ECE 1001/1002 Introduction to Robotics
Lab #2: The Touchless Switch

Objectives

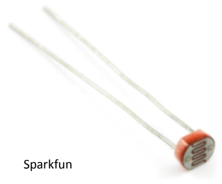
Introduction to Digital Input, and the photoresistor

Requirements and Signoffs

Lab #1 first had you turn the LED on and off by moving wires, about like you do with wall switches. The second part had the Arduino run by itself, endlessly turning an LED on and off. In this lab, you will again have and LED to turn on and off, but now you will control when it's on and off without moving a switch or wires. You will do that with a light-controlled switch—when light shines on it the LED will be off, and when light doesn't shine on the switch, the LED will be on. Just like you want with a night-light, for instance.

Light Control Circuit

The kit includes a photoresistor, which looks like the picture below, with a squiggly line on top. The squiggly line is made of a semiconductor, when exposed to light the squiggly semiconductor line conducts electricity somewhat well (like a 1,000Ω resistor), and when in the dark, it conducts electricity very poorly (like a 50,000Ω to 1,000,000Ω resistor depending on how dark).

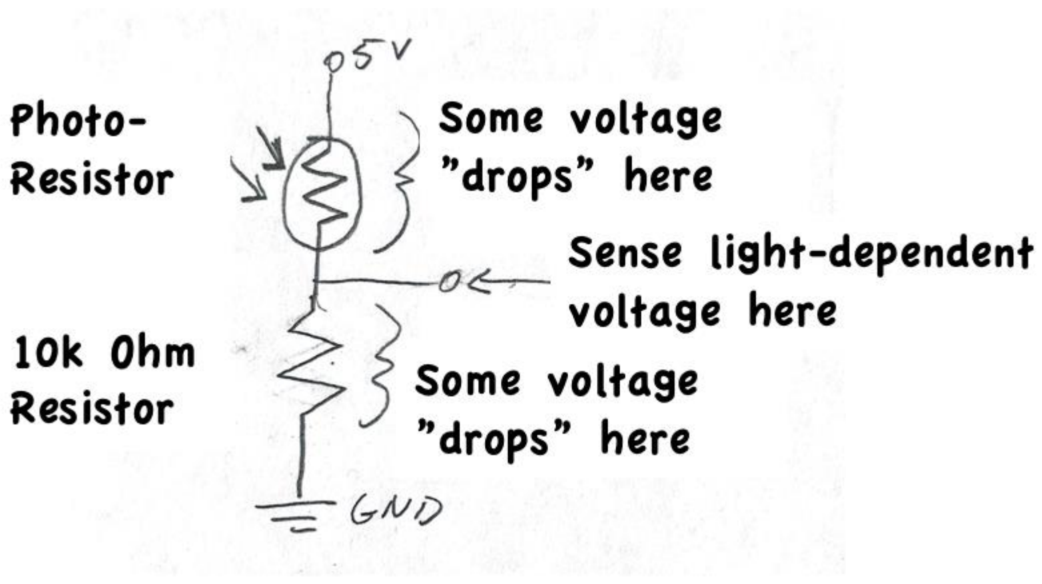


A circuit called a voltage divider can be easily built which will produce a "LOW" voltage when the resistor is exposed to light, and a "HIGH" voltage when not exposed to light[1] (in later labs the photoresistor will be used to provide much finer light sensitivity). These low and high voltage values will not be 0V and 5V, but the Arduino can sense the difference between these voltages, accepting the higher voltage as a HIGH signal, and the lower voltage as a LOW signal.

As shown below, the photoresistor and a 10,000Ω resistor are connected in a series combination between 5V and Ground. One leg of the photoresistor is connected to 5V (the photoresistor is not polarized, so either leg can be connected to 5V). The other leg is connected to the 10,000Ω resistor which is then connected to ground. Some of the 5V power supply voltage is "dropped" across the photoresistor, and some is dropped across the 10,000Ω resistor.

Between the photoresistor and 10,000Ω resistor a variable voltage can be sensed which depends on the amount of light shining on the photoresistor.

