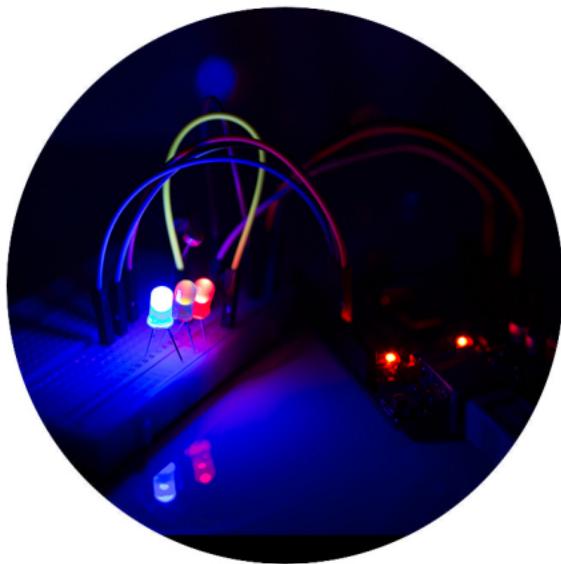


Mood Lamp



Index

Parts List	1
Introduction	2
Hardware	3-5
Program	6-11
Common Errors	12-13
Exercises	14-15
Sneak Peek	16

Parts List



Light Dependent
Resistor (LDR)
x 1



Red LED
x 1



Green LED
x 1



Blue LED
x 1



Regular
Jumper Wires
x 8 (+2 extra)



U-shaped
Jumper Wires
x 6



2.2k OHM
Resistor
x 1



Breadboard
(830 Tie-points)
x 1



UNO R3
(Arduino-compatible)
x 1



USB Cable
x 1



Chinese Paper Lantern
x 1

Introduction

Welcome to the Creation Crate club!

You've taken your first step in learning how to build electronics.

What are we Creating?

Today, we're making a mood lamp!

This lamp changes colours and only turns on when it's dark.

How do we make it?

In two steps:

1. Building the hardware.

&

2. Programming it.

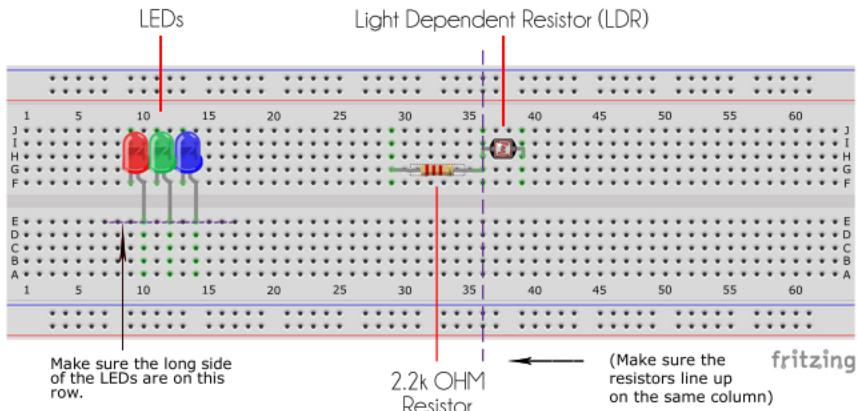
We're going to build the hardware first, and then write the program to make it run!

Turn the page to
get started!



Hardware

1. First, let's connect the components!



2. Now, let's connect the jumper wires!

Note: The wire colours are interchangeable - they all do the same thing!

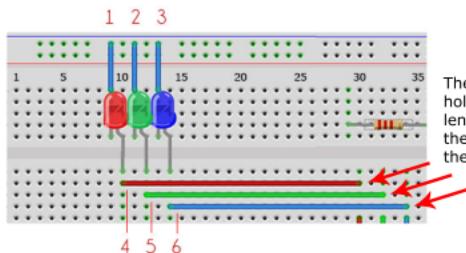


U-shaped



Regular

Attach the U-shaped wires to the 6 locations shown below.
(Don't bend them!)



These ends will end up in different holes depending on your wire lengths. This is fine! Just keep them flat and connect the rest of the parts as shown.

