Institute for Aerospace Studies
UNIVERSITY OF TORONTO

# LABORATORY I

# Sensor & Actuator Programming

ROB301 Introduction to Robotics
Fall 2015

# 1 Introduction

This is the first lab of ROB301—Introduction to Robotics. A total of four single-session labs of progressing complexity are the lead-in to a two-session project. Labs I and II have no written deliverables; however, students must attend the sessions and demonstrate functionality to the TAs. Labs III and IV do require written reports in addition to the demonstrations to the TAs during the lab sessions. Each of these four labs are meant to be short, easy to implement, and designed to exercise important robotics concepts which are presented during the lectures. The project will require the teams to design and program a robot to complete an obstacle course while achieving simple tasks along the way using knowledge gained from the labs. A demonstration and report for the project are required.

Teams of three students will work together for each of the four labs and project. See the TA for an assignment of a team number, workstation, and LEGO kit.

# 2 Purpose

This lab strives to familiarize you with the equipment and software used for all labs as well as the project for ROB301. One of the most important aspects of robotics is the ability of the robot to interact with its environment, be that sensing its surroundings or moving around within it. By the end of this session, you should be able to write a simple Java program, upload it to the EV3 brick, send signals to drive all actuators, and receive information from all sensors.

# 3 Equipment & Software

The hardware platform used for ROB301 are LEGO Mindstorms EV3 kits. The benefit of these kits is their reliability and robustness. In engineering applications, one of the foremost struggles is powering, reading, and sending information to sensors and actuators. With the LEGO kits, all sensors and actuators are powered through the main EV3 Brick thus allowing for quick and easy implementation with minimal need for debugging.

The standard LEGO Mindstorms icon-based programming language is not powerful or flexible enough for the purposes of these labs. Instead, the EV3 Bricks have been reprogrammed with leJOS firmware allowing for programming in Java. The primary advantage of this is that it allows users to implement object-oriented programming quickly and easily.

Java shares some similar syntax to C++ but is different in that it does not need to be manually compiled into a machine-language. Rather, Java is implemented by a Java virtual machine which allows it to be interpreted directly. The result of this is that C++ is typically leaner and faster (ideal for high-computation applications) while Java tends to have few implementation dependencies and is very portable. Sample code will be provided to help get you started and links to resources for syntax details are available at the end of this document.

## 3.1 LEGO Mindstorms EV3 Kit

Each team will be assigned a LEGO Mindstorms EV3 Kit which will be numbered. Please do not remove the kits from SF4102 and please do not mix parts with kits from other teams. The following subsections highlight some of the included contents.

### 3.1.1 LEGO Parts

The LEGO Mindstorms EV3 Education Core Set comes with a total of 541 pieces (including the sensors, actuators, and Brick). In addition, a LEGO Mindstorms Expansion Pack will be provided for the project.

### 3.1.2 EV3 Brick

The EV3 Brick is the main controller for the labs and project. There are four output ports (labelled A-D) for actuators and four input ports (labelled 1-4) for sensors. The EV3 Bricks have been programmed to run leJOS firmware for this course. A charging cable is provided in the kit and can be plugged into the charge port found beneath the input ports when the internal rechargeable battery is low. To connect to a computer, use the provided USB cable where the microUSB connector plugs into the "PC" port found next to the output ports.


Figure 1: LEGO Mindstorms EV3 Brick

**Powering ON.** To turn ON the EV3 Brick, press the centre button. A red light will turn on behind the navigation pad and the leJOS software will begin the boot process. It may take up to 1.5 minutes to reach the main menu.

**Powering OFF.** To turn OFF the EV3 Brick, press the back button located beneath the lower left side of the screen, navigate using the left or right buttons to the "X" for no or "checkmark" for yes to shutdown. The EV3 Brick will then begin the shutdown procedure.

**Navigating the Menu.** At the main menu, the navigation pad (up, down, left, and right) allow you to navigate the options. Press the centre button to select and press the back button (located beneath the lower left side of the screen) to step back up a level in the menu. If you have programs installed, navigate to the "Programs" option and select using the centre

button to list the installed programs. Navigate to the desired program and select using the centre button to run it.

