Lab 8: Arduino Smart Car: the BOE-BOT

"Let's not kid ourselves here; robots already run most of our world. We'll be their butlers soon enough." -Eric Stoltz

1. Objectives

This lab gives you a chance to combine many things you already learned in ES 50 into an autonomous moving electronic system – BOE-BOT. In particular, you will use feedback from sensors to provide inputs to the brain of the whole "operation" – Arduino – that will then take necessary action to navigate a BOE-BOT. You will:

- Play with an electronic system that uses sensors, feedback, microcontrollers and servo motors
- Program your system to navigate a track and follow a path

This lab will hopefully provide additional inspiration for your final projects ☺.

2. Background

The BOE-BOT - a simple robot designed for educational use - consists of an Arduino microcontroller that is connected to two servo motors (from now on called servos ☺) and an under-carriage array of infrared transmitters and sensors. Here, we discuss these three parts in more detail:

The Arduino: By now you should be well- acquainted with the Arduino, which serves as the 'brains' of the BOE-BOT, taking in information about the surroundings from the sensors, and using that information to direct the BOEBOT's motion through the help of the servos.

Figure 1: The BOE-BOT platform, together with Arduino and breadboard

The servos: A servo is a special type of motor system that is used to achieve precise rotations in hobby and other projects (ex: the steering in a radio-controlled (RC) car, the surfaces on an RC airplane, etc). Each servo consists of a motor, a gearing system, and a controller. Based on the system input, which comes in as a periodic series of pulses, the controller will adjust the system so that the output shaft is rotated to a particular position. Normally, servos only have a limited degree of rotational freedom (90 -180 degrees). The BOE-BOT servos, however, have been modified to rotate 360 degrees so that they can power the robot's wheels.

Note: In the ES 50 lab we have 50-ish servos ready for all kinds of projects. ☺

The IR sensor array: This consists of three infrared transmitter/sensor pairs on the bottom of the BOE-BOT. For this lab, the infrared transmitters are basically just LEDs, and the sensors are photo-transistors (you will work more with these in laser lab). The pairs are located at left, center, and right positions relative to the BOE-BOT chassis, and they can give positional feedback to the microcontroller. The transmitter/sensor pairs work in the following way:

- Each transmitter sends out pulses of infrared light that are reflected by whatever surface the BOE-BOT is traveling over and received by the corresponding sensor.
- Each sensor compares the strength of the received pulse to a pre-set threshold, and then sends a 1 or a 0 back to the microcontroller based on the result of the comparison. The thresholds on these particular sensors have been calibrated to distinguish between light and dark surfaces.
- In this lab, you will program the BOE- BOT control to follow a line. Here, the Arduino will be the controller and the transmitter/ sensor array will provide feedback. The motion of the servos (and the attached wheels) will be the system output.

3. Bringing the BOE-BOTS to life, and helping them take their first steps

Please follow the steps below to complete this lab exercise. Lettered points contain questions that need to be addressed in your lab write-up.

Awakening the BOE-BOT:

1. Examine your BOE-BOT. Identify the Arduino, the power input, the servos, and the under-carriage infrared transmitter/sensor array.
    a. What pins on Arduino are used to connect the infrared sensors?

2. Download the *BOE_BOT_simple.ino* program from the website and load it into Arduino. This program contains the framework necessary to write some simple programs that will control the BOE-BOT servos.
    a. What does the BOT do when the given code is executed?

First Movements:

3. By adjusting the commands in the maneuver function, you can control how each servo rotates. Take a look at where maneuver is defined at the end of the code to gain a better idea of how this works. Now alter the code to make the BOT do some turns.
    a. What commands are necessary to make a sharp left turn?
    b. What commands are necessary to make a gentle right turn?

