BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 460 (July 2023)
Optoelectronics Laboratory

Final Project Report

Section: G1 Group: 06

Screen Safety & Optical Efficiency assessment with a spectrometer

Course Instructors:
Dr. Muhammad Anisuzzaman Talukder
Tanushri Medha Kundu

Signature of Instructor: ______________________________

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. <u>Type the student ID and name, and put your signature</u>. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: ______________ **Full Name: Subhan Zawad Bihan** **Student ID: 1906037**	**Signature:** ______________ **Full Name: Abrar Md. Mahir** **Student ID: 1906039**
Signature: ______________ **Full Name: Archishman Sarkar** **Student ID: 1906051**	**Signature:** ______________ **Full Name: Bokhtiar Foysol Himon** **Student ID: 1906056**
Signature: ______________ **Full Name: Nilotpaul Kundu Dhrubo** **Student ID: 1906060**	

Table of Contents

1 Abstract

Everyday electronic devices are filled with screens that can harm our eyes. A special spectrometer is constructed to evaluate the viewing safety and optical efficiency of screens by analysing the emission spectrum at various wavelengths. Real time spectrum analysis helps us to find the ultraviolet content And figure how harmful it might be for viewing.

2 Introduction

Blue light is a high-energy, short-wavelength light within the visible spectrum, typically ranging from 380 to 500 nanometers. It is emitted naturally by the sun and artificially by digital screens, LED lights, and fluorescent lighting.

Potential Harm of Blue Light:

1. **Eye Strain and Fatigue**: Prolonged exposure to screens can lead to digital eye strain, causing discomfort, dryness, and blurred vision.
2. **Sleep Disruption**: Blue light suppresses melatonin, a hormone that regulates sleep, disrupting natural sleep-wake cycles when exposed at night.
3. **Retinal Damage:** Some studies suggest prolonged exposure to intense blue light may contribute to retinal damage over time and increase the risk of macular degeneration.

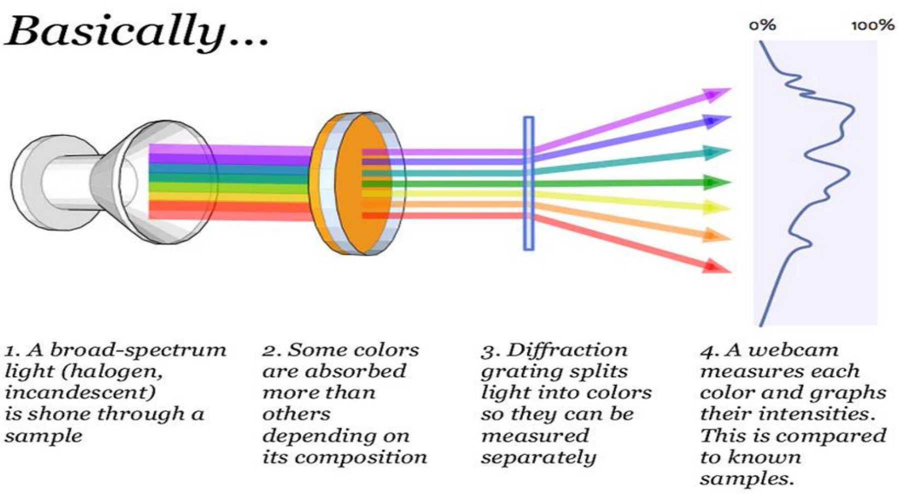
This harmful blue light can be analyzed using a spectrometer. Here is a brief introsuction

A **Spectrometer** is an analytical instrument used to measure and analyze the properties of light over a specific portion of the electromagnetic spectrum. It helps determine the intensity, wavelength, and sometimes the polarization of light, providing insights into the composition, structure, and physical properties of materials.

