Interfacing and LIDAR Implementation for Space Communication

Francisca Vasconcelos*[1] and Jose Velazco†[2]

[1]Massachusetts Institute of Technology

[2]NASA Jet Propulsion Lab, Caltech

August 21, 2017

Abstract

The work is split among two projects: a novel amplification system for the Deep Space Network (DSN) and a new means of optical communication between CubeSats. The first project, a Spatial Power Combining Amplifier (SPCA), will eventually require 20,000 small amplifiers working in unison to produce several GHz of power. These amplifiers will autonomously self-regulate and connect to a central framework, accessible by operators of DSN antennas. In the development of a prototype of this system, we focus on the creation of a graphical user interface that allows operators to visualize and alter the state of individual or several amplifiers. This interface has been implemented as a React webpage, running on a node.js server with a MongoDB database. The second project, an Inter-Satellite Omnidirectional Optical Communicator (ISOOC), will be used to establish up to thirty optical communication links between CubeSats in flock formations. This will require precise regulation of relative distance and velocity. In the development of a prototype of this system, we focus on the creation of a LIDAR system that can scan a 3D region to create a depth map. The system consists of a laser, steerable MEMs mirror, and receiver.

*francisc@mit.edu

†Jose.E.Velazco@jpl.nasa.gov

Contents

1 Spatial Power Combining Amplifier [SPCA]

1.1 Background

Currently, the Deep Space Network (DSN) antennas at Goldstone, Canberra, and Madrid generate their beams through amplification by klystrons. These large machines are comprised of old, limited technology. When one breaks, it can take weeks and millions of dollars to repair. For this reason, we propose the Spatial Power Combining Amplifier (SPCA). Rather than a few large klystron amplifiers, this system consists of 20,000 small amplifiers, which will self-regulate and be easily replaceable.

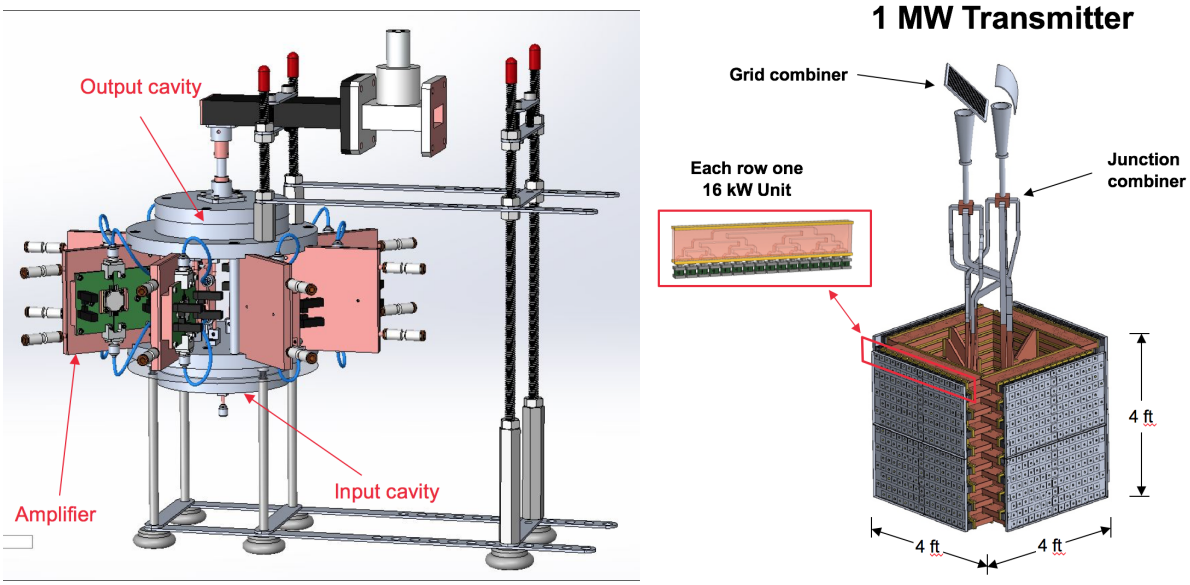

Figure 1: SPCA system currently under development at JPL.

Emerging SPCA technologies offer many advantages over klystrons and traveling-wave tube amplifiers in common use. SPCAs have a lower operating voltage, which allows them to consume less power and be supplied by less complicated power sources. The structure of SPCAs avoid RF breakdown and thermal issues prevalent in klystrons and traveling wave tubes, and thus offer improved reliability a more graceful degradation. SPCAs also circumvent the issue of the solid state power combiners of the microstrip boards becoming single point failures. SPCAs can achieve stable and reliable high-power microwave amplification in a compact structure through the use of reliable solid-state devices. SPCAs are four times smaller than comparable klystrons and traveling-wave tubes.

SPCA designs have the potential to produce X-band (8.4 GHz), Ka-band (32 GHz), and W-band

(94 GHz) for both radar and communication applications. They offer a factor of four reduction in size that could potentially revolutionize phase array radar, communications, food processing, and microwave materials processing. All these applications cut through several arenas including science, the military, and industry. SPCAs could also be the key to wireless power beaming, which could potentially be implemented to beam power to small spacecraft or planetary rovers, in addition to the many potential applications in commercial market.

1.2 Project Goals

The overall objectives for the project are to integrate SPCA components with a network of microcontrollers and program the microcontrollers appropriately. The SPCA under development at JPL (see figure 1) has 18 smart board modules, each consisting of a solid-state amplifier, other RF components and a microcontroller. These microcontrollers will read relevant parameters from each board (temperature of the amplifier, gain, condition of each RF component, etc.), then send these data to a master microcontroller. We must program the microcontrollers on each smart board to read all the parameters of interest, program the master microcontroller, network all the microcontrollers, and develop a graphic user interface to monitor the data amassed from all the microcontrollers. The GUI will be programmed to allow real-time monitoring of the data delivered by each microcontroller and to provide statistical and time history information of the logged parameters.

The GUI is the primary focus of this literature.

1.3 System Architecture

We were tasked with creating a prototype of the system that will eventually be used to meet all these specifications. This led to several design choices and the development of a system architecture.

