Massachusetts Institute of Technology

Robotics: Science and Systems I

Lab 4: Light Sensors and Braitenberg Behaviors

Distributed: Monday, 2/23/09, 3pm

Due: Monday, 3/2/09, 3pm

Objectives and Lab Overview

Your objective in this lab is to understand how to integrate and use sensors on your robots. You will add two light sensors to your robot. You will mount the sensors and calibrate them. Then you will use the sensors to control the robot's movement according to the light levels it senses from them.

This lab will give you the technical skills to incorporate and use a light sensor. It will allow you to experiment with fairly simple means of reacting to sensory input and observe or predict a variety of behaviors for your robot.

Time Accounting and Self-Assessment: Make a dated entry called "Start of Light Sensor Lab" on your Wiki's Self-Assessment page. Before doing any of the lab parts below, answer the following questions:

- **Electronics**: How proficient are you at working with electronics (as of the start of this lab)? (1=Not at all proficient; 2=slightly proficient; 3=reasonably proficient; 4=very proficient; 5=expert.)
- **Motion Control**: How proficient are you at crafting robot motion algorithms using feedback from light sensors? (1=Not at all proficient; 2=slightly proficient; 3=reasonably proficient; 4=very proficient; 5=expert.)

To start the lab, you should have:

- Your robot from previous lab
- Two light sensors, wire, connectors, heat shrink tubing, heat gun (shared), soldering iron and solder
- A photocopied handout on Braitenberg vehicles from the book entitled "Vehicles: Experiments in Synthetic Psychology" by Valentino Braitenberg.

Check for any errata for the lab in the course wiki, and for any additional materials linked from the lab handouts webpage.

This lab requires a working solution for the MotorControl lab (not the Chassis lab): a whole-robot velocity controller which accepts angular velocity commands for each wheel. You may choose either to use your group's solution or the solution we have published. If you have choose to use our posted solution to the MotorControl lab instead of your own, then you will need to change all the references in your `LightSensors` java files from

```
import MotorControl.*;
```

to

```
import MotorControlSolution.*;
```

Physical Units

We remind you to use MKS units (meters, kilograms, seconds, radians, watts, etc.) throughout the course and this lab. In particular, this means that *whenever you state a physical quantity, you must state its units*. Also show units in your intermediate calculations.

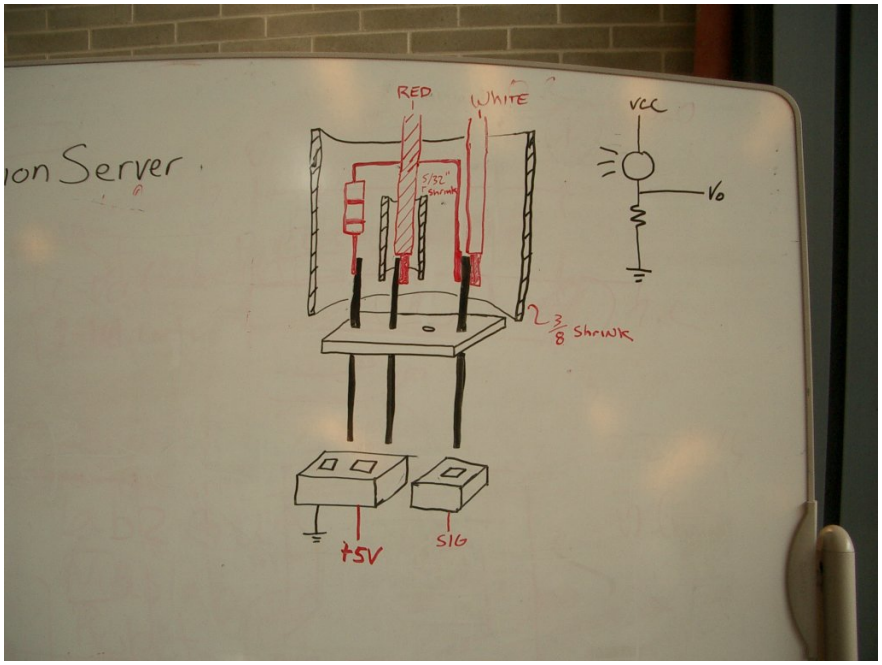

Figure 1: Wiring diagram of the light sensor.

