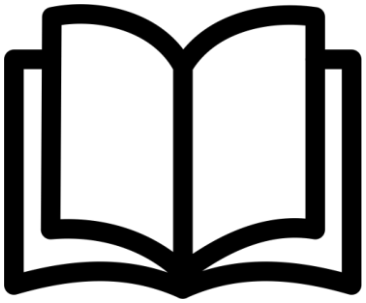
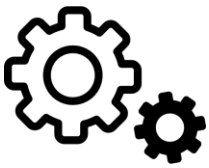
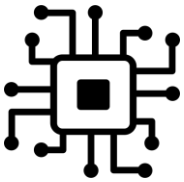
Manuel d’utilisation du système FuniBot



Date de création : 22 février 2021

Numéro de version : 1.0.0





PRÉPARATION MÉCANIQUE

Piliers

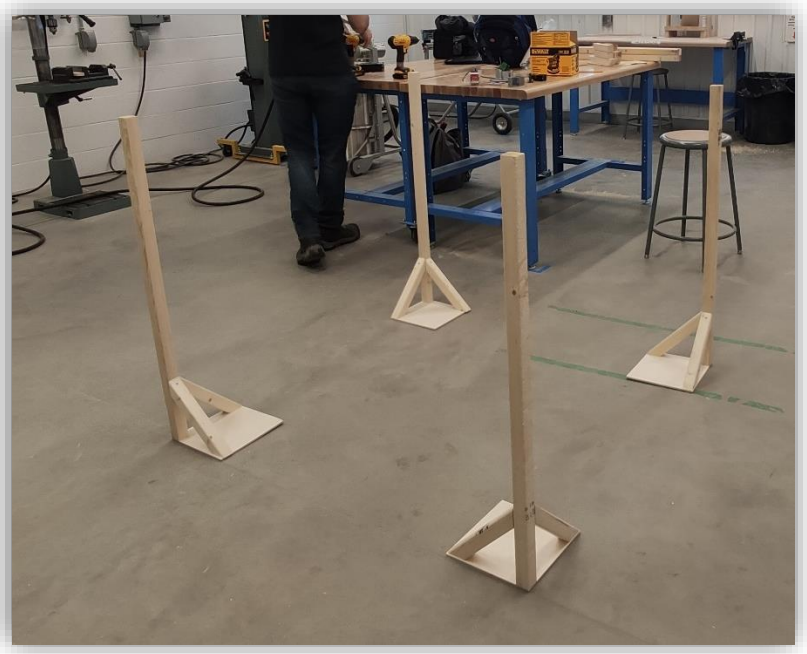


Figure 1 : 4 piliers fabriqués selon la mise en plan

Matériel nécessaire pour 1 pilier :

- 1x poutre de 38 mm x 38 mm x 1150 mm (aussi appelé 2x2 au Canada et USA)
- 5x vis à bois n° 6 1 1/2"
- 1x plaque de bois de 238 mm x 238 mm x 5 mm

Plan d'assemblage :

1) Découper un pilier d'une longueur de 1000 mm dans la poutre de bois.
2) Découper les deux pièces qui supportent le pilier selon le schéma présenté à l'Annexe pilier.
3) Positionner le pilier dans le coin d'une plaque et percer sur une longueur de 30 mm la plaque et la base du pilier. Le trou devrait se situer au centre du pilier, dans le sens de la longueur. Voir le rectangle bleu de la Figure 2 comme guide au perçage.
4) Utiliser une vis à bois pour fixer le pilier à la plaque.
5) Positionner une pièce qui supporte le pilier sur la plaque, comme illustré à l'Annexe pilier.
6) Percer sur une longueur de 30 mm en se référant au rectangle rouge à la Figure 2.
7) Utiliser une vis à bois pour fixer une pièce qui supporte le pilier au pilier.

