

ROBOLAB Programming Tutorial
Spring 2006
Evan Lemley
UCO Physics and Engineering Department

1. Introduction to Lego Robot Kits
2. ROBOLAB Video (skip this)
3. Introduction to ROBOLAB Software, the RCX, and Software Configuration
4. Pilot Level Programming Introduction
5. Pilot Level One
 - (a) Demo Program
 - (b) Write a program to turn on run a motor forward (right arrow) on port A for 5 seconds. **Program = Pilot 1**
6. Pilot Level Two
 - (a) Demo Program
 - (b) Write a program to turn on motor A at power level one and motor C at power level C (both motors the same direction) until the touch sensor is released. **Program = Pilot 2**
 - (c) Try rotating the wire attached to motor A by 90/180/270 degrees on port A. Try stacking two touch sensors on port 1.
7. Pilot Level Three
 - (a) Run Once vs. Continuous Run
 - (b) Load and Save
 - (c) Wait for +/- 5 % change of light sensor
 - (d) Demo Program (Don't worry about the lamps)
 - (e) Change the Demo Program so that when a touch sensor on port 1 is depressed the motors reverse direction. Also when a touch sensor on port 3 is released the program should stop. **Program = Pilot 3A**
 - (f) Change your program so that motors reverse when a touch sensor on port 3 is depressed and the program ends when a light sensor on port 1 is *uncovered* (i.e. 5% more light being received). Try making the light sensor end the program when it *is covered*. **Program = Pilot 3B**
8. Pilot Level Four
 - (a) Multi Step Programs – Deleting and Adding Steps, Scrolling Through Steps
 - (b) Demo Program
 - (c) Modifiers for Time, Touch Sensors, and Light Sensors
 - (d) Write a program to test the sensitivity of the light sensor. **Program = Pilot 4**
 1. Delete Step #2 (only Step #1 should remain).
 2. Change the touch sensor to a light sensor on port 2.

3. Alter the light percentage to smaller and larger values and use the program with a sheet of white paper with a piece of black electrical tape on it (you may want to try different colored items as well).
4. Observe what will and will not make the program stop for different light percentage values.

9. Inventor Level Programming Introduction

- (a) The Functions Palette
- (b) Help – Ctrl-H
- (c) Tools – press Tab to change
 1. Select (arrow)
 2. Placement (open hand)
 3. String (spool)
 4. Text (letter and cursor)
 5. Change Value (pointer finger)
- 6. Space Bar toggles between Select and String**
- (d) Make a Demo program to play with the tools
- (e) Removing Bad Strings – **Ctrl-B**

10. Inventor Level One

- (a) Demo Program – Note must use stop signs to stop power to ports.
- (b) Write a program to turn on a motor on port B for 10 seconds then stop.
- (c) Write a Program to wait for the touch sensor to be pushed in then rotates the motors on ports A & C to the left for 6 seconds. The program should then reverse the rotation of both motors until the touch sensor is pushed again.

11. Inventor Level Two

- (a) Function Palette – *Wait For, Modifiers, Structures* Commands
 1. Wait For – Time, Light Sensor, and Touch Sensor.
 2. Modifiers – Output Ports, Input Ports, Power Level, Numeric Constants
Power Levels –

RoboLab Power Level	Real Output Port Power (%)
1	12%
2	25%
3	50%
4	67%
5	100%

3. Structures – Jump and Land (Go To loops and infinite loops).
4. Write a program to (use the general inputs and outputs) **Optional**
 - i. Wait until the touch sensor on port 3 is pressed to turn on Motor A at half power
 - ii. Keep the motor on until the touch sensor is released

- iii. Turn the motor back on (in the opposite direction) when the light sensor (port 1) has increased by more than 15%.
- iv. After seven seconds of running the motor end the program.

12. Inventor Level Three

- (a) Multitasking Structure – independent portions of your program after a branch
- (b) Looping Structure – Use numeric constant to specify number of times.
- (c) Forks – Touch Sensor, Light Sensor – Fork Merge
- (d) Write a program that checks to see if the light level is below 50. If it is below 50 then lamp A is turned on at full power. If it is less than 50 then Lamp A should be off. **Optional**
- (e) Write a program that when the light sensor (on port 2) is greater than 40% motors A and B run in the same direction at power level 3 (50% power) and when the light sensor drops below 40% motor A changes direction and increases to 4 (67%) and motor B decreases to 1 (12%). **Program - Inventor Level 3A**
- (f) Modify the **Inventor Level 3A** program so that when the light sensor on port 2 **is not** below 40% **and** another light sensor on port 1 **is** below 40% that motor A runs at power level 1 (12%) and motor B runs at power level 4 in the opposite direction(67%). **Program - Inventor Level 3B**

13. Inventor Level Four

- (a) More functions
 - 1. Containers are just variables – Note can branch on Container Value
 - 2. Multiple colored jumps and lands may be used for nested loops

14. Line Following

- (a) Build a vehicle
- (b) Program it to follow a line (single width of the electrical tape on tile floor in the hallway outside the PSE1 Lab (Howell 163) under normal lighting conditions).
- (c) Vehicle must be able to make a right turn and a left turn.
- (d) Vehicle that finishes course in shortest time or most completely wins!
- (e) Mechanical Constraints and Suggestions
 - 1. Only one (1) RCX
 - 2. Only two (2) motors
 - 3. Only two (2) light sensors
 - 4. The motors are high speed low torque – to control the direction of your bot better you want lower speed and higher torque. You need to build a gear system to achieve this! Look for gearing suggestions for Legos on the web – the following links might help.

<http://home.wanadoo.nl/eggplant/lego/index-transmissions-main.htm>

<http://www.weirdrichard.com/gears.htm>

<http://www.texbrick.com/ideas/gears/>

<http://www.marshall.edu/LEGO/lessonplans/Houston/LEGOprojects.html>

Look for more yourself.

(f) Team Suggestions:

1. Team Leader should manage project.
2. **Everyone needs to learn to program.** Trust me it works best this way.
3. Some might work on detailed logic of the program while others work on mechanical design. Obviously someone needs to make sure logic and mechanical design fit together.
4. Test Often (but try to conserve batteries). Make very simple programs to test with. I have seen teams flounder to get their program to work because they missed something very basic at the beginning.

Examples:

- i. Figure out what light sensor values are necessary to be sensitive to different colored tape on the tile floor (we will show you where to test since lighting conditions and other variables are different than in HOH 106).
 - ii. Figure out proper motor powers for your gearing situation. Note you will likely have to do some early trial and error to help you decide on a gearing system.
5. Don't get too focused on a particular test pattern you might make to test since to win you need a bot that responds well in a general fashion.