CIS 316

# Light Reactive Lamp

## Parts Required:

Light Dependent Resistor (LDR) x 1

Red LED x 1

Green LED x 1

Yellow LED x 1

2.2k OHM Resistor x1 (Ideally we will use a 2.2k but you can place 2 1k resistors in a series for this lab) - <https://www.youtube.com/watch?v=qqhz3mGdgjc>

Regular Jumper Wires x 8 (+2 extra)

U-shaped Jumper Wires or Regular Jumper Wires x6

Breadboard

USB cable (to connect the pc to the Arduino)

Arduino x 1

Optional - Light shard or Paper Lantern

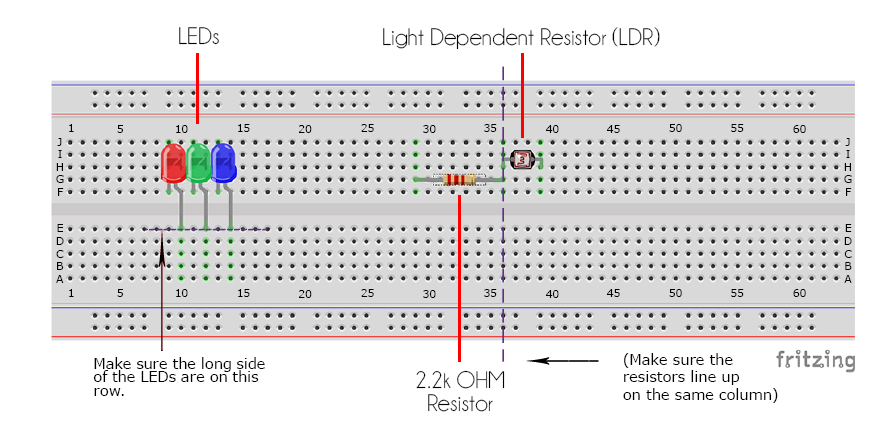
Goal:

To create a lamp that change colors and only turns on when it is dark.

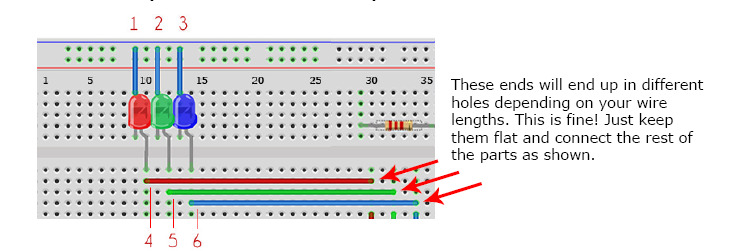
In Part 1 we will build the hardware, and in Part 2 we will program it.

## Part 1 – Building the Lamp

Step 1 Connect the components

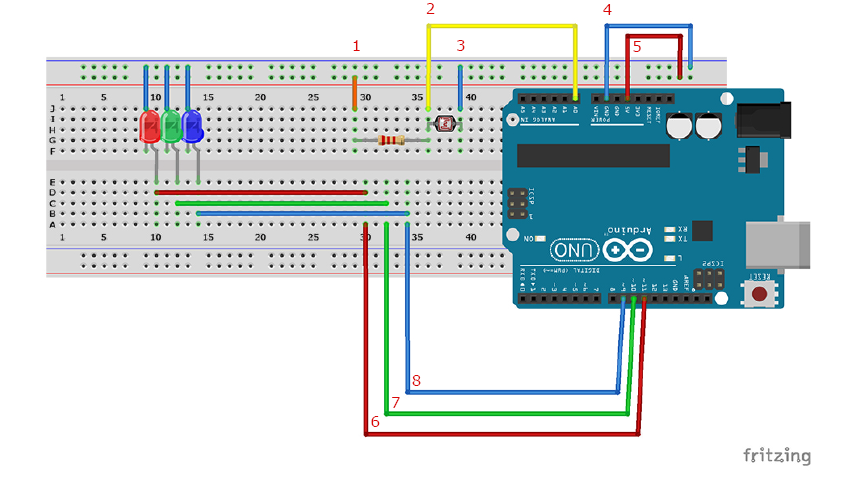


Step 2 – Connect the Jumper Wires – (If you do not have flat jumper wires I would recommend turning the wires to the side if you plan to use a lamp.



