ME 224 Final Project – Spring 2008

1. Objectives:
   - Design and implement an experiment to realize the control of a commercially available robot
   - Enhance your knowledge of LabVIEW, data acquisition, feedback control, and MEMS sensing

2. Methodology:
   - Sensing: Familiarization with MEMS gyroscope sensor to obtain the information of angular motion – this involves calibration of the gyroscope
   - Actuation: Assembling of the Boe-Bot, centering of the servo motors and programming to control its motion
   - Control: Using LabVIEW and the data acquisition card for feedback control and data acquisition to control the navigation of robot along a user input path (see examples below)

3. Outcome:
   - The outcome of this final project will be a LabVIEW user-interface program, which can control the robot to travel along any specified path.
   - A final report and a final exam with on site demonstration
   - During the final exam the instructor will ask questions about the project to any of the students in the group. Each student will be asked specific questions related to the project activities.

4. Schedule:
   - Groups will be assigned by Friday, May 23rd. Each member of the group may take responsibility for a particular activity BUT all team members must become familiar and knowledgeable to be able to answer final project exam questions. Each group should contact the TA to obtain the needed materials such as robot kits and gyroscope sensors as soon as possible. (These material should be returned to TA at the end of the class)
   - From next week, each group will present final projects activities (see below). Half of the groups will present on Mondays and the other half on Wednesdays.
   - Demonstrations will take place the last week of classes. The final report is due the Friday before finals week.
   - The final project exam (closed book) will take place on June 12th from noon to 2 PM.

5. Reference:
   - The Manual of Gyroscope sensor (ADXRS150EB)
   - The Manual of Boe-Bot Robot Kit
   - Final project report of spring 2004 (http://clifton.mech.northwestern.edu/~me224/)
6. Materials:
   - LabVIEW and data acquisition card (available in B100, Ford Building)
   - Gyroscope sensor (ADXRS150EB) from Analog Devices
   - Boe-Bot Robot Kit ([www.parallax.com](http://www.parallax.com), #28132)

Routes, to be followed:

1) [Diagram]

2) [Diagram]

3) [Diagram]

4) [Diagram]
Boe-Bot Control - Activities

Introduction
In this lab you will use all of the skills you have developed throughout the quarter in order to program a small robot (Boe-Bot, Parallax Inc.) to follow a desired path. The main elements involved in this process are the BASIC Stamp microcontroller, the PBASIC programming language, electromechanical servos, a MEMS gyroscopic sensor, and, of course, LabVIEW as a user interface and control system.

Main Components

**BASIC Stamp**
BASIC Stamp is essentially a small microcontroller that contains a CPU, memory, a clock, a power supply and external input/output. When working with small RC models, BASIC Stamp is preferred because of the ease of use of the PBASIC programming language.

A Boe-Bot is a small scale robot with servo motor control that is simplified by the use of BASIC Stamp. The program controls the speed of the servos by controlling its output signal pulse time. Shorter pulses results in more servo movement and thus a higher speed in one wheel. Once servo manipulation has been mastered, a program can be written with different tasks where each performs a specific motion such as forward movement or turning right/left. This then is compiled onto the BASIC Stamp chip on the Boe-Bot so that a virtual code of commands is available on the robot.

**Gyroscope sensor (ADXRS150EB)**
The gyroscope sensor provided on the Boe-Bot from Analog Devices is the first commercially available surface-micromachined angular rate sensor with integrated electronics (Figure 1).

![Figure 1: The ADXRS150EB chip from Analog Devices, mounted on the Boe-Bot.](image)

This sensor measures the speed at which an object is turning in the yaw direction (rotation about the z axis, normal to the plane of the board). The angular rate measured is quantified by an output voltage that is proportional to the set sensitivity of the gyroscope. The full scale capability of the sensor is limited to 150 degrees/second which limits the angular rate that the
gyroscope can detect. However, the sensitivity of 12.5 millivolts/degree/second makes the output voltage readings very easy to distinguish.

In this lab, the angular position of the Boe-Bot will be important as an indication of whether or not the robot is staying on course. This can easily be obtained by integrating the angular rate measured over time. This calculated angular position relative to its starting orientation can then be manipulated to yield the difference in the object’s instantaneous position from the starting position in degrees.

