

Bluetooth Boe-Bot[®] Robot

Controlled by Microsoft Robotics Studio (#28118)

The Boe-Bot Robot Kit for Microsoft Robotics Studio (MSRS) is a compact (approx 14 x 17 x 12 cm) mobile robot with the ability to roam autonomously while communicating over Bluetooth wireless with a nearby PC running MSRS. Figure 1 highlights some of the kit's major components including its BASIC Stamp 2 microcontroller brain, Board of Education prototyping platform, and eb500 Bluetooth communication AppMod. Figure 1 also highlights some of the Boe-Bot robot's features including an aluminum chassis with slots for mounting a variety of sensors and actuators, differential drive wheels connected to Parallax continuous rotation servos, and a solderless breadboard for building sensors and indicators.

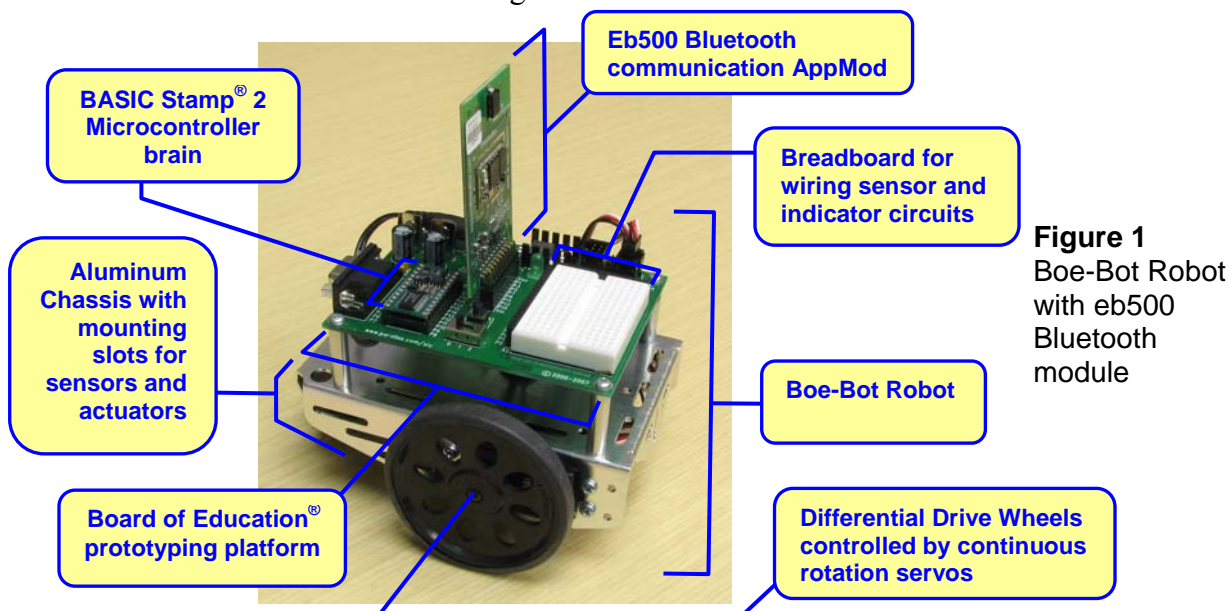



Figure 1
 Boe-Bot Robot
 with eb500
 Bluetooth
 module

 **The Board of Education** (the “Boe” in Boe-Bot) has a myriad of other components and features including headers for connecting the breadboard circuits to power, ground, and BASIC Stamp I/O pins, servo and programming ports, and power connectors, regulators, indicators, and switches. It is often used in conjunction with Stamps in Class Tutorials and kits that introduce electronics, programming and robotics topics in the United States and abroad with texts translated into more than 15 languages.

MSRS makes it possible to write a variety of services, including robot sensor monitoring and motion control code, on your PC with popular languages such as Microsoft Visual C#, Visual BASIC, and Visual Programming Language (VPL). Regardless of the language, the code relies on other services that communicate with the robot (serial over Bluetooth in the case of the Boe-Bot) that request sensor measurements and issue control commands. The Boe-Bot robot's BASIC Stamp Microcontroller runs a PBASIC program that processes these control commands, monitors sensors, controls its continuous rotation drive servos, and replies to MSRS' information requests. With this arrangement, MSRS puts an extensive service library and the PC's processing and data storage abilities at your disposal for Boe-Bot robotics applications.

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.
- Configure MSRS serial communication settings.
- Fix bugs in MSRS (1.5).
- Try an example MSRS user interface (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 (1.5), and Microsoft Visual C# 2005 Express Edition.

Hardware and Software List

- (1) Boe-Bot Kit for Microsoft Robotics Studio, which 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, 6.....Connect, 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 4.....Navigation program examples
- Chapter 5.....Navigation with Whiskers (contact switch sensors)
- Chapter 7, 8.....Navigation with Infrared 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 Serial Communication Module

The eb500 module provides the BASIC Stamp with a serial link to the PC via Bluetooth wireless. Before plugging your eb500 into the Board of Education's X1 socket, there are three very important steps (the next three checklist instructions) that you should follow.



Before you plug your eb500 into the Board of Education

Hot socketing and electrical signals transmitted by or to P5 or P1 can damage the eb500! Before plugging your eb500 into the Board of education, follow the next three checklist instructions to make sure it doesn't happen to your eb500.

- √ Open End.bs2 (also shown below) with the BASIC Stamp Editor and download it to the BASIC Stamp:

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

End.bs2 eliminates the possibility that whatever program was previously downloaded to the BASIC Stamp and stored in its EEPROM program memory might send signals to P5 or P1. All I/O pins initialize to input, and it takes commands such as OUTPUT, HIGH, LOW, FREQOUT, SEROUT, or PULSOUT to make an I/O pin transmit signals. Since End.bs2 doesn't have any of those commands, it sets all the I/O pins to input, and then puts the BASIC Stamp into low power mode.

- √ Turn off the Board of Education's power (set the 3-position switch to 0).
- √ Make sure no circuits on the Board of Education's breadboard are connected to I/O sockets P1 or P5. (It's also a good idea to make sure no breadboard circuits are connected to P0 and P6.)

At this point, it is safe to plug your eb500 into the Board of Education's X1 socket, but there's one last thing to check before doing so.

- √ 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 27 before continuing with these instructions.**



Figure 2
Checking eb500 Barcode Label for SER C

- √ Alright, now plug the eb500 module into the Board of Education AppMod header as shown in Figure 3. Note that the chips on the eb500 face the breadboard.



Figure 3
Plugging the eb500 into
the Board of Education

*(Excerpt from eb500
Users Manual)*

- ✓ 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.

