University of Nevada, Reno

Development of a multispectral albedometer and deployment on an unmanned aircraft for evaluating satellite retrieved surface reflectance over Nevada's Black Rock Desert

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Atmospheric Science

by

Jayne M. Boehmler

Dr. W. Patrick Arnott/Thesis Advisor

May, 2018

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

JAYNE BOEHMLER

Entitled

Development Of A Multispectral Albedometer And Deployment On An Unmanned Aircraft For Evaluating Satellite Retrieved Surface Reflectance Over Nevada's Black Rock Desert

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

W. Patrick Arnott, Advisor

James C. Barnard, Committee Member

Kristin A. Lewis, Committee Member

Wendy M. Calvin, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

May, 2018

Abstract

Accurate atmospheric aerosol characteristics derived from satellite measurements are needed over a variety of land surfaces. Inhomogeneous and bright surface reflectance across California and Nevada may be a contributing factor in the discrepancies observed between ground based and satellite-retrieved atmospheric aerosol optical depth (AOD). We developed and deployed a compact and portable instrument to measure albedo to evaluate a major factor that influences the accuracy of AOD retrievals. The instrument functions as a spectral albedometer using two Hamamatsu micro-spectrometers with a spectral range from 340 –780 nm for measuring incident and reflected solar radiation at the surface. The instrument was operated on an unmanned aircraft system (UAS) to control areal averaging for comparison with satellite derived albedo from NASA Moderate Resolution Imaging Spectroradiometer (MODIS) and Land Satellite 7 Enhanced Thematic Mapper Plus (Landsat-7 ETM+). The instrument was deployed on October 5th, 2017 under clear skies over Nevada's Black Rock Desert to investigate a region of known high surface reflectance. It was found that satellite retrieved surface reflectance underestimated measured surface albedo at this location, indicating the need for more albedo measurements to validate satellite retrievals over areas of complex terrain in the Western U.S. This study demonstrates the viability of obtaining hyperspectral surface albedo measurements via UAS as an intermediary between fixed-point ground measurement and space-borne observations.

Acknowledgments

I first and foremost wish to thank my advisor, Dr. W. Patrick Arnott for his personal investment of time, energy and belief in this project. Thank you for your encouragement to participate in workshops and attend conferences, involving me in contracted projects, and allowing me the freedom to set my own schedule. My deepest appreciation to Chris Stevens who was heavily involved with the early stages of the instrument development as well as Michael Filicchia. Dr. Adam Watts, for his expert piloting skills and willingness to let us use his aircraft. My committee members for their time and commitment. S. Marcela Loria-Salazar for her guidance and unwavering support of me. All members of Dr. Arnott's research group, thank you for your comradery and assistance with testing of the instrument. Grant Francis and Marco Giordano for their help with the field expedition to the Black Rock Desert. Dr. Heather Holmes and her research group for their valuable feedback and weekly meetings to discuss each other's research. Funding for this project was supported through the Clemons-Magee Professorship of my advisor. Thank you to the UNR Physics Department for their resources and use of their facilities; to Adafruit for their open source hardware and software; and to NASA for providing publicly accessible satellite data for MODIS and Landsat. Lastly, my family and friends for putting up with all of my nonsense.

Table of Contents

List of Figures

Chapter 1 Introduction

1.1 Background

Atmospheric processes are driven by the global distribution of solar energy absorbed and reflected by the Earth's surface. The amount of energy that the Earth absorbs or reflects over a given area is dependent on surface cover. Albedo, an important driver of the Earth's climate system, is a measure of surface reflectivity. The Earth's radiative balance can be affected by small changes in albedo such as those due to land use change, deforestation, fires, snow and ice cover. The Earth's average global albedo is being affected by anthropogenic activities such as urbanization, and the presence of aerosols in the atmosphere which can be deposited onto snow (Schmitt et al., 2015). Accurate measurements of surface albedo are needed for understanding the climatological ramifications of land use change, reducing uncertainties in global climate models (Brovkin et al., 2013), and improving satellite retrievals of aerosol optical depth (AOD) (Zhang et al., 2016). However, the effects of albedo changes on global radiative forcing are still highly uncertain due to the wide range of estimates of anthropogenic and natural land cover change (Myhre, Bréon, Aamaas, & Jacob, n.d.). More comprehensive methods for accurately measuring regional albedo over time are needed.

Quantifying albedo is made complicated because it varies in both space and time (Jonsell, Hock, & Holmgren, 2003) and it is highly dependent on solar zenith angle. It is a dimensionless quantity that can be defined as the ratio of the solar irradiance reflected from the Earth's surface to that which is incident upon it (He, Liang, & Song, 2014).

Albedo is measured as the hemispherical reflectivity of a surface as a function of wavelength (Taha, 1997); therefore, an ideal albedo measuring device must have a good cosine response and be multispectral. Broadband albedo ground measuring devices, such as pyranometers, are widely used in the field yet only provide a single average measurement across a wide spectral range. Global networks, such as NOAA's Surface Radiation (SurfRad) network and the Department of Energy's Atmospheric Radiation Measurement (ARM) network utilize broadband pyranometers and narrowband radiometers on fixed towers for measuring albedo but are limited by the spatial footprint they can sense. The limitations of sparsely distributed broadband albedo measurements at the surface are reduced by the use of satellites to estimate surface albedo over areas where towers are not present. Yet, satellites used for measuring atmospheric parameters have their own shortcomings.

Earth observing satellites are capable of providing global coverage of surface albedo; however, accurately estimating albedo from space-borne platforms can be challenging due to the variability in spatial and temporal conditions of the surface and atmosphere. Additionally, there are not enough ground-based measurements to evaluate satellite retrievals, and spatial interpolation of existing ground-based measurements may not fully represent local or even regional areas. Acquiring satellite-derived measurements with high accuracy can be especially challenging over regions of complex terrain and in semi-arid environments such as the Western U.S. (Loría-Salazar et al., 2016; Sorek-Hamer et al., 2015).

Loria-Salazar et al. (2016) found that satellite retrievals of aerosol optical depth (AOD) can be affected by underlying high surface reflectance; specifically, Zhang et al. (2016), report that an error of 0.01 in estimated surface reflectance has been shown to translate to an error of 0.1 in satellite retrieved AOD. This is a particularly strong problem in the Western U.S., and it hinders the ability to ground truth columnar aerosol estimations and surface reflectance observations over these areas.

Unmanned Aircraft Systems (UAS) and other small aircraft observations have the potential to provide cost-effective, low-altitude columnar and surface measurements for atmospheric science applications. They can provide optical observations of the surface with greater accuracy than conventional high-altitude satellites and manned airplanes due to the reduced effect of atmospheric extinction and higher spatial resolution (Uto, Seki, Saito, Kosugi, & Komatsu, 2016). Reductions in sensor size has resulted in smaller and more robust instrumentation capable of flying onboard small unmanned aircrafts. Additionally, smaller instruments for UAS are designed to be portable in contrast to existing commercial pyranometers which are designed to be stationary. Other handheld field spectroradiometers are often bulky, expensive and require additional equipment to operate. We address the need for more portable ground-based measurements for evaluating satellite retrievals over areas of known high surface reflectance through the development of a novel multispectral albedometer for measuring hemispherical albedo. We introduce a new method for obtaining albedo measurements, through a unique instrument design which enables the instrument to be flown on UAS.

1.2 Satellite Platforms

Moderate Resolution Imaging Spectroradiometer (MODIS) is an instrument onboard NASA's Terra and Aqua satellites which collects global atmospheric measurements twice daily, once in the morning (Terra) and again in the afternoon (Aqua). The MOD09GQ/MYD09GQ surface reflectance product for Terra and Aqua satellites, respectively, is obtained after atmospherically correcting top of the atmosphere (TOA) radiance measurements (E. F. Vermote et al., 2002). MODIS obtains TOA radiance measurements over 36 wavelength bands, of which 7 bands are corrected to provide surface albedo [bands 1 (620-670 nm), 2 (841-876 nm), 3 (459-479 nm), 4 (545-565 nm), 5 (1230-1250 nm), 6 (1628-1652 nm), and 7 (2105-2155 nm)] (E. F. Vermote, Roger, & Ray, 2015). In addition, MODIS provides a surface albedo 8-day best product (MOD09Q1) which is a composite of MOD09GQ and contains the best possible observation over an 8-day period selected based on high observation coverage, low view angle, the absence of clouds or cloud shadow, and aerosol loading.

Similar to MODIS, NASA's land satellite 7 enhanced thematic mapper plus (Landsat 7 ETM+) obtains surface albedo values at 8 spectral bands with higher spatial resolution than MODIS (30 m compared to 250 m – 1 km, respectively). Landsat7 ETM+ spectral coverage ranges from visible to infrared with specific band designations from 450-520 nm, 520-600 nm, 630-690 nm, 770-900 nm, 1550-1750 nm, 2090-2350 nm, and 10400-12500 nm for bands 1-7, respectively and an additional panchromatic band from 520-900 nm (Barsi, Lee, Kvaran, Markham, & Pedelty, 2014).

The presence of gases in the atmosphere absorb and scatter both incoming and reflected sunlight. To account for the alteration in observed radiant energy caused by these gases it is necessary to apply an atmospheric correction to the observed top of the atmosphere radiance. Information on gaseous concentrations present in the atmosphere between the observing satellite and the Earth's surface at the time of measurement is gathered through other ground and satellite observations; Atmospheric inputs of ozone and pressure are acquired from the National Centers for Environmental Prediction (NCEP) while aerosol and water vapor are derived directly from MODIS ("Modis Land Surface Reflectance - Home," n.d.). The atmospheric corrections used by Landsat and MODIS to derive surface reflectance are similar. Atmospheric conditions such as water vapor, ozone, geopotential height, aerosol optical thickness, and digital elevation are input along with Landsat data into the "Second Simulation of a Satellite Signal in the Solar Spectrum" (6S) radiative transfer models to generate the surface reflectance product (USGS, 2018). The MODIS atmospheric correction assumes a Lambertian surface and adjusts the atmospheric correction algorithm for non-Lambertian surfaces as well as for heterogeneous landscapes.

The surface albedo products are meant to represent surface conditions as they would be if the measurement were made at the ground surface (E. F. Vermote et al., 1997) and therefore, it is fair to expect that the albedo measurements obtained using the instrument developed in this study should agree with satellite values. Previous studies have evaluated satellite surface albedo using ground-based networks as well as inter-

comparisons between space-borne instruments. (Claverie, Vermote, Franch, & Masek, 2015; Liu et al., 2017; Pinty et al., 2011). Generally, MODIS satellite and ground-based measurements of albedo agree over vegetated landscapes (Heikkinen et al., 2007; Maiersperger et al., 2013; Mira et al., 2015). We aim to provide insight for comparison over complex, semi-arid desert terrain using unmanned aircraft.

1.3 State of the Art

Broadband albedometers are available and widely used by the scientific community. Broadband albedometers and pyranometers can be used for validating broadband albedo products; however, spectral albedo products are rarely validated using ground measurements (Zhou, Wang, & Liang, 2018). Typical high resolution spectral measuring devices like the Analytical Spectral Device (ASD) Field spectroradiometer are only able to sense over a small area. The micro-spectrometers utilized in our albedometer allow it to be a portable and an inexpensive alternative to other hyperspectral radiometers for spectral albedo validation studies. The albedometer developed in this work also features a suite of additional sensors to measure pressure, humidity, surface and air temperature, track position and altitude, record tilt, take photos, and communicate via radio, making it an all-in-one albedo measuring device.

Previous work to develop albedometers has consisted mainly of long poles with a commercial pyranometer or spectrometer attached to the end to be used for surveying on foot (van der Hage, 1992). This technique is inconveniently heavy and often requires

extra equipment to operate and data log. These studies also lack the ability to mount the instrument on an unmanned aircraft system. Although, albedometers have been previously mounted onto small planes (Coddington et al., 2008; Wendisch et al., 2001), fewer studies to measure albedo from UAS have been performed. Uto et al. (2016) developed a low-cost hyperspectral whiskbroom imager for UAS applications using similar Hamamatsu micro-spectrometers to those used in the present work. Kipp & Zonen advertise one study by Goodale & Fahey Labs at Cornell University who mounted CMP6 and CMP3 pyranometers to a UAS ("Novel Estimation of Albedo Using a Drone Pyranometer - Kipp & Zonen," n.d.). In this application the airborne pyranometer faced downward to measure reflected solar radiation and a nearby tower equipped with an additional pyranometer was used for measurements of downwelling solar radiation. This process not only limits the study location to be near a fixed tower but the downwelling solar radiation reaching the surface may vary from the location of the tower to the location of the UAS. In this case, albedo can only be obtained during post processing of the data whereas, the albedometer we have developed obtains incoming and reflected radiation simultaneously at the same location and displays albedo values in real time through radio communication to a handheld ground control device.

This thesis is organized as follows. Chapter 2 and 3 describe the design and development of the albedometer. Chapter 4 details the deployment of the instrument to Nevada's Black Rock Desert (BRD). Chapter 5 presents the results of our albedo measurements from the BRD campaign and discusses comparison to satellite retrieved values. Chapter 6 includes

our thoughts and limitations of the comparison study as well as future developments and applications for the instrument.

Chapter 2 System Specifications

The instrument consists of two parts. The first part is a measuring device mounted to the aircraft (~300 g) which houses two micro-spectrometers and six additional sensors (figures 1.a, 1.b, and 1.c). The second part is a handheld display and ground control device (~ 133 g) to initiate collection and display real-time data from the measuring device (figure 1.d). The measuring device is enclosed in a 3D-printed polylactic acid (PLA) casing with a custom mount built-in to the design of the box (figure 1.c). Both parts are powered by 9V batteries and can operate for multiple hours. *Teensy3.6* and *3.2* microcontrollers are used to control signal processing for each part, respectively ("Teensy USB Development Board," n.d.). The *Teensy3.6* microcontroller has a built-in real-time clock with battery backup capability for time and date. To measure albedo, two micro-spectrometers manufactured by Hamamatsu Photonics, each with a spectral range of 340 –780 nm, are utilized; one for obtaining the downwelling solar radiation and the other for measuring the solar radiation reflected from the surface. Albedo values range from 0 to 1 and are calculated as the ratio between reflected light from the surface (downward facing spectrometer) and incident light (upward facing spectrometer) after first subtracting out the dark counts, dividing by the integration time, and applying a transfer function (equation 1). Equations for modeling the dark current and deriving the transfer function are further discussed in Chapter 3.

$$Albedo\ (\lambda) = \frac{Downward\ facing\ spectrometer}{Upward\ facing\ spectrometer}.$$

(eq. 1)

The uncertainty for albedo measurements is also calculated with every measurement for each wavelength (equation 2). $Spec_1$ and $Spec_2$ are the counts (with dark counts subtracted) from the upward facing and downward facing spectrometers, respectively.

$$Uncertainty\ (\lambda) = Albedo * 0.5 * \sqrt{\frac{1}{Spec_1} + \frac{1}{Spec_2}}.$$

(eq. 2)

Additional components on the instrument include a GPS for position, altitude, and time; a digital level and compass for measuring instrument orientation; temperature, pressure and humidity sensors; an infrared sensor to measure ground temperature; a camera for measuring sky conditions; a radio for two-way communication between the devices; and a micro SD card for recording data (figure 2). Specific connections for all components in the design of the albedometer are shown in the circuit board schematic in Appendix 1 and the code used to run the instrument is provided in Appendix 2.

