Created by Minwoo Kim, Hankyu Jang, Jisoo Min and Sooryeon Lee.

Environment for moving a robot brick for learning to program in Python

##########################################################################

How to connect NXT to Computer with Python Bluetooth?

You need to

1. Download Python 2.7
2. Download nxt-python
3. Download Pybluez
4. Download cs1robot.py

Downloading nxt-python

* nxt-python is Python interface for programming the Lego Mindstorm NXT
* <http://code.google.com/p/nxt-python/downloads/detail?name=nxt-python-2.2.2.zip>
* Download and uncompress the file
* Double click install.bat in nxt-python-2.2.2 directory
* It invokes setup.py script in the same directory
* It copies the nxt package files to the proper python directory (Lib\site-package)

Downloading Pybluez

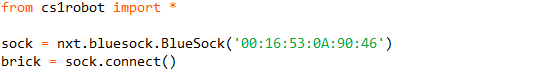
* https://code.google.com/p/pybluez/downloads/detail?name=PyBluez-0.20.win32-py2.7.exe&can=2&q=

Download cs1robot.py and set it to the proper Python directory (Lib\site-package)

##########################################################################

Connecting the brick

Import cs1robot.py and use function BlueSock with parameter of your robot’s Bluetooth address.



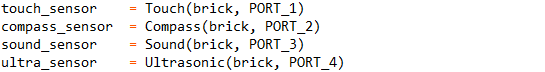
Connecting the motors to your brick

The ultra\_motor is the motor with the ultra\_sensor.



Connecting the sensors to your brick

Connect each sensors to your brick. The PORT numbers where you connect doesn’t matter.



Initializing the robot

The order of the motors and sensors are very important. Please initialize motors and sensors in this way. Then you can create a `hubo` instance of the Robot class with motors and sensors.



Play around with your robot hubo!



Here are the basic methods you can use

\*\*\* Sensors \*\*\*

Touch Sensor

* is\_pressed( ) : if pressed return True, else return False

Ultrasonic Sensor

* get\_distance( ): return distance (distance in centimeters range 0~254). If the distance cannot be read, return -1

Hitech Compass Sensor

* get\_heading() : return degree to north. (0 ~ 359)

\*\*\* Motors \*\*\*

* get\_tacho() : return the tachometer count
* reset\_position(relative) : reset the counters. Parameter relative = True or False
* run(power=100, regulated=False) : motor run continuously. If regulated=True, synchronization starts working. When calling on Synchronized motors: run(power = 100)
* brake() : stop the motor. When called on Synchronized motors: brake both motors at the same time.
* idle() : stop motor whatever it’s doing. It smoothly stops.
* weak\_turn(power, tacho\_units) : turn motor for the specified distance.
* turn(power, tacho\_units, brake = True, timeout = 1): power is a value between -127 and 128 (greater than 64 is recommended).
* Tacho\_units is the number of degrees to turn the motor. (values smaller than 50 are not recommended)
* Brake is whether or not to hold the motor after the function exits.

\*\*\* Cs1robot module methods \*\*\*

* face\_north() : using the hitech compass sensor, the robot faces north
* front\_is\_clear() : Return True if there is nothing in front of the robot using ultrasonic sensor. Otherwise, return False (near 50cm).
* right\_is\_clear() : The ultrasonic sensor faces right side and detect if there is any object in the right side. Return True if there is nothing in the right side (near 50cm). Otherwise, return False
* left\_is\_clear() : The ultrasonic sensor faces left side and detect if there is any object in the left side. Return True if there is nothing in the left side (near 50cm). Otherwise, return False.
* move\_forward() : The ultrasonic sensor faces forward at first and go straight until the distance between the robot and any obstacle becomes 50cm. If the touch sensor is pressed, the collision occurred and the robot stops immediately.
* face\_front() : The direction of ultrasonic sensor is changed to the front side.
* face\_back() : The direction of ultrasonic sensor is changed to the back side.
* move\_backward() : The ultrasonic sensor faces backward at first when this function is called. The, the robot goes backward until the distance between the robot and any obstacle becomes 50cm.
* turn\_robot(x) : The robot turns x degrees clockwise then stop.
* get\_revolution() : Function computes the revolution of the robot. Function gets input from the user about distance between two wheels and radius of wheel. Then, compute the revolution.