Robot Projects for RobotBASIC
Volume I: The Fundamentals

Copyright February 2014 by John Blankenship
All rights reserved

Project 3: Measuring Distances

Previous projects have provided some fundamental insights involving programming the RobotBASIC simulator and using those programs with the RobotBASIC Robot. This project will explore how ranging sensors can measure the distance to nearby objects.

One of the ranging sensors on the RobotBASIC Robot is shown in Figure 3.1. One of the two metallic cylinders on the sensor acts as a speaker, the other as a microphone. In order to measure the distance to an object, the speaker creates a sound wave pulse that is above the range of human hearing (ultrasonic). This wave moves out from the robot and is detected by the microphone when it bounces back after hitting an object.

Figure 3.1: This ranging sensor can measure the distance to nearby objects.

The time it takes the sound wave to return can be used to determine how far away an object is from the robot. All the calculations are performed by the robot and returned to the user in increments of .5 inch. The simulator can perform distance measurements too, but the units are in pixels.

The function **`rRange()`** or **`rRange(0)`** returns the distance to objects directly in front of the robot as demonstrated in Figure 3.2. Notice that the robot is located at a random distance from the top wall. The distance to the wall is then stored in the variable **`d`**, and

then printed in a sentence by using several parameters separated by commas. Remember, if you are not familiar with any of the commands used a program, you can obtain more information from the RobotBASIC HELP file.

```
rLocate 400,50+Random(200)
d=rRange()
print "The distance to the wall is ", d, " pixels."
```

Figure 3.2: This short program demonstrates how easy it is to measure distances.

The **`rRange()`** function can also measure distance to objects that are not directly in front of the robot by providing a parameter that specifies at what angle the distance should be measured. For example, you can measure the distance to an object directly right of the robot with **`rRange(90)`**. A parameter of -45 would effectively point the ranging sensor left of center by 45°. The simulator actually allows any angle (increments of 1°) either to the left (negative numbers) or right (positive numbers) of center. The real robot's ranging capabilities are not quite as versatile.

Differences

The real robot has five ranging sensors spaced every 45° around the front of the robot. When you ask it to measure the distance to objects at some angle, it will give you the reading of the sensor that is closest to the specified angle. For example, if you asked for a reading at 50°, it would return the value associated with the right side sensor mounted at 45°. This limitation does not usually present a problem, but just remember to request readings in increments of 45° if you want the simulation to be representative of the real robot's capabilities. Note: The RB-9 Robot could have been built with one ranging sensor mounted on a motor so that it could be rotated to any requested angle, but this method was not used because it would be far slower than using five sensors. Custom robot's built with our RROS chip though, can have this capability.

The RB-9 differs in another ways too. For example, the simulator can measure any distance shown on the screen. The real robot's measuring limit is only a few feet depending on conditions. Soft objects, such as stuffed animals for instance, are harder to detect because they absorb the sound wave instead of reflecting it. Any time the real robot cannot find an object within its range, it will return a value of 255.

Finally, the sound wave emitted by the real robot is very cone-shaped, meaning it can detect objects in a relatively large region as shown in Figure 3.3. In contrast, the simulated robot only detects objects along a straight-line path away from the sensor. In practice, the cones for each of the real sensors tend to start where another leaves off, so that there are no large blind spots in the field of view. Due to limitations of the real sensors though, there is a small blind spot very close to the robot (see Figure 3.3).

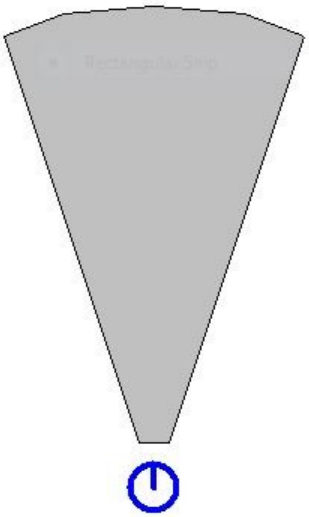

Figure 3.3: The ranging sensors can detect object in a cone-shaped area.

The program in Figure 3.4 shows how to read the distances at various angles to random objects, as shown in Figure 5.

```
rLocate 400,500
// Draw some random objects
Line 100+Random(100),400-Random(150),100+Random(200),550,10,BLUE
Circle 200,270,350+Random(100),300+Random(100),RED,RED
Rectangle
550+Random(100),300+Random(100),690,400+Random(200),GREEN,GREEN
// read the distances at 45 degree intervals
a=rRange(-90)
b=rRange(-45)
c=rRange()
d=rRange(45)
e=rRange(90)
// Draw lines to make it easy to see where readings are taken
line 0,500,799,500
line 400,500,0,100
line 400,500,400,0
line 400,500,799,100
// print the distances at appropriate places
xyString 320,480,a
xyString 280,420,b
xyString 370,400,c
xyString 490,420,d
xyString 450,480,e
end
```

Figure 3.4: This program shows the readings to random objects as shown in Figure 3.5.

