RobotBASIC Projects for the Parallax Scribbler S3

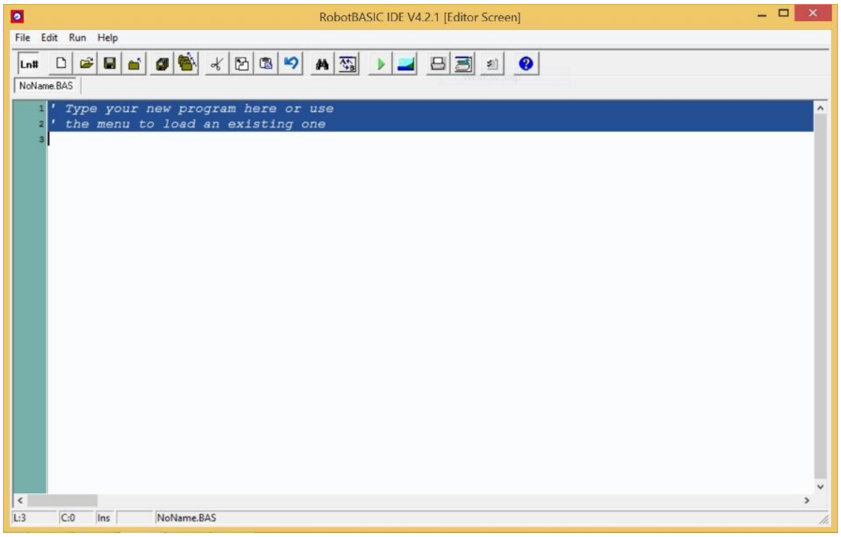
John Blankenship

Color Figures for Printed Book

Text only Figures not included.

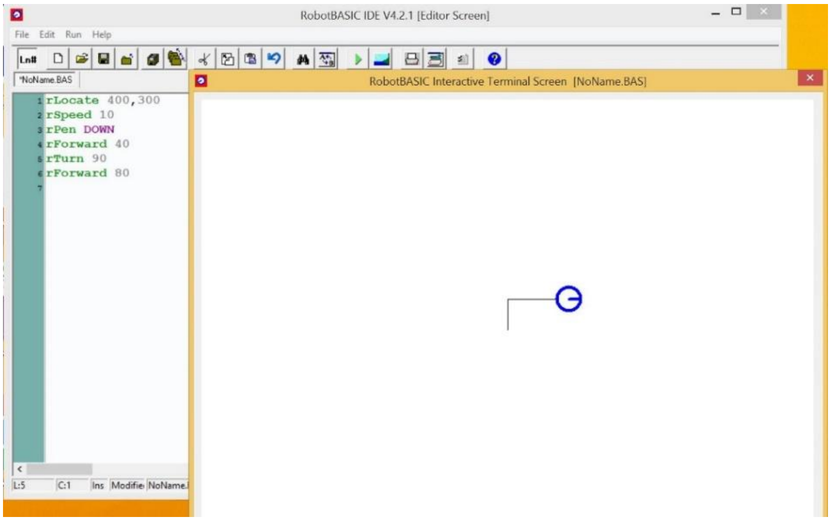
Chapter 1

Introduction to Programming with RobotBASIC



```
' Type your new program here or use
' the menu to load an existing one
```

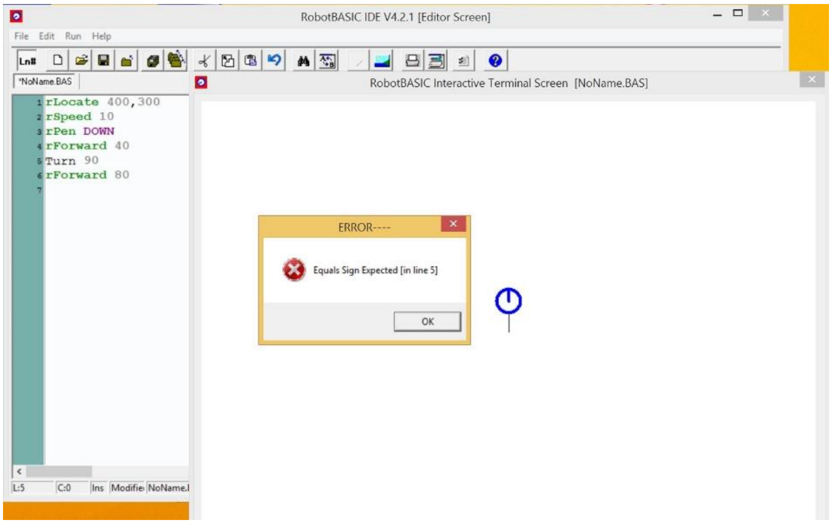
Figure 1.1: RobotBASIC's IDE


RobotBASIC IDE V4.2.1 [Editor Screen]
File Edit Run Help
*NoName.BAS
RobotBASIC Interactive Terminal Screen [NoName.BAS]
L:5 C:1 Ins Modifie NoName.

```
rLocate 400,300
rSpeed 10
rPen DOWN
rForward 40
rTurn 90
rForward 80
```

Figure 1.3: When you run the program in Figure 1.2, the robot responds as shown.

Figure 1.4: The Parallax S3 robot can be controlled by RobotBASIC.



RobotBASIC IDE V4.2.1 [Editor Screen]

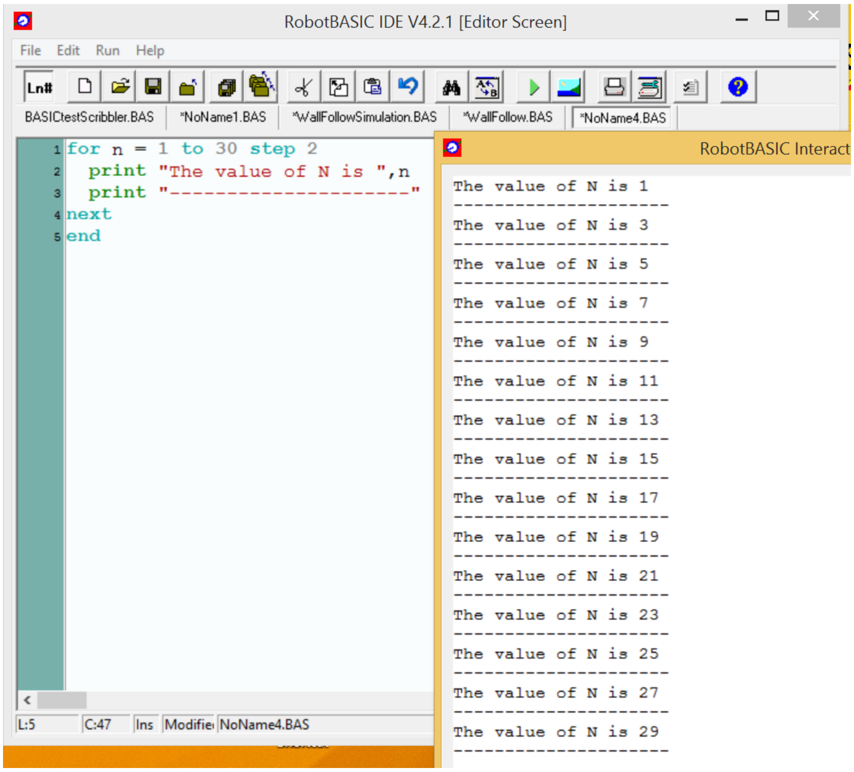
RobotBASIC Interactive Terminal Screen [NoName.BAS]

```
1 rLocate 400,300
2 rSpeed 10
3 rPen DOWN
4 rForward 40
5 Turn 90
6 rForward 80
7
```

ERROR----

Equals Sign Expected [in line 5]

Figure 1.5: Some errors are bad enough to confuse the system.



```
for n = 1 to 30 step 2
  print "The value of N is ",n
  print "---------------------"
next
end
```

```
The value of N is 1
---------------------
The value of N is 3
---------------------
The value of N is 5
---------------------
The value of N is 7
---------------------
The value of N is 9
---------------------
The value of N is 11
---------------------
The value of N is 13
---------------------
The value of N is 15
---------------------
The value of N is 17
---------------------
The value of N is 19
---------------------
The value of N is 21
---------------------
The value of N is 23
---------------------
The value of N is 25
---------------------
The value of N is 27
---------------------
The value of N is 29
---------------------
```

Figure 1.9: This partial view of the output window shows the program’s output.

Chapter 2

Enhancing the S3

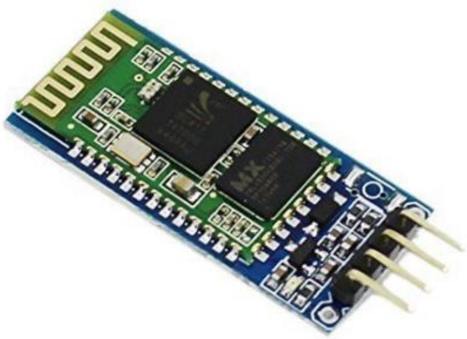


Figure 2.1: A Bluetooth transceiver can allow the S3 to communicate with RobotBASIC.

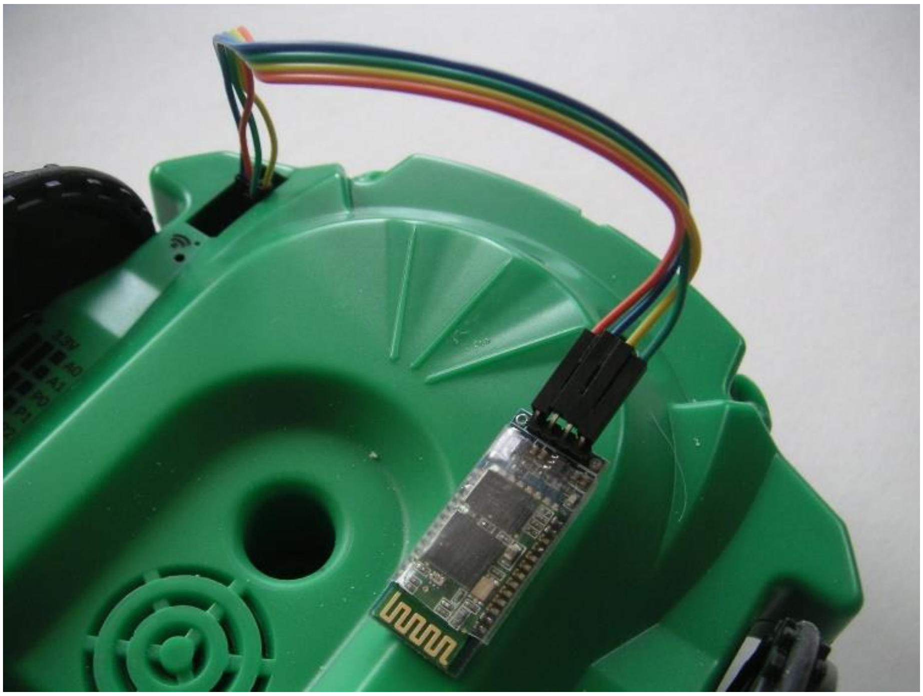


Figure 2.3: If Bluetooth is your only modification, 4 cables are all you need.

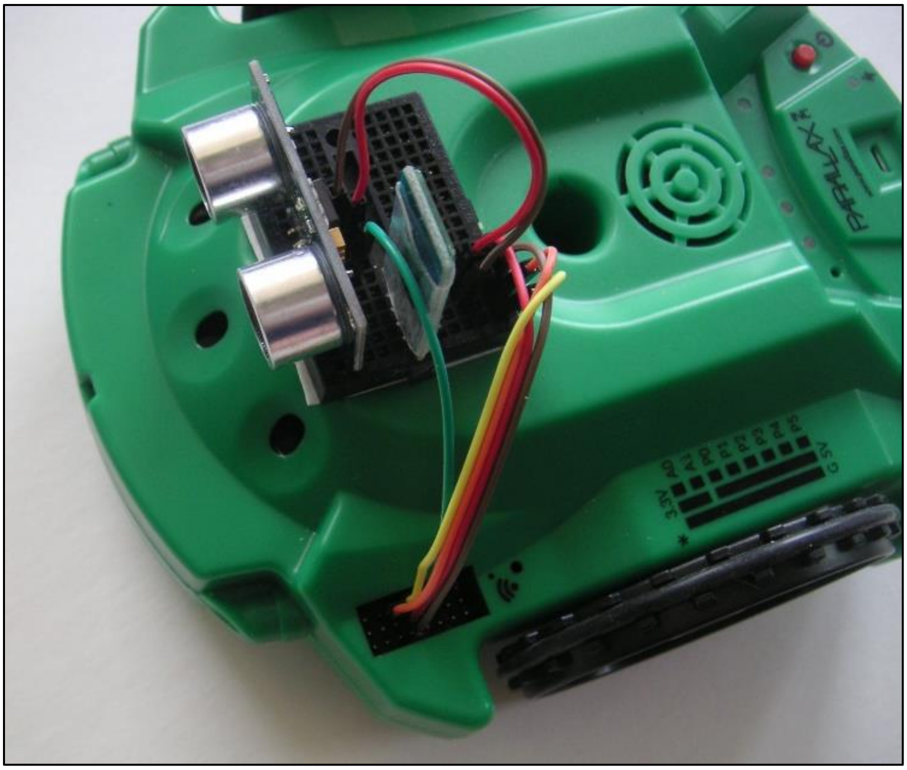


Figure 2.4: Mounting a small breadboard on the S3 makes it easy to interface a Bluetooth transceiver and a Ping sensor.

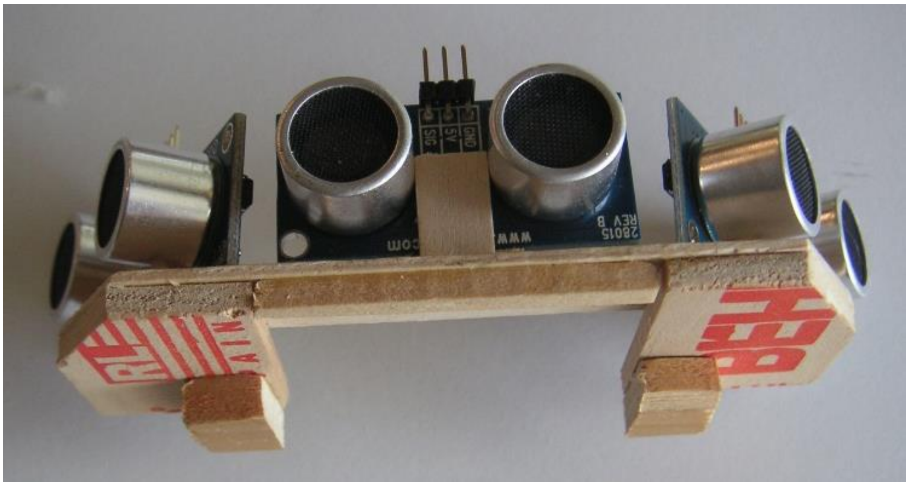


Figure 2.6: My first prototype.

