Jaypee Institute Of Information Technology
Department Of Computer Science & Engineering And Information Technology

विद्या तत्व ज्योतिसमः

ClearLens: Investigating Chromatic Influence on Image Clarity Using NodeMCU

COURSE NAME : COMPUTER ORGANISATION AND ARCHITECTURE LAB
COURSE CODE : 15B17CI373
PROGRAM : B.Tech. CSE, 3rd Year 5th Semester

SUBMITTED BY:

Name	Enrollment Number	Batch
Sneha Srijaya	23103310	B11
Sneha Gusain	23103327	B11
Kritika Bhatt	23103329	B11
Arya Dhawan	23103330	B11

UNDER THE SUPERVISION OF:
Dr. Naveen Chauhan
Dr. Pawan Kumar Upadhyay

ClearLens: Investigating Chromatic Influence on Image Clarity Using NodeMCU

Problem Statement

In imaging systems, the clarity and quality of captured images depend heavily on illumination conditions. While many studies focus on light intensity or angle, the impact of light color (chromaticity) on image clarity is less explored, particularly using affordable setups like a laptop webcam interfaced with NodeMCU. This project aims to analyze how different colored lights positioned in front of a laptop webcam affect the clarity of images captured, identifying which colors optimize image sharpness and contrast.

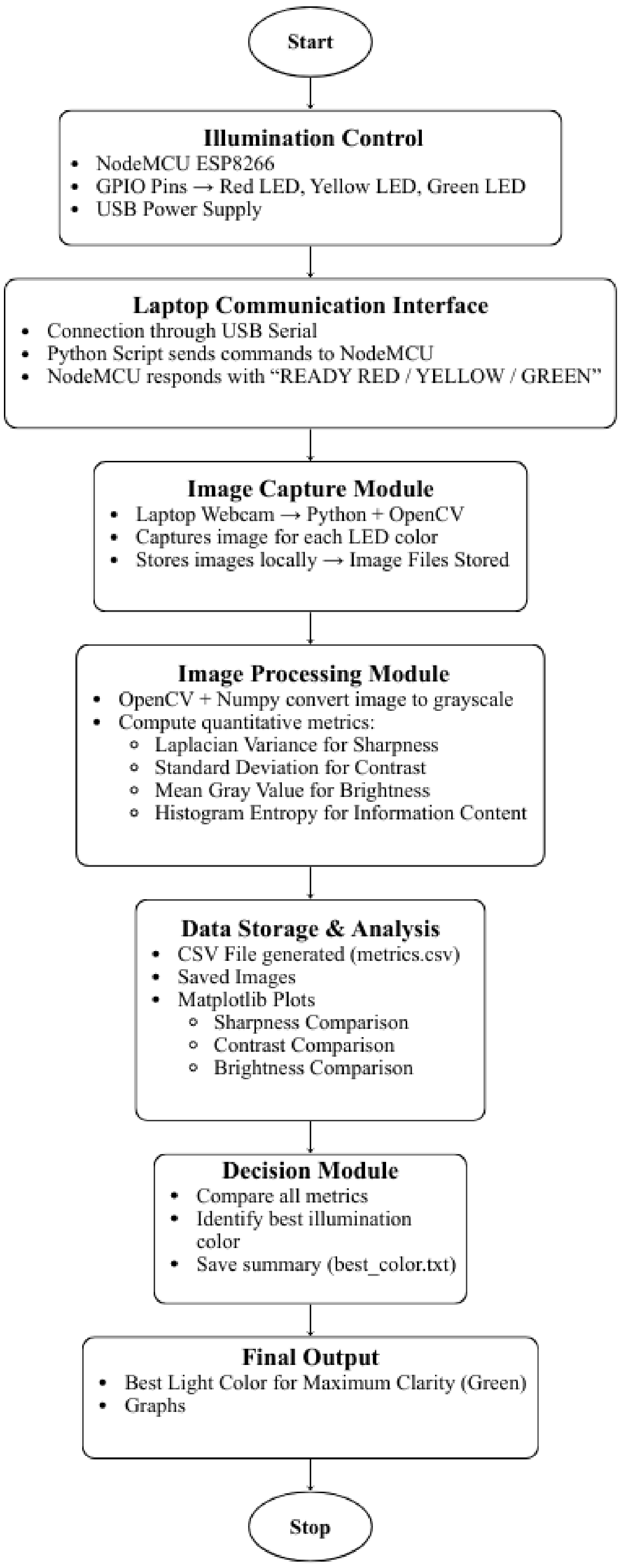
Abstract

ClearLens explores the influence of varying light colors on image clarity using a laptop webcam controlled by NodeMCU. By illuminating a target object sequentially with different colors of light and capturing corresponding images, the project employs OpenCV-based image processing techniques to quantify clarity metrics such as sharpness, contrast, and noise. The goal is to determine which chromatic lighting conditions produce the highest quality images. This low-cost, scalable approach combines embedded hardware control with advanced image analysis, offering insights applicable in optical imaging, machine vision, and quality control domains.