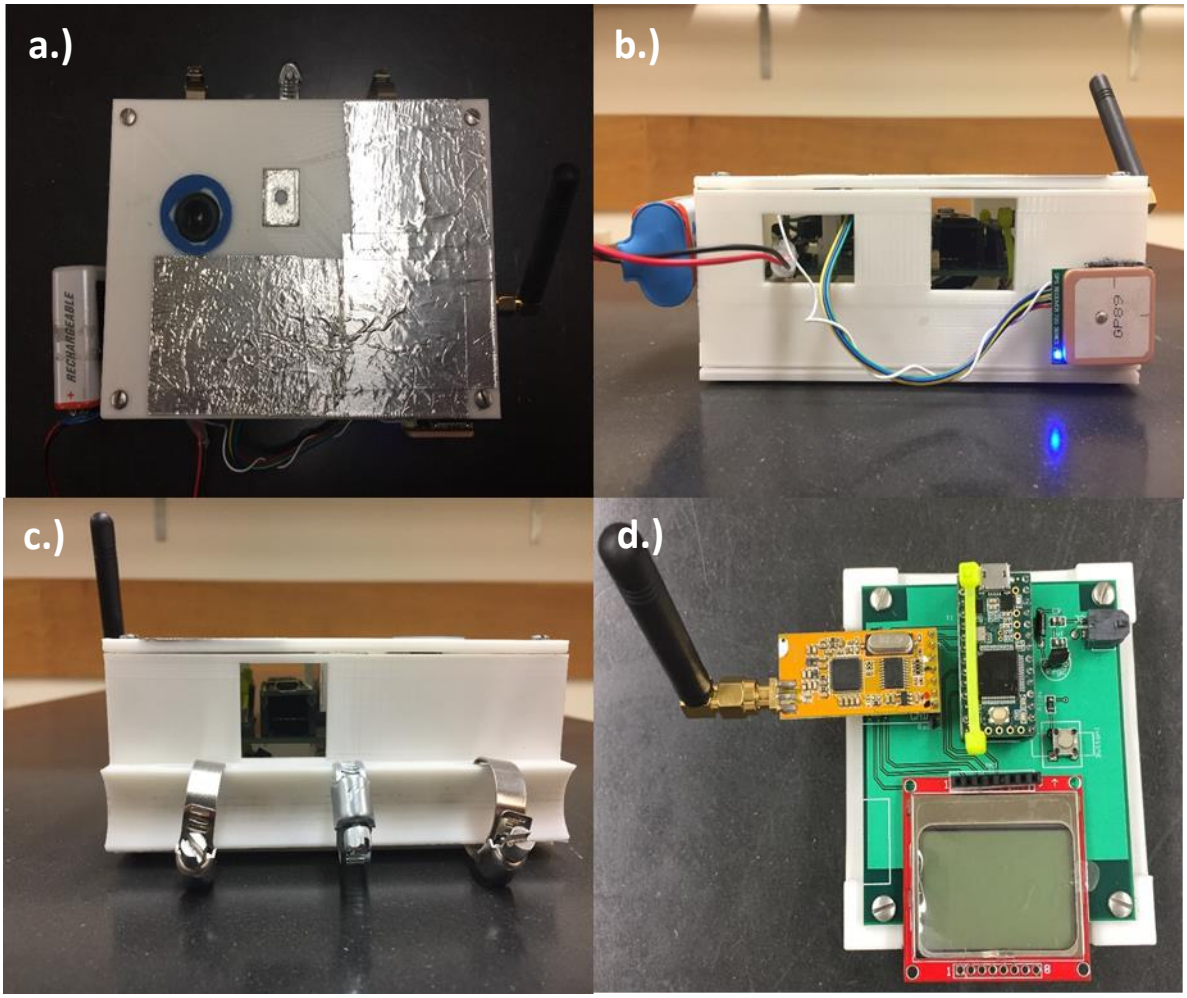


Figure 1: Albedometer design. 1.a. Top view of measuring device showing upward facing spectrometer and camera. Aluminum tape was added to maintain cool temperatures inside the box and a UV/IR filter was placed over the camera to capture more natural looking images. 1.b. Side view of measuring device showing the GPS and 9V battery which sit outside of the box. 1.c. Side view of measuring device showing the custom 3D-printed mount built-in to the box. 1.d. Ground control device showing radio for communicating to the measuring device, a button for initiating measurements, a screen for printing resulting albedo in real-time, and the Teensy 3.2 microcontroller.

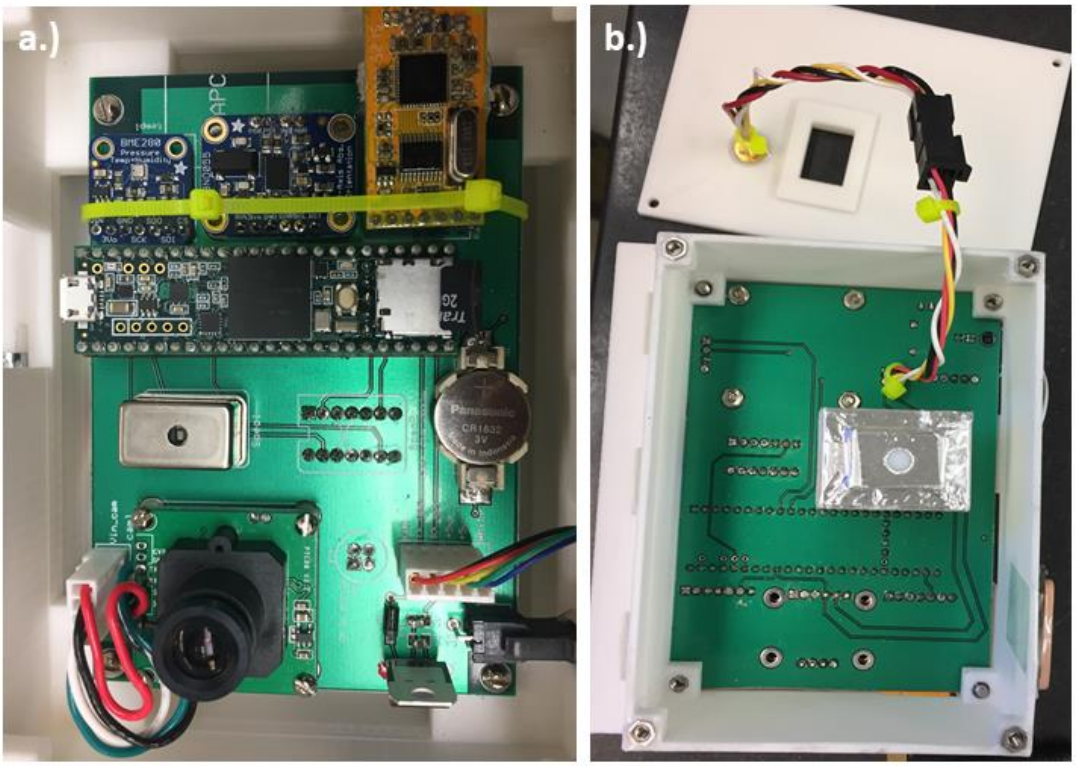


Figure 2: Albedometer components. 2.a. Top view of printed circuit board including components (from left to right, top to bottom): BME280 temperature sensor, BNO055 absolute orient, APC220 radio, Teensy 3.6 microcontroller, C12666MA micro spectrometer, Back-up battery, VC0706 camera, UBX-G7020 GPS. 2.b. Bottom view of printed circuit board including components: C12666MA spectrometer with diffuser, and MLX90614 Infra-Red (IR) sensor.

2.1 Components

2.1.1 C12666MA Micro-spectrometer

The Hamamatsu micro-spectrometers used for obtaining albedo feature an ultra-compact design with size dimensions 20.1 x 12.5 x 10.1 mm and mass of 5 g. The manufacturer specifications indicate a spectral range from 340 to 780 nm and a spectral resolution of 15 nm. In this application we only considered 400 to 750 nm due to low counts below 400 nm.

2.1.2 Teensy 3.6/3.2 microcontroller

A 32-bit, 180MHz ARM processer controls the functionality of the system. The microcontroller performs analog to digital conversion with a 13-bit read resolution and is programmed using the Arduino integrated development environment.

2.1.3 BME 280 pressure, temperature, & humidity sensor

Digital readings of pressure, temperature and humidity were obtained in conjunction with every observation. The Bosch sensor is able to measure conditions within the control box with a response time of 1 s and was incorporated into the instrument design using the inter-integrated circuit (I2C) interface. The pressure and temperature measurements were useful for determining the height of the UAS above the surface.

2.1.4 BNO055 absolute orientation

The Bosch absolute orientation sensor was used to measure the tilt angle of the instrument relative to the vertical coordinate. The level reading was used as a data qualifier. Only measurements obtained when the aircraft was within a 5-degree offset in the x- and y- horizontal directions were used in our analysis.

2.1.5 VC0706 TTL Serial Camera

The onboard camera developed by Adafruit Industries was used to document the sky conditions at the time of measurement. The images were saved and serve as an additional

data qualifier for properties of the radiation field at the time of measurement, e.g., detecting cloud cover.

2.1.6 UBX-G7020 GPS

The geographic position, altitude and time of each observation is obtained and recorded with every measurement. This information was used for geo-referencing satellite albedo measurements and to verify the height above ground level of each measurement. The GPS time was used in addition to the real time clock on the Teensy.

2.1.7 APC220 radio

Radios on both devices were used to establish two-way communication between the payload and the ground control unit. The ground control unit is used to initiate a measurement. the measuring device onboard the UAS sends measurements to the ground for display to evaluate operations in near real time.

2.1.8 Nokia Screen

Periodic updates of each measurement were printed to a screen on the hand-held unit and resulting albedo was plotted after each measurement.

2.1.9 MLX90614 Infra-Red (IR) sensor

The onboard IR thermometer faces downward to capture noncontact measurements of surface temperature with a temperature range from -70° to 380°C and a temperature accuracy of ±0.5°C. The detector has a field of view of approximately 100° with a peak

zone around 0° where the measured value is the average temperature of all objects in the field of view.

2.1.10 SD card

All data parameters and camera images were saved to a 2-GB SD card. The SD port is built into the Teensy 3.6 microcontroller.

2.1.11 Real-time clock

The real-time clock is built-in to the Teensy 3.6 microcontroller and was used for recording the time at which each measurement was taken. It is manually set once upon installation and reports both time and date.

Chapter 3 System Calibration and Testing

3.1 Diffuser Transmissivity

In order to control the amount of solar radiation entering the detector, diffusers were 3D printed and fitted to each spectrometer. The diffusers were characterized for their transmissivity, angular response, and fluorescence. The transmissivity of the 3D-printed diffusers was tested using an Ocean Optics HR2000 spectrometer. Overall the diffusers allow 0.1% of light through and even less below 400 nm (figure 3). It was found that below 400 nm the diffusers let in very little light and because of this our study only focuses on 400 nm and above. Additional motivation for characterizing the transmissivity of the 3D-printed diffusers was to check for any unwanted fluorescence. It was found that certain types of PLA fluoresced, however the final PLA diffusers used in the instrument design showed no signs of fluorescence. This experiment demonstrated that the PLA diffuser spectral response was similar to commonly used PTFE diffusers.

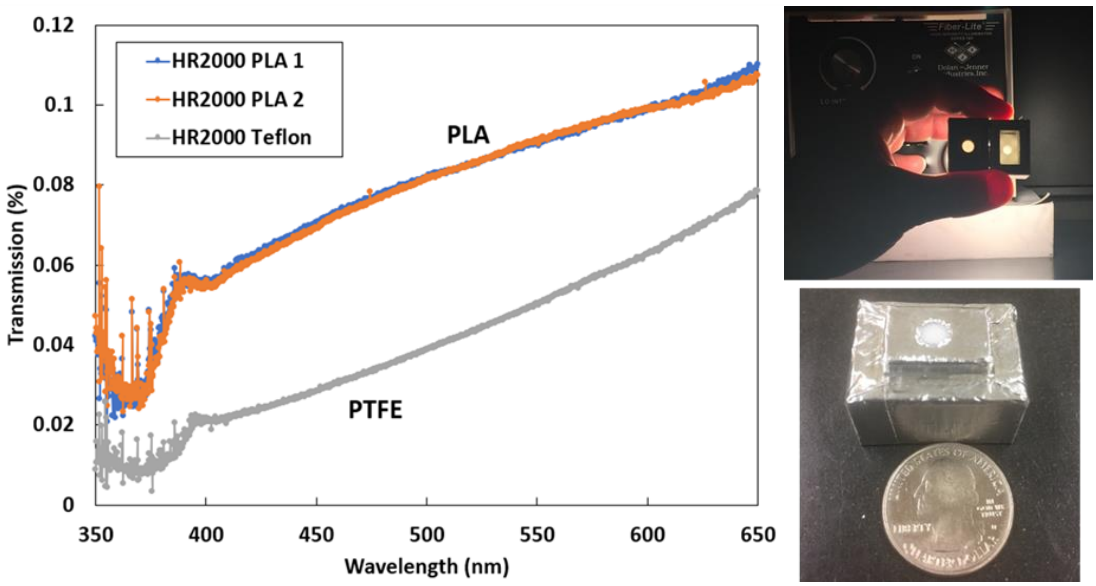


Figure 3: Transmissivity of PLA and Teflon diffusers using Ocean Optics HR2000 spectrometer. Both types of diffusers allowed very little light through (<1%). PLA was incorporated into the instrument design over PTFE due to the fact that it allowed slightly more light through. The transmissivity decreases rapidly below 400 nm and for this reason we chose to limit the spectral range of our results to 400 nm.

3.2 Angular Response

In order to be considered a proper solar irradiance detector, the instrument should have a response that scales with the cosine of the zenith angle. An experiment to test the cosine weighting of the instrument was performed using a light source and a lens. The set up for testing the cosine response of the detector is shown in figure 4. The onboard absolute orientation sensor was used to record the zenith angle and measurements were taken every few degrees through controlled tilting of the instrument. Overall, the instrument has a good cosine response which is made apparent by the cosine curve generated by plotting the detector counts as a function of zenith angle (figure 4).

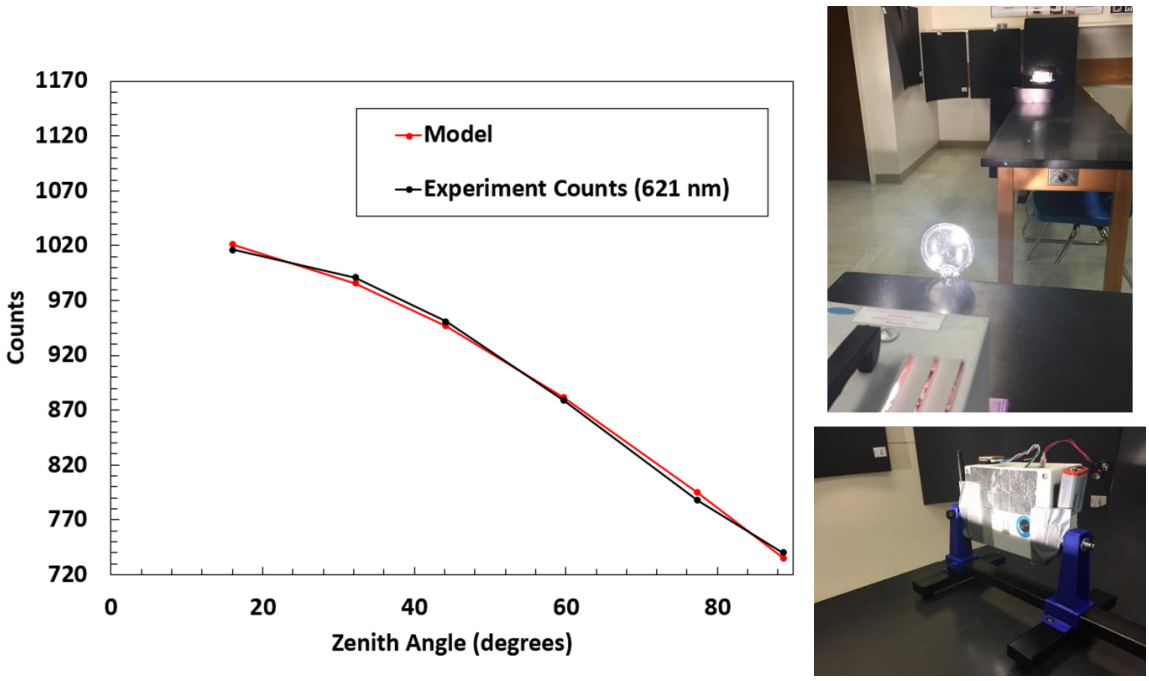


Figure 4: The cosine response of the instrument was measured to ensure proper use as an irradiance detector. Counts from the spectrometer at 621 nm were recorded from tilting the detector every few degrees. The experimental set-up involved a light source and a lens to focus the light evenly onto the detector.

3.3 Temperature Compensation

An experiment to model the dark counts of the spectrometers was performed using two environmental chambers: a toaster oven and a freezer. The spectrometer along with a temperature sensor were breadboarded and subjected to extreme operating temperatures. A second-degree polynomial fit was taken from the resulting curve and equations for modeling the dark counts with respect to temperature in degrees centigrade were found for each spectrometer (equations 3.a and 3.b). Temperature inputs for equations 3.a and 3.b (temp) are obtained from the onboard BME 280 pressure, temperature, & humidity sensor.

$$Modeled\ Dark_{spectrometer1} = 0.010593 * (temp^2) + 0.062132 * temp + 719.9529. \quad \text{(eq. 3.a)}$$

$$Modeled\ Dark_{spectrometer2} = 0.010715 * (temp^2) + 0.062741 * temp + 727.0078. \quad \text{(eq. 3.b)}$$

The dark count range uncertainty was found to be approximately 15 counts, where typical total counts are around 6000. The same procedure was done for each spectrometer, and it was found that the two spectrometers differed by less than 10 counts. Experimental analysis was performed such that the dark counts were averaged over all the wavelengths. Differences in the response times of the temperature sensor and the spectrometer caused the resulting hysteresis curve (figure 5). Discrepancies in the results could be due to the fact that the temperature sensor was not in direct contact with the spectrometer, and therefore did not represent the spectrometer actual temperature but instead the environmental temperature. Per manufacturer recommendations and to avoid condensation, we did not test below 5 °C.

