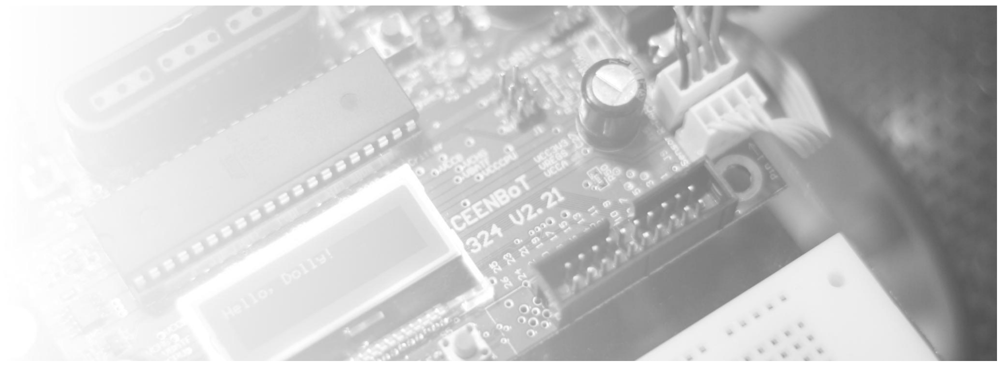
Mobile Robotics I: Lab 4

Light Sensing with Photo-resistors

CEENBoT™ Mobile Robotics Platform Laboratory Series
CEENBoT v2.21 – '324 Platform



Alisa N Gilmore, P.E., *Instructor, Course & Lab Developer*

The Peter Kiewit Institute of Information Science & Technology
Department of Computer & Electronics Engineering
University of Nebraska-Lincoln (Omaha Campus)

Rev 1.02

(Blank)

Purpose

In this laboratory exercise you will create and incorporate two *photo-resistor light-sensors* and connect them to the Analog-to-Digital Converter (ADC) inputs on your CEENBoT's controller board. You will use light sensing with photo-resistors to implement Reactive and Behavior-based controllers. You will first use light sensor data to create several reactive controllers based on light intensity inspired by Valentino Braitenberg's *Vehicles*. Next, you will build upon the IR_AVOID and CRUISE behaviors created in the previous lab to incorporate a new LIGHT_FOLLOW behavior in a formalized behavior-based control structure.

Lab Objectives

By following the directions in this lab, you are expected to achieve the following:

- Design and create photo-resistor sensors and sensor cables for use in a robotics light sensing application. Wire these onto the CEENBoT controller board and experiment with their positioning on the front of the robot to achieve the best field results. Experiment with the values generated by the photo-resistor light sensors in various light conditions.

- Get to know the ADC subsystem module of the CEENBoT API in order to read voltage inputs wired to the CEENBoT's ADC port and manipulate their values in C-code.

- Implement Valentino Braitenberg's *Vehicles* to see first hand the impact of simple reactive controllers and the characteristics exhibited by your robot under simple motor-sensory couplings.

- Add upon the behavior-based control paradigm introduced in the previous lab by creating a new Light_Follow behavior and integrating it with the IR_AVOID and CRUISE behaviors with the aid of a formalized behavior control structure in C with fixed priorities. At the conclusion of this step, you should have a random wanderer robot that avoids obstacles and homes in on a light source, when appropriately triggered.

Requirements

Preliminary Readings

- Read about the *ADC subsystem module* chapter in the *CEENBoT-API: Programmer's Reference* manual as this lab will require that you make use of the ADC (Analog-to-Digital converter) facilities of the CEENBoT-API.

- For information about Sensors: *The Robotics Primer* by M. Mataric Chapters 7-8 and *Robot Programming: A Practical Guide to Behavior-based Programming* by J. Jones Chapter 9.

- For information about Braitenberg's Vehicles: *Vehicles: Experiments in Synthetic Psychology by* Valentino Braitenberg. 1984. Vehicles 2 and 3.

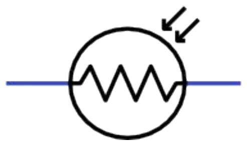
Required Equipment

- CEENBoT, platform '324 v2.21.
- Means to program your CEENBoT. (i.e., USB or Serial ISP programmer).

