Final Documentation

MAY14-08 3D LIDAR

Members

Nicolas Cabeen – Project Leader

Todd Wegter – Communications & Mechanical

Xiang "Peter" Wang – Software

Eric VanDenover – Electrical

Advisor

Koray Celik

Contents

Nomenclature

Laser - a device that emits light through a process of optical amplification based on the stimulated emission of electromagnetic radiation

LIDAR - term combining ‘light’ and ‘radar’ - device which measures distance by illuminating a target with a laser and analyzing the reflected light

2D LIDAR - a LIDAR which performs a sweeping scan with the laser along one axis to measure the distances along that axis - this gives two dimensional data

3D LIDAR - a LIDAR which performs a sweeping scan with the laser along two axes to measure the distances of a two dimensional area - this gives three dimensional data

Pointcloud - a set of data points in some coordinate system

PWM - Pulse Width Modulation, a signal modulation technique that alters the duration of the pulse to encode information

ROS - Robot Operating System, an open source software framework for robot software development

RS-232 - moderate speed serial bus used to communicate between the PC and servo controller

RS-422 - high speed serial bus necessary for communicating the vast amount of data a LIDAR creates

Servo - a DC motor and transmission which turns to a specific angle given a PWM signal

Stepper Motor - a brushless motor which is controlled through ‘step’ commands to its electromagnetic coils

Problem/Need Statement

3D LIDAR is a very expensive piece of technology which can cost upwards of $100,000 for a commercial unit. This cost is too substantial to use this sensor in many applications. However, a 3D LIDAR is the best device for accurate and precise volumetric scanning. We would like to make this device available at a lower cost and reach out to the robotic community by making it more accessible. Schools, researchers, and the Iowa State Lunabotics Team could all benefit from having this technology available to them. The ISU Lunabotics Team currently needs a sensor that can detect objects in extreme environments in order to compete at NASA's Lunabotics Mining Competition. Also, Vermeer Corporation is interested in using this technology on a new automatic tractor design. Without an advanced sensor, the design may be unsafe or unable to complete its tasks.

Market Survey

The market for a 3D LIDAR is niche at best, which is partially responsible for the continued high prices. 3D LIDARs are really only used for research at this point, so the demand isn't there to lower the price yet. The typical response, then, is to create one's own 3D LIDAR by tilting or rotating a standard 2D LIDAR along the second axis to generate the 3D pointcloud of data. There are several cases where this has successfully been done, and at least one company that commercially produces a 3D LIDAR using a spinning or tilting 2D LIDAR. MIT has used such device on their ROAMS robot, a robot which autonomously navigates their campus making 3D maps (http://www.technologyreview.com/news/416331/making-3d-maps-on-the-move/?a=f). Michael Bosse and Robert Zlot used a spinning 2D LIDAR to collect 3D pointclouds in their research for developing a method to eliminate error from 3D scans taken from a moving scanner (http://ubuntuone.com/0aaEvPEHt89dBFIv9B1quk). Hobbyists on a budget are also applying this technique with cheaper 2D LIDARs to create 3D pointclouds (https://sites.google.com/site/mechatronicsguy/turquoisebot/laser-nodder). 3DLS currently sells several different models of spinning or tilting 3D LIDARs based off the LMS 200 series. They also sell kits for converting an existing LMS 200 series LIDAR into a 3D LIDAR (http://www.3d-scanner.net/index.html). We can see from all the work others have done in this area that this is a reasonable and feasible solution to the problem of expensive 3D LIDARs.

Research has also shown that there are tools available for making the collection of the 3D pointcloud easier. ROS, the Robot Operating System (http://wiki.ros.org/), provides native 3D pointcloud collection. It even provides a framework specifically for collecting this pointcloud from a tilting 2D LIDAR. This OS is becoming the industry standard for these kinds of projects, and has a great support community, including tutorials for building 3D pointclouds with a tilting 2D LIDAR.

Our background research has shown that creating a 3D LIDAR from a tilting 2D LIDAR is a feasible and reasonable solution to the cost problem of a native 3D LIDAR. Since several different groups of varying skill levels have created this kind of device, we determined that should be feasible for us to do the same. ROS's native support for this operation was also very helpful.

Concept Sketch

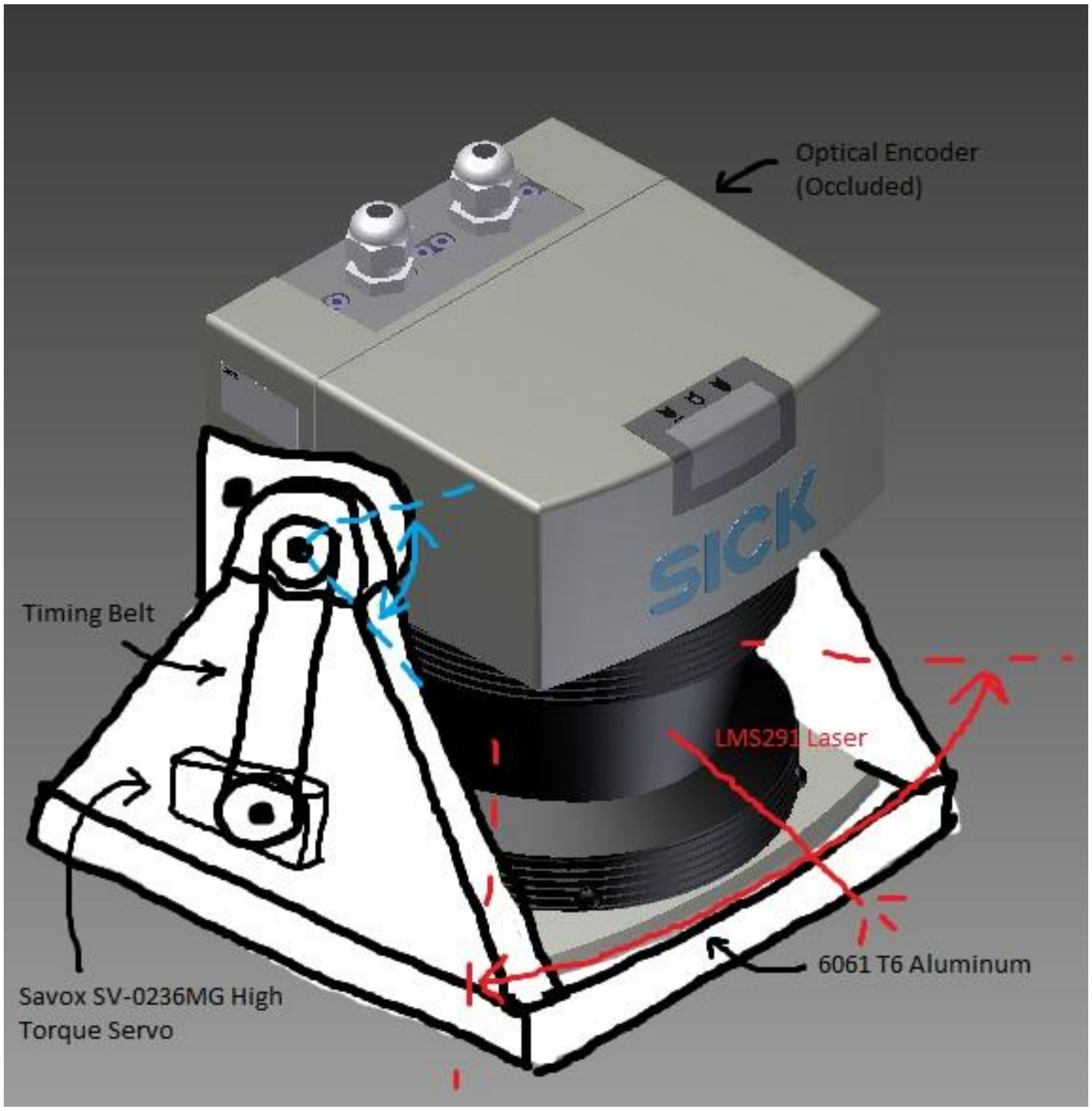


Figure 1 - System Concept Sketch

System Design

System Block Diagram

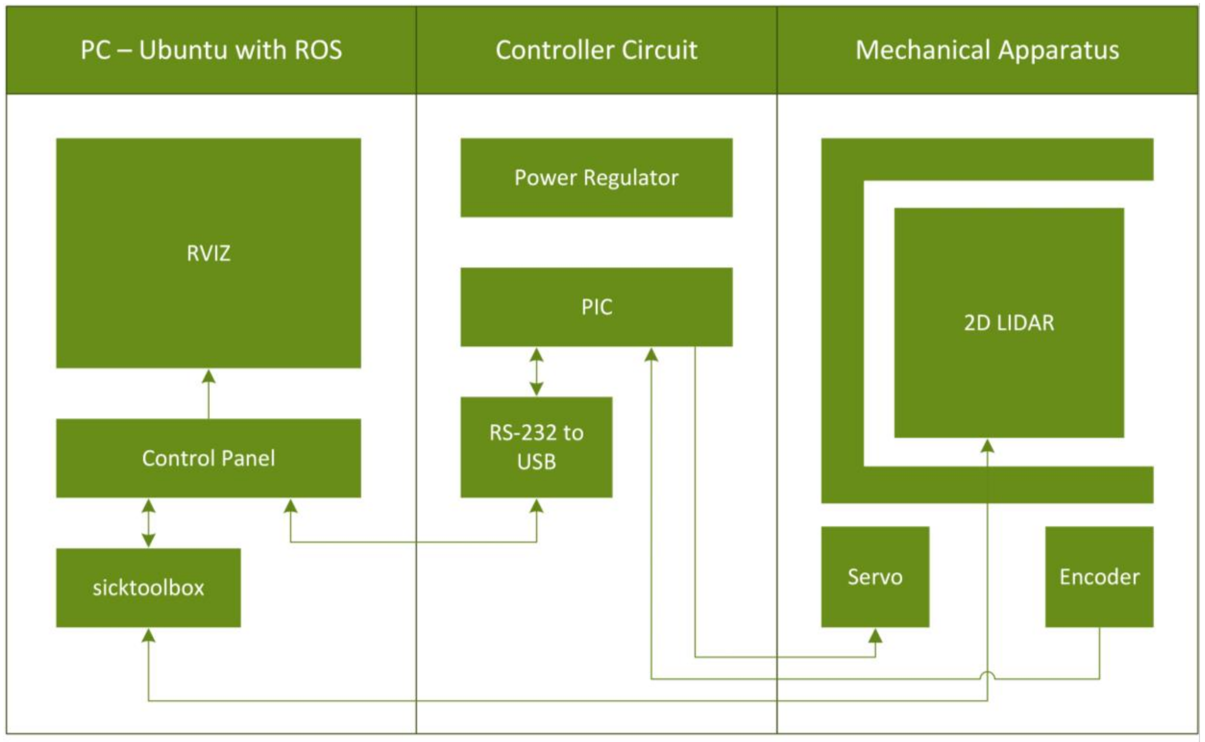


Figure 2 - System Block Diagram

Deliverables

The expected deliverables for this project includes a GUI providing basic control of the servo and LIDAR as well as displaying the scan data in human readable format (ie - pictorially). This should be developed with open-source software and operate on a readily available Linux distribution.

The scanning apparatus will be able to securely hold a 2D LIDAR scanner and precisely position it. The servo should connect to a servo controller or a generic microcontroller, which will be connected to the PC. The LIDAR will communicate with either the microcontroller or the PC directly.

The testing and demonstration of this project will at minimum be the scanning apparatus resting on a table with the PC by its side displaying the output. This will be running in real-time, allowing visitors to interact with the scan and view the output. If time permits, we will mount the apparatus on an RC mobile base and communicate with a PC wirelessly.

Functional Requirements

1. Scanning

Each scan must take less than 1 second to complete

Each scan must start no more than 0.5 seconds after the previous scan completes

2. Mechanical

Servo controller must provide positional feedback (ie - be closed-loop)

Servo must withstand 600 oz/in of torque

Scanning apparatus must be able to scan at least 90 degrees

3. Software

Deliver scan results in a human-readable visualization

Scan results should be visible to the user no more than 0.5 seconds after a scan completes

Use RS-422 to communicate with LIDAR to achieve necessary baud rate

Non-Functional Requirements

1. Hardware

Hardware should connect to PC via USB

Apparatus should use the SICK LIDAR that was provided

2. Software

User should be able to operate the system with minimal configuration and set-up

3. Cost

Any needed software should be open source or widely available

Cost of prototype must be much less than a 3D LIDAR

Functional Decomposition

System Description

To clarify our design, we have divided the system into several subsystems as illustrated in the above block diagram. The PC subsystem integrates with a 2D LIDAR and collects scan measurements for visualization. The mechanical subsystem is a custom-designed mechanical platform and bracket for the LIDAR that can be easily rotated by an industrial servo attached with a timing belt. The LIDAR will be mounted on the platform so that when the servo rotates, the LIDAR nods up and down. The servo will be controlled by the third and final subsystem, a control board that communicates with the PC and also provides closed-loop positional feedback with an external encoder mounted on the LIDAR.

The PC software allows users to control the entire system. When the software collects measurements from the LIDAR and the control board, it correlates the two datasets and creates a 3D graphical representation for the user.

The end goal will be to integrate this system into another larger system, such as large machinery or the Iowa State Lunabotics robot. These larger systems operate can be dangerous without sensors to detect obstacles, targets, and pedestrians.

Identification of Subcomponents

PC Software

- ROS
- RVIZ package
- sicktoolbox package
- PCL package
- Servo controller serial communication
- QT GUI, embedded control panel
- Launch script
- Data processing threads

Control Board (Software)

- UART communication
- Servo positioning
- Encoder feedback
- Closed-loop control

Control Board (Hardware)

- PIC-18F2580 microcontroller
- FT232RL UART via USB chip
- Power regulation circuit

Mechanical Apparatus

- Mechanical components
    - Base plate
    - LIDAR supports
    - Servo mount
    - Encoder mount
    - LIDAR bracket
    - Timing belt
    - Bearings
    - Rubber grommet feet
- SICK LMS291 LIDAR
- Savox SV-0236MG servo
- Avago HEDM-5500 optical encoder
- RS-422 USB serial cable adapter
- Power
    - 24V 10A DC power supply (LIDAR)
    - 0-30V 13A 400W adjustable DC power supply (Servo and Servo Controller)

Identification of Third-Party Components

PC Software

- ROS
- RVIZ package
- sicktoolbox package

Control Board (Hardware)

- PIC-18F2580 microcontroller
- FT232RL UART via USB chip

Mechanical Apparatus

- Mechanical components
    - Timing belt
    - Bearings
    - Rubber grommet feet
- SICK LMS291 LIDAR
- Savox SV-0236MG servo
- Avago HEDM-5500 optical encoder
- RS-422 USB serial cable adapter
- Power
    - 24V 10A DC power supply (LIDAR)
    - 0-30V 13A 400W adjustable DC power supply (Servo and Servo Controller)

System Analysis

PC Software

ROS

Robot Operating System (ROS) is a set of open-source software libraries that integrate as a framework to build robot applications. In our application, we are using the current distribution of ROS called Hydro Medusa. Because ROS libraries are built using C++, this allows us to write our own C++ code and makefiles. By using catkin build tools, which is also part of ROS, we can invoke our makefiles to build our software.

