

6

experimenting with the EV3 infrared components

In this chapter, you'll learn about the Remote Infrared (IR) Beacon and the Infrared (IR) Sensor. In addition to measuring the proximity of objects, the IR Sensor can detect infrared signals from the Remote IR Beacon, allowing you to send commands to your robot just as you send commands to a television with a remote control. The IR Sensor can also estimate its distance and orientation with respect to the Remote IR Beacon; this cool feature will allow you to do fun things with your robots like play tag, chase prey, or locate and reach a mission base.

remote IR beacon

The Remote IR Beacon is powered by two AAA batteries. It has holes on each side, which make it easy to incorporate into a LEGO Technic model. As you can see in Figure 6-1, it has four small buttons (labeled 1, 2, 3, and 4), a large button (9), and a red Channel Selector switch (12). The Channel Selector lets you choose among four channels, so you can use up to four Remote IR Beacons at once. A number engraved in red plastic in the small circular window shows the current channel. (As long as each remote is set to a different channel, its signal won't interfere with other remotes' signals.)

The large button (9) turns on Beacon Mode. When in Beacon Mode, the device transmits a continuous signal until any button is pressed or until one hour has elapsed. The IR Sensor can estimate the proximity and the heading to a beacon set in Beacon Mode. This feature allows a robot to follow a moving beacon or find its distance and heading relative to a fixed beacon.

The four small buttons (1, 2, 3, and 4) send commands (the numbers in the following list) to the IR Sensor using two infrared light-emitting diodes (LEDs) at the front of the remote. (The plastic that houses these LEDs is a dark blue filter that lets only infrared light pass through it.)

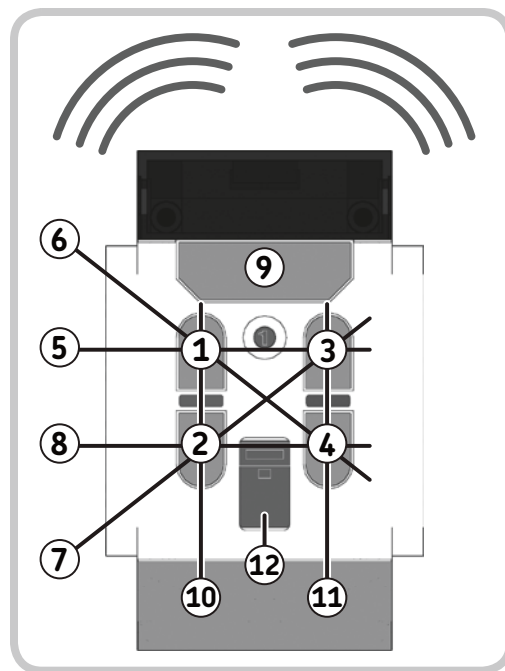


Figure 6-1: The Remote IR Beacon

- 0 No button is pressed and Beacon Mode is off.
- 1 Button 1
- 2 Button 2
- 3 Button 3
- 4 Button 4
- 5 Buttons 1 and 3
- 6 Buttons 1 and 4
- 7 Buttons 2 and 3
- 8 Buttons 2 and 4
- 9 Beacon Mode is on.
- 10 Buttons 1 and 2
- 11 Buttons 3 and 4

using the remote IR beacon as a remote

Let's see how we can use the Remote IR Beacon as a simple remote control for your robot using only the IR Control App (and no programming—yet). In the EV3 Brick menu, go to the **Apps Tab** (third from the left) and open the **IR Control App**. You should see a screen like Figure 6-2(a).

There are two modes to choose from, as shown in Figure 6-2. In the mode shown in Figure 6-2(a), you can control the motors using Remote IR Beacon channels 1 and 2; in the mode shown in Figure 6-2(b), you can control the motors using channels 3 and 4. To switch between modes, press the **Enter** button on the EV3 Brick.

In the first mode, with the Remote Channel Selector (labeled 12 in Figure 6-1) on channel 1, you can control a motor connected to port B with buttons 1 (forward) and 2 (backward), and you can control a motor connected to port C with buttons 3 (forward) and 4 (backward). While in the same mode, you can control motors connected to ports A and D with another remote on channel 2. The second mode works similarly but receives commands from remotes set on channels 3 or 4.

