Path Following Robot With Gyroscopic Sensing

Final Project – ME 224
Professor Espinosa
Northwestern University
June 10, 2005

James Lovsin
Erica Morales
Dan Sheehan
Josh Widzer
Table of Contents

Abstract 4
I. Introduction 5
II. Equipment 6
III. Activity 1: Gyroscope Familiarity 6
    Self-Test 7
    Calibration 7
IV. Activity 2: Robot Familiarity 12
    Servo Centering 12
    Assembly 12
    Basic Paths 12
V. Activity 3: Path Programming 13
    Basic Stamp Program 13
    LabVIEW Program 14
VI. Results 15
VII. Future Work 16
VIII. Conclusion 17
IX. References 18

Appendices

Appendix A: LabVIEW Calibration Program 19
Appendix B: Calibration Data 21
Appendix C: Basic Stamp Path Program 22
Appendix D: LabVIEW Path Program 24
List of Figures

1. Specified Route 5
2. Theta versus Integral of Voltage 9
3. Angular Rate versus Voltage 11
4. LabVIEW Program Structure 14

List of Tables

1. First Attempt Calibration Data 8
2. Second Attempt Calibration Data 10
3. X and Y Deviation from Starting Point 15
4. Angle Deviation from Starting Point 15
Abstract

As the final project for Mechanical Engineering 224: Experimental Engineering, our group designed a robot that would follow a predetermined path. To ensure the path was followed as close as possible, a gyroscope sensor was used to ensure the robot turns with the correct turn angles and to ensure the robot goes straight on straight-aways. The gyroscope output was read using a LabVIEW program. However, the actual control of the robot was done using a Basic Stamp chip. The LabVIEW was interfaced with the Basic Stamp chip using the outputs on the DAQ data acquisition board and the inputs on the Basic Stamp chip. Depending on the reading, the LabVIEW program would send signals to the Basic Stamp chip telling it how to move. The finished product was an automated 2-wheel cart that follows a preprogrammed path with gyroscopic feedback control on turns and straight-aways. Overall, the project achieved its objectives to realize the control of a commercially available robot and enhance our knowledge of LabVIEW, the DAQ data acquisition board, feedback control, and MEMS (microelectromechanical systems) sensing.
I. Introduction

The focus of the Mechanical Engineering 224: Experimental Engineering class has been on using LabVIEW programming, the DAQ data acquisition board, and simple circuits to acquire experimental data and implement controls. This was done through a series of experiments ranging from temperature controllers to MEMS (microelectromechanical systems) sensor performance assessments. This project is the culmination of these experiences. The objectives of this project are:

- To design and implement an experiment to realize the control of a commercially available robot
- To enhance our knowledge of LabVIEW, data acquisition, feedback control, and MEMS sensing

This is to be achieved by creating a robot (2-wheel cart) that travels along a specified route. Figure 1 shows the specified route. The robot will utilize feedback from a MEMS gyroscopic sensor to ensure it is accurately following the path.

To create the robot mentioned above, several activities were completed. The first activity involves becoming familiar with the MEMS gyroscope sensor.

![Figure 1: Specified Route](image)

Note: The distances given are relative values. For each straight line segment, the robot moves forward with feedback for the specified number of seconds to achieve the path.
This consists of both a self-test to ensure the sensor is working and calibration so the output data can be converted to a useful form. The second activity involves becoming familiar with the robot that is to be used. This consists of building the robot, centering the servos, and executing basic movements. The third activity entails the actual path programming. A basic stamp program was created to do basic movements and a LabVIEW program was created to control the movements and implement the gyroscopic sensing feedback.

II. Equipment

In this project, three basic pieces of equipment were required:

- A computer with LabVIEW and a data acquisition card
- A gyroscope sensor (ADXRS150EB) from Analog Devices
- A Boe-Bot Robot Kit (www.parallax.com, #28132)

The computer with LabVIEW and a data acquisition were available in the ITEC building.

