CG1111 Engineering Principles and Practice I

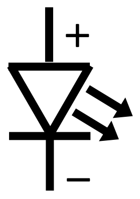
Photoelectric Sensors: Characterization & Applications

(Week 10, Studios 1) **[Graded Lab Report]**

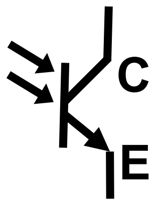
Time	Duration (mins)	Activity
0:00	10	Briefing
0:10	60	Activity #1: IR Proximity Sensor Characterization
1:10	80	Activity #2: LDR Colour Sensor
2:30	5	Final discussions and wrap-up

Introduction:

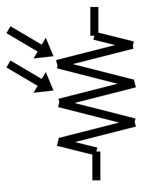
- In this studio, we will be exploring two major photoelectric sensors – infrared (IR) and light dependent resistor (LDR). They are commonly used to estimate distance (proximity) or determine the presence (or absence) of an object by making use of IR and visible light respectively.
- IR sensor consists of an IR LED (emitter) and an IR phototransistor (detector).
- IR LED is a special purpose LED that emits light in the IR range of wavelength.
- IR phototransistor works like a normal transistor, where the amount of collector current produced depends on the amount of current at its base, except that the base current is controlled by the amount of IR light incident on the phototransistor. When its base region detects IR illumination, a current proportional to the amount of illumination starts to flow from its collector (C) to its emitter (E) thus creating a potential difference across the C and E terminals that varies proportionally to the amount of illumination.
- When the IR LED and phototransistor are placed side by side and an opaque object is placed in front of them, this indirect incidence setup of the IR sensor enables it to function as a proximity sensor (as in Activity #1).
- The distance of the opaque object from the IR sensor affects the amount of reflected IR light incident on the IR detector, which in turns affects the amount of voltage drop across the IR detector that can be used to estimate the proximity of the object from the sensor.
- LDR is essentially a light-controlled variable resistor whose resistance varies inversely proportional to the intensity of light. In the dark, the LDR resistance can be as high as several mega ohms, while under bright light its resistance is only in the order of a few 100 ohms.
- The IR LED, IR phototransistor and LDR are represented symbolically in electric circuits as shown below.



IR LED



IR Phototransistor


LDR

Activity #1: IR Proximity Sensor Characterization (60 mins)

Introduction:

In this activity, we will be building our own IR proximity sensor using an IR LED and an IR phototransistor arranged in an indirect incidence setup. The circuit schematic is as follows.

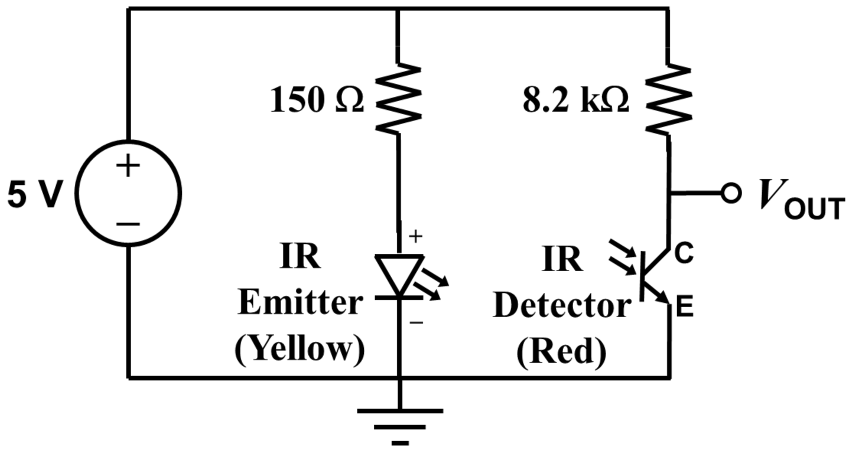


Figure 1. IR sensor circuit for determining its optimal working distance.

Objectives:

- To understand the operation of an IR sensor through observing its electrical properties
- To be able to setup an indirect incidence sensing circuit using IR emitter and detector
- To learn about sensor sensitivity and the importance of optimal sensor working distance

Equipment and Materials:

- IR LED (Emitter)
- IR Phototransistor (Detector)
- 5% tolerance resistors {150 Ω and 8.2 kΩ}
- Breadboard and connecting wires
- USB breakout cable + your own USB charger
- Digital multimeter
- Ruler (15 cm)
- White “maze wall” board

Procedure:

1. Construct the circuit shown in Figure 1 on your breadboard. Ensure that the IR emitter (dotted yellow) and the IR detector (dotted red) are placed in a straight line, close to each other with their protruding lenses (the bumps) **facing the same direction**, preferably out of the nearest breadboard edge and avoid facing any potential IR source, e.g. sunlight. With reference to the *IR Emitter* and *IR Detector datasheets* provided, which piece of information hints that the IR pair should be placed close to each other for optimum result? Briefly explain your choice.

