Dataintelliage

SHREYA, TANNU, TANNU2, AKANSHA

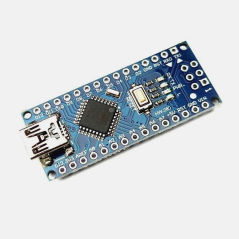
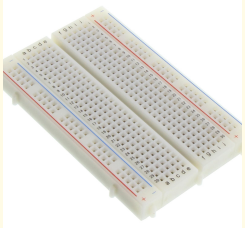
LIGHT INTENSITY CONTROLLER

Project Overview

An intensity controller for light using hand gestures involves the integration of sensors, image processing, and communication modules to allow users to adjust the brightness of lights through specific hand movements.

- The light source, equipped with a controllable intensity feature, adjusts its brightness based on the received commands.
- This could involve dimming, brightening, or toggling the light on/off.

Components

1	2	3	4	5
				

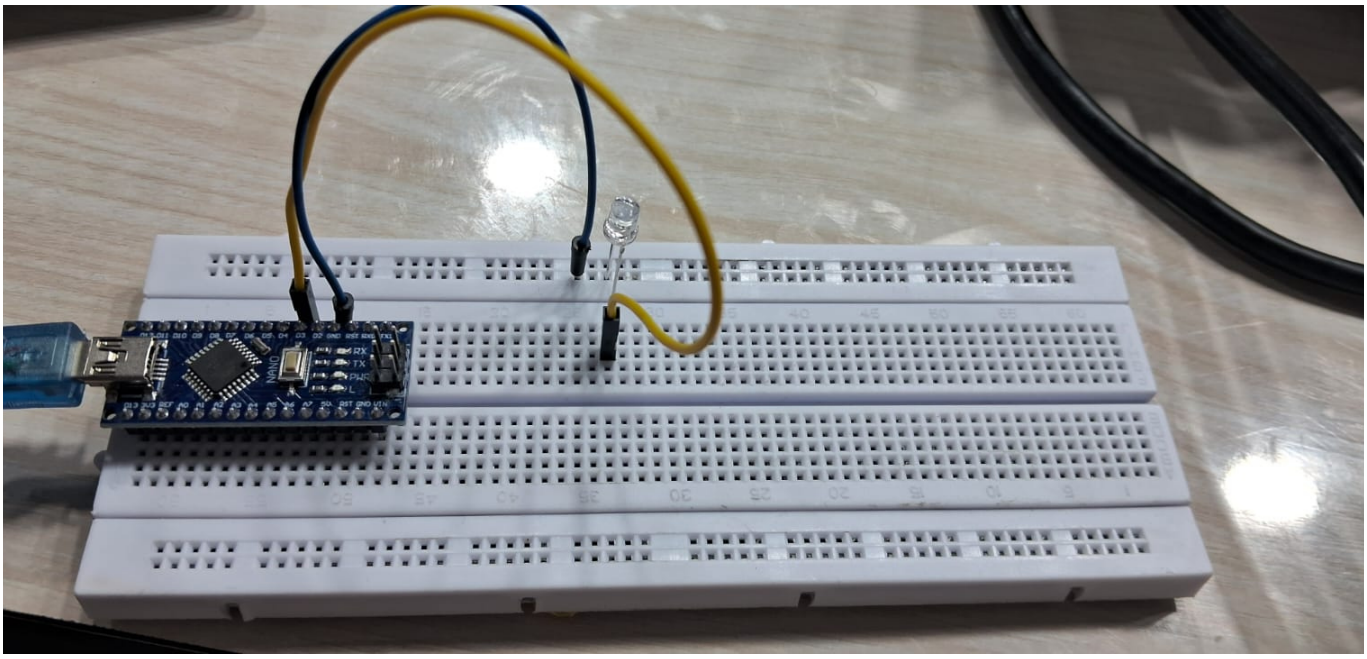
1. **Arduino Nano V3**
2. **LED Light**
3. **Breadboard**
4. **Jumper Wire**
5. **USB Cable (Type B)**

Connections

- Arduino Nano V3 - Breadboard

- **Arduino Nano V3 - LED Light**

+ve	**-**	**D3**
-ve	**-**	**GND**



Usage

The usage of an intensity controller for light using hand gestures brings about several practical applications and advantages in various settings. Here are some common scenarios where such a system can be beneficial:

Smart Homes:

- In smart home environments, users can control the brightness of lights in different rooms through hand gestures. For example, a simple upward or downward swipe can adjust the intensity of ambient lighting in the living room, bedroom, or kitchen.