Project Objectives

1. **Investigate the impact of light color on image clarity**
   Analyze how different chromatic lighting conditions (Red, Green, Yellow) affect sharpness, contrast, and noise in webcam-captured images.
2. **Develop a low-cost, NodeMCU–based imaging platform**
   Integrate a laptop webcam and controllable RGB LED lighting system using NodeMCU for illumination control.
3. **Automate image acquisition under varied lighting conditions**
   Implement synchronized LED color switching and image capture via a Python script to ensure consistency and repeatability.
4. **Implement clarity evaluation metrics using OpenCV**
   Use quantitative measures such as Laplacian variance (sharpness), histogram spread (contrast), and noise estimation to assess image quality.
5. **Create a data storage and visualization framework**
   Store captured images and computed metrics locally, and visualize results using tools like Matplotlib.
6. **Identify optimal illumination colors for maximum clarity**
   Determine which light colors enhance image sharpness and contrast when using an affordable webcam setup.

Flowchart


Start
Illumination Control
NodeMCU ESP8266
GPIO Pins → Red LED, Yellow LED, Green LED
USB Power Supply
Laptop Communication Interface
Connection through USB Serial
Python Script sends commands to NodeMCU
NodeMCU responds with "READY RED / YELLOW / GREEN"
Image Capture Module
Laptop Webcam → Python + OpenCV
Captures image for each LED color
Stores images locally → Image Files Stored
Image Processing Module
OpenCV + Numpy convert image to grayscale
Compute quantitative metrics:
Laplacian Variance for Sharpness
Standard Deviation for Contrast
Mean Gray Value for Brightness
Histogram Entropy for Information Content
Data Storage & Analysis
CSV File generated (metrics.csv)
Saved Images
Matplotlib Plots
Sharpness Comparison
Contrast Comparison
Brightness Comparison
Decision Module
Compare all metrics
Identify best illumination color
Save summary (best_color.txt)
Final Output
Best Light Color for Maximum Clarity (Green)
Graphs
Stop

Project Workflow

1. **Setup**
   NodeMCU connected to a USB laptop webcam and colored LEDs (Red, Green, Yellow) positioned near the object for illumination control.

2. **Light Control**
   LED color switching is controlled using NodeMCU (ESP8266), programmed via Arduino IDE, with resistors for handling higher LED currents.
3. **Image Capture**
   For each selected LED color, the webcam captures an image of the target object, synchronized with illumination changes, and stores it.
4. **Image Processing**
   Captured images are processed using OpenCV to measure clarity, where sharpness is calculated with Laplacian variance, contrast with histogram spread, and noise through estimation techniques.
5. **Data & Visualization**
   All captured images and computed clarity metrics are locally saved (CSV). Visualization is done using Matplotlib and plots are saved.
6. **Analysis & Outcome**
   Results are compared across different illumination colors to determine which chromatic condition provides the highest image clarity (maximum sharpness, contrast, and minimal noise).

