RidgeSoft™

Programming Your Robot to Perform Basic Maneuvers

www.ridgesoft.com

Revision 1.0

Introduction

This tutorial will teach you how to program the IntelliBrain™-Bot educational robot to perform basic maneuvers. You will also learn how to use the RoboJDE™ API documentation and the *IntelliBrain User Guide* to guide your IntelliBrain programming efforts. In addition, you will learn about controlling hobby servos from your Java program.

Although this tutorial is based on the IntelliBrain-Bot, the maneuvering techniques you learn can be applied to just about any two-wheeled differential drive robot.

Before You Get Started

If you are not already familiar with writing robotics programs using RoboJDE™ we recommend you start with the tutorial: *Creating Your First IntelliBrain Program*, which is available from the RidgeSoft web site, www.ridgesoft.com.

Maneuvering Your Robot

What is a Differential Drive Robot?

Many mobile robots, including the IntelliBrain-Bot educational robot, are based on a two wheeled differential-drive design. A robot based on this design uses two independently powered wheels to enable it to steer. As you can see in Figure 1, by controlling the direction of each of your robot's two main drive wheels, you can make your robot perform simple maneuvers. If you program your robot to turn both of its wheels in the forward direction, it will move forward. If you program your robot to turn its wheels in opposite directions, it will rotate clockwise or counterclockwise, depending on the direction the wheels turn.

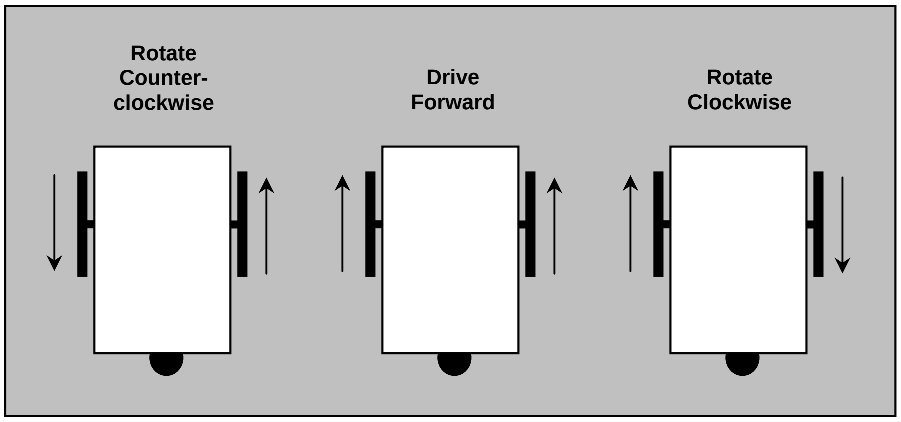


Figure 1 - Maneuvering a Differential Drive Robot

Driving Your Robot's Wheels

Many mobile robots, like the IntelliBrain-Bot educational robot, use hobby servo motors to drive their wheels. The examples in this tutorial are based on a robot powered by servos.

DC motors are another popular way to drive a robot's wheels. We not will discuss using DC motors in this tutorial, although, the maneuvering techniques we discuss can be used equally well for a DC motor powered robot. If you intend to use small DC motors to power your robot, you can interface to them and control them using the IntelliBrain Expansion Board.

Standard hobby servo motors have built-in control circuitry and mechanics designed to rotate the servo's output shaft to a specific position and hold that position. However, it has become common to modify servos for continuous rotation. By removing mechanical stops and disabling position sensing circuitry, you can make many hobby servo motors into continuous rotation servo motors. In fact, it has become so common, that you can now buy servos manufactured for continuous rotation, so you don't have to modify them yourself. The servos included in the IntelliBrain-Bot kit are continuous rotation servos.

Controlling Servos

While continuous rotation servos can power your robot's wheels – just like a conventional DC motor – you still have to control them using the servo's control circuitry. Servo control circuitry is designed to allow the position output shaft to be set and maintained at that position. However, when servos have been modified for continuous rotation the circuitry can be used to control the power supplied to the output shaft and its direction. You will have to learn a bit more about controlling servos before you can start programming your robot to maneuver.

Hobby servos are controlled by periodically pulsing the servo's control signal, as shown in Figure 2. The control signal is typically the white or yellow wire attached to the servo. You can think of a pulse as switching the power on momentarily, then switching it back off. A 1 millisecond pulse commands the servo to position its output shaft at one end of its range (0%). A 2 millisecond pulse commands the servo to position the output shaft at the other end of its range (100%). By varying the pulse duration between these two extremes, the shaft can be moved to an intermediate position.