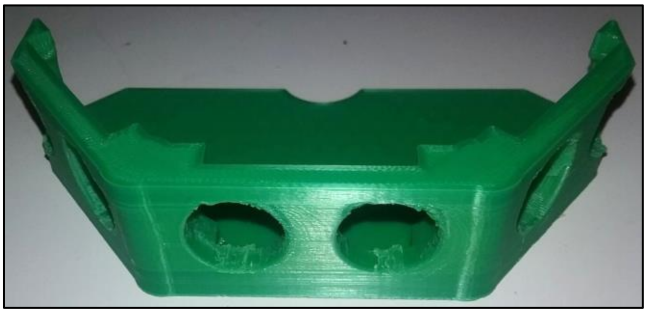


Figure 2.7: This 3D-printed mount makes it easy to install the Ping sensors in the proper orientation.

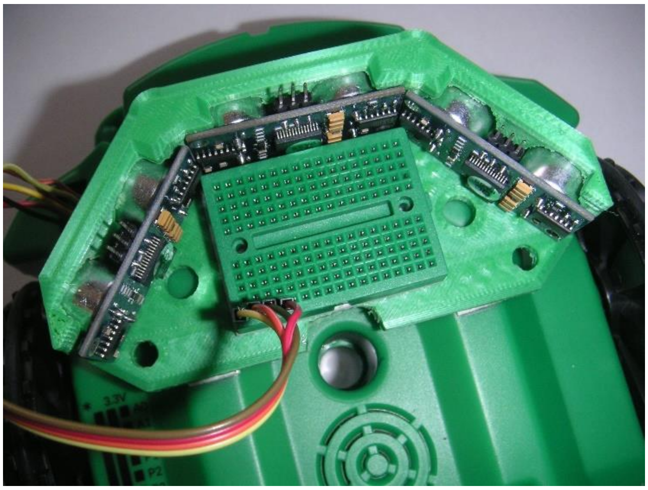


Figure 2.8: The mount with Pings and breadboard. Note the wires for the Bluetooth transceiver.

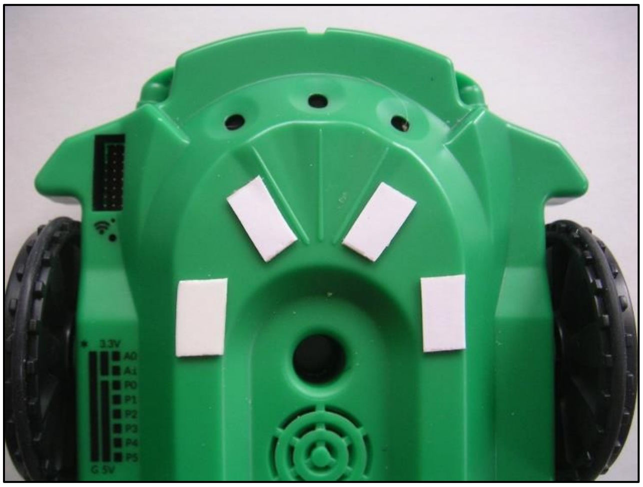


Figure 2.9: The mount is easily attached to the S3 with four small pieces of double-sided tape.

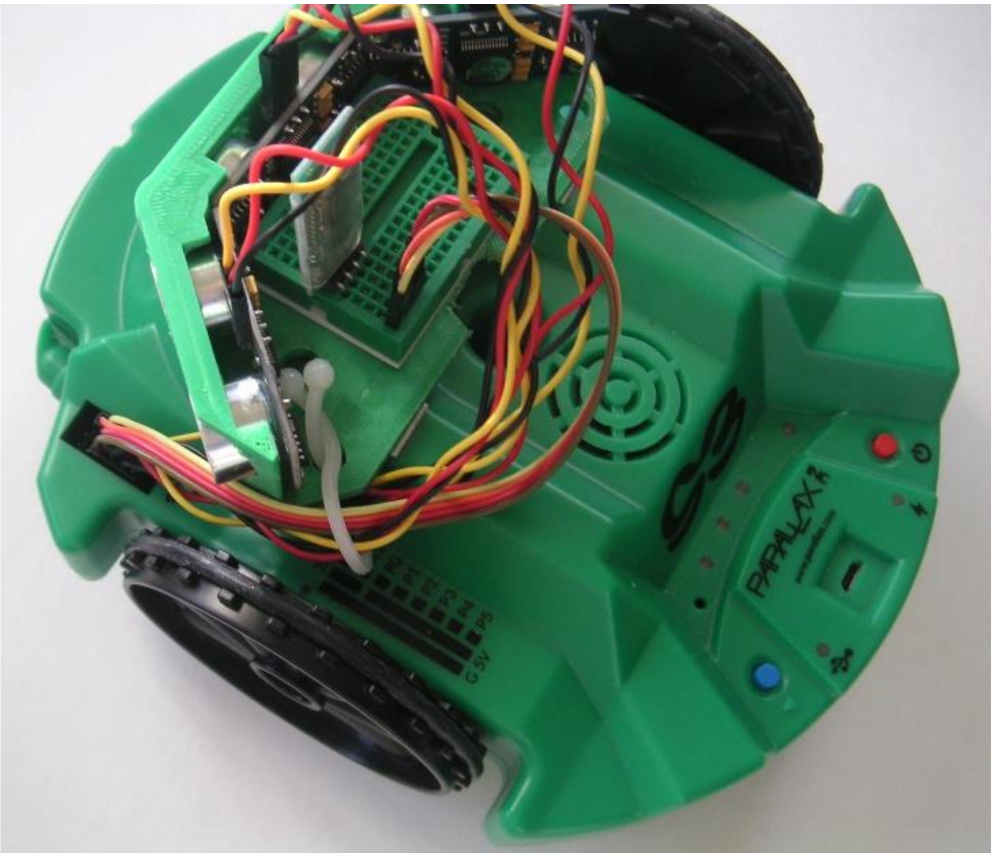


Figure 2.10: Wiring the circuit is the last step in preparing the hardware.

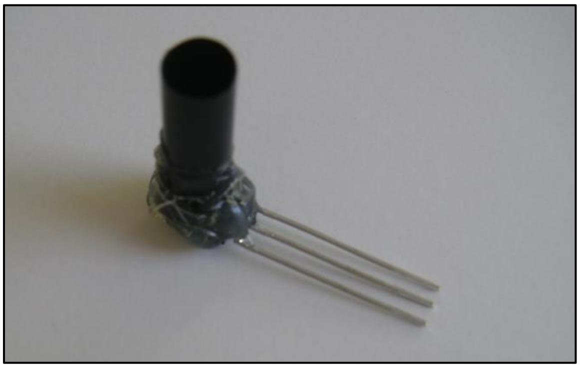


Figure 2.11: The RobotBASIC Beacon Detector can identify eight different beacons.

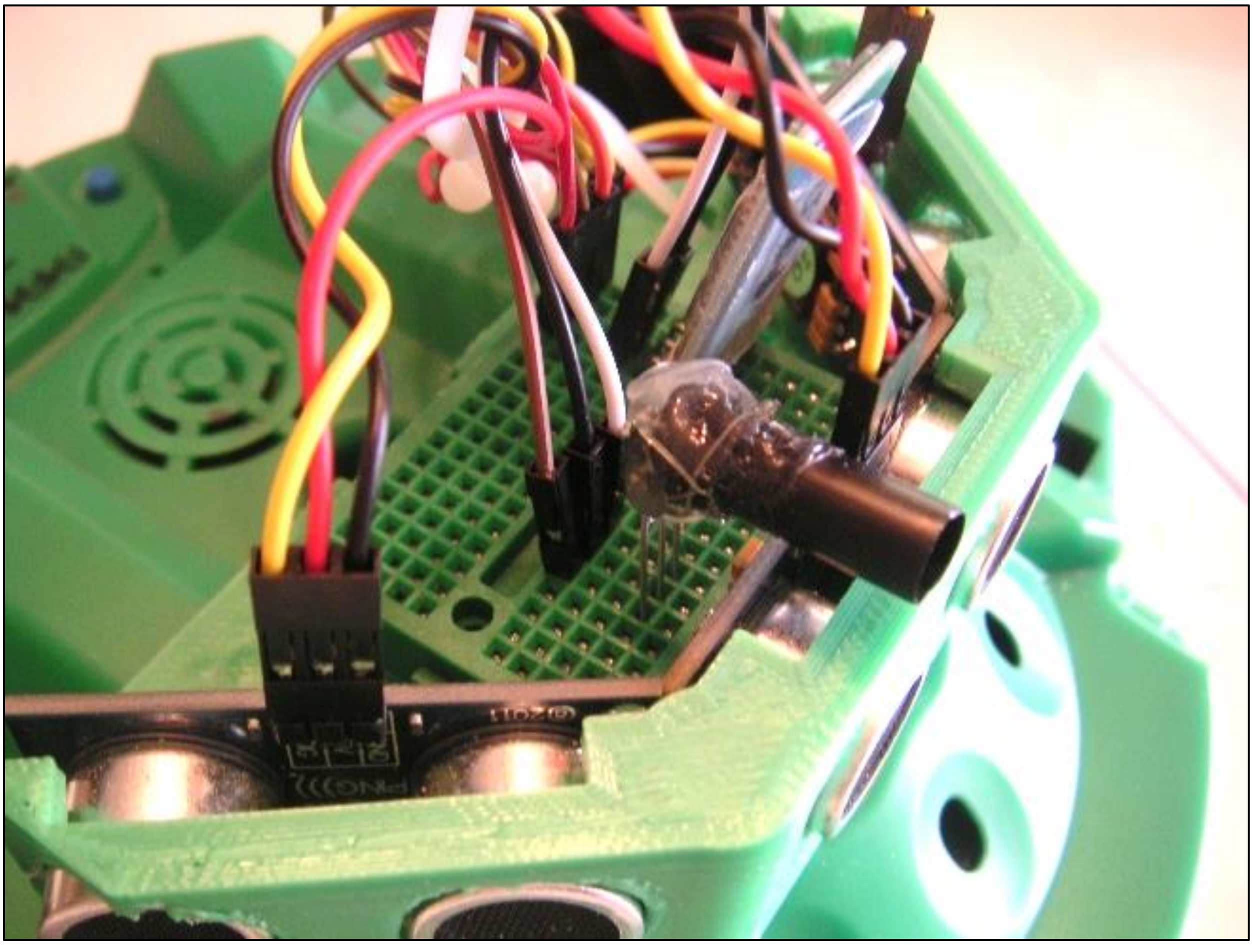


Figure 2.12: The detector plugs into the breadboard and faces forward to identify beacons.

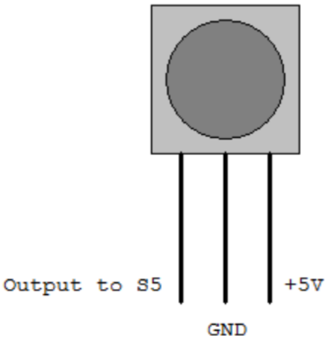
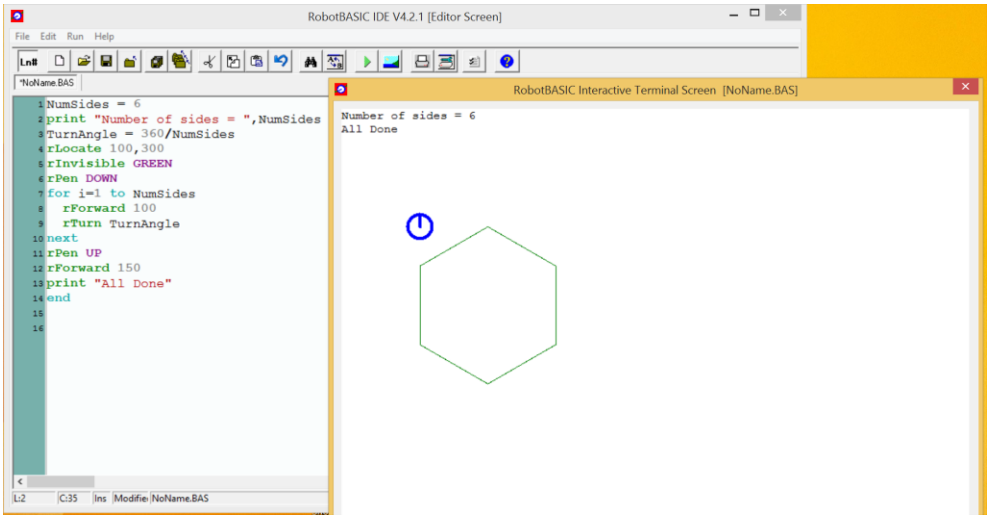


Figure 2.13: Beacon Pins (looking from the front, into the tube)

Chapter 3

Controlling the Robot’s Movements


RobotBASIC IDE V4.2.1 [Editor Screen]
File Edit Run Help
Ln#
*NoName.BAS
RobotBASIC Interactive Terminal Screen [NoName.BAS]
L:2 C:35 Ins Modifie NoName.BAS

```
NumSides = 6
print "Number of sides = ",NumSides
TurnAngle = 360/NumSides
rLocate 100,300
rInvisible GREEN
rPen DOWN
for i=1 to NumSides
  rForward 100
  rTurn TurnAngle
next
rPen UP
rForward 150
print "All Done"
end
```

```
Number of sides = 6
All Done
```

Figure 3.2: The simulated robot can draw.