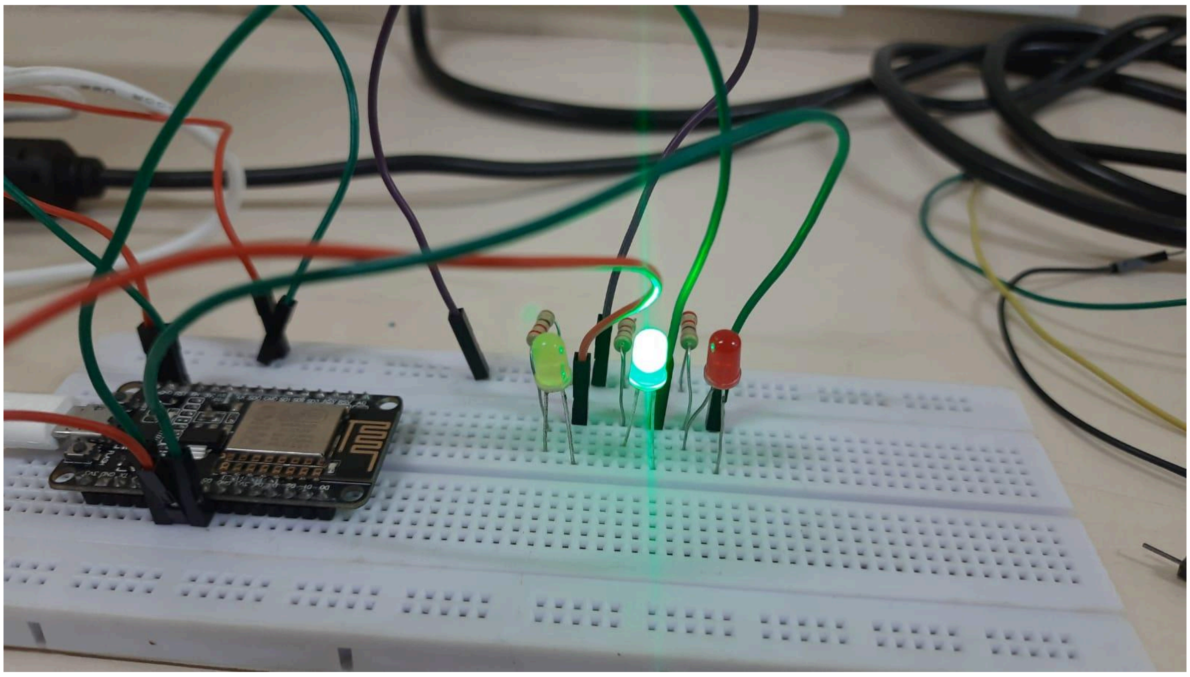
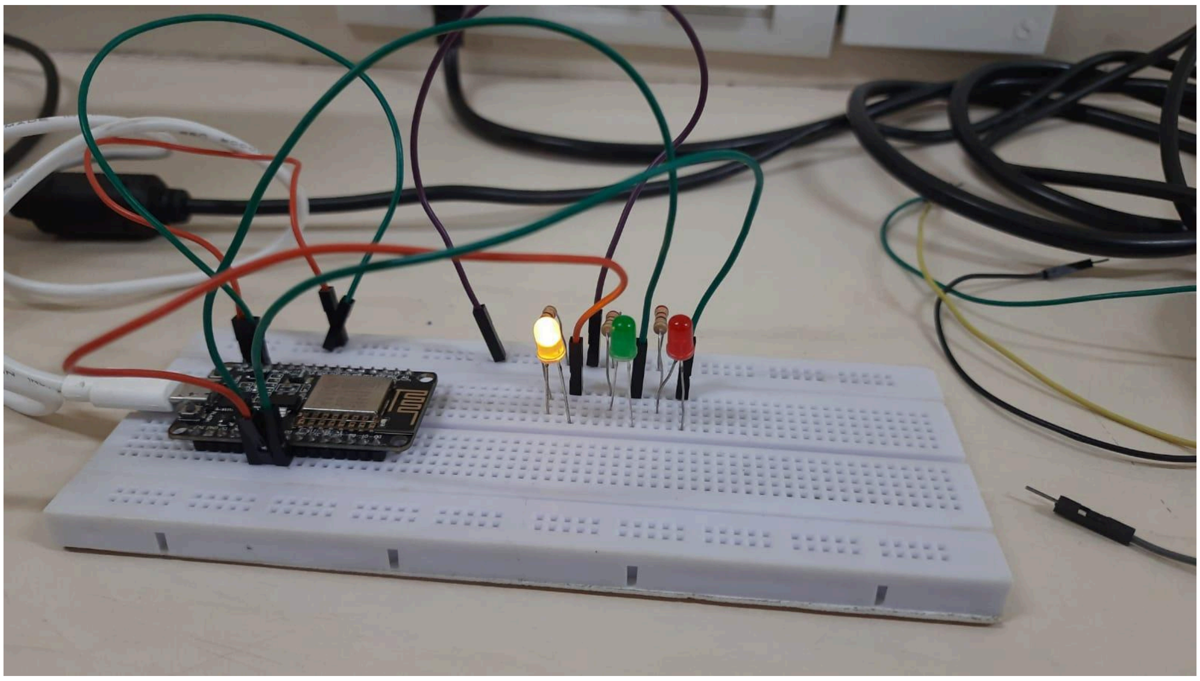
Requirements