### 3.1.3 Actuators

There are a total of three actuators included in your kit: large servo motor (2) and medium servo motor (1). A brief description of these actuators follows.

Figure 2: Large Servo Motor and Medium Servo Motor

**Large Servo Motor.** This motor is meant for high-torque, low-speed applications. It is an ideal candidate for drive actuation. The maximum speed is $170$ rpm and a stall torque of $40\ \mathrm{N\,cm^{-1}}$.

**Medium Servo Motor.** This motor is meant for low-torque, high-speed applications. It is an ideal candidate for actuation requiring higher precision. The maximum speed is $250$ RPM and a stall torque of $12$ N/cm.

### 3.1.4 Sensors

There are a total of five sensors included in your kit: touch (2), ultrasonic (1), colour (1), and gyro (1). A brief description of these sensors follow.

Figure 3: Touch, Ultrasonic, Gyro, and Color sensors

**Touch.** This sensor is used to detect physical contact. Since it is essentially a push-release button, there are only two states: *depressed* and *not depressed*. This type of sensor is commonly placed at the front of mobile robotic platforms and is used as a bumper/detector.

**Ultrasonic.** This sensor is used to measure distance. It emits ultrasonic waves and then detects their echoes in order to calculate the distance to an object. Typical ranges are up to $250$ cm with $\pm 1$ cm accuracy. In addition to measuring distance, this sensor can also be used to detect other ultrasonic signals emitted from the environment.

**Color.** This sensor has three functions: *color measurement*, *ambient light measurement*, and *reflected light measurement*. In color measurement mode, the sensor can differentiate between seven different colors: none (0), black (1), blue (2), green (3), yellow (4), red (5), white (6), and brown (7). In ambient light measurement mode, the sensor measures the light intensity of the surroundings. In reflected light measurement mode, the sensor emits light from an LED and measures the amount reflected back from nearby obstacles which can be used to calculate the distance to an object. Typical ranges are up to $50$ cm with $\pm 1$ cm accuracy.

**Gyro.** This sensor has two functions: *measure angle* and *measure angular velocity*. Typical accuracies for the two modes are $\pm 3^\circ$ and $\pm 440^\circ\,\mathrm{s}^{-1}$, respectively. This sensor can be used to measure the incline a robot is traversing or to aid in stabilization of unstable platforms.

## 3.2 leJOS: Java for LEGO Mindstorms

The EV3 Bricks used throughout this course have been replaced by leJOS firmware. LeJOS runs a Java virtual machine on the EV3 Brick which permits Java programs to be uploaded via USB and run directly from the leJOS interface.

### 3.2.1 Eclipse

In order to prepare programs using the Java programming language, Eclipse, an integrated development environment (IDE), has been installed on the computer workstations provided. Eclipse can be accessed from the desktop of the ROB301 account on the computer workstation. Upon initiating Eclipse, the user is prompted to select a workspace for programs to be stored. The default is "C:\Users\ROB301\workspace"

### 3.2.2 Java Basics

Java is an object-oriented programming language where everything is an *Object* that has states and behaviours. A *Class* defines the template that describes what behaviours/states that objects within it may possess. To use an input or output, you must *import* it so that the

program will be able to access it. Future labs will explore the capabilities of Java for our EV3 Brick, but for now, the focus will remain on communicating with the screen, sensors, and actuators provided in your kit.

# 4 Task Programming

This section will walk you through writing your first leJOS program and uploading it to the EV3 Brick.

## 4.1 Getting Started

Follow these steps to create, write, and save your first program:

1. Open Eclipse from the desktop of your workstation. Select your workspace.
2. Select File > New > Project > leJOS EV3 > leJOS EV3 Project > Next
3. Type "HelloROB" as the Project Name.
4. Click Finish. You should see HelloROB appear in the Workspace under Package Explorer on the left menu.
5. Right-click on the project HelloROB and select New > Class
6. Type "HelloROB301" as the Name.
7. Check the box that says "public static void main(String[] args)"
8. Click Finish. Your new Class should appear.
9. Type in the code to match that of below. This program outputs the test message "Hello ROB301!"