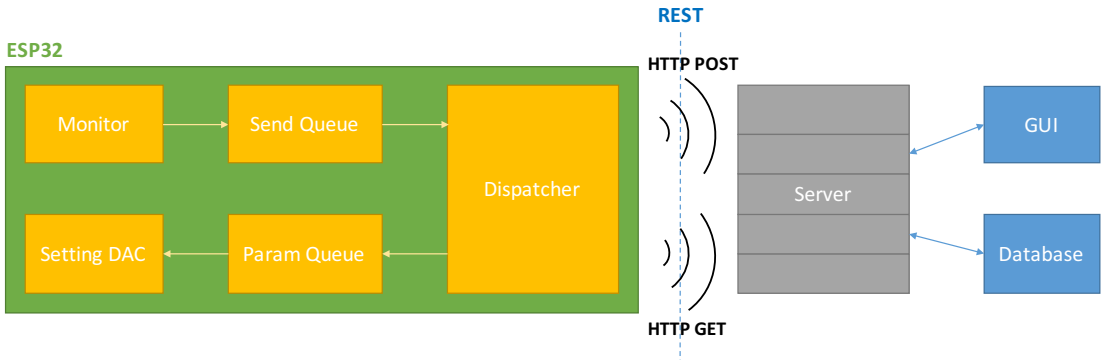

Figure 2: The system architecture of our SPCA prototype.

In order to monitor the amplifiers and enable communication to a central server, the ESP32 was chosen as a regulatory microcontroller. Each amplifier has an associated ESP32, meaning when there are eventually 20,000 amplifiers, there will be 20,000 ESP32s. These ESP32 boards can communicate with one another, as well as a central server via HTTP GET and POST requests. This server is implemented in node.js. While it would be ideal to have the ESP32s communicate via WiFi to the server (as it is currently implemented), power noise concerns may eventually result in a switch to physical ethernet connections.

The server also connects to the GUI and database. The database is implemented in MongoDB and is used to store parameter value data from the ESP32s. The GUI is implemented as a React webpage, so that it is accessible anywhere, on any platform.

1.4 Graphical User Interface

The current version of the GUI is divided into three pages, which are described in detail in the following sections. It was programmed in React, a language created by Facebook that combines Javascript, HTML, and CSS. React was chosen for its expandability, increasing popularity, and speed. Plotting is implemented with Recharts, a React library. The Codecademy ReactJS Tutorial I was used to learn the language, in order to program the GUI.

1.4.1 Health Monitor Page

The Health Monitor Page displays the overall condition of the system and allows the operator to adjust the ESP32 parameters.

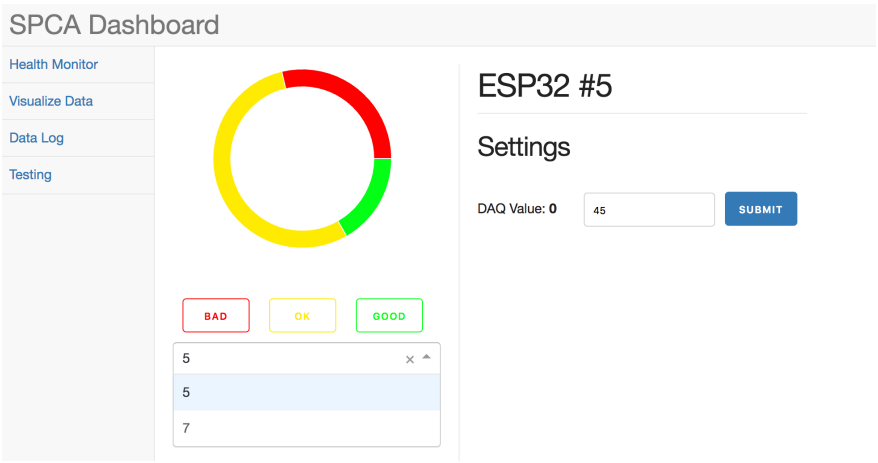

Figure 3: A screenshot of the Health Monitor Page, with ESP32 ID 5 selected.

Since eventually 20,000 amplifiers will be in use, there needs to be an efficient way to know which ESP32s are in need of maintenance. This functionality is implemented via a color-coated pie-chart and menu. The pie-chart allows the operator to quickly understand the relative number of ESP32 that are in"Good", "OK", and "Bad" condition, providing him/her with a sense of the overall health of the system. There is a search bar below that allows the operator to search among all the ESP32 IDs. The three buttons above the search bar limit the searchable space to the ESP32s with the corresponding condition. Since the operator will most likely be using the tool to manipulate ESP32s in poor condition, this allows him/her to quickly see a list of the amplifiers urgently requiring maintanence.

Upon clicking on an ESP32 ID, another menu opens, corresponding to only that ESP32. This menu allows the operator to change the DAC value of the ESP32. This has been tested and verified to work with a voltmeter. As more parameters of the board are connected to the GUI, this menu will be expanded to allow modification of those values as well.

1.4.2 Data Visualization Page

The Data Visualization Page allows the operator to see plots of different parameter values over time (for a specific or several ESP32s). It is split into two modes, real-time and historic, to allow the operator to either monitor the current state of the system or look back on previous states.

Both modes contain an ESP32 ID selection menu. This drop down lists all the ESP32 IDs and narrows the list as values are typed in the search bar. The user can select as many ESP32 IDs as they would like and the corresponding values display in the plot. A random color generator is used to select a color for each selected ESP32 ID. There is also a drop down that allows the user to select which parameter they would like to plot. This feature has not been fully implemented since currently only one parameter value is being sent from the ESP32. However, the structure is there and should be easily adjustable.

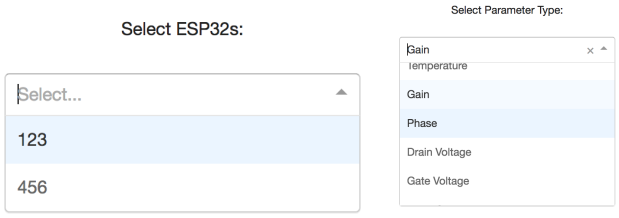

Figure 4: Screenshots of the ESP32 ID and parameter search/selection dropdowns.

The real-time plot is the default when the page is loaded. The time settings are greyed out (disabled), since the current values will be used. As soon as the user selects at least one ESP32 ID and a parameter, the values begin displaying. An animation moves the incoming points across the plot until 20 values are plotted, in which case they just begin sliding across the screen. If no value is received from the ESP32, a null point is plotted, allowing the operator to distinguish missing data from lag.