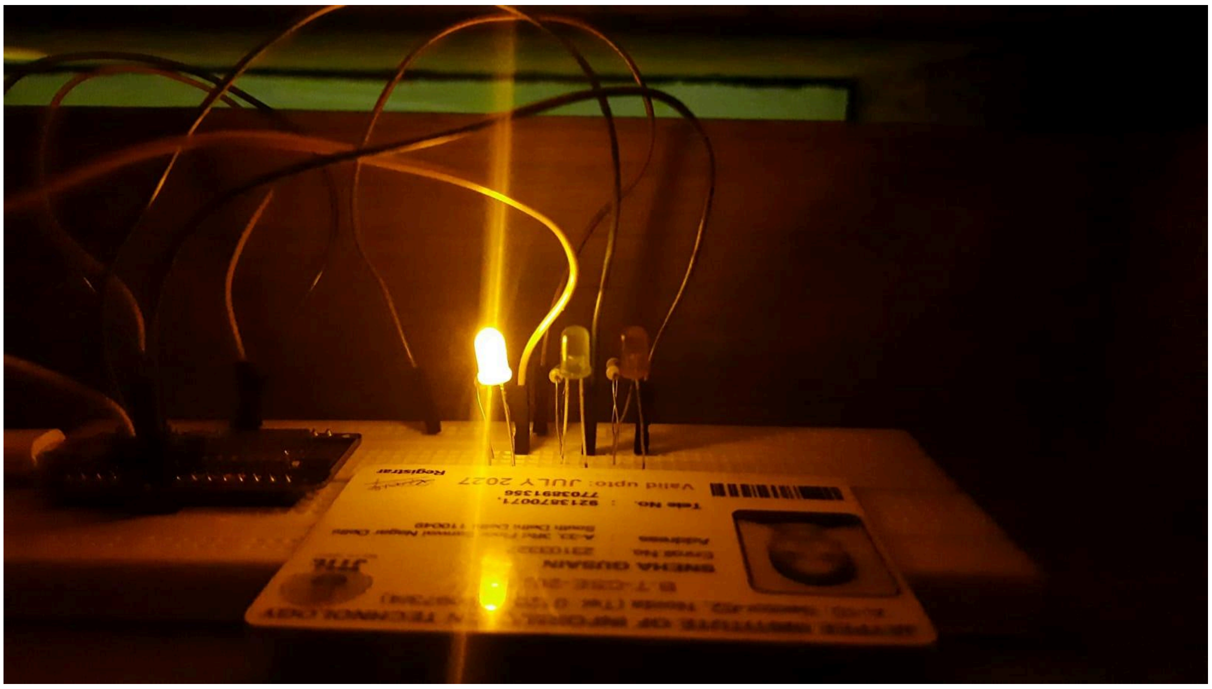
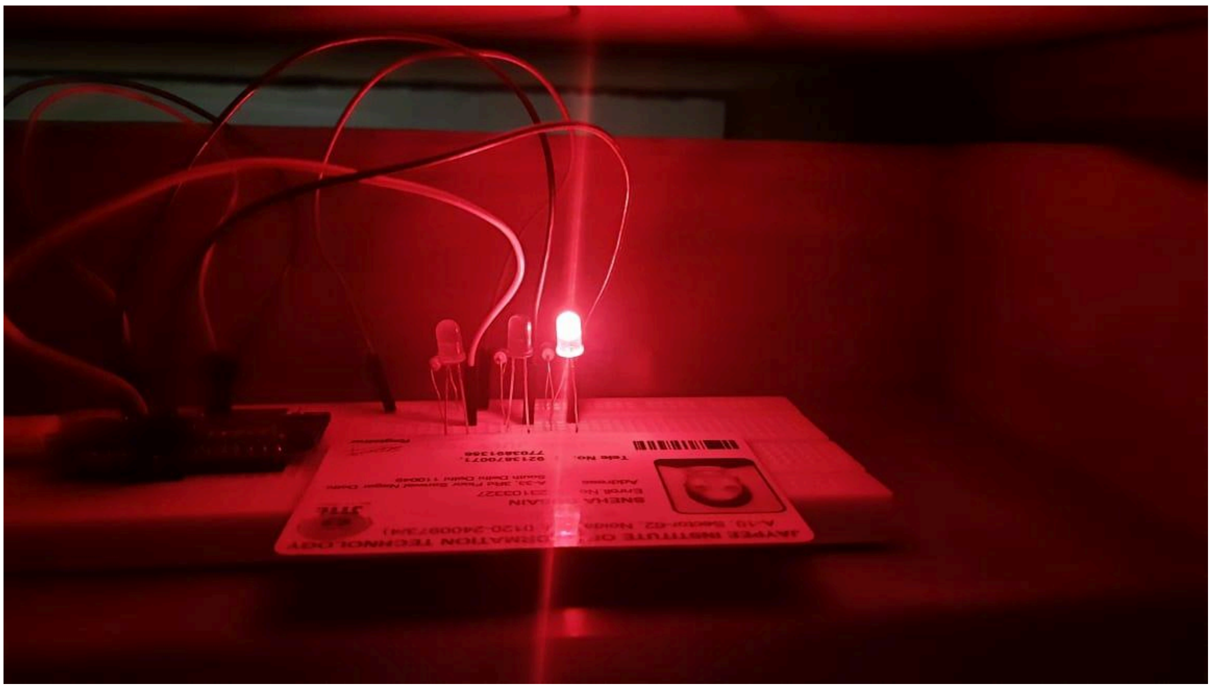
- **Hardware:**
  - NodeMCU ESP8266
  - Laptop webcam connected via USB
  - Individual colored LEDs (Red, Yellow, Green)
  - Resistors
  - Jumper Wires
  - Breadboard