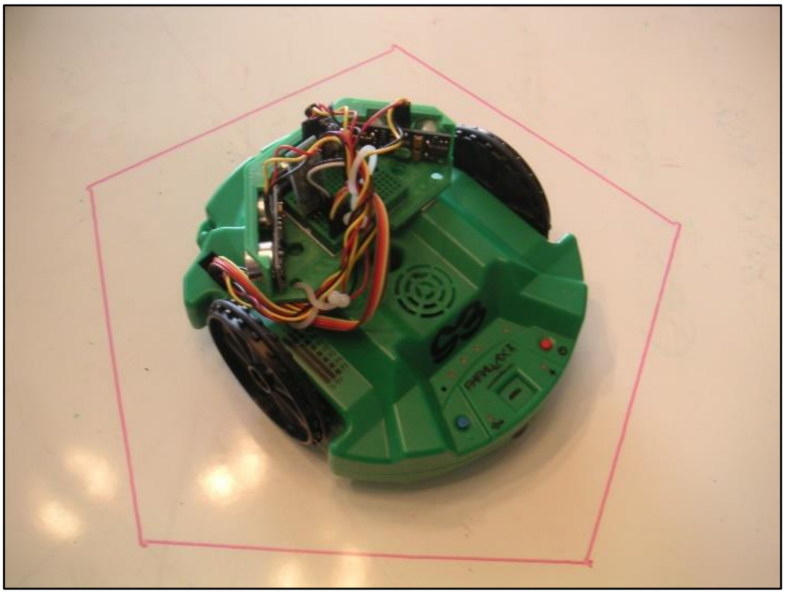


Figure 3.4: A white board makes a great reusable surface for the S3.

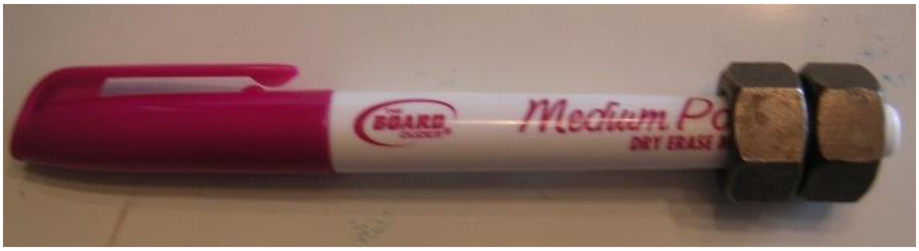


Figure 3.5: A little weight on the marker makes it draw much better

Chapter 4

More on Movement

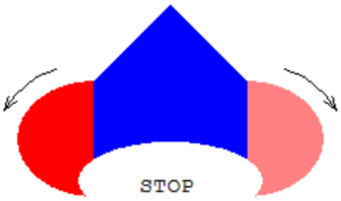


Figure 4.1: Placing the mouse over different areas of this diagram will move the robot.

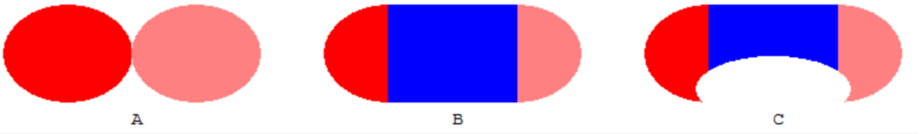


Figure 4.3: These drawings show how overlapping objects help create the control panel.

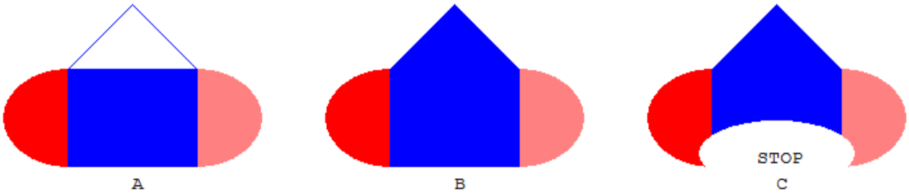


Figure 4.4: These steps almost complete the control panel.

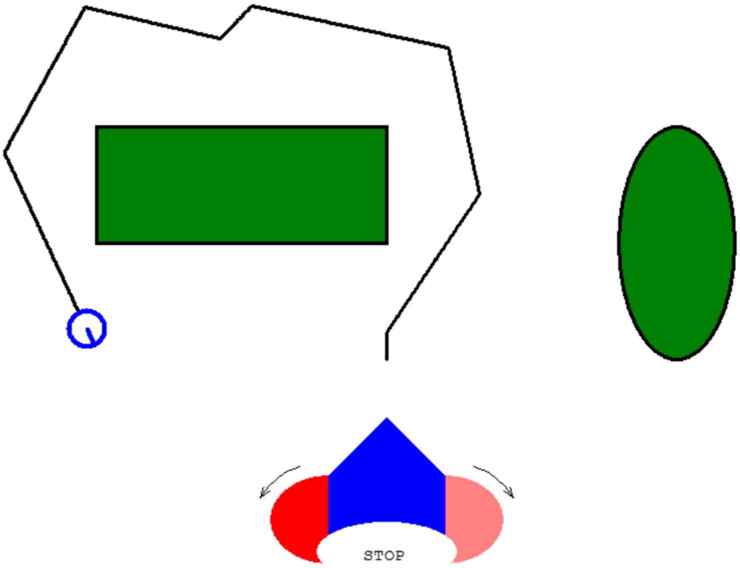


Figure 4.5: The robot rotates when it turns.

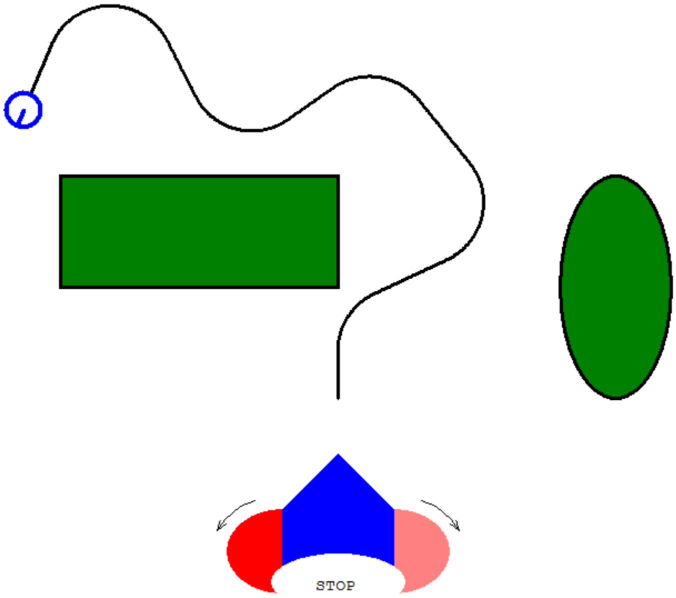


Figure 4.6: This style of turns may be preferred by some.

Chapter 5

Sensor Controlled Movement

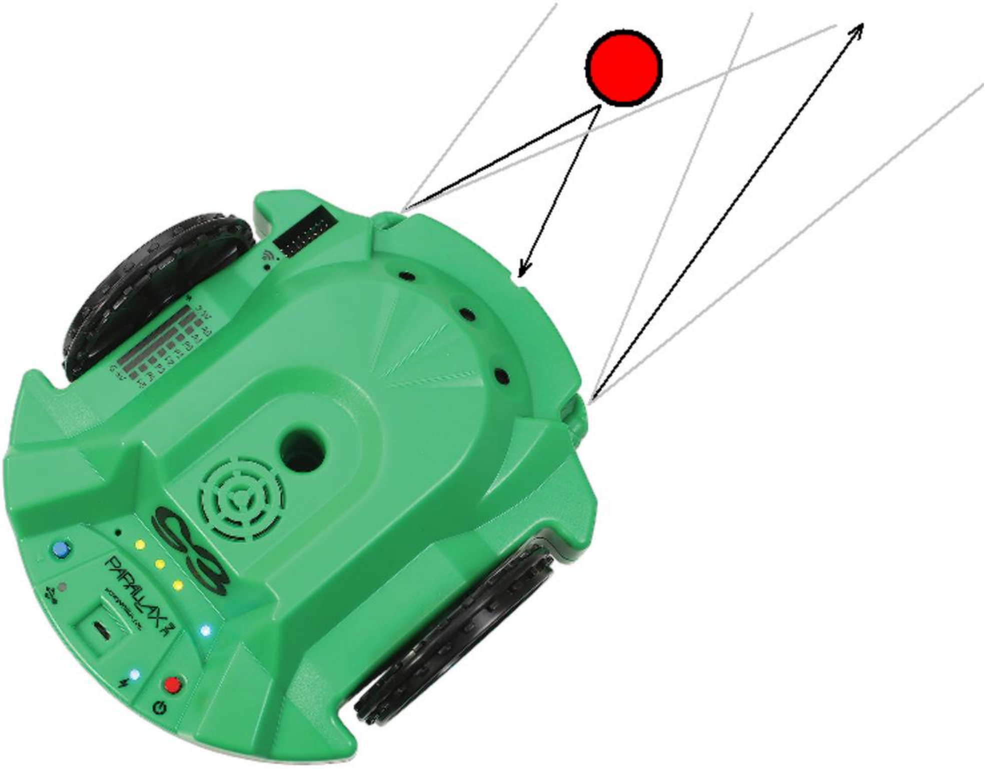


Figure 5.1: IR light allows the S3 to detect nearby objects.

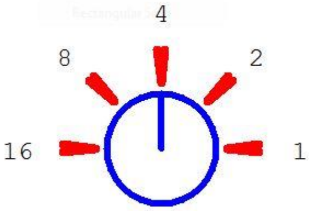


Figure 5.2: The simulator has five perimeter proximity sensors.

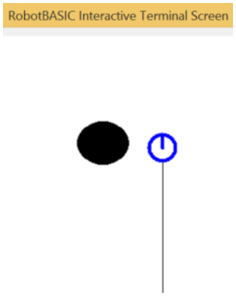


Figure 5.5: The robot stopped because of the simulator’s side sensor.

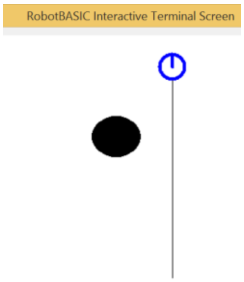


Figure 5.6: The simulated robot now cannot see to its side.

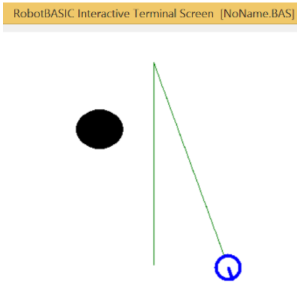


Figure 5.8: The robot now turns away from the wall at a random angle.

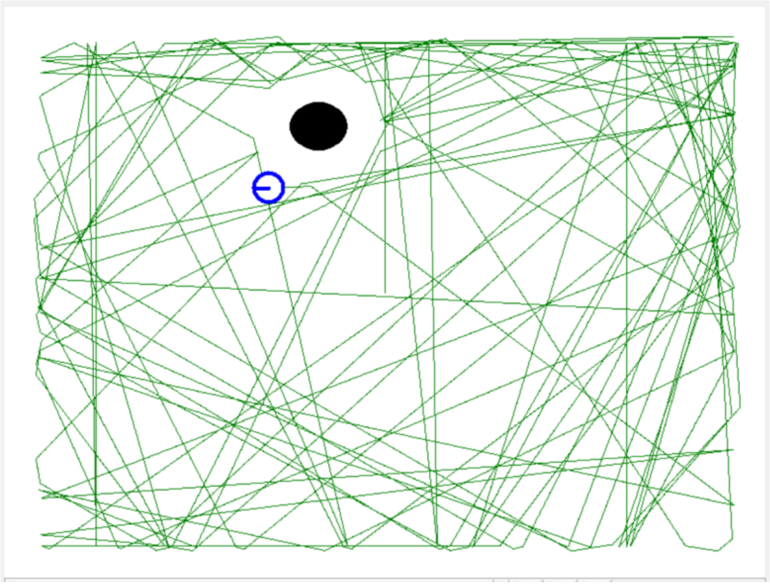


Figure 5.9: The robot roams randomly while avoiding objects.

Chapter 6

Finding a Light

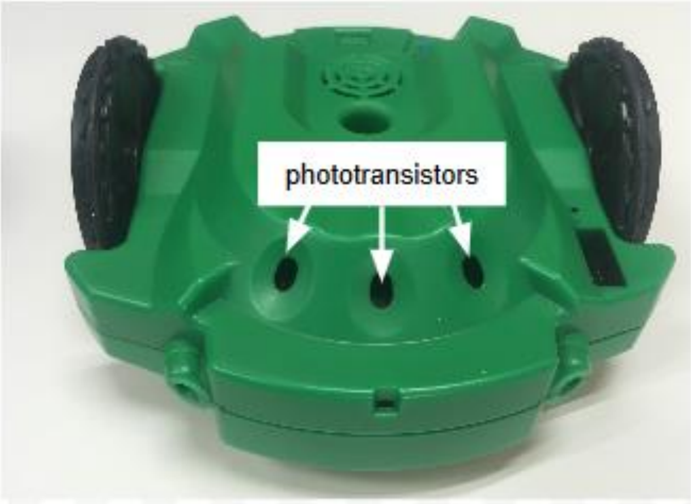
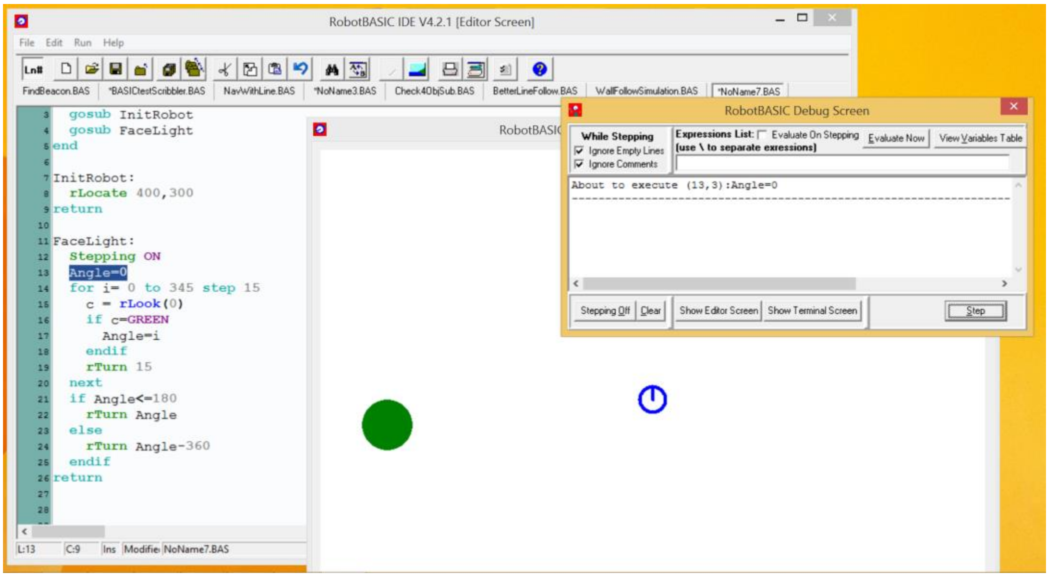


Figure 6.1: There are three light sensors on the front of the S3.



