Learning Roomba: Student’s Guide
Module 1 Robotics Introduction

Drew Housten (dhousten@gmail.com)
1 Introduction

Welcome to Learning Roomba. Unlike many of your other classes, this should be fun and exciting. Why? Because you get to control robots. Robotics can sound complicated, but you can easily have a robot do some tasks. Through these modules, you will get a chance to solve problems using a robot and have it perform seemingly complicated behaviors. This Module is an introduction to the topic and will get you started with the development environment.

2 What is a Robot?

There are many different definitions of a robot. One definition is simply a machine that helps humans. There are many, many machines that fall under that definition. For example, a toaster would then be a robot. However, if you ask people, most would not consider a toaster a robot. So, the definition needs to be adapted a little. Another definition is slightly more complicated: a machine that intelligently helps humans. The problem with this definition is that it is not exactly clear. What makes a machine intelligent? We do not have a robot IQ test to verify that it meets the requirements. Some people say that the robot must be able to think and act like a human. If so, there are very few machines that even come close to meeting that requirement.

Unfortunately, there are no definitions that are exact. Instead, roboticists (people that work with robots) rely on a working definition. The definition that is commonly used, and the one used by these materials, is an artificial device that senses its environment, uses the information to make decisions, and then acts in the environment. How the robot does this highly varies. Even the environment that the robot works in is not consistent. However, notice that this definition is not perfect. Would you consider a television a robot? Many people would answer no. But, the television is an artificial device that senses its environment (remote control signal), makes decisions (decides what to do based on the remote control code), and acts (changes the channel, volume, settings, etc.). Or, consider a remote controlled airplane. If the airplane is not a robot, then consider many current Unmanned Aerial Vehicles (UAVs). Most current UAVs are piloted by a human remotely, so they are really expensive remote controlled airplanes. But, based on our working definition, both the remote controlled airplanes as well as the UAVs are considered robots. They sense the environment (wind speed, pitch, remote signals, etc.), decide how to act based on the signals, and act (change wing flaps, propeller velocity, etc.).

Robots do not necessarily need to look or act like a human. Robots that do so (See Figure 1 and Figure 2) are called anthropomorphic, which means artifacts with human features or a man-like appearance. Other examples of robots that are not anthropomorphic include the Sony AIBO robot-dog (Figure 3), iRobot Roomba, and industrial robots (Figure 4). These are just a few exam-
There is a big difference between robots that occur in science fiction and those that are real, working systems. Robots in science fiction movies and books, such as those that occur in *Star Wars*, *Star Trek*, *I, Robot*, and *Lost In Space*, tend to have human-like intelligence and reasoning abilities. In reality, robots are not yet at that point. Technology is quickly advancing, so reality and science fiction may shortly merge, but at the moment, it is not there yet. The robot that you will be working with is a real robot rather than a science fiction robot.
The robot that you will be using is an iRobot Roomba (Figure 5). The Roomba was originally created as a robotic vacuum cleaner. It will clean the floors by semi-intelligently navigating around the room and then going to a dock to charge itself. The human just needs to tell it to clean and then it does it without any other human interaction. It has sensors to detect walls, obstacles, dirt, virtual walls, and cliffs, uses that information to decide how it should proceed in the environment, and acts by turning on its vacuuming motors and controlling the drive motors on the wheels.

You are going to be using the Roomba in a slightly different way. You will be writing programs that make the decisions about how to act. Your programs will have access to all the same sensors and motors, but how the robot moves is up to you.

4 Building a robot

A robot generally includes mechanical, electrical, and software components, bringing together people with different backgrounds and interests. The mechanical components include the physical structure of the robot. Without mechanical engineers, the robot would not have a shape, would not have a physical structure, and would not stay together. The electrical components include the sensors, power systems, motors, and computers. Without electrical engineers, the sensing and acting capabilities would not exist, and, based on our work-
ing definition of a robot, the robot would no longer be a robot. The software components include the components that use the computer to make decisions about how to act. Without computer scientists, the robot would not have decision making capabilities and would be nothing more than a simple machine. In these Learning Roomba modules, you will be building the software for the Roomba.

Computation is a process of sending instructions to a computer processor. The computer processor will execute those instructions which will cause some outcome. In our case, the processor will send and receive electrical signals to and from the robot’s motors and sensors. The software that you write will perform some computation. The sequence of these computations will (hopefully) cause the robot to exhibit intelligent behaviors.

