Simple Bluetooth Programming for Battle Bots

Purpose: This handout will show you the process I go through with my students to create remote controlled battle bots. They are similar to sumo-bots but a require more complex programming (than Sense Line>Drive Forward> Smash) as each battle bot must be remotely controlled using Bluetooth.

Step 1: Establish Bluetooth Control

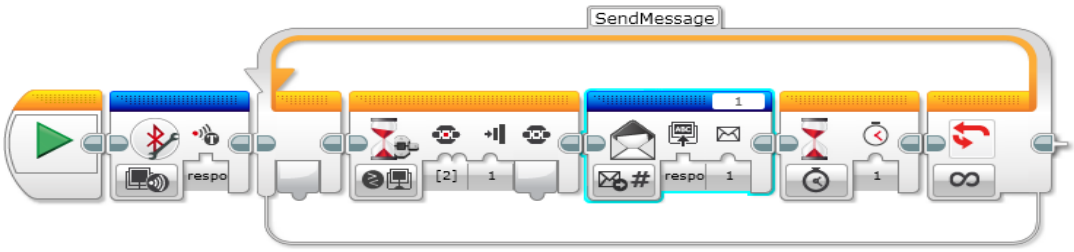
There are many tutorials available on how to do this. My favorite is a YouTube video available from *Builder Dude 35* which can be found at: **http://bit.ly/27VORmT**

If you're unsure how to do this, I have prepared a separate handout titled **Establishing a Bluetooth Connection Between two EV3 Robots.**

Step 2: Practice Simple Connection

To begin with I show my students how to write a program that has the controller bot send a signal upon push of a button to the responder robot and has the responder robot play a sound when it receives a signal. This is a nice first step because there is no building involved. For the sake of simplicity I have named my robots "controller" and "responder."

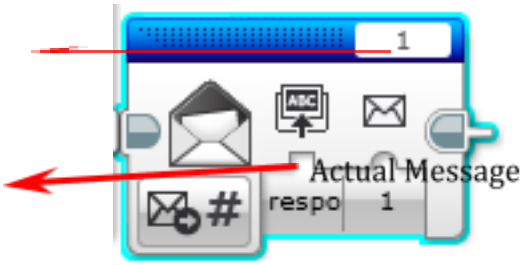
Here is the program for the controller bot:



In this program I did the following:

1) Initiated Bluetooth connection between the controller and responder robot. (They had already been connected earlier.)
2) Created a loop so I could send the command over and over again.
3) Waited for a push of the center button on the EV3.
4) Sent the numeric message “1” to the responder robot using message title “1.”
5) Then I waited a second and repeated.

Let’s take a closer look at the Send Message icon.