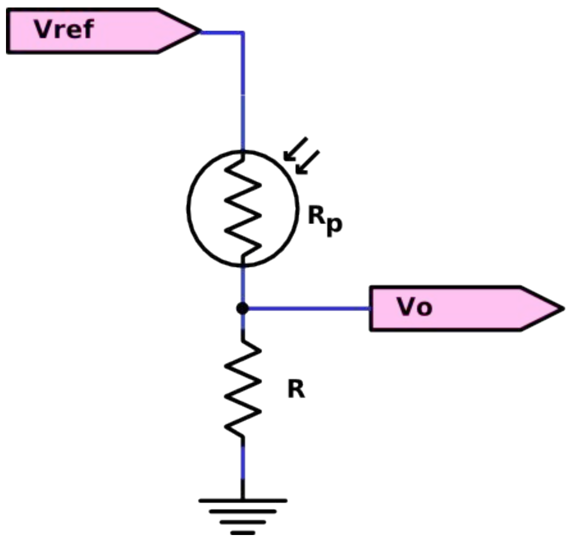
Background

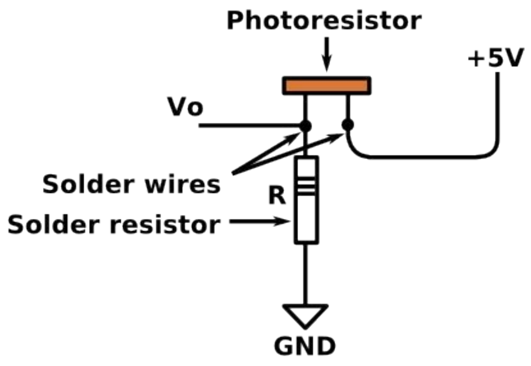
THE PHOTO-RESISTIVE DEVICE

A *photo-resistor* is a semiconductor device whose resistance is a function of light intensity. The schematic symbol for the photo-resistor is shown below:



Because the resistance of the photo-resistor varies with light intensity, the *current* that flows *through* it also varies with light intensity. However, we want to monitor a voltage, not a current, since the ADC (Analog-to-Digital Converter) on the micro-controller takes voltage measurements. We will be able to monitor a voltage from the photo-resistor by creating a simple voltage divider circuit, as shown below. The photo-resistor used in this lab is designed to have a maximum resistance in the absence of light. As light intensity increases, its resistance decreases. As a result, as light intensity increases, the voltage, Vo, in the voltage divider circuit will also increase. You will monitor this voltage Vo as a measure of the light intensity seen by the photo-resistor.





The top left shows the voltage divider configuration you will use. The top right shows how you will solder and attach wires and the resistor to the photo-resistor. If you need to, you can also view photos on the construction of your photo-resistor cable here:

http://roboticsprimer.sourceforge.net/workbook/HowTo:_Make_A_Photoresistor_Cable

You should end up with three wires per photo-resistive device: one for the +5V supply, one for the output voltage Vo, and the other for ground (GND).

When configured in this way, the output voltage is given as follows:

$$V_o = \left(\frac{R}{R + R_p} \right) \cdot V_{\text{ref}}$$

That is, the output voltage is a factor of the *voltage-divider* network formed by the photo-resistor's resistance and your chosen value for R. A good range of values for R should be between 2K to 10K.

> Take the following example – suppose you choose R to be $10\,K\Omega$ and that the photo-resistor is rated to vary between $3\,K\Omega$ (maximum light reception) and $200\,K\Omega$ (no light reception). Then the voltage output using the previous equation is given by:
>
> $$0.0476 \cdot V_{\text{ref}} \;\leq\; V_o \;\leq\; 0.7692 \cdot V_{\text{ref}}$$
>
> So, if Vref = 5V (as it *will* be), then your output voltage will vary as given:
>
> $$0.238\text{V} \;\leq\; V_o \;\leq\; 3.846\text{V}$$

NOTE: Here, the highest voltage value represents maximum light intensity, and the lowest voltage value represents a dark or no light condition.

Alternatively, instead of using the rated values shown above for the photo-resistor used in this lab (Manufacturer no. Jameco CDS001-8001), you can design your sensor to be precisely based on the resistance you measure for the minimal light reception (dark, covered) and the maximum light reception in the environment you will use the sensor. To do this, use a multi-meter to measure the resistance across the photo-resistor in these two situations. The first situation is the darkest light your robot will see. Cover up the photo-resistor entirely and measure the resistance. The second situation is the brightest light your robot will see, such as next to the light source. Use these measured resistance values in the above equation for the upper and lower limits on R_p and calculate your R value accordingly.