How It Works:

1. **Light Source**: Light interacts with a sample (through reflection, absorption, or emission).
2. **Dispersing Element**: after the light goes through a slit, a prism or diffraction grating separates the light into its component wavelengths. Diffraction makes all the wavelength present in the source diffract into a unique angle. Thus, a spectra form
3. **Detector**: Measures the intensity of light at different wavelengths from where we can identify the UV content the light may have

The working of a spectrometer is described below pictorially



Using a webcam (detector) and a compact disc (DVD) as a diffraction grating, the device splits the emission spectrum of the screen. The webcam captures the spectrum, and a PC (using Python algorithms) then analyzes the relative radiation intensity at various wavelengths. Heres the process flow

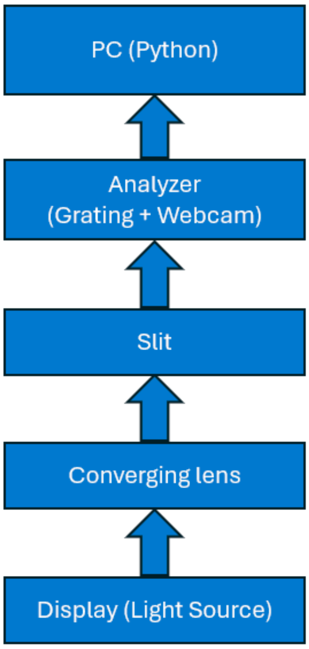


Figure 1.1: Process flow in the spectrometer

3 Design

3.1 Problem Formulation and Analysis

In modern life, digital screens have become common and concerning, raising concerns about their impact on health and energy efficiency. Screens emit blue light, which is known to cause potential harm, including eye strain, sleep disruption, and long-term retinal damage. Simultaneously, the energy efficiency and optical performance of these screens are critical for sustainable technology development.

Despite these concerns, there is limited accessible and cost-effective technology to quantitatively evaluate screen safety and optical efficiency. This project addresses this gap by using a compact, affordable spectrometer setup—combining a webcam, a DVD diffraction grating, and Python-based data analysis—to measure the emission spectrum of digital screens. The goal is to assess the intensity of harmful wavelengths and evaluate optical efficiency, thereby providing a practical solution for screen safety assessment.

3.2 Scope Identification

It can be used in laboratories or households to measure if a device emits harmful contents of the visible or even UV-A spectra.

3.3 Tools Used

1. **Webcam**: Acts as a detector to capture the emission spectrum of the screen.
2. **Compact Disc (**DVD**)**: Serves as a diffraction grating to split light into its component wavelengths.
3. **Python Algorithms**: Used for analyzing the captured spectrum and determining the relative intensity of light at various wavelengths.
4. **PC**: Processes the data from the webcam and performs spectral analysis.
5. **Accessories:** Lens (for focusing light), Cardboard Box, Glue, Tape, Thermal Insulating Plastic, etc.

This simple and cost-effective setup enables precise measurement of screen emission properties.

3.4 Hardware Diagram

The hardware Diagram can be shown as below:

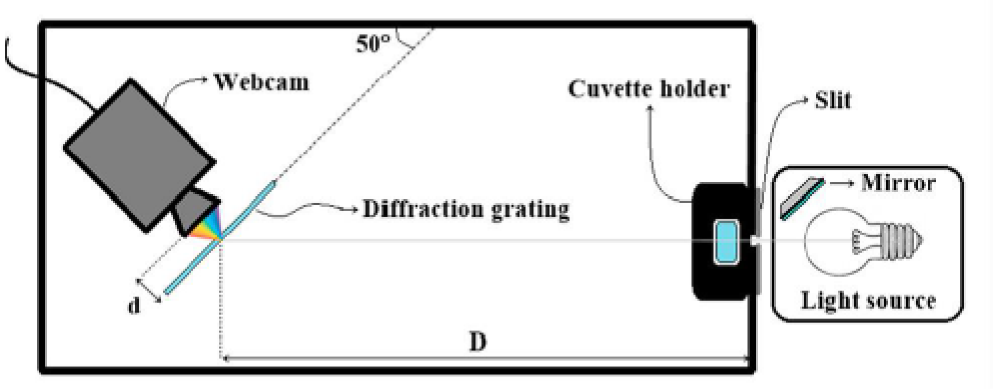


Fig: simple diagram of the spectrometer

3.5 Design Method

The project utilizes a cost-effective spectrometer setup to analyze the emission spectrum of digital screens. A webcam is used as the detector, while a compact disc (DVD) serves as a diffraction grating to split the light into its component wavelengths. The webcam captures the diffracted light spectrum, which is then processed and analyzed using Python-based algorithms on a PC. These algorithms calculate the relative intensity of radiation at various wavelengths, focusing on harmful blue light emissions and overall optical efficiency. This design ensures affordability, simplicity, and accuracy in assessing screen safety and performance.

