Design Lab 8: Turning Heads

You can use any computer that runs Python.

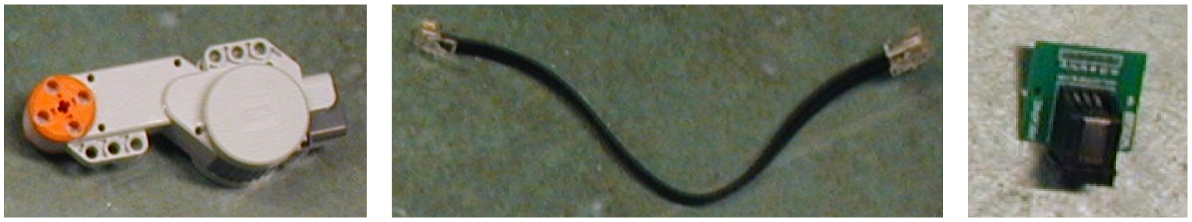
- **Athena machine:** Do `athrun 6.01 update` and `add -f 6.01`.
- **Lab laptop:** Do `athrun 6.01 update`.
- **Personal laptop:** Download design lab 8 zip file from course web page.

Code is in `lab8/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' that we can put on the robot that will be able to sense and track light. Last week, you designed a brain to control the robot to point at a light. The goal for this week and the next is to build a circuit that controls the neck motor on the head to get faster tracking of the light.

1 Lego Motor

We use a Lego motor shown below for the "neck" of the robot head.



The motor attaches to a 6-pin proto board connector via a short cable with RJ-11 connectors that are similar to those used for telephones. Notice that the two ends of the cable are different: the locking clip is centered on one end and offset on the other. The end with the centered clip goes into the connector, the end with the offset clip goes into the motor. The motor is driven via pins 5 and 6 of the connector.

The motor is designed to be driven with a voltage difference between 0 and 10 V across its terminals. Try it out as follows.

- Connect the power supply terminals labeled +15 V and **GND** to the power rails of your separate proto board using alligator clip leads. Adjust the power supply voltage to 0 V. (Yes, really 0).
- Plug a 6-pin RJ-11 connector into the proto board and connect it to a standalone motor (do **not** use a pre-built head).
- Turn off the power supply and wire pins 5 and 6 of the connector to the power and ground rails of the proto board.

- Turn on the power supply.
- Connect a multimeter to measure the voltage from the power supply.
- Adjust the power supply voltage between 0 and 10 V and note the relation between motor speed and applied voltage.
- Swap the connections to power and ground. What happens?
- What is the minimum voltage required to make the motor turn?

Check Yourself 1. $V_{min} =$

- Remove the connection between the motor and power rails of the proto board.
- Adjust the power supply to +10 V, then turn it off.
- Measure the resistance R_m of the motor using the multimeter.

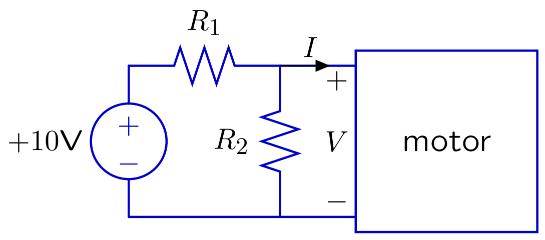
Check Yourself 2. $R_m =$

1.1 Controlling the motor with resistors

Our goal is to control the motor electronically. We will ultimately mount the motor on the robot and use the robot's power supply, which is constant at 10 V. The point of this section is to find a way to use a constant-voltage power supply to get a range of motor speeds.

First, think about how we might control the velocity with resistors. One way might be to use a voltage divider to generate a control voltage between 0 and 10 V, and then use this control voltage to drive the motor.

Consider the following resistor circuit for generating the control voltage, where $R_1 = R_2 = 1000\Omega$.

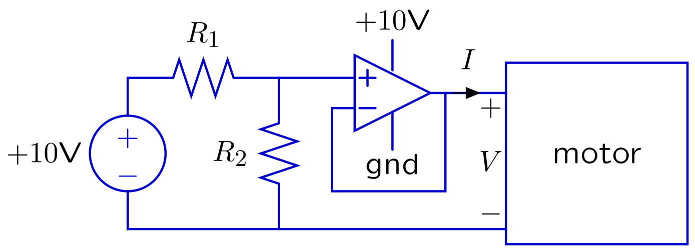


Build the circuit on your proto board. Measure the voltage across the motor and observe the motor's behavior.