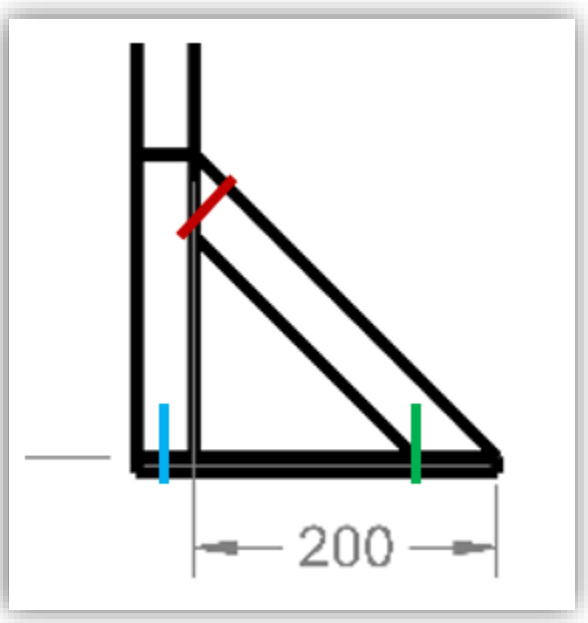


Figure 2 : Perçage du pilier

8) À l'aide d'une équerre, vérifier que le pilier est perpendiculaire à la plaque.
9) Percer la plaque et le support du pilier en se référant au rectangle vert sur la Figure 2.
10) Fixer les deux pièces à l'aide d'une vis.
11) Répéter les étapes 5) à 10) pour le deuxième support du pilier.

Détails supplémentaires :

Pour un montage en deux dimensions, deux piliers sont nécessaires.

Bobine

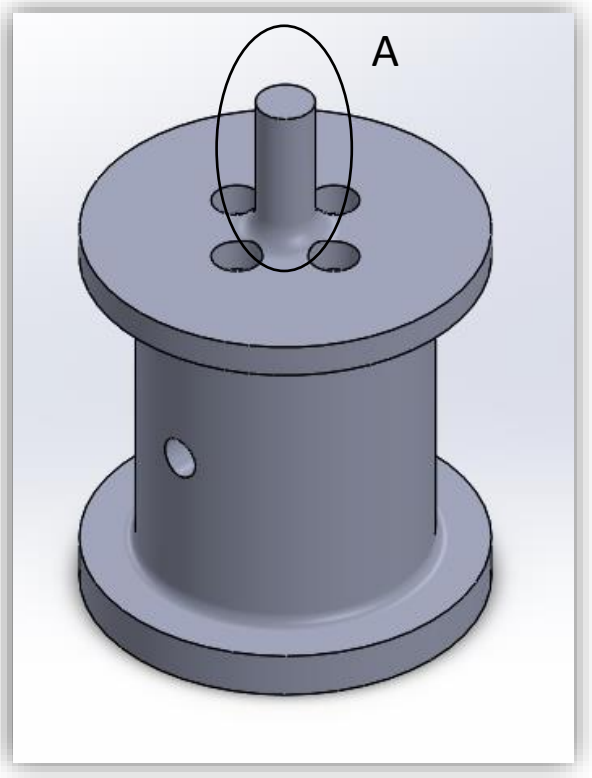


Figure 3 : Bobine imprimée en 3D

Matériel nécessaire pour 1 bobine :

- 4x vis n° 2 x 6mm

Plan d'assemblage :

1) Visser la base de la bobine à la partie tournante du moteur.
2) Entrer la zone A dans la zone B de la Figure .
3) Découper un câble de 2 mètres de long et enrouler la moitié autour de la bobine, en entrant le bout dans le trou au centre de la bobine, y faire un nœud.

Détails supplémentaires :

Pour un montage en deux dimensions, deux bobines sont nécessaires.

Support

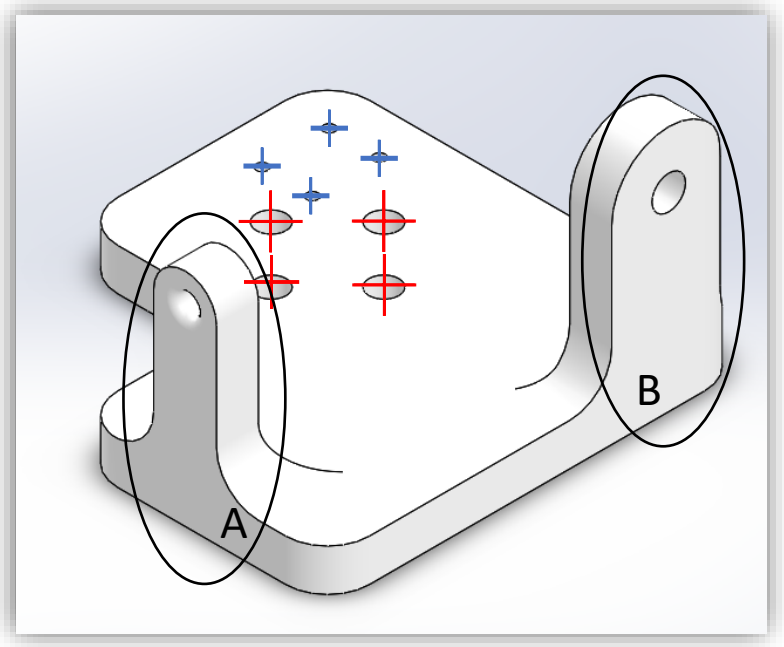


Figure 4 : Support du moteur imprimé en 3D

Matériel nécessaire pour 1 support :

