IE2070
Embedded Systems
2nd Year, 2nd Semester

Assignment

IR controlled Lamp Circuit

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the

Bachelor of Science Special Honors Degree in Information Technology

4th May 2024

Declaration

I certify that this report does not incorporate without acknowledgment, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief it does not contain any material previously published or written by another person, except where due reference is made in the text.

Registration Number: IT22073228

Name: D.W.A.I. Abeynayake

Introduction

The goal of this project is to create an infrared (IR) remote controller using an Arduino Uno microcontroller to control an LED bulb remotely. Four white LEDs and one RGB LED with multiple colors make up the lamp. Users can turn on and adjust the LED lamp by using an IR remote controller. The goal of this research is to give a thorough overview of the remote-controlled LED lamp system's development process.

Components

• Arduino Uno Board

• Dot Board

• IR sensor 1838

• IR Remote Controller

• White LED 4

• Multicolor (RGB) LED 1

• Soldering Iron

• Soldering Lead

• 9V battery

• 220Ω resistors

• Jumper Wires

Circuit Implementation

• Use jumper wires to connect the Arduino board to the dot board.

• Solder the IR sensor onto the dot board, making that the signal pin of the receiver is linked to pin 2, the digital pin on the Arduino board.

• Solder a resistor (220Ω) in series with the anode of each white LED and connect each LED's cathode to a different digital pin on the Arduino board (pins 5, 6, 9, and 10).

• Attach the RGB LED's common cathode to the dot board's ground, and link pins 13, 12, and 11 on the Arduino board to individual color pins (R, G, B) via resistors (220Ω).

• Use a 9-volt battery to supply the circuit with power.