Step 3 – Connect the Hardware. For this step you will attached the remaining 8 wires as shown in the diagram.

Note: The picture makes it look like the arduino is mounted to the breadboard. It is not, you can put it anywhere as long as you have connected the cables.



Optional: Step 4 – This is where you would place a paper lantern over the lights or the breadboard.

## Part 2 – Programming

In this part we will program the arduino to control the lights. You will need to us the Arduino Softwrae (IDE) for this part of the lab.

Step 1 – Open the Arduino Software (IDE)

Step 2 – Open a new program by navigating to ‘File > New’

Step 3 – Delete all of the existing text/code

Step 4 – Navigate to ‘File > Save’ and save the program as “316Lampxx” where xx is your initials.

Step 5 – Type the program into your software.

Information: Lines that start with // or /\* indicate comments. // is used for single line comments and /\* \*/ are used in multi line comments. Lines that begin like this are invisible to the program and are provided as explanations and not actual code. You are not required to add the comments to the program just the code. If you are creating a program to distribute to others or plan to review code at a later date, it is considered best practice to provide comments.

/\* Lamp Lab Information

This lamp should cycle through a color spectrum and will only turn on when it is dark.

Examples of color equations:

Red = sin(x)

Green = sin(x + PI/3)

Yellow = sin(x +2PI/3)

These equations are used in the program to calculate the brightness of the LEDs. \*/

//Input user defined variables. Think of variables like containers they are used to store information.

int pulseSpeed = 5;  
// This value controls how fast the mood lamp runs. You can replace this with any whole number.   
int ldrPin = 0; // LDR in Analog Input 0 to read the surrounding light.  
int redLed = 11; // red LED in Digital Pin 11.  
int greenLed = 10; // green LED in Digital Pin 10.  
int yellowLed = 9; // yellow LED in DIgital Pin 9.  
// These are the pins we are using with the Arduino. You can see the numbers on the board itself.  
int ambientLight;   
// This variable stores the value of the light in the room.  
int power = 150;  
// This variable controls the brightness of the lamp (2-255).  
float RGB[3];  
// This is an ‘array’. It can hold 3 values: RGB[0], RGB[1], and RGB[2].We’ll use this to store the values of the Red, Yellow, and Green LEDs.  
float CommonMathVariable = 180/PI;  
/\* We will be using the value of 180/PI a lot in the main loop, so to  
save time, we will calculate it once here in the setup and store it   
in CommonMathVariable. Note: it is PI, not P1 \*/  
  
/\* Step 2: Create Setup Loop  
This ‘loop’ is not really a loop. It runs once in the beginning to create  
the default values for our LEDs. \*/  
  
void setup () {  
  
pinMode (redLed,OUTPUT);  
pinMode (greenLed,OUTPUT);  
pinMode (yellowLed,OUTPUT);  
// This tells the Arduino to send data out to the LEDs.  
  
digitalWrite (redLed,LOW);  
digitalWrite (greenLed,LOW);  
digitalWrite (yellowLed,LOW);  
// This sets all the outputs (LEDs) to low (as in off), so that they do not turn on during startup.  
  
} // Opening brackets must be accompanied by closing brackets.  
  
/\* Step 3: Create Main Loop  
The previous sections are where we set up the variables. This section is   
where we put them to work! This part of the program is a ‘loop’. It   
repeats itself over and over again, making a small change in the   
brightness of the LEDs each time - this creates a smooth transition in   
color. \*/  
  
void loop () {  
  
for (float x = 0; x < PI; x = x + 0.00001) {  
RGB[0] = power \* abs(sin(x \* (CommonMathVariable))); // Red LED.  
RGB[1] = power \* abs(sin((x + PI/3) \* (CommonMathVariable))); // Green LED.  
RGB[2] = power \* abs(sin((x + (2 \* PI) / 3) \* (CommonMathVariable))); // Yellow LED.  
ambientLight = analogRead(ldrPin);  
// This reads the light in the room and stores it as a number.  
  
if (ambientLight > 600) {  
// This ‘if statement’ will make the lamp turn on only if it is dark. The darker it is, the higher the number.   
analogWrite (redLed,RGB[0]);  
analogWrite (greenLed,RGB[1]);  
analogWrite (yellowLed,RGB[2]);  
// These ‘analogWrite’ statements will send the values calculated above to the LEDs.   
} // Don’t forget to close this ‘if statement’ with a bracket!  
  