When you send a control pulse to a continuous rotation servo, rather than moving the shaft to a particular position and holding that position, the pulse instead controls the power (more accurately, the torque) and direction of the shaft. Therefore, when you command the servo to the 0% position with a 1 millisecond pulse it actually results in the servo applying maximum power in the reverse direction. Similarly, when you command the servo to the 100% position it applies maximum power in the forward direction. If you command the servo to the 50% position it applies zero torque.

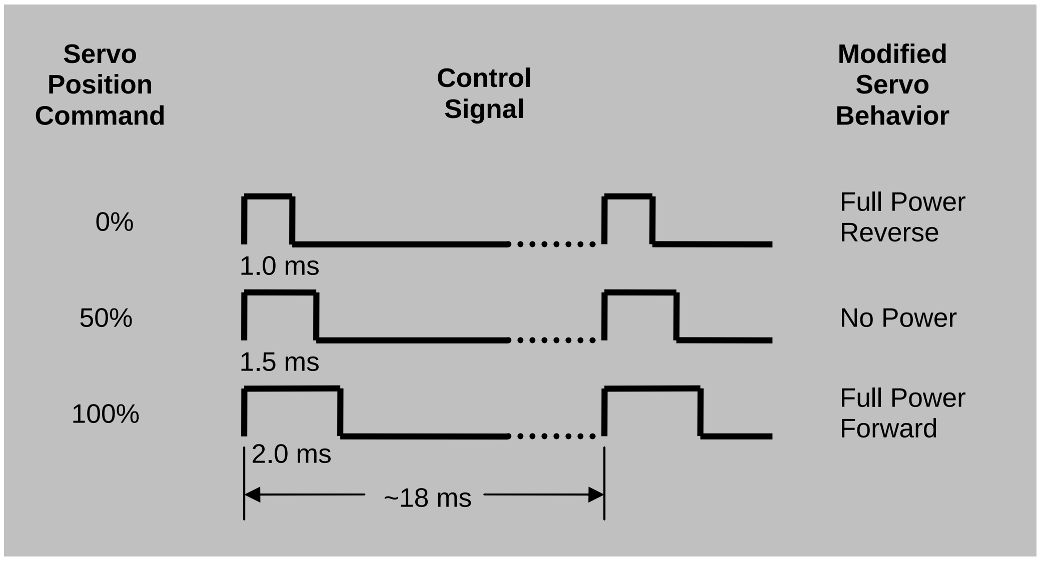


Figure 2 - Servo Control Signaling

Fortunately, the details of generating servo control pulses are managed for you by RoboJDE. You only need to be concerned with setting the power and direction of the servo. You do this by giving the servo position commands which, for a continuous rotation servo, actually control its power and direction.

Programming Your Robot

Let's start by programming the robot to go straight ahead for five seconds. The first thing you will need to do is create a software project in RoboJDE. This step is described in the tutorial titled *Creating Your First IntelliBrain Program.* We will use the program from that tutorial as our starting point. If you haven't already created that program, now would be a good time to do it.

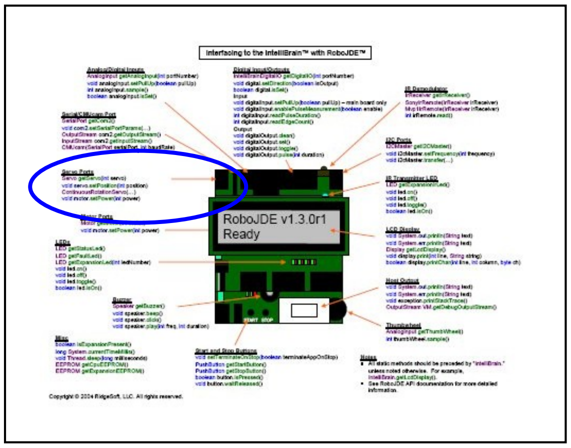
If you didn't have this tutorial to guide you, you would first need to determine how to command the servo motors from your program. In addition to teaching you how to program the robot to maneuver, this tutorial will also guide you through the process of using the documentation to facilitate programming the robot.

Obviously, from the discussion in the previous section, you will need to determine how to command the servo motors from your Java program in order to make your robot maneuver. The first steps in doing this is to consult the IntelliBrain User Guide and the API documentation.