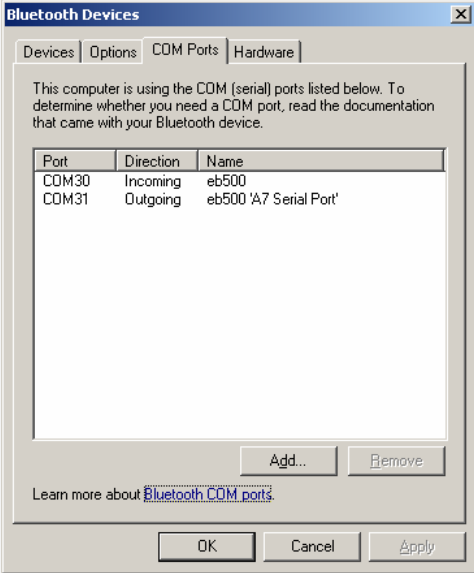
- √ Find out which COM port number your eb500 'A7 Serial Port' has been assigned, and make a note of it. If you are unsure how to do this, see the information accompanying Figure 4.

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.
- √ If you see two eb500 COM port entries, make sure to choose the port with the name: eb500 'A7 Serial Port'.

Figure 4: eb500 'A7 Serial Port'



Port	Direction	Name
COM30	Incoming	eb500
COM31	Outgoing	eb500 'A7 Serial Port'

First Boe-Bot Test Circuit

Figure 5 shows a schematic of the continuous rotation servos you will connect so that the BASIC Stamp can control the Boe-Bot's motion along with light emitting diode (LED) and piezoelectric speaker indicator circuits that you will build for getting MSRS status indications from your Boe-Bot. These devices should first be connected to the Boe-Bot robot's Board of Education platform and tested with the BASIC Stamp 2 microcontroller and a PBASIC program. Knowing that they work properly now can help simplify trouble shooting in later steps.

- √ Turn off the Board of Education's power by setting its 3-position switch to 0.
- √ Build the circuit shown in Figure 5 with the aid of the wiring diagram in Figure 6.

Figure 5: Basic Boe-Bot Robot Schematic

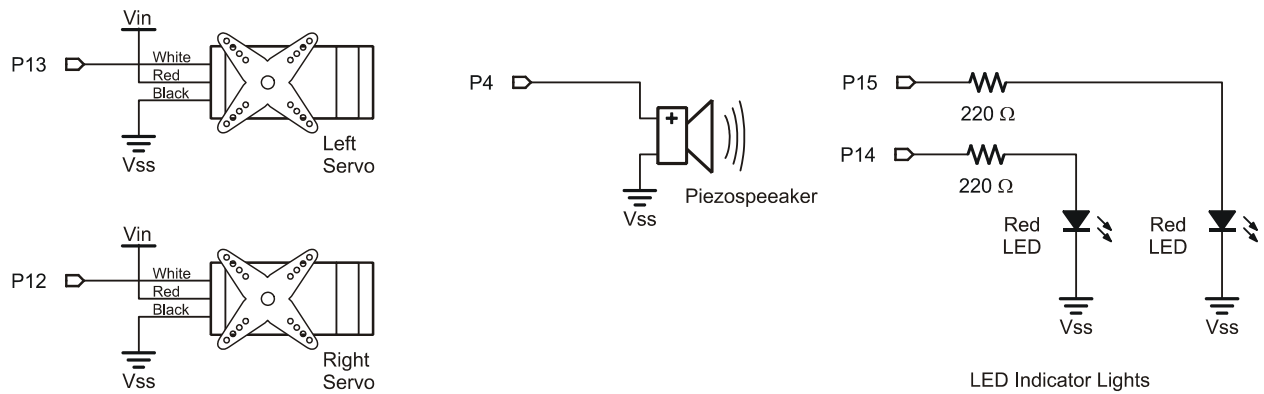
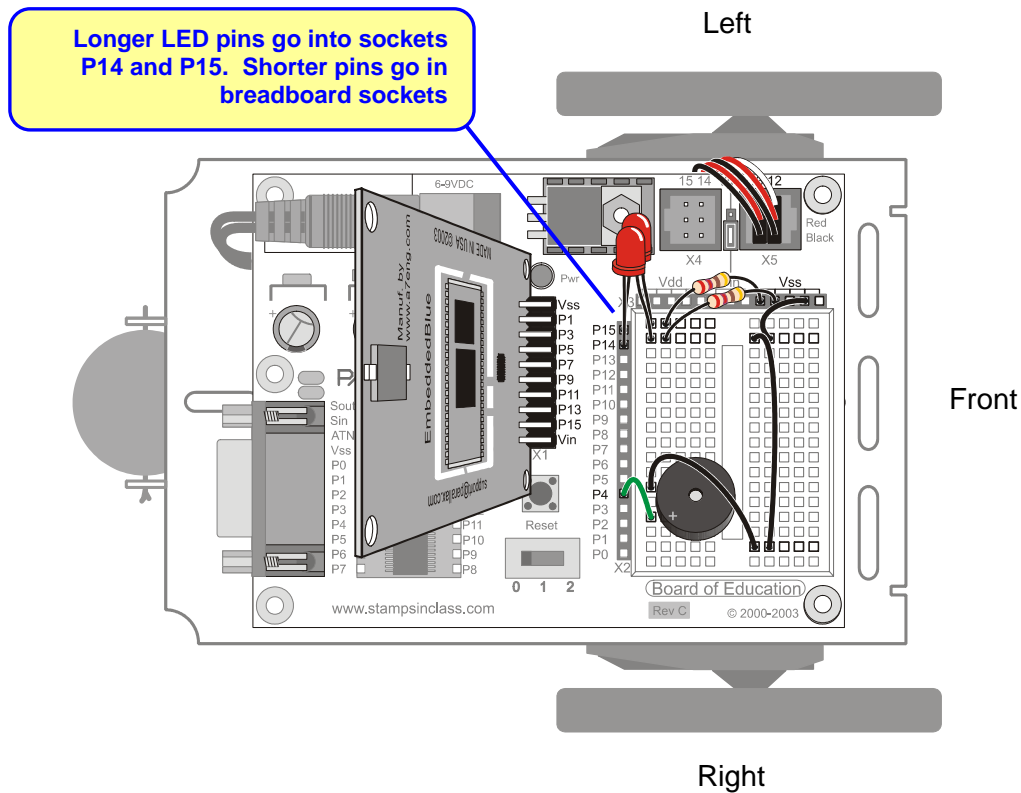


Figure 6: Basic Boe-Bot Robot Schematic



- √ Set the Board of Education's 3-position switch to position-1.
- √ Download and unzip “Bluetooth Boe-Bot Example Code for MSRS v1.2.zip” from the Boe-Bot Kit for Microsoft Robotics Studio page at www.parallax.com.
- √ Load TestSpekerLedsServos.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-2. 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 6 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.