Part 1: Incorporating the sensors

Recall from the lecture on sensors that a light detecting sensor is incorporated using a voltage divider. The voltage at the analog input pin of your ORCboard is determined by the voltage between the resistor and the photocell.

1. Consult the wiring of the light sensor on the exemplar, and Figure 1. *Write down the formula for the voltage divider which gives V_{out} in terms of $V_{in} = 5$V, the photocell resistance R_{photo}, and the resistor R_1. If your photocell has a resistance of 180 Ohms in bright light and 200k Ohms in the shade, what is the voltage range of your light sensor as a function of R_1?*
2. Suppose your photocell has an average resistance around 47k Ohm in ambient light. *What is a desirable value for the resistor?* You can measure the real range of resistance of your photo cell by connecting it to your multimeter set to the ohm meter setting.
3. Suppose you use a resistor an order of magnitude larger than your answer above. *What will be the effect? What can you do to compensate?*
4. A TA will demonstrate how to build and mount each light sensor on the robot. For each sensor, place the signal pin of the connector you make into header 12 or 13 on the ORCboard, exactly as on the exemplar robot. *Using ORCspy, what range of values do you see during an interval of ambient light?* Shine a light in front of the sensors. *What range of values do you see as you close the interval (with roughly discrete steps in distance) between the light source and the sensor?* Cover the sensor with your palm. *What range of values do you see now?* Record these values on your wiki. Plot this data to characterize your sensor.

Deliverables: Make a new page on the wiki named "LightSensors Lab Report Group N." Add a picture of your completed sensors to the wiki. Answers the questions above and put the sensor data plot on the wiki. Please include a description of how you mounted the sensors and let us know what was easy and what was difficult.

Part 2: Calibrating the Sensor

It is important to calibrate sensors relative to ambient light because ambient light affects your sensor readings. Ambient light varies from room to room or even in the same room at different times of the day when you have different lighting conditions. Ideally, you would like for your robot to adapt to ambient light.

To keep it simple, in this lab your robot should run a calibration method that explicitly senses the ambient light upon startup and sets an offset term and a linear scale term. Later, another method will use these stored values to return a calibrated reading from the light sensors at run-time.

1. Fill in the `calibrate()` method in `Photocell.java` to set the instance variables `offset` and `scale`. Have your method take multiple sensor measurements of the ambient light over a duration of at least 5 seconds (the frequency is up to you). A method named `getRawValue()` is already provided in `Photocell.java` which returns the raw value of the sensor.

   Assume that 5 volts is always the maximum reading, and that it corresponds to saturation of your sensor. Assume that the light sensor readings scale linearly with the light level. Make your calibrated values range from 0 to 100 . That is, 0 should indicate ambient light and 100 should indicate saturation (bright light).

   Based on the sensor behavior you saw in Part 1 via the ORCspy, is a linear scaling sufficient? What other scaling could be done to improve the information from the sensors?

2. Fill in the `getValue()` method in `Photocell.java` which returns the calibrated value of the light sensor, based on its current reading, `offset`, and `scale`.

Deliverables: Answer to the question on linear scaling being sufficient.

Part 3: Searching for a Light Source

In this part of the lab you will use the light sensors to guide the motion of your robot toward a **fixed** (i.e. non-moving) light source. Ultimately, your robot will search for a light source, head toward it, and stop and beep once it arrives.

Hint 1: Use a bright, omnidirectional light source, and hold it at the same height as the light sensors. A bare 75W incandescent light bulb works well. Be careful not to touch it or let it touch your robot, since it is hot.