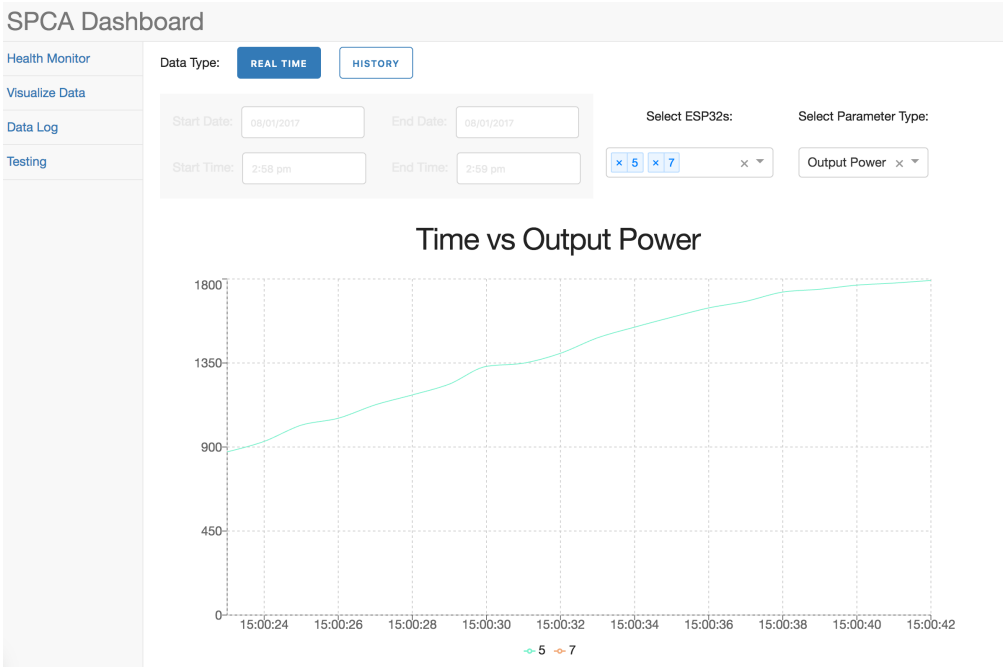

Figure 5: A screenshot of the realtime plotting of the Data Visualization Page (only ESP32 5 is connected).

When the button for the historical data plot is toggled, the previously greyed-out menu with time settings is unlocked. This allows the user to select a time range of data to display on the plot. The default is the most recent minute, but the user can adjust the start date and time as well as the end date and time. Depending on the range and number of ESP32s selected, this data may take a while to load from the database and display, so a loading symbol is used to let the operator know that it is working. When the plot displays, all the points in the range are displayed. However this could be thousands of points, so four buttons were added to the bottom of the graph: pan left (<), zoom in (+), zoom out (-), and pan right (>).

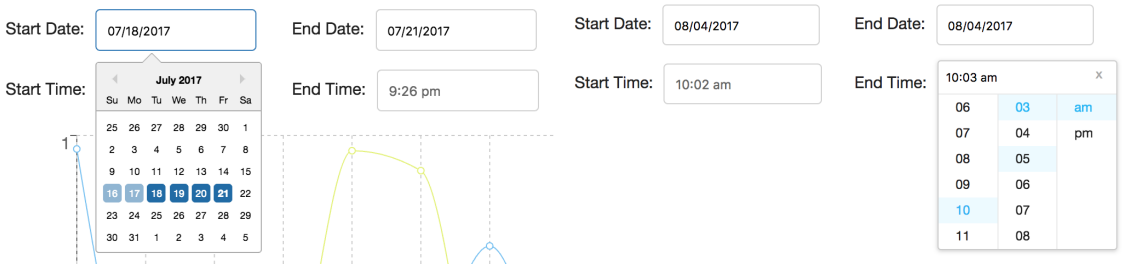

Figure 6: The date and time selection menus of the historical data plot.

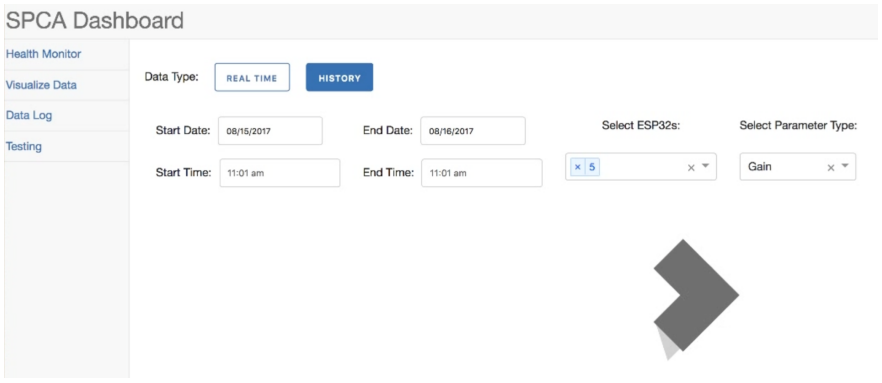

Figure 7: A screenshot of the historical data plot loading.

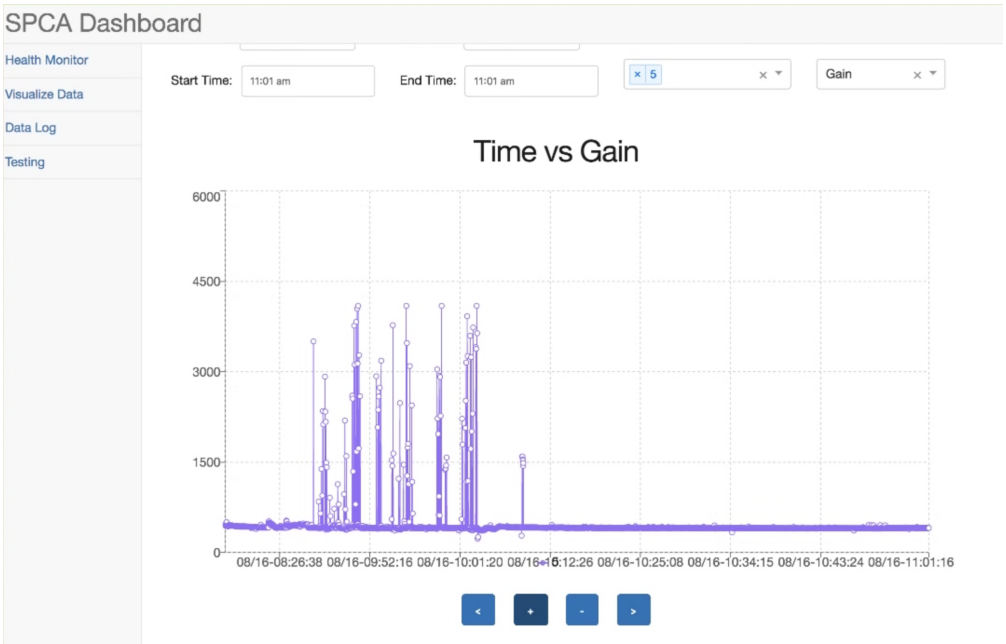

Figure 8: A screenshot of the historical data plotting of the Data Visualization Page.