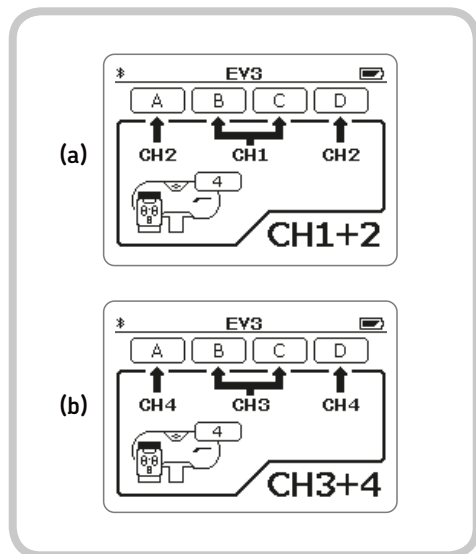


Figure 6-2: The IR Control App. To switch between controlling channels 1 and 2 (a) and channels 3 and 4 (b), press the **Enter** button on the EV3 Brick.

The IR Control App makes it easy to remotely control a wheeled robot like ROV3R. You'll also find it useful when building and testing a motor-powered mechanism, since you can test the mechanism by turning the motor forward and backward without having to build a test program.

To test out the remote control, build ROV3R in any of the versions from Chapter 2 and connect the motors to ports B and C. Now start the IR Control App on the EV3 Brick, take the Remote IR Beacon, and select channel 1. You should be able to use the small buttons on the remote to drive ROV3R around. Table 6-1 shows how you would control ROV3R.

table 6-1: controls for a differential drive robot such as ROV3R

Buttons pressed	Motion
1 & 3	Drive forward.
2 & 4	Drive backward.
1 & 4	Spin right.
2 & 3	Spin left.
1	Turn right by pivoting on the right wheel.
2	Turn left by going backward and pivoting on the right wheel.
3	Turn left by pivoting on the left wheel.
4	Turn right by going backward and pivoting on the left wheel.

NOTE If the controls aren't working at first, make sure that the IR Sensor is connected to port 4 and that the IR Control App is in the right mode, receiving commands from channel 1 (or channel 2, if you connected the motors to ports A and D).

See? Now you can use the Remote IR Beacon as a remote control for ROV3R, without having to create a program for it! This same setup works for real-world vehicles, such as tanks or tracked vehicles like excavators. The human driver controls these vehicles by moving two levers, and each lever controls the motor that drives the track on the corresponding side. This is just like pressing the Remote IR Beacon's buttons 1 and 2 or 3 and 4. With the Remote IR Beacon and some extra programming, you can also control vehicles with different steering, as you'll see in Chapter 12 with the SUP3R CAR.

using sensor blocks and data wires

A robot uses the data provided by its sensors to perceive the world around itself. In Chapters 3, 4, and 5, we compared sensor readings against thresholds to trigger Wait blocks or Switch blocks. To directly access sensor readings, we can use the Sensor blocks (found in the Programming Palette with the yellow tab).

Each Sensor block has several modes that serve different functions. In Measure mode, Sensor blocks provide measurements as numeric values to other blocks. In Compare mode, they compare measured values against a threshold to provide logic values (see “Understanding Data Types” on page 89 for a discussion of these). Some Sensor blocks (like the Motor Rotation block and the Timer block) also have a Reset mode, which resets their measured values to 0.

The IR Sensor block has a Proximity mode (which measures the distance to the nearest object), a Remote mode (which receives commands from the Remote IR Beacon), and a Beacon mode (which estimates the robot’s proximity and heading in relation to the Remote IR Beacon).

Let’s get some data from the IR Sensor block. Build ROV3R with Front IR Sensor (page 31), and create the program shown in Figure 6-3 by adding blocks as described in Chapter 5. The IR Sensor block is set in **Measure Proximity** mode, while the Move Steering block is in **On** mode. The key idea of this simple program is to take the Proximity value provided by the IR Sensor block and send it to the Move block for use as a Steering input. ROV3R should steer according to the distance the sensor measures to an object. This simple program makes ROV3R spin in place until you place your hand in front of the IR Sensor; then it will follow your hand, going straight until you remove your hand.