If the speaker does not make a tone, double-check your wiring. Make sure each speaker pin shares a row of five sockets with the two adjacent wires shown in Figure 6. The P4 wire should be plugged into the same breadboard row as the piezospeaker's + pin, and the wire that connects to Vss after a couple more jumpers should share a row with the piezospeaker's other pin.

MSRS Driver Code for the Boe-Bot Robot

After loading the BoeBotControlForMsrs.bs2 “driver code” into the Boe-Bot’s BASIC Stamp, it will be ready to interact with MSRS.

- √ 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 more until it establishes communication with MSRS running on a nearby PC.

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

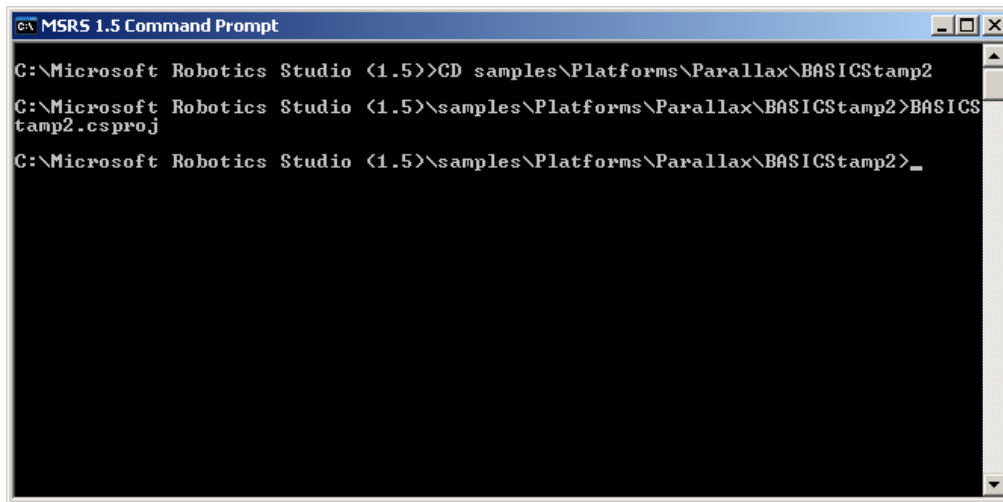
Although the Boe-Bot is now ready to start communicating with MSRS, MSRS has to be configured to communicate with the Boe-Bot.

MSRS Serial Port Settings

All the serial port references in Microsoft Robotics Studio should be updated to the eb500 A7 Serial Port number you determined in Figure 4 on Page 6.

- √ 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... → Command Prompt.)
- √ Navigate to the BASIC Stamp 2 samples folder by typing in CD samples\Platforms\Parallax\BASICStamp2, and then press Enter. (See Figure 7.)
- √ Next type BASICStamp2.csproj at the prompt and then press Enter. The command prompt will open the project into Microsoft Visual Studio.

Figure 7: 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 8.)

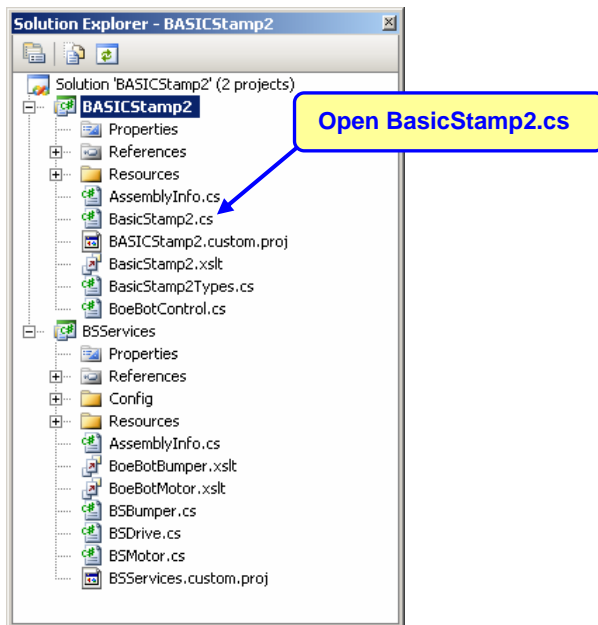


Figure 8
BASICStamp2.cs in
Visual Studio's Solution Explorer

- √ Find the line that reads `private const int DefaultSerialPort = 3;` (See Figure 9) and update the value to the COM port number from your Bluetooth devices window. For example, with the information from Figure 4, the statement would be changed to `private const int DefaultSerialPort = 31;`

Figure 9: Updating the COM Port

```
namespace Microsoft.Robotics.Services.BasicStamp2
{
    // the boe control code uses a polling loop that blocks a thread
    // The ActivationSettings attribute with Sharing == false makes the runtime
    // dedicate a dispatcher thread pool just for this service
    [ActivationSettings(ShareDispatcher=false,ExecutionMode=ExecutionMode.Asynchronous)]
    [Contract(Contract.Identifier)]
    [DisplayName("Boe-Bot BASIC Stamp 2")]
    [Description("Provides access to the Parallax Boe Bot using BASIC Stamp 2.")]
    public class BasicStamp2Service : DsspServiceBase
    {
        private const int DefaultSerialPort = 3;

        [InitialStatePartner(Optional=true, ServiceUri="BasicStamp2.config.xml")]
        private BasicStampState _state = null;

        [EmbeddedResource("Microsoft.Robotics.Services.BasicStamp2.BasicStamp2.xslt")]
        string _transform = null;
    }
}
```

Replace with your Bluetooth Serial Port#

- ✓ If the Config folder in the BSServices project is not expanded, you may need to click the + next to BSServices as well as the + next to Config in the Solution Explorer pane.
- ✓ Double-click Parallax.BoeBot.Config.xml and Parallax.MotorIrBumper.Config.xml to open them. (See Figure 10.)

