WIDE AREA DRONE SITUATIONAL AWARENESS

By

Nathan Donovan & Team 18061

A Thesis Submitted to The Honors College

In Partial Fulfillment of the Bachelors degree
With Honors in

Mechanical Engineering

THE UNIVERSITY OF ARIZONA

M A Y 2 0 1 9

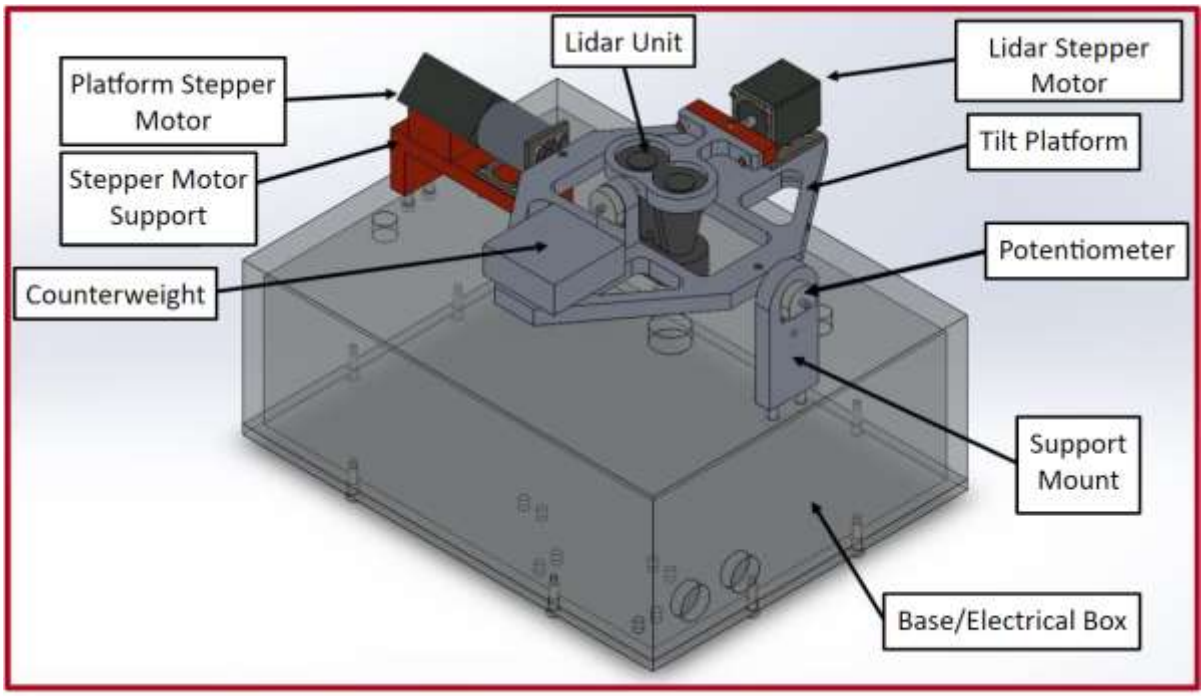
Approved by:

Dr. Gary Redford
Aerospace and Mechanical Engineering

Abstract:

The paper Wide Area Drone Situational Awareness gives an in-depth walk through the Eyes in the Dark (EitD) drone detection system. The EitD system was designed to scan a portion of the sky for the presence of a drone. If the system detects a drone, EitD then uses its ability to alert a user to the detected object. Raytheon created the project to help secure small private airports from the ever-growing threat of small hobby drones. The EitD prototype satisfies all functional requirements except for the 400m distance requirement, which was not obtainable within the budget for the project. Both physical testing and theoretical analysis is presented to demonstrate the systems capabilities and areas of future work.

Final Report:
Eyes in the Dark System



By: Sarah Rimsza, Mark Sackett, Logan Knott, Anthony Smith, Nick Busker, Nate Donovan

Table of Contents

1.0 Scope

This report is for the Eyes in the Dark System (EitD) sponsored by Raytheon Missile Systems (RMS). EitD is a low cost, portable detection system to detect the presence of drones. The increase in the availability and popularity of hobby unmanned aircraft has become a dangerous security issue for small private airports and concert arenas.

The EitD drone-detection system detects unmanned aircraft within 35 seconds and notifies system operators via PC so they can act to avoid the unmanned aircraft. The system displays the location of unmanned aircraft by giving the height and azimuth bearing of the unmanned aircraft relative to the system. The system covers a full 30-degree field of regard at up to 400 meters.The EitD system uses a lidar rangefinder as the beam source and distance-measurement device for unmanned aircraft detection. Two motors scan the detecting beam across a 30-degree field of regard. The system sits atop an electrical box housing the microcontroller and support circuit used to power and steer the motors, as well as collect recordings from the lidar. The position and readings are output from the microcontroller to the user's PC for verification and display on the graphical user interface. The software package is written in C++ and inputted into a Matlab script which generates a user-friendly display and interface for field use.

2.0 System Block Diagram

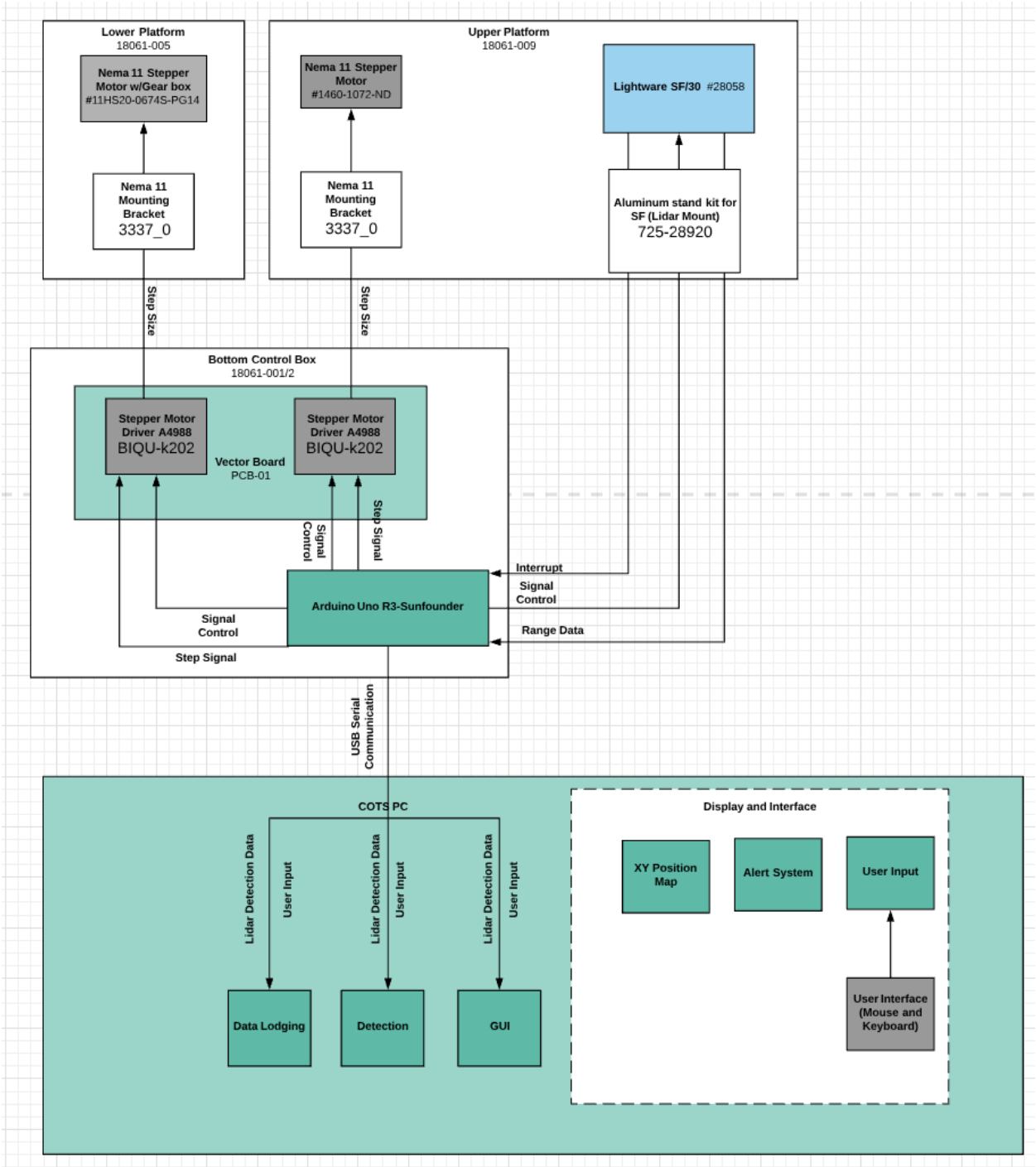


Figure 1. EitD System Block Diagram

As shown above in Figure 1. The EitD System Block Diagram contains four main blocks. These blocks are the Upper Platform which uses the Nema 11 motor to rotate the Lidar, the Lower Platform which uses a Nema 11 motor with gearbox to rotate the entire upper platform, the

