MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.01—Introduction to EECS I
Spring Semester, 2008

Design Lab and Homework 10, Issued Thursday, April 17

This handout contains design lab 10 as well as homework problems that are to be written up and handed in, together with questions from software lab 10, in your design lab on Thursday or Friday, April 24 or 25.

Please write up and hand in answers to the following questions: 7, 11, 12, 13, 14 and 15 from the design lab. Also, write up and hand in answers to homework questions 17 and 19.

Design Lab: Computer-based control

Do `athrun 6.01 update` and you will find the relevant files in the `lab10` directory.

This lab requires a laptop, a NIDAQ box, your head from last week, a power supply, and the usual circuit-building and testing stuff.

Warning: As you've seen in previous labs, some of the clip leads are bad. Make sure to test the ones you are using. You can just set your meter to the "sound" setting, and touch both ends of the clip-lead with the leads from the meter; the meter will beep if it's good.

Python to NIDAQ Input/Output

In this lab, and then in the final-project labs, we will be using the NIDAQ box for both input and output, between a Python program and a circuit. The NIDAQ box has the following ports that will be of interest to us:

- AIGND: analog input ground
- AI0—AI7: analog inputs 0 through 7
- AOGND: analog output ground
- AO0: analog output 0

The NIDAQ box can read voltage differences between AIGND and AI0, between AIGND and AI1, etc. It can actually set a voltage difference between AO0 and AOGND. It can actually read and set voltages between $-10V$ and $+10V$, but we will only use voltages in the range $0V$ to $+10V$.

You might wonder why, if the NIDAQ box can generate voltage differences, we need to use the power supply. The answer is that the NIDAQ box can only generate relatively small currents, but not necessarily large enough ones to drive our motor. Luckily, we know a way to convert a voltage sustained by a low-current source to that same voltage, but with more current: an op-amp connected to a voltage source that can supply more current. So, we'll use the $0V$ and $+12V$ from the power supply as the supply rails for op-amps in our circuit.

Staying Grounded

Things can get very confused here, very easily, because we have three wires labeled "ground": NIDAQ AIGND, NIDAQ AOGND, and 0V from the power supply. You cannot necessarily depend on them all being at the same voltage level (unless all of the devices are plugged into the the wall with a three-pronged plug). To keep all your "grounds" equal, you should start by connecting NIDAQ AIGND, NIDAQ AOGND, and power supply 0 together on the bottom board of your head. And you might as well connect the black lead of your meter to this value, while you're at it.

Read and writing

In lab 7, we read voltages from the NIDAQ box in a Python program. Now we'll see how we can use Python to command voltages to outputs on the NIDAQ box.

1. Connect a wire to the AO0 port of your NIDAQ box
2. Start the NIDAQ server by opening a new terminal window and typing

   ```
   > ./NIDAQserver --output
   ```

   Now, wait a somewhat uncomfortably long time, and numbers should start streaming out. Just leave that window alone.

   If you see a "fatal error"" message before the numbers start streaming, just ignore it.
3. Now, from IDLE, open the file `test.py`, and run it to load its definition. Type `writeTest()` at IDLE's shell prompt.

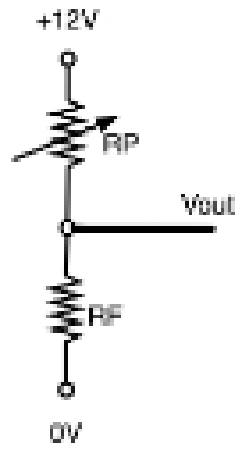
Here is the text of part of the `test.py` file. The `writeNIDAQ` procedure takes a voltage value and "writes" it out on the port AO0 of the NIDAQ box.

```
# Write values from 0 up to 9.5 volts to NIDAQ AO0
# Go in steps of 0.5, every 2 seconds
def writeTest():
    v = 0.0
    while True:
        print v
        writeNIDAQ(v)
        time.sleep(2.0)
        v = v + 0.5
        if v > 9.5: v = 0.0
```

You should see the voltage on the meter increment from 0V, in 0.5-Volt steps, every 2 seconds, until it reaches 9.5V, and then wraps around. You may notice that occasionally one of the commands is missed; this is a known problem, which won't affect subsequent parts of this lab, when we will be sending commands very frequently.

Python phototaxis

Now, let's see if we can replicate what we did in lab 9, having the head turn to follow the light, but with a Python program in control. It will be useful to put the photo-sensors in a resistor-divider configuration, like this:



Question 7: Design a circuit that will let you read the two values from the two photo-sensors on your head assembly, so they can be read in Python via the NIDAQ box. Remember that they need to be in the range 0V to +10V. You don't need more than the three wires that we have going up to the head; if you have a problem seeing how to do this, talk to an LA.

Checkpoint

Show your circuit design to your LA

It will simplify the process of doing this lab if you unwire the circuits that you have currently built on the bottom board of your head and start fresh. (You should leave the op-amp chips where they are; we'll be using those).

Question 8: Build your circuit on the bottom board of your head. Connect the AI0 and AI1 ports to the photo-sensor circuit outputs.

Question 9: The code below (included in `test.py`) repeatedly reads a list of 8 numbers from the NiDAQ box and prints the first two, which are the voltages between AI0 and AIGND and between AI1 and AIGND.

```
def readTest():
    while True:
        values = readNIDAQ()
        print values[0], values[1]
        time.sleep(1.0)
```

Run the program and try it. Observe how the output voltages change, and how they are affected when you shine a light on the head.

Checkpoint

Demonstrate reading the light sensors to your LA

Now that you can read the photo-sensors, it's time to turn to the task of driving the motor so that you can create feedback loop. Your program will read the sensors and use NIDAQ box to produce a voltage for the motor.

But there's a complication: The NIDAQ box can generate outputs in the range 0—+10V and the motor needs to turn in both directions. So, you'll need to generate a "virtual ground" again, this time, at +5V, and connect that to one side of the motor. Recalling tutor problem PS.10.1.5, you can use a 15KΩ resistor and a 10KΩ resistor to get the appropriate ratio in your divider.