Code Implementation

```
#include <Arduino.h>
#include <IRremote.h>
//function prototypes
void volumeUp();
void volumeDown();
void reduceBrightness();
void increaseBrightness();
void turnOn();
void turnOff();
void cycleColors();
void resetForColorToggle();
void resetToWhite();
void updateLEDs();
void setAllLEDs(bool state);
void setWhiteLEDs(int count);
void setRGBLED(int ledPin);
// Pin definitions for LED connections
#define RED_LED_PIN 13
#define GREEN_LED_PIN 12
#define BLUE_LED_PIN 11
#define WHITE_LED_1 5
#define WHITE_LED_2 6
#define WHITE_LED_3 9
#define WHITE_LED_4 10
// Pin definitions for IR receiver
#define IR_RECEIVE_PIN 2
// IR Remote button codes
#define VOLUME_UP_CODE 21 // Button code for Volume Up
#define VOLUME_DOWN_CODE 7 // Button code for Volume Down
#define CHANNEL_DOWN_CODE 69 // Button code for Channel Down
#define CHANNEL_UP_CODE 71 // Button code for Channel Down
// State variables
int ledState = 0;
bool toggleColors = false;
int colorCounter = 1;
int led_brightness[4] = {255, 255, 255, 255};
int loopback = 0;
void setup() {
 Serial.begin(9600);
 IrReceiver.begin(IR_RECEIVE_PIN);
 // Initialize white LEDs as outputs
 pinMode(WHITE_LED_1, OUTPUT);
 pinMode(WHITE_LED_2, OUTPUT);
 pinMode(WHITE_LED_3, OUTPUT);
 pinMode(WHITE_LED_4, OUTPUT);
 // Initialize RGB LEDs as outputs
```

```
 pinMode(RED_LED_PIN, OUTPUT);
 pinMode(GREEN_LED_PIN, OUTPUT);
 pinMode(BLUE_LED_PIN, OUTPUT);

}
void loop() {
 if (IrReceiver.decode()) {
 IrReceiver.resume();
 int irCode = IrReceiver.decodedIRData.command;
 Serial.println(irCode);
 if (irCode == VOLUME_UP_CODE) {
 volumeUp();
 } else if (irCode == VOLUME_DOWN_CODE) {
 volumeDown();
 } else if (irCode == CHANNEL_DOWN_CODE) {
 reduceBrightness();
 } else if (irCode == CHANNEL_UP_CODE) {
 increaseBrightness();
 }
 }
}
void volumeUp() {
 if (toggleColors) {
 cycleColors();
 } else {
 turnOn();
 }
 updateLEDs();
 delay(200);
}
void volumeDown() {
 if (ledState > 4 && !toggleColors) {
 resetForColorToggle();
 } else if (toggleColors) {
 resetToWhite();
 } else {
 turnOff();
 }
 updateLEDs();
 delay(200);
}
void reduceBrightness() {
 int i = ledState - 1;
 for(i = 0; i < 4; i++){
 led_brightness[i] -= 10;
 if (led_brightness[i] < 0) {
 led_brightness[i] = 0;
 }
```

```
 updateLEDs();
 }
}
void increaseBrightness() {
 int i = ledState - 1;
 for(i = 0; i < 4; i++){
 led_brightness[i] += 10; // Increase brightness by 10 units
 if (led_brightness[i] > 255) { // Assuming maximum brightness is 255
 led_brightness[i] = 255;
 }

 updateLEDs();
 }
}
void turnOn() {
 ledState++;
 if (ledState > 7) {
 loopback++;
 ledState = 7;
 toggleColors = true;
 colorCounter = 1;
 }
}
void turnOff() {
 ledState--;
 led_brightness[ledState] = 255;
 if (ledState < 0) {
 ledState = 0;
 }
}
void cycleColors() {
 colorCounter++;
 if (colorCounter > 3) {
 colorCounter = 1;
 loopback++;
 }
}
void resetForColorToggle() {
 ledState--;
 if (ledState <= 0) {
 ledState = 0;
 }
}
void resetToWhite() {
 colorCounter--;
 if (colorCounter < 1 && loopback > 0) {
 colorCounter = 3;
 loopback--;
```

```
 if (loopback <= 0) {
 toggleColors = false;
 loopback = 0;
 }
 }
}
void updateLEDs() {
 if (!toggleColors) {
 // Handle white LEDs and RGB LEDs for non-color toggle mode
 switch (ledState) {
 case 0: setAllLEDs(LOW); break; // All off
 case 1: setWhiteLEDs(1); break;
 case 2: setWhiteLEDs(2); break;
 case 3: setWhiteLEDs(3); break;
 case 4: setWhiteLEDs(4); break;
 case 5: setRGBLED(RED_LED_PIN); break; // Red LED on
 case 6: setRGBLED(GREEN_LED_PIN); break; // Green LED on
 case 7: setRGBLED(BLUE_LED_PIN); break; // Blue LED on
 }
 } else {

 // Handle RGB LED for color toggle mode
 switch (colorCounter) {
 case 0: setWhiteLEDs(4); break; // All off
 case 1: setRGBLED(RED_LED_PIN); break; // Red LED on
 case 2: setRGBLED(GREEN_LED_PIN); break; // Green LED on
 case 3: setRGBLED(BLUE_LED_PIN); break; // Blue LED on
 }
 }
}
void setAllLEDs(bool state) {
 digitalWrite(WHITE_LED_1, state);
 digitalWrite(WHITE_LED_2, state);
 digitalWrite(WHITE_LED_3, state);
 digitalWrite(WHITE_LED_4, state);
 digitalWrite(RED_LED_PIN, state);
 digitalWrite(GREEN_LED_PIN, state);
 digitalWrite(BLUE_LED_PIN, state);
}
void setWhiteLEDs(int count) {
 digitalWrite(RED_LED_PIN, LOW);
 digitalWrite(GREEN_LED_PIN, LOW);
 digitalWrite(BLUE_LED_PIN, LOW);
 if (count >= 1) {
 analogWrite(WHITE_LED_1, led_brightness[0]);
 } else {
 digitalWrite(WHITE_LED_1, LOW);
 }
```

```
 if (count >= 2) {
 analogWrite(WHITE_LED_2, led_brightness[1]);
 } else {
 digitalWrite(WHITE_LED_2, LOW);
 }
 if (count >= 3) {
 analogWrite(WHITE_LED_3, led_brightness[2]);
 } else {
 digitalWrite(WHITE_LED_3, LOW);
 }
 if (count >= 4) {
 analogWrite(WHITE_LED_4, led_brightness[3]);
 } else {
 digitalWrite(WHITE_LED_4, LOW);
 }
}
void setRGBLED(int ledPin) {
 digitalWrite(RED_LED_PIN, LOW);
 digitalWrite(GREEN_LED_PIN, LOW);
 digitalWrite(BLUE_LED_PIN, LOW);
 digitalWrite(ledPin, HIGH);
}
```