To send the sensor output to the Steering input, we need to set up a Data Wire. (Data Wires carry values from one block to another.) To create the Data Wire, use your mouse to click

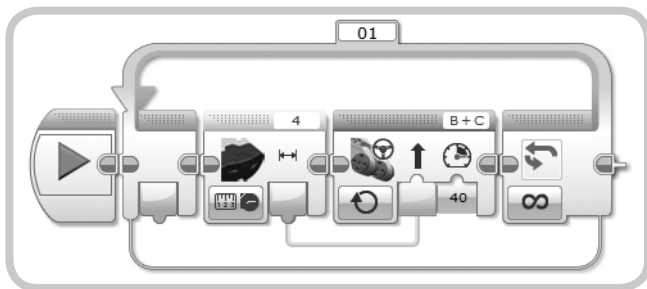


Figure 6-3: This program makes the robot steer according to the distance measured by the IR Sensor.

and hold the **Sensor block output** and drag the wire—left to right—to the **Steering input** (Figure 6-4). When you place the mouse cursor on an output, its shape should change into a wire spool. When you click an output, a wire plug appears. When you drag the wire near a block’s inputs, all inputs that can accept that type of data should be highlighted in blue. Place the plug on your desired input, and release the left mouse button.

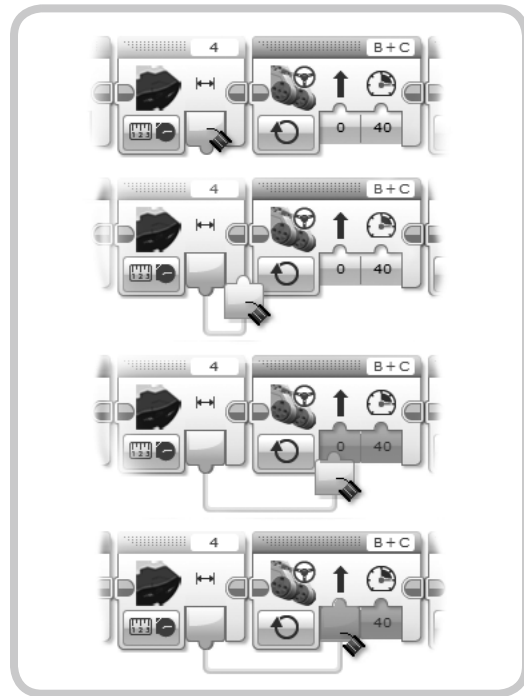


Figure 6-4: To add a Data Wire, click a block output. A plug appears on the end of the wire. Drag the plug to another block’s input. The wire automatically follows the plug on the screen.

UNTANGLING DATA WIRES

To delete a Data Wire, click and drag its end slightly away from the input (the reverse of what we did in the last two steps shown in Figure 6-4). To move a Data Wire, just click and drag it. To make the EV3 software rearrange and compact the wire (in case things are getting messy), double-click the wire.

Remember that the block that provides the output value must precede the block that receives the value in its input and that the blocks are executed in sequence from left to right: The output block where the wire begins must be to the left of the input block where it ends. However, a Data Wire can skip over many blocks and connect distant blocks.

Download and run the program in Figure 6-3. What happens? ROV3R should spin until you put your hand near the sensor. Once your hand is near, ROV3R should go toward your hand and follow it as you slowly pull it away.

How does this all work? When the IR Sensor measures a large distance (when no object is near), its Proximity output sends a high value, around 80 to 90 percent. When you place your hand in front of the sensor, its Proximity output drops closer to 0 percent. The Proximity value is carried by the Data Wire into the Steering input of the Move Steering block, which accepts values from -100 to 100 (percentage of steering). When the value is high, the robot spins; when it's low, the robot will go almost straight.

EXPERIMENT 6-1

What does the program in Figure 6-3 do if you connect the Data Wire to the Power input, setting the Steering input to 0?

EV3 software features for debugging programs

In Chapter 5, you learned that the Port View tab in the Hardware Page lets you see the values of sensors even while a program is running. But the EV3 Software allows you to do even more!

Since the Data Wires carry data, you can display the sensor's current readings on the wire. Place the mouse pointer over a Data Wire, and a small window pops up displaying the current value, as shown in Figure 6-5. The number in the pop-up window changes continuously because the blocks are inside a loop that runs forever, very quickly. Note that this feature works only if you run the program from the EV3 Software using the Controller (Figure 5-3 on page 71). It won't work if you run the program from the EV3 Brick menu, even if the Brick is connected to the EV3 Software.

