



# **Lambertian Diffuse Reflectance - MAE 157 Project**

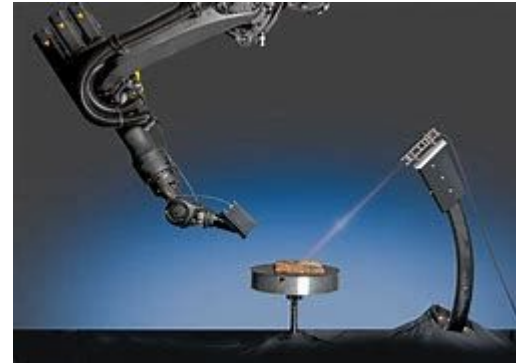
Group 1

# Project Overview

---

# Experimental Purpose

- Replicate Lambert's cosine law to show the reflectance of a near diffuse surface
- Determine the light absorption of this supposed diffuse surface and approximate the reflectivity coefficient of it using the Bidirectional reflectance distribution function
- Use the Lambertian model to demonstrate the effect of sampling rate on image resolution
- Design a scaled down simple goniometer



Goniometer

# Theoretical Overview

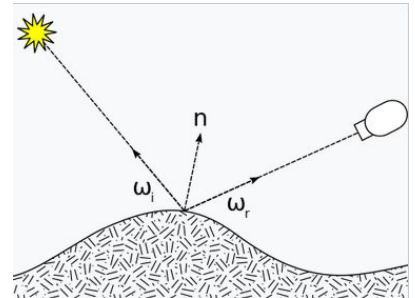
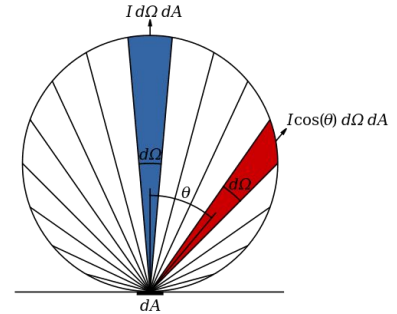
—

# Theory: Lambert & Bidirectional reflectance

- Lambert's cosine law says that the radiant intensity reflecting off of an isotropic material is directly proportional to the cosine of the angle  $\theta$  between the observer's line of sight and the surface normal  $I = I_0 \cos \theta$ 
  - The reflectivity coefficient  $\rho$  is the ratio of irradiance to radiance ( $0 < \rho < 1$ )
  - Reflectance behavior is described by the unit sphere
  - Assumes no absorption of light by the surface  $\rightarrow$  conservation of energy applies
  - Light intensity is independent of viewing angle  $\rightarrow$  observed illuminance (brightness) is equivalent in all directions
- The bidirectional reflectance distribution function (BRDF) defines how light is reflected at a surface. The function takes two direction inputs, composed of four independent angles

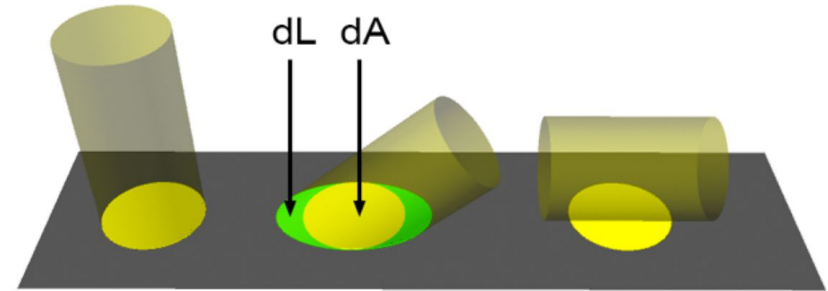
$$f(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{L(\theta_r, \phi_r)}{E(\theta_i, \phi_i)} \left( \frac{1}{\text{sr}} \right) \quad f_{\text{lambertian}}(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{\rho_d}{\pi}$$

- Various BRDF parameters are combined to account for body reflection (diffuse) and surface reflection (specular scattering) of a line from a surface (i.e microfacets used to approximate surface irregularities)
- Specular scattering is a result of surface irregularities, material properties
  - For this experiment an attempt was made to create a diffuse surface hence specular scattering was assumed to be 0



## Theory: Differential Beam Area

- The laser when striking a surface will produce a cross section of  $dA$  when it is normal to the surface
- As the incident angle of the beam shifts the cross-section of the beam with the surface becomes larger than  $dA$ 
  - The photons are distributed over a larger region than  $dA \rightarrow dA + dL$
- $dA$  receives less energy as the angle between the beam and the normal ( $\theta_i$ ) increases
- This forms the basis of Lambert's law



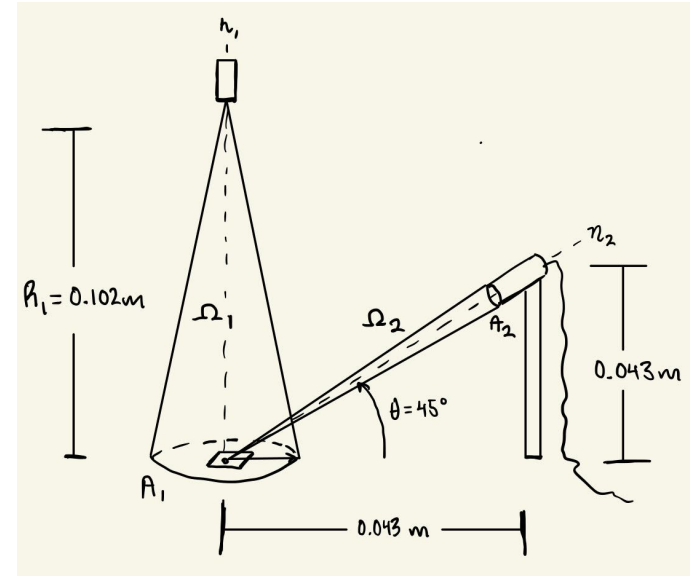
# Theory: Defining our System

Maximum power output  $\Phi = 5 \text{ mW}$

Assumptions:

- The incident light is a perfect laser light (inverse square law does not apply)
- Conservation of Energy applies
- The material emits diffusely  $\rightarrow$  unit sphere defines reflectance geometry
- Light is evenly distributed through space
- The Difference between radiance and irradiance is the object reflectivity

To determine the best model for our apparatus and electrical configuration we approximated the range of light intensities we are going to be working with in this experiment so we can calibrate the laser over the corresponding lux range. All theoretical calculations are made from  $\theta_s = 0$  at a fixed sensor distance.



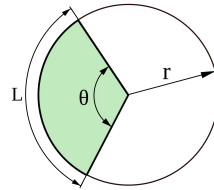
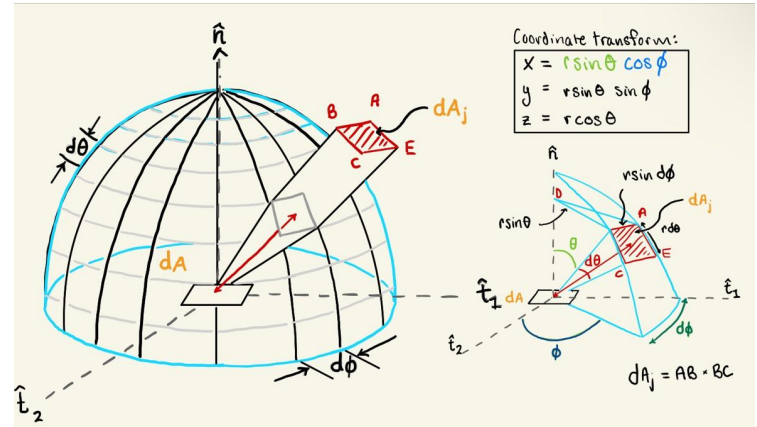
$$A_{\text{photoresister}} = \pi \cdot (0.005\text{m})^2 = 7.85 \cdot 10^{-5} \text{m}^2$$

$$A_{beam} = \pi \cdot (0.002 \text{ m}^2) = 1.26 \cdot 10^{-5} \text{ m}^2$$

$$R_1 = 0.102 \text{ m}, R_2 = 0.0608 \text{ m}$$

- 2D angles projected in 3D
- Path light travels through

$$d\Omega_{i,r} = \frac{dA'}{r^2} = \frac{dA \cos(\theta_{i,r})}{r^2}$$


$$d\theta = \frac{dl}{r}$$

$$\Omega_1 = \frac{A_{beam} \cdot \cos(\theta_1)}{R_1^2} = \frac{1.26 \cdot 10^{-5} m^2 \cdot \cos(0)}{(0.102 m)^2} = 1.21 \cdot 10^{-3} sr$$

$$\Omega_2 = \frac{A_{\text{photoresistor}} \cdot \cos(\theta_1)}{R_2^2} = \frac{7.85 \cdot 10^{-5} \text{ m}^2 \cdot \cos(0)}{(0.0608 \text{ m})^2} = 2.12 \cdot 10^{-2} \text{ sr}$$

$$\Phi_g = 0.005 W, \Phi_v = (0.005 w) \cdot (73.1 \frac{W}{lm}) = 0.37 lm$$

- Convert radiometric functions to photometric
- Luminosity function  $v(\lambda)$  gives the spectral efficacy at a given wavelength