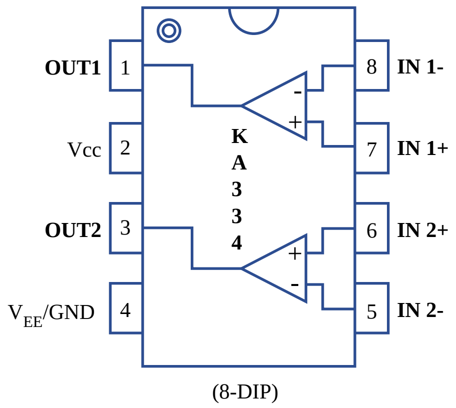
Check Yourself 3. Solve the circuit, treating the motor as a resistor. Pay careful attention to how the mathematical model of the circuit's behavior corresponds with the actual the behavior of the motor. Recall your measurements in the first part of this lab; pay particular attention to the voltage across the motor.

1.2 Buffering the motor voltage

We can add an op amp to *buffer* the output of the resistor network so that the resistors function as a voltage divider while the resulting voltage drives the motor. A simple buffer circuit is shown below.



We use op amps (KA334) that are packaged so that two op amps fit in an 8-pin (dual inline) package, as shown below.



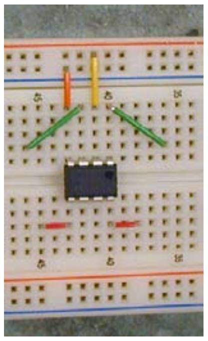


Figure by MIT OpenCourseWare.

The spacing of the pins is such that the package can be conveniently inserted into a breadboard as shown above (notice that a dot marks pin 1). The top center (yellow) wire in the picture above shows the connection of the op amp to the positive power supply (+10 V). The shorter (orange) wire to the left of center shows the connection to ground. The diagonal (green) wires indicate connections to the outputs of the two amplifiers, and the short horizontal (red) wires indicate connections to the two inverting (−) inputs.

Build the circuit on your proto board. Measure the voltage across the motor and observe the motor's behavior.

Check Yourself 4. Compare the behaviors of the circuit with and without the buffer.

Now, replace the two resistors in the voltage divider with a potentiometer.

Check Yourself 5. Step the potentiometer through various settings (1/4 turn, 1/2 turn, 3/4 turn) and observe the behavior of the motor. We'll ask you to compare this behavior to a simulation below.

Checkoff 1. Explain to a staff member the results of your experiments.

2 Circuit Simulations

We saw in Lab 6 that we can simulate circuit layouts using CMax. As we explore more complex circuits, we will want to simulate circuits without going through the trouble of doing a complete wire layout in CMax. In fact, CMax derives a `Circuit` component level (as in this week's Software lab) representation of the circuit from the layout and then simulates it.

Here's a procedure for simulating the circuit above (you can find this file in the lab distribution: `circuitSimulateTest.py`).

```
def motorTest(test):
    (nsteps, signal) = test
    circ = [OpAmp('v+', 'v-', 'vo'),
            Wire('vo', 'v-'),
            Pot('gnd', 'v+', '10v'),
            Connector('Motor', ['1', '2', '3', '4', 'vo', 'gnd']),
            Power('10v'),
            Ground('gnd'),
            Probe('Pos', 'vo'),
            Probe('Neg', 'gnd')]
    simulate.runRealCircuit(circ, signal, nsteps = nsteps)
>>> motorTest(oneStep.testSignal())
```

Note that you cannot do `OpAmp('v+', 'vo', 'vo')` to build a buffer; a `Wire` can be used to connect the output to the negative input.

The function `simulate.runRealCircuit` takes a list of circuit component instances and a test signal, runs the simulations and produces graphs. The Python definition of an appropriate signal can be imported from the test files that you use with CMax, for example `oneStep.testSignal()` returns a tuple (number of simulation steps, signal), as shown above.

We have the following classes of components, some you saw in Software Lab, and some new ones:

- `Resistor(value, node1, node2)`
- `Wire(node1, node2)`
- `OpAmp(posNode, negNode, outNode)`
- `Connector(type, pinNodes)`: the `type` argument can be: `'Motor'`, `'Head'`, and `'Robot'`; the `pinNodes` argument is a list of node names.
- `Power(node)` and `Ground(node)`
- `Probe(type, node)`: the `type` argument is either `'Pos'` or `'Neg'`.
- `Pot(node1, node2, node3)`: represents a $5K\Omega$ potentiometer, with `node2` being the middle terminal, whose voltage will vary. The resistance between `node1` and `node2` is $\alpha(5K\Omega)$, while the resistance between `node3` and `node2` is $(1-\alpha)(5K\Omega)$, where $0 \leqslant \alpha \leqslant 1$.