RVIZ Package:

RVIZ display function receives data from the LIDAR thread and output the information on the computer screen graphically. This graphical information must be human readable and relay the distance and direction measurements. A control panel GUI is embedded with RVIZ to launch the scan. The GUI will have buttons and fields available to receive input from the user.

The graphical output displayed in RVIZ is plotted as a point cloud. This gives the user a 3D perspective of what is been scanned. This virtual 3D representation will be processed and drawn by the CPU, not a GPU. There are two strong reasons for this design choice: 1) this expands the number of systems that can run our software and makes our solution more portable; 2) the additional computational power of a GPU is not required for the relatively simple 3D rendering necessary for our solution.

sicktoolbox Package:
Sicktoolbox is a library that allows the software to interface with our SICK LMS291 device. This library includes function calls to the LIDAR such as configuring the device to different modes and resolutions, initializing the device, reading and retrieving scan datas from the device, and un-initializing the device.

PCL Package:
PCL or Point Cloud Library is a software library that provides an interface for point cloud processing. Data obtained from the LIDAR can be calculated and mapped to x-y-z coordinates. By creating a cloud object, these points can be added to the cloud to form a 2D/3D image.

Servo Controller Serial Communication:
This function is responsible for sending commands to and receiving data from the hardware. The servo controller uses UART serial communication and the LIDAR communicates with hexadecimal codes transmitted over RS-422 cable.

Data Processing:
The data processing component translates the raw data from the LIDAR into a properly formatted two dimensional matrix. The coordinates of the matrix are directly related to the position which was scanned and the values in each position contain the corresponding distance measurement.

Launch Script:
A bash launch script is used to configure ROS and other external libraries to create the necessary system environment for our solution. Upon exit of our application, the script also follows a graceful termination sequence of all external libraries.

Control Board (Software)

UART Communication

The control board is designed to communicate with a PC using UART over USB. This sub-component handles transmission of character strings and reception of individual character bytes. Reception is controlled with an interrupt flag so no polling is necessary. Transmission does not use interrupts since the system should only transmit when the goal angle has been reached by the servo and is not based on a timer.

Our control board was unable to transmit ASCII characters with a value of 17 (0x11) and 19 (0x13) which correspond to Device Control 1 and 3. To work around this issue, we translate these two characters to 117 and 119, respectively. This is safely outside of our normal transmission range of -30 to 70.

Servo Positioning

Our servo is capable of 180 degrees of motion controlled with a PWM signal. Upon reception of a new servo position, we translate the angle into a 0-180 degree range and calculate the pulse widths required for the desired position. The true servo positioning code is interrupt based while the pulse widths are calculated in the main loop. Throughout our development and testing process, we determined the calculations are too intensive to be handled within the interrupt handler. The calculations take approximately 500 clock cycles which causes enough change in the pulse width to alter the servo position by nearly 10 degrees.

Encoder Feedback

Since our servo cannot provide the positional accuracy of a stepper motor, we needed external positioning feedback. Using transition interrupts, we were able to track the LIDAR's physical position accurately and precisely. This feedback is used to determine when the LIDAR has reached the desired angle and triggers a transmission to the PC.

Closed-Loop Control

We were able to implement closed-loop positioning control by incorporating the encoder feedback with the servo positioning. By comparing the goal angle with the actual angle measured by the encoder, the servo position can be altered to move the LIDAR to the correct angle. Because of the speed of our desired scan, closed-loop control is not active in our final solution and is only operational when the control board is placed into "human readable debug mode" as described in Appendix B.

Control Board (Hardware)

PIC-18F2580 microcontroller

The microcontroller is the basis of the control board and runs the software described in the previous section. The external crystal allows the microcontroller to operate at 40MHz, the maximum clock speed of the hardware. By having a fast clock speed and efficient coding, the control board can compute a large amount of calculations without lagging in speed. If the board is not fast enough, the PWM signal will be skewed and send the servo to the wrong angle. Port B of the microcontroller is connected to the transistor that sends power to the control pin of the servo and also to each encoder port. The controller has plenty of I/O pins to handle the amount of devices that the system requires. This chip can also be reprogrammed for future designs via the Mini-Fit Jr 6-pin port on the PCB.

FT232RL UART via USB chip

The microcontroller communicates with a UART signal and the system requires a USB connection. In order to achieve a USB connection, we use the FT232RL. When powered and wired correctly, the chip provides a UART connection between the microcontroller and PC over a USB 2.0 cable.

Power regulation circuit

Since the PCB components require 5V and the board is powered with 7.2V, we designed a power regulation circuit to step the voltage down. This circuit uses a LM2675-5 to provide a steady 5V to power the components and protects them from any power surges or 'brown outs.'

Mechanical Apparatus

Mechanical Components

A vast majority of the parts used to build the mechanical apparatus were designed and manufactured in-house. These parts, including the base plate, LIDAR supports, LIDAR bracket, servo mount, and encoder mount were all custom machined out of 6061 T6 aluminum. Machining these parts ourselves allowed us to design and build exactly what we needed while keeping costs low. This also meant that we spent a great deal of time machining these parts. Other standard components, such as the timing belt and pulleys, bearings, screws, and rubber feet, were purchased off the shelf for ease and cost-effectiveness.

SICK LMS291 LIDAR

The SICK LMS291 LIDAR is a proven 2D LIDAR, capable of 180 degree scans with 0.5 degree resolution. It is also capable of operating reliably outdoors. The most important feature of the LMS291, however, is its range: 80 meters. This range is key to the requirements set forth for our project. The street price of an LMS291 is roughly $5,000, allowing this project to be very affordable in comparison to true 3D LIDAR systems available with an 80 meter range (approx. $100,000).

Savox SV-0236MG Servo

This servo is an affordable high torque metal gear giant scale servo. We selected it for our project based primarily on its price, strength, and durability. To nod the LMS291 LIDAR reliably, we needed a servo which had adequate muscle and wouldn't strip from the torque of reversing the nod direction. This servo met these requirements. Furthermore, our advisor, Koray, had personal experience with this servo, and could vouch for its robustness.

Avago HEDM-5500 Optical Encoder

We selected this encoder for its high resolution and self-contained package. The HEDM-5500 is a quadrature encoder providing 1000 steps per revolution with 4 unique ticks per step. This provides sufficient resolution to measure the rotational position of the LIDAR to within 0.09 degrees. Since this particular encoder is a self-contained package, we didn't need to supply our own encoder wheel, and integration into the system was simple.

RS-422 USB Serial Cable Adapter

The SICK LMS291 is capable at transmitting data over either RS-242 or RS-422. While RS-232 is more common, RS-422 allows for much higher throughput. This increased bandwidth was necessary for our project in order to achieve our desired scan speed. We used a USB to RS-422 adapter to utilize the faster speeds of RS-422 while still being able to use the system with nearly any computer.

Power

We used two independent DC power supplies in our project: one to power the LIDAR and the other to power the servo controller and the servo. A 24V 10A DC power supply was used for the LIDAR, ensuring that it had sufficient power available for operation. The servo and servo controller were powered with a 0-30V 13A 400W adjustable DC power supply adjusted to 7.2V. This voltage is optimal for operation of the servo, and is regulated with a switching regulator down to 5V for the servo controller. Using two separate supplies was much simpler than designing a regulator to take 24V down to 7.2V while supplying sufficient current for the servo.

Detailed Design

Input/Output Specification

System Level

User to ROS

The user is able to interact with the apparatus via a ROS user interface. The user is able to issue input and configuration commands to control the apparatus and the LIDAR. ROS has a window that displays a 3D graphical representation of the scan measurements. System status is returned to the user through the UI as well. Using ROS, the user has the option to export settings and an image file of the 3D graphical representation.

ROS to LIDAR

ROS communicates with the LIDAR via a USB-to-DB9 converter cable using the RS-422 protocol. ROS sends configuration and request commands to the LIDAR. The LIDAR returns status signals and scan measurements. The initialization begins at a baud rate of 9,600 bps and transitions to the operational rate of 500 kbps.

ROS to Servo Controller

ROS also communicates with the servo controller board via UART over USB. Desired servo positions are sent as a signed 8-bit value (signed char). The FT232RL chip on the controller board converts USB to UART and simulates a serial port to the PC. After the signal is processed on the board, a PWM signal is sent to the servo to physically move the LIDAR to the specified angle. When the servo is in position, the controller board returns the returns current position as a signed 8-bit value (signed char). All data transmissions are at a baud rate of 57,600 bps.

Optical Encoder to Servo Controller

An optical encoder mounted on the apparatus provides feedback of the true angle achieved by the servo's motion. The encoder communicates using quadrature encoding with the servo controller. The servo controller then can use this information to reposition the servo as necessary.

PC Debug Interface to Servo Controller

We have implemented a debug interface for the servo controller to increase the visibility of internal servo controller operations. The debug interface is accessed by connecting to the serial device using a third-party serial terminal such as "PuTTY" for Windows or "screen" for UNIX systems. The connection should be established with a baud rate of 57,600 bps. Once connected, the user can hold 'h' for approximately 3 seconds to transition the servo controller into debug mode. Once in debug mode, the servo position and encoder value will be sent from the controller to the terminal in a clear, human-readable string format. Full specifications are included in Appendix B.

Subcomponent I/O

ROS

RVIZ Display

The RVIZ display window shows a real-time 3D graphical representation of the incoming scan data from the LIDAR coupled with the servo's position.

Servo Start Angle Control

This box accepts integer input between -25 and 65. If valid, it becomes the bottom limit of the vertical scan range. This value must be less than or equal to the stop angle.

Servo Stop Angle Control

This box accepts integer input between -25 and 65. If valid, it becomes the top limit of the vertical scan range. This value must be greater than or equal to the start angle.

Start Scan Control

The Start Scan control button initializes the LIDAR and signals the servo to begin scanning and collecting data.

Stop Control

The Stop control button stops sending signals to the servo controller to cease nodding the apparatus. It also signals ROS to halt collecting data from the LIDAR and gracefully exits.

Servo

The Savox SV-0236MG accepts PWM signals from the servo controller as input. It does not electrically return any signals but instead physically outputs mechanical energy to move the timing belt and LIDAR

Optical Encoder
The Avago HEDM-5500 optical encoder optically reads markings on its internal encoder wheel attached to the same shaft as the LIDAR. The encoder sends ticks back to the servo controller using quadrature encoding.

User Interface Specification
When the system is under operation, the user primarily interacts with the graphical control panel interface for configuration of the physical hardware and the 3D visualization of the scan measurements. The user is able to configure the LIDAR and servo at the beginning of operation. Once the user has launched the scan, the 3D visualization is generated in another window and can be freely rotated to achieve any desired viewing angle. Snapshots of the scan can also be taken.

UI samples are included in Appendix A.

Software Specification
PC Software
Software Functionality
The PC software functions as the unifying piece between the LIDAR, servo controller, and the user. It sends commands to and receives scan measurements from the LIDAR and controls the servo controller while simultaneously processing the measurements to generate a 3D visualization for the user.

Languages
- Bash
- C++
- CMakeList

Libraries
- ROS
- RVIZ package
- sicktoolbox package
- PCL package
- QT GUI

Compiler
We compiled using catkin_make, a ROS compilation package.

Architecture
Please refer to the architecture diagrams in Appendix A.

Servo Controller Software

Software Functionality

The servo controller software's primary function is generating PWM signals to move the servo while monitoring the position by reading measurements from the optical encoder. It receives positioning commands via USB from a PC and replies with real-time positioning measurements.

Languages

- C

Libraries

- PIC18F Library

Compiler

We compiled with the Microchip XC8 compiler with the MPLAB X IDE on Linux.

Hardware Specification

Our design work is focused on the servo controller and the mechanical apparatus which supports the LIDAR and nods it up and down. Other major components are from third parties. This specification lays out the requirements for our project's hardware.

Desktop Computer

- Must have at least two USB 2.0 ports
- Should be running Ubuntu 12.04LTS
- Should have at least 2GB of RAM
- Should have at least 32GB of hard disk space (for operating system, files, and software)
- Should have at least a 1GHz processor

USB 2.0 Cable with Converter to RS-422 (DB9)

- Should handle conversion from USB 2.0 to RS-422 and vice versa
- Should simulate a serial port over USB
- Should not be longer than three feet
- Should not use external power

RS-422 Cable

- Must be long enough to easily connect to the LIDAR
- Must be able to securely fasten to the LIDAR for operation

LIDAR

- Must be a SICK LMS-291 LIDAR

USB 2.0 Cable with Mini-B Connector

- Should not be longer than five meters

Servo Controller

- FT232RL
    - Provides the computer with a UART com port over USB Mini-B port
- PIC18F2580 Microcontroller
- LM2675 Voltage Regulator
- Must be supplied with 7.2V
- See Electrical CAD for more details

Encoder

- Must be quadrature encoded
- Should be an optical encoder
- Should provide resolution finer than 0.5 degrees per tick

Servo

- Savox SV-0236MG Servo
    - 416.6 – 555.5 oz-in of torque
    - 0.17-0.21 sec/60 degrees
    - Accepts 6V-7.4V

Mechanical Apparatus

- 6061 T6 Aluminum
- See Mechanical CAD for details

Power Supply

- 24V 10A DC Power Supply (LIDAR)
- 0-30V 13A 400W Adjustable DC Power Supply (Servo and Servo Controller)

Test Plan

Our 3D LIDAR solution is comprised of three different subsystems: PC software, a mechanical apparatus, and a servo controller PCB. Each of these subsystems was independently tested and verified before integration into the whole system. These tests were designed to show that each subsystem has met its functional requirements. Finally, we tested the whole system to ensure that it functioned as intended.

Software Testing

We tested the software subsystem independently in a controlled environment to ensure specific functionality. Much of our testing correlated with the development cycles of the PCB and mechanical development. We tested our system with simulated LIDAR and servo measurements, and developed several smaller test applications for subcomponent testing. Finally, we tested the portability of our solution by installing it on multiple systems. This allowed us to verify that our software will work in many different environments.

