Design Document Rev. 2

MAY14-08 3D LIDAR

Members

Nicolas Cabeen – Project Leader

Todd Wegter – Communications & Mechanical

Xiang “Peter” Wang – Software

Eric VanDenover – Electrical

Advisor

Koray Celik

Contents

Problem/Need Statement **3**
Concept Sketch **3**
System Block Diagram **4**
Functional Requirements **4**
Non-Functional Requirements **5**
System Description **5**
Identification of Subcomponents **5**
Interface Software 5
Scanning 5
Structure and Movement 5
Power Supplies 5
Identification of Third-Party Components **6**
Interface Software 6
Scanning 6
Structure and Movement 6
Power Supply 6
System Analysis **6**
Interface Software 6
Scanning 7
Structure and Movement 7
Power Supply 8
Supply the power necessary for each component to operate. **8**
Input/Output Specification **8**
System Level 8
Subcomponent I/O 8
User Interface Specification **10**
Software Specification **10**

Software Breakdown 11
Hardware Specification 11
Desktop Computer 11
USB 2.0 Cable with Converter to RS-422 (DB9) 11
RS-422 Cable 12
LIDAR 12
USB 2.0 Cable with Mini-B Connector 12
Servo Controller 12
Encoder 12
Servo 12
Mechanical Apparatus 12
Power Supply 12
Test Plan 13
Software Unit Testing 13
Mechanical Stress Testing 13
Electrical Testing 13
System Test Environment 13
Overall Function Testing 13
Simulation, Modeling, and Prototypes 14
Software Prototype 14
PCB Prototype 14
Mechanical Apparatus Prototype 14
Mock Product Installation Environment 14
Mechanical CAD 15
Electrical CAD 16
Software Design 17
Standards 17
RS-422 17
RS-232 18
Universal Serial Bus 2.0 18
Laser Classification 18
SAE 18

Problem/Need Statement

3D LIDAR is a very expensive piece of technology which can cost upwards of $100,000. 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 Lunabotics team currently needs a sensor that can detect objects in extreme environments.