3.6 Hardware Design

We have designed our hardware by making a black box where no light will enter without our desired light. We made a slit in the box for the source light to be entered. We used black tape and blade to make the slit width as small as possible. Then we setup a webcam. The position of the camera was set by doing some trial and errors. A grating was also implemented in front of the camera for getting the diffraction. The camera was set making an angle with the light coming because for getting required diffraction angle. Camera is set at an angle around 40 degrees. IR filter removed to get the infrared range. Whole Body is Black colored and covered with Thermal Insulating Plastic to reduce IR (detected by webcam). A Lens is used to focus the light of a display (e.g. phone screen) on the slit.

Below are the images of our hardware setup

Figure: View from above

Fig: Using mobile as a source and the black covered spectrometer

4 Implementation of Software

We used Python OpenCV library to capture live video feed from our webcam. Then used the frames from the video to extract the spectrum from it.

4.1 Source Code

Below is the **Analyzer.py** code

```
import numpy as np

# Constants
h = 6.626e-34  # Planck's constant (Js)
c = 3.0e8      # Speed of light (m/s)
e = 1.602e-19  # Joules to eV

file_path1 = "C:\\Users\\SBihan\\Desktop\\Theremino\\Spectrum_bg.txt"
file_path2 = "C:\\Users\\SBihan\\Desktop\\Theremino\\Spectrum_v3.txt"

def integrate_intensity(file_path, start_wavelength, end_wavelength):
    """
    Integrates the intensity with respect to wavelength over a specified range.

    Parameters:
        file_path (str): Path to the spectrum data file.
        start_wavelength (float): Start wavelength for integration.
        end_wavelength (float): End wavelength for integration.

    Returns:
        float: The calculated integral of intensity.
    """

    data = np.loadtxt(file_path, skiprows=1)

    # Extract wavelength and intensity columns
    wavelengths = data[:, 0]
    intensities = data[:, 1]

    # Filter the data within the desired range
    mask = (wavelengths >= start_wavelength) & (wavelengths <= end_wavelength)
    filtered_wavelengths = wavelengths[mask]
    filtered_intensities = intensities[mask]

    # Calculate the integrand: I(lambda) * lambda. h*c is a constant so doesn't matter for ratio
    integrand = filtered_intensities * filtered_wavelengths

    # Perform numerical integration using the trapezoidal rule
    integral = np.trapz(integrand, filtered_wavelengths)

    return integral

def integrate_photon_energy(file_path, start_wavelength, end_wavelength):
    """
    Integrates intensity * photon energy with respect to wavelength over a specified range.
```

```
    Parameters:
        file_path (str): Path to the spectrum data file.
        start_wavelength (float): Start wavelength for integration (in nm).
        end_wavelength (float): End wavelength for integration (in nm).

    Returns:
        float: The calculated integral of intensity * photon energy (in eV·nm).
    """

    data = np.loadtxt(file_path, skiprows=1)

    # Extract wavelength and intensity columns
    wavelengths = data[:, 0]  # in nm
    intensities = data[:, 1]

    # Filter the data within the desired range
    mask = (wavelengths >= start_wavelength) & (wavelengths <= end_wavelength)
    filtered_wavelengths = wavelengths[mask]
    filtered_intensities = intensities[mask]

    # Perform numerical integration using the trapezoidal rule
    integral = np.trapz(filtered_intensities, filtered_wavelengths)

    return integral

visible_light_photons = integrate_intensity(file_path2, 380, 750) - integrate_intensity(file_path1, 380, 750)
visible_light_energy = integrate_photon_energy(file_path2, 380, 750) - integrate_photon_energy(file_path1,
                                                                                          380, 750)

blue_light_photons = integrate_intensity(file_path2, 380, 495) - integrate_intensity(file_path1, 380, 495)
blue_light_energy = integrate_photon_energy(file_path2, 380, 495) - integrate_photon_energy(file_path1, 380, 495)

ir_photons = integrate_intensity(file_path2, 750, 1200) - integrate_intensity(file_path1, 750, 1200)
ir_energy = integrate_photon_energy(file_path2, 750, 1200) - integrate_photon_energy(file_path1, 750, 1200)

blue_light_perc = 100 * blue_light_energy / visible_light_energy
visible_light_efficiency = 100 * visible_light_energy / (visible_light_energy + ir_energy)

print(f"Short-wavelength light percentage (in visible spectrum): {blue_light_perc}%")
print(f"Display Efficiency: {visible_light_efficiency}%")
```

