CS148
Building Intelligent Robots

Week 1 : Sensors and Line Following *Out: 13 Spt 2005*

Preliminary Tasks *before 15 Spt 2005, 9am*

This week's lab will introduce you to the lego touch and light sensors. Your main task is to write a calibration light routine, and use it to make your tankbot into a linefollower. *Every project you will do this semester will depend on light sensor calibration.* There is no preliminary reading for this lab, but you must come to class with brickOS set up on your account, and an already assembled tankbot. During the lab, you may find the brickOS section of the website to be helpful. Everything in this lab you can also find in the HOWTO, command reference, and code samples pages in the brickOS section of the website.

NOTE: You must setup brickOS on your cs account, and you must come to lab with your already assembled tankbot.

Lab 1: Sensors and Line Following *15 Spt 2005*

Done during lab; please see a TA for checkoff after each section.

0.1

The MindStorms light sensors measure light intensity (for the most part). The light sensor can be activated in two modes: *active* and *passive.* In *active* mode, the light sensor also emits a red light, enhancing the range and accuracy of the sensor quite a bit. In *passive* mode, the red light is not activated. The active mode is usually much more accurate and stable. For the purpose of this class you will probably always want to use active mode. One thing to note is that at close distances to objects (let's say, a playing field of some sort), using active mode will return readings that correspond more to reflectivity rather than intensity. The boundary between reflectivity and intensity seems to be about 4–5 inches away from whatever object you're trying to read.

To set the mode, *active* or *passive*, of a light sensor, use the commands:

- `ds_active(&SENSOR_X);` and `ds_passive(&SENSOR_X)` where X is the number of the sensor (1, 2, or 3).
- To read the light sensor values, access sensor number X as a constant directly using `LIGHT_X`. This value is set approximately every quarter millisecond by brickOS.

Task:

- First, write a program that will display the value of a light sensor connected to the RCX brick (pick an input slot, any slot). Choose a suitable time interval between updates and use the `lcd_int(int)` command to display the value read in by the light sensor.
- As with all sensors, the light sensor is highly vulnerable to changes in the ambient environment and thus experimental values should almost never be used as a good indicator. Augment the program you just wrote with a calibration phase. You will then be given a sheet with three color-coded regions: black, green, and white. When the new program starts, it should determine the average value of the black, green, and white areas and then run. While running, your program should be able to correctly identify the three colors in various ambient light conditions by outputing to the LCD either "black", "green", or "white".

0.2

Touch sensors will almost always be used as binary values. To access the value of a touch sensor as a Boolean, use the constant `TOUCH_X` (where X is the sensor number). This will return a 1 if the sensor is pressed and 0 if the sensor it not.

Task:

- Write a program that recognizes touches on the bump sensor. Your program should give some indication (beep, LCD print out, spins its motors wildly) that a connected touch sensor has been depressed.

0.3

Since there are only three inputs on the RCX, we must use them as efficiently as possible. One way to do this is to put a touch and light sensor on the same input. Access the input as you would a light sensor. You will find that when the touch sensor is depressed, the light sensor reading will read a certain range of values outside the norm of a light sensor. In fact, the readings will also vary with the strength of the depression of the touch sensor.

Task:

- Find out the exact value (or value range) of the light/touch sensor input when the touch sensor is depressed. Use this value range to improve upon your program so that is recognizes all three colors in various conditions and will also display "bump" on the LCD if the touch sensor is depressed. *Hint: use the first task from 1.1 to visualize light and touch sensor readings and then incorporate that into calibration.*

0.4

The purpose of this section is to become more familiar with the many quirks of the light sensor, as it is the most important—and annoying—sensor you will be using. You must build a robot that will follow a line on the floor. The line will be about 1″ wide and of a significantly different color than the floor. You cannot assume the line is darker than the floor, or vice-versa. Your robot should have a calibration phase that takes into account this fact and ambient conditions such that your robot could work on any line in any room with any lighting conditions.

When the robot gets to the end of a line, it should not travel along the same line in the opposite direction. Your robot may or may not start on a line after the calibration phase. If it doesn't start on a line, it should go straight forward. If it hasn't found a line within 7 seconds of dead time at any point during its task, it should stop.

Some problems you might run into:

- How many light sensors should you use?
- How should you design the calibration phase?
- How can you shield the light sensor so a minimal amount of light leaks into it? (Note: good shielding is extremely helpful!)

- Should you design your robot so the line takes precedence or the floor takes precedence? In other words, should your robot follow only the line you tell it to follow or any line that is different from the floor? (Both are perfectly acceptable.)

Note: Light sensors introduce a lot of inconsistencies that can't easily be predicted. The only way to catch and fix them is by trial and error.