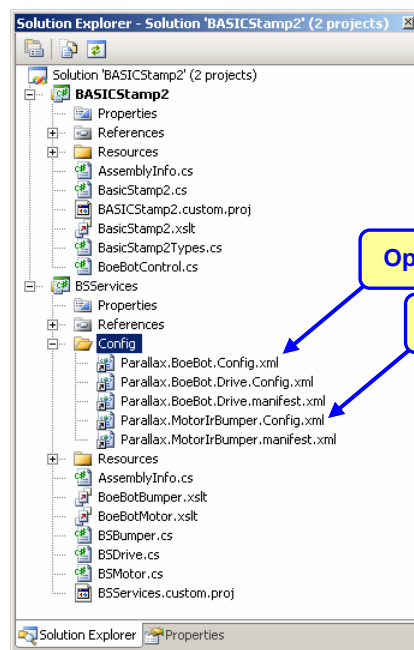
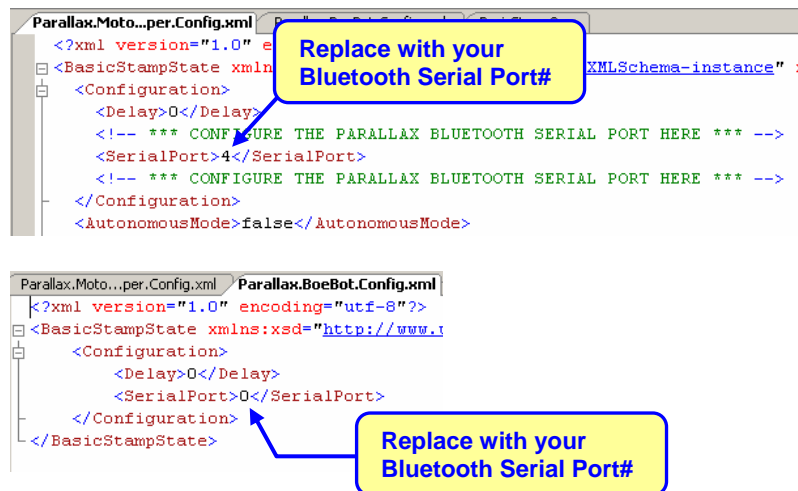


Figure 10: Opening the configuration files

- ✓ Update the values between `<SerialPort>` and `</SerialPort>` shown in Figure 11 with your Bluetooth COM port number. Make sure to use the same value you used in BASICStamp2.cs, and update the value in both files.

Figure 11: More Serial Port Settings



- ✓ 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.)

MSRS Bug Fixes

Microsoft made some changes to the Boe-Bot services for MSRS (1.5), but did not update references to an older MSRS (1.0) contract in some of the manifests. Follow these instructions to find and fix the bugs in the MSRS BASICStamp2 solution.

- ✓ Press CTRL + H to open the Find and Replace window shown in Figure 12.
- ✓ Enter this reference to the old MSRS (1.0) contract into the Find what field: <http://schemas.microsoft.com/robotics/2006/06/basicstamp2.html>
- ✓ Enter this reference to the new contract into the Replace with field (the 6 in 2006 changes to 7): <http://schemas.microsoft.com/robotics/2007/06/basicstamp2.html>
- ✓ Make sure to specify the Entire Solution in the Look in field.
- ✓ Make sure there are no extra spaces the html in either reference.
- ✓ Click Replace all.
- ✓ Verify that 7 occurrences were replaced.
- ✓ Click Save All.
- ✓ Rebuild the solution. (Right-click Solution 'BASICStamp2' (2 projects), and select Rebuild Solution.)

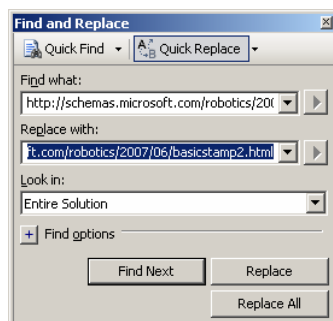


Figure 12: Find and replace all instances of text in the solution

First MSRS Example

Control the Boe-Bot Robot with the MSRS Simple Dashboard Service

This first example demonstrates using the Simple Dashboard service to remotely control the Boe-Bot robot like you might with a game controller thumbstick.

- √ 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:

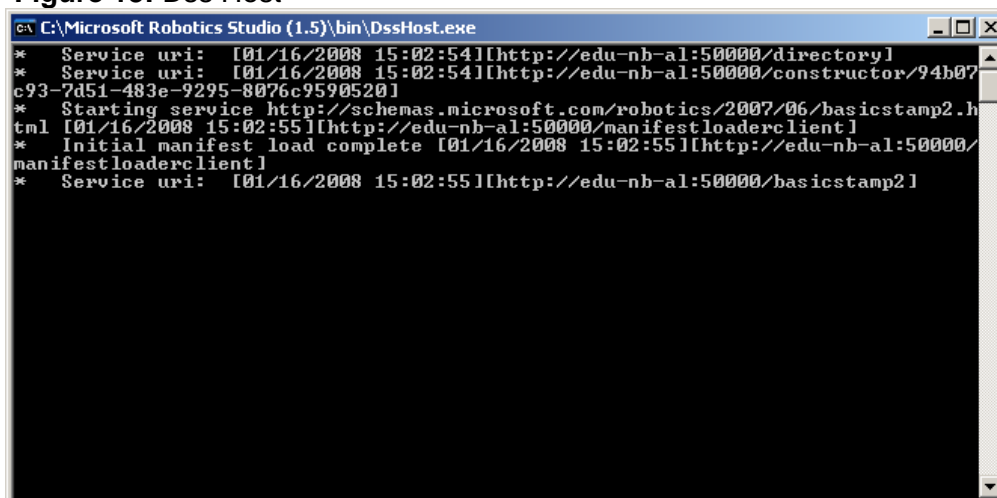
```
bool _autonMode = false;
```

- √ Click the Save all button.
- √ If you had to change the `_autonMode` declaration, rebuild the 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 BoeBotControlForMrs.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.

During the first few seconds after starting the Visual Studio debugging session, three things should happen:

- (1) The MSRS DSS Host application similar to the one in Figure 13 should appear.
- (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).

Figure 13: Dss Host



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.

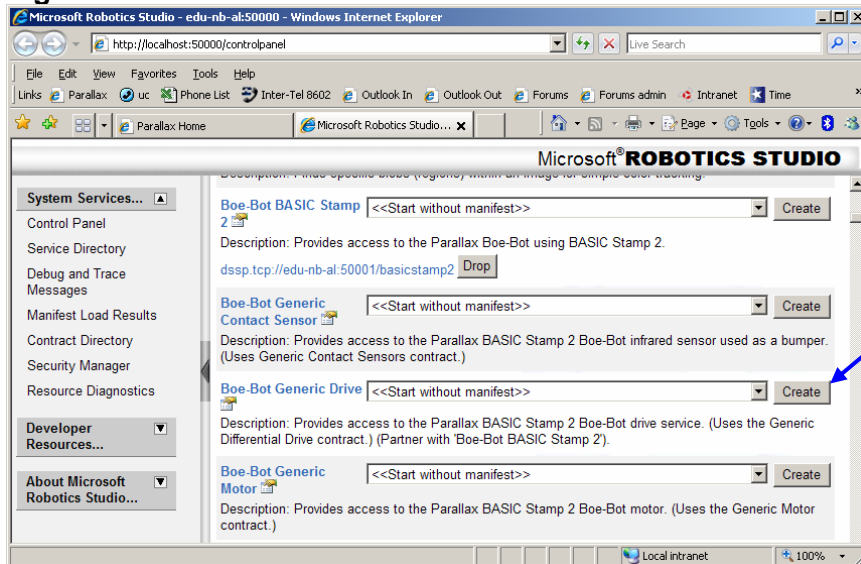
- ✓ Launch Microsoft Internet Explorer and enter this into the Address field:

<http://localhost:50000/controlpanel>

A Microsoft Robotics Studio Control Panel should appear.

