Real Time Boe-Bot Control

ME 224 Final Project
Fall 2005
Professor: Horacio Espinosa
TA: Ravi Agrawal

Team:
Patrick Luckow
Joni Stegeman
Ingrid Lin
Giovanni Wuisan
Brent Willson
Abstract
We would like to be able to control our robot through real time wireless communication with a user-friendly interface. We also want the turning angle of our robot to be displayed, so if necessary its path can be corrected. We plan to do this using Bluetooth wireless technology, basic stamp II programming, and LabVIEW version 7.1 or higher. LabVIEW 7.1 is necessary because it contains sub VI’s compatible with Bluetooth technology. We expect problems establishing a connection between the wireless card on the robot and the computer. We also expect trouble integrating basic stamp II and LabVIEW. We hope to resolve these issues by doing research, looking for available solutions and finally trial and error.

Introduction

Objectives
The primary goal of this project is to design and implement an experiment to realize real time wireless control and feedback analysis of a commercially available robot.

Methodology
We had to design an experiment that would allow us to gain the necessary insight to complete our primary goal within 5 weeks. This experiment required several parts to be integrated effectively to yield real-time control of our robot. We first had to assemble our Boe-bot, center the servomotors, and understand its motion. From there, we had to familiarize ourselves with a MEMS gyroscope sensor that would provide feedback on the position of the robot. Then we had to develop a control strategy for our Boe-bot so we could navigate along any path. Finally, we had to find a way to achieve this Boe-bot control wirelessly in real-time. This would require a flexible user interface that could accept input commands and could then send those commands to the robot to gain control as well as output relevant data to the user interface to facilitate accurate control. From there, we would achieve real-time control through constant feedback, altering our controls as necessary.

Enhance Knowledge
One secondary goal throughout this project has been to enhance our knowledge of the various components necessary for the control of the Boe-bot. The experiences in this project enhanced our knowledge of LabVIEW, data acquisition, feedback control, Basic Stamp 2, and MEMS sensing. We also gained an understanding of wireless communication using Bluetooth technology. The knowledge gained during this project will prove invaluable in future experiences given that we started this project with little background in these areas.
Final Outcome

The desired outcome of this project is the real-time control of our robot through wireless communication. More specifically, the final result will consist of a LabVIEW user-interface program which can be used to control the robot along specified routes. This program should also receive data from the robot wirelessly and display the total angle the robot has turned. The real-time control comes into play when we are forced to alter routes at various points along the course. Feedback analysis is performed through getting the angular position of the robot through LabVIEW and comparing that to the actual angular position. The final deliverables consist of a final report and final presentation including an on-site demonstration of our robot control.

Materials

We have several materials and resources available to us to complete this project:

- LabVIEW and data acquisition card
- Gyroscope sensor (ADXRS150EB) from Analog Devices
- Boe-Bot Robot Kit #28132
  - Boe-Bot Robot
- Two servo motors
- Board of Education
- Wireless Bluetooth DBT-120
Testing Results

Actuation

Boe-Bot Assembly

We received the Boe-bot fully assembled from a previous 224 project team. However, we had to gain an understanding of the components and their interactions. We also had to understand how to power the robot and send that power to the necessary components. Finally, we had to understand the various modes of the robot; there are three modes on the Boe-bot, off (0), on and receive data (1), and run programs involving motion (2). The Boe-bot is made up of an aluminum chassis, two servo motors, and a printed circuit board known as the Board of Education (BOE). The breadboard allows smooth integration of additional electronic components depending on the desired application. It is required in order to use the gyroscope and ADC. Due to lack of space on the provided breadboard, our group attained a secondary breadboard to attach the ADC. This was mechanically attached to the Boe-bot chassis below the Board of Education.
Center Servos

The Boe-bot achieves motion through two servomotors, which are attached to the base of the robot. These servomotors rotate two wheels that propel the robot in the desired direction based on the control program. In order to ensure the proper direction of travel for the Boe-bot we had to first center these servomotors to make sure that they would rotate at the same speed. If left un-centered, the robot would not follow the proper path during the control phase of the project.

Centering the servos was a bit challenging at first because the adjustment involves a potentiometer, which lies within the core of the servo casing.