After calculating the expected range for V_o, you will then read this voltage value in through the ADC and determine its meaning, in terms of presence or absence of light, which takes us to the next topic.

THE ADC SUBSYSTEM MODULE

The ATmega324 on the v2.21 platform of the CEENBoT has an 8-channel ADC. That is, the ADC has 8 analog inputs that can be monitored (one at a time). Channels **`0`**, **`1`**, and **`2`** are already taken up to monitor battery current, battery voltage, and charger voltage levels. However, channels **`3`** to **`7`** are *free* for your use. You can attach the voltage output (Vo) from your photo-resistive devices into one one of these available channels. These channels are located in via ***`J3 Header`*** on the controller board via pins **`1`** through **`5`** on ***`J3`*** (corresponding with ADC channels **`3`** to **`7`**). These are once again, tabulated below (see next page):

ADC Channel	**Pin on J3 Header Connector (on '324 Controller Board)**
ADC Channel 3	1
ADC Channel 4	2
ADC Channel 5	3
ADC Channel 6	4
ADC Channel 7	5
VCC (5V)	**19**
GND (0V)	**20**

Note: Please pay *careful* attention to where pin numbers and where you connect your wires.

Next, once your 'hardware' is configured, you can read analog voltage data via the ADC using the API. The API allows you to 'sample' data by using the '`ADC_sample()`' function. This function returns a 10-bit representation (ADC code) of the analog voltage (in a 16-bit variable of *type* `ADC_SAMPLE`).

You then take this ADC code, and covert it to it's voltage representation using the following equation:

$$V_{\text{ADC}} = \left(\frac{\text{ADC}_{\text{code}}}{1024} \right) \cdot V_{\text{ref}}$$

Where 'Vref' is the voltage reference used by the ADC, which for our purpose, it will be 5V. Here's a quick 'snippet' on how you would read analog data via ADC channel 3.

```
#include “capi324v221.h”

void CBOT_main( void )
{

    ADC_SAMPLE sample;  // Storage for ADC code.
    float      voltage; // Storage for 'voltage' representation of ADC code.

    // Open the ADC subsystem module.
    ADC_open();

    // Set the voltage reference (we want 5V reference).
    ADC_set_VREF( ADC_VREF_AVCC );

    // Set the channel we will sample from.
    ADC_set_chan( ADC_CHAN3 );

    // Okay, now SAMPLE it!
    sample = ADC_sample();
```

(*Continued on next page*)

(*Continued from previous page*)

```
        // Convert it to meaningful value.
        voltage = ( sample / 1024 ) * 5;

        // Now do whatever it is you need to do with this information..., etc.

    } // end CBOT_main()
```

This is just a *quick sample*.

Note: To read analog voltage data using the API please take a look at the *ADC Subsystem module* chapter in the *CEENBoT-API: Programmer's Reference* manual for details on how to use the ADC using the API. There is a 'snippet' that shows how to open and initialize the ADC, along with how to 'collect ADC samples' and convert them to their meaningful voltage values.

ROBOT PROGRAMMING

In this lab you will first implement reactive controllers inspired by Valentino Braitenberg's *Vehicles*. In reactive controllers there is a tight coupling between sensing and action. Next, you will write a Light_Follow behavior designed to make the robot home in on a light source. You will then add this behavior to a program that also includes the IR_AVOID and CRUISE behaviors created in the previous lab to incrementally build upon behaviors. In this way, you will have a random wanderer robot that avoids obstacles and when a high enough intensity light source appears, begins to follow it. In order to do this you will need to structure your program using a formalized behavior-based control structure in C that approximates a type of multitasking among behaviors, along with fixed priorities, to be presented in class.

Directions

Part I

1. Create two photo-resistor sensor cables by soldering the connections as shown in the Background section, for an appropriately selected resistor. State the resistor value you chose for the circuit along with the range of voltage values you expect from the voltage divider circuit for min and max light conditions. Connect the sensor cables to power, ground and two available ADC inputs on the CEENBoT.

2. Arrange the photo-resistors on the CEENBoT using tape or other means to position them on the front of the CEENBoT, on the left and right sides of the robot, either facing forward, at a diagonal. You can play around with the best positioning during the lab.

