Design Lab 7: For Your Eyes Only

You'll need a lab laptop for the second part of the lab.

- **Athena machine:** Do `athrun 6.01 update` and `add -f 6.01`.
- **Lab laptop:** Do `athrun 6.01 update`.
- **Personal laptop:** Download design lab 7 zip file from course web page.

Code is in `~/Desktop/6.01/lab7/designLab/`.
For reference on connector pin-outs or soar brain input/output, see the *Robot Infrastructure Guide* **in the reference section of the course web page.**

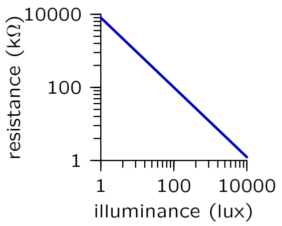
Our ultimate goal is to build a 'head' with eyes and neck that can turn and track a light. This week you will connect light sensors to your robot and write a brain that can make the robot move to face a bright light.

Many systems are made up of a combination of special-purpose electronics and general-purpose computation. Today, we will build such a system. You will build a small circuit to connect two light sensors to the robot and, thence, to a lab laptop, allowing soar brain software to 'read' voltages from your board and 'command' voltages back to it.

1 See the Light

Our first task is to use a pair of photoresistors to construct light sensors (eyes) for the robot head. A photoresistor is a two-terminal device whose electrical resistance depends on the intensity of light incident on its surface.

A photoresistor is made from a high resistance material (e.g., cadmiun sulfide). Incident photons excite the electrons – liberating them from the atoms to which they are normally held tightly – so that the electrons can move freely through the material and thereby conduct current.



The net effect can be characterized by plotting electrical resistance as a function of incident light intensity, as shown (qualitatively) above (notice that the axes are logarithmically scaled). Normal room lighting is between 10 and 100 lux. Illuminance near a 60 watt light bulb (as we will use in lab) can be greater than 10,000 lux.

The details of the behavior of the photoresistor will depend on its particular design and will vary substantially even among "identical" parts. However, in all cases, we expect to see a very high resistance in the dark and a low resistance near a lamp. One reasonable approximation to the

Athena is MIT's UNIX-based computing environment. OCW does not provide access to it.

behavior of a photoresistor is: $R = \frac{R_0}{I}$, where R is resistance, I is illuminance and R_0 is a (large) constant, representing resistance when the illuminance is 1 lux (dark).

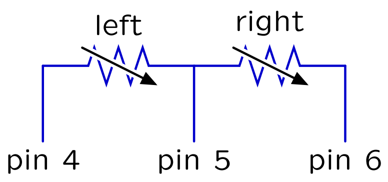
When thinking about the behavior of a photoresistor, keep in mind that illuminance will drop as the square of the distance from a light source. So, resistance will increase as you move sensor away from the light source.

1. We will start by measuring the resistance of the photosensors in different lighting conditions. Get a robot head and use a red cable to connect the 8-pin connector on the head to a second 8-pin connector (from one of the supply bins). Plug the second 8-pin connector into an otherwise empty protoboard. This is the `head` connector.

Head Connector

pin 1:		neck pot (top)
pin 2:		neck pot (center)
pin 3:		neck pot (bottom)
pin 4:		photoresistor (left)
pin 5:		photoresistor (common)
pin 6:		photoresistor (right)
pin 7:	V_{M+}	Motor drive +
pin 8:	V_{M-}	Motor drive −

2. Notice that there are two photosensors on the head. The photosensors are wired to the `head` connector as shown below.



Connect your meter to pins 4 and 5 of that connector, using short wires as probes into the protoboard. Switch your meter to measure resistance (the scale is marked Ω), and be sure you understand the scales (you can always measure a known resistor to help figure out the scale). Get a silver hand-held lamp and plug it in to a power strip.

Check Yourself 1. Measure the photoresistance (Ω) of **each photosensor individually** under the following lighting conditions.

with ambient light (two values):	
three feet in front of lamp (two values):	
one foot in front of lamp (two values):	

3. Design a circuit that uses one photoresistor (plus one or more additional resistors) to generate a voltage that is large under bright conditions and small under dark conditions. The robot provides a 10 V voltage source. Make sure there is at least a 2 V difference between the generated voltages in ambient conditions and when there's a lamp one foot away. Sketch your circuit below. *Hint: Think about how a potentiometer generates a voltage that depends on the values of its resistances.*

Check Yourself 2. What voltage do you expect for the following lighting conditions?

with ambient light:	
three feet in front of lamp:	
one foot in front of lamp:	

4. The robot has in it a set of converters. A-to-D (analog to digital) converters take analog voltages from pins 1, 3, 5, and 7 of the robot connector, sample them on each soar cycle, and encode them in binary so that they may be read by the computer, ultimately showing up as a list of values in the `analogInputs` attribute of a `io.SensorInput` object. A single D-to-A (digital to analog) converter takes a value specified by the `voltage` argument to the `io.Action` constructor, and converts it into a voltage which is made available on pin 6 of the robot connector; we will be using this output voltage in later weeks.