Evaluate `circuitSimulateTest.py` via Run Module in Idle. Then you can type `motorTest(oneStep.testSignal())` in the Python shell. This will run the simulation and produce several graphs:

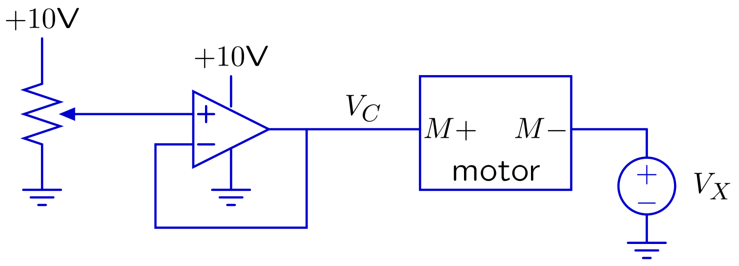
- **Probe** (in green): When there is a probe in the circuit, this graph shows the voltage measured across the probes.
- **Motor** (in red): When there is a motor in the circuit, then simulations **assume that the motor is attached to a potentiometer**, which turns as the motor turns. One of the motor graphs is the α value of the motor potentiometer, which measures the motor angle. The other motor graph shows rotational velocity (radians/sec) for the motor. Remember that the potentiometer has a finite range of rotation, so with a constant voltage, the motor will quickly reach the end of the range and stop. **When you're using a real robot head, driving it up against the end of the range in this way risks tearing the head apart.**
- **Input** (in blue): When there is an external input to the simulation, such as a potentiometer, this graph shows the input value, for example, the value α for a potentiometer, which goes between 0 and 1.

Check Yourself 6. Make sure you understand the meaning of the motor rotational velocity and motor pot alpha graphs. Compare the simulated behavior to the actual behavior of the circuit you built.

3 Bidirectional Speed Controller

The circuit you built above controls the speed of a motor. That circuit allows the motor to turn fast or slow (depending on the choice of resistors or the pot setting), but only in one direction. To make our robot head turn both left and right, we need to design a bidirectional speed controller.

The circuit above only turns in one direction because the op-amp operates from a single +10 V power supply. We are limited to a single +10 V power supply, because it is the only power supply available from the robots for which we are building the "head." A simple approach to this problem (using a (5KΩ) potentiometer) is to connect the motor as follows:



Check Yourself 7. What value of V_X gives the most symmetric (around 0) range of speeds for the motor?

$V_X =$

The key new component in the bidirectional speed controller is the voltage source V_X. Design a circuit to implement this voltage source (using only a fixed 10 V supply, which is all that's available from the robot).

Check Yourself 8. Can you implement V_X with a simple voltage divider? Explain.
Modify the circuit diagram above to include your circuit for supplying V_x.

Build a `Circuit` simulation of your bidirectional control circuit by completing the definition of the `biDirectional` procedure in `circuitSimulateTest.py`. It should contain a circuit definition that is a modification of the one in the `motorTest` procedure. When the α value of the pot is near zero, the motor should spin quickly in one direction; when α is 0.5, the motor should be stopped, and when α is near 1, the motor should spin quickly in the other direction.

The `biDirectional` circuit can be tested with the `threeSteps.testSignal(1.0)` test, which simulates turning the potentiometer to three different values.

Checkoff 2. Demonstrate your working simulation. Explain the relation between motor speed and potentiometer angle. Demonstrate that you can generate both positive and negative speeds. Explain how your circuit accomplishes bidirectional speed control.

Save a plot of the input signal and the associated output signal, as well as the procedure that defines your circuit.
Mail these results to your partner. We will discuss these at your interview.

4 Pointing

Your goal for the remainder of this lab and during next week's lab is to design and compare two circuits for controlling the neck motor on the head so as to point the head quickly and accurately towards a bright light (as you did last week by turning the robot, but hopefully more quickly).

4.1 Design Criteria

We will start by considering what properties we want our circuit to have:

- **Fast**: The head should line up with the light as quickly as possible.
- **Stable**: The head should not oscillate.
- **Uniform**: The behavior of the head should be nearly independent of the brightness of the light and of the distance of the light to the head.
- **Accurate**: The head should point accurately at the light, as demonstrated by the laser pointer.

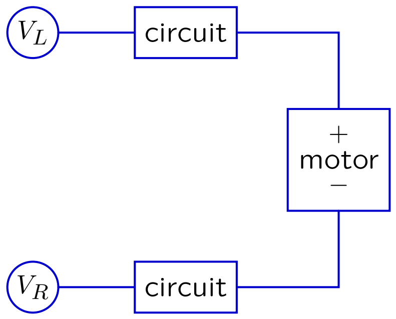
There are a few key points to keep in mind when thinking about a design.