Learning How to Use IntelliBrain Features from the Documentation

Take a moment now to find the section in the *IntelliBrain User Guide* regarding the main board servo ports. These are the ports the IntelliBrain-Bot uses to power and control the two servos it uses to drive its wheels. From the programming example in the user guide, you can see that your program can obtain a Servo object for a particular port by using the “getServo” method in the “IntelliBrain” class.

A second way that you can learn where to start with an IntelliBrain feature to use is to view the quick reference document (IntelliBrainAPI.pdf), shown in Figure 3. Take a moment to locate the quick reference document and identify information regarding programming the servo ports.


Interfacing to the IntelliBrain™ with RoboJDE™

Analog/Digital Inputs
AnalogInput getAnalogInput(int portNumber)
void analogInput.setPullUp(boolean pullUp)
int analogInput.sample()
boolean analogInput.isSet()

Serial/CMUcam Port
SerialPort getCom2()
void com2.setSerialPortParams(...)
OutputStream com2.getOutputStream()
InputStream com2.getInputStream()
CMUcam(SerialPort serialPort, int baudRate)

Servo Ports
Servo getServo(int servo)
void servo.setPosition(int position)
ContinuousRotationServo(...)
void motor.setPower(int power)

Motor Ports
Motor [illegible]
void motor.setPower(int power)

LEDs
LED getStatusLed()
LED getFaultLed()
LED getExpansionLed(int ledNumber)
void led.on()
void led.off()
void led.toggle()
boolean led.isOn()

Buzzer
Speaker getBuzzer()
void speaker.beep()
void speaker.click()
void speaker.play(int freq, int duration)

Misc
boolean isExpansionPresent()
long System.currentTimeMillis()
void Thread.sleep(long milliseconds)
EEPROM getCpuEEPROM()
EEPROM getExpansionEEPROM()

Copyright © 2004 RidgeSoft, LLC. All rights reserved.

Digital Input/Outputs
IntelliBrainDigitalIO getDigitalIO(int portNumber)
void digital.setDirection(boolean isOutput)
boolean digital.isSet()
Input
void digitalInput.setPullUp(boolean pullUp) – main board only
void digitalInput.enablePulseMeasurement(boolean enable)
int digitalInput.readPulseDuration()
int digitalInput.readEdgeCount()
Output
void digitalOutput.clear()
void digitalOutput.set()
void digitalOutput.toggle()
void digitalOutput.pulse(int duration)

Start and Stop Buttons
void setTerminateOnStop(boolean terminateAppOnStop)
PushButton getStartButton()
PushButton getStopButton()
boolean button.isPressed()
void button.waitReleased()

IR Demodulator
IrReceiver getIrReceiver()
SonyIrRemote(IrReceiver irReceiver)
Mvp1IrRemote(IrReceiver irReceiver)
int irRemote.read()

I2C Ports
I2CMaster getI2CMaster()
void i2cMaster.setFrequency(int frequency)
void i2cMaster.transfer(...)

IR Transmitter LED
LED getExpansionIrLed()
void led.on()
void led.off()
void led.toggle()
boolean led.isOn()

LCD Display
void System.out.println(String text)
void System.err.println(String text)
Display getLcdDisplay()
void display.print(int line, String string)
boolean display.printChar(int line, int column, byte ch)

Host Output
void System.out.println(String text)
void System.err.println(String text)
void exception.printStackTrace()
OutputStream VM.getDebugOutputStream()

Thumbwheel
AnalogInput getThumbWheel()
int thumbWheel.sample()

Notes
- All static methods should be preceded by "IntelliBrain." unless noted otherwise. For example, IntelliBrain.getLcdDisplay().
- See RoboJDE API documentation for more detailed information.

Figure 3 - IntelliBrain API Quick Reference

Finally, the RoboJDE class library API (Application Programming Interface) documentation contains the most detailed information on programming your robot. It is essential that you become familiar with using the API documentation because you will need to refer to it frequently as you program your robot. The RoboJDE API documentation is in standard Javadoc format. This is the format used to document most Java APIs. Becoming proficient at using the RoboJDE API documentation will make you proficient at using similar documentation for other Java programming projects.