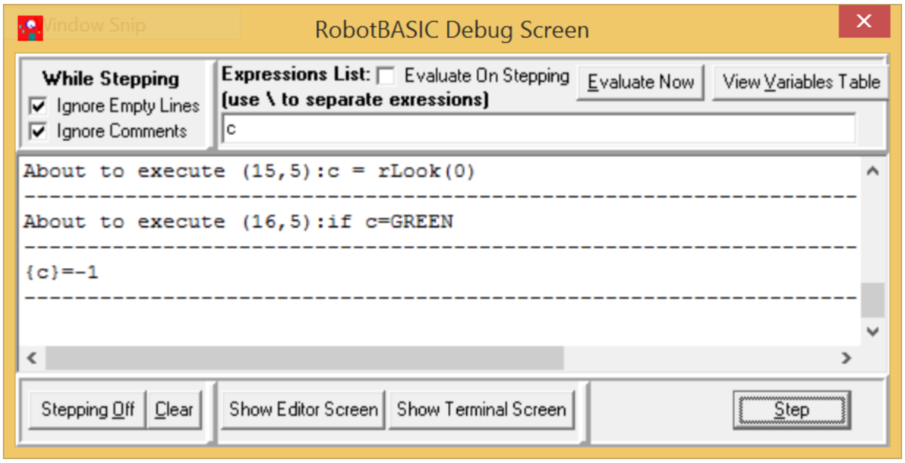
```
  gosub InitRobot
  gosub FaceLight
end

InitRobot:
  rLocate 400,300
return

FaceLight:
  Stepping ON
  Angle=0
  for i= 0 to 345 step 15
    c = rLook(0)
    if c=GREEN
      Angle=i
    endif
    rTurn 15
  next
  if Angle<=180
    rTurn Angle
  else
    rTurn Angle-360
  endif
return
```

While Stepping
☑ Ignore Empty Lines
☑ Ignore Comments

Expressions List: ☐ Evaluate On Stepping
(use \ to separate exressions)

```
About to execute (13,3):Angle=0
------------------------------------------------------------------
```

Figure 6.5: Arrange the edit, output, and debug windows so you can see everything.



```
c
```

```
About to execute (15,5):c = rLook(0)
-------------------------------------------------------------------
About to execute (16,5):if c=GREEN
-------------------------------------------------------------------
{c}=-1
-------------------------------------------------------------------
```

Figure 6.6: You can check the value of variables and expressions.

Chapter 7

Line Sensors: Following a Line

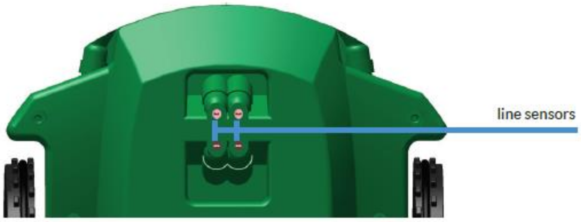


Figure 7.1: The S3 has two line sensors on the bottom of the robot.

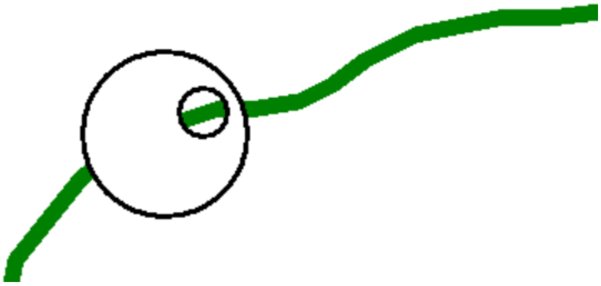


Figure 7.2: Imagine you are the robot.

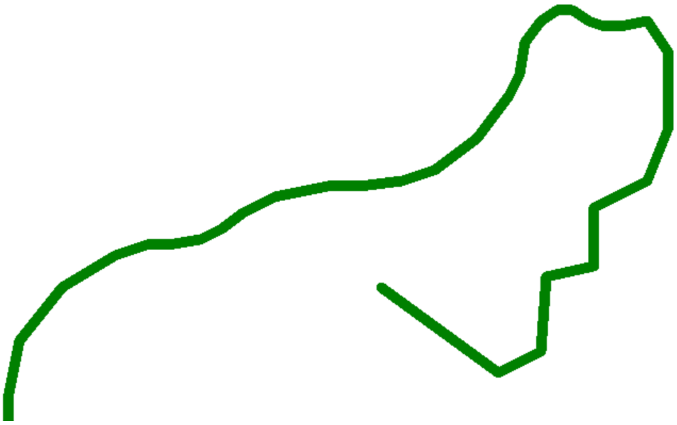


Figure 7.4: This is the line to be followed.

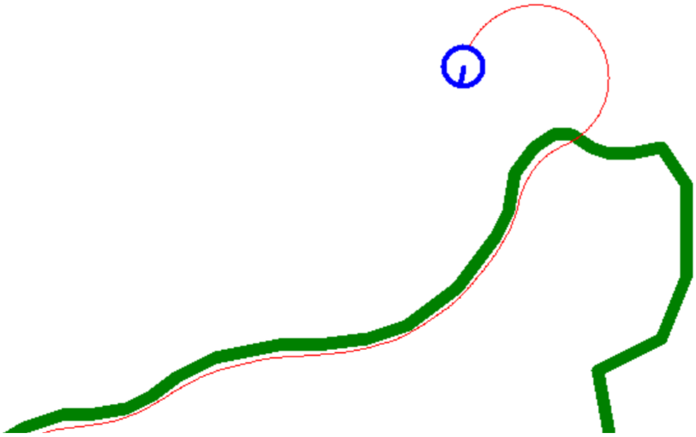


Figure 7.5: The robot handles gentle curves with ease, but loses the line when it turns sharply.

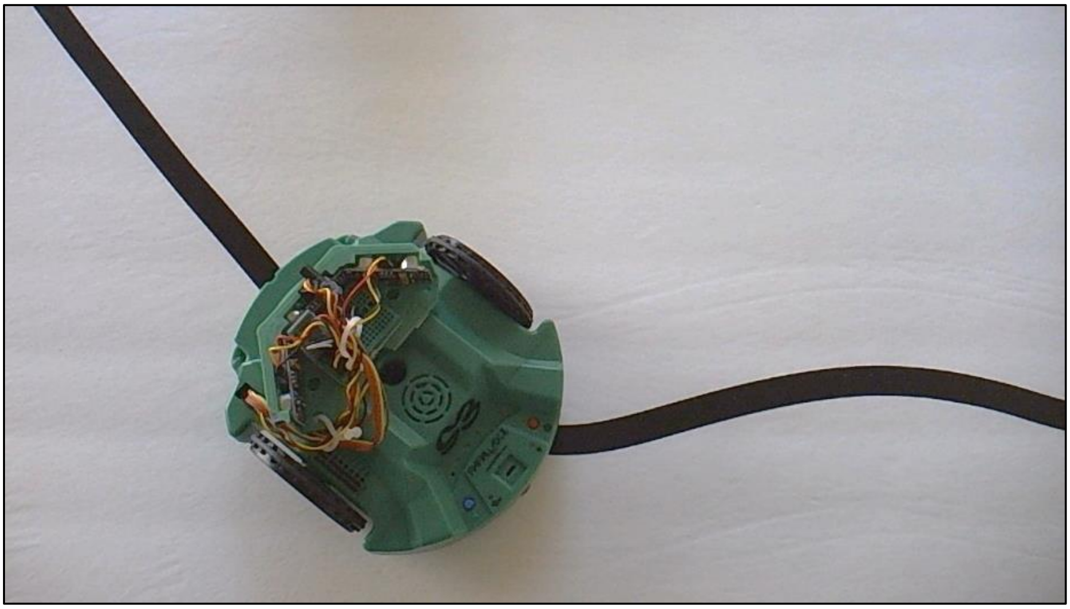


Figure 7.6: Foam board and electrical tape makes creating a line easy.

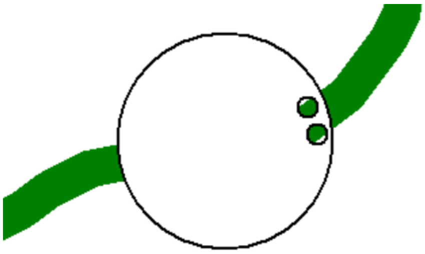


Figure 7.7: We can improve the line following behavior if we use both sensors.

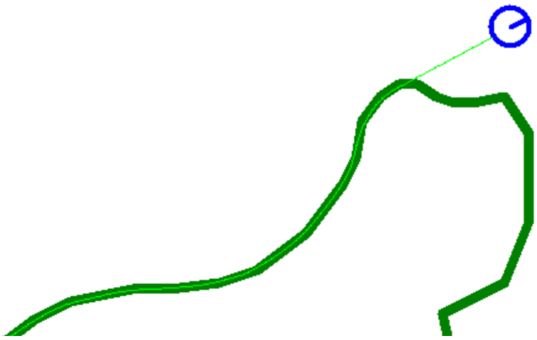


Figure 7.10: Notice how the robot now stays centered on the line.

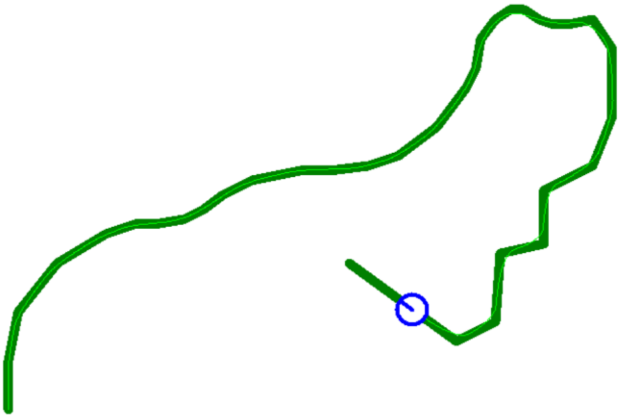


Figure 7.12: The robot now follows the line perfectly.

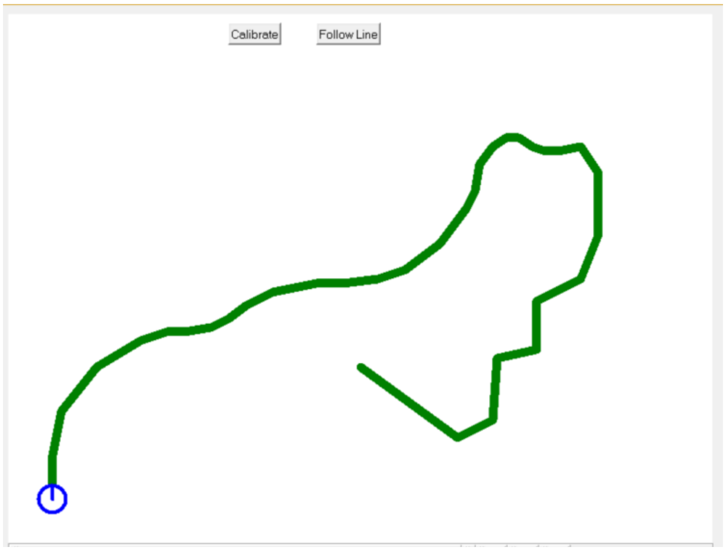


Figure 7.14: These buttons allow you to calibrate the line sensors or follow the line.

Chapter 8

Ranging Sensors: Measuring Distance

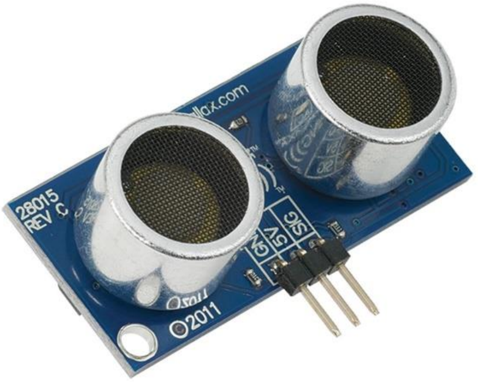


Figure 8.1: This ultrasonic sensor can measure distances to objects.

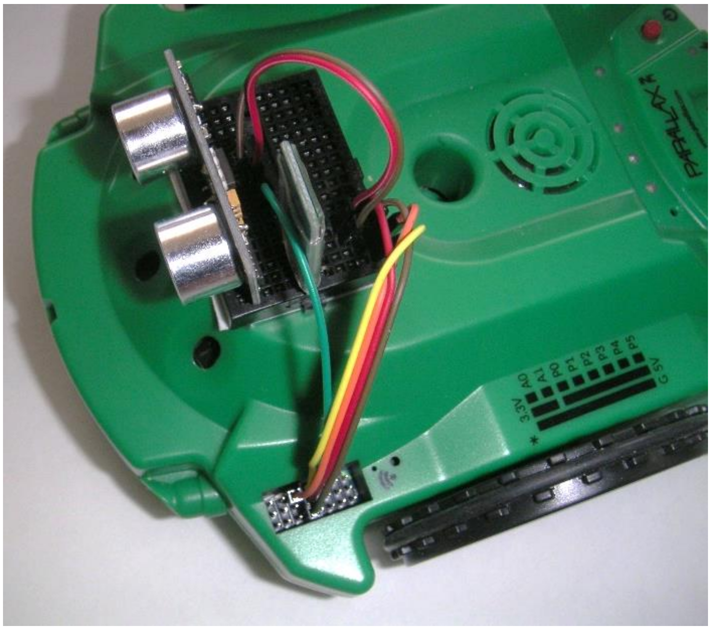


Figure 8.2: A single Ping sensor can be mounted on your robot as shown.