Concept Sketch

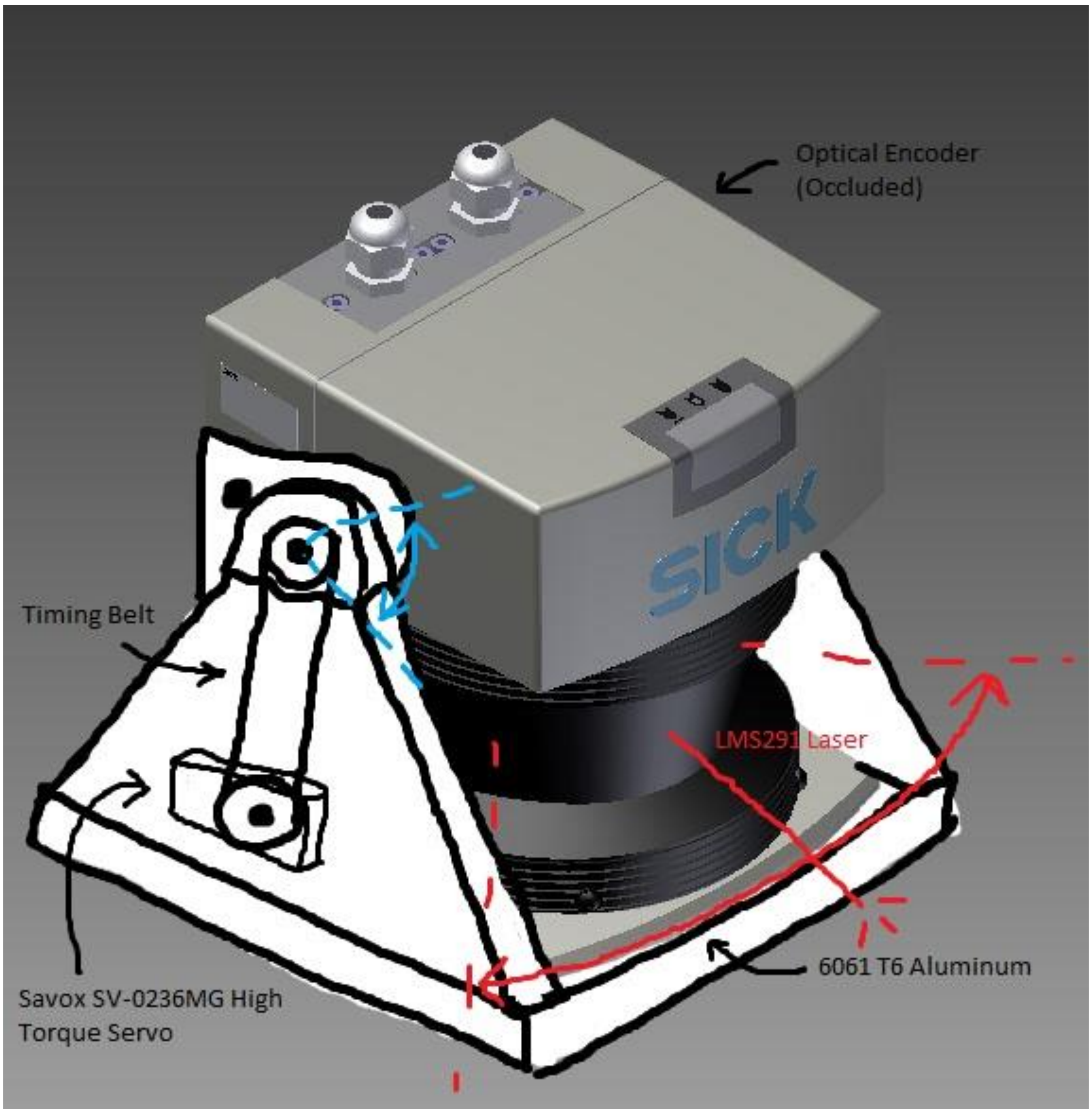

Figure 1 - System Concept Sketch

System Block Diagram

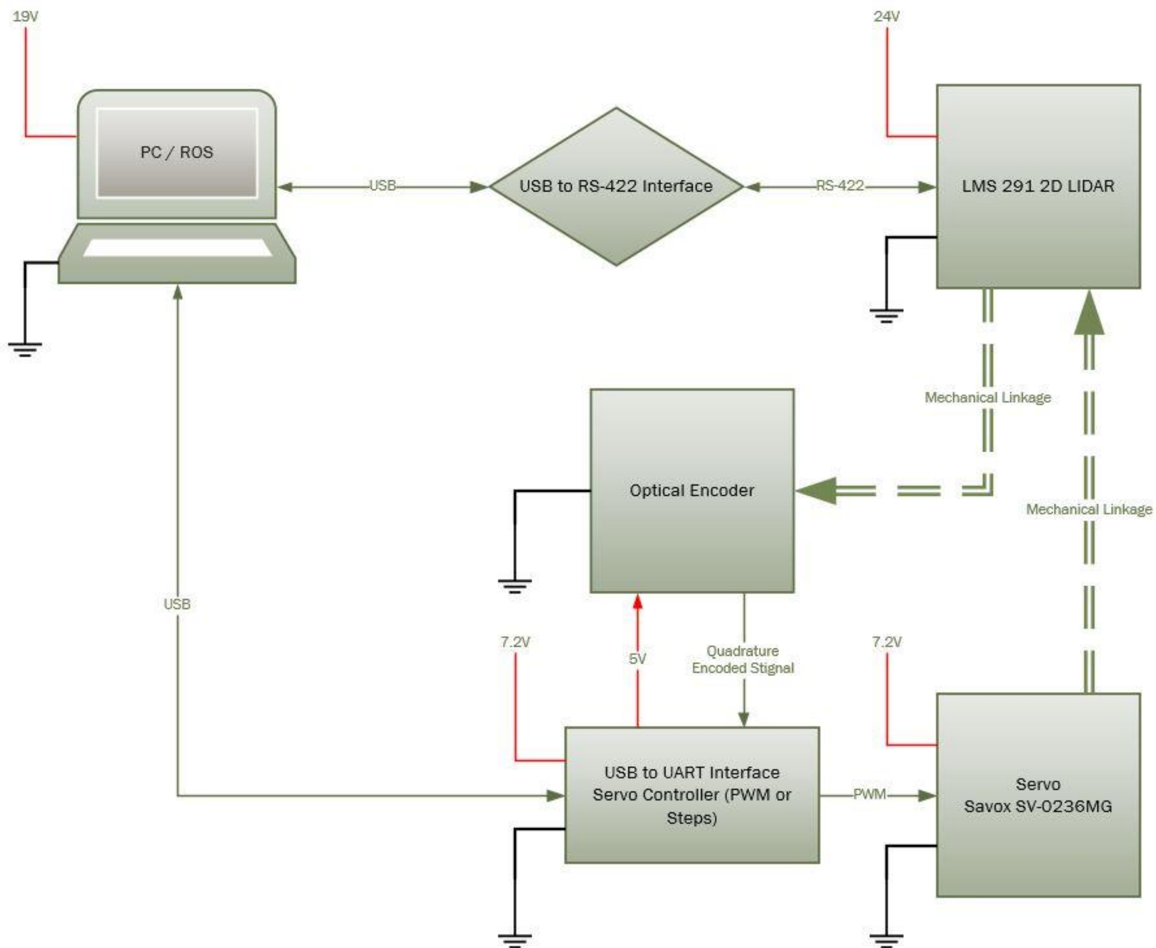

Figure 2 - System Block Diagram

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 setup must provide positional feedback (ie - be closed-loop)
   Servo must withstand 600 oz/in of torque
   Scanning apparatus must scan at least 120 degrees

3. Software
   Deliver scan results in a human-readable format
   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
   Prototype should be tested while mounted on the provided wheeled cart

2. Software
   User should be able to operate the system with one window

3. Cost
   Any needed software should be open source or widely available
   Cost of prototype must be much less than a 3D LIDAR

System Description

The LIDAR system creates a three dimensional representation of scanned data via a computer. Users interact with the computer interface to control scanning hardware. This hardware provides the computer with data which is then converted into a human readable format for easy viewing.

Identification of Subcomponents

Interface Software

Display
Communication
Data Processing

Scanning

Signal Converter (USB to RS-422)
SICK LMS-291 LIDAR

Structure and Movement

Mechanical Apparatus
Servo
Servo Controller

Power Supplies

Identification of Third-Party Components

Interface Software

Desktop Computer running Ubuntu 12.04LTS
ROS

Scanning

SICK LMS-291 LIDAR
USB-to-RS-422 converter

Structure and Movement

Savox SV-0236MG Servomotor
PIC-18F2580 Microcontroller
FT232RL USB to UART Converter
Encoder (TBD)

Power Supply

24V 10A DC power supply (LIDAR)
0-30V 13A 400W adjustable DC power supply (Servo and Servo Controller)

System Analysis

Interface Software

Display: The display function will need to receive data from the Data Processing function and output the information on the computer screen graphically. This graphical information must be human readable and relay the distance and direction measurements. It will also need to have buttons and fields available to receive input from the user.

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: This part of the software will translate the raw data from the LIDAR into a properly formatted two dimensional matrix. The coordinates of the matrix will be directly related to the position which was scanned and the values in each position will contain the corresponding distance measurements.

Scanning