The gyroscope sensor is used to detect angular velocity. The gyroscope sensor used in this experiment is the ADXRS150EB from Analog Devices. This is a MEMS device fabricated using micromachining techniques. The advantages of this sensor are that it is small and can be manufactured at a low cost. As oppose to classical gyroscopes which utilize a spinning mass to detect the spin, MEMS gyroscope sensors rely on the principle of the Coriolis Effect. The force created by the Coriolis Effect is applied to a mass which is already being electrostatically vibrated and causes a secondary vibration. This secondary vibration is measured using capacitive pickoff structures. This change in capacitance results in a change in the output voltage of the chip. The output signal goes through signal conditioning to produce a usable signal. In this way, the output voltage is related to the angular velocity.

The Boe-Bot robot kit is from Parallax, Inc. The Boe-Bot robot is a two-wheel cart driven by two servo motors. The brain of the Boe-Bot is a Basic Stamp chip. Included on the basic stamp assembly is a small bread board for incorporating various sensors.

III. Activity 1: Gyroscope Familiarity

Before gyroscopic feedback control can be incorporated into the robot, the MEMS gyroscope sensor must be understood. There are two steps to accomplishing this. First, a self-test is needed to ensure the chip is functioning correctly. Second, calibration is needed so the relationship between output voltage and angular displacement can be understood.
Self-Test

A self-test actuates the each of the structures and electronics of the chip as if the chip were subjected to an angular rate. For given applied voltages, the chip should have appropriate RATEOUT responses. If the RATEOUT responses do not correspond to the specified self-test values, then the chip is either not wired properly or the chip is damaged. For detailed accounts of a self test procedure, the reader is referred to [1] and [4].

The self-test was conducted with an adjustable power supply, a multimeter, and an oscilloscope. When the chip was appropriately wired according to the self test procedure and the RATEOUT was wired to an oscilloscope, the RATEOUT was initially observed to be 2.5 V. Applying 4.6 V to pin 10 the RATEOUT was observed to drop to 3.2 V. Similarly, applying 4.6 V to pin 11 the RATEOUT was observed to drop to 1.81 V. Since these RATEOUT values correspond to the specified self-test values given by the chip’s data sheet, the chip was determined to be wired correctly and functioning properly.

Calibration

The chip is designed to measure a change in capacitance associated with a vibration of a proof mass of the chip. With signal conditioning circuits integrated onto the chip, the change in capacitance can be related to a change in voltage which is the RATEOUT response discussed in the self-test section. This chip relates a voltage to an angular rate. Although the chip is very sensitive, the chip must be calibrated so that the sensing information provided by the chip can be used effectively.

Since position control of the Boe-Bot is desired, a deterministic model must be developed to relate a change in voltage with a change in angle. LabVIEW was used to acquire voltage readings at consistent sample times and then send the data to Microsoft Excel for data analysis. The calibration program sampled the voltage signal every 100 ms and sent the voltage signal and a timestamp to an excel spreadsheet. The calibration program is presented in Appendix A. Two calibration techniques were attempted with varying results. Both calibration techniques relied on linear regression to produce a deterministic model.

In a first calibration attempt, a deterministic model that related a change in voltage to a change in angle was attempted to be derived from experimental data. The following model was presupposed:

\[ \Theta_i = B_0 + B_1 \sum V_i \Delta t \]  (1)

Where \( B_0 \) and \( B_1 \) are the intercept parameter and slope parameter, respectively. The two parameters are to be obtained by using an ordinary least squares linear regression of experimental data. Additionally, \( \Delta t \) is the sample rate.
With known angles, the Boe-Bot was rotated by hand for a given amount of time. Data was acquired in LabVIEW and then analyzed with Excel. Since the chip relates a change in voltage to a change in angular rate, in order to arrive at a change in angle the change in voltages need to be integrated. To this end, the integral of the change in voltage needs to be acquired to be related to a change in angle. A simple numerical integration approximation was used consisting of the sum of the voltage multiplied by the sample rate. Data was acquired for several trials of turns of 90 degrees and turns of 180 degrees. The data was plotted and regressed in Excel. Table 1 presents the data obtained. Figure 2 shows the regression plot.

