

# **Turntable demonstrator**

Technical instructions



## Table of contents

Basic explanation of the programs.....	3
Robots.....	3
Robot 1.....	3
Robot 2.....	3
Arduino.....	4
Scale.....	4
setup().....	4
loop().....	4
Round().....	4
Master.....	4
setup().....	4
initializeRobot1() and initializeRobot2().....	4
InitializeMotor().....	5
loop().....	5
readButtons().....	5
getBluetooth().....	5
robot1Pick(int button_pressed).....	5
getBalanceWeight().....	6
chooseBox(int weight).....	6
moveRobot2().....	6

## Add library to the Arduino program


You first need to add the four libraries (HX711, LiquidCrystal\_I2C, PCF8574, and Wire) on the Arduino environment. You will find these libraries here [Turntable/program/arduino/datat](#). When you are in the Arduino environment, click on Sketch, then on Include Library, and finally an Add .zip Library... You can now search for the libraries you want and add them to your Arduino environment.

## Upload and launch the program on the Arduino board

Once the program is written, you can upload it to your Arduino board. To do so, you first need to tell to your Arduino environment on which type of board you will upload the program. Click on Tools, then Board, and search for the right board. For the scale program it is “Arduino/Genuino Uno”, and for the master program it is “Arduino/Genuino Mega or Mega 2560”. Then you need to specify on which Port your board is connected. Click on Tools, then on Port, and click on the one where there is the name of the board on it – for example, COM3 (Arduino/Genuino Mega or Mega 2560). You can finally click on the upload button (the one with an arrow).

## Launch the program on Niryo Studio

When programs are uploaded on their board, you need to launch and upload programs on Niryo Studio and on the robot.

Launch Niryo Studio, connect to the robot, click on this icon  then click on the two arrows icon to upload your code, search for the right program and upload it. Then click on the green arrow to start the program on the robot.

## Basic explanation of the programs

### Robots

#### Robot 1

We define the robot's pins as Input (pin 1A) or Output (pin 1B), then we move to the start position. We then enter the loop.

We put the pin 1B to state Low, this pin defines if the robot is doing something or not. Then, we check the state of pins 1A. If it is High, we move to the object position, we take the object, we put it on the scale and we put the pin 1B to state High.

#### Robot 2

The program's structure for the second robot is quite the same as the first one.

We define the robot's pins as Input (pin 1A, 1B and 1C) or Output (pin 2A), then we move to the start position. We then enter the loop.

We put the pin 2A to state Low, this pin defines if the robot is doing something or not. Then, we check the state of pins 1A, 1B and 1C. If one of these three pins is High, and the two others are Low, we move to the defined position (Box 1 = 1A/ Box 2 = 1B/ Box 3 = 1C), we take the object on the scale, we put it in the corresponding box and we put the pin 2A to state High.

## Arduino

### Scale

We begin by including the libraries that we will use (HX711, for the scale, and LiquidCrystal\_I2C, for the LCD screen). We then define the pins for the scale and the LCD screen. And we then enter the loop() function.

#### setup()

We start the serial port, which mainly be used for debugging. We then offset the scale so that it prints 0 g when there is nothing on it. Finally, we initialize the LCD screen (we clear the printing, we activate the backlight, we define the number of rows and columns, we set the cursor to the defined starting point, and we print « Niryo 2019 », we then clear everything after three seconds).

Once all setup done, we enter the loop() function.

#### loop()

We check the scale's push button state. If it is pressed, we tare the scale thanks to function defined in the library. Then, we read and keep in memory ten values, we then take the average of these ten values. Then, we round the average value thanks to the Round(float float\_number) function.

#### Round()

We take the average float value given by the scale and we cast it into an integer. Then, we look at the difference between the float and the integer, if it is superior or equal to 0.5, we increment the integer value and return it. If this is not the case, we just return the integer value.

Once we have a round value, we print on the scale the weight of the object.