Below is the code for spectrometer.py

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.collections as mcollections
import tkinter as tk
```

```python
from tkinter import filedialog

def find_spectrum_line(frame):
    """
    Extracts the intensity profile along a horizontal line across the entire frame.
    """
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    intensity = np.mean(gray_frame, axis=0)
    return intensity

def map_pixels_to_wavelengths(intensity, wavelength_range=(380, 1200)):
    """
    Maps pixel positions in the intensity profile to wavelengths in the specified range.
    """
    num_pixels = len(intensity)
    wavelengths = np.linspace(wavelength_range[0], wavelength_range[1], num_pixels)
    return wavelengths

camera_index = 1  # Update this index based on your camera setup
cap = cv2.VideoCapture(camera_index)

if not cap.isOpened():
    print("Error: Unable to access the camera")

# Add before the main loop
wavelength_accumulator = []
intensity_accumulator = []

# Parameters
wavelength_range = (300, 1200)  # Wavelength range in nm
# Define the wavelength range for visible spectrum
visible_min, visible_max = 380, 750

print("Press 'q' to quit")

plt.ion()  # Enable interactive mode for real-time plotting

try:
    while True:
        ret, frame = cap.read()
        if not ret:
            print("Error: Unable to capture frame")
            break

        # Extract the intensity profile
        intensity = find_spectrum_line(frame)
        wavelengths = map_pixels_to_wavelengths(intensity, wavelength_range)

        # Accumulate data
```

```
    wavelength_accumulator.append(wavelengths)
    intensity_accumulator.append(intensity)

    # Find indices for UV and IR wavelengths
    uv_indices = [i for i, x in enumerate(wavelengths) if x >= wavelength_range[0] and x < visible_min]
    ir_indices = [i for i, x in enumerate(wavelengths) if x > visible_max and x <= wavelength_range[1]]
    visible_indices = [i for i, x in enumerate(wavelengths) if x >= visible_min and x <= visible_max]

    # Extract corresponding intensities
    uv_wavelengths = wavelengths[uv_indices]
    uv_intensities = intensity[uv_indices]

    ir_wavelengths = wavelengths[ir_indices]
    ir_intensities = intensity[ir_indices]

    visible_wavelengths = wavelengths[visible_indices]
    visible_intensities = intensity[visible_indices]

    plt.clf()
    # plt.plot(wavelengths, intensity, color='black', linewidth=1, zorder=1)  # Thin black line

    points = np.array([visible_wavelengths, visible_intensities]).T.reshape(-1, 1, 2)
    segments = np.concatenate([points[:-1], points[1:]], axis=1)

    lc = mcollections.LineCollection(segments, cmap='rainbow',
                                     norm=plt.Normalize(visible_wavelengths.min(), visible_wavelengths.max()))
    lc.set_array(visible_wavelengths)
    lc.set_linewidth(2)
    plt.gca().add_collection(lc)

    plt.plot(uv_wavelengths, uv_intensities, color='black', linewidth=1, zorder=1)
    plt.plot(ir_wavelengths, ir_intensities, color='black', linewidth=1, zorder=1)

    # Create scatter plot with custom colormap
    scatter = plt.scatter(wavelengths, intensity, c=wavelengths, cmap='rainbow', vmin=visible_min, vmax=visible_max,
s=5, zorder=2)
    plt.colorbar(scatter, label='Wavelength (nm)')

    plt.scatter(uv_wavelengths, uv_intensities, color='black', s=5, zorder=2)
    plt.scatter(ir_wavelengths, ir_intensities, color='black', s=5, zorder=2)
    plt.xlabel('Wavelength (nm)')
    plt.ylabel('Intensity')
    plt.title('Real-Time Intensity Spectrum')
    plt.pause(0.001)