- 4x vis à bois n° 6 1 1/2"
- Moteurs XM430-W350 et ses accessoires

Plan d'assemblage :

1) Visser le moteur par en dessous pour avoir la partie tournante du moteur face à la zone B.
2) Visser dans les 4 coins du bout du pilier à partir des 4 trous identifiés en rouge, de façon à avoir la zone A vers l'intérieur de la zone de travail.

Détails supplémentaires :

Pour un montage en deux dimensions, deux supports sont nécessaires.

Support du cadre

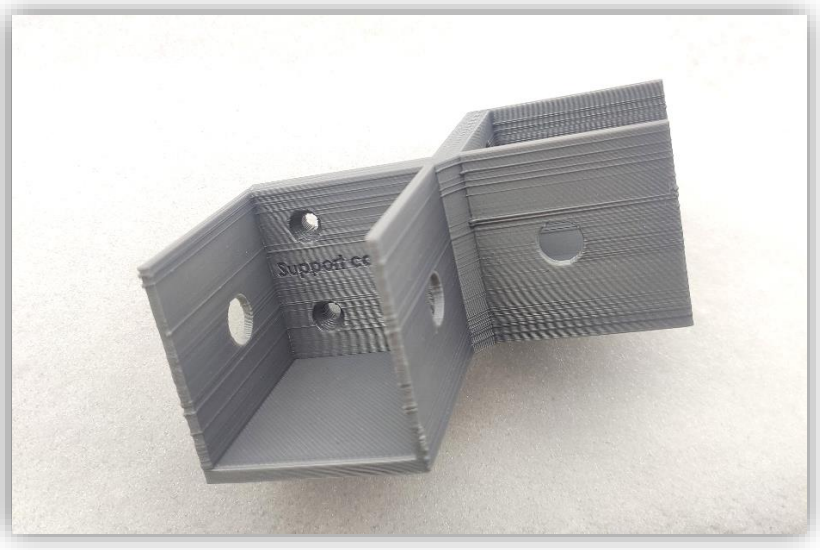


Figure 5 : Support du cadre imprimé en 3D

Matériel nécessaire pour 1 support du cadre :

- 4x vis à bois n° 6 3/4"

Plan d'assemblage :

1) À l'aide d'un ruban à mesurer, mesurer une distance verticale de 850 mm depuis le dessus de la plaque du pilier. Marquer d'un trait la distance sur le pilier.
2) Positionner le support du cadre sur le pilier au-dessus du trait rouge, représenté à la Figure 6. Percer le pilier au travers des trous du support du cadre.
3) Visser le support du cadre au pilier à l'aide des 4 vis.

Détails supplémentaires :

Pour un montage en deux dimensions, deux supports du cadre sont nécessaires.

Cadre

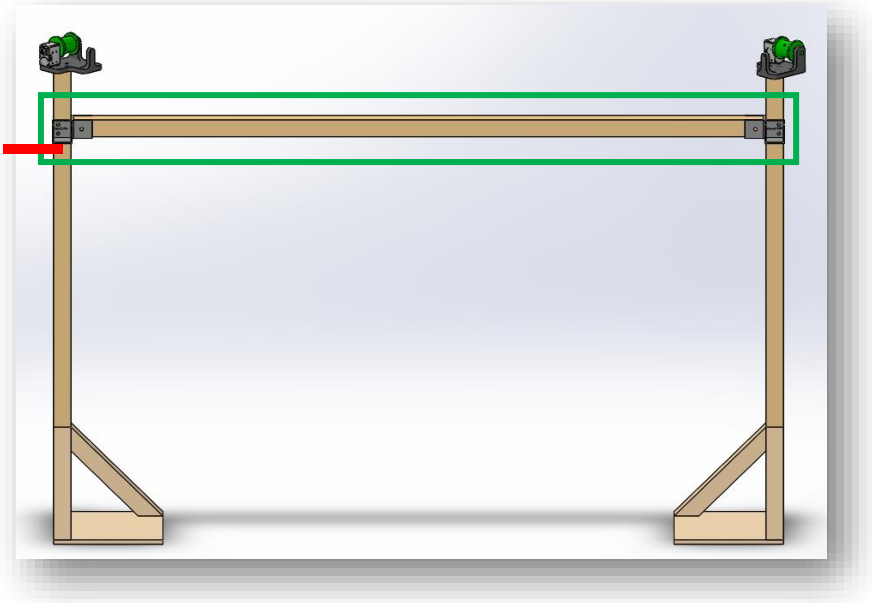


Figure 6 : Cadre de soutien entre les piliers

Matériel nécessaire pour 1 cadre :

