

Bluetooth Boe-Bot[®] Robot

Controlled by Microsoft Robotics Studio

(#28118)

The Bluetooth Boe-Bot Robot Kit for Microsoft Robotics Studio (MSRS) is a Parallax Boe-Bot Robot and an A7 Engineering eb500-SER Bluetooth module. The eb500 module makes it possible for the Boe-Bot robot's BASIC Stamp 2 microcontroller brain to communicate wirelessly with MSRS running on a nearby PC. The BASIC Stamp microcontroller runs a small PBASIC program that controls the Boe-Bot robot's servos and optionally monitors sensors while it communicates wirelessly with MSRS.

MSRS makes it possible to write robot sensor monitoring and motion control code (called services) on your PC with popular languages such as Microsoft Visual C# and Visual BASIC. This code relies on other services that communicate serially with the Boe-Bot robot via the Bluetooth connection to request sensor measurements and issue control commands. With this arrangement, MSRS makes puts an extensive service library and the PC's processing and data storage abilities at your disposal for robotics applications.



To learn more about MSRS and robot control, check out articles in the Spring 2007 issue of ROBOT Magazine and the January issue of Scientific American.

This guide will help you get started with the Boe-Bot Robot, eb500 Bluetooth module, and Microsoft Robotics Studio (MSRS) as you follow these steps:

- Download and install the software packages.
- Set up and test the Boe-Bot and eb500 hardware.
- Set up and test the Bluetooth serial connection between PC and the eb500 module.
- Load a PBASIC “driver” program into the Boe-Bot robot's BASIC Stamp microcontroller. This program, monitors the Boe-Bot robot's sensors, controls its motors, and communicates with MSRS.
- Try an example MSRS UI service that controls the Bluetooth Boe-Bot with a virtual joystick.
- Experiment with the Microsoft Robotics Studio service that communicates with and controls the Boe-Bot robot.
- Use Microsoft Robotics Tutorials to learn more about writing UIs and services. Examples include displaying sensor readings, coding for control based on sensor events, and writing a UI to control the Boe-Bot robot.



Software and Operating System

The examples presented in this guide used Microsoft Robotics Studio (1.0) and Microsoft Visual C# 2005 Express Edition on Windows XP Professional. Depending on which editions of Microsoft Robotics Studio, Visual C# and operating system you are using, some steps in this guide may be slightly different from what you will have to do to make the examples work.

Required Hardware and Software

To test the Bluetooth Boe-Bot with Microsoft Robotics Studio examples in this guide, you will need a fully functional (assembled and tested) Boe-Bot robot and an eb500 Bluetooth AppMod. The examples in this document use software which is available for free download, including the Parallax BASIC Stamp Editor, Microsoft Robotics Studio (1.0), and Microsoft Visual C# 2005 Express Edition.

Hardware and Software List

- (1) Bluetooth Boe-Bot Robot Kit for Microsoft Robotics Studio includes:
 - (1) Parallax Boe-Bot Robot Kit
 - (1) A7 Engineering eb500-SER Bluetooth AppMod
- (1) PC with Bluetooth wireless
- (1) BASIC Stamp Editor – *free download from www.parallax.com.*
- (1) Microsoft Visual C# – *the Express Edition is a free download from msdn.microsoft.com.*
- (1) Microsoft Robotics Studio (1.0) – *also a free download from msdn.microsoft.com.*



USB Bluetooth Adaptors provide an inexpensive and easy to obtain upgrade for PCs that don't have Bluetooth wireless built-in. Here are two adaptors the author has used with various machines and operating systems at the time of this writing:

- Targus USB Bluetooth 2.0 Adaptor with EDR – worked with Microsoft Windows XP and Vista.
- D-Link DBT-120 – worked with Microsoft Windows XP, but not with Windows Vista.

If you have an adaptor or operating system information you would like to see added to this list, please contact editor@parallax.com.

Boe-Bot Robot and Board of Education® Platform

If you've never worked with the Boe-Bot robot and BASIC Stamp® microcontroller before, the best place to start is with the *Robotics with the Boe-Bot* text that comes with the kit. As a minimum, work through the activities in the Getting Started with the BASIC Stamp and Boe-Bot list below. For extra information on the differential drive navigation and sensor monitoring techniques used with this document's examples, continue through the Background on the Servos and Sensors list.

Getting Started with the BASIC Stamp and Boe-Bot

Robotics with the Boe-Bot v2.2

- Chapter 1, Activity #1 – 4Getting started with the BASIC Stamp
- Chapter 2, Activity #3, 4, 6Connect, center and test the servos
- Chapter 3, Activity #1 – 3Assemble and test the Boe-Bot

Background on the Servos and Sensors

Robotics with the Boe-Bot v2.2

- Chapter 4Navigation program examples
- Chapter 5Whiskers (contact switch sensors)
- Chapter 7, 8Infrared for object and distance detection



Make sure to connect and "center" the servos before assembling the Boe-Bot!

For instructions on how to connect and center the servos, see Robotics with the Boe-Bot, Chapter 2, Activity #3 & #4. For assembly instructions, see Chapter 3, Activity #1.

eb500 Bluetooth Module

The eb500 module plugs into the X1 socket on the Board of Education as shown in Figure 1, and it provides the BASIC Stamp with a serial link to the PC via Bluetooth wireless. While it's not necessary to disconnect your Board of Education from the Boe-Bot, it is necessary to make sure the power is turned off when you plug in the eb500. It is also necessary to make sure that no electrical signals are being sent to P5 and P1, either by the BASIC Stamp or external circuits.



Electrical signals transmitted to P5 or P1 could damage the eb500!

You can ensure the eb500 doesn't receive electrical signals by following two simple steps before plugging it into the Board of Education:

- ✓ Make sure no circuits are connected to I/O sockets P0, P1, P5, and P6 on the Board of Education.
- ✓ Open End.bs2 with the BASIC Stamp Editor and download it to the BASIC Stamp:

```
' End.bs2
' {$STAMP BS2}
END
```

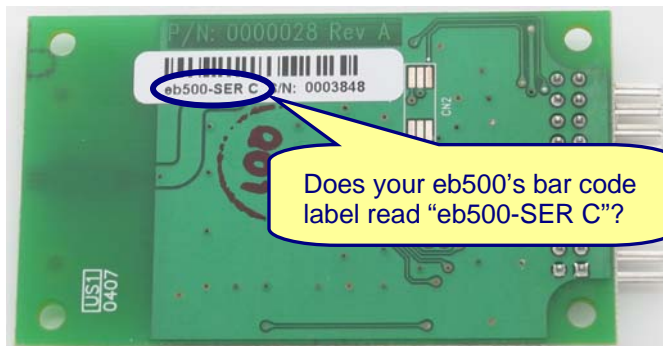


Figure 1: Plugging the eb500 into the Board of Education

(Excerpt from eb500 Users Manual)

- ✓ Turn off the Board of Education's power (set the 3-position switch to 0).
- ✓ Check the white barcode label on your eb500. **If it reads “eb500-SER C” as shown in Figure 2, go to Appendix B on page on page 22 before continuing with these instructions.**

Figure 2: Checking eb500 Barcode Label for SER C



- ✓ Plug the eb500 module into the Board of Education AppMod header as shown in Figure 1.
- ✓ Turn the power back on (move the 3-position switch to 1).
- ✓ Set up the Bluetooth connection between your PC and the eb500. The PIN code/Passkey is 0000.



For an example of setting up a Bluetooth connection between the PC and eb500, see the eb500 Users Manual, pages 31 - 35 and 67 - 72.

- ✓ Find out which COM port number your eb500 'A7 Serial Port' has been assigned, and make a note of it. If you are unsure how to do this, see the information box below.

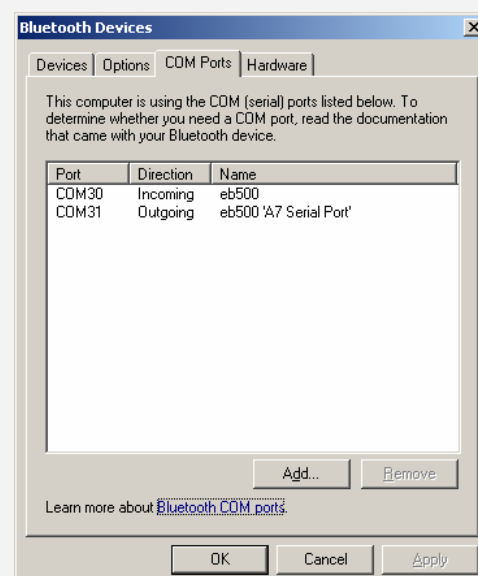


Instructions for identifying this COM port can differ with various PC's Bluetooth radios and their accompanying software.

