Design, Construction, and Analysis of a Solid State Nearly-Isotropic Light Source

by

Paul Kovacs
B.Sc., Brock University, 2013

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Physics and Astronomy

Design, Construction, and Analysis of a Solid State Nearly-Isotropic Light Source

by

Paul Kovacs
B.Sc., Brock University, 2013

Supervisory Committee

Dr. J. Albert, Supervisor
(Department of Physics and Astronomy)

Dr. R. Sobie, Departmental Member
(Department of Physics and Astronomy)

Dr. S. Briggs, Outside Member
(Department of Chemistry)

ABSTRACT

Isotropic radiators are known to be a useful tool across a wide range of applications, from applications in light dosimetry in human/animal tissue to calibration of sensitive laboratory equipment. While the benefits are known, constructing such a tool has proven to be difficult. Currently, there are no commercially available isotropic or nearly isotropic radiators. Previous attempts at constructing an isotropic radiator have been limited to a $\pm 10\%$ isotropy level. This thesis covers the design, construction, and analysis of a *nearly* isotropic optical light source. The constructed source has extrema anisotropies, over a solid angle of approximately 3π steradians, of $^{+3\%}_{-3\%}$ (with 95% of that solid angle being within $\pm 2.5\%$ anisotropy).

Contents

List of Tables

List of Figures

Abbreviations

ALTAIR	Airborne Laser for Telescopic Atmospheric Interference Reduction
A	Ampere
API	Application Program Interface
CAD	Canadian Dollar
CI	Confidence Interval
DC	Direct Current
FLRW	Friedmann-Lemaître-Robertson-Walker
LED	Light Emitting Diode
NIST	National Institute of Standards and Technology
sr	Steradian
USB	Universal Serial Bus
V	Volt
W	Watt

List of Symbols

M	Absolute Magnitude
A	Active Area of the Photodiode
m, m_1, m_2	Apparent Magnitude
C	Conversion Impedance of the Photosensor Amplifier (V/A)
d	Distance from Photodiode to Diffusing Sphere (m)
ρ_i	Energy Density of Component i (of the Universe)
ω_i	Equation of State Parameter of Component i (of the Universe)
a	Expansion Factor of the Universe
f, f_1, f_2	Flux (W/m^2)
f_i	Flux Contribution of LED i
G	Gravitational Constant
H	Hubble Parameter
d_L	Luminosity Distance
R	Photosensitivity of the Photodiode (A/W)
P_i	Pressure of Component i (of the Universe)
P_{pd}	Power Measured by the Photodiode
I	Radiant Intensity (W/sr)
ϕ	Radiated Flux (W/m^2)
$L_\odot$	Solar Luminosity ($\sim 3.846 \cdot 10^{26}$ W)
$M_\odot$	Solar Mass ($\sim 1.989 \cdot 10^{30}$) kg

Ω	Solid Angle (sr)
K	Spatial Curvature of the Universe
L	Total Luminosity (W)
V_{pd}	Voltage Output From Photosensor Amplifier (V)

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Justin Albert, my supervisor who has provided an abundance of guidance and support throughout this thesis.

Chris Secord and Jeff Trafton, for their help in designing and constructing the measurement apparatus.

Nicolas Braam, for his help getting the stepper motors working to automate measurements, and general guidance with electronics throughout the thesis.

DEDICATION

I would like to dedicate this thesis to:

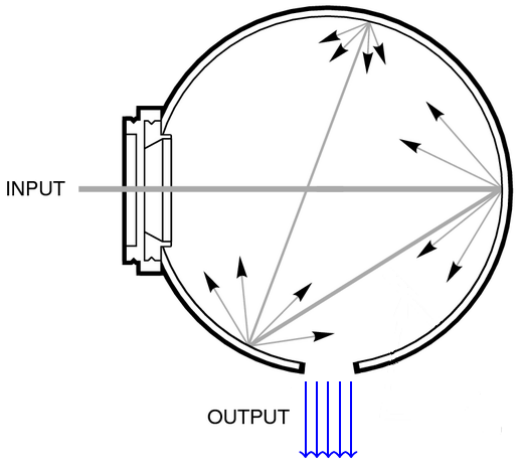
my parents, their love and support have been paramount to my success.

Chapter 1

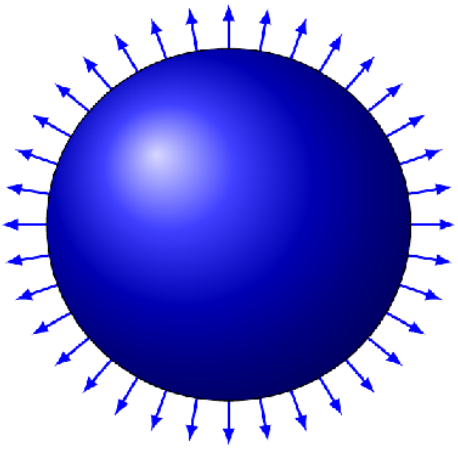
Introduction

An estimate of the global market for optical integrating spheres is of the order $100 million CAD per annum [1]. Integrating spheres are optical instruments which spatially integrate radiant flux [2, 3], as shown in Fig. 1.1(a). An integrating sphere consists of a hollow spherical cavity that is covered with a white diffuse reflective coating on the interior. Small holes in the spherical cavity are used for entry/exit ports. Light through an entry port to the interior surface is uniformly distributed by multiple scattering over the Lambertian surface coating [4]. While the largest application of integrating spheres is to measure the total reflectance or transmittance from scattering materials, they are also widely used to calibrate electronic imaging devices [5]. An isotropic or *nearly* isotropic light source — if such a device were commercially available and also as robust as a typical integrating sphere — could perform many of the tasks for which integrating spheres are currently used. Thus, such a light source could potentially obtain a significant fraction of the annual market for integrating spheres. A nearly isotropic light source could also provide additional techniques for optical, radiometric, photometric and spectroscopic calibrations.

An isotropic light source is an isotropic radiator which radiates in a part of the electromagnetic spectrum. A perfect isotropic radiator would emit equal amounts of radiant intensity across a spherical shell of 4π steradian around a point (the source), as shown in Fig. 1.1(b). A *nearly* isotropic light source emits radiant intensity which is nearly equal around a subset of 4π steradians of the source. This thesis intends to provide a detailed explanation of the design, construction, and analysis of a nearly isotropic light source. To characterize the isotropy level of the constructed light source, a measurement apparatus was constructed to move a photodiode around a spherical shell. The photodiode (a detailed description provided in section 3.1.4) was



(a) A model cross-section of an integrating sphere showing the internal reflections, and the spatially integrated output light (blue arrows).



(b) A model of an isotropic light source showing the light distribution in the plane of the paper around the source (blue arrows).

Figure 1.1: Model integrating sphere and isotropic light source diagrams.

used to measure the radiant intensity (I) defined as [6]:

$$I = \frac{\partial \phi}{\partial \Omega}, \tag{1.1}$$

where ϕ is the radiated flux emitted by the source and Ω is the solid angle. The radiant intensity was measured in units of watt per steradian, and was measured at multiple regular intervals around the source using the photodiode. The photodiode measured the irradiance via the produced photocurrent. The current was determined using conversion factors provided by a photosensor amplifier.

Some of the many potential applications of an isotropic light source include: light dosimetry in human and animal tissue [7], characterizing optical properties of oceans and lakes [8, 9], facial recognition [10], and photometric calibration of ground-based telescopes and camera optics [11, 12].

While the benefits of using an isotropic source are known, construction of such a source has proven to be difficult. Several previous attempts to construct an isotropic light source involved using an initial source (strobe/fiber) at the base of a Lambertian diffusing sphere [13, 14]. Another method used was to mount a light emitting diode (LED) on each surface of a cube and cover each LED with a spherical diffusing cap, then to mount the cube itself inside a sand-blasted glass sphere to provide extra

diffusion [8]. These methods have been used in laboratory settings, and are only isotropic to $\pm 10\%$, or worse [8, 13, 14]. There is room for major improvement for precise and reliable optical calibration.

This thesis describes the construction and study of a nearly isotropic solid-state light source. The source itself is a candidate calibrated source for a project aimed to determine the optical transmission properties of the atmosphere, known as ALTAIR, and which is outlined in the next chapter [15]. The following chapters will describe the construction details, measurement uncertainty, and difficulties of a near-isotropic light. The outlined optical source is isotropic to $O(4\%)$ or better over $\frac{3}{4}$ of the total solid angle of a sphere.

Chapter 2

The ALTAIR Project

The **A**irborne **L**aser for **T**elescopic **A**tmospheric **I**nterference **R**eduction (ALTAIR) project aims to reduce photometric uncertainty due to the atmosphere in current and future optical and near-infrared telescope surveys. The project will provide a precision photometric reference calibration of $O(0.1\%)$ uncertainty above the Earth's atmosphere using in-situ calibrated light sources. This chapter will cover the basics of how photometric surveys are executed, dark energy, and how ALTAIR can help improve the precision on dark energy parameters.

2.1 Standard Candles

There exist many different methods to measure cosmic distances across the Universe. A typical method is to use "standard candles." A standard candle is an astrophysical object with an intrinsic luminosity (L, in watts). If such an object exists, one can compute its luminosity distance (d_L) by the flux (f, in watts per m^2) measured from the object, and using an inverse square law about its flux [16]:

$$d_L = \sqrt{\frac{L}{4\pi f}}, \tag{2.1}$$

assuming there is zero light absorption in the path between object and observer.

Outlined are two types of astrophysical sources which can approximate standard candles. The first class of objects is known as Cepheid variable stars, or Cepheids. The second class of objects is known as Type 1a supernovae.

Cepheids are supergiant stars which have a mean luminosity $\sim 400 - 40000 L_\odot$. At

first glance, Cepheids do not appear to have a standard luminosity (with a variation of 3 orders of magnitude) to act as a standard candle. However, Cepheids are unstable stars which pulsate due to their instability. The pulsation periods also have a large variation $1.5 - 60$ days. An American astronomer, Henrietta Leavitt, identified a relationship between the luminosity and period of these variable stars [17]. Using the period-luminosity relationship, Cepheid variable stars can be used as standard candles. One underlying issue with the use of Cepheids as a standard candle is they are only luminous enough to measure distances up to ~ 20 Mpc. However, on this scale, the Universe is not considered to be isotropic or homogeneous. To obtain distances > 20 Mpc, much more luminous sources are needed [16].

Type 1a supernovae are much more luminous approximate standard candle sources than Cepheids. There exist several different types of supernovae, which can be classified by their spectral lines. Type 1a supernovae are identified by absorption lines of singly ionized silicon with hydrogen present. Type 1a supernovae are strongly hypothesized to be produced in a binary star system in which a white dwarf accretes mass from its companion star. Once the mass of the white dwarf exceeds the so-called Chandrasekhar limit ($\sim 1.4 M_{\odot}$ - where $M_{\odot}$ represents solar mass), the point which gravitational force exceeds electron degeneracy pressure, the star will undergo a supernova explosion. From basic physical principles, the Chandrasekhar limit should be valid for all accreting type 1a supernovae in the limit of zero spin. Therefore, their corresponding peak brightness is relatively standardized, making them an excellent candidate for a standard candle [18]. However, there is still an intrinsic spread on the absolute magnitude produced from such a phenomenon, limiting the precision needed to make precise cosmological measurements. M. Hamuy et al. demonstrated that there is a close relationship between the absolute magnitude brightness emitted from the explosion and its corresponding light curve: a supernova with a broader light curve will have an absolute magnitude larger than a supernova with a more narrow light curve [19].

Astronomers use the apparent magnitude system to define how bright a star is when observed from Earth. If there are two sources with apparent magnitudes, m_1 and m_2, with measured fluxes, f_1 and f_2, then their apparent magnitudes are related by [16]:

$$m_1 - m_2 = -\frac{5}{2} \log_{10}\Big(\frac{f_1}{f_2}\Big). \tag{2.2}$$

To establish a convention to compare the absolute brightness of stars on the same

footing (i.e. as observed from a standard distance) the absolute magnitude (denoted by M) of a source is defined using its luminosity distance [16]:

$$m - M = 5\log_{10}\Big(\frac{d_L}{10 \text{ parsecs}}\Big), \tag{2.3}$$

Therefore if M is known, in order to determine the luminosity distance, one must simply measure the apparent luminosity of the source.[1] The light curve corrected value of a typical Type 1a supernova is now known to be $M = -19$ with little variation. Hence, Type 1a supernovae are useful as standard candles [19]. These Type 1a supernovae have demonstrated their ability to map the expanding universe, as shown in Fig. 2.1(a), which led to the discovery of dark energy.

2.2 Dark Energy

The Friedmann-Lemaître-Robertson-Walker (FLRW) metric equations describe the expansion of the universe under the assumption that the initial conditions were homogeneous and isotropic. Two equations derived from the metric are used to describe the non-static Universe, which are known as the Friedmann equation and Friedmann acceleration equation respectively [16]:

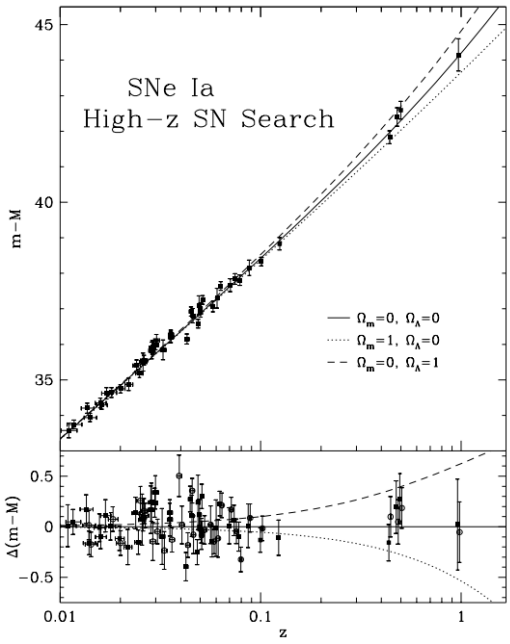
$$H^2 = \Big(\frac{\dot{a}}{a}\Big)^2 = \frac{-Kc^2}{a^2} + \frac{8\pi G}{3}\sum_i \rho_i, \tag{2.4}$$

and

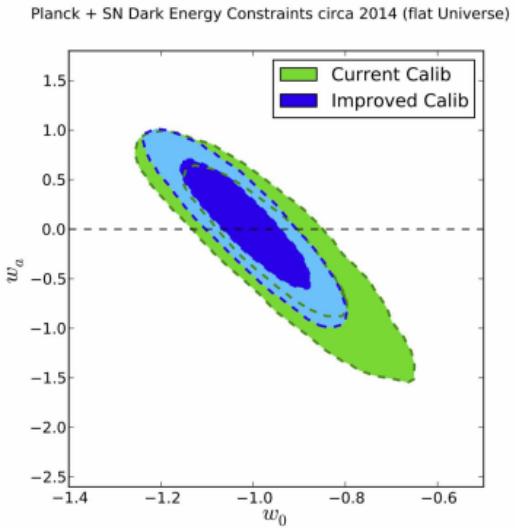
$$\Big(\frac{\ddot{a}}{a}\Big) = \frac{-4\pi G}{3}\sum_i \Big(\rho_i + \frac{3P_i}{c^2}\Big), \tag{2.5}$$

where a is the expansion factor of the Universe and the dot notation denotes the time derivative, K is the spatial curvature of the Universe, G is the gravitational constant, ρ_i is the effective energy density of each component, and P_i is the pressure of each component. The summation is over each component of the Universe: matter, radiation, and dark energy. Each component has a corresponding equation of state parameter $\omega_i = \frac{P_i}{\rho_i}$. A large area of interest in modern cosmology is to determine the equation of state parameter for dark energy to a high resolution, to see if dark energy is consistent with being a cosmological constant over both space and time,

[1] Implicit in this brief overview is that there exists no light extinction from interstellar space, nor from the Earth's atmosphere.



(a) Apparent magitude vs redshift of multiple type 1a supernovae plotted with 3 different universe models. [23]



(b) Constraints on the equation of state parameters of dark energy. [22]

Figure 2.1: Type 1a supernovae as a probe for dark energy.

or not. Generally, the dark energy equation of state is constrained using distance measurements (typically from standard candles/rulers). The equation of state for dark energy need not be constant in time. There exist many different parameterization models, one which is of particular interest takes linear form, suggested by E. Linder [20]:

$$\omega(a) = \omega_0 + (1 - a)\omega_a. \tag{2.6}$$

Constraints on the parameters of dark energy with current and upcoming supernovae 1a dark energy projects (plus the Planck CMB mission) with current calibration standards and projected improvements from enhanced calibration methods can be seen in Fig. 2.1(b). It is estimated that the figure of merit [21] will improve by a factor of 2.4 with better calibration standards [22].

It is important to note that many of the current probes for dark energy leave a degeneracy between the equation of state and the spatial curvature of the universe. Obtaining tighter bounds on the equation of state parameter for dark energy will help eliminate or verify cosmological models.

2.3 ALTAIR

Observational cosmology has historically been plagued with statistical and systematic uncertainties. There have been major reductions in statistical uncertainties with the construction of larger scale surveys. Improved detector technologies enable these large surveys to take measurements at greater redshifts (distances) while simultaneously taking more accurate measurements due to the better technology available today. While these improvements have greatly reduced major uncertainties, a dominant source of measurement error is due to atmospheric attenuation.