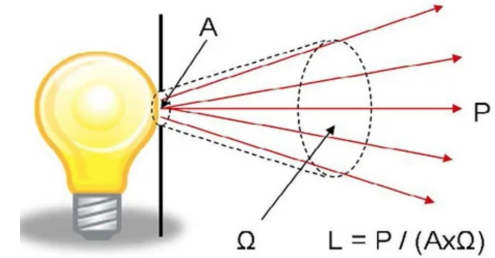
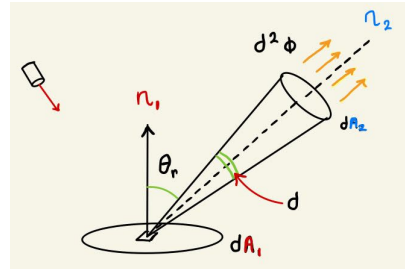
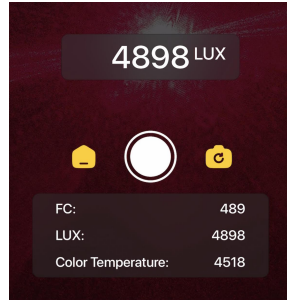
$$K_m = (683 \frac{W}{lm}) \cdot v(\lambda) = (683 \frac{W}{lm}) \cdot v(650 \text{ nm}) = (683 \frac{W}{lm}) \cdot (0.107)$$

$$K_m = 73.1 \frac{w}{lm} \text{ (used to convert watts to lumens in our experimental calculations)}$$

$$I_s = \frac{d\Phi}{dA} = \frac{0.005 \text{ W}}{1.26 \cdot 10^{-5} \text{ m}^2} = 397 \frac{\text{W}}{\text{m}^2}, I_v = \frac{0.37 \text{ lm}}{1.26 \cdot 10^{-5} \text{ m}^2} = 2.9 \cdot 10^4 \frac{\text{lm}}{\text{m}^2}$$



## Lux measured by iPhone camera



## Theoretical Incident intensity of photoresistor

( directly underneath the laser)

$$I_v = \frac{\Phi_v}{A_1} = \frac{0.37 \text{ lm}}{7.85 \cdot 10^{-5} \text{ m}^2} = 4713.4 \text{ flux}$$

$$E = \int_0^{\pi/2} \int_0^{2\pi} L_i(\theta, \phi) \cos(\theta) \sin(\theta) d\phi d\theta = \int_{\Omega} L_i(\omega) \cos(\theta) d\omega$$

## Irradiance:

Strength of illumination out of the sample surface. Lambertian definition:

## Radiance:

Radiation power which is emitted from the laser and propagates in solid angle  $\Omega$  per unit area

## Incident Radiance:

$$L_i = \frac{I_e}{\Omega_1} = \frac{397 \frac{\text{W}}{\text{m}^2}}{1.21 \cdot 10^{-3} \text{ sr}} = 3.3 \cdot 10^6 \frac{\text{W}}{\text{m}^2 \text{ sr}}$$

$$\text{Sphere Area} = \int_0^{\pi/2} \int_0^{2\pi} \sin \theta r^2 d\phi d\theta = 2\pi r^2$$

### Emission rate from A<sub>1</sub> to A<sub>p</sub>

$$q_{1 \rightarrow 2} = I_{e,1}(\theta, \phi) \cdot A_1 \cdot \cos(\theta_r) \cdot \Omega_2$$

Since  $A_1$  is a Lambertian surface, the irradiance  $E$  is distributed over the entire diffuse hemisphere and must be normalized by  $\pi$  in the equation  $I_{e,1}(\theta, \phi) = \frac{E}{\pi}$  where  $E = \rho \cdot L_i$  and we assume 100% emission and zero absorption so  $\rho = 1$  and  $E = 397 \frac{W}{m^2}$

$$\Phi_{e,1 \rightarrow 2} = \frac{397 \frac{W}{m^2}}{\pi} \cdot (1.26 \cdot 10^{-5} m^2) \cdot \cos(45) \cdot 2.12 \cdot 10^{-2} sr = 2.4 \cdot 10^{-5} W$$

$$\Phi_{v,1 \rightarrow 2} = 2.4 \cdot 10^{-5} W \cdot (73.1 \frac{lm}{W}) = 1.7 \cdot 10^{-3} lm$$

$$I_{e,1 \rightarrow 2} = \frac{\Phi_{e,1 \rightarrow 2}}{A_p} = \frac{2.4 \cdot 10^{-5} W}{7.85 \cdot 10^{-5} m^2} = 0.31 \frac{W}{m^2}$$

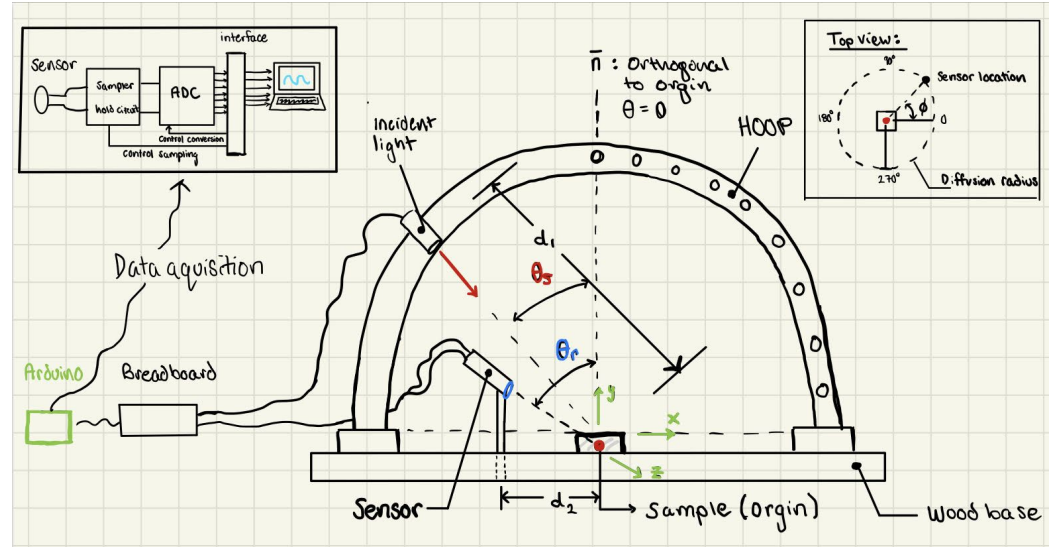
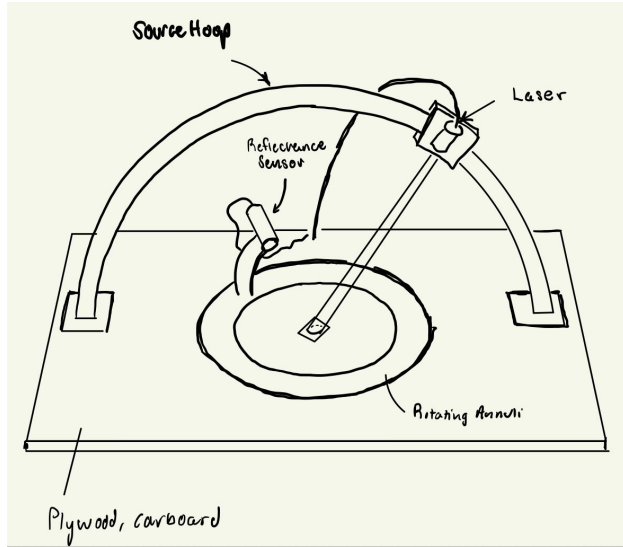
$$I_{v,1 \rightarrow 2} = \frac{\Phi_{v,1 \rightarrow 2}}{A_p} = \frac{1.7 \cdot 10^{-3} lm}{7.85 \cdot 10^{-5} m^2} = 21.65 \text{ flux}$$

$\therefore$  With a laser power output of 0.005 W (or 0.37 lm) approximately 0.000024 W (or 0.0017 lm) is transmitted to the photoresistor at a rate of  $0.31 \frac{W}{m^2}$  or 21.65 flux

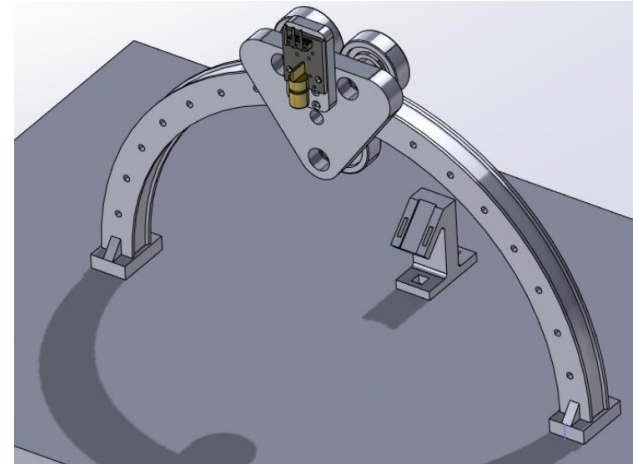
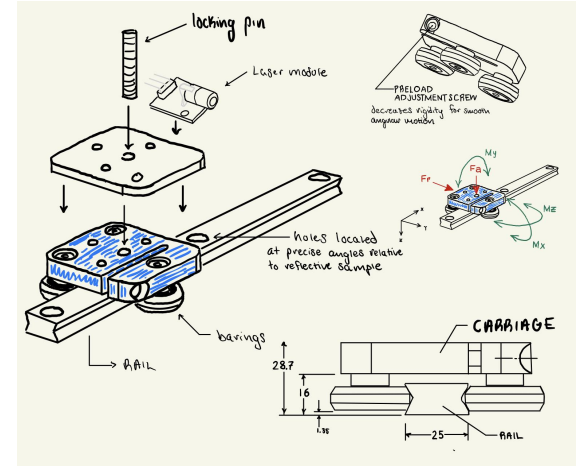
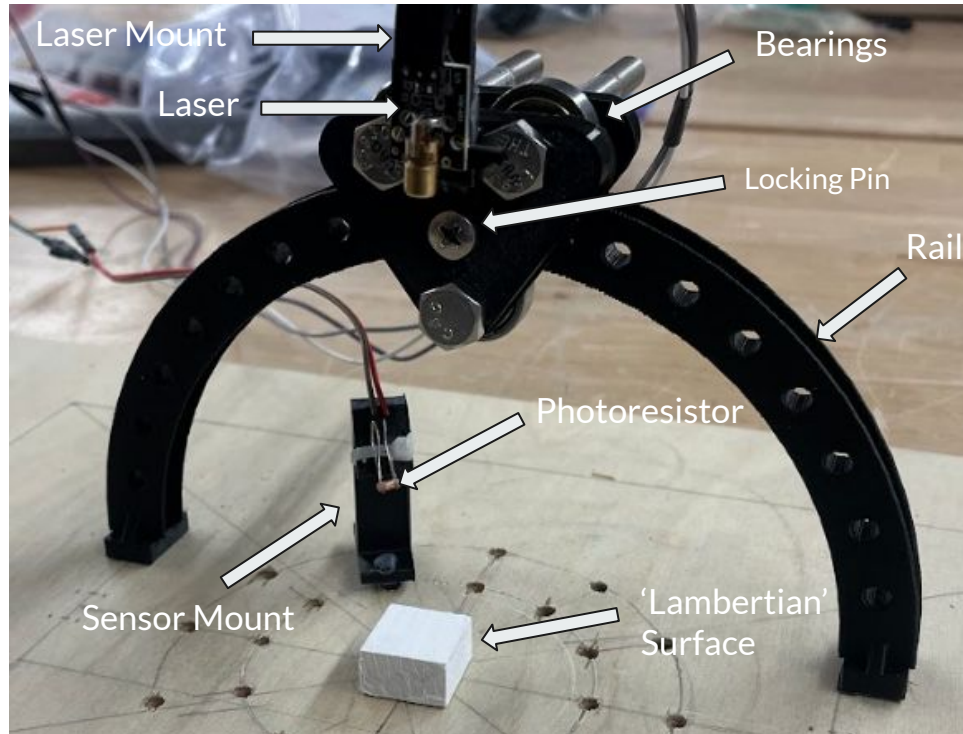
# Experimental Set-Up