- 2x boulons M6
- 2x écrous M6
- 1x poutre de 38 mm x 38 mm x 1219 mm (aussi appelé 2x2 au Canada et USA)

Plan d'assemblage :

1) Déposer la poutre dans les supports du cadre comme indiqué dans l'encadré vert à la Figure 6.
2) Percer la poutre au travers des trous des supports du cadre en utilisant une mèche du même diamètre que le trou.
3) Insérer les boulons dans les trous et les fixer au montage à l'aide des écrous.

Détails supplémentaires :

Pour un montage en deux dimensions, un seul cadre est nécessaire.

PRÉPARATION ÉLECTRIQUE

Alimentation microcontrôleur

Pour faire fonctionner le microcontrôleur afin de lui transmettre le code FuniBot (voir p.11), il suffit d'alimenter le OpenCR par le port micro USB relié à un ordinateur (voir point A de la Figure 7

Erreur ! Source du renvoi introuvable.).

Pour faire fonctionner les moteurs et le microcontrôleur, il suffit de brancher le OpenCR avec une alimentation de 12 V et de placer l'interrupteur en position « *On* » *(*voir point B et C de la Figure 7). Il est possible d'utiliser le bloc d'alimentation fournit avec l'achat du microcontrôleur.

Alimentation des moteurs

Pour alimenter les moteurs, il suffit de les brancher dans un des ports de communication TTL du OpenCR (voir point D de la

).