    # Show the live webcam feed
    cv2.imshow('Spectrometer View', frame)

    # Handle key events
```

```
        key = cv2.waitKey(1) & 0xFF
        if key == ord('q'):
            break

finally:
    # Cleanup
    cap.release()
    cv2.destroyAllWindows()
    plt.close()

    # Prompt for file saving
    root = tk.Tk()
    root.withdraw()  # Hide the main window

    # Open file dialog to choose save location
    file_path = filedialog.asksaveasfilename(
        defaultextension=".txt",
        filetypes=[("Text files", "*.txt"), ("All files", "*.*")],
        title="Save Spectrum Data"
    )

    # Save data if a file path is selected
    if file_path:
        # Calculate time-averaged wavelengths and intensities
        avg_wavelengths = np.mean(wavelength_accumulator, axis=0)
        avg_intensities = np.mean(intensity_accumulator, axis=0)

        # Create a formatted array for saving
        data = np.column_stack((avg_wavelengths, avg_intensities))

        # Save to file with formatted header
        np.savetxt(file_path, data,
                   header='  nm\t\t\tI',
                   fmt='%.2f\t\t%.5f',
                   comments='')

        print(f"Spectrum data saved to {file_path}")
```

Code explanation:

The Python script "Spectrometer.py" is designed to collect real-time spectral data of refracted light from a webcam, covering the UV-A, visible, and infrared (IR) regions. The script computes a running average of the spectral data, which is then saved to a text file at the end of the process. This data collection is performed both with and without a light source, similar to how radioactivity measurements are conducted.

The second script, "Analyser.py," processes the spectral data from both the light source (with illumination) and the background (without illumination). The background data is subtracted from the light source data to isolate the energy difference. The script then calculates the integral of intensity with respect to wavelength across the desired spectral regions and provides the ratio or percentage of the results.

5 Design Analysis and Evaluation

5.1 Novelty

1. **Cost-Effective Design**: Unlike conventional spectrometers that are expensive and complex, this project uses affordable components like a webcam and a DVD, making it accessible to a wider audience.
2. **Detecting Blue lights:** while the regular spectrometers detect light of all colors we focus on the harmful blue lights. We detect to refrain from using or taking regular intervals between using them
3. **Focus on Screen Safety**: While traditional spectrometers are used for general spectral studies, this project specifically targets the health and efficiency concerns of digital screens, addressing real-world problems.

5.2 Design Considerations