Circuit Diagram

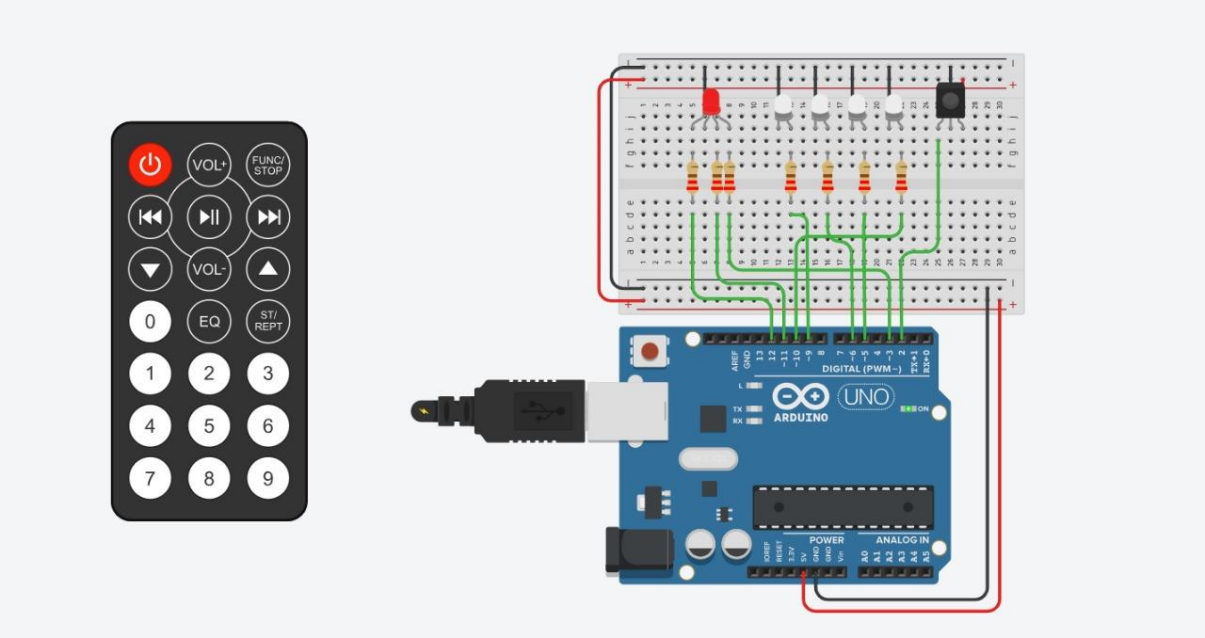

Figure 1: TinkerCad Diagram

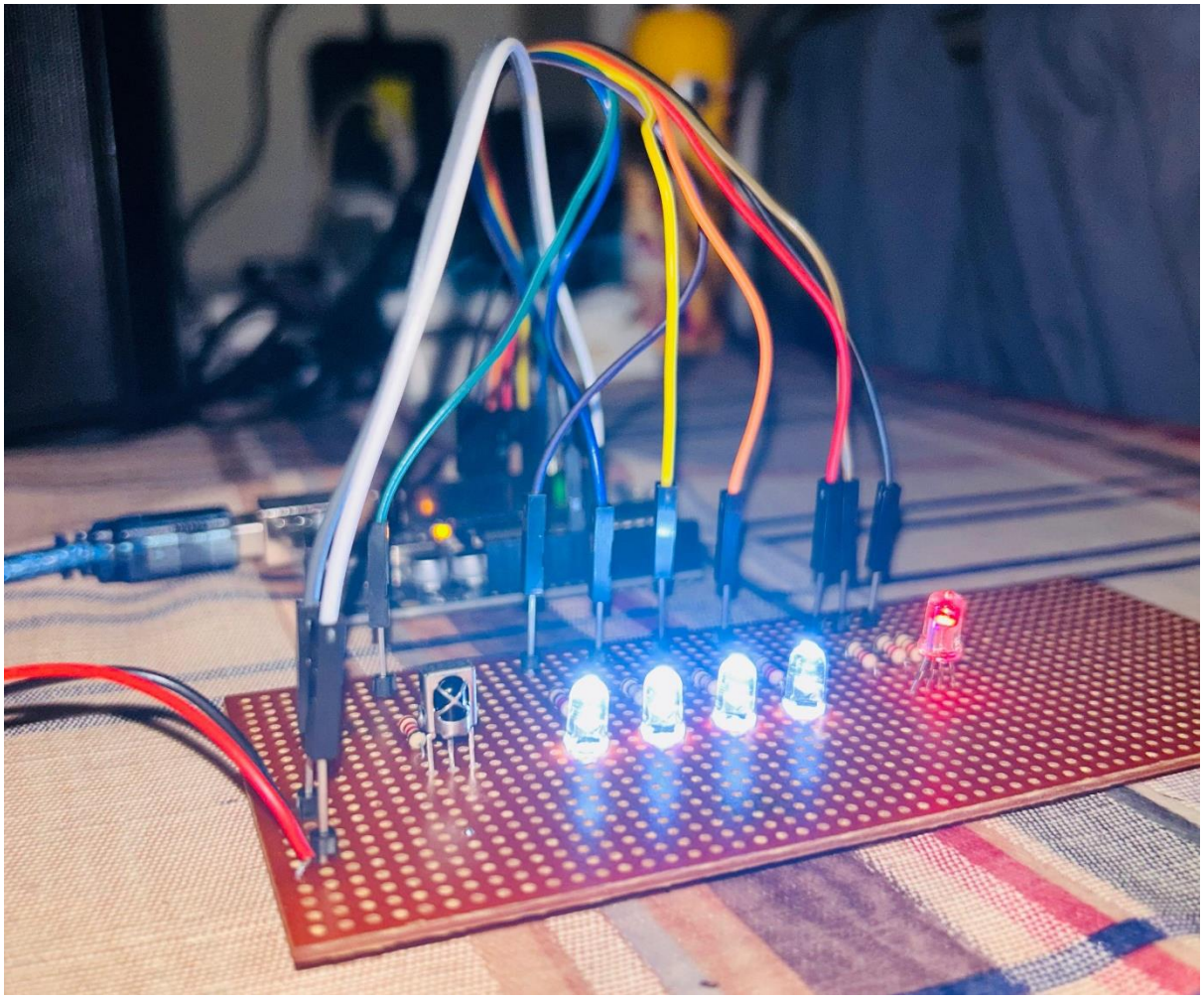

Figure 2: Dot Board Diagram

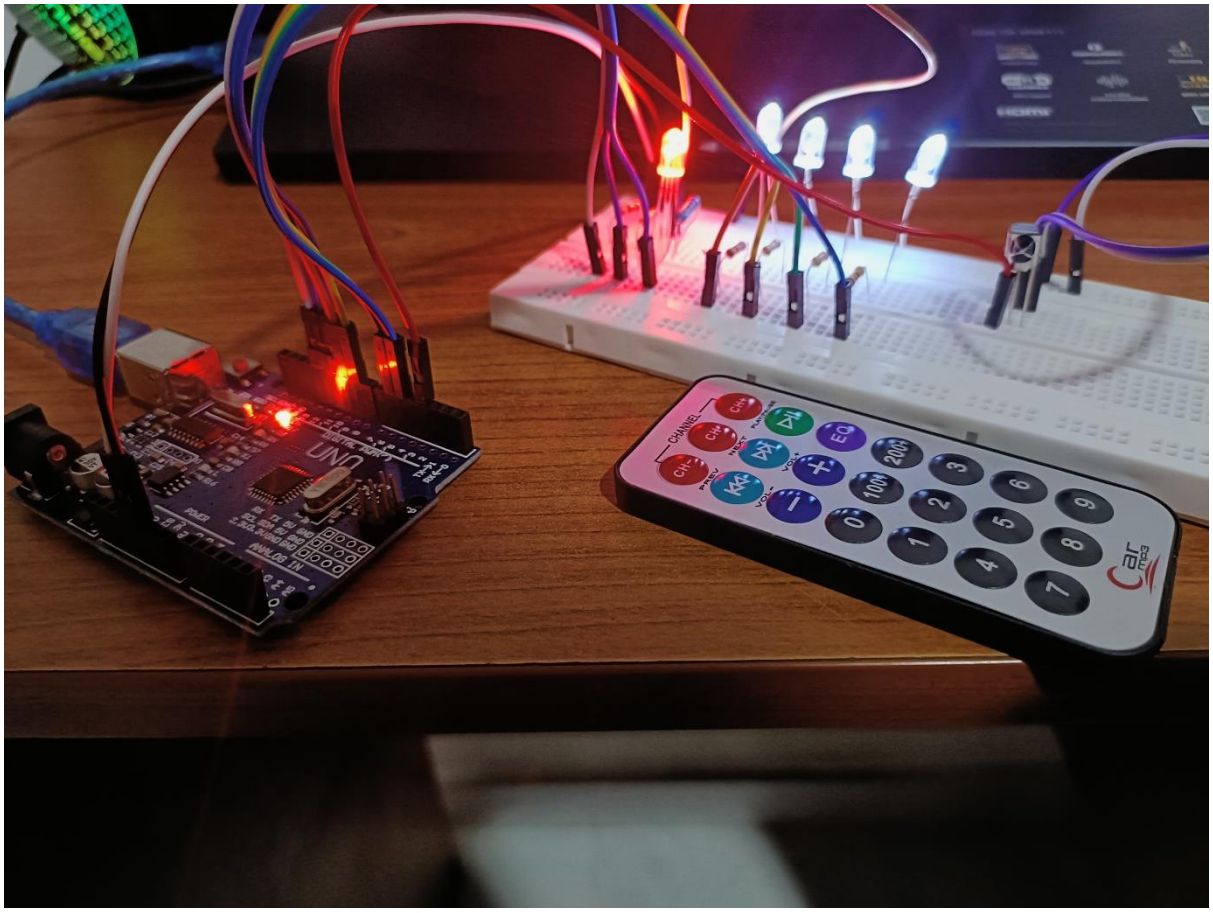

Figure 3: Breadboard Diagram

Testing and Validation

Prior to being implemented in a dot board, the circuit was tested in a breadboard to guarantee its dependability and performance.

Using the Arduino IDE serial monitor, the codes for the buttons on the IR remote controls were obtained.

The circuit was first designed and tested using Tinkercad. After making sure that Tinkercad's circuit implementation worked correctly, the circuit was built on a breadboard.

By pressing the volume up and down buttons, sequential LED activation and deactivation was tested, and the LEDs turned on in the correct order every time.