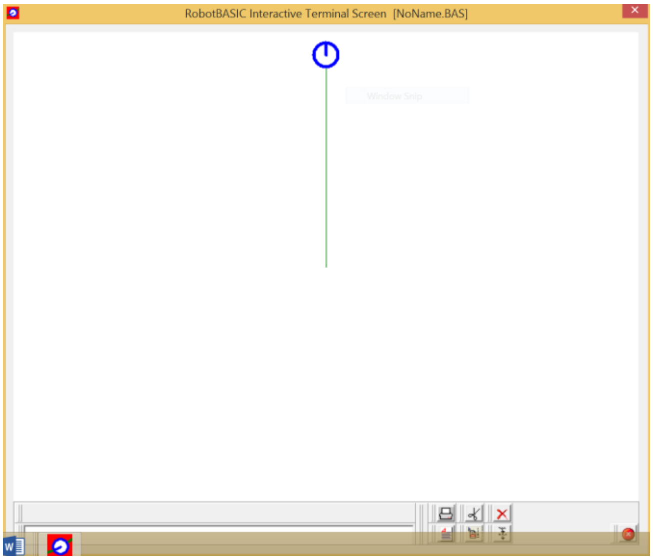


Figure 8.6: The robot was initialized in the center of the screen and moves until it reaches a wall.

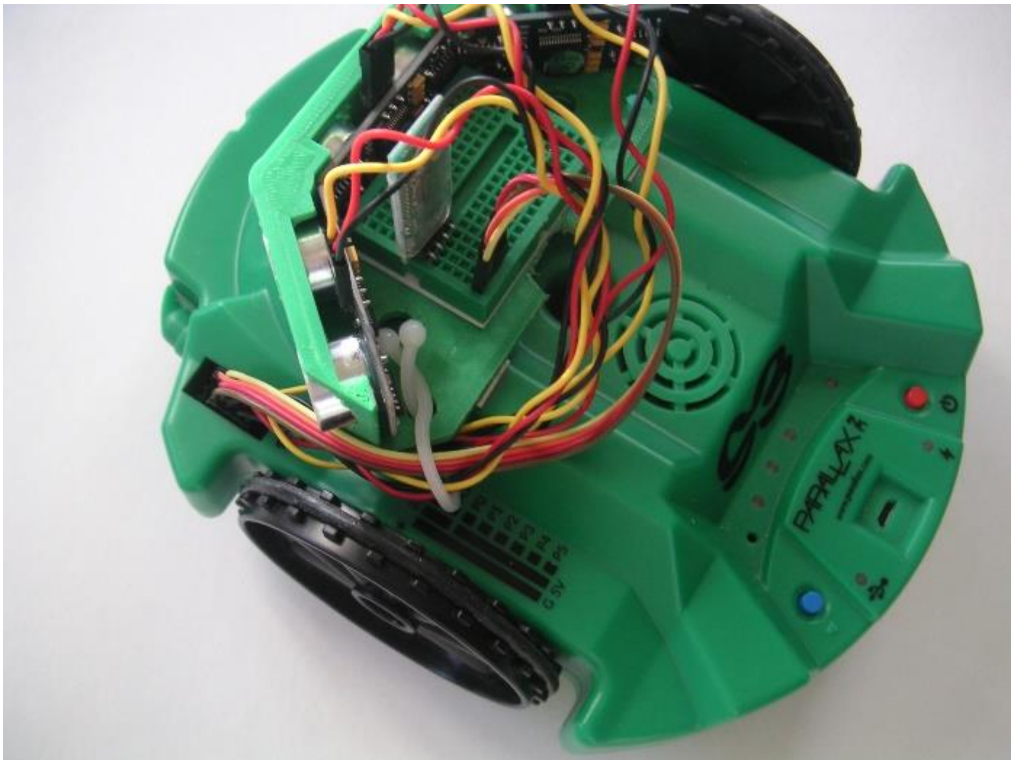
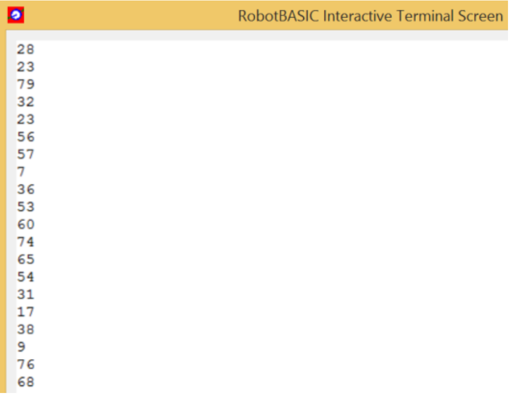
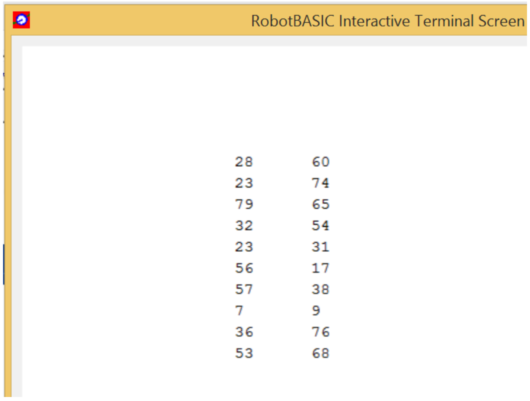


Figure 8.3: Mounting three Pings greatly improves your S3's capabilities.



```
28
23
79
32
23
56
57
7
36
53
60
74
65
54
31
17
38
9
76
68
```

Figure 8.9: This is one set of random numbers generated by Figure 8.8.



```
28        60
23        74
79        65
32        54
23        31
56        17
57        38
7         9
36        76
53        68
```

Figure 8.11: It is easy to create a two-column output.

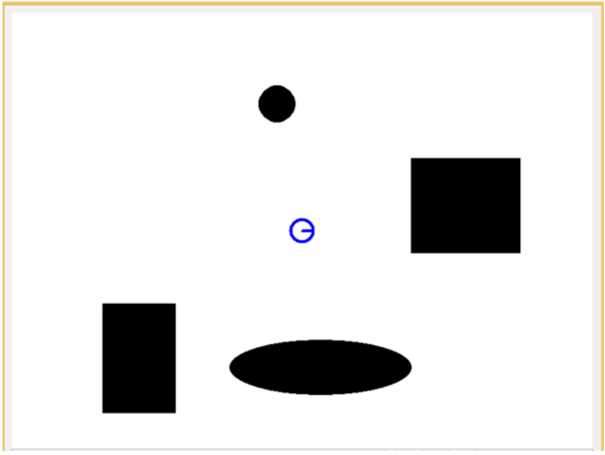


Figure 8.14: This is the environment for the simulated robot.

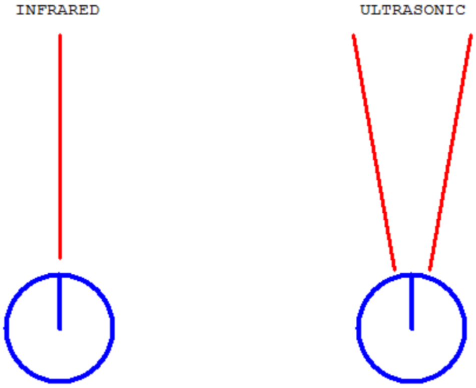


Figure 8.15: Ultrasonic rangers have a wider beam than IR.

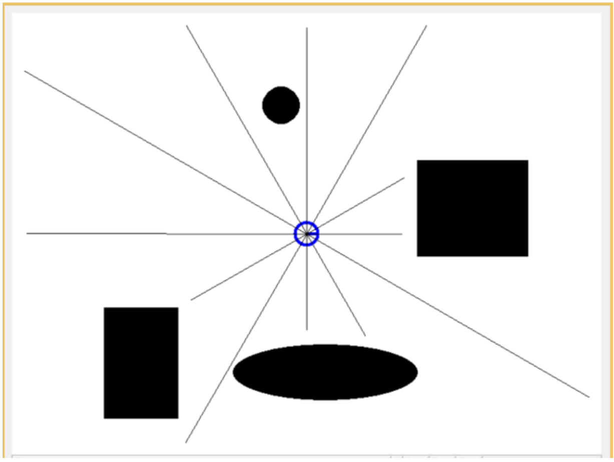


Figure 8.16: These lines represent the distances measured at each position.

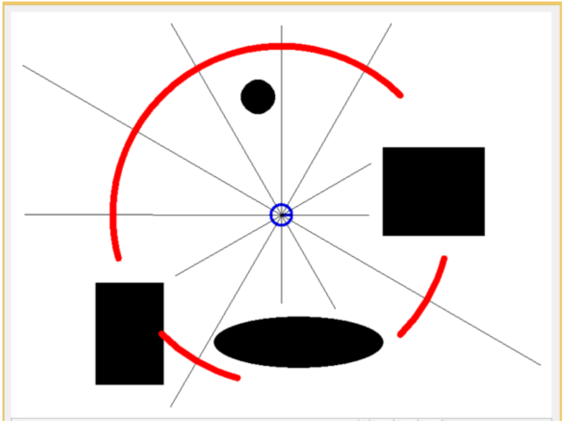


Figure 8.17: The arcs show open areas as deemed by the scan data.

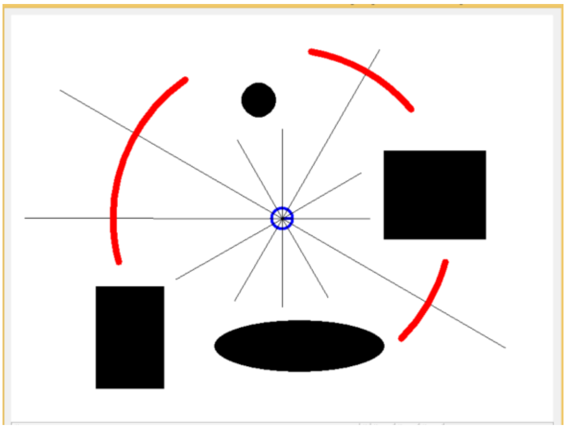
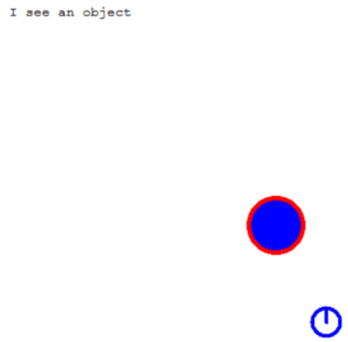
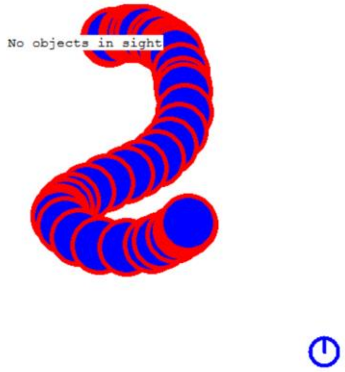


Figure 8.18: The ultrasonic scan more accurately detects the open areas in the environment.



```
I see an object
```

Figure 8.20: As you move the ball with the mouse, the program displays whether it sees the ball or not.



```
No objects in sight
```

Figure 8.21: If you do not erase the ball, it leaves

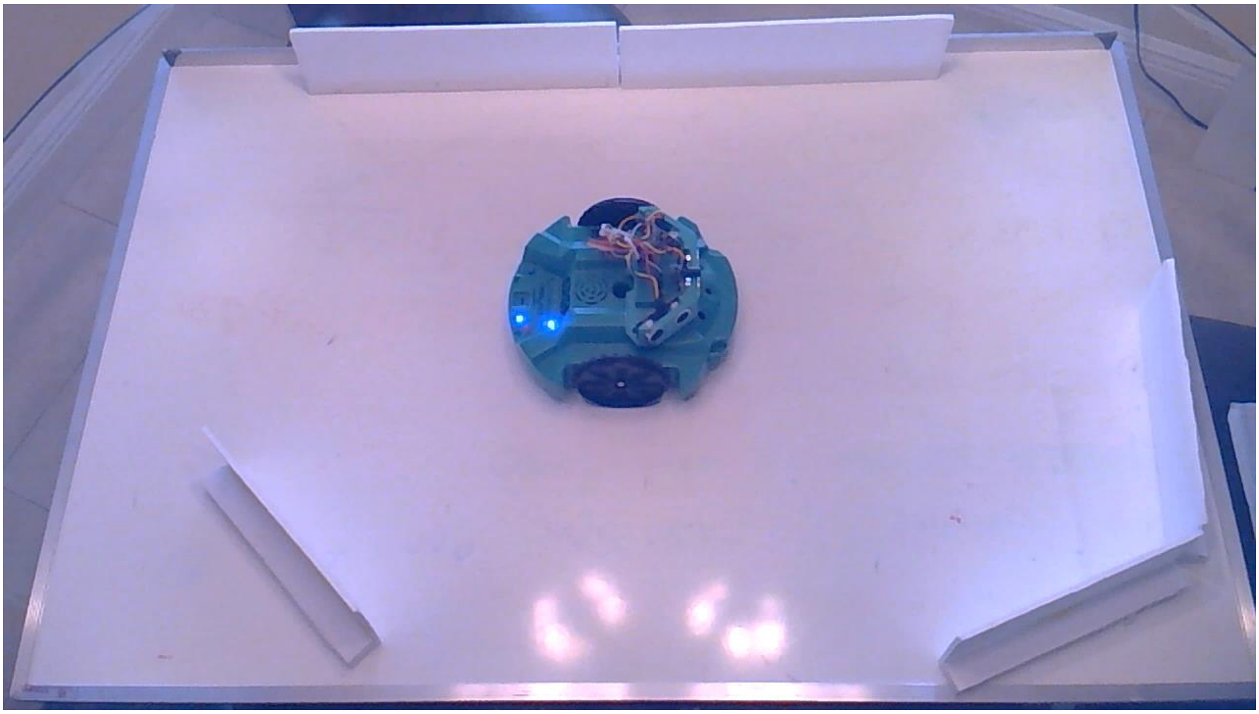


Figure 8.22: The program used for the simulator can direct the **S3** to scan the area around it.

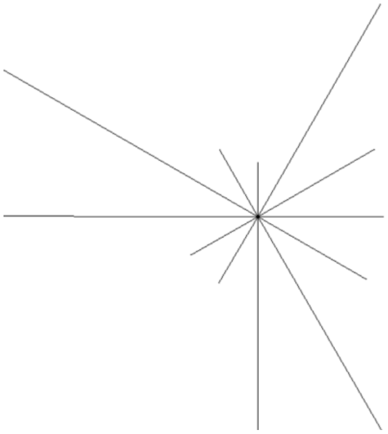


Figure 8.23: The scan lines drawn by the program correspond to the objects in Figure 8.20.