<table>
<thead>
<tr>
<th>Angle (deg)</th>
<th>Integral dV (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>13158</td>
</tr>
<tr>
<td>180</td>
<td>6555</td>
</tr>
<tr>
<td>180</td>
<td>7024</td>
</tr>
<tr>
<td>90</td>
<td>12694</td>
</tr>
<tr>
<td>90</td>
<td>12774</td>
</tr>
<tr>
<td>90</td>
<td>5353</td>
</tr>
</tbody>
</table>

Table 1: First Attempt Calibration Data

Table 1 shows that there is significant scatter in the integral of dV.
From the correlation coefficient ($R^2$) of the regression, the design team concluded that this data set is not appropriate for the chip calibration. The experimental setup and model may have contributed to this poor correlation. The exact positioning and timing to complete a rotation of the Boe-Bot by hand were not extremely accurate. Additionally, the numerical approximation of the integral of the angular rates is considered to be effective only as a first approximation. More sophisticated numerical integration techniques such as the Trapezoidal rule or Simpson’s rule would most likely give more accurate results.

In a second calibration attempt, a deterministic model that related a change in voltage to a change in angular rate was attempted to be derived from experimental data. The following model was presupposed:

$$\frac{d}{dt}(\Theta_i) = B_0 + B_1 V_i$$

(2)

Where $B_0$ and $B_1$ are the intercept parameter and slope parameter, respectively. The two parameters are to be obtained by using an ordinary least squares linear regression of experimental data.
Formula two provides for a more accurate calibration because a deterministic model is being specified for what the sensor is intended to sense. Once an angular rate is known, further programming and calculations can be conducted to allow the Boe-Bot to rotate at an angular rate for a specified amount of time to arrive at a desired angle.

In this calibration attempt, Basic Stamp is used to rotate the Boe-Bot at a given angular rate. The angular rate was determined by timing one revolution of the Boe-Bot. For the determined angular rates of 2.06 rad/s, 3.05 rad/s, and 1.78 rad/s data was acquired with LabVIEW and then analyzed in Excel. Appendix B presents data from one of the trials conducted. Table 2 presents the data obtained. Figure 3 shows the regression plot.

<table>
<thead>
<tr>
<th>Angular Rate (rad/s)</th>
<th>Voltage Value (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.78</td>
<td>1.25376</td>
</tr>
<tr>
<td>1.78</td>
<td>1.244694</td>
</tr>
<tr>
<td>1.78</td>
<td>1.240739</td>
</tr>
<tr>
<td>3.05</td>
<td>4.589909</td>
</tr>
<tr>
<td>3.05</td>
<td>4.560954</td>
</tr>
<tr>
<td>3.05</td>
<td>4.580176</td>
</tr>
<tr>
<td>2.06</td>
<td>3.904574</td>
</tr>
<tr>
<td>2.06</td>
<td>3.892855</td>
</tr>
<tr>
<td>2.06</td>
<td>3.867415</td>
</tr>
</tbody>
</table>

Table 2: Second Attempt Calibration Data
Although the $R^2$ value for this regression is only 0.64, this calibration may be considered to be superior to the first calibration attempt. Developing a more robust way of determining the Boe-Bot’s angular rate and conducting more trials would most likely improve the regression results. The deterministic model obtained from this regression will be used as a starting point in path programming.
IV. Activity 2: Robot Familiarity

An important task before beginning path programming was to become familiar with the robot. The first task in this activity was to assemble the robot. The second task in this activity was to center the servos on the robot. Finally, the robot was to be tested by creating programs to make the robot do simple maneuvers.

Servo Centering

Before the robot was assembled, it was necessary to center the servo motors. Initially, when the stop signal is applied to the servo motors, the motors start turning because the servo motors are not pre-adjusted at the factory. Hence, it is necessary to adjust them so that the servo motors are stopped when the stop signal is sent to them. This is done using a screwdriver. Using the Basic Stamp to send a pulse of 1.5 ms to the motor, the screwdriver is inserted into the access hole and twisted to adjust the potentiometer such that the motor servos are stopped. For further information on how to center the servos the reader is referred to [3].

Assembly

Once the servos were centered we could go about assembling the Boe-Bot using the provided instruction manual. The Boe-Bot kit came ready to assemble except for the AA batteries, which were not included. Assembly began by mounting the topside hardware and the servos onto the chassis. After the battery pack was attached, the tail wheel and drive wheels were respectively pinned and screwed on. Completion was achieved after appending the board with BASIC Stamp 2 onto the chassis’ standoffs. Following assembly it was important to test the Boe-Bot to avoid mistakes with the robot’s behavior.