Mechanical Stress Testing

The components of our design which are under the greatest stress were first tested in SolidWorks' built-in Finite Element Analysis software. This verified that they would be able to sustain the loads that they are subjected to. For our tests, we applied a 100N force (about twice the weight of the LIDAR since it swings back and forth and the rotation point isn't perfectly at the center of mass) to the parts. This found that our critical parts had a factor of safety over 40. Knowing that most engineering projects find a factor of safety of 10 to be sufficient, we deemed that our parts were more than adequate. Please review the two FEA reports included in Appendix A for further details.

System testing has shown that our mechanical design is more than sufficient for supporting and moving the LMS 291 LIDAR. Some thread locker was needed to ensure that the timing belt system wouldn't come loose, but other than that, our testing has revealed no problems with the mechanical design.

Servo Controller Board Testing

This subsystem was first tested on a breadboard before ordering the PCB. With the breadboard, we tested both basic software functionality as well as the electrical characteristics of the board. This breadboard served as a proof of concept and showed that our design worked. So we ordered the PCB. This was then assembled and tested for proper functionality with the established servo control and UART communication code.

As the code base for the servo controller was further developed, testing shifted to software. One of the biggest issues we had at one time was inconsistency in the production of the PWM signal controlling the servo. Testing and team code review found that a large amount of the PWM calculation was being performed in the timer interrupt for the PWM signal generation. Moving this calculation to the main code loop solved the problem since this large calculation was no longer throwing off the timing of the PWM. Any other problems we encountered with this code were solved in a similar fashion.

System Environment Testing

The final product was tested in several environments to ensure that it can work under a reasonable range of conditions. Primarily, we tested the system both indoors and outdoors to verify that different background lighting did not affect the system. A collection of sample scans has been included in Appendix A.

Simulation, Modeling, and Prototype

Software Prototype

Thanks to the modular model provided by ROS, we were able to design and test our software at a relatively granular level. We were also able to swap designs with relative ease since we adhered to the MVC software design model. For example, we were able to create multiple revisions of our graphical RVIZ window that looked very different with confidence that they would integrate with the rest of our design as long as they used the same interface.

PCB Prototype

Before ordering our PCB, we first built and tested a breadboard equivalent of the circuit for electrical and software testing. Once we confirmed the functionality of the circuit, we proceeded with the PCB order and continued our development on that platform.

Mechanical Apparatus Prototype

We developed a 3D model of our mechanical apparatus well before we began machining components. We then performed rigorous machinability testing, made several weight reduction changes, and simulated the operation of our solution. With our completed design drafted, we were able to minimize our construction material costs and maximize our throughput in the machining lab. To assist in this step, we acquired precise models of third-party components such as the LIDAR and servo directly from the manufacturers.

Standards

RS-422

RS-422 is a technical standard that specifies electrical characteristic of a digital signaling circuit. RS-422's full title is ANSI/TIA/EIA-422-B. We used RS-422 to interface from the computer to the LIDAR. It is bidirectional and can operate in full duplex so we can both send commands and receive data from the LIDAR at the same time. Since we do not want to interrupt the scanning data, the full duplex capabilities will be crucial so that we can change necessary settings without dropping incoming scan measurements. We also discussed using RS-232 for the interface to the LIDAR, but went against it since we need the higher baud rate offered by RS-422 (a maximum of 10Mb/s).

RS-232

RS-232 is a communication protocol standardized in the 1960s. It is a slow protocol (maximum baud rate 112.5 Kbps) but is also very easy to implement. We used RS-232 on our servo controller board to communicate with the PC through a RS-232-to-USB converter.

Universal Serial Bus 2.0

Universal Serial Bus (USB) is a serial communication standard, but includes some guaranteed functions atypical of other serial communication protocols. Most of the details of the standard are outside the needs of our project, but most notably USB is much faster than other protocol standards discussed. USB claims data transfer rates of up to 5Gbps. For our uses however, we use USB to simplify our RS-422 and RS-232 connections since our computer (and most modern computers) does not have DB-9 ports for RS-422 or RS-232 connections.

Laser Classification

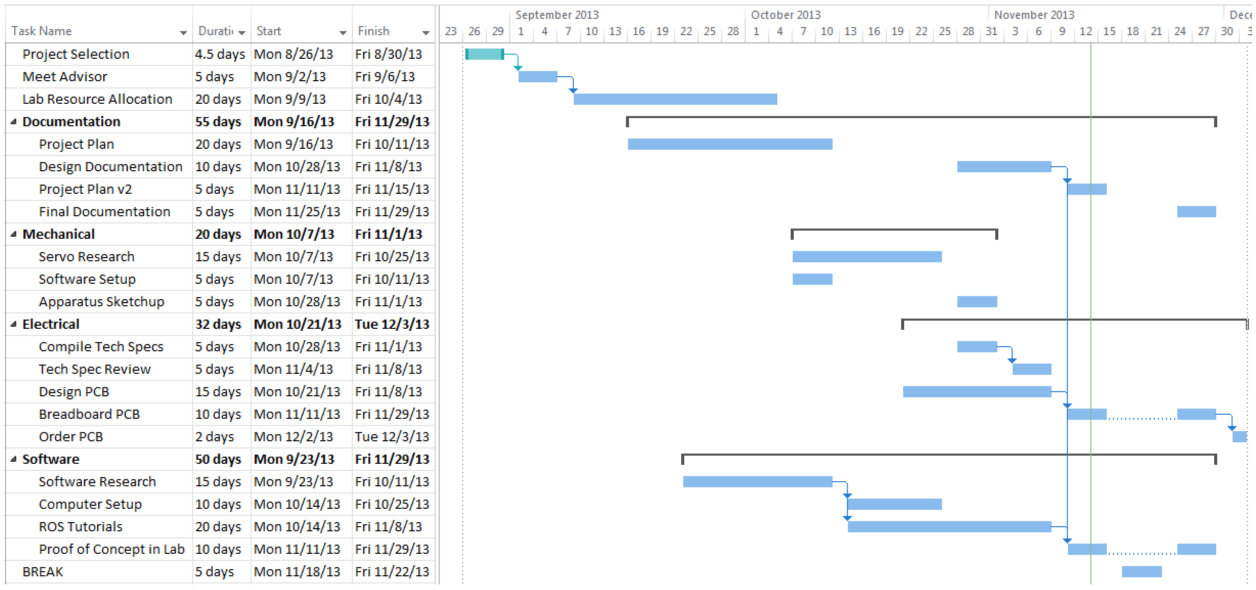
Since the LIDAR is intended to be used to scan areas potentially involving humans, it is crucial that the laser does not pose a danger to anyone crossed by the beam. Because of this, our project needed to adhere to the laser classifications outlined by the IEC 60825-1 standard. Specifically, the laser must be a Class 1 laser to be guaranteed to be eye-safe. The laser in the Sick LMS-291 LIDAR we have acquired meets the safety requirements for Class 1 lasers. Since we left the LIDAR unaltered from production state (that is, we did not open the casing), we can be confident that our project meets Class 1 standards.

SAE

Our project also involved physical machining and was subject to many measurement standards. Through research we found two primary standards for making threads: ISO/IEC 80000 and SAE Threading Standards. The SAE standards are more common than the metric standard for machining tools on campus, so we have decided our best option would be to use SAE threads in our design. In fact, Dr. Celik mentioned that he personally owns SAE threading tools that we could use for our project. Both standards would be satisfactory to meet our needs, but availability of machining tools leads to the best option of using SAE.

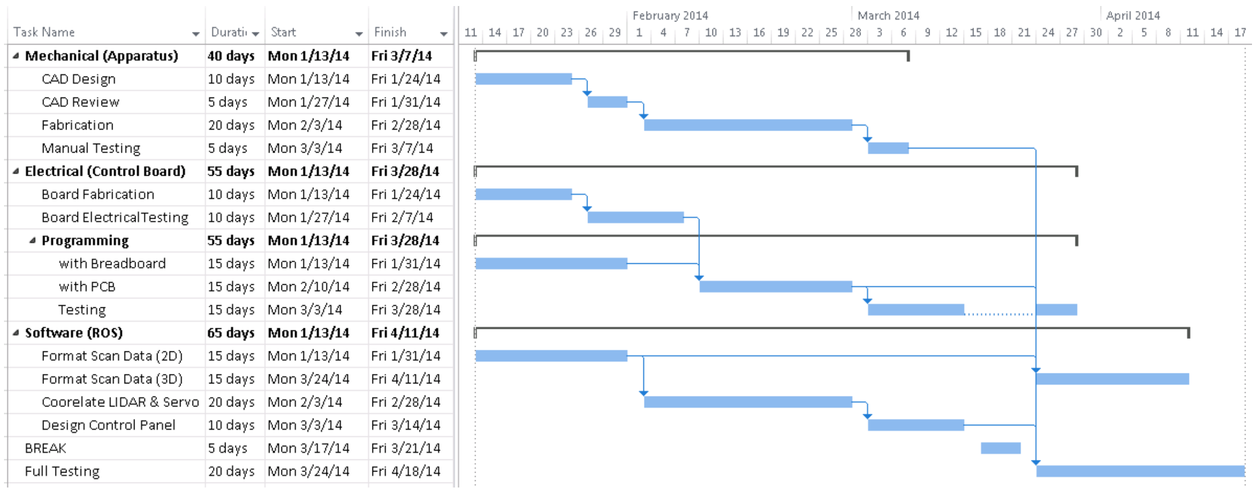
Project Plan

Schedule



Task Name	Duration	Start	Finish
Project Selection	4.5 days	Mon 8/26/13	Fri 8/30/13
Meet Advisor	5 days	Mon 9/2/13	Fri 9/6/13
Lab Resource Allocation	20 days	Mon 9/9/13	Fri 10/4/13
Documentation	**55 days**	**Mon 9/16/13**	**Fri 11/29/13**
Project Plan	20 days	Mon 9/16/13	Fri 10/11/13
Design Documentation	10 days	Mon 10/28/13	Fri 11/8/13
Project Plan v2	5 days	Mon 11/11/13	Fri 11/15/13
Final Documentation	5 days	Mon 11/25/13	Fri 11/29/13
Mechanical	**20 days**	**Mon 10/7/13**	**Fri 11/1/13**
Servo Research	15 days	Mon 10/7/13	Fri 10/25/13
Software Setup	5 days	Mon 10/7/13	Fri 10/11/13
Apparatus Sketchup	5 days	Mon 10/28/13	Fri 11/1/13
Electrical	**32 days**	**Mon 10/21/13**	**Tue 12/3/13**
Compile Tech Specs	5 days	Mon 10/28/13	Fri 11/1/13
Tech Spec Review	5 days	Mon 11/4/13	Fri 11/8/13
Design PCB	15 days	Mon 10/21/13	Fri 11/8/13
Breadboard PCB	10 days	Mon 11/11/13	Fri 11/29/13
Order PCB	2 days	Mon 12/2/13	Tue 12/3/13
Software	**50 days**	**Mon 9/23/13**	**Fri 11/29/13**
Software Research	15 days	Mon 9/23/13	Fri 10/11/13
Computer Setup	10 days	Mon 10/14/13	Fri 10/25/13
ROS Tutorials	20 days	Mon 10/14/13	Fri 11/8/13
Proof of Concept in Lab	10 days	Mon 11/11/13	Fri 11/29/13
BREAK	5 days	Mon 11/18/13	Fri 11/22/13

Figure 3 - Semester 1 Gantt Chart



Task Name	Duratio	Start	Finish
Mechanical (Apparatus)	**40 days**	**Mon 1/13/14**	**Fri 3/7/14**
CAD Design	10 days	Mon 1/13/14	Fri 1/24/14
CAD Review	5 days	Mon 1/27/14	Fri 1/31/14
Fabrication	20 days	Mon 2/3/14	Fri 2/28/14
Manual Testing	5 days	Mon 3/3/14	Fri 3/7/14
Electrical (Control Board)	**55 days**	**Mon 1/13/14**	**Fri 3/28/14**
Board Fabrication	10 days	Mon 1/13/14	Fri 1/24/14
Board ElectricalTesting	10 days	Mon 1/27/14	Fri 2/7/14
Programming	**55 days**	**Mon 1/13/14**	**Fri 3/28/14**
with Breadboard	15 days	Mon 1/13/14	Fri 1/31/14
with PCB	15 days	Mon 2/10/14	Fri 2/28/14
Testing	15 days	Mon 3/3/14	Fri 3/28/14
Software (ROS)	**65 days**	**Mon 1/13/14**	**Fri 4/11/14**
Format Scan Data (2D)	15 days	Mon 1/13/14	Fri 1/31/14
Format Scan Data (3D)	15 days	Mon 3/24/14	Fri 4/11/14
Coorelate LIDAR & Servo	20 days	Mon 2/3/14	Fri 2/28/14
Design Control Panel	10 days	Mon 3/3/14	Fri 3/14/14
BREAK	5 days	Mon 3/17/14	Fri 3/21/14
Full Testing	20 days	Mon 3/24/14	Fri 4/18/14

Figure 4 - Semester 2 Gantt Chart

Work Breakdown

We first elected Nicolas Cabeen as the group leader and Todd Wegter as the group communicator. The group leader delegates tasks and helps make crucial decisions when the group is in disagreement. The communicator arranges meeting times with everyone, coordinates communications between the group and the advisors, and submits the weekly report. From here the project can easily be broken down into three categories: mechanical, electrical, and computer. Todd Wegter is taking lead on the mechanical portion of the project. This includes designing the apparatus that moves the LIDAR, fabricating any components required, and connecting any required equipment to the base. Eric VanDenover is in charge of the electrical component of the design which consists of any circuit design or fabrication required and connecting the LIDAR, servo motor, and controller to the computer. The computer portion has two parts. Nicolas Cabeen is working on the communication between the computer, servo, and LIDAR. Xiang "Peter" Wang has taken charge of interpreting the data from the LIDAR and servo to make a graphical representation of it.

Resource Requirements

We will require the following resources for our development:

- Development lab space
- Machining lab time
- 2D LIDAR
- Industrial-grade servo
- Servo controller/microcontroller
- RS-422 to USB converter box
- Linux PC
- Multi-output Power Supply (0-24V)
- Aluminum for apparatus construction
- Bearings for apparatus construction

Early Risk Identification/Mitigation

Risks	Mitigations
We are not able to find a servo in our budget	Use surplus parts or find sponsorship to find a servo
The LIDAR and servo cannot collect data quickly enough to be usable	We can compress data and drop resolution as needed to meet speed requirements
The inherent risk of shock injury from working with electrical equipment	Follow lab policies and procedures Wear a ground strap
The provided equipment is very expensive and we may not be able to replace it if broken	Read and know the technical specifications of the equipment Follow proven procedures Don't work by yourself or when you're tired or impaired

Acknowledgements

Dr. Koray Celik

The team would especially like to thank Koray for his countless hours of advice, direction, and assistance for this project. His ability to balance between providing clear guidance and giving us the challenge to design, develop, and learn on our own allowed us to accomplish more than we could have hoped.