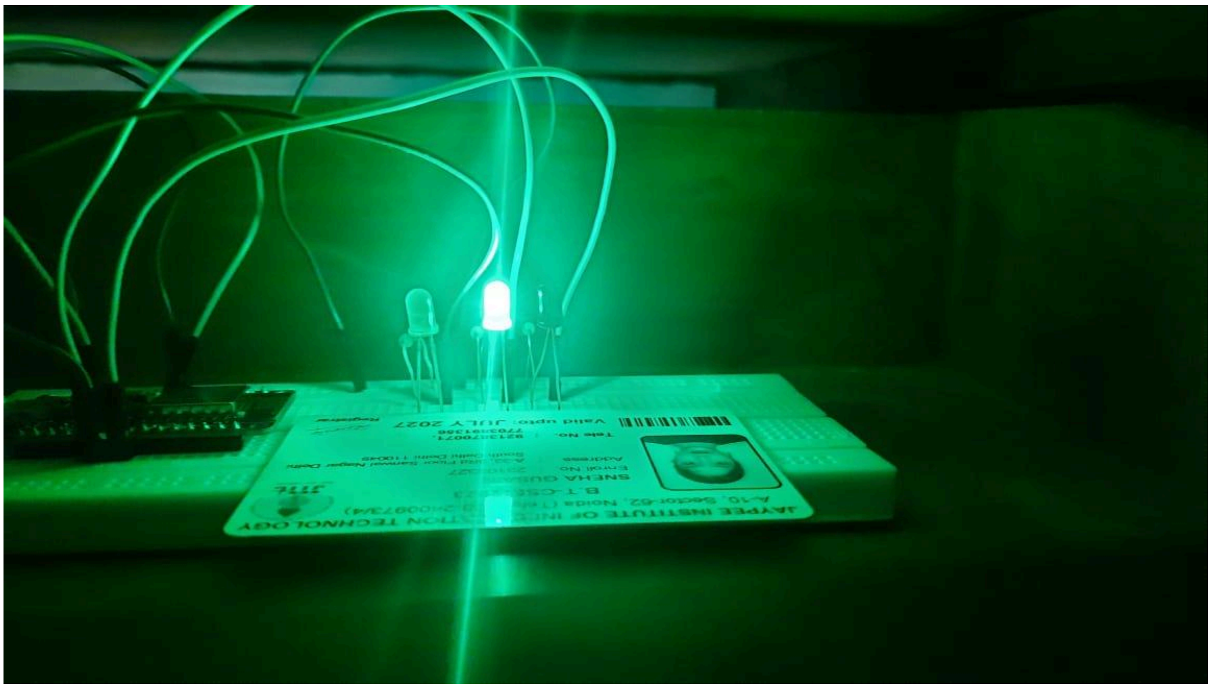
Component	**Quantity**	**Specification**
NodeMCU	1	ESP8266
Red, Green, Yellow LEDs	1 each	-
Resistors	3	220 Ω
Jumper Wires	7	-
Breadboard	1	-

- **Software:**
  - Arduino IDE
  - Python 3 for scripting and automation
  - OpenCV for image capture and clarity analysis
  - NumPy and Matplotlib for numerical processing and visualization