[1] It's just as easy to reverse that, producing a HIGH voltage when the resistor is in the dark. If you don't follow the instructions, this may happen to you.

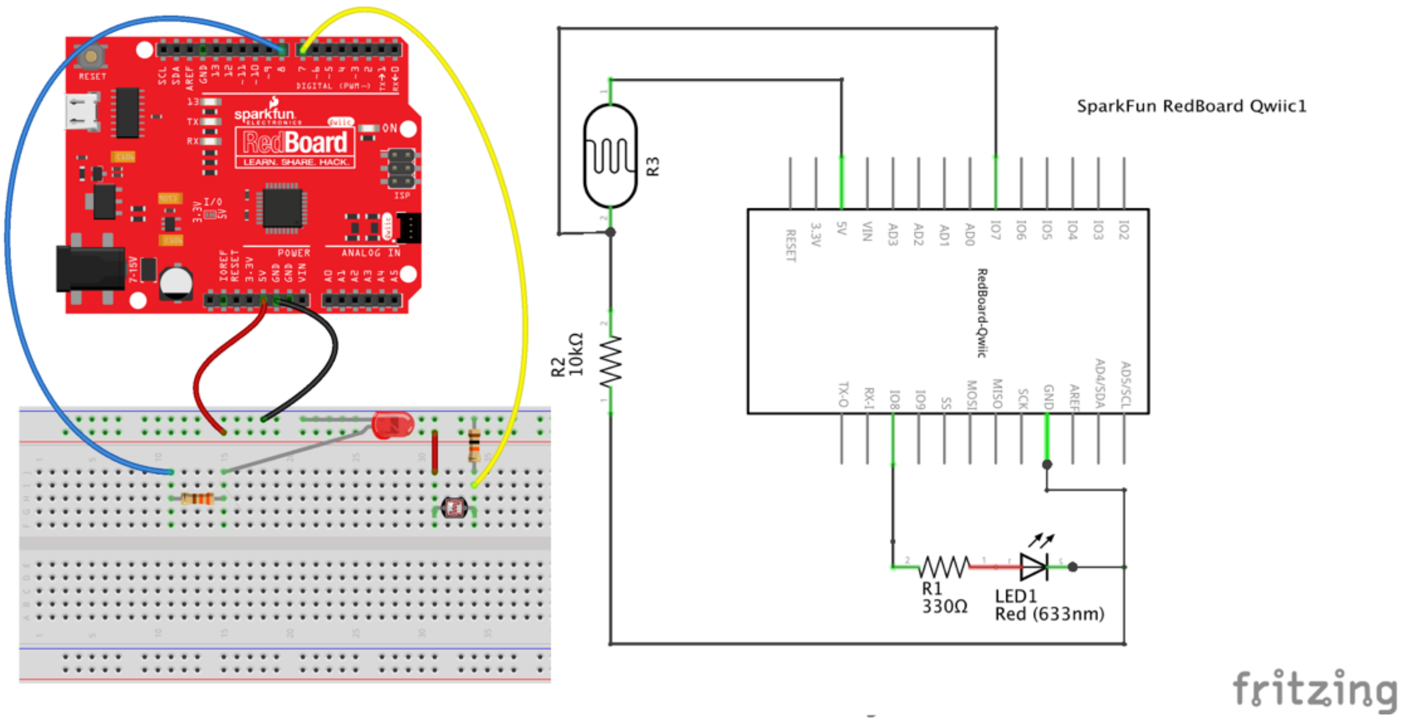

5V
Photo-Resistor
Some voltage "drops" here
Sense light-dependent voltage here
10k Ohm Resistor
Some voltage "drops" here
GND

- If the light is very bright on the photoresistor, the photoresistor's resistance will be smaller; only a fraction of the 5V will be dropped in the photoresistor, and the sense wire will have a fairly high voltage on it.
- If the photoresistor is in the dark (for instance, covered a hand. Be careful, though, don't touch the photoresistor leads, it's not dangerous but it might change the reading) the resistance in the photoresistor will be quite high, and almost the entire 5V will drop across the photoresistor. That will leave the sense wire with a low voltage on it, close to ground level.