Notice in Figure 6-5 that the header of each block contains diagonal stripes (they are animated in the software). These animated stripes indicate the blocks currently being executed. Both this *Execution Highlight* feature and the real-time Data Wire pop-up display really help with *debugging* programs (that is, finding and fixing errors, or *bugs* in programming jargon).

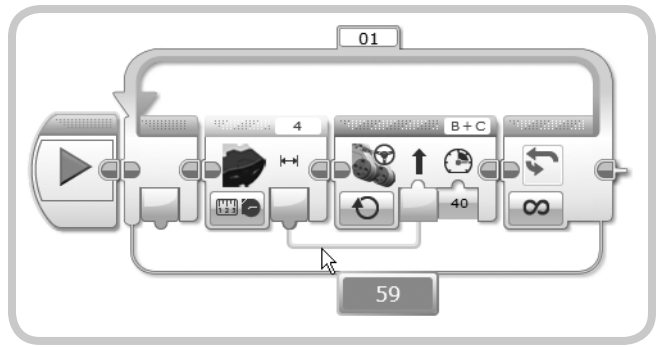


Figure 6-5: Place the mouse pointer over a Data Wire to display a pop-up window with the wire's current value. The blocks currently being executed are highlighted with animated diagonal stripes.

For example, you can use these features to see whether a Data Wire is carrying the expected values or whether the program is stuck somewhere due to a stalled Wait block.

NOTE The origin of the terms *bug* and *debugging* is curious and controversial. It's rumored that the term *bug* originated back in the 1950s, when computers were as big as walk-in closets. Back then, real bugs (moths and roaches) sometimes snuck into those huge relay-based computers, causing electrical and mechanical problems. Engineers had to literally remove the bugs to get the computer to work correctly!

displaying data nicely with the text block

Let's add some blocks to the program in Figure 6-5 that will display the IR Sensor values on the EV3 Brick screen as the robot follows a hand. Using Figure 6-6 as reference, add a Text block (Data Operations palette, red header) and a Display block to the program.

Text blocks have only one mode, Merge mode, which combines strings of text provided by its inputs **a**, **b**, and **c**. A *string* is just a bit of text with any combination of letters, numbers, spaces, or symbols: `!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~`. To enter characters into the Text block, you can either type text into one of its input fields or connect a Data Wire from another block to one of its inputs.

If we had connected the IR Sensor output directly to the Display Block Text input, the numeric values would have converted to text automatically. However, when many numeric values are displayed, what they mean is not always clear. You can use the Text block to generate more meaningful strings

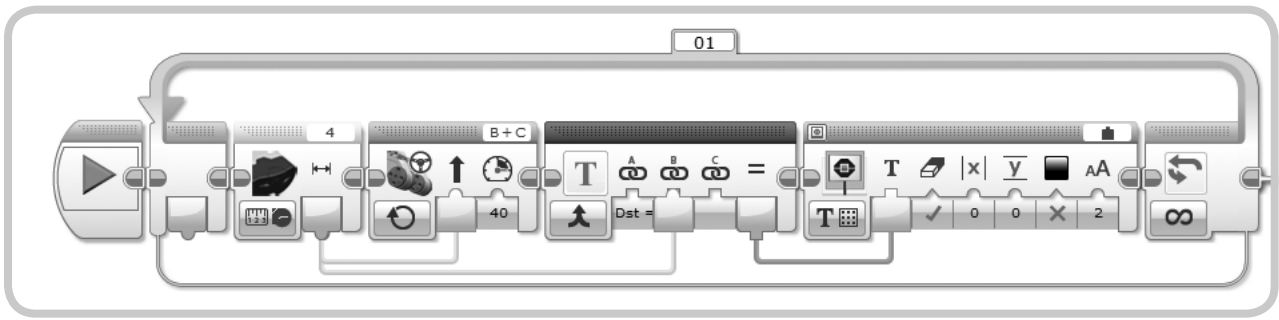


Figure 6-6: You can display meaningful messages on the EV3 Brick screen using the Text block. This program is similar to the one in Figure 6-3.

by wrapping text around numeric values. For example, you could create a string like *Proximity is 20%*, where the number 20 comes from a Data Wire, or the Text block could report *Distance = 40*, where 40 is a sensor reading.