else {  
digitalWrite (redLed,LOW);  
digitalWrite (greenLed,LOW);  
digitalWrite (yellowLed,LOW);  
}  
// This ‘else statement’ will only activate if the ‘if statement’ above does not (ie. If it is too bright in the room). The LEDs will turn off.   
  
for (int i = 0; i < 3; i++) {  
// This loop calculates the delay for each color depending on its current brightness. Brighter LEDs will change color slower and vice versa.  
  
if (RGB[i] < 1) {  
delay (20 \* pulseSpeed);  
}  
   
else if (RGB[i] < 5) {  
delay (10 \* pulseSpeed);  
}  
  
else if (RGB[i] < 10) {  
delay (2 \* pulseSpeed);  
}  
  
else if (RGB[i] < 100) {  
delay (1 \* pulseSpeed);  
}   
// ’else if’ means only one of these conditions can activate at a time.   
  
else {}  
// This blank ‘else statement’ is a fail-safe mechanism. It instructs the program to do nothing if the conditions above do not activate. This prevents the program from generating errors when calculating delays.   
}  
  
delay(1);  
// This delay gives the light dependent resistor time to settle and give accurate readings.   
}  
} // Don’t forget to close with these brackets!

Step 6 – Connect the board to the computer with the usb cable.

Step 7 – Navigate to ‘Tools > Port’ then select the port that is connected to your Arduino. If you have trouble finding your post you can try disconnecting another USB device.

Step 8 – Navigate to ‘Tools > Board’ and select the Arduino.

Step 9 – Navigate to ‘Sketch > Upload’

You should see Done uploading at the bottom of your screen. If you see errors in your code god back and correct the errors.

Step 10 - The project should now be working. If it is not working review the suggestions below.

If you have errors review the following possible issues. If your program is working move on to Step 11.

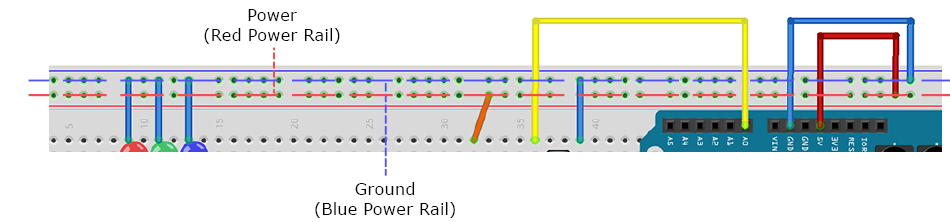
Program issues:

- if a line of your code is highlighted red, there is an error on that line. Correct the error and try again.

- common issues include: typos, missing ‘;’ and missing/extra brackets. (examples ‘{‘ ‘(‘) You must always close an open bracket or you may have too many brackets. Both of these issues will cause errors. Make sure all brackets are placed correctly.

Hardware issues: (if your code uploads fine but the project still is not working there are errors with the hardware setup.

- review if the wires are connected to the correct power rail (row) See image provided below



- pins/wires are not aligned properly. (Columns) Review where the pins occur along the column to verify they are lining up.

Step 11 – Now that the project is setup, complete the following.

Exercise 1 – Change the overall brightness of the lamp. Find the variable that controls this, and change its value. Hint: The max value is 255

**Answer: (Provide in your answer how do you accomplish this, and include the lines of code you changed)**

I changed the values calculated for each led in the RGB array by changing them from sin to cos and tan.

RGB[0] = power \* abs(cos(x \* (CommonMathVariable))); // Red LED.

RGB[1] = power \* abs(cos((x + PI/3) \* (CommonMathVariable))); // Green LED.

RGB[2] = power \* abs(tan((x + (2 \* PI) / 3) \* (CommonMathVariable))); // Yellow LED.

Exercise 2 – Make the lamp change colors faster. Hint: There are 3 ways to do this.

What are two ways this can be accomplished? Provide examples of how the code would be changed.