Accurate photometry on the Earth's surface is heavily reliant on knowledge of the total transmittance of the atmosphere. The transmittance can depend on many factors. The atmosphere scatters, absorbs, and reflects throughout the electromagnetic spectrum. Uncertainties on atmospheric transmittance, and knowledge of absolute flux and spectra from stellar sources, have left photometric surveys limited to $O(1\%)$ uncertainty. ALTAIR is an initiative to characterize the total atmospheric transmittance and provide a precisely understood optical source at multiple wavelengths, reducing the total photometric uncertainty to $O(0.1\%)$ [15]. To achieve this goal, ALTAIR will loft laboratory-level calibrated sources $\sim$ 20 km into the sky, which are monitored *in-situ* via on-board photodiodes (Fig. 2.2). ALTAIR utilizes helium weather balloons for its ascents. This is a relatively cheap way to obtain the altitudes necessary for the project. The descent is initiated through the use of a "cut-down" mechanism. Once the balloon has reached the desired altitude, it is cut from the balloon releasing a steerable parafoil. The steerable parafoil makes for a relatively simple recovery of the payload.

There exist many unknowns about the transmission properties of the atmosphere. For accurate ground-based astronomy, much needs to be known about atmospheric attenuation. How does it vary with wavelength? What is the temporal dependence? ALTAIR can answer these important questions and benefits from the fact that it is retrieved after each flight. This enables the ability to mount different sources (an integrating sphere, isotropic light, radio waves, etc.) for calibration across a broader range of the electromagnetic spectrum. ALTAIR is also bound by neither location nor time. The payload can be transported to the necessary site with the required calibrated equipment at any point in time, in contrast to satellites which are restricted in time and location by the satellite's orbit.

The light source constructed for this thesis project is intended to determine what

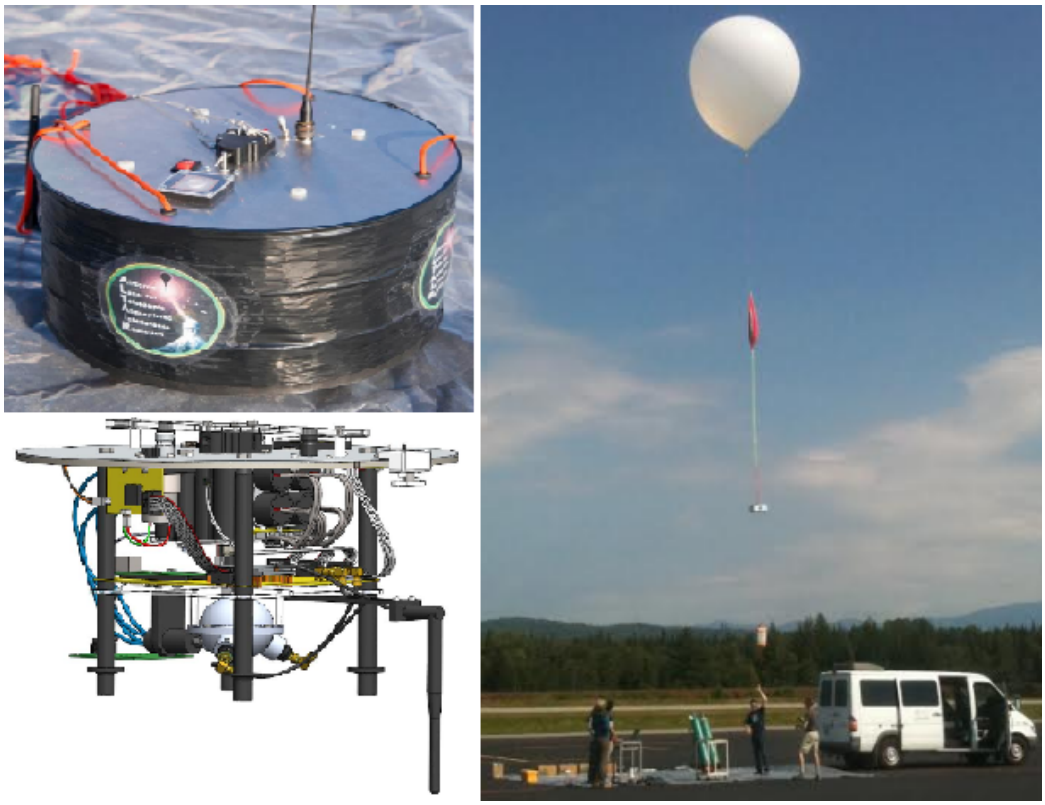


Figure 2.2: A concept design of ALTAIR (bottom left), a constructed ALTAIR payload (top left) and a launch of a flight (right).

level of isotropy can be achieved by a solid state light source, and will be a model source for the ALTAIR payload.

Chapter 3

Experimental Technique

This chapter will provide an overview of the construction and specifications of the light source, the measurement apparatus used to determine the radiant intensity, and the computer software written to operate the measurement apparatus. The chapter will also provide a detailed explanation of the steps taken to reduce the anisotropy of the light source.

3.1 Experimental Setup

3.1.1 The Light Source

The solid state isotropic light source was constructed from LEDs attached to a 60-sided polyhedral die 3D-printed from polyamide plastic. The 60-sided die had a diameter of 3.0 cm with 4-sided polygonal faces, each with an area of ~ 0.45 cm^2. A small hole was drilled on the corner of 3 faces to mount the die onto a steel post.

The LEDs used for this experiment were Lite-On Inc. model LTST-T670TBKT, purchased from DigiKey. The LEDs had a peak emission wavelength of 470 nm (Fig. 3.1 - Left) and a viewing angle of 120° (Fig. 3.1 - Right) after which the intensity falls to half the peak intensity [24]. A single LED was glued on each face of the polyhedral. LEDs were similarly oriented on each face. Current limiting resistors (270 Ω) were used to prevent overcurrenting of the LEDs. All connection were made with 40 gauge enamel coated copper wire. The materials used for the source can be seen in Fig. 3.2.

LEDs were wired such that each had its own voltage source. The ability to control the applied voltage of each LED enabled individual calibration, allowing for greater

control of the total anisotropy.

The wires were threaded in between each LED (on the surface of the die) and ran to a common spot (at the post holding the die). To ensure reliable connections, two methods were used to strip the enamel coating. The first uses a straight edge razor to carefully scratch off the enamel. This technique proved to be difficult as the small wires could be cut easily. The second, better method uses a soldering iron (at a high temperature) to melt off the enamel coating. The negative end of each LED was connected to one row of a 20×2 pin header. The positive ends of each LED were connected to circuit boards with one face plated in copper. The copper cover face of each circuit board was attached to the other row of the 20×2 pin header. Working with 40 gauge wires proved to be a difficult task. The wires themselves were very small, thin, frail, and had a tendency to break if worked with extensively. The debugging of electrical issues was also quite difficult for a few reasons:

- There are at least 120 wires with no coordination system in place.
- Wires would often break while tracing the connections due to low mechanical strength.
- To fix a broken wire, the enamel coating had to be removed - often resulting in wires cut by accident.
- Even operating under a microscope, it was difficult to see if the enamel coating had been properly stripped to ensure a good connection.

With these factors, debugging connections proved to be a delicate task as fixing one issue could induce other issues even when handled with extreme care. Throughout this project, faulty wires/wire connections have caused the majority of electrical issues encountered.

The circuit board was used to mount the 270 Ω current limiting resistors for each LED. The resistors allowed for a larger applied voltage range, enabling more accurate precision of the luminosity of each LED. Each circuit board serviced 20 LEDs, yielding a total of 3 boards. The voltage sources were provided by 4-channel analog output devices ("Phidgets") purchased from Phidgets Inc. Phidgets are easy-to-use microcontrollers which connect directly to a computer via USB with a very comprehensive set of application program interfaces (APIs) and code examples provided by Phidget Inc. Each phidget had a maximum analog output voltage of ± 10

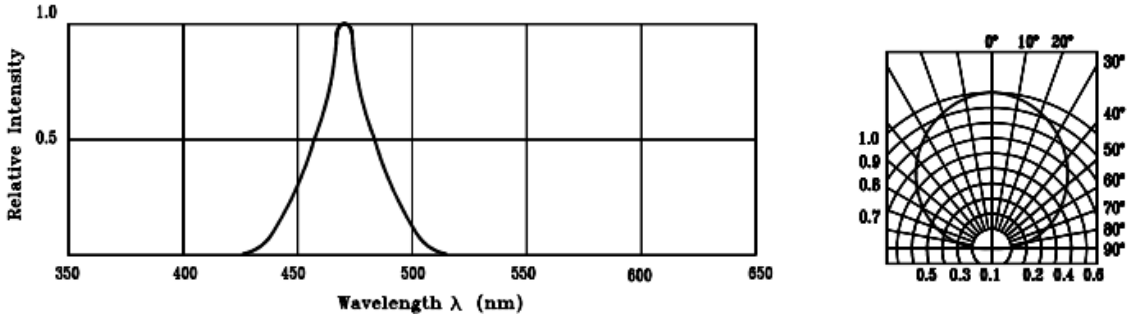


Figure 3.1: (Left) The relative intensity vs wavelength of the LEDs used for this project, peak intensity occurs at 470 nm. (Right) The spatial distribution of the LEDs used for this project, half intensity occurs at 60°. [24]

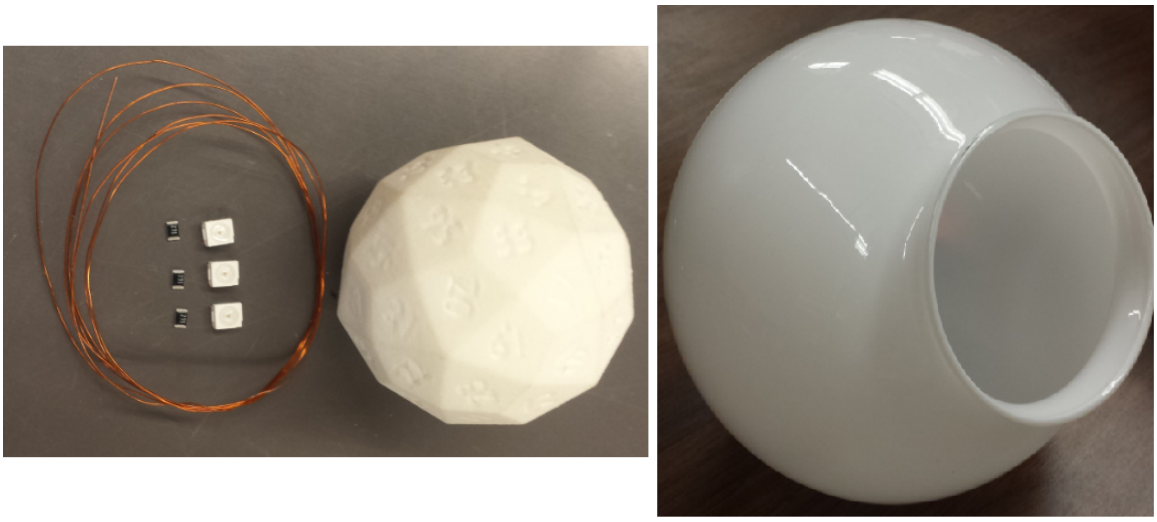


Figure 3.2: The parts used for the source: (Left)A 60-sided polyhedral die, 40 gauge enamel coated wire, 470 nm surface mount LEDs, and 270Ω resistors; (Right) the diffusing sphere.

V direct current with a resolution of ±4.8 mV. The phidgets provided each LED its own voltage source, yielding tunable LEDs.

Each pin header was connected to 5 phidgets using a ribbon cable, yielding a total of 15 phidgets. To manage multiple phidgets, software code was written to control the applied voltage of each LED, discussed in more detail in section 3.1.5.

The polyhedral die with the LEDs and wires attached to it was completely painted black. This was done by masking each LED with small circular masking tape dots and then spray painting the die with the wires and LEDs attached. The black paint promoted light absorption and minimized the reflection of light, reducing these physical difficulties to improve isotropy. The LED ball was then placed within a glass diffusing sphere to smooth anisotropy. The glass diffusing sphere was Lithonia model 11980-BN (seen in Fig. 3.2). The diffusing sphere had a radius of 7.5 cm, and had a circular entrance hole of radius 2.7 cm for the LED ball to be placed within in the sphere [25]. Polar candela distribution and isofootcandle plots of the diffusing sphere show a high degree of symmetry (Fig. 3.3) [26]. The additional data is provided in Appendix B.

3.1.2 Measurement Apparatus

We constructed an apparatus with the objective of measuring the radiant flux at each point on a spherical shell, where the light source (LED ball and diffusing sphere) was located at the origin. The apparatus aimed to achieve four goals:

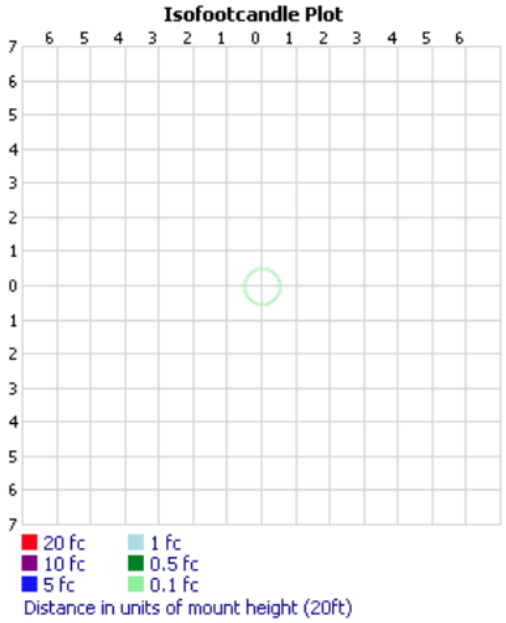
1. To maintain a constant distance between the photodiode and the glass diffusing sphere (Fig. 3.4(a))

2. To ensure the photodiode measurements were taken with the photodiode surface normal aligned with the diffusing sphere surface normal (Fig. 3.4(b))

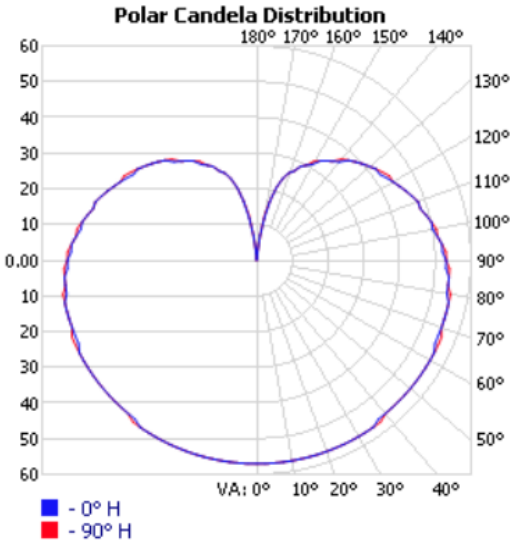
3. To have the photodiode centered on the surface normal of the diffusing sphere (Fig. 3.4(c))

4. To locate the LED ball in the center of the diffusing sphere which encases it (Fig. 3.4(d)

The first three requirements will be referred to as alignment errors and the fourth as a centering error. If the LED ball was not located at the center of the diffusing



(a) Isofootcandle plot of the diffusing sphere. The plotted circle represents where the footcandle (intensity of light on surface equal to 1 lumen per square foot) levels are the same, the number labeled on the axes indicates the distance to either side of the source in multiple of the mount height (20ft).

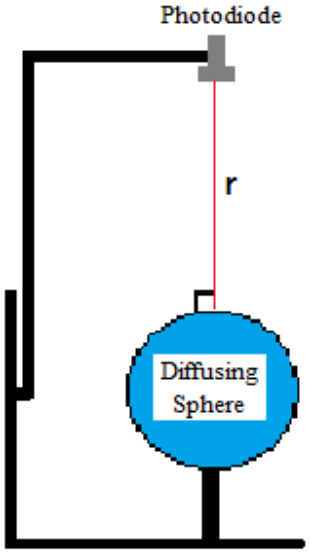


(b) Polar candela distribution of the diffusing sphere. The plotted lines represent the distribution of luminous intensity, the left axis labels show equivalent intensity lines (in candelas), and the bottom/left/top labels show the vertical angle at which the measurement was taken. Plot shows two distributions, one taken at 0° horizontal angle, and another taken at 90° horizontal angle.

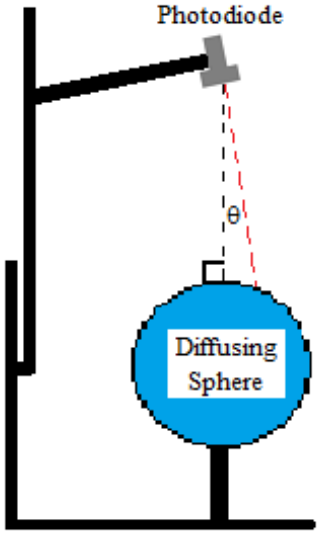
Figure 3.3: Photometric data of the glass diffusing sphere used for the isotropic light source. Data was provided by the manufacturer, Lithonia [26].

sphere, we would expect a dipole in the luminosity data. A more in-depth discussion of these errors will be presented in Chapter 4.