Signal Conversion: This function will simulate a serial port over USB for communication between the LIDAR and the computer. A USB to RS-422 converter will handle the conversion from USB (computer) to RS-422 (LIDAR) and vice versa.

LIDAR: This device will receive commands from the computer and complete two dimensional scans accordingly. It will need to send the corresponding output to the computer as soon as it is available.

Structure and Movement

Mechanical Apparatus: The apparatus will need to hold the weight of the LIDAR and allow it to rotate up and down (in a "nodding" movement). It will also need to allow the servomotor to mount to the side and drive the LIDAR.

Servo: The servo will receive a PWM signal from the servo controller and rotate the LIDAR +/- 60 degrees. To do this at a high rate of speed it will need a large amount of torque and be accurate.

Servo Controller: The servo controller will communicate positioning data to the computer from the encoders and provide the PWM signal to drive the servomotor. It will also need to receive positioning instructions for the servo from the computer.

Encoders: The encoder will improve accuracy and provide feedback for the servo position. It will optically read the position and provide a quadrature encoded signal to the servo controller.

Power Supply

Supply the power necessary for each component to operate.

Input/Output Specification

System Level

User to ROS

The user will be able to interact with the apparatus via a ROS user interface. The user will be able to issue input and configuration commands to control the apparatus and the LIDAR. ROS will have a window (rviz - a GUI plugin for ROS) that will display a graphical representation of the scan measurements. System status will be returned to the user through the UI as well. Using ROS, the user will have the option to export settings and an image file of the scan measurements.

ROS to LIDAR

ROS will communicate with the LIDAR via a USB-to-DE9 converter cable using the RS-422 protocol. ROS will send configuration and request commands to the LIDAR. The LIDAR will return status signals and scan measurements.

ROS to Servo Controller *to Servo*

ROS will also communicate with the servo controller board via UART over USB. On the controller board, we will include a FT232RL converter to convert USB to UART and simulate a serial port. After the signal is processed on the board, a PWM signal will be sent to the servo to physically move the apparatus to the specified angle.