Basic Paths

With the Boe-Bot now assembled, it was necessary to determine if everything was functioning properly. To do this, several basic commands were programmed into the Basic Stamp and run. The first of these commands was to drive forward for three seconds. The second of these was to turn left for three seconds. Finally, the Boe-Bot was commanded to go turn right for three seconds. For further information about these commands the reader is referred to [3]. These commands were repeated until satisfactory results were obtained.
V. Activity 3: Path Programming

The final activity is programming the robot to follow the predetermined path using gyroscopic feedback control. One of the problems faced in this activity is how to control the robot. The challenge was somehow incorporating the feedback control. To overcome this challenge, our group decided to use both a Basic Stamp program and a LabVIEW program. The Basic Stamp program would control the motion of the robot, while the LabVIEW program would take readings from the gyroscope and control the movement of the robot based on these readings. To communicate between the Basic Stamp chip and LabVIEW, a ribbon cable was connected to the inputs of the Basic Stamp chip and the outputs on the DAQ data acquisition card.

Basic Stamp Program

The Basic Stamp program is fairly straightforward. The complete Basic Stamp program is presented in Appendix C. There are four variables. These variables are the inputs (0-3) of the Basic Stamp. Depending on which of these inputs are activated, the robot will behave in a certain way. Activation of these inputs is controlled by the LabVIEW program. Once these variables are initialized in the code, the program enters an infinite “do” loop. This loop consists of four if-statements. These if-statements check to see which inputs are activated. If input 0 is activated, then the program will enter a subroutine called STO. If input 1 is activated, the program will enter a subroutine called Forward. If input 2 is activated, then the program will enter a subroutine called Right. If input 3 is activated, then the program will enter a subroutine called Left.

STO tells the robot to stop by sending the servos the appropriate pulse train.

Forward tells the robot to go forward, driving the motors in the same direction at the same speed. It also incorporates feedback control. Depending on whether input 2 or input 3 is also activated, the program will then enter the subroutine Forright or Forleft, respectively. These subroutines correct the path to ensure a straight line. For example, if the robot begins to drift left on a straight path, input 2 will be activated and the left motor’s speed will be increased to put the robot back on the path. If the robot begins to drift right, input 3 will be activated and the right motor’s speed will be increased to put the robot back on the path.

Right tells the robot to turn to the right. This turn is accomplished by driving the servos at the same speed but in opposite directions; the left servo forward and the right servo backward. This essentially causes the robot to rotate about its center point in the clockwise direction.

Left tells the robot to turn to the left. This turn is also accomplished by driving the servos at the same speed but in opposite directions. However, in this case, the left servo is driven backwards and the right servo is driven forward.
LabVIEW Program

The LabVIEW program is the high level control software used in the path following of the Boe-Bot. Generally, the program samples the chip’s voltage reading every 10 ms and through the calibration formula relates it to an angle. With this information, the program makes a position determination and sends output voltages to the Basic Stamp inputs. As discussed in the previous section, the Basic Stamp program is responsible for the direct servo activation of the Boe-Bot whereas the LabVIEW program is responsible for controlling how and when the servos will be activated.

The LabVIEW program consists of nine sequence structures. In these sequence structures various case structures, loops, and formula nodes exist. The case structures and loop structures typically establish position control. For example, a certain loop will run until a desired angle is reached. The formula nodes typically convert voltage readings to angular readings. The specified route from Figure 1 has been divided into nine tasks. Each task corresponds to one of the nine case structures. Figure 4 shows the structure of the program.

Figure 4: LabVIEW Program Structure

In Figure 4, green numbers represent straight line sequence structures and blue numbers represent turn structures. The bold number beneath the turn structure specifies the angle that the Boe-Bot will turn. Step nine terminates the program by sending zero voltage to
all of the Basic Stamp inputs. Notice here that the program implements feed-back control for both angle turns and during straight line travel. For small perturbations on a straight line path, the program can return the Boe-Bot to the desired straight line path. The full program is presented in Appendix C.