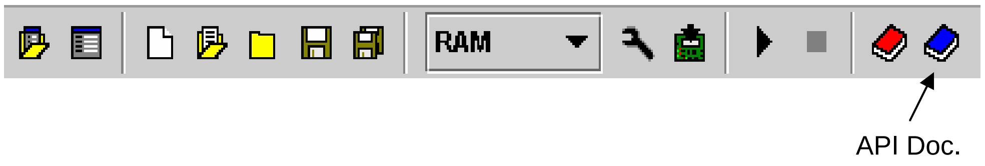
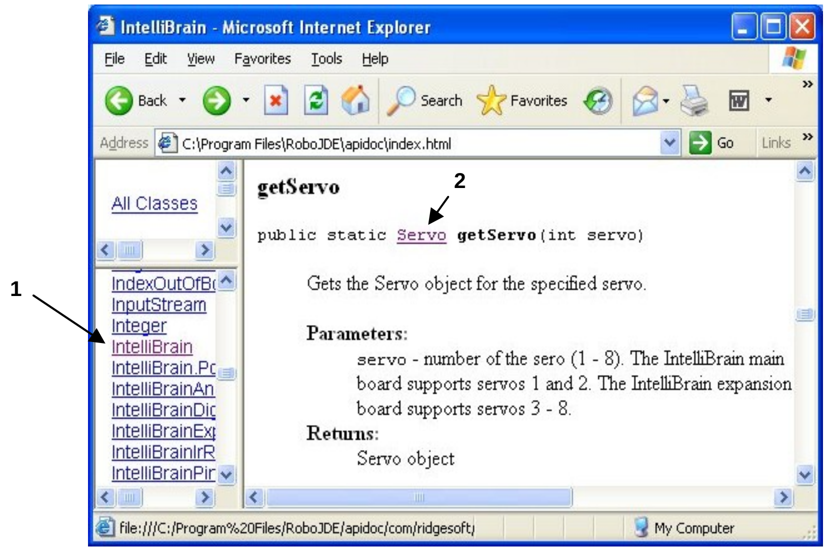


Figure 4 - API Documentation Button on RoboJDE Tool Bar

Click the API documentation button on the RoboJDE tool bar (shown in Figure 4) to display the API documentation in your web browser. This will launch your web browser to display the documentation, as shown in Figure 5.

Normally, when you are learning to use a new feature of the IntelliBrain controller, the best place to start learning about that feature's API is the documentation of the IntelliBrain class. You can display the IntelliBrain class's documentation by scrolling to and clicking on the class name in the list of classes, as indicated by reference 1 in Figure 5. This will display documentation for the class in the pane on the right hand side. By browsing through the class documentation, you can find the methods to access various features of the IntelliBrain controller.


IntelliBrain - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address C:\Program Files\RoboJDE\apidoc\index.html
Go
Links
All Classes
IndexOutOfB
InputStream
Integer
IntelliBrain
IntelliBrain.Pc
IntelliBrainAn
IntelliBrainDic
IntelliBrainExp
IntelliBrainIrR
IntelliBrainPir
1
2
file:///C:/Program%20Files/RoboJDE/apidoc/com/ridgesoft/
My Computer

getServo

```
public static Servo getServo(int servo)
```

Gets the Servo object for the specified servo.

Parameters:
servo - number of the sero (1 - 8). The IntelliBrain main board supports servos 1 and 2. The IntelliBrain expansion board supports servos 3 - 8.
Returns:
Servo object

Figure 5 - IntelliBrain API Documentation

Interfacing to IntelliBrain Servo Ports

Since you must learn how to control servos from your program in order to make your robot move, you will need to use the documentation for the IntelliBrain class to find a method related to using the servo ports. Browse through the IntelliBrain class documentation until you find a method related to the servo ports.

You should have found the documentation for the getServo method (shown in Figure 5). By reading this documentation you will learn this method returns a Servo object for the servo port number you specify. Noting that the left wheel's servo is attached to servo port 1 and the right wheel's servo is attached to servo port 2, you can now extend your program to obtain a servo object for each servo.