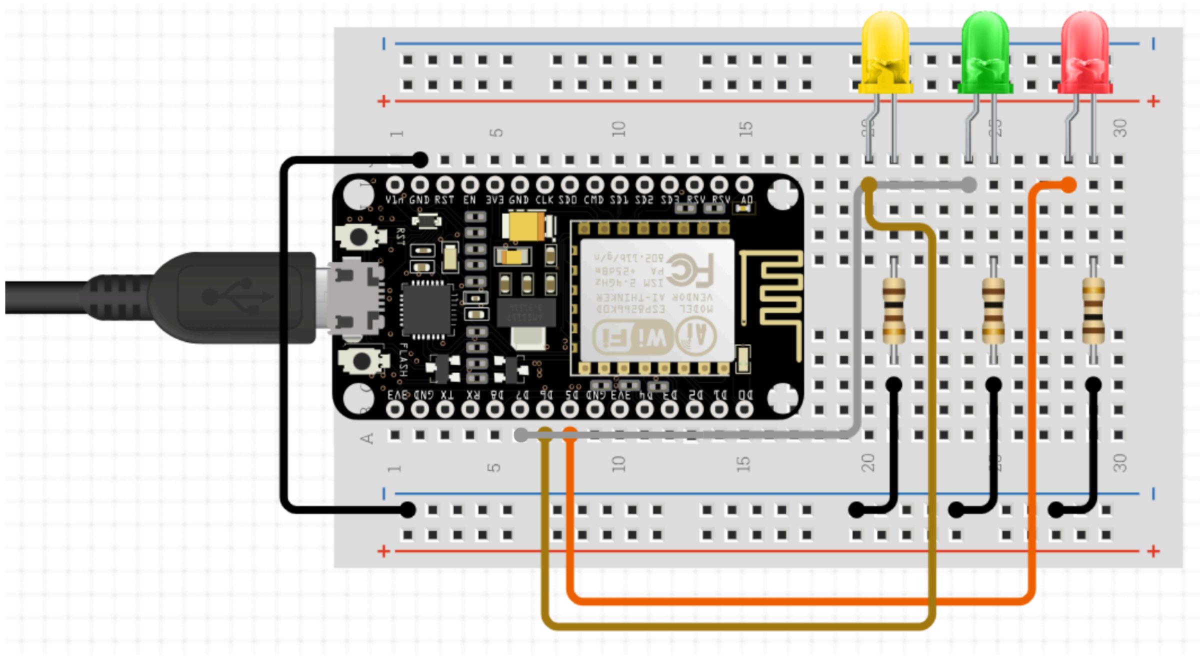
Working Model



Schematic/Circuit Diagram



Connections and Implementation Details/ Pin Configuration

Connections for NodeMCU (ESP8266):

- VIN (NodeMCU) → USB 5V supply
- GND (NodeMCU) → Breadboard – rail (common ground)

Breadboard Power Connections:

- NodeMCU GND → Breadboard – rail

- All LED grounds connect to the common ground rail

Connections for LEDs:
Red LED:

- Anode (long leg) → D5 (GPIO14)
- Cathode (short leg) → GND (breadboard) through a 220 Ω resistor

Yellow LED:

- Anode → D6 (GPIO12)
- Cathode → GND through a 220 Ω resistor

Green LED:

- Anode → D7 (GPIO13)
- Cathode → GND through a 220 Ω resistor

USB Power Supply:

- NodeMCU powered via Micro-USB cable
- Provides 5V input and regulated 3.3V output for the LED circuit

Code

Arduino IDE Code for NodeMCU to switch LEDs

```
int RED_PIN = D5;
int YELLOW_PIN = D6;
int GREEN_PIN = D7;

String incoming = "";
int state = 0;
// 0 = RED, 1 = YELLOW, 2 = GREEN

void setup() {
  Serial.begin(115200);

  pinMode(RED_PIN, OUTPUT);
  pinMode(YELLOW_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);

  digitalWrite(RED_PIN, LOW);
  digitalWrite(YELLOW_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);

  Serial.println("ESP_READY");
}
```

```
void loop() {
  // Listen for commands from laptop
  if (Serial.available()) {
    incoming = Serial.readStringUntil('\n');
    incoming.trim();

    if (incoming == "NEXT") {
      state = (state + 1) % 3;  // cycle 0 → 1 → 2 → repeat
      setLED(state);
    }
  }
}

void setLED(int s) {
  if (s == 0) {
    digitalWrite(RED_PIN, HIGH);
    digitalWrite(YELLOW_PIN, LOW);
    digitalWrite(GREEN_PIN, LOW);
    Serial.println("READY RED");
  }

  if (s == 1) {
    digitalWrite(RED_PIN, LOW);
    digitalWrite(YELLOW_PIN, HIGH);
    digitalWrite(GREEN_PIN, LOW);
    Serial.println("READY YELLOW");
  }

  if (s == 2) {
    digitalWrite(RED_PIN, LOW);
    digitalWrite(YELLOW_PIN, LOW);
    digitalWrite(GREEN_PIN, HIGH);
    Serial.println("READY GREEN");
  }
}
```