1.4.3 Data Log Page

The Data Log Page provides the operator access to raw data and allows him/her to pull data from the database. It is split into two modes, real-time and historic, to allow the operator to either monitor the current state of the system or look back on previous states.

In both cases, there is a drop down menu that allows the user to select the ESP32 ID of the device they would like to see data from (note that only one can be selected at a time). Both also contain an "Export to CSV" button, which downloads a formatted CSV containing all of the data being displayed in the table. There is a "# of Rows" textbox that allows the user to specify the number of rows that should show up in the table. This allows the user to also specify the number of values to be output to the CSV.

	A	B	C	D	E	F	G	H	I	J
1	key_id	var0	var1	var2	var3	var4	var5	var6	var7	
2	0	Time	Output Powe	Temperature	Gain	Phase	Drain Voltag	Gate Voltage	Gate Current	
3	2017-08-16T	8/16/17 10:58	452	452	452	452	452	452	452	
4	2017-08-16T	8/16/17 10:58	453	453	453	453	453	453	453	
5	2017-08-16T	8/16/17 10:58	450	450	450	450	450	450	450	
6	2017-08-16T	8/16/17 10:58	449	449	449	449	449	449	449	
7	2017-08-16T	8/16/17 10:58	456	456	456	456	456	456	456	
8	2017-08-16T	8/16/17 10:58	453	453	453	453	453	453	453	
9	2017-08-16T	8/16/17 10:58	449	449	449	449	449	449	449	
10	2017-08-16T	8/16/17 10:58	448	448	448	448	448	448	448	
11	2017-08-16T	8/16/17 10:58	452	452	452	452	452	452	452	
12	2017-08-16T	8/16/17 10:58	455	455	455	455	455	455	455	
13	2017-08-16T	8/16/17 10:58	454	454	454	454	454	454	454	
14	2017-08-16T	8/16/17 10:58	456	456	456	456	456	456	456	
15	2017-08-16T	8/16/17 10:58	453	453	453	453	453	453	453	
16	2017-08-16T	8/16/17 10:58	454	454	454	454	454	454	454	
17	2017-08-16T	8/16/17 10:58	452	452	452	452	452	452	452	

Figure 9: A screenshot of CSV file generated by GUI (opened in Excel).

When the user selects an ESP32 ID, the default real-time table begins filling in values. Values populate the table until the specified number of rows (default of 20) is met, in which case the values begin scrolling up. There is pause button that allows the user to stop the data stream. When the user presses start, the data will resume streaming. If the user adjusts the number of rows, the table will wipe the data and begin repopulating until it reaches that value (beginning to scroll again).

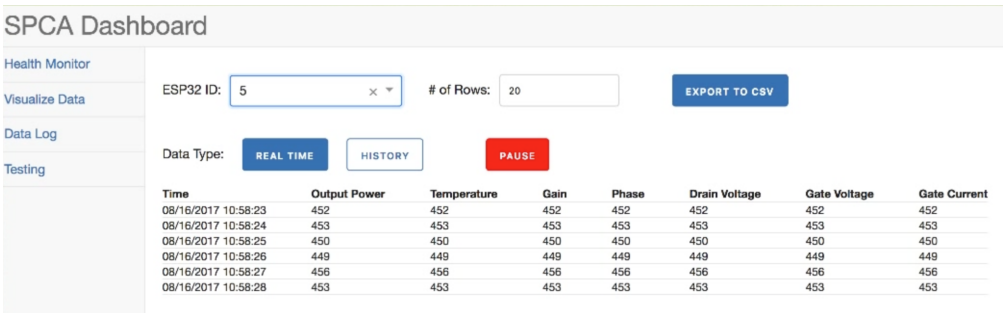

Figure 10: A screenshot of the realtime table of the Data Log Page.

If the user selects the historical data button a new table will display, containing values in the specified time range. There is a time range selection menu (the same as in the Data Visualization Page), that allows the operator to change the dates and times between which he/she want to see data. If the number of values is greater than the number of specified rows, only a portion of the values will be displayed (Page 1). There are buttons at the bottom that allow the user to move forward or backwards pages. If they want more or less values to display at any one time, they must change the number of rows. The smaller the number of rows, the faster the data loads. While data is loading, a loading symbol displays (as with the historical data plots).

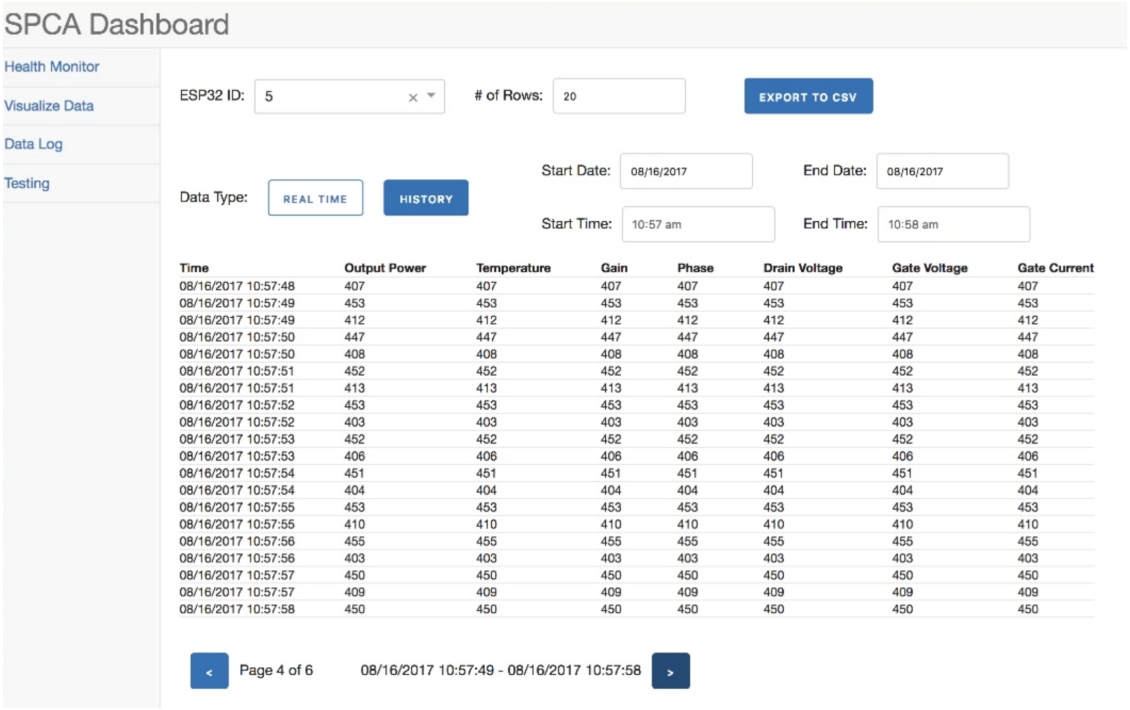

Figure 11: A screenshot of the historical data table of the Data Log Page.

