**CE-8 Lab 6: More Sensors**

**Important Notes:**
1. Please put your comments to your program; otherwise, there will be some points off for not doing so.
2. All programs should be printed from either MATLAB or PBASIC editor. (not from MS Word)
3. Labs are individual work. You can ask your lab-mates, but you never turn in your work as a group. That means your code will be different from the person you asked questions to.
4. Labs are due at the beginning of the lab you’re attending to; otherwise, it is considered as late reports.

TA: HyukChoong ‘Chuck’ Kang
Email: h kang@soe.ucsc.edu

**Lab Objectives**

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

1. **Identify** how to use subroutines and the stall sensor.
2. **Create** a program that uses subroutines and the stall sensor that enables your robot to avoid objects and be able to unstick itself.

**Subroutines**

So far you've been able to make a program that commands your robot to do various things based on the state of a sensor. As your programs get larger and more complicated you'll need a way to keep them organized. One of the ways we do this in many programming languages is by using subroutines. Subroutines are a way for you to put a chunk that might be used often away from the rest of your program and then 'call' it up my using the name that you give it. Let's look at an example. Let's say that I have a program that makes my robot constantly go straight or turn to left depending on different circumstances. Now instead of having to write,

```
PULSOUT MotorLeft, 3000
PULSOUT MotorRight, 3000
```

every time I want my robot to go straight wouldn't it be nice if I could just write,

```
GOSUB go_straight
```

instead? To make a subroutine we first need to give it a name. In the example above the name of my subroutine is **go_straight**. I still need to write the code for go_straight somewhere in my program. We will put the subroutines before our **DO...LOOPS**. The code would look like,

```
go_straight:
PULSOUT MotorLeft, 3000
PULSOUT MotorRight, 3000
RETURN
```

Notice the **RETURN** command, this tell the program where our subroutine ends and instructs the program to go back to the place in where the subroutine was called from. So the entire program would look like,

```pascal
' {STAMP BS2}
' {PBASIC C 2.5}
MotorRight    PIN 12
MotorLeft     PIN 13

go_straight:
    PULSOUT MotorLeft, 3000
    PULSOUT MotorRight, 3000
    RETURN
DO
    GOSUB go_straight
LOOP
```

While this isn't a very complicated program, it illustrates how a subroutine works. When the GOSUB command is used it will go to the spot in the program that has the matching name. It will then return to the place it was called from once it hits the RETURN command.

### Stall Sensor

The scribbler has a built in sensor that can detect if the robot's wheels are not turning. We can use this sensor to detect when we have hit a wall. The sensor is attached to pin 7. This sensor outputs a 0 if the wheels are not stuck and a 1 if the wheels are stuck. We can check the stall sensor by simply using a PIN directive and checking that pin directly. For instance,

```pascal
Stall    PIN 7

IF Stall = 1 THEN
    'Were Stuck'
ENDIF
```

### Exercises

1. Program your robot to go in a straight line, if it detects that it is stalled have the robot backup and turn away from the wall and keep on going. In this way your robot will 'bounce' around the room. Use subroutines for when you want it to turn.
2. Combine what you did last week with the stall sensor so that it will try to avoid obstacles, but if it gets stuck have it backup and keep going in a different direction. Also program it so that if it detects something in front of it with the obstacle sensor it will turn away from it rather then stopping.