2. Place a ruler on the table top, with its **0 cm marking right under the lenses**. Hold the breadboard and ruler down with adhesive tape or a thin layer of blu-tack if necessary as it is important to measure the distance accurately in this activity.
3. Check that your USB breakout cable is supplying a voltage of about 5 V before connecting it to the circuit.
4. Now place the white "maze wall" board provided at the 1 cm mark. To be consistent, the side of the board that is facing the sensor should always be exactly on the mark and parallel to the sensor.
5. Measure V_{OUT} (to 2 decimal places) of the detector with the digital multimeter and record the reading in Table 1.
6. Repeat the above two steps for all other distances in Table 1. Record your observations.
7. Scatter plot V_{OUT} vs distance. Properly label it and include it in your report.
8. Based on your plot, briefly describe the **relationship** between V_{OUT} and the distance of the object from the sensor. What can you say about the **sensitivity** of this IR sensor operating in this arrangement?
9. In your opinion, what is the **optimal working distance** for this IR sensor?

Table 1. IR Proximity Sensor: Distance and V_{OUT} relationship.

Distance between object & sensor (cm)	V_{OUT} (V)
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Activity #2: LDR Colour Sensor (80 mins)

Introduction:

White light is made up of three primary colours – red, green and blue; all colours can be derived from components of these three colours. Different object absorbs different amount of each of the three colours of light, resulting in the intensity of the reflected light for each of the three colours to vary. In other words, the colour of an object we perceive is the colour (or a combination of colours) of light being reflected by that object while all the other colours get absorbed.

Based on this principle, we will build a simple colour sensor which consists of an LDR, a tri-colour RGB LED lamp and an Arduino Uno, as shown in Figure 2. At intervals apart, the Arduino turns on one colour of the RGB LED lamp at a time that shines directly at an object. The LDR resistance varies according to the reflected light intensity it perceives, for each of the three colours shone. For each colour, the Arduino analog-to-digital converter (ADC) at port A0 reads in the corresponding voltage. The program then calculates and concludes by assigning a corresponding RGB colour code (e.g. 255, 255, 255 for white) based on its calibrated maximum light intensity ranges (one range for each colour, deduced from the intensity difference between the white and black samples) and the intensities (one intensity value for each colour) of the reflected colour lights from the object.

If the calibration and experiment are done correctly, the RGB colour code generated should yield the actual colour of the object using any decent RGB calculator or colour code chart – this also serves to verify that the sensor is indeed working properly. Hence, the concepts of sensor response time, sensor calibration, verification and sensor design for a specific application will also be covered in this activity.

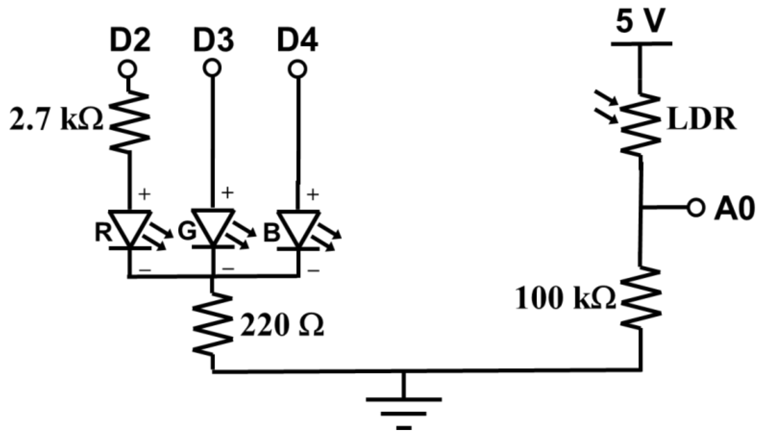


Figure 2. Colour sensor circuit consists of an LDR, a RGB LED and an Arduino Uno (ports shown).

