

Démonstrateur Balance

Instructions techniques

NIRYO

Table des matières

Ajouter une bibliothèque au programme Arduino.....	3
Téléverser le programme sur la carte Arduino.....	3
Lancer le programme sur Niryo Studio.....	3
Principe général de fonctionnement des programmes.....	3
Robots.....	3
Robot 1.....	3
Robot 2.....	3
Arduino.....	4
Balance.....	4
setup().....	4
loop().....	4
Round().....	4
Master.....	4
setup().....	5
initializeRobot1() et initializeRobot2().....	5
loop().....	5
readButtons().....	5
getBluetooth().....	5
moveRobot1(int button_pressed).....	5
getBalanceWeight().....	6
chooseBox(int weight).....	6
moveRobot2().....	6


Ajouter une bibliothèque au programme Arduino

Il faut d'abord ajouter les 4 bibliothèques (HX711, LiquidCrystal_I2C, PCF8574, et Wire) dans l'environnement Arduino. Vous trouverez les bibliothèques ici [Scale/program/arduino/data](#). Une fois dans l'environnement Arduino, cliquez sur Croquis puis sur Inclure une bibliothèque, et sur Ajouter la bibliothèque .ZIP... Cherchez la bibliothèque que vous voulez puis ajoutez-là à l'environnement Arduino.

Téléverser le programme sur la carte Arduino

Une fois le programme écrit, vous pouvez le téléverser sur la carte Arduino. Pour ce faire, vous devez d'abord indiquer à l'environnement Arduino, sur quel type de carte vous allez mettre le programme. Cliquer sur Outils, puis sur Type de carte, et choisissez la bonne carte. Pour le programme sur la balance, la carte est « Arduino/Genuino Uno », et pour le programme Master, la carte est « Arduino/Genuino Mega ou Mega 2560 ». Ensuite, vous devez spécifier sur quel port la carte est connectée. Cliquer sur Outils, puis Port et choisissez le port avec le nom de la carte – par exemple, COM3 (Arduino/Genuino Mega ou Mega 2560). Vous pouvez enfin cliquer sur le bouton de téléversement (celui avec une flèche).

Lancer le programme sur Niryo Studio

Lorsque les programmes sont sur leur carte, vous devez lancer les programmes sur les robots. Pour cela, lancez l'application Niryo Studio, connectez-vous au robot, cliquez sur cet icône () , et cliquez sur les deux flèches pour ajouter votre programme, cherchez le bon et ajoutez-le sur Niryo Studio. Vous pouvez ensuite cliquer sur la flèche verte pour démarrer le programme sur le robot.

Principe général de fonctionnement des programmes

Robots

Robot 1

On définit les pins du robot en tant qu'Input (pin 1A, 1B, 1C et 2A) et Output (pin 2B), puis on va sur la position de départ. On rentre ensuite dans la boucle.

On met le pin 2B à l'état bas, ce pin définit si le robot effectue une action ou non. Ensuite, on évalue les pins 1A, 1B, 1C et 2A. Si l'un des 4 pins est à l'état haut et que les 3 autres sont à l'état bas, on effectue le déplacement à la position définie (Position 1 = 1A/ Position 2 = 1B/ Position 3 = 1C/ Position 4 = 2A), on prend le poids, on le dépose sur la balance et on met le pin 2B à l'état haut.

Robot 2

La structure du programme du second robot ressemble beaucoup à celle du premier.

On définit les pins du robot en tant qu'Input (pin1A, 1B et 1C) et Output (pin 2A), puis on va sur la position de départ. On rentre ensuite dans la boucle.

On met le pin 2A à l'état bas, ce pin définit si le robot effectue une action ou non. Ensuite, on évalue les pins 1A, 1B et 1C. Si l'un des 3 pins est à l'état haut et que les 2 autres sont à l'état bas, on effectue le déplacement à la position définie (Box 1 = 1A/ Box 2 = 1B/ Box 3 = 1C), on prend le poids sur la balance, on le dépose dans le bac correspondant et on met le pin 2A à l'état bas.

Arduino

Balance

On commence par inclure les bibliothèques qui nous seront utiles (HX711, pour la balance, et LiquidCrystal_I2C, pour l'écran LCD). On définit ensuite les pins pour la balance et pour l'écran LCD. Puis on rentre dans la fonction setup().

setup()

On démarre le port série, qui sera principalement utilisé pour le debug. Puis on effectue un offset sur la balance pour que celle-ci affiche bien zéro lorsqu'il n'y a pas de poids dessus. Et enfin, on initialise l'écran LCD (on efface tout ce qu'il y a dessus, on active le rétroéclairage, on définit le nombre de lignes et de colonnes, on se positionne à un point de départ défini, et on écrit « Niryo 2019 », puis on efface tout après trois secondes).

