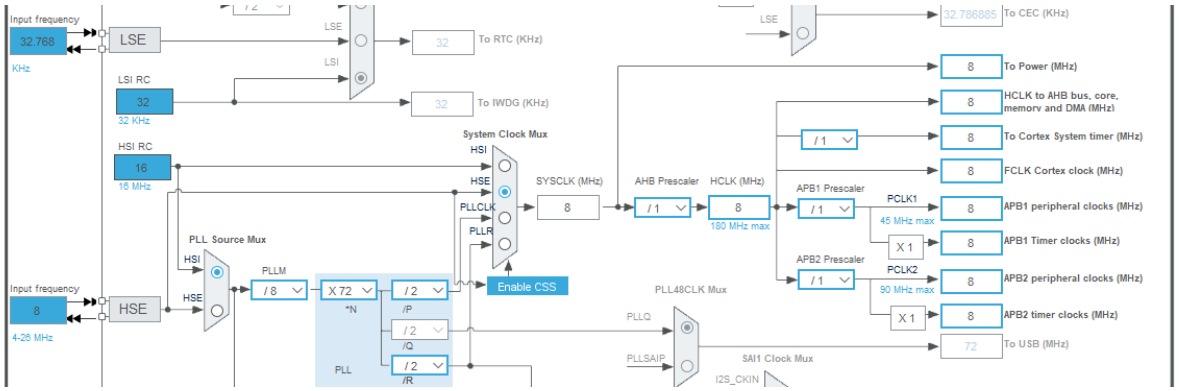
Montage avec batterie de la carte nucléo et des moteurs

Voici aussi le montage pour utiliser la carte OpenCR pour brancher la batterie et faire les branchements pour les moteurs.



Configuration Horloge

Pour passer avec la version avec FreeRtos, voici la configuration qui semble fonctionnelle



Ce qu'il y'a d'important dans le programme quand on veut écrire ou lire des infos des moteurs dynamixel est de penser à activer la réception ou la transmission sur la broche du half-duplex avant chaque opération.

Exemple ci-dessous pour activer la transmission et ensuite faire un Transmit sur les moteurs d'une commande. Il est utile d'activer la réception juste après pour recevoir la réponse.

Tout cela peut être organisé dans une fonction pour que tout soit automatique et ne pas activé la réception et transmission à chaque fois. Ce qui est le cas dans la librairie.

```
HAL_HalfDuplex_EnableTransmitter(&huart1);
HAL_UART_Transmit(&huart1, test_pkt, sizeof(test_pkt), 100);
HAL_HalfDuplex_EnableReceiver(&huart1);
```

Exemple d'activation de la transmission et réception pour une commande

La données reçus est dans le buffer de la fonction Callback de l'interruption. Fonction à mettre généralement dans le USER CODE 4 sur stm32cubeide.

```
void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size){
    printf("bon");
    HAL_UARTEx_ReceiveToIdle_IT(&huart1, rx, sizeof(rx));
}
```

C'est dans cette fonction qu'on peut ensuite traiter cette donnée reçue.

Utilisation d'une valeur *extern* pour utiliser partout la variable reçue du Rx définie dans le fichier :

dxl_protocol2.h

Voici la variable

```
//variable du buffer de lexture du Rx
extern volatile uint8_t rx[64];
```

et dans le main.c

```
volatile uint8_t rx[64];
```

Pour toute la configuration des moteurs : lecture et écriture, on peut retrouvez tous les éléments à ces adresses :

Table de control Dynamixel

Pour ce qui est de la vitesse de configuration des moteurs, regarder les fonctions *Goal Velocity* et *Velocity limit*.

Attention à ne pas faire fonctionner les moteurs à trop grande vitesse et trop longtemps pour ne pas les chauffer et les endommager.

Fonctions Moteurs

La fonction *Dxl_SetOperatingMode* permet de choisir le mode des moteurs conformément à la doc Le mode 1 pour le mode vitesse et le mode 3 pour le mode de position, selon le mode, les fonctions utilisées sont différentes et donc le mode de déplacement aussi. Pour le mode 1, mode vitesse on utilise *Dxl_MoveVel()* pour choisir une vitesse fixe constante. Et le mode 3, ***Dxl_MovePos****()* pour choisir le mode de position du moteur ; voir la doc pour plus d'informations.