Bottom Control box houses the vector board, microcontroller and power supply, the final large green box represents our connection to a COTS PC which is our user interface.

3.0 Technical Data Package (TDP)

3.1 System Requirements

3.1.1 Interface Requirements

- 3.1.1.1 System Interface: The EitD system shall interface with a COTS computing device.
  - 3.1.1.1.1 Detection Notification: The EitD system shall notify a user when a retroreflective surface is detected.
  - 3.1.1.1.2 Detection Distance: The EitD system shall display the distance to the detected retroreflective relative to the location of the system.
  - 3.1.1.1.3 Detection Bearing: The EitD system shall output the bearing of the retroreflective surface in relation to a local North, marked on the system.
- 3.1.1.2 Power Supply: The EitD system shall receive 110-120V via a standard U.S. wall outlet.

3.1.2 Mechanical Requirements

- 3.1.2.1 Scanning Subsystem: The EitD system shall scan the field of regard within 35 seconds.

3.1.3 Performance Requirements

- 3.1.3.1 Distance Performance: The EitD system should detect a retroreflective surface at a distance between 400-600 meters.
- 3.1.3.2 Field of Regard Performance: The EitD system shall detect a retroreflective surface within a field of regard up to 15 degrees.
- 3.1.3.3 Field of Regard Performance: The EitD system shall determine the retroreflective surface's Az/El bearing within ±2 degrees of its true position.

3.1.4 Customer Constraints

- 3.1.4.1 Cost Constraint: The project cost shall remain under $4000.00.
- 3.1.4.2 Portable Constraint: The EitD system shall weigh no more than 15 pounds.
- 3.1.4.3 Operating Constraint: The EitD system shall be able to operate for 8 hours in a lab environment.

3.2 System Architecture

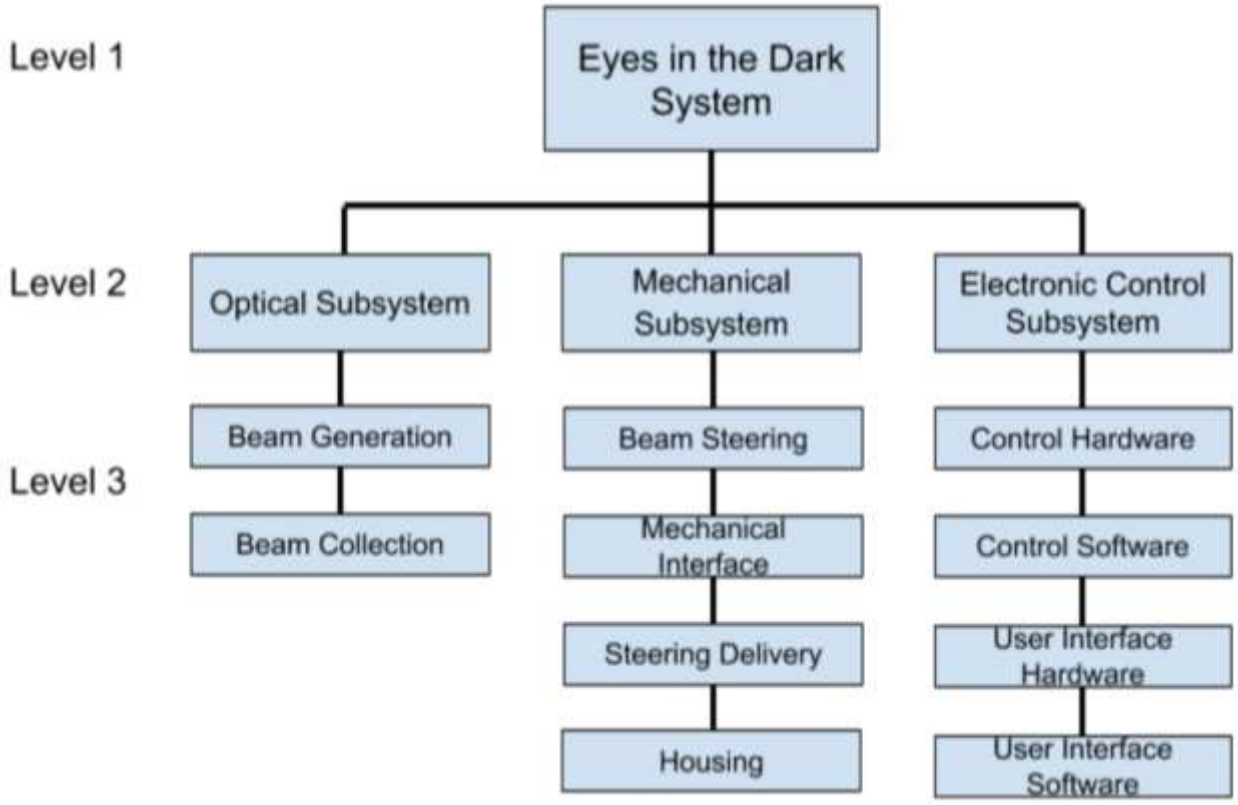


Figure 2. EitD System Architecture

The EitD system was split into three subassemblies represented in the system Architecture. The three subsystems are the Optical Subsystem, Mechanical Subsystem, and Electronic Control Subsystem.

3.3 System Requirements Flowdown Matrix

Table 1: EitD Tracebility Test Matrix							
		Sub-Assembly					
System Requirement	V M	Optical Sub-System	V M	Mechanical Sub-System	V M	Electronic Control Sub-System	V M
3.1.1.1 System Interface:	D					1.0 Direct Flow	D
3.1.1.1.1 Detection Notification:	D					2.0 Direct Flow	D
3.1.1.1.2 Detection Distance:	D					3.0 Direct Flow	D

3.1.1.1.3 Detection Bearing:	D					4.0 Direct Flow	D
3.1.1.2 Power Supply:	I	1.0 Allocated, 5V	I	1.0 Allocated, 60V	I	5.0 Allocated, 12V	I
3.1.2.1 Scanning Subsystem:	T A			2.0 Direct Flow	A T		
3.1.3.1 Distance Performance:	T A	2.0 Direct Flow	A T				
3.1.3.2 Accuracy Performance:	T A			3.0 Derived, scanning step size no greater than 2 degrees.	A T		
3.1.3.3 Field of Regard Performance:	T A			4.0 Derived, scan a field of regard of at most 15 degrees.	A T		
3.1.4.1 Cost Constraint:	A	4.0 Allocated, $1500	A	5.0 Allocated, $1000	A	6.0 Allocated, $1000	A
3.1.4.2 Portability Constraint:	D A	5.0 Allocated, 7lbs	D A	6.0 Allocated, 4lbs	D A	7.0 Allocated, 4lbs	D A

Verification Definitions

Test - validated by evaluating or executing an item under controlled conditions, configurations, and inputs to observe the response as specified by the requirements, the results are quantified and analyzed