3. Experiment with the ADC functions in the API to read in the values from each sensor. Get a feel for the values the sensors generate in various light conditions by creating a program to display the voltage value of each sensor onto the LCD. Record in a table your initial ADC voltage values generated for each of the following conditions by **each** sensor: ambient light on the lab table, ambient light on the lab floor, in dark (cover the sensor), next to a bright artificial light source, in sunlight (in a hall near a window). You should complete a table of 2 columns (for each sensor) by 5 rows (for these scenarios). Are the sensors equally matched in their voltage readings or is their a significant difference in voltage values read by each in similar light conditions? In this step, be sure you have an adequate range in voltage values that will enable you to distinguish between various light conditions.

4. The first program you will write a reactive controller inspired by Valentino Braitenberg's vehicle experiments. Read about Vehicles 2 and Vehicles 3 in his book. In this step you are to create a vehicle that is wired with excitatory connections where each sensor is connected to the motor on its same side (Vehicle 2a). Write a program that controls left and right wheel speeds based on the light intensity seen by the left and right photo-resistors. Observe and record how the robot behaves when: (a) the light source is directly in front of the robot (b) the light source is to one side of the robot. Is there anything about the robot's behavior that surprises you?

5. Next, repeat the above exercise, except now cross the connects between motors and sensors, so that the left light sensor controls the right motor's speed, and vice versa (Vehicle 2b). Observe and record how the robot behaves when: (a) the light source is directly in front of the robot (b) the light source is to one side of the robot. Is there anything in the robot's behavior that surprises you?

6. Now repeat step 4, except use inhibitory connections so that each light sensor is connected in an inhibitory manner to the motor on the same side (Vehicle 3a). Observe and record how the robot behaves when: (a) the light source is directly in front of the robot (b) the light source is to one side of the robot. Is there anything about the robot's behavior that surprises you?

(*Continued on next page*)

(*Continued from previous page*)

7. Repeat step 6, except now cross the connects between motors and sensors, so that the left light sensor controls the right motor's speed in an inhibitory manner, and vice versa (Vehicle 3b). Observe and record how the robot behaves when: (a) the light source is directly in front of the robot (b) the light source is to one side of the robot. Is there anything in the robot's behavior that surprises you?

8. **BONUS:** Can you create a reactive controller that will cause the robot to orbit the light source?

Part II

1. Next, you will continue from the previous lab in your implementation of a behavior-based controller. First, create a Light_Follow behavior designed to make the robot home in on a light source, when the average value of light intensity between the two photo-resistors is greater than some minimum value (that you will determine.) Represent the actions triggered by this behavior using a Finite State Machine before coding.

2. Add this behavior to a program that also includes the IR_AVOID and CRUISE behaviors created in the previous lab to incrementally build upon behaviors. In this way, you will have a random wanderer robot that avoids obstacles and when a high enough intensity light source appears, begins to follow it. In order to do this you will need to structure your program using a formalized behavior-based control structure in C that approximates a type of multitasking among behaviors, along with fixed priorities.

3. **BONUS:** Create an additional behavior called Light_Observe, that is triggered when the Light_Follow behavior gets within a distance of 6-8 inches from the light source (you have to determine what intensity and voltage this represents). To keep the robot from following the light source too closely, Light _Observe behavior will stop the robot for a set about of time as if causing it to observe the light source, then back up and quickly turn away from the light source. Represent the actions triggered by this behavior using a Finite State Machine before coding.

Questions to consider in your Report

1. How did you end up positioning your photo-resistors on the front of the CEENBoT? Why did you choose such positioning?

2. Describe whether the Light_Follow and Light_Observe behaviors are ballistic or servo behaviors.

3. Answer all the questions posed throughout the Directions section of this lab.

Deliverables

Demonstrations

You will demonstrate your robot's execution of these new capabilities during the designated class meeting or lab time.

C-code

Include a printout of your CEENBoT program. You may include *snippets* of your code and embed them in your lab report as you discuss how your program works, but a separate attachment of your entire source code must be included as part of your report.

Lab Report

The lab report should include all of the following:

- **Title Page** – Include *Course Number*, *Course Title*, *Instructor Name, Your Name*, *Lab Name*, & *Due Date*.

- **Overview** Section (a brief paragraph of what the lab was about, and the purpose behind it). Please also make sure you touch upon the following as well:

  1. Resources used: (human, text, online or otherwise).
  2. Time invested in this project.
  3. A high-level description of your robot and program.
  4. If this was a *team assignment*, state how tasks were delegated and split up among you.
  5. Known problems with your solution (e.g., the robot will not work on the *carpet*, or... it breaks if you make it go further than 5 ft (for whatever reason), etc.)