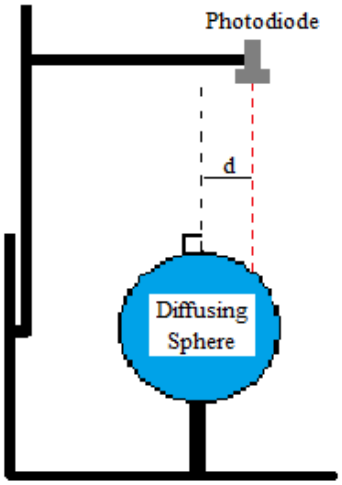
The measurement apparatus (Fig. 3.6) was constructed using a metal framework to maintain the structural integrity of the device (minimizing distortions from moving and shaking of the device during measurements). The entire apparatus rested on a bearing which enabled smooth movement in the azimuthal direction (longitudinal motion). A right angle brace connected the circular part of the base (which rested on the bearing) to the arm responsible for changing the elevation (latitudinal motion). The photodiode was located at the end of the arm. The movement was controlled through the use of stepper motors. One motor attached directly to the arm which provided elevation travel, and the other attached to the bearing via a toothed belt to prevent any slipping due to the inertia of the measurement apparatus. The stepper motors had 200 steps per revolution (1.8°/step), as well as the ability to take full, half, quarter or an eighth of a step (through the functionality of the micro-controller), yielding a sub-step size of 0.225°. The stepper motor responsible for the azimuthal movement contained a "gear system", and there was not a simple 1:1 ratio for a stepper motor step to actual rotation of the photodiode. To calibrate the stepping ratio, 180° markings were placed on the bearing using a ruler and a protractor, and a reference point was marked on a stationary part of the apparatus. The number of steps to achieve 180° of travel was determined using a simple gear ratio equation, and then refined via testing. Using this calibration method it was found that 23 steps at 1/4 stepping rate is equivalent to 1° (yielding an uncertainty of $0.04° = \frac{1}{23}$). The stepper motor responsible for elevation movement was attached directly to the photodiode arm where each step was 1.8°, and this was also cross-checked using a digital goniometer. The stepper motors were activated through the use of a simple micro-controller seen in Fig. 3.7. The micro-controllers relied on a voltage source to control the stepping, the direction of each step, and whether the stepper motor was held rigid or not. The code was written in Python to drive the stepper motors by an analog output phidget. Each microcontroller required its own phidget for operation, requiring an additional two phidgets for the project, yielding a total of 17 phidgets. The micro-controllers for the phidgets step on the falling edge of a clock cycle.



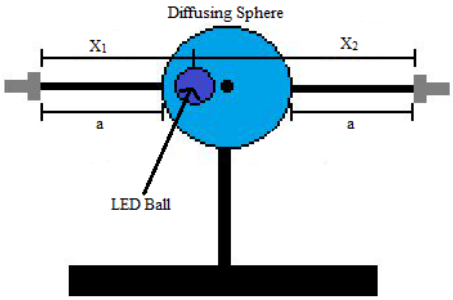
(a) Maintaining a constant distance.



(b) Maintaining measurements normal to the surface.



(c) Photodiode deviation from surface normal.



(d) Centering LED ball inside the diffusing sphere.

Figure 3.4: Apparatus requirements [27].

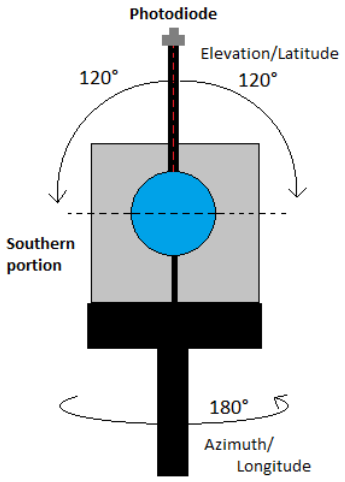


Figure 3.5: The range of motion of the photodiode.

3.1.3 Dark Enclosure

The measurement apparatus itself was located inside a light-tight enclosure, or dark enclosure, made to eliminate any background light which might exist. The enclosure was made using wood, cardboard, matte black paint, and steel struts for structural support. Everything in the enclosure was painted using the matte black paint. In reality, some photons may have leaked through the black enclosure, resulting in some background light. To compensate for this, background measurements were taken and subtracted from the measured flux. Inside the enclosure, the apparatus rested on top of a black table which constricted space on the southern portion of the source (Fig. 3.5). This limited the elevation (ϕ) measurements to cover $(117 \pm 0.04)^\circ$ (coarse resolution plots) and $(118.8 \pm 0.04)^\circ$ (fine resolution plots) of the possible 180°. The solid angle of a circular cone (in steradians) with vertex at the origin, with conical axis of symmetry through an equator of a sphere, and which subtends maximal and minimal elevation angles of that sphere equal to $\pm\phi$ whose cross-section subtends the elevation angle 2ϕ is described as:

$$\Omega = 2\pi(1 - \cos\phi). \tag{3.1}$$

The propagated error on the solid angle is thus described by:

$$\delta\Omega = 2\pi(\delta\phi \cdot \sin\phi). \tag{3.2}$$

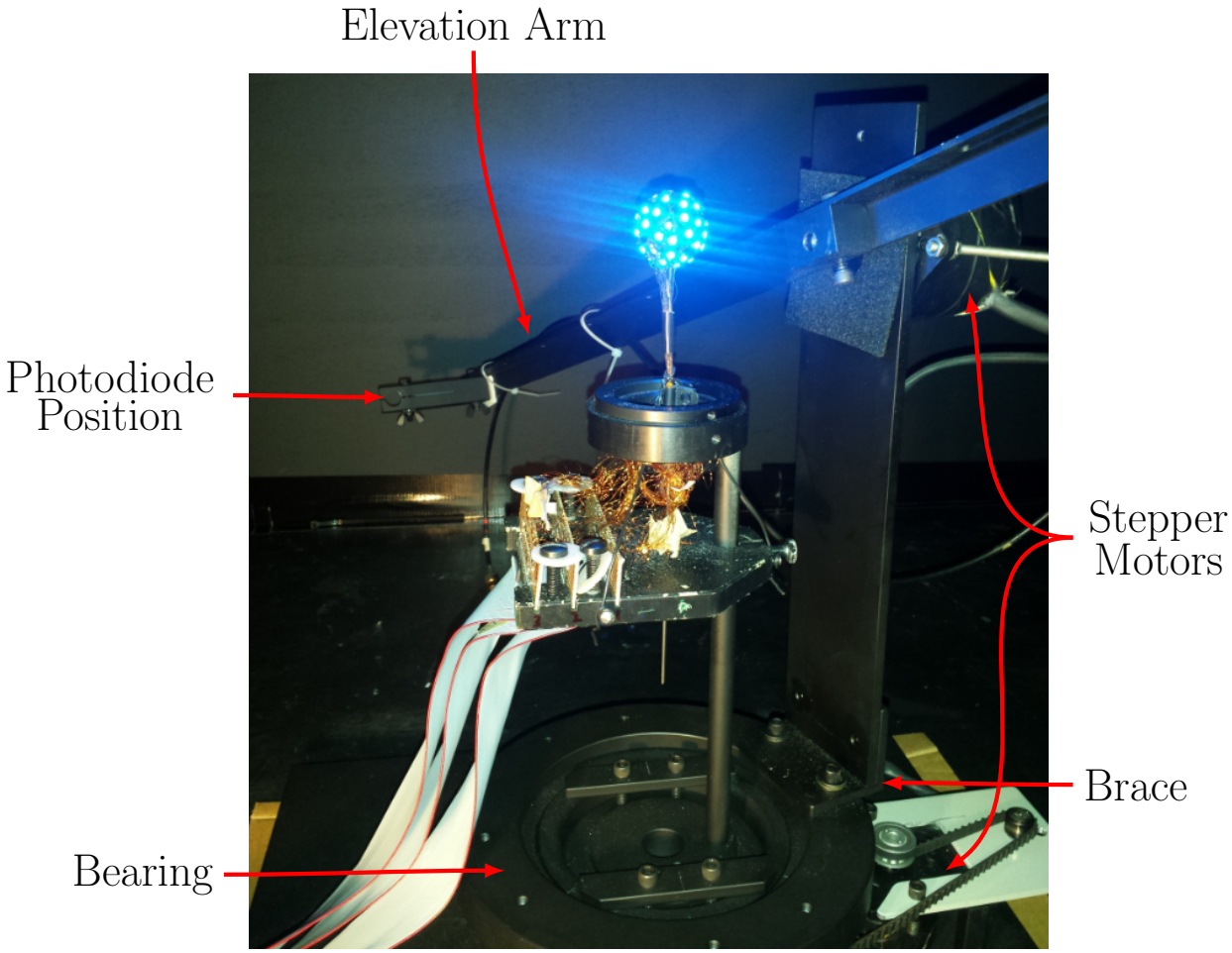


Figure 3.6: The measurement apparatus with light source attached (without the glass diffusing sphere).

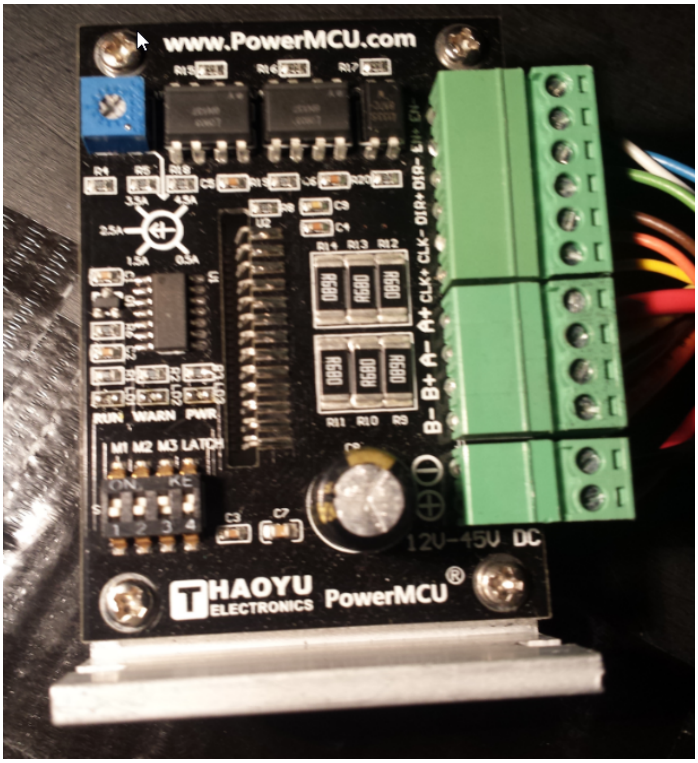


Figure 3.7: The microcontrollers used for controlling the stepper motors.

From equation 3.1 and 3.2 the measurement apparatus is able to measure $(2.91 \pm 0.07)\pi$ sr (for coarse resolution plots: $\phi = (117.00 \pm 0.04)^\circ$) and $(2.96 \pm 0.07)\pi$ sr (for fine resolution plots: $\phi = (118.80 \pm 0.04)^\circ$) of the possible 4π sr.

3.1.4 Photodiode and Photosensor Amplifier

To measure the photon flux produced from the light source at a given area, a silicon photodiode — purchased from Hamamatsu, model S2281 – was used. The photodiode had an active area of 100 mm^2, with sensitivity in the spectral range of 190 to 1100 nm, with a peak sensitivity occurring at approximately 960 nm, and measured at a 10 kHz freq range. The photodiode was calibrated at the National Institute of Standards and Technology (NIST). The calibration determined the radiant power responsivity of the photodiode and the uniformity over the active area.

The photodiode was used in conjunction with Hamamatsu's photosensor amplifier, model C9329. The photosensor amplifier output stream was in hexadecimal format. The measured hexadecimal value was converted into a voltage, with propagated uncertainty, using a formula provided in the photosensor amplifier operation manual [28].

3.1.5 Computer Code

Our computing software, written in Python, commanded the Phidgets in ascending order of their corresponding serial number. This was necessary to ensure a consistent placement of each LED in memory (e.g., the LED marked LED#4 in the program would physically be the same LED on the polyhedral die upon each start). The software initialized the applied voltages from a list (represented as a text file). The voltage list contained one voltage per line (60 lines total) in decimal form. Error catches were written into the program to prevent (1) improper inputs, which include invalid keystrokes, and/or (2) applying too large a voltage (each Phidget had a threshold of $\pm$10V DC max). The program allowed the operator to change the voltage of any LED easily, turn off all LEDs (by applying 0V), and save any changes made during runtime (i.e., updating the voltage list). The "Phidgets" software package provided by Phidgets Inc. was used for development. The software was also responsible for handling the data output. At each measurement location, the photodiode was read out to the photosensor amplifier (Hamamatsu C9329) which was connected to the computer via a serial input. The software recorded the photodiode output voltage,

azimuth angle, and elevation angle at each measurement. The photodiode was then moved by the apparatus to the next location, and the process was repeated until the whole $(\frac{4}{3}+2)\pi$ solid angle was covered. The computer code can be seen in Appendix A.

The software also provided the ability to test the stepper motors and photodiode, which was used to aid the setup of the experiment.

The "Basemap" software package was used to visualize the data as isotropy maps. The basemap package provides many different data projection for visualization.

3.2 Isotropy Measurements

The radiant intensity, defined in 1.1, was measured using a NIST-calibrated photodiode (Fig. 3.8(a)). The photodiode has an active area (A), records a voltage (V_{pd}), hence conversion factors provided by the photodiode (R) and photosensor amplifier specifications (C) manual [28] were used to convert each measurement into a flux power density (irradiance, in watts/m^2), given by:

$$\phi_{\mathrm{pd}} = \frac{V_{\mathrm{pd}}}{C \cdot R \cdot A}, \tag{3.3}$$

where V_{pd} is the readout voltage from the photosensor amplifier, C is the conversion impedance specified by the photosensor amplifier (in volts/ampere), R is the photo sensitivity of the photodiode (in amperes/watt), and A is the active area of the photodiode. The flux power density measurements taken with the photodiode will be dependent on its distance to the glass diffusing sphere. To account for the small perturbations of this distance, the inverse square law is applied so that each measurement estimates the flux at the surface of the diffusing sphere. Furthermore, the photodiode measurements will have a cosine dependence of the angle with respect to the surface normal of the diffusing sphere [29]. Therefore, the irradiance at the surface of the diffusing sphere was given by:

$$\phi_{\mathrm{sphere}} = \phi_{\mathrm{pd}} \cdot \frac{\cos\theta}{d^2}. \tag{3.4}$$

From the photosensor amplifier/photodiode specification manual [28] and the NIST calibration report [30], the numerical values of the conversion factors were found to be:

(a) The photodiode used in this experiment, Hamamatsu model S2281

(b) The conical cap used to calibrate each LED.

Figure 3.8: The photodiode and a conical cap constructed for calibration.

- $C = 1.0 \cdot 10^7$ V/A
- $R = 0.2394 \pm 0.0005$ A/W (i.e., an uncertainty of 0.22%)
- $A = 100 \pm 0.008$ mm^2 (i.e., an uncertainty of 0.008%)

The photosensor amplifier manual provides an uncertainty for the output (discussed in detail in section 4.2), which would incorporate the uncertainty of C within; hence the uncertainty of C for the conversions is assumed to be zero.

3.2.1 Calibration

To center the LED ball within the glass diffusing sphere, it was necessary to calibrate each LED individually so that there was no initial prominent bulge in the light distribution. The presence of an initial bulge could have resulted in artificial centering of the LED ball within the diffusing sphere (i.e. to compensate for the extra light distribution where the bulge has occurred). A conical cap made from cardboard and electrical tape (seen in Fig. 3.8(b)) was attached to the photodiode. This was used to calibrate individual LEDs. The conical cap achieved two things:

1. To ensure a constant measuring distance, when flux measurements were taken the cap was able to rest firmly on the polyhedral die.

2. To standardize the amount of light captured on each measurement.

The calibration process was initiated (and performed with the photodiode connected directly to a voltmeter) by applying a voltage of 3.0 V to an arbitrary LED, which produced a baseline output voltage of 0.30 V(V_{pd}, via the photocurrent). The applied voltages of all other 59 LEDs were set such that their output voltage matched that of the baseline measurement of 0.30 V. The largest form of uncertainty on this measurement was keeping the photodiode perpendicular to the LED's surface normal, as these measurements were done by hand (holding the photodiode).

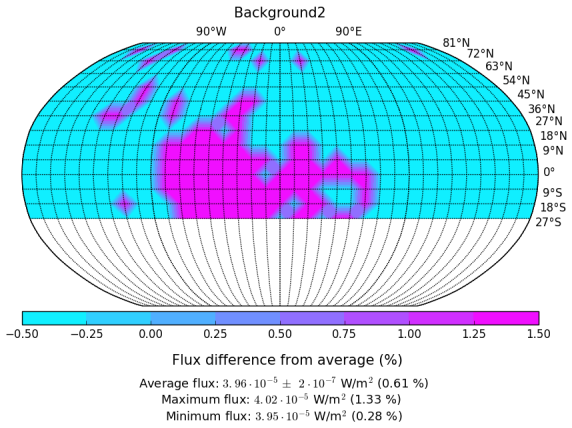
3.2.2 Background Light Elimination

To account for any photons which may have seeped through the constructed black box, background measurements were taken. A set of 4 background measurements were taken and then averaged. The averaged value was subtracted from all flux measurements to eliminate background light. The results (seen in fig. 3.9) showed a relatively constant background disrupted by statistical fluctuations from the photodiode, and/or the photosensor amplifier. On average, this statistical fluctuation was $7 \cdot 10^{-7}$ W/m^2 which represented 1.6% of the average background flux ($3.96 \cdot 10^{-5}$ W/m^2), and 0.009% of the average of the source when illuminated ($3.41 \cdot 10^{-2}$ W/m^2). As this fluctuation was such a small portion of the total signal, it can be eliminated as a major source of error. The background itself contributed $\sim 0.5\%$ of the total signal from the source when illuminated.