2 Inter-Satellite Omnidirectional Optical Communicator [ISOOC]

2.1 Background

An advanced omnidirectional inter-satellite optical communicator has been proposed to enable high bandwidth (1Gb/s) full-duplex optical communication between small spacecraft that are separated by over 200 km distance, called the Inter-Satellite Omnidirectional Optical Communicator (ISOOC). This communicator will utilize a dodecahedron geometry that can fit inside a 1u block (CubeSat standard). Avalanche Photo Detector (APD) arrays and gimbal-less MEMS scanning

mirrors will be used for direction of arrival calculation and beam steering. The ultimate goal is to integrate the communicator into a small spacecraft to enable data transfer, data relay and relative navigation control. The hope is to maintain a structured, precise CubeSat formation through ISOOC measurements and communication.

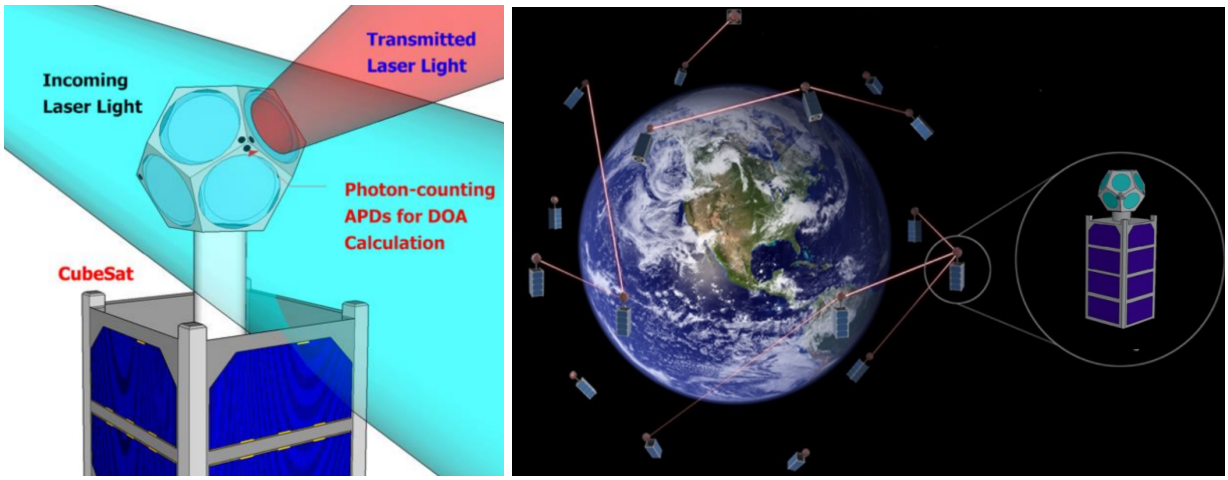

Figure 12: Drawings of the ISOOC functionality and desired application, a distributed network of CubeSats establishing several communication links.

This literature focuses on two tasks to improve the ISOOC design and functionality.

2.2 Transmitter Design

The first task in working on the ISOOC was redesigning the transmitter (TX) CAD file so as to fit the necessary components. The CAD files given were correct in terms of the optics angle and distance measurements, however they had been produced before the component dimensions were known or with components of a different size. The transmitter is composed of an 850nm laser that sends pulses through a collimated lens to a mirror which reflects the light onto a steerable MEMs mirror (controlling the direction in which the light it transmitted). These adjustments were made using SolidWorks and tested by 3D printing the files on a FormLab resin printer. The changes were successful and screw holes were added, allowing the enclosure to be screwed together. A significant change was necessary to fit the MEMs mirror, which was originally assumed to have no back-board.

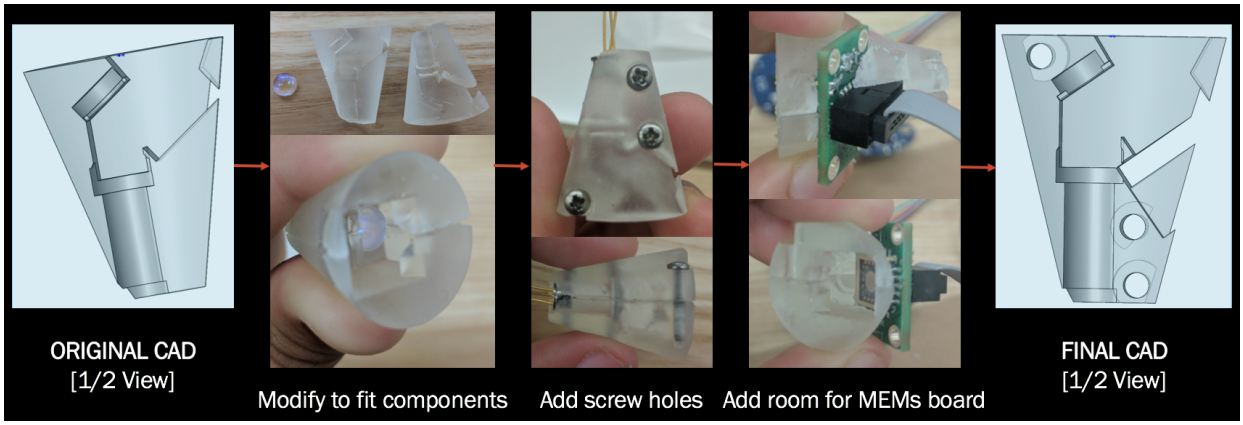

Figure 13: The redesign of the ISOOC TX, changing CAD measurements and testing with 3D prints.

2.3 LIDAR

The second task in working on the ISOOC was the development of a LIDAR (Light Detection and Ranging) system. While the lab wants a LIDAR system for general purposes and other projects, the ability to use LIDAR with the ISOOC is critical to CubeSat flock formation and provides other interesting applications. The goal of this work was to prove that LIDAR depth maps can be achieved using the ISOOC set-up.