Add the following two lines to your program after the line that displays the program version number:

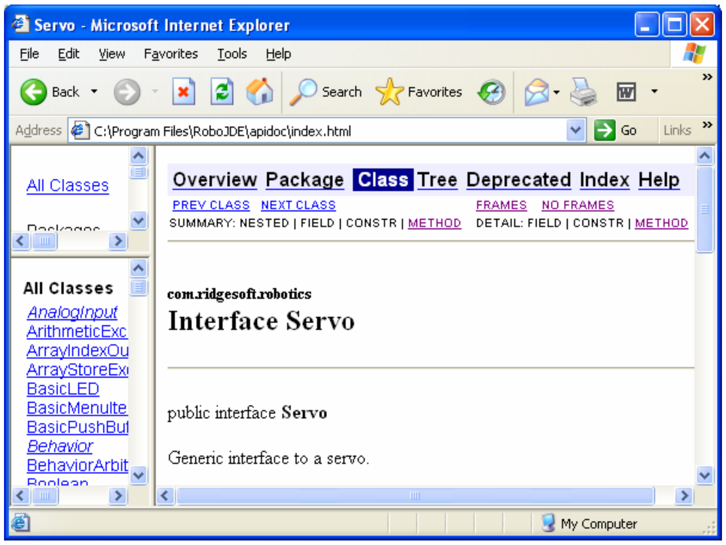
```
    Servo leftServo = IntelliBrain.getServo(1);
    Servo rightServo = IntelliBrain.getServo(2);
```

Since your program now refers to another type of object, Servo, you must also add an import statement at the top of your program. This statement is required to enable the Java compiler to find the Servo class. The import statement provides the full name of the class. The full name of a class includes its "package" name which is used to avoid problems that could arise with duplicate class names.

You will need to refer to the documentation for the Servo class to find its location (package name) in the library. Do this by clicking on the Servo link in the documentation for the getServo method, as shown by reference 2 in Figure 5. The documentation for the Servo class will be displayed, as shown in Figure 6. (It turns out Servo is an interface, not a class, but you don't need to be concerned about the distinction for the purposes of this tutorial.)

By viewing the documentation you can see that the Servo interface is in the package com.ridgesoft.robotics in the class library. Therefore, add the following import statement to your program just after the existing import statement:

```
    import com.ridgesoft.robotics.Servo;
```


Servo - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Program Files\RoboJDE\apidoc\index.html Go Links
All Classes
Packages
Overview Package Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
All Classes
AnalogInput
ArithmeticExc
ArrayIndexOu
ArrayStoreEx
BasicLED
BasicMenuIte
BasicPushBu
Behavior
BehaviorArbit
Boolean
My Computer

com.ridgesoft.robotics
Interface Servo

public interface **Servo**

Generic interface to a servo.

Figure 6 - Servo Interface Documentation

Programming Your Robot to Drive Forward

Your program now has objects – leftServo and rightServo – that enable it to control the servos. Your next step is to consult the documentation for the Servo interface to determine how to you use these objects to control the servos.

Browse the API documentation for the Servo interface to see what methods it provides. You will see the interface provides two methods, setPosition and off. As the documentation indicates, the off method turns the servo off and the setPosition method sets the servo's position between 0 and 100.

Recalling our previous discussion, the position setting command for a servo which has been modified for continuous rotation actually sets its power and direction. Setting the position to 0 corresponds to full reverse power. Setting the position to 100 corresponds to full power forward, and setting the position to 50 corresponds to no power. Note: setting the power to 50 is not the same as calling the off method. The off method stops sending command pulses to the servo, whereas setting the power to 50 sends command pulses to the servo.

You must command both servo motors to apply power in the forward direction to cause your robot to move forward. You can do this by turning each servo on at full power. However, you also need to take into account that the servos are mounted opposed to each other, which results in the sense of the commands being opposite for the two servos.   In order to make both wheels rotate in the forward direction, you must command the left servo to rotate counterclockwise while commanding the right servo to rotate clockwise. To move the robot forward at maximum speed, you must set the left servo position to 100 and the right servo position to 0. After setting these power levels the robot will drive forward until you command it to stop by calling the off method for each servo.

Add the following lines to your program to program your robot to move forward for 5 seconds:

```
     leftServo.setPosition(100);
     rightServo.setPosition(0);
     Thread.sleep(5000);
     leftServo.off();
     rightServo.off();
```

You can now test your program by building and downloading it to your robot. Once you have the program loaded into your robot, disconnect the serial cable, set the robot on the floor and press the START button. Your robot should drive straight ahead for 5 seconds.