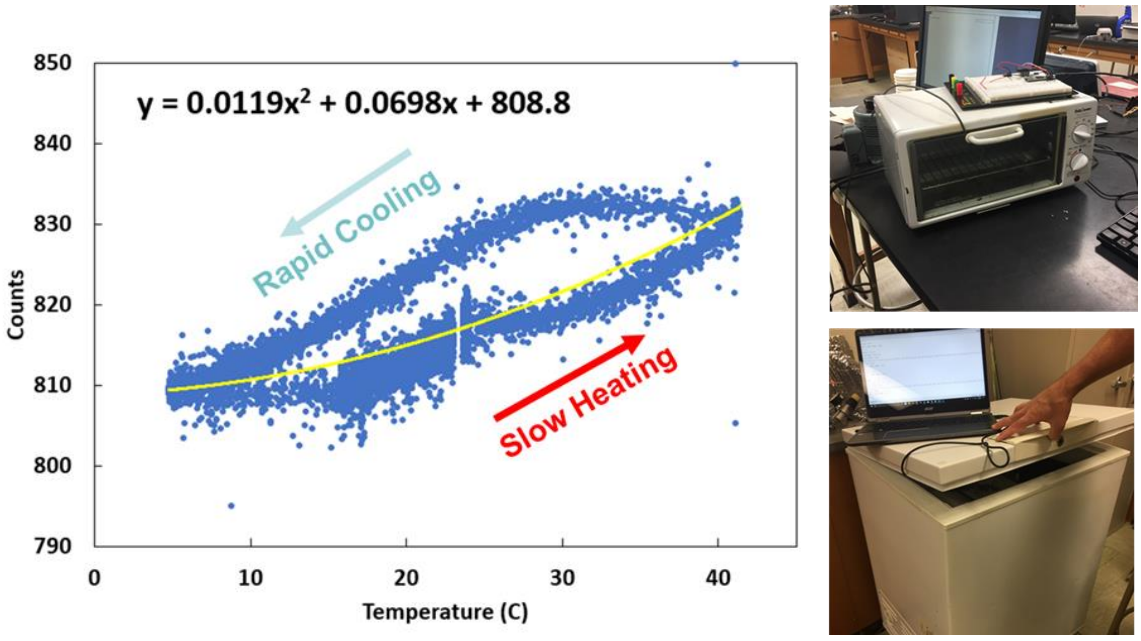


Figure 5: The two spectrometers onboard the instrument were tested and corrected for their temperature dependence. The hysteresis curve is a result of the temperature measurements and the spectrometer counts not changing at the same rate. A second-degree polynomial fit was taken from the resulting curve and equations for modeling the dark counts of the spectrometer with respect to temperature were incorporated. This is done due to provide the dark counts while the instrument is flying.

3.4 Transfer Function

A transfer function was calculated in order to correct for the differences between the two spectrometers and the slight variation in their diffusers (equation 4). Dark subtracted counts from each spectrometer were used for calculation in equation 4. The instrument was carefully flipped to obtain an upward and downward facing measurement for each spectrometer.

$$H(\lambda) = \sqrt{\frac{Spec_2\ Downward}{Spec_1\ Downward} * \frac{Spec_2\ Upward}{Spec_1\ Upward}}\ .$$

(eq. 4)

This was done by taking multiple measurements over the same surface. Nine measurements taken over a grass and concrete covered area were averaged for each wavelength (figure 6). The average was then applied to the output of one spectrometer in order to equal the other when measuring the same irradiance.

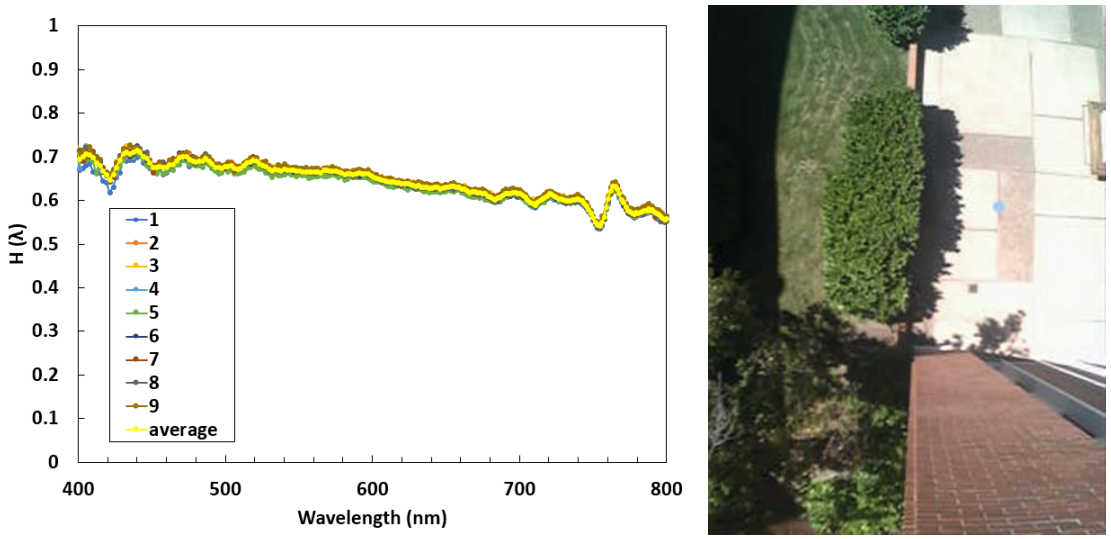


Figure 6: A transfer function to account for the differences in the two micro spectrometers was calculated. Multiple measurements were taken over the same surface and the average was applied to one spectrometer in order to "equal" the other.

Final albedo is calculated according to equation 5, where dark counts are subtracted from spectrometer counts and normalized by the integration time. The transfer function (H) is then applied to the upward facing spectrometer, $Spec_1$.

$$Albedo\ (\lambda) = \frac{\frac{Spec_2 - Dark}{int\ time}}{\left(\frac{Spec_1 - Dark}{int\ time}\right) * H(\lambda)}. \quad \text{(eq. 5)}$$

3.5 Preliminary Experiments

Initial testing of the instrument was performed over heterogeneous surfaces around the University of Nevada, Reno (UNR) campus. Surface reflectance of common surface types were examined: vegetation, dead vegetation, concrete, asphalt, blue paint, and mixed vegetation (figure 7). For verification, the results were qualitatively compared by eye with the USGS (United States Geological Survey) online spectral library. Overall the instrument performed well with appropriate vegetation, blue paint, and asphalt spectral signatures.

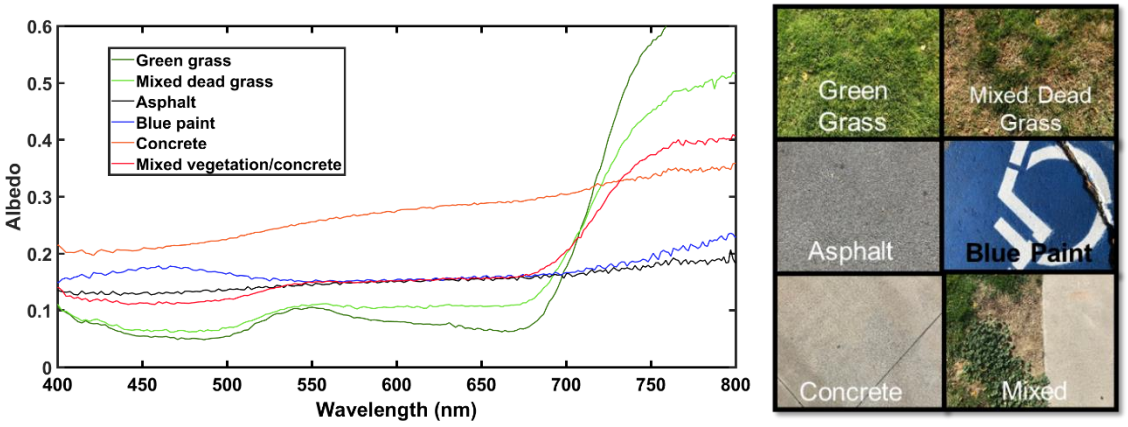


Figure 7: Initial testing of the instrument was performed over various surfaces (right) around the University of Nevada, Reno campus. The observed spectral signatures align with expected signatures for the examined surface types. The data collected here were obtained using the instrument in the handheld version.

In addition to the tests performed at UNR, a prototype of the instrument was taken to the Peruvian Andes for glacier albedo measurements. In this application, the instrument was used in the handheld mode and mounted on a trekking pole in order to facilitate use in

mountaineering applications (figure 8). The Peru expedition served to validate the portability, ease of use of the instrument, and functionality of the instrument as a handheld device. Lastly, for establishing proof of concept, the instrument was first flown onboard an unmanned aircraft at Silver Knoll Ranch located north of Reno. Preliminary airborne data was retrieved successfully with no radio interference between the aircraft and the ground control unit. The instrument mounting design proved to be sufficient for flying. The Silver Knoll Ranch expedition was meant as a practice run for the Black Rock Desert deployment.

Figure 8: Left photo. Undergraduate Chris Stevens using the Albedometer prototype mounted to the end of a trekking pole in handheld mode on a glacier in the Cordillera Blanca, Peru. Right photo. Albedometer mounted to end of copper pole on DJI Matrix 600 Pro hexacopter at Silver Knoll Ranch for a test flight.

Chapter 4 Nevada Black Rock Desert Measurements

The instrument was deployed in Nevada's Black Rock Desert under clear sky conditions on October 5th, 2017. The apparently homogeneous terrain of the Black Rock Desert was chosen as our study location for its known high surface reflectance. The quintessential surface of the Black Rock Desert is representative of other areas known to also have a high surface reflectance in the Western U.S. To obtain albedo, the measuring device was mounted onto a DJI Matrix 600 Pro Hexacopter. The rotary wing aircraft has dimensions 525 × 480 × 640 mm with a total weight (including batteries) of 10 kg and a recommended payload weight of 5.5 kg. With all payload attachments onboard, the aircraft was capable of flying for approximately 20 minutes. The instrument was mounted onto a carbon fiber pole which extended out from the aircraft to limit the aircraft's influence on the radiation field (figure 9). The aircraft was manually piloted over two surface types at four heights above ground level (AGL): 100, 200, 300 and 393 feet to simulate the spatial sensing area of MODIS. At 100ft AGL the ground field of view matches the 500 m spatial resolution of MODIS with a detector field of view of ~166° (Appendix 4). Over 90% of the measured signal is received with an 80° instantaneous field of view (IFOV) and was therefore assumed for the ground IFOV calculation. Flights were made as close to solar noon as possible and during flights the instrument was oriented to face the sun to avoid disturbances to the radiation field due to shadowing.

Figure 9: Configuration of instrument mounted to UAS. A long pole, approximately 2 m in length, was used to extend the instrument away from the body of the aircraft. This was done to limit the effects of the aircraft on albedo measurements, specifically those which would change the surrounding radiation field.

Measurements were obtained over two locations, designated in figure 10, as the red and blue circles for road and nonroad, respectively. The physical appearance of the two locations varied, however their compositions were believed to be similar. In one location, denoted “road”, the ground surface had been consistently driven over as a means in and out of the yearly Burning Man event, which had taken place the previous month. The “nonroad” location showed less evidence of vehicle tracks and showed no distinct disturbance from car tracks. In other words, the tracks appearing over the location of the nonroad observations were sparse and random compared to the road location. Five measurements were obtained at four different heights above each road and nonroad location. Per Federal Aviation Administration (FAA) Part 107 regulation, our flight height was restrained to below 400 feet (PART 107 - SMALL UNMANNED AIRCRAFT SYSTEMS, 2016).


a.)
Black Rock Desert
ROAD
HOME
Reno
N
90 mi


b.)
NONROAD
ROAD
HOME
N
2 mi

Figure 10: Field site locations. 10.a. Overview of Black Rock Desert (BRD) located North of Reno. 10.b. Zoomed-in Google Earth image over the BRD showing proximity to annual Burning Man Festival (the half circle). 10.c. Zoomed in Google Earth image over the location where measurements were made. The blue circle (most north) represents where our "nonroad" (40.749586, -119.261153) measurements were taken, the red circle (most south) represents the "road" area (40.748192, -119.258969) and the black circle in the middle indicated as "HOME" (40.748345, 119.263186) was the location where we were standing.

Chapter 5 Albedo Measurements and Discussion

Comparisons of UAS and satellite retrieved surface reflectance values over Nevada's Black Rock Desert (BRD) are presented. Five albedometer measurements at each height AGL were averaged and compared to single pixel values from MODIS and LANDSAT. For MODIS, the pixel value for road and nonroad areas were the same due to the large spatial resolution (250 m for band 1 and 500 m for bands 3 and 4). The 30 m resolution of LANDSAT7 ETM+ allowed for distinguished pixels to compare to road and nonroad areas.

In general, nonroad measurements exhibited a higher albedo than those obtained over the road location; likely due to the surface of the road area being more non-Lambertian. Measured albedo over nonroad locations ranged from ~0.35 at 400 nm to ~0.60 at 750 nm and from ~0.30 at 400 nm to ~0.50 at 750 nm over road locations (figures 11, 12, 13, and Appendix 3). Over both road and nonroad locations albedo tended to decrease with increasing height AGL. In other words, albedo measurements made closer to the surface were slightly greater than those made hundreds of feet above the surface. This can likely be attributed to the differences in the amount of atmosphere present between the albedometer and the surface. At greater heights AGL, there is more atmosphere to contribute to the scattering of reflected shortwave radiation. Additionally, at greater heights the detector is sensing over a larger spatial area which could contribute to the overall variability in the measurements. The observed range in measured albedo from the lowest height to the highest height (100 to 393 ft) was within 0.05 over road and nonroad locations, indicating that the effect of height AGL was less than the effect of road or

nonroad location. Additionally, at greater heights above the surface the field of view of the detector is also greater, and therefore albedo measurements are averaged over larger areas, giving rise to differences in albedo with change in height.

In comparison to Aqua MODIS, measurements obtained with the albedometer were higher across all MODIS bands, with errors of ~26% averaged over road and ~34% averaged over nonroad areas (figure 11). A similar trend was observed for comparison to Terra MODIS, with all bands underestimating albedometer values by an average percent error of approximately 26% for road and 34% for nonroad. (figure 12). When compared to MODIS 8-day best values, measured and retrieved values were closer in range than daily MODIS retrievals (figure A4 in Appendix 3). In comparison to LANDSAT 7 ETM+, measurements obtained with the albedometer were again higher across all bands (figure 13). However, LANDSAT 7 ETM+ values were closer to albedometer measured values than MODIS (percent error ~15% over road and ~14% over nonroad), likely due to the enhanced spatial resolution.

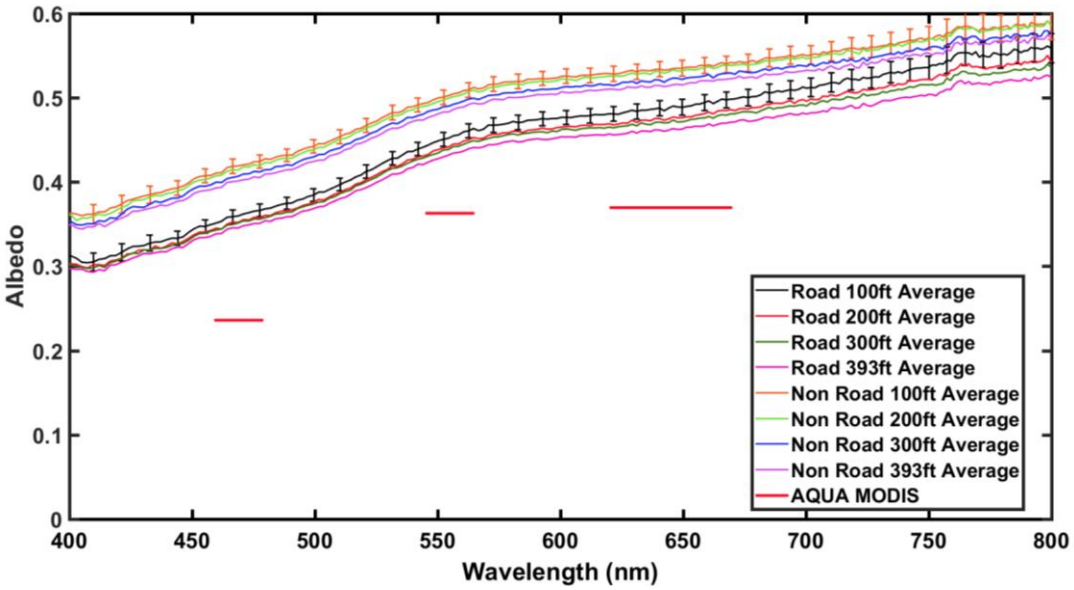


Figure 11: Albedometer measurements obtained over Nevada's Black Rock Desert on October 5th, 2017 and comparison to AQUA MODIS retrieved surface reflectance.