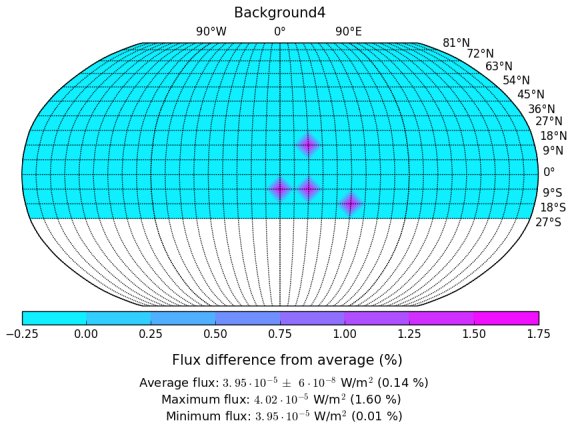
3.2.3 Anisotropy Reduction

The LEDs were approximately uniformly distributed around the 60-sided polyhedral ball. To reduce the anisotropy, each LED was fine-tuned. Every LED was on a self-contained circuit with a variable voltage source (a phidget) controlling the brightness of the LED. "Isotropy maps" were generated by taking flux measurements on a spherical shell around the source. The flux extrema were examined manually (plotted visually) and appropriate LEDs were fine-tuned (i.e. their voltage was changed) to reduce the flux domain.

The locations of all LEDs were determined by illuminating one LED, and then generating an isotropy map of the single LED. The position of maximum flux was selected as the LED location (Fig. 3.10). This method was repeated for each of the

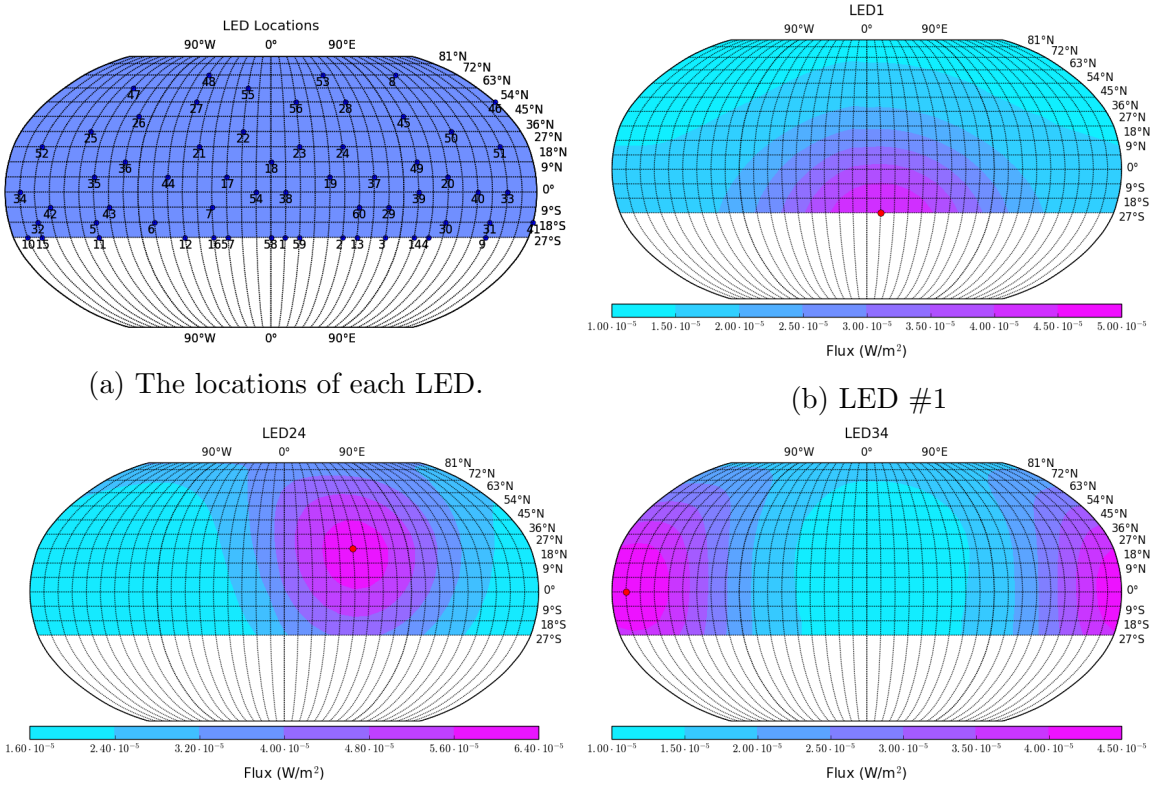


(a) Background measurement #2



(b) Background measurement #4

Figure 3.9: Two sample background measurements are shown. The relatively small range of excursions from the flux average, given by the maximum and minimum flux values measured by the photodiode within each background measurement, was equivalent in magnitude across all background measurements.



(a) The locations of each LED.

(b) LED #1

(c) LED #24

(d) LED #34

Figure 3.10: LED locations and example isotropy maps of three different LEDs.

60 LEDs. Two ambiguities were present when locating LEDs:

1. The first occurrence of a maximum was selected (if a maximum occurs at two or more measurements).

2. The accuracy was restricted to the flux measurement intervals (i.e. the resolution of the isotropy map).

It should also be noted that since LEDs were distributed approximately uniformly across the LED ball and flux measurements on the southern half were restricted due to physical limitations (discussed in section 3.1.2), we see a higher density of LEDs on the south portion of the light source. The distribution of LEDs can be seen in Fig. 3.10(a) with examples of single LED isotropy maps in Fig. 3.10(b), 3.10(c), and 3.10(d).

Once the location of each LED was identified, further anisotropy reduction was achieved easily by fine-tuning appropriate LEDs. The total flux at any given point

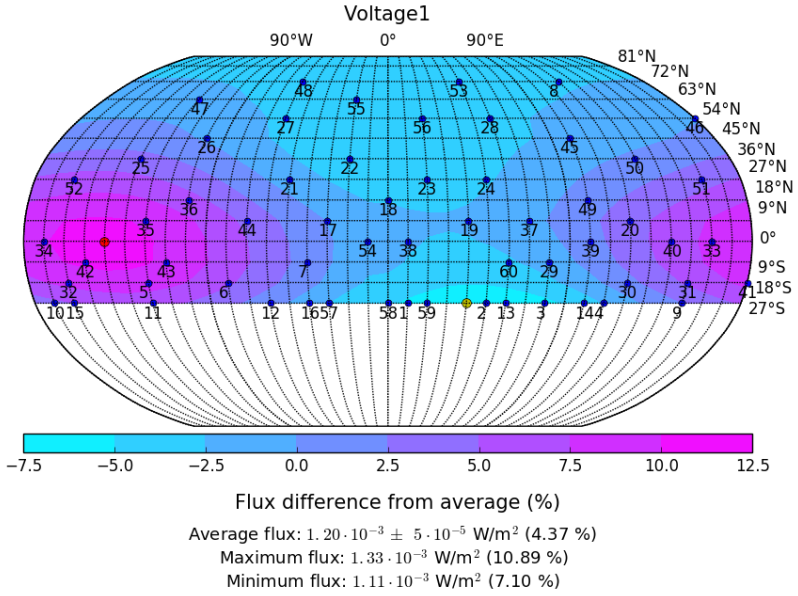
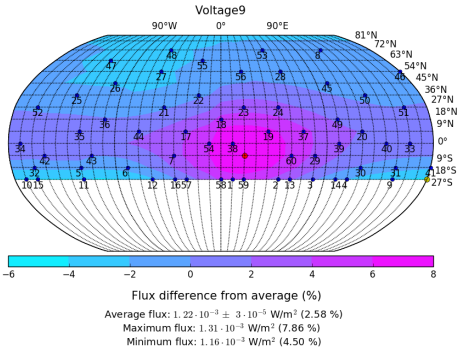


Figure 3.11: The isotropy plot of the first LED adjustment.

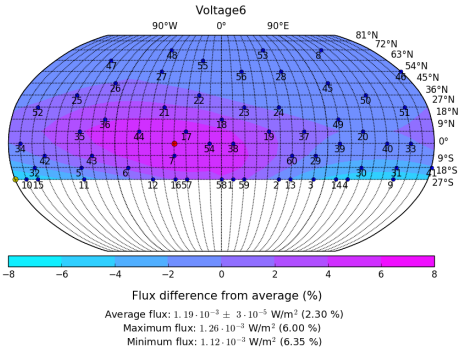
around the source, at a constant distance, is a superposition of the flux emitted by each LED:

$$F_{\text{total}}(\theta, \phi) = \sum_{i=1}^{60} f_i(\theta, \phi), \tag{3.5}$$

where $f_i(\theta, \phi)$ is the flux contribution of LED i at point (θ, ϕ) around the source. While each LED contributed to the total flux at a given point, only one (in some cases more than one) LED was adjusted on each iteration of anisotropy reduction. The strategy taken to fine-tune each LED was to identify the extrema regions in the previous isotropy plots and adjust the applied voltage to the LEDs located nearest to the extrema. For example, in the last iteration of the centering process (Appendix C.1(f)) there is a clear maximum flux region located near LED #19, #38, and #60. The maximum point is closest (spatially) to LED #19, hence only LED #19s applied voltage was adjusted from $2.92V \rightarrow 2.50V$. Fig. 3.11 shows the result of the voltage adjustment on LED #19. The maximum region shifted to encompass a new LED subset and a revised decision was made on which LED to adjust. Typically, only one LED was adjusted at a time so that the anisotropy was reduced in the most controlled manner. Occasionally, two or more LEDs were adjusted. Such adjustments were made when it was necessary to both reduce an LED's luminosity in the maximum region and increase an LED's luminosity in the minimum region. By limiting adjustments at any given iteration, the anisotropy was smoothed out gradually. The choice to decrease



(a) Difference of maximum to average was larger hence luminosity of LED #38 was *decreased* ($3.0V \rightarrow 2.5V$)



(b) Difference of minimum to average was larger hence luminosity of LED #10 was *increased* ($3.03V \rightarrow 3.6V$)

Figure 3.12: Demonstration of how the choice to increase or decrease luminosity was made.

the luminosity in maximum regions or increase the luminosity in minimum regions was driven by the magnitude of the difference between the extrema and average flux. If the difference of the maximum to the average was larger than the minimum to the average, as in Fig. 3.12(a), then an LED's luminosity was decreased. If the difference of the minimum to the average is larger than the maximum to the average, as in Fig. 3.12(b), then an LED's luminosity is increased. This was the strategy used to fine-tune the LEDs and reduce the total anisotropy.

Chapter 4

Discussion of Measurement Errors

Voltage measurements taken with the photodiode were restricted by the photosensor amplifier resolution ($V_{pd} \pm \delta V_{pd}$), the photodiode active area ($A \pm \delta A$), and the photosensitivity of the photodiode ($R \pm \delta R$). Each measurement was also subject to positional errors ($d \pm \delta d$ and $\theta \pm \delta\theta$). Such errors were introduced in section 3.1.2. Applying standard error propagation rules to the flux equation 3.4, the error associated with each measurement is defined by:

$$\frac{\delta\phi_{\text{sphere}}}{\phi_{\text{sphere}}} = \sqrt{\left(\frac{2 \cdot \delta d}{d}\right)^2 + \left(\frac{\delta V_{\text{pd}}}{V_{\text{pd}}}\right)^2 + (\delta\theta \cdot \tan\theta)^2 + \left(\frac{\delta R}{R}\right)^2 + \left(\frac{\delta A}{A}\right)^2}, \tag{4.1}$$

where δ represents the standard error of the measurement. The following section will provide a detailed analysis on how these errors are estimated for anisotropy calculation.

4.1 Distance from Photodiode to Diffusing Sphere

The distance from the photodiode to the glass diffusing sphere (d) was measured at 513 points using a physical ruler. Each measurement was cross-checked with a wooden yard stick and a tape measure, both which have a standard error of ± 0.5 mm. The glass diffusing sphere was centered in the measurement apparatus during construction. However, measurements of d at each position showed a 0.7 cm total fluctuation. The fluctuation in d was strictly a function of the elevation angle (ϕ), independent of the azimuth angle (θ), as seen in fig. 4.1. Measurements of d taken at the northern pole of the light source were 51.3 cm and measurements of d at the

southern edge were 50.6 cm. These measurements assumed the ruler resolution of ±0.5 mm as the error. Therefore, the error assumed for each distance measurement is:

$$\delta d = 0.0005 \text{ m}. \tag{4.2}$$

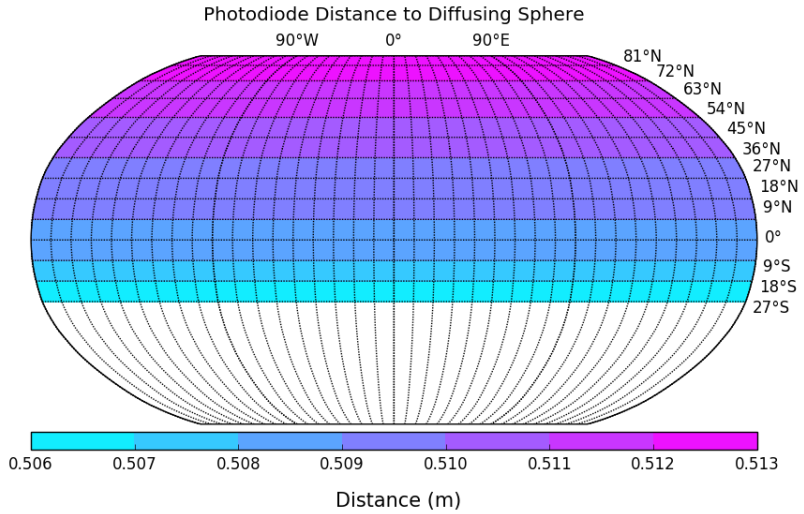


Figure 4.1: Distances from photodiode to glass diffusing sphere.

Table 4.1 outlines the error contributions of each distance measurement at the min, max, and average distance values when considering a measurement error described in equation 4.7. Using the average distance of 51.0 cm, calculated from the 513 measurements, the uncertainty due to the distance measurement $\left(\frac{2\cdot\delta d}{d}\right)$ induced a relative uncertainty of approximately 0.2% on the total radiant intensity at the surface of the glass diffusing sphere.

Table 4.1: Systematic uncertainties due to photodiode distance to glass diffusing sphere.

		d (m)		
		worst	estimated	best
		0.506	0.510	0.513
δd (m)	0.0005	0.002	0.002	0.002

Systematic relative uncertainties due to the distance measurement between the photodiode and glass diffusing sphere $\left(\frac{2\cdot\delta d}{d}\right)$. Uncertainty calculate at: min, max, and mean distances.

4.2 Photosensor Amplifier Resolution

The Hamamatsu C9329 photosensor amplifier was used in junction with the photodiode to record the photocurrent. The photosensor amplifier specification manual [28] indicates the output has peak-to-peak noise of 0.5 mV. Hence, the error on the voltage measurements was assumed to be:

$$\delta V_{\text{pd}} = 0.0005 \text{ V}. \tag{4.3}$$

Table 4.2 outlines the error contributions of each voltage measurement at the min, max, and mean voltage measured, when considering a voltage measurement error described in equation 4.3. Using an average voltage reading of 1.935 V, the uncertainty due to the photosensor resolution $\left(\frac{\delta V_{\text{pd}}}{V_{\text{pd}}}\right)$ induced a relative uncertainty of approximately 0.03% of the total radiant intensity at the surface of the glass diffusing sphere.

Table 4.2: Systematic uncertainties due to the photosensor amplifier resolution.

		V_{pd} (V)		
		worst	estimated	best
		1.518	1.935	2.350
δV_{pd} (V)	0.0005	0.0003	0.0003	0.0002

Systematic relative uncertainties due to the voltage measurement of the photodiode $\left(\frac{\delta V_{\text{pd}}}{V_{\text{pd}}}\right)$. Uncertainty calculated at: min, max, and mean voltages.

4.3 Angle Between Surface Normals

As described in section 3.1, each flux measurement was assumed to have been taken such that the photodiode was situated normal to the surface of the glass diffusing sphere. The angle between the surface normals of the photodiode and the diffusing sphere (θ) was not physically measured. As (~ 0) was a small value, it would have required highly precise measurement to limit experimental error. A toy Monte Carlo simulation was performed to obtain a reasonable estimate of the experimental error which was due to the photodiode and diffusing sphere surface normal alignment.