To the right is an example of finding the eb500 'A7 Serial Port' with a D-LINK DBT-120 Wireless Bluetooth 2.0 USB Adaptor.

- ✓ Right-click the Bluetooth devices icon in your system tray and select Show Bluetooth devices.
- ✓ Click the COM Ports tab and find out what COM port your eb500 'A7 Serial Port' is connected to. Make a note of it.

Figure 3: eb500 A7 Serial Port



First Boe-Bot Test Circuit

The first service that will be tested is the Robotics Studio Simple Dashboard user interface (UI). You will be able to use this UI to drive the Boe-Bot around by controlling a virtual joystick with your mouse. Some circuits need to be built on the Boe-Bot before testing the UI, and they should be tested with the BASIC Stamp before attempting to use them with the UI.

✓ Build the circuit shown in Figure 4 with the aid of the wiring diagram in Figure 5.

Figure 4: Basic Boe-Bot Robot Schematic

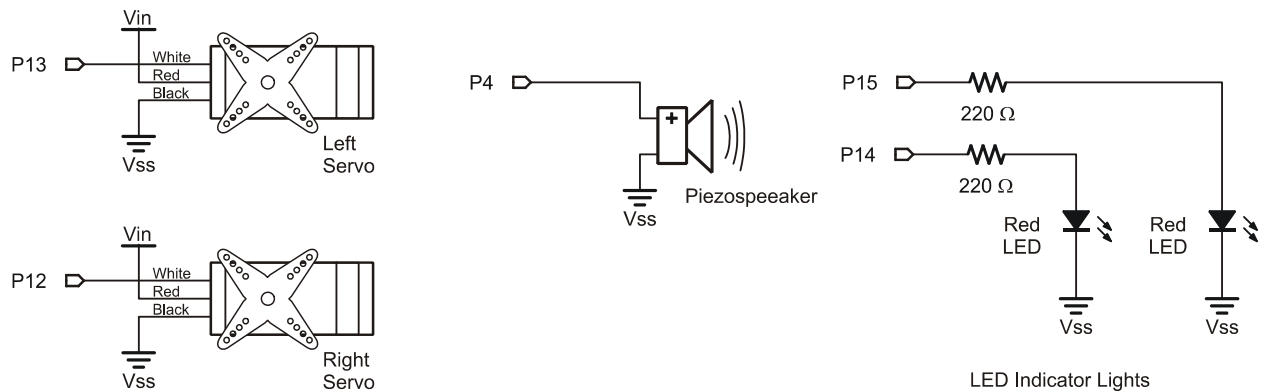
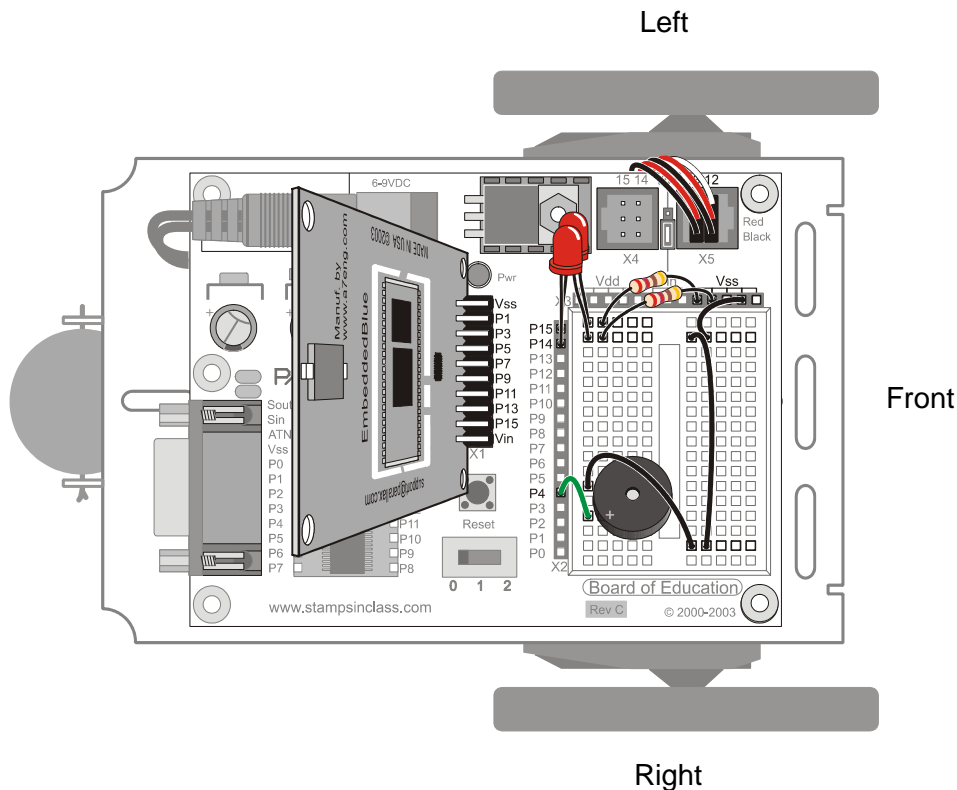


Figure 5: Basic Boe-Bot Robot Schematic



- ✓ Set the Board of Education's 3-position switch to position-1.
- ✓ Load TestSpeakerLedsServos.bs2 into the BASIC Stamp Editor and download it to the BASIC Stamp.
- ✓ Press and hold the Board of Education's Reset button as you slide the 3-position switch to position-3. Then, let go of the Reset button.
- ✓ Verify that the Boe-Bot beeps, turns its lights on and off, and then goes a short distance forward, followed by a rotate left, then a rotate right, then backward a short distance. (Figure 5 shows the Boe-Bot front, left and right for this test.)



If the Boe-Bot backs up, rotates right, then left, then goes forward, it means the servo cables are swapped. The left servo cable should be connected to port 13, and the right servo should be connected port 12.

If the lights do not turn on, double-check your wiring, and especially make sure the LED's long pins plug into P14 and P15.

Programming the Boe-Bot Robot to Communicate with Microsoft Robotics Studio

- ✓ Connect the Boe-Bot's Board of Education to your computer with the programming cable.
- ✓ Open BoeBotControlForMsrs.bs2 with the BASIC Stamp Editor software.
- ✓ Download the program to the BASIC Stamp (CTRL + R).

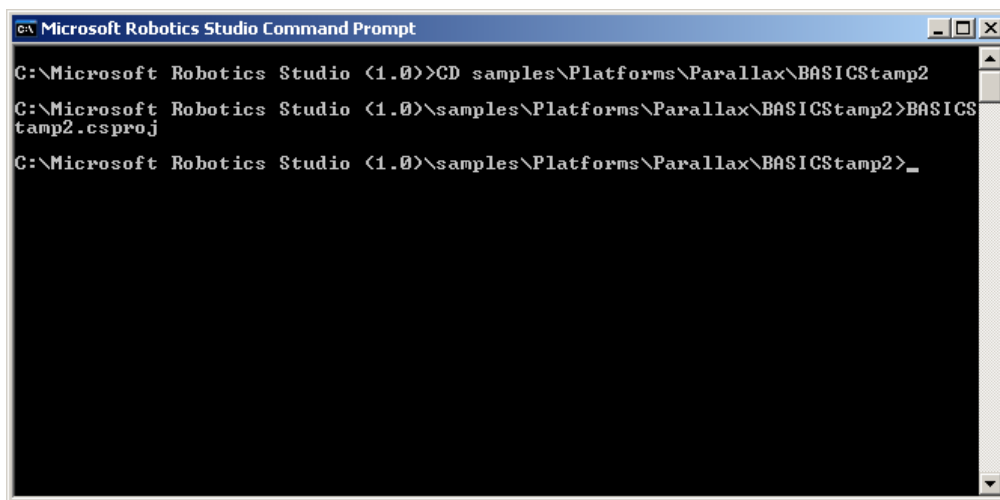
The BASIC Stamp Editor software will automatically open a Debug Terminal window. The Boe-Bot should emit one brief high pitched chip as it flashes the P15 LED on/off. Then, it should do nothing else until it starts getting instructions from the PC.