Let's give this a try. Enter **Dst =** (with a space after the equal sign) into **Text field A**. Set the Display block to **Text Grid** mode. Then select **Wired** instead of static text by clicking the **Text field** in the header (as shown in Figure 6-7). The Display block should show a new Text input, where you will provide the variable text data to be displayed. Set the Clear Screen input to **True** (as indicated by the check mark under the eraser icon) so that the block will clear the screen every time it is executed.

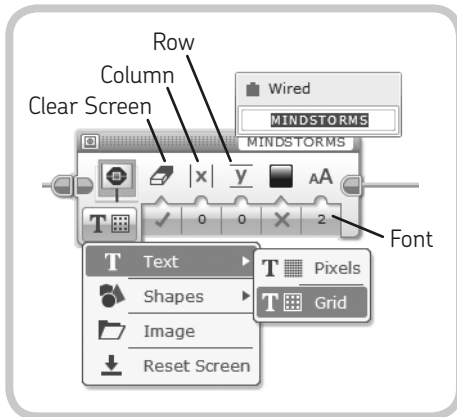


Figure 6-7: Configure the Display block to show text on a grid, with the text input coming from a wired input.

Text Grid mode allows you to display the text aligned to a grid of rows (Y input) and columns (X input). The dimension of a cell grid is one character (Normal font = 0; Bold font = 1). A character set in Large font (2) is two rows and two columns wide.

Now drag a new Data Wire from the IR Sensor output to the Text block's second input, and drag another Data Wire from the Text block's output to the Text input of the Display block. (You'll now have two Data Wires coming from one output.) Download and run the program. The robot's behavior

should be the same as before, but onscreen you should see a nice big report of the IR Sensor's distance readings.

WARNING You can have many Data Wires coming out of a single output, but you cannot connect multiple Data Wires to a single input.

understanding data types

In the programs above, you used either numbers or text as data. There's a third type of data: *logic values* (true or false), which are often used to express the result of a comparison. To help differentiate the data types, Numeric, Text, Logic, Numeric Array, and Logic Array inputs have different plug shapes, and the corresponding Data Wires have different colors. These are shown in Figure 6-8.

The plug shapes and colors are as follows:

- * Numeric inputs/outputs have a rounded shape, and the wire is yellow.
- * Logic inputs/outputs have a triangular shape, and the wire is aqua.
- * Text inputs/outputs have a square shape, and the wire is orange.
- * Numeric Array inputs/outputs have a double rounded shape, and the wire is thick and yellow.
- * Logic Array inputs/outputs have a double triangular shape, and the wire is thick and aqua.

NOTE You will learn about arrays in Chapter 13.

data type conversion

The EV3 Software will not allow you to attach a Data Wire to the wrong type of input. For example, you can't connect a

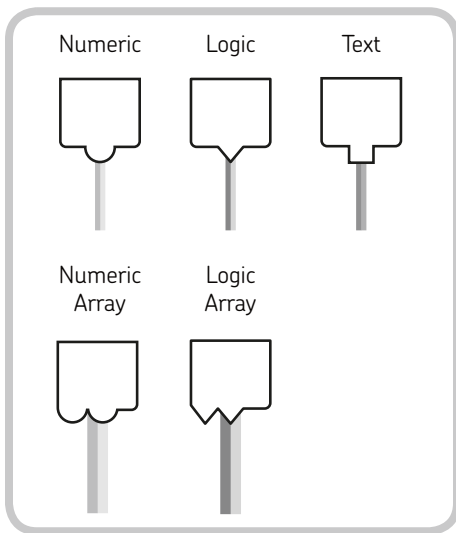







Figure 6-8: Different data types are distinguished by their input/output plug shape and wire color.

Text output to a Numeric input with a Data Wire. However, you can convert some types of data from one to another other automatically simply by connecting an output of one type to an input of another type.

As a visual guide, a data type can be converted if its plug can fit into another. For example, because the triangular shape can fit into the round or square shape, we know that a logic value can be converted into a number or text. Likewise, because the square shape does not fit into the round or triangular shape, we know that text can't be converted to a number or logic value. Table 6-2 lists all possible conversions.

table 6-2: the automatic type conversions

From	To	Resulting data
Numeric	Text	 A number becomes a text string. For example, 3.1415 would become the text string "3.1415".
Logic	Numeric	 The logic value True becomes 1; the logic value False becomes 0.
Logic	Text	 The logic value True becomes the text string "1"; the logic value False becomes the text string "0".
Logic	Logic Array	 The logic value becomes the first and only element of the resulting logic array.
Numeric	Numeric Array	 The numeric value becomes the first and only element of the resulting numeric array.