5 Why Robotics can be Difficult

As mentioned in our definition, there is a big difference between real robots and science fiction robots. Making robots act like humans has proven to be a difficult task. Part of the reason for this is that nobody knows exactly how the human brain works. Consider the following seemingly simple task: go to the kitchen. Ask a human to do that and most people would not hesitate to be able to do so. What does it take for a robot to do that? Unless we specifically tell the robot, it does not have any concept of what a kitchen is. We could tell it that it is a room with food and appliances to cook the food. But, what is a room? food? appliances? Just formulating the task such that a robot can understand it is a difficult problem. Even when the robot understands the task, it has to know how to get there and where it currently is. To do so, it needs to have a map of the environment, build a map, or wander randomly until it reaches the kitchen. If it has a map, it also needs to know where it is, how to read the map, and how to navigate the map. What does the robot do if there is a closed door or other obstruction? What does it do if something on the robot breaks down?
This is just one example and demonstrates only a few of the challenges that roboticists face. Software is increasingly becoming intelligent, but the human brain is extremely complex.

6 Programming your Roomba

Programming is the act of writing software programs. We are going to step through writing and running your first program with the Roomba. The program will cause the Roomba to go forward for 3 seconds, stop, turn in place to the right for 5 seconds, and then stop.

1. Open BlueJ - BlueJ is the development environment that you will use to write your programs. To begin, open BlueJ. On MacOSX, click on the BlueJ icon on the dock (Figure 6). On Windows, find the BlueJ shortcut in the “Start” menu.

![Figure 6: Starting BlueJ from the MacOSX Dock](image)

2. Create a new Project - Projects are ways of organizing code files that are related. Go to the “Project -> New Project...” menu option (Figure 7). Type your name in the “File” field, choose “Desktop” in the dropdown list, and click the “Create” button. From this point on, we will use this project to write all code.

![Figure 7: Creating a new project in BlueJ](image)

3. Create a new Class - Classes are files that you will write programs in. To create a new class, click the “New Class...” button. Fill in the name of “MyRoombaProgram” and click the “Ok” button. A class file should appear in the interface (See Figure 8).
Figure 8: A new class was created in BlueJ

4. Write the program - Double-click on the newly created “MyRoombaProgram” class. Remove all the code that was written and copy the program listed in Figure 9. Instead of “localhost”, use the server address provided by your teacher. Instead of “Your_Name”, use your name. The file should look like Figure 10.

5. Compile the program - Compiling the program checks for errors and prepares it so that it can be executed. To do so, click the “Compile” button. If there were no errors, the bottom of the class file should show “Class compiled - no syntax errors”. If it did not compile, make sure that the code matches the example provided in Figure 9.

6. Run the program - Click on the Project window (should be in the background). Right-click (or ctrl-click if you do not have a 2 button mouse) MyRoombaProgram. A pop-up menu should appear (See Figure 11). Click on “void main(String [] args)”. A new window will appear. Click the “Ok” button. A new window will appear and will start to show the output of your program. If everything is setup properly, a Roomba will begin to move.

6.1 Simple Java Programming Introduction

The sample program in Figure 9 may look cryptic if you have not programmed before. Code is written in a Programming Language. A programming language is a set of rules that you must follow so that the code can be translated into a form that the computer processor can understand. The code example above is written in the Java programming language and is the language that we will use for all future exercises.
import roomba.roombanetwork.services.userservice.*;

public class MyRoombaProgram{
    public static void main(String [] args){
        UserService.setServerAddress("localhost");
        UserService.setName("Your_Name");

        Roomba roomba = new Roomba();
        roomba.forwardForTime(.3,3);
        roomba.turnForTime(.3,5);

        UserService.disconnect();
        System.exit(1);
    }
}

---

**Figure 9: Quick Roomba code example**

This is a very basic introduction to the Java programming language. The above example program is evaluated line-by-line.

1. import roomba.roombanetwork.services.userservice.*;

   This line tells the compiler which libraries to use. This will be the same for all exercises and does not need to change.

2. public class MyRoombaProgram{

   This line specifies the name of the class. This must match the class file-name and was set when we created the new class in BlueJ.

3. public static void main(String [] args){

   This line starts a method. A method is a group of commands that can be called together and a result can be returned. This method is special since its name is “main”. The main method is the method that is called when the class is executed. Methods and other blocks of code are enclosed in matching curly brackets (“{” and “}”).

