Boe-Bot Final Project

Final Report

Mechanical Engineering 224
Professor Espinosa

Thursday December 8, 2005

Ashley Nidiffer
Tom Cohlmia
Eric Yim
Scott Moffat
# Table of Contents

I. Introduction  
II. Boe-Bot Assembly and Servo Adjustment  
III. Gyroscope Calibration  
IV. Communication  
   a. Serial  
   b. Wireless  
V. Programming  
   a. Basic Stamp  
   b. Labview  
VI. Conclusions  

**Appendices:**  

Appendix A: Gyroscope Calibration Program (BASIC Stamp)  
Appendix B: HyperTerminal Program (BASIC Stamp)  
Appendix C: Labview Program to establish Bluetooth Connection with while loops.  
Appendix D: Labview Program to establish Bluetooth Connection with for loops  
Appendix E: Calibrated Labview program to establish Bluetooth connection and create a final user interface.
**Introduction:**

The goal of this project was to communicate with the Boe-Bot robot such that we could get it to travel along a pre-determined path, such as that seen in figure 1. To do this requires many distinct disciplines (electronics, mechanics, computer programming and wireless communication) working together side by side. The goal is to communicate from the computer to the robot wirelessly in order to inform the robot what path it needs to take. Further, the gyroscope on board the Boe-Bot will communicate, using a closed-loop feedback system, in order to keep the Boe-Bot on a straight path.
Boe-Bot Assembly and Servo Adjustment:

Fortunately the Boe-Bot came mostly pre-assembled. The main components such as the chassis and board of education had already been connected with four support posts. However, during the process of the project, it was necessary to uninstall and install a new board of education as well as fully secure the servomotors after they had been centered. Centering the servomotors was really the first task for the group. Using the attached BASIC stamp program, we ran each servomotor and adjusted the internal potentiometer so that the motors stopped turning. This provided that PULSOUT 12, 750 would be the center position. The 12 refers to which motor (R or L) and 750 refers the angular velocity. A number higher than 750 would turn the motor one way, while a number below 750 would turn the motor the other way. We next secured the servomotors to the chassis with four screws and attached the wheels to those motors. Now our Boe-Bot was ready to move.

![Boe-Bot Assembly and Servo Adjustment](image)

Figure 2—Boe Bot
Gyroscope Calibration:

The gyroscope on board the Boe-Bot outputs a voltage between 0 and 5 volts depending on the angular velocity of the Boe-Bot. The gyroscope outputs a voltage of 2.5 volts at rest and increases this voltage output if the Boe-Bot rotates one way, while decreasing the output with rotation occurs in the opposite direction. In order to make this voltage output useful, we needed to calibrate the gyroscope so that we would know what each voltage value meant in terms of an angular velocity.

We first used BASIC stamp to run the program seen in Appendix A to rotate the Boe-Bot at increasing angular velocities. As the Bot rotated, we used Labview to record the voltage outputs of the gyroscope in real-time and write them to an excel file. We rotated the Boe-Bot from the digital velocity values of 700 up to 800 for 20 seconds each. During each trial we recorded the number of Bot rotations such that we could calculate an angular velocity in terms of degrees/second. A sample of the plot obtained from one of
these trials can be seen in Figure 3. We took the average of the voltage readings taken in each trial and plotted them versus the observed Bot angular velocity. This plot, seen in Figure 4, gave a linear relationship between gyroscope voltage output and Boe-Bot angular velocity. It can easily be seen that this linear relationship is fairly accurate, with an $R^2$ value of .9889. This information will be needed in using the voltage reading from the gyroscope to predict the actual angular velocity of the Boe-Bot.

**Communication: Serial**

Before going on too much further, it is important to document the evolution of communication methods we employed in order to control the Boe-Bot. First, we communicated with the Boe-Bot using the serial cable that came with it. This was mainly used in the beginning of the process in order to load BASIC stamp programming onto the board of education. This form of communication was used in the centering of the servomotors and the gyroscope calibration.
Communication: Wireless

One of the main challenges of this project was to communicate wirelessly with the Boe-Bot. Today’s society has truly begun to embrace and depend on wireless technology and it is important to understand its format, advantages, as well as shortcomings. Since our group was very unfamiliar with how wireless technology worked, we had to research Bluetooth software and hardware. Bluetooth is the wireless format we are using with our Boe-Bot.

After familiarizing ourselves with the software, our first task was to establish a connection with the wireless communication card on the Boe-Bot, the eb500. We were able to do this fairly easily using the USB Bluetooth device and the software that accompanied it. Once a connection was established, we found it quite easy to transmit text wirelessly in “data mode.” Practically speaking, this accomplishment served no purpose other than confirming that a connection had been made and information could in fact be transmitted. The next steps in establishing useful communication with the Boe-Bot will be discussed in the “Programming” section of this document.

Programming:

Programming was an extremely big part of getting the Boe-Bot to perform the desired tasks involved in the project. The initial programming was done in BASIC Stamp. We discussed these programs earlier, as we used BASIC Stamp in order to center the servomotors. Next, we tackled the problem of actually getting the Boe-Bot to respond to our wireless commands. Using the BASIC stamp program in Appendix B, we
were able to get the Boe-Bot to respond to commands we would input in HyperTerminal in real-time. Using the input commands 0, 1, 2, 3 and 4 we were able to move the Boe-Bot clockwise, counterclockwise, forward or simply tell it to stop.

