INF-2202 (Fall 2013)
Assignment #1

Lars Ailo Bongo
larsab@cs.uit.no
Ibrahim Umar
ibrahim.umar@uit.no

22-08-2013

Abstract

For this assignment, you will work with the lego Mindstorms[1] robot. Each robots consists of three (touch[2], ultrasonic[3], and gyro[4]) sensors and two motors. This robot is a self-balancing robot, which means it needs to be programmed in order for it to stand straight on its own. One can achieve this by coordinating the two motors based on information given by the gyro sensor. The programming will be done in Python, which can interface directly to the robot via a USB cable or Bluetooth connection. Your task is to develop and evaluate thoroughly a program that will enables the robot to be able to self-balanced itself while capable in receiving simultaneous movement instructions. The said robot should also able to recover itself by backtracking those instructions whenever it encounters an obstacle that renders itself as impossible to move further forward. The implementation MUST incorporate efficient concurrent processing.

1 Expected Goals

Your task is to develop and evaluate thoroughly a program that will:

1. enables the robot to be able to self-balanced itself at any moments,
2. able to receive simultaneous movement instructions
3. able to recover itself by backtracking instructions whenever it encounters an obstacle that renders itself as impossible to move further forward

It is important that the robot have to maintain its balance at any moment, while obeying to user instructions. Use the gyro sensors effectively to achieve this.

The acceptable inputs are in a form of console-typed text using this format (FORWARD / BACK / LEFT / RIGHT, <x>). Variable x is the degree of rotation for LEFT / RIGHT instructions and distance in cm for FORWARD / BACK instructions. For example if you want the robot to move forward for 10cm, the command issued should be <FORWARD, 10>.

The input console is going to be non-blocking, which means that you can write as many instructions as you want and the console will not stop receiving any inputs while an operation is currently taking place. Entered instructions shall be executed sequentially one after another in the order that they are entered.

Issuing the command "PAUSE" / pushing the sensor button on the robot will immediately stops the currently running instruction. Pushing the sensor button again or issuing "CONTINUE" command will make the robot to resume the stored instructions. Be mind that PAUSE condition will not prevent new instruction inputs from the console.

Ultrasonic sensors is useful for detecting obstacles. Your robot must be able to detect an obstacle and re-trace back its way to the end-state of the last instruction whenever it encounters a blockage on its way forward and immediately proceed to the preceeding instruction. For example a robot is issued a sequence of: <FORWARD, 30>, <LEFT, 30>, <BACK, 10>instructions. However after moving forward for 10cm the robot encounters an obstacle on its way. In this situation the robot should respond by move backward for 10cm, returning its state to the end-state of previous instruction and then proceed to execute the next waiting instruction, which is <LEFT, 30>, and so on.

The implementation MUST incorporate efficient concurrent processing. This is very useful for further developing a much complex robotics system, especially one that utilise a much complex computation on a multiple core system. For example robots that incorporate facial recognition, gesture detection, etc.

Your assignment report must include a detailed analysis on the correctness of your implementation in approaching the outlined goals. Employed testing scenarios that leads to your conclusion are also need to be described. You also need to outline on how concurrent processing will be beneficial in this kind of robot operation as well as for more complex robotics operation.

You are also required to measure the response latency comparison between concurrent implementation of the Lego robot and non-concurrent (for example using a simple "global" while loop on the main procedure) implementation. Explain your findings in the same report.

NOTE: Due to our department request, you are ***NOT*** *allowed to disassemble the robots.*

2 Deadlines and Scoring Criteria

The assignment will commence on Thursday, 22 August 2013 and its resulting report (and codes) shall be submitted by 19 September 2013, both at the end of the scheduled colloquium class (at 14.00). Any late submission will not be accepted and a FAIL mark will be given as your score.

Submit your report and code to this email: `ibrahim.umar@uit.no` by the deadline. If there any discrepancies on the submission time, only the email receive timestamp will be taken into consideration.

The PASS / FAIL mark for this assignment will be assessed based on these criteria (not in order of preference):

1. The ability to demonstrate a program that performs the outlined goals
2. Originality of the implementation
3. The ability to develop an efficient and optimal concurrent program

4. The ability to write a good analysis and report

Those getting the FAIL marks will be given a second chance at the discretion of the responsible lecturers.

3 Preliminaries

This is a group assignment. Form a group of 5 person to work together on this assignment.

3.1 System Setup

These are the things that you will need to get up and running:

1. A self-balancing Lego robot
2. Either Windows, Mac OS X or Linux (recommended) OS
3. Python 2.6 or later (But not 3.x)
4. The nxt-python package (`https://code.google.com/p/nxt-python/`)
5. USB cable or Bluetooth connection. For minimum latency, a USB connection is recommended.

Refer to `https://code.google.com/p/nxt-python/wiki/Installation` for the installation guide of the *nxt-python*. After installing, go ahead take a look at the */examples* directory for various example scripts. From there on you should be able to get started in programming the robot.

3.2 Programming References

Here are some of the programming references related to the self-balancing lego robots that we are using:

1. `http://www.hitechnic.com/blog/gyro-sensor/htway/`
2. `http://www.techbricks.nl/My-NXT-projects/nxt-self-balancing-segway-nxtway-robot.html`

The links above contains valuable information and example source codes (in C-like language) that you can use as a starting point for implementing the self-balancing robot using *nxt-python*. I have also prepare a code snippets that demonstrate a way to obtain information from the available sensors:

```
import nxt.locator, time
from nxt.sensor.hitechnic import *
from nxt.sensor import *

b = nxt.locator.find_one_brick()  # Find an NXT brick and connect

s1 = Touch(b, PORT_1)             # Touch sensor, in port 1
s2 = Ultrasonic(b, PORT_2)        # Ultrasonic sensor, in port 2
```

```
s4 = Gyro(b, PORT_4)                # Gyro sensor, in port 4

s4.calibrate(); #calibrate the gyro sensor

for i in range(1000):
    print 'Accel: '+ str(s4.get_sample())       # accel data from gyro
    print 'Touch_Up: '+ str(s1.get_sample())    # touch data
    print 'Ultrasonic: '+ str(s2.get_sample()) # obstacle data
    time.sleep(0.2)

b.sock.close()  #close bluetooth connection
```

Also take a look at */examples/cnc.py* from the *nxt-python* package for an example of concurrent motor control.

If you need a concise programming references on threading with Python, this particular site: `http://pymotw.com/2/threading/` might be come in handy.

- GOOD LUCK! -

References

[1] http://mindstorms.lego.com/en-us/default.aspx

[2] http://mindstorms.lego.com/en-us/products/default.aspx#9843

[3] http://mindstorms.lego.com/en-us/products/default.aspx#9846

[4] http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NGY1044