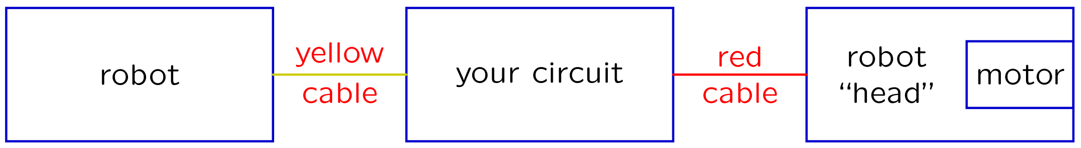
You can connect your circuit to a robot via a yellow 8-pin cable, which is coming out of the robot. We'll connect this cable to the board using a connector that is exactly the same as the head connector; to help keep this distinct from the red cable connecting your board to the head, remember: 'red' — 'head'. The pins of the robot connector are wired up as follows:

Robot Connector

pin 1:	V_{i1}	analog input #1
pin 2:	$+10\,\text{V}$	power (limited to $0.5\,\text{A}$)
pin 3:	V_{i2}	analog input #2
pin 4:	ground	
pin 5:	V_{i3}	analog input #3
pin 6:	V_o	analog output
pin 7:	V_{i4}	analog input #4
pin 8:	$+5\,\text{V}$	power (limited to $0.5\,\text{A}$)

Plug a second 8-pin connector on your proto board. Connect power and ground on your board to the corresponding pins (pins 2 and 4) on the robot connector.

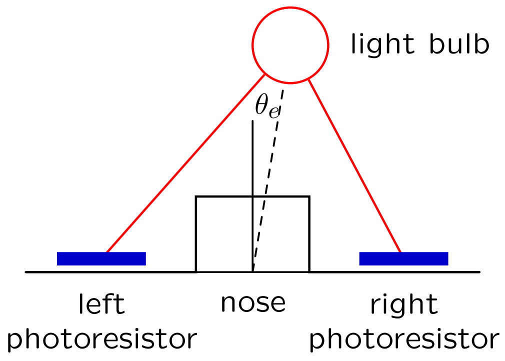
5. Build two photoresistor circuits: one to generate the voltage V_L from the left photoresistor and one to generate the voltage V_R from the right photoresistor. Here is how the whole system should be configured:



Attach the head to the lego plate on the front of the robot (sometimes putting a couple of lego bricks in between will make this easier), connect the yellow robot cable to your board, and turn on the robot. Use your multimeter to make sure that you are getting reasonable values for V_R and V_L. You can use your finger to obscure each of the sensors in turn and see that the voltages behave as expected.

6. Connect V_L to analog input #2 (pin 3) on the **robot** connector and connect V_R to analog input #3 (pin 5).
7. Take your robot and laptop near one of the standing lamps on the sides of the room. Connect the robot to the laptop, turn on the robot, and start soar, telling it to run the brain in `eyeDataBrain.py`. Line up your robot in front of the lamp, so that the head is pointing at the lamp and the robot is about a meter from the lamp. Now turn the robot **clockwise** by 90 degrees. This will turn the robot through 180 degrees, click Stop on `soar` when it has fully turned. Three plots should appear when you click Stop: the brightness on the left and right eyes as well as the difference between them. **Take a screenshot and save this data.**

We want to write a new soar brain that will make the robot turn so that the head is directly pointed toward a bright light anywhere in front of the robot. It should do this as accurately as possible. Examine your plots and decide how you want to approach this. The photoresistors are separated by a "nose" that casts a shadow, and thereby controls the amount of light on the two photoresistors (see top-down view at right).



Checkoff 1. **Part a.** Explain your plots to a staff member.
Part b. Describe your approach to pointing at the light. Be prepared to justify your approach.

2 Bull's Eye

Go back to your original table and debug using a silver lamp. When you think your program is roughly right, then go test it using a standing lamp.

Modify `turnToLightBrainSkeleton.py` to implement your approach to pointing the robot towards the light. It will generate one plot, which you can ignore for now. The voltages from the photosensors can be read as follows:

```
# io.SensorInput().analogInputs is a list of all 4 analog inputs
left = inp.analogInputs[1]
right = inp.analogInputs[2]
```

To enable us to see how well your pointing approach works, we have provided some low-power laser pointers that are mounted on the robot head. To power the laser, you need to plug the yellow and red wires from the laser into the robot connector. The yellow wire goes to pin 4 (ground) and the red wire goes to pin 8 (+5V power). **Do not plug the laser into pin 2!!**. We want the laser to move quickly and reliably to make the red dot fall on the lampshade.

First, tune your controller so that the laser lands reliably on the lampshade. You will probably have to compensate for discrepancies in the responses of the two eyes to the light. Be prepared to discuss your approach for dealing with this.

Once your controller's accuracy is adequate, you can focus on its speed. When you stop the brain, you will see a plot of the robot's angle as a function of the number of time steps. An estimate of the *settle time* of the signal will be printed to the output window. Recall that the settle time of the signal is the number of steps required for the signal to converge. If your signal is oscillating, it will report a number that is near the end of the signal.

Use the following procedure to collect data for your controller:

- Position the robot at a 45 degree angle from the lamp.
- Start the brain.
- Stop the brain when the robot stops moving, or when it is clear that it will not converge.

For each trial, save the plot and record the associated settle time and gain value (or whatever parameters your controller uses). Generate three plots with substantially different behaviors, one of which is oscillatory. Except in the oscillatory case, each trial should end with the laser on the lampshade. **You should save these plots (and associated parameters and settle times) for your interview**.

Finally, experimentally optimize your controller with respect to settle time. You don't need to to spend too long on this, but you should try several more parameter settings, and save the plot, settle time, and parameters for the controller you find to have the best behavior. Save a plot of your best controller for your interview as well.

Checkoff 2. **Part a.** Discuss your approach to the problem.
Part b. Demonstrate your best controller.
Part c. Discuss the plots, parameters, and settle times for several controllers with different behaviors.

2.1 For fun...

After you're done with everything else, change your brain so that it follows the light. It should move forward or backward to keep a desired distance from the light, as well as rotating towards the light.

- Mail your brain, plots, parameters, and settle times to your partner.
- Disassemble the circuit board; return both 8-pin connectors and cables to the supply bins.