- Remember that op-amps can't produce voltage values outside the range of supplied voltages (0V to 10V in our case).
- Recall that the speed of the motor is proportional to the voltage difference across it, and that for fast response you want this difference to be large.
- You can think of the circuit as a controller in a feedback system. What is the gain of the circuit? Higher gains will give you higher speeds, but may cause oscillation. What controls the overall gain? How can you change the gain? Are there limits to the gains that you can choose in your circuit?
- The sensors you have in your circuit do not have identical response to light. How will this affect the behavior of the circuit? Can you compensate?

4.2 A reference design

In last week's lab, you built simple circuits to produce voltages proportional to the brightness of the light falling on each photoresistor.

One simple design for a pointing circuit is to place a voltage derived from one photosensor on one side of the motor and a voltage derived from the other photosensor on the other side. Here is the basic structure of the circuit:



The difference in voltages across the motor can be described as $k(V_L - V_R)$, where k can be thought of as a gain. Be sure that you understand how to adjust k in your circuit.

- Fill in the design, using op-amps and resistors to make sure you can get a good range of voltages across the motor. **Label each node in your diagram with a name you will use in the `Circuit` specification.**.

- Build a `Circuit` description for the circuit. You will need to use a head connector in your circuit, to give you access to the photoresistors and the motor on the head assembly.

```
Connector('Head', ['neck pwr', 'neck signal', 'neck gnd', 'left eye',
                   'common eye', 'right eye', 'motor pos', 'motor neg']),
```

  You can see a diagram of what's connected to each pin in the *Robot Infrastructure Guide* in the reference section of the course web page. You should connect any unused pins to the ground node.

- You can place your circuit definitions in `circuitSimulateTest.py`. We have placed the beginnings of such a definition (`eyeNeckCircuit`) in that file; you can test it using

```
eyeServo.testSignal(dist=3.0)
```

This input signal simulates moving a light back and forth (changing instantaneously between angles $\pi/2$ and π). The ideal circuit would track this input as closely as possible. Note that you can change the simulated distance to the light, by changing the argument `dist` in `eyeServo.testSignal(dist=3.0)`.

Check Yourself 9. Simulate your circuit at distances of 1 and 3 for gains 1, 5 and 10 (or a high enough gain so that it stops working). Why does it stop working?

Debugging: You might want to simplify your problem while debugging so that you can better understand what's going on.

- One way to simplify is to disconnect the motor in your circuit and place probes where the motor connections would be. That way you can observe what your circuit is "commanding" the motor without having any real motion.
- Another way to simplify is to use a simpler input signal; `eyeServo.simpleSignal(dist=3.0)` models a light source at a constant angle. You can use that with the motor connected to make sure that your circuit converges to the target angle.

Checkoff 3. Discuss your design and the result of your simulations. Be sure you can address all the issues in the Design Criteria section (speed, accuracy, stability, uniformity). **For your next interview, keep your `Circuit` definition, a circuit drawing with the same node labels as your `Circuit` definition, and the simulation results.**

Be sure to:
- Mail your plots and the `Circuit` definitions to your partner.
- Return both 8-pin connectors and the cables to the supply bins.

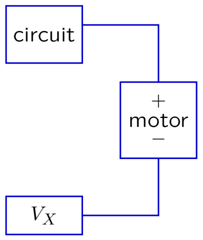
5 Upload Problems

The upload problems for this week are to finish and document two circuits for the pointing task. Upload a document that has the following components:

- A circuit drawing for the reference design. Include also a `Circuit` specification and your simulations for different gains and distances of the light.
- A circuit drawing for the alternative design described below. Include also a `Circuit` specification and your simulations for different gains and distances of the light.

5.1 An alternative design

We can use the bidirectional motor controller as the basis for an alternative design in which both photoresistors are used to compute a voltage on the positive terminal of the motor and there is a fixed voltage on the negative terminal of the motor.



There are a couple of key design issues in this approach:

- How should the voltage on the positive terminal of the motor relate to the amount of light on the photosensors (L and R)? Let's consider using a voltage that is proportional to $L/(R + L)$.

Check Yourself 10. How do we get this using a very simple circuit?

- With this voltage on one terminal of the motor and a fixed voltage on the other, write an algebraic expression for the voltage difference across the motor.
- You must ensure that you get bidirectional behavior – the motor must be able to turn both ways. You need to think carefully about the ranges of values of the voltages. Consider carefully how to introduce gains higher than 1. **The first problem on the Tutor homework exercises due this week offers an idea on how to do this.**

Check Yourself 11.

1. Draw your complete design and **label the nodes**.
2. Build a `Circuit` description for the circuit and simulate it at distances of 1 and 3 for gains 1, 5 and 10.

Be prepared to discuss its behavior as part of the first checkoff in Lab 9 (next week). Keep your circuit diagram, the `Circuit` description, and the simulation results for your next interview.