MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.081—Introduction to EECS I
Spring Semester, 2007

Midterm assessment

Issued: Thursday, April 12
April 19 at the beginning of lab

We are calling this a midterm *assessment* rather than a midterm *exam*, to emphasize that the purpose of this assignment is primarily to help you assess how well you are understanding the course material, rather than to help us determine your grade. In point of fact, this "midterm" counts no more than any other weekly assignment in grading. We'll grade it with check, plus, and minus—just as with the other assignments—and we'll use it to give you some feedback on how you are doing in the course. The other purpose of this assignment is to give you some sense of the kind of thing you should expect on the final exam.

You should work on this as you would any homework assignment, using whatever resources you have available. But in order for this to be an accurate assessment of *your* understanding, we suggest you work on this assessment alone and not ask the staff or other students for help.

Be sure to explain your answers. Since we are using this assignment for diagnostic information, we care not only about the correctness of your answers, but also about the clarity of your explanations. (And please make your answers legible: anything we can't decipher is certainly unclear.)

Problem 1: Programming

SoaR, as you recall, contains a procedure `sonarDistances()`, which returns the list of distances reported by the robot sonars. For this problem, assume that there is only one sonar, and that the procedure `readSonar()` returns that sonar reading as a single number.

You've undoubtedly noticed that the robot sonars sometimes give flaky readings. So it might be useful to have a procedure like `readSonar`, but which returns the entire history of sonar readings, or the average of the sonar readings over several calls, or the minimum and maximum readings. In this problem, we'll ask you to implement such a procedure. Please use only the basic functions built into Python (e.g., don't use fancy packages you might find on the Web).

We'll start with the following class:

```
class rangeTracker:
    def __init__(self):
        self.saved=[]

    def values(self):
        self.saved.append(readSonar())
        return self.saved
```

The `rangeTracker` class behaves as follows (assuming that the first few sonar readings are 10, 9, 11, and 12).

```
>>> rt=rangeTracker()
>>> rt.values()
[10]
>>> rt.values()
[10, 9]
>>> rt.values()
[10, 9, 11]
>>> rt.values()
[10, 9, 11, 12]
```

1. Add methods to the `rangeTracker` class to return the maximum and minimum of the saved values (one method for each).

2. The *mean*, or average, of a set of N numbers $x_1 \ldots x_N$ is the sum of x's over the number of x's:

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

   Add a method to the `rangeTracker` class that returns the mean of the sonar readings.

3. The *standard deviation* of a set of numbers is a measure of how spread out the numbers are around the mean. It can be computed as

$$\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

   Add a method to the `rangeTracker` class that returns the standard deviation of the sonar readings.

4. The above parts of the problem asked you to add new methods to the `rangeTracker` class. An alternative way to add new methods—without modifying the original class—is to use inheritance.

   Define a new class called `trackerWithReset`, a subclass of `rangeTracker`, that has a `reset` method that reinitializes the list of saved values to the empty list.

In assessing your performance on this programming problem, we're interested not only in whether your programs returns the right answers, but in whether your code is and shows good style. The definition of "good style" is somewhat idiosyncratic, but for us, it includes things like

- Do you reuse pieces of code to achieve compactness and clarity (for example, does your code for computing the standard deviation make use of your code for computing the mean)?
- Do you take advantage of list comprehension, rather than writing explicit `for` or `while` loops?

Suggestion: It is *not* necessary for you to test and debug your code. We won't be taking off points for trivial syntax errors. Similarly, on the final exam, we may ask you to write code, but we certainly won't expect you to run and debug it.

If you do want to test your code for this midterm, however, you'll find that this may be awkward to do, since it relies on having `readSonar` available. For testing purposes, you could use a dummy procedure like:

```
counter = 0
def readSonar():
    global counter
    counter=counter+1
    return counter
```

This will return "sonar values" that are consecutive integers, and you can reset the `counter` to get repeatable results for testing.

Helpful hint: The mean of the set of numbers 1, 2, 3, ..., 10 is 5.5, and the standard deviation is 2.8723.

Problem 2: Feedback control

In lab for week 6, you programmed the robot to drive down the center of a narrow corridor, maintaining a desired distance d_{desired} from the center. You used a control algorithm based on the error

$$e[n] = d_{\text{desired}}[n] - d[n]$$

where $d[n]$ is measured distance from the center. You modeled the robot's dynamics by

$$\begin{aligned} d[n+1] &= d[n] + V\delta t\,\theta[n] \\ \theta[n+1] &= \theta[n] + \delta t\,x[n] \end{aligned}$$

where $\theta[n]$ is the angle of the robot heading, $x[n]$ is the robot's angular velocity, V is the robot's forward speed, and δt is the time step. You then implemented a control law that produced values for $x[n]$.

Suppose $x[n]$ is specified according to the control law

$$x[n] = Ke[n-2]$$

that is, x is proportional to the error two time-steps ago.

1. Give a polynomial whose roots are the natural frequencies of the system that relates the input sequence $d_{\text{desired}}[n]$ to the output sequence $d[n]$.
2. Suppose we choose K so that $KV(\delta t)^2 = 1$. What are the natural frequencies? Is the system stable? How about for $KV(\delta t)^2 = -1$?