### Master

We begin by including the libraries that we will use (PCF8574, for the I2C expanders, and Wire, for the communication with the scale thanks to pins SDA and SCL). We then define the three I2C expander's addresses. We then enter into the setup() function.

#### setup()

We start the serial 1 port, which mainly be used for debugging. We then initialize the I2C expanders and the two robots thanks to initializeRobot1() and initializeRobot2() functions.

#### initializeRobot1() and initializeRobot2()

Put all robot's pins to state Low.

*Example: `pcf8574_robot1.write(0, LOW);` for pin 1A of robot 1.*

And we initialize the motor with `initializeMotor()` function.

### **InitializeMotor()**

This function define the pins of the stepper motor and its driver (Direction, Step, Enable). The Enable pin is used turn on and off the robot to make it works only when we need it. So we begin by put the pin to state High, to turn off the motor.

Once all setup done, we enter the `loop()` function.

### **loop()**

We start the serial 2 port which will be used for Bluetooth communication.

The rest of the program is executed only if pins 1B, for robot 1, and 2A, for robot 2, are Low. These pins represent the state of the robots, if they are Low, it means that the robots are not doing anything so they are ready to receive commands.

We check the push buttons state thanks to the `readButtons()` function.

### **readButtons()**

If a push button is pressed, and the others are not, we return the corresponding number (0, 3, 6 ou 9). If none of the push button is pressed, we return the number -1.

Back in the `loop()` function, we check if the serial2 port is available, if it is, it means that the Bluetooth application is connected. In that case, we enter the `getBluetooth()` function.

### **getBluetooth()**

If one of the buttons on the application is pressed, we return the corresponding number(1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or 11), through the serial2 port.

From here, the program is divided in two parts, the one with physical push buttons on one hand and the Bluetooth application on the other hand, however the structure is the same.

If the number returned, from the physical push buttons or from the Bluetooth application, is strictly positive, we send this number to the `robot1Pick(int button_pressed)` function.

### **robot1Pick(int button\_pressed)**

According to the number given, we rotate the turntable to the good position. To do so, we rotate the motor x (the desired position given by the parameter) time the number of step corresponding to one position. We have twelve positions on the turntable, so we have an angle of 30 degrees between each position (360/12), the motor needs to execute 1600 steps to make a complete rotation, there is a ratio of 2/1 between the motor and the turntable, which means that to do a complete rotation with the turntable, we need to execute 3200 steps, so to go from one position to the next one, the motor needs to execute 107 steps (3200/30 round to the superior integer).

Example: If the parameter received is 3, we need to move to the fourth position of the turntable, so we move the motor of 3x107 steps.

We then put the pin 1A to state High to send the command and move the robot, then we wait until it is done, to know when the robot has finished his move, we check pin 1B, when it is High, the robot is done. Once it is, the pin used for the move (1A) goes Low to stop sending the command. We finally rotate the turntable in the opposite direction to go back to its initial state.

Back in the loop() function, we get the weight measured by the scale thanks to the getBalanceWeight() function.

### getBalanceWeight()

This function is used to get the weight of the object send by the scale through the serial1 port, and return this value.

We then send this value to the chooseBox(int weight) function.

### chooseBox(int weight)

The weight received defines the box in which the object must be put. If the weight is inferior or equal to 50 g, the function returns the number 1, if it is superior to 50 g but inferior or equal to 100 g, the function return the numbers 2, in all the other cases, the function returns the number 3.

We then send the returned number to moveRobot2(int box) function.

### moveRobot2()

According to the number given, we put the corresponding pin of the robot 2 to state High. Below, it is the table showing the correspondence between possible parameters robot 2 pins:

Parameter	1	2	3
Pin to state High	1A	1B	1C

When the corresponding pin is High, the robot move and we wait until it is done, to know when the robot has finished is move, we check pin 2A, when it is High, the robot is done. Once it is, the pin used for the move (1A, 1B, 1C) goes Low to stop sending the command.

When the robot 2 has made his move, the loop gets back to the starting point and wait for a button, physical or in the Bluetooth application, to be pressed.