VI. Results

With the LabVIEW virtual instrument an interface was created allowing a user to push a button to start the Boe-Bot. After communicating with the BasicStamp program, the robot performed the assigned path with successful results. To achieve satisfactory results time was devoted to calibration and debugging. We performed a repeatability test comprised of ten trials to see how accurate the Boe-Bot executed its task. We made sure to start the robot at the same starting point for each trial. The x and y distance from the starting and endpoint were measured to see how much deviation there was. LabVIEW would output the angle the robot made each time during the path, with the aide of a MEMS gyroscopic sensor. The gyroscope worked well in conjunction with the robot. In a situation in which the robot got caught in the ribbon cable the robot will still turned to the desired angle. The reliability test data is presented in Table 3 and Table 4. The X error and Y error data is plotted and presented in Figure 5. Notice that the results precise with respect to the actual values.

<table>
<thead>
<tr>
<th></th>
<th>X Error (mm)</th>
<th>Y Error (mm)</th>
<th>Theta Error (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>20.1</td>
<td>17.5</td>
<td>3.15</td>
</tr>
<tr>
<td>Standard Error</td>
<td>7.68022442</td>
<td>10.84486566</td>
<td>1.901023116</td>
</tr>
</tbody>
</table>

Table 3: Deviation from Starting Point

<table>
<thead>
<tr>
<th>Desired Angle (degrees)</th>
<th>90</th>
<th>143.1</th>
<th>-53.1</th>
<th>-180</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Value (degrees)</td>
<td>90.568</td>
<td>143.727</td>
<td>-53.702</td>
<td>-180.581</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0.422027</td>
<td>0.357026</td>
<td>0.281772</td>
<td>0.256361</td>
</tr>
</tbody>
</table>

Table 4: Angle Results
Although the robot worked well, there was some error in the robot’s movements. The errors can be a result of the program’s loop delay; the sample rate could be increased as opposed to the current 10 milliseconds. Human error while measuring the angles with a protractor and the distances with a tape measure can cause some inaccuracy. Another reason that can be attributed to the error is that Windows, which is used on the computer, is not a real-time operating system. This can be fixed by using Linux as opposed to Windows. We could also perform more trials enabling us to have a larger sample.

**VII. Future Work**

Although we are very satisfied with the performance of our robot, there are a few possibilities for improvement. First of all, our robot utilized a cable to communicate between the Basic Stamp chip and the LabVIEW program. This achieved the desired results but led to inconveniences, including the cable becoming twisted and the fact that our robot was “tethered” to a computer. A possibility for future work is to somehow get rid of this cable. This could possibly be accomplished by using the Basic Stamp for all controls and computing.
In conclusion, we have successfully accomplished the task to calibrate a gyroscope, assemble a Boe-Bot and program the robot to execute the assigned path. The gyroscope measured angular velocity, through which we found the robot’s angle. Both LabVIEW and BasicStamp were used in conjunction for this project. LabVIEW allowed the user to begin the robot and controlled the BasicStamp program, which dictated the robot’s direction. Our final results were precise and satisfactory. We are confident in our Boe-Bot’s performance.
References


[3] A. Lindsay, Robotics with the Boe-Bot Parallax Inc.

   http://www.analog.com/UploadedFiles/Data_Sheets/778386516ADXRS150_B.pdf

1. **Appendix A: LabVIEW Calibration Program**

Figure A.1 shows the front panel of Calibration.vi and Figure A.2 shows the block diagram of Calibration.vi. Note that Calibration.vi has a sequence structure with two frames.

![Front Panel Image](image-url)

**Figure A.1: Front Panel**

![Block Diagram Image](image-url)

**Figure A.2: Block Diagram**
Figure A.2: Block Diagram
Appendix B: Calibration Data

Table B.1 presents the first thirty raw data values for a second attempt calibration trial.