Analysis - determined by models or by interpreting results using technical or mathematical models, simulations, or analysis

Demonstration - validated by observing the item in operation

Inspection - formal examination using one or more of the five senses, simple physical manipulation, and mechanical and electrical gauging and measurement

3.4 Software Design

3.4.1 Arduino Microcontroller

The EitD system runs on the arduino platform, specifically an Arduino Mega, running modified c++, designed with both the MatLab Gui interface, and a command line interface in mind. Controller software includes a main application that can be used in either Command Line or Gui interfaces, with the primary structure implementing a split mode/state-based design. This main application includes 3 modes, with additional command line only modes, for test verification implemented in separate software branches (FOR, Bearing, Distance, etc.). The main protocol includes a setup and test mode, a directional scan mode, and the primary scanning mode. The system controls the two stepper motors used to steer the Lidar beam, as

well as take readings from the two potentiometers and Lidar, allowing for simultaneous closed loop movement of the Lidar Beam, while taking measurements.The system also employs the power of the Arduino Mega's extra serial ports, to communicate with the PC and the SF30-C Lidar at the same time. System synchronization and interfacing are all done at startup of the main protocol, with all commands and actions optimized to be as fast as possible to allow for the best scan speed possible over the systems FOR. Users control the system state and can set various test/scan parameters through one of the 3 modes in the main protocol. Once setup the user can start the scan mode and the system will continue to run autonomously until instructed to stop. The closed loop nature of the system will allow it to correct itself in the event an error propagates over time, effectively allowing the system to run indefinitely once set up by the user, while allowing interfacing at any time.

3.4.2 MatLab GUI

The EitD Graphical User Interface is developed in MATLAB's App Designer GUI building environment. App Designer can export GUIs as standalone executables circumventing the need for current versions of MATLAB on the client PC. It has a general Workflow of:

{Show flowchart for app workflow}

It supports [N-number] of Modes.

1. The Run Mode

   This mode interfaces with the back-end Arduino to display LRF reading to the table and plot the location on plot.

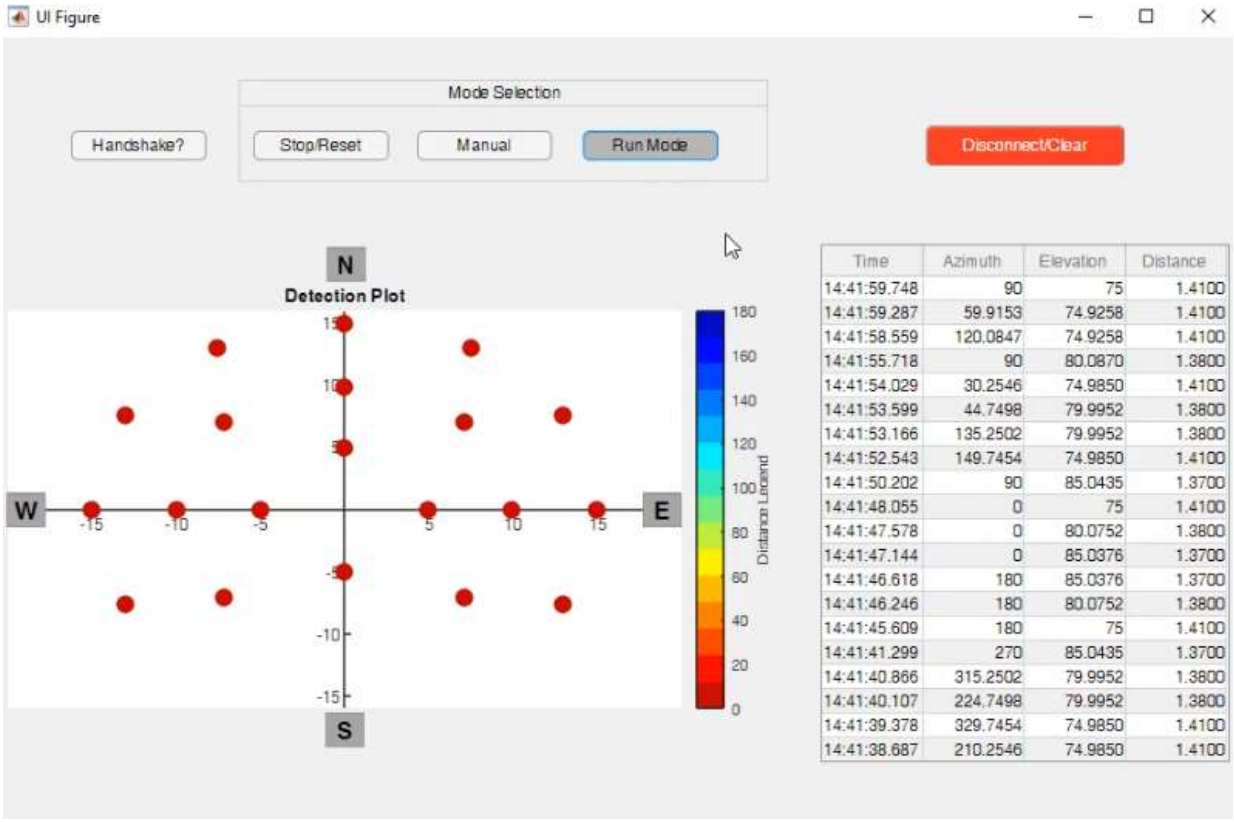


Figure 3. System GUI

3.5 Electronics Drawings

The Electrical/Wiring diagram depicts all of the electronic components attached in the electrical base the Mega board controls the system for the majority. It connects directly to the motor controllers as well as the LIDAR. From the two motor controllers two coils connect to the motors themselves. In addition to the motors, a power supply is also connected to each of the motor controllers. Finally, the Arduino itself is hooked up to a Computer via USB connection.