The project is designed to be simple, affordable, and reliable. A webcam is used as a detector for its accessibility and sensitivity, while a compact disc (DVD) serves as a low-cost diffraction grating to split light into wavelengths. Proper alignment of the light source, DVD, and webcam is crucial for accurate spectral capture. Python algorithms enable precise and customizable data analysis. The setup is portable, user-friendly, and suitable for both educational and practical applications.

5.3 Considerations to public health and safety

This project addresses public health concerns by analyzing blue light emissions from screens, which are linked to eye strain, sleep disruption, and potential long-term retinal damage. By providing insights into screen safety, it empowers users to make informed decisions about screen usage and protection. The cost-effective approach also promotes widespread adoption, enabling more people to assess and mitigate potential health risks associated with digital screens.

5.4 Considerations to environment

The project promotes energy efficiency by evaluating the optical performance of screens, encouraging the use of sustainable and eco-friendly display technologies. Its cost-effective and reusable setup reduces reliance on expensive, resource-intensive equipment, minimizing environmental impact.

5.5 Considerations to cultural and societal needs

This project addresses the societal need for safer screen usage as digital devices become integral to work, education, and entertainment. By providing accessible tools for assessing screen safety, it supports health awareness across diverse communities and encourages responsible technology use.

5.6 Investigations

5.7 Data

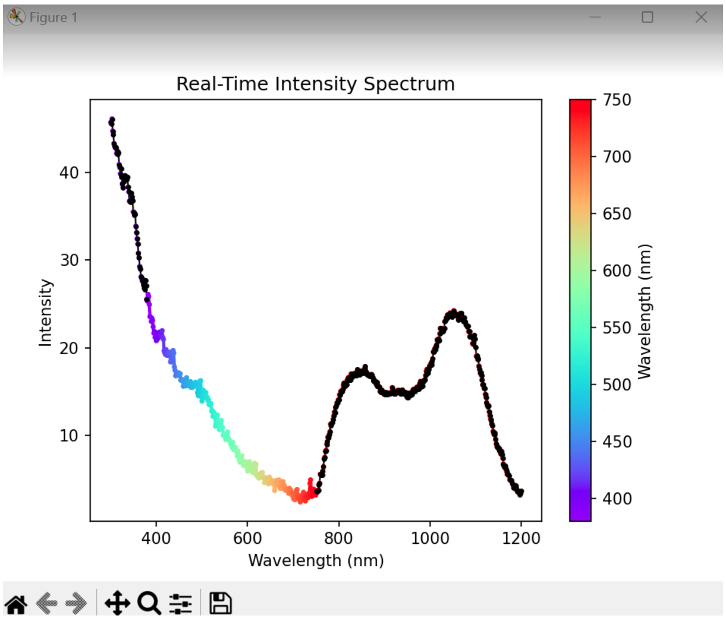


Fig: White Torch (Phone) Spectrum

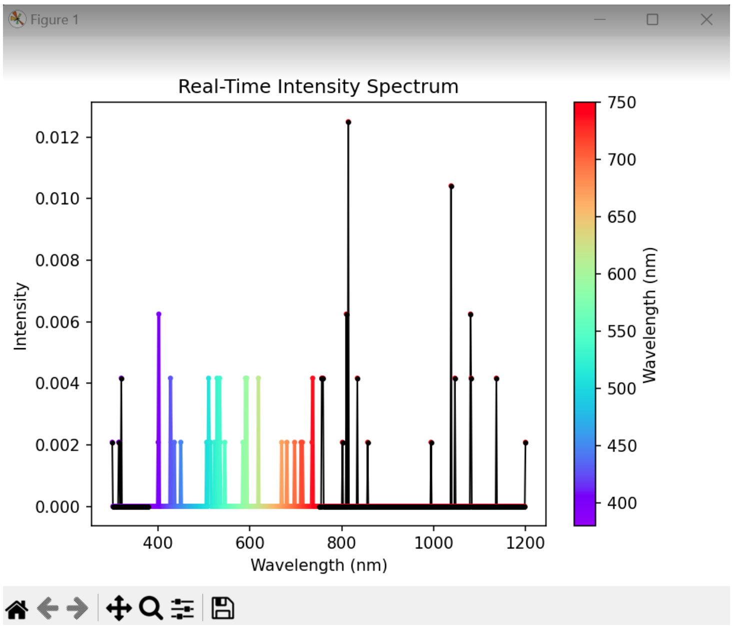


Fig: Mobile Spectrum when video with blue content was played

Data for spectrum_bg.txt (partial)

nm	I
300.00	0.00974
301.41	0.00335
302.82	0.00136
304.23	0.00034
305.63	0.00042
307.04	0.00010
308.45	0.00058
309.86	0.00058
311.27	0.00090
312.68	0.00056
314.08	0.00106
315.49	0.00040
316.90	0.00102
318.31	0.00020
319.72	0.00060
321.13	0.00022
322.54	0.00116
323.94	0.00054

Data for spectrum_torch.txt (partial)

nm	I
300.00	47.60349
301.41	47.38061
302.82	47.97808
304.23	46.72106
305.63	46.17363
307.04	44.90902
308.45	44.37764
309.86	43.56297
311.27	43.59831
312.68	43.06284
314.08	43.07778
315.49	42.76178
316.90	42.72809
318.31	41.66621
319.72	41.26138
321.13	40.59692
322.54	40.84114
323.94	40.29000
325.35	39.78443
326.76	39.29056
328.17	39.99864
329.58	39.54448
330.99	39.60427
332.39	39.14422

Data for spectrum_v1.txt (partial)

nm	I
300.00	0.01560
301.41	0.00673
302.82	0.00367
304.23	0.00132
305.63	0.00163
307.04	0.00094
308.45	0.00179
309.86	0.00114
311.27	0.00273
312.68	0.00179
314.08	0.00318
315.49	0.00219
316.90	0.00349
318.31	0.00169
319.72	0.00258
321.13	0.00180
322.54	0.00401
323.94	0.00202
325.35	0.00232
326.76	0.00150
328.17	0.00268
329.58	0.00212
330.99	0.00388

Data for spectrum_v2.txt (partial)

nm	I
300.00	0.00483
301.41	0.00167