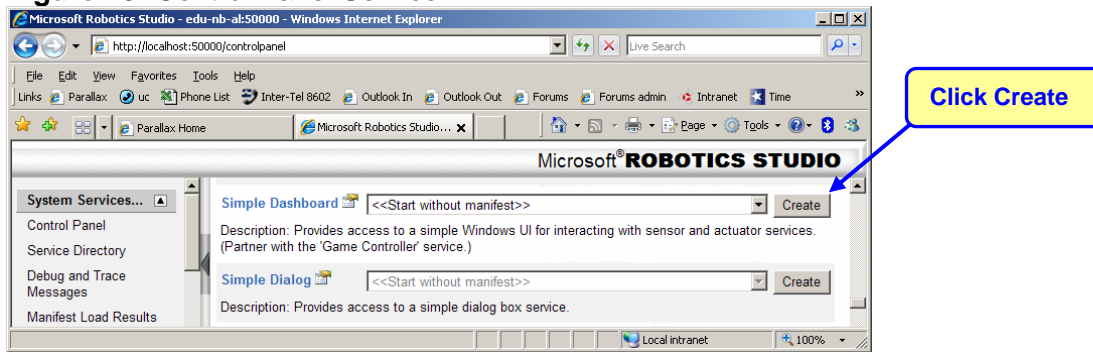
- ✓ Scroll down and find the Boe-Bot Generic Drive service shown in Figure 14.
- ✓ Click the Boe-Bot Generic Drive entry's Create button.

Figure 14: Control Panel Service



- ✓ Scroll down and find the Simple Dashboard service shown in Figure 15.
- ✓ Click the Simple Dashboard entry's Create button.

Figure 15: Control Panel Service



The Dashboard user interface (upper half shown in Figure 16) should appear. If it doesn't, 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 /bsdrive/drive service will be listed as running.

- ✓ Double-click /bsdrive/drive.
- ✓ Click the Drive button.
- ✓ Click and drag the round eyeball control's crosshairs (See Figure 16) 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.

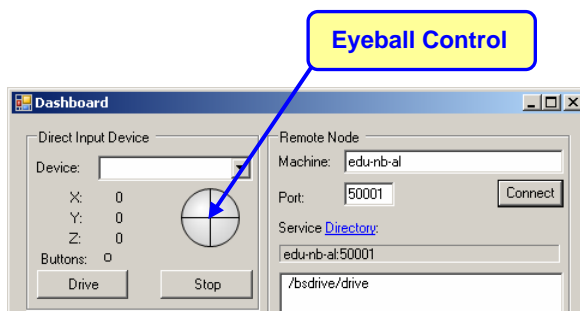


Figure 16
Dashboard Service
User Interface

Semiautonomous Navigation

This next example uses infrared (non contact) and whisker (contact) object detection circuits to gather information that MSRS can use to guide the Boe-Bot Robot through a field of obstacles. The example highlights how the Boe-Bot and Microsoft Robotics Studio can share different levels of robotic control and decision-making. In the first part of this example, the Boe-Bot communicates all its infrared object detection observations to MSRS, which makes the navigation decisions based on this information and directs the Boe-Bot's motion accordingly.

In the second part of this example, the BoeBotControl.cs file gets modified so that, in the event of a whisker contact, the Boe-Bot takes control, halting forward motion immediately and notifying MSRS of the event. This preemptive Boe-Bot response prevents it from overrunning the object during the fraction of a second it takes for the notification to be sent and for the MSRS reply to come back over the Bluetooth serial connection. Next, the Boe-Bot waits until it receives a message from MSRS instructing it which evasive maneuver to perform. After the Boe-Bot completes its maneuver, it resumes taking instructions from MSRS. Although it's difficult to tell that the Boe-Bot halted motion and waited for a reply, the difference is can be seen by comparing it to the fully MSRS controlled version of the same events.

The infrared object and whisker contact detection circuits have to be built and tested on the Board of Education platform's breadboard in a manner similar to the LED and piezospeaker indicator circuits. Recall that the indicator circuits were built and tested with a PBASIC program executed by the BASIC Stamp to make sure they were wired correctly before controlling them with MSRS. The same procedure should be followed here to detect and correct wiring mistakes, especially since the circuits are more complex. Reason being, if wiring mistakes are left undetected and uncorrected, the system might either fail to report objects that are in its path or in other cases report objects that are not actually there. Either way, the misinformation can lead to erratic behavior that can be difficult to trouble shoot. So, follow these step-by-step circuit building and testing instructions carefully to ensure that the infrared and whisker object detection circuits that are working correctly.

For the full story of what's happening with the infrared and whisker object detection circuits and code, try chapters 5, 7, and 8 in *Robotics with the Boe-Bot* before continuing here. While going through these chapters, the eb500 should be disconnected from the Board of Education's AppMod Header. When you resume from here, make sure to disconnect any circuits from P5 and P1, and reload the BoeBotControlForMsrs.bs2 "driver".

The infrared LEDs are the clear LEDs in your kit.

- √ Insert the infrared LED into the shield assembly as shown in Figure 17.
- √ Make sure the LED snaps into the larger part of the housing.
- √ Snap the smaller part of the housing over the LED case and onto the larger part.

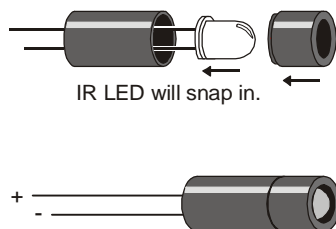


Figure 17
Snapping the IR LED
into the Shield
Assembly

- √ Build the circuit shown in the Figure 18 schematic with the Figure 19 wiring diagram as a guide.

TIP: Use your PDF reader to zoom in on the breadboard circuit in Figure 19 for a close-up view of the correctly wired circuit.