As you can see, numeric and logic values can be converted into text (shown in quotation marks), but text cannot be directly converted into a numeric or logic value. Also, a logic value can be converted to a numeric value—but not the other way around. Computers (like the EV3 Brick) represent all kinds of data using only the binary digits 0 and 1, which are equivalent to the truth states False and True, respectively. By convention, we convert the True logic value to 1 and False to 0, but the EV3 Software doesn't know how to directly convert any numeric value to a logic value. In fact, there could be several options. For example, should any nonzero number be converted to True? Or should any number less than a certain threshold be converted to False?

With some programming effort, we can overcome the limits of direct data conversion: You'll learn how to convert numeric values into logic values in Chapter 7 and how to convert text into a numeric value in Chapter 14.

DIGGING DEEPER: DECIMAL NUMBERS

The Numeric type represents numbers that can be positive or negative and can have digits after the decimal point; that means the EV3 Brick supports floating-point arithmetic operations, such as dividing 3 by 10, the result of which is 0.3. If you did the same operation on a system that supports only integers (numbers without digits after the decimal point), the result would be 0. Pretty different, isn't it?

following the remote IR beacon

By using the IR Sensor in Measure Beacon mode, you can make ROV3R follow the Remote IR Beacon. ROV3R will drive toward the remote as long as it can detect it. The Remote IR Beacon can be thought of as a kind of landmark the robot can use to determine its relative position and orientation.

You can reuse this concept in many creative ways. For example, you could build a robot that plays tag by attaching the Remote IR Beacon to your belt and having the robot chase you! Or you could put the beacon in the corner of a room and have the robot return to it, even after a long exploration, like a home base.

WARNING The Remote IR Beacon must be in Beacon Mode to be detected by the IR Sensor in Measure Beacon mode! To enable Beacon Mode, press button 9 on the Remote IR Beacon. To disable Beacon Mode, press any other button.

NOTE Data Wires can pass through a Switch block when the Switch block is in Tabbed View. As soon as you drag a Data Wire through the border of a Loop block or a Tabbed Switch block, a tunnel appears. To pass values through Switch blocks in Flat View, you need to use variables. (You'll learn how to use variables in Chapter 12.)

The key to creating programs like these lies in using the Proximity and Heading outputs of the IR Sensor block (in Measure Beacon mode) as Power and Steering inputs for a Move Steering block. For example, in the program shown in Figure 6-9, the robot will steer toward the beacon with steering action that is proportional to the measured heading: The greater the heading, the greater the steering action. When the heading is near 0 (frontal), the robot will drive straight toward the beacon.

Likewise, the robot's speed will be proportional to its distance from the beacon. When the beacon is far away, the robot will travel fast; when it's close by, the robot will slow to a stop. If the beacon is not detected, the robot will glide to a stop. (This program would be clearer if the Switch block was in Flat View, but I had to set it in Tabbed View to pass Data Wires into it.)

Here's how the program works. First of all, the entire sequence is set to repeat forever by using a Loop block in Unlimited mode.

The IR Sensor block in Measure Beacon mode has its **Channel** input set to **1**. It has three outputs:

- * The first output, *Heading*, provides the sensor's heading to the Remote IR Beacon. The values range from a low of -25 (indicating that the beacon is directly to the left of the beacon) to a high of 25 (which indicates the beacon is directly to the right). A value of 0 says that the beacon is directly ahead of the sensor.
- * The second output provides the *Proximity* of the Remote IR Beacon. Its values range from 0 (the nearest position) to 100 (the farthest).