3.6 Part Drawings

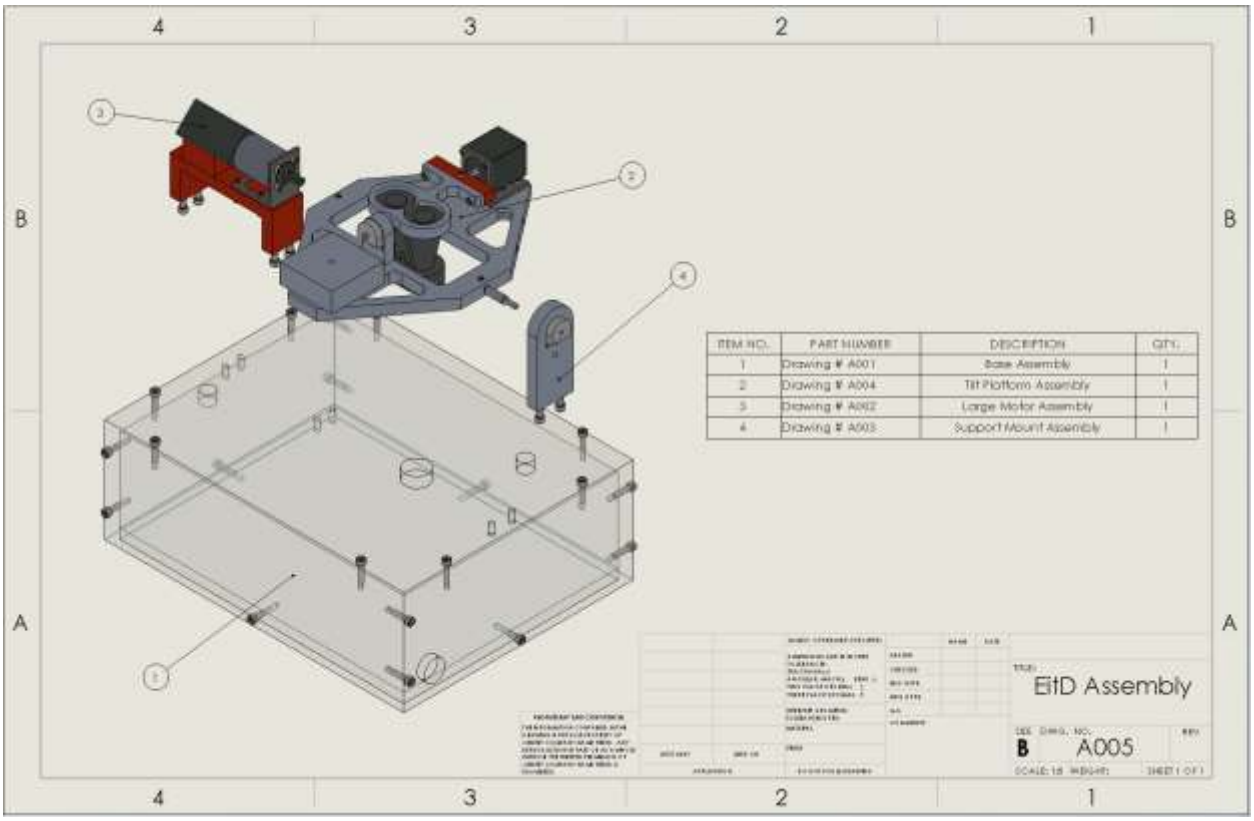


Figure 4. Drawing #A005 EitD Assembly

Figure 4 shows the complete assembly broken down into major components including the base, large support, tilt platform, and large motor assembly.

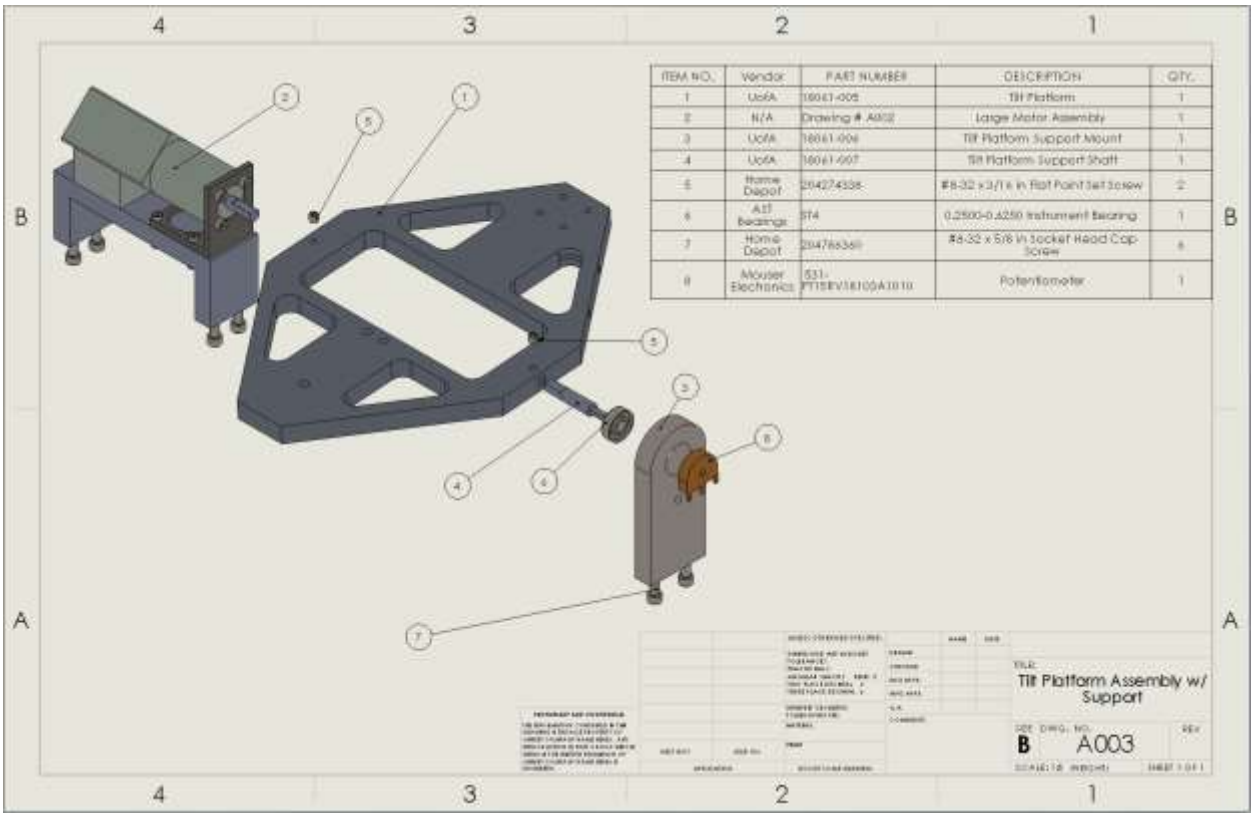


*Figure 5. Drawing #A003 Tilt Platform Assembly w/ Suppor*t

Figure 5 shows the connection of the tilt platform (part #18061-005) with the large support and large motor assembly.

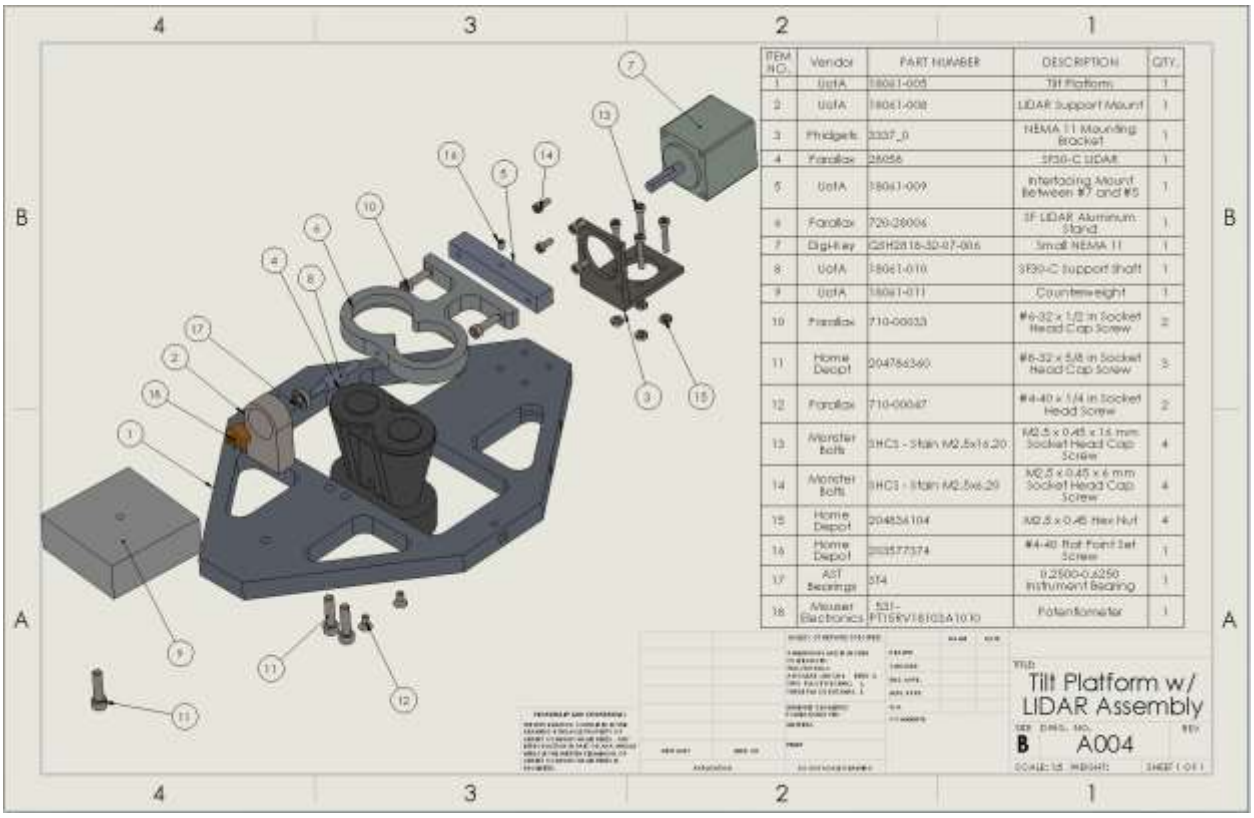


Figure 6. Drawing #A004 Tilt Platform w/ LIDAR Assembly

Figure 6 shows the placement of parts on the tilt platform (part #18061-005), including the SF30-C laser rangefinder, NEMA11 motor (QSH2818), and counterweight.