Chapter 9

Following a Wall

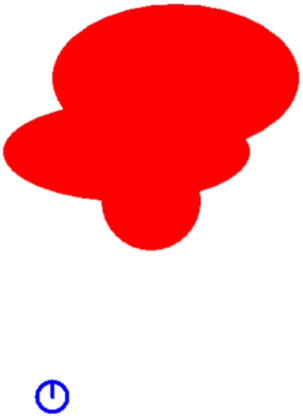


Figure 9.2: This output is produced by the program in Figure 9.1.

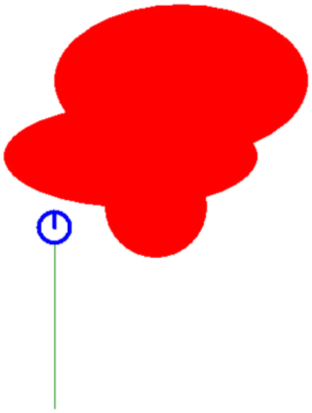
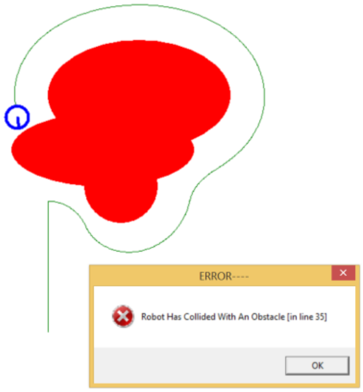


Figure 9.4: The code in Figure 8.3 moves the robot as shown here.



Robot Has Collided With An Obstacle [in line 35]

Figure 9.6: The robot follows the wall until it reaches a protrusion.

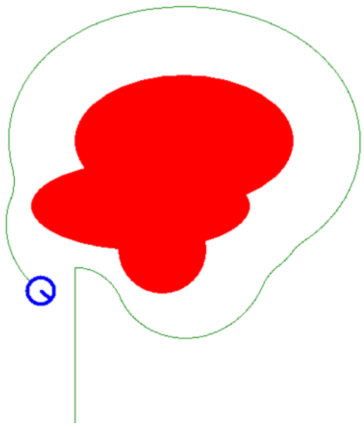
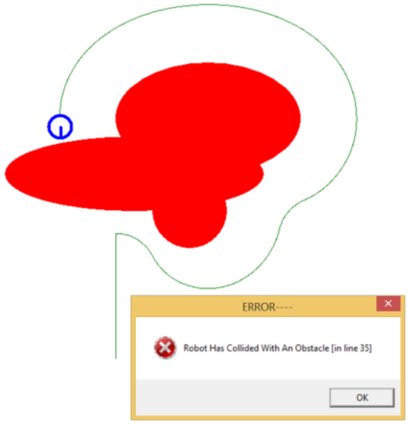
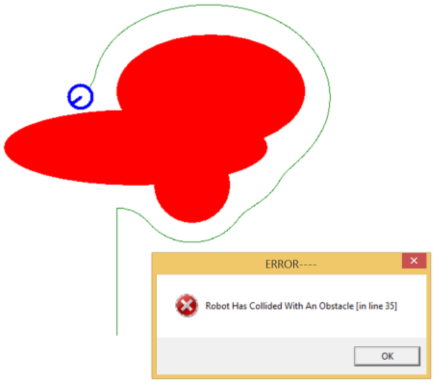


Figure 9.7: Move the robot further from the wall seems to work.



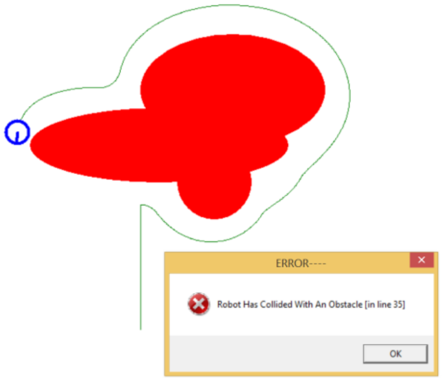
Robot Has Collided With An Obstacle [in line 35]

Figure 9.8: Changing the wall results in same error as before.



Robot Has Collided With An Obstacle [in line 35]

Figure 9.9: If the robot looks ahead slightly it seems to work better.



Robot Has Collided With An Obstacle [in line 35]

Figure 9.10: The robot is now almost perfect

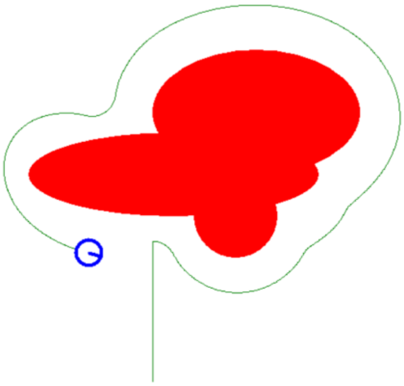


Figure 9.11: The simulated robot seems to handle all situations.

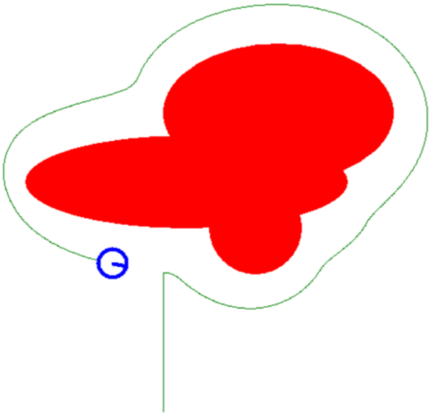


Figure 9.12: Simulating an ultrasonic scan works well.

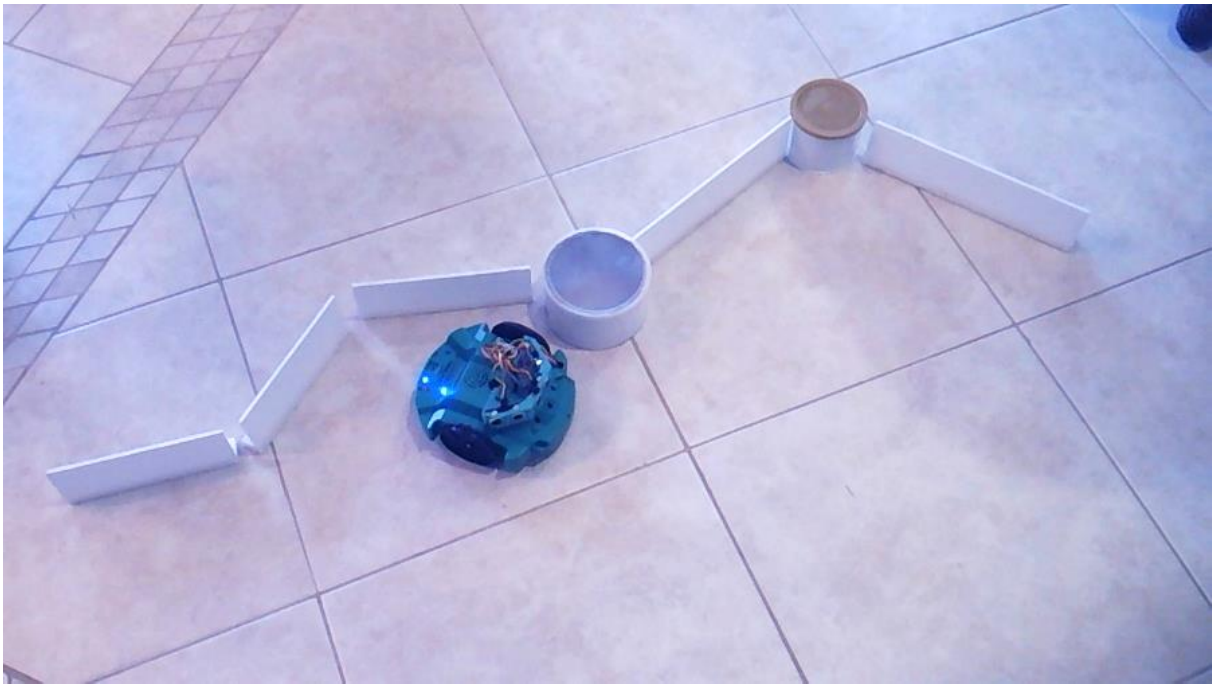


Figure 9.13: Wall modules make building real walls and obstacles easy.

Chapter 10

A Virtual Sensory System

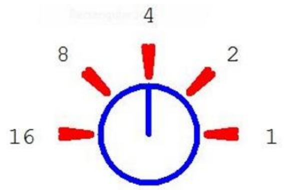


Figure 10.1: The simulator has five FEEL sensors.

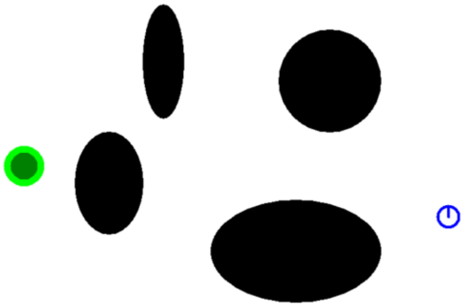


Figure 10.2: This environment is just random enough to force the robot to find a pathway to the goal.

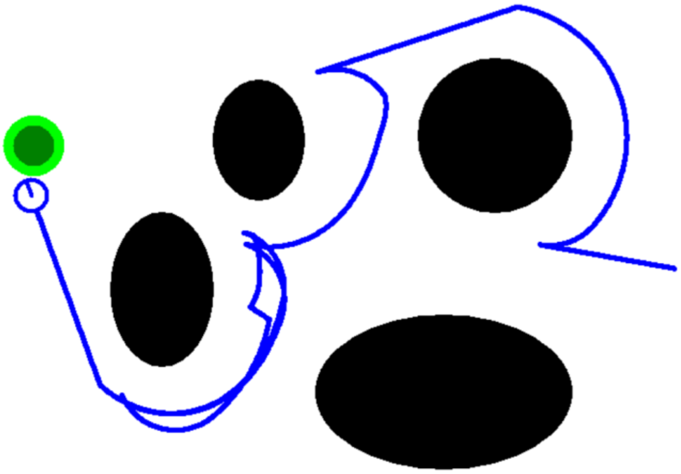


Figure 10.4: The algorithm seems to work.

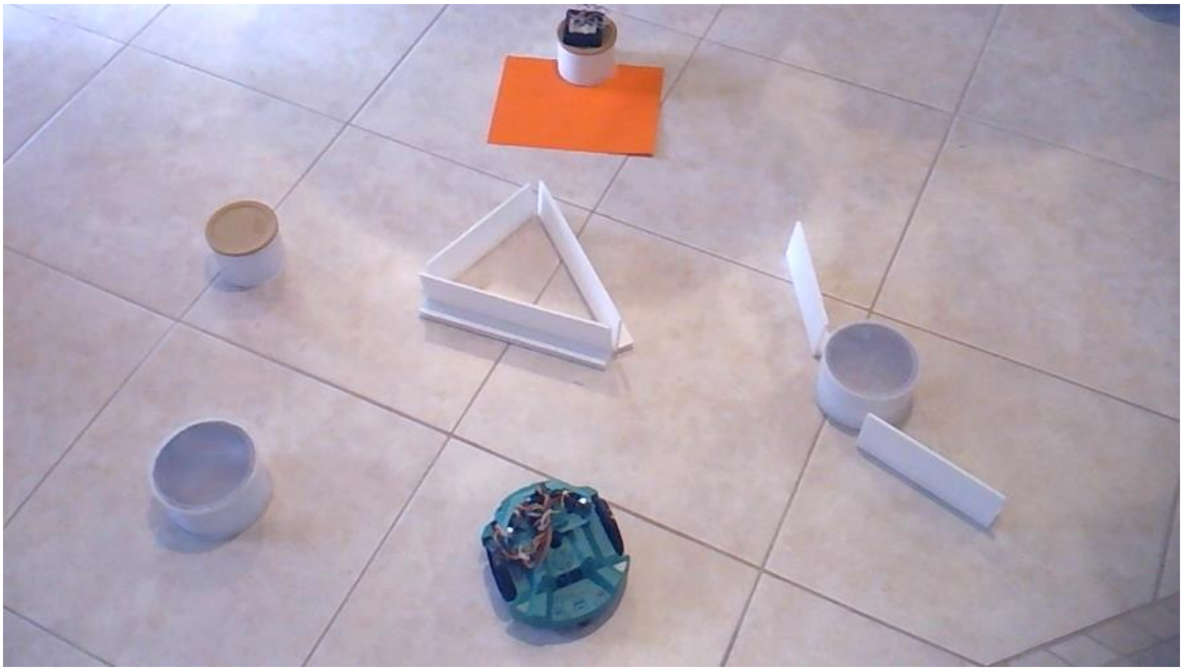


Figure 10.9: This is a sample environment for beacon detection.

Chapter 11

Combining Behaviors: Avoiding Objects on a Line

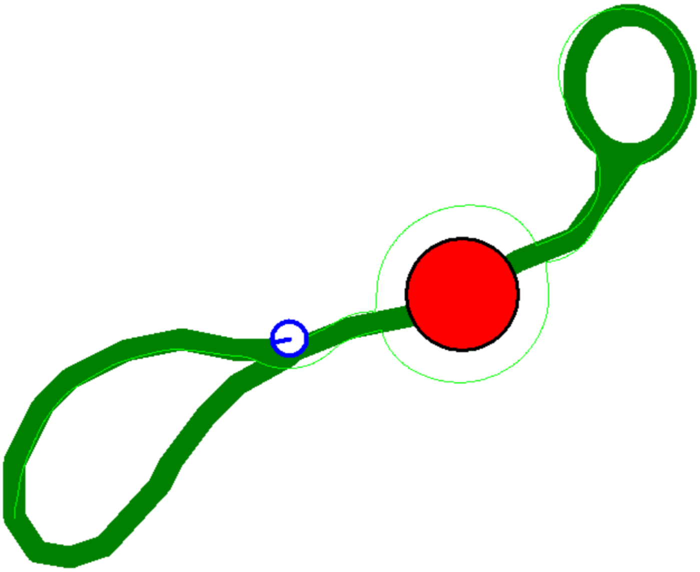


Figure 11.1: The robot avoids objects blocking its path on the line.