- ✓ Close the Debug Terminal.
- ✓ Disconnect the programming cable.

Boe-Bot Remote Control with a Robotics Studio Service

- ✓ If you do not already have a Microsoft software development package with Visual C#, download Microsoft Visual C # 2005 Express Edition from msdn.microsoft.com and install.
- ✓ If you have not already done so, download and install the latest version of Microsoft Robotics Studio.
- ✓ Run the Robotics Studio Command Prompt. (Click Start, then select all Programs → Microsoft Robotics Studio... → Microsoft Robotics Studio Command Prompt.)
- ✓ Navigate to the BASIC Stamp 2 samples folder by typing in `CD samples\Platforms\Parallax\BASICStamp2`, and then press Enter. (See Figure 6.)
- ✓ Next type `BASICStamp2.csproj` at the prompt and then press Enter. The command prompt will open the project into Microsoft Visual Studio.

Figure 6: Opening BASICStamp2.csproj with the Robotics Studio Command Prompt



- ✓ If the code for BASICStamp2.cs does not show in the editor pane, double-click the file in the Solution Explorer. (See Figure 7.)

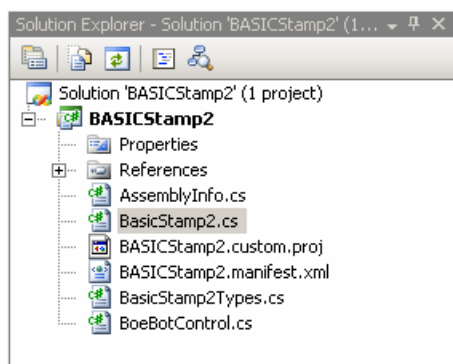


Figure 7: BASICStamp2.cs in Visual Studio's Solution Explorer

- ✓ Find the line that reads `_state.Configuration.SerialPort = 4;` (See Figure 8) and update the value to the COM port number from your Bluetooth devices window. For example, with the information from Figure 3, the statement would be `_state.Configuration.SerialPort = 31;`

```
protected override void Start()
{
    if (_state == null)
    {
        _state = new BasicStampState();
        _state.AutonomousMode = false;
        _state.Configuration = new Config();
        _state.Configuration.SerialPort = 4;
        _state.Configuration.Delay = 0;
        _state.Connected = false;
        _state.FrameCounter = 0;
        _state.InfraRed = new InfraRed();
        _state.MotorSpeed = new MotorSpeed();
        SaveState(_state);
    }
}
```

Figure 8: Updating the COM Port

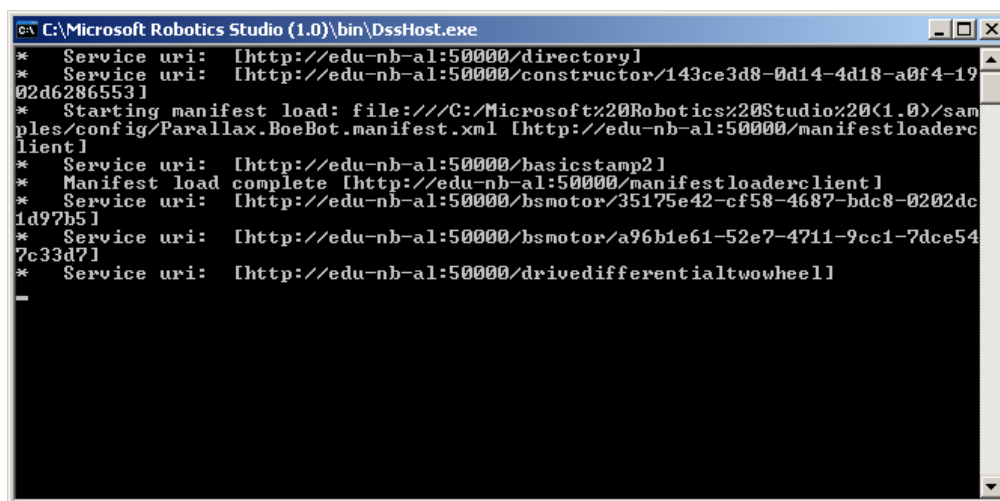
- ✓ Build the project (Right-click BASICStamp2 in the Solution Explorer and select Build. If the Save File As window appears, verify that the File name is BASICStamp2.sln and click Save.)
- ✓ Right-click Solution 'BASICStamp2' (1 project) and select Add → Existing project.
- ✓ In the Add Existing Project window, navigate into the Microsoft Robotics Studio folder's \samples\Platforms\Parallax\BSServices folder and select BSServices.csproj. Then click Open. Visual Studio will add this project, and it should appear in the Solution Explorer window.
- ✓ Click the + next to the BSServices Project's Config folder in the Solution Explorer pane.
- ✓ Double-click the Parallax.BoeBot.Config.xml file.
- ✓ Update the value in this line `<ComPort>6</ComPort>` with your Bluetooth COM port number. (Make sure to use the same value you used in BASICStamp2.cs).
- ✓ Find BoeBotControl.cs in the Solution Explorer and double-click it.
- ✓ Make sure BoeBotControl.cs's `_autonMode` variable is initialized to `false`. You shouldn't have to change it if this is your first time in the file. It should already look like this:

```
bool _autonMode = false;
```

- ✓ Click the Save all button.
- ✓ Right-click Solution 'BASICStamp2' (2 projects), and select Build Solution.
- ✓ Make sure the power to your Boe-Bot is on for both microcontroller and servo ports (3-position switch to position-2).
- ✓ Press and release the Board of Education's Reset button. The BoeBotControlForMsrs.bs2 program the BASIC Stamp is running and should make the Boe-Bot emit one brief high-pitched chip as it flashes the P15 LED on/off. Then, it should do nothing else until it starts getting instructions from the PC.
- ✓ Click the green Start Debugging (F5) button on Visual Studio's button bar or press F5 to run the project in debug mode.

At this point, three things should happen:

- (1) A second Microsoft Robotics Studio Command window should appear similar to the one shown below.
- (2) The green LED indicator light on your EB500 next to the D1 label should also turn on, indicating a Bluetooth connection.
- (3) The Boe-Bot should beep twice as it flashes its LED indicators (the ones you wired onto the breadboard).



```
C:\Microsoft Robotics Studio (1.0)\bin\DssHost.exe
* Service uri: [http://edu-nb-al:50000/directory1]
* Service uri: [http://edu-nb-al:50000/constructor/143ce3d8-0d14-4d18-a0f4-1902d62865531]
* Starting manifest load: file:///C:/Microsoft%20Robotics%20Studio%20(1.0)/samples/config/Parallax.BoeBot.manifest.xml [http://edu-nb-al:50000/manifestloaderclient1]
* Service uri: [http://edu-nb-al:50000/basicstamp21]
* Manifest load complete [http://edu-nb-al:50000/manifestloaderclient1]
* Service uri: [http://edu-nb-al:50000/bsmotor/35175e42-cf58-4687-bdc8-0202dc1d97b51]
* Service uri: [http://edu-nb-al:50000/bsmotor/a96b1e61-52e7-4711-9cc1-7dce547c33d71]
* Service uri: [http://edu-nb-al:50000/drivedifferentialetwowheel1]
```

Trouble Shooting



If the eb500's green light did not come on, click Visual Studio's square stop button to exit the debugging session and double check to make sure that your COM port entries are correct and the solution file compiled. If that doesn't fix the problem, consult the eb500 Users Manual, and try following their Bluetooth and HyperTerminal connection examples to make sure you can get a successful connection.

If the Boe-Bot did not respond with the speaker and LEDs, try pressing and releasing the Board of Education's Reset button.

Now that the Microsoft Robotics Studio has established a Bluetooth communication channel with the Boe-Bot, try these next steps to launch a Simple Dashboard Service from the Robotics Studio Control Panel. The result will be a dashboard window with a virtual joystick that you can use to control the Boe-Bot's motion.