Maneuvering:

4. Now it is time to see how using feedback can make our BOTS much more autonomous. Open the *BOE_BOT.ino*. You will notice this code is similar to the one you just looked at. There are, however, more lines of code that deal with the signals received from the IR

sensors. While it is not necessary to understand all of this code, make sure you get a decent idea of the functionality before moving on.

a. Google the function *writemicroseconds()*. An entry from the Arduino website should be easy to find. What does this function do? The code that sets servo speed is as follows:

```
servoLeft.writeMicroseconds(1500 + speedLeft); // Left servo speed
servoRight.writeMicroseconds(1500 - speedRight); // Right servo speed
```

b. Why does the left servo add your speedLeft value while the right servo subtracts the speedRight value?

Hint: The maneuver function is written as follows: maneuver(x, y, z);

c. What are x, y, and z? (Hint: Look at the definition of maneuver() at the bottom of the code. Your answers should be more than just variable names.)

5. Upload this code to your BOT. Once the upload is complete unplug the USB cable to keep the BOT from trying to take a trip off the bench. Notice that in the Arduino program window right below the text editor, the size of your program and the memory capacity of the Arduino is listed.
   a. What is the total size of the BOE-BOT program(in bytes)?
   b. What is the total size of the Arduino memory(in bytes)?
   c. What fraction of the total memory is taken up by our program?

On the Track!

6. Now take the BOT to the track. Place the BOT with the center sensor on a straight piece of track. When you are ready for the BOT to run (!), plug in battery pack.
   a. What does the robot do and what happens when it reaches a corner?

7. Its now time for your BOT to learn how to make it around the track. Notice that we have only written the code to direct the BOT if the center sensor reads black and the other sensors read white. Otherwise the BOT stops dead. Your mission is to write in and specify the other necessary conditions in order for your BOT to quickly traverse the track.
   Note: One point will be awarded to the fastest two teams in each lab section.

   a. In your lab report, insert a print out of the code you used to make your BOT navigate the track.

   b. Make sure that your TF records the time it takes your robot to complete a circuit unassisted and then upload your time to the ES50 BIG BOARD. Eternal honor and glory (and the above, by default) will be awarded to the fastest team in ES50!

Planning for BOE-BOT Version 2.0

8. Even if your BOE-BOT beat all the competition in the track race, as an engineer (and/or entrepreneur), it's time for you to begin thinking about the next, improved version of your BOE-BOT. This involves your own analysis of problems encountered and solutions that you recommend.

    a. Describe any problems you encountered in the performance of your BOEBOT. This could include things like not seeming to respond in the way you thought you programmed it to, stopping all of sudden, running off track....whatever you noticed.

    b. Now describe your solution in enough detail that someone else in the class could read your notes, pick up your BOT and 'make it better'. Does the 'fix' have to do with software (improving the code), hardware (changing the sensors or detectors, or changing their positions), the environment (i.e., the track) or maybe the 'brains' (Arduino)?

    Be sure to submit your answers to these questions to your TF in your lab report.

Optional: Adding new sensors and giving your BOTS greater capabilities.

What you've done in lab today can be just the beginning: we want you to realize that there is a VAST world of possibilities once you connect Arduino controllers/brains to different kinds of sensors and actuators.

For example, in the lab we have some other BOE-BOT-compatible sensors, such as the inexpensive Parallax "Ping" sensors (Google it!) that work by sending out an extremely high-frequency sound wave and 'wait' for that signal's echo to return to the sensor; since the sound wave travels at a constant speed, by counting the time between the transmission of a signal and the reception of the "echo," the Ping sensor is able to keep very accurate track of distance. This kind of 'echo-location' scheme can allow your BOTS to go 'off-track' in a good way: they can now sense and (through the magic of the Arduino brain) avoid walls, obstacles and each other (!).

Most of the work you have to do in adding new sensors to BOE-BOTS is in the reprogramming the BOT's brain in order to respond to a world seen and sensed by the new sensor. If you have time have fun with this.

Note: we did not provide any instructions on purpose. It is time for you to start doing some independent work as prep for final project. Have fun! ☺