Since the Boe-bot was fully assembled, we had to use our engineering prowess to adjust the potentiometer without removing the servo motors from the robot itself. In order to know at what point the servos were centered we had to send a program to the Boe-bot which contained a center signal. The center signal instructs the servo motor to remain stationary once centered; this point was easily identified because any motion of the servo resulted in a whining noise and visible motion in the wheels connected to the servos. The basic stamp program with the center signal follows:

```基本
; ($STAMP BS2)
; {$PBASIC 2.5}
DEBUG "Program Running"
DO
  PULSOUT 13, 750
  PAUSE 20
LOOP
```
This program sends a center signal to one of the servomotors. This center signal is expressed physically below:

![Diagram of center signal](image)

The basic stamp “PULSOUT” command has two arguments, the duration argument and the appropriate pin argument. The duration argument, which is 750 in this case, is multiplied by 0.000002 seconds in order to calculate the signal duration. In this case the signal is 1.5ms in duration. The pin argument, 12 or 13 in this application, simply refers to the I/O pin P12 and P13 respectively. The “PAUSE 20” argument pauses the pulse for 20ms. We ran the program for each servo separately and adjusted the potentiometer until the motion and noise stopped.

After completing the servo-centering step of this project we wrote a program in basic stamp to verify that our servos were centered properly. We verified this by two methods. First we set each motor to 750 and ran the program. This should keep the robot motionless and it did. Next we wrote a program that should make the robot follow a straight path. In order to do this we had to further understand the “PULSOUT” command. Because our center signal calibration used a duration argument of 750, that value became our “center” point; arguments larger than 750 result in clockwise rotation while arguments smaller than 750 result in counterclockwise rotation. Since the servos are mounted on either side of the chassis, one must rotate CW while the other must rotate CCW to yield straight-line movement. Our straight path program verified that our servos were centered properly.

However, during later experiment, we realized that since the two servos are not perfectly identical, to make the Boe-bot run along a straight path, pulses of 815 and 700 were required, rather than the expected 800 and 700.
**Sensing**

**Gyroscope Background**

Two primary sensing devices used in research and industry include accelerometers, and gyroscopes. Accelerometers measure linear motion and gravity at the same time. These results cannot be isolated or decoupled without the introduction of a gyroscope. With an accelerometer, the effects of gravity confound the signal magnitude, and an additional difficulty associated with the device is being required to integrate twice in order to obtain displacement information.

The gyroscope is the best approach to sensing because it measures rotational motion but is not influenced by gravity, as is an accelerometer. Unlike accelerometers, both the frequency and magnitude information is accurate down to a DC level or zero frequency. Only a single integration is necessary to obtain angular displacement information. Gyroscopes offer a much better sensing alternative to accelerometers.

The schematic for our particular gyroscope sensor (ADXRS150EB) follows, along with a parts layout of the particular components.

*Figure 1. ADXRS150EB Schematic*

*Figure 2. ADXRS150EB Parts Layout*
Gyroscope Calibration

Before being able to use our gyroscope for sensing applications we first had to perform a self-test to ensure it was functioning properly. After verifying this, our next step was calibration. Calibration is necessary in order to associate an angular velocity with each voltage readout from the gyroscope. With this information we could then determine how far our robot turns based on the voltage readouts. The total angular rotation is important for the final objective of the project, which is achieving real-time control, because we can instruct the robot to alter its path based upon feedback from the gyroscope.

Our gyroscope calibration involved two iterations. The first was to understand what method we would use to calibrate our gyroscope and generate some sample data points. The second iteration was to generate a more substantial data set. The goal was to generate enough data to be able to evaluate any sort of error in our calibration technique. With this error we would then have an understanding of how accurately we could control the robot.

The strategy involved in our gyroscope calibration was first to write a basic stamp 2 control program that would instruct the boe-bot to rotate 360 degrees without moving laterally from its position. We then developed a LabVIEW program to record the gyroscope output during the run. We repeated these steps for several angular accelerations. The LabVIEW program used to record our data is attached in Appendix 2.