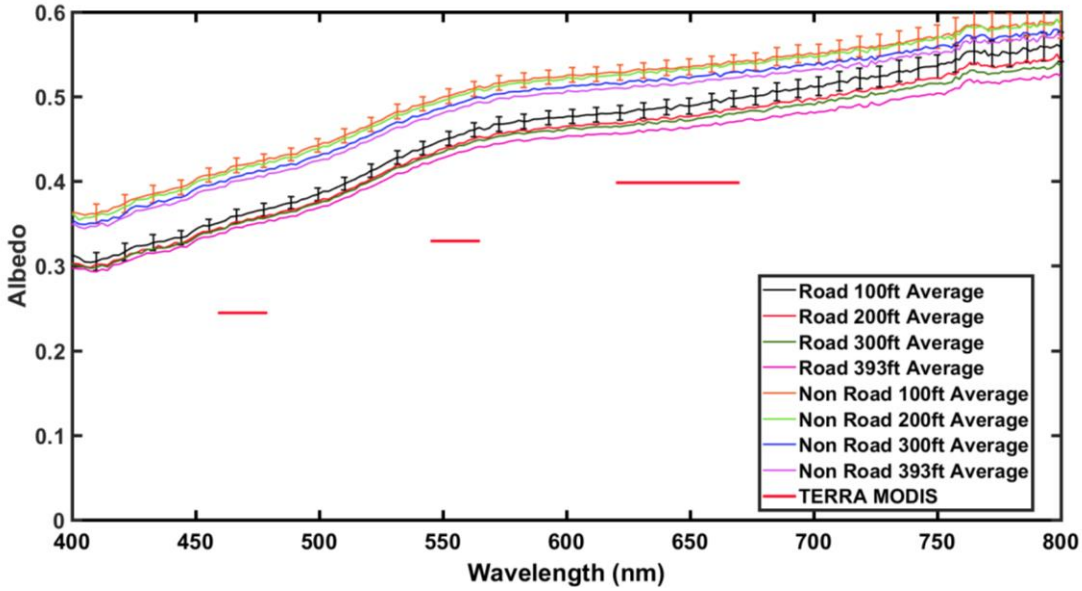


Figure 12: Albedometer measurements obtained over Nevada's Black Rock Desert on October 5th, 2017 and comparison to TERRA MODIS retrieved surface reflectance.

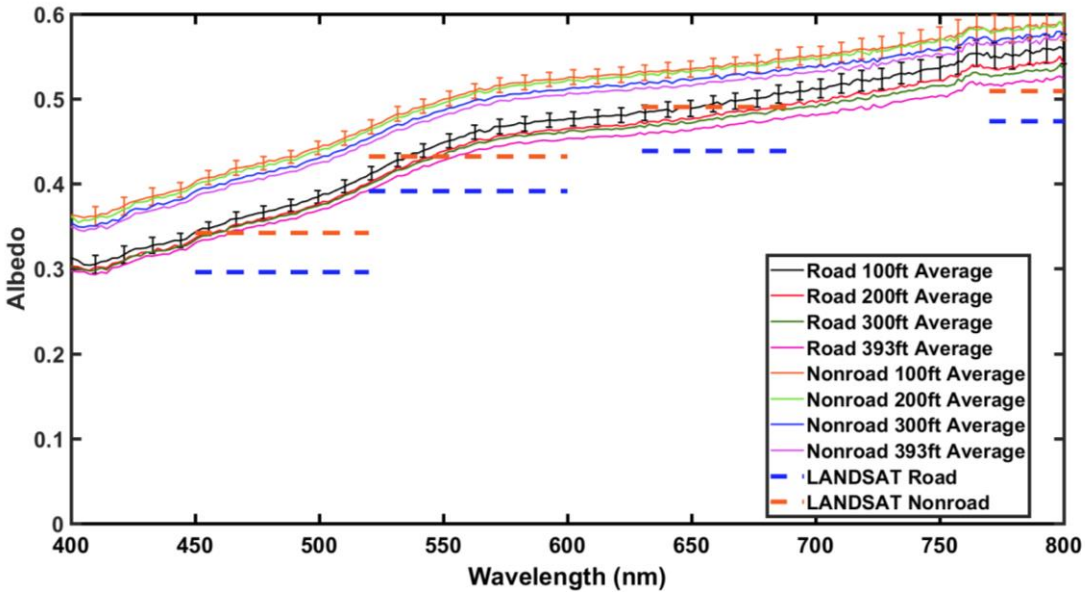


Figure 13: Albedometer measurements obtained over Nevada's Black Rock Desert on October 5th, 2017 and comparison to LANDSAT 7 ETM+ retrieved surface reflectance.

To address the sense of variability in the satellite products, histograms were generated for neighboring LANDSAT pixels around road and nonroad locations (figure 14 and 15). Local values were found to be in the vicinity of albedometer measurements for both road and nonroad locations indicating localized homogeneity. However, there appears to be a wider range of albedo values over the road location compared to the nonroad location which implies a greater variation in the road surface. Measured albedo from the UAS were in the range of neighboring pixel values for LANDSAT. Histograms of neighboring pixels for MODIS were not used due to the high spatial resolution of the sensor.

In addition to neighboring pixels, a histogram of pixel values across the entire Black Rock Desert was made to assess the homogeneity of the desert surface (figure 16). The region chosen to represent the Black Rock Desert is shown in figure 17 as the blue

shaded area. All valid pixels in the region of interest are incorporated into the histogram in figure 16, excluding missing data and over saturated pixels. The histogram had a wide range of albedo values across the entire desert for bands 1, 2, and 3, while band 4 albedo values were mostly within 0.4 – 0.5 range. This indicates that the Black Rock Desert is heterogeneously bright. The more consistent grouping of observed pixel values at band 4 is likely representative of a common surface type known to be present in dry lake beds such as the BRD. The saline minerals present in playas exhibit absorption features in the near-infrared bands, and their reflectance in the VNIR is highly dependent on moisture content (Crowley, 1991). The spread of albedo values obtained from LANDSAT across the BRD is likely a consequence of some parts being more wet than others for at least certain parts of the year. Brightness of the BRD playa is therefore likely to be sensitive to weather patterns and seasonal variation.

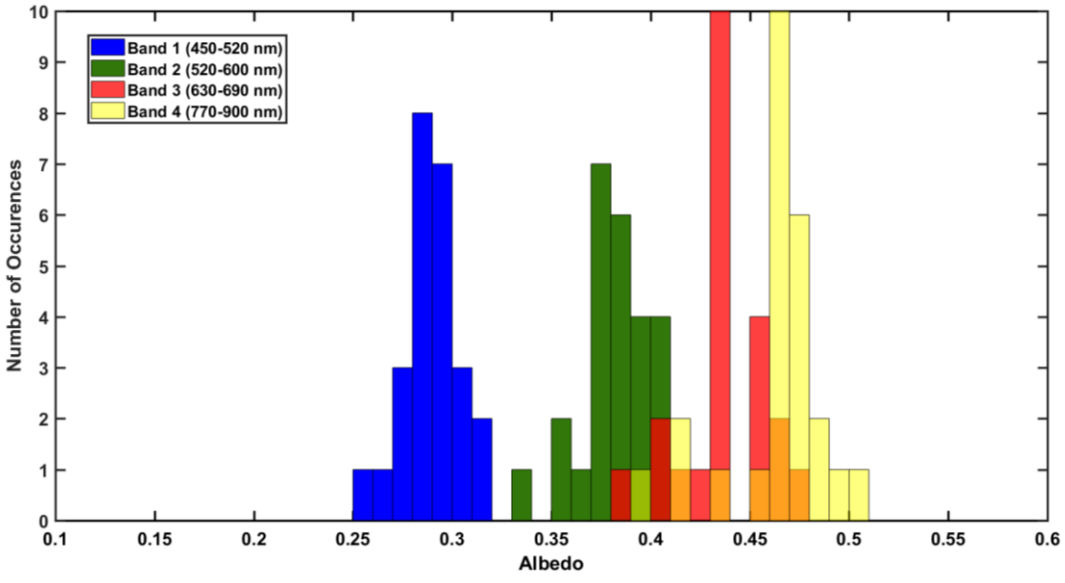


Figure 14: Histogram of neighboring LANDSAT pixels over road location on October 5th, 2017.

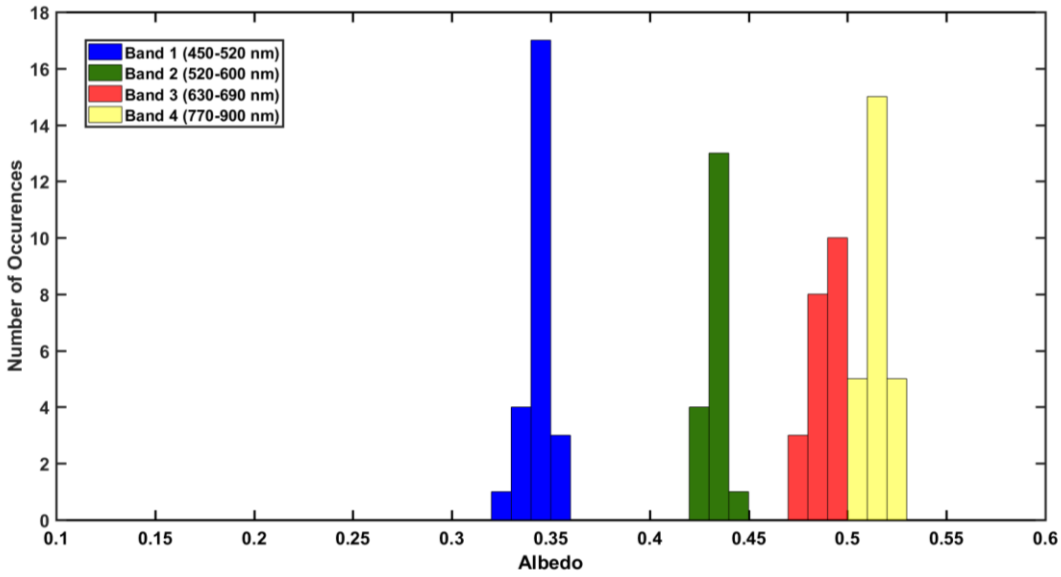


Figure 15: Histogram of neighboring LANDSAT pixels over nonroad location on October 5th, 2017.

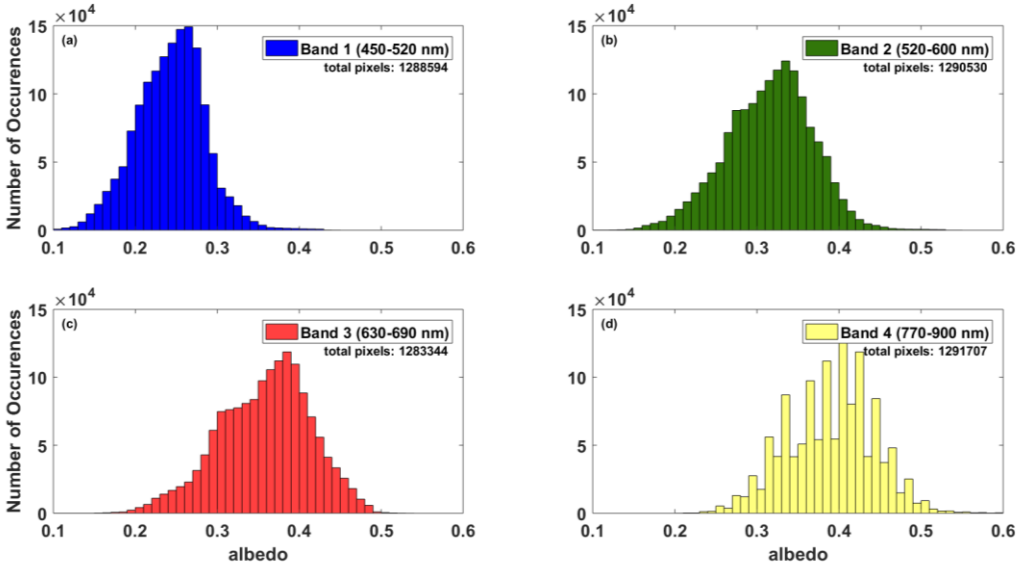


Figure 16: Histogram of all LANDSAT pixels over Nevada's Black Rock Desert on October 5th, 2017. 16.a. Band 1 (450-520 nm). 16.b. Band 2 (520-600 nm). 16.c. Band 3 (630-690 nm). 16.d. Band 4 (770-900 nm).

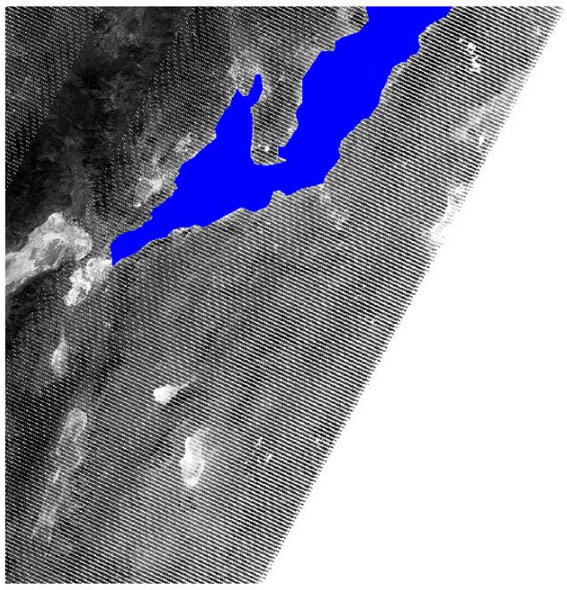


Figure 17: LANDSAT tile over Nevada's Black Rock Desert. Region of interest used to generate LANDSAT histogram is shaded in blue.

Chapter 6 Conclusions

The instrument development goal of this research was to develop a lightweight multispectral instrument to measure albedo. The primary application was to measure the albedo of a representatively bright area of the Black Rock Desert for comparison with MODIS albedo, ultimately in relation to issues with aerosol optical depth retrievals from MODIS observations. Our albedometer also serves as a general tool for quantifying surface albedo at a variety of time and length scales. It has broad applications for use in climate change research and environmental monitoring. The instrument is an inexpensive alternative to existing commercial devices and could serve as a great addition to field instrumentation. The design of the instrument allows for measuring the albedo of glaciers in mountaineering environments, for airborne measurements from aircraft, and for ground-based measurements over complex terrain.

We deployed the instrument for a desert environment to evaluate our measurements against satellite retrieved surface albedo. This was done to evaluate a major factor that influences the accuracy of satellite AOD retrievals. The discrepancies observed between ground based and satellite-retrieved atmospheric aerosol optical depth in the Western U.S. are likely due to the underestimation of surface reflectance over complex terrain. MODIS and LANDSAT surface reflectance consistently underestimated measured albedo across all spectral bands. This underestimation in surface reflectance likely contributes to an overestimation in observed aerosol optical depth. It was found that LANDSAT retrieved values agreed more with measured albedo values than MODIS did, likely due to the finer spatial resolution of LANDSAT.

The instrument was flown onboard an unmanned aircraft system to introduce a novel technique for taking airborne albedo measurements. We demonstrate that it is possible to accurately measure albedo at low altitudes using an unmanned aircraft system. Slight increases in albedo with decrease in height AGL indicate the need for possibly applying an atmospheric correction to measurements even when flown a few hundred feet above the surface, and for interpretation of the effective field of view of the instrument. Use of UAS provides for obtaining albedo measurements over a large spatial area at a much lower cost compared to using a high-altitude plane without compromising accuracy of results.

Future work could consist of developing the instrument for radiance measurements as well as for Bidirectional Reflectance Distribution Function (BRDF) measurements. Efforts to increase the wavelength range of the instrument would provide more thorough albedo measurements. Additional measurements over the entire Black Rock Desert are needed for a more comprehensive data set for comparison to satellite retrievals, as well as expanding the study to other areas known to have a high surface reflectance in the Western U.S. Additional comparison to other airborne platforms, such as manned aircraft for measuring surface reflectance, could also be explored. Analysis of seasonal variations in albedo over the Black Rock Desert could be useful for improving satellite retrievals of AOD and for general climate studies. Incorporating ground-based sun photometer measurements collocated with albedo measurements would provide additional comparison to assess satellite retrievals. Overall, the instrument has potential to be

transformed into a stationary device and could be designed to be mounted onto a tower with a more weather-proof case and a long-term power supply. Open sourcing the instrument design and software including the circuit board design, the 3D printed box, and the code used to operate the device is underway.