On a second note, the team would also like to thank Koray for providing us with a dedicated workspace in his research lab. The confidence to leave our project on the desk at night and find it in the same condition the next morning was simply awesome.

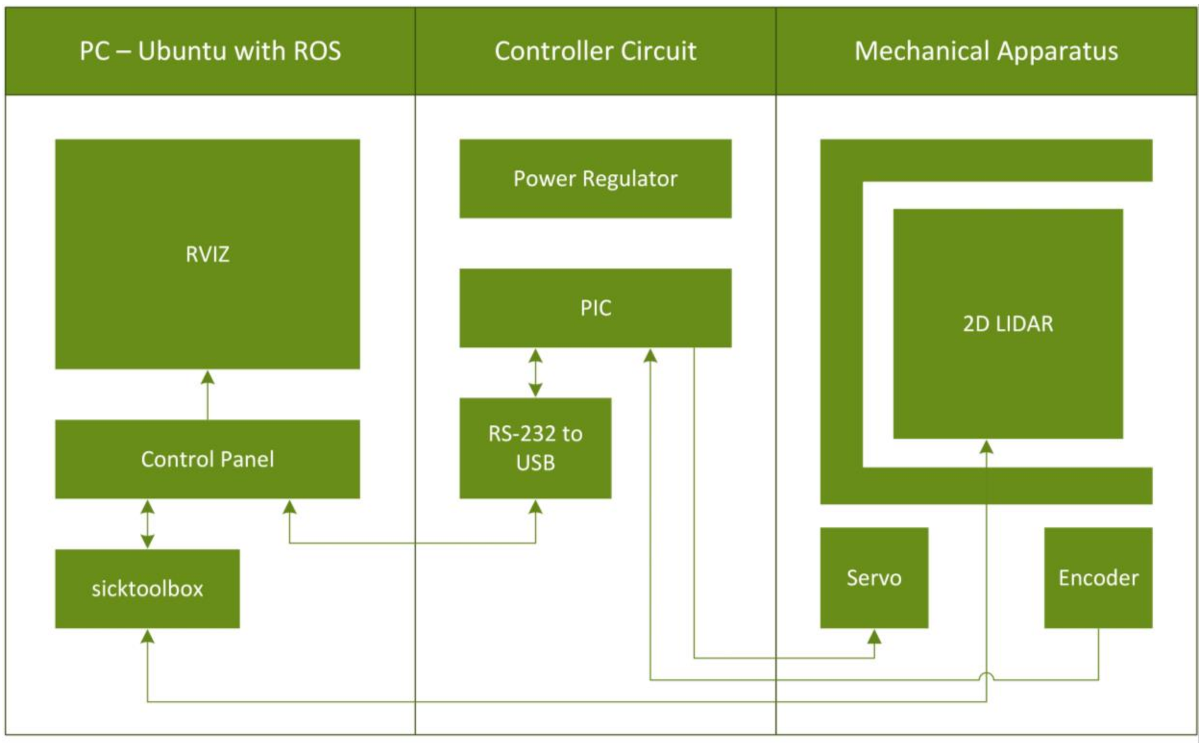
Leland Harker

Special thanks to Lee Harker for his innumerable hours of machining time and invaluable expertise. Without him, our project would never have left our computer screens and joined the physical realm.

Appendix A: Diagrams and Pictures

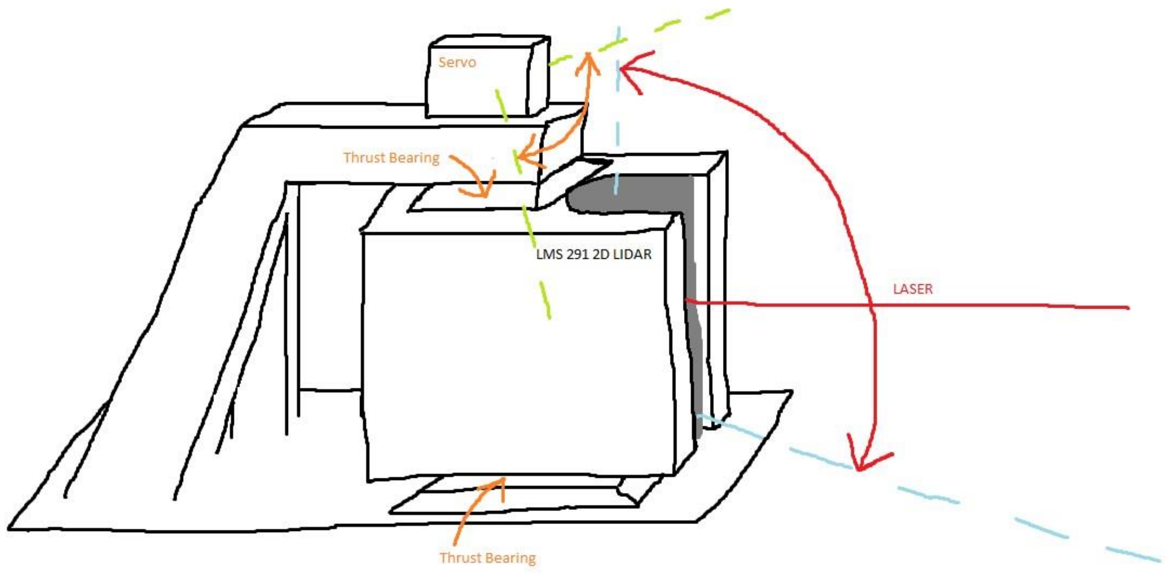
System

Block Diagram

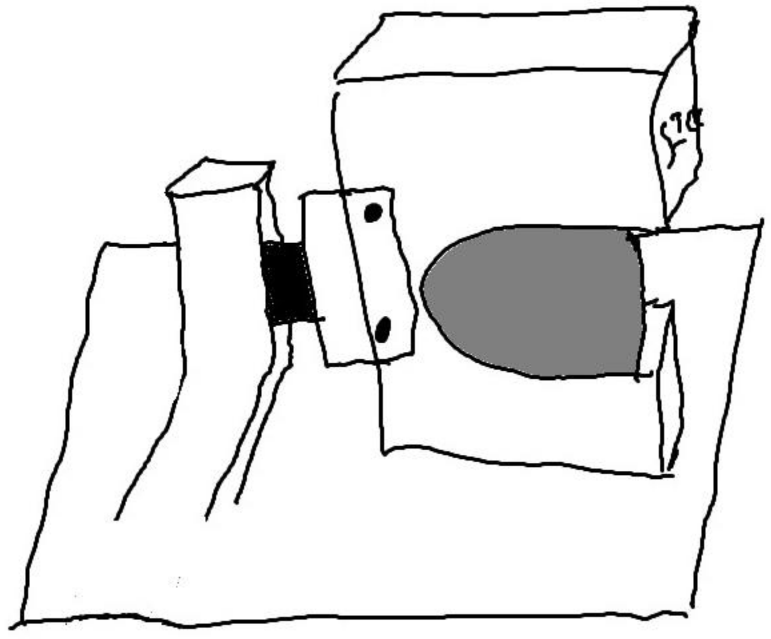


Sketches

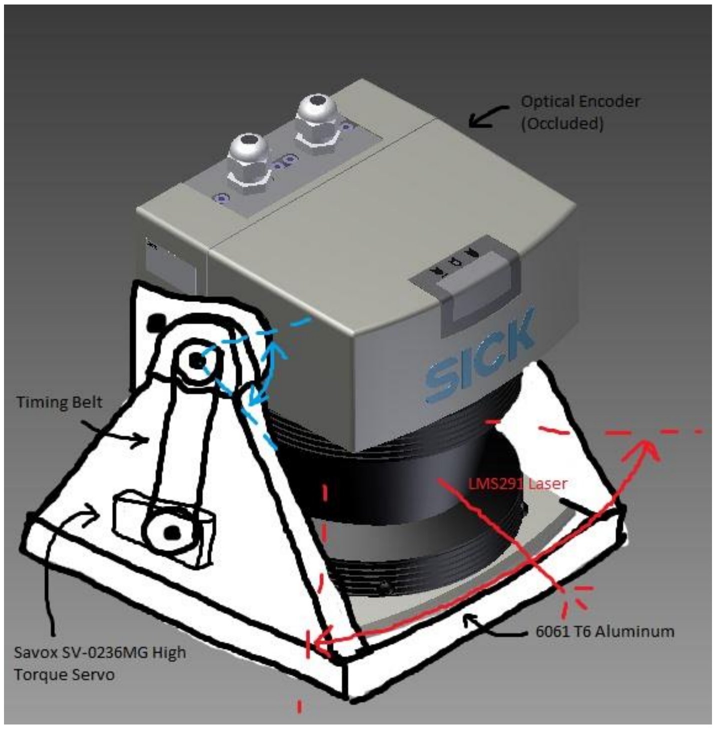
Alternative Design 1 – Horizontal Rotation



Alternative Design 2 – Polar Rotation

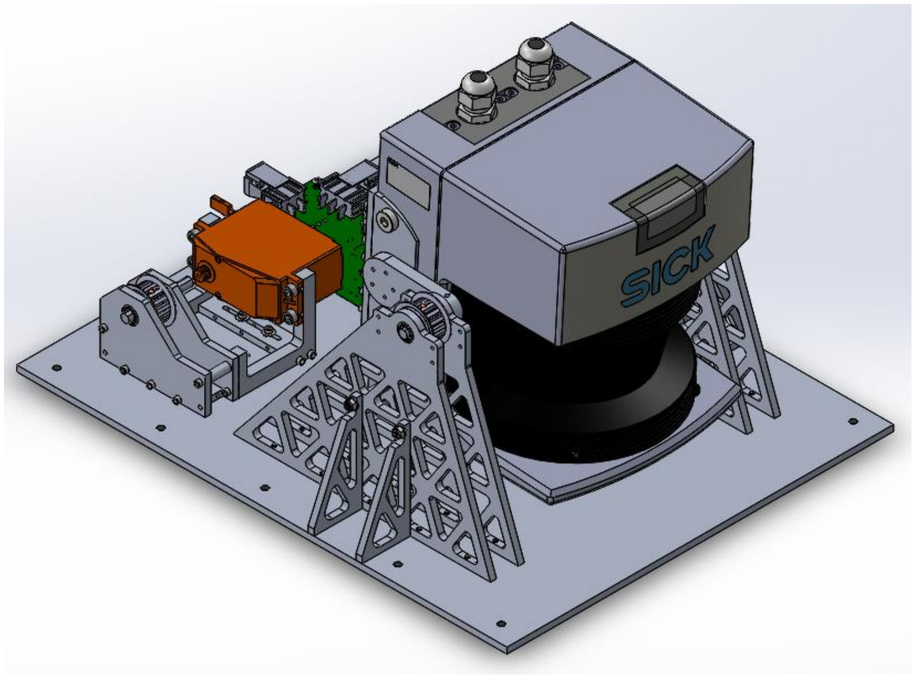


Final Design Sketch – Vertical Rotation



Mechanical Apparatus

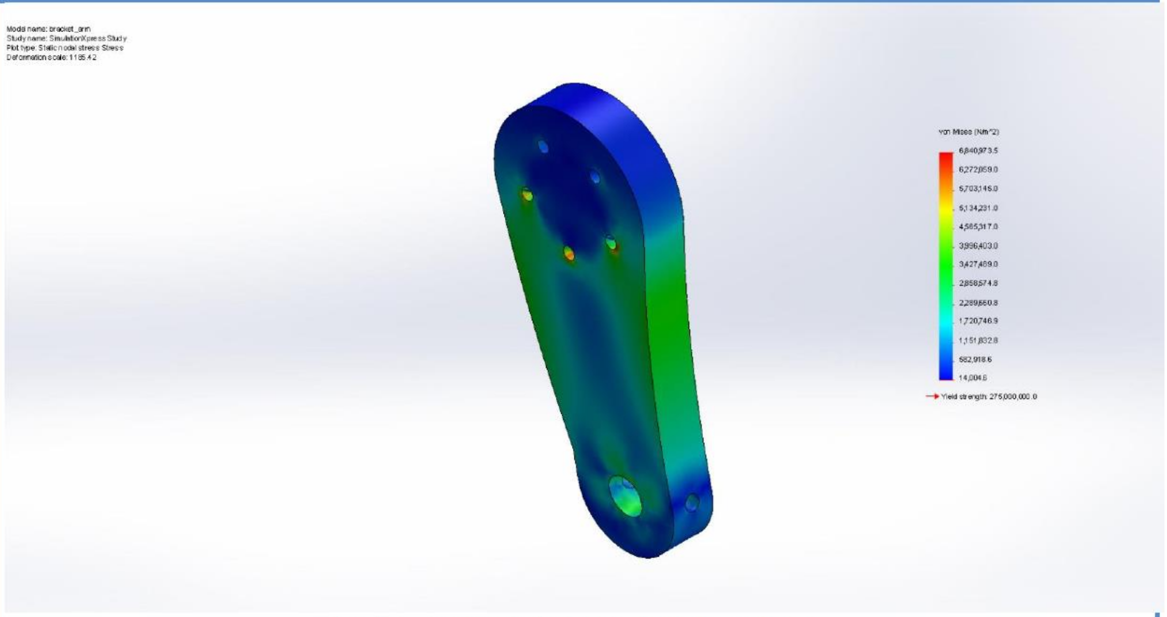
CAD – 3D Isometric View



Stress Tests

LIDAR Bracket Arm

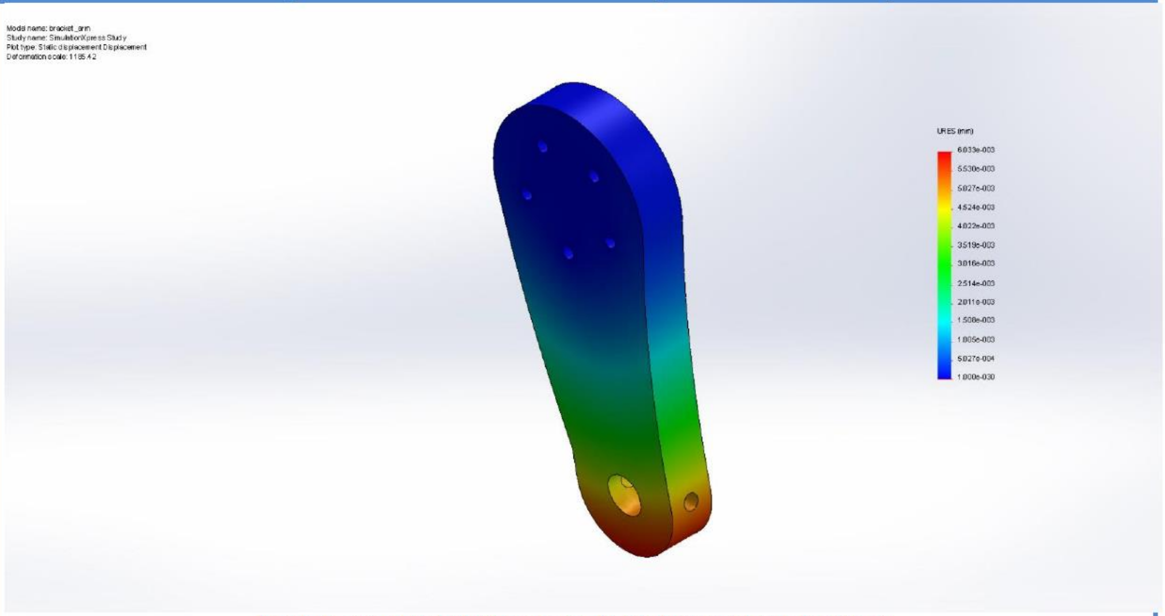
Name	Type	Min	Max
Stress	VON: von Mises Stress	14004.6 N/m^2 Node: 13151	6.84097e+006 N/m^2 Node: 15014



