

## Experiment #3: Following Light

Light has many applications in robotics and industrial control. Some examples include sensing the edge of a roll of fabric in the textile industry, determining when to activate streetlights at different times of the year, when to take a picture, or when to deliver water to a crop of plants.

# 3

To sense the presence and intensity of light we'll build a couple of photoresistor circuits on our Boe-Bot. A photoresistor is a light-dependent resistor (LDR) that covers the spectral sensitivity similar to that of the human eye. The active elements of these photoresistors are made of Cadmium Sulfide (CdS). Light enters into the semiconductor layer applied to a ceramic substrate and produces free charge carriers. A defined electrical resistance is produced that is inversely proportionate to the illumination intensity. In other words, darkness produces high resistance, and high illumination produces very small amounts of resistance.

The specific photoresistors included in the Boe-Bot kit are EG&G Vactec (#VT935G). If you need additional photoresistors they are available from Parallax and electronic component suppliers (parts listing is included in Appendix A). The specifications of these photoresistors are shown in Figure 3.1:

Photoresistor Specifications

Resistance (Ohms)					Peak Spectral Response nm	$V_{MAX}$	Response Time @ 1 fc (ms, typ.)	
10 Lux 2850K			Dark				Rise (1-1/e)	Fall (1/e)
Min	Typ.	Max.	Min.	Sec.				
20K	29.0K	38K	1M	10	550	100	35	5

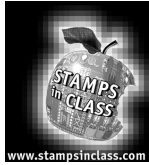
Figure 3.1: EG&G Vactec Photoresistor Specifications

Illuminance is a scientific name for the measurement of incident light. The unit of measurement of illuminance is commonly the "foot-candle" in the English system and the "lux" in the metric system. While using the photoresistors we won't be concerned about lux levels, just whether or not illuminance is higher or lower in certain directions. Based on this data the Boe-Bot will turn towards the light. For more information about light measurement with a microcontroller, take a look at Earth Measurements Experiment #4, Light on Earth and Data Logging.

The topics we'll explore in this experiment also relate to program structure. Some of the variables that will need to be customized for your Boe-Bot are how far to travel before checking the photoresistors, how much to turn when a photoresistor detects light, and how to arrange your program to do these tasks with the smallest number of steps.

## Experiment #3: Following Light

---



### Parts Required

You'll need the following parts for these experiments:

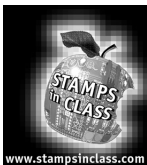
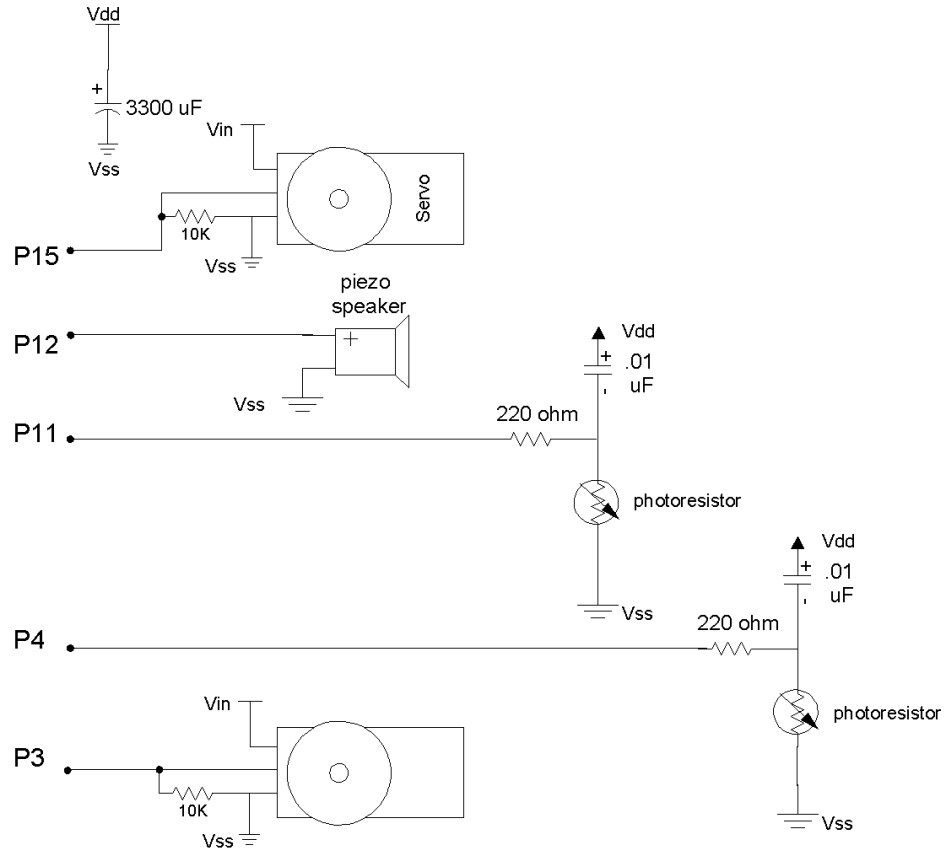
- (1) fully constructed Boe-Bot with servos
- (1) 3300 uF capacitor
- (1) piezo speaker
- (2) 0.01 uF capacitor
- (2) 10K Ohm resistors
- (2) 220 Ohm resistors
- (2) photoresistors
- (misc.) jumper wires



