CSCE 236 Embedded Systems, Spring 2012
Lab 4

In Class: Thursday, February 23, 2012

Names of Group Members:

1 Instructions

This is a group assignment to work on during class. You only need to hand in one copy of this, but make sure that the names of all of your group members are on this sheet to receive credit. Complete all of the sections below and make sure to get the instructor or TA to sign off where required.

2 Manual Analog to Digital Converter

In this section of the lab, you will manually configure the analog to digital converter (also know as ADC or A2D). You will then test it by creating a voltage divider and reading the values at different points on the voltage divider.

Refer to the datasheet and configure the ADC in the setup function by:

- Setting the ADC voltage reference to AVcc. This connects the voltage reference to the external voltage supply, which on the Arduino is 5V.
- Enabling the ADC by setting the appropriate bit in the `ADCSRA` register.
- Setting the ADC clock prescaler to an appropriate value for the Arduino. Refer to table 24-5 in the datasheet. As noted, in section 24.4, the successive approximation circuitry in the ADC requires an input clock frequency between 50kHz and 200kHz.

Now create a voltage divider on your bread board using 3 different resistors (of your choosing, whatever you have available). Connect 1 end of the divider to ground and the other to the 5V power rail. Now connect an ADC port to each of the 4 different voltage levels created by the voltage divider (GND, between R1 and R2, between R2 and R3, and 5V).

In your loop function, write code to read each of the 4 ADC values and output the four values over the serial port. To read the ADC, you must:

- Set the `ADCMUX` register appropriately (do not overwrite the voltage source bits).
- Start the conversion (look for `ADCS` bit).
- Wait for the conversion to complete (by monitoring the `ADCS` bit).
- Read the value out of the `ADC` register.

Checkoff: *What clock divider did you end up using and what is the final ADC clock value?*

Checkoff: *How do you convert from the ADC value to the actual voltage on the pin? Do the voltages you read on your voltage divider match the theoretical values given the resistor values you have?*

3 Reflectance Sensor

In this section of the lab, you will hook up infrared (IR) reflectance sensors to your robot. These sensors emit an IR light and then measure how much is reflected back. If the IR sensor is over a white surface a lot of IR will be measured, while if it is over a black surface little will be seen. The distance to the surface also makes a difference.

For this part of the lab, you can either read the ADC manually as you configured it above, or use the Arduino `analogRead(...)` command. I would recommend using the `analogRead(...)` command or creating functions that encapsulate the ADC reading code above. Regardless, you are expected to know how both work.

The reflectance sensors have three wires. Black should be connected to ground, red to 5V, and the green wire to one of your ADC ports. Perform basic experiments to characterize the performance of the sensors. You can use the black line on the following page as a reference.

Checkoff: *As you move the sensor closer to a surface, does the ADC value increase or decrease?*

Checkoff: *How close do you have to be to an object to have a difference of more than 100 ADC values between a black and white object?*

4 Project 1 Competition Overview

The first project competition will be held Tuesday, March 6, 2012. A written report will be due Thursday, March 8, 2012. Additional details on this competition and the report will be forthcoming.

As an overview, this competition will involve having your robot follow a line as quickly as possible using the IR sensors. In addition to speed, your robot will be judged on the smoothness of operation. Your robots will compete against each other on a variety of courses containing different types of curves and turns.

Start by mounting the IR sensors on your robot as demonstrated in class. Then write code that will turn your robot towards a line if one sensor sees the line and the other does not or to drive straight if both sensors see the line. Start by using slow forward and turning speeds, then try increasing the speeds so you can operate faster. Initially you should transition between three different driving states: straight, left turn, and right turn. Once you have this working, you can try to create smoother turns by doing a more gradual transition between these states or based on the values of the sensors. You should also consider what action to take if your robot completely looses sight of the line. You can use the line on the following page as a reference.

Additional details on this competition, including questions you should answer will be given next week. Note that this is an individual assignment, but you can use time in lab to get started.