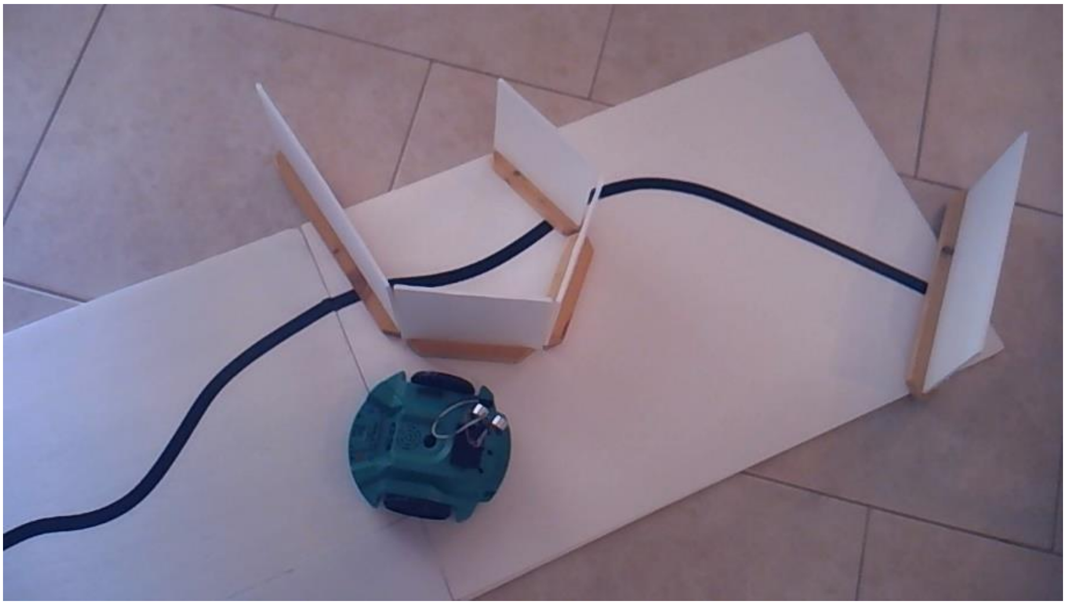


Figure 11.4: This is a reasonable environment for this chapter's challenge.

Chapter 12

Autonomous Navigation

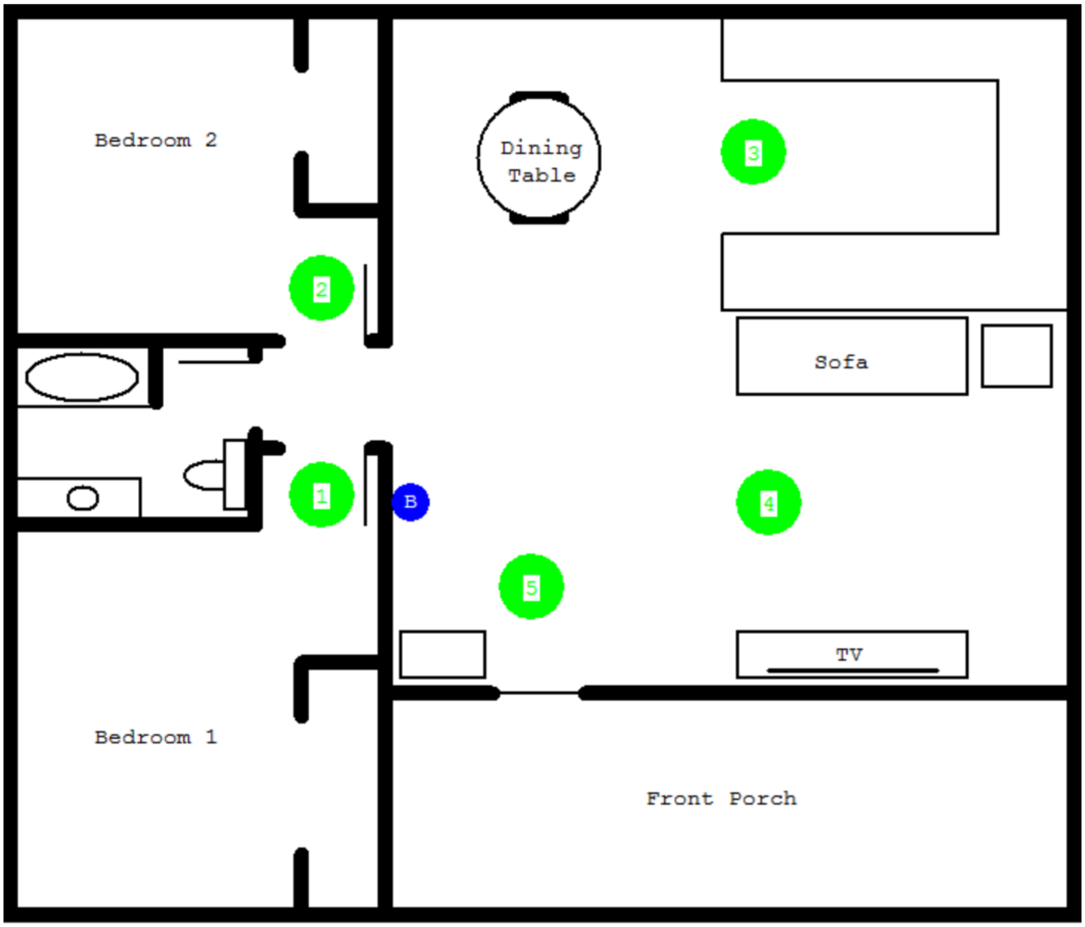


Figure 12.1: A realistic environment for the simulated robot.

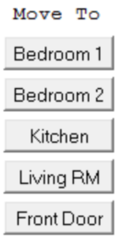


Figure 12.2: These buttons control the robot’s movements.

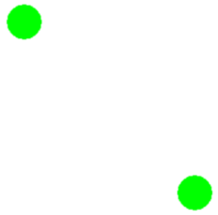


Figure 12.7: Assume we want the robot to travel between these two locations.

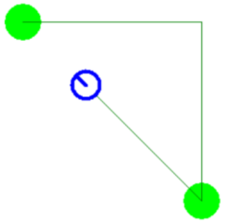


Figure 12.9: The robot is accurately moving back to its starting position.

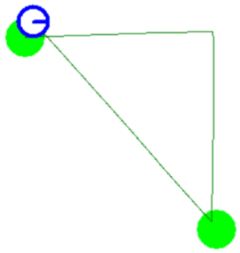
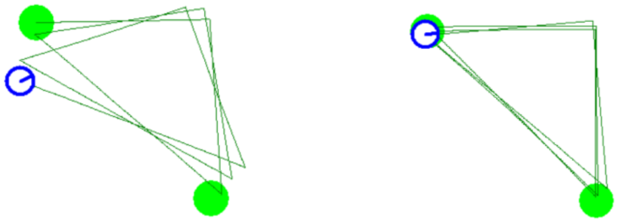


Figure 12.10: The movements are inaccurate when error is introduced.



3% error

1% error

Figure 12.11: Each new path is worse than the last.

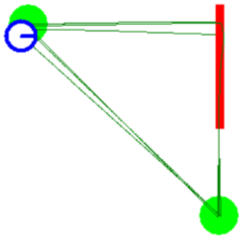


Figure 12.12: One method for correcting errors in the robot’s movements.

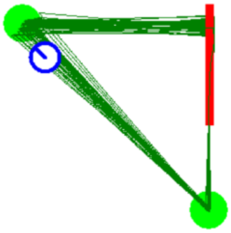


Figure 12.13: With a closed-loop movement, the robot constantly gets back on track.

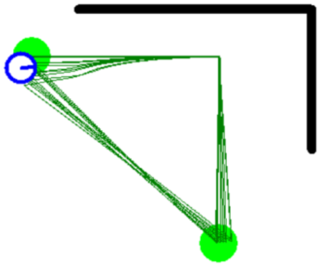


Figure 12.16: Following a wall can also re-align the robot.

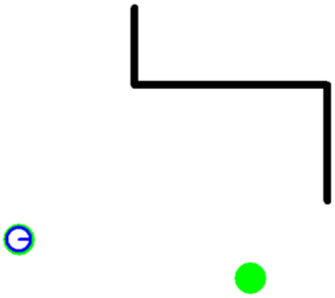


Figure 12.18: The robot can easily miss the corner without the beacon to guide it.

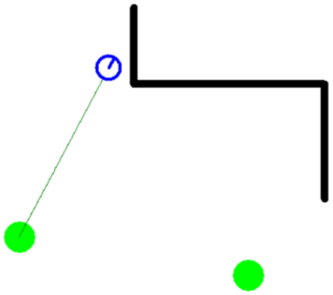


Figure 12.19: Sometimes even a small error can cause big problems.

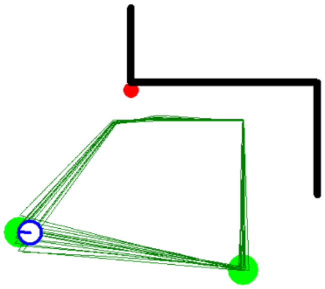
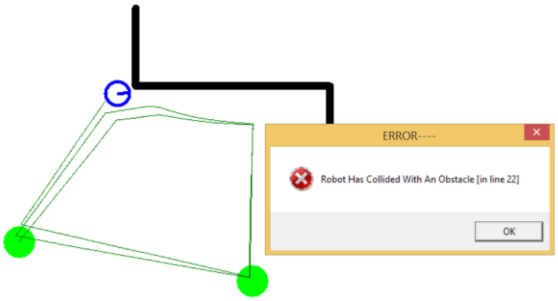


Figure 12.20: The beacon makes sure the robot arrives to the right of the corner.



```
Robot Has Collided With An Obstacle [in line 22]
```

Figure 12.21: Without the beacon, the robot’s error can prevent it from finding the wall.

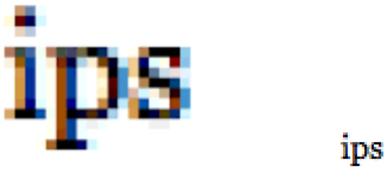


Figure 12.23: Extra pixels are added by windows to make characters appear smoother.

Appendix A

A RROS Overview and Advanced RROSS Options

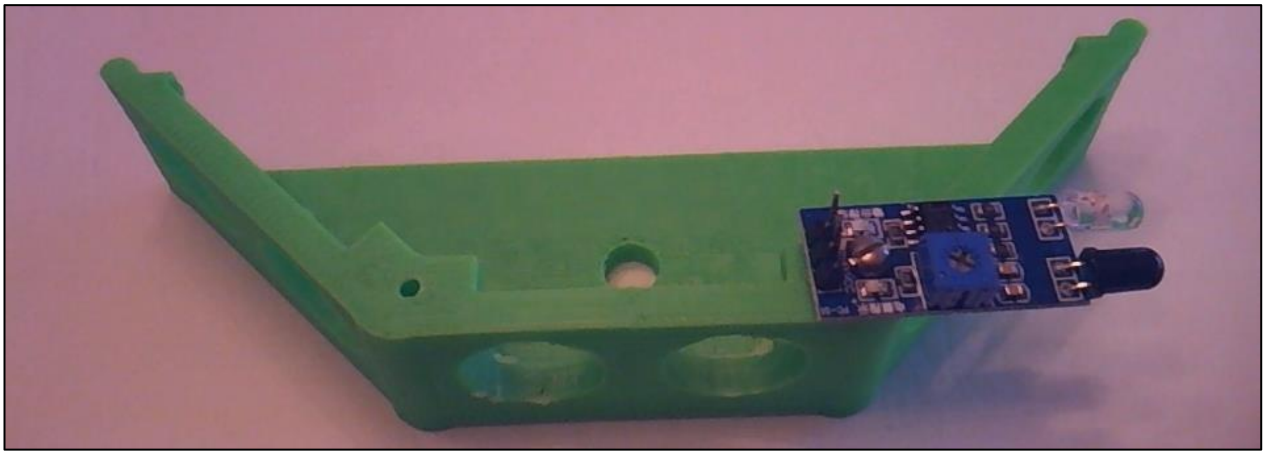


Figure A:3: Digital side sensors can be mounted on the corners of the Ping mount.

Appendix B

RobotBASIC Beacons

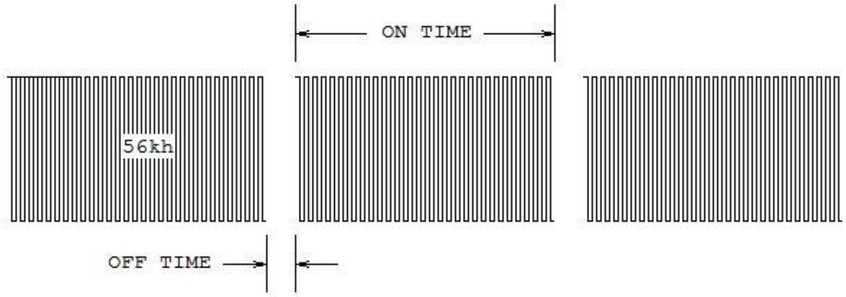


Figure B.1: Each beacon generates this waveform.

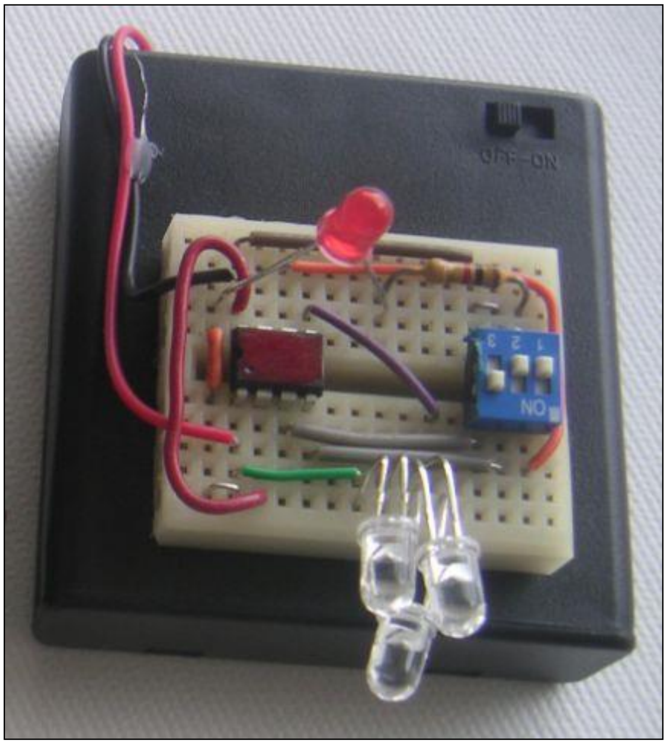


Figure B.2: Our preprogrammed chip makes it easy to create your own beacons.

