CIS 1.5 (Science Section)
Brooklyn College
Professor Langsam

Assignment #3-Robot Exercises

Programming the Lego NXT ™ Robot using the nxtOSEK C++ development system consists of using the nxtOSEK class definitions (as well as several prewritten libraries) together with the standard C++ language. In this assignment you will write several simple programs to control the robot.

A Quick Reference Guide to nctOSEK C++ may be found here: http://lejos-osek.sourceforge.net/html/index.html. We will be using only a small portion of the many methods described in the guide.

The following header files should be included at the top of your programs:

```
     #include <robot.h>
     #include <robotTask.h>
```

A (very) Brief Introduction to Object Oriented Programming

Before we can write programs to control the robot, we must first understand how we can access the various components of the NXT robot. Each component of the robot is described by a ***class*** definition. A class contains a description of the components ***attributes*** (often called ***members***) as well as ***methods*** that may be applied to that component. In order to use the class we must create (or instantiate) an ***object*** of the class by which we will refer to that component.

For example, suppose we wish to display the message “Hello Robot” on the lcd. We would write:

```
Lcd screen;     // This creates an object called screen of
                // the Lcd class

screen.putf("s", "Hello Robot");    // putf and disp are
                                    // methods of the Lcd
                                    // class and are applied
screen.disp();                      // to the screen object
```

As another example, suppose we wish to have the wheel connected to output port B turn in a positive direction. We would write:

```
Motor motorB(PORT_B);    // This creates an object called
                         // motorB of the Motor class which
                         // is assigned to port B

motorB.setPWM(50);       // setPWM is a method which sends
                         // a signal to the port associated
                         // with the object motorB
```

A complete program to display a message on the lcd and move the robot forward for five seconds follows:

```
// Simple program that moves the robot forward for 5 seconds

#include "robot.h"
#include "roboTask.h"

//define objects
Lcd lcd;
Clock tiktok;
Motor motorB(PORT_B);
Motor motorC(PORT_C);

//function prototypes
void moveForward(int, int);

int main()
{
    int time = 5;
    int power = 50;      // power may range between -100 to +100
    lcd.clear();
    lcd.putf("s", "Moving Forward");
    lcd.disp();

    moveForward(time, power);

    return 0;
}
```

```
// moveForward moves the robot forward for a given time
// and power
void moveForward(int time, int power)
{
    motorB.setPWM(power);       // set motor PWM value
    motorC.setPWM(power);

    tiktok.wait(time*1000);     // wait in a loop for duration
                                // in msec

    motorB.reset();             // stop motor and set motor
    motorC.reset();             // encoder value to 0

}
```

You will be using the following nxtOSEK classes and methods in your programs:

Lcd		
	`clear()`	clears the lcd
	`putf(“s”, “string”)`	puts the string “string” in the lcd buffer
	`disp()`	displays the contents of the lcd buffer
Clock		
	`wait(msec)`	Wait in a loop for duration in msec
Motor		
	`setPWM`	Set motor PWM value
	`reset`	Stop motor and set motor encoder count to 0.

Write individual functions to perform the following tasks:

1. `moveFoward(int time, int power) -`
   The robot should move forward for time seconds at power level power and then stop.
2. `moveBackwards(int time, int power) –`
   The robot should move backwards for time seconds at power level power and then stop.

3. `spinRight(int time, int power) –`
   The robot should spin to the right for time seconds and then stop (The motors should spin in opposite directions.)
4. `spinLeft(int time, int power) –`
   The robot should spin to the left for time seconds and then stop
5. `uneven(int time, int leftPower, int rightPower) –`
   The robot should move forward with its left motor on forward at power level `leftPower` and its right motor on forward at power level `rightPower` for `time` seconds and then stop.

Have your main program do the following:

1. Call `moveFoward` followed by `moveBackwards`. Repeat this 5 times. Use `time = 5` seconds and `power = 50`. Display the direction of travel on the LCD.
2. Do the same for `spinRight` and `spinLeft`.
3. Call `uneven` with its left motor on forward at power level 64 and its right motor with power level 32. It should move for 8 seconds and then stop.
4. When the robot is finished have it display “DONE” on the LCD.

Test your program using the NXT Simulator in the CPlusVEBot system. Submit a printout of your program. Be sure to use meaningful variables, proper style and to comment your program as described in class.