The first attempt at gyroscope calibration involved recording the calibration data at two angular velocities, one positive and one negative. The output data in terms of Voltage vs. Time follows for the positive angular velocity is below. The dip in the signal at 9000 ms is simply the end of the BS2 program before the program runs a second time.
We decided that after completing this first iteration of gyroscope calibration that it would be desirable to understand the magnitude of error in our testing methodology. As mentioned earlier, we thus decided to carry out further and more extensive gyroscope calibration. The first plot below shows our calibration data points fitted with a linear approximation. Looking at the plot it is clear that our reference voltage, the voltage associated with zero angular velocity is just below 2.5 volts. Using an oscilloscope, we determined that the zero-rotation voltage output of the gyroscope was 2.36V. Low voltages from 0 to 2.5 volts are associated with negative angular velocities and higher voltages from 2.5 to 4.5 volts are associated with positive angular velocities. This information will prove invaluable when we need to understand the position of our robot during our final run.

The gyroscope calibration plot below is similar to the first but shows the relative error inherent in each calculation. It is clear, given the minimal distance between the error points and the original point and the $R^2$ correlation value, that error in the calibration measurements was minimal.
Analog to Digital Converter (ADC)

After calibrating our gyroscope sensor to our satisfaction, the next step was to implement an analog to digital converter (ADC) that would allow us to utilize the full range of gyroscope in conjunction with the Boe-bot. The ADC will split the signal into 8 channels, allowing us to send and receive a number from 0-255 ($2^8$) instead of simply 0 or 1 which is what the pins on the Boe-bot read. An analog to digital converter uses the comparator principle to determine whether or not to turn on a particular bit of binary number output. The conversion from analog to digital involves comparing the analog input to a particular standard that we can set. In this manner we will be able to use our ADC for feedback control. Our controls will respond to the ADC input according to the calibration factor we previously calibrated. Once the comparison is made, appropriate changes in navigation commands can then be made.

On a more detailed level, the particular ADC we are using is an 8-bit converter that uses the gyroscope output as its input and sends values to 8 difference pins designated 0 to 7. The power for converter function comes directly from the boe-bot, which can supply the necessary 5 volts required which is convenient because the range of the gyroscope is about 0-5 volts as well. A diagram of the ADC used in this project lies below. Pins D0 to D7 are the eight pins associated with the 8-bit analog to digital converter.

Our primary finding relating to the operational function of the ADC is that it has two modes that we have to activate. For the chip to read the data, pin 8 (RD) has to be on LO (ground), then to instruct the chip to send the binary values to the output pins, pin 8 has to be switched to HI (5 Volts).

![ADC Diagram]

DB, DW, OR N PACKAGE (TOP VIEW)

ANLG IN (LSB) D0 1 20 VCC
D1 2 19 NC
D2 3 18 OFLW
D3 4 17 D7 (MSB)
WR/RDY 5 16 D6
MODE 6 15 D5
RD 7 14 D4
INT 8 13 CS
GND 9 12 REF+
10 11 REF−

FN PACKAGE (TOP VIEW)
**Control**

**LabVIEW**

Before being able to efficiently control our robot, we were required to make a few calculations to understand how long the pulse commands should be in order for the robot to turn a specified number of degrees or move forward a particular distance. In order to make these calculations we modified a labview program developed by a classmate. We used our gyroscope calibration data to understand the angular or linear distance traversed for a particular set of commands over time. A portion of the program used is attached in Appendix 3.

**Basic Stamp II**

Our first step in the control of our robot was to write a basic stamp II program designed to command our robot to travel any path. After successfully completing this step and developing a confidence in our basic stamp programming abilities, our next step toward the control of our robot was to write a program that would control our robot wirelessly. The program written for this wireless control follows. It is a modification of the original “monkeydo” program from the user manual of the wireless module, page 58. This program uses the Windows program “HyperTerminal” as an interface to send basic stamp commands to the boe-bot, which then responds accordingly. On a more specific level, 1 = turn right, 2 = turn left, 3 = move forward. The initial program follows:
'{$STAMP BS2}$
LOW 12
LOW 13

LMotor CON 12
RMotor CON 13

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

CmdData VAR Byte
DEBUG "connecting",CR
Connect:
PAUSE 1000
SEROUT 1,84,[:\"con 00:0C:84:00:0C:A6\",CR]
'SERIN 0,84,[:\"WAIT\"\"ACK\",CR]]
DEBUG "waiting", CR
'\'WaitForConnection:
' IF IN14 = 0 THEN WaitForConnection
DEBUG "connected, attempt commands",CR

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

Turn_right:
PULSOUT LMotor,LFwdFast
PULSOUT RMotor,RFwdFast
SEROUT 1,84,[:\"1\"]
GOTO Main

