**Activity #2**  
**Aim:** Object Detection and Avoidance

An interesting thing about the IR detectors is that their outputs are just like the whiskers. When no object is detected, the output is high; when an object is detected, the output is low.

**Program For IR Object Detection/Avoidance**

Each IR LED circuit receives the `freqout` signal. Immediately after sending the `freqout` signal the output state of the IR detector in the pair needs to be checked and recorded. After the information is recorded, it can be compared using the `if...then` statements.

`Robotics! v1.5, Program Listing 4.2: Roaming with Adjusted for IR Pairs.

```
' {Stamp bs2}                       Stamp Directive.

----- Declarations ---------------

pulse_count var byte              ' For...next loop counter.
left_IR_det var bit               ' Two bit variables for saving IR
detector output values.
right_IR_det var bit

----- Initialization --------------

output 2                           ' Set all I/O lines sending freqout
output 7                           ' signals to function as outputs
output 1                           ' Program start/restart signal.
freqout 2, 2000, 3000              ' Set Fl2 and Fl3 to output-low.
low 12                             
low 13                             

----- Main Routine -----------------

main:                               ' Detect object on the left.

freqout 7, 1, 38500                 ' Send freqout signal - left IRLED.
left_IR_det = in8                    ' Store IR detector output in RAM.
detector output values.
right_IR_det = in0                   ' Detect object on the right.

' Repeat for the right IR pair.

' With the exception that values stored in RAM are used instead of
' input register values, the decision making process is the same as
' the one used in Program Listing 3.2.
```
if left IR det = 0 and right IR det = 0 then u turn
if left IR det = 0 then right turn
if right IR det = 0 then left turn

' The commands from this point onward are identical to
' Program Listing 3.2: Roaming with Whiskers.

forward: ' If no detect, one forward pulse.
    pulsout 12, 500
    pulsout 13, 1000
    pause 20

goto main ' Check again.

'------ Navigation Routines -------

left turn: ' Left turn routine.
    gosub backward
    for pulse count = 0 to 35
        pulsout 12, 500
        pulsout 13, 500
        pause 20
    next
    goto main

right turn: ' Right turn routine.
    gosub backward
    for pulse count = 0 to 35
        pulsout 12, 1000
        pulsout 13, 1000
        pause 20
    next
    goto main

u_turn: ' U-turn routine.
    gosub backward
    for pulse count = 0 to 75
        pulsout 12, 1000
        pulsout 13, 1000
        pause 20
    next
    goto main

'------ Navigation Subroutine ------

backward: ' Used by each navigation routine.
    for pulse count = 0 to 75
        pulsout 12, 1000
        pulsout 13, 500
        pause 20
    next
return
How the Roaming with Whiskers Adjusted for IR Pairs Program Works

Two bit variables, left_IR_det and right_IR_det, are added for capturing and holding the IR detectors’ output states.

```
declarations:
pulse_count var byte
left_IR_det var bit
right_IR_det var bit
```

The main routine has four additional commands, two for checking the output of each IR detector. Each freqout command sends a 1 ms unfiltered freqout signal to the IR LED circuit in the pair. The value at the input connected to the IR detector’s output is saved as a bit variable in the BASIC Stamp’s RAM. For example, the command `freqout 7, 1, 38500` is followed immediately by the statement `left_IR_det = in8`. This command sets the value of `left_IR_det` equal to the input at P8, the I/O pin connected to the left IR detector’s output.

```
main:
check_IR_pairs:
freqout 7, 1, 38500
left_IR_det = in8
freqout 1, 1, 38500
right_IR_det = in0
```

The saved bit values for each IR detector output can be used. With one exception, the navigation routines that are executed according to `if...then` statements are identical to those originally used in Program Listing 3.2: Roaming with Whiskers. The `if...then` statements themselves are changed to accommodate the need to capture and store the output from each IR detector; whereas, the `if...then` statements in the whiskers program used the input values directly.

```
if left_IR_det = 0 and right_IR_det = 0 then u_turn
if left_IR_det = 0 then right_pulse
if right_IR_det = 0 then left_pulse
```

Task

As with Program Listing 3.2, you can fine tune the end arguments in the `for...next` loops to fine tune the Boe-Bot’s turning and backing up behaviors.