**LabVIEW and the Data Acquisition Card**

LabVIEW, a visual programming language, will be used to control the Boe-Bot in this lab. Your program should include a user interface for entering desired paths, control of the robot in response to entered paths, and correction in movement based on angular position readings from the gyroscope. The details of how to complete these programs are explained in the activities section.

The National Instruments Data Acquisition Card (DAQ), as shown in Figure 2, will be used to communicate between the LabVIEW program and the robot’s BASIC Stamp chip and gyroscope. Digital outputs are connected to the robot so that logical high outputs can be delivered to a specified port to activate a response from the BASIC Stamp program. The analog readings of the gyroscope are also read via the data acquisition card and interpreted in LabVIEW.

![Figure 2: The National Instruments DAQ.](image)

**Summary**

Now that each main component has been discussed, it is important to step back and realize the main purpose of each one. The BASIC Stamp chip and PBASIC language are used to establish servo control, LabVIEW is the main interface and command of the robot, the DAQ transfers information from the gyroscope (which provides sensing capabilities) to the computer, and also
transfers commands from LabVIEW to the robot. With all these working correctly in conjunction, the end result is a robot that responds to inputs and travels in the specified direction.

Activities
The following activities describe the necessary steps to becoming familiar with the robot, and then put the pieces together so that a working robot can be tested.

Activity 1: Servo Manipulation
A servo is a device used to provide control of an operation through feedback. In this case, the operation being controlled by the servo is the rotation of a wheel and therefore the movement of the robot. The feedback is the user input of the time delay between voltage pulses to the motor. This controls the speed at which the servo rotates. This in turn controls the speed and direction of the robot. Now that the mechanics are understood, the next step is to develop the feedback program to control the servos.

The servos are controlled by PBASIC. Only five lines of code are necessary to manipulate the wheels. In this activity, the calibration of the servo will be done and BASIC Stamp programming will be applied to control the rotation of the corresponding wheels.

![Figure 3: Boe-Bot wheel attached to servo.](image)

Activity 1.1: Rotating the Wheels
The PBASIC code necessary to rotate the wheels will be written. Before beginning it is important to recognize that the PBASIC PULSOUT command’s duration argument, if greater than 750, will cause the servo to rotate counterclockwise. If it is less than 750, the servo will rotate clockwise. The two servos are attached to pins P12 and P13.

Procedure
1. Attach the serial cord to the robot and the other end to the computer.
2. Open the BASIC Stamp editor, which can be installed from the included CD.
3. Write a simple program to control one servo (P12).
4. Repeat step 3 for the P13 servo by changing the 12 in the PULSOUT command line to 13. Observe rotation in both directions.

5. Now that you are familiar with the PBASIC code required to have the servos rotate, we will move on to having the servos rotate at the same time.
   a. Amend the program in step 3 to now include both servos.
   b. Change the duration of servos 12 and 13 to make the robot move in a reasonably straight line.
   c. Try setting it so the robot turns (servos rotate in opposite directions).
   d. Adjust the speed of the servos by changing the PULSOUT values of both servos.

**Activity 1.2: Calibrating the Servos**

Now that servo speed and direction has been studied you must be able to stop their rotations as well. This is done by changing the duration argument for each servo (you must determine the value for each servo). Normally, the servos can be manually calibrated by adjusting the potentiometers. Since they are hard to reach, it is easier to simply find the value which corresponds to zero rotation for each servo.

**Procedure**

1. Using the code formulated in Activity 1.1 study each servo independently until the correct duration value is found.
   a. Write the code to control the rotation of Servo 12.
   b. Enter an estimate for the duration value that will stop rotation.
      i. Change this value until the servo stops rotating, and record the value.
      ii. Repeat this for Servo 13 and record this value as well.

2. Store this information somewhere convenient. It will be needed in later parts of the lab. This information is the calibration of your servos. At this value, durations larger will result in counterclockwise rotation and smaller will result in clockwise rotation.