Model name: bracket_arm
Study name: SimulationXpress Study
Plot type: Static nodal stress Stress
Deformation scale: 1185.42

bracket_arm-SimulationXpress Study-Stress-Stress

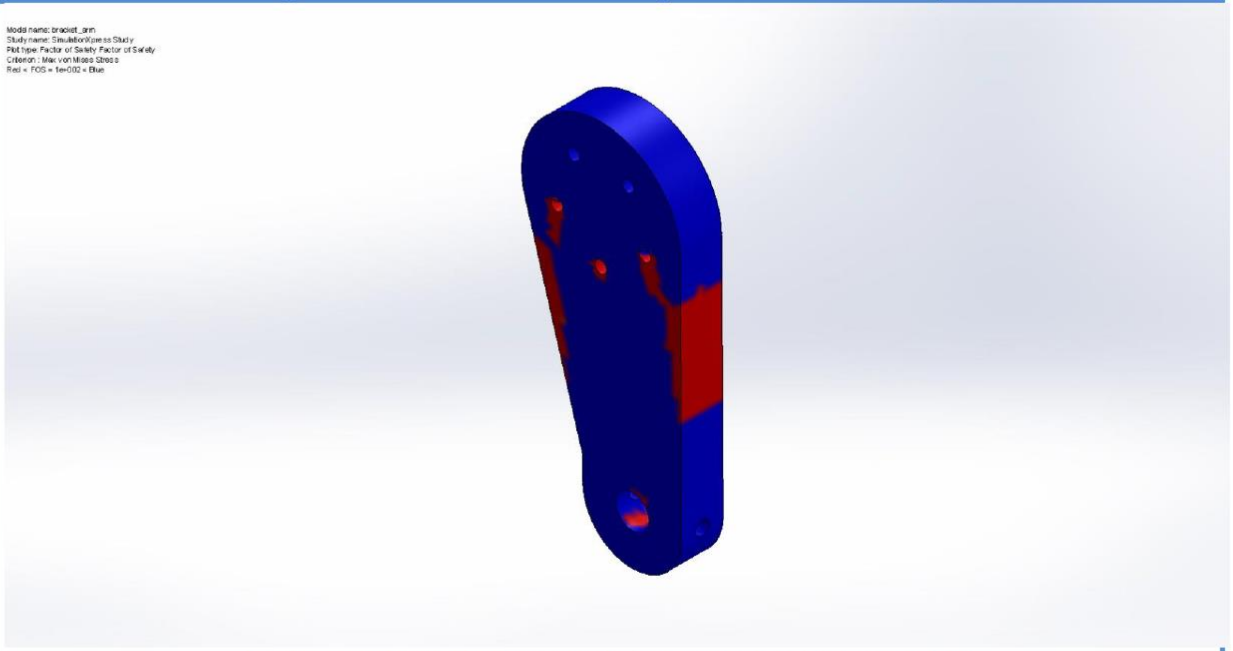
Name	Type	Min	Max
Displacement	URES: Resultant Displacement	0 mm Node: 1	0.00603256 mm Node: 13912



Model name: bracket_arm
Study name: SimulationXpress Study
Plot type: Static displacement Displacement
Deformation scale: 1185.42

bracket_arm-SimulationXpress Study-Displacement-Displacement

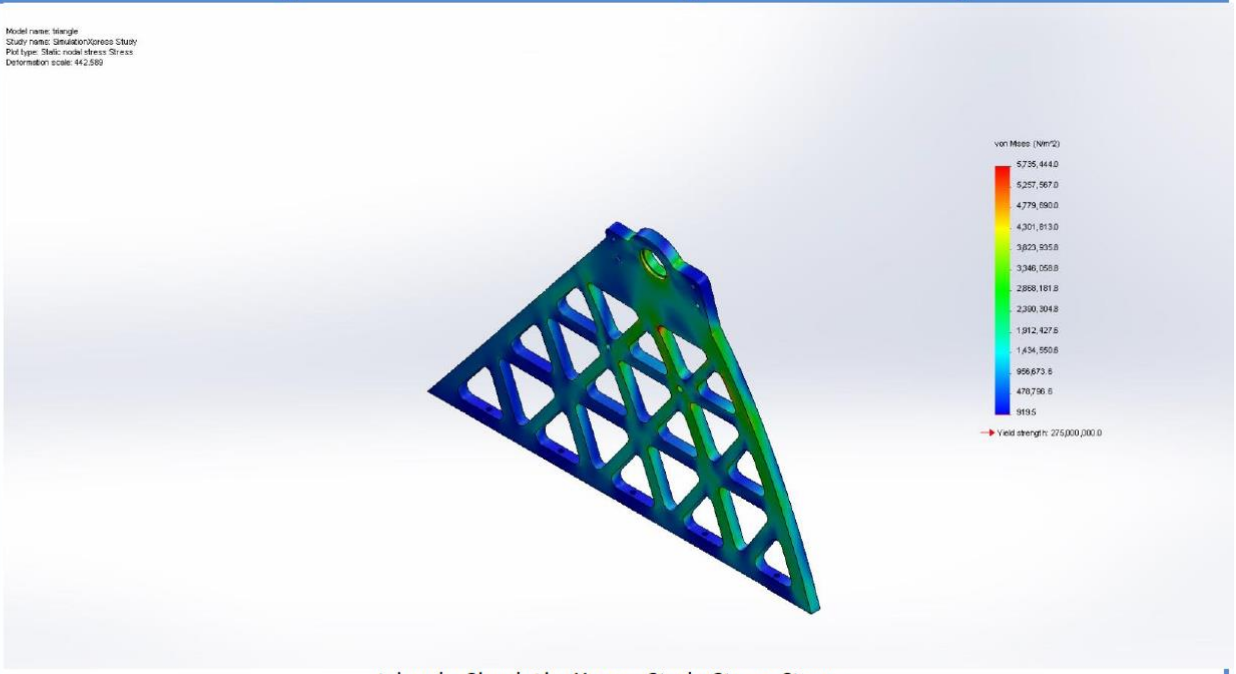
Name	Type	Min	Max
Factor of Safety	Max von Mises Stress	40.199 Node: 15014	19636.5 Node: 13151


Model name: bracket_arm
Study name: SimulationXpress Study
Plot type: Factor of Safety Factor of Safety
Criterion : Max von Mises Stress
Red < FOS = 1e+002 < Blue

bracket_arm-SimulationXpress Study-Factor of Safety-Factor of Safety

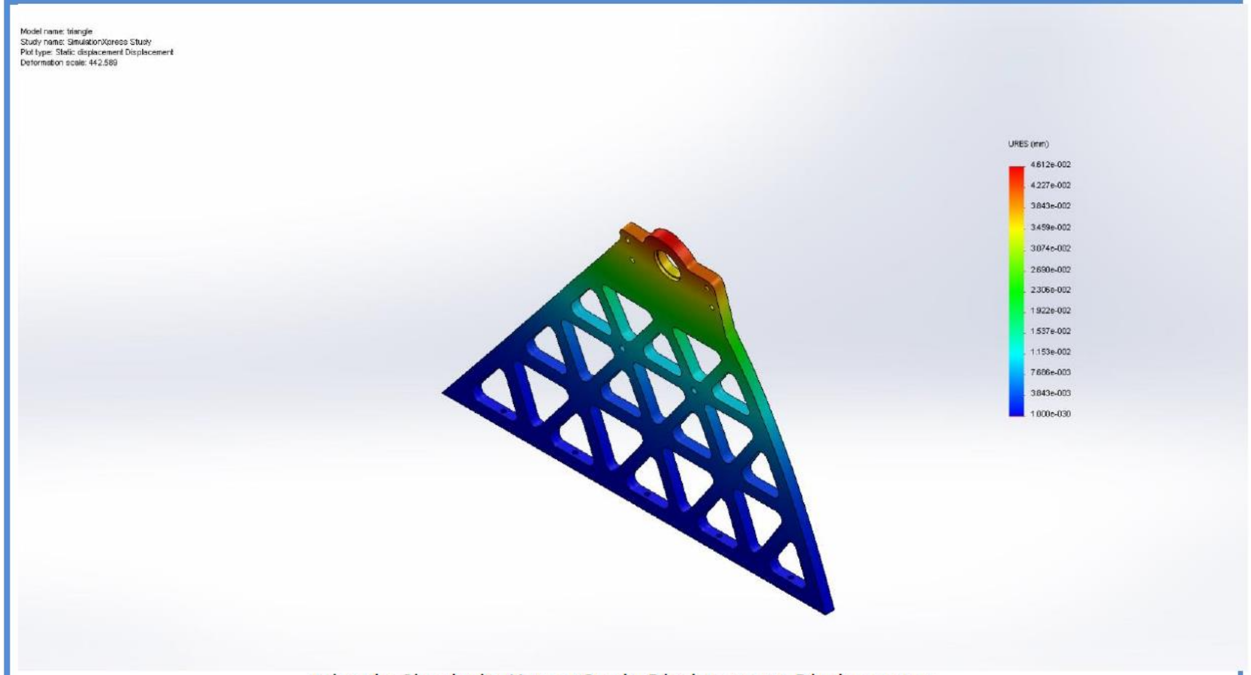
Main Triangle Piece

Name	Type	Min	Max
Stress	VON: von Mises Stress	919.516 N/m^2 Node: 9935	5.73544e+006 N/m^2 Node: 17886



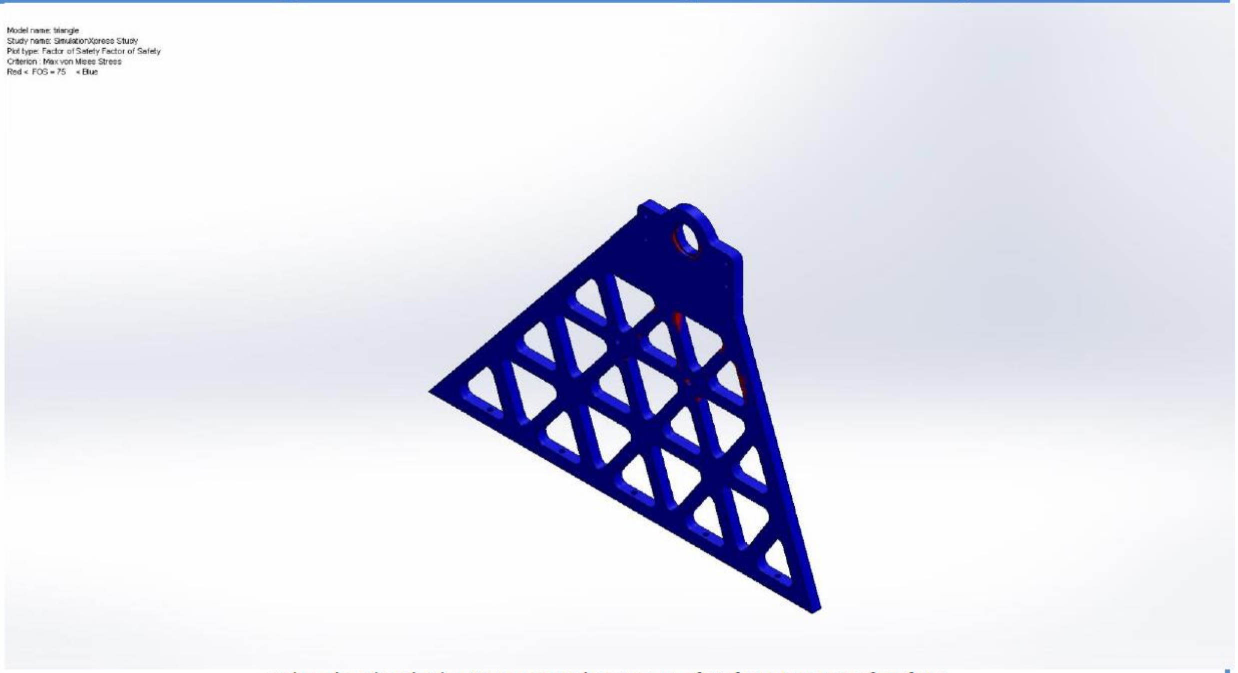
triangle-SimulationXpress Study-Stress-Stress