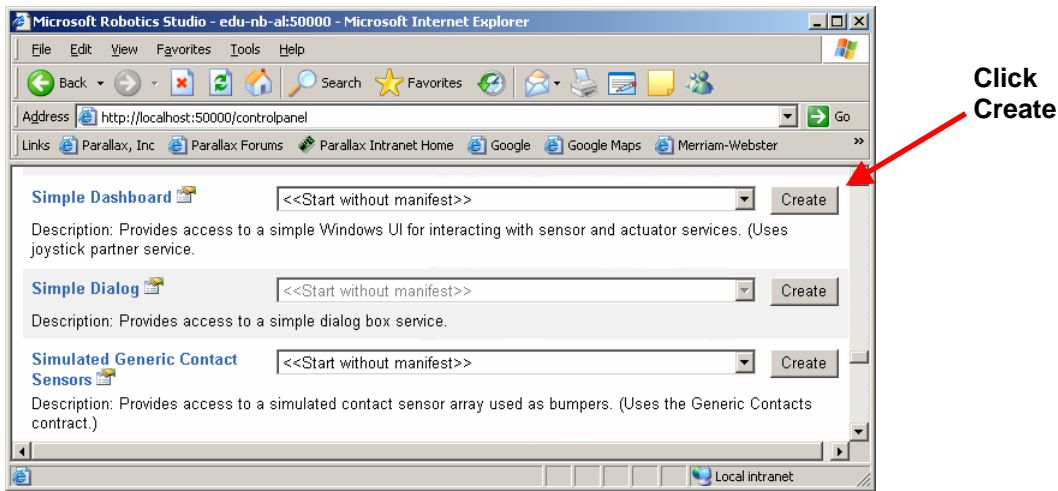
- √ Launch Microsoft Internet Explorer and enter this into the Address field:

<http://localhost:50000/controlpanel>

A Microsoft Robotics Studio Control Panel should appear.

- √ Scroll down and find the Simple Dashboard service shown in Figure 9.
- √ Click the Simple Dashboard entry's Create button.

Figure 9: Control Panel Service



The Dashboard user interface (upper half shown in Figure 10) should appear. If it doesn't appear, check your Windows taskbar because it may have appeared behind another window.

- ✓ Type the word *localhost* into the Machine field.
- ✓ Type 50001 into the Port field.
- ✓ Click Connect.

The contents of the machine field should change from *localhost* to your computer's name, and the */drivedifferentiawheel* service will be listed as running.

- ✓ Double-click */drivedifferentiawheel*
- ✓ Click the Drive button.
- ✓ Click and drag the round eyeball control's crosshairs (See Figure 10) to drive the Boe-Bot around as a joystick-operated device.
- ✓ When you're done, click Visual Studio's square Stop Debugging (Shift + F5) button to end the dashboard debugging session.

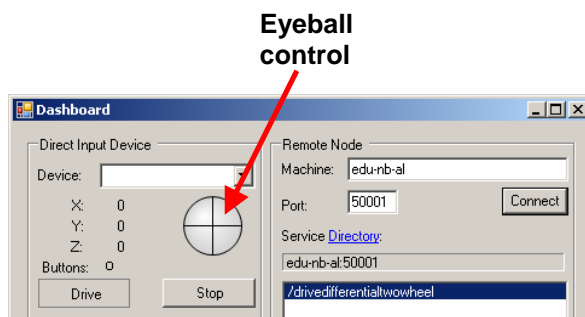



Figure 10: Dashboard Service User Interface

Trying out a Microsoft Robotics Tutorial

Another example of controlling your Boe-Bot with a PC window can be found in Microsoft Robotics Studio Tutorial #4. Here is a link to an article that explains how to load and run the pre-written tutorial code to control the Boe-Bot with another Windows user interface, shown in Figure 11. Additional Microsoft Robotics Tutorials are included, starting on page 14 of this document.

<http://www.devx.com/dotnet/Article/33216>



Wait for communication to be established before clicking the buttons!

The Boe-Bot beeps twice (and the LEDs flash twice) to indicate that it has established communication with Microsoft Robotics Studio. Wait for those two beeps before clicking the buttons shown in Figure 11.

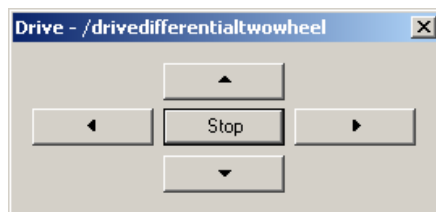


Figure 11: A Windows User Interface Example

Semiautonomous Navigation Test

In this example, the Boe-Bot and Microsoft Robotics Studio share Boe-Bot control and decision-making. With the infrared object detection navigation example, the Boe-Bot lets Robotics Studio make most of the decisions. However, the BoeBotControl.cs file is modified so that, in the event of a whisker contact, the Boe-Bot takes control. First, it halts forward motion and notifies Robotics Studio of the event. Then, it waits until it receives a message from Robotics Studio. The message from Microsoft Robotics Studio will instruct the Boe-Bot to perform one of three maneuvers that reside in its .bs2 file. After the Boe-Bot completes its maneuver, it resumes taking instructions from Robotics Studio. It all happens pretty quickly, and it's difficult to tell the difference between the fully autonomous and the Robotics Studio-controlled actions.

- ✓ If you tried the Microsoft Robotics Tutorials, retrace your steps in the Boe-Bot Remote control with a Robotics Studio Service section to the point where you are looking at this declaration `bool _autonMode = false;` in the BoeBotControl.cs file.
- ✓ Build the circuit shown in the Figure 12 schematic with the Figure 13 wiring diagram as a guide.

Figure 12: Boe-Bot Circuit with Infrared and Whisker Object Detectors

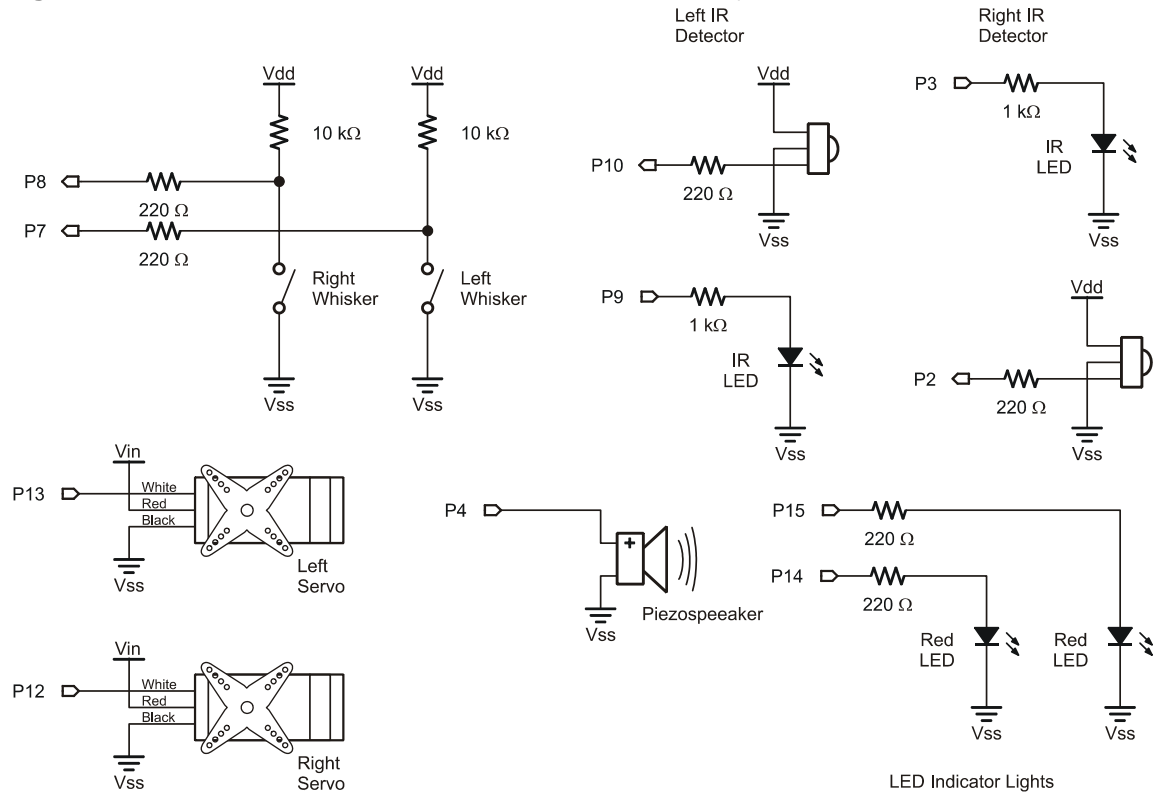
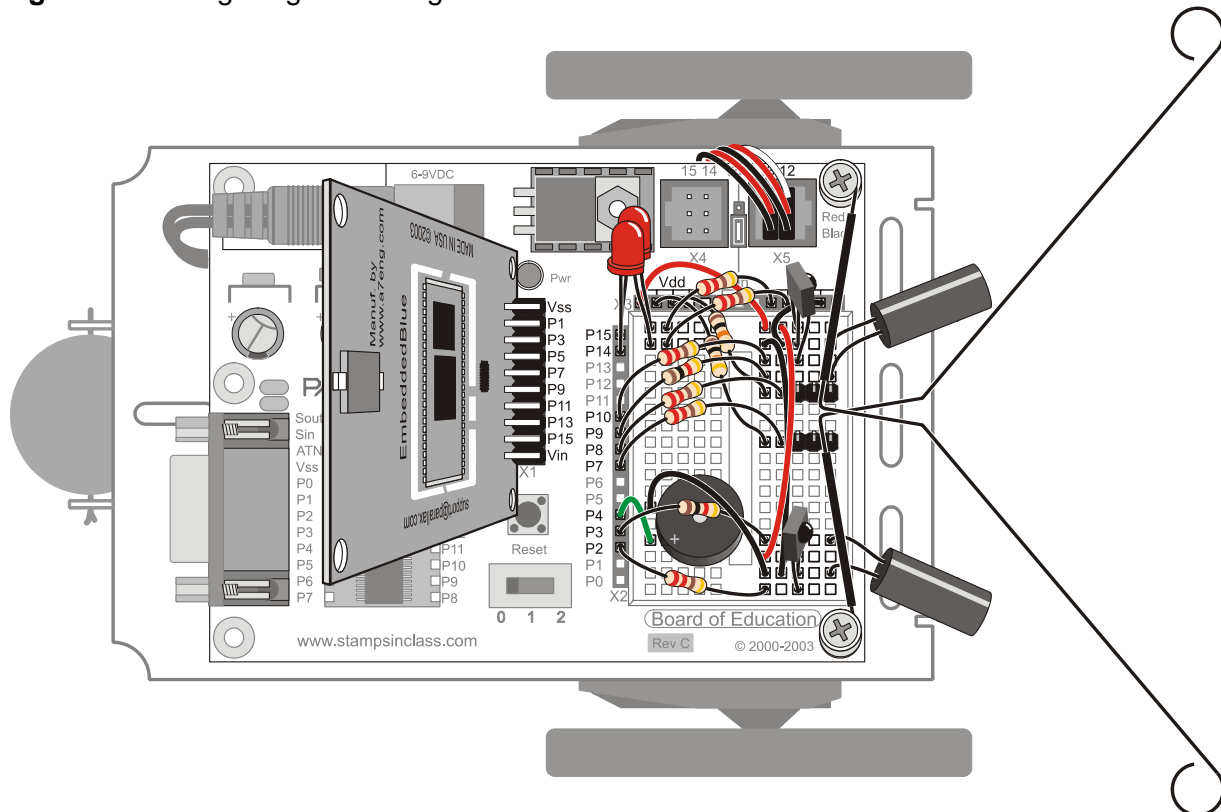