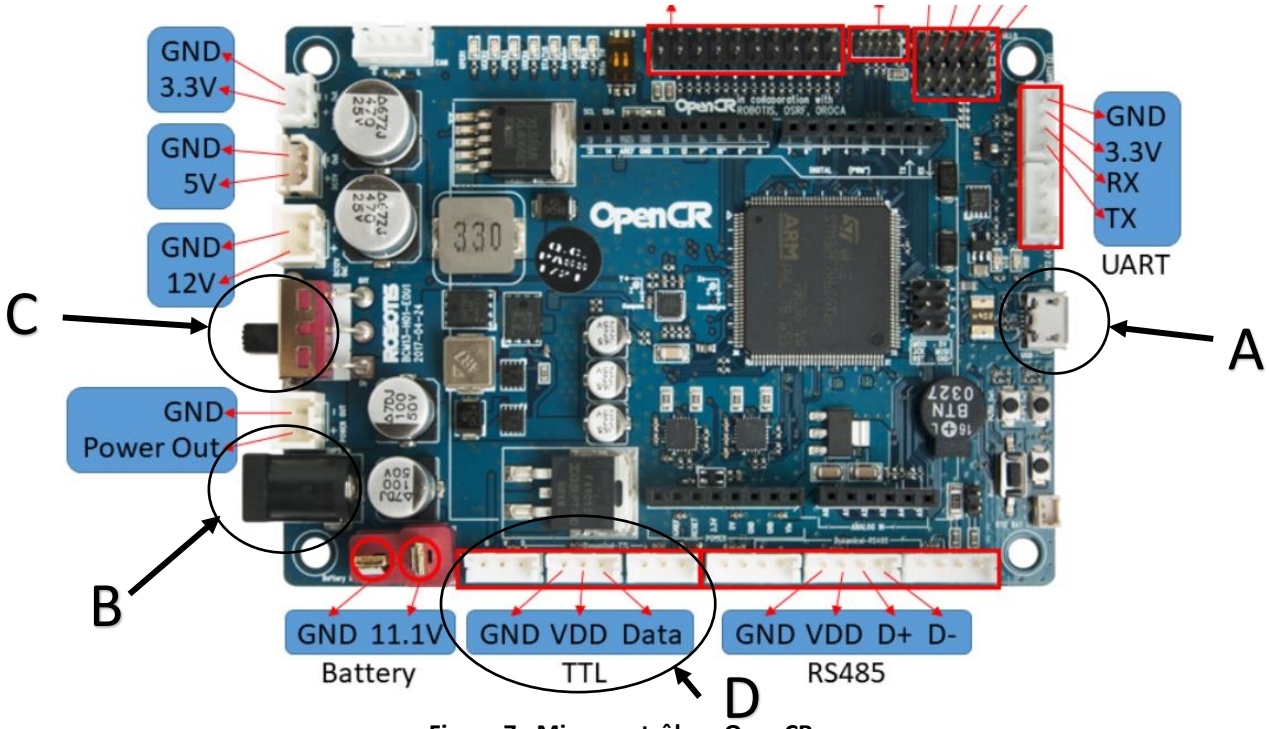


Figure 7 : Microcontrôleur OpenCR

Pour alimenter plus d'un moteur à la fois, il est possible de les connecter en série lorsque les trois ports de communication TTL sont utilisés, comme illustré à la Figure 8.

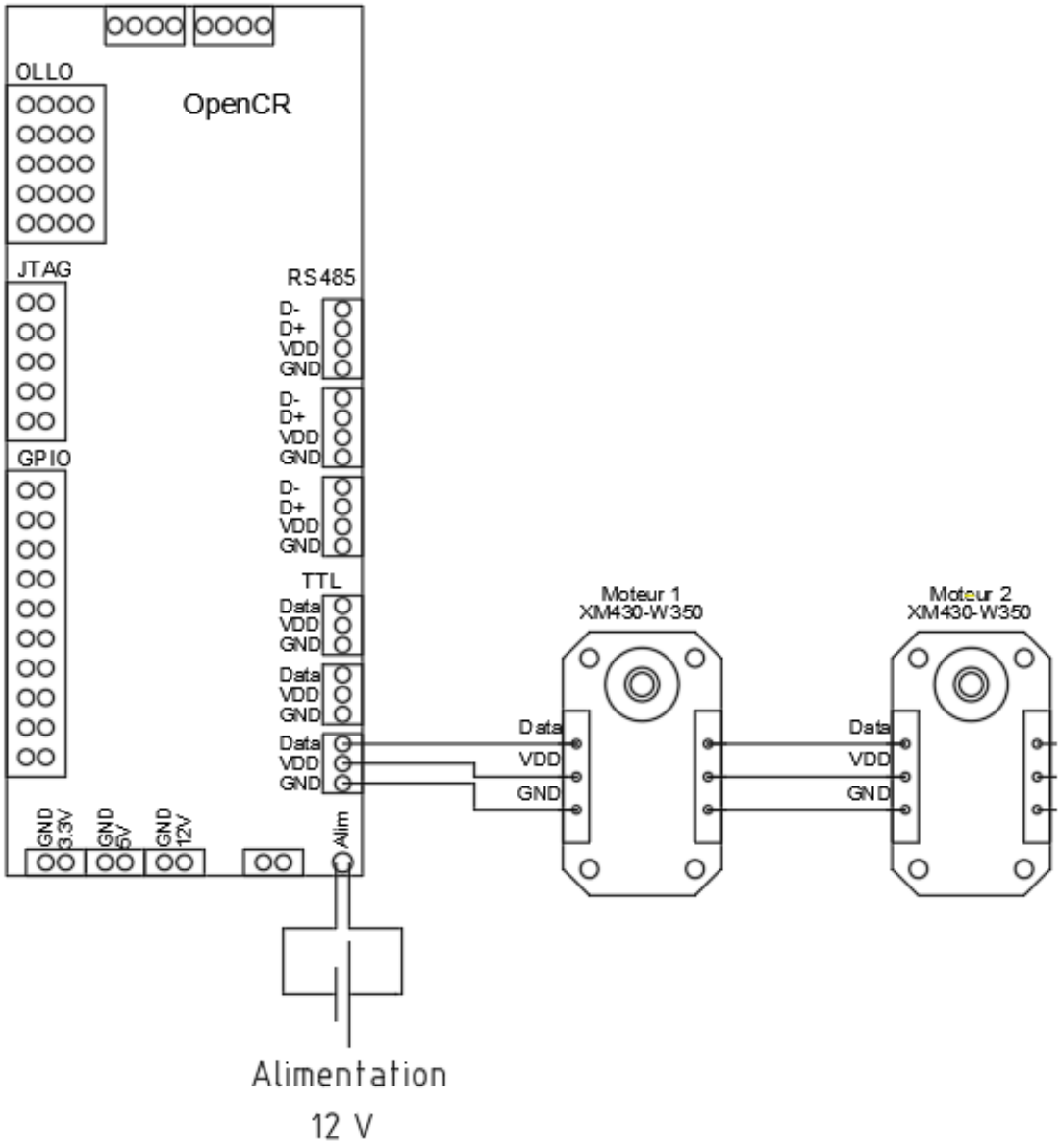


Figure 8 : Exemple d'alimentation de moteurs en série

PRÉPARATION LOGICIELLE

Préparation de l'environnement de développement

Git-lfs

Avant de cloner le répertoire git, il faut installer le logiciel git-lfs, puis l'initialiser. Pour cela, il suffit de lancer la commande suivante une fois l'installation terminée.

```
> git lfs install
```

Git-lfs est disponible au téléchargement ici.