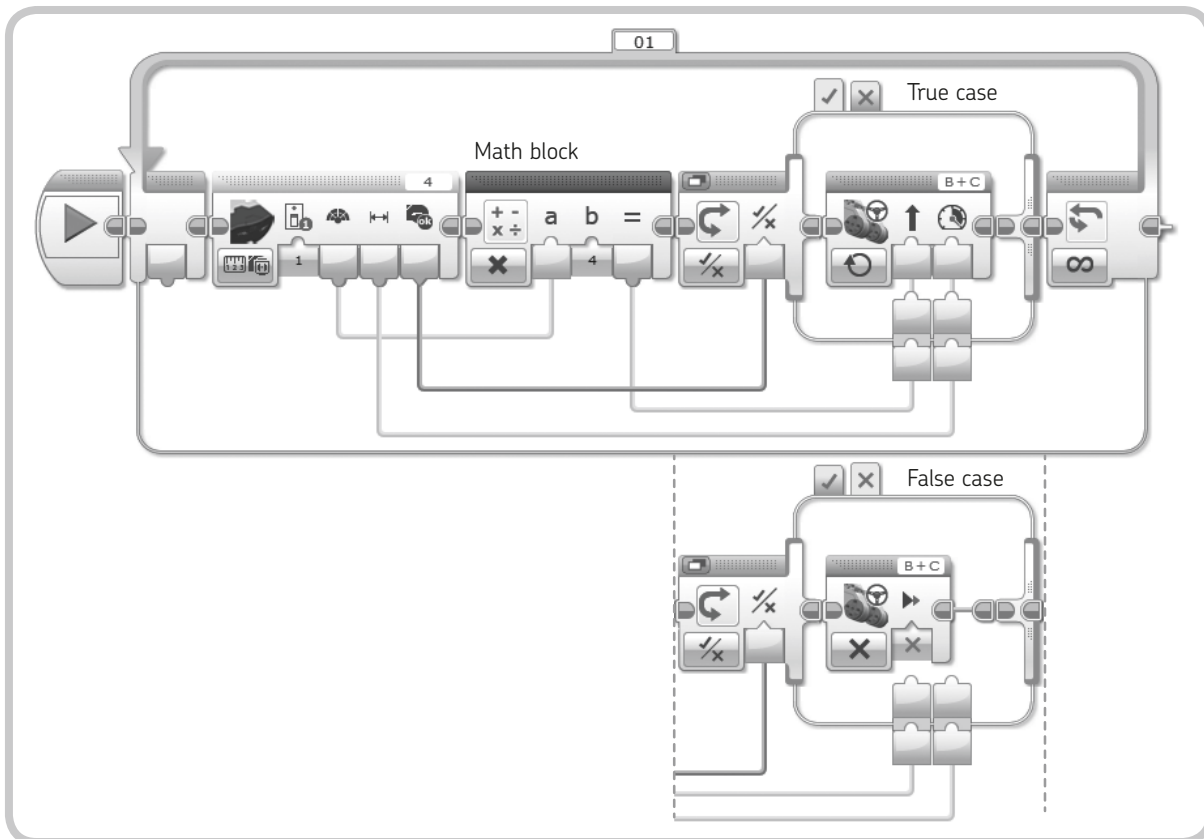


Figure 6-9: The beacon-following program. Both the True and False cases of the Tabbed Switch block contain blocks. Data Wires can go in and out of a switch only in Tabbed View.

* The third output provides the *Detected* state. The value is False when Beacon Mode is off or the signal is not detected, and it is True when the Beacon Mode signal is detected.

We need to take these three outputs and send them to other blocks in the program. First, we'll send the IR Sensor block Heading output to the Steering input in the Move Steering block. However, we can't use the raw data, because we need a number from -100 to 100 for the Steering input and the Heading output only ranges from -25 to 25. To solve this problem, we use a Math block in **Multiply** mode to multiply the value coming from the IR Sensor block Heading output by 4 before sending that value to the Steering input.

Next, we want to send the Proximity output to the Power input of the Move Steering block. Since the ranges of the values are the same, we use a Data Wire to connect them directly.

DIGGING DEEPER: ROBOT LOCALIZATION

A robot that is aware of its position and can navigate through an environment is one of the most interesting and challenging topics in mobile robotics research. Why not take up the challenge with EV3?

For the purposes of this experiment, you could use up to four Remote IR Beacons, set at known coordinates and on different channels, to build a robust localization system for your EV3 robot. This would give you up to four proximity and heading measurements from the IR Sensor. However, heading measurements to beacons are very inaccurate, so use them only as a very rough reference. On the other hand, proximity measurements should be reliable, as long as the IR Sensor's line of sight to the beacon is unobstructed and the beacon is more or less straight ahead of the sensor.

