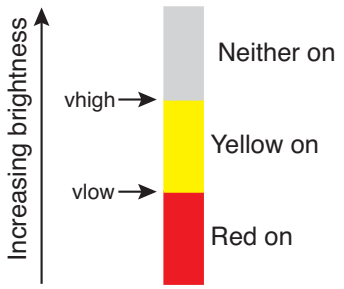
Living with the Lab
Fall 2012
v: November 21, 2012

Ambient Light Indicator
Gerald Recktenwald
gerry@me.pdx.edu

1 Assignment

Build the circuit and write the Arduino code for a device that uses a red and a yellow LED to indicate ambient light levels. At low light levels, turn the red LED on. At intermediate light levels, turn the yellow LED on. At high light levels, turn both LEDs off. The relationship between brightness and the LED status is depicted in the sketch to the right. The boundaries between the LED settings are delineated by `vlow` and `vhigh`, which are values read on an analog input pin of the Arduino.



Note that the color of the LED is arbitrary. It is helpful to be able to refer to the specific colors when describing the assignment and the Arduino code.

1.1 Concepts Used

- Analog input from a photo-resistor.
- Voltage divider circuit.
- `if ...else` construct.

1.2 Variations

- Use several LEDs arranged in a row. Turn on one LED at at time to indicate light level, where the position of the "on" LED changes from left to right (say) when the ambient light level increases. This variation introduces the potential for using an array.
- Instead of having the bar-graph LED display turn one LED on, turn on more LEDs as the light level increases.
- Use PWM to control the brightness a single LED, where the brightness increases (or decreases) with the ambient light level.

2 Solution

The following sections explore the three main components of a solution to this assignment: the voltage-divider circuit for the photo-resistor, reading the output of the voltage divider, and the code logic to control the LED display. This document could be used as a case study by instructors during an active-learning class, or for self-study by students. The bulk of the material is presented as a series of questions and answers.

2.1 Photo-resistor Circuit

The light sensor is called a photo-resistor or photocell. The top face is an exposed semiconductor with an electrical resistance dependent on the amount of light incident on the semiconductor. As the light intensity increases, the resistance of the semiconductor decreases. Limor Fried provides a good overview of photo-resistors at `http://www.ladyada.net/learn/sensors/cds.html`.

An Arduino can only measure voltage directly. The simplest way to get a voltage output from the photo-resistor is to put it into a voltage divider with a resistor having a fixed resistance. Therefore, the first step in the solution is to build the voltage divider circuit.

Photograph from
`www.bkbelectronics.com`

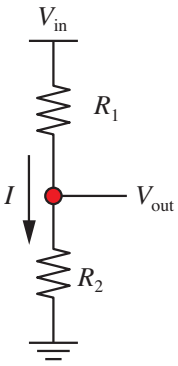
1. Before building the circuit, use a multimeter to measure the resistance of the photo-resistor in different light conditions, e.g., cover the sensor with your finger for the low light condition, and hold the sensor in the sunshine for the bright light condition. Does the resistance increase or decrease with an *increase* in ambient light levels?

   Answer: Resistance *decreases* as the photo-resistor is exposed to *increasing* light levels. For the photo-resistor in the Sparkfun kit, the resistance value varies from the low kΩ to above 20 kΩ.

2. Derive the equation for V_{out}/V_{in} of the voltage divider in the sketch below.

   Answer:

   $$\frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2} \qquad (1)$$

   To obtain this formula, start by applying Ohm's law to the series combination of resistors, and to the fixed resistor. Let I be the current flowing through both resistors to ground.