2.3.1 Applications

The simplest form of LIDAR consists of sending out a pulse of light and waiting to see how long it takes to return after bouncing off another object. If the clock is precise, this can result in extremely accurate distance measurements. For ISOOC CubeSats this would work, because they are in low Earth orbit, meaning no obstructions. However, to improve the implementation, if the CubeSat clocks were synced extremely accurately (which could be done by sending out synchronization signals from an Earth antenna) the CubeSat that would be deflecting the beam could send back a new pulse, increasing the signal strength to allow a farther range. This ultimately would be used to measure the distance between CubeSats, allowing them to maintain precise flock formation.

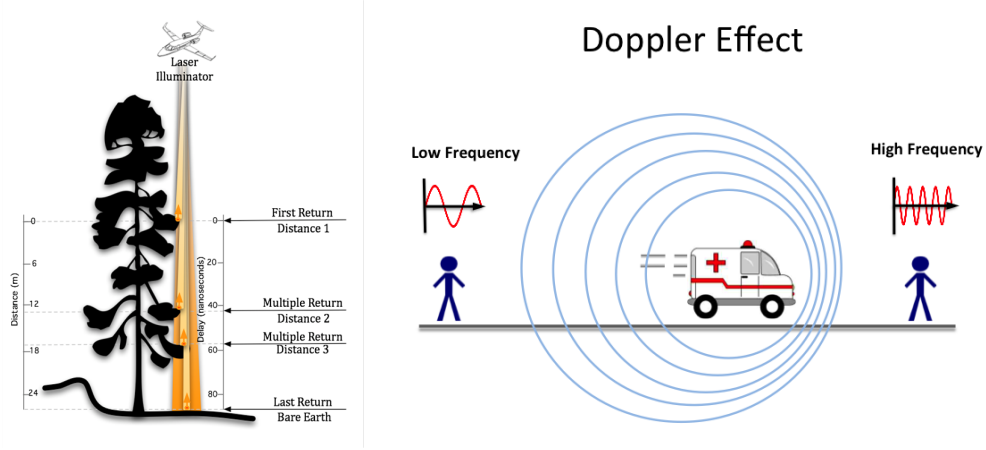

Figure 14: How to measure relative distance and velocity with LIDAR.

Another application of LIDAR is in relative velocity measurements between the CubeSats. Exploiting the Doppler Effect, if the the distance between the CubeSats is increasing [decreasing], the frequency with which the light pulses arrive back to the transmitting CubeSat will decrease [increase]. If the CubeSats need to change formation or are drifting apart, we can tell using this technique.

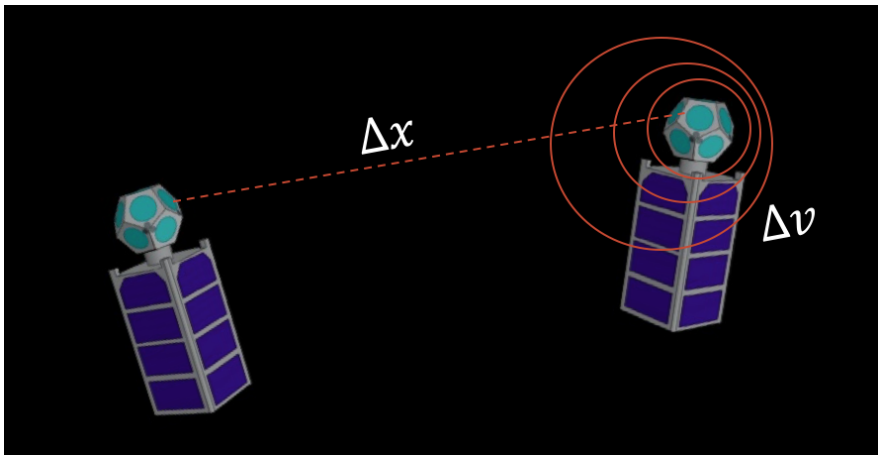

Figure 15: Use of ISOOC LIDAR distance and velocity measurements for CubeSat flock formations.

Of all the applications of the LIDAR system, probably the most interesting is the creation of 3D depth maps. By exploiting the fact that the ISOOC TX is steerable and can scan a field, we can create perform a distance measurement at each point in a plane, allowing us to create 3D reconstruction of the scene. This can be done using all the ISOOC transmitters at once, allowing for the construction of a 360 degree depth map! If the ISOOC were put on, say, a rover this could allow

the device to double as both a communication device (to talk to other small, assistant rovers) and instrumentation tool (allowing the rover to create 3D images of its surrounding).

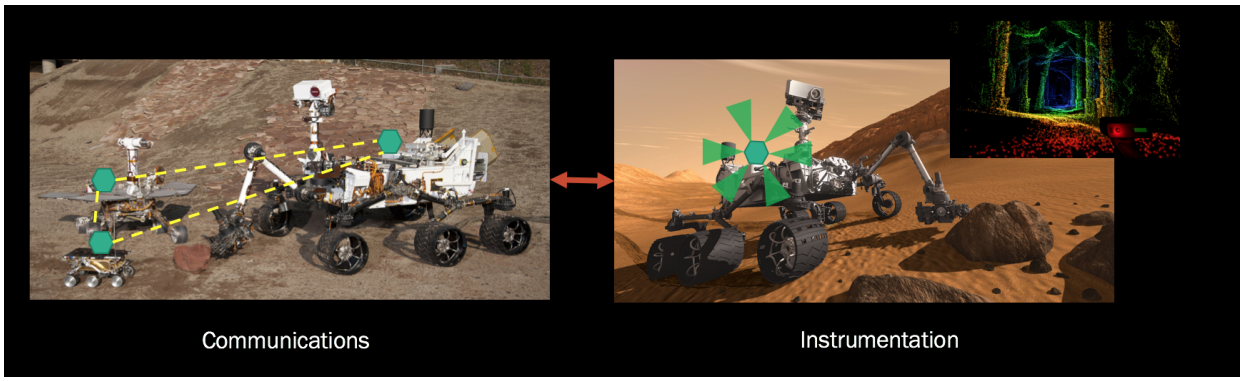

Figure 16: Use of ISOOC for communication and instrumentation (via LIDAR depth maps).

2.3.2 Hardware

The first step in designing the LIDAR system was purchasing a rangefinder to send the IR pulses and measure return time (this was purchased since precision is crucial and the ISOOC FPGAs are still under development). We settled on the Lightware SF11 rangefinder, which can measure up to 150m in distance (although when we tested it seemed to max at 130m). The module was deconstructed and the circuit board, with the attached laser (TX) and APD (RX), as well as the lens components and IR filter were pulled out to build the LIDAR system.

Figure 17: The Lightware SF11 rangefinder disassembled for use in the LIDAR system.