Optical Encoder to Servo Controller

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

Subcomponent I/O

ROS

rviz Display: The rviz display window will show a real-time graphical representation of the incoming scan data from the LIDAR coupled with the servo's position.

Servo Start
Angle Control: This box will accept integer input between 0-180. If valid, it will send a signal to the servo control board to adjust the servo accordingly. This value must be less than or equal to the stop angle.

Servo Stop
Angle Control: This box will accept integer input between 0-180. If valid, it will send a signal to the

servo control board to adjust the servo accordingly. This value must be greater than or equal to the start angle.

Servo Speed Control: This box will accept a floating point number specifying the speed of rotation for the servo. The valid range is TBD pending final specifications of our servo. If valid, it will send a signal to the servo control board to adjust the servo accordingly.

LIDAR Resolution Control: This box will allow the user to specify the scan resolution in degrees. For the SICK LMS-291, acceptable values are 0.25, 0.5, and 1. If valid, it will send a signal to the LIDAR to configure it appropriately.

LIDAR Communication Rate Control: This box will allow the user to specify the serial communication speed to and from the LIDAR. For the SICK LMS-291, acceptable values are 9600, 19200, 38400, and 500000. If valid, it will send a signal to the LIDAR to configure it and reconfigure ROS to communicate at the new rate. The default value will be 9600 to correlate to the default speed set by the LIDAR during power-on.

Capture Control: The Capture control button will take a snapshot of the rviz window and prompt the user to save the image file to a disk.

Export Control: The Export control button will allow the user to export the raw scan data collected from the LIDAR to a CSV file. It will prompt the user to save the file to a disk.

Start Scan Control: The Start Scan control button will initialize the LIDAR and servo to begin scanning and collecting data.

Stop Control: The Stop control button will immediately send a signal to the servo controller to cease nodding the apparatus. It will also signal ROS to halt collecting data from the LIDAR.

Servo

The Savox SV-0236MG will accept PWM signals from the servo controller as input. It will not electrically return any signals, but instead will physically output mechanical energy to move the apparatus.

Optical Encoder

The optical encoders (specific model - TBD) will optically read markings placed on the LIDAR/mechanical apparatus. The encoder will communicate back to the servo controller using quadrature encoding.

User Interface Specification

When the system is under operation, the user will mainly be interacting with the graphical interface that is embedded to control the physical hardware. The software will be listening for data inputs from the user and execute them properly. The user needs to be able to configure the LIDAR and servo at any given time. The user should also be able to start and stop scanning, take screenshots while operating, and export data as a CSV file when the corresponding button is clicked. The figure below shows a rough design as for how the software UI would look like.