Une fois les configurations terminées, on entre dans la boucle.

loop()

On commence par inspecter l'état du bouton sur la balance. Si celui-ci est appuyé, on tare la balance grâce à une fonction déjà définie dans la bibliothèque. Ensuite, on lit et on stocke dix valeurs, puis on effectue une moyenne de ces dix valeurs. Ensuite, on effectue un arrondi de cette valeur moyenne grâce à la fonction Round(float float_number).

Round()

On retourne la valeur flottante à laquelle on ajoute 0.5, casté en entier. Le cast permet de tronquer la valeur flottante, si elle est supérieure à 0.5, on renvoie l'entier supérieur.

Une fois la valeur arrondie, on affiche sur la balance le poids de l'objet.

Master

On commence par inclure les bibliothèques qui nous seront utiles (PCF8574, pour les expanders I2C, et Wire, pour la communication avec la balance grâce aux pins SDA et SCL). On définit ensuite les adresses des 3 expanders I2C. Puis on rentre dans la fonction setup().

setup()

On démarre le port série 1 qui sera utilisé pour le debug. On initialise ensuite les expanders I2C, ainsi que les deux robots avec les fonctions initializeRobot1() et initializeRobot2().

initializeRobot1() et initializeRobot2()

Mets tous les pins des robots à l'état bas.

Exemple : pcf8574_robot1.write(0, LOW); pour le pin 1A du robot 1.

Une fois les configurations terminées, on entre dans la boucle.

loop()

On démarre le port série 2 qui servira à la communication Bluetooth.

La suite du code est effectuée uniquement si les pins 2B, pour le robot 1, et 2A, pour le robot 2, sont à l'état bas. Ces pins représentent l'état des robots, s'ils sont à l'état bas, cela signifie que les robots n'effectuent aucune action et sont donc prêt à recevoir des commandes.

On inspecte d'abord l'état des boutons poussoirs grâce à la fonction readButtons().

readButtons()

Si un bouton est appuyé, et que les autres ne le sont pas, on retourne le chiffre correspondant au bouton appuyé (1, 2, 3 ou 4). Si aucun bouton n'est appuyé, on retourne le chiffre 0.

De retour dans la boucle, on regarde si le port série 2 est disponible, si c'est le cas, cela signifie que l'application Bluetooth est connectée. Dans ce cas, on rentre dans la fonction getBluetooth().

getBluetooth()

Si un bouton de l'application est appuyé, on retourne le chiffre correspondant au bouton appuyé (1, 2, 3 ou 4), via le port série 2.

À partir d'ici, le programme se divise en deux parties, celle avec les boutons physique d'un coté et celle avec l'application Bluetooth de l'autre, cependant le fonctionnement est le même. Si le chiffre retourné, via les boutons ou l'application est strictement supérieur à zéro, alors on passe en paramètre ce chiffre à la fonction moveRobot1(int button_pressed).

moveRobot1(int button_pressed)

En fonction du chiffre passé en paramètre, on met le pin du robot 1 correspondant à l'état haut.

Voici ci-dessous, le tableau montrant la correspondance entre les paramètres possible et les pins du robot 1 :

Paramètre	1	2	3	4
Pin à l'état Haut	1A	1B	1C	2A

Lorsque le pin correspondant est à l'état haut, le robot effectue son déplacement et on attend qu'il ait fini, en inspectant le pin 2B, et en attendant qu'il passe à l'état Haut. Une fois fait, le pin utilisé pour le déplacement (1A, 1B, 1C, 2A) repasse à l'état Bas pour arrêter d'envoyer la commande.

De retour dans la boucle, on récupère le poids mesuré par la balance grâce à la fonction `getBalanceWeight()`.

`getBalanceWeight()`

Cette fonction récupère simplement l'information de poids envoyé par la balance sur le port série 1, et retourne cette valeur.

On passe ensuite ce poids à la fonction `chooseBox(int weight)`.

`chooseBox(int weight)`

Le poids passé en paramètre définit le bac dans lequel le poids doit être placé. Si le poids est inférieur ou égal à 50 g, la fonction retourne le chiffre 1, s'il est supérieur à 50 g mais inférieur ou égal à 100 g, le programme retourne le chiffre 2, dans tous les autres cas, le programme retourne le chiffre 3.

On passe ensuite ce chiffre en paramètre de la fonction `moveRobot2(int box)`.

`moveRobot2()`

En fonction du chiffre passé en paramètre, on met le pin du robot 2 correspondant à l'état haut.

Voici ci-dessous, le tableau montrant la correspondance entre les paramètres possibles et les pins du robot 2 :

Paramètre	1	2	3
Pin à l'état Haut	1A	1B	1C

Lorsque le pin correspondant est à l'état haut, le robot effectue son déplacement et on attend qu'il ait fini, en inspectant le pin 2A, et en attendant qu'il passe à l'état Haut. Une fois fait, le pin utilisé pour le déplacement (1A, 1B, 1C) repasse à l'état 0 pour arrêter d'envoyer la commande.

Lorsque le robot 2 a fini son déplacement, la boucle retourne à son point de départ et attend qu'un bouton, physique ou sur l'application, soit appuyé.