Class 1: HelloROB301

```
import lejos.hardware.Button;
public class HelloROB301 {
   public static void main(String[] args) throws Exception {
      System.out.println("Hello ROB301!");
      Button.waitForAnyPress();
   }
}
```

The first line is where the centre-button of the EV3 Brick is imported, the fourth line prints the text in quotations on the screen of the EV3 Brick, and the fifth line prevents the program from ending until the centre-button is pressed.

## 4.2 Uploading to the EV3 Brick

To upload this program onto the EV3 Brick, follow these steps:

1. Turn ON the EV3 Brick.
2. Connect the computer to the EV3 Brick using the USB cable provided.
3. When leJOS has fully booted on the EV3 Brick, in Eclipse select Run $>$ Run As $>$ leJOS EV3 Program.
4. You should see a loading screen on the EV3 Brick display.
5. After loading, the program will run automatically.
6. You should see "Hello ROB301!" on the EV3 Brick display - congratulations!

You can rerun the program from the EV3 Brick any time without having to reload it by navigating to Programs using the navigation pad and then selecting the desired program from the list available.

## 4.3 Actuator Characterization

Now that the EV3 Brick can be programmed, we can explore how the robot interacts with the world. This section seeks to quantify the performance of the two different actuator types, the large servo motor and medium servo motor.

The following code activates a motor (either type) which is connected to Port A of the EV3 Brick. A new class named "MotorTest" was defined for this example.

Class 2: MotorTest

```
import lejos.hardware.motor.*;
public class MotorTest {
   public static void main(String[] args) throws Exception {
      Motor.A.setSpeed(90);
      Motor.A.rotate(360);
   }
}
```

The first line is where motors on all ports of the EV3 Brick, the fourth line sets the rotation rate of the motor to $90^\circ\ \mathrm{s}^{-1}$, and the fifth line rotates the motor $360^\circ$.

### Investigation

1. Change the rotation angle from $360^\circ$ to $-720^\circ$ and upload the program again. Observe the change in operation.

2. Change the rotation rate from $90^\circ\,\mathrm{s}^{-1}$ to $360^\circ\,\mathrm{s}^{-1}$ and upload the program again. Observe the change in operation. What is the maximum rotation rate in °/s for each motor?

3. Refer to Additional Resources at the end of this document for a link to a complete list of leJOS EV3 motor options.

## 4.4 Sensor Characterization

Next we explore how the robot observes its surroundings. This section seeks to quantify the information received from the four different sensor types. For convenience, connect your sensors as follows: Touch (Port 1), Ultrasonic (Port 2), Gyro (Port 3), and Color (Port 4). It will be useful to poll the sensors constantly to observe changes easily; for that, we will use a simple *while loop* which will exit when the centre button is pressed.

### 4.4.1 Touch Sensor

Since the Touch sensor has only two possible states, undepressed ("0") and depressed ("1"), the output of this sensor requires no calibration. The following code polls and outputs the state of the Touch sensor which is connected to Port 1 of the EV3 Brick. A new class named "TouchTest" was defined for this example.

Class 3: TouchTest

```
import lejos.hardware.Button;
import lejos.hardware.lcd.*;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.EV3TouchSensor;
public class TouchTest {
   public static void main(String[] args) {
      EV3TouchSensor touch = new EV3TouchSensor(SensorPort.S1);
      LCD.clear();
      while (!Button.ENTER.isDown()) {
         int sampleSize = touch.sampleSize();
         float[] touchsample = new float[sampleSize];
         touch.fetchSample(touchsample, 0);
         LCD.clear();
         System.out.println(touchsample[0]);
      }
      touch.close();
   }
}
```

The first four lines imports the centre button on the EV3 Brick, LCD display on the EV3 Brick, sensor ports on the EV3 Brick, and the Touch sensor. The seventh line initializes the Touch sensor at Port 1. The *while loop* is active until the user presses the centre button the the EV3 Brick and is constantly polling and displaying the Touch sensor state.