4.0 Acceptance Test Procedure

4.1 Detection Distance

4.1.1 Introduction: This procedure outlines the distance acceptance test on the EitD system for the Lidar component. This test determines the maximum distance the *lightware* SF30C Lidar can reliably detect a retroreflective surface. The Lidar will not be mounted in the system for this test, but as there are no obstructions in its scanning path when fully assembled, the performance should be the same as when installed in the system.

4.1.2 Reference Documents: EitD SRD

4.1.3 Required Test Equipment: The list of required test equipment includes

Description	Model Number	Accuracy
Computer	PC	N/A
Interface Cable	N/A	N/A
Lidar	SF30c	0.25m
Corner Cube	UA Optics (1.5”)	N/A
Tripods	UA Library	N/A
Mirror	PF05-03-G01	R = 72.5 %
Lidar Stand	720-28006	N/A
lightware screws	710-00033	N/A
ND Filters	FS-3	+/- 4%

4.1.4 Table of Tests:

Test #	Test	Requirement
1.1	Detection Distance	400 meters

4.1.5 Step-by-step Procedure:

0-400m

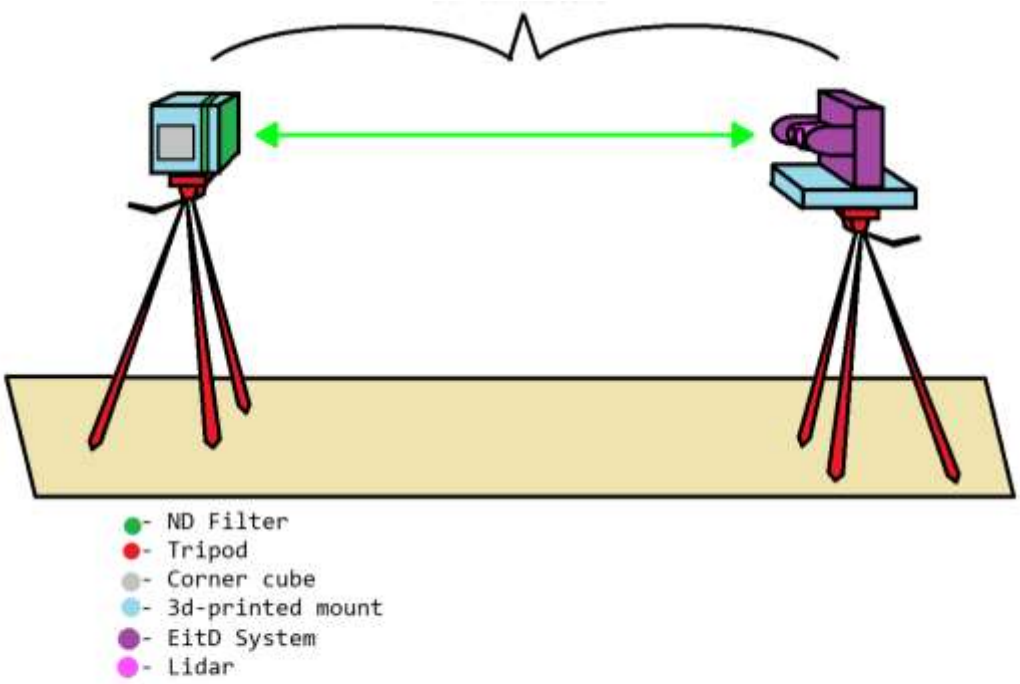


Figure 7. Distance Test Set-Up

a. Attach Lidar to its Aluminum stand with screws provided by *lightware.*
b. Secure stand to tripod so Lidar is pointed horizontal.
c. Affix corner cube to 3D printed mount.
d. On adjacent tripod attache the corner cube and mount.
e. Connect Lidar to the computer through the interface cable.
f. Task Lidar to start taking distance measurements.
g. Once Lidar is reporting the distance to corner cube, measure out 15 yards using the measuring wheel and place tripod with corner cube on it.
H. Use a laser to help align the corner cube and lidar.
i. Continue this set up to distances out to 400m in 15 yard intervals, recording the lidar measurements at each distance in the detection sheet.

4.1.6 Support Requirements: The test should be performed in the optics lab

4.2 EitD Mechanical Performance Acceptance Tests

4.2.1 Introduction: This procedure outlines the mechanical performance tests on the EitD system for FOR size and scan time. The tests verify the range of motion on the mechanical stages ensuring a 30-degree field of regard is possible and a full scan can occur in less than or equal to 35 seconds.The scanning test will be conducted under both full load and no load to resolve both extremes.

4.2.2 Reference Documents: EitD SRD and FOR + Scan time test programs
4.2.3 Required Test Equipment: The list of required test equipment includes

Description	Model Number	Accuracy
Computer	PC	N/A
Interface Cable	N/A	N/A
EitD system	N/A	N/A
Protractor	N/A	±0.5°

4.2.4 Table of Tests:

Test #	Test	Requirement
1.1	FOR Test	30 degrees
1.2	Scan Time Test	≤35 seconds

4.2.5 Step-by-step Procedure:

a. Place assembled EitD system on top of optical bench and connect to a power supply.
b. Connect system to the PC using the interface cable.
c. Start up the FOR test plan and run it.
d. Align a protractor with the large rotating stage so the reference axis of the protractor is parallel with the stage, do **NOT** move the protractor from this point on.
e. Click next on the FOR test program, then measure/record the angle of the stage on the test data sheet.
f. Click next on the FOR test program again and measure/record the new angle of the stage.
g. Click next finish on the FOR test program.
h. Open the scan time script and run it.
i. Record the output time from the command window on the test data sheet.

4.2.6 Support Requirements: Two people will be required for the test procedure

4.3 EitD Bearing Accuracy Acceptance Test

4.3.1 Introduction: This procedure outlines the acceptance test for the accuracy of the EitD

system. This test verifies the EitD system can report a detection location with ± 2 degrees. The system's FOV-FOR ratio allows the system to meet the requirement if it is calibrated. The calibration process will occur with the system fully assembled and the Lidar installed.

4.3.2 Reference Documents: EitD SRD, Scan script, and Black Cardboard backdrop

4.3.3 Required Test Equipment: The list of test equipment includes

Description	Model Number	Accuracy
Computer	PC	N/A
Interface Cable	N/A	N/A
Lidar	SF30c	0.25m
Corner Cube	UA Optics (1.5")	N/A
Cloth Backdrop	Black Card Board 5x5yds	N/A
Mirror	PF05-03-G01	R = 72.5 %
EitD System	0.13 degrees	N/A