   $$V_{in} = I(R_1 + R_2) \quad \text{and} \quad V_{out} = IR_2$$

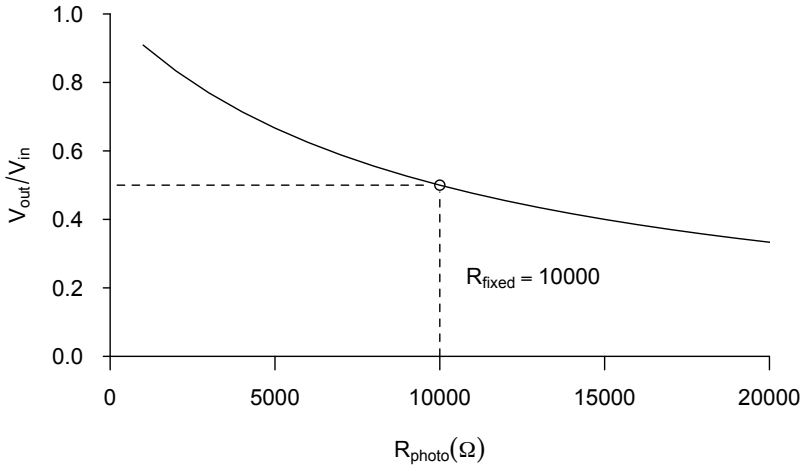
   Take the ratio and cancel the I

   $$\frac{V_{out}}{V_{in}} = \frac{IR_2}{I(R_1 + R_2)}$$

3. Should the photo-resistor be located "above" or "below" the fixed resistor? Why?

   Answer: When the photo-resitor is located on the upper half of the voltage divider, an increase in light intensity causes an increase in V_{out}. Therefore, for aesthetic and logical reasons, the photo-resistor should be located "above" the fixed resistor. However, the circuit would function in either configuration, and in some applications it may be desirable for the output to decrease as the brightness increases. A more detailed explanation follows.

   The following figure is a plot of Equation (1) for $R_2 = 10\,\text{k}\Omega$ and $1\,\text{k}\Omega \leq R_1 \leq 25\,\text{k}\Omega$. As R_1 decreases, V_{out}/V_{in} increases.



Assume that the photo-resistor is in the "upper" half of the voltage divider, i.e., in the circuit schematic, R_1 is the photo-resistor. As the light level increases, the resistance of the photo-resistor decreases. We can deduce the trend in the formula by looking at the limit as $R_1 \to 0$

$$\lim_{R_1 \to 0} \left[\frac{V_{\text{out}}}{V_{\text{in}}} \right] = \lim_{R_1 \to 0} \left[\frac{R_2}{R_1 + R_2} \right] = \lim_{R_1 \to 0} \left[\frac{R_2}{0 + R_2} \right] = 1$$

At the other extreme, the trend for decreasing light levels is apparent from taking the limit as $R_1 \to \infty$,

$$\lim_{R_1 \to \infty} \left[\frac{V_{\text{out}}}{V_{\text{in}}} \right] = \lim_{R_1 \to \infty} \left[\frac{R_2}{R_1 + R_2} \right] = \lim_{R_1 \to \infty} \left[\frac{R_2/R_1}{1 + R_2/R_1} \right] = \lim_{R_1 \to \infty} \left[\frac{0}{1} \right] = 0$$

Technically, there is no reason why the fixed resistor must be above the photo-resistor. However, aesthetically and intuitively it makes more sense to have V_{out} increase with increasing ambient light levels.

4. Assume that the nominal resistance of the photo-resistor in ambient light conditions is $10\,\text{k}\Omega$. What is a "good" value for the fixed resistor in the voltage divider?

   Answer: A "good" value for the fixed resistor is equal to the resistance of the photo-resistor at ambient light levels. For the resistors available in the Arduino Inventor's kit, use a fixed $10\,\text{k}\Omega$ resistor.

2.2 Analog Input

The Arduino Uno has six analog input pins. Connect one of those pins to the voltage divider.

1. Where should the analog input pin be connected to the voltage divider? In other words, where on the physical circuit should the jumper wire to the analog input pin be connected? Where is the ground wire for the analog input connected?

   Answer: The analog input pin is connected to the point between the photo-resistor and the fixed resistor. That location is indicated as V_{out} in the schematic.

There is no separate connection to ground because the analog input circuit uses the common ground on the Arduino board. If a multimeter is used to measure V_{out}, the negative probe of the multimeter needs to be connected to the ground pin on the Arduino.