Note the use of new commands in Figure 3.4. The statement **`line a,b,c,d`** draws a line from coordinates **`a,b to c,d`**. The statement **`Rectangle a,b,c,d`** will draw a rectangle whose upper left corner is **`a,b`** and its lower right corner is **`c,d`**. The circle statement is similar to the rectangle in that it draws a circle (or ellipse) inside the area defined by the coordinates as they were used in the rectangle command. Remember, you can always use the HELP file to get more information about RobotBASIC statements. You can also just write some small test programs to see what various commands and parameters do. Experimenting in this way can be a very effective way of learning.

Another new command in Figure 3.4 is **`xyString`**. It is very similar to a print statement, but the information printed can be located at any **`x,y`** position on the screen (as specified by the first two parameters following the command itself).

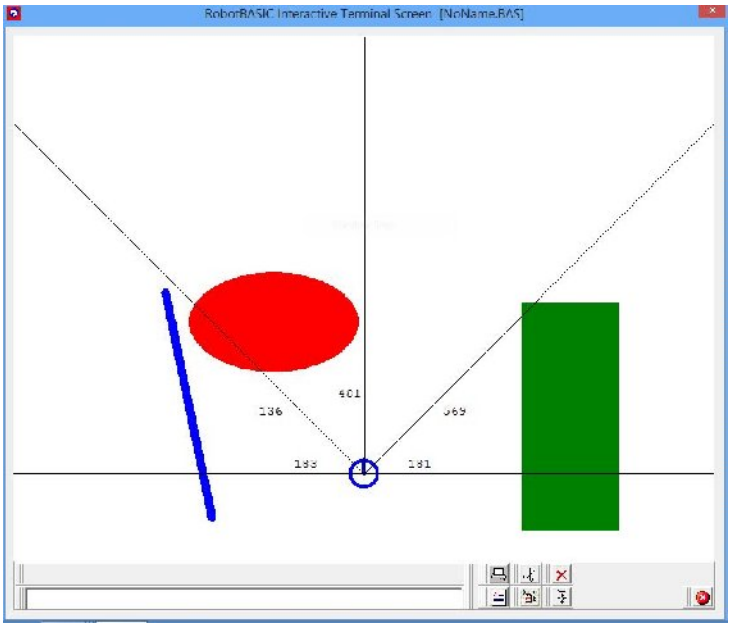

Figure 3.5: This screen shot helps visualize how the distance readings are taken.

Measuring distances with the real robot is just as easy as demonstrated by the program in Figure 3.6. It draws the lines, as before, to make it easy to see where the readings are made. In this case though, the values displayed will represent the distances to real objects near the real robot. Have someone walk around the robot or place objects at various distances and notice how the reading change on the screen. Remember, the readings are in units of .5 inch, so for example, a reading of 40 indicates 20 inches.

```
#include "RB-9.bas"
gosub InitRROScommands
PortNum = 5 // set equal to your Bluetooth Port Number
```

```
gosub InitializeRealRobot
line 0,500,799,500
line 400,500,0,100
line 400,500,400,0
line 400,500,799,100
// continually print the distances at appropriate places
while true
  xyString 320,480,rRange(-90)
  xyString 280,420,rRange(-45)
  xyString 370,400,rRange()
  xyString 490,420,rRange(45)
  xyString 450,480,rRange(90)
wend
```

Figure 3.6: This program prints the distances measured by the real robot.

Limitations

Faulty readings can be obtained when there are too many objects close to the real robot. This can cause the sound to reflect from object to object before returning to the robot, perhaps being detected from a different sensor than the one that emitted the sound. The best results are obtained if the surface area of objects are perpendicular to the sound beam. If the angle is severe, then it is possible that the sound wave will just bounce away without returning at all. This will make it impossible for the robot to see the object.

Suggestions for Study

Graphics is an exciting aspect of programming. Examine the program in Figure 3.4 to ensure that you understand how the random objects are drawn.

If you have a real robot, run the program in Figure 3.6 and verify that the real robot can measure distances to objects as described. While you should not expect the readings to be perfect, they should be reasonably accurate when appropriate objects are within the robot's measuring range.