Energy Conservation:

- Users can easily dim lights when full brightness is not necessary, contributing to energy efficiency. This is particularly useful in areas with natural light, where the system can adapt the artificial lighting to complement the available daylight.

Accessibility:

- Hand gesture-controlled light intensity provides an accessible and intuitive way for individuals with mobility challenges to manage their lighting environment.

This can be especially beneficial for those who may find it difficult to reach physical switches.

Entertainment and Mood Lighting:

- In entertainment settings or during events, users can create dynamic lighting effects by adjusting the intensity of lights with specific hand gestures. This adds an interactive and immersive element to the overall experience.

Workplace Settings:

- In offices or workspaces, users can control the brightness of task lighting or overhead lights to suit their preferences. This can contribute to a more comfortable and personalized work environment.

Public Spaces:

- Implementing hand gesture-controlled lighting in public spaces, such as museums, galleries, or exhibition halls, can enhance the visitor experience. Users can control the lighting to focus on specific exhibits or create dynamic visual effects.

Healthcare Environments:

- In healthcare settings, such as hospitals or nursing homes, hand gesture-controlled lighting can be used to create a soothing and comfortable atmosphere for patients. Caregivers can easily adjust lighting conditions without physical contact.

Interactive Displays:

- Hand gesture-controlled lighting can be integrated into interactive displays or installations. This allows users to engage with the environment and control the lighting as part of an interactive experience.

Code

- **Arduino Code**

```
const int redPin = 3;

void setup()

{
```

```
Serial.begin(9600);
pinMode(redPin, OUTPUT);
}
void loop()
{
while (Serial.available() > 0){
int red = Serial.parseInt();
if (Serial.read() == '\n')
 {
  red = constrain(red, 0, 255);
  analogWrite(redPin, red);
  Serial.print(red);}
 }
}
```

Open CV Code

```
import serial
import cv2
from cvzone.HandTrackingModule import HandDetector
import math

serialcomm = serial.Serial('COM18', 9600)
serialcomm.timeout = 1

cap = cv2.VideoCapture(0)
detector = HandDetector(detectionCon=0.5, maxHands=1)
```

```
while True:
    success, img = cap.read()

    # Check if the frame is not empty
    if not success or img is None:
        continue

    img = cv2.resize(img, (500, 500))
    hands, _ = detector.findHands(img)

    if hands:
        hand = hands[0]
        fingers = detector.fingersUp(hand)

        if len(hand["lmList"][4]) >= 2:
            x1, y1, _ = hand["lmList"][4][:3]  # Coordinates of the tip of the index finger
        else:
            continue  # Skip this iteration if the landmark structure is unexpected

        if len(hand["lmList"][8]) >= 2:
            x2, y2, _ = hand["lmList"][8][:3]  # Coordinates of the tip of the middle finger
        else:
            continue  # Skip this iteration if the landmark structure is unexpected
```

```
        cv2.circle(img, (int(x1), int(y1)), 7, (0, 255, 255), 1)
        cv2.circle(img, (int(x2), int(y2)), 7, (0, 255, 255), 1)
        cv2.line(img, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)

        distance = int(math.sqrt((x2 - x1)**2 + (y2 - y1)**2) * 1.0)
        distance = int((distance / 110) * 255)
        newline = '\n'

        if 0 < distance < 256:
            cv2.putText(img, str(distance), (20, 30), cv2.FONT_HERSHEY_COMPLEX, 0.7, (255,
255, 255), 1)
            serialcomm.write(str(distance).encode())
            serialcomm.write(newline.encode())

    cv2.imshow('Image', img)

    if cv2.waitKey(20) & 0xFF == 27:
        break

cv2.destroyAllWindows().
```