**Answer: Provide an explanation of how you can accomplish this, and the lines of code that you have changed to make this work.**

The first two ways that come to mind are to change the pulseSpeed variable to something else:

int pulseSpeed = 15;

OR you could change the delay here:

if (RGB[i] < 1) {

delay (30 \* pulseSpeed);

}

else if (RGB[i] < 5) {

delay (20 \* pulseSpeed);

}

else if (RGB[i] < 10) {

delay (5 \* pulseSpeed);

}

else if (RGB[i] < 100) {

delay (2 \* pulseSpeed);

}

Exercise 3 – Create a new user-defined variable to control how dark it would need to be for the lamp to turn on. You will call this variable ‘ambientLimit’ You will use this variable to replace the default number in the code. Hint: You will have to declare the variable as an ‘int’ and the value can range from 0-1024.

**Answer: Provide an explanation on how you accomplished this and a copy of the code used or changed.**

Declare ambientLimit as an int equal to 500 and then check to see if ambientLight is greater than ambientLimit.

// int ambientLight;

int ambientLimit = 500;

// This variable stores the value of the light in the room.

if (ambientLight > ambientLimit) {

// This ‘if statement’ will make the lamp turn on only if it is dark. The darker it is, the higher the number.

analogWrite (redLed,RGB[0]);

analogWrite (greenLed,RGB[1]);

analogWrite (yellowLed,RGB[2]);

// These ‘analogWrite’ statements will send the values calculated above to the LEDs.

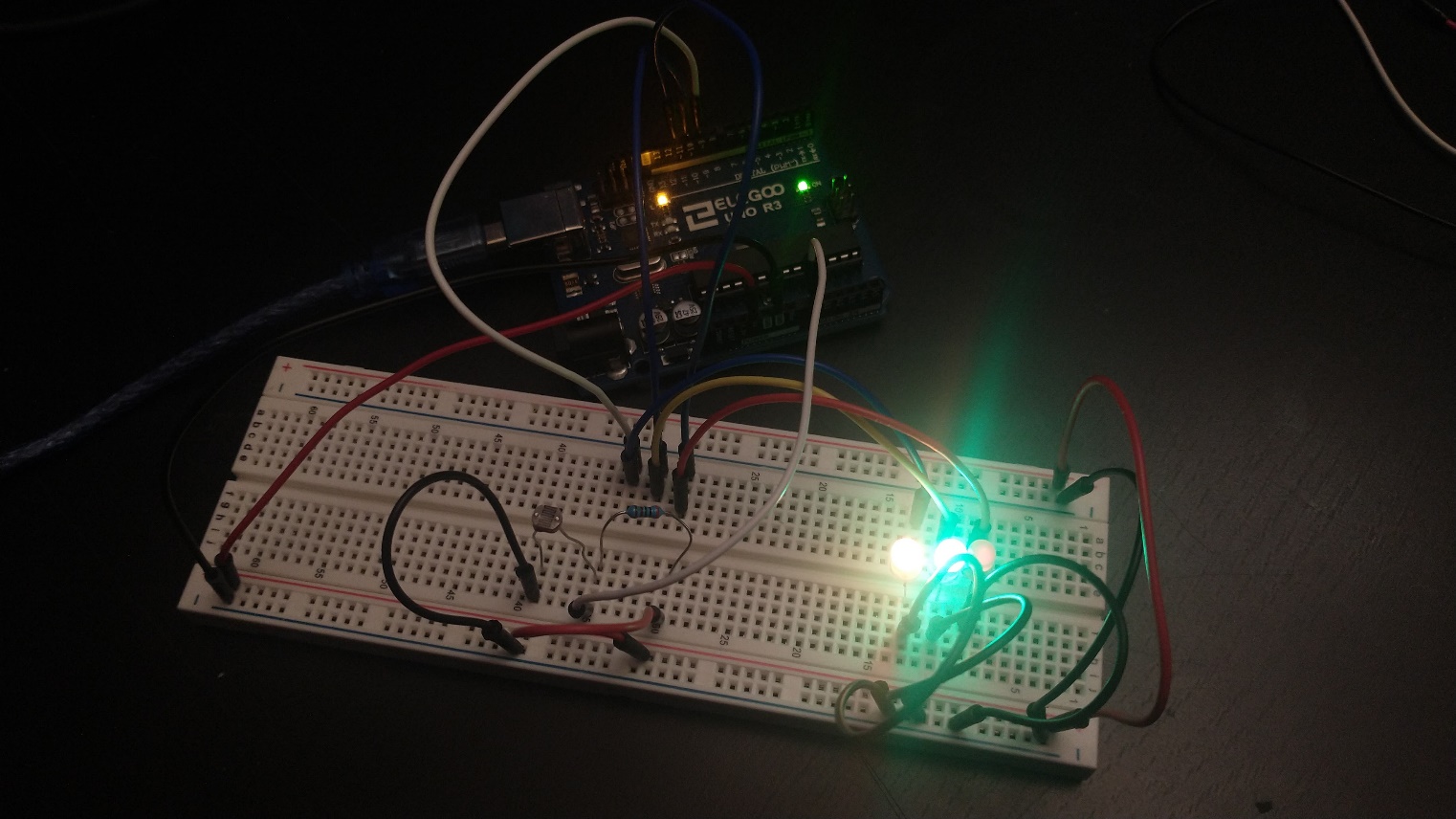
} // Don’t forget to close this ‘if statement’ with a bracket!

