Lab Report: Project 4 – Color Mixing Lamp

Lucas Porter

Spring 2025 TESY 3300

Trevor P. Robinson, PhD

Lab 3 – Color Mixing Lamp

2/9/2025

Abstract

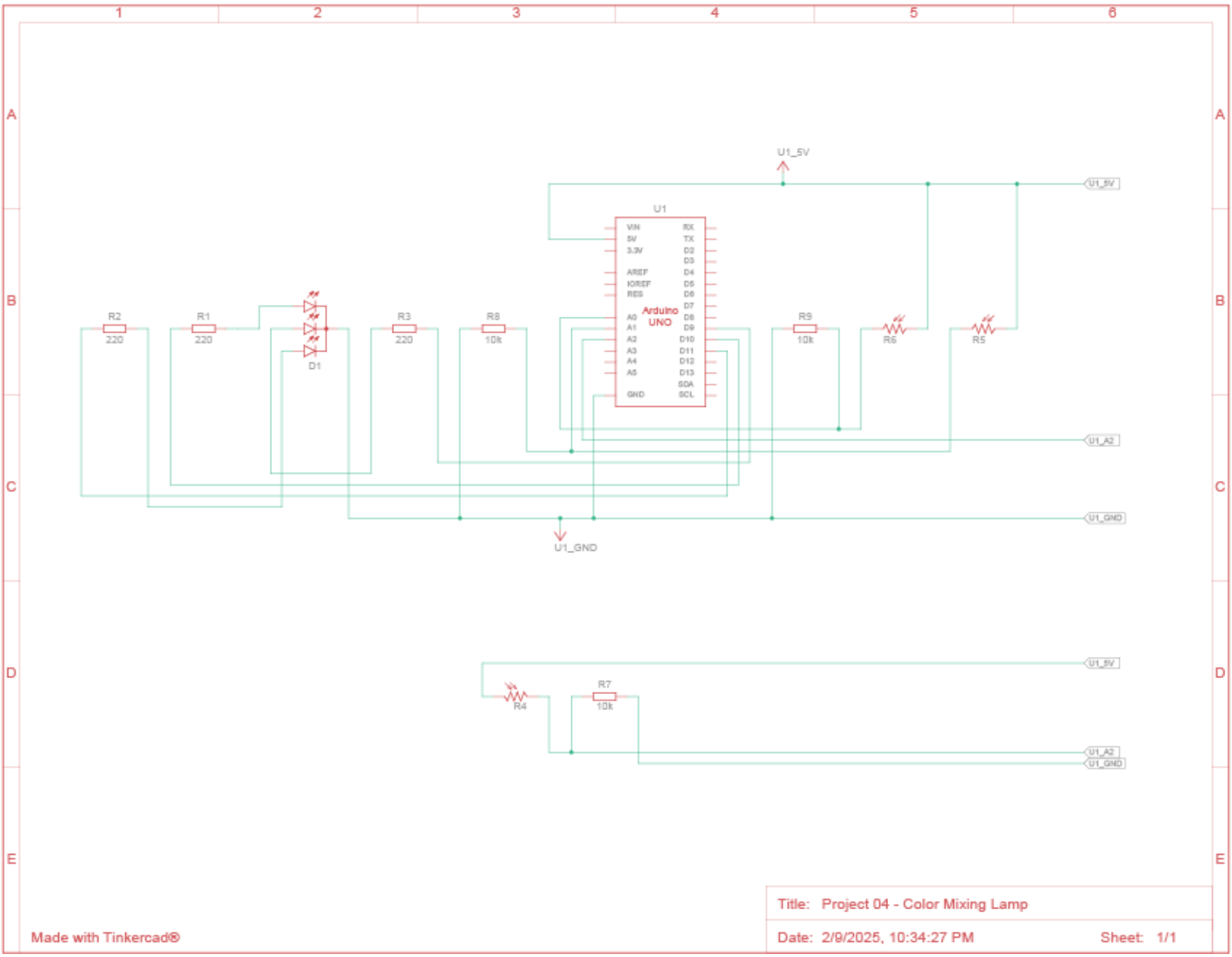
The goal of this lab was to use a photoresistor to control the brightness and color of an LED using an Arduino. The project explored how analog sensors provide variable input and how Pulse Width Modulation (PWM) can simulate an analog output using digital signals. By using multiple photoresistors, the color of an RGB LED was adjusted based on the amount of light detected by each sensor. This experiment demonstrated the relationship between light intensity and resistance, as well as how microcontrollers process analog input to control digital output.

Materials

1. Arduino Uno Board
2. Breadboard
3. RGB LED
4. Three Photoresistors
5. Three 10kΩ Resistors
6. Jumper Wires
7. USB Cable
8. Computer with Arduino IDE
9. Strips of Paper (for light isolation)

Procedure

Circuit Diagram



Steps

1. Connected the Arduino's 5V and GND pins to the power and ground rails of the breadboard.
2. Placed the RGB LED on the breadboard and connected its red, green, and blue pins to Arduino digital PWM pins.
3. Connected the common cathode of the LED to the ground rail.
4. Placed three photoresistors on the breadboard, each corresponding to a different LED color.
5. Connected one leg of each photoresistor to 5V and the other leg to both an analog input pin and a 10kΩ pull-down resistor to ground.
6. Uploaded the Arduino sketch and tested the circuit.

