*COS 116*
*The Computational Universe*
***Laboratory 3: Controlling the Robot I***

---

**You should hand in your lab report at the beginning of lecture on Tuesday, February 23. Include the following:**

- **Reports for pseudocode tests: parts 1, 2, 3**
- **The 2 programs and resulting pictures that you have created in Part 4**
- **Responses to questions printed in bold (number them by Part/Experiment)**

Today you'll meet Scribbler, the robot we'll be using in several labs this term. Scribbler is a very capable little machine. It has two independently driven wheels, three status lights, and a small speaker. It can detect light sources, obstacles in its path, and lines printed on the "road." Best of all, the Scribbler can easily be programmed to perform a multitude of tasks.

For this class, we've created a way to control the Scribbler using simple "pseudocode" instructions. These statements resemble English commands, and they'll help you grasp some important concepts about computation without having to understand a complete programming language.

In this lab you will learn the building blocks of pseudocode by analyzing, testing and writing a series of pseudocode files. Then, in two short experiments, you'll use the robot and the provided pseudocode to explore the robot's capabilities. You won't be writing any pseudocode this week-- that will be the focus of the next week's lab.

Your TA will issue you the following items:

- The Scribbler, with 6 AA batteries
- A black serial cable
- A gray serial-to-USB adapter
- Ruler
- Construction paper
- Tape
- Permanent marker

Please return unused materials at the end of the session.

If you get stuck at any point, feel free discuss with another student or a TA. However, you are not allowed to copy another student's program or answers.

You can print your programs directly from Scribbler Control Panel, or copy and paste them into a program like Microsoft Word. (Click "Copy as Text" from the Edit menu.)

## Scribbler Software Installation

The course staff has created a simple interface for the Scribbler robot, called the Scribbler Control Program (SCP). SCP was written by a Princeton computer science undergrad. You'll need to install SCP on the Friend 007 laptops before you can begin this lab. To install SCP, visit

**http://www.cs.princeton.edu/courses/archive/spring08/cos116/instruct.php**

and follow the instructions. If you are installing your Scribbler on your personal computer, you'll find the relevant instructions there as well.

## General Test Procedure

In this lab you will test the robot with several pseudocode files. Keep detailed notes about each test and hand in them when you finish. Follow this procedure for each test:

1. **Use Scribbler Control Panel to open the test file.** In the Help menu, select Sample Programs. Select a test file from the list.

2. **Examine this pseudocode. Write in your notes what you think it will make the robot do.**

3. **Load the file on to the robot and observe the robot's behavior.**

   a) **Attach your robot to the computer.**
   Use the black cable to attach the robot to the gray serial-to-USB adapter. Attach the adapter to the USB port (on the right) on the Friend 007 laptop.

   b) **Switch the robot on.**

   c) **Download the pseudocode to the robot.**
   In the Tools menu, click Download to Robot (equivalently, press F5).
   - *Note: If the program asks for a "COM port", select "COM 4".*

   d) **Unplug the cable from the robot.**

   e) **Place the robot where it will have enough room to run the test. (For some tests, you will need to use the floor.)**

   f) **Press the robot's button twice (rapidly) to start it.**

   g) **Observe the robot's behavior. Some tests will provide additional instructions.**

h) **If the robot is still moving when you are finished with your observations, press the button once more to stop it.**

i) **To conserve the batteries, you should turn off the robot if you're not going to use it for a few minutes.**

4. **Did the robot's behavior differ from what you predicted? If so, figure out why and write about it in your notes.**

## Part 1: Basic Motion, Lights, and Sounds

1. **Test the robot with the file "01 Forward, Back"**

```
1  Move Forward for 1s
2  Pause 0.5s
3  Move Back for 1s
4  END
```

   Remember, you should use the testing procedure described on the previous page for all the tests in this week's lab.

2. **Test the file "02 Spin Left, Spin Right"**

```
1  Spin Left for 1s
2  Pause 0.5s
3  Spin Right for 1s
4  END
```

3. **Test the file "03 Turn Left, Turn Right"**

```
1  Turn Left for 1s
2  Pause 0.5s
3  Spin Left for 0.7s
4  Pause 0.5s
5  Turn Right for 1s
6  END
```

   In your notes, explain the difference between spinning and turning. Hint: Carefully observe the motion of the robot's wheels while performing a spin and a turn.

4. **Test the file "04 LEDs"**

```
1  LED: ON,  OFF,  OFF
```

```
Pause 1s
LED: ON,  ON,  OFF
Pause 1s
LED: ON,  ON,  ON
Pause 1s
LED: OFF,  OFF,  OFF
END
```

Notice that, unlike the motor commands, the LED ("Light Emitting Diode") commands have a lasting effect on the state of the LEDs.