302.82	0.00215
304.23	0.00043
305.63	0.00127
307.04	0.00041
308.45	0.00111
309.86	0.00041
311.27	0.00124
312.68	0.00069
314.08	0.00236
315.49	0.00207
316.90	0.00697
318.31	0.00206
319.72	0.00253
321.13	0.00083
322.54	0.00285
323.94	0.00092
325.35	0.00148
326.76	0.00037
328.17	0.00174

Data for spectrum_v3.txt (partial)

nm	I
300.00	0.01719
301.41	0.00585
302.82	0.00326
304.23	0.00086
305.63	0.00122
307.04	0.00064
308.45	0.00212
309.86	0.00150
311.27	0.00311
312.68	0.00218
314.08	0.00406
315.49	0.00225
316.90	0.00590
318.31	0.00157
319.72	0.00240
321.13	0.00140
322.54	0.00375
323.94	0.00153
325.35	0.00211

Data for spectrum_v4.txt (partial)

nm	I
300.00	0.01719
301.41	0.00585
302.82	0.00326
304.23	0.00086
305.63	0.00122
307.04	0.00064
308.45	0.00212
309.86	0.00150
311.27	0.00311
312.68	0.00218
314.08	0.00406
315.49	0.00225
316.90	0.00590
318.31	0.00157
319.72	0.00240
321.13	0.00140
322.54	0.00375
323.94	0.00153
325.35	0.00211
326.76	0.00107
328.17	0.00318
329.58	0.00143
330.99	0.00369
332.39	0.00144

5.8 Results

Torch Short-wavelength light percentage (in visible spectrum): 49.82%
Display Efficiency: 34.74%

Display Video-1: Short-wavelength light percentage (in visible spectrum): 27.56%
Display Efficiency: 63.96%

Display Video-3: Short-wavelength light percentage (in visible spectrum): 35.57%
Display Efficiency: 51.26836337796343%

High-intensity blue light: Short-wavelength light percentage (in visible spectrum): 78.30%
Display Efficiency: 78.074%

6 Reflection on Individual and Team work

ID	Task
1906037	Software python
1906039	Hardware Setup
1906051	Hardware Setup
1906056	Software python
1906060	Hardware Setup

7 Cost Analysis

Components	**Price (BDT)**
DVD	150
Webcam	1200
Lens	200
Black tape	60
PVC board	200
Glue Stick	60
Insulating Poly	20
Total	**1890**

8 Future Work

Future work on this project can focus on improving the accuracy and versatility of the spectrometer setup. Using a stronger lens would better the spectral data. A better Black room would reduce noise generated randomly. Advanced machine learning algorithms could be integrated to enhance the analysis of spectral data and automatically classify harmful emission levels. Expanding the system to evaluate a wider range of devices, such as LEDs, smartphones, and televisions, would increase its applicability. Additionally, creating a user-friendly mobile application or software interface could make the tool more accessible to non-experts. Research could also explore the integration of alternative low-cost diffraction gratings and sensors for greater precision and scalability. Finally, collaborating with educational institutions and industries could help standardize this tool for widespread use in health, research, and sustainability efforts.

9 References

Link 1: https://www.instructables.com/DIY-Low-Cost-Spectrometer/?fbclid=IwZXh0bgNhZW0CMTAAAR1Ebac3NjzkwonpJbCszZXKClQAGTEuKk3vBUpZHDSqMP9GXxCqkQA0gtg_aem_e9XoGf4LiZrvdmcB-L8H_Q