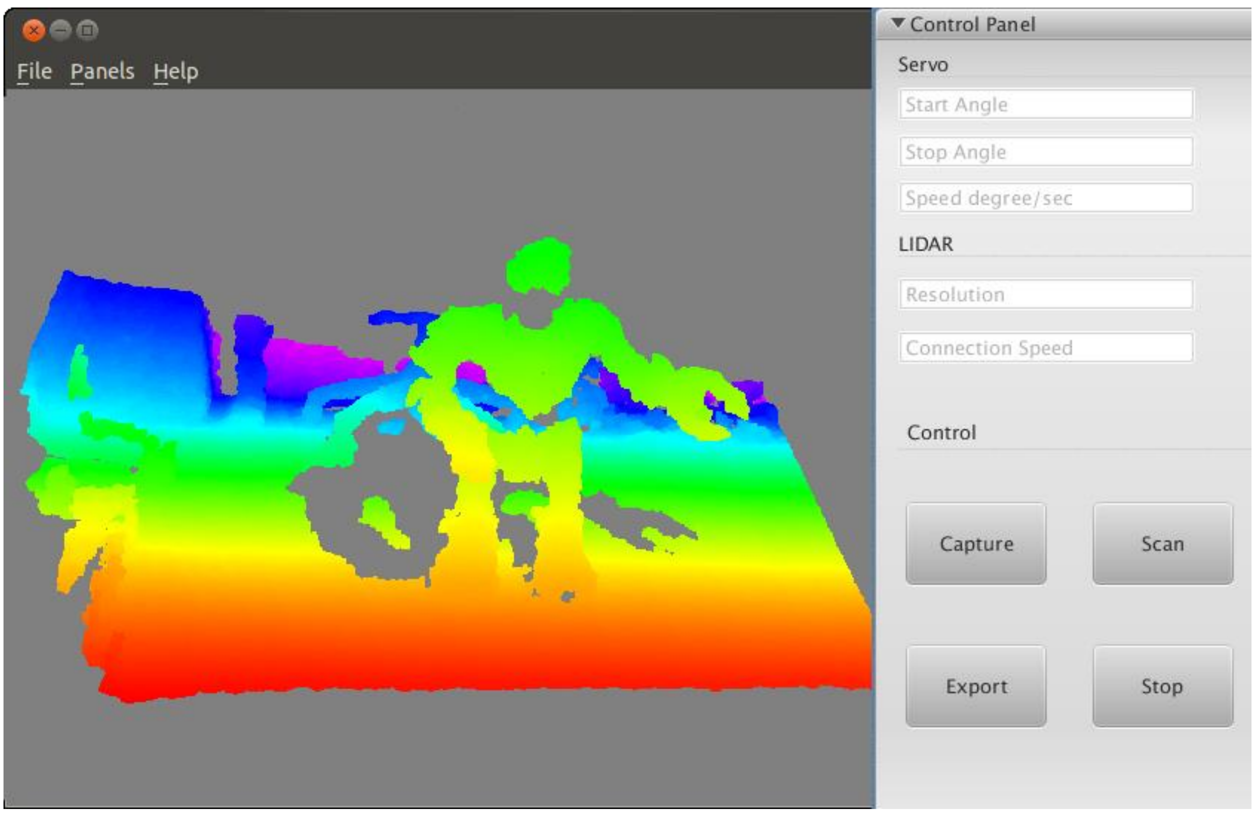

Figure 3 - Photo courtesy upenn.edu. Available at http://mediabox.grasp.upenn.edu/roswiki/depth_image_proc.html

Software Specification

We will be using ROS (Robot Operating System), a software framework for robot software development, to implement the graphical software. ROS is open-source and it provides many libraries that we will use to build the GUI, interface with the LIDAR, and communicate with serial ports. These libraries include sicktoolbox_wrapper, rosserial, rqt, roscpp, etc. Below are the software specifications.

Languages: ROS hydro
C++

Compiler: Using rosmake which uses the cmake family
Specified in rostoolchain.cmake of rosbuild
-CMAKE_C_COMPILER uses the arm-linux-gcc
-CMAKE_CXX_COMPILER uses the arm-linux-g++

Software Breakdown

Graphical Screen: The screen display will be using plc_ros, a point cloud library, to plot the points in a 3 dimensional view. Then it will use rviz to display the view.

Control Button Panel: The control button panel will be integrated into rviz using the rqt library. rqt provides a sub-package called rqt_gui and can be used to implement GUIs. Also the library roscpp will be used to help with this implementation, as well as to export files.

Hardware Interface: sicktoolbox is a driver for the SICK LIDAR and sicktoolbox_wrapper is a library that provides the API to talk to the LIDAR. We will be using these software to control and obtain data for the LIDAR. We will also be using rosserial to communicate with the control board which will send PWM signals to the servomotor.

Hardware Specification

We are designing the servo controller and mechanical apparatus for this project. All other hardware will be a third party component.

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 not exceed one kilometer in length (we will only use a few feet of cable)

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
- To be determined

Servo

- Savox SV-0236MG Servomotor
    - 416.6 – 555.5 oz-in of torque
    - .17-.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

The 3D LIDAR will be comprised of three different subsystems: software, electrical, and mechanical. Each of these subsystems will need to be independently tested and verified before integration into the whole system. These tests will be designed to show that each subsystem has met its functional requirements. Finally, we will test the whole system to ensure that it functions as intended.

Software Unit Testing

We will be testing the software subsystem independently in a controlled environment to ensure specific functionality. By their nature, unit tests are a more controlled environment. This will allow us to verify that our software will work in many different environments before conducting any system tests.

Mechanical Stress Testing

The mechanical component of our system will be over-built such that finite stress tests should be unnecessary. However, if system testing reveals any weaknesses in the mechanical design, we will address and resolve these issues as needed.

Electrical Testing

Our 3D LIDAR will have a custom servo controller. This system will first be tested on a breadboard before ordering a PCB. The tests for this subsystem will be largely dependent on proper operation of its code, so these tests will be testing not only electrical functionality, but the functionality of its code. After proper operation of the design is confirmed, a PCB will be ordered and tested for proper operation.