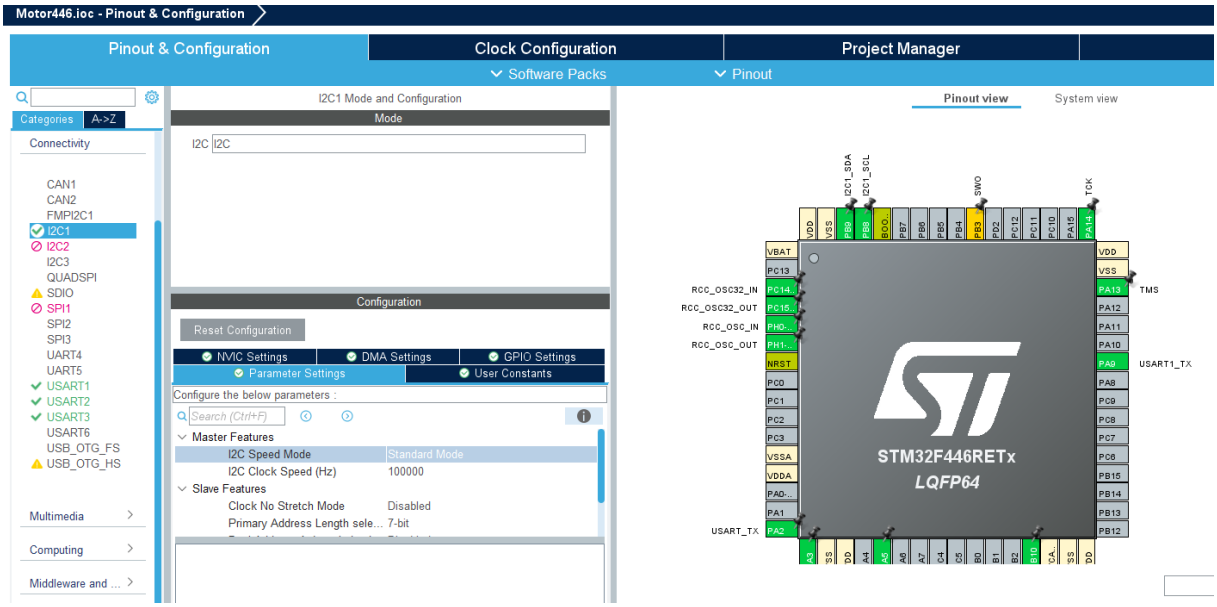
Voici l'ensemble des fonctions qui peuvent être utilisé et leurs paramètres :

`HAL_StatusTypeDef Dxl_Ping(uint8_t id);`	Pour Ping avec le bon id et vérifier que le bon moteur est bien disponible
`HAL_StatusTypeDef Dxl_TorqueEnable(uint8_t id, uint8_t enable);`	Permet d'activer le couple du moteur avec l'id indiqué du moteur et 0 ou 1 pour activer ou désactiver le couple. Sans le couple le moteur ne pourra pas tourner avec les autres fonctions
`HAL_StatusTypeDef Dxl_MovePos(uint8_t id, uint32_t position);`	Fonction à utiliser avec le mode 3 (*Position Control Mode*) Pour faire fonctionner le moteur avec des positions
`HAL_StatusTypeDef Dxl_MoveVel(uint8_t id, uint32_t speed);`	Fonction pour utiliser le moteur avec une vitesse définie
`HAL_StatusTypeDef Dxl_SetOperatingMode(uint8_t id, uint8_t mode);`	Mode de fonction de moteur à définir (mode 3 par défaut) doc
`HAL_StatusTypeDef Dxl_SetProfileVelocity(uint8_t id, uint32_t velocity);`	Fonction pour définir une vitesse moyenne du moteur
`HAL_StatusTypeDef Dxl_ReadVel(uint8_t id, uint32_t *vel);`	Lecture de la vitesse avec un pointer en paramètres
`HAL_StatusTypeDef Dxl_ReadPresentPosition(uint8_t id, uint32_t *position);`	Lecture de la position du moteur

Centrale Inertielle

L'IMU utilisé est normalement le MPU6050, l'adressage et les fonctions sont simples pour lire les données du capteur, mais on préférera utiliser une bibliothèque pour simplifier la lisibilité du programme. Voici le lien de la librairie pour la centrale Inertielle : https://github.com/svenikea/MPU6050

Pour la configuration de I2C pour la carte Nucléo :



Bien alimenté dans l'alimentation du composant qui est 3V3.

Voila une idée du montage pour la centrale Inertielle :