—

# Apparatus Overview

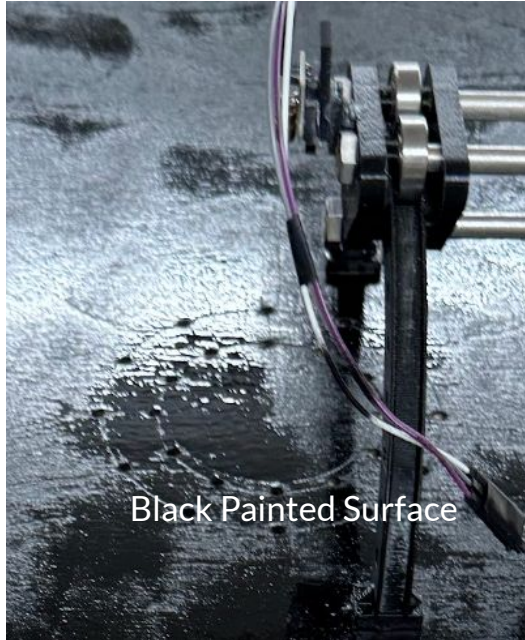


# Mechanical System

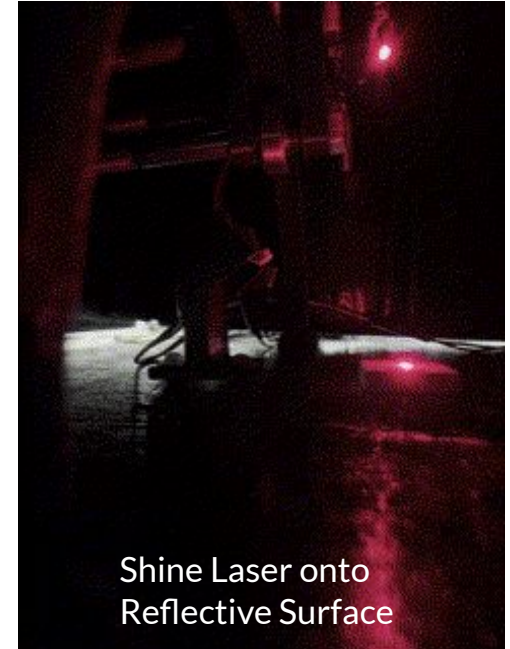


# Mechanical System

---



Cover with Black Box to  
Remove Ambient Light





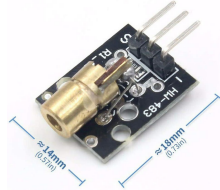
# Electrical System

1. **Arduino UNO Microcontroller**

- a. 5V supplied by arduino

2. **KY-008 Arduino Laser**

- a.  $\lambda=650\text{ nm}$ ,  $\Phi=5\text{ mW}$

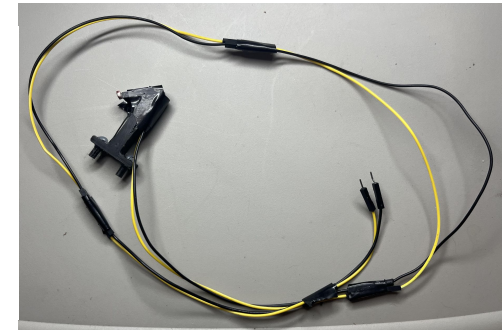
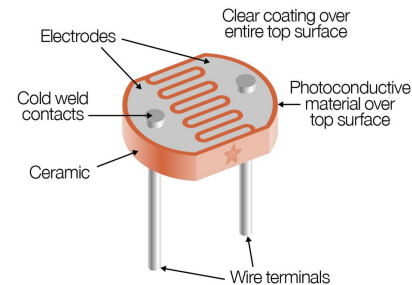
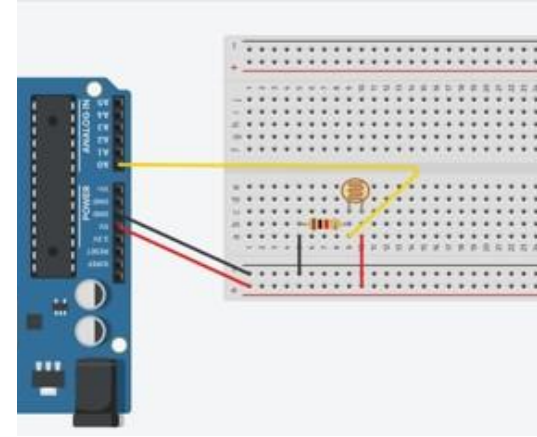
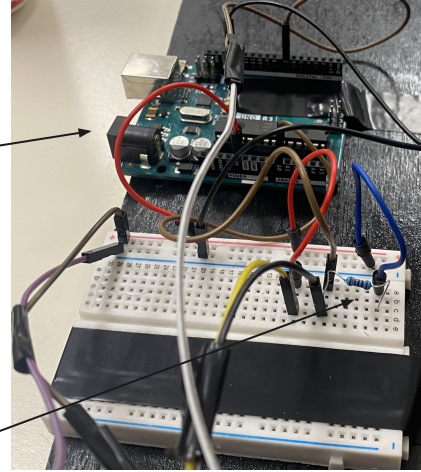


3. **Fixed resistor**

- a.  $R_2=1000\Omega$

4. **Photoresistor**

- a. Photoresistance values range from  $R_p=10\text{k}\Omega$  to  $100\text{k}\Omega$  based on amount of lux ( $\text{lumens} / \text{m}^2$ ) through the resistor and the fixed resistor in the circuit.



# Electrical System

Voltage Divider:

- Applying ohm's law allows us to convert analog voltage input from arduino into resistance of photoresistor

Sample Calculations:

$$V_{out} = \text{Analog} \cdot \frac{5}{1023}$$

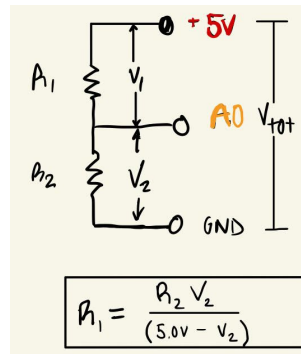
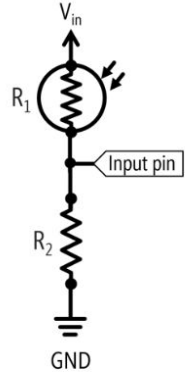
$$R_{\text{photoresistor}} = \frac{R_2 V_2}{5 - V_2} = \frac{(1000\Omega)(2.48)}{5 - 2.48} = 984\Omega$$

$$V_{LED} = \frac{90}{270} \cdot 5V = 1.67V$$

note: 100% current flow corresponds to a PWM of a=255

$$\text{Lux} = \text{digital output} \cdot 0.1243 = 255 \cdot 0.1243 = 32.7 \frac{\text{lm}}{\text{m}^2}$$

$$R_{\text{photoresistor}} = \frac{R_2 V_2}{5 - V_2} = \frac{(1000\Omega)(2.48)}{5 - 2.48} = 984\Omega$$



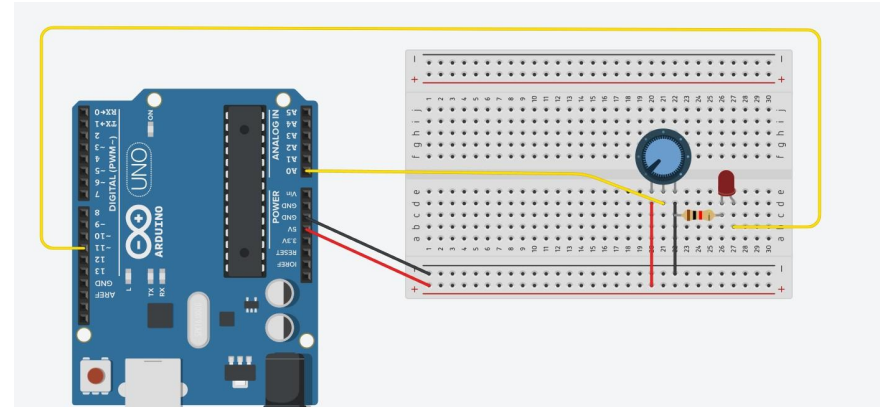
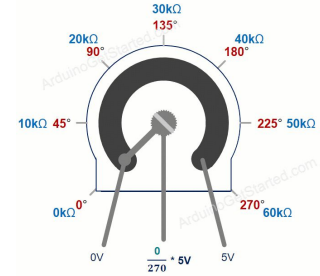


# LED-Photoresistor calibration

Because the laser operates on a digital signal, we were unable to modulate its voltage. The red LED is used as a reference because it is a reasonable representation of reflected laser light since it emits diffusely, corresponds to the same range of intensities as the reflected laser light, and is the same wavelength.