Figure 13: Wiring Diagram for Figure 12



- ✓ Insulate the whiskers with shrink-wrap tubing or electrical tape as shown in Figure 14.
- ✓ Make sure the metal part of each whisker is sitting between the front two pins of the male-male headers. The whisker wire should not be resting against the header posts until the Boe-Bot bumps into an object.

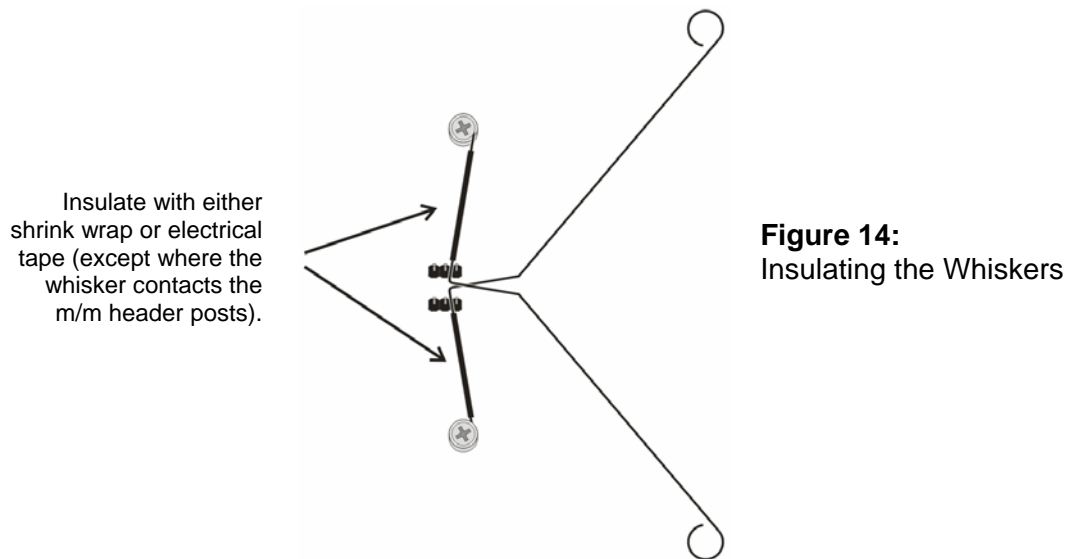


Figure 14:
Insulating the Whiskers

- ✓ Open TestWhiskersAndIr.bs2 in the BASIC Stamp Editor and download it to the BASIC Stamp.
- ✓ Watch the Debug Terminal as you put your hand in front of each infrared detector. The output should change from 1 to 0 when you do so.



If the IR detectors don't work, check your wiring and resistor values. The IR LEDs have two different length pins. In Figure 13, the upper IR LED is a clear LED inside the black cylinder. Its short pin should plug into the second row from the top, and the long pin should plug into the fourth row from the top. Likewise, the lower IR LED's short pin should plug into the second row from the bottom, and the longer pin should plug into the fourth row from the bottom.

- ✓ Press each whisker. As the wire contacts the metal male-male standoff post that's plugged into the breadboard, the Debug Terminal should display a 0 for that whisker. Otherwise, it should display a 1.



In most cases, the IR detectors will detect your hand as you reach to depress the whisker.

- ✓ Repeat the procedure in the Programming the Boe-Bot to Communicate with Microsoft Robotics Studio section that starts on page 6 to reload BoeBotControlForMsrs.bs2 into the BASIC Stamp.
- ✓ In Microsoft Visual Studio, change the `bool _autonMode` declaration in the BoeBotControl.cs file so that it initializes to `true` instead of `false`:

```
bool _autonMode = true;
```



Important: undo the changes to BoeBotControl.cs before trying Microsoft Robotics Studio Tutorials

Before going back to Microsoft Robotics Tutorials or the remote control panel, you will need to change `_autonMode` back to `false`, save all, and build the BasicStamp2 solution file.

- ✓ Find the four lines shown below (in `BoeBotControl.cs`), and make sure that `wFlag = DisableWhiskers();` is commented and `wFlag = EnableWhiskers();` is not.

```
irFlag = EnableIr();  
wFlag = EnableWhiskers();  
//wFlag = DisableWhiskers();  
digFlag = EnableDigitalSensors();
```

- ✓ Save all files.
- ✓ Right-click Solution 'BASICStamp2' (2 projects) in the Solution explorer and select Build.
- ✓ Click Debug (or press F5).
- ✓ Make sure the 3-position switch on the Board of Education is in position-2.

The Boe-Bot should now roam semi-autonomously (mostly under Robotic Studio's control), and navigate around both contact and infrared-detected obstacles.



If the Boe-Bot is changing direction for no apparent reason, try removing it from direct sunlight by covering any nearby windows. Some fluorescent light fixture ballasts may also create infrared interference. Also make sure the Whiskers do not touch the 3-pin headers on the breadboard until pressed by an object that's in the way.

- ✓ To make objects invisible to infrared so that the Boe-Bot will need to detect them with whiskers, wrap the object(s) in black electrical tape. Black vinyl wall base also tends to be invisible to IR. To test it without electrical tape, simply reach down between the IR detectors and press the whiskers with your finger.

Microsoft Robotics Tutorials

Microsoft Robotics Studio's `\samples\RoboticsTutorials` directory has several tutorials that will help you get familiar with writing robot control code for your PC. Before getting started with this, your `BoeBotControl.cs` file has to be returned to its original configuration.