2. What range of voltages should we expect to measure for V_{out}? How is V_{in} provided to the physical voltage divider circuit?

   Answer: The absolute limits for the range of V_{out} are $\min(V_{\text{out}}) = 0$ and $\max(V_{\text{out}}) = V_{\text{in}}$.

   Five volts is the maximum allowable voltage for analog input on an Arduino Uno. The natural way to satisfy this constraint for the photo-resistor circuit is to connect V_{in} to one of the 5V supply pins on the Arduino board.

   As a practical matter, values of $V_{\text{out}} = 0$ or $V_{\text{out}} = 5$ are never attained. The next question explores the actual range of V_{out} values encountered.

3. The output of the voltage divider is measured with the statement

   ```
   vphoto = analogRead( photo_pin );
   ```

   where `photo_pin` is an integer ($0 \leq$ `photo_pin` ≤ 5) designates the number of the analog input pin on the Arduino. Use the voltage divider formula and documentation of the `analogRead` function to predict the likely range of values stored in `vphoto`. For the sake of this exercise, assume that $R_2 = 10\,\text{k}\Omega$, $\min(R_1) = 3\,\text{k}\Omega$ and $\max(R_1) = 25\,\text{k}\Omega$.

   Answer: When the ambient light is at its *maximum* brightness the photoresistor will have its minimum resistance. Thus, at $\min(R_1) = 3\,\text{k}\Omega$,

   $$\frac{V_{\text{out}}}{V_{\text{in}}} = \frac{10}{3+10} = 0.769$$

   and the value returned by `analogRead` is

   $$\texttt{vphoto} = 0.769 \times 1023 = 787$$

   where the result of the multiplication has been rounded to the nearest integer value. Remember that the value returned by `analogRead` is an `int`.

   Repeat the calculation for $R_1 = 25\,\text{k}\Omega$.

   $$\frac{V_{\text{out}}}{V_{\text{in}}} = \frac{10}{25+10} = 0.286 \qquad \Longrightarrow \qquad \texttt{vphoto} = 0.286 \times 1023 = 293$$

   Therefore, for $R_2 = 10\,\text{k}\Omega$ and $3\,\text{k}\Omega \leq R_1 \leq 25\,\text{k}\Omega$, the analog input values should lie in the range $293 \leq$ `vphoto` ≤ 787. Note that only a fraction of the analog input range is used. This is characteristic of using voltage dividers to read sensors with a variable resistance response to a stimulus.

4. What range of voltages correspond to the range of `analogRead` values identified in the preceding question?

   Answer: For `vphoto` $= 293$:

   $$V_{\text{out}} = 0.286 \times V_{\text{in}} = 0.286 \times 5\,\text{V} = 1.43\,\text{V}$$

   For `vphoto` $= 787$:

   $$V_{\text{out}} = 0.769 \times V_{\text{in}} = 0.769 \times 5\,\text{V} = 3.65\,\text{V}$$

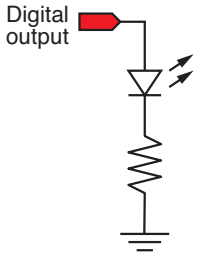
   Therefore, although the analog to digital converter on the Arduino Uno is capable of reading inputs from 0 V to 5 V, only part of that range is used by the voltage divider circuit.

The following table summarizes the results of the preceding calculations

Ambient light	R_1	$\frac{V_{out}}{V_{in}}$	`vphoto`	V_{out}
Maximum	3 kΩ	0.769	787	1.43 V
Minimum	25 kΩ	0.286	293	3.65 V

2.3 Logic for Controlling LED Display

A typical circuit for driving an LED is shown to the right.



Review questions:

1. Why is the resistor necessary?
2. Does it matter whether the resistor or the LED is "on top"?
3. What Arduino function (or functions) is used to turn a digital pin "on"?

After the basic circuit is set up, consider the following questions when developing the Arduino code.