The rangefinder board is connected via UART to a Teensy 3.5, allowing serial communications. Originally we tried using a Tiva C, but the serial communication code compatible with the rangefinder would not work. Also connected to the Teensy is the MEMs driver board, which steers the MEMs mirror to point the light pulses. There were a lot of problems with these boards, which ended up being caused by the power supply (we needed an industrial grade power supply for the amplification

to work as desired). The other intern, MItchell, who had originally worked with the MEMs driver boards was using the Tiva C microcontroller, meaning all the pin breakouts needed to be changed to comply with the Teensy.

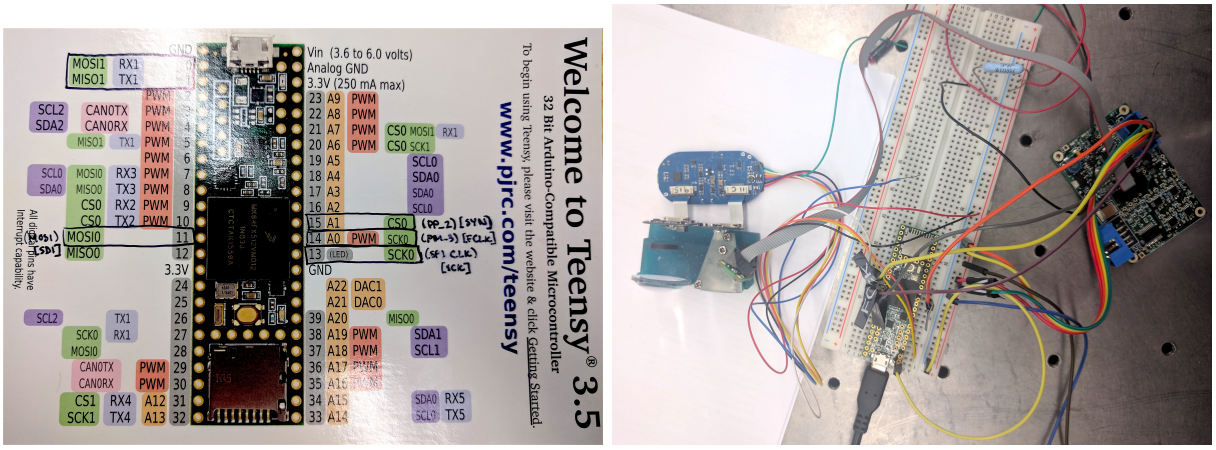

Figure 18: Pin breakout for MEMs driver and rangefinder serial on Teensy and a photo of the actual circuit breadboarding.

2.3.3 Assembly

The rangefinder needed to be deconstructed so that the laser could be put into the ISOOC transmitter case (allowing us to steer the direction of the beam for 3D depth maps). In order to hold all the components a casing/base was designed and CADed. The final base was printed using a Formlab resin 3D printer.

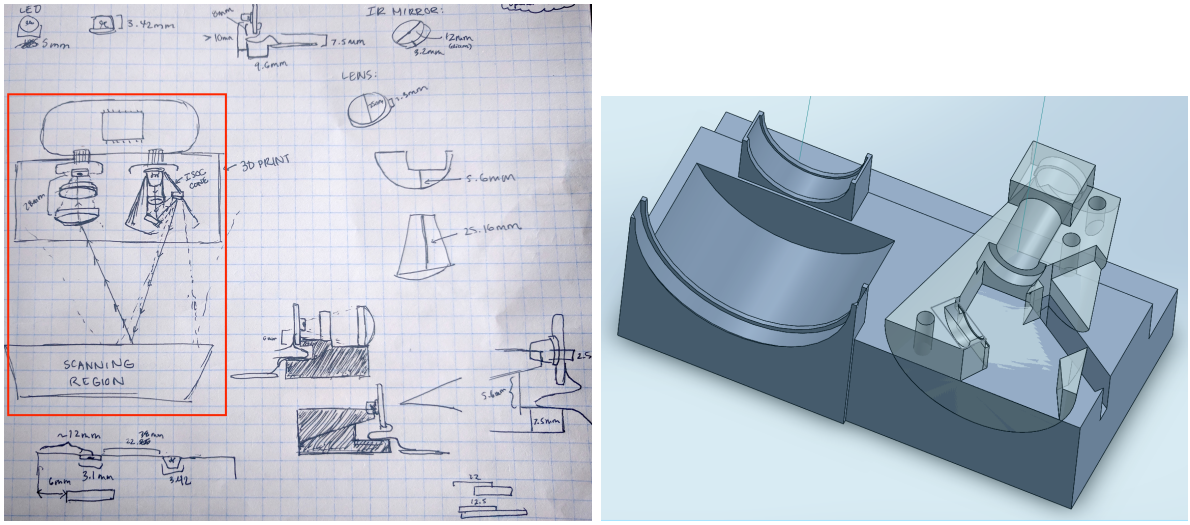

Figure 19: Measurements, design, and CAD of the LIDAR enclosure/base.

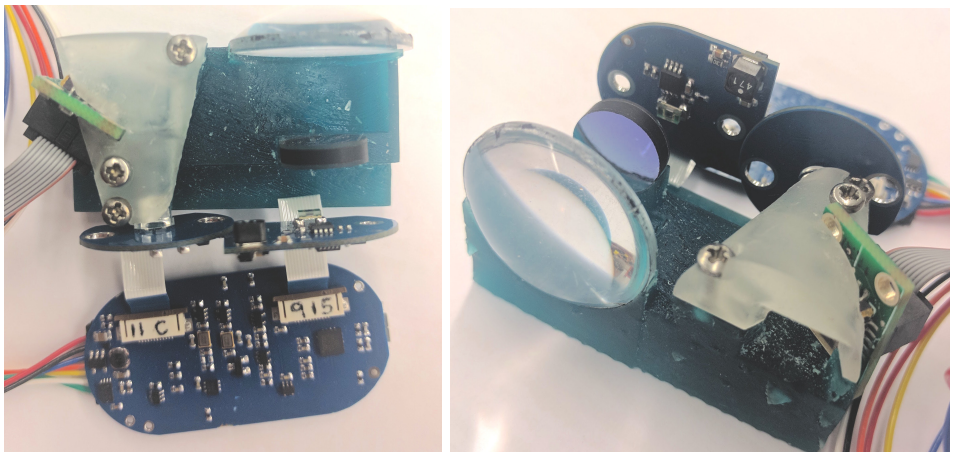

Figure 20: Photos of the final LIDAR assembly, consisting of the rangefinder board, transmitter components (obscured view: collimator lens, mirror, MEMs mirror), receiver lens, and IR filter.