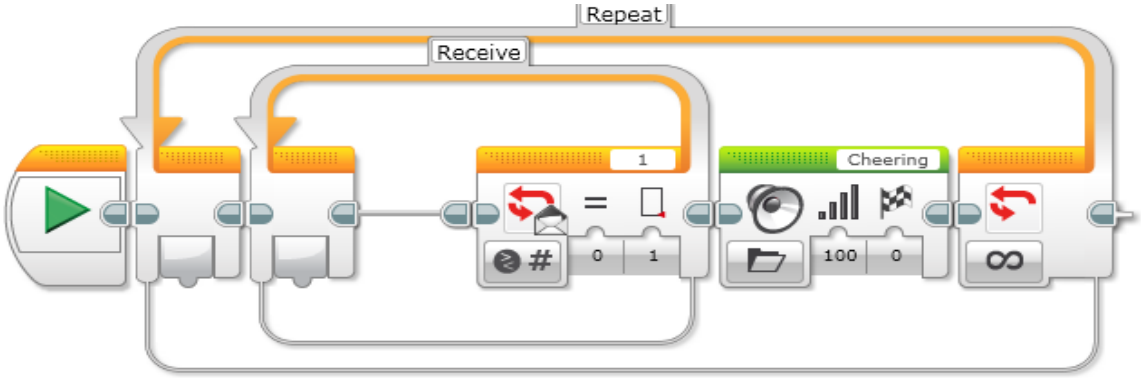


Title of Message

Click here to choose Send/Receive a numeric, text, or logic message

Name of Robot Message is Being Sent To

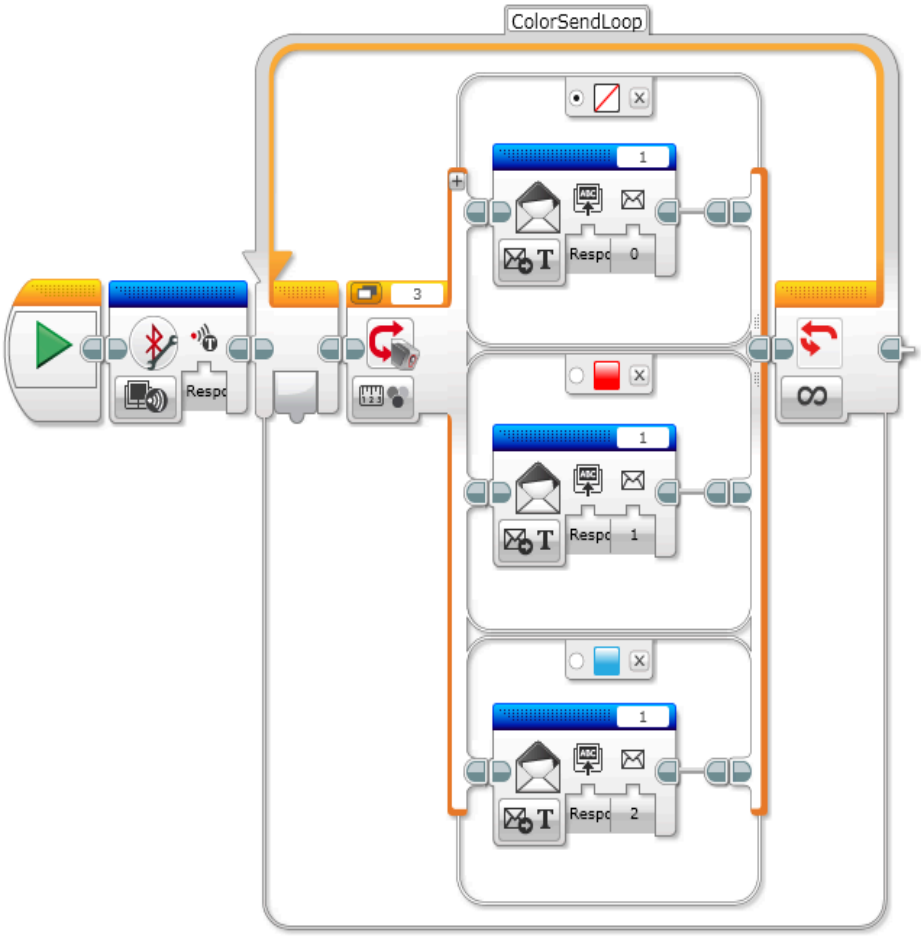
Here’s the program for the responding robot:



In this program I created two loops. The first waits to receive the message “1” in a message titled “1.” (We will see later why using message titles is important.) When it receives the message “1” it ends the first loop and plays the “Cheering” sound. Then it loops to the beginning and waits for the message again.

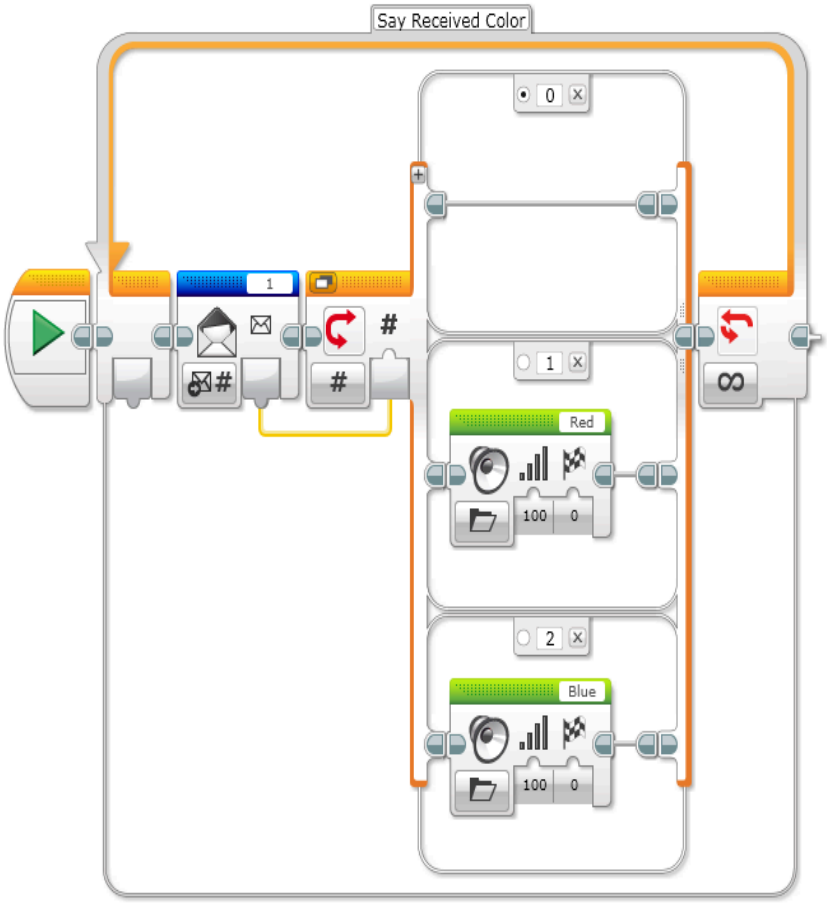
Step 3: Practice a Simple Switch to send Different Messages