- ✓ Double-click `BoeBotControl.cs` in the Solution Explorer.
- ✓ Update the `_autonMode` declaration so that it initializes to `false` instead of `true`.

```
bool _autonMode = false;
```

- ✓ Uncomment the line `wFlag = DisableWhiskers();` and then comment the line `wFlag = EnableWhiskers();`. It should look like this:

```
irFlag = EnableIr();  
//wFlag = EnableWhiskers();  
wFlag = DisableWhiskers();  
digFlag = EnableDigitalSensors();
```

- ✓ Click Save.
- ✓ Click Build → Build Solution (F6).

Before following the tutorials step-by-step and hand-entering the code, it's a good idea to run the prewritten code to make sure it works. Here are three examples of opening and running the completed Robotics Tutorial 1 code.

Robotics Tutorial 1

The end result of this tutorial is code that displays whether or not the Boe-Bot's right IR detectors detect an object in a DssHost.exe window.

- ✓ Use the Microsoft Robotics Studio Command Prompt to open this file:

`\samples\RoboticsTutorials\Tutorial1\CSharp\RoboticsTutorial1.csproj`

- ✓ In the Solution Explorer, click the + next to the Config folder.
- ✓ Double-click Parallax.MotorIrBumper.Config.xml.
- ✓ Update the serial port configuration to the correct value for your Bluetooth COM port. Continuing with the example from Figure 3, the value 31 is used below.

```
<!-- *** CONFIGURE THE PARALLAX BLUETOOTH SERIAL PORT HERE *** -->
<SerialPort>31</SerialPort>
<!-- *** CONFIGURE THE PARALLAX BLUETOOTH SERIAL PORT HERE *** -->
```

- ✓ In the Solution Explorer, double-click RoboticsTutorial1.cs.
- ✓ Find the text "`Ouch - the bumper was pressed.`" and modify it to read "`Object detected!`".
- ✓ Double-click Properties in the Solution Explorer pane. This should open up the RoboticsTutorial1 Properties.
- ✓ Click the Debug tab to view the Start Options. It should look similar to Figure 15, except the command line arguments field will probably be empty.
- ✓ Copy the text below into the command line arguments field:

`-p:50000 -t:50001 -m:"Samples\Config\RoboticsTutorial1.manifest.xml" -m:"samples\config\Parallax.MotorIrBumper.manifest.xml"`

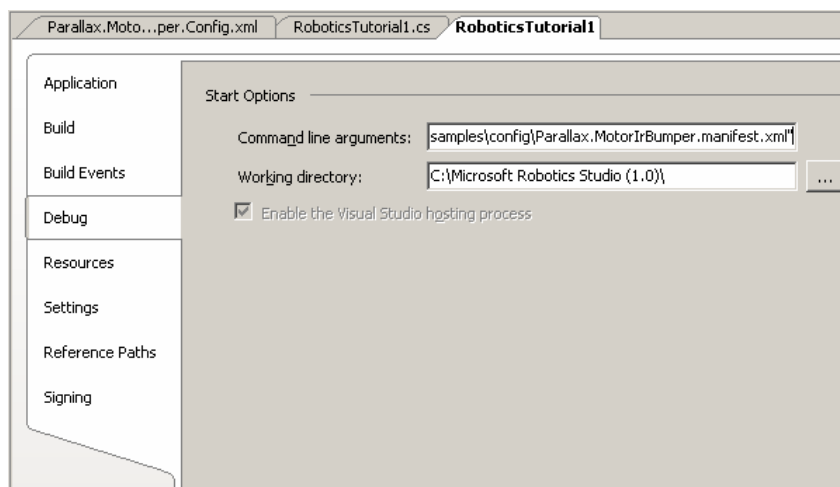
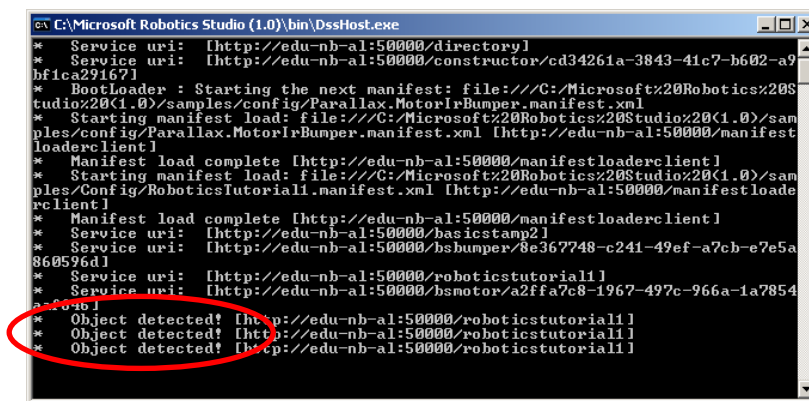


Figure 15:
RoboticsTutorial1
Properties Debug Tab

- ✓ Click Save All.

- ✓ Click Build → Build Solution (F6).
- ✓ Make sure the IR detectors are directed so that they will not detect a nearby object. You may also want to make sure any nearby windows are covered to eliminate interference from direct sunlight.
- ✓ Make sure your Boe-Bot is on. Since this tutorial is only for testing one of the IR sensors, the 3-position switch on the Boe-Bot's Board of Education can be in position-1.
- ✓ In Visual Studio, click the Start Debugging (F5) button.

The DssHost.exe window shown in Figure 16 will appear. The Boe-Bot should beep twice (and the LEDs will blink twice) indicating that the communication link between Microsoft Robotics Studio and the Boe-Bot has been established. Each time you wave your hand in front of the Boe-Bot's right IR detector, it will display a new "Object detected!" message.



```

C:\Microsoft Robotics Studio (1.0)\bin\DssHost.exe
* Service uri: [http://edu-nb-al:50000/directory1]
* Service uri: [http://edu-nb-al:50000/constructor/cd34261a-3843-41c7-b602-a9bf1ca22167]
* Bootloader : Starting the next manifest: file:///C:/Microsoft%20Robotics%20Studio%20(1.0)/samples/config/Parallax.MotorIrBumper.manifest.xml
* Starting manifest load: file:///C:/Microsoft%20Robotics%20Studio%20(1.0)/samples/config/Parallax.MotorIrBumper.manifest.xml [http://edu-nb-al:50000/manifestloaderclient1]
* Manifest load complete [http://edu-nb-al:50000/manifestloaderclient1]
* Starting manifest load: file:///C:/Microsoft%20Robotics%20Studio%20(1.0)/samples/Config/RoboticsTutorial1.manifest.xml [http://edu-nb-al:50000/manifestloaderclient1]
* Manifest load complete [http://edu-nb-al:50000/manifestloaderclient1]
* Service uri: [http://edu-nb-al:50000/basicstamp2]
* Service uri: [http://edu-nb-al:50000/hsbumper/8e367748-c241-49ef-a7cb-e7e5a860596d1]
* Service uri: [http://edu-nb-al:50000/roboticstutorial1]
* Service uri: [http://edu-nb-al:50000/hsmotor/a2ffa7c8-1967-497c-966a-1a7854a2a0461]
* Object detected! [http://edu-nb-al:50000/roboticstutorial1]
* Object detected! [http://edu-nb-al:50000/roboticstutorial1]
* Object detected! [http://edu-nb-al:50000/roboticstutorial1]
  
```

Figure 16: Object Detected Messages in the DssHost.exe Window

Robotics Tutorial 2

The end result of this tutorial is that you can wave your hand in front of the Boe-Bot's right IR detector to turn its motion (a circular path) on and off. The process is similar to the previous tutorial in the following ways:

- 1) The tutorial is opened with the Microsoft Robotics Studio Command prompt.
- 2) The Parallax.MotorIrBumper.Config.xml file (in the Config folder) is opened and the SerialPort configuration is updated.
- 3) The Command line argument field in the project Properties' Debug tab is updated with a file path to the Parallax hardware manifest.
- 4) The project is executed with the Microsoft Visual Studio's Debug feature.

Here is each step along with checklist instructions.

1) The tutorial is opened with the Microsoft Robotics Studio Command prompt.

- ✓ Use the Microsoft Robotics Studio Command Prompt to open this file:

\samples\RoboticsTutorials\Tutorial2\CSharp\RoboticsTutorial2.csproj.