References

Barsi, J., Lee, K., Kvaran, G., Markham, B., & Pedelty, J. (2014). The Spectral Response of the Landsat-8 Operational Land Imager. *Remote Sensing*, *6*(10), 10232–10251. https://doi.org/10.3390/rs61010232

Brovkin, V., Boysen, L., Raddatz, T., Gayler, V., Loew, A., & Claussen, M. (2013). Evaluation of vegetation cover and land-surface albedo in MPI-ESM CMIP5 simulations. *Journal of Advances in Modeling Earth Systems*, *5*(1), 48–57. https://doi.org/10.1029/2012MS000169

Claverie, M., Vermote, E. F., Franch, B., & Masek, J. G. (2015). Evaluation of the Landsat-5 TM and Landsat-7 ETM + surface reflectance products. *Remote Sensing of Environment*, *169*, 390–403. https://doi.org/10.1016/j.rse.2015.08.030

Coddington, O., Schmidt, K. S., Pilewskie, P., Gore, W. J., Bergstrom, R. W., Roman, M., … Schaaf, C. C. (2008). Aircraft measurements of spectral surface albedo and its consistency with ground-based and space-borne observations. *Journal of Geophysical Research-Atmospheres*, *113*(D17), D17209. https://doi.org/10.1029/2008JD010089

Crowley, J. (1991). Visible and Near-Infrared (0.4-2.5 Mu-M) Reflectance Spectra of Playa Evaporite Minerals. *Journal of Geophysical Research-Solid Earth*, *96*(B10), 16231–16240. https://doi.org/10.1029/91JB01714

He, T., Liang, S., & Song, D.-X. (2014). Analysis of global land surface albedo climatology and spatial-temporal variation during 1981-2010 from multiple satellite products. *Journal of Geophysical Research-Atmospheres*, *119*(17). https://doi.org/10.1002/2014JD021667

Heikkinen, P., Pulliainen, J., Kyro, E., Sukuvaara, T., Suokanerva, H., & Kontu, A. (2007). Comparison of MODIS surface reflectance with mast-based spectrometer observations using CORINE20001and cover database. In *2007 IEEE International Geoscience and Remote Sensing Symposium* (pp. 4117–4119). https://doi.org/10.1109/IGARSS.2007.4423755

Jonsell, U., Hock, R., & Holmgren, B. (2003). Spatial and temporal variations in albedo on Storglaciären, Sweden. *Journal of Glaciology*, *49*(164), 59–68. https://doi.org/10.3189/172756503781830980

Liu, Y., Wang, Z., Sun, Q., Erb, A. M., Li, Z., Schaaf, C. B., … SanClements, M. (2017). Evaluation of the VIIRS BRDF, Albedo and NBAR products suite and an assessment of continuity with the long term MODIS record. *Remote Sensing of Environment*, *201*, 256–274. https://doi.org/10.1016/j.rse.2017.09.020

Loría-Salazar, S. M., Holmes, H. A., Patrick Arnott, W., Barnard, J. C., & Moosmüller, H. (2016). Evaluation of MODIS columnar aerosol retrievals using AERONET in semi-arid Nevada and California, U.S.A., during the summer of 2012. *Atmospheric Environment*, *144*(Supplement C), 345–360. https://doi.org/10.1016/j.atmosenv.2016.08.070

Maiersperger, T. K., Scaramuzza, P. L., Leigh, L., Shrestha, S., Gallo, K. P., Jenkerson, C. B., & Dwyer, J. L. (2013). Characterizing LEDAPS surface reflectance products by comparisons with AERONET, field spectrometer, and MODIS data. *Remote Sensing of Environment*, *136*, 1–13. https://doi.org/10.1016/j.rse.2013.04.007

Mira, M., Weiss, M., Baret, F., Courault, D., Hagolle, O., Gallego-Elvira, B., & Olioso, A. (2015). The MODIS (collection V006) BRDF/albedo product MCD43D: Temporal course evaluated over agricultural landscape. *Remote Sensing of Environment*, *170*, 216–228. https://doi.org/10.1016/j.rse.2015.09.021

Modis Land Surface Reflectance - Home. (n.d.). Retrieved April 30, 2018, from http://modis-sr.ltdri.org/

Myhre, G., Bréon, F.-M., Aamaas, B., & Jacob, D. (n.d.). 8 Anthropogenic and Natural Radiative Forcing, 82.

Novel Estimation of Albedo Using a Drone Pyranometer - Kipp & Zonen. (n.d.). Retrieved April 29, 2018, from http://www.kippzonen.com/News/803/Novel-Estimation-of-Albedo-Using-a-Drone-Pyranometer#.WtETRIjwZPa

PART 107 - SMALL UNMANNED AIRCRAFT SYSTEMS, 14 C.F.R § 107 (2016). Retrieved from https://www.ecfr.gov/cgi-bin/text-idx?SID=dc908fb739912b0e6dcb7d7d88cfe6a7&mc=true&node=pt14.2.107&rgn=div5#se14.2.107_115

Pinty, B., Taberner, M., Haemmerle, V. R., Paradise, S. R., Vermote, E., Verstraete, M. M., … Widlowski, J.-L. (2011). Global-Scale Comparison of MISR and MODIS Land Surface Albedos. *Journal of Climate*, *24*(3), 732–749. https://doi.org/10.1175/2010JCLI3709.1

Schmitt, C. G., All, J. D., Schwarz, J. P., Arnott, W. P., Cole, R. J., Lapham, E., & Celestian, A. (2015). Measurements of light-absorbing particles on the glaciers in the Cordillera Blanca, Peru. *The Cryosphere*, *9*(1), 331–340. https://doi.org/10.5194/tc-9-331-2015

Sorek-Hamer, M., Kloog, I., Koutrakis, P., Strawa, A. W., Chatfield, R., Cohen, A., … Broday, D. M. (2015). Assessment of PM2.5 concentrations over bright surfaces using MODIS satellite observations. *Remote Sensing of Environment*, *163*, 180–185. https://doi.org/10.1016/j.rse.2015.03.014

Taha, H. (1997). Urban climates and heat islands: albedo, evapotranspiration, and anthropogenic heat. *Energy and Buildings*, *25*(2), 99–103. https://doi.org/10.1016/S0378-7788(96)00999-1

Teensy USB Development Board. (n.d.). Retrieved April 29, 2018, from https://www.pjrc.com/teensy/

USGS. (2018, March). Landsat 4-7 Surface Reflectance (LEDAPS) Product. Department of the Interior U.S. Geological Survey. Retrieved from https://landsat.usgs.gov/sites/default/files/documents/ledaps_product_guide.pdf

Uto, K., Seki, H., Saito, G., Kosugi, Y., & Komatsu, T. (2016). Development of a Low-Cost Hyperspectral Whiskbroom Imager Using an Optical Fiber Bundle, a Swing Mirror, and Compact Spectrometers. *Ieee Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *9*(9), 3909–3925. https://doi.org/10.1109/JSTARS.2016.2592987

van der Hage, J. C. H. (1992). Interpretation of Field Measurements Made with a Portable Albedometer. *Journal of Atmospheric and Oceanic Technology*, *9*(4), 420–425. https://doi.org/10.1175/1520-0426(1992)009<0420:IOFMMW>2.0.CO;2

Vermote, E. F., El Saleous, N., Justice, C. O., Kaufman, Y. J., Privette, J. L., Remer, L., … Tanré, D. (1997). Atmospheric correction of visible to middle-infrared EOS-

MODIS data over land surfaces: Background, operational algorithm and validation. *Journal of Geophysical Research: Atmospheres*, *102*(D14), 17131–17141. https://doi.org/10.1029/97JD00201

Vermote, E. F., Roger, J. C., & Ray, J. P. (2015, May). MODIS Surface Reflectance User's Guide. NASA. Retrieved from http://modis-sr.ltdri.org/guide/MOD09_UserGuide_v1.4.pdf

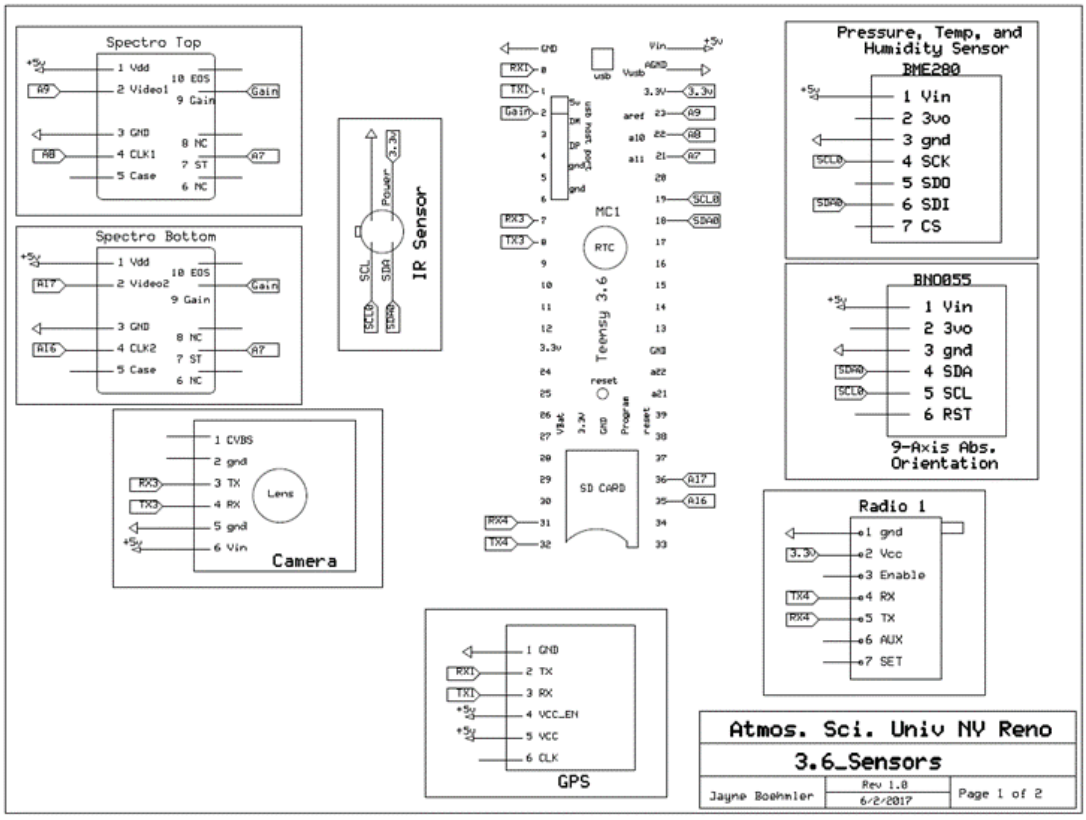
Vermote, Eric F, El Saleous, N. Z., & Justice, C. O. (2002). Atmospheric correction of MODIS data in the visible to middle infrared: first results. *Remote Sensing of Environment*, *83*(1–2), 97–111. https://doi.org/10.1016/S0034-4257(02)00089-5

Wendisch, M., Müller, D., Schell, D., & Heintzenberg, J. (2001). An Airborne Spectral Albedometer with Active Horizontal Stabilization. *Journal of Atmospheric and Oceanic Technology*, *18*(11), 1856–1866. https://doi.org/10.1175/1520-0426(2001)018<1856:AASAWA>2.0.CO;2

Zhang, H., Kondragunta, S., Laszlo, I., Liu, H., Remer, L. A., Huang, J., … Ciren, P. (2016). An enhanced VIIRS aerosol optical thickness (AOT) retrieval algorithm over land using a global surface reflectance ratio database. *Journal of Geophysical Research: Atmospheres*, *121*(18), 2016JD024859. https://doi.org/10.1002/2016JD024859

Zhou, H., Wang, J., & Liang, S. (2018). Design of a Novel Spectral Albedometer for Validating the MODerate Resolution Imaging Spectroradiometer Spectral Albedo Product. *Remote Sensing*, *10*(1), 101. https://doi.org/10.3390/rs10010101

Appendix 1


Spectro Top
1 Vdd
2 Video1
3 GND
4 CLK1
5 Case
10 EOS
9 Gain
8 NC
7 ST
6 NC
Gain
A9
A8
A7
Spectro Bottom
1 Vdd
2 Video2
3 GND
4 CLK2
5 Case
10 EOS
9 Gain
8 NC
7 ST
6 NC
Gain
A17
A16
A7
IR Sensor
Power
SCL
SDA
3.3v
SCL0
SDA0
Camera
1 CVBS
2 gnd
3 TX
4 RX
5 gnd
6 Vin
Lens
RX3
TX3
MC1
RTC
Teensy 3.6
SD CARD
reset
Program
GND
RX1
TX1
Gain
RX3
TX3
RX4
TX4
A9
A8
A7
A17
A16
SCL0
SDA0
Pressure, Temp, and Humidity Sensor
BME280
1 Vin
2 3vo
3 gnd
4 SCK
5 SDO
6 SDI
7 CS
BNO055
1 Vin
2 3vo
3 gnd
4 SDA
5 SCL
6 RST
9-Axis Abs. Orientation
Radio 1
1 gnd
2 Vcc
3 Enable
4 RX
5 TX
6 AUX
7 SET
GPS
1 GND
2 TX
3 RX
4 VCC_EN
5 VCC
6 CLK
Atmos. Sci. Univ NV Reno
3.6_Sensors
Jayne Boehmler
Rev 1.0
6/2/2017
Page 1 of 2

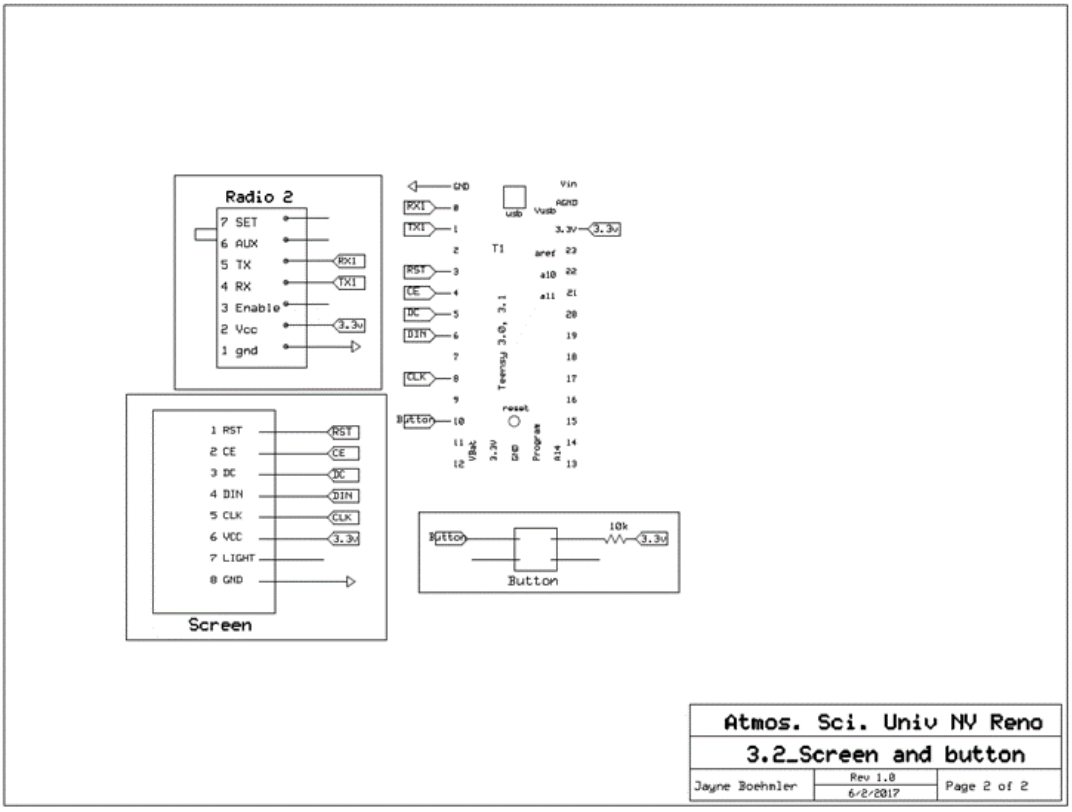


Figure A1: Albedometer circuit board schematic for measuring device (top) and handheld ground control until (bottom).