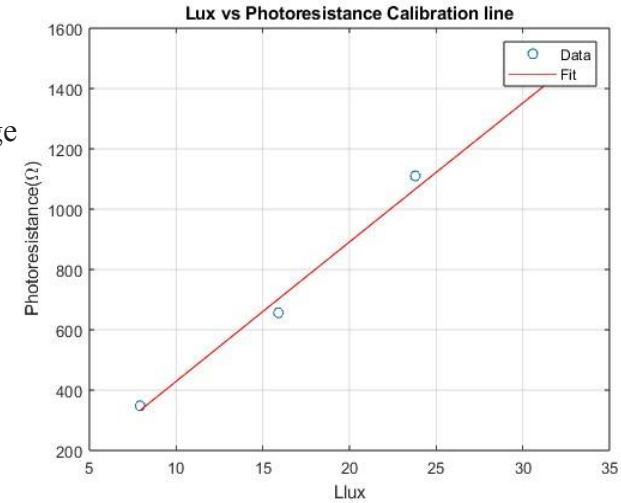
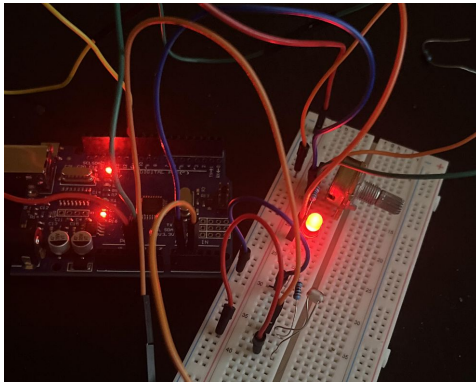
- Record light intensity data from the photoresistor at 100%, 75%, 50%, 25%, and 0% by adjusting the potentiometer.
- Based on the calibration equation

$$\text{lux} = (\text{digital output}) * 0.1243$$



## Calibration Fit line

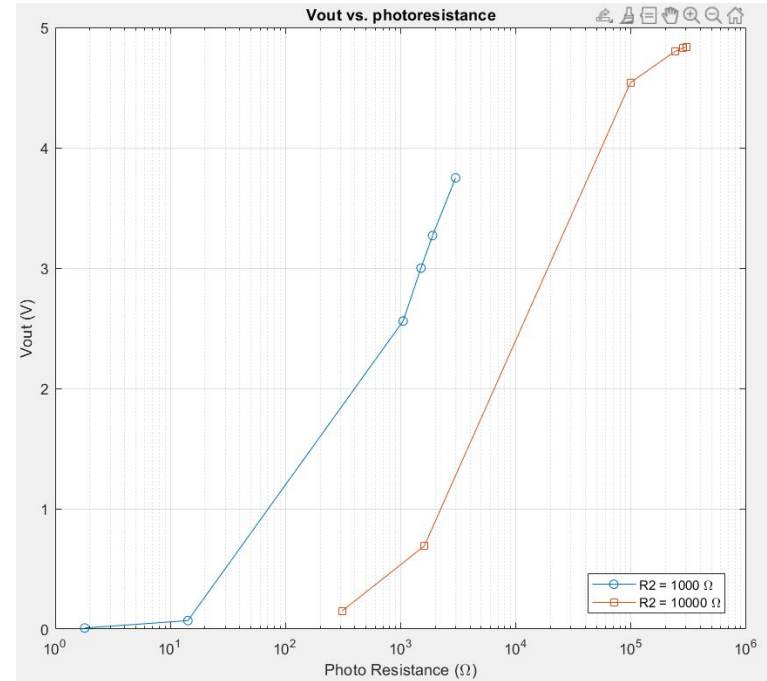
- The line is a linear fit with little deviation from the trend. This indicates that in the range of lux we are using there is a linear relationship between resistance and lux.
- Quick exposure to intense light causes the photocell to saturate and the readings will scale based on the environment the cell was calibrated in. This means the photocells should be calibrated with a known source which is close to the approximate light intensity of our laser.



| Potentiometer Flow | LED Voltage (V) | $V_{out}$ | Photoresistor Analog | $R_p$ | Illuminance (lux) |
|--------------------|-----------------|-----------|----------------------|-------|-------------------|
| 25%                | 1.25            | 1.29      | 263                  | 348   | 7.92              |
| 50%                | 1.67            | 1.98      | 403                  | 656   | 15.9              |
| 75%                | 3.75            | 2.63      | 538                  | 1110  | 23.8              |
| 100%               | 5               | 2.97      | 608                  | 1463  | 32.7              |

# Fixed Resistor Analysis

- Varied intensity of iPhone flashlight (50 lm at max power)
  - Rational to using iPhone is that it is a calibrated light source with strong tolerances allowing easy estimation of luminous intensity
- Plot output voltage vs. photoresistance on logarithmic scale
- When  $R_2 = 1000\Omega$ , there is a larger variation in voltage output at higher light intensities which gives more precise measurements
- Our entire voltage range falls into the expected resistance range of  $R_2 = 1000\Omega$  which is the best option for our system



# Data & Results

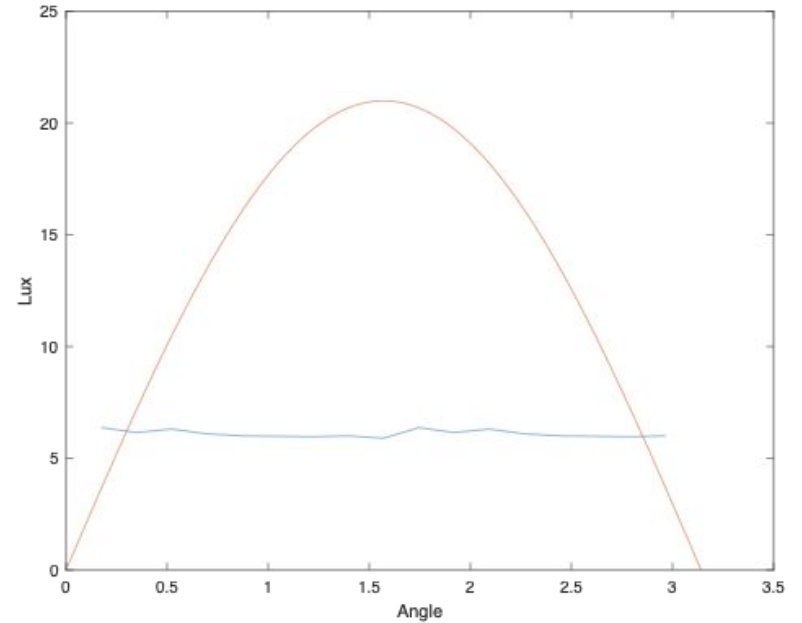
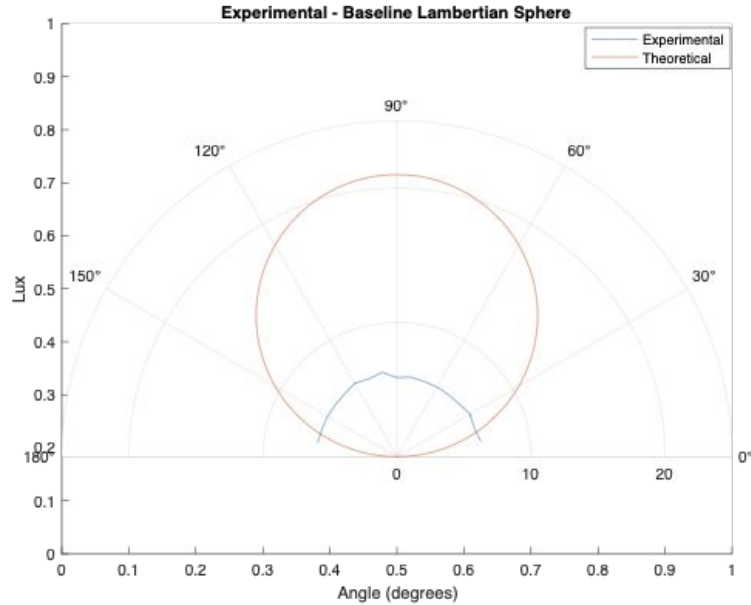
—

## Data Acquisition and Analysis

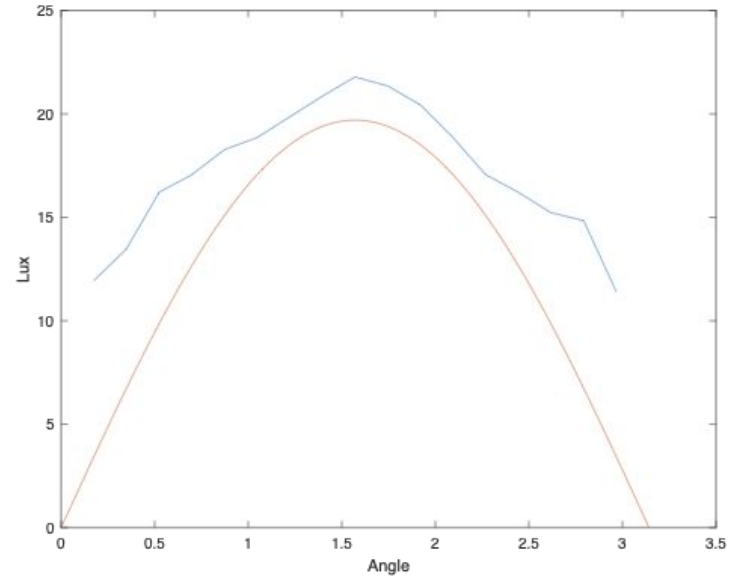
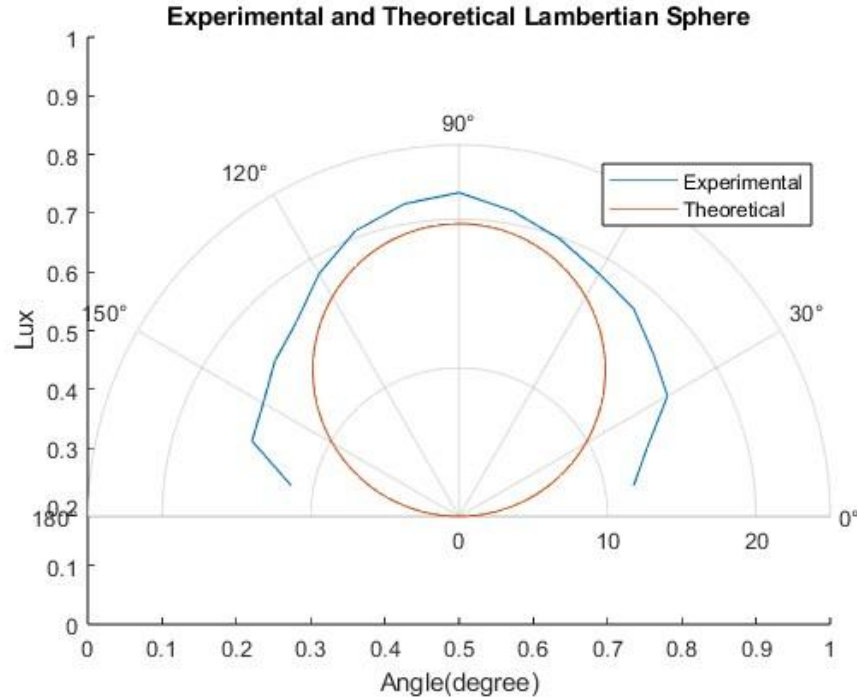