Name	Type	Min	Max
Displacement	URES: Resultant Displacement	0 mm Node: 136	0.0461163 mm Node: 731



triangle-SimulationXpress Study-Displacement-Displacement

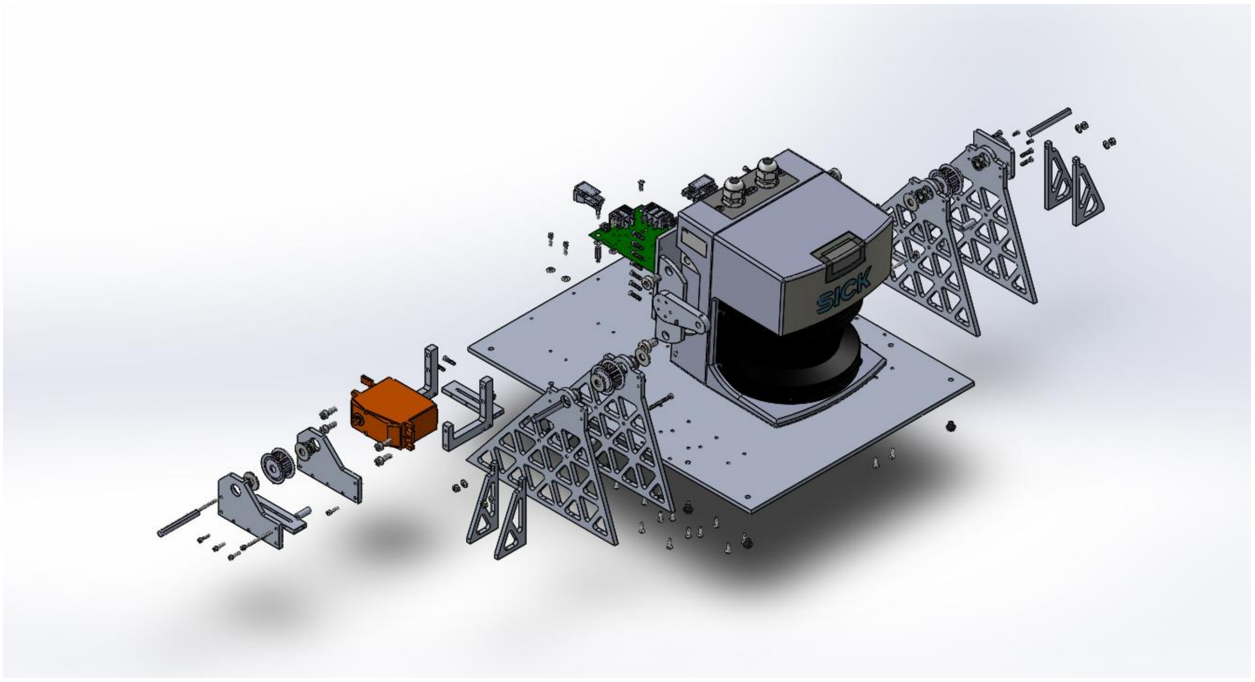
Name	Type	Min	Max
Factor of Safety	Max von Mises Stress	47.9475 Node: 17886	299071 Node: 9935



Model name: triangle
Study name: SimulationXpress Study
Plot type: Factor of Safety Factor of Safety
Criterion : Max von Mises Stress
Red < FOS = 75 < Blue

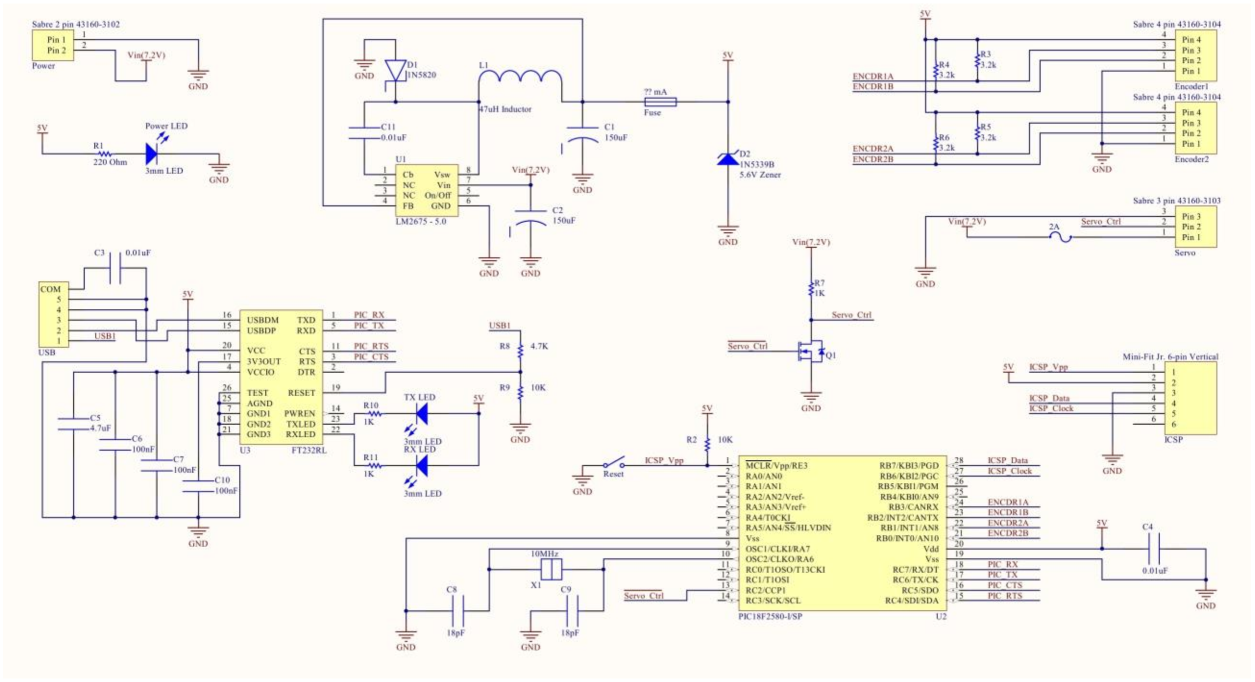
triangle-SimulationXpress Study-Factor of Safety-Factor of Safety