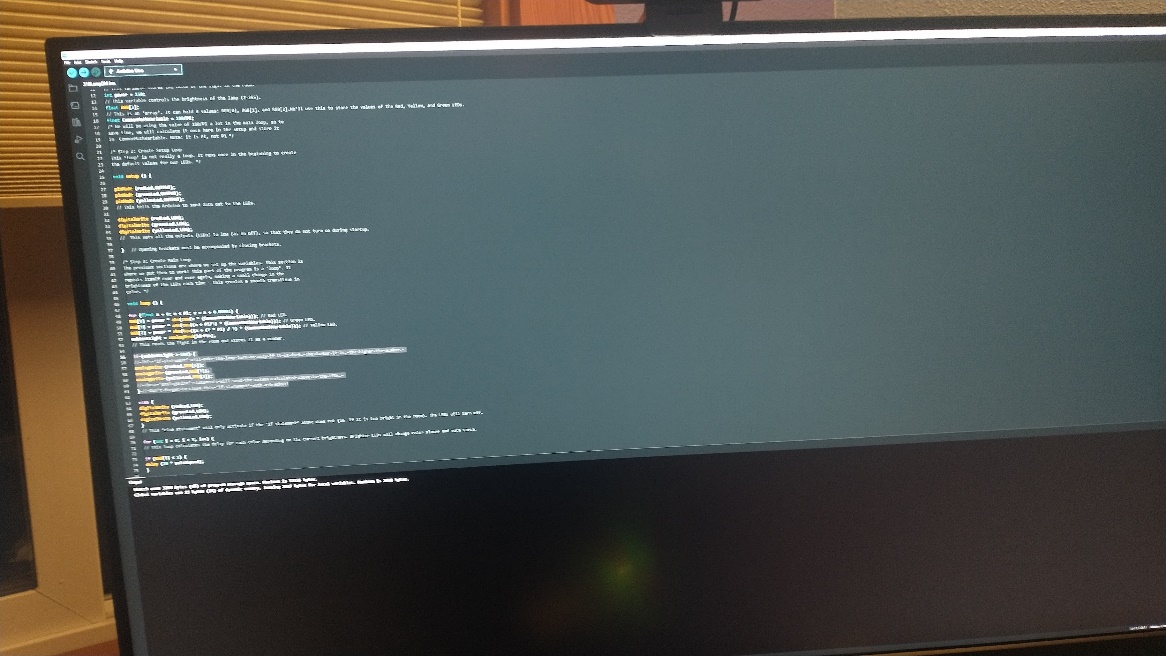
Exercise 4 – Move the 2.2k OHM resistor to another spot on the board without moving the light dependent resister. Make the project run. Important Hint: You will have to use another jumper wire to complete this step.

**Answer: Provide an explanation on how this can be accomplished. You can use pictures to demonstrate the changes.**

I did not physically do this, but I think I can explain how it is done. If I moved the resistor to the other side of the breadboard, I could maintain functionality by running a jumper from the row for each leg of the resistor to the row it is currently mounted on.

Exercise 5 – Provide pictures of the desktop and the kit assembled for this lab, or check off the lab in class with the instructor.





Deliverable:

Turn in the completed document to canvas.