There was a small shake in the measurement apparatus when the photodiode moved from one measurement location to the next. This could have resulted in

small fluctuations of theta. It was assumed that the apparatus was built such that theta was equal to zero, with imperfections due to the photodiode movement. It was also assumed that these imperfections were normally distributed. Thus, the simulation sampled theta from a normal distribution with a mean of zero. In order to construct the sampling distribution, it was necessary to calculate its standard deviation. Calculation of the standard deviation required some limiting theta value. Thus, the upper extreme and lower extreme of theta were measured.

Examination of the joint on the metal arm which held the photodiode showed a variation of 4° in total when forcefully pushed to the extrema. Therefore, the limiting extrema values were assumed to be ±3°. Using the standardize z-distribution with an alpha value of 0.01 (thus, 99% of the sampled values would fall within the limits), the standard deviation was found to be 1.165, as

$$z = \frac{x - \mu}{\sigma}. \tag{4.4}$$

Since the cosine function is symmetric about zero, the absolute value of each sampled theta was stored as a value. A typical sampling distribution (with $n = 1000000$) can be seen in Fig. 4.2. Theta was sampled n times, and the standard error of all n samples was used as the standard error for theta:

$$\delta\theta \simeq 0.7°. \tag{4.5}$$

Table 4.3 outlines the error contribution of the angle between surface normals of the photodiode and glass diffusing sphere. The table outlines the error at the min, max, and mean angles measured in the Monte Carlo simulation. The table also explores a variation of the standard error for theta ($\delta\theta$) to demonstrate the importance of minimizing this measurement. Using an average sampling mean of 0.9° and angle measurement error described in equation 4.5, the uncertainty due the angle between surface normals of the photodiode and glass diffusing sphere ($\delta\theta \cdot \tan\theta$) induced a relative uncertainty of approximately 0.02% of the total radiant intensity.

4.4 Photodiode Photosensitivity

The photosensitivity of the photodiode (R) was used to convert the recorded output voltage from the photosensor amplifier into watts. Calibration of the photodiode [30]

Table 4.3: Systematic uncertainties due to angle between surface normals.

		θ (°)		
		min.	mean	max.
		~ 0	0.9	~ 3
$\delta\theta$ (°)	0.1	~ 0	0.00003	0.00009
	0.7	~ 0	**0.0002**	0.0006
	1.5	~ 0	0.0004	0.0014

Systematic relative uncertainties due to the theta measurement of the surface normals of the photodiode and glass diffusing sphere ($\delta\theta \cdot \tan\theta$). Uncertainties calculated at the min, max, and mean theta values measured in the Monte Carlo simulation. Uncertainties are also explored with different standard errors for theta. The bolded value was the estimated uncertainty for this measurement throughout the experiment.

determined that at a wavelength of 470 nm (that of the LEDs used in the experiment) $R = 0.2394$ A/W. The calibration showed and uncertainty on R of 0.22%. Hence, the error of the photodiode's photosensitivity was assumed to be:

$$\delta R = 0.0005 \text{ A/W}. \tag{4.6}$$

The uncertainty due to the photodiode's photosensitivity ($\frac{\delta R}{R}$) induced a relative uncertainty of approximately 0.2% of the total radiant intensity.

4.5 Photodiode Active Area

The active area of the photodiode (A) and associated uncertainty (δA) is given in the photodiode operation manual [28]. The photodiode had an active area of 100 mm^2, with a diameter uncertainty of 0.1 mm. Propagating this uncertainty through the equation for the area of a sphere ($A = \pi r^2$) the estimated uncertainty of the total area was found to be:

$$\delta A = 0.008 \text{ mm}^2. \tag{4.7}$$

Considering the manufacturer measured active area of 100 mm^2, the uncertainty due to the photodiode's active area ($\frac{\delta A}{A}$) induced a relative uncertainty of approximately 0.008% of the total radiant intensity.

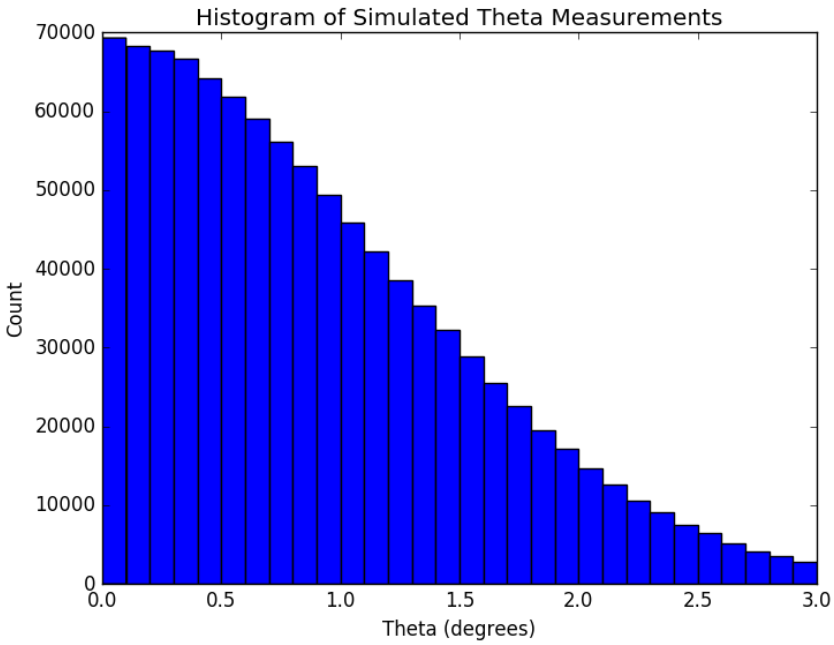


Figure 4.2: Histogram of a Monte Carlo simulation of theta.

4.6 Overview

Table 4.4 outlines the best case scenario, worst case scenario, and our estimate of the relative error for each term in equation 4.1. Table 4.4 assumes the standard theta error ($\delta\theta$) to be that presented in equation 4.5.

There are two dominate sources of uncertainty within this experiment: the distance between the photodiode and the diffusing sphere, and the photosensitivity of the photodiode. The distance measurement could be improved upon by using a laser measurement device. This option was explored but appeared to be unreliable for this experiment because the laser measurement device was pointed by hand, resulting in large variations from measurement to measurement. To reliably improve on the distance measurement with a laser measurement device, the apparatus would be required to maintain the orientation of each laser measurement. The photodiode's photosensitivity was calibrated at national standards and therefore does not leave much room for improvement at the time of this experiment.

The angular displacement between the photodiode and diffusing sphere surface normals was the most fluctuating source of uncertainty which showed fluctuations up to two orders of magnitude.

Considering the average voltage reading ($V_{\text{pd}} = 1.935$ V), average distance measurement ($d = 0.510$ m), and average theta ($\theta = 0.9°$), the total relative uncertainty presented in equation 4.1 is found to be:

$$\frac{\delta\phi_{\text{sphere}}}{\phi_{\text{sphere}}} = 0.0003. \tag{4.8}$$

Table 4.4: Overview of systematic errors.

	$\frac{2\cdot\delta d}{d}$	$\frac{\delta V_{\text{pd}}}{V_{\text{pd}}}$	$\delta\theta \cdot \tan\theta$	$\frac{\delta R}{R}$	$\frac{\delta A}{A}$	$\frac{\delta\phi_{\text{sphere}}}{\phi_{\text{sphere}}}$
best	0.002	0.0002	0.00003	0.002	0.00008	0.003
estimated	0.002	0.0003	0.0002	0.002	0.00008	**0.003**
worst	0.002	0.0003	0.0004	0.002	0.00008	0.003

Overview of each systematic error. The bolded value represents the estimated experimental limitation of this project (0.3%).

Chapter 5

Results

The experiment resulted in a total of 93 individual isotropy plots. These plots include the initial measurement of the setup, five centering iterations, and 87 LED adjustment iterations. Of the 87 LED adjustment iterations, 70 were taken with an azimuth spacing of 10° and an elevation spacing of 9°, yielding a total of 518 measurements per isotropy plot. The other 17 LED adjustment iterations were taken with an azimuth spacing of 5° and an elevation spacing of 5.4°, yielding a total of 1679 measurements per isotropy plot. Scans with 518 measurements took approximately 55 minutes to complete, and scans with 1679 measurements took approximately 3 hours to complete. All scans were taken with a photodiode distance as discussed in section 4.1. Anisotropy progression from LED adjustments can be seen in Fig. 5.1. This plot shows the standard deviation of a given data set relative to the mean of the data set. The red dotted line indicates the experimental error limitations of this project. With the outlined design for a nearly isotropic light, it was relatively easy to achieve an isotropy level close ($\sim 1\%$) to that of the experimental limits of this project. The large spike in this plot between isotropy plot numbers 21 and 22 was due to debugging a broken wire. Smaller spikes were situations where reducing an extremum resulted in a larger flux spread; however continuing voltage adjustments showed further anisotropy reduction.

The source was left illuminated for a 24 hour period to test its consistency over time. Fig. 5.2(a) shows the isotropy plot of LED adjustment iteration # 43 and Fig. 5.2(b) shows the result of the source being illuminated for a 24 hour period. There was a very small change in the standard deviation anisotropy from 1.78% to 1.77%. The extrema anisotropies also showed a small changes, from $^{+4.48\%}_{-2.49\%}$ to $^{+4.46\%}_{-2.47\%}$. The overall light distribution showed subtle changes on a measurement per

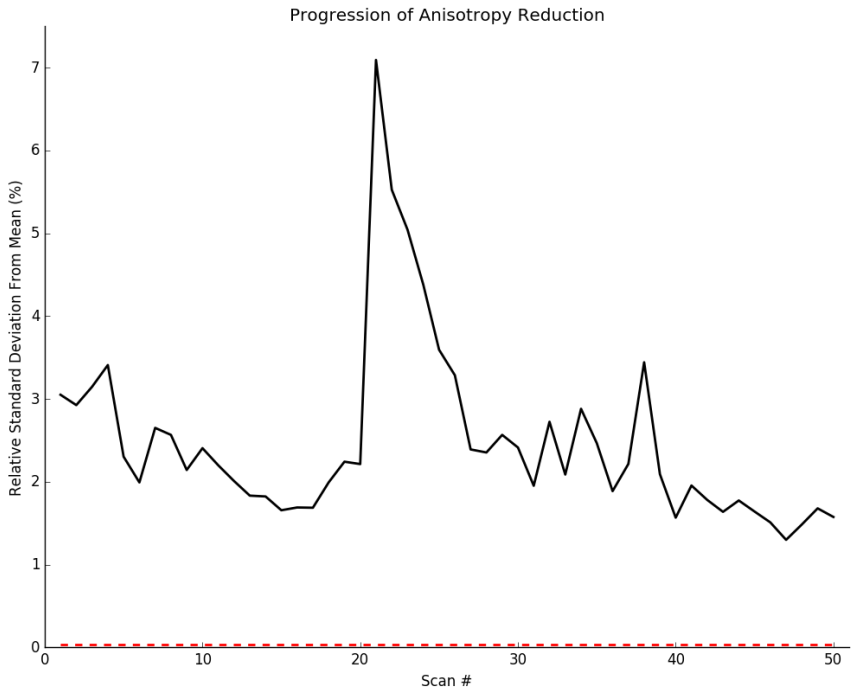
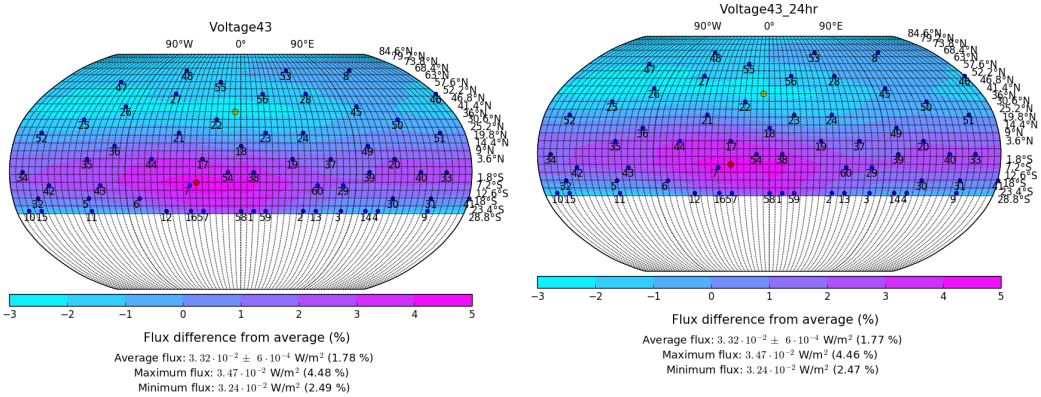


Figure 5.1: The progression of the anisotropy reduction, measured in terms of the standard deviation of all flux measurements to the average. The red dotted line indicates the experimental error threshold.



(a) LED adjustment iteration # 43.

(b) LED adjustment iteration #43 after being illuminated for 24 hours.

Figure 5.2: Demonstration of source consistency over a 24 hour period.

measurement basis, but the overall light distribution was maintained throughout the 24 hour period. In a laboratory setting, the outlined source was able to reliably maintain its anisotropic characteristics over a prolonged period of time.

The lowest standard deviation anisotropy level achieved with the source was 1.30% ($4.10 \cdot 10^{-4}$ W/m^2), which occurred on LED adjustment iteration # 48. The lowest extrema anisotropy (i.e. source with the smallest flux domain) level achieved with the source was $^{+2.98\%}_{-2.68\%}$, also occurred on LED adjustment iteration # 48. The average flux produced by such a source was $3.42 \cdot 10^{-2}$ (W/m^2). The peak flux was $3.52 \cdot 10^{-2}$ (W/m^2), and the minimum flux was $3.33 \cdot 10^{-2}$ (W/m^2). Two scans of this source were taken, both seen in Fig. 5.3. The anisotropy of this source is provided in table 5.1 along with 90% and 95% confidence intervals.

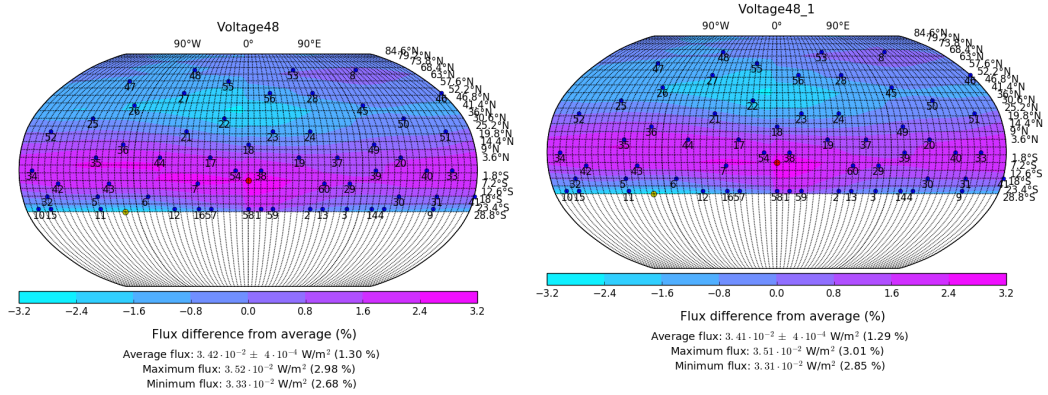
Table 5.1: Overview of source with lowest anisotropy – LED adjustment # 48.

Avg. Flux (W/m^2)	Extrema (W/m^2)	Std. (W/m^2)	90% CI (W/m^2)	95% CI (W/m^2)
3.42×10^{-2}	$^{+3.52\times10^{-2}(2.98\%)}_{-3.33\times10^{-2}(2.68\%)}$	$\pm4.10 \times 10^{-4}(1.30\%)$	$\pm7.30 \times 10^{-4}(2.13\%)$	$\pm8.70 \times 10^{-4}(2.54\%)$

The average flux, extrema flux, standard deviation, 90 % confidence intervals (CI), and 95% confidence intervals of the source with the lowest anisotropy.

There are many improvements which could be made for the next generation of an isotropic light source.

- A wire coding scheme would make debugging broken or detached wires much easier.



(a) The first scan at LED iteration # 48.

(b) The second scan at LED iteration # 48.

Figure 5.3: The lowest standard deviation anisotropy level achieved in the experiment, occurred at LED adjustment iteration # 48. Two scans were taken at this iteration, taken 24 hours apart (the source was turned off during the 24 hour waiting period).

- Increasing the size of the wires would have two benefits. (1) A larger wire would be plastic wrapped which would be *much* easier to strip, compared to the enamel coating on the wires used in this project, ensuring quality electrical connections. (2) A larger wire would also have additional mechanical strength making accidental wire breakage more infrequent.

- Automating the LED adjustment process would be a major improvement to this project. If the illumination scaling of each LED is known across the entire source, it may be possible to develop an algorithm to automate these adjustments. The overhead of this improvement would be large; automation would eliminate the trial-and-error element of this project, and could likely produce a source of the same anisotropy level (or better) in fewer iterations.

Chapter 6

Conclusion

A solid state light source was constructed from 60 LEDs glued nearly uniformly on the faces of a 60 sided die. A measurement apparatus was constructed to move a photodiode around a spherical shell with the light source centered at the origin. The photodiode was used to measure the flux on regular intervals around the source. All measurements were taken within a constructed dark enclosure.