Look at the circuit below. The LED control circuit is the same as the last lab you just finished; the Arduino's pin 8 is connected to a 330Ω resistor and LED in series, with the other end of the LED making a complete circuit to ground. That makes the computer-controlled light.

The yellow wire is now the sense wire, and it is connected to the Arduino's pin 7. If pin 7 is configured as an input, then the software running on the Arduino can check the voltage value on pin 7, and if the voltage is LOW (which happens when the photoresistor is in the dark) the Arduino software could, for instance, turn on the LED. And when the voltage on pin 7 is HIGH, which happens when the photoresistor is exposed to light, the software can turn off the LED connected to pin 8. This means the Arduino can make decisions and actions by itself based on input from the world. This decision-making process is called an algorithm.

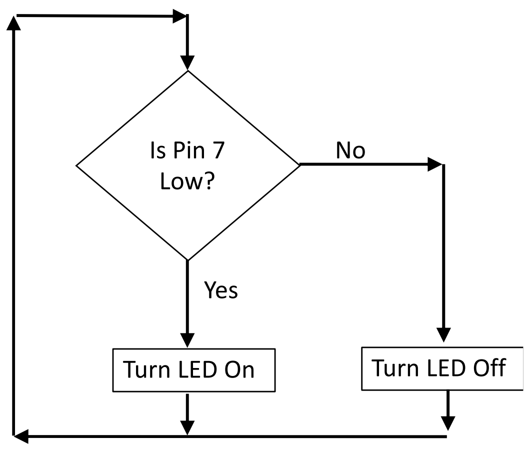
Build the circuit below now. In addition to the LED and resistor from Lab 1, you need the photoresistor and a 10kΩ resistor.



Light Controlling Software Testing

The software for turning on the light is a simple algorithm; the Arduino checks the light level with the photoresistor connected to pin 7 (actually, the Arduino checks the voltage level, which is dependent on the light level) and turns the LED on if the light level is low. Then the void loop () Arduino code will repeat the light check. This is such a simple algorithm that it might seem obvious. But the details can be confusing, so looking carefully may help.

Flow charts are a common way to illustrate algorithms. Flow charts are nice because they graphically represent concepts; you can even trace actions out with your finger. A flow chart is made with action rectangles and question diamonds. The questions must be yes/no questions. And for each case “yes” or “no” there is an arrow leading to an action. Then the arrows loop back to ask the question again, just as the loop (); statement in Arduino code does.



The question asked in the diamond of the flow chart is this: "is Pin 7 Low?" That's the question the Arduino will make a decision based on. If the voltage level on pin 7 corresponds to a low logic level, the photoresistor is in the dark and the LED should be turned on. If the voltage on pin 7 is not low (that means of course, it is a high logic level) then the LED should be turned off, since the light level is high

The setup again initializes pin 8 as an output (for the LED), but now pin 7 is initialized as an input. This input reads the voltage which changes due to light hitting the photoresistor.

```
void setup()
{
  pinMode(8, OUTPUT);   // initialize pin 8 as an output to control the LED
  pinMode(7, INPUT);   // initialize pin 7 as an input to read the light sensor
}
```

In the main loop (), there is now decision-making logic corresponding to the flow chart:

- If the voltage on pin 7 is low, the LED is turned on
- Otherwise, the LED attached to pin 8 is turned off

Study the code below, can you see this logic?

```
if (digitalRead(7) == LOW)
  {
  digitalWrite(8, HIGH);
  }
else
  {
  digitalWrite(8, LOW);
  }
```

If you can't that's OK, let's go through it. The logic to turn the LED on is this: If the voltage on pin 7 is low, the LED is turned on. To do this the code has an "if" statement. If statements (and various other logic control statements) are always followed by a parenthesis (). Inside the parenthesis is a question (or sometimes, a series of questions). If whatever is inside the parentheses () is true (or yes, or HIGH or “1”) then the Arduino will do whatever is between the curly braces {}. Stripping out the comments the control statement is this:

```
if (digitalRead(7) == LOW)
   {
   digitalWrite(8, HIGH);
   }
```

The two equal signs == is the way logic questions in the C language can ask if something equals a specific value. In this case the question is this: is pin 7 equal to a LOW voltage level? Computer languages require specific, unambiguous statements. A single equals sign “=” has several meanings in common use and specific mathematical use. To be unambiguous, C uses two equal signs to ask the question: does something equal something else? More on this later!