- Individual photocells perform differently due to manufacturing and other variations
- Photoresistor takes 50 measurements over 10 sections for the position the laser is at
- Manually move the laser to vary the incident angle
- Photoresistor remains in a constant location at  $45^\circ$  to the diffuse surface
- Photoresistor provides analog value
- Analog is then converted to digital which is then fed into the voltage divider equation to determine the resistance
- Using the established lux versus resistance fit curve the resistance was converted to lux
- This was then plotted against the angles the resistances were measured at to create the cosine graph and a polar representation against the unit circle that has been scaled by the maximum lux.

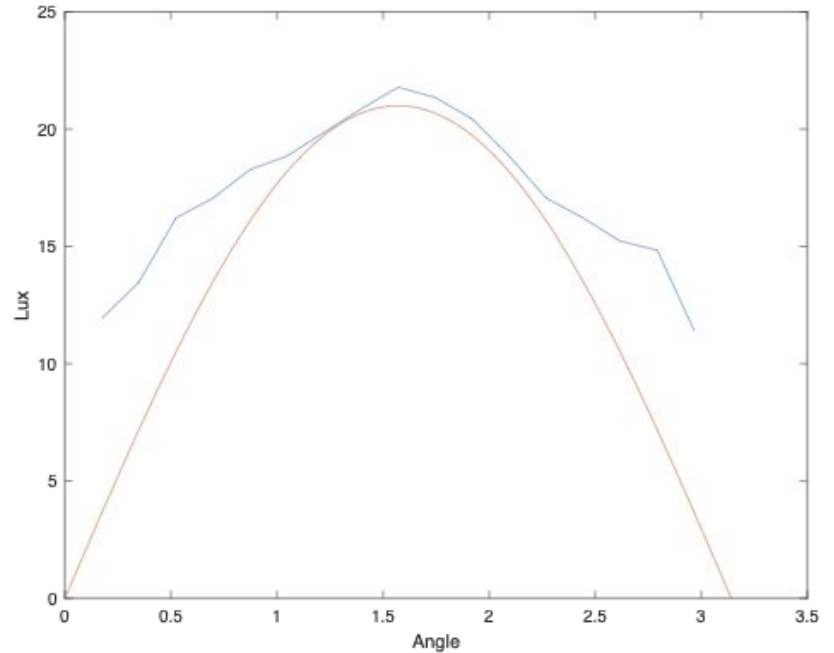
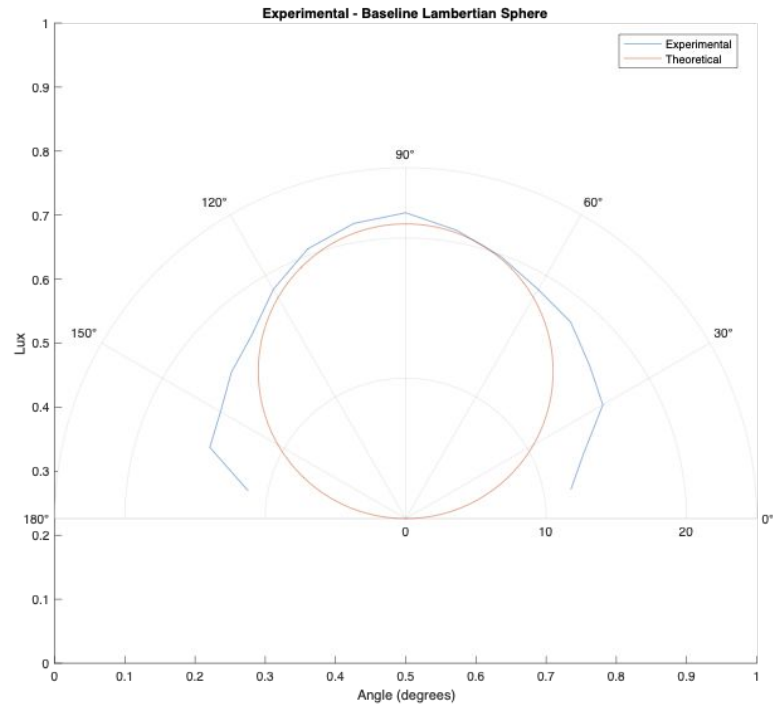
# Baseline Lambertian Sphere



# Experimental vs Theoretical Lambertian Sphere

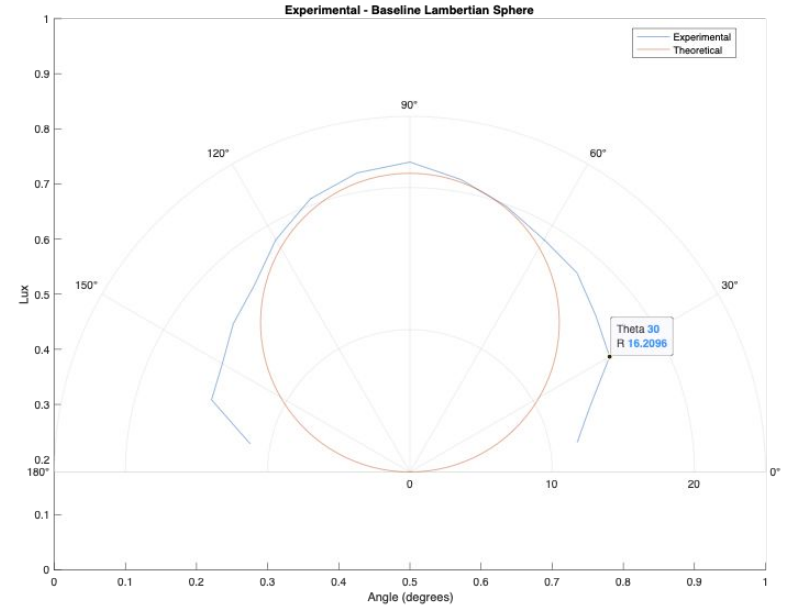
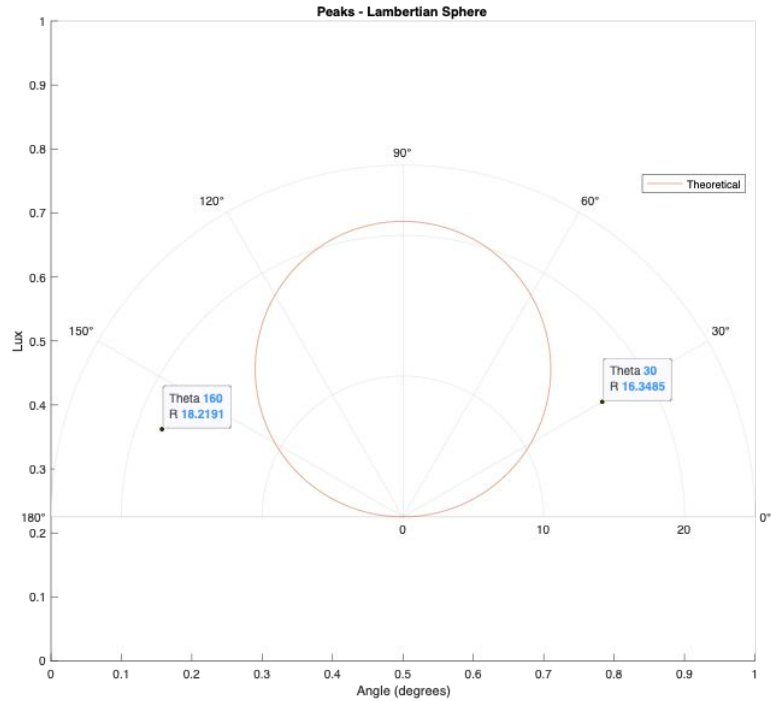


# Experimental - Baseline Lambertian Sphere





# Peaks - Lambertian Sphere



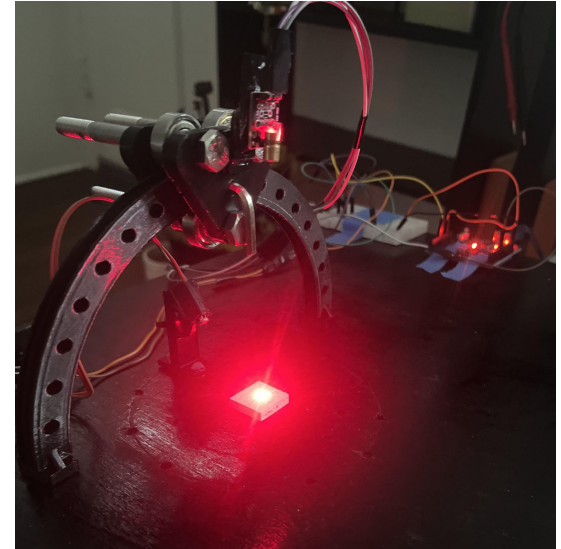
# Approximated Reflectivity

At  $\theta_s=0$  the photoresistor gives a resistance of  $880\Omega$  corresponding to 19.8 lux relative to our fit line which is an accurate value compared to our theoretical calculation of 21.7 lux.