3. At this point, there are a lot of numbers you need to know… full speed forward and reverse durations, servo numbers, stopping durations, etc. Now put them all in your program as variables so things are easier to understand and use.
   a. The BASIC Stamp manual is posted on the course website on Clifton
   b. Search for the section on variables (pay attention to the data types!).
   c. Make variables for all the following (one each for the left servo and right servo):
      i. Servo number
      ii. Full speed forward
      iii. Full speed reverse
      iv. Stop
   d. Use the variables in your program to control the robot. Now if you need to change, say the stop duration, you only need to change it in one place.


**Activity 2: Gyroscope Calibration and Programming**

**Activity 2.1: Gyroscope Calibration**

The gyroscope is a MEMS device that measures the angular velocity about its z-axis (normal to the plane of the board). It operates similarly to an accelerometer, but responds to changes in angular displacement, rather than changes in translational velocity. The gyroscope’s internal circuitry converts this displacement to a voltage proportional to angular velocity. This activity will help you connect the gyroscope and calibrate the output voltage to angular velocity.

**Procedure**

1. The gyroscope is mounted on a 20 pin ADXR5150EB evaluation board. The pins are numbered 1 (starting from the square solder pin) through 20, counting counterclockwise. Be sure to include the four spaces without a physical pin in the numbering.
   a. Connect the pins according to the gyroscope data sheet.
   b. Check the high- and low-voltage self-tests at pins 10 and 11 individually and record the RATEOUT response.
   c. Repeat for pin 11.

2. Calibrate the output voltage (RATEOUT) to angular velocity. Notice that as you rotate the gyroscope in the z-axis direction, the voltage changes from 0V to 5V. At no angular velocity, the output voltage should be around ~2.5V. Because the output voltage is proportional to angular velocity, the equation for angular velocity should be as follows: $\omega = \alpha \nu + \beta$ where $\omega$ is angular velocity in rad/s, $\nu$ is output velocity from the gyroscope, and $\alpha$ and $\beta$ are constants. What characteristics do you think these constants signify?
   a. Build a LabVIEW program that will record an input voltage every .01s for about 5–10 seconds and write them to a spreadsheet file.
   b. Connect the RATEOUT to the DAQ.
   c. Place the gyroscope or Boe-Bot on a rotating surface (e.g., swivel chair) and make sure that the gyroscope’s center and z-axis are aligned with the axis of rotation. Note that since the gyroscope can’t be centered on the Boe-Bot’s axis of rotation, you can’t just write a program to make the Boe-Bot spin and take measurements, even though that would be preferable if possible.
   d. Run the LabVIEW program without rotating the chair and save the spreadsheet file. Use Excel to find the average output voltage and plot this on a graph as 0 rad/s.
   e. Repeat step d, but this time rotate the chair at a fixed angular velocity for 180 degrees in the clockwise direction (do the best you can). Find the average voltage and the average angular velocity and add this test point to the graph.
      i. You will want to use a timer in LabVIEW to time how long it took to rotate the gyro 180 degrees. Then the average angular velocity will be $\omega_{avg} = \pi/ t (\text{sec})$.
      ii. Take the entire voltage history during the spinning and average all the points. This is $V_{avg}$.
   f. Repeat step e several more times using different clockwise angular velocities.
g. Repeat step e several more times using different counter-clockwise angular velocities.

h. Decide on a sign convention to distinguish counterclockwise and clockwise directions.

i. Once the points are all plotted on the same graph of $\omega_{\text{avg}}$ vs. $V_{\text{avg}}$, find the linear best fit and record the constants $\alpha$ and $\beta$.

**Activity 2.2: Getting angular displacement from angular velocity**

In this activity you will take the equation from Activity 2.1 and implement it into a LabVIEW program that will output the angular displacement. Recall that angular displacement is found by taking the integral of the angular velocity.

$$\theta(t) = \int_0^t \omega d\tau + \theta_0$$  \hspace{1cm} (1)

where $\theta_0$ is the initial angular displacement.

**Procedure**

1. Build a LabVIEW program that uses the equation found in Activity 2.1 to take voltage from the gyroscope as an input and output angular velocity.

2. Use a for loop and shift registers to integrate the angular velocity and output angular displacement.
   a. The easiest way to do this is to multiply the angular velocity by $dt$ during each iteration and add it to the previous iteration. Use the same numeric constant to feed the time delay and this multiplier, in case you need to adjust the time interval.
   b. Convert the angular displacement to degrees.