Figure 18: Boe-Bot Circuit with Infrared and Whisker Object Detectors

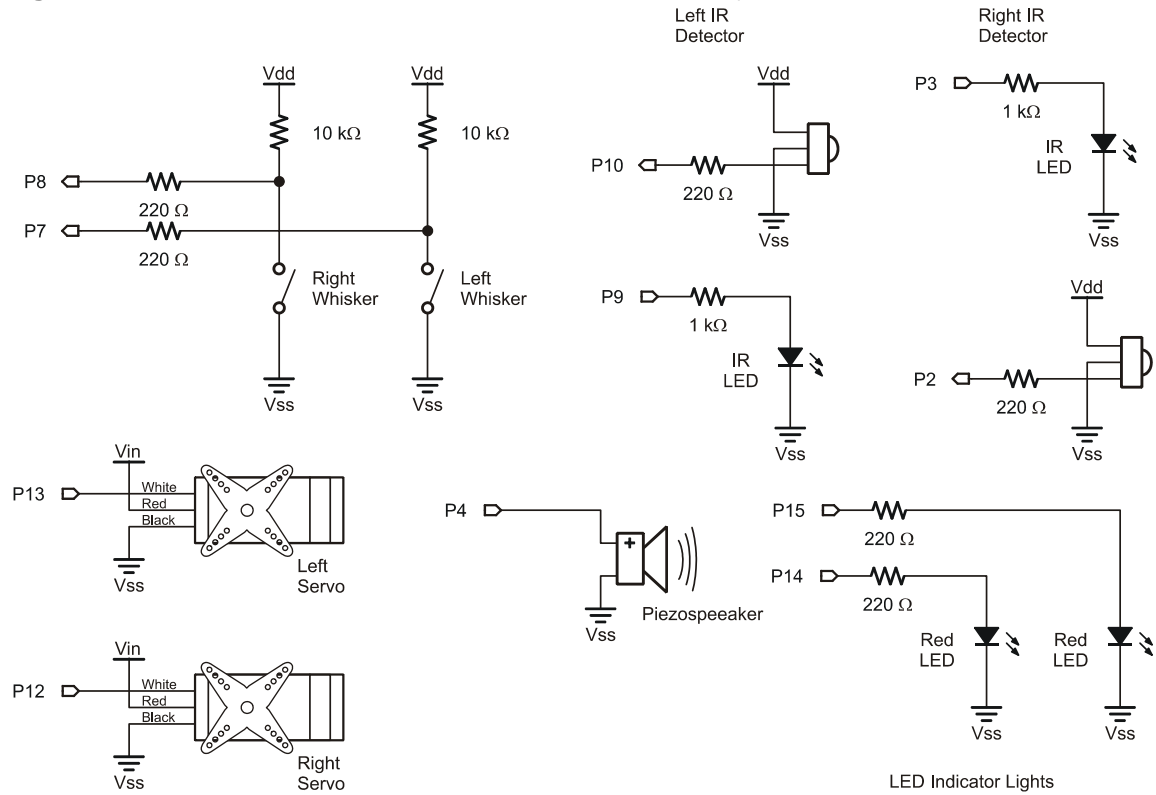
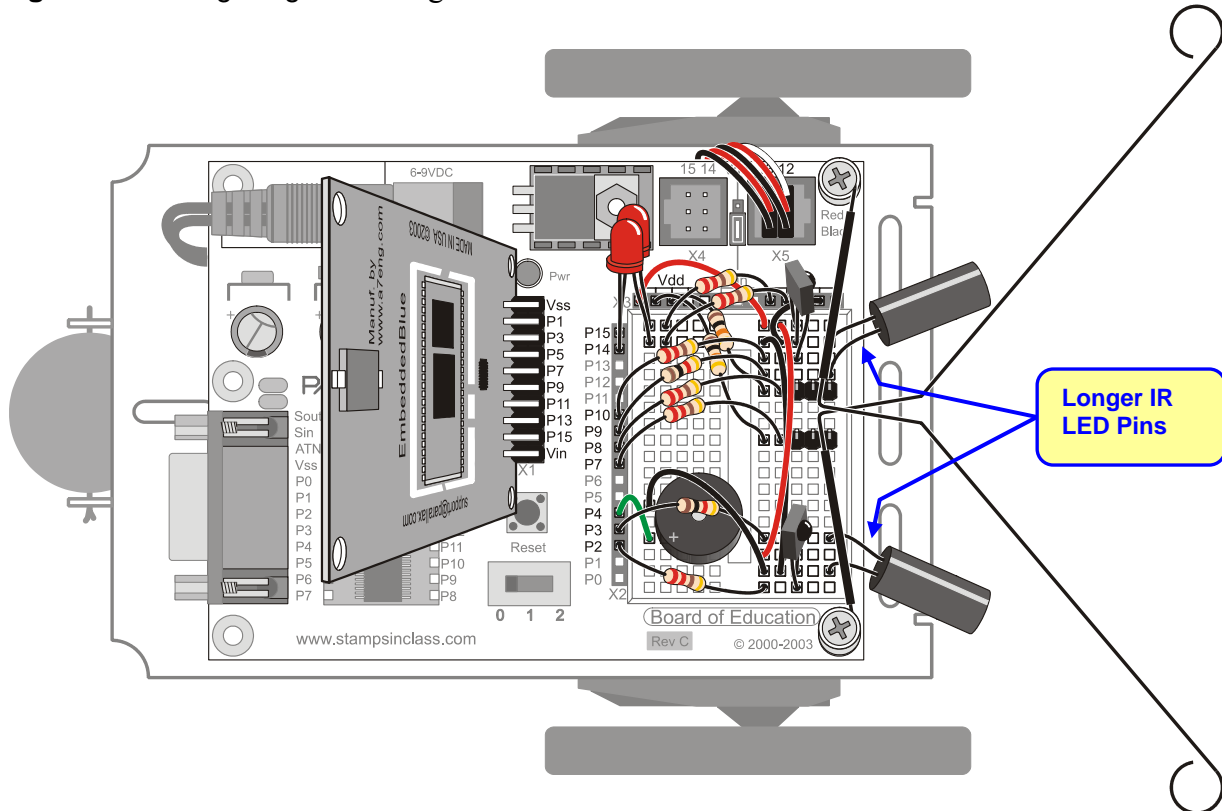
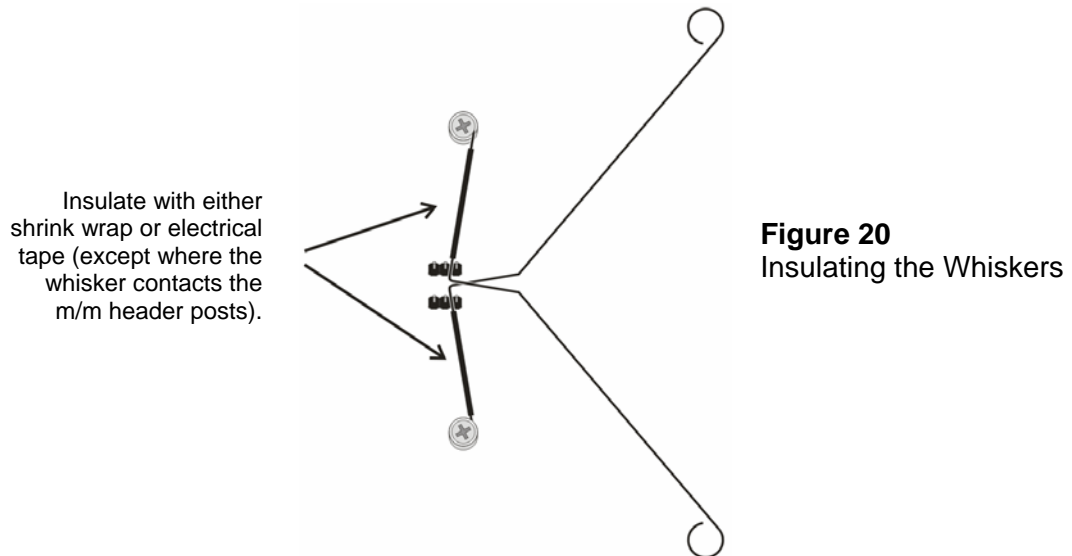


Figure 19: Wiring Diagram for Figure 18




- √ Insulate the whiskers with shrink-wrap tubing or electrical tape as shown in Figure 20, so it will not make contact with the infrared LED leads.
- √ Make sure the metal part of each whisker is resting between but not touching the front two pins of the male-male headers. The whisker wire should not touch either of the header posts until the Boe-Bot bumps into an object.



