

Duke/DPS Robotics Program

10/15

Lesson Plan 1

Goal: *Familiarity with the NXT brick and with basic movement in RobotC.*

Materials Needed: *Assembled Brookbot, sensors optional*

Part 1: The NXT Brick

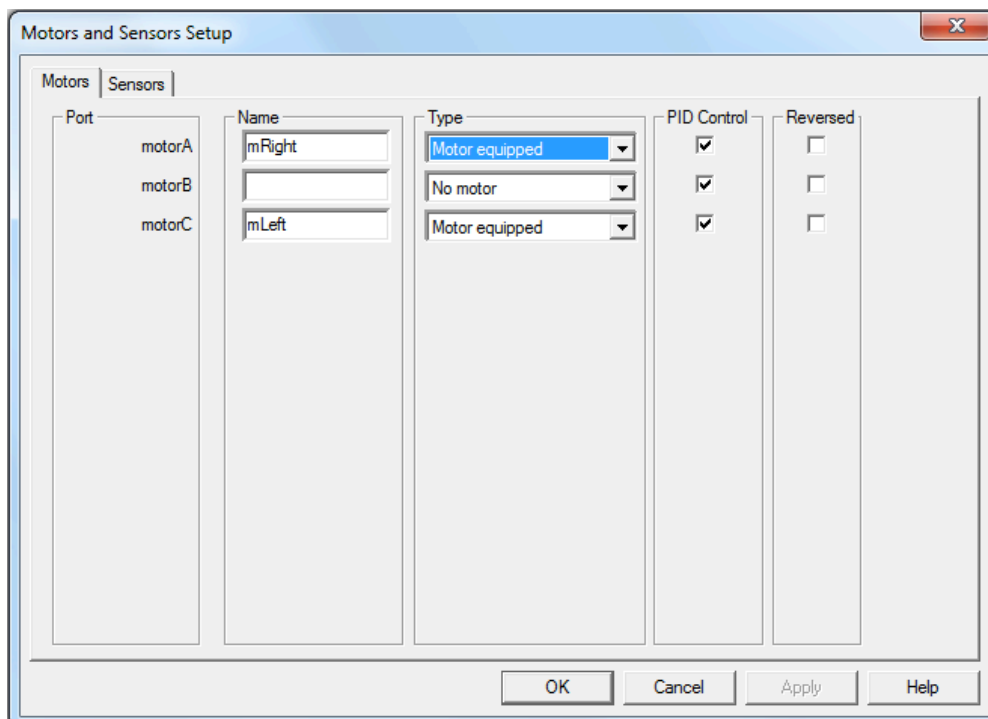
The Lego NXT has 4 sensor ports, 3 motor ports, 4 buttons, and a screen. Ask your students which of these are input and which are output. If they don't know, provide examples of I/O with a computer: mouse and keyboard are input while monitor and speakers are output.

Make sure to show them the battery, the charger, the battery life indicator, and how to power the robot on and off.

Part 2: RobotC

A. *Basic Program Structure*

The first thing you should do for any program is click Robot -> Motors and Sensors Setup. Choose "Motor equipped" in the dropdown menu for each active motor. Optionally, you can give your motors names, such as "mLeft" and "mRight."



In RobotC, every program must have a task called main. For now, all the code you write will go inside main's curly braces .

```
task main()
{
|
}
```

B. *Moving Forward*

To move the robot, set both motors to a power level of 50. Do this by accessing the motor array with a particular motor. You can either use the default names, like motorA and motorB, or the names you gave your motors in setup, like mLeft and mRight. The following code is equivalent:

```
task main()
{
  motor[motorA] = 50;
  motor[motorC] = 50;
}
```

```
task main()
{
  motor[mRight] = 50;
  motor[mLeft] = 50;
}
```

C. *Time*

Presumably, you want your robot to stop rather than going on forever. To stop your robot, set the motor power to zero. To wait for a specific amount of time, use the wait1Msec() function. If you want to wait for a whole second, pass in 1000 as a parameter.

```
task main()
{
  motor[motorA] = 50;
  motor[motorC] = 50;
  wait1Msec(5000);
  motor[motorA] = 0;
  motor[motorC] = 0;
}
```

D. *Turning*

The robot is not built for the wheels to tilt like on a car. How, then, can we get the robot to turn? Get your students to figure out the solution to this question: power only one wheel, or spin the wheels in opposite directions. Explain the difference between these two approaches.

So to implement this in RobotC, set one motor's power to zero, or give it a negative value to have it spin the opposite direction.

```
task main()
{
  motor[motorA] = 50;
  motor[motorC] = 0;
  wait1Msec(5000);
}
```

```
task main()
{
  motor[motorA] = 50;
  motor[motorC] = -50;
  wait1Msec(5000);
}
```

Part 3: Dead-Reckoning Challenge!

Program your robot to navigate a very simple course. Explain the concept of dead reckoning: navigation based on precise foreknowledge of the environment. The faster the better, but try not to hit the edges of the course!