It would be reasonable to assume the difference between the theoretical and experimental values is due to the absorption of the material thus

$$\therefore \rho = .91$$

which indicates that 9% of the light that reaches the sample is absorbed by the material



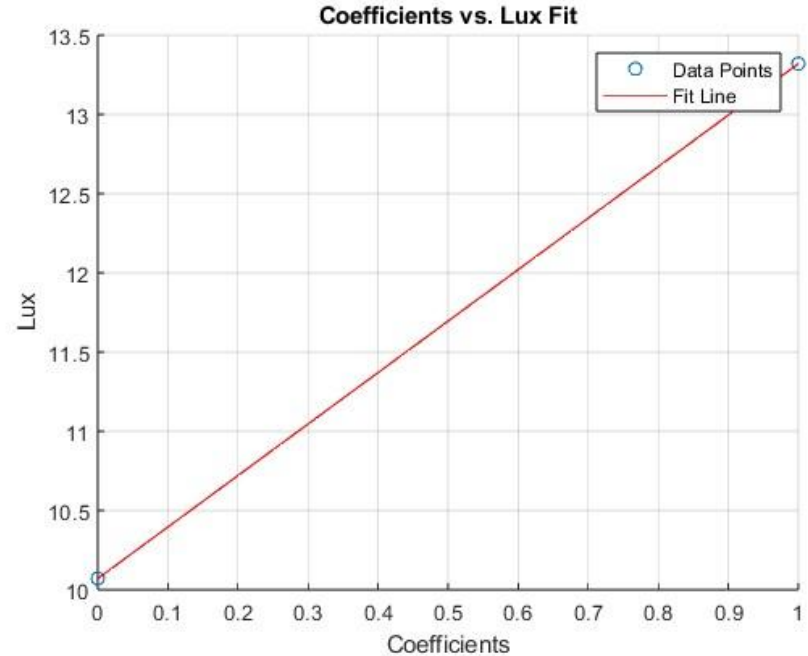
## Results - Lambert's cosine law and BRDF model

---

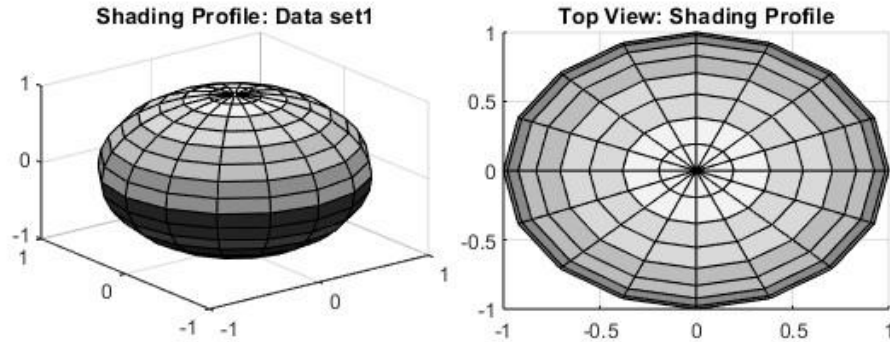
- Baseline reference of Lambertian Sphere with no sample present on laser (Plot 1)
  - Critical to understand the instrument signature and characterize the baseline light scattering (otherwise the calculated reflectivity coefficient is meaningless)
  - As desired this baseline is a small fraction of the experimental results
- Measured emission pattern of wood acrylic sample compared to Ideal Lambertian distribution (Plot 2)
  - The measured results of the
- Comparative analysis by subtracting the baseline results from the measured values (Plot 3)
  - The sample resembles the characteristic behavior of Lambertian reflectance.
  - As expected removing the baseline reference values from the true experimental results helped account for ambient reflection and other factors allowing the final model to be close to the theoretical shape
  - This subtraction helps account for electronic noise and potential positioning errors.
- Overall, saw that this simple set up was able to create a close Lambertian surface that due to surface imperfections still saw light scattering something that is notable in the additional peaks at  $160^\circ$  and  $30^\circ$ 
  - These reflection values changed significantly when the sample was rotated  $90^\circ$  further proving this point
- By being able to calculate the absorbance of light by the material we also showed how this surface is not a true Lambertian surface since it is showing the reality that not all light is being reflected let alone diffusely

# Contour Shading

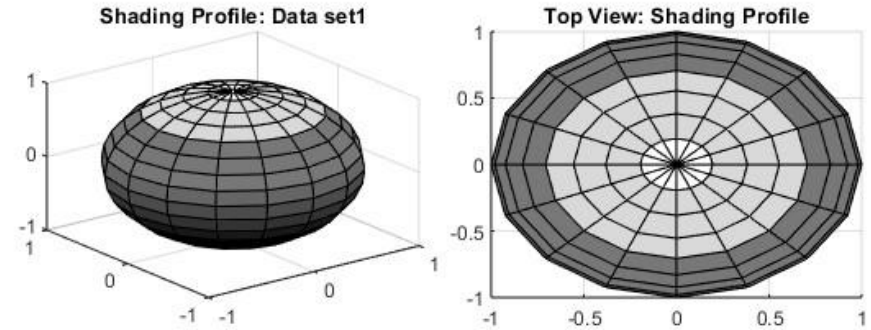
- The uncoated side of the sample is used to create variance in the shading profile
- $\theta_i$  is constant at zero and for the first data set  $\theta_r$  is varied in increments of 10 degrees until it reaches 90. In the second data set we used increments of 30 degrees
- Illuminance measurements are correlated to brightness to establish a basis for the shading profile
- The maximum recorded lux is given a coefficient of 1 and the minimum corresponds to zero. All other coefficients are approximated based on the trend



### Resolution produced by 8 pixels:



### Resolution produced by 4 pixels:



- The shading produced with 8 data points (pixels) is a higher resolution than the sphere with only 4 data points
- Pixels are like bins that store information about the color and reflectance of the sphere, the larger the sampling rate the more bins fit into a window which describes an area on the sphere.
- A lower sampling rate approximates multiple colors as one
- Shading is what produces a 3D profile in images
- Precise measurement of reflected light from various orientations is a crucial part of producing hyper realistic rendering models

# Error Analysis

—

# Error Analysis



- Arduino's ADC conversion
  - The precision depends on characteristics such as resolution and linearity
- Fixed resistors
  - $R2 = 1000\Omega$  gives a tolerance of  $\pm 2\%$  which effects the sensitivity of the photosensor and scales the resulting photoresistance.
  - Regular calibration minimizes discrepancies
- Assumptions
  - 100% flux through sensor, light evenly distributed, 100% reflectance, lambertian surface
  - The calculated absorbance demonstrates that our sample is not a lambertian surface
  - The laser is likely not perfect so the estimated light intensity at the surface is likely an over approximation.
- Sample surface irregularities cause light to reflect in an unpredictable manner
- Photoresistor
  - These sensors are more commonly used to detect large changes in lux (on or off) rather than precise measurements
  - Additionally we noted that these sensors would often become saturated with ambient light and then give a systematic error compared to prior measurements take
  - In general photoresistors are most sensitive to light differences at lower lux levels (i.e., in the black box)
- System Tolerances
  - 3D printed jigs have dimensional tolerances that can become a slight factor in the calculations based on radius
  - Similarly the hole locations can have deviations from what we assumed their angular values to be
  - Finally the is the fact that the laser is not perfectly normal to the surface and can have an angular tilt despite best efforts
  - Laser then may not be fully pointed at the center of the annulus

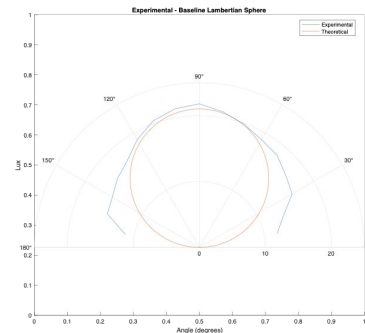
# Conclusion

---

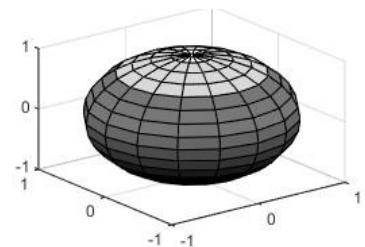


# Conclusion

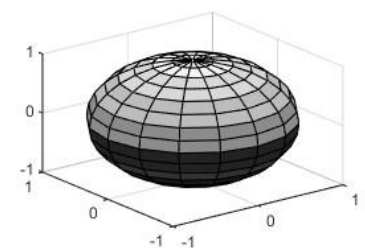
- Lambert's law describes diffuse reflectance behavior from an ideal surface and the corresponding BRDF model gives a basis for determining the reflectivity of a material
- Our apparatus allows us to record illuminate at various incident and viewing angles which gives a more precise approximation of the sample materials reflectance behavior
- The approximated reflectivity is only meaningful when a baseline is established. By removing the baseline from the results our data fits more closely to the ideal Lambertian trend
- By changing the orientation of the surface and remeasuring the reflectance when the incident light is at  $160^\circ$  and  $30^\circ$  (orientation of the two peak measurement aside from  $90^\circ$ ) we showed the measured illumination varied appreciably thus confirming that the in our data is attributable to sample surface imperfections
- Lastly we showed the nature of reflectance on a non diffuse surface and how sampling rates can then affect the contour produced in rendered images