Turn_left:
PULSOUT LMotor,LFwdFast
PULSOUT RMotor,RRevFast
SEROUT 1,84,[:\"2\"]
GOTO Main

move_forward:
PULSOUT LMotor,LRevFast
PULSOUT RMotor,RFwdFast
SEROUT 1,84,[:\"3\"]
GOTO Main

Hold:
GOTO Main
Feedback and Basic Stamp II

The following two subroutines were added to the Basic Stamp program to accommodate gyroscope feedback:

Calc:

```plaintext
LOW 8
value = (1*IN0) + (2*IN1) + (4*IN2) + (8*IN3) 
+ (16*IN4) + (32*IN5) + (64*IN6) + (128*IN7) 
HIGH 8
V = (value*420/256+50)
theta = (145*V - 31700)/100
dsum = theta*ts/100
sum = sum + dsum
angle = sum/14
RETURN
```

Zerovar:

```plaintext
value = 0
V = 0
dsum = 0
theta = 0
sum = 0
RETURN
```

Calc converts the 8 binary signals we output from the ADC to one signal. This signal is then converted to a voltage. The equation for theta incorporates our gyroscope calibration data. The data is then integrated using a basic algorithm to output a value for the total angular rotation.

Zerovar simply resets the calculation. It is run whenever the robot is told to go forward or backwards.

The final Basic Stamp II program can be seen in Appendix 5.

LabVIEW and Basic Stamp 2

One necessary step in the process for the completion of this project is to understand how to integrate LabVIEW, the program used for our user-interface, and Basic Stamp 2, the program used to control the boe-bot. We made two attempts at achieving communication between these two programs and achieved it successfully the second time around.

Our first attempt was using a set of LabVIEW files and macros developed by a professor from Michigan State University. Screen captures of his LabVIEW files can be seen in Appendix 4.
Wireless Communication

Bluetooth

In order to control our Boe-bot wirelessly we had to first gain an understanding of Bluetooth technology. Bluetooth allows communication between electronic devices using short-range radio. There are two primary modes used with our particular Bluetooth components. Command mode allows the eb500 to accept commands while Data mode will not accept new commands. The key for this project is to maintain command mode so that we can continually send commands to our boe-bot in real-time to maintain control throughout our final demonstration.

As mentioned earlier, we were able to successful modify the “monkeydo” program to yield simple wireless control of the boe-bot through the “HyperTerminal” interface. The program used to find the address of the Boe-bot to ensure a connection with the computer can be found in Appendix 1. From here we must be able to successfully integrate this program into LabVIEW. After integrating our monkeydo program with LabVIEW we then gained a set of controls integrating, LabVIEW, HyperTerminal, and Basic Stamp II wirelessly. The screen captures for our final LabVIEW interface can be found in Appendix 6. Both the front panel and block diagram are shown.

Conclusion

In the end we were able to achieve substantially more than we originally thought possible. We struggled with wireless communication but in the end we were able to successfully achieve wireless integration. We were able to use the eb500 to communicate between the PC and the Boe-bot. We were also able to achieve real-time control of our robot. Using HyperTerminal, we could control our robot as well as display the angle it was turning in real-time. We were also able to control our robot using LabVIEW output the proper angle it turned after each command. One future step for this project would be to investigate ways to shorten the time delay between pulse commands to allow for smooth motion.
References

http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/adc.html


www.parallax.com

http://www.analog.com/

http://clifton.mech.northwestern.edu/~me224/

The documentation that came with the Boe-bot and eb500 module
APPENDIX 1

The following basic stamp II program was run to ensure our eb500 was making a connection with the computer.

GetAddress.bs2 gets the address of the eb500 module you are working with

'($Stamp BS2)
szData ViR Byte(20)
'Wait for the eb500 radio to be ready
PAUSE 1000
'Get the eb500 Bluetooth Address
SEROUT 1,84,["get address",CR]
SERIN 0,84,[WAIT("ACK",CR)]
'Read the local address from the get command
SERIN 0,84,[STR szData\17]
SERIN 0,84,[WAIT(CR,"\"\")]
.szData(17) = 0
DEBUG "Local eb500 address: ",STR szData\17,CR
APPENDIX 2

Following are the block diagrams associated with our gyroscope calibration data recording LabVIEW program.
APPENDIX 3

The following program was used to calculate pulse durations for particular angles and distances traveled by the Boe-bot.