System Test Environment

The 3D LIDAR will be tested in several environments to ensure that it can work under a reasonable range of conditions. For example, this means we will be testing the system as a whole both indoors and outdoors to verify that different background lighting doesn't affect the system. This doesn't mean that the 3D LIDAR should work in a hail storm, but we will be showing through our test environments that it will work in any environment one would expect a laptop to work in.

Overall Function Testing

We will verify the functional operation of the 3D LIDAR by showing that the system performs as expected in several common environments (for example, indoors, outside on a cloudy day, outside on a sunny day, etc.).

Simulation, Modeling, and Prototypes

Software Prototype

Thanks to the modular model provided by ROS, we are able to design and test our software at a relatively granular level. We are also able to swap designs with relative ease since we are adhering to the MVC software design model. For example, we are able to create multiple revisions of our graphical rviz window that look very different and yet know they will work with the rest of our design as long as they use the same interface.

PCB Prototype

We have finished a first draft layout for our servo control board and are currently in the process of developing a breadboard model of it. This will allow us to ensure our design is functioning and identify problem spots before final production.

Mechanical Apparatus Prototype

We have also begun creating a 3D model of our mechanical apparatus. Once completed, we will be able to minimize our construction material costs and reduce the amount of time machining the parts physically.

Mock Product Installation Environment

We will attach the apparatus and LIDAR to a table or mobile base for demonstration purposes. This setup will give us a basic functional equivalent of an end-user's installation.