4 pixels



8 pixels



# Appendix

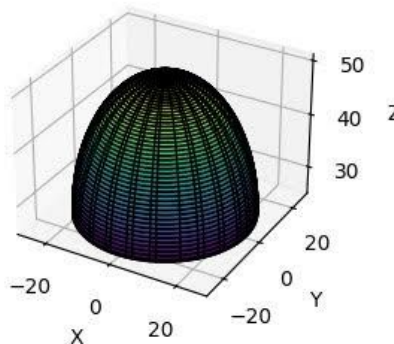
---

# Theory: Rendering Equation

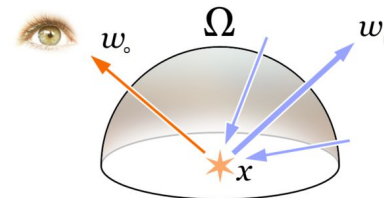
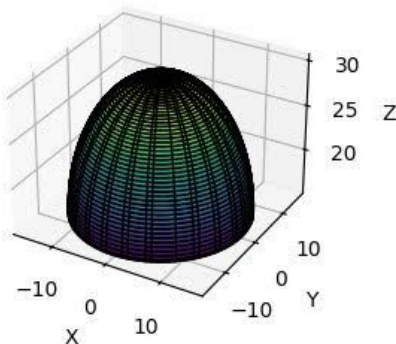
$$L(\omega_r) = \int_{\Omega} f_r(\omega_i, \omega_r) L_i(\omega_i) \cos(\theta) d\omega_i$$

Rendered diffuse hemisphere relative to our systems constraints. (Axis given in lux)

Diffusion at max power



Diffusion at 60% Power



# Reference Research:



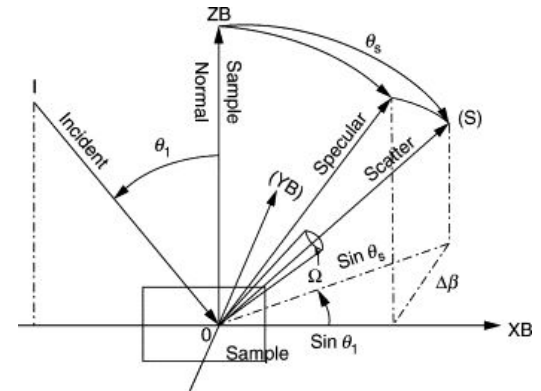
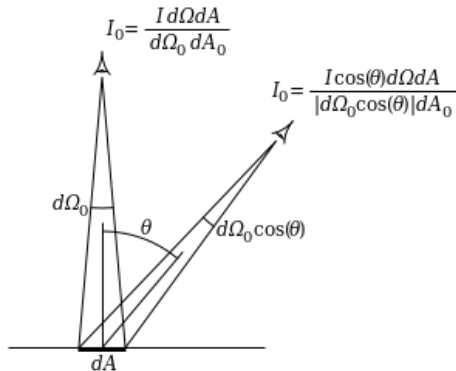
1. [Measuring and modeling Anisotropic reflection - UCSD](#)
2. [Efficient BRDF Importance Sampling Using A Factored Representation - Princeton University](#)
3. [Reflection Models - Stanford University](#)
4. Bidirectional reflectance distribution function expressed in terms of surface scattering modes - European conference of computer vision
5. [Notes on the Ward BRDF : Cornell program of computer graphics](#)
6. [A Survey of BRDF Representation for Computer Graphics : Princeton University](#)
7. [Determining shape and reflectance of Lambertian, Specular, and Hybrid Surfaces using extended surfaces : Internation workshop of industrial applications of computer vision, Robotics Institute, Carnegie Mellon University](#)
8. [Experimental analysis of BRDF models : MIT](#)
9. [Lambertian Model](#)
10. [Reflectance of rough surfaces : First principles of computer vision](#)
11. [Crash course in BRDF implementation : MIT](#)
12. [Bidirectional reflectance distribution function expressed in terms of surface scattering modes](#)
13. [Measuring Anisotropic reflection : Lighting systems research group, Berkeley Laboratory](#)
14. [Physically-Based Rendering Cook-Torrance Reflectance Model](#)
15. [Measuring Luminance with a digital camera](#)
16. [BRDF - UCSD](#)
17. [Analytic Methods for Simulated Light Transport - Yale University](#)

# (More) Theory: Generalized Bidirectional Reflectance Model (BRDF)

The generalized BRDF, denoted  $f_r(\omega_i, \omega_r)$ , is a function which describes the behavior of light reflected off a surface. It takes two directions as inputs, composed of four independent angles, two of them in the interval  $(0, 90)$  and the other pair in the periodic range  $(0, 360)$ .

Intuitively the BRDF represents, for each incident angle, the amount of light that is scattered at each outgoing angle. Dimensions can be added and dropped from the BRDF to better represent a specific system and increase the accuracy of the measurement.

$$f(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{L(\theta_r, \phi_r)}{E(\theta_i, \phi_i)} \left( \frac{1}{sr} \right)$$



# Rendered Sphere Data

Obtained by rotating photoresistor position ( $\theta_r$ ) along the hoop and keep  $\theta_s$  constant

| $\theta_r (rad)$ | Analog | $R_1$ | $lux (\frac{lm}{m^2})$ | Coefficient |
|------------------|--------|-------|------------------------|-------------|
| 0                | 377    | 582   | 13.34                  | 1           |
| 10               | 374    | 577   | 13.23                  | 0.95        |
| 20               | 366    | 558   | 12.83                  | 0.85        |
| 30               | 360    | 543   | 12.51                  | 0.74        |
| 40               | 344    | 506   | 11.71                  | 0.55        |
| 50               | 323    | 462   | 10.77                  | 0.19        |
| 60               | 319    | 454   | 10.60                  | 0.13        |
| 70               | 318    | 449   | 10.50                  | 0.14        |
| 80               | 307    | 429   | 10.07                  | 0           |

Data: Contour 1

| $\theta_r (rad)$ | Analog | $R_1$ | $lux (\frac{lm}{m^2})$ | Coefficient |
|------------------|--------|-------|------------------------|-------------|
| 0                | 390    | 618   | 14.11                  | 1           |
| 30               | 378    | 587   | 13.45                  | .85         |
| 60               | 346    | 511   | 11.82                  | .47         |
| 90               | 311    | 437   | 10.24                  | 0           |

Data: Contour 2

## Derivation of Lambert BDRF:

$$f_{\text{Lambertian}}(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{\rho}{\pi}$$

The Albedo,  $\rho(x)$ , (also called *Bi-hemispherical reflectance*) at point  $x$  is the fraction of light that a surface reflects or ratio of total outgoing power to total incident power at the surface interface. If all light is all reflected, the albedo is equal to 1. It can be used to denote how bright a surface is (not to be confused with how reflective a surface is). Real Time Rendering 4th Edition defines albedo as the *hemispherical-directional reflectance* which measures the amount of light reflected along a given direction for incoming light in any direction within the hemisphere around the surface normal. It can be represented as:

$$\rho = \int_{\Omega} f_r(x, \omega_i, \omega_o)(\omega_i \cdot N) d\omega_i$$

where

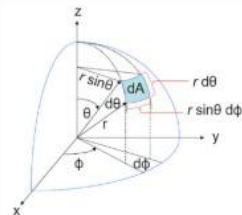
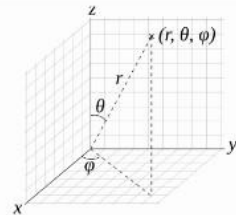
$f_r$  = BRDF at point  $x$  for incoming radiance along  $\omega_i$  and outgoing radiance along  $\omega_o$

$(\omega_i \cdot N)$  = weakening factor for incoming direction  $\omega_i$  and normal  $N$

$\Omega$  is used to denote integration over hemisphere

Since the Lambertian BRDF is a model of diffuse reflectance it is invariant to the viewing direction. In other words, it is constant and can be taken out of the integral.

$$\rho = f_{\text{Lambert}} \int_{\Omega} (\omega_i \cdot N) d\omega_i \rightarrow f_{\text{Lambert}} = \frac{\rho}{\int_{\Omega} (\omega_i \cdot N) d\omega_i}$$



For  $\int_{\Omega} (\omega_i \cdot N) d\omega_i$  we integrate a small area with length  $dr$  and arc length  $r d\theta$  in polar coordinates hence the length of the arc is  $r \sin\theta d\theta$ . Since the direction vector and normal are assumed to be unit length and the weakening factor is invariant to the azimuth angle, we can write

$\int_{\Omega} (\omega_i \cdot N) d\omega_i$  as

$$\int_{\Omega} \cos\theta d\omega_i = \int_{\theta=0}^{\frac{\pi}{2}} \int_{\phi=0}^{2\pi} \cos\theta \cdot r d\theta \cdot r \sin\theta d\phi = r^2 \int_{\theta=0}^{\frac{\pi}{2}} \sin\theta \cos\theta d\theta \int_{\phi=0}^{2\pi} d\phi$$

applying the trigonometric identity  $\sin(2\theta) = 2\sin(\theta)\cos(\theta)$  and substituting  $r = 1$  for a unit hemisphere we can rewrite this as

## Lambertian BRDF derivation contd...

$$\int_{\theta=0}^{\frac{\pi}{2}} \frac{\sin 2\theta}{2} d\theta \int_{\phi=0}^{2\pi} d\phi$$

Substituting  $\Theta = 2\theta$  such that  $\frac{d\Theta}{d\theta} = 2$

$$\int_{\Theta=0}^{\pi} \frac{\sin \Theta}{2} \frac{d\Theta}{2} \int_{\phi=0}^{2\pi} d\phi = \frac{1}{4} \left[ -\cos \Theta \right]_{\Theta=0}^{\pi} \left[ \phi \right]_0^{2\pi} = \frac{1}{4} \times 2 \times 2\pi$$

$$\int_{\Omega} \cos \theta d\omega_i = \pi$$

$$\therefore f_{\text{lambert}} = \frac{\text{albedo}}{\pi}$$



# Code

---

## Rendering Equation (Python Code):

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def lambertian_reflectance(theta):
    """
    Calculate Lambertian reflectance for a given angle.

    Parameters:
    - theta: Angle in radians.