1. Suggest ways of setting thresholds for "low light levels" and "medium light levels" as required in the assignment.

   Answer: There are at least three ways to do this. **(1)** Build the circuit and write an Arduino program to send the values from `analogRead` to the host PC using `Serial.print`. Expose the sensor to different light levels and choose threshold of `analogRead` values at appropriate light levels. This procedure is an informal calibration of your entire system.

   (2) Use the computations in a preceding exercise to predict the values returned by `analogRead`. Note that this requires knowledge of the value of the fixed resistor and samples of the photo-resistor output measured with a multimeter. You will need to verify that the system works by testing it at different light levels. The experiments will be a check on your calculations.

   (3) A more sophisticated technique would involve making readings in the `setup` function to determine threshold levels at runtime. That approach raises the question of a user interface: how does the user know when to expose the sensor to the different light levels that define the thresholds? One approach would be to blink each of the LEDs for a period of time during which the user should be holding the photo-resistor in different light levels. Another technique involves using a button or other switch to confirm when the sensor is in a calibration environment. Two such confirmations would be necessary.

2. Should the program use `if` constructs, or `if ...else` constructs?

   Answer: The use of the word "should" is a red herring. It's possible to create a working program using only `if` constructs, and it's possible to create a working program using only `if...else` constructs. As will be shown in the following exercises, the `if ...else` construct results in cleaner code that is easier to read and understand. An example program that only uses `if` constructs is given for completeness.

3. The code snippet to the right shows an attempt to make the LEDs turn on when the output of the voltage divider circuit is below the two thresholds, `vlow` and `vhigh`. Running this code on an Arduino (with the appropriate additions of the `setup` function and global definitions for `yellow_LED_pin` and `red_LED_pin`) results in the wrong behavior. The LEDs turn on when the photo-resistor is exposed to dim light conditions, but the LEDs do not turn off again when the photo-resistor is exposed to brighter light. Suggest a fix for this code.

```
void loop() {
  int vlow = 600, vhigh = 700;

  vphoto = analogRead( photo_pin );
  if ( vphoto <= vlow ) {
     digitalWrite( red_LED_pin, HIGH );
  }
  if ( vphoto <= vhigh ) {
    digitalWrite( yellow_LED_pin, HIGH );
  }
}
```

Answer: Use `else` clauses to turn the LEDs off when the condition tested in the `if` clause evaluates to FALSE. The code to the right shows how.

This is arguably the best solution to the assignment. Alternative Arduino codes are explored in the remainder of this document.

```
void loop() {
  int vlow = 600, vhigh = 700;

  vphoto = analogRead( photo_pin );
  if ( vphoto <= vlow ) {
     digitalWrite( red_LED_pin, HIGH );
  } else {
     digitalWrite( red_LED_pin, LOW );
  }
  if ( vphoto <= vhigh ) {
    digitalWrite( yellow_LED_pin, HIGH );
  } else {
     digitalWrite( yellow_LED_pin, LOW );
  }
}
```

4. Having seen the solution to the preceding question, a student thinks it would be nice to use a single `if ...else` structure to turn the LEDs on an off. The code to the right almost works. What is the incorrect behavior of this program? Try to figure out the behavior without running the program. After you are reasonably sure of your explanation, test your understanding by running the code on an Arduino.

```
void loop() {
  int vlow = 600, vhigh = 700;

  vphoto = analogRead( photo_pin );
  if ( vphoto <= vlow ) {
     digitalWrite( red_LED_pin, HIGH );
  } else if ( vphoto <= vhigh ) {
    digitalWrite( yellow_LED_pin, HIGH );
  } else {
     digitalWrite( red_LED_pin, LOW );
     digitalWrite( yellow_LED_pin, LOW );
  }
}
```

Answer: This program has two flaws. The conditions that clearly expose the cause of the incorrect behavior are a little difficult to recognize and isolate.

Once the yellow LED is turned on, it stays on until the the ambient light is brighter than the higher threshold (`vhigh`). As the light level is decreased, the yellow LED turns on as it should. As the light level decreases further, the red LED turns on (as it should) and the yellow LED *remains* on (which it shouldn't). An obvious indicator of this flaw is that both LEDs are on at the same time.