- **Background** Section (discuss some of the theory addressed in this particular lab).

- **Procedure** – Briefly describe the lab procedures for each lab – what were you asked to do? What did the process consist of? How *did you* approach it?

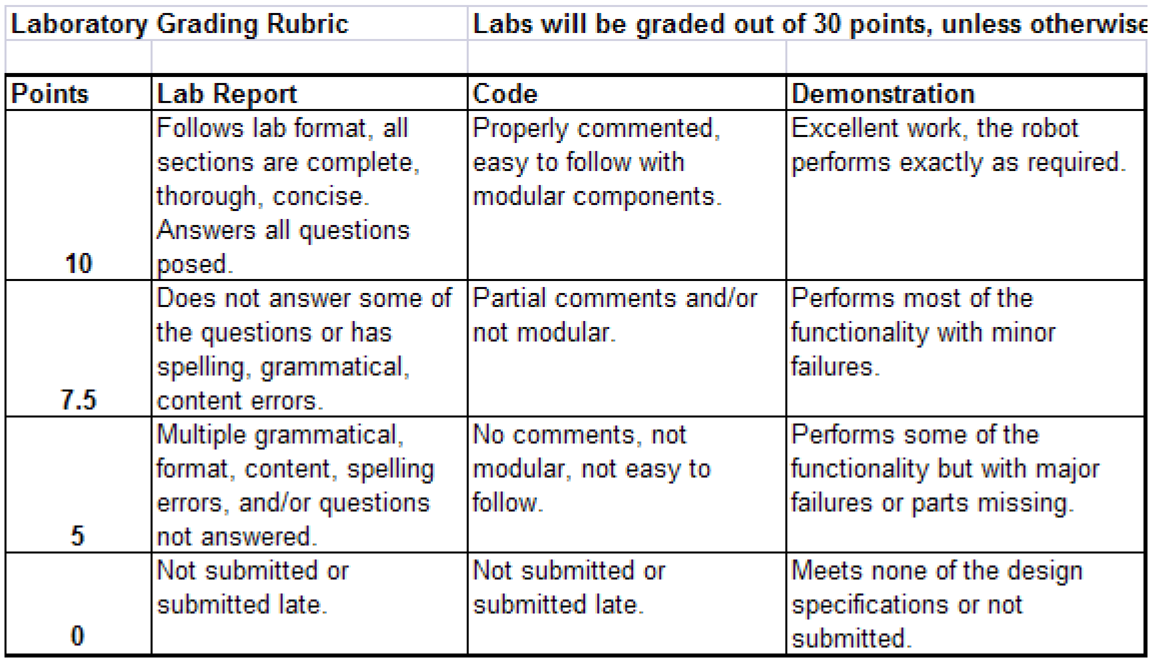
- **Discuss your Source Code** – Discuss and explain any relevant details behind the motivation and manner in which you have written your source code. <u>You can *embed* [small] portions on of your code that are relevant to the discussion for clarity, but please ensure to keep a complete copy of your source code as a separate attachment</u> (see *Source Code* section below). You may also have the reader *refer* to particular pages of your source code attachment instead.

- **Results –** Use this section to answer *questions* posted throughout your lab exercise. In particular those given out in the '*Directions*' section. In the '*Procedure'* section, feel free to refer the reader to *this* section for additional info – that is, there is no need to repeat information.

- **Conclusion** – Reiterate the objective of the lab, discuss what you have learned and any further comments you might have.

- **Source Code** – Attach your C-program also.

Grading



Laboratory Grading Rubric		Labs will be graded out of 30 points, unless otherwise	
Points	**Lab Report**	**Code**	**Demonstration**
10	Follows lab format, all sections are complete, thorough, concise. Answers all questions posed.	Properly commented, easy to follow with modular components.	Excellent work, the robot performs exactly as required.
7.5	Does not answer some of the questions or has spelling, grammatical, content errors.	Partial comments and/or not modular.	Performs most of the functionality with minor failures.
5	Multiple grammatical, format, content, spelling errors, and/or questions not answered.	No comments, not modular, not easy to follow.	Performs some of the functionality but with major failures or parts missing.
0	Not submitted or submitted late.	Not submitted or submitted late.	Meets none of the design specifications or not submitted.