Python Script for Image Capture and Metrics Analysis

```
import serial
import time
import cv2
import os
import numpy as np
```

```python
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd

# 1. SETUP SERIAL + CAMERA

ser = serial.Serial("COM4", 115200)
cam = cv2.VideoCapture(0)

colors = ["RED", "YELLOW", "GREEN"]

# Create required directories
os.makedirs("results/saved_imgs", exist_ok=True)
os.makedirs("results/metrics/plots", exist_ok=True)

# 2. METRIC FUNCTIONS

def compute_metrics(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    metrics = {}
    metrics["sharpness"] = cv2.Laplacian(gray, cv2.CV_64F).var()
    metrics["contrast"] = np.std(gray)
    metrics["brightness"] = np.mean(gray)

    hist = cv2.calcHist([gray], [0], None, [256], [0, 256]).flatten()
    hist /= hist.sum()
    metrics["entropy"] = -np.sum(hist * np.log2(hist + 1e-10))

    return metrics

# 3. CAPTURE IMAGES USING HANDSHAKE WITH NODEMCU

def capture_images():
    for color in colors:

        # Send handshake: request next LED
        ser.write(b"NEXT\n")
        print("Sent: NEXT")

        # Wait for NodeMCU to confirm LED is ON
        while True:
            line = ser.readline().decode().strip()
            print("Received:", line)
            if line == f"READY {color}":
                print(f">>> {color} LED active. Capturing image…")
                break

        # Capture image
        ret, frame = cam.read()
        filename = f"results/saved_imgs/{color}.jpg"
        cv2.imwrite(filename, frame)
        print(f"Saved {filename}")

        time.sleep(0.5)  # small buffer
```

# 4. COMPUTE METRICS FOR EACH IMAGE

```
def analyze_metrics():
    metrics_list = []

    for color in colors:
        img_path = f"results/saved_imgs/{color}.jpg"
        img = cv2.imread(img_path)

        if img is None:
            print(f"Error: Could not load {img_path}")
            continue

        m = compute_metrics(img)
        m["color"] = color
        metrics_list.append(m)

    df = pd.DataFrame(metrics_list)
    df.to_csv("results/metrics/metrics.csv", index=False)

    print("\nMetrics saved to results/metrics/metrics.csv\n")
    return df
```

# 5. DETERMINE BEST LIGHTING SHADE

```
def find_best(df):
    best_row = df.loc[df["sharpness"].idxmax()]
    best_color = best_row["color"]

    with open("results/metrics/best_color.txt", "w") as f:
        f.write(f"Best lighting condition: {best_color}\n")
        f.write(best_row.to_string())

    print(f"\n>>> BEST LIGHTING: {best_color}\n")
    return best_color

# 6. PLOT COMPARISON GRAPHS

def plot_graphs(df):

    # SHARPNESS
    plt.figure(figsize=(6,4))
    plt.bar(df["color"], df["sharpness"])
    plt.title("Sharpness Comparison")
    plt.xlabel("Color")
    plt.ylabel("Sharpness")
    plt.savefig("results/metrics/plots/sharpness_plot.png")
    plt.close()

    # CONTRAST
    plt.figure(figsize=(6,4))
    plt.bar(df["color"], df["contrast"])
    plt.title("Contrast Comparison")
    plt.xlabel("Color")
    plt.ylabel("Contrast")
    plt.savefig("results/metrics/plots/contrast_plot.png")
    plt.close()

    # BRIGHTNESS
    plt.figure(figsize=(6,4))
    plt.bar(df["color"], df["brightness"])
    plt.title("Brightness Comparison")
    plt.xlabel("Color")
    plt.ylabel("Brightness")
    plt.savefig("results/metrics/plots/brightness_plot.png")
    plt.close()

    print("Plots saved in results/metrics/plots/")

# 7. MAIN

print("Starting capture process…\n")
capture_images()

print("\nComputing metrics…\n")
```

```
df = analyze_metrics()

print("\nDetermining best color…\n")
best = find_best(df)

print("\nGenerating comparison graphs…\n")
plot_graphs(df)

print("\n\n>>> PROCESS COMPLETE! <<<\n")

cam.release()
```