### 4.4.2 Ultrasonic Sensor

The Ultrasonic sensor has two measurement modes: *getDistanceMode* for distance measurement and *getListenMode* to check for sonic signals. The latter will not be discussed and the default is the former so no special settings are necessary. The following code polls and outputs the state of the Ultrasonic sensor which is connected to Port 2 of the EV3 Brick. A new class named "SonicTest" was defined for this example.

Class 4: SonicTest

```
import lejos.hardware.Button;
import lejos.hardware.lcd.*;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.EV3UltrasonicSensor;
public class SonicTest {
   public static void main(String[] args) {
      EV3UltrasonicSensor sonic = new EV3UltrasonicSensor(SensorPort.S2);
      LCD.clear();
      while (!Button.ENTER.isDown()) {
         int sampleSize = sonic.sampleSize();
         float[] sonicsample = new float[sampleSize];
         sonic.fetchSample(sonicsample, 0);
         LCD.clear();
         System.out.println(sonicsample[0]*100);
      }
      sonic.close();
   }
}
```

The first four lines imports the centre button on the EV3 Brick, LCD display on the EV3 Brick, sensor ports on the EV3 Brick, and the Ultrasonic sensor. The seventh line initializes the Ultrasonic sensor at Port 2. The *while loop* is active until the user presses the centre button the the EV3 Brick and is constantly polling and displaying the Ultrasonic sensor state. Note that line 14 is multiplied by 100.

#### Investigation

1. In what units is the output displayed? (Use a ruler for a quick estimate.)
2. What is the approximate maximum range of your Ultrasonic sensor?

### 4.4.3 Gyro Sensor

The Gyro sensor has two measurement modes: *getAngleMode* for tilt angle and *getRateMode* for tilt rate. Both of these will be investigated. The following code polls and outputs the states of the Gyro sensor which is connected to Port 3 of the EV3 Brick. A new class named "GyroTest" was defined for this example.

Class 5: GyroTest

```
import lejos.hardware.Button;
import lejos.hardware.lcd.*;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.EV3GyroSensor;
public class GyroTest {
   public static void main(String[] args) {
      EV3GyroSensor tilt = new EV3GyroSensor(SensorPort.S3);
      LCD.clear();
      while (!Button.ENTER.isDown()) {
         int sampleSize = tilt.sampleSize();
         float[] tiltsample = new float[sampleSize];
         float[] ratesample = new float[sampleSize];
         tilt.getAngleMode().fetchSample(tiltsample, 0);
         tilt.getRateMode().fetchSample(ratesample, 0);
         LCD.clear();
         System.out.println(tiltsample[0] + " " + ratesample[0]);
      }
      tilt.close();
   }
}
```

The first four lines imports the centre button on the EV3 Brick, LCD display on the EV3 Brick, sensor ports on the EV3 Brick, and the Gyro sensor. The seventh line initializes the Gyro sensor at Port 3. The *while loop* is active until the user presses the centre button the the EV3 Brick and is constantly polling and displaying the Gyro sensor states. Note that line the angle and rate are polled separately using their respective modes.

#### Investigation

1. Where is angle $0^\circ$ defined?

### 4.4.4 Color Sensor

The Color sensor has, in fact, four measurement modes but we will only investigate three: *getColorIDMode* for identifying observed colors, *getRedMode* for measuring reflected light,

and *getAmbientMode* for measuring ambient light. Since some modes of the Color sensor require active lighting from the sensor itself, the three modes are investigated separately.

**Color Identification.** The following code polls and outputs the states of the Color sensor in *getColorIDMode* which is connected to Port 4 of the EV3 Brick. A new class named "ColorIDTest" was defined for this example.