Alternative : télécharger une livraison

Si vous souhaitez utiliser une version stable plutôt que la version de développement, vous pouvez télécharger la dernière version stable (ou une version antérieure) à partir de ce lien. Assumez que lorsque ce guide parle de la racine du répertoire, vous devez travailler dans le dossier correspondant une fois l'archive décompressée.

Python et pip

Assurez-vous d'avoir Python 3.5 ou une version plus récente d'installée, ainsi que « pip » et « venv » pour cette version de Python.

Si vous installez Python sur Windows, « pip » devrait être inclus. Vous pouvez ensuite installer « venv » avec la commande suivante :

```
> python -m pip install -U venv
```

Sur Linux, ils peuvent être dans des « package » différents, comme par exemple, sur Ubuntu. Essayer d'invoquer « python -m pip » ou « python -m venv » vous indiquera le nom de ces « package ».

Environnement virtuel et dépendances du projet

La suite doit être accomplie à partir de la racine du répertoire git.

Si votre Python est accessible par la commande « python », vous pouvez lancer le script Python suivant pour préparer votre espace de travail :

```
> python pip_script.py
```

Ce script crée un environnement virtuel et y installe les dépendances du projet. Si votre Python n'est pas accessible via la commande « python » (par exemple, c'est « python3 » sur Ubuntu 18.04, ou « py » sur Windows lors d'une installation via le Microsoft Store), vous pouvez faire cette action manuellement avec la suite de commandes suivante, en remplaçant « python » par votre commande Python :

```
> python -m pip install -U pip
```

```
> python -m venv .venv
```

La commande suivante dépend de votre système ou de votre shell :

```
Windows CMD > %cd%\.venv\Scripts\activate.bat

Windows PowerShell > .\.venv\Scripts\Activate.ps1

Windows Bash (git – bash) > source ./.venv/Scripts/activate

Linux/Mac/BSD (bash, zsh, etc.) > source ./.venv/bin/activate
```

Cette commande est identique pour tous les shell :

```
> python -m pip install -r requirements.txt
```

Avant de développer et de tester le code Python, assurez-vous d'activer votre environnement virtuel avec l'une des quatre commandes ci-dessus, selon votre système, et ce, même si vous avez utilisé « pip_script.py ».

Pour valider la mise en place de l'environnement, déplacez-vous dans la racine du répertoire, activez l'environnement virtuel, et lancez les tests automatisés avec la commande suivante :

```
> pytest
```

Si les tests roulent et passent, votre environnement est prêt.

Si vous utiliser un IDE comme PyCharm, la création et l'activation d'un environnement virtuel peuvent être réalisés différemment ou automatiquement. Renseignez-vous pour votre IDE si vous le souhaitez.

Facultatif : Utiliser VSCode pour votre développement

Si vous souhaitez utiliser VSCode pour votre développement Python, voici une liste d'extensions recommandées pour vous faciliter la vie :

- **Python (Microsoft) -> Support pour Python.**
- **Pylance (Microsoft) -> Support amélioré pour Python, et analyse statique.**
- Python Auto Venv (Mattias Edlund) -> Active automatiquement l'environnement virtuel lors de l'ouverture d'un terminal dans VSCode.
- Visual Studio IntelliCode (Microsoft) -> Améliore IntelliSense, pour des meilleures suggestions automatiques, selon le contexte.
- Code Runner (Jun Han) -> Ajoute un bouton pour exécuter du code. Pas nécessaire pour Python, mais plus pratique que le bouton d'exécution intégré à l'extension Python. Il est possible de configurer l'extension pour qu'elle exécute le script dans le terminal intégré plutôt que dans l'onglet « Output », ce qui permet de taper des commandes pour un programme CLI interactif, par exemple.

Les extensions en gras sont fortement recommandées pour travailler avec Python. Les autres sont facultatives, mais apportent des fonctionnalités pratiques.

Notez que certaines extensions ont des dépendances et vous demanderont d'installer ces dépendances par une notification dans le coin inférieur droit de la fenêtre de VSCode lors de leur installation. Lorsque demandé, installez aussi ces dépendances.