Before moving on to MSRS navigation with the IR and whisker object detectors, test them to make sure they are free of wiring errors and working properly. Here's how:

- √ Open TestWhiskersAndIr.bs2 in the BASIC Stamp Editor and download it to the BASIC Stamp.
- √ Verify that the output of each detector with no obstacles is 1.
- √ Verify that the output of each detector with an obstacle in front of it is 0.

	<p>IR Detector Trouble Shooting: If the Debug Terminal did not correctly display object detection status, it means there's probably a mistake in the wiring. One of the most common mistakes is connecting an IR LED's cathode terminal where its anode terminal should have been connected and visa versa. To rule this out, verify that the "longer pins" indicated in Figure 19 are plugged into the correct rows shown in the wiring diagram.</p>
---	--

- √ 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.

	<p>In most cases, the IR detectors will also detect your hand as you reach to depress the whisker.</p>
---	---

If the IR and whisker object detectors both correctly detected object presence/absence, you are now ready to make MSRS direct the Boe-Bot's navigation based on its object detection sensor reports.

- √ Repeat the procedure in the MSRS Driver Code for the Boe-Bot Robot section that starts on page 9 to reload BoeBotControlForMrs.bs2 into the BASIC Stamp.
- √ In Microsoft Visual Studio, change the `bool _autonomousMode` declaration in the BoeBotControl.cs file so that it initializes to `true` instead of `false`:

```
bool _autonomousMode = true;
```

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

The Boe-Bot should now roam under full MSRS control avoiding objects it detects with infrared.


Next, let's add whisker sensing. When the Boe-Bot makes contact with the whiskers, it stops immediately and then tells MSRS what happened and waits. It will receive a reply with an evasive action maneuver from MSRS in less than a second and start executing the maneuver.

- √ 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.
- √ Make sure the 3-position switch on the Board of Education is in position-2.
- √ Click the Debug button (or press F5).

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

	<p>If the Boe-Bot changes direction for no apparent reason, try the following:</p> <ul style="list-style-type: none"> √ Make sure the IR LED's are directed level to the ground or even a few degrees above the horizontal so that the floor cannot reflect infrared that can be misinterpreted as an object. √ Remove it from direct sunlight by covering any nearby windows. √ Some fluorescent light fixture ballasts may also create infrared interference. See <i>Robotics with the Boe-Bot</i>, Chapter 7, Activity #2. √ 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 objects 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 whisker(s) 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 _autonomousMode = 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 code:

Robotics Tutorial 1

This tutorial's example code displays whether or not the Boe-Bot's left IR detector detects an object in a DssHost.exe window.

- √ Use the Microsoft Robotics Studio Command Prompt to open this file into the Visual Studio editor:

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

- √ In the Solution Explorer, click the + next to the Config folder.
- √ In the Solution Explorer, double-click Robotics Tutorial1.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 21, except the command line arguments field may be empty or possibly contain a manifest for a different robot.
- √ 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"
```

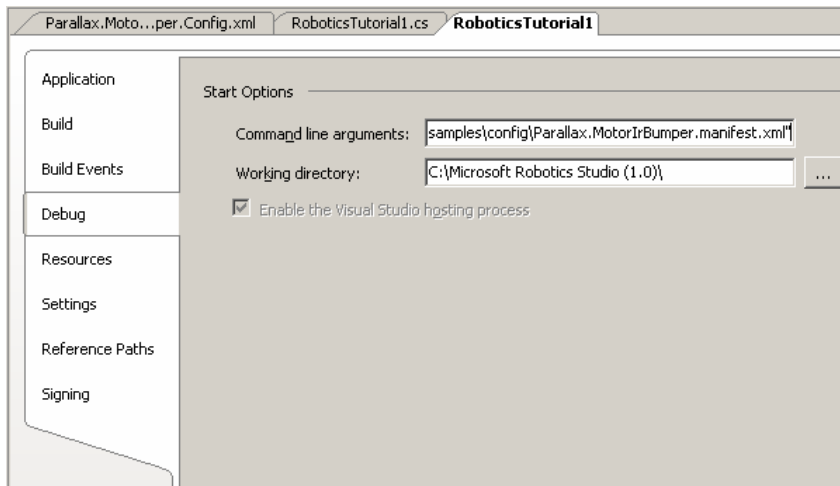


Figure 21
RoboticsTutorial1
Properties Debug Tab

- ✓ Click Save All.
- ✓ Click Build → Build Solution (F6).
- ✓ Make sure to point the Boe-Bot in a direction so that its IR detectors will not detect a nearby object. You may also want to make sure any nearby windows are covered to eliminate interference from direct sunlight.
- ✓ Since this activity does not involve motor control, your Boe-Bot robot’s 3-position switch can be in position-1, which cuts power to the servo motors while still supplying power to the rest of the Board of Education.
- ✓ In Visual Studio, click the Start Debugging (F5) button.

The DssHost.exe window shown in Figure 22 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 left IR detector, it will display a new “Object detected!” message.

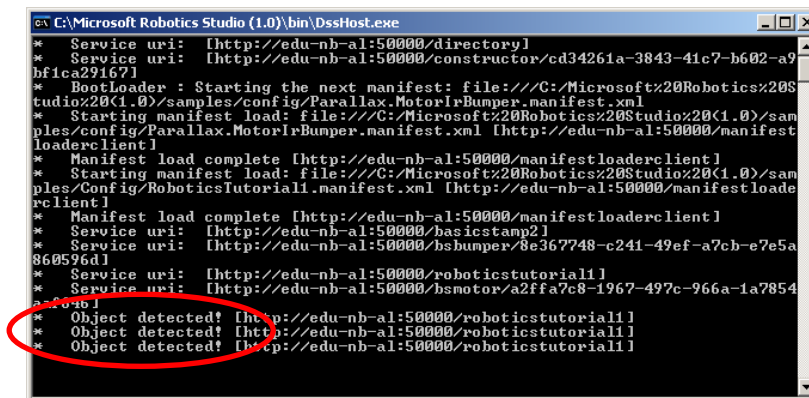


Figure 22
Object Detected Messages in
the DssHost.exe Window

Robotics Tutorial 2

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

- 1) The tutorialName.csproj project file is opened with the Microsoft Robotics Studio Command prompt.
- 2) The Command line argument field in the project Properties' Debug tab is updated with a file path to the appropriate Parallax hardware manifest(s).
- 3) 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 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 21.
- √ 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"
```