### Build It!

While building this circuit it would be practical to move all of the servo connections and speakers towards the part of the breadboard closest to the BASIC Stamp. The schematic for this project is shown in Figure 3.2.

Figure 3.2: Light Sensing Boe-Bot Schematic  
The capacitor sizes for the photoresistors are not critical. These may also be 0.1 or 1.0 uF.



### Program It

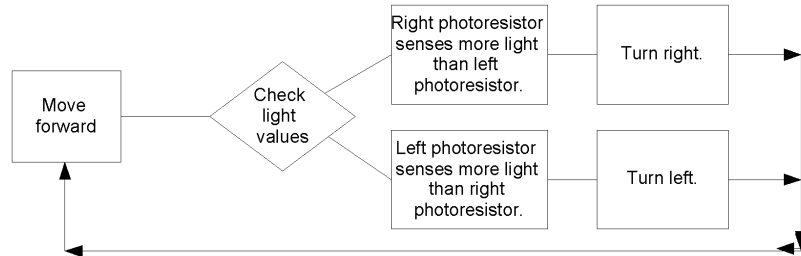
The Boe-Bot must be programmed to follow light, making turns towards more light. Consider how we want the Boe-Bot to execute the series of commands. One approach is to build a flowchart of the steps the Boe-Bot should execute to find the light and move in that direction. An example is shown in Figure 3.3.

## Experiment #3: Following Light

---

Figure 3.3: Basic Operational Flow Chart

The main decision that needs to be made is in response to the question "where's the light?". Other decisions relating to "how often" to check the photoresistors and "how far" to travel are also important.



There are some decisions not shown in this flowchart. For example, how far to travel before checking the photoresistors, and how much to turn in order to track the light without missing it entirely. And how do we know if the two photoresistors are measuring light properly? We'll verify the light values by generating sound with a piezospeaker.

### Using the Photoresistor

The photoresistors have a non-linear response to light. To measure their output we'll use a resistor/capacitor circuit. The `RCTime` command measures the charge (or discharge) time of a resistor/capacitor circuit. When `RCTime` executes, it starts a counter that increments ever  $2\ \mu\text{s}$ . It stops this counter as soon as the pin is no longer in the starting state. The starting state of our pin will be 0 (0 volts). When the circuit charges to 1.4 V (and the pin is considered "high") the command will stop and store the time value in a variable. The command works as shown below, and an example is shown in Program Listing 3.1:

```
RCTime 4, 0, rightLDR
    ^===== I/O pin with photoresistor
    ^===== starting state (0 or 1)
    ^===== variable location to store the charge time
```