The constructed source had an average flux, over a solid angle of 2.96π steradians, of 3.42×10^{-2} (W/m^2) with 90% (95%) of that solid angle being within $\pm 2.13\%$ ($\pm 2.54\%$) anisotropy. The extrema anisotropies of this source were found to be $^{+3.52\times10^{-2}(2.98\%)}_{-3.33\times10^{-2}(2.68\%)}$ (W/m^2). The experimental limitations of the constructed setup were estimated to be at the 0.3% level. The anisotropy levels achieved with this source were approaching that of the limitation of this experiment. To improve upon this current method, an automation of the LED fine tuning process is suggested, eliminating the trial and error aspect of this experiment.

Appendix A

Computer Code

The software code (written in python) used to control the measurement apparatus was:

```
#!/hepuser/pkovacs/Thesis/IsotropicLightSource/bin/python

__author__="Paul Kovacs"

#Basic imports
import sys
import math
import serial
from time import time
from time import sleep

#Phidget specific imports
from Phidgets.PhidgetException import PhidgetException
from Phidgets.Devices.Analog import Analog

##############################################################################
#########################  USER-DEFINED FUNCTIONS  ###########################
##############################################################################

""" displayDeviceInfo
Display if the device was attached, the device type, device serial number, and
the version of device.
"""
def displayDeviceInfo(device):
    print("|------------|----------------------------------|--------------|------------|")
    print("| Attached |-----------------Type-----------------| Serial No. |--- Version -|")
    print("|------------|----------------------------------|--------------|------------|")
    print("| %8s |- %30s |- %10d |- %8d |" % (device.isAttached(), \
        device.getDeviceName(), device.getSerialNum(), device.getDeviceVersion()))
    print("|------------|----------------------------------|--------------|------------|")
    print("Number of analog outputs: %i" % (device.getOutputCount()))
    print("Maximum output voltage: %d" % (device.getVoltageMax(0)))
    print("Minimum output voltage: %d" % (device.getVoltageMin(0)))
```

```
"""AnalogAttached
Event handler for attached phidgets (analog device).
"""
def AnalogAttached(e):
    attached = e.device
    print("Analog %i Attached!" % (attached.getSerialNum()))
    PHIDGETFILE.write("Analog %i Attached! \n" % (attached.getSerialNum()))

"""AnalogDetached
Event handler for detached phidgets (analog device).
"""
def AnalogDetached(e):
    detached = e.device
    print("Analog %i Detached!" % (detached.getSerialNum()))
    PHIDGETFILE.write("Analog %i Detached! \n" % (detached.getSerialNum()))

"""AnalogAttached
Event handler for phidget errors.
"""
def AnalogError(e):
    try:
        source = e.device
        print("Analog %i: Phidget Error %i: %s" % (source.getSerialNum(), e.eCode, \
            e.description))
    except PhidgetException as e:
        print("Phidget Exception %i: %s" % (e.code, e.details))

"""GetSerialNum
Function will find the number ("num") of phidgets requested, and return a sorted
list of their serial numbers in ascending order.
"""
def GetSerialNum(num):
        sn = [0]*num
        for x in range((num)):
                try:
                        DUMMY[x]=Analog()
                        DUMMY[x].openPhidget()
                        DUMMY[x].waitForAttach(1000)
                        sn[x] = DUMMY[x].getSerialNum()
                except RuntimeError as e:
                        print("Runtime Exception: %s" % e.details)
                        print("Could not append analog device to list.")
                        print("Exiting.")
                        exit(1)

        for x in range((num)):
                try:
                        DUMMY[x].closePhidget()
                except RuntimeError as e:
                        print("Runtime Exception: %s" % e.details)
                        print("Exiting.")
```

```
                exit(1)
        sn.sort()
        return sn

"""CreateAnalogObjects
Function will assign each attached phidget to its proper assignement based on
their serial numbers.
"""
def CreateAnalogObjects():
        lights = [None]*(numPhidgets-2)
        count = 0
        for x in range((numPhidgets)):
                if (SN[x]==ELEVATION_SN):
                        try:
                                elevation=Analog()
                        except RuntimeError as e:
                                print("Runtime Exception: %s" % e.details)
                                print("Exiting....")
                                exit(1)
                elif(SN[x]==AZIMUTH_SN):
                        try:
                                azimuth=Analog()
                        except RuntimeError as e:
                                print("Runtime Exception: %s" % e.details)
                                print("Exiting....")
                                exit(1)
                else:
                        try:
                                lights[count]=Analog()
                                count += 1
                        except RuntimeError as e:
                                print("Runtime Exception: %s" % e.details)
                                print("Exiting....")
                                exit(1)

        for x in lights:
                try:
                        SetOnHandlers(x)
                except PhidgetException as e:
                        print("Phidget Exception %i: %s" % (e.code, e.details))
                        print("Exiting....")
                        exit(1)
        try:
                SetOnHandlers(elevation)
                SetOnHandlers(azimuth)
        except PhidgetException as e:
                print("Phidget Exception %i: %s" % (e.code, e.details))
                print("Exiting....")
                exit(1)
        return lights, elevation, azimuth

"""SetOnHandlers
Sets necessary handlers on.
```

```
"""
def SetOnHandlers(device):
        device.setOnAttachHandler(AnalogAttached)
        device.setOnDetachHandler(AnalogDetached)
        device.setOnErrorhandler(AnalogError)

"""OpenPhidgets
Opens up each phigets for communication/interaction.
"""
def OpenPhidgets():
        cnt = 0
        for x in range(len(SN)):
                if (SN[x]==ELEVATION_SN):
                        try:
                                ELEVATION.openPhidget(ELEVATION_SN)
                        except PhidgetException as e:
                                print("Phidget Exception %i: %s" % (e.code, e.details))
                                print("Exiting....")
                                exit(1)
                elif(SN[x]==AZIMUTH_SN):
                        try:
                                AZIMUTH.openPhidget(AZIMUTH_SN)
                        except PhidgetException as e:
                                print("Phidget Exception %i: %s" % (e.code, e.details))
                                print("Exiting....")
                                exit(1)
                else:
                        try:
                                LIGHTS[cnt].openPhidget(SN[x])
                                cnt += 1
                        except PhidgetException as e:
                                print("Phidget Exception %i: %s" % (e.code, e.details))
                                print("Exiting....")
                                exit(1)
"""AttachPhidgets
Waits for each phidget to attach to the pc.
"""
def AttachPhidgets():
        cnt=0
        for x in range(len(SN)):
                if (SN[x]==ELEVATION_SN):
                        try:
                                ELEVATION.waitForAttach(1000)
                        except PhidgetException as e:
                                print("Phidget Exception %i: %s" % (e.code, e.details))
                                try:
                                        ELEVATION.closePhidget()
                                except PhidgetException as e:
                                        print("Phidget Exception %i: %s" % (e.code, \
                                            e.details))
                                        print("Exiting....")
                                        exit(1)
                                print("Exiting....")
```

```
                        exit(1)
                elif (SN[x]==AZIMUTH_SN):
                        try:
                                AZIMUTH.waitForAttach(1000)
                        except PhidgetException as e:
                                print("Phidget Exception %i: %s" % (e.code, e.details))
                                try:
                                        AZIMUTH.closePhidget()
                                except PhidgetException as e:
                                        print("Phidget Exception %i: %s" % (e.code, \
                                           e.details))
                                        print("Exiting....")
                                        exit(1)
                                print("Exiting....")
                                exit(1)
                else:
                        try:
                                LIGHTS[cnt].waitForAttach(1000)
                                cnt += 1
                        except PhidgetException as e:
                                print("Phidget Exception %i: %s" % (e.code, e.details))
                                try:
                                        LIGHTS[cnt].closePhidget()
                                except PhidgetException as e:
                                        print("Phidget Exception %i: %s" % (e.code, \
                                           e.details))
                                        print("Exiting....")
                                        exit(1)
                                print("Exiting....")
                                exit(1)

"""ClosePhigets
Closes the phidget object.
"""
def ClosePhidgets():
        for x in LIGHTS:
                try:
                        x.closePhidget()
                except PhidgetException as e:
                        print("Phidget Exception %i: %s" % (e.code, e.details))
                        print("Exiting....")
                        exit(1)
        try:
                AZIMUTH.closePhidget()
                ELEVATION.closePhidget()
        except PhidgetException as e:
                print("Phidget Exception %i: %s" % (e.code, e.details))
                print("Exiting....")
                exit(1)

"""EnableChan
Enables the output channels of the provided phidget ("device").
"""
```

```
def EnableChan(device):
        for x in range(4):
                device.setEnabled(x,True)
                device.setVoltage(x,0)

"""DisableChan
Disables the output channels of the provided phidget ("device").
"""
def DisableChan(device):
        for x in range(4):
                device.setVoltage(x,0)
                device.setEnabled(x,False)
"""Step
Steps a stepper motor by one step.
    device - the phidget operating the stepper motor.
    num (int) - the number of steps to take.
    direction - the direction to step: 0 for CW, 1 for CCW.
    sleeptime - the time to wait between each voltage pulse.
"""
def Step(device, num, direction, sleeptime):
        device.setVoltage(1,direction)
        for i in range(num):
                device.setVoltage(0,4)
                sleep(sleeptime)
                device.setVoltage(0,0)
                sleep(sleeptime)

"""On
Turns on all LED from the voltage file provided.
"""
def On():
        print "Turning on all LEDs.\n"
        count = 0
        for analog in LIGHTS:
                for x in range(4):
                        try:
                                analog.setVoltage(x, float(VOLTAGE_NEW[count]))
                                print("Voltage set to " + str(VOLTAGE_NEW[count]) + \
                                    " on LED #" + str(count+1))
                                count = count + 1
                        except PhidgetException as e:
                                print("Phidget Exception %i: %s" % (e.code, e.details))
                                print("Could not set voltage and/or enable LED #" + \
                                    str(count))
                                print("Exiting.")
                                exit(1)

"""Off
Turns off all LEDs.
"""
def Off():
        print "Turning off all LEDs.\n"
        for analog in LIGHTS:
```

```
                for x in range(4):
                        analog.setVoltage(x,0)

"""Update
Updates the voltage list with any changes made.
"""
def Update(voltages_new):
        savefile = raw_input("Name of new voltage file?")
        f = open("/tmp/%s.txt" % savefile, "w")
        f.write("\n".join(map(lambda x: str(x), voltages_new)))
        #f.close()
        #f = open('list2.txt','w')
        #f.write("\n".join(map(lambda x: str(x), old)))
        f.close()

"""MeasureFlux
Measures the flux from the photodiode.
"""
def MeasureFlux(PD):
        while True:
                PD.flushInput()
                line = PD.readline()
                line = line.strip()
                if (len(line) == 9 and line[4] == ','):
                        nums = line.split(',')
                        value = int(nums[0],16)
                        output = float(value)*5/32767
                        break
        return output

"""AskInt
Ask user for an interger.
"""
def AskInt(question):
        while True:
                try:
                        answer = int(raw_input(question))
                        break
                except:
                        print "Must be an integer. \n"
        return answer

"""AskFloat
Ask user for a float
"""
def AskFloat(question):
        while True:
                try:
                        answer = float(raw_input(question))
                        break
                except:
                        print "Must be an float. \n"
        return answer
```

```
######################### END OF USER-DEFINED FUNCTIONS ######################
############################ START OF MAIN PROGRAM ############################
#Initializing Variables
numPhidgets = 17
flag = 0
elevation_angle = 90
azimuth_angle = 0
state = 0
substate = 0
flux = 0
motor = 0
engage = 0
startTime = 0
lon_catch = False
lat_catch = True
catch = False

azimuth_range = 18     #number of AZIMUTH_STEPSIZE to step
elevation_range = 26    #numer of ELEVATION_STEPSIZE to step

photodiode = serial.Serial(port='/dev/ttyS0',baudrate=19200, stopbits=2)

#specify the stepper motor phidgets by their serial numbers
ELEVATION_SN = 276956
AZIMUTH_SN = 281316
LIGHTS = [None]*(numPhidgets-2)
DUMMY = [None]*(numPhidgets)


#Stepper motor calibration variables
AZIMUTH_CALIBRATION = 230       # 230 steps = 10 degrees
ELEVATION_CALIBRATION = 5       # 5 steps = 9 degrees
AZIMUTH_STEPSIZE = 10
ELEVATION_STEPSIZE = 9

#the voltage file
VOLTAGE_FILE = open("Voltages/%s.txt" % sys.argv[1], 'r')
VOLTAGE_OLD =    [line.strip() for line in VOLTAGE_FILE]
VOLTAGE_NEW = list(VOLTAGE_OLD)

#used for debugging
PHIDGETFILE = open("/tmp/%s.err" % sys.argv[1], "w")

#Setting up phidgets
SN = GetSerialNum(numPhidgets)
LIGHTS, ELEVATION, AZIMUTH = CreateAnalogObjects()
OpenPhidgets()
AttachPhidgets()

for x in LIGHTS:
        displayDeviceInfo(x)
```

```
displayDeviceInfo(ELEVATION)
displayDeviceInfo(AZIMUTH)

print("Welcome to the Isotropic Light Source Project.\n")

#enabling the channels on each phidget.
for x in LIGHTS:
        EnableChan(x)
EnableChan(AZIMUTH)
EnableChan(ELEVATION)

state = AskInt("Menu: \n\t1. Scan Source. \n\t2. Source Options \n\t3. Step Testing\
    \n\t4. Time Test \n\t0. Quit \n")

while (state != 0):

        #Complete scan function
        if (state == 1):
                catch = False
                name =  raw_input("Name of save file (without extension)")
                DATA = open("/tmp/%s.dat" % name, "w")
                density = AskInt("Density of Map? \n\t1. High \n\t2. Medium \n\t0. Low\n")
                if (density == 1):
                        AZIMUTH_CALIBRATION = 23       # 23 steps = 1 degree
                        ELEVATION_CALIBRATION = 1        #1 steps = 1.8 degrees
                        AZIMUTH_STEPSIZE = 1
                        ELEVATION_STEPSIZE = 1.8
                        azimuth_range = 180     #number of AZIMUTH_STEPSIZE to step
                        elevation_range = 130   #numer of ELEVATION_STEPSIZE to step
                if (density == 2):
                        AZIMUTH_CALIBRATION = 115       # 115 steps = 5 degree
                        ELEVATION_CALIBRATION = 3        #3 steps = 5.4 degrees
                        AZIMUTH_STEPSIZE = 5
                        ELEVATION_STEPSIZE = 5.4
                        azimuth_range = 36     #number of AZIMUTH_STEPSIZE to step
                        elevation_range = 44   #numer of ELEVATION_STEPSIZE to step
                else:
                        AZIMUTH_CALIBRATION = 230       # 230 steps = 10 degrees
                        ELEVATION_CALIBRATION = 5        #5 steps = 9 degrees
                        AZIMUTH_STEPSIZE = 10
                        ELEVATION_STEPSIZE = 9
                        azimuth_range = 18     #number of AZIMUTH_STEPSIZE to step
                        elevation_range = 26   #numer of ELEVATION_STEPSIZE to step

                PHIDGETFILE.write("##### START OF A SCAN #####\n")
                substate = AskInt("Does device need to be centered? \n1. Yes. \n0. No. \n")

                if (substate == 1):
                        print("Stepper motors not engaged, manually move photodiode to \
    home position. Press enter when set.\n")
                        AZIMUTH.setVoltage(2,4)
                        ELEVATION.setVoltage(2,4)
                        chr = sys.stdin.read(1)
```

```
                        AZIMUTH.setVoltage(2,0)
                        ELEVATION.setVoltage(2,0)
                        print("First center the azimuth travel.\n")

                        steps = AskInt("Take how many steps? (0 to continue, + for CW, \
 ---------------------- - for CCW)\n")

                        while (steps != 0):
                                if (steps > 0):
                                        Step(AZIMUTH,steps,0,0.1)
                                if (steps < 0):
                                        Step(AZIMUTH,abs(steps),4,0.1)

                                steps = AskInt("Take how many steps? (0 to continue, + \
 ------------------------------------ for CW, - for CCW)\n")


                        print("Azimuth travel calibrated! \n Now, center elevation \
 ---------------------- travel.\n")

                        steps = AskInt("Take how many steps? (0 to continue, + for CW, \
 ---------------------- - for CCW)\n")

                        while (steps != 0):
                                if (steps > 0):
                                        Step(ELEVATION,steps,0,0.1)
                                if (steps < 0):
                                        Step(ELEVATION,abs(steps),4,0.1)

                                steps = AskInt("Take how many steps? (0 to continue, \
 ---------------------------------- + for CW, - for CCW)\n")

                        print("Elevation travel calibrated!\n Centering: COMPLETE ... \
 ---------------------- Sending to first position.")

                        Step(ELEVATION,ELEVATION_CALIBRATION*(elevation_range/2),4,0.1)
                        substate = 0

                flag = 0
                azimuth_angle = -180
                elevation_angle = -((ELEVATION_STEPSIZE*(elevation_range/2))%90)

                print "Press Enter to begin scan."
                chr = sys.stdin.read(1)
                startTime = time()
                flux = MeasureFlux(photodiode)
                DATA.write("ELEVATION\tAZIMUTH\tFLUX\n")
                DATA.write("%.2f\t%.2f\t%.10f\n" % (elevation_angle, azimuth_angle, flux))
                print elevation_angle, azimuth_angle, flux
                for x in range(elevation_range):
                        for y in range(azimuth_range):
                                if (flag == 0):
                                        Step(AZIMUTH,AZIMUTH_CALIBRATION,4,0.01)
```

```
                        if (catch == False):
                                azimuth_angle = azimuth_angle + \
                                  AZIMUTH_STEPSIZE
                        else:
                                azimuth_angle = azimuth_angle - \
                                  AZIMUTH_STEPSIZE
                        sleep(1)
                        if (catch == False and azimuth_angle == 0):
                                continue
                        flux = MeasureFlux(photodiode)
                        DATA.write("%.2f\t%.2f\t%.10f\n" % \
                           (elevation_angle, azimuth_angle, flux))
                        print elevation_angle, azimuth_angle, flux
                if (flag == 1): ######step the other direction.######
                        Step(AZIMUTH,AZIMUTH_CALIBRATION,0,0.01)
                        if (catch == False):
                                azimuth_angle = azimuth_angle - \
                                  AZIMUTH_STEPSIZE
                        else:
                                azimuth_angle = azimuth_angle + \
                                  AZIMUTH_STEPSIZE
                        sleep(1)
                        if (catch == False and azimuth_angle == 0):
                                continue
                        flux = MeasureFlux(photodiode)
                        DATA.write("%.2f\t%.2f\t%.10f\n" % \
                           (elevation_angle, azimuth_angle, flux))
                        print elevation_angle, azimuth_angle, flux

if (flag==1):
        flag=0
else:
        flag=1


if (elevation_angle == 90 and flag == 0 and catch == False):
        print "Here"
        if (catch == False):
                catch = True
                azimuth_angle = 180
        flag = 1
        flux = MeasureFlux(photodiode)
        DATA.write("%.2f\t%.2f\t%.10f\n" % (elevation_angle, \
            azimuth_angle, flux))
        print elevation_angle, azimuth_angle, flux
        for y in range(azimuth_range):
                Step(AZIMUTH,AZIMUTH_CALIBRATION,4,0.01)
                azimuth_angle = azimuth_angle - AZIMUTH_STEPSIZE
                sleep(1)
                flux = MeasureFlux(photodiode)
                DATA.write("%.2f\t%.2f\t%.10f\n" % \
                   (elevation_angle, azimuth_angle, flux))
```

```
                        print elevation_angle, azimuth_angle, flux
                if (elevation_angle == 90 and flag == 1 and catch == False):
                        if (catch == False):
                                if (azimuth_angle == -180):
                                        catch = True
                                        azimuth_angle = 180
                                else:
                                        catch = True
                                        azimuth_angle = 0
                        flag = 0
                        flux = MeasureFlux(photodiode)
                        DATA.write("%.2f\t%.2f\t%.10f\n" % (elevation_angle, \
                            azimuth_angle, flux))
                        for y in range(azimuth_range):
                                Step(AZIMUTH,AZIMUTH_CALIBRATION,0,0.01)
                                azimuth_angle = azimuth_angle + AZIMUTH_STEPSIZE
                                sleep(1)
                                flux = MeasureFlux(photodiode)
                                DATA.write("%.2f\t%.2f\t%.10f\n" % \
                                   (elevation_angle, azimuth_angle, flux))
                                print elevation_angle, azimuth_angle, flux

                Step(ELEVATION,ELEVATION_CALIBRATION,0,0.5)
                if (catch == False):
                        elevation_angle = elevation_angle + ELEVATION_STEPSIZE
                else:
                        elevation_angle = elevation_angle - ELEVATION_STEPSIZE
                sleep(3)
                if (catch == False and azimuth_angle == 0):
                        continue
                flux = MeasureFlux(photodiode)
                DATA.write("%.2f\t%.2f\t%.10f\n" % (elevation_angle, \
                 azimuth_angle, flux))
                print elevation_angle, azimuth_angle, flux

        #last sweep through azimuth travel
        for y in range(azimuth_range):
                if (flag == 0):
                        Step(AZIMUTH,AZIMUTH_CALIBRATION,4,0.01)
                        if (catch == False):
                            azimuth_angle = azimuth_angle + AZIMUTH_STEPSIZE
                        else:
                            azimuth_angle = azimuth_angle - AZIMUTH_STEPSIZE
                        sleep(2)
                        flux = MeasureFlux(photodiode)
                        DATA.write("%.2f\t%.2f\t%.10f\n" % (elevation_angle, \
                            azimuth_angle, flux))
                        print elevation_angle, azimuth_angle, flux
                if (flag == 1): ######step the other direction.######
                        Step(AZIMUTH,AZIMUTH_CALIBRATION,0,0.01)
                        if (catch == False):
                            azimuth_angle = azimuth_angle - AZIMUTH_STEPSIZE
                        else:
```

```
                                azimuth_angle = azimuth_angle + AZIMUTH_STEPSIZE
                            sleep(2)
                            flux = MeasureFlux(photodiode)
                            DATA.write("%.2f\t%.2f\t%.10f\n" % (elevation_angle, \
                                azimuth_angle, flux))
                            print elevation_angle, azimuth_angle, flux
                DATA.close()

                ##This could depend on varible "density"
                print "Sending device back to first position."
                Step(ELEVATION,ELEVATION_CALIBRATION*(elevation_range),4,0.1)
                print ("Total elapsed time: %s seconds" % (time()-startTime))

                state = AskInt("Menu: \n\t1. Scan Source. \n\t2. Source Options \n\t3. \
                    Step Testing \n\t4. Time Testing \n0. Quit \n")

        #LED adjustment function
        if (state == 2):
                substate = AskInt("Source Options: \n\t1. Turn all LEDs on \n\t2. Turn \
                    all LEDs off. \n\t3. Update a voltage \n\t0. Back \n")

                while (substate != 0):
                        if (substate == 1):
                                On()
                        elif (substate == 2):
                                Off()
                        elif (substate == 3):
                                led = AskInt("Which LED would you like to update? (1-60)\
                                    \n")
                                voltage = AskFloat("Change voltage to? \n")

                                while (voltage > 8):
                                        voltage = AskFloat("Error - voltage must be < \
                                            8.0v - Change voltage to? \n")

                                if(led>=100):
                                        Off()
                                        led = (led/100)
                                        print "turning on device " + str(led)
                                        LIGHTS[led-1].setVoltage(0,voltage)
                                        LIGHTS[led-1].setVoltage(1,voltage)
                                        LIGHTS[led-1].setVoltage(2,voltage)
                                        LIGHTS[led-1].setVoltage(3,voltage)
                                else:
                                        #Off()
                                        print "device " + str(int((math.floor((led-1)/4)\
                                            ))+1) + " output " + str((led-1)%4) + "(LED " \
                                                + str(led) + ") set to: " + str(voltage) +\
                                                    "v\n"
                                        #selecting the correct LED
                                        LIGHTS[int((math.floor(abs((led-1)/4))))]\
                                            .setVoltage((((led-1)%4)), float(voltage))
                                        VOLTAGE_NEW[(led-1)] = voltage
```

```
                substate = AskInt("Source Options: \n\t1. Turn all LEDs on. \n\
\t2. Turn all LEDs off. \n\t3. Update a voltage. \n\t0. Back \n")

        state = AskInt("Menu: \n\t1. Scan Source. \n\t2. Source Options \n\t3. \
Step Testing \n\t4. Time Testing \n\t0. Quit \n")

    #Stepper motor testing function
    if (state == 3):
        substate = AskInt("Options: \n\t1. Step \n\t2. Enable/Disable \n\t0. \
Back \n")
        while (substate != 0):
            if (substate == 1):

                motor = AskInt("Which stepper motor would you like to \
work with? \n\t1. Azimuth \n\t2. Elevation \n")
                steps = AskInt("Take how many steps? (0 to continue, + \
for CW, - for CCW)\n")

                if (motor == 1):
                    AZIMUTH.setVoltage(2,0)
                if (motor == 2):
                    ELEVATION.setVoltage(2,0)

                while (steps != 0):
                    if (steps > 0 and motor == 1):
                        Step(AZIMUTH, steps,0,4.01)
                    if (steps < 0 and motor == 1):
                        Step(AZIMUTH, abs(steps),0,0.01)
                    if (steps > 0 and motor == 2):
                        Step(ELEVATION, steps,0,0.5)
                    if (steps < 0 and motor == 2):
                        Step(ELEVATION, abs(steps),4,0.5)

                    steps = AskInt("Take how many steps? (0 to quit, \
+ for CW, - for CCW)\n")

                substate = AskInt("Options: \n\t1. Step \n\t2. \
Enable/Disable \n\t0. Back \n")

            if (substate == 2):
                engage = AskInt("Stepper motor activation: \n\t1. \
Enable \n\t2. Disable \n\t0. Back")

                if (engage == 1):
                    AZIMUTH.setVoltage(2,0)
                    ELEVATION.setVoltage(2,0)
                if (engage == 2):
                    AZIMUTH.setVoltage(2,4)
                    ELEVATION.setVoltage(2,4)

                substate = AskInt("Options: \n\t1. Step \n\t2. \
Enable/Disable \n\t0. Back \n")
```

```
                state = AskInt("Menu: \n\t1. Scan Source. \n\t2. Source Options \n\t3. \
                    Step Testing \n\t4. Time Testing \n\t0. Quit \n")

        #time testing function.
        if (state == 4):
                name =  raw_input("Name of save file (without extension)")
                TIMEDATA = open("/tmp/%s.dat" % name, "w")
                TIMEDATA.write("TIME\tFLUX\n")
                delay =  AskInt("How many seconds between measurements? \n")
                nummeasurements = AskInt("How many measurements? \n")
                for i in range(nummeasurements):
                        flux = MeasureFlux(photodiode)
                        TIMEDATA.write("%i\t%.10f\n" % ((i)*delay, flux))
                        print i*delay, flux
                        sleep(delay)

                state = AskInt("1. Scan Source. \n2. Source Options \n3. Step Testing \
                    \n4. Time Testing \n0. Quit \n")


print("Press Enter to quit....")

chr = sys.stdin.read(1)

print("Closing...")

#closing all phidgets, photodiode, and files.
for x in LIGHTS:
        DisableChan(x)
DisableChan(AZIMUTH)
DisableChan(ELEVATION)
ClosePhidgets()
PHIDGETFILE.close()
photodiode.close()

#Saves voltages to new text file
ans = raw_input("Save voltages before quitting? (y/n)")
while ans not in ['n', 'N', 'no', 'No']:
        if ans.strip() in ['y','Y', 'Yes','yes']:
                Update(VOLTAGE_NEW)
                break
        ans = raw_input("Save voltages before quitting? (y/n)")

print("Done.")
```

