Design Lab 10 *6.01 – Fall 2011*

Robot Pets

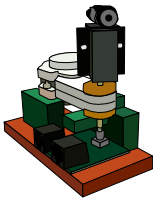
Goals: Design Lab 10 focuses on building and testing a controller a simple "pet robot." You will:

- Demonstrate an integrated analog and digital controller for the robot to follow a light.
- Integrate the sonar sensors with your controller.

Resources: This lab should be done with a partner. Each partnership will need a lab **laptop** as well as the following:

Robot

Lamp



Robot head

Two eight-pin connectors

Red cable

Resistors, as needed

Black motor cable

Do `athrun 6.01 getFiles`. The relevant file (in `~/Desktop/6.01/designLab10`) is

- `roverBrainSkeleton.py`: Brain file for implementing your pet robot controller.
- `boundaryFollower.pyc`: State machine for a boundary follower from **Design Lab 2**.

If you and your partner already completed Design Lab 9, retrieve your board and make sure that it still works.
If you did not complete it, then complete it now.

Some of the software and design labs contain the command athrun 6.01 getFiles. Please disregard this instruction; the same files are available on the 6.01 OCW Scholar site as a .zip file, labeled Code for [Design or Software Lab number].

1 Pet Robot

We would like our robot to follow a bright light around the room. The head you built in **Design Lab 9** is capable of turning much more quickly than the (heavy) robot, so we will construct a two-level control system in which the head turns to track moving light and the robot body turns so as to keep the head pointing forward relative to the body. This is analogous to your visual system, where your eyes move quickly to track motion and your head turns in the direction of gaze.

Use the head as configured in the previous part, so that it automatically turns toward a bright light. Mount the head on the robot body facing forward (same direction as the sonar array).

Step 1. Design and implement a robot behavior that uses signals from its head to turn the robot toward a bright light.

You can read the voltages from pins 1, 3, 5, and 7 of the robot connector from a `soar` brain as the list of four values `inp.analogInputs`, where `inp` is an instance of the `io.SensorInput` class. **For some robots, pin 1 gives bad readings, so if you're having trouble, try switching your input to the other pins.**

Write a `soar` brain to implement your controller. The output of the controller should be an `Action` that specifies the rotational velocity `rvel` of the robot. We have provided a skeleton in `roverBrainSkeleton.py`

Hint: You can debug the robot behavior by tilting the robot backwards so that the wheels do not touch the ground and watching to see that the behavior is reasonable before unleashing your robot on the world. Also, start with the black cable disconnected, so that you can manually turn the head and observe if the wheels turn correctly.

Demonstrate that your robot turns toward a bright light. What is the highest gain (in the software control loop) for which you get stable responses?

Step 2. We would like to rework the brain and circuit to make the robot's behavior depend on its proximity to the light. If the light is off, the robot should stand still in obedience. If the light is on, the robot should approach the light, positioning itself approximately half a meter from the bulb.

Check Yourself 1. What control variable(s) from the head is/are important for determining proximity to the light? Explain.

For this behavior, the `soar` brain will need access to not just the neck pot but also some measure of the light intensity. Figure out how to make your circuit provide this information, and make whatever connections are needed, using one or more of pins 1, 3, 5, and 7 on the **robot connector**. **Talk to a staff member for ideas if you don't see how to do it for your circuit.**

Checkoff 1. **Wk.10.2.1**: Demonstrate your pet robot's basic behaviors, including facing the light, approaching the light, retreating from it, and patiently waiting when there is no light. For extra brownie points, try parallel parking.

Save all code and plots, and mail them to your partner (for the next interview).

2 More behaviors

We would like to extend the behavior of the robot to take into account the sonar sensors so as to get more interesting behaviors.

Use **state machine combinators** (much like in **Design Lab 3**) to construct a brain that combines the boundary follower from **Design Lab 2** with your light-following controller.

When the light is bright enough and the path towards the light is unblocked, the robot should follow the light, otherwise the robot should follow the wall.

You can try using your boundary follower if you have it handy, otherwise you can uncomment the following line in `roverBrainSkeleton.py`

```
from boundaryFollower import boundaryFollowerClass
```

This will define a state machine class `boundaryFollowerClass` that implements the boundary follower. It can be used to control the robot. If you replace

```
mySM = mySMClass()
```

in the brain file with

```
mySM = boundaryFollowerClass()
```

the robot should follow walls.

Step 3. Now, combine your light-following state machine and the boundary follower state machine.

If you have some other interesting behavior you would like to implement instead, talk to a staff member about it.

Checkoff 2. **Wk.10.2.2**: Demonstrate your pet robot's extended behavior.

Take your board apart.
Remove the 8-pin connectors from your circuit board and put them back where you got them.
Keep the op-amps and potentiometers and put them back where you got them.
You can put wires that are in good shape back in the wire kit. Otherwise, you can discard resistors and most wires.
Make sure you turn off the power on your multi-meter and your robot.

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011