Programming Your Robot to Rotate

Now that you know how to program your robot to move forward, getting it to rotate in place is trivial. All you have to do is make the wheels go in opposite directions instead of the same direction. Add the following statements to make your robot rotate clockwise:

```
     leftServo.setPosition(100);
     rightServo.setPosition(100);
     Thread.sleep(5000);
     leftServo.off();
     rightServo.off();
```

Add these statements to make your robot rotate counterclockwise:

```
leftServo.setPosition(0);
rightServo.setPosition(0);
Thread.sleep(5000);
leftServo.off();
rightServo.off();
```

Programming Your Robot to Drive in a Square

You can make your robot drive in a square simply by programming it to perform a sequence of moving forward and rotating 90 degrees four times in a row. You will need to adjust the sleep time during the rotation step so your robot rotates 90 degrees. For the IntelliBrain-Bot, changing the sleep duration to 625 should make the robot rotate approximately 90 degrees. However, the exact value depends on characteristics of your individual robot and the condition of the batteries. You can also adjust the size of the square by adjusting the sleep time in the forward movement step.

Rather than repetitively adding statements to the program to drive straight, turn right, drive straight, turn right, and so on, you can use a loop to reduce the number of statements in the program. Since a square has four identical sides joined by four identical angles you can program your robot to drive in a square by performing two steps: 1) drive forward and 2) turn 90 degrees. By repeating these steps four times, in a loop, your robot will drive in a square.

Add the following statements to program your robot to drive in a square:

```
for (int i = 0; i < 4; ++i) {
    // drive forward
    leftServo.setPosition(100);
    rightServo.setPosition(0);
    Thread.sleep(5000);

    // rotate clockwise approximately 90 degrees
    leftServo.setPosition(100);
    rightServo.setPosition(100);
    Thread.sleep(625);
}
leftServo.off();
rightServo.off();
```

Download and test your program. You will most likely find that your robot does not drive in a perfect square. This is primarily due to the angles of the square not being exactly 90 degrees. If the robot turns less than 90 degrees at each corner, increase the sleep time. If the robot turns more than 90 degrees at each corner, reduce the sleep time.

Conclusion

The key to maneuvering a differential-drive robot is controlling the direction and speed of the two wheels. By programming the robot to turn both wheels in the same direction, it will go forward. By programming your robot to turn its wheels in opposite directions, it will rotate in place. In addition, you can program your robot to perform other move maneuvers such as arcing left or right by applying different amounts of power to each wheel.

You can program your robot to perform more sophisticated maneuvers by performing timed sequences of basic maneuvers. By programming your robot to perform four repetitions of moving straight and turning 90 degrees it will drive in a square pattern.

As you experiment with your robot, you will observe that using time as the basis for navigation has its limitations. As the batteries drain, the behavior of your robot will change. Also, your robot's performance will vary depending on the surface it operates on. The difference in friction between different surfaces will have a significant effect on how your robot performs.

Unfortunately, as the control software for your robot is currently implemented, it provides no mechanism for the robot to account for variations in conditions, such as battery charge and friction, which affect its performance. The robot's control system is said to be "open loop" because it lacks "feedback" to account for variations in conditions which affect your robots performance.

You can improve the consistency and accuracy of your robot's maneuvering by adding sensors to provide feedback to its control software. One way you can do this is by adding wheel encoders that sense the position of each wheel, which is a topic we will leave for another tutorial.

You have also learned about controlling hobby servos. You can use servos to control the position of an arm or other device. Servos can be modified for continuous rotation to be used like conventional motors. Servos are commonly used for robotics applications because they are inexpensive, lightweight, easy to mount and easy to control.

Finally, this tutorial demonstrated how to use the class library API documentation and the *IntelliBrain User Guide* to learn how to program your robot. By using the documentation you can learn how to access features of the IntelliBrain controller you have not previously used.

Exercises

1. Browse to the documentation on the getServo method of the IntelliBrain class in API documentation.

2. Find the getServo method in the IntelliBrain API quick reference.
3. Locate the discussion of using servo ports in the *IntelliBrain User Guide*.
4. Find documentation on the Servo interface in the class library API documentation.
5. Program your robot to drive straight ahead for exactly 12 inches. Using the same program, run the robot on a different surface. How far did it go? Why does it not go the same distance on all surfaces?
6. Program your robot to rotate exactly 180 degrees. Using the same program, run the robot on a different surface. How far did it rotate? Why does it not rotate the same amount on all surfaces?
7. Program your robot to navigate in a triangle or some other shape you choose.
8. Program your robot to arc to the left or right as it moves forward by applying a different amount of power to each wheel. What happens when you increase or decrease the difference in power?
9. Program your robot to navigate a particular shape, then change the batteries to a set of batteries that has more or less charge remaining. How does this affect the navigation?