Exploded View

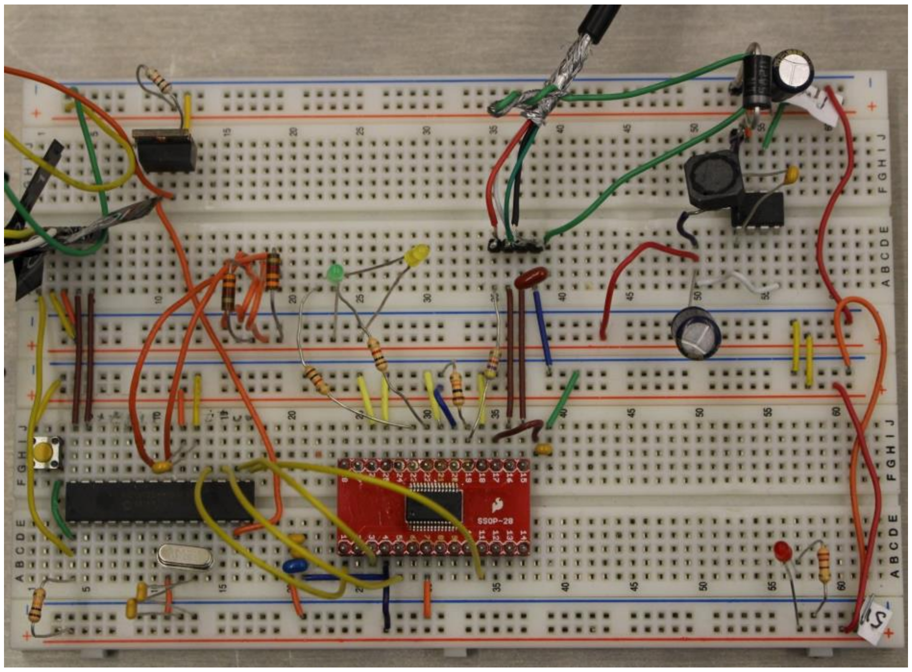


Control Board

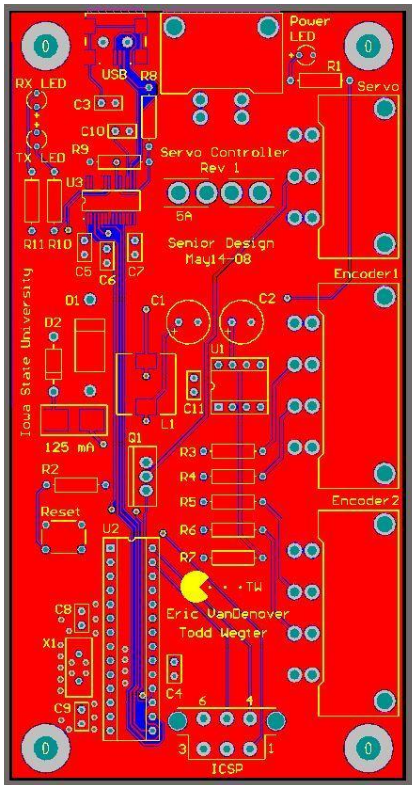
Schematic

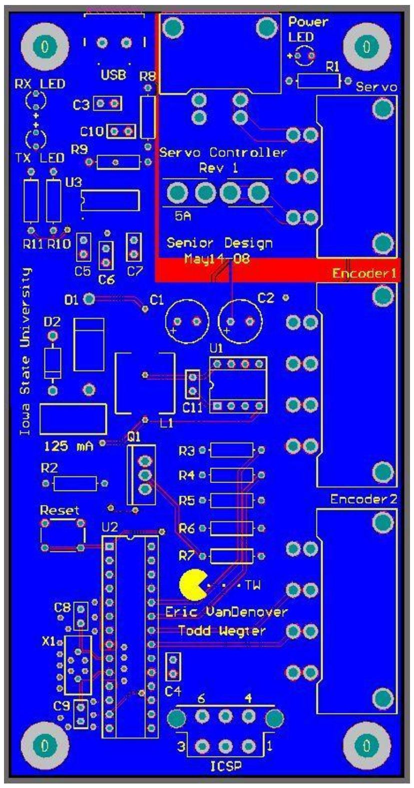


Breadboard

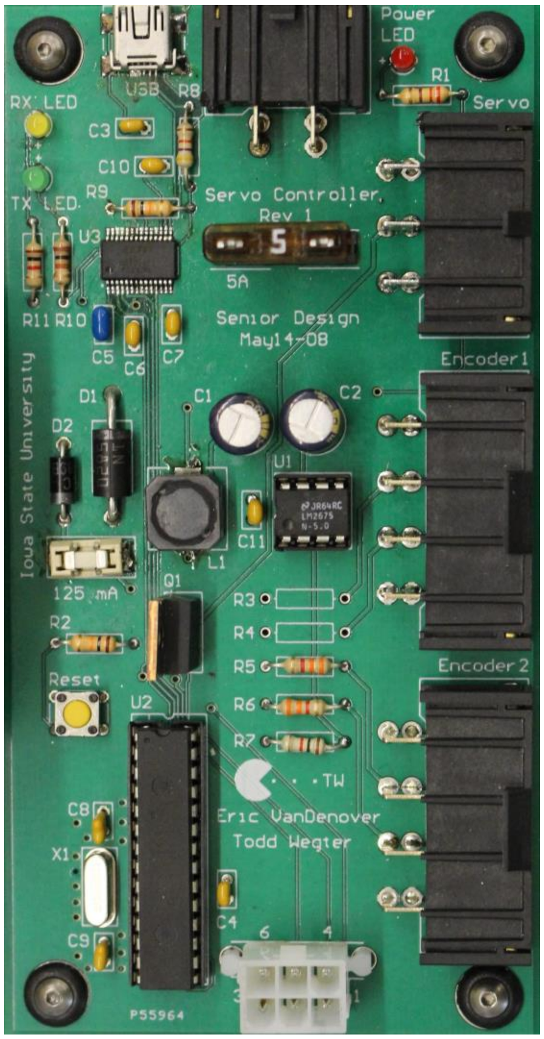


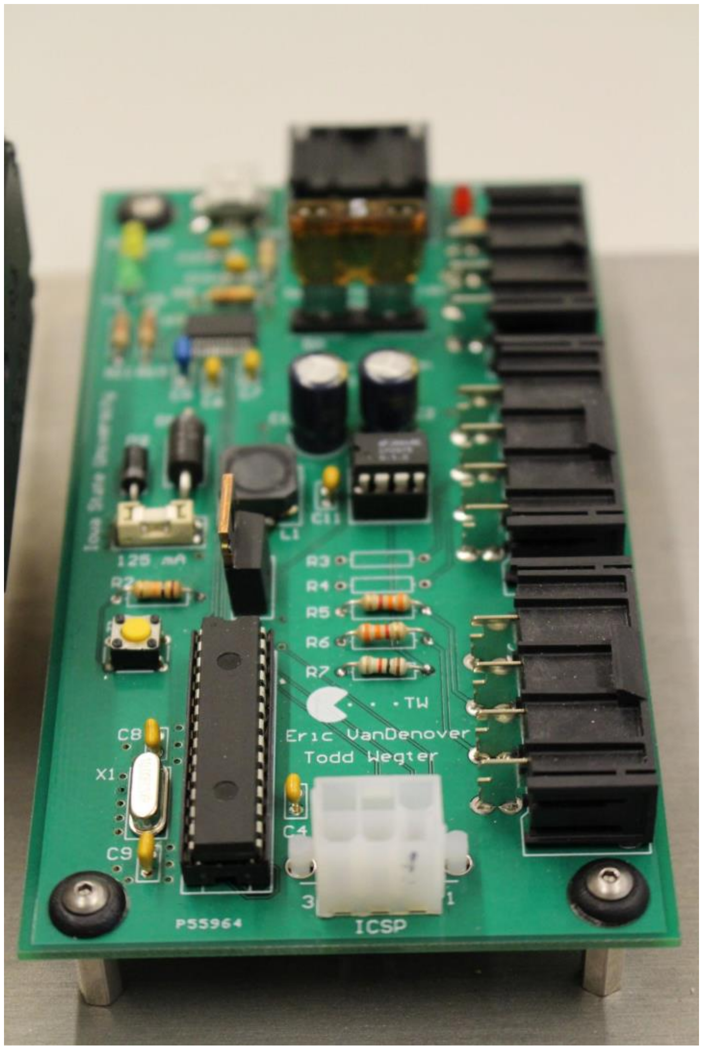
Layout





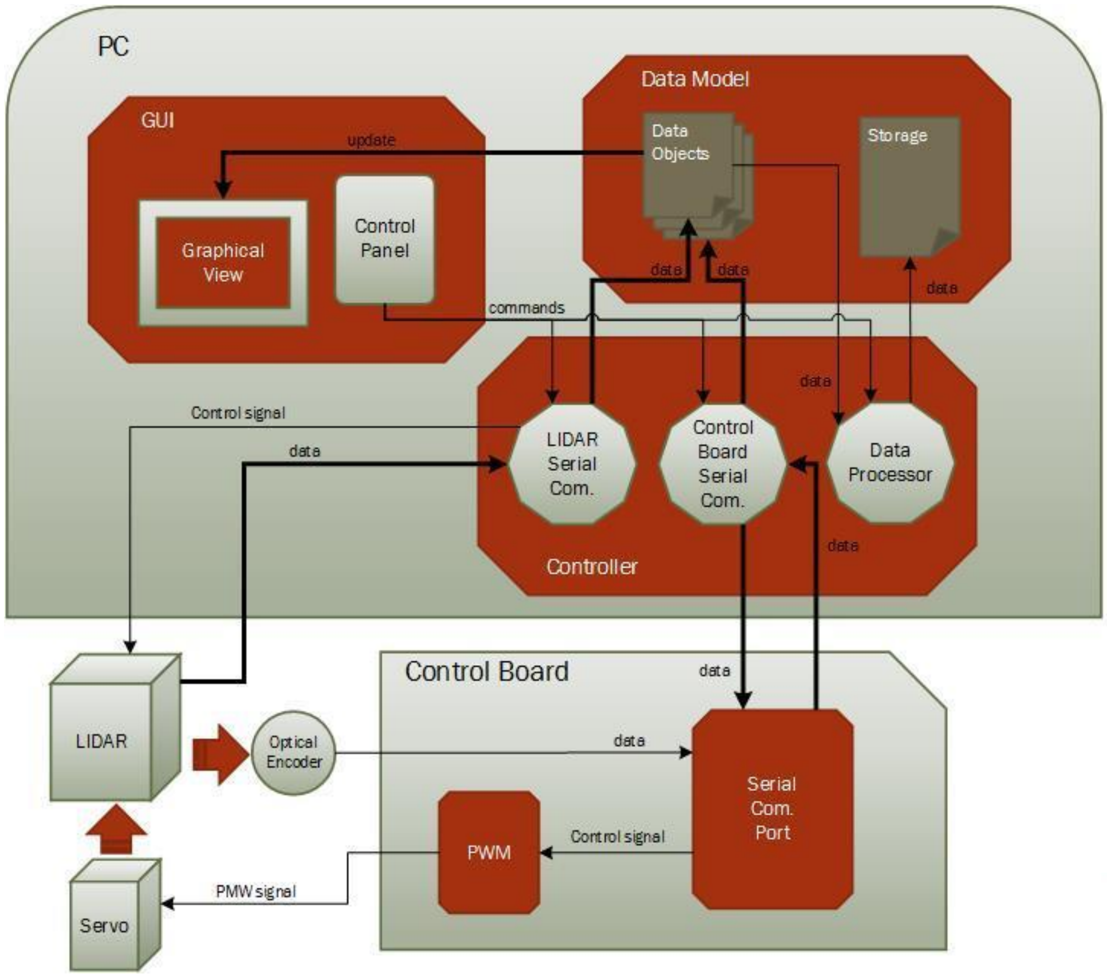
PCB


Power LED
USB
R8
R1
Servo
RX LED
TX LED
C3
C10
R9
Servo Controller Rev 1
U3
5A
R11 R10
C5 C6 C7
Senior Design May14-08
Encoder1
Iowa State University
D1
D2
C1
C2
U1
C11
L1
125 mA
Q1
R2
R3
R4
R5
R6
R7
Reset
U2
Encoder 2
TW
Eric VanDenover
Todd Wegter
C8
X1
C9
C4
6
4
P55964


Iowa State University
125 mA
R2
R3
R4
R5
R6
R7
L1
C11
TW
Eric VanDenover
Todd Wegter
C8
X1
C9
C4
P55964
ICSP
3
1