Note: $dt$ is based on the time between iterations. If this part is running concurrently with other tasks and functions in a larger LabVIEW program (and it will in later sections), the $dt$ value might need to be increased so that other parts of the program have time to execute. Experiment with larger delay values like .03s or .05s until the program operates correctly.

**Activity 3 - Putting it together in LabVIEW**

**Activity 3.1: Controlling BASIC Stamp with LabVIEW**

This activity will guide you through installing a BASIC Stamp program into the Boe-Bot, and then creating a LabVIEW program to control the Boe-Bot through outputs that will act as inputs for the BASIC Stamp. The BASIC Stamp Syntax manual is posted on the course web site on Clifton.

**Procedure**

1. Define 4 Inputs, 0-3, which correspond to the I/O pins of the BASIC Stamp chip.
   *(Note: This step is just a precaution. Basic assumes the I/O pins are inputs unless specified as output)*

2. Use If statements that will execute a command depending on the input
3. Here your variables will come in handy. Define 4 commands: Stop, Forward, Left, and Right, which do the following (Note: When defining the Stop command, do not write “Stop” into the program, as this is its own command in the Basic language. Use “Pause,” “Sto,” or “Still.” Similarly, if you have a variable called “Left” or “Right,” you may need to change those function names.)
   a. STOP: Set both servos to their stop durations.
   b. FORWARD: Set both servos to their full-speed forward durations.
   c. LEFT: Set the right servo to full speed forward and the left servo to full speed reverse.
   d. RIGHT: Set the left servo to full speed forward and the right servo to full speed reverse.

4. Upload the BASIC Stamp into the Boe-Bot and disconnect the serial cable. (Note: The BASIC Stamp code is saved onto the Boe-Bot in its memory. Unlike RAM, turning off the Boe-Bot will not erase the code)

5. Create a LabVIEW program that will take an input of initial distance (in iterations) and then stop after the “for” loop is complete.
   a. Create a Stacked Sequence with two frames.
   b. In the initial frame, create a “for” loop with the number of iterations as an input. Inside the loop, create four digital outputs 1-4, and set the output that corresponds to the forward command in BASIC Stamp to True. Set the others to False. Create a timer set to 20ms (to send the pulse every 20 ms).
   c. In the second frame, simply set the output that corresponds to STOP in BASIC Stamp to true, and set the others equal to false (Note: Without this step, the Boe-Bot will keep moving indefinitely).

6. Connect the DAQ so it outputs digital signals from LabVIEW to the Boe-Bot
   a. Connect 4 wires to the DAQ outputs you specified in LabVIEW
   b. Connect these wires to the corresponding I/O pins of the BASIC Stamp. These are labeled P0 to P3 on the board.

7. Test the LabVIEW program and verify that it controls the Boe-Bot

**Activity 3.2: Distance Calibration**

In this activity, you will use the LabVIEW program you created in Activity 3.1 to experimentally find the number of for loop iterations that correspond to different distances. You will plot these points to find the equation that relates number of iterations to distance, and modify the LabVIEW program to take distance instead of number of iterations as input. At the completion of this activity, you should be able to specify a distance for the Boe-Bot to travel.

*Note: Here you may have the same problem with timing inconsistencies as noted in Activity 2.2.*

**Procedure**

1. Relate distance to number of iterations
   a. Set the Boe-Bot where it can roll in a straight line. Mark its starting point.
   b. Input any reasonable number of iterations into the distance input of LabVIEW, and run it. Measure how far the Boe-bot moved.
   c. Repeat several times, then change to a different iteration value and repeat several times.
2. Finding Best Fit/incorporating Relationship into LabVIEW
   a. Graph the results and find the best fit. What type of regression do you expect? Should you set the y intercept to be 0? Why?
   b. Create a formula node that will take an input distance and convert it to number of iterations and input it into the “for” loop.

Activity 3.3: Controlling Turning
In this activity, you will use the VI you created to take the real angle measurement of the gyroscope to make turns. Instead of using a “for” loop with iterations to control the input into the Boe-Bot, you will be using a while loop, which will make the Boe-Bot turn until it has reached a specified angle.