    Returns:
    - Reflectance at the given angle.
    """
    return np.cos(theta) / np.pi

# Given parameters
original_power = 1000 # Laser power in mW
wavelength = 650e-9 # Wavelength in meters (650 nm)
incident_radiance = original_power / (2 * np.pi) # Assuming power is in mW/sr for the upper hemisphere

# Assuming 100% diffusive reflectance (Lambertian model)
reflectance = 1.0

# Set up the hemispherical geometry
theta_vals = np.linspace(0, np.pi/2, 100) # Limiting to the upper hemisphere
phi_vals = np.linspace(0, 2 * np.pi, 100)
```

```
# Create a meshgrid for theta and phi
theta, phi = np.meshgrid(theta_vals, phi_vals)
```

```
# Calculate the radiance for each point on the hemisphere
radiance = incident_radiance * reflectance *
lambertian_reflectance(theta)
```

```
# Convert spherical coordinates to Cartesian
coordinates
x = radiance * np.sin(theta) * np.cos(phi)
y = radiance * np.sin(theta) * np.sin(phi)
z = radiance * np.cos(theta)
```

```
# Calculate and print the radius of the original hemisphere
radius = np.max(np.sqrt(x**2 + y**2 + z**2))
print(f"Radius of the original hemisphere: {radius}")
```

```
# Create a 3D plot for the original hemisphere
fig = plt.figure()
ax = fig.add_subplot(121, projection='3d')
ax.plot_surface(x, y, z, cmap='viridis',
edgecolor='k')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_zlabel('Z')
ax.set_title('Original Hemisphere')
```

```
# Calculate and visualize a second hemisphere with 60% of the power
reduced_power = 0.6 * original_power
incident_radiance_reduced = reduced_power / (2 * np.pi)
radiance_reduced = incident_radiance_reduced * reflectance *
lambertian_reflectance(theta)
x_reduced = radiance_reduced * np.sin(theta) * np.cos(phi)
y_reduced = radiance_reduced * np.sin(theta) * np.sin(phi)
z_reduced = radiance_reduced * np.cos(theta)
```

```
# Calculate and print the radius of the reduced-power hemisphere
radius_reduced = np.max(np.sqrt(x_reduced**2 + y_reduced**2 +
z_reduced**2))
print(f"Radius of the reduced-power hemisphere: {radius_reduced}")
```

```
# Print the reduction in radius
radius_reduction = radius - radius_reduced
print(f"Reduction in radius: {radius_reduction}")
```

```
# Create a 3D plot for the reduced-power hemisphere
ax = fig.add_subplot(122, projection='3d')
ax.plot_surface(x_reduced, y_reduced, z_reduced, cmap='viridis', edgecolor='k')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Reduced-Power Hemisphere')
```

```
# Show the plots
plt.show()
```

## Photoresistor data collection program:

(Arduino IDE)



```
int V2;
int Vout;
int totalReadings = 0; // Counter
for the total number of readings
const int laserPin = 7;
const int photoResistorPin = A0;
const double R2 = 1000;
const float referenceVoltage = 5;
unsigned long startTime;
float sumReadings = 0.0; //
Variable to accumulate the sum of
all readings
float firstReading = 0.0; //
Variable to store the first reading
```

```
void setup() {
    pinMode(laserPin, OUTPUT);
    Serial.begin(9600);
    digitalWrite(laserPin, LOW);
    startTime = millis(); // Record the start time
}

void loop() {
    digitalWrite(laserPin, HIGH);
    delay(100);
    Vout = analogRead(photoResistorPin);
    float V2 = Vout * (5.0 / 1023.0);
    float photoResistance = (R2 * V2) / (5.0 - V2);

    //float photoResistance = (5/(V2-1))*R2;
    Serial.println(photoResistance);

    // Increment the total readings count
    totalReadings++;
}
```

# Photoresistor data collection program contd..



```
// Accumulate the sum of all readings
sumReadings += photoResistance;

// Store the first reading
if (totalReadings == 1) {
    firstReading = photoResistance;
}

// Check if 10 seconds have passed
if (millis() - startTime >= 10000) {
    Serial.println("-----");
    Serial.println("t = 10 s");
    Serial.print("n_tot: ");
    Serial.println(totalReadings);

    // Calculate and print the average of all readings
    float average = sumReadings / totalReadings;
    Serial.print("AVG: ");
    Serial.println(average);

    // Calculate and print the slope of the line
    float slope = (photoResistance - firstReading) /
totalReadings;
    Serial.print("Slope: ");
    Serial.println(slope);


    while (true) {
        // Loop indefinitely, you can add any cleanup code here if
needed
    }
}
delay(100);
}
```

# Lux to Resistance Calibration

## Arduino IDE


```
% Given data
photoresistor = [348; 656; 1110; 1463];
flux = [7.92; 15.9; 23.8; 32.7];
% Fit a polynomial of degree 2 to the data
degree = 2;
fitCoefficients = polyfit(flux, photoresistor, degree);
% Generate fitted values based on the fit curve
fitValues = polyval(fitCoefficients, flux);
% Calculate the slope of the fitted curve
derivativeCoefficients = polyder(fitCoefficients);
slope = polyval(derivativeCoefficients, flux);
% Approximate lux at a resistance of 880
targetResistance = 429;
targetFlux = polyval(polyfit(photoresistor, flux, degree), targetResistance);
% Print the approximate lux value
fprintf('Approximate lux at resistance %d: %.2f\n', targetResistance, targetFlux);
% Plot the data, the fit curve, the slope, and the target point
figure;
plot(flux, photoresistor, 'o', 'DisplayName', 'Data Points');
hold on;
plot(flux, fitValues, 'r-', 'DisplayName', 'Fit Curve');
plot(flux, slope, 'g-', 'DisplayName', 'Slope');
plot(targetFlux, targetResistance, 'bx', 'DisplayName', 'Target Point');
hold off;
xlabel('Flux');
ylabel('Photoresistor Resistance');
title('Polynomial Fit, Slope, and Target Point for Photoresistor Resistance vs Flux');
legend('show');
grid on;
```

## Coefficient Scaling MatLab Code: (for sphere rendering)



```
% Given data points
lux_values = [13.32, 10.07];
coefficients = [1, 0];
% Perform linear regression (fit a line)
fit_coefficients = polyfit(coefficients, lux_values, 1);
% Generate fitted values based on the fit line
coefficients_range = linspace(min(coefficients), max(coefficients), 100);
fit_line = polyval(fit_coefficients, coefficients_range);
% Plot the data points and the fit line
figure;
scatter(coefficients, lux_values, 'o', 'DisplayName', 'Data Points');
hold on;
plot(coefficients_range, fit_line, 'r-', 'DisplayName', 'Fit Line');
hold off;
xlabel('Coefficients');
ylabel('Lux');
title('Coefficients vs. Lux Fit');
legend('show');
grid on;
```


## Coefficient Scaling Code contd..



```
% Given data points
lux_values = [13.32, 10.07];
coefficients = [1, 0];
% Perform linear regression (fit a line)
fit_coefficients = polyfit(coefficients, lux_values, 1);
% Generate fitted values based on the fit line
coefficients_range = linspace(min(coefficients), max(coefficients), 100);
fit_line = polyval(fit_coefficients, coefficients_range);
% Plot the data points and the fit line
figure;
scatter(coefficients, lux_values, 'o', 'DisplayName', 'Data Points');
hold on;
plot(coefficients_range, fit_line, 'r-', 'DisplayName', 'Fit Line');
hold off;
xlabel('Coefficients');
ylabel('Lux');
title('Coefficients vs. Lux Fit');
legend('show');
grid on;
```



## Sphere Generator MatLab Code:



```
% Create a 2x2 tiled layout
t = tiledlayout(2, 2);
% Manually adjust the shading of each ring in tile 1
nexttile;
[x, y, z] = sphere(16);
% Initialize shading values
shading_values = zeros(size(z));
% Manually input shading values for each ring
for i = 1:size(z, 1)
    prompt = ['Enter shading value for Ring ' num2str(i) ' (between 0 and 1): '];
    shading_values(i, :) = input(prompt);
end
% Plot the sphere with custom shading
s = surf(x, y, z, shading_values);
title('Shading Profile: Data set1');
% Create a flat sphere with the same shading profile in tile 2
nexttile;
flat_sphere = surf(x, y, z, shading_values);
view(0, 90); % Set the view to make it flat
title('Top View: Shading Profile');
% Create an interpolated sphere with the same shading profile in tile 3
nexttile;
interp_sphere = surf(x, y, z, 'FaceAlpha', 1, 'FaceColor', 'interp');
shading interp;
title('Interpolated Sphere');
% Set the colormap to black and white
colormap(gray);
% Adjust layout
t.TileSpacing = 'compact';
t.Padding = 'compact';
```

## Ideal Lambertian Sphere vs Experimental Data Polar Plot (Matlab)

```
clc;close all; clear;
r2 = 1000;
angles=(pi/180)*([-80,-70,-60,-50,-40,-30,-20,-10,0,10,20,30,40,50,60,70,80]+90);
v2=(5/1023)*[226,268,333,351,375,386,404,421,436,429,414,386,351,333,311,302,210];
r1 = r2*v2./(5-v2);
resistance =
[880,852,825,805,767,761,748,634,493,838,825,779,757,742,672,623,543].';
lux
=[19.74,19.14,18.56,18.13,17.31,17.18,16.90,14.45,11.44,18.83,18.56,17.57,17.09,16.77, 15.27,14.22, 12.51].';
coeff = polyfit(resistance,lux,1);
figure(3)
plot(lux,resistance);
hold on
lux = r2lux(r1,coeff)+5;
y = coeff(1)*resistance+coeff(2);
plot(y,resistance);
angletheory = (pi/180)*linspace(0,180,100);
theory = 19.7*sin(angletheory);
figure(1)
title('Experimental and Theoretical Lambertian Sphere' )
xlabel('Angle(degree)' )
ylabel('Lux')
ax = polaraxes;
polarplot(angles,lux);
hold on
lux_at_sphere_angle = r2lux(r2, coeff) + 5;
```

```
polarplot(angletheory,theory);
thetalim(ax,[0 180])
legend('Experimental','Theoretical','Location','northeast','Orientation','vertical');
figure(2)
plot(angles, lux,angletheory, theory);
xlabel('Angle')
ylabel('Lux')
function lux =
r2lux(resistance,coeff)
    lux =
(coeff(1)*resistance)+coeff(2);
end
```