Code

```
const int greenLEDPin=9;
const int redLEDPin=10;
const int blueLEDPin=11;

const int redSensorPin=A0;
const int greenSensorPin=A1;
const int blueSensorPin=A2;

int redValue=0;
int greenValue=0;
int blueValue=0;

int redSensorValue=0;
int greenSensorValue=0;
int blueSensorValue=0;

void setup() {
Serial.begin(9600);

pinMode(greenLEDPin, OUTPUT);
pinMode(redLEDPin, OUTPUT);
pinMode(blueLEDPin, OUTPUT);
}

void loop() {
redSensorValue=analogRead(redSensorPin);
greenSensorValue=analogRead(greenSensorPin);
blueSensorValue=analogRead(blueSensorPin);

Serial.print("Raw Sensor Values \t red: ");
Serial.print(redSensorValue);
Serial.print("\t green: ");
Serial.print(greenSensorValue);
Serial.print("\t blue: ");
Serial.println(blueSensorValue);

redValue=redSensorValue/4;
greenValue=greenSensorValue/4;
blueValue=blueSensorValue/4;

Serial.print("Mapped Sensor Values \t red: ");
Serial.print(redValue);
Serial.print("\t green: ");
Serial.print(greenValue);
Serial.print("\t blue: ");
Serial.println(blueValue);

analogWrite(redLEDPin, redValue);
analogWrite(greenLEDPin, greenValue);
analogWrite(blueLEDPin, blueValue);

}

```

Discussion

1. **Your Arduino cannot output an analog voltage. How does it "fake it" for dimming an LED, for example? What is this principle called?**
   The Arduino simulates an analog output by rapidly turning a digital pin on and off at high frequency. This technique is called Pulse Width Modulation (PWM). The human eye perceives this rapid switching as a smooth dimming effect.

2. **What is actually happening to the LED as we see it fade in and out?**
   The LED is rapidly switching between fully on and fully off. By varying the proportion of time the LED is on versus off (duty cycle), the brightness appears to change smoothly.

3. **What does a photoresistor do? Why are they connected to analog pins in this lab?**
   A photoresistor changes its resistance based on the amount of light it receives. More light decreases resistance, while less light increases it. They are connected to analog pins because their resistance varies continuously, and an analog input can detect these variations.

4. **How many digital pins are there on the UNO board?**
   The Arduino Uno has 14 digital pins.

5. **What pins on the Arduino generate PWM signals?**
   The PWM-enabled digital pins on the Arduino Uno are 3, 5, 6, 9, 10, and 11.

6. **What is the function of the analogWrite() function?**
   The analogWrite() function sets the PWM duty cycle for a pin, effectively controlling brightness for LEDs or speed for motors.

7. **A call to analogWrite() is on a scale of 0 - 255, such that** analogWrite(255) requests a 100% duty cycle, and analogWrite(127) is a 50% duty cycle.

8. **The process of using a pulse signal to represent information is called** Pulse Width Modulation (PWM).

9. **How could you use this lab to let you know if it is a nice day outside while you are working inside?**
   A photoresistor could be placed near a window, and its readings could determine if it is bright outside. If the sensor detects a high level of light, an LED could light up or a message could be displayed on a screen.

10. **What other sensors can you use to control the LED's color?**

    Temperature sensor (adjust LED color based on temperature)

    Sound sensor (change color based on noise levels)

    Motion sensor (change color based on movement detected)

Troubleshooting

1. **Issue:** The photoresistors were not isolated well enough, causing interference between them.
   Solution: Placed small strips of paper between the sensors to better isolate the light exposure for each one. Also, turned off room lights to prevent stray light from affecting readings.

2. **Issue:** Swapped color pins in the code, causing incorrect LED colors.
   Solution: Corrected the pin assignments in the code so that each color responded to the intended photoresistor.

Conclusion

At first, the circuit functioned, but the colors were not well controlled because the photoresistors were picking up too much stray light. Once the sensors were isolated with small strips of paper, the LED colors became more distinct and controllable. The LED dimmed smoothly most of the time, but at the lowest brightness levels, it would briefly flash before turning off completely. This behavior persisted despite attempts to adjust different parameters in the code. It may have been due to how the Arduino handles low PWM values. The photoresistors worked correctly once wiring issues were fixed.

This project demonstrated how analog sensors can provide dynamic control over digital outputs. By using PWM, the Arduino was able to smoothly dim and blend LED colors based on real-time sensor input. This type of setup could be adapted for practical applications, such as ambient lighting that adjusts based on surrounding light levels. Understanding how to properly isolate sensors and troubleshoot unexpected behaviors was an important takeaway from this lab.