2) *The Parallax.BoeBot.Config.xml file is opened and the SerialPort configuration is updated.*

- ✓ In the Solution Explorer, click the + next to the Config folder.
- ✓ Double-click Parallax.MotorIrBumper.Config.xml.
- ✓ Update the serial port configuration to the correct value for your Bluetooth COM port. Continuing with the example from Figure 3, the value 31 is used below.

```
<!-- *** CONFIGURE THE PARALLAX BLUETOOTH SERIAL PORT HERE *** -->
<SerialPort>31</SerialPort>
<!-- *** CONFIGURE THE PARALLAX BLUETOOTH SERIAL PORT HERE *** -->
```

3) *The Command line argument field in the project Properties' Debug tab is updated with a file path to the Parallax hardware manifest.*

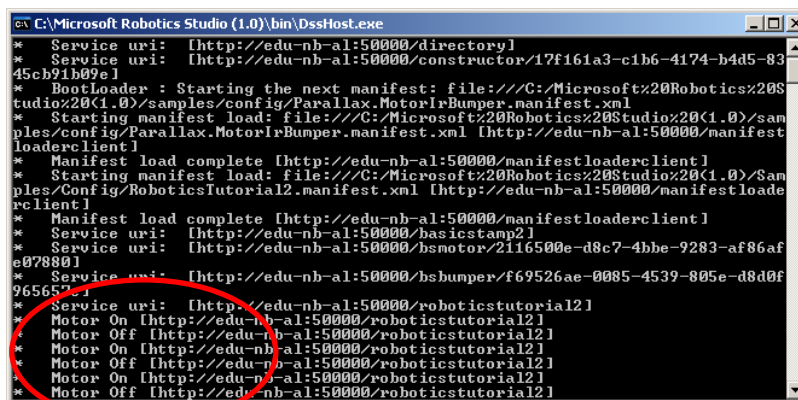
- ✓ Double-click Properties in the Solution Explorer pane. This should open up the RoboticsTutorial1 Properties.
- ✓ Click the Debug tab to view the Start Options. It should look similar to Figure 15, except the command line arguments field will probably be empty.
- ✓ Copy the text below into the command line arguments field:

```
-p:50000 -t:50001 -m:"Samples\Config\RoboticsTutorial2.manifest.xml" -m:"samples\config\Parallax.MotorIrBumper.manifest.xml"
```

4) *The project is executed with the Microsoft Visual Studio's Debug feature.*

- ✓ Click Save All.
- ✓ Click Build → Build Solution (F6).
- ✓ Make sure the IR detectors are directed so that they will not detect a nearby object. Also, cover any nearby windows to eliminate interference from sunlight.
- ✓ Make sure your Boe-Bot is on. This tutorial involves Boe-Bot motion, so the 3-position switch on the Boe-Bot's Board of Education has to be in position-2.
- ✓ Click Start Debugging (F5).

The DssHost.exe window shown in Figure 17 will appear again. Each time you wave your hand in front of the Boe-Bot's right IR detector, it will toggle the Boe-Bot's motors on/off. The Boe-Bot will travel in a fairly tight circle when the motors are running.



```
C:\Microsoft Robotics Studio (1.0)\bin\DssHost.exe
* Service uri: [http://edu-nb-al:50000/directory1]
* Service uri: [http://edu-nb-al:50000/constructor/17f161a3-c1b6-4174-b4d5-8345cb91b09e1]
* BootLoader : Starting the next manifest: file:///C:/Microsoft%20Robotics%20Studio%20(1.0)/samples/config/Parallax.MotorIrBumper.manifest.xml
* Starting manifest load: file:///C:/Microsoft%20Robotics%20Studio%20(1.0)/samples/config/Parallax.MotorIrBumper.manifest.xml [http://edu-nb-al:50000/manifestloaderclient1]
* Manifest load complete [http://edu-nb-al:50000/manifestloaderclient1]
* Starting manifest load: file:///C:/Microsoft%20Robotics%20Studio%20(1.0)/Samples\Config\RoboticsTutorial2.manifest.xml [http://edu-nb-al:50000/manifestloaderclient1]
* Manifest load complete [http://edu-nb-al:50000/manifestloaderclient1]
* Service uri: [http://edu-nb-al:50000/basicstamp21]
* Service uri: [http://edu-nb-al:50000/bsmotor/2116500e-d8c7-4bbe-9283-af86afe078091]
* Service uri: [http://edu-nb-al:50000/bsbumper/f69526ae-0085-4539-805e-d8d0f9656521]
* Service uri: [http://edu-nb-al:50000/roboticstutorial21]
* Motor On [http://edu-nb-al:50000/roboticstutorial21]
* Motor Off [http://edu-nb-al:50000/roboticstutorial21]
* Motor On [http://edu-nb-al:50000/roboticstutorial21]
* Motor Off [http://edu-nb-al:50000/roboticstutorial21]
* Motor On [http://edu-nb-al:50000/roboticstutorial21]
* Motor Off [http://edu-nb-al:50000/roboticstutorial21]
```

Figure 17: Motor on/off Messages in the DssHost.exe Window

Robotics Tutorial 4

The end result of this tutorial is a user interface window you can use to select Boe-Bot maneuvers. The user interface is shown in Figure 11 and repeated in Figure 18 below for convenience.

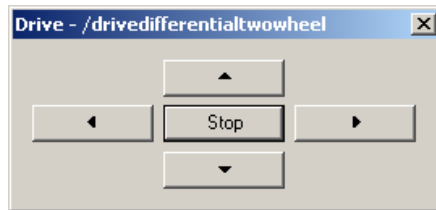


Figure 18: A Windows User Interface Example

As with the previous tutorial, there are four major steps to getting it up and running:

- 1) The tutorial is opened with the Microsoft Robotics Studio Command prompt.
- 2) The Parallax.BoeBot.Config.xml file is opened and the SerialPort configuration is updated.
- 3) The Command line argument field in the project Properties' Debug tab is updated with a file path to the Parallax hardware manifest.

Note that the manifest file in the Solution explorer is different. Make sure this is reflected in the Command line argument field.

```
-p:50000 -t:50001 -m:"Samples\Config\RoboticsTutorial4.manifest.xml" -m:"samples\config\Parallax.BoeBot.manifest.xml"
```

- 4) The project is executed with the Microsoft Visual Studio's Debug feature.



Wait for communication to be established before clicking the buttons!

The Boe-Bot beeps twice (and the LEDs flash twice) to indicate that it has established communication with Microsoft Robotics Studio. Wait for those two beeps before clicking the buttons.

Appendix A: Communication Protocol

The protocol was designed to make it possible for the single-threaded BASIC Stamp microcontroller to stop paying attention to incoming packets whenever it needs to perform sensor monitoring and servo control tasks without ever missing a packet. The protocol also makes it possible for the BASIC Stamp to periodically update the servo control signals at a rate that is independent of the rate the packets are delivered by the Bluetooth serial link.

The protocol accomplishes this by making the PC transmit a given packet repeatedly until it receives a reply packet from the BASIC Stamp. Although the BASIC Stamp's reply packet may contain other information, it has an acknowledgement feature in the form of an index byte. The BASIC Stamp increments this index byte and puts it in the reply packet. The PC must use the incremented index in the next packet it transmits to the BASIC Stamp. The BASIC Stamp uses the index to discard any repeats of the old packet that arrive while it is polling for the next packet. The BASIC Stamp starts polling immediately after sending the reply packet, but it has to filter for and discard repeats of the previous packet because its polling and robot control loop may repeat several times during packet delivery delays introduced by the Bluetooth system.

The BASIC Stamp polls the serial data for the 5-byte packet's start byte (the value 255) for up to 5 ms before each repetition of its servo control and sensor monitoring loop. The start byte is followed by the message index, then the command byte, and two data bytes as shown below. The 255 start byte followed by an index provides an implied CRC, which is necessary above and beyond what's included in the Bluetooth protocol. Reason being, the BASIC Stamp might resume polling in the middle of a serial byte that is relayed by the EB500 Bluetooth module. None of the other 4 bytes are allowed to contain 255, which makes it impossible for the BASIC Stamp to erroneously receive a 255 byte followed by the correct index value, regardless of when it starts polling for the next packet. The net loss of 2 bits of values (255 illegal in four bytes) is preferable to loading the packet with 8 more bits for CRC, which would decrease the protocol's bandwidth.