Il est aussi possible de configurer des profils de débogage, en appuyant sur « F5 » lorsque le curseur est dans un script Python, puis en ajoutant des profils au fichier « launch.json ». Il est ainsi possible d'ajouter des paramètres d'entrée avec la clé « args ». Par exemple, voici un profil simple qui lance le fichier Python sélectionné, avec l'argument « -f config.yaml » :

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python avec config ",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "args": ["-f", config.yaml"],
            "console": "integratedTerminal"
        }
    ]
}
```

Utilisation de l'API

La documentation sur l'API est contenue dans ce fichier. Le module principal est le module « funibot ». Le module « funibot_json_serial » contient l'abstraction de la communication série, et est utilisé par le module Funibot. Un objet de ce module (FuniSerial) doit être construit manuellement, et passé au constructeur de l'objet principal Funibot. De plus, l'objet FuniSerial doit être construit avec un objet Serial, du « package » pip « serial ».

Utilisation de l'interface de tests CLI pour l'API

Le module « cli_demo.py » peut être lancé directement pour tester le FuniBot sans développer à l'aide de l'API. Il peut aussi servir de référence et d'exemple d'utilisation de l'API.

« cli_demo.py » prend un argument en ligne de commande, « -f », qui permet de spécifier le fichier de configuration (au format YAML) à utiliser. Ce fichier de configuration doit contenir certaines informations permettant d'initialiser partiellement le FuniBot.

Ce lien présente un exemple de fichier de configuration. Toutes les composantes des vecteurs sont en millimètres. Les noms ou identifiants des poteaux peuvent être choisi (dans l'exemple, il s'agit de « moteur1 » et « moteur4 », en fonction des identifiants internes des moteurs utilisés sur chacun des poteaux), mais les autres clés doivent être telles quelles.

Ce lien décrit les commandes utilisables dans le programme « cli_demo.py », et décortique aussi la structure du fichier de configuration.

Avant de tenter de déplacer le FuniBot, il faut calibrer le robot en position. Il suffit de mesurer la longueur de chacun des câbles, et de transmettre cette information au robot à l'aide de la commande « cable » telle que décrite dans le document précédemment mentionné. Après cela, le robot sera prêt à être déplacé en position.

Configuration du microcontrôleur

Le microcontrôleur utilisé est le OpenCR. Le guide d'utilisation de celui-ci est disponible en suivant ce lien.

Le téléchargement de la librairie ArduinoJson est requis pour faire fonctionner le code du microcontrôleur. La librairie est disponible en suivant ce lien. Elle est aussi disponible directement via le gestionnaire de librairies de Arduino IDE.

Ce code doit être envoyé dans le microcontrôleur.

Tous les fichiers de ce répertoire sont requis pour que le code fonctionne.

Configuration des moteurs

L'utilisation de deux moteurs Dynamixel XM430-W350 est requis.

L'ID des deux moteurs doivent être différents.

La fréquence de communication des deux moteurs doit être paramétrée à 4.5M baud.

Le guide d'utilisation des moteurs est disponible en suivant ce lien.

Le guide de dynamixel workbench est disponible en suivant ce lien.

Les ID des moteurs doivent être entrés dans ce fichier.

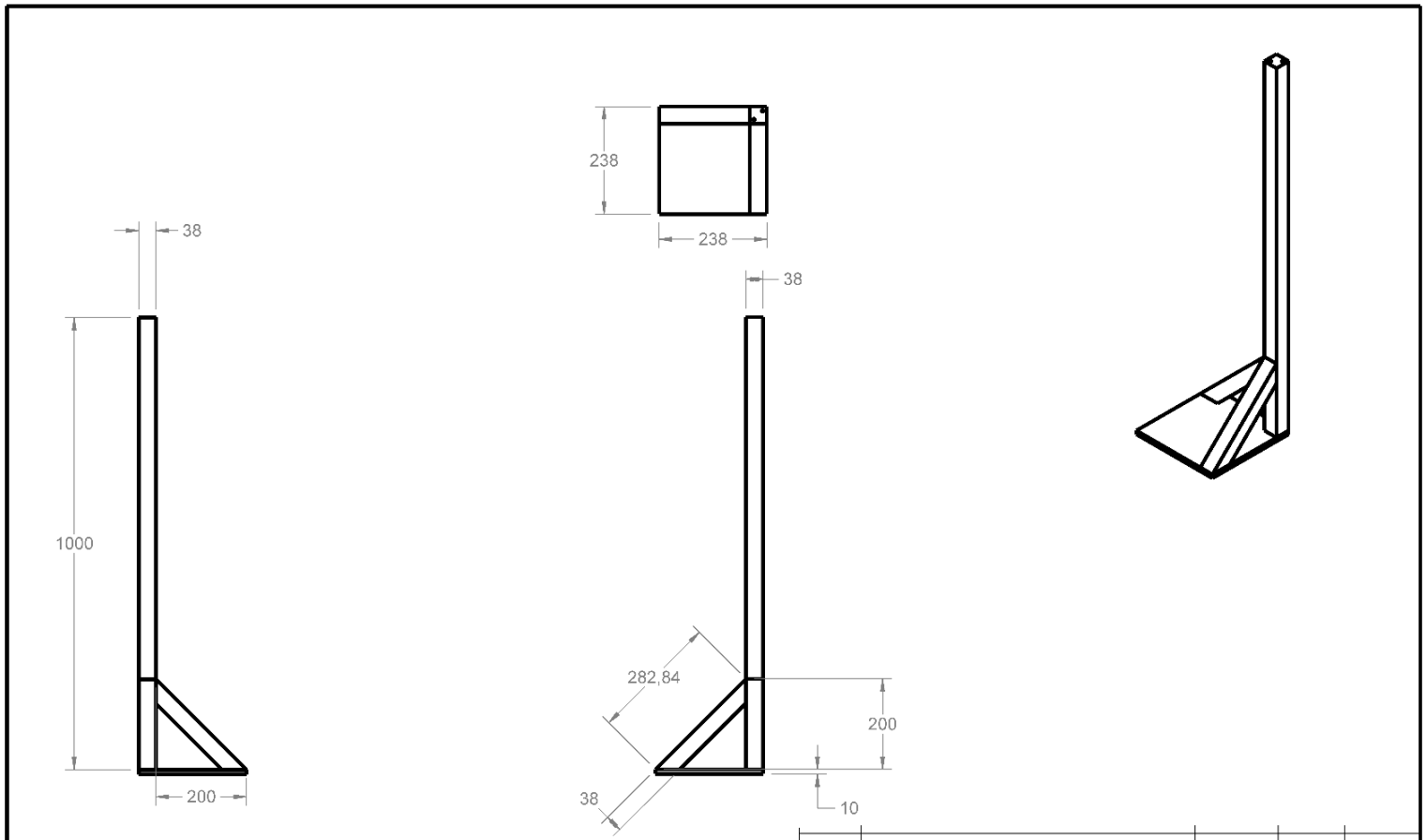
Les deux premières valeurs du tableau de la ligne 15 doivent être modifiées :