3) 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 23 will appear again. Each time you wave your hand in front of the Boe-Bot's left 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/directory]
* Service uri: [http://edu-nb-al:50000/constructor/17f161a3-c1b6-4174-b4d5-8345cb91b09e]
* 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/manifestloaderclient]
* Manifest load complete [http://edu-nb-al:50000/manifestloaderclient]
* Starting manifest load: file:///C:/Microsoft%20Robotics%20Studio%20(1.0)/samples/Config/RoboticsTutorial2.manifest.xml [http://edu-nb-al:50000/manifestloaderclient]
* Manifest load complete [http://edu-nb-al:50000/manifestloaderclient]
* Service uri: [http://edu-nb-al:50000/basicstamp2]
* Service uri: [http://edu-nb-al:50000/bsmotor/2116500e-d8c7-4bbe-9283-af86afe07880]
* Service uri: [http://edu-nb-al:50000/bsbumper/f69526ae-0085-4539-805e-d8d0f965657]
* Service uri: [http://edu-nb-al:50000/roboticstutorial2]
* Motor On [http://edu-nb-al:50000/roboticstutorial2]
* Motor Off [http://edu-nb-al:50000/roboticstutorial2]
* Motor On [http://edu-nb-al:50000/roboticstutorial2]
* Motor Off [http://edu-nb-al:50000/roboticstutorial2]
* Motor On [http://edu-nb-al:50000/roboticstutorial2]
* Motor Off [http://edu-nb-al:50000/roboticstutorial2]

```

Figure 23
Motor on/off Messages in the DssHost.exe Window

Robotics Tutorial 4

The end result of this tutorial is a user interface window in Figure 24 you can use to select Boe-Bot maneuvers.

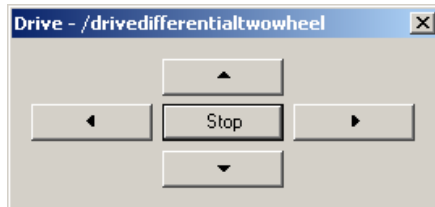



Figure 24
A Windows User Interface Example

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

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

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

- 3) 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.

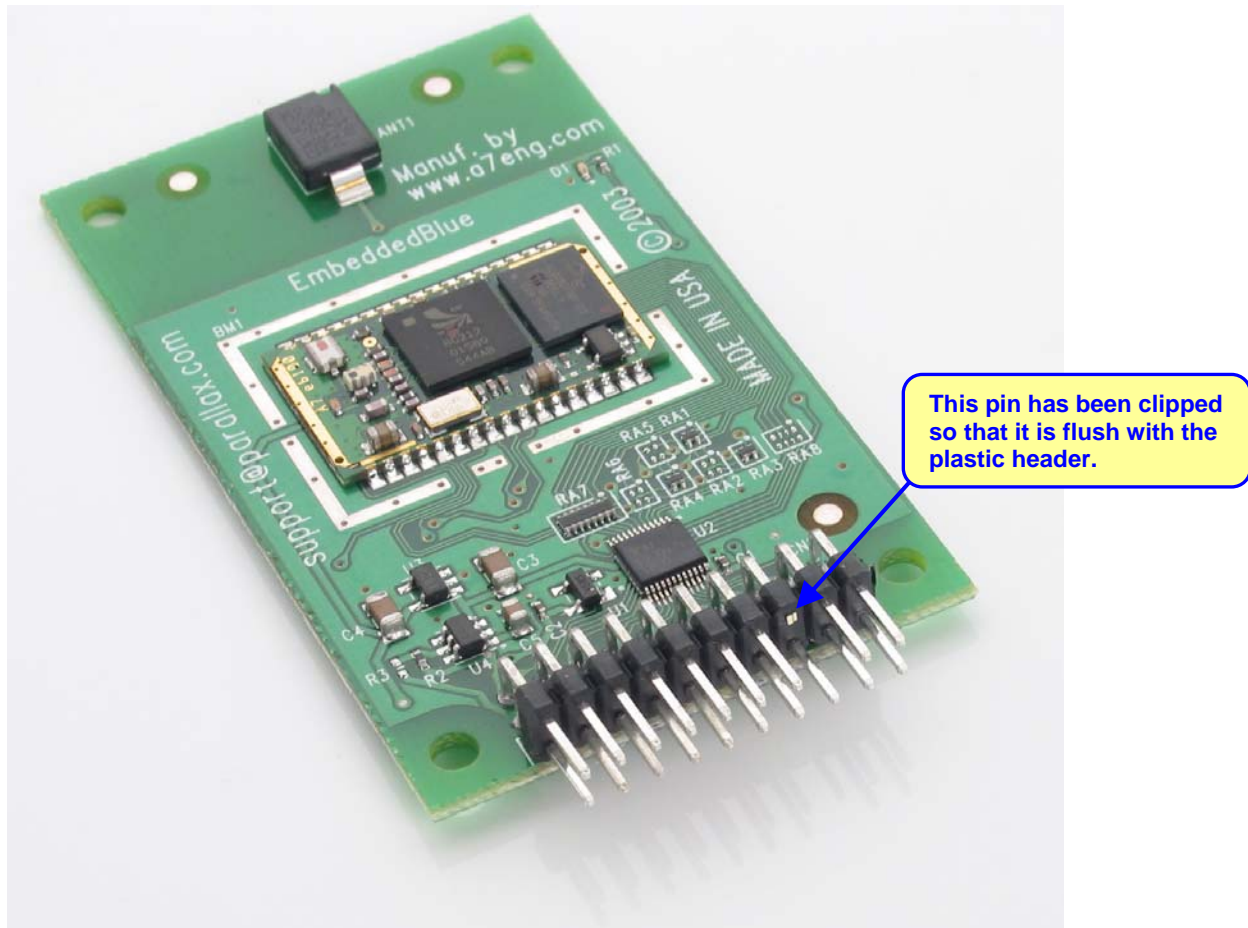
Start Byte	Message Index	Command	Data 1	Data 2
0BASIC Stamp instructs PC to repeat handshake. (The PC's version of this message is command = 192.)				
255	index	0	don't care	don't care
1 BASIC Stamp initiates handshake				
255	index	1	don't care	don't care
2 PC confirms receipt of handshake initiation				
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	xxxxLRxx	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 on page 17. 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 17 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.