One challenge was assessing whether or not the IR beam was actually making it out of the transmitter. When the rangefinder circuit was put into the casing, the APD was not receiving any pulses, meaning no distances were being measured. After confirming that both the TX laser and RX APD were working, calculations of the beam divergence were made to ensure that too much of the light was not being absorbed before making it to the collimator lens. These calculations matched with the actual measurements, meaning the problem was not divergence. So, we tested for alignment issues (since a mirror bounces the pulses onto the MEMs mirror, which then directs the beam). This was done using a phone camera. It quickly became apparent that the light was making it out of the transmitter, but farther to the side and lower than expected, meaning the the beams were not being deflected back to the receiver. This was corrected by the MEMs mirror scanning and adjustment of the MEMs mirror in the transmitter casing.

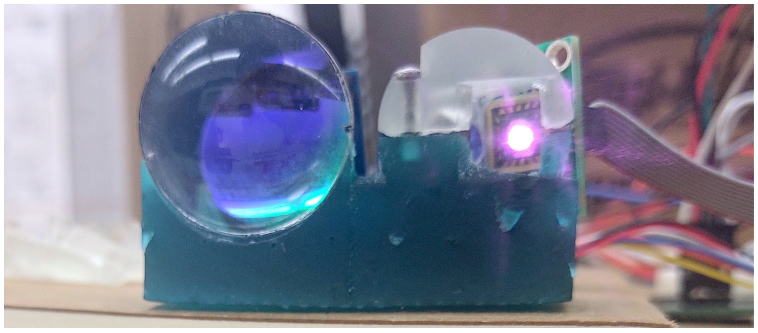

Figure 21: Photo taken with Nexus 5X camera to assess where IR light was coming out of LIDAR TX (can see purple/pink IR light reflecting off of MEMs mirror).

2.3.4 Software

In order to create a 3D depth map, the MEMs mirror must scan the plane and measure the depth at each point. Code for this was written using the Arduino IDE (in C) and uploaded to the Teensy. A for loop was written telling the MEMs mirror to sweep across the plane, row by row. At each point in the row, a command is sent to the rangefinder board, requesting the distance measured when the light is pointing in that direction. As soon as a value is received, the MEMs mirror points to the next point.

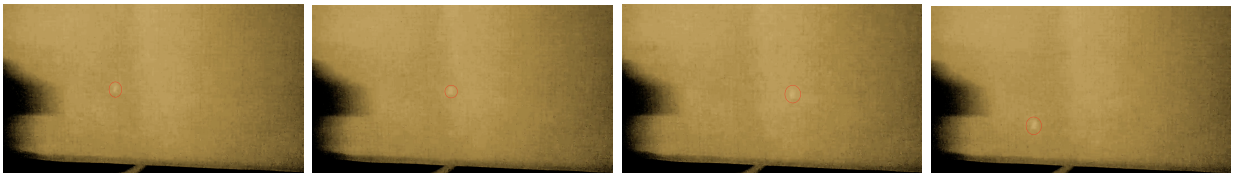

Figure 22: Frames from a video of the actual LIDAR system sweeping across a sheet of paper (the light dot is the IR reflection). The frames have been edited to make the light more noticeable.

In order to visualize the depth map, a python script was written to read the serial output of the Teensy and plot the values with matplotlib. When used with the moving MEMs mirror, we were able to create some rudimentary depth maps!

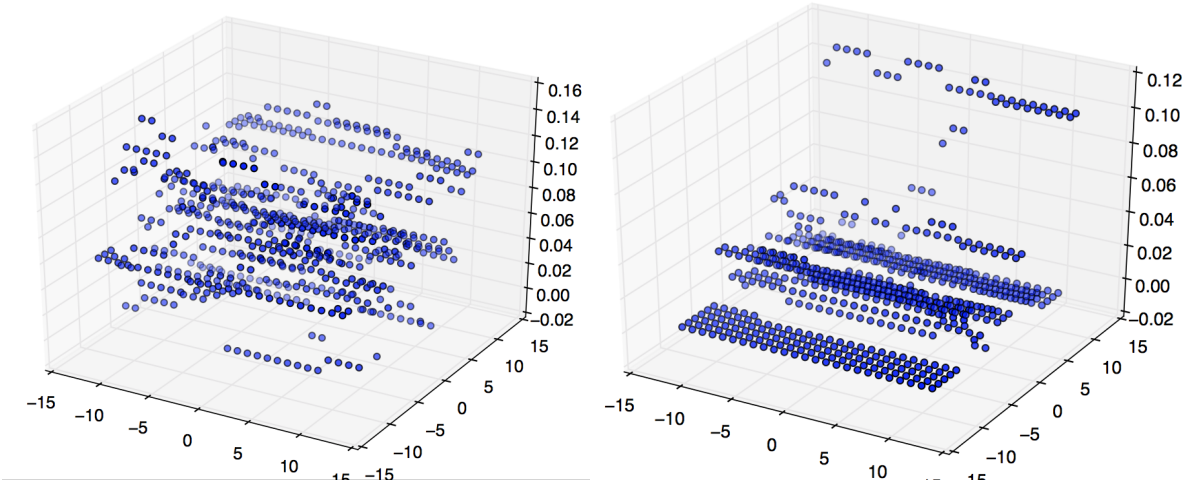

Figure 23: Sample 3D depth maps created by the LIDAR system.

In order to achieve better results, improvements must be made to the hardware. However, the proof of concept holds!

3 Acknowledgements

I primarily would like to thank my supervisor, Jose Velazco, for providing me with the amazing opportunity to work in his lab for ten weeks and taking the time to meet with me on a nearly daily basis. I have learned so much about the field through his guidance and expertise. I would also like to thank other 333K employees, including Andrew Janzen, Mark Taylor, and John Huleis, as well as my office mate, Roland Liou, for their guidance and company. I learned so much about the culture, workings of JPL, and different projects through discussions with them. I would like to acknowledge all the other interns that I worked with in 333K this summer: Amy Weber, Adrian Meza, Alex Huang, Andrew DeNucci, Benjamin Cary, David Campeau, Mitchell Gu, and Nima Mohseni. I would like to especially thank Alex, who acted as a mentor on the SPCA project, and Mitchell, who helped debug the ISOOC LIDAR system. I would like to acknowledge JPL, NASA, and Caltech for organizing the summer intern program and hosting a number of interesting events and lectures to make the summer even more enjoyable. Finally, I would like to thank my parents for helping me with the logistics of getting to Caltech/JPL and supporting me in all my scientific endeavors.