5. **Test the file "05 Sound – Scale"**

```
Play Sound for 0.5s at Frequency 262Hz
Play Sound for 0.5s at Frequency 294Hz
Play Sound for 0.5s at Frequency 330Hz
Play Sound for 0.5s at Frequency 349Hz
Play Sound for 0.5s at Frequency 392Hz
Play Sound for 0.5s at Frequency 440Hz
Play Sound for 0.5s at Frequency 494Hz
Play Sound for 0.5s at Frequency 523Hz
END
```

## Part 2: Sensors and Branches

1. **Test the file "06 If, Light"**

```
If <Light from any 1 side> Then
{
   Play Sound for 1s at Frequency 440Hz
}
Else
{
   LED: ON,  ON,  ON
}
END
```

First try starting the robot with your hand covering its light sensors, then try starting it with open light sensors. What happens after you close the light sensors again? Remember, pseudocode commands are performed *in sequence*.

2. **Test the file "07 If, Obstacle"**

```
1  If <Obstacle on Either Side> Then
2  {
3     Play Sound for 1s at Frequency 440Hz
4  }
5  Else
6  {
7     LED: ON,  ON,  ON
8  }
9  END
```

First start the robot where there is nothing in front of it for several feet. Then hold your hand about three inches ahead of the robot and try again. What happens after you move your hand away?

3. **Test the file "08 If Not"**

```
1  If Not <No Obstacle> Then
2  {
3     Play Sound for 1s at Frequency 440Hz
4  }
5  Else
6  {
7     LED: ON,  ON,  ON
8  }
9  END
```

Compare this test to the previous one. Are they logically equivalent? Is there any difference in the robot's behavior? Can you think of another way to express line 1 in the pseudocode from step 1 above?

## Part 3: Loops

1. **Test the file "09 Do For n Times"**

```
1  Do for 3 Times
2  {
3     Play Sound for 0.25s at Frequency 392Hz
4  }
5  Play Sound for 1.5s at Frequency 330Hz
6  END
```

2. **Test the file "10 Do While"**

```
1  Do While <No Obstacle>
```

```
{
   Move Forward for 0.2s
}
Play Sound for 1s at Frequency 440Hz
```

Before you start the robot, place it at least three feet away from an obstacle (your bag, the wall, etc.). Allow the robot to move forward until it encounters the obstacle. When do you hear the sound?

### 3. Test the file "11 Do Forever"

```
Do forever
{
   If <Obstacle on Either Side> Then
   {
      Play Sound for 1s at Frequency 440Hz
   }
   Else
   {
      Move Forward for 0.2s
   }
}
END
```

This time place the robot at least three feet from a *movable* obstacle. After the robot encounters the obstacle, remove the obstacle and note the robot's behavior. What happens if you place the object back in the robot's path?

### 4. Test the file "12 Loop, Line Sensor"

```
Do forever
{
   If <Line on Either Side>
   {
      LED ON, OFF, OFF
   }
   Else
   {
      LED OFF, OFF, OFF
   }
}
END
```

The line sensor is best used within a loop, as shown here. To test it, draw a heavy black line (at least 1 inch thick) on a sheet of construction paper.

Slowly move the robot back and forth over the line and observe the behavior of the LED.

(The line sensor is very sensitive. If it doesn't behave the way you expect, ask the TA for help.)

## Part 4: Motion by Dead-reckoning

You can move the robot along a pre-determined path by specifying the motor speed and duration of each leg of the journey. This kind of navigation that does not use any external landmarks is called *dead reckoning.* (This is a sailor's term from the 17th century.) One drawback of dead reckoning is that any error in the measurement of speed or direction accumulates over time, causing the robot's position to become less and less accurate. Another complication here is that the Scribbler lacks a speedometer, and your program will use motor power to approximate actual speed.

In this section you'll use dead reckoning to make the Scribbler draw some simple figures. You'll also see how to apply basic geometry to plan the robot's course.

**Before you begin, calibrate your robot's motors to ensure that it can drive straight. To calibrate:**

1) **Plug the serial cable into the robot, and turn the robot on.**
2) **Run the Scribbler Control Program. You may need to re-install SCP on your computer first; if so, follow the same procedure as in Lab 2.**
3) **From the Tools menu, select "Calibrate Motors", and then follow the instructions.**

### Experiment 1 — Draw a square

*Instructions required:* Forward, Spin Right, Pause, Do for *n* times

**Program the Scribbler to draw a square with sides approximately 4 inches in length. Use trial and error to get the distances and angles right. Use a loop to avoid writing the same instructions four times.**

*Tips for accurate drawing:* Tape the paper to your lab table so it doesn't slip. Always test the robot on the paper, since angles and distances may come out differently on other surfaces. Always insert a 0.2 second pause between motor commands to give the robot time to come to a stop.