The 5-byte Packet

255	message index	command	data 1	data 2
-----	---------------	---------	--------	--------

255 Start byte. For the sake of simplicity and reliability, no other byte is allowed to store a 255.

message index Incremented by the BASIC Stamp and returned to the PC in a confirmation packet that may contain different command and data bytes.

command A value from 0 to 254 that selects an action that the BASIC Stamp is programmed to take. See the Command Set section below for the table of 18 predefined commands; the rest are available for customization.

data 1 and data2 Bytes that contain the command's arguments, or data from the BASIC Stamp's reply.

Command Set

This command set was designed with 255 possible values (0 to 254) to leave room for lots of enhancements and modifications. Only 18 values have been predefined.

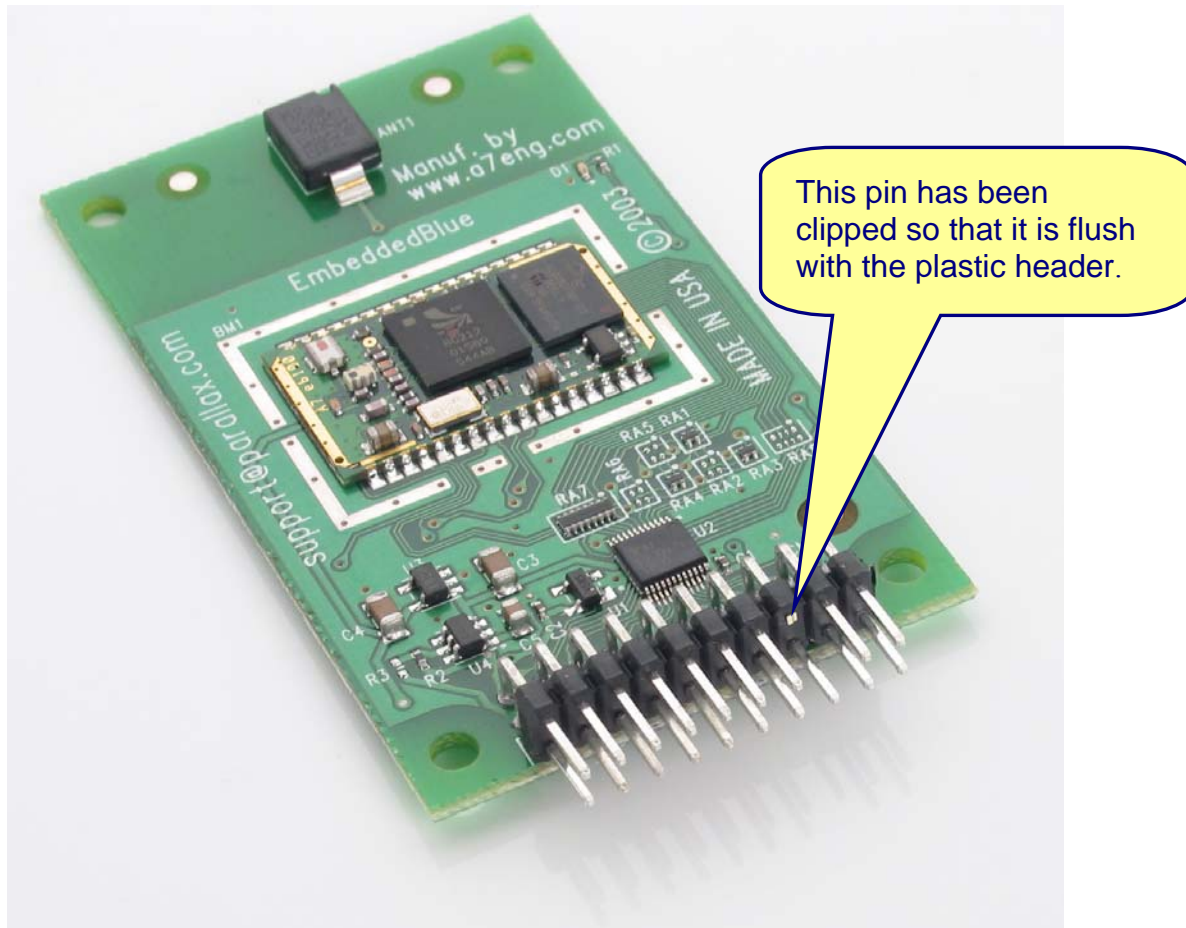
Start Byte	Message Index	Command	Data 1	Data 2
0BASIC Stamp instructs PC to repeat handshake. (The PC's version of this message is command = 192.)				
255	index	0	don't care	don't care
1 BASIC Stamp initiates handshake				
255	index	1	don't care	don't care
2 PC confirms receipt of handshake initiation				
255	index	2	don't care	don't care
3 BASIC Stamp instructs PC to send next command				
255	index	3	don't care	don't care
32 PC transmits Boe-Bot left and right servo speeds Left and right servo speeds are 0 to 200 where: <ul style="list-style-type: none"> • 0 is full speed clockwise • 200 is full speed counterclockwise • 100 is full stop. Note: For most Parallax Continuous rotation servos, the response is close to linear from about 60 to 140.				
255	index	32	left speed	right speed
33PC instructs Boe-Bot to select a preprogrammed maneuver. The current character options are: <ul style="list-style-type: none"> • "U" - back up and make a u-turn • "L" - back up and make a left turn • "R" - back up and make a right turn 				
255	index	33	character	don't care
64 PC instructs Boe-Bot to send Ir detector states <ul style="list-style-type: none"> • x is don't care • L is the state of the left IR detector and • R is the state of the right. Note that states are active-low.				
255	index	64	xxxxxxLR	don't care
65 PC instructs Boe-Bot to send Whisker states bits are xxxxLRxx, where L is the state of the left whisker and R is the state of the right. Note that states are active-low.				
255	index	65	xxxLRxx	don't care

Command Set (continued)

Start Byte	Message Index	Command	Data 1	Data 2
96 PC instructs Boe-Bot to play a tone on the speaker <ul style="list-style-type: none"> duration is in 50 ms units (example 20 -> 1 second) frequency is in 50 Hz units (example 20 -> 1 kHz) 				
255	index	96	duration	frequency
97PC instructs Boe-Bot to set the states of up to two I/O pins Each byte holds the command (0 to 4) in the upper nibble and the pin (0 to 15) in the lower nibble. Command: <ul style="list-style-type: none"> 0 - take no action 1 - Make pin an output 2 - Make pin an input 3 - Make pin output-high 4 - Make pin output-low 				
255	index	97	command/pin	command/pin
98PC instructs Boe-Bot to delay for a specified number of milliseconds. <ul style="list-style-type: none"> Do nothing for the specified number of milliseconds (word value). 				
255	index	98	low byte	high byte
128PC instructs Boe-Bot to enable Digital Sensors Causes every BASIC Stamp reply Data 1 byte to contain xxxlRlR <ul style="list-style-type: none"> l- left whisker r - Right whisker L - Left IR R - Right IR 				
255	index	128	don't care	don't care
129PC instructs Boe-Bot to Enable IR IR is implemented, but it does not need any enable/disable at this time.				
255	index	129	don't care	don't care
130PC instructs Boe-Bot to Enable Whiskers Configures Boe-Bot to halt forward motion upon whisker contact without first consulting with the PC. Since message round trips tend to be in the 160 ms, this mode prevents the Boe-Bot from colliding with an object it contacts before it gets a command back from the PC.				
255	index	130	don't care	don't care
160 - 162 ..Disable versions of 128 - 130				
255	index	160, 161, or 162	don't care	don't care
192PC instructs BASIC Stamp to reset message index and begin the handshake.				
255	index	97	command/pin	command/pin

Appendix B: Special Instructions for eb500-SER C

The eb500 SER C modules send an output signal on a pin that interferes with P3 I/O pin signals on the Board of Education. This in turn interferes with the IR LED in the schematic and wiring diagram on page 12. The easiest way to fix this problem is to clip off the pin as shown here.



Although it's not recommended, an alternative to clipping this pin would be to modify the circuit on page 12 so that the IR LED connected to P3 is instead connected to P10. If you do this, you will also have to modify *BoeBotControlForMrs.bs2* so that it sends IR LED signals on P10 instead of P3. Another thing to keep in mind is that the eb500 pin that's interfering with P3 serves no useful purpose, so it's better to remove it and free up a BASIC Stamp I/O pin for use with other circuits.

If you have an eb500 module labeled SER C but are either not comfortable with clipping off the pin, or if you make a mistake that damages the module while doing so, you may return it for a replacement. For more information, see the *eb500 SER C Notice* on the eb500 product page at www.parallax.com.