```
DynamixelWorkbench dxl_wb;

uint8_t liste_moteurs[4] = {3, 2, 4, 1};

void moteurSetup()
{
```

L'ID du moteur présent au pôle 0 doit être entré à l'index 0 du tableau listes_moteurs et l'ID du moteur présent au pôle 1 doit être entré à l'index 1 du même tableau.

Le pôle 0 est le premier pôle du fichier de configuration et le pôle 1 est le deuxième. Voir la documentation du programme CLI pour une description du fichier de configuration. De façon plus générale (en utilisant l'API directement), le pôle 0 est le premier envoyé par l'API, et le pôle 1 est le deuxième. Le programme CLI envoie les pôles dans le même ordre que leur apparition dans le fichier de configuration.

Annexe pilier



NO	RÉVISIONS	DATE	PAR	APP. PAR

NOTE PRÉLIMINAIRE:
TOUTES LES DIMENSIONS SONT EN MM

UNIVERSITÉ DE SHERBROOKE
FACULTÉ DE GÉNIE
DÉPARTEMENT GÉNIE ROBOTIQUE

TOLÉRANCES GÉNÉRALES (SAUF INDICATION CONTRAIRE)			
FRACTIONS ± 1/32"	NOM: WILLIAM BRUNEAU	DATE: 21 FÉVRIER 2021	PROJET: NÉOROBROOKE
1 DÉCIMALES ± .015	VÉRIFIÉ PAR: CAMILLE DURIVAGE	ÉCHELLE: 1:12 QTÉ: 4	TITRE: MISE EN PLAN ASSEMBLAGE_PILIER V2
2 DÉCIMALES ± .010	MATIÈRE: PLA		FORMAT: A NO DESSIN: ASSEMBLAGE_PILIER V2
3 DÉCIMALES ± .005			
ANGLE ± 1°			
FINI DE SURFACE:			