Lab Objectives

By the end of this lab you should be able to:

1. Identify how the robot’s object detection sensor works.
2. Use repeat loops and if/else statements to continuously get information from the robot’s sensors.
3. Create a program that uses sensor information to dictate the behavior of the robot.

Infrared Object Detection Sensor

The Scribbler 2 is equipped with several sensors, one of which can be used for detecting obstacles that are in the robot’s path. The Scribbler 2 performs object detection by first shining infrared light from its two infrared emitters. If an object is in the robot’s path, the infrared light reflects off the obstacle and bounces back to its infrared detector. The two emitters take turns shining infrared light in front of the robot, which enables the infrared detector to sense whether the reflected light came from the right or left emitter. The pictures below show the locations of the infrared detector and emitters, and an illustration of the robot performing object detection.

The robot can “see” light objects (like the white front of the sneaker) better than very dark objects because dark objects do not reflect as much infrared light.

We can get information from the object detection sensor by using the command:

\[ s2.obstacle(side, threshold) \]
This command returns a value of 255 if an object has been detected and a value of 0 if no object was detected. The variable `side` can be set to either `s2#RIGHT` or `s2#LEFT` to designate which side to check. The variable `threshold` sets the threshold for the infrared detector and can be any value between 0 and 100. At high threshold values, only very close objects will be detected. At lower values, farther objects (and possibly the rolling surface itself) will be detected. If the `object_threshold` is equal to zero, then the default threshold setting will be used. For this lab we will be using a `object_threshold` value of 20.

**Decisions With Sensors**

Once we have some sensor information, we want to use it to control the behavior of the robot depending on what the sensor “sees.” Since we want the robot to perform some action if the sensor “sees” something, we will use `if/else` statements. Recall that an `if/else` statement has the form:

```plaintext
if(condition)
    some action
else if (condition)
    some action
else
    some action
```

where the truthfulness of `condition` dictates which action will be performed. Each `if` and `elseif` can have one or multiple conditions. When more than one condition is desired, we must use **boolean operators** to show the relationship between the different conditions. The table below shows the three boolean operators used in the SPIN programming language.

<table>
<thead>
<tr>
<th>Boolean Operator</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AND</strong></td>
<td>if (x == 1) and (y == 1)</td>
<td>The if statement is executed if both x and y equal 1.</td>
</tr>
<tr>
<td><strong>OR</strong></td>
<td>if (x == 1) or (y == 1)</td>
<td>The if statement is executed if either x or y equal 1.</td>
</tr>
<tr>
<td><strong>NOT</strong></td>
<td>if (x == 1) and not (y == 1)</td>
<td>The if statement is executed if x is equal to 1 and y is not equal 1.</td>
</tr>
</tbody>
</table>

Using `if/else` statements is good for deciding what the robot should do based on what the sensor “sees,” but `if/else` statements only use the sensor information once. If we want to continually monitor what the sensor “sees,” then we need to put our `if/else` statements inside of an infinite `repeat-loop` along with the command used to get information from the sensor. Recall that a `repeat-loop` is the structure used in the SPIN language to create loops and can be used the same way `for-loops` and `while-loops` are used in MATLAB. The basic syntax for creating an infinite `repeat-loop` involves using just the `repeat` command without any conditions as follows:
In the context of this lab, some action would represent the command for getting information from the sensor along with the if/else statements that are used to make decisions based on that information.

Controlling the Robot's Motion

In the first two exercises we will use sensor information to turn the robot's LEDs on and off. This is good practice for understanding how to interact with the robot's sensors, but we want to be able to control the robot's motion as well as its LEDs.

To begin with, we need to get the robot moving forward. In the previous lab we learned how to use the command

```plaintext
s2.wheels_now(leftMotor,rightMotor,time)
```

to turn the robot in place. This command can also be used to move the robot forward in a straight line by setting the variables leftMotor and rightMotor to the same value. If we put this command inside an infinite repeat-loop, then the robot will move forward indefinitely.

When the robot moves forward and senses an obstacle, we want it to turn away from the object that is blocking its path. We can use the command

```plaintext
s2.turn_deg_now(degrees)
```

to turn the robot by setting the variable degrees to a value between ±1 and ±180. Positive values will turn the robot counterclockwise and negative values will turn the robot clockwise.

The two motion commands above are preemptive commands, so they will override and stop any motion command that comes before them. This means that if the robot has been commanded to move forward continuously, but we want it to turn when it "sees" an obstacle, we need to put in a wait command after the turn command to ensure that the robot performs a full turn before it continues its forward motion. The command we can use for this is

```plaintext
s2.wait_stop
```

This command tells the robot to wait until a given motion command has been fully executed before executing the next motion command. For exercise 3, we will need to use this command after each s2.turn_deg_now(degrees) command.

Exercises

1. Program your Scribbler 2 to make the left LED turn on when an object is on the left-hand side of the robot's field of vision, and the right LED turn on when an object is on the right-hand side of the robot's field of vision. **Hint:** use a small delay in your code (i.e., s2.delay_tenths(1)) after turning on/off the LEDs.
2. Using your program from exercise 1, add some code that makes the center LED turn on if an object is directly in front of the robot (i.e., in both fields of vision).

3. Using your program from exercise 2, add some code that makes your robot turn based on what it “sees.” Start by commanding your robot to continuously drive forward at full speed (i.e., 255). If the robot “sees” an object on the right-hand side of its field of vision, make it turn left 90 degrees. If the robot “sees” an object on the left-hand side of its field of vision, make it turn right 90 degrees. If the robot “sees” an object directly in front of it (i.e., the center LED turns on), make it turn 180 degrees.

Note: For this lab you don’t need to comment every line of your code. Only include enough comments to sufficiently explain your programs.