Mechanical CAD

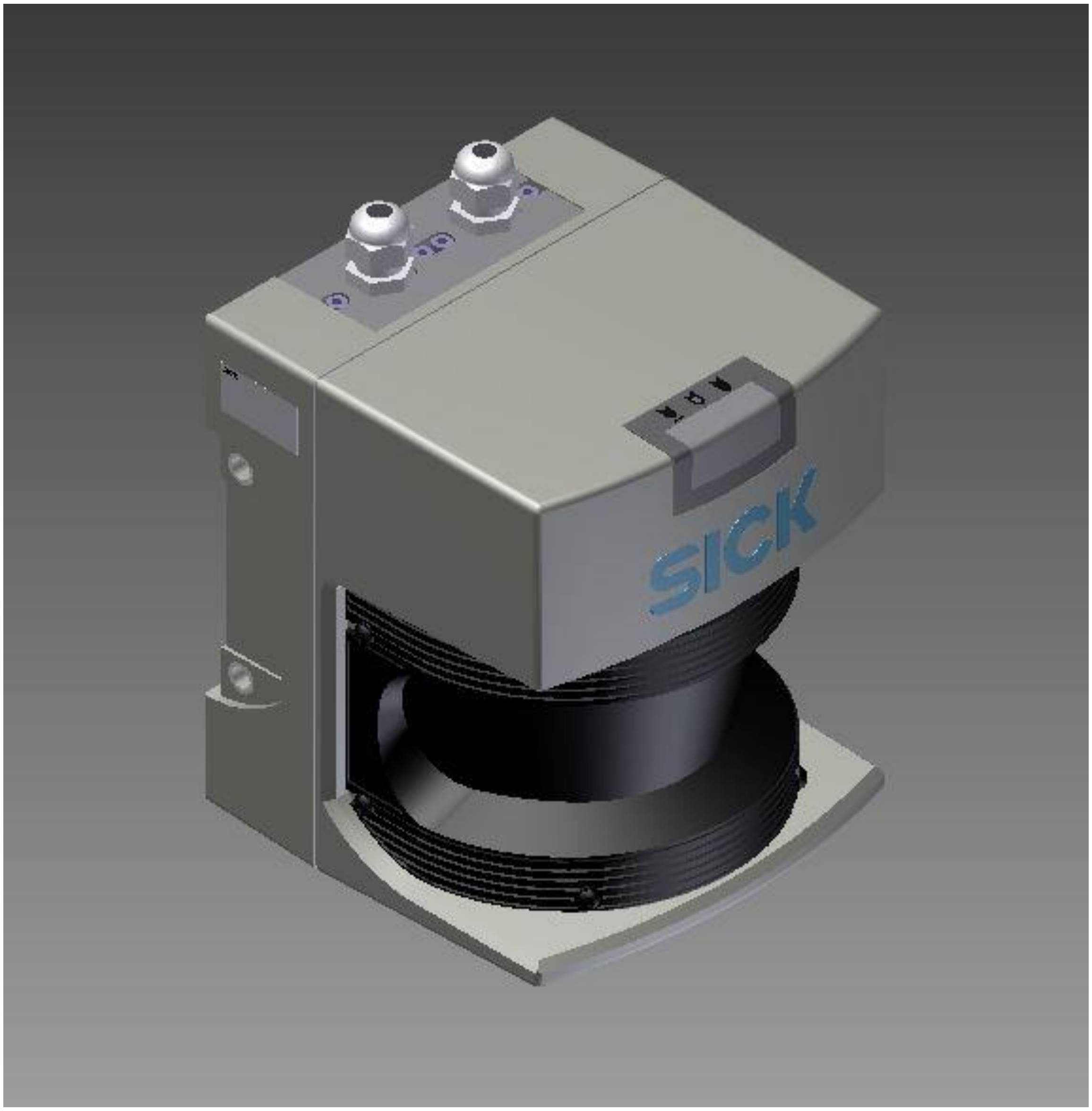

Figure 4 - SICK LMS-291 2D LIDAR

This is a CAD model of the SICK LMS-291 LIDAR. We will soon be adding the CAD of the mechanical apparatus and servo.