'Program Listing 3.1

```
scale          con    100    'adjust value to make audible frequency
rightLDR       var    word
leftLDR        var    word
```

```
right_light:
low 4          'discharge the capacitor
pause 100
rctime 4,0,rightLDR 'measure the time to charge
debug home, dec rightLDR 'display the charge time
```

```
freqout 12,100,rightLDR*scale+100 `play a tone, more light = lower pitch
pause 10

left_light:
low 11
pause 100
rctime 11,0,leftLDR
debug ? leftLDR
freqout 12,100,leftLDR*scale+100
pause 10

goto right_light
```

With more light present on the photoresistors the charge time of the resistor / capacitor circuit is shorter, and the lower frequency of the tone. In order to generate an audible range of tones adjust the `scale` constant while exposing the photoresistors to varying light levels.

You may have noticed that when you point the Boe-Bot's photoresistors towards an even light (or dark) area of the room the `rightLDR` and `leftLDR` values are not equal. This is a result of variations between capacitors and tolerance of resistors. You can "calibrate" the photoresistor returning the lower `RTime` value by adding the difference between the two photoresistors. For example, if the `rightLDR` returns `RTime` values that are lower than the `leftLDR` by a value of 5, you could make the following modification to your code:

```
.
.
right_light:
low 4
pause 100
rctime 4,0,rightLDR
rightLDR=rightLDR + 5
debug home, dec ? rightLDR
freqout 12,100,rightLDR*scale+100
pause 10
.
.
```

This calibration will only be effective over a narrow range of `RTime` values since the photoresistors are non-linear.

## Experiment #3: Following Light

---

### Source Code Example

Program Listing 3.2 is an example application of light following. Readings are taken on each photoresistor and the Boe-bot turns toward the brighter reading.

```
'Boe-Bot Program for Light Following with Sound Feedback  
'Program Listing 3.2
```

```
'Define Variables and Constants
```

```
'-----  
left_servo      con    15  
right_servo     con     3  
turnvalue       con     6  
scale           con    100  
speed           con    100  
speaker         con     12  
rightLDR        var    word  
leftLDR         var    word  
x               var    word  
'-----
```

```
'Main Program
```

```
'-----  
forward:  
for x=1 to 10  
pulsout left_servo,750-speed  
pulsout right_servo,750+speed  
pause 20  
next  
  
right_light:  
high 4  
pause 10  
rctime 4,1,rightLDR  
debug "R:",dec rightLDR  
freqout speaker,100,rightLDR*scale+100  
  
left_light:  
high 11  
pause 10  
rctime 11,1,leftLDR  
debug " L:",dec leftLDR,cr
```

```
freqout speaker,100,leftLDR*scale+100

if rightLDR < leftLDR then left
if leftLDR < rightLDR then right

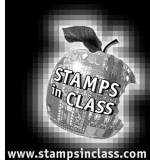
right:
high 0
for x=1 to turnvalue
  pulsout left_servo,750-speed
  pulsout right_servo,750-speed
  pause 20
next
low 0
goto forward

left:
high 14
for x=1 to turnvalue
  pulsout left_servo,750+speed
  pulsout right_servo,750+speed
  pause 20
next
low 14
goto forward
'-----
```

What would happen if the light readings were the same on both sides?

Experiment with the source code by changing different software and hardware parameters. You'll find that the Boe-Bot is not aware of objects since it has no feedback for sensing objects, and can easily run into walls and chairs. It may also be necessary to shield the edges of the photoresistors with a straw or heat-shrink tubing to reduce light reflection from the side.

In Experiment #4 we'll introduce infrared communication for proximity sensing.



## Challenge!

1. Program the Boe-Bot to hide from light. The Boe-Bot should move towards the darkest corner of the room, and when a specific level of luminance is reached it should stop in place and generate a siren signal.
2. Restructure Program Listing 3.2 to use the `gosub` statement to execute a series of tasks.
3. Can the Boe-Bot follow a light source with just one photoresistor? Instead of comparing readings from two sensors, write a program that takes two readings (in slightly different directions) with one sensor and then goes toward the brighter reading.