Note: You'll probably want to use a computer or a calculator to find the roots of polynomials. The Python root finder we used in class can compute roots of polynomials only up to third order. For higher orders, you can use Matlab on Athena. To do this, log in to Athena, and at the Athena prompt, type

```
athena% add matlab
athena% matlab
```

and wait for Matlab to start up. Then to find, for example, the roots of $x^4+2x^3+3x^2+4x+5=0$, you'd do:

```
>>> roots([1,2,3,4,5])
ans =
   0.2878 + 1.4161i
   0.2878 - 1.4161i
  -1.2878 + 0.8579i
  -1.2878 - 0.8579i
```

Problem 3: Circuits

On March 22 in lab, you used the National Instruments box to measure the protoboard's voltage supply, which can generate voltages from 0 to +15 volts. You may have also, in the optional part of the lab, connected a Lego motor to the power supply and experimented with measuring the voltage and current through the motor.

As noted in the lab, the speed of the motor varies with the voltage applied. It also turns out that the torque of the motor is proportional to the current through the motor. If the current is too small, there won't be enough torque, and the motor will stall.

The lab (optional part) asked you to try driving the motor with +7.5V by setting the the protoboard at +15V and connecting the motor across two 1KΩ resistors. This was observed not to work. (See figure 1(a).)

1. The circuit that drives the motor consists of voltage sources and resistors. It is a two-terminal circuit that can be viewed as a 1-port, as shown in figure 1(b). Sketch the Thévenin equivalent model of the 1-port, giving the values of V_{TH} and R_{TH}.

2. Write a couple of sentences to explain, as clearly as you can, why the circuit of figure 1(a) fails.

3. Show how to use an op-amp to fix the circuit, i.e., to drive the motor with +7.5V, even though the protoboard source is set at +15V. Sketch the resulting circuit and explain clearly why it achieves the desired result.

4. Suppose we connect the motor to the Protoboard supply as shown in figure 2, where R_x denotes a potentiometer, i.e., a variable resistance.

   Say which (if any) of the following statements will be true, and explain why, as clearly as you can.

   - The voltage at the node marked `a` remains constant as R_x varies.
   - The current through the motor remains constant as R_x varies.

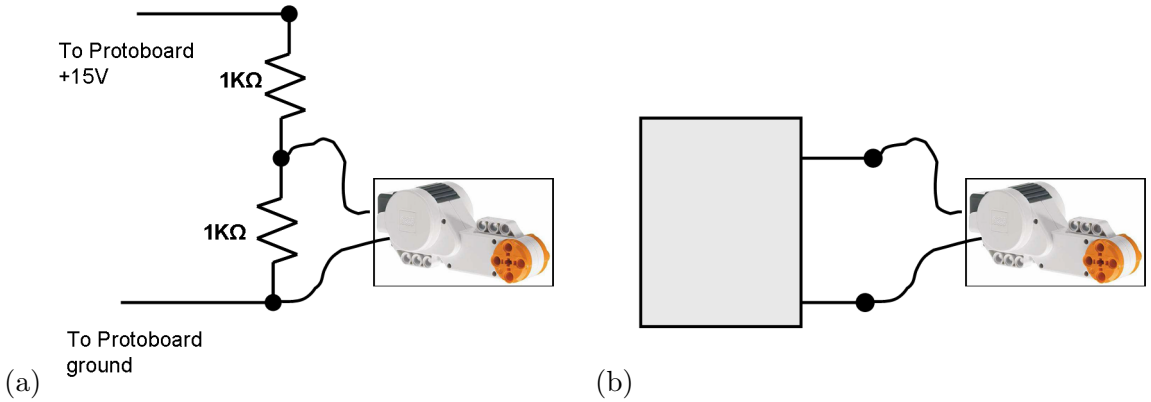

Figure 1: (a) The circuit you tried in lab, which did not work. (b) Abstracting the motor driver as a 1-port.

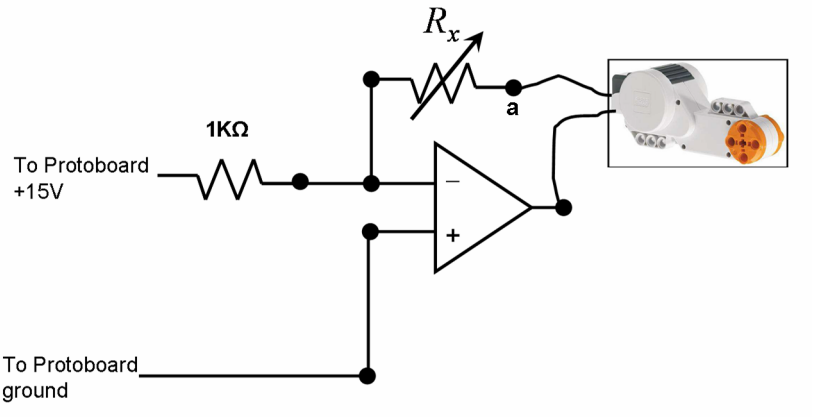

Figure 2: Hooking the motor to the Protoboard supply, using an op-amp.