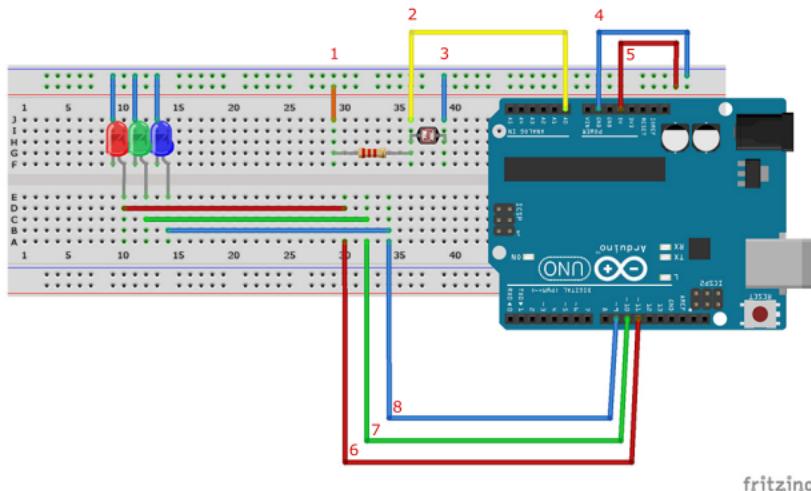
We are using U-shaped wires here because they are flat. We want to keep the left side of the breadboard as flat as possible so that the paper lantern can sit on top.

Hardware

3. Now, let's attach the rest of the (8) wires as shown below.
(You can bend these ones!)

Note:

The UNO R3 may seem like it's attached to the breadboard, but this is for illustration purposes only. You can put it anywhere as long as the wires connect!

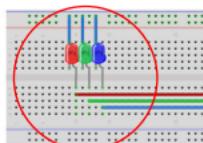


For a clearer view of this diagram, visit

<http://mycreationcrate.com/month-1>

Password: **QBXD42**

4. Time to assemble the paper lantern! Place it over the LEDs:



That's it for the hardware!

Now What?

Awesome!

You've built the hardware and it looks great!

...but it doesn't do anything yet.

In order for the mood lamp to run, you need to program it!

Electronics is a combination of hardware and programming. You can't have a working project with only one half.

How do we start programming?

1. Let's start by going to [*https://www.arduino.cc/en/Main/Software*](https://www.arduino.cc/en/Main/Software).
2. Download the Arduino Software (IDE).
3. Open the Arduino program and navigate to 'File > New'.
4. Delete all of the existing text/code.
(Note: the program is made up of lines of code)
5. Navigate to 'File > Save' and save the program as "Mood Lamp".
6. Turn the page and start programming!

Disclaimer

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program. If not, see:

<<http://www.gnu.org/licenses/>>.

Program

Note: Lines that start with “//” or “/*” indicate comments. These lines are invisible to the program. They are explanations only, and are not actual code. Actual code will be in **this colour**. You only need to enter the actual code into the program.

Let’s Begin!

(Type **the code** below into the program)

```
/*Creation Crate Mood Lamp
```

This lamp smoothly cycles through a colour spectrum.

It only turns on when its surroundings are dark.

Colour equations (we’ll be using these later):

Red = $\sin(x)$

Green = $\sin(x + \pi/3)$

Blue = $\sin(x + 2\pi/3)$

These equations are how the program will calculate the brightness of the LEDs.

Step 1: Input User Defined Variables

Think of variables like containers - they’re 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 blueLed = 9; // blue LED in Digital Pin 9.
```

// These are the pins we are using with the UNO R3 (Arduino-compatible). 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, Blue, and Green LEDs.

Program

```
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 (blueLed,OUTPUT);  
    // This tells the UNO R3 to send data out to the LEDs.  
    digitalWrite (redLed,LOW);  
    digitalWrite (greenLed,LOW);  
    digitalWrite (blueLed,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  
colour. */  
  
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)));  
        // Blue LED.
```

Program

```
ambientLight = analogRead(IdrPin);
// 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 (blueLed,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 (blueLed,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 colour depending on its
current brightness. Brighter LEDs will change colour 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);
}
```

Program

```
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!
```

Almost There!

Now that you've entered all of the code, it's time to see if it works!

Mac Users

Go to <http://mycreationcrate.com/month-1> and download the Mac OS driver before continuing. If you are not using a Mac, or have already done this in a previous project, you can skip this step.

Password: **QBXD42**

1. Connect your UNO R3 (Arduino-compatible) board to your computer with the USB cable provided.
2. Navigate to 'Tools > Port' and select the port connected to your UNO R3. (If you have trouble finding the right port, try disconnecting other USB devices).
3. Navigate to 'Tools > Board' and select "Arduino/Genuino Uno".
4. Navigate to 'Sketch > Upload'. You should see this at the bottom:

Done uploading.

(If the program doesn't upload, there are errors in the code. Don't worry! Errors are very common - I'd be more surprised if you didn't have any! See the next page for possible solutions).

5. Your project should now be working!
6. Solve the exercises on Pages 14-15 to test your understanding of the project.

Not working?
See why on the next page.



Common Errors

Note: If your project is working, you can skip this section!

Programming Errors

If a line is highlighted red like below, there is an error on that line!
Fix this error first and try again.

A screenshot of a code editor window. The code is written in C-like syntax, likely for an Arduino microcontroller. It includes calculations for RGB values based on ambient light and a sine wave, and a loop that turns on three LEDs (red, green, blue) sequentially. Two specific lines of code are highlighted in red, indicating they are causing errors:

```
RGB[1] = power * abs(sin((x+P1/3)*(CommonMathVariable)));
RGB[2] = power * abs(sin((x+(2*PI)/3)*(CommonMathVariable)));
```

Below the code, two error messages are displayed in a red box:

- 'P1' was not declared in this scope
- 'PI' was not declared in this scope

A "Copy error messages" button is visible in the top right corner of the error box.

These are the most common programming errors:

1. *Typos* - The words have to be exact! Make sure your code matches the code in this booklet. Note: Letters are case-sensitive.
2. *Missing ';' -* Any time you write a line of code, it has to end with a ';'. The only exceptions in this code are lines that start with 'void', 'if', 'else', or 'for'.
3. *Missing/extra brackets -* Any time you use an opening bracket '(' or '{', you must have a closing bracket to match it ')' or '}'. Having too many brackets will also create errors. Lastly, make sure these are placed correctly!

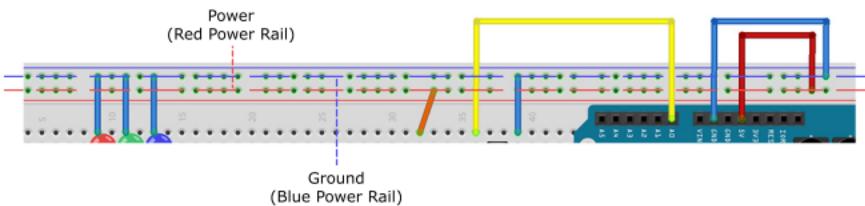
Common Errors

Hardware Errors

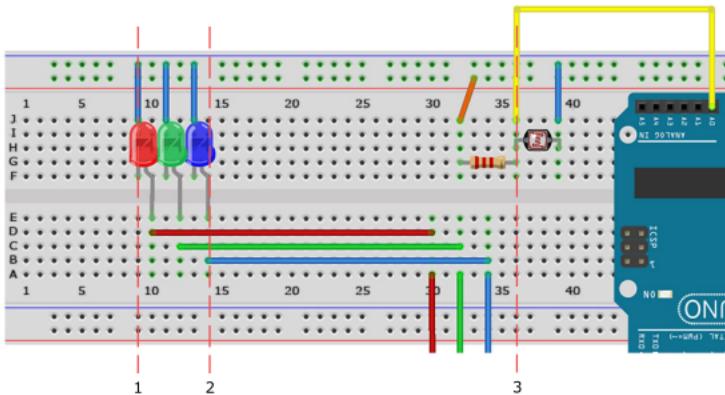
If your code uploads fine, and the project is still not working, there are error(s) with the hardware setup!

Here are some common hardware errors:

1. *Wires are not connected to the correct power rail. (Rows)*



2. *Pins/Wires are not aligned properly. (Columns)*



The 3 columns indicated are common areas for mistakes. Make sure the parts in these columns line up!

Exercises

Solve these problems and write the answers below.

- 1) Let's start by changing the overall brightness of the mood lamp.
Find the variable that controls this, and change its value.
Hint: The max value is 255.

Answer:

- 2) Make the lamp change colours faster!
Hint: There are 3 ways to do this.

Answer:

- 3) Create a new user-defined variable to control how dark it needs to be for the lamp to turn on. Call this variable 'ambientLimit'. Use this variable to replace the default number in the code.
Hint: You will have to declare this variable as an 'int'. The value can range from 0-1024.

Answer:

Exercises

Bonus Hardware Exercise

4) Move the 2.2k OHM Resistor to another spot on the board without moving the Light Dependent Resistor. Make the project run!

Hint: You will have to use one of the extra jumper wires included in the box

Solutions available at <http://mycreationcrate.com/month-1>

Password: QBXD42

Sneak Peek

Here's a sneak peek at next month's project!

Can you guess what it does?