Class 6: ColorIDTest

```
import lejos.hardware.Button;
import lejos.hardware.lcd.*;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.EV3ColorSensor;
public class ColorIDTest {
   public static void main(String[] args) {
      EV3ColorSensor color = new EV3ColorSensor(SensorPort.S4);
      LCD.clear();
      while (!Button.ENTER.isDown()) {
         int sampleSize = color.sampleSize();
         float[] idsample = new float[sampleSize];
         color.getColorIDMode().fetchSample(idsample, 0);
         LCD.clear();
         System.out.println(idsample[0]);
      }
      color.close();
   }
}
```

The first four lines imports the centre button on the EV3 Brick, LCD display on the EV3 Brick, sensor ports on the EV3 Brick, and the Color sensor. The seventh line initializes the Color sensor at Port 4. The *while loop* is active until the user presses the centre button the the EV3 Brick and is constantly polling and displaying the Color sensor state. The *ColorIDMode* returns a number between 0 and 7 depending on the color observed as defined in Section 3.1.4.

### Investigation

1. Check to see if the color output matches the description in Section 3.1.4 when observing various colored objects (use colored LEGO pieces from your kit).

**Reflected Light.** The following code polls and outputs the states of the Color sensor in *getColorRedMode* which is connected to Port 4 of the EV3 Brick. A new class named "ColorRedTest" was defined for this example.

Class 7: ColorRedTest

```
import lejos.hardware.Button;
import lejos.hardware.lcd.*;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.EV3ColorSensor;
public class ColorRedTest {
   public static void main(String[] args) {
      EV3ColorSensor color = new EV3ColorSensor(SensorPort.S4);
      LCD.clear();
      while (!Button.ENTER.isDown()) {
         int sampleSize = color.sampleSize();
         float[] redsample = new float[sampleSize];
         color.getRedMode().fetchSample(redsample, 0);
         LCD.clear();
         System.out.println(redsample[0]);
      }
      color.close();
   }
}
```

The first four lines imports the centre button on the EV3 Brick, LCD display on the EV3 Brick, sensor ports on the EV3 Brick, and the Color sensor. The seventh line initializes the Color sensor at Port 4. The *while loop* is active until the user presses the centre button the the EV3 Brick and is constantly polling and displaying the Color sensor state. The *ColorRedMode* returns a number between 0 and 1 depending on the amount of reflected light observed.

## Investigation

1. Activate the sensor over a light-colored surface, then a dark-colored surface and observe the change.

2. What is the output when the sensor is over half of the light-colored surface and half of the dark-colored surface?

**Ambient Light.** The following code polls and outputs the states of the Color sensor in *getColorAmbientMode* which is connected to Port 4 of the EV3 Brick. A new class named "ColorAmbientTest" was defined for this example.

Class 8: ColorAmbientTest

```
import lejos.hardware.Button;
import lejos.hardware.lcd.*;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.EV3ColorSensor;
public class ColorAmbientTest {
   public static void main(String[] args) {
      EV3ColorSensor color = new EV3ColorSensor(SensorPort.S4);
      LCD.clear();
      while (!Button.ENTER.isDown()) {
         int sampleSize = color.sampleSize();
         float[] ambsample = new float[sampleSize];
         color.getAmbientMode().fetchSample(ambsample, 0);
         LCD.clear();
         System.out.println(ambsample[0]);
      }
      color.close();
   }
}
```

The first four lines imports the centre button on the EV3 Brick, LCD display on the EV3 Brick, sensor ports on the EV3 Brick, and the Color sensor. The seventh line initializes the Color sensor at Port 4. The *while loop* is active until the user presses the centre button the the EV3 Brick and is constantly polling and displaying the Color sensor state. The *ColorAmbientMode* returns a number between $0$ and $1$ depending on the amount of ambient light observed.

### Investigation

1. Activate the sensor away from a light-source, then near a light-source and observe the change.

# 5 Conclusion

You should now be able to use the LEGO EV3 sensors/actuators in basic applications.

# 6 Additional Resources

1. LEGO Mindstorms - `http://mindstorms.lego.com/`
2. Java Programming - `http://www.javaprogrammingforums.com/`
3. leJOS - `http://sourceforge.net/p/lejos/wiki/Home/`
4. leJOS EV3 API - `http://www.lejos.org/ev3/docs/overview-summary.html`