Appendix 2: Arduino code used to run both parts of the instrument is uploaded to the Teensy 3.2 and 3.6 microcontrollers.

```
/* Albedometer, Teensy 3.6 */
/* UAV VERSION with calculated dark current */


/* Push button
take 1 measurement
push again to take another
Also includes printing of K, High gain setting, teensy avg 32, set integration time of 100ms,
and temperature modeled dark current, and USES the new transfer fcn for
the Auto Gain off setting.
*/


//Libraries
#include <TinyGPS.h>                    /* For the GPS */
#include <Adafruit_Sensor.h>
```

```
#include <Adafruit_BME280.h>      /* For Temp/Pressure/Humidity Sensor */
#include <Adafruit_MLX90614.h> /* For IR Sensor */
#include <Adafruit_VC0706.h>      /* For Camera */
#include <Adafruit_BNO055.h>      /* For Absolute Orient - Level Sensor */
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <TimeLib.h>
#include <utility/imumaths.h>


//Function Calls void sendByte();
void getAmbientTemp(); void getObjectTemp(); void getTemp();
void getHumidity(); void getPressure(); void getAltitude(); void getGPS();
void saveData();
void takePicture();
void gpsdump(TinyGPS &gps);


//Set Up GPS TinyGPS gps;
#define HWSERIALgps Serial4        /* Serial port used for the NMEA GPS      */


//Set Up Camera
#define HW Serial3
File myFile;        //Create file object
#define chipSelect    BUILTIN_SDCARD        // Assign memory card pin
Adafruit_VC0706 cam = Adafruit_VC0706(&HW);


//Initialize I2C Adafruit_BME280 bme;

//Set Up Level, Absolute Orient
#define BNO055_SAMPLERATE_DELAY_MS (100) Adafruit_BNO055 bno =
Adafruit_BNO055(55); char tempArray[9] = {0};
char roll[9] = {0}; char pitch[9] = {0}; float rollSpec1; float pitchSpec1;

// Leveling data
float RollBeforeSpec1UP; float RollAfterSpec1UP; float RollBeforeSpec2DOWN; float
RollAfterSpec2DOWN; float PitchBeforeSpec1UP; float PitchAfterSpec1UP; float
PitchBeforeSpec2DOWN; float PitchAfterSpec2DOWN;



//IR Sensor
Adafruit_MLX90614 mlx = Adafruit_MLX90614();


//Declarations
float pressure;              //hPa float ambientTemp;          //C* float objectTemp;          //C* float
temp;                    //C* float alt;                    //m float humid;                //%

//Sets Sea Level Pressure
#define SEALEVELPRESSURE_HPA (1013.25)
```

```
//Set up Spectrometers
#define SPEC_GAIN 2 // Pin 2 on Teensy, recognized as an int.
#define SPEC_ST 21 // Could also be pin A7 instead. 21


#define SPEC_CLK1 22 // Could also be pin A8 instead. 22
#define SPEC_VIDEO1 23 // Could also be pin A9 instead. 23
#define SPEC_CLK2 35 // Could also be pin A16 instead. 35
#define SPEC_VIDEO2 36 // Could also be pin A17 instead. 36
#define SPEC_CHANNELS 256 float albedo[SPEC_CHANNELS];
float AlbedoUncertainty[SPEC_CHANNELS];
float Spectra_Added[SPEC_CHANNELS];
/*New Transfer Function */
float Transfer_function[] = {0.954923333, 0.937852222,0.913888889, 0.
912672222, 0.898741111, 0.894436667, 0.903385556,
0.872847778, 0.858934444, 0.862938889, 0.829883333, 0.831866667, 0.822824444,
0.802516667, 0.791152222, 0.761364444,
0.764121111, 0.768255556, 0.764497778, 0.750393333, 0.720847778, 0.692708889,
0.70551, 0.683398889, 0.682781111, 0.700182222,
0.699522222, 0.713965556, 0.720002222, 0.70739, 0.700038889, 0.713603333, 0.
730401111, 0.733724444, 0.72549, 0.719953333,
0.694745556, 0.701234444, 0.707104444, 0.705844444, 0.698347778, 0.687364444,
0.681735556, 0.668065556, 0.660224444,
0.647616667, 0.660777778, 0.676687778, 0.693, 0.706662222, 0.711295556, 0.
707105556, 0.710726667, 0.716373333, 0.711425556,
0.700721111, 0.696773333, 0.687167778, 0.675865556, 0.676202222, 0.678375556,
0.675817778, 0.676655556, 0.683166667,
0.683127778, 0.691454444, 0.700194444, 0.701926667, 0.701757778, 0.69476, 0.
691638889, 0.688636667, 0.691371111, 0.691084444,
0.696582222, 0.691938889, 0.685112222, 0.677226667, 0.676144444, 0.674444444,
0.677757778, 0.678672222, 0.681453333,
0.674698889, 0.672387778, 0.675157778, 0.680414444, 0.686135556, 0.689041111,
0.692237778, 0.690282222, 0.684472222,
0.680803333, 0.677956667, 0.673438889, 0.669104444, 0.672331111, 0.67088, 0.
669056667, 0.671634444, 0.671347778,
0.670944444, 0.670657778, 0.668023333, 0.667627778, 0.667168889, 0.667353333,
0.666344444, 0.666994444, 0.666038889,
0.66539, 0.665121111, 0.670146667, 0.670466667, 0.667706667, 0.669128889, 0.
66865, 0.665842222, 0.663966667, 0.660752222,
0.661375556, 0.662138889, 0.660935556, 0.662338889, 0.663663333, 0.662363333,
0.662248889, 0.661492222, 0.65838, 0.654621111,
0.652583333, 0.649706667, 0.648165556, 0.64569, 0.643672222, 0.643812222, 0.
642907778, 0.640745556, 0.640752222,
0.639022222, 0.639, 0.638508889, 0.636614444, 0.636237778, 0.633847778, 0.
636166667, 0.632191111, 0.632194444, 0.631566667,
0.628933333, 0.629292222, 0.632225556, 0.630586667, 0.627615556, 0.629836667,
0.630588889, 0.630912222, 0.632557778,
0.63323, 0.632095556, 0.631777778, 0.627152222, 0.627358889, 0.622523333, 0.
61973, 0.619978889, 0.620083333, 0.619374444,
0.617984444, 0.617403333, 0.616094444, 0.611122222, 0.608258889, 0.604852222,
0.602013333, 0.605594444, 0.60917, 0.612921111,
0.617433333, 0.617485556, 0.617442222, 0.619876667, 0.618827778, 0.616728889,
```

```
0.616822222, 0.612081111, 0.608291111,
0.602998889, 0.595821111, 0.593583333, 0.590915556, 0.596264444, 0.60118, 0.
603705556, 0.607088889, 0.611355556, 0.61588,
0.615461111, 0.609636667, 0.608892222, 0.604813333, 0.604051111, 0.60208, 0.
599631111, 0.599982222, 0.599966667,
0.603222222, 0.602982222, 0.604894444, 0.60307, 0.599796667, 0.595015556, 0.
587808889, 0.580437778, 0.571736667, 0.560137778,
0.55003, 0.544388889, 0.54208, 0.550808889, 0.562103333, 0.585604444, 0.
603884444, 0.622484444, 0.63257, 0.633951111,
0.627596667, 0.616226667, 0.602278889, 0.595341111, 0.584896667, 0.578542222,
0.574517778, 0.570485556, 0.570013333,
0.572608889, 0.572247778, 0.573077778, 0.575721111, 0.577791111, 0.578355556,
0.582221111, 0.578144444, 0.580181111,
0.5731, 0.570548889, 0.570327778, 0.566117778, 0.562016667, 0.55669, 0.
559457778, 0.555164444, 0.556283333, 0.557306667,
0.555628889, 0.556217778};


float spec1DataUp[SPEC_CHANNELS];
float spec2DataDown[SPEC_CHANNELS];


float wavelength1[SPEC_CHANNELS];
float wavelength2[SPEC_CHANNELS];


float darkData1[SPEC_CHANNELS]; float darkData2[SPEC_CHANNELS]; float Dark1_1;
float Dark2_2;


float kk1 = 0;
float kk2 = 0;


uint16_t intTimeSpec1 = 100;
uint16_t intTimeSpec2 = 100;
uint16_t integrationTime1_1 = 100; // Spec 1, UP
uint16_t integrationTime2_2 = 100; // Spec 2, DOWN


uint16_t Dark1 = 950;    // For Spectrometer C12666MA
uint16_t Dark2 = 950;    // For Spectrometer C12666MA


int darkCurrentState = 0;
char darkCurrentCom[5] = {0};


int dataState = 0;
char dataCom [5] = {0};
int buttonState = 0;
char buttonCom[5] = {0};
```

```
// Bools
bool AutoGain = false; boolean newData = false; boolean isLevel = false;

/******************GPS Constants*********************/ float gpslat = -99.99 ; /*
decimal degrees */ float gpslon = -99.99 ; /* decimal degrees */ float gpsalt = -99.99
; /* meters */
float gpsspeed = -99.99 ; /* kilometers / hour */
float gpscourse = -99.99 ; /* degrees */
int gpsyear = -99 ; /* e.g. 2016 */
int gpsmon = 99 ; /* 1 thru 12 UTC */ int gpsday = 99
; /* 1 thru 31 UTC */ int gpshour = 99 ; /* 0 thru 59 UTC */ int gpsmin
= 99 ; /* 0 thru 59 UTC */ int gpssec = 99 ; /* 0 thru
59 UTC */

// Saving to SD
char filename[] = "00000000.txt"; // Data


void setup() {
Serial.println("Made it in set up"); HW.begin(9600); //camera
//Open Serial to Teensy Serial.begin(9600); Serial5.begin(9600); //Radio
HWSERIALgps.begin(9600); // GPS
//bool status; bme.begin(); bno.begin(); mlx.begin();
setSyncProvider(getTeensy3Time);


/*****Set up the Teensy analog to digital conversion******/ analogReadResolution(13); // Do 13
bit analog read resolution on the Teensy. analogReadAveraging(32); // Do 1 measurements and
average them for every
analog input measurement.


//pinMode(SPEC_EOS, INPUT); pinMode(SPEC_GAIN, OUTPUT); pinMode(SPEC_ST, OUTPUT);
pinMode(SPEC_CLK1, OUTPUT);
pinMode(SPEC_CLK2, OUTPUT);
//pinMode(SPEC_CLK3, OUTPUT);


digitalWrite(SPEC_ST, HIGH); digitalWrite(SPEC_CLK1, HIGH); digitalWrite(SPEC_CLK2, HIGH);
//digitalWrite(SPEC_CLK3, HIGH);
digitalWrite(SPEC_GAIN, LOW); //set low for HIGH Gain set high for LOW Gain


if (!SD.begin(chipSelect)) { Serial.println("Card failed, or not present");
// don't do anything more:
return;
}


//Sets Image Size
//cam.setImageSize(VC0706_640x480); // biggest cam.setImageSize(VC0706_320x240);
// medium
//cam.setImageSize(VC0706_160x120); // small
//uint8_t imgsize = cam.getImageSize();
```

```
//Wavelength  calibration readWavelength(1); readWavelength(2);

} // End of Set up void loop() {
Serial.println("Made it in loop");
if (Serial.available()) {
time_t t = processSyncMessage();
if (t != 0) {
Teensy3Clock.set(t); // set the RTC
setTime(t);
}
}    // End of Time processSync


do{
int i = 0;
//delay(2000);
//Reads in a button push over radio as an array and outputs as an int if (Serial5.available() > 0
&& newData == false) {
delay(100); //allows all serial sent to be received together while (Serial5.available() && i < 5) {
Serial.println("Data is writing");
buttonCom[i++] = Serial5.read();
}
buttonCom[i++] = '\0'; newData = true; Serial.println(buttonCom);
}
if (newData == true) { buttonState = atoi(buttonCom); Serial.println(buttonState); buttonState =
HIGH;
isLevel = false;
}
} while (buttonState == LOW);


while (buttonState == HIGH) // Giant while button is pushed, begin taking data
{
// Send Status Spec 1 UP
char StatusSpec11[10] = "z111&"; Serial5.write(StatusSpec11);

// Get and Store Level sensors_event_t spec2; bno.getEvent(&spec2);
RollBeforeSpec1UP =    (float)spec2.orientation.y; PitchBeforeSpec1UP = (float)spec2.orientation.z;

readSpectrometer(1, 1);    // Spectrometer 1, UP (1)


// get and store level sensors_event_t spec3; bno.getEvent(&spec3);
RollAfterSpec1UP =    (float)spec3.orientation.y;
PitchAfterSpec1UP = (float)spec3.orientation.z;


// Send Status Spec 2 DOWN
char StatusSpec22[20] = "z221&"; Serial5.write(StatusSpec22);

// get and store level sensors_event_t spec4; bno.getEvent(&spec4);
```

```
RollBeforeSpec2DOWN = (float)spec4.orientation.y; PitchBeforeSpec2DOWN =
(float)spec4.orientation.z;

readSpectrometer(2, 2); // Spectrometer 2, DOWN (2)

// get and store level sensors_event_t spec5; bno.getEvent(&spec5);
RollAfterSpec2DOWN = (float)spec5.orientation.y; PitchAfterSpec2DOWN =
(float)spec5.orientation.z;

// Get everything else getAmbientTemp(); getObjectTemp(); getTemp(); getHumidity(); getPressure();
getAltitude(); getGPS();
// Send Status: Saving Photo char StatusPhoto[20] = "z60&"; Serial5.write(StatusPhoto); takePicture();
delay(100);


// Send Status: Saving Raw char StatusRaw[20] = "z80&"; Serial5.write(StatusRaw);

/* Save all of the raw data: wavelength, darks, 4 raw spectras */
saveRAWSpecData(wavelength1, spec1DataUp, spec2DataDown, integrationTime1_1,
integrationTime2_2);

/* Do all of the calculations */
// Send Status: Calculating Albedo char StatusAlbedo[20] = "z90&"; Serial5.write(StatusAlbedo);

/* Subtract the dark current */
Dark1_1 = 0.010593*pow(temp,2) + 0.062132*temp + 719.9529; Dark2_2 = 0.010715*pow(temp,2)
+ 0.062741*temp + 727.0078; for(int i=0; i < SPEC_CHANNELS; i++){
spec1DataUp[i] -= Dark1_1;
spec2DataDown[i] -= Dark2_2;
//darkData1[i] = Dark1_1; // Added to print out, not necessary
//darkData2[i] = Dark2_2; // Added to print out, not necessary
} // End of Subtract Dark
//Serial.println(Dark1_1);
//Serial.println(Dark2_2);
/* Uncertainty calculation PT.1 */
for(int i=0; i < SPEC_CHANNELS; i++){
Spectra_Added[i] = (0.5)* sqrt(1.0/abs(spec1DataUp[i]) + 1.
0/abs(spec2DataDown[i]));
}


/* Divide by the integration time */
divideIntTime(1,spec1DataUp, integrationTime1_1, 1); // Spec #, data, integration time, 1 for UP
divideIntTime(2,spec2DataDown, integrationTime2_2, 2); // Spec #, data, integration time, 2 for
DOWN

/* Transfer Function calculation H() */
for(int i=0; i < SPEC_CHANNELS; i++){
spec1DataUp[i] = Transfer_function[i]*spec1DataUp[i];
}


/* Albedo calculation */
```

```
spectraRatio(1, spec1DataUp, spec2DataDown); // Spec ratio (1 or 2), Spec 1 UP minus dark, Spec 2 down minus dark

/* Uncertainty calculation PT.2 */ for(int i=0; i < SPEC_CHANNELS; i++){ AlbedoUncertainty[i] = Spectra_Added[i] * albedo[i];
}


/* Save Data */
saveData();


/** Format and send Albedo **/
float PeakSpec = 0;
float MinSpec = 1000000;
char yPixel[84];
char Pixel[9];
float constrainedSpectra[SPEC_CHANNELS];
float OrigPeakSpec = 0;
char Peak[9]={0};


for (int j = 0; j < SPEC_CHANNELS; j++) {        // Finds peak value in original Albedo
if (albedo[j] > OrigPeakSpec) {
OrigPeakSpec = albedo[j];
}
}
// Send over the peak of albedo to display on screen sprintf(Peak, "q%02.02f&", OrigPeakSpec);
Serial5.write(Peak);


for (int j = 0; j < 84; j++) {        // Finds peak value and minimum value in constrained spectra
constrainedSpectra[j] = ((albedo[3*j] + albedo[3*j+1] +
albedo[3*j+2])/3);    // Averages every 3 values in array if (constrainedSpectra[j] > PeakSpec) {
PeakSpec = constrainedSpectra[j];
}
if (constrainedSpectra[j] < MinSpec){ MinSpec = constrainedSpectra[j];
}
}


float den = PeakSpec - MinSpec;
for (int j = 0; j < 84; j++) {
yPixel[j] = 40-(40 * (PeakSpec - constrainedSpectra[j])/ den);
sprintf(Pixel, "a%0d&", yPixel[j]); Serial5.write(Pixel); Serial.println(Pixel);
delay(10);
}


delay(100); buttonState = LOW; newData = false;
Serial.println(buttonState);


} // End button State High buttonState = LOW;
```

```
}   // End of Main Loop


void readWavelength(int SpecNum) {
if (SpecNum == 1) {
float A_0 = 3.157284930e2; float B_1 = 2.382758890; float B_2 = -4.532653713e-4; float B_3 = -
9.592362306e-6; float B_4 = 2.283177122e-8;
float B_5 = -2.319360095e-11;
for (int i = 0; i < SPEC_CHANNELS; i++) {
wavelength1[i] = A_0 + B_1 * i + B_2 * pow(i, 2) + B_3 * pow(i, 3) + B_4 * pow(i, 4) +
B_5 * pow(i, 5);
}
}
else if (SpecNum == 2) {
float A_0 = 3.187695698e2; float B_1 = 2.384544110; float B_2 = -6.291373514e-4; float B_3 = -
7.755157198e-6; float B_4 = 1.487105328e-8;
float B_5 = -1.104337883e-11;
for (int i = 0; i < SPEC_CHANNELS; i++) {
wavelength2[i] = A_0 + B_1 * i + B_2 * pow(i, 2) + B_3 * pow(i, 3) + B_4 * pow(i, 4) +
B_5 * pow(i, 5);
}
}
}   // End of Read Wavelength


/*// Calculates the fixed dark value based on the integration time of each spectra
float Fixed_Dark(float intTime){
float temp = 725.0+.08125*((float)intTime-100.0);    // These numbers were obtained through testing
the dark current, excel
return temp;
} // End of Fixed Dark
*/
void divideIntTime(int SpecNum, float data[SPEC_CHANNELS], uint16_t intTime, int x){

for (int i = 0; i < SPEC_CHANNELS; i++) {
data[i] = (data[i]/(float)intTime);
}


// Return data to the correct array if (SpecNum == 1 && x == 1) {
for (int i = 0; i < SPEC_CHANNELS; i++) {
spec1DataUp[i] = data[i];
}
}
if (SpecNum == 2 && x == 2) {
for (int i = 0; i < SPEC_CHANNELS; i++) {
spec2DataDown[i] = data[i];
}
}
} // End of divide by intTime


void spectraRatio(int sR, float data1[SPEC_CHANNELS], float data2[SPEC_CHANNELS]){
if (sR == 1) {
```

```
for(int i = 0; i < SPEC_CHANNELS; i++){
albedo[i] = (data2[i]/data1[i]);
}
}
} // End of spectraRatio


void readSpectrometer(int SpecNum, int x) // Spectrometer (1 or 2), spectra
(first or second)
{
uint16_t intTimeNew;              // Initialize integration time variable, for
Step 3 and auto gain.
uint16_t PeakSpec = 0;          // Initialize Spectral peak variable to determine integration time
variable.
uint16_t Dark = 0;                // Initialize dark variable, changes based on
spectrometer being used.
bool PeakCheck = false;         // Initialize the while loop to find spectra peak value.
//int delay_time = 35;          // delay per half clock (in microseconds).   This ultimately controls the
integration time.
int delay_time = 1;               // delay per half clock (in microseconds).   This
ultimately controls the integration time.
int read_time = 35;               // Amount of time that the analogRead()
procedure takes (in microseconds) (different micros will have different times)
int accumulateMode = false; float data[SPEC_CHANNELS]; int idx = 0;
int k = 0;
int SPEC_CLK = 0;
int SPEC_VIDEO = 0;


// Determine which spectrometer is in use to set variables to match:


if (SpecNum == 1) { SPEC_CLK = SPEC_CLK1; SPEC_VIDEO = SPEC_VIDEO1;
Dark = Dark1;   // Dark 1 is background and comes from global variables.
intTimeNew = intTimeSpec1;
}
else if (SpecNum == 2) { SPEC_CLK = SPEC_CLK2; SPEC_VIDEO = SPEC_VIDEO2;
Dark = Dark2; //   Dark 2 is background and comes from global variables.
intTimeNew = intTimeSpec2;
}


while ((PeakCheck == false) && (k < 5)) {
k++;
// Step 1: start leading clock pulses
for (int i = 0; i < SPEC_CHANNELS; i++) { digitalWrite(SPEC_CLK, LOW);
delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH);
delayMicroseconds(delay_time);
}


// Step 2: Send start pulse to signal start of integration/light collection
digitalWrite(SPEC_CLK, LOW); delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH);
digitalWrite(SPEC_ST, LOW); delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, LOW);
```

```
delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH); digitalWrite(SPEC_ST, HIGH);
delayMicroseconds(delay_time);

// Step 3: Integration time -- sample for a period of time determined by the intTime parameter
int blockTime = delay_time * 8;
long int numIntegrationBlocks = ((long)intTimeNew * (long)1000) / (long)blockTime;
for (int i = 0; i < numIntegrationBlocks; i++) {
// Four clocks per pixel
// First block of 2 clocks -- measurement digitalWrite(SPEC_CLK, LOW);
delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH); delayMicroseconds(delay_time);
digitalWrite(SPEC_CLK1, LOW); delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH);
delayMicroseconds(delay_time);

digitalWrite(SPEC_CLK, LOW); delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH);
delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, LOW); delayMicroseconds(delay_time);
digitalWrite(SPEC_CLK, HIGH); delayMicroseconds(delay_time);
}