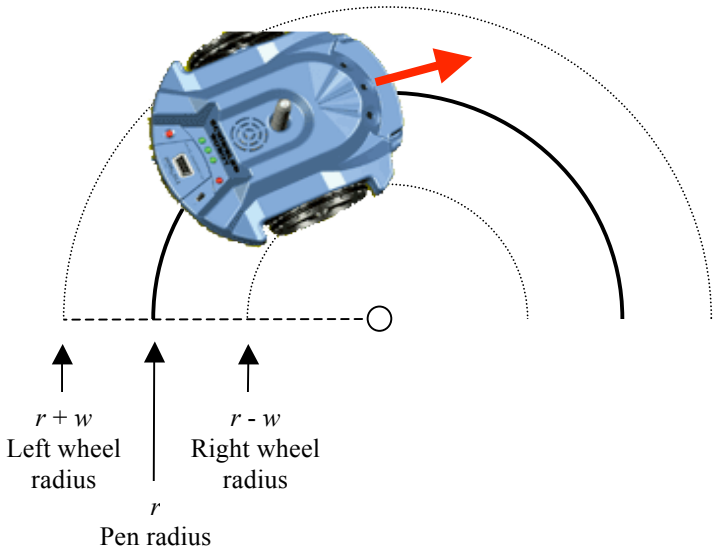
---

To draw more complicated pictures, you'll need to take advantage of the Scribbler's Advanced Motor Controls. These allow greater control over the speed of each wheel.

**Click on the *Motor* command in Scribbler Control Panel, and check the box next to "Enable Advanced Motor Controls."** Notice the additional sliders that become available. You can use these to set the speed of each motor from 100% (full power, forward) to -100% (full power, reverse). Try clicking each of the Basic Motor Control commands: Forward, Reverse, Left Turn, Right Turn, Left Spin, Right Spin. Note the Advanced Motor Control speed settings corresponding to each basic command.

One use of Advanced Motor Controls is to make the Scribbler turn along a path with a specific radius. Suppose you wanted to draw a circle with radius $r$, as shown at right. If the distance between the robot's wheel is $2w$, then in one full circle the left wheel will travel a distance of $2\pi(r + w)$, while the right wheel will travel a distance of $2\pi(r - w)$. Say it takes $t$ seconds to draw the circle. Then: *left wheel speed* $= 2\pi(r + w) / t$, and *right wheel speed* $= 2\pi(r - w) / t$. This yields the proportion:

$$\frac{right\ speed}{left\ speed} = \frac{r - w}{r + w}$$




**Measure *w* (half the distance between the wheels). Assume the left motor speed is 100% and solve the proportion to find the speed for the other motor that results in a curve with a 6 inch radius. What about a 3 inch radius? A 1 inch radius?**

**Experiment 2 — Draw a circle**

*Instructions required:* Advanced motor controls

**Program the Scribbler to draw a circle with a radius of approximately 5 inches. Use the proportion above to set the motor powers. Use trial and error to make sure the circle is complete without retracing too much of the line. Your program should use only a single instruction.**

## Part 5: Experiments

### Experiment I: How consistent is the robot's motion?

The movement commands you have seen so far operate by sending a certain amount of power to the robot's motors for a set length of time. In this experiment you will determine how accurately the robot can repeat a sequence of movements.

1. **Tape a piece of construction paper to the lab table so that it doesn't slip.**

2. **Insert a pen into the robot's center hole and place the robot on the construction paper.**

3. **Load the "01 Forward, Backward" pseudocode. Run it and allow the robot to trace out its path with the marker. Does the robot return exactly to its starting position?**

4. **Repeat the trace five more times. Start in a different spot each time.**

5. **Find the average distance the robot traveled in 1 second of motion. What is the range of distances traveled across the trials?**

6. **Examine the traces. Are they identical? Speculate in your notes about what might cause any differences you observe. If you were building a more sophisticated robot, how might you correct for such differences?**

## Experiment II: Using the robot to measuring reflectance

The robot's obstacle sensor detects objects in the robot's path. It works by projecting a beam of infrared light in front of the robot. If an obstacle is present, some of this light is reflected back towards the robot where it is detected by a sensor.

At what distance will the robot detect an object? The answer depends on how much infrared light the object reflects. Objects that reflect little light are hard for the sensor to "see", so the robot will only notice them at close range. The robot can sense highly reflective objects from farther away, since they send back more light. You can use this behavior to measure how much infrared light different objects reflect.

1. **Pick three objects that reflect different amounts of light.**
   For example, you might use a white sheet of construction paper, a dark colored notebook, and your hand. (Infrared reflectance is sometimes, but not always, correlated with reflectance of visible light.)

2. **Load the file "07 If, Obstacle" on to your robot.**

3. **Point the obstacle sensor at the first object and start the robot. Find the closest distance where the robot reliably detects the object.**

4. **Repeat for the other two objects.**

5. **Compare the distances you observed. Which object reflected the most infrared light? Did any of your measurements surprise you?**