The second flaw is that once the light levels are low enough to turn on both LEDs, the red LED is not turned off again until the light level is *above* the higher threshold, i.e., until the light level is above the level that turns off the yellow LED. This flaw also has the obvious behavior that both both LEDs are on at the same time.

One solution is to add two lines of code for turning off the red LED and the yellow LED when they are not supposed to be turned on. That works, but the code is less clear and direct than the solution given for the preceding exercise. Reading the code, one may ask, "why is the red LED turned off for two different conditions?". The solution in the code to the right has the additional disadvantage that the logic for the red and yellow LEDs is mingled. In the preceding solution, there was a single `if...else` construct for each LED, which would make it simpler to remove one or the other LEDs if that change was desired.

```
void loop() {
  int vlow = 600, vhigh = 700;

  vphoto = analogRead( photo_pin );
  if ( vphoto <= vlow ) {
    digitalWrite( red_LED_pin, HIGH );
    digitalWrite( yellow_LED_pin, LOW );
  } else if ( vphoto <= vhigh ) {
    digitalWrite( red_LED_pin, LOW );
    digitalWrite( yellow_LED_pin, HIGH );
  } else {
    digitalWrite( red_LED_pin, LOW );
    digitalWrite( yellow_LED_pin, LOW );
  }
}
```

5. How would you implement the solution without using an `else` clause? Does this approach have any practical or aesthetic weaknesses?

   Answer: One solution is given in the code to the right. Two separate `if` constructs are used for each LED. One `if` clause determines whether to turn each LED on. A second `if` clause determines whether to each LED off.

   Aesthetically, this code is inferior because it is wordier than the solution that uses a single `if ...else` construct for each LED. Furthermore, it may cause someone unfamiliar with the intent of the author to wonder why the second test is necessary. Logically, the second test is redundant because any condition that fails the first test will pass the second test. (N.B. This depends on the nature of the test. For example, what if the logical test in the first clause was `vphoto < vlow`?)

```
void loop() {
  int vlow = 600, vhigh = 700;

  vphoto = analogRead( photo_pin );
  if ( vphoto <= vlow ) {
    digitalWrite( red_LED_pin, HIGH );
  }
  if ( vphoto > vlow ) {
    digitalWrite( red_LED_pin, LOW );
  }
  if ( vphoto <= vhigh ) {
    digitalWrite( yellow_LED_pin, HIGH );
  }
  if ( vphoto > vhigh ) {
    digitalWrite( yellow_LED_pin, LOW );
  }
}
```

   Given a choice, it is preferable to write code that conveys the desired outcome in the most straightforward and direct way. Hence the first solution using the `if...else` construct is preferred.

   Apart from the aesthetic weakness of this solution, the code makes the microcontroller perform two unnecessary tests on each pass through the loop. For this simple application, the inefficiency does not matter. In more complex situations, loss of efficiency due to unecessary work may be unacceptable.

6. A student is determined to satisfy the assignment using only `if` constructs. The student is also aware of the inefficiencies in the preceding solution. The code to the right appears to work. Do you see any problem with it?

```
void loop() {
  int vlow = 600, vhigh = 700;

  vphoto = analogRead( photo_pin );
  digitalWrite( yellow_LED_pin, LOW );
  if ( vphoto <= vlow ) {
    digitalWrite( yellow_LED_pin, HIGH );
  }
  digitalWrite( red_LED_pin, LOW );
  if ( vphoto <= vhigh ) {
    digitalWrite( red_LED_pin, HIGH );
  }
}
```

Answer: The code works by turning off both LEDs before deciding to turn them on. This will introduce an imperceptible off/on cycling of the LEDs that may make the LEDs appear to be very slightly dimmer than when they are on without the off/on cycling.

2.4 Practical Questions

1. Depending on the location and orientation of the photo-resistor relative to the LEDs, the photo-resistor reading may be affected by whether or not the LEDs are illuminated. You could fix this by placing a piece of cardboard between the light sensor and the LEDs, or by placing a short tube around the light sensor. The longer the tube, the more sensitive the sensor will be to the *direction* of the light source.