Procedure
   1. Copy the VI from Activity 2.2 which outputs the current absolute angle of the gyroscope. Insert it into the LabVIEW Program from Activity 3.2.
   2. Create two more frames in the stacked sequence
   3. In the first new frame, create a while loop around the 4 digital outputs.
   4. Create a desired angle input, and set up the while loop so it continues while the gyroscope angle is less than the desired angle.
   5. Set the output corresponding to right (or left) to true, set all others to false.
   6. Go to the last frame and set the Stop output to true and the rest to false.
   7. Test your program!

Activity 3.4: Feedback Control
In this activity, you will use the absolute angle reading from the gyroscope to detect if the Boe-Bot is going off its straight path during straight-aways. This activity will require you to edit both the BASIC Stamp and LabVIEW program from the previous activities, so that if the gyroscope detects the Boe-Bot is veering off course, the Boe-Bot will turn slightly to correct it.

Procedure
   1. Create control conditions in BASIC Stamp
      a. In the BASIC Stamp program you created in Activity 3.1, create 2 “If” statements under the Forward condition so that these conditions will run only when the Forward input is true (when the Boe-Bot is traveling along a line)
      b. Create If statements that execute commands SlightL and SlightR for IN2=1 and IN3=1 respectively. Remember, these commands will be executed only if the Forward condition is true as well (IN1=1).
      c. Define these commands so that SlightL will make the Boe-Bot veer left (to correct for veering right) and SlightR will make it veer right. *(Hint: There are two ways to do each. You can either slow down one servo or increase the speed of the other)*
   2. Edit the LabVIEW program to output conditions to the BASIC Stamp to correct the Boe-Bot.
      a. Go to the frame in the stacked sequence that controls the distance.
      b. Determine a buffer angle so that if the Boe-Bot is traveling in a direction higher than that angle, the correction procedure will execute *(Note: Setting this
too high will make the Boe-Bot zigzag, and too low and the correction procedure will constantly occur; both may affect distance)
c. Edit the digital output 2 such that if the Boe-Bot is traveling in a direction greater than the buffer angle plus the desired angle, it will send a 1 to IN2 to correct by veering left (Error! Reference source not found.).
d. Do the same for output 3, except now the condition is if the direction is less than the desired angle minus the buffer angle, it will correct by veering right.

Note: Remember to leave the forward output at true, as the correction conditions only run if the forward input (IN1) is true as well.

Activity 3.5: Traveling along a pre-determined path

In this activity you will control the Boe-Bot to travel along a pre-determined path, with all the distances and angles as inputs in LabVIEW. Simply add frames to the stacked sequence (however many you need to travel the route). The sequences should follow the pattern: distance, stop, turn, stop, etc. There are two things to watch out for:

1. The angles you input for turning have to be absolute. If the Boe-Bot has already turned 90 degrees and you want it to turn 90 degrees again, you must enter 180 into the second angle input.
2. The feedback control for the straight-aways must be edited such that the desired angle is the last angle you put in. Again, it’s all absolute angles.

Note: Adding the gyroscope measurement VI into your LabVIEW control program may alter the duration of the time step, which will affect how far the Boe-Bot travels. If you find that after Activity 3.3 and Activity 3.4 the distances or angles are unreliable, you may have to repeat Activity 3.2 again to find the new relationship between iteration and distance, or fiddle with the time delay.

**Activity 4: Test Run**

In this activity you will use the code that you have developed to test whether or not your robot is traveling the desired path.

**Procedure**

1. The path is provided here.

   ![](image)

   2. Calculate the angle of the third turn and the distance of the final straightaway.
   3. Using the information provided in the last activity, program the robot to follow this course.
   4. Observe, and correct values until the robot is able to adequately follow the course.
Activity 5: Customizing

Your Boe-Bot has now been controlled successfully. As a final step, you will be assigned some level of customization. For example, repeat activity 4 with a user input (arbitrary) path, optimization for speed, optimization for accuracy, etc. Tune the numbers you used, change the algorithms, or otherwise alter the setup to achieve this goal.