The next challenge was to configure Labview in such a way that it could send the correct action inputs to the Boe-Bot wirelessly. The end goal of this would be to input a specific path into a Labview “front panel” interface, click “run,” and the Boe-Bot would respond by traveling the specified path. Getting Labview to talk to the Boe-Bot wirelessly proved to be an extremely difficult task. As can be seen in the program in Appendix C, we employed a Bluetooth element in Labview to first establish a connection with the correct Bluetooth device. Next, we expanded the program to that seen in Appendix D. This program writes a given user input to the Boe-Bot for a specific time. The “for loop” is used to control how long this command is sent in real-time to the Boe-Bot. For example, the for loop might run for 5 seconds, which will translate into the Boe-Bot going forward for 5 seconds. We originally used “while loops,” but found that the Boe-Bot would only respond to the first loop command. The successive loops would run, but the commands would not reach the Boe-Bots servomotors to produce results.

The next step involved calibrating the Boe-Bot so that the user could input distances and directions, and the Labview program would convert that to accurate speeds and command durations to move the Boe-Bot correctly. This involved iterative debugging in order to create the correct conversion formulas within the Labview program. We would input a command for the Boe-Bot to turn counterclockwise for 5 seconds and recorded the rotational distance it moved. We did this for several trials, as well as several trials at 10 seconds. This allowed us to develop a formula to equate a
given directional user input (in degrees) into a correct length of time the Boe-Bot must rotate to achieve the desired turn. We ran similar trials to move the Boe-Bot clockwise, forward and backward. This calibration can be seen in figures 5 and 6. Thus, when we finished, the user could input the 4 distances and turns required for the robot to travel the desired path. The final Labview program can be seen in Appendix E. Again, this program is used in conjunction with the BASIC Stamp program of Appendix B, which had been loaded onto the Boe-Bot previously.

**Setbacks and Considerations:**

In any project, comes the inevitable number of problems or setbacks that prevent smooth progress toward the desired goal. We would like to detail some of those problems here for consideration, as well as future concern. First, early on in the process our board of education completely malfunctioned and had to be replaced. The eb500 also malfunctioned and needed to be replaced. This caused a tremendous delay because we were without the capability of testing our wireless communication for 4 or 5 days. We
later learned that a “stop” program must be loaded onto the Boe from BASIC Stamp before the eb500 card is plugged in. This program turns off the output from all pins before the card is inserted, thus eliminating any unwanted signals that could fry the delicate card components.

Next, in working to connect the gyroscope to the Boe-Bot we realized it was necessary to use an analog to digital converter (ADC) in order to convert the analog output of the gyroscope to a digital value. We ordered this chip on Monday November 28, with 2-day shipping, but were unable to receive it in time due to shipping delays. It is for this reason that we were forced to leave the gyroscope out of the project. To compensate we made sure to create a Labview program that can direct the Boe-Bot through the entirety of the path without the closed-loop feedback adjustments that the gyroscope would provide for.

**Conclusion:**

In the end we are extremely satisfied with the status of the Boe-Bot. We were able to get the Bluetooth wireless communication fully integrated. In addition, we used BASIC Stamp and Labview to communicate a pre-determined path to the Boe-Bot. Some errors we encountered were inaccurate movement from the Boe-Bot once the program maneuvers were communicated. Our only theory for this problem was due to the delay in communication from the wireless application. The first turn would be accurate, however the successive turns were usually off by a few degrees. We concluded that the second angular commands were overlapping with the first command, and thus
producing an inaccurate turn. Through a little debugging, we were able to correct this problem.

The next steps we would like to take would be to obtain an ADC chip and use that to fully integrate the gyroscope into the robot. This would eliminate the maneuver accuracy problem, as the closed loop feedback would keep the robot on track throughout its path.

Finally, we were very proud and grateful to get the opportunity to utilize wireless communication in this project. The future of communication in our society is one that does not include wires. In learning about the applications of wireless technology and how to integrate that into a real-world project, we are all better prepared to utilize this vital knowledge in future projects outside of Northwestern.
Appendix A:

Gyroscope Calibration Program (BASIC Stamp):

' {$STAMP BS2}
' {$PBASIC 2.0}
speed CON 760

FOR counter = 1 TO 500

    PULSOUT 12,speed
    PULSOUT 13,speed
    PAUSE 20

NEXT
END

Gyroscope Calibration: Labview Data Collection (see attached for full program)
Appendix B:

HyperTerminal Program (BASIC Stamp)

'{$STAMP BS2}
LOW 12
LOW 13

LMotor CON 13
RMotor CON 12

LFwdFast CON 850
LRevFast CON 650
RFwdFast CON 695
RRevFast CON 850

CmdData VAR Byte

Connect:
PAUSE 1000
SEROUT 1,84,["con 00:13:46:4d:6b:30",CR]
SERIN 0,84,[WAIT("ACK",CR)]

WaitForConnection:
' IF IN5 = 0 THEN WaitForConnection

Main:
SERIN 0,84,[DEC1 CmdData]
DEBUG DEC1 CmdData
BRANCH CmdData,[Hold, Turn_Right, Turn_Left, Move_Fwd]
GOTO Main

Move_Fwd:
PULSOUT LMotor,LFwdFast
PULSOUT RMotor,RFwdFast
'SEROUT 1,84,["5"]
GOTO Main

Turn_Right:
PULSOUT LMotor,LFwdFast
PULSOUT RMotor,RRevFast
GOTO Main

Turn_Left:
PULSOUT LMotor,LRevFast
PULSOUT RMotor,RFwdFast
GOTO Main

Hold:
GOTO Main
Appendix C:

Labview Program to establish Bluetooth Connection, using while loops. (see attached for full program)
Appendix D:

Labview Program to establish Bluetooth Connection, preliminary program using for loops (see attached for full program)
Appendix E:

Final Labview Program: (See attached for full program)