4.3.4 Table of Tests:

Test #	Test	Requirement
1.1	Accuracy Calibration	± 2°

4.3.5 Step-by-step Procedure:

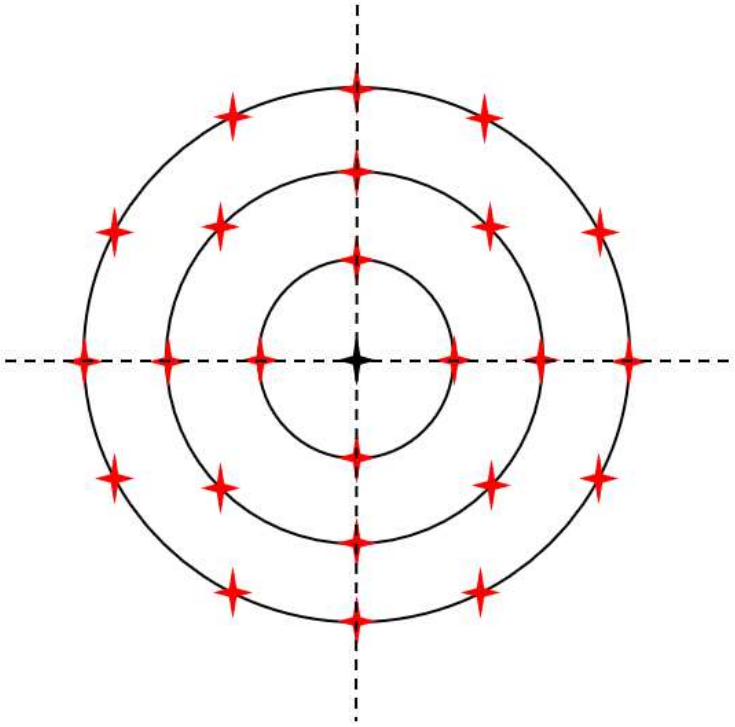


Figure 8. Poster-board Backdrop Pattern

a. Place assembled EitD system on top of optical bench and connect to a power supply.
b. Connect the system to the PC with the Interface cable.
c. Mark cloth backdrop in accordance with the cardboard backdrop pattern.
d. Attach a small velcro spot to each mark.
e. Attach the backdrop to the ceiling directly above the EitD system.
f. Place corner cube in its mount at the center position on the cloth backdrop.
g. Task lidar to take distance measurements and ensure the lidar can detect the corner cube at the EitD system's neutral position. If the lidar cannot detect the corner cube at the neutral position, adjust the position of the EitD system.
h. Bring up the scan script on the PC and press run, once the system detects the cube, confirm it is still reporting the cube at the origin. If the position is not the origin adjust the output program and then run again until system reads the corner cube at the origin.
i. Next move the corner cube to another mark on the back drop and repeat step h. for the new position.
j. Repeat step I for all the marked positions on the back drop.
k. After calibration is complete, move the cube to a location **NOT** marked on the back drop and run the EitD system.
l. Record the output position in the test data sheet.
m. Repeat steps k and l for 2 more points.
n. Verify the reported positions match the measured positions within the accuracy

specified on the test data sheet.

4.3.6 Support Requirements: Once the backdrop is put up and made taught, it should not be moved until the test is complete.

5.0 Models and Analysis

5.1 Analysis

Table 2. Analysis/ Model Summary

Name/ Reference	Requirement	Tool	Inputs	Outputs
Accuracy Performance Model Req 3.1.3.2	Plus or minus 2 degrees of true position	Excel	Scanning system step size Reflective Object Distance Reflective Surface SIze Scanning Pattern	Detection accuracy with regard to angle
Scanning Subsystem Req 3.1.2.1	<35 seconds	Written Calculations/ Excel	Distance FOR Angle Beam Divergence Motor Min Step Angle Motor RPM Line Transition Time	Total Time to Scan Field of Regard
Field of Regard Performance Req 3.1.3.3	Total 30 Degrees	Written Calculations/ Excel	Distance Scan Time Beam Divergence Motor Min Step Angle Motor RPM Line Transition Time	Field of Regard degrees
Cost Constraint Req 3.1.4.1	< $4,000	Excel	Parts Costs Number of Parts Poster Materials Polos Tax Shipping	Total Cost Project

Portability Constraint Req 3.1.4.2	< 15 lbs	Excel	Motor Weight Motor Driver Weight Microcontroller Weight Shaft Weight Upper/Lower Platform Weight LIDAR Weight Bolts/Caps Weights Weight of all Materials	Total Weight of System
Distance Performance Req 3.1.3.1	400-600m	MATLAB/ Written Calculations	Divergence Angle Laser Power Retroreflector Area Detector Area Target Reflectance Lens Transmission Range	Power Incident on Detector

Table 2 shows the analysis summary for EitD system. It contains all requirements verified by analysis ("A" in the SRVM) for the project. Note that all requirements have been met in the analyses/ models below.

5.1.1 Accuracy Performance, Scanning Subsystem, and Field of Regard Performance Analysis

Below are the basic calculations and math behind calculating scan time given the field of regard of 30 degrees total at an accuracy of +/- 2 degrees. The analyses/model for the Accuracy Performance, Scanning Subsystem, and Field of Regard Performance have been combined in Figure 3 below. These analyses were combined due to the dependency of each requirement on the other.

Scan Time Key Assumptions:

1. Scanning area is square, I.E. scans/line = number of lines
2. 295 is max RPM, scaling factor 1 reduces this to reasonable output
3. Transition time is the mechanical time to move to the next line

Basic math behind scan time:

1. Steps per Line $(\frac{Step}{Line})$

$$\frac{\theta_{FOR}}{d\theta} = \frac{Step}{Line}$$

2. Line Scan Time $(\frac{Sec}{Line})$

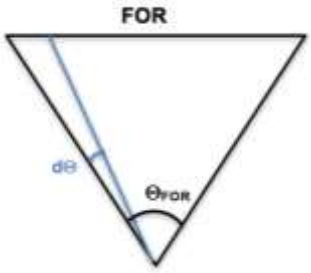
Note: Purely Dimensional Analysis

$$\frac{Sec}{Line} = \frac{Step}{Line} * \left[RPM(Scaling\ factor) * \frac{1min}{60s} * \frac{\beta Step}{Rev}\right]^{-1}$$

$$\frac{Sec}{Line} = \frac{Step}{Line} * \left(\frac{Step}{Sec}\right)^{-1}$$

$$\frac{Sec}{Line} = \frac{Sec}{Line}$$

$$\beta = \frac{360°}{d\theta}\ where\ d\theta = step\ angle$$



3. Total Time (T)

$$T = \frac{Sec}{Line} * (\#of\ Lines)(scalling\ factor\ 2) + (Transition\ Time)(\#\ of\ lines + 1)$$

Inputs		Outputs	
System Defined	Timing Estimation	Geometric System	Time and Measurements
Distance (m)	Stop-to-Scan Time Mult (float)	Motor step distance (cm)	Steps per measurement K (int)
400	2	78.54	1
FOR Angle Total (deg)	Line Transition time Added (ms)	Beam radius at Distance (cm)	System Step Angle (deg)
30	50	68.00	0.1125
Beam Divergence (mrad)	Stepper Motor Acceleration Mult	Diagonal Overlap (cm)	Steps per line
1.7	0.5	24.93	267
Motor min single step angle		Square Overlap (cm)	Line Scan Time (ms)
0.1125		57.46	33.941
Stepper Motor RPM		Diagonal Overlap Percent of beam	Total Steps
295		36.7%	71289
			Total Time Est (seconds)
			31.524

Figure 9. Scan Time, Accuracy Performance, Field of Regard Results

5.1.2 Cost Constraint Analysis

Below is the analysis for the Cost Constraint. The project must cost less that $4,000, and as shown below, the cost model shows that the project will cost approximately $2,772.43.

Figure 10. Cost Model

5.1.3 Portability Constraint Analysis

Below is the analysis for the Portability Cost Constraint. The EitD system must weigh less than 15 pounds. The figure below, Figure *10* shows the total weight of the system to be 9.36 pounds.