<table>
<thead>
<tr>
<th>Distance 1</th>
<th>Angle 1</th>
<th>Ang Pin 12 Pulse 1</th>
<th>Ang Pin 13 Pulse 1</th>
<th>Ang Loop pulse duration 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distance 2</th>
<th>Angle 2</th>
<th>Ang Pin 12 Pulse 2</th>
<th>Ang Pin 13 Pulse 2</th>
<th>Ang Loop pulse duration 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distance 3</th>
<th>Angle 3</th>
<th>Ang Pin 12 Pulse 3</th>
<th>Ang Pin 13 Pulse 3</th>
<th>Ang Loop pulse duration 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distance 4</th>
<th>Angle 4</th>
<th>Ang Pin 12 Pulse 4</th>
<th>Ang Pin 13 Pulse 4</th>
<th>Ang Loop pulse duration 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distance 5</th>
<th>Angle 5</th>
<th>Ang Pin 12 Pulse 5</th>
<th>Ang Pin 13 Pulse 5</th>
<th>Ang Loop pulse duration 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
APPENDIX 4

The following program was developed by a professor at Michigan State University that we unsuccessfully attempted to implement into this project.
APPENDIX 5

This is a basic stamp II program modification of the program shown within the report. It allows the user to specify a right or left turn as well as forward or backward motion. It will also calculate the angle the robot has turned. This information is based on the calibration data of our gyroscope as well as a properly functioning ADC. It can be used with HyperTerminal or LabVIEW.

Controlprogram.bs2
Modified Monkey see program that serves to control our robot

```
' |$STAMP BS2
LOW 12
LOW 13

LMotor CON 12
RMotor CON 13

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

value VAR Word
theta VAR Word
sum VAR Word
V VAR Word
t VAR Word
counter VAR Word
dsum VAR Word
pulsecount VAR Word
angleturn VAR Word
angle VAR Word
t s = 10

CmdData VAR Byte
DEBUG "connecting", CR
Connect:
PAUSE 1000
SEROUT 1,84,"["con CC:OC:84:00:0A:87", CR]
'SERIN 0,84,[WAIT("ACK",CR)]
DEBUG "waiting", CR
' IF IN1A = 0 THEN WaitForConnection
DEBUG "connected, attempt commands", CR

Main:
SERIN 0,84,[DEC1 CmdData]
DEBUG DEC1 CmdData
BRANCH CmdData,[Hold, Turn_Right, Turn_Left, Move_backward, Move_forward]
GOTO Main
```
move_backward:
    FULSOUT LMotor, Lwdfast
    FULSOUT RMotor, Rwdfast
    GOSUE Zerovar
    GOSUE Calc
    SEXECUT 1, 0.94, [DEC angle]
    GOTO Main

Turn_right:
    FULSOUT LMotor, Lwdfast
    FULSOUT RMotor, RrevFast
    GOSUE Calc
    SEXECUT 1, 0.94, [DEC angle]
    GOTO Main

turn_left:
    FULSOUT LMotor, LrevFast
    FULSOUT RMotor, Rwdfast
    GOSUE Calc
    SEXECUT 1, 0.94, [DEC angle]
    GOTO Main

move_forward:
    FULSOUT LMotor, LrevFast
    FULSOUT RMotor, RrevFast
    GOSUE Zerovar
    GOSUE Calc
    SEXECUT 1, 0.94, [DEC angle]
    GOTO Main

Hold:
    GOTO Main

Calc:
    LOW 8
    value = (1*INC) + (2*IN1) + (3*IN2) + (4*IN3) + (5*IN4) + (6*IN5) + (7*IN6) + (8*IN7)
    HIGH 8
    V = (value*420/256+50)
    theta = (145*V - 31700)/100
    dsum = theta*ts/100
    sum = sum + dsum
    angle = sum/14
    RETURN

Zerovar:
    value = 0
    V = 0
    dsum = 0
    theta = 0
    sum = 0
    RETURN
APPENDIX 6

This is the LabVIEW program that we modified to get a connection between the Boe-bot and PC as well as to make LabVIEW compatible with the basic stamp II program written above. It allows LabVIEW to control the robot and display the angle it has turned.
Open a connection to the selected service and begin reading and writing data as necessary for the specified service. When finished, close the connection. It will graph the data from the gyroscope in real time, allow the user to control the robot, and will display the angle the robot has turned in real time on the front panel.