// Step 4: Send start pulse to signal end of integration/light collection digitalWrite(SPEC_CLK,
LOW);
delayMicroseconds(delay_time);
digitalWrite(SPEC_CLK, HIGH); digitalWrite(SPEC_ST, LOW); delayMicroseconds(delay_time);
digitalWrite(SPEC_CLK, LOW); delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH);
digitalWrite(SPEC_ST, HIGH); delayMicroseconds(delay_time);

// Step 5: Read Data 2 (this is the actual read, since the spectrometer has now sampled data)
idx = 0;
for (int i = 0; i < SPEC_CHANNELS; i++) {
// Four clocks per pixel
// First block of 2 clocks -- measurement digitalWrite(SPEC_CLK, LOW);
delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH); delayMicroseconds(delay_time);
digitalWrite(SPEC_CLK, LOW);

// Analog value is valid on low transition if (accumulateMode == false) {
//          unsigned long stTime = micros() ;
data[idx] = analogRead(SPEC_VIDEO);
//          unsigned long eTime = micros() ;
//          unsigned long del=eTime-stTime;
//          Serial.print("microsecs for read =");
//          Serial.println(del)     ;


} else {
data[idx] += analogRead(SPEC_VIDEO);
}
idx += 1;
if (delay_time > read_time) delayMicroseconds(delay_time - read_time);     // Read takes about
135uSec

digitalWrite(SPEC_CLK, HIGH);
delayMicroseconds(delay_time);
```

```
// Second block of 2 clocks -- idle digitalWrite(SPEC_CLK, LOW); delayMicroseconds(delay_time);
digitalWrite(SPEC_CLK, HIGH); delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, LOW);
delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH); delayMicroseconds(delay_time);
}


// Step 6: trailing clock pulses
for (int i = 0; i < SPEC_CHANNELS; i++) { digitalWrite(SPEC_CLK, LOW);
delayMicroseconds(delay_time); digitalWrite(SPEC_CLK, HIGH);
delayMicroseconds(delay_time);
}
// Return data to the correct array if (SpecNum == 1 && x == 1) {
for (int i = 0; i < SPEC_CHANNELS; i++) {
spec1DataUp[i] = data[i];
}
}
if (SpecNum == 2 && x == 2) {
for (int i = 0; i < SPEC_CHANNELS; i++) {
spec2DataDown[i] = data[i];
}
}


// Auto Gain


if (AutoGain == false) return; // When Autogain is not used, just return the measured spectra.

PeakSpec = 0;
for (int j = 0; j < SPEC_CHANNELS; j++) {    // Finds the peak value in the spectra.
if (data[j] > PeakSpec) {
PeakSpec = data[j];
}
}
//if (PeakSpec >= 5000 && PeakSpec <= 6500) { // if Spectral peak is ok, then don't change the
integration time
if (PeakSpec >= 3000 && PeakSpec <= 3470) { // Changed for low gain, to be below saturation,
and reduced range with hope for more consistent albedos
if (SpecNum == 1 && x == 1) {
integrationTime1_1 = intTimeNew;
kk1 = k;
//intTimeSpec1 = intTimeNew;
}
if (SpecNum == 2 && x == 2) { integrationTime2_2 = intTimeNew; kk2 = k;
//intTimeSpec2 = intTimeNew;
}


if (SpecNum == 1 && x == 1) {
for (int i = 0; i < SPEC_CHANNELS; i++) {
spec1DataUp[i] = data[i];
}
}
if (SpecNum == 2 && x == 2) {
```

```
for (int i = 0; i < SPEC_CHANNELS; i++) {
spec2DataDown[i] = data[i];
}
}
PeakCheck = true;                          // This will exit the While loop
with good data.
}
else {
if (k == 5){ // Unsuccessful in finding a new integration time.   Reset to the last value.
if (SpecNum == 1 && x == 1) {
integrationTime1_1 = intTimeNew;
//intTimeSpec1 = intTimeNew;
}
if (SpecNum == 2 && x == 2) {
integrationTime2_2 = intTimeNew;
//intTimeSpec2 = intTimeNew;
}
if (SpecNum == 1 && x == 1) {
for (int i = 0; i < SPEC_CHANNELS; i++) {
spec1DataUp[i] = data[i];
}
}
if (SpecNum == 2 && x == 2) {
for (int i = 0; i < SPEC_CHANNELS; i++) {
spec2DataDown[i] = data[i];
}
}
} // End k == 5
float intTimeNewF = (float)intTimeNew * (float)(3200 - Dark) / (float)(PeakSpec - Dark + 0.01); //
if Spectral peak is not ok, then change integration time based on equation
intTimeNew = (uint16_t)abs(intTimeNewF);    // Protected against zero in
denomenator
intTimeNew = constrain(intTimeNew, 1, 3000);
if (PeakSpec >= 3475) intTimeNew = 2; // Setting Integration time when massive overload might
have happened.
}


}     // End while loop to check Auto Gain


}     // End of readSpectrometer()


/******** Everything else subroutines ********/


void sendByte()
{
dataState = HIGH; itoa(dataState, dataCom, 10); Serial5.write(dataCom); Serial.println(dataCom);
delay(100);
dataState = LOW;
}
```

```
void saveRAWSpecData(float wavelength[SPEC_CHANNELS], float Spec1_1[SPEC_CHANNELS],
float Spec2_2[SPEC_CHANNELS], uint16_t integrationTime1_1, uint16_t integrationTime2_2){
File myFile;
myFile = SD.open("RawData.txt", FILE_WRITE);
if(myFile)
{
myFile.println("\n"); myFile.print("RTC Date & Time:"); if(month() < 10) myFile.print('0');
myFile.print(month()); myFile.print("/");
if(day() < 10) myFile.print('0'); myFile.print(day()); myFile.print("/"); myFile.print(year());
myFile.print("  ");
if(hour() < 10) myFile.print('0');
myFile.print(hour());
myFile.print(":");
if(minute() < 10) myFile.print('0'); myFile.print(minute()); myFile.print(":");
if(second() < 10) myFile.print('0');
myFile.println(second()); myFile.print("Integration Time for Spec 1 UP: ");
myFile.println(integrationTime1_1); myFile.print("Integration Time for Spec 2 DOWN: ");
myFile.println(integrationTime2_2); myFile.println("Wavelength, Spectra1_Up, Spectra2_Down"); for(int
i=0; i<SPEC_CHANNELS; i++)
{
myFile.print(wavelength[i]); myFile.print(","); myFile.print("  "); myFile.print(Spec1_1[i],5);
myFile.print(","); myFile.print("  "); myFile.print(Spec2_2[i],5); myFile.println("\n");
}
}
myFile.close();
}


void getAmbientTemp()
{
ambientTemp = mlx.readAmbientTempC();
}
void getObjectTemp()
{
objectTemp = mlx.readObjectTempC();
}
void getTemp()
{
bme.takeForcedMeasurement();
temp = bme.readTemperature();
}
void getHumidity()
{
bme.takeForcedMeasurement();
humid = bme.readHumidity();
}
void getPressure()
{
bme.takeForcedMeasurement();
pressure = (bme.readPressure() / 100.0F);
}
void getAltitude()
{
```

```
bme.takeForcedMeasurement();
alt = bme.readAltitude(SEALEVELPRESSURE_HPA);
}


void gpsdump(TinyGPS &gps)
{
long lat, lon;
float flat, flon;
unsigned long age, date, time, chars;
int year;
byte month, day, hour, minute, second, hundredths;
unsigned short sentences, failed; gps.get_position(&lat, &lon, &age); gps.f_get_position(&flat, &flon,
&age); gps.get_datetime(&date, &time, &age);
gps.crack_datetime(&year, &month, &day, &hour, &minute, &second,
&hundredths, &age);


gpslat = flat                                                    ; gpslon = flon
; gpsalt     = gps.f_altitude()                                  ; gpsspeed = gps.f_speed_kmph()
; gpscourse = gps.f_course()                                     ; gpsyear     = year
; gpsmon       = static_cast<int>(month)                         ; gpsday       = static_cast<int>(day)
; gpshour     = static_cast<int>(hour)                          ; gpsmin       = static_cast<int>(minute)
;
gpssec       = static_cast<int>(second)                        ;



gps.stats(&chars, &sentences, &failed);
}


void getGPS()
{
bool newdata = false                    ; unsigned long startup = millis() ; while (millis() - startup <
1000L) {
if (HWSERIALgps.available()) { char c = HWSERIALgps.read(); if (gps.encode(c)) {
newdata = true       ;
break                    ; /* Obtained data, bail out of the while loop */
}
}


}
if (newdata) gpsdump(gps);
/*Serial.print(gpslat, 5); Serial.print(","); Serial.print(gpslon, 5); Serial.print(","); Serial.print(gpsalt);
Serial.print(","); Serial.print(gpsspeed); Serial.print(","); Serial.println(gpscourse);
*/
}


void saveData()
{
filename[0] = hour()/10 + '0'; filename[1] = hour()%10 + '0'; filename[2] = '-';
```

```
filename[3] = minute()/10 + '0'; filename[4] = minute()%10 + '0'; filename[5]= '-';
filename[6] = second()/10 + '0';
filename[7] =second()%10 + '0';


myFile = SD.open(filename, FILE_WRITE);
if (myFile)
{
myFile.println("GPS_Date,GPS_Time, Latitude, Longitude, Altitude, Course_Degrees");
myFile.print(gpsday);
myFile.print("/"); myFile.print(gpsmon); myFile.print("/"); myFile.print(gpsyear);
myFile.print(","); myFile.print(gpshour); myFile.print(":"); myFile.print(gpsmin); myFile.print(":");
myFile.print(gpssec); myFile.print(","); myFile.print(gpslat); myFile.print(","); myFile.print(gpslon);
myFile.print(","); myFile.print(gpsalt); myFile.print(","); myFile.println(gpscourse); myFile.print("\0");

myFile.print("RTC Date & Time:"); if(month() < 10) myFile.print('0'); myFile.print(month());
myFile.print("/");
if(day() < 10) myFile.print('0'); myFile.print(day()); myFile.print("/"); myFile.print(year());
myFile.print("    ");
if(hour() < 10) myFile.print('0'); myFile.print(hour()); myFile.print(":");
if(minute() < 10) myFile.print('0'); myFile.print(minute()); myFile.print(":");
if(second() < 10) myFile.print('0'); myFile.println(second()); myFile.print("Temperature: ");
myFile.println(temp); myFile.print("Pressure: "); myFile.println(pressure); myFile.print("Humidity: ");
myFile.println(humid); myFile.print("Altitude: "); myFile.println(alt); myFile.print("Ambient Temp: ");
myFile.println(ambientTemp); myFile.print("Object Temp: "); myFile.println(objectTemp);
myFile.println("Roll & Pitch (roll, pitch): ");
myFile.print("Before Spec1 UP: "); myFile.print("("); myFile.print(RollBeforeSpec1UP);
myFile.print(","); myFile.print(PitchBeforeSpec1UP); myFile.println(")"); myFile.print("After Spec1 UP:
"); myFile.print("("); myFile.print(RollAfterSpec1UP); myFile.print(",");
myFile.print(PitchAfterSpec1UP); myFile.println(")"); myFile.print("Before Spec2 DOWN: ");
myFile.print("("); myFile.print(RollBeforeSpec2DOWN); myFile.print(",");
myFile.print(PitchBeforeSpec2DOWN); myFile.println(")"); myFile.print("After Spec2 DOWN: ");
myFile.print("("); myFile.print(RollAfterSpec2DOWN); myFile.print(",");
myFile.print(PitchAfterSpec2DOWN); myFile.println(")");
myFile.print("Dark Spectrometer 1 (UP): ");
myFile.println(Dark1_1);
myFile.print("Dark Spectrometer 2 (DOWN): ");
myFile.println(Dark2_2);
myFile.print("K Spectrometer 1 (UP): ");
myFile.println(kk1);
myFile.print("K Spectrometer 2 (DOWN): ");
myFile.println(kk2);
myFile.println("");


myFile.println("Wavelength, Surface_Albedo, Albedo_uncertainty, Spectra1_Up, Spectra2_Down");
for(int i = 0; i < SPEC_CHANNELS; i++)
{
myFile.print(wavelength1[i]); myFile.print(","); myFile.print("          "); myFile.print(albedo[i],5);
myFile.print(","); myFile.print("          ");
myFile.print(AlbedoUncertainty[i], 5);
myFile.print(",");
myFile.print("          "); myFile.print(spec1DataUp[i],5); myFile.print(","); myFile.print("          ");
myFile.print(spec2DataDown[i],5);
```

```
myFile.println("");
}
myFile.println("\n");
} else { Serial.println("Error!");
}
myFile.close();
}


void takePicture()
{
if (cam.begin()) { Serial.println("Camera Found:");
} else {
Serial.println("No camera found?");
return;
}


// delay(100);
delay(3000);


if (! cam.takePicture()) Serial.println("Failed to snap!");
else
Serial.println("Picture taken!");


char imgfile[] = "00000000.JPG";   // Camera for (int i = 0; i < 1; i++) {
imgfile[0] = hour()/10 + '0'; imgfile[1] = hour()%10 + '0'; imgfile[2] = '_';
imgfile[3] = minute()/10 + '0'; imgfile[4] = minute()%10 + '0'; imgfile[5]= '_';
imgfile[6] = second()/10 + '0';
imgfile[7] =second()%10 + '0';
if (! SD.exists(imgfile)) {
break;
}
}
myFile = SD.open(imgfile, FILE_WRITE);
uint16_t jpglen = cam.frameLength();


while (jpglen > 0) {
uint8_t *buffer;
uint8_t bytesToRead = min(64, jpglen); // change 32 to 64 for a speedup but may not work with
all setups!
buffer = cam.readPicture(bytesToRead);
myFile.write(buffer, bytesToRead);
jpglen -= bytesToRead;
}
myFile.close();
}


time_t getTeensy3Time()
{
```

```
return Teensy3Clock.get();
}


unsigned long processSyncMessage() {
unsigned long pctime = 0L;
const unsigned long DEFAULT_TIME = 1357041600; // Jan 1 2013
}

// end Albedo_3.6_UAV
```

```
// Ground control device code
// Radio communication

#include <Adafruit_GFX.h>            /* For the display */
#include <Adafruit_PCD8544.h>    /* For the display */
#include <SPI.h>    /* Serial Communication */
#include <Wire.h> /* I2C Communication */


#define SPEC_CHANNELS 256
Adafruit_PCD8544 display = Adafruit_PCD8544(8, 6, 4, 3, 2); // For the LCD display void
recAlbedo();
void recvWithStartEndMarkers();
bool printAlbedo = false;


const int buttonPin = 10;
int buttonState = 1;
char buttonCom[5] = {0};
char darkCurrentCom[5] = {0};
int darkCurrentState = 0;


//Temp Arrays for receiving data const byte numChars = 32;
char receivedChars[numChars];
char tempChars[numChars];
//char ab[84];
char measType = '0';


//Initilization of Parsed data
char messageFromPC[numChars] = {0};
int integerFromPC = 0; char dataCom[5] = {0}; int dataState = 0; uint16_t xPixel[84]; uint16_t
yPixel[84];
float wavelength[SPEC_CHANNELS];


//float rollDark = 0.0;
float rollSpec1; float pitchSpec1; float albedo1; float albedo2[72]; String Status; float OrigPeak;
```

```
boolean newData = false; boolean newByte = false; boolean isLevel = false; boolean GotPeak = false;

void setup() {


pinMode(buttonPin, INPUT);


display.begin(); display.setContrast(60); display.setTextSize(1); display.setTextColor(BLACK);
Serial.begin(9600); Serial1.begin(9600);

} // End Set Up void loop() {
display.clearDisplay();    //For the display display.setCursor(0,0);    //For the display
display.println("Press the\nbutton \nto begin \nmeasurements"); display.display();

buttonState = digitalRead(buttonPin);
while (buttonState == LOW)
{
Serial.println("button pressed"); itoa(buttonState, buttonCom, 10); buttonState = LOW;
Serial.println(buttonCom); Serial1.write(buttonCom);

//Read in everything while(isLevel == false){
measType = 'z';
recvWithStartEndMarkers();
if (newData == true){ strcpy(tempChars, receivedChars); float NumberCode = atof(tempChars);

if (NumberCode == 45){ display.clearDisplay();    //For the display display.setTextSize(1);
display.setCursor(12,20);    //For the display display.println("Dark Taken"); display.display();
newData = false;
}
if (NumberCode == 50){ display.clearDisplay();    //For the display display.setTextSize(1);
display.setCursor(25,15);    //For the display display.println("Begin\n    Leveling..."); display.display();
newData = false;
}
if (NumberCode == 60){ display.clearDisplay();    //For the display display.setTextSize(1);
display.setCursor(25,15);    //For the display display.println("Saving\n      Photo..."); display.display();
newData = false;
}
if (NumberCode == 70){
display.clearDisplay();    //For the display
display.setTextSize(1); display.setCursor(8,20);    //For the display display.println("Flip over...");
display.display();
newData = false;
}
if (NumberCode == 80){ display.clearDisplay();    //For the display display.setTextSize(1);
display.setCursor(12,15);    //For the display display.println("Saving Raw\n        Data...");
display.display();
newData = false;
}
if (NumberCode == 90){ display.clearDisplay();    //For the display display.setTextSize(1);
display.setCursor(12,15);    //For the display display.println("Calculating\n        Albedo...");
display.display();
newData = false;
```

```
break;
}
if (NumberCode == 111){ display.clearDisplay();    //For the display display.setCursor(0,0);    //For
the display display.setTextSize(1); display.println("Taking\nSpectrometer\nmeasurement");
display.setTextSize(2);
display.setCursor(20,29);    //For the display display.print("1 UP");
display.display();
newData = false;
}
if (NumberCode == 221){ display.clearDisplay();    //For the display display.setCursor(0,0);    //For
the display display.setTextSize(1); display.println("Taking\nSpectrometer\nmeasurement");
display.setTextSize(2);
display.setCursor(8,29);    //For the display display.print("2 DOWN");
display.display();
newData = false;
}
if (NumberCode == 121){ display.clearDisplay();    //For the display display.setCursor(0,0);    //For
the display display.setTextSize(1); display.println("Taking\nSpectrometer\nmeasurement");
display.setTextSize(2);
display.setCursor(8,29);    //For the display display.print("1 DOWN");
display.display();
newData = false;
}
if (NumberCode == 211){ display.clearDisplay();    //For the display display.setCursor(0,0);    //For
the display display.setTextSize(1); display.println("Taking\nSpectrometer\nmeasurement");
display.setTextSize(2);
display.setCursor(20,29);    //For the display display.print("2 UP");
display.display();
newData = false;
}
}
}
while(GotPeak == false){ measType = 'q'; recvWithStartEndMarkers(); if (newData == true){
strcpy(tempChars, receivedChars); OrigPeak = atof(tempChars);
GotPeak = true;
}
}


int z = 0;
float peak = 0;
while(z<72)
{
measType = 'a'; recvWithStartEndMarkers(); if (newData == true)
{
//recvWithStartEndMarkers(); strcpy(tempChars, receivedChars); albedo1 = atof(tempChars); albedo2[z]
= albedo1;
z++;
//memset(tempChars, 0, sizeof(tempChars));
//memset(receivedChars, 0, sizeof(receivedChars));
delay(10);
newData = false;
}
}
```

```
//Display albedo
display.clearDisplay();    //For the display display.setCursor(0,41);    //For the display
display.setTextSize(1); display.setTextColor(BLACK); display.print("Peak=");
display.println(OrigPeak);          // Displays Peak value on screen
for (int i = 0; i < 72; i++){ Serial.println(albedo2[i]);
display.drawPixel(i, albedo2[i], BLACK);    // draw a single pixel
}
display.display();
delay(2500); // Hold Albedo plot on display for 2.5 Seconds buttonState = HIGH;
}
} // End Loop


//Function that recieves data with starting marker * and end marker &
void recvWithStartEndMarkers() {
static boolean recvInProgress = false;
static byte ndx = 0;
char startMarker = measType;
char endMarker = '&';
char rc;


while (Serial1.available() > 0 && newData == false) {
rc = Serial1.read();


if (recvInProgress == true) {
if (rc != endMarker) { receivedChars[ndx] = rc; ndx++;
if (ndx >= numChars) {
ndx = numChars - 1;
}
}
else {
receivedChars[ndx] = '\0'; // terminate the string recvInProgress = false;
ndx = 0;
newData = true;
}
}
else if (rc == startMarker) {
recvInProgress = true;
}
}
}


void recByte()
{
int i = 0;
if(Serial1.available() >0 && newByte == false)
{
while (Serial1.available() && i<5)
{
dataCom[i++] = Serial1.read();
}
```

```
dataCom[i++] = '\0';
newByte = true;
}
if (newByte == true)
{
dataState = atoi(dataCom);
}
}

// End ground control device code
```