Part #	Description	Weight (lb)	Quantity	Total Weight (lb)
18061-001	Base	3.51	1	3.51
18061-002	Electrical Base	1.66	1	1.66
N/A	Electrical Components	1.25	1	1.25
11HS20-0674S-PG14	NEMA 11 w/ Gearbox	0.88	1	0.88
18061-005	Tilt Platform	0.71	1	0.71
N/A	Counterweight	0.35	1	0.35
QSH2818-32-07-006	NEMA 11	0.24	1	0.24
18061-006	Tilt Platform Support	0.14	1	0.14
3337_0	NEMA 11 Mounting Bracket	0.07	2	0.14
18061-003	Large Motor Mount	0.09	1	0.09
28058	SF30 Laser Rangefinder	0.08	1	0.08
204786360	#8-32 x 5/8 Socket Head Screw	0.003	25	0.075
720-28006	Aluminum Stand for SF30	0.07	1	0.07
18061-008	LIDAR Support Mount	0.03	1	0.03
SHCS-Stain M2.5x16.20	M2.5 x 0.45 x 16 Socket Head Screw	0.003	8	0.024
18061-004	Motor Support	0.02	1	0.02
SCHS-Stain M2.5x6.20	M2.5 x 0.45 x 6 Socket Head Screw	0.002	8	0.016
18061-009	Motor/Stand Interface	0.01	1	0.01
18061-007	Large Support Shaft	0.01	1	0.01
SR4	0.2500-0.6250 Instrument Bearing	0.01	2	0.02
204808021	M3.0 x 0.45 x 6 Socket Head Screw	0.002	4	0.008
204786360	#5-40 x 5/8 Socket Head Screw	0.002	4	0.008
531-PT15RV18103A1010	Potentiometer	0.003	2	0.006
710-00033	#6-32 x 1/2 Socket Head Screw	0.002	2	0.004
204836104	M2.5 x 0.45 Hex Nut	0.0004	8	0.0032
710-00047	#4-40 x 1/4 Socket Head Screw	0.001	2	0.002
204274338	#8-32 x 0.1875 Socket Set Screw	0.0008	2	0.0016
18061-010	Small Support Shaft	0.001	1	0.001
203577374	#4-40 x 0.1875 Socket Set Screw	0.0004	1	0.0004
Total				9.36

Figure 11. Portability Constraint Analysis

5.1.4 Distance Performance Analysis

The performance of the EitD system can be characterized by the following radiometric model. Because the emitted radiation is a relatively focused beam of light, the EitD system must scan across the FOR until incidence is achieved on a reflective target. Figure 6 details the ideal case where the LRF encounters an on-axis target.

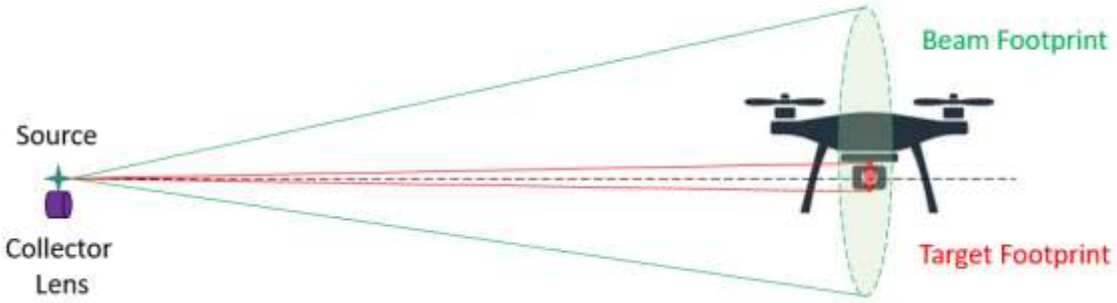


Figure 12. Drone Detection Example. Note the Beam Footprint overfills the reflective target area

The drone optic limits the amount of reflected light that can be returned, thus it is the system stop. The system can be unfolded about the target to reveal the footprint of the original source and the projected area of the target at twice the distance. The area subtended by projecting the target an additional distance is referred to as the “Lens Plane”. The ideal case for detection is when the reflected beam exactly fills the receiver. The Receiver comprises of a basic radiometer, which is a lens that is a focal length away from a detector. If all the radiation incident on the lens will focus down to the detector, the power on the lens will become the signal power. This is true if the projected target area overfills or exactly fills the lens area. Figure 7 shows the unfolded system with the target as the system stop, and the projected area of the target onto the lens plane exactly filling the lens (purple). The original “Beam Footprint” is shown as reference.

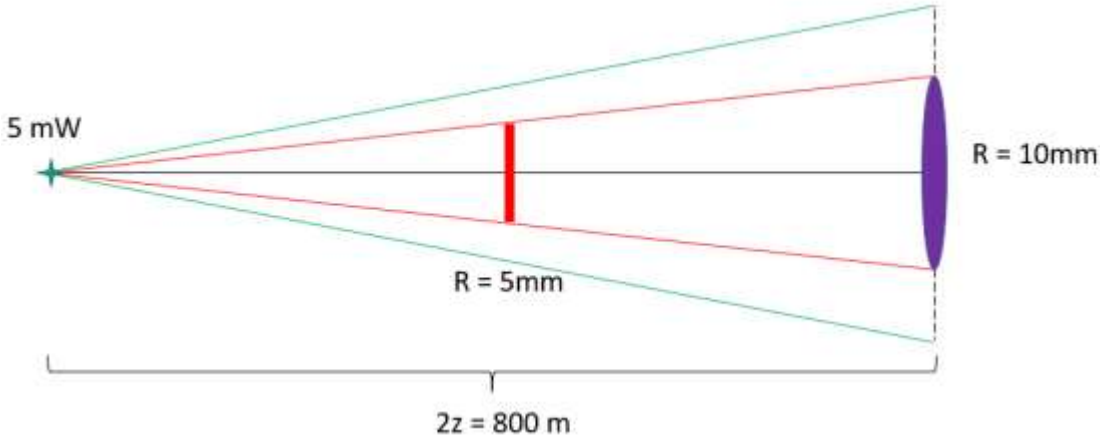


Figure 13. Lidar Distance Performance Analysis

This system is operating in the far-field, as the distance between the source and the drone is much larger than their areas. Therefore, the following equation is valid:

$$E = \frac{I}{d^2}$$

Recall that Irradiance is the ratio of power to target area so the power incident on the lens is:

$$\Phi = \frac{I_s * A_{det}}{d^2}$$

The intensity of the laser can be written as:

$$I = \frac{\Phi}{\Omega}$$

Where the solid angle is defined as:

$$\Omega = \pi * sin^2(\theta_{1/2})$$

. This yields the equation:

$$\Phi = \frac{\Phi_s * A_{fp}}{4 * d^2 * \pi * sin^2(\theta_{1/2})}$$

The lens, and the drone optic are not perfect reflectors nor transmitters, so the equation must consider the reflectivity and transmission at the operating wavelength:

$$\Phi = R * \tau * \frac{\Phi_s * A_{fp}}{4 * d^2 * \pi * sin^2(\theta_{1/2})}$$

- Transmission of lens = 0.8
- Reflectivity of target = 0.3
- Half angle = 1.2 mrad

$$\phi_{lens} = 130\ nW$$

$$I_{det} = 52\ nA$$

```
>> Radiometry_unfolded
Power at Lens Plane = 1.302084e-07 W
Signal Current = 5.208336e-08 A
```

Figure 14. Lidar Distance Performance MatLab Analysis. The Signal Current is based on an Edmund Optics Avalanche Photodiode (#58-261)

Gaussian Beam Approach to Detection

We have now successfully calculated the power incident on the lens plane given the on-axis scenario using a uniform beam. Next, we can expand the conversation to include the gaussian behavior of the output beam.