When the channel up and channel down buttons were engaged to adjust brightness, certain inaccuracies appeared in the result. The errors were fixed by taking the necessary actions. The final results of testing outcomes confirmed the successful operation of the remote-controlled LED lamp.

Discussion

With a few small adjustments made to maximize simplicity and performance, the functionality of this project satisfies the objectives listed in the project specifications.

Pressing the volume up button activates each LED individually. The multicolor LED changes colors constantly when the volume up button is pressed repeatedly. On the other hand, when the volume down button is hit, the LEDs go out. The channel up and channel down controls regulate the LEDs' brightness.

The lamp's performance was assessed during the development phase, all the while guaranteeing its dependability and functionality. Nevertheless, there were certain difficulties with the creation of the circuit schematic and the implementation of the code.

Instead of decreasing brightness when the channel down button was pressed, the first white LED, which was attached to the Arduino board's third pin, stopped responding when the brightness of the white LEDs was reduced. As a result, the LED's functionality was restored and the Arduino pin that was attached to it was changed.

Better brightness control, higher-quality components, and better design achieved by rearranging the components can all help to assure that this project will continue to improve in the future.

References

- H. L. ModTech, “Control LED’s with an IR remote in tinkercad circuits! #arduino,” 02-Oct2020. [Online]. Available: https://www.youtube.com/watch?v=qHPZSCOWPD8.