In this program, I have again initiated a connection. Then I placed a color switch inside a forever loop. The color switch is measuring color. If it measures no color (the default case) it sends a numeric message of “0.” If it measures red it sends a “1” and if it measures blue it sends a “2.” You could expand to more colors by adding another case in your switch. By having the default “no color” option, I help ensure that my responding robot doesn’t say “red” or “blue” when nothing is seen. I like sending numeric messages, but you could easily send text message containing the name of the color instead.



Here is the program for the responding program:

In this program I have placed a switch inside a loop. In front of the switch is a Message icon set to receive a numeric message. The received message is wired into a number switch. If a "0" is received (indicating no color) then nothing happens and the loop repeats. If a "1" is received the robot says "Red" and if a "2" is received the robot says "Blue." I made "0" the default, to make sure the robot says nothing if no color is sensed.



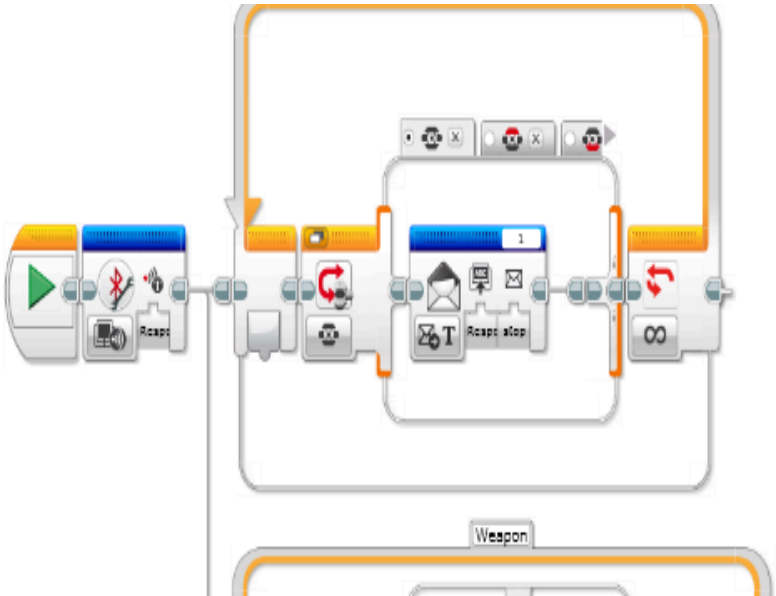
Step 4: Programming a Fully Controllable Battle Bot