<table>
<thead>
<tr>
<th>D Value</th>
<th>Timestamp (ms)</th>
<th>Voltage (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1610</td>
<td>602108864</td>
<td>3.930664</td>
</tr>
<tr>
<td>1603</td>
<td>602108992</td>
<td>3.913574</td>
</tr>
<tr>
<td>1603</td>
<td>602109056</td>
<td>3.913574</td>
</tr>
<tr>
<td>1593</td>
<td>602109184</td>
<td>3.88916</td>
</tr>
<tr>
<td>1602</td>
<td>602109248</td>
<td>3.911133</td>
</tr>
<tr>
<td>1589</td>
<td>602109376</td>
<td>3.879395</td>
</tr>
<tr>
<td>1604</td>
<td>602109504</td>
<td>3.916016</td>
</tr>
<tr>
<td>1603</td>
<td>602109568</td>
<td>3.913574</td>
</tr>
<tr>
<td>1614</td>
<td>602109696</td>
<td>3.94043</td>
</tr>
<tr>
<td>1611</td>
<td>602109760</td>
<td>3.933105</td>
</tr>
<tr>
<td>1599</td>
<td>602109888</td>
<td>3.903809</td>
</tr>
<tr>
<td>1611</td>
<td>602109952</td>
<td>3.933105</td>
</tr>
<tr>
<td>1603</td>
<td>602110080</td>
<td>3.913574</td>
</tr>
<tr>
<td>1599</td>
<td>602110144</td>
<td>3.903809</td>
</tr>
<tr>
<td>1602</td>
<td>602110272</td>
<td>3.911133</td>
</tr>
<tr>
<td>1602</td>
<td>602110336</td>
<td>3.911133</td>
</tr>
<tr>
<td>1604</td>
<td>602110464</td>
<td>3.916016</td>
</tr>
<tr>
<td>1594</td>
<td>602110592</td>
<td>3.891602</td>
</tr>
<tr>
<td>1599</td>
<td>602110656</td>
<td>3.903809</td>
</tr>
<tr>
<td>1594</td>
<td>602110784</td>
<td>3.891602</td>
</tr>
<tr>
<td>1586</td>
<td>602110848</td>
<td>3.87207</td>
</tr>
<tr>
<td>1577</td>
<td>602110976</td>
<td>3.850098</td>
</tr>
<tr>
<td>1589</td>
<td>602111040</td>
<td>3.879395</td>
</tr>
<tr>
<td>1600</td>
<td>602111168</td>
<td>3.90625</td>
</tr>
<tr>
<td>1608</td>
<td>602111232</td>
<td>3.925781</td>
</tr>
<tr>
<td>1620</td>
<td>602111360</td>
<td>3.955078</td>
</tr>
<tr>
<td>1610</td>
<td>602111488</td>
<td>3.930664</td>
</tr>
<tr>
<td>1604</td>
<td>602111552</td>
<td>3.916016</td>
</tr>
<tr>
<td>1615</td>
<td>602111680</td>
<td>3.942871</td>
</tr>
<tr>
<td>1618</td>
<td>602111744</td>
<td>3.950195</td>
</tr>
</tbody>
</table>

Table B.1 Raw Calibration Data
Appendix C: Basic Stamp Path Program

The code for the basic stamp path program is given below:

' Robotics with the Boe-Bot - PathProgram.bs2
' Closed loop path control for the Boe-Bot.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

'-----------------[Variables]-----------------

INPUT 0
INPUT 1
INPUT 2
INPUT 3

DO
Main:
  IF ( IN0 = 1 ) THEN Sto
  IF ( IN1 = 1 ) THEN Forward
  IF ( IN2 = 1 ) THEN Right
  IF ( IN3 = 1 ) THEN Left
LOOP

END

Sto:
  PULSOUT 13, 750
  PULSOUT 12, 750
  PAUSE 18
  GOTO Main

Forward:
  IF ( IN2 = 1 ) THEN Forright
  IF ( IN3 = 1 ) THEN Forleft
  PULSOUT 13, 700
  PULSOUT 12, 800
  PAUSE 18
  GOTO Main

Forright:
  PULSOUT 13, 600
  PULSOUT 12, 800
PAUSE 18
GOTO Main

Forleft:
PULSOUT 13, 700
PULSOUT 12, 950
PAUSE 18
GOTO Main

Right:
PULSOUT 13, 775
PULSOUT 12, 775
PAUSE 18
GOTO Main

Left:
PULSOUT 13, 725
PULSOUT 12, 725
PAUSE 18
GOTO Main
Appendix D: LabVIEW Path Program

Figure D.1 shows the front panel of PathProgram2.vi and Figure D.2 shows the block diagram of PathProgram2.vi. Note that PathProgram2.vi has a sequence structure with nine frames.

![Figure D.1: Front Panel](image1)

![Figure D.2: Block Diagram](image2)
Figure D.2: Block Diagram
Notice: Alternatives of case structures are not shown in this appendix.