4. UserService.setServerAddress("localhost");
Figure 10: MyRoombaProgram class file in BlueJ

```java
import roomba.roombawork.services.userService.*;

public class MyRoombaProgram{
    public static void main(String [] args){
        userService.setServerAddress("192.168.1.188");
        userService.setName("drew");
        Roomba roomba = new Roomba();
        roomba.forwardForTime(1,3);  
        roomba.turnForTime(3,5);  
        userService.disconnect();  
        System.exit(0);  
    }
}
```

Figure 11: Running a program in BlueJ
This line is a statement that is setting information about the Roomba server address. All statements must end with a semi-colon (";"). The server address is a way that your program can communicate with the server, which gets your program's messages to the Roomba. An argument is passed to the “setServerAddress” method of the “UserService” class. The argument is the address of the server and should be provided by your teacher.

5. UserService.setName("Your Name");

This line is another statement that is setting the service name. This is mainly used for debugging. Debugging is a process of finding problems in the program. The argument to the setName method should be your name.

6. Roomba roomba = new Roomba();

This statement creates a Roomba object. An object is an instance of a Class with information that is related to the object. Running this command will communicate to the server to find a Roomba that is not being used by anyone. If you instead want to connect to a specific Roomba, pass an argument specifying the name of the Roomba. For example, you would type the following line:

Roomba roomba = new Roomba("BobTheRoomba");

7. roomba.forwardForTime(.3,3);

This statement calls the method forwardForTime on the roomba object that we created on the previous line. The first argument is the speed to move forward at and the second argument is the time that the Roomba should travel for. This statement will be discussed in more detail in Module 3.

8. roomba.turnForTime(.3,5);

This statement calls the method turnForTime on the roomba object that we created. The first argument is the speed to turn at and the second argument is the time that the Roomba should turn for. This statement will be discussed in more detail in Module 3.

9. UserService.disconnect();

This statement disconnects the program from the server and releases control over the Roomba.

10. System.exit(1);

This statement causes the program to quit.
11. }
This closes the method "main" that was opened on line 3.

12. }
This closes the class “MyRoombaProgram” that was opened on line 2.

Other important programming concepts include print statements, comments, variables, and control blocks. They are described here.

• Print Statements: Print statements show some text on the console output when the program is running. They can be used for debugging purposes to know where the program is in its execution. To add a print statement, add a statement like:
  
  System.out.println("My Output");

  but replace “My Output” with the text that you want printed to the console.

• Comments: Comments are lines that are for the programmer’s use and ignored during execution. A comment is either a line that begins with two forward-slashes (“//”) or a block that begins with “/*” and ends with “*/”. An example of both types is shown here:
  
  // This creates a new Roomba object
  Roomba roomba = new Roomba();
  /* This is also a comment, but can be on multiple lines */

• Variables: Variables are identifiers that represent an object or information. In the example above, “roomba” is a variable that represents an instance of the “Roomba” class. The variable can be used to store information. You will see how these are used in other modules.

• If/Then Control Block: An if/then control block can be used to evaluate some condition and based on the result of the evaluation do one thing or another. Consider this example:
  
  if(roomba.getLeftBump() == true){
    System.out.println("Left Bump");
  } else{
    System.out.println("No Left Bump");
  }

  The method “getLeftBump” will return true if the Roomba that is represented by the object “roomba” has its left bumper sensor pressed. It will return false if the left bumper sensor is not pressed. If “getLeftBump” is
true, then the block will be executed and the text “Left Bump” will be displayed on the console output. If not, the else block will be executed and the text “No Left Bump” will be displayed on the console output.

- While loops: A while loop continues executing a block of code until a condition is met. An example is provided here:

```java
while (roomba.getLeftBump() == false) {
    System.out.println("No Left Bump");
}
```

As long as the Roomba’s left bumper sensor is not pressed, the text “No Left Bump” will repeatedly be printed to the console. As soon as the bumper sensor is pressed, the program will continue to the next statement.

If the above examples do not make sense, it is recommended that you look at other resources for more information on programming in Java.

7 Homework

Don’t run away! Homework can sound boring, but it can also be fun. For this homework assignment, find examples around you of robots. Some examples were provided, but what else are you familiar with? What tasks or responsibilities do they perform? How effective are they? Are they real or are they science fiction? Find a couple examples and answer those questions for each of them.