If the question (digitalRead(7)==LOW) evaluates to yes or TRUE, then the Arduino will do what’s between the curly braces which is to turn on the LED attached to pin 8:

```
{
  digitalWrite(8, HIGH);
}
```

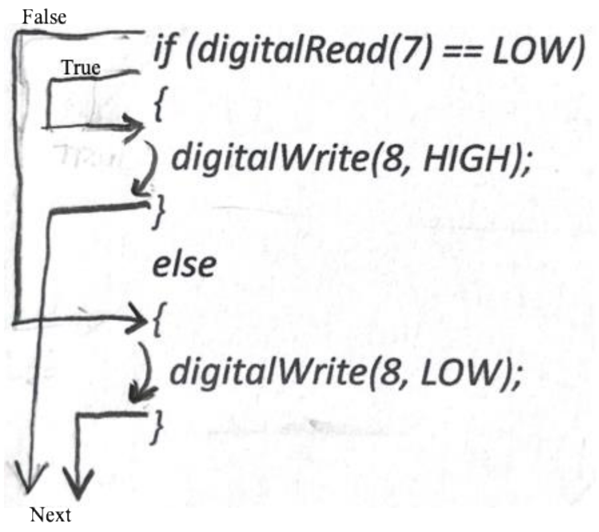
Is it confusing that a LOW voltage value (or NO or FALSE or “0”) level evaluates to a true statement? Logic can be confusing, which is one reason flow charts are helpful.

The next statement with the “else” is easier, it simply is a kind of default action, which happens if the question is answered as false, which happens with a HIGH voltage value, meaning the input **is not LOW**.

```
else
 {
 digitalWrite(8, LOW);
 }
```

For the situation where the “if” statement above it evaluates to a false condition (or NO, or LOW or a “0”) then the LED on pin 8 should be turned off. That happens, If the photoresistor sees light, it has a low resistance, which causes a high voltage and the “if statement” to be false. When an “if statement” is false the things between the following curly braces {} are not done. Instead, the program execution skips around them, to whatever is below. When there is an “else” statement after an “if” statement, the Arduino will do whatever is inside the “else” statement’s curly braces {} whenever the if question is false.

The Arduino will never do both the commands in the “if” statement and the commands in the “else” statement; only one set of these commands will be done.



```
if (digitalRead(7) == LOW)
{
digitalWrite(8, HIGH);
}
else
{
digitalWrite(8, LOW);
}
```

Normally, every "if" question should have an "else" option for the cases when the "if" evaluates as false. Even when there is no action to take, having the "else" statement will make your logic easier to follow and understand by others.

Compile the code and upload it to the properly wired Arduino. Does the LED turn on when the photoresistor is in the dark? How light does it need to be to turn on? What if you use the bright LED flashlight from your phone to illuminate the photoresistor? Try various conditions. If you have trouble getting the light to switch, try moving the photoresistor from the 5V to 3.3V will make it more reliable. Ask for help if you are confused.

Change the hardware and software for signoff

Now that you've got a light switch operating without touch, modify the hardware and software so that that the input is taken from pin 9, and the LED is driven from pin 6. Test your new arrangement, once you have it working, show it to the instructor or TA for signoff.

Your instructor will either have you print off a sign-off sheet to sign or have a roster to check your name off. To receive credit for this lab, you *must* get the sign-off before the due date (see Canvas), during class time or with a TA during their hours. The instructor will not signoff outside of class hours.