For best performance, avoid building things around the IR Sensor that could shield the beacon's signal. You want to maximize the sensor's field of view (especially sideways).

By merging the *eteroceptive* (external) measurements that the robot takes of its environment with its *proprioceptive* (internal) measurements of the distance it traveled and changes in orientation (using the rotation sensors in the Servo Motors, for example), you can create a robot with the ability to precisely determine its location and navigate. This technique requires some complex math, but the LEGO EV3 has enough computational power to handle it.

We use the Detected output of the IR Sensor block to choose between two cases of a Tabbed Switch block. When the Detected state is True, a Move Steering block in On mode is executed. Since the Move Steering block's Power and Steering inputs are connected to Data Wires coming from the IR Sensor block, the robot's speed and direction will change according to its distance and heading relative to the Remote IR Beacon.

When the Detected state is False, a Move Steering block in Off mode is executed, and the robot will stop moving.

The Move Steering block has the *Brake at End* input set to False, so the robot will glide to a stop when the beacon disappears or when Beacon Mode is turned off.

using the basic operations of the math block

The Math block is a Data Operations block. Depending on its mode, it performs mathematical operations on numeric inputs, producing the result as an output. The Math block can handle both integers and numbers with significant digits after the decimal point. The available operations are listed in Table 6-3.

table 6-3: basic operations of the math block

Mode	Inputs	Output
Add	a, b	$a + b$
Subtract	a, b	$a - b$
Multiply	a, b	$a \times b$
Divide	a, b	a / b
Absolute Value	a	a if $a \geq 0$, $-a$ if $a < 0$ (Result is always positive.)
Square Root	a	\sqrt{a}
Exponent	a, n	a^n

You'll learn about the Advanced mode of the Math block in Chapter 7. In Advanced mode, you can enter a formula in the Block Equation field that involves up to four operands.

EXPERIMENT 6-2

Expand the program in Figure 6-9 to display the IR Beacon's Proximity and Heading outputs on the EV3 Brick screen. You'll need Display blocks and Text blocks. To display more lines of text without having them overlap, you'll have to set different values for the Row input of the Display blocks (the input with the red Y icon). Experiment with various ways to display the messages using the fonts (Normal, Bold, and Large) and colors. And because you will probably use more than one Display block, remember to set the Clear Screen input to False in any Display block that follows. Otherwise, that block will clear the text displayed by the preceding one.

When a Display block with the Clear Screen input set to False shows a text string on a row that previously contained a longer string, the old string will not be cleared completely. Any characters from the old string that are beyond the length of the new string will remain displayed, and the screen will look messy. To avoid that without completely clearing the display, just enter a series of spaces in the Text block's c input field when building the text string. For example, if one of the Text blocks has "DIST = " as input a and a Data Wire carrying the data from the Proximity output of the IR Sensor block as input b, enter a few blank spaces as input c.

WARNING If you divide a number by zero, the result will be an error shown as *Inf* (infinite) on the Display block, with a sign depending on the dividend. If you use this value for the Rotations input of a Move block, the motors will turn forever. If you compute the square root of a negative number, the output will be an error displayed as ---- on the Display block. Errors displayed as ---- are interpreted as zero when they are used as inputs. (Errors are still sent through Data Wires connected to the output.)

conclusion

In this chapter, you learned all the features of the Infrared devices included in the EV3 set, the Remote IR Beacon and the IR Sensor. You learned how the Remote IR Beacon can be used with the IR Sensor to build a remote-controlled robot, and you discovered how to make your robot follow the beacon. You also learned how to read sensor data and how to transmit data between blocks using Data Wires. You read about the different types of data that you can use in your programs, how to pass them from one block to another, and how to display them on the EV3 screen. Finally, you saw how to use the Math block to perform simple operations. Along the way, you learned how to drive a tracked excavator!

EXPERIMENT 6-3

Create a program that plays a tone whose frequency is proportional to the Proximity output of the IR Sensor in Proximity mode. You'll need a Sound block in Tone mode, with the Proximity output wired to a Math block whose result is wired to the Sound block's Frequency input. Try a frequency range from 300 to 3000 Hz. For example, use the formula

$$\text{Frequency} = 10 * \text{Proximity} + 330$$