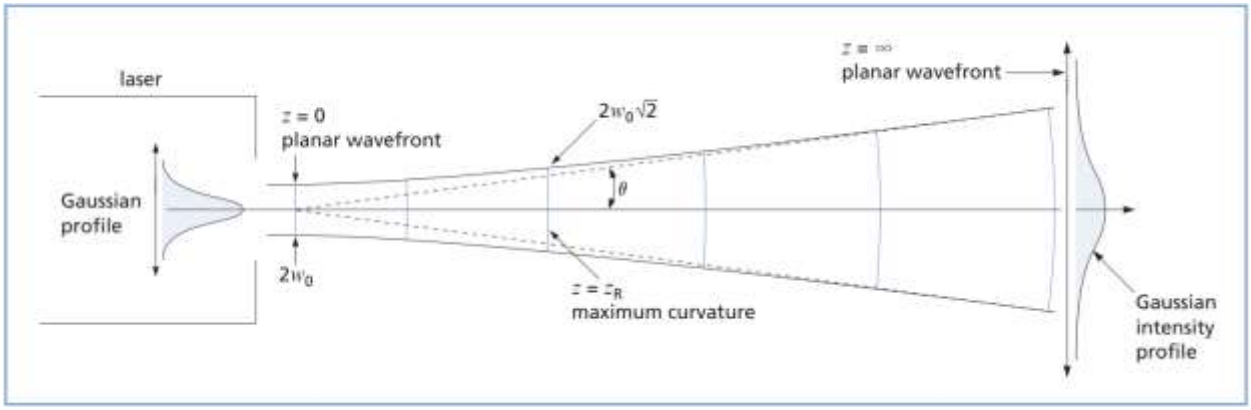


Figure 15. Far-Field Gaussian Irradiance Distribution

Melles Griot. Gaussian Beam Optics. Intensity is denoting W/m^2, this report attributes that unit to irradiance.

Because most laser operate in the TEM00 (Gaussian) mode, the non-uniform irradiance profile improves the signal received, as more power is concentrated in the center.

Recall all the equations for Gaussian beams:

- Beam Radius

$$W(z) = w_0\sqrt{1+\left(\frac{z}{z_R}\right)^2}$$

- Wave-front Radius of Curvature

$$R(z) = z[1+\left(\frac{z_R}{z}\right)^2]$$

- Rayleigh Range with respect to Beam Waist

$$z_R = \frac{n\,\pi\,w_0^2}{\lambda_0}$$

- Beam Divergence (Note this is a half angle)

$$\theta_{1/2} = \frac{\lambda_0}{n\,\pi\,w_0}$$

- Power through an Aperture

$$P(r,z) = P_0\left[1 - exp\ exp\left\{\frac{-2r^2}{w^2(z)}\right\}\right] P_0 = \frac{1}{2}\pi E_0 w_o^2 = \frac{\phi_s w_0^2}{2A_s}$$

We are interested in the power of the laser passing through the “aperture” in the unfolded system, where the aperture is the collecting lens.

The power equation, P(r,z), can be expressed in terms of beam waist:

$$P(r,z) = P_0\left[1 - exp\ exp\left\{\frac{-2\ (n\,\pi r\ w_0^2)^2}{(n\pi w_0^2)^2 + (\lambda_0 z)^2}\right\}\right]$$

And in terms of wavelength, divergence, and refractive index:

$$= \frac{\phi_s \lambda_0^2}{2\pi A_s * \left(n\theta_{1/2}\right)^2}\left[1 - exp\ exp\left\{\frac{-2\left(n\pi r\theta_{1/2}\right)^2}{\lambda_0^2 + \left(n\pi\theta_{1/2}^2 z\right)^2}\right\}\right]$$

P0 is defined to be the total power of the laser. This is the quoted specification of power. Thus, P0 may be simplified to just the quoted power specification.

The attached Spreadsheet shows calculations for four scenarios: a COTS hobby laser (DigiKey 1568-1849-ND), a THOR Labs laser diode (CPS532), and our Lidar (Lightware SF30) at average and peak powers.

Unfolded Radiometry Model				
	Laser Diode	THOR labs CPS532	SF30 Average Power	SF30 Peak Power
Inputs	Inputs			
Target				
Range [m]	400			
r_t [m]	5.00E-03			
ref_t [] reflectivity	0.3	0.3	0.348	0.348
tilt_t [rad]	0			
Lens/Detector	EO 58-261		Hamamatsu S14645-02	
r_lens [m]	1.00E-02		1.10E-02	
A_lens [m]	3.14E-04			
A_det [m^2]	7.80E-07		1.900E-07	
t_lens	0.9	0.9	0.9	0.9
A_lp (lens plane)	3.14E-04			
r_lensPlane	1.00E-02			
Responsivity [A/W]	4.00E-01	4.00E-01	0.49	0.49
Laser Inputs				
p_s [W]	5.00E-03	4.50E-03	1.80E-02	20
r_source [m]	6.00E-03	2.40E-03	2.550E-02	2.550E-02
Div Angle [rad]	2.40E-03	5.00E-04	3.491E-03	3.491E-03
lam [m]	5.32E-07	5.32E-07	9.05E-07	9.05E-07
halfAngle [rad]	1.200E-03	5.000E-04	3.491E-03	3.491E-03
Calculations				
Laser Intensity				
solidAngle_s [sr]	4.524E-06	7.85E-07	3.83E-05	
I_s [W/sr]	1.11E+03	5.73E+03	4.70E+02	
First Order Radiometry				
	Laser Diode	THOR labs CPS532	SF30 avg	SF30 peak
Outputs				
Power @ LP				
P_lp [W]	1.465E-07	7.594E-07	7.229E-08	8.033E-05
I_sig [A]	5.859E-08	3.038E-07	3.542E-08	3.936E-05
checks				
is target system stop?	TRUE	TRUE	TRUE	TRUE
Gaussian Beam Radiometry				
P01 [W]	5.000E-03	4.500E-03	1.800E-02	2.000E+01
P(basic)	1.085E-06	5.621E-06	5.586E-07	6.206E-04
P(r,z,n,lam,th,Refl(lam))	2.93E-07	1.52E-06	1.51E-07	1.68E-04
I_sig [A]	1.17E-07	6.07E-07	7.39E-08	8.21E-05

Figure 16. Radiometry Table

The first sections contain information relating to the target such as distance, physical radius, and reflection coefficients assuming a PMMA/Si detector target.

Next, the lens detector assembly is defined. Recall that the lens is sufficiently large to capture all the light “transmitted” by the target. Because this is non-imaging, a Fresnel Lens like an Edmund Optics 32-681 Fresnel Lens, can be employed. It transmits 90% in the visible to the near IR. We assume all the light is focused onto an avalanche photodiode. At the green wavelengths, an Edmund Optics 58-261 APD is chosen. The SF30 operates at 908 nm, so a Hamamatsu Si APD is chosen.

The sources named above are also defined. The Digi-key laser diode quoted the divergence angle as the full-angle, while the other sources conventionally specify them as the half angle. After quickly calculating the solid angles over which they propagate, the source intensities are found and used in the first-order calculations. The power incident on the collection lens in the ten to hundred nW scale. When considering the Gaussian properties of propagation, we see a factor of 10 improvement across the selection.

It is distinctly apparent that the lowest signal comes from the LRF at its average strength. This is due to the relatively large divergence angle. While this large value aids in mechanical scan speed of the finished EiTD unit, less of the total flux is incident on the target. It does have a slightly larger collection aperture. An LRF employed with an APD like the Hamamatsu S14645, will still have a signal that is three orders of magnitude greater than the dark noise of the detector at average emitter power. The Peak power improves the signal an additional factor of three which indicates a strong possibility of detection.

5.1.5 Operating Constraint

To meet our operating constraint, the system was tested in a lab environment to see the temperature response of critical electrical components over time. Of note, the motors had the strongest possibility of overheating. The Platform Stepper Motor with the attached gearbox required less power draw than the Lidar Stepper Motor during the use case. However, both will draw maximum current when holding a load. (Appendix 9.2) Thus, the Lidar Stepper motor would be the at-risk unit to overheat. A hand-held Forward Looking InfraRed (FLIR) Camera was affixed to a tripod and set to measure the temperature profile of both stepper motors as they actively held their respective loads. Measurements were taken every ninety seconds as the system reached thermal equilibrium. Figure 17 shows the temperature after an hour of the Lidar Stepper Motor plateaued at 119 degrees fahrenheit, and the Platform Stepper Motor plateaued at approximately 87 degrees. Additionally, the FLIR images show the temperature profile at the initial, thirty minute, and hour mark. The last image shows the interior of the system housing after 1 hour. Because Lidar Stepper Motor plateaued to 119 degrees after an hour, the team is confident of the system’s performance over an extended period of use.