Electrical CAD

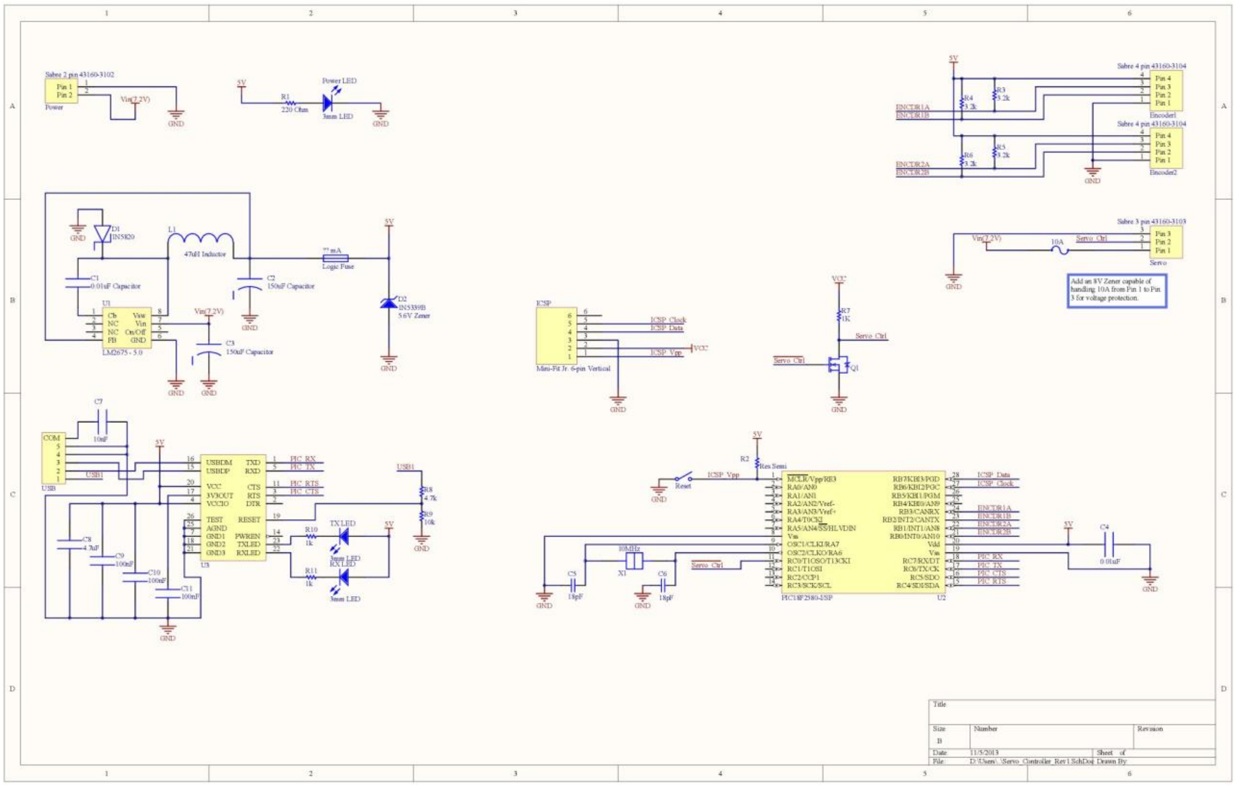

Figure 5 - Servo Controller Scematic

Software Design

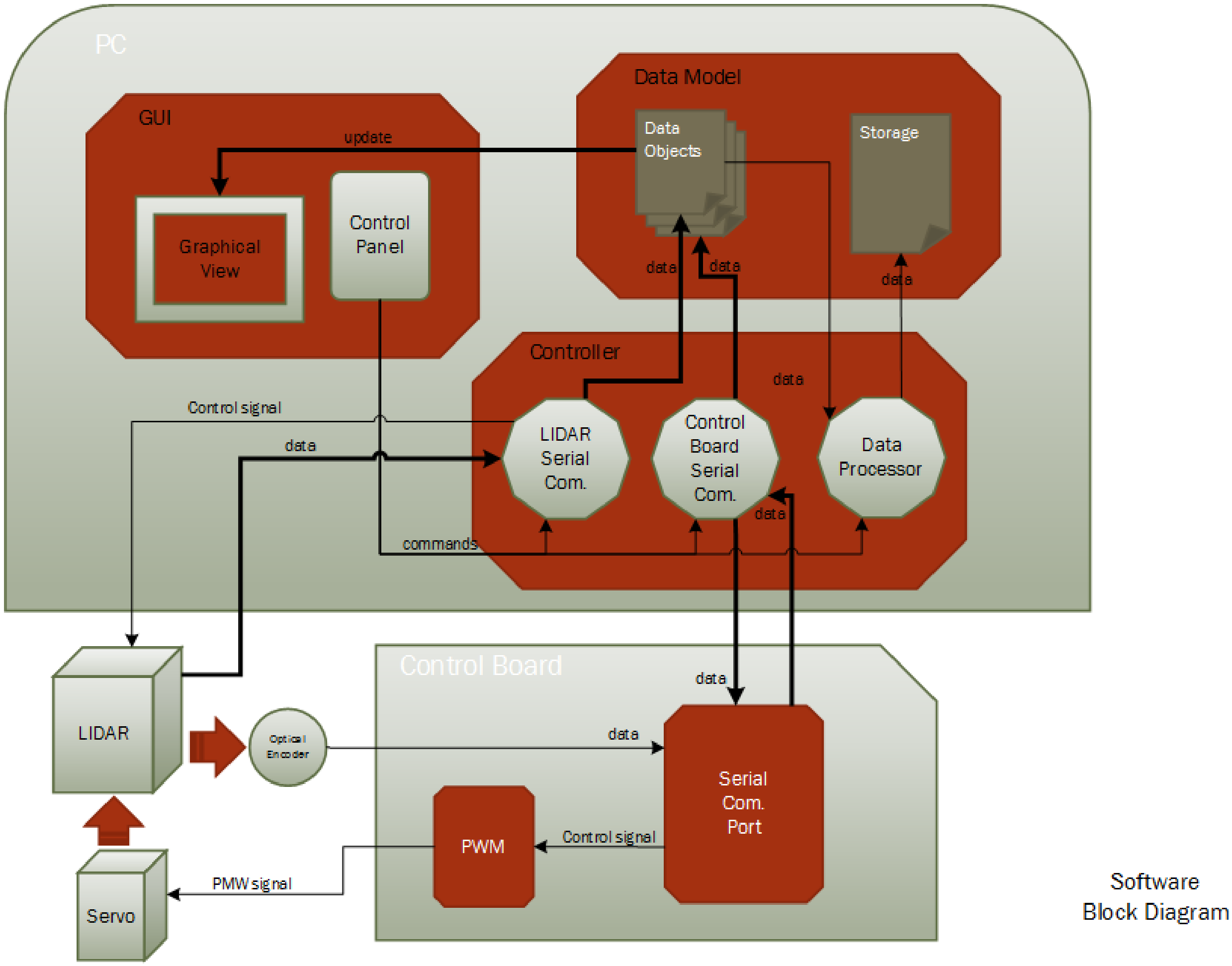

Figure 6 - Software Design Diagram

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 will be using RS-422 to interface from a 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 data. We also discussed using RS-232 for the interface to the LIDAR, but went against it since we will need the higher baud rate offered by RS-422 (10Mb/s).

RS-232

RS-232 is a communication protocol standardized in the 1960s. It is a very slow protocol (max baud rate 112.5 Kbps) but is also very easy to implement. We will use RS-232 on our servo controller PCB 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 will use USB to simplify our RS-422 and R-S232 connections since our computer (and most modern computers) doesn't have 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 will need 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. As long as we leave the LIDAR unaltered from production state (that is, not open the casing), we can be confident that our project will meet Class 1 standards once we have converted the 2D LIDAR into a 3D scanner.

SAE

Our project also involves physical machining and will be 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.