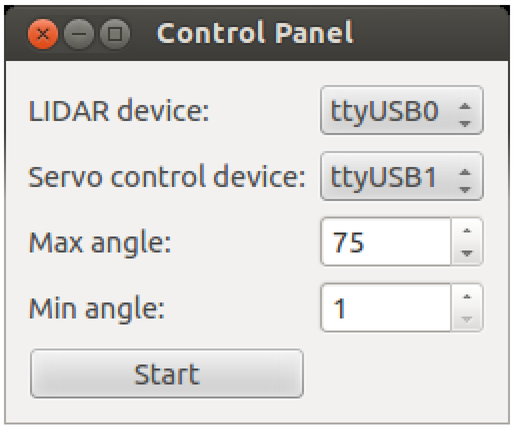
PC Software

Block Diagram



GUI

Control Panel



LIDAR device: ttyUSB0

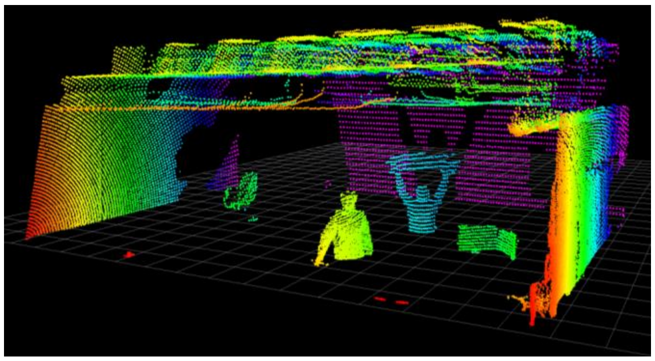
Servo control device: ttyUSB1

Max angle: 75

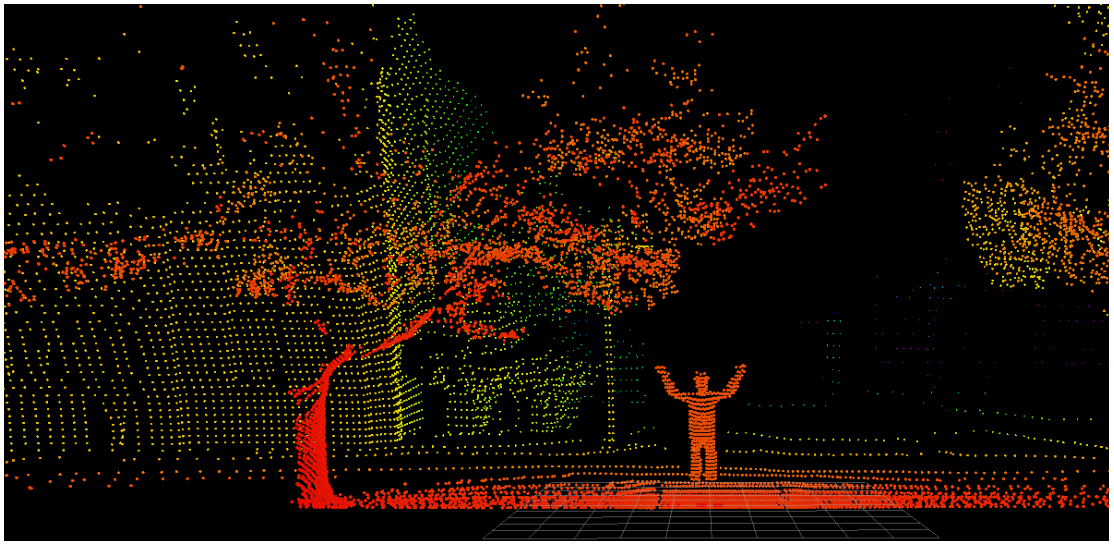
Min angle: 1

Sample Outputs

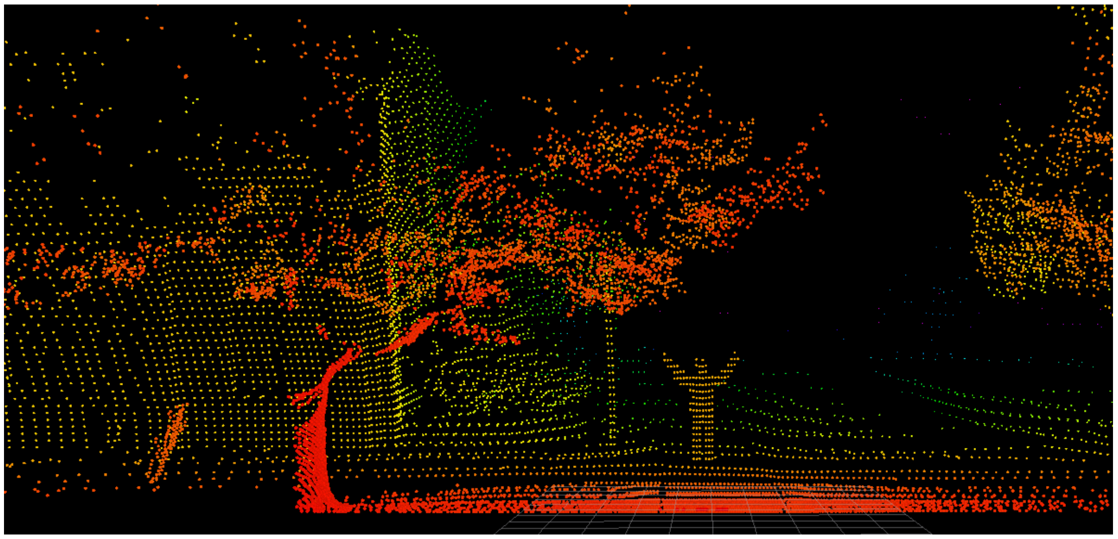
Indoor Scan Test



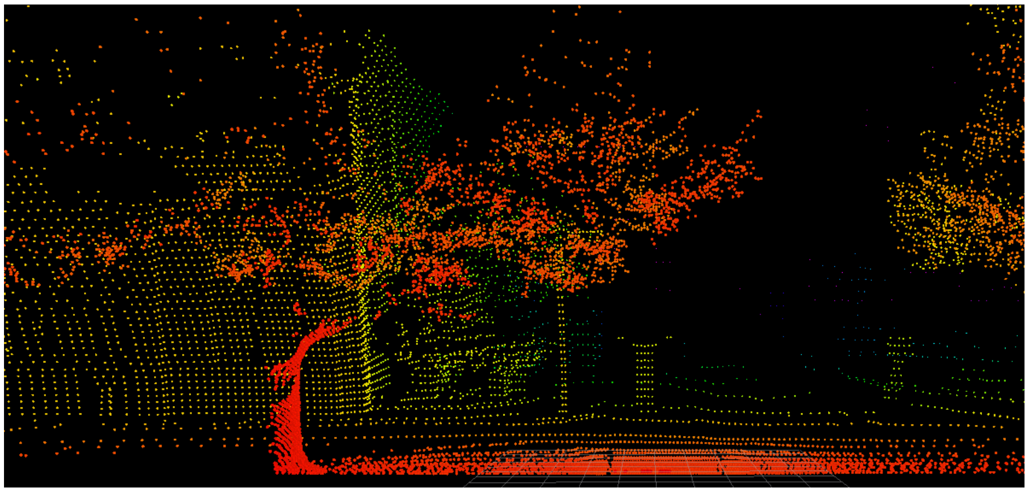
Human Detection Test – 3.5 meters



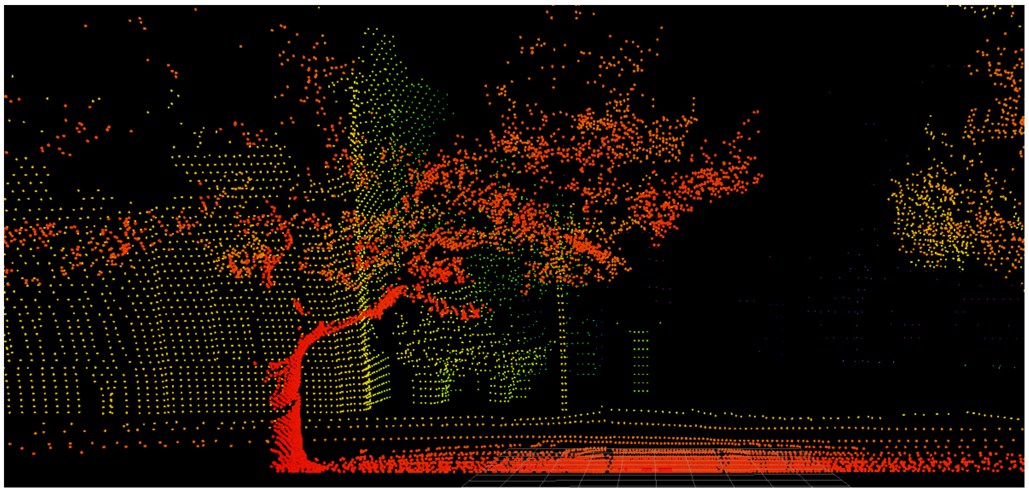
Human Detection Test – 7.2 meters



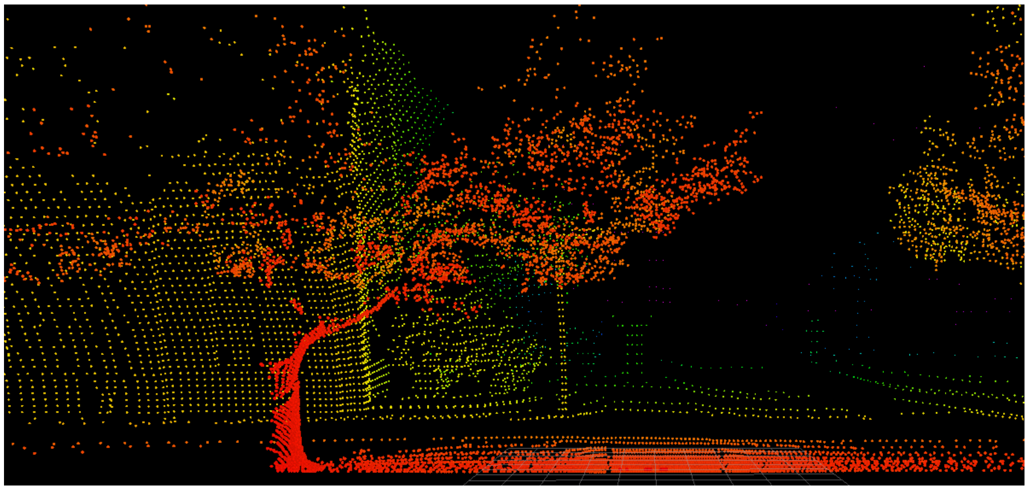
Human Detection Test – 10.9 meters



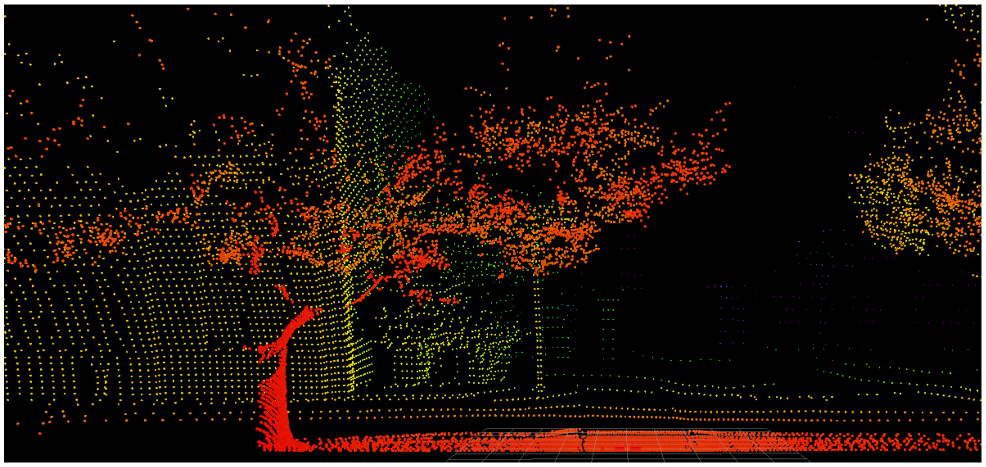
Human Detection Test – 14.6 meters



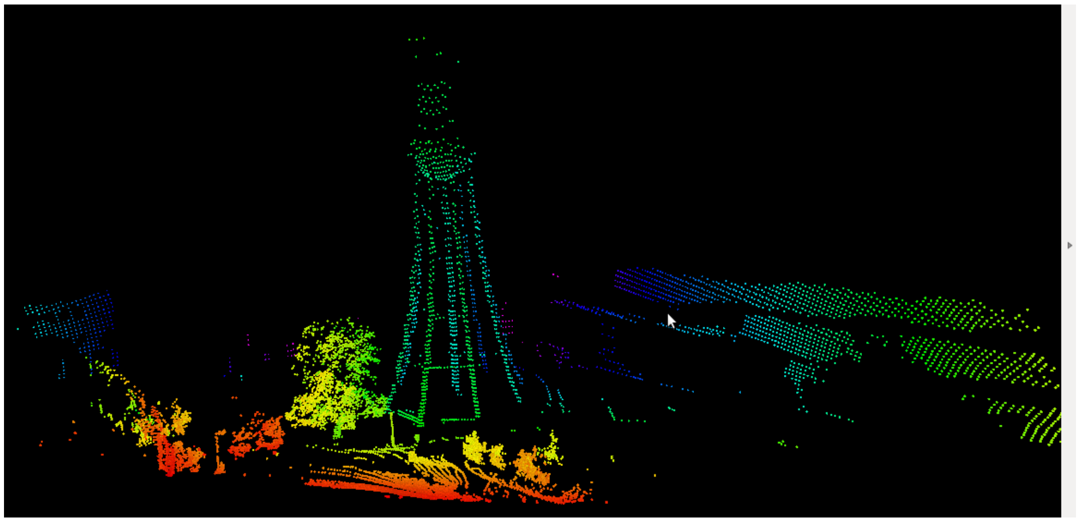
Human Detection Test – 18.3 meters



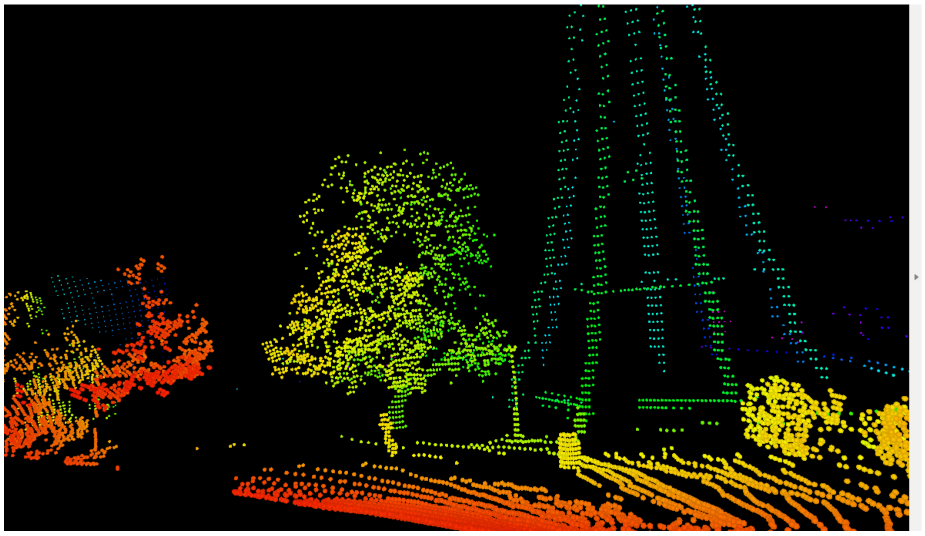
Human Detection Test – 22.7 meters



Outdoor Scan Test – Marston Water Tower



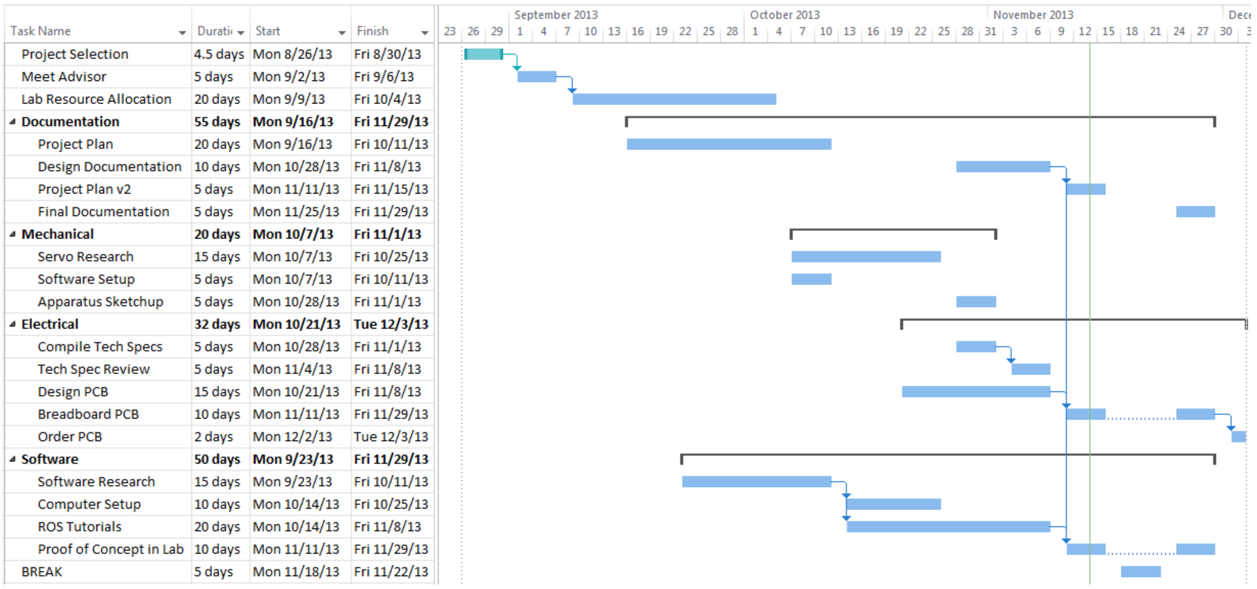
Outdoor Scan Test – Tree



Planning

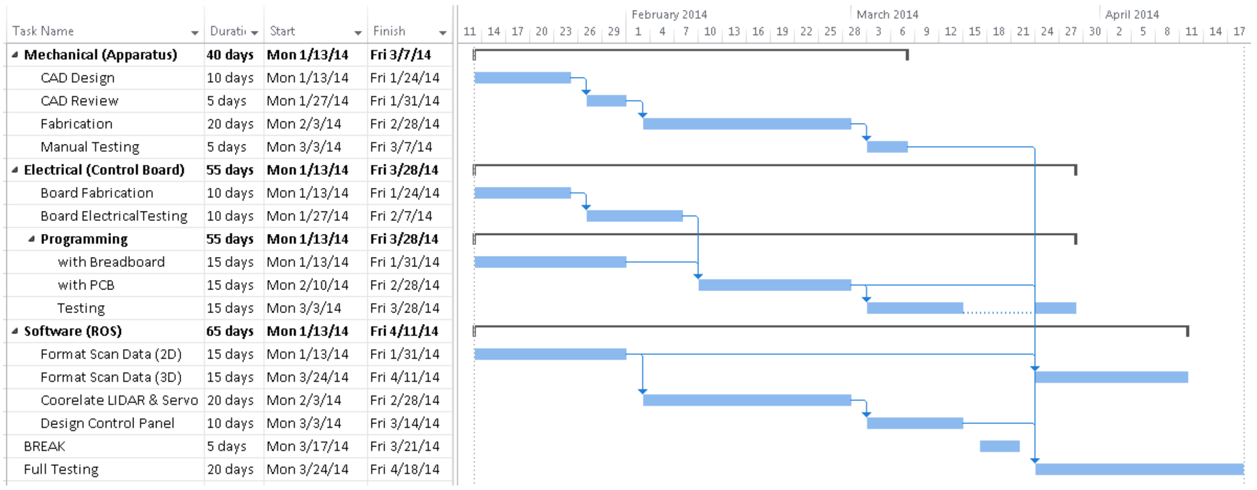
Schedule

Semester 1



Task Name	Duration	Start	Finish
Project Selection	4.5 days	Mon 8/26/13	Fri 8/30/13
Meet Advisor	5 days	Mon 9/2/13	Fri 9/6/13
Lab Resource Allocation	20 days	Mon 9/9/13	Fri 10/4/13
Documentation	**55 days**	**Mon 9/16/13**	**Fri 11/29/13**
Project Plan	20 days	Mon 9/16/13	Fri 10/11/13
Design Documentation	10 days	Mon 10/28/13	Fri 11/8/13
Project Plan v2	5 days	Mon 11/11/13	Fri 11/15/13
Final Documentation	5 days	Mon 11/25/13	Fri 11/29/13
Mechanical	**20 days**	**Mon 10/7/13**	**Fri 11/1/13**
Servo Research	15 days	Mon 10/7/13	Fri 10/25/13
Software Setup	5 days	Mon 10/7/13	Fri 10/11/13
Apparatus Sketchup	5 days	Mon 10/28/13	Fri 11/1/13
Electrical	**32 days**	**Mon 10/21/13**	**Tue 12/3/13**
Compile Tech Specs	5 days	Mon 10/28/13	Fri 11/1/13
Tech Spec Review	5 days	Mon 11/4/13	Fri 11/8/13
Design PCB	15 days	Mon 10/21/13	Fri 11/8/13
Breadboard PCB	10 days	Mon 11/11/13	Fri 11/29/13
Order PCB	2 days	Mon 12/2/13	Tue 12/3/13
Software	**50 days**	**Mon 9/23/13**	**Fri 11/29/13**
Software Research	15 days	Mon 9/23/13	Fri 10/11/13
Computer Setup	10 days	Mon 10/14/13	Fri 10/25/13
ROS Tutorials	20 days	Mon 10/14/13	Fri 11/8/13
Proof of Concept in Lab	10 days	Mon 11/11/13	Fri 11/29/13
BREAK	5 days	Mon 11/18/13	Fri 11/22/13

Semester 2



Task Name	Duration	Start	Finish
Mechanical (Apparatus)	**40 days**	**Mon 1/13/14**	**Fri 3/7/14**
CAD Design	10 days	Mon 1/13/14	Fri 1/24/14
CAD Review	5 days	Mon 1/27/14	Fri 1/31/14
Fabrication	20 days	Mon 2/3/14	Fri 2/28/14
Manual Testing	5 days	Mon 3/3/14	Fri 3/7/14
Electrical (Control Board)	**55 days**	**Mon 1/13/14**	**Fri 3/28/14**
Board Fabrication	10 days	Mon 1/13/14	Fri 1/24/14
Board ElectricalTesting	10 days	Mon 1/27/14	Fri 2/7/14
Programming	**55 days**	**Mon 1/13/14**	**Fri 3/28/14**
with Breadboard	15 days	Mon 1/13/14	Fri 1/31/14
with PCB	15 days	Mon 2/10/14	Fri 2/28/14
Testing	15 days	Mon 3/3/14	Fri 3/28/14
Software (ROS)	**65 days**	**Mon 1/13/14**	**Fri 4/11/14**
Format Scan Data (2D)	15 days	Mon 1/13/14	Fri 1/31/14
Format Scan Data (3D)	15 days	Mon 3/24/14	Fri 4/11/14
Coorelate LIDAR & Servo	20 days	Mon 2/3/14	Fri 2/28/14
Design Control Panel	10 days	Mon 3/3/14	Fri 3/14/14
BREAK	5 days	Mon 3/17/14	Fri 3/21/14
Full Testing	20 days	Mon 3/24/14	Fri 4/18/14

Cost Analysis

	Market Value	Our Cost
SICK LMS-291 LIDAR	$6000	
RS-422 Cable and Adapter	$60	
Savox XV-0236MG Servo Motor	$110	$100
Servo Controller	(Varies)	
Parts		$50
PCB Fabrication		$0
Desktop Computer	(Varies)	
Software	$0	
Power Supplies	$100	$100
Optical Encoder	$50	$50
Mechanical Apparatus Materials	$300	$300
Machining Time	$1500+	$0
Total	**$8120+**	**$600**

Appendix B: Operation Manual

Software Installation Guide

1. Install Ubuntu

    Note: ROS supports Ubuntu 12.04 (Precise), Ubuntu 12.10 (Quantal), and Ubuntu 13.04 (Raring). We recommends Precise since it was used for our development.

2. Install ROS Hydro: Desktop full install

    http://wiki.ros.org/hydro/Installation/Ubuntu

3. Install the ROS sicktoolbox library

```
apt-get install ros-hydro-sicktoolbox
```

4. Download our solution from GitHub

    https://github.com/ncabeen/May14-08

5. Build our solution software

```
cd <download directory>/May14-08/lidar/

source /opt/ros/hydro/setup.sh

catkin_make
```

Operation Guide

General Operation

1. Connect the 24V power source to the LIDAR.

2. Connect the 7.2V power source to the servo controller.

3. Connect the LIDAR to the PC with an RS-422 converter cable.

4. Connect the servo controller to the PC with a USB cable.

5. Launch the application:

```
<download directory>/May14-08/startup_script.bash
```

Servo Controller Debug Mode

While the PCB is powered and all components are connected, the microcontroller can be accessed via the PC's terminal to view the data directly.

1. Follow steps 1-4 from the "General Operation" guide above.

2. Open 'Terminal' on the PC.

3. Use the screen command to access the microcontroller:

```
screen /dev/<path-to-microcontroller-device> 57600
```

4. Hold down the 'h' key on the keyboard for approximately 3 seconds. Text should begin to scroll in the terminal window.

5. Use the following keys to control the system:

    '1' - go to -98 degrees

    '2' - go to -45 degrees

    '3' - go to 0 degrees

    '4' - go to 45 degrees

‘5’ - go to 63 degrees

‘z’ - go down 1 degree

‘x’ - go up 1 degree

‘t’ - toggle the encoder feedback loop on or off

‘m’ - go back to “machine mode”

‘ctrl+a’ then ‘\’ then ‘y’ to exit screen