Results and Analysis

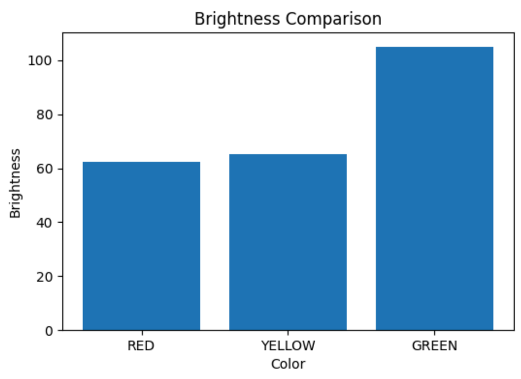
Images captured under **Red, Yellow,** and **Green** illumination were evaluated using four clarity metrics: **sharpness, contrast, brightness,** and **entropy**. The results are summarized below:

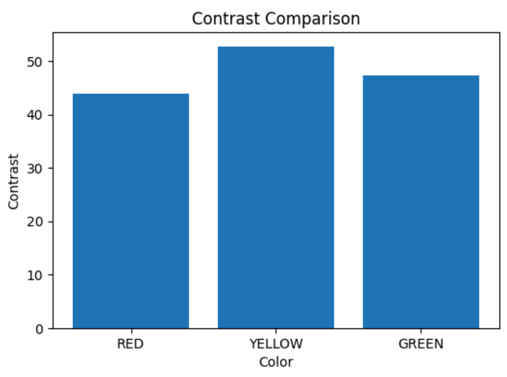
Color	Sharpness	Contrast	Brightness	Entropy
Red	30.74	43.87	62.35	6.58
Yellow	46.29	52.84	65.34	7.28
Green	276.72	47.28	105.10	6.94

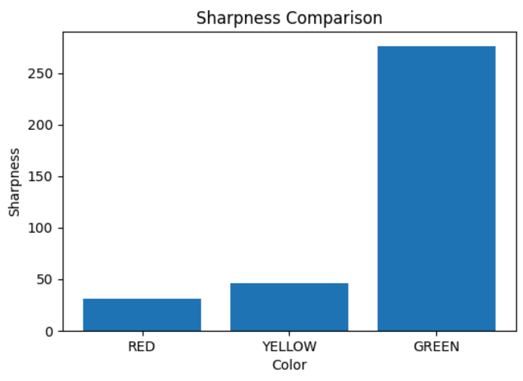
Best Light per Metric:

- Sharpness: Green (highest edge clarity)
- Brightness: Green (strongest illumination)
- Contrast: Yellow (best separation of text and background)
- Entropy: Yellow (richest information content)

Graph Interpretation:







- The Sharpness plot clearly shows Green outperforming all others by a large margin.
- The Contrast Plot shows Yellow having the highest contrast.
- The Brightness Plot shows Green producing the brightest image, followed by Yellow and Red.

Overall Best Illumination:

Although Yellow performed well in contrast and entropy, Green illumination produced the best overall image clarity due to significantly higher brightness and sharpness.

Conclusion

This project investigated how different illumination colors: Red, Yellow, and Green, affect the clarity of images captured using a laptop webcam. Using a controlled hardware setup with LED lights and automated image capture, several image quality metrics were computed, including sharpness, brightness, contrast, and entropy.

The results showed that Green illumination produced the clearest images overall, with significantly higher sharpness and brightness compared to Red and Yellow. This aligns with the behavior of modern camera sensors, which are designed with a stronger response to green wavelengths due to the Bayer filter pattern. Yellow performed well in contrast and entropy, indicating richer texture and better differentiation of image regions, while Red consistently showed the lowest clarity across all metrics.

Overall, the findings demonstrate that illumination color has a measurable impact on image clarity, and among the tested conditions, Green light provides the most effective illumination for capturing sharp and well-defined images. The metric-based comparison and plotted graphs further validate this conclusion. This experiment also highlights how low-cost hardware like LEDs and microcontrollers can be used to analyze and optimize imaging conditions in real environments.

References

1. Howse, J. (2013). *OpenCV computer vision with python* (Vol. 27). Birmingham, UK: Packt Publishing.

2. Chakraborty, S., & Aithal, P. S. (2024). Communication Channels Review For ESP Module Using Arduino IDE And NodeMCU. *International Journal of Applied Engineering and Management Letters(IJAEML)*, *8*(1), 1-14.
3. Wang, Y., & Durmus, D. (2025). The Effect of Color Contrast on the Visual Clarity of Images of Complex Indoor Environments. *Buildings*, *15*(7), 1157.
4. Wu, W. C., Zhao, H., & Liu, W. W. (2009). Effects of illumination on image quality in precision vision measurement. *J. Shanghai Jiaotong Univ*, *43*, 391-395.
5. https://nodemcu.readthedocs.io/en/release/
6. https://docs.opencv.org/4.x/index.html