Appendix 3 Albedo measurements at each altitude.

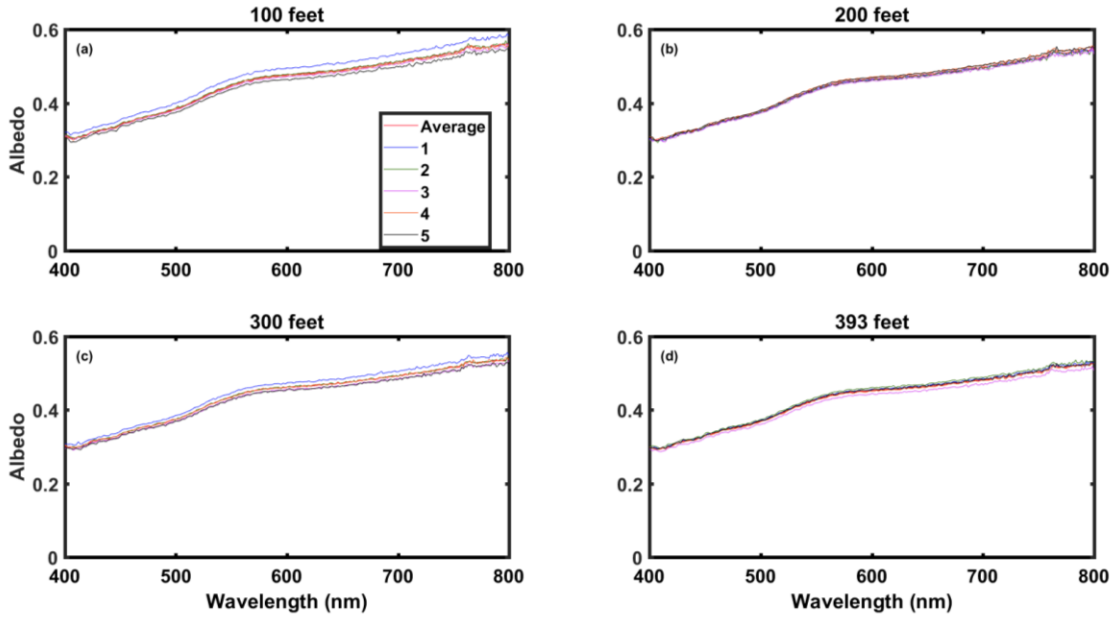


Figure A2: Albedo measurements obtained from UAS on October 5th, 2017 in Nevada's Black Rock Desert over road location. (a.) Measurements obtained 100 ft AGL. (b.) Measurements obtained 200 ft AGL. (c.) Measurements obtained 300 ft AGL. (d.) Measurements obtained 393 ft AGL.

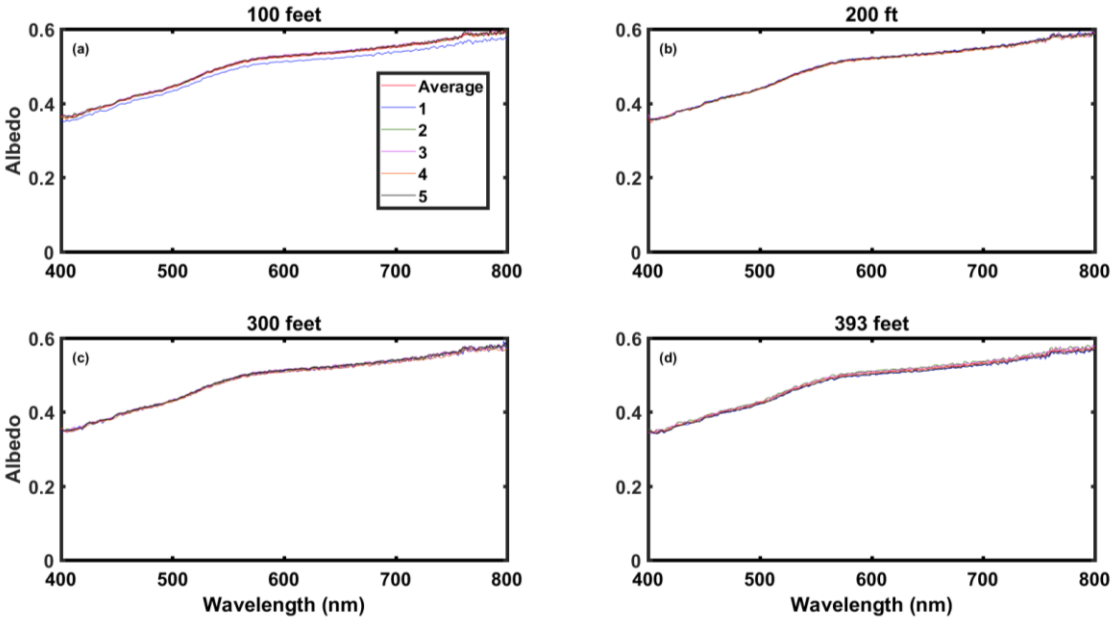


Figure A3: Albedo measurements obtained from UAS on October 5th, 2017 in Nevada's Black Rock Desert over nonroad location. (a.) Measurements obtained 100 ft AGL. (b.) Measurements obtained 200 ft AGL. (c.) Measurements obtained 300 ft AGL. (d.) Measurements obtained 393 ft AGL.

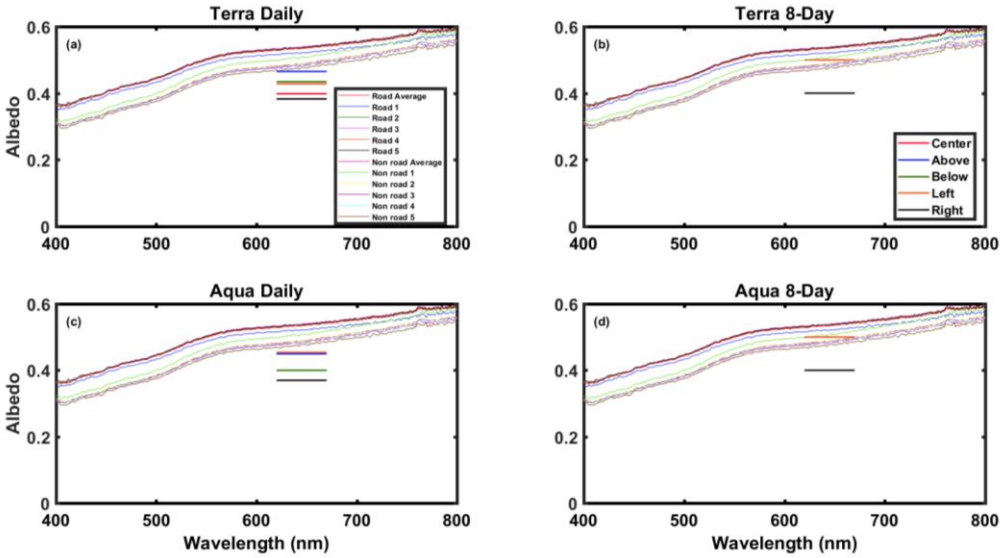


Figure A4: MODIS (MOD/MYD09GQ) and MODIS 8-day best (MOD/MYD09Q1) band 1 (620-670 nm) surface reflectance (250 m resolution) for Terra (a and b) and AQUA (c and d) comparison to measured values over road and nonroad locations. MODIS pixels directly over the measurement site are designated as "center" while immediately neighboring pixels are taken to be "above", "below", "left", and "right" in relation to the center pixel.

Appendix 4 Ground instantaneous field of view (GIFOV).

$$D \approx 2h \tan(\theta). \text{ (eq. A1)}$$

In equation A1, D is the ground field of view, h is the height above ground level, and 2θ is the angular field of view of the detector.

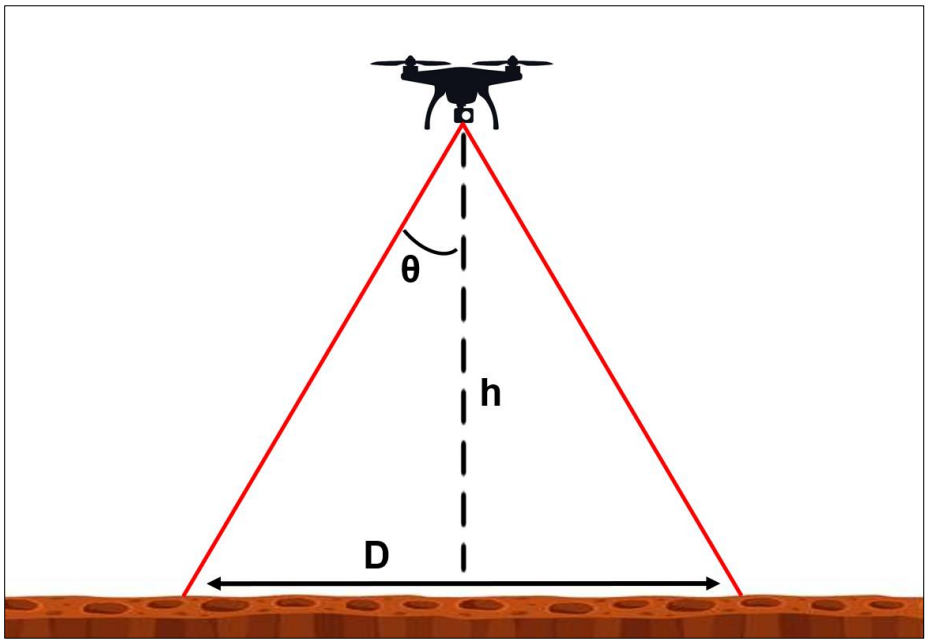


Figure A5: Diagram of the detector field of view while flying. A change in height (h) above the ground level will affect the spatial area that the instrument senses at the surface (D).