The software code (written in python) for the toy Monte Carlo simulation was:

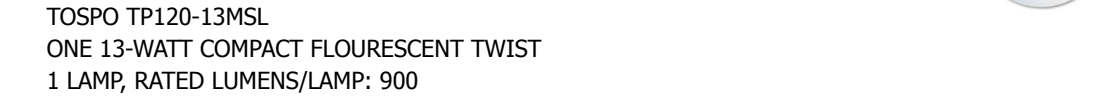
```
import matplotlib.pyplot as plt
import numpy as np

"""runThetaMonteCarloSimulation
Performs a ntrial theta 'measurements' via a Monte Carlo Simulation
"""
def runThetaMonteCarloSimulation(ntrials, std=1.53, limit=5):

    degrees = np.zeros(ntrials)

    for i in range(0,ntrials):

        degree = limit

        #reject samples outside the allowed region
        while degree >= limit or degree <=-limit:
            degree = np.random.normal(scale=std)

        degrees[i] = np.abs(degree)

    return degrees

n = 1000000
limit = 3.0
std = limit/2.575

theta = runThetaMonteCarloSimulation(n, std=std, limit=limit)

print theta.mean(), theta.std(), theta.min(), theta.max()

#plot histogram of points.
fig = plt.figure()

ax1 = fig.add_subplot(1, 1, 1)

plt.hist(theta, bins=30)
plt.xlabel("Theta (degrees)")
plt.ylabel("Count")
plt.title("Histogram of Simulated Theta Measurements")

plt.show()
```

Appendix B

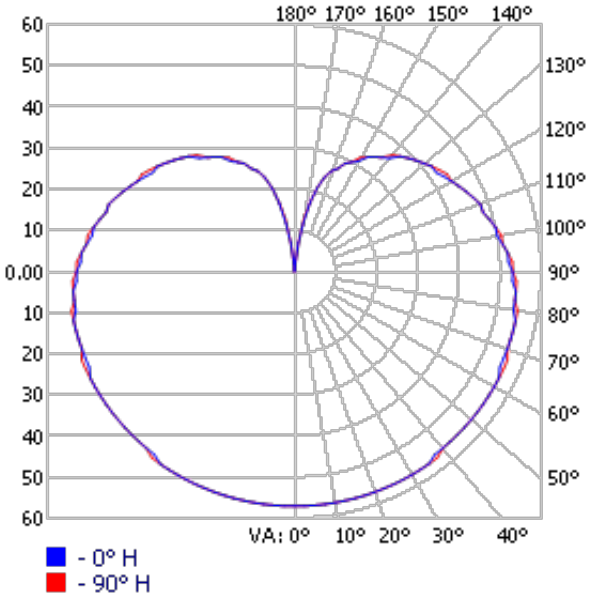
Diffusing Sphere Photometric Data

INDOOR PHOTOMETRIC REPORT
CATALOG: 11981 (13W TWIST)

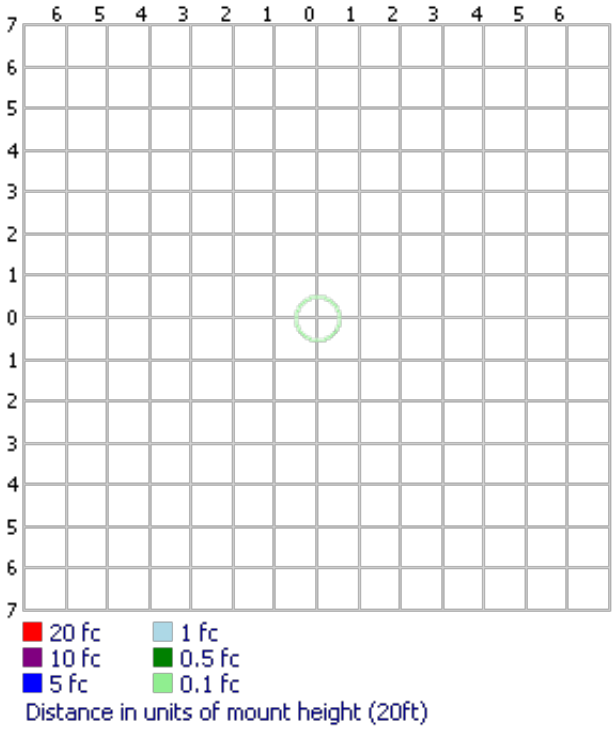
TEST #:	LTL20713
TEST LAB:	ACUITY BRANDS LIGHTING CONYERS LAB
TEST DATE:	12/24/2013
CATALOG:	11981 (13W TWIST)
DESCRIPTION:	GLOBE FLUSH MOUNT WITH WHITE GLASS DIFFUSER AND TWIST LAMP
SERIES:	GLOBE
LAMP CATALOG:	TOSPO TP120-13MSL
LAMP:	ONE 13-WATT COMPACT FLOURESCENT TWIST
LAMP OUTPUT:	1 LAMP, RATED LUMENS/LAMP: 900
BALLAST / DRIVER:	INTEGRATED LAMP/DRIVER
INPUT WATTAGE:	12.5
LUMINOUS OPENING:	VERTICAL CYLINDER (DIA : 6", H: 6")
CIE CLASS:	GENERAL DIFFUSE
MAX CD:	57.0 AT HORIZONTAL: 0°, VERTICAL: 0°
SPACING CRITERION:	@ 0 = 1.51 / @ 90 = 1.53
EFFICIENCY:	68.2%



Polar Candela Distribution



Isofootcandle Plot



ZONAL LUMEN SUMMARY

ZONE	LUMENS	% LAMP	% LUMINAIRE
0-30	48.0	5.3%	7.8%
0-40	83.7	9.3%	13.6%
0-60	177.3	19.7%	28.9%
60-90	172.0	19.1%	28%
70-100	173.3	19.3%	28.2%
90-120	154.2	17.1%	25.1%
0-90	349.3	38.8%	56.9%
90-180	264.2	29.4%	43.1%
0-180	613.5	68.2%	100%

LUMENS PER ZONE

ZONE	LUMENS	% TOTAL	ZONE	LUMENS	% TOTAL
0-10	5.4	0.9%	90-100	56.5	9.2%
10-20	16.2	2.6%	100-110	52.0	8.5%
20-30	26.4	4.3%	110-120	45.7	7.4%
30-40	35.7	5.8%	120-130	38.3	6.2%
40-50	43.4	7.1%	130-140	30.2	4.9%
50-60	50.2	8.2%	140-150	21.5	3.5%
60-70	55.3	9.0%	150-160	13.4	2.2%
70-80	58.1	9.5%	160-170	6.0	1%
80-90	58.7	9.6%	170-180	0.6	0.1%

AVERAGE LUMINANCE (CD/M2)

	0	22.5	45	67.5	90
0	3125	3125	3125	3125	3125
45	1910	1910	1910	1910	1910
55	1899	1899	1899	1899	1899
65	1912	1947	1947	1947	1947
75	2025	2025	2025	2025	2025
85	2184	2184	2184	2184	2184