Objectives:

- To understand the operation of an LDR through observing its electrical properties
- To learn about the importance of sensor response time, calibration and verification
- To be able to design a colour sensor using an LDR, a RGB LED and an Arduino Uno

Equipment and Materials:

- LDR (7 mm diameter)
- RGB LED lamp (common cathode)
- 5% tolerance resistors {220 Ω, 2.7 kΩ and 100 kΩ}
- Breadboard and connecting wires
- Arduino Uno and a USB cable
- USB breakout cable + your own USB charger
- Digital multimeter
- 5 x White 80 gsm A4 paper (aka White Sample)
- 1 x Black 160 gsm A4 paper (aka Black Sample)
- 1 x Colour 160 gsm A4 paper (random colour, aka Colour Sample)
- 2 x plastic rectangular drawer trays (for supporting the Samples above the sensor circuit)

Procedure:

A. <u>LDR Response Time</u>
Sensors do not change output state immediately when an input parameter change occurs. In order for the colour sensor to work properly, the LDR response time must be known before the Arduino can be programmed to read in its value. We shall find out how long it takes for the LDR to stabilize before it can be used reliably.

1. Construct the colour sensor circuit as shown in Figure 2. The LDR and the RGB LED should be placed close to but not obstructing each other. Particularly, take note that the **LDR flat top should not be covering the RGB LED and it should be slightly higher than the LED lens** on the breadboard to avoid being shone on directly by the LED.
2. Copy and paste the Arduino code that examines the LDR response time from the text file **LDRresponse.txt** onto a new sketch and save it under an appropriate file name. Study the code and explain why `pinmode()` instruction for the LDR sensor pin is not necessary (hence missing in the code) and what the line of code `Serial.print(millis());` does.
3. Place the two plastic rectangular trays on their sides to provide support to the sample. The rested sample should be **roughly 5 cm from the table top or about 1 – 3 cm away from the LDR flat top**.
4. Get ready your three samples. You will be using them immediately one after another, in the sequence of White, Black and Colour.
5. Compile your sketch and evoke the Serial Monitor from Tools.
6. Follow the instructions on the Serial Monitor. You have **5 seconds to get ready each required sample**.
7. The Serial Monitor continuously prints a millisecond timestamp follows by a comma and the most current calculated intensity value for the colour of light shown in the preceding colour header.
8. When "Experiment Completed." is shown, cut and paste all the numbers, including their colour headers, to a new text file. Save the text file somewhere you can retrieve it easily.

9. Open an Excel file and under Data tab, choose **Get External Data -> From Text** (or its equivalent command in your version of Excel) to import the data from the text file, and ensure the **Delimited** and **Comma** delimiter options are both chosen.
10. Treating the first timestamp of each colour to be 0 sec (i.e. at the origin), use Excel to generate the new time data.
11. The raw data may contain some negative values at the beginning of data collection, especially for Red light. Replace all these negative values with zeros for simplicity.
12. Scatter plot the graphs of R, G and B values vs time, preferably all in one plot. Include your plot(s).
13. Determine the **LDR response time** from your graphs. It is the time your LDR takes to stabilize.

B. Importance of Proper Calibration

1. Refer to the original code used for the LDR response time experiment. Identify the function responsible for the **sensor calibration**. It is a process that you have to go through each time the board is reset.
2. Should the calibration function be **omitted** so as to increase the availability of the sensor system? Why or why not? Think in terms of the sensor's overall performance.
3. What happens if a **not-so-white** object is presented as white sample or a **not-so-black** object is presented as black sample? Explain in terms of the computed RGB code values of the colour sample.

C. Colour Sensor Verification **[NOT GRADED]**

1. Copy and paste the second Arduino code from the text file **ColourSensor.txt** onto a new sketch and save it under an appropriate file name.
2. Use the same colour sensor circuit as shown in Figure 2 and prepare the trays for support as before.
3. Compile your sketch and evoke the Serial Monitor from Tools.
4. Follow the instructions on the Serial Monitor and perform the calibration as before.
5. When "Colour Sensor Is Ready." is shown, go ahead and test the colour sensor with any object whose RGB code you are interested in finding out. You may swap your colour sample with your neighbour. Note down the RGB code for one particular object.
6. Visit any online RGB calculator, such as https://www.w3schools.com/colors/colors_rgb.asp, to verify that your colour sensor works. It is likely that the colour you get from the calculator is a darker shade of the actual colour of the object. Do you know why?
7. Now that you know how your colour sensor operates (LDR response time, calibration details, behaviours of the functions in the code), feel free to enhance your colour sensor, either through hardware or software, or even by tweaking the calibration process, such that it can give fairly accurate RGB code that matches the colour of your object.
8. Note down the new RGB code. Compare it with the one obtained in step 5 above. Is you sensor performing better now? Regardless, you have just completed the sensor design for a specific application.

END OF STUDIO SESSION