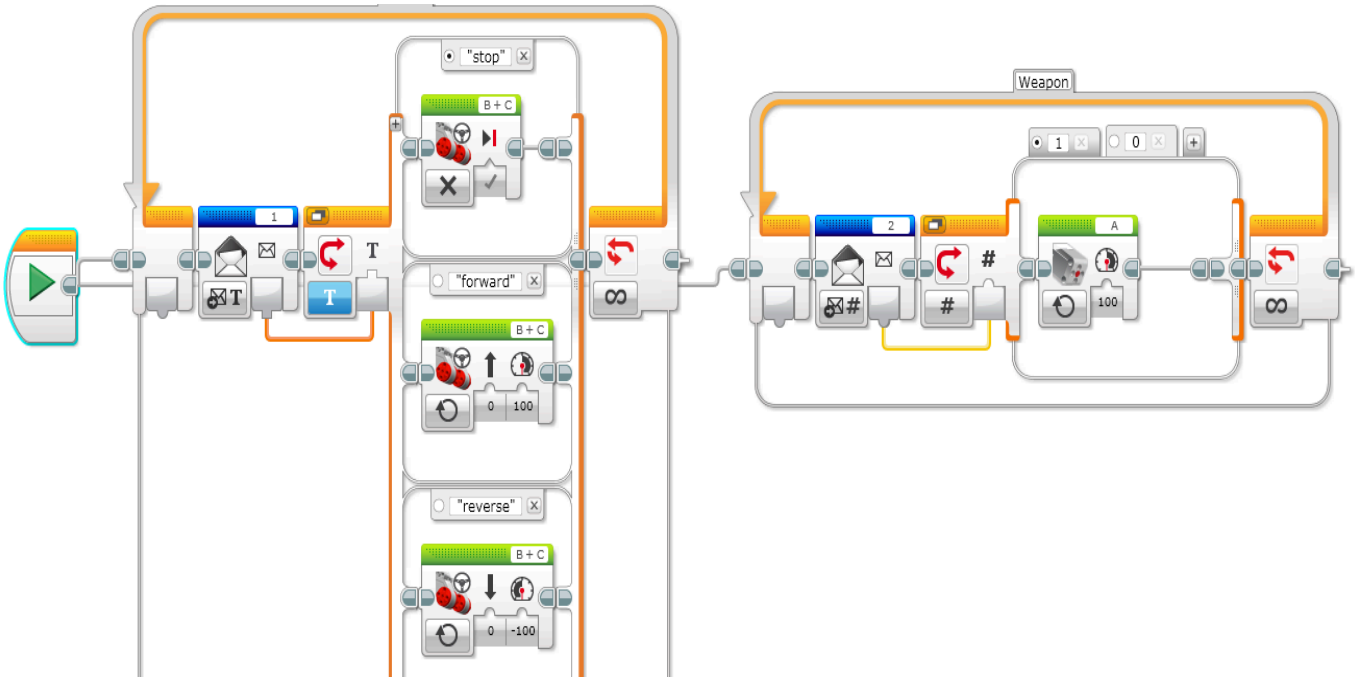
To begin with our controller bot is going to have a touch sensor attached to Port 1. The Battle Bot is going to have left and right drive motors attached to ports B and C respectively, and a medium motor wielding a weapon of amazingness attached to port A. This is the most common setup my students use with one EV3 kit; you can of course design any controller/battler combination you like once you learn the basics. The programs below are written for the robots I just described.

There are two loops with switches in this first program. This is because the we want to be able to run the “weapon” connected to port A separately from the drive motors.

The top loop is in “tabbed” view. This means that all of the cases are arranged in tabs, rather than vertically – this takes up less space. After initiating Bluetooth connection, a Brick Button switch is placed inside the loop. The default is to send the word “stop” if no button is pushed. This is the simplest way I have found for the Battle Bot to stop when nothing is happening, If the top button is pushed, the text message “forward” is sent and so on with “reverse,” “left,” and “right” for the corresponding buttons.

For this loop utilizing the touch sensor please notice that the title of the message has been set to “2.” This allows clear differentiation between the driving commands and the weapon commands. If the touch sensor is pushed in the numeric message “0” is sent and if it is not pushed in, the numeric message “0” is sent.





This is the responding program. I have left the first switch out of tabbed view so you can see at least three of the options in the switch. Text received via Bluetooth is wired into a numeric switch. The default is to turn motors off if the word stop is received. Otherwise I have the forward and reverse commands, and cut off are similar commands to turn left or right. Also wired from the start icon (not wired following the first infinite loop – it would never run that way) is a second loop with a switch. This one is waiting for a numeric message in message title "2." If it's "1" it turns the "A" motor on, if it's a "0" (default) it turns motor A off.

Extensions –

This is only one way of many to program a remote controlled ‘bot with Bluetooth. You could use Tank Move for steering or add more conditions or another touch sensor to allow you to control steering and driving separately so you can turn forward or backward to the right or left.

You could also use the rotation sensor built into the EV3 motor on a controller bot to vary the speed of the Battle Bot.