COEFFICIENTS OF UTILIZATION - ZONAL CAVITY METHOD

EFFECTIVE FLOOR CAVITY REFLECTANCE: 20%

RCC %:	***80***				***70***				***50***			***30***			***10***			***0***
RW %:	70	50	30	0	70	50	30	0	50	30	20	50	30	20	50	30	20	0
RCR: 0	.74	.74	.74	.74	.69	.69	.69	.39	.59	.59	.59	.51	.51	.51	.43	.43	.43	.39
1	.65	.60	.56	.53	.60	.56	.52	.27	.47	.45	.42	.40	.38	.36	.33	.31	.30	.26
2	.58	.51	.45	.41	.53	.47	.42	.20	.40	.36	.33	.33	.30	.28	.27	.25	.23	.20
3	.52	.44	.38	.33	.48	.40	.35	.16	.34	.30	.26	.28	.25	.22	.23	.20	.18	.15
4	.47	.38	.32	.27	.43	.35	.30	.13	.30	.25	.22	.25	.21	.18	.20	.17	.15	.12
5	.43	.34	.27	.23	.39	.31	.25	.11	.26	.22	.18	.22	.18	.15	.18	.15	.13	.10
6	.39	.30	.24	.19	.36	.28	.22	.10	.24	.19	.15	.20	.16	.13	.16	.13	.11	.09
7	.36	.27	.21	.17	.33	.25	.19	.08	.21	.17	.13	.18	.14	.11	.14	.12	.09	.07
8	.34	.24	.18	.14	.31	.23	.17	.07	.19	.15	.12	.16	.13	.10	.13	.10	.08	.06
9	.31	.22	.17	.13	.29	.21	.15	.06	.18	.13	.10	.15	.11	.09	.12	.09	.07	.06
10	.29	.20	.15	.11	.27	.19	.14	.06	.16	.12	.09	.14	.10	.08	.11	.09	.07	.05

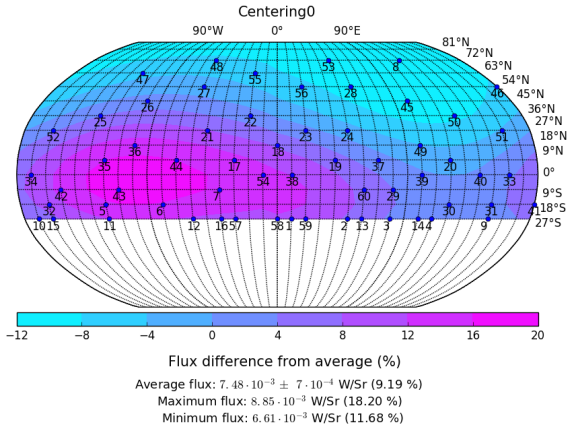
CANDELA TABLE - TYPE C

	0	22.5	45	67.5	90
0	**57**	**57**	**57**	**57**	**57**
5	**57**	**57**	**57**	**57**	**57**
10	**57**	**57**	**57**	**57**	**57**
15	**57**	**57**	**57**	**57**	**57**
20	**57**	**57**	**57**	**57**	**57**
25	**57**	**57**	**57**	**57**	**57**
30	**57**	**57**	**57**	**57**	**57**
35	**57**	**57**	**57**	**57**	**57**
40	56	56	56	56	56
45	56	56	56	56	56
50	56	56	56	56	56
55	56	56	56	56	56
60	56	56	56	56	56
65	55	56	56	56	56
70	55	55	55	55	55
75	55	55	55	55	55
80	54	54	54	55	55
85	54	54	54	54	54
90	53	53	53	53	53
95	52	52	52	52	52
100	50	50	51	51	51
105	49	49	49	49	49
110	48	48	48	48	48
115	46	46	46	46	46
120	44	44	44	44	44
125	42	43	43	43	43
130	41	41	41	41	41
135	39	39	39	39	39
140	36	36	37	37	37
145	34	34	34	34	34
150	31	31	32	32	32
155	29	29	29	29	29
160	26	26	26	26	26
165	21	21	21	21	21
170	12	13	12	13	13
175	2	2	2	2	2
180	0	0	0	0	0

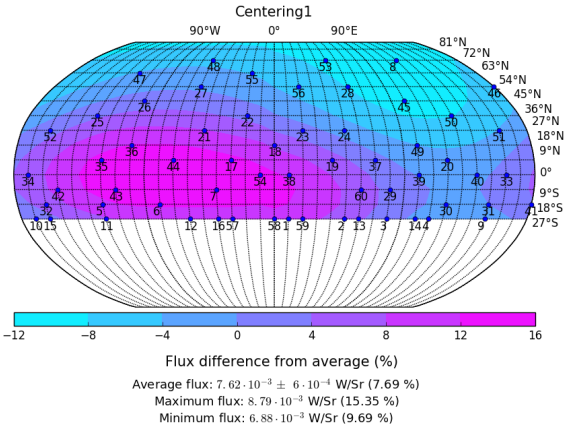
Appendix C

LED Ball Centering

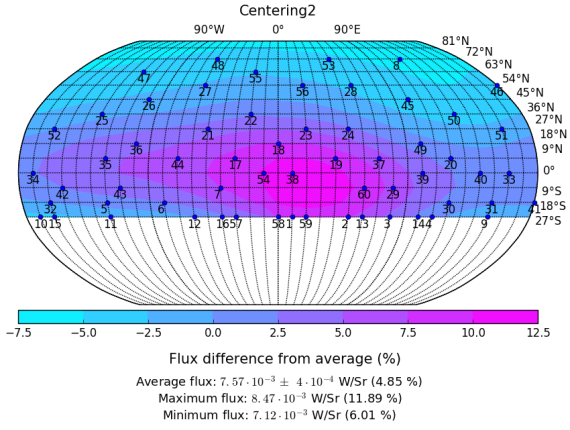
Once each LED was calibrated to a regulated output voltage of 0.30 V, we attempted to center the ball within the diffusing sphere. Without being able to measure the distance from surface to surface physically, the strategy used to center the LED ball was to make an isotropy map, look for a dipole in the data, and then slightly adjust the location of the LED ball to reduce the dipole. The LED ball was mounted atop a length of three plastic coated wire. These wires were used to relocate the LED ball to reduce the dipole. Fig. C.1 shows the plots of the centering process. The plots start with "$Centering0$" then "$Centering1$, $Centering2$, etc..." Fig. C.1(a) and C.1(b) show a clear dipole in the data (hot on one side, and cold on the opposing side). The flux domain increases from centering adjustment from Fig. C.1(c) to C.1(d), from $(-6.01\%, +11.89\%)$ to $(-9.17\%, +13.38\%)$, a good demonstration of the estimation required for each adjustment. In the final adjustment (Fig. C.1(f)), the flux domain did not show a significant change from the previous adjustment (Fig. C.1(e)), and the centering process was halted. Overall, the centering process improved the flux domain from $(-11.68\%, +18.20\%)$ to $(-9.20\%, +12.56\%)$, and improved the standard deviation from the mean from 9.19% to 4.50%. To improve on this process, it is imperative to provide a measurement of the relative location of the LED ball, with the ability to "undo" a given change easily. This way multiple iterations could be performed and the best setting could be selected for further development.



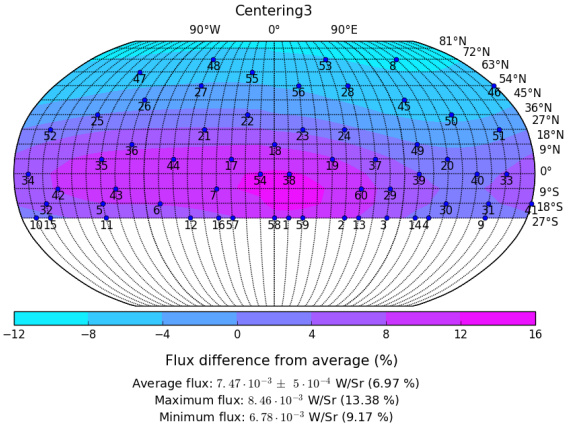
(a) Centering iteration #0 (Initial Measurement)



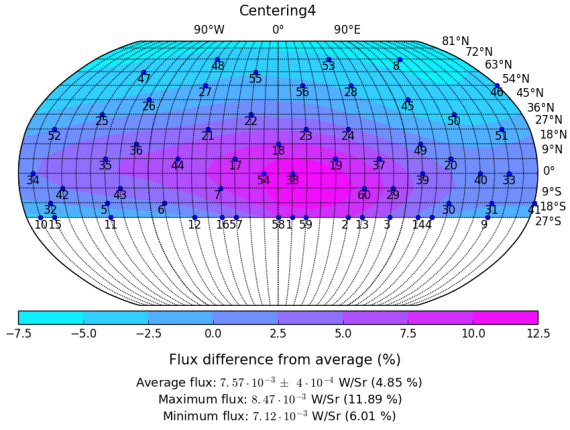
(b) Centering iteration #1



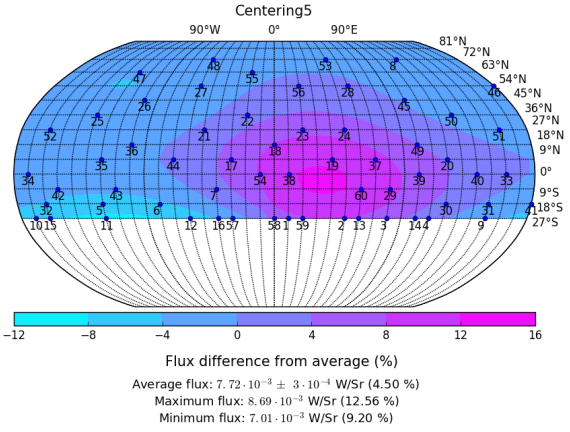
(c) Centering iteration #2



(d) Centering iteration #3



(e) Centering iteration #4



(f) Centering iteration #5

Figure C.1: Isotropy plots of LED ball centering.

References

[1] "U.S. Securities and Exchange Commission Form 8-k/a," X Rite Inc. Grandville, MI, USA, February, 1995. https://www.sec.gov/Archives/edgar/data/790818/0000926044-95-000006.txt. A Labsphere Audit, 1993-1994. Retrieved on June 18th 2016.

[2] R. Ulbricht, "Die bestimmung der mittleren raumlichen lichtintensitat durch nur eine messung," *ETZ* **21** (1900) 595; Sci. Abst., 3, abst. 2154 (1900); as cited in E. Rosa and A. Taylor, "Theory, construction, and use of the photometric integrating sphere," *Journal of the Franklin Institute* **195**, no. 1, (1923) 107–109.

[3] G. McKee, "Integrating spheres: Collecting and uniformly distributing light." http://www.photonics.com/EDU/Handbook.aspx?AID=25122. Retrieved on July 5th 2016.

[4] J. Lambert, *Photometrie: Photometria Sive de Mensura et Gradibus Luminis, Colorum et Umbrae*. W. Engelmann, Leipzig, Germany, 1760. https://books.google.ca/books?hl=en&lr=&id=6WpJAAAAYAAJ&oi=fnd&pg=PA3&dq=Photometria+Sive+de+Mensura+et+Gradibus+Luminis,+Colorum+et+Umbrae&ots=vPMpepRGw_&sig=c2K7UKgBW5K1b0lLE5BlzETEAWg#v=onepage&q=Photometria%20Sive%20de%20Mensura%20et%20Gradibus%20Luminis%2C%20Colorum%20et%20Umbrae&f=false. Retrieved on July 5th 2016.

[5] K. Carr, "Integrating sphere radiometry & photometry," technical guide, Labsphere Inc., North Sutton, NH, USA, 1997. https://www.labsphere.com/site/assets/files/2550/a-guide-to-integrating-sphere-radiometry-and-photometry.pdf. Retrieved on July 5th 2016.

[6] "Thermal insulation – heat transfer by radiation – physical quantities and definitions," standard, International Organization for Standardization, Geneva, Switzerland, 1989. http://www.iso.org/iso/catalogue_detail.htm?csnumber=16943. Retrieved on Sept. 24th 2015.

[7] W. Star and J. Marijnissen, "Calculating the response of isotropic light dosimetry probes as a function of the tissue refractive index," *Applied Optics* **28** no. 12, (1989) 2288–2291. https://www.osapublishing.org/ao/abstract.cfm?uri=ao-28-12-2288.

[8] J. Aguilar, A. Albert, P. Amram, M. Anghinolfi, G. Anton, S. Anvar, F. Ardellier-Desages, E. Aslanides, J. Aubert, R. Azoulay, *et al.*, "Transmission of light in the deep sea water at the site of the antares neutrino telescope," *Astroparticle Physics* **23** no. 1, (2005) 131–155. http://www.sciencedirect.com/science/article/pii/S0927650504001902.

[9] R. Maffione and J. Jaffe, "The average cosine due to an isotropic light source in the ocean," *Journal of Geophysical Research: Oceans* **100** no. C7, (1995) 13179–13192. http://dx.doi.org/10.1029/95JC00461.

[10] K.-C. Lee, J. Ho, and D. Kriegman, "Nine points of lights: Acquiring subspaces for face recognition under variable lightning," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.*

[11] J. Albert, "Satellite-mounted light sources as photometric calibration standards for ground-based telescopes," *The Astronomical Journal* **143** no. 1, (2012) 1–8. http://stacks.iop.org/1538-3881/143/i=1/a=8.

[12] C. Stubbs and J. Tonry, "Toward 1% photometry: End-to-end calibration of astronomical telescopes and detectors," *The Astrophysical Journal* **646** no. 2, (2006) 1436–1444. http://stacks.iop.org/0004-637X/646/i=2/a=1436.

[13] R. Brown, R. Honey, and R. Maffione, "Isotropic light source for underwater applications," in *Proc. SPIE*, vol. 1537 of *Underwater Imaging, Photography, and Visibility*, pp. 147–150. 1991. http://dx.doi.org/10.1117/12.48879.

[14] J. Marijnissen and W. Star, "Quantitative light dosimetry in vitro and in vivo," *Lasers in Medical Science* **2** no. 4, (1987) 235–242. http://link.springer.com/article/10.1007/BF02594166.

[15] J. Albert, "Airborne laser for telescopic atmospheric interference reduction." http://projectaltair.org/, 2012.

[16] B. Ryden, *Introduction to Cosmology.* Addison Wesley, San Francisco, USA, 1st ed., 2006.

[17] H. S. Leavitt and E. C. Pickering, "Periods of 25 variable stars in the small magellanic cloud.," *Harvard College Observatory Circular* **173** (1912) 1–3. http://articles.adsabs.harvard.edu/full/1912HarCi.173....1L/0000002.000.html?high=4dea8c437115295.

[18] M. Roos, *Introduction to Cosmology.* Wiley, Chichester, West Sussex, England Hoboken, NJ, 3rd ed., 2003.

[19] M. Hamuy *et al. Astron. J.* **106** (1993) 2392, as cited in L. Amendola and S. Tsujikawa, *Dark Energy: Theory and Observations.* Cambridge University Press, Cambridge, England, 1st ed., 2010.

[20] E. Linder, "Exploring the expansion history of the universe," *Phys. Rev. Lett.* **90** no. 9, (2003) 091301. http://link.aps.org/doi/10.1103/PhysRevLett.90.091301.

[21] A. Albrecht *et al.*, "Report of the dark energy task force," arXiv:0609591v1 [astro-ph]. http://arxiv.org/abs/astro-ph/0609591.

[22] J. Albert, M. Fagin, Y. Brown, C. Stubbs, N. Kuklev, and A. Conley, "Precision calibration via artificial light sources above the atmosphere," arXiv:1207.1938 [astro-ph.IM]. https://arxiv.org/abs/1207.1938.

[23] P. Garnavich *et al.*, "Constraints on cosmological models from hubble space telescope observations of high-z supernovae," *The Astrophysical Journal Letters* **493** no. 2, (1998) L53. http://iopscience.iop.org/article/10.1086/311140/meta.

[24] Lite-On Electronics, Inc., *LTST-T670TBKT Datasheet*. http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTST-T670TBKT.pdf. Retrieved on Feb 6th 2015.

[25] Lithonia Lighting, *Indoor Decorative Globe Flush Mount*. http://www.acuitybrandslighting.com/library/ll/documents/specsheets/glof.pdf. Retrieved on Sept 15th 2016.

[26] Lithonia Lighting, *Visual Photometric Tool*. http://www.visual-3d.com/tools/photometricViewer/default.aspx?id=39602. Retrieved on Sept 15th 2016.

[27] K. Youakim, "Developing and testing a solid state isotropic light source for use on the altair project," 2014. An undergraduate thesis.

[28] Hamamatsu: Solid State Devision, *Photosensor Amplifier Type No. C9329 OPERATION MANUAL*, 2016. https://www.hamamatsu.com/resources/pdf/ssd/c9329_kacc1100e.pdf. Retrieved on Aug. 2010.

[29] CVI: Melles Griot, *Introduction to Laser Beam and Spectral Measurement*, 2009. http://www.astro.caltech.edu/~lah/ay105/pdf/e1-photodiode.pdf. Retrieved on Nov 15th 2016.

[30] NIST Report of Calibration (NIST Test #39077C – Spectral Responsivity) for Hamamatsu Photodiode S2281, S/N 8F097, issued Feb. 10, 2010 to Dr. J. Albert.