Hint 2: During the daytime, very bright light enters the room through the large windows. Think about what direction you want the robot to be facing during the cablibration–*should it face toward or away from the windows? Why?*

1. In `Behavior.java`, implement `search()` to find the direction of a bright light source (note that a convenience `Behavior.setDesiredAngularVelocity()` method has been provided). It is up to you to define what is considered a "bright" light. The `search()` function is called repeatedly (from `Behavior.go()`) until it returns `true`. *How far away can the light source be for the robot to detect it in your implementation?* If the robot does not see any light at the start (that is, it is not directly pointed a the light source), it should search for it by rotating. *What will your robot do when there are two light sources? Or when a single light source is equally spaced from both light sensors?* Test your algorithm for 3 different distances and 3 different orientations of the light source and plot a graph of each experiment (it is up to you to define a sensible graph; total 6 experiments).
2. Implement `Behavior.goToLight()`, this is invoked right after `search()` returns `true`. After the initial `search()`, the robot should not make any more turns. Make the speed of the robot decrease in proportion to its distance from the light source, and beep (a `beep()` method is provided in `Behavior.java`) when the robot is close to the light source (again, by your definition).

Deliverables: Document your methods and prepare a written explanation of the algorithms for the locate and tracking methods. Include your test data (screen shots of runs work well) and the answers to the questions above.

Part 4: Braitenberg Behaviors

In his book entitled *Vehicles: Experiments in Synthetic Psychology*, Valentino Braitenberg describes a series of thought experiments in which vehicles with simple internal structure behave in unexpected ways. He experimented with

simple control mechanisms that generate complex behaviors with psychological interpretations such as aggression, love, foresight, and even optimism. All these behaviors have very simple internal representation and it is surprising that the resulting perceived response of the robot can have such complex manifestation. In his book, Braitenberg gives these machines as evidence for the *law of uphill analysis and downhill invention* which refers to how it is more difficult to guess internal structure of a creature from the observation of behavior than it is to create the structure that gives the behavior.

The simplest Braitenberg vehicle is equipped with one sensor and one motor. This is described in the Braitenberg handout. We shall skip implementing this vehicle but it is worth noting the vehicle responds linearly to sensory input. That is, it would run at a fast motor speed when there are high sensor values and slow motor speed at low sensor values. It is similar to the functionalities you implemented in Part 3. Your next tasks are to implement several Braitenberg vehicles with your robot.

1. Implement `Behavior.vehicle2a()` and invoke it from the appropriate spot in `Behavior.go()`. Connect the behavior of each light sensor to the motor on the same side. When the sensor gets closer to a source, its corresponding motors moves more quickly. As the sensor moves away from a light source, the corresponding motor slows down. This is termed excitatory response. Test your program and record how your robot responds to fixed light sources in 3 different locations, as well as its response to a moving light source. *What emotion does your robot's behavior most resemble?*
2. Implement `Behavior.vehicle2b()`. Connect the behavior sensor to the motor on the opposite side of the robot. *Write your prediction on what will happen under different light source positions and test the robot's response in 3 different example settings, plus in a setting with a moving light source. How do your predictions match the robot's response? What emotion does your robot's behavior most resemble?*
3. Implement `Behavior.vehicle3a()` and `Behavior.vehicle3b()`. You can implement Vehicle 3a by generalizing your code for Vehicle 2a, making the connections inhibitory rather than excitatory. That is, when the sensor gets closer to a source its corresponding motor should slow down, and when the sensor gets farther from a source its corresponding motor should speed up. *Repeat your predictions and experiments recording them.* Likewise, for Vehicle 3b, generalize your code for Vehicle 2b, making the connections inhibitory rather than excitatory. *Write your prediction on what will happen under different light source positions and test the robot's response in 3 different example settings, plus in a setting with a moving light source. How do your predictions match the robot's response? What is the difference between the two vehicles' behavior under that same light source conditions. What emotions best describe your robot's behaviors?*
4. Design and implement your own vehicle behavior. For example, consider the Wimpy Robot behavior: your robot gets attracted by a light source, investigates with great curiosity for a while, then gets scared, turns around, and runs in the opposite direction as far and as fast as it can! Some other options include varying excitation and inhibition or varying the sensor response curve to create new behaviors. You can call the function you implement in the `Behaviors.CREATIVE` section of `Behavior.go()` *Repeat your predictions and experiments recording them on your wiki. Select your favorite and report on the algorithm and observed results in detail.*

Deliverables: Include a detailed report on all the questions above. Be sure to explain your **documented** *algorithms and tests. Be prepared to demonstrate your work on the due date marked at the top of this document.*