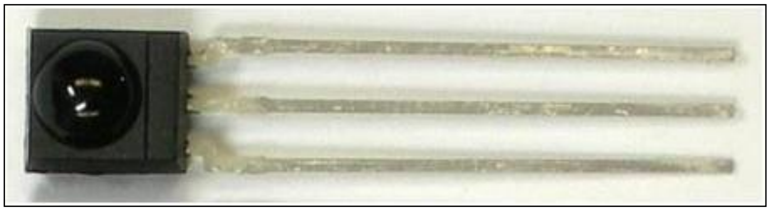


Figure B.3: This small device serves as a beacon detector.

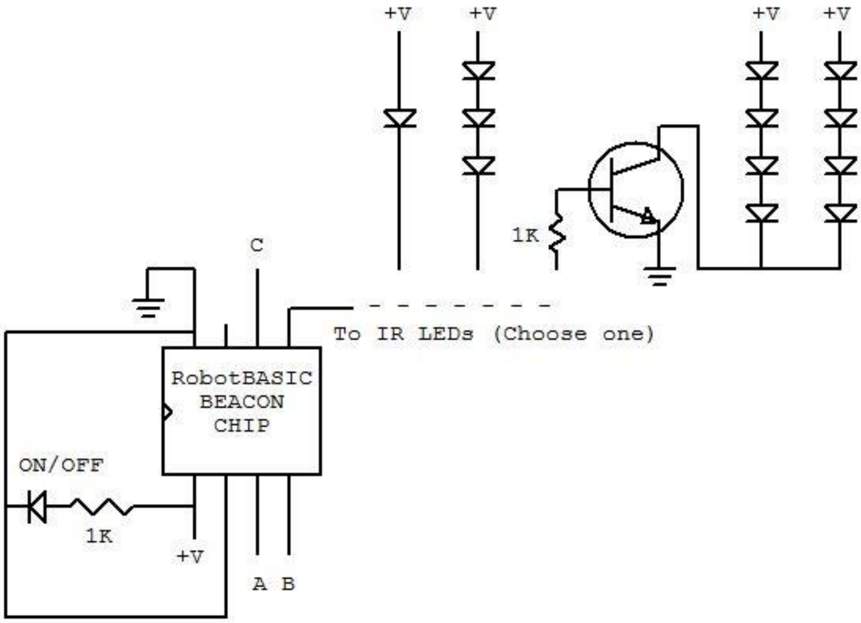


Figure B.4: The RobotBASIC Beacon Chip makes it easy to build your own beacons.

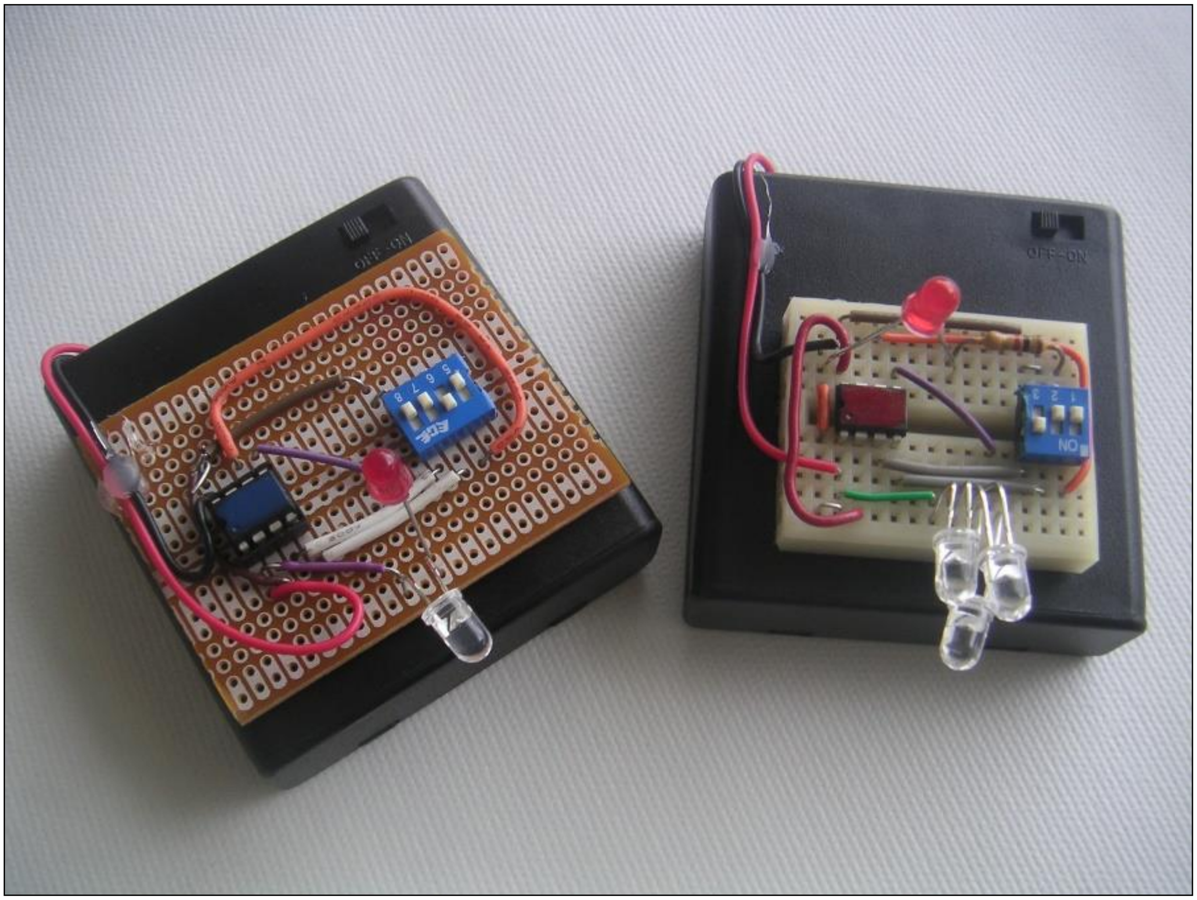
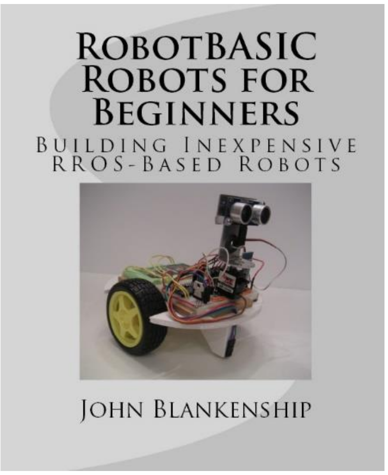


Figure B.6: Two methods of constructing beacons.

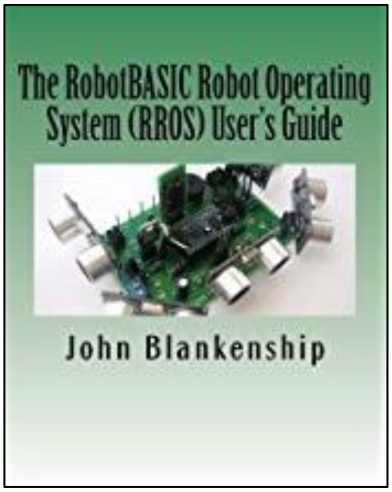
Appendix C

Resources

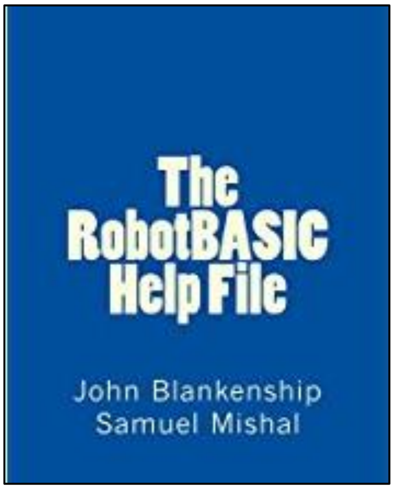
These are only some of our publications and the list is constantly growing. Search Amazon.com to see everything that is currently available.


ROBOTBASIC ROBOTS FOR BEGINNERS
BUILDING INEXPENSIVE RROS-BASED ROBOTS
JOHN BLANKENSHIP

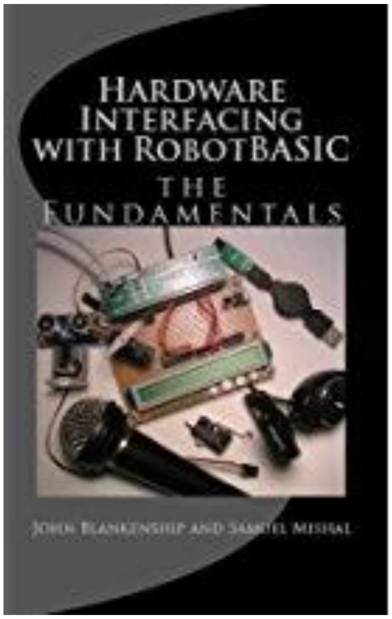
If you want to build your own robot (something similar to the S3 but far less expensive) this is what you need. It provides step-by-step details on both construction and programming.



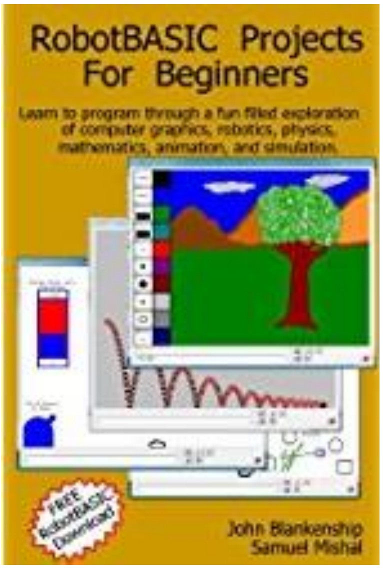
The RROS Manual is free for download on the RROS tab at www.RobotBASIC.org, but a printed version is preferred by many readers. It provides details of everything the RROS can do as well as many example programs. It is not written for beginners, but supplies a massive amount of information about building RobotBASIC robots using the RROS.



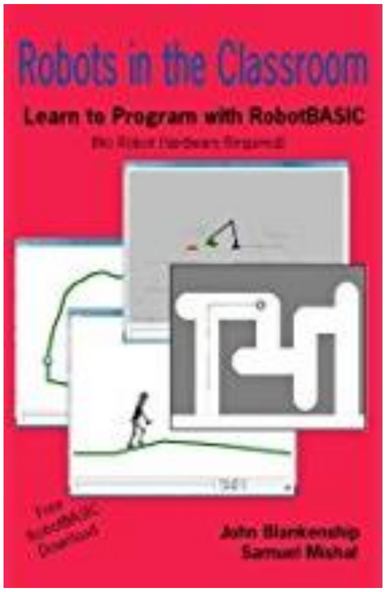
RobotBASIC's help file is integrated into the language, but a printed version is available to those wanted the convenience to read it anywhere at any time. You could print out the help file, but it will need a lot of reformatting and probably use an entire ink cartridge.


HARDWARE INTERFACING WITH ROBOTBASIC
THE FUNDAMENTALS
JOHN BLANKENSHIP AND SAMUEL MISHAL

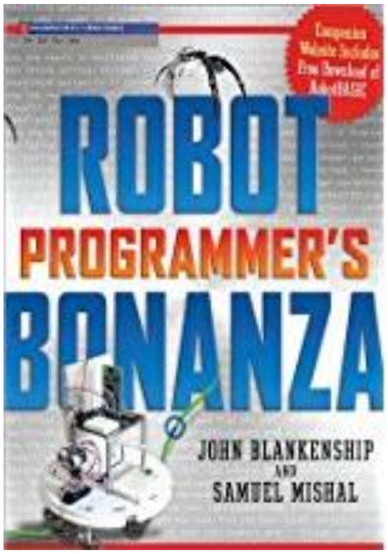
If you want to learn more about hardware and how sensors and motors can be controlled from RobotBASIC programs, then this is a good place to start. You will also be introduced to vision, speech, voice recognition and interfacing with microprocessors.


RobotBASIC Projects For Beginners
Learn to program through a fun filled exploration of computer graphics, robotics, physics, mathematics, animation, and simulation.
FREE RobotBASIC Download
John Blankenship
Samuel Mishal

This book will teach you about programming by writing simple animations, simulations, video games, and other graphical applications. If you are a beginner and want to learn about programming, this book is our number one recommendation. The projects will make you love programming.

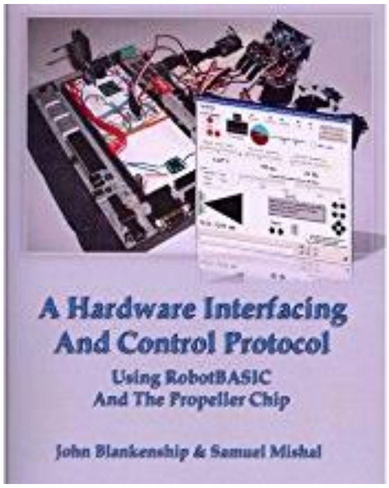


This book teaches you about programming like the above book, but all of the example programs are robotic related.



This book provides more details about developing algorithms for robotic behaviors than any RobotBASIC publication. Nearly everything is done with the simulator so learning how to program a robot will never be easier or less expensive.

This is one of our most advanced books and definitely not for beginners, but if you really want to learn how to build a real robot, this is for you. Arlo is a mobile robot with two arms, vision, speech, voice recognition, an animated face, and much, much more. It is built on the RROS system so S3 users should find it an easy transition to a much more complex robot. Many of the principles discussed for Arlo will apply to the S3.



This is another of our advanced books. It provides programs that interface with a variety of sensors and the Parallax's Propeller processor (used internally for the S3). If you are a Propeller fan, you need this book.

Hopefully these resources will help you expand your skills and knowledge of robotics. Learning to program can be fun and exciting but it also develops critical thinking and problem solving skills that can carry over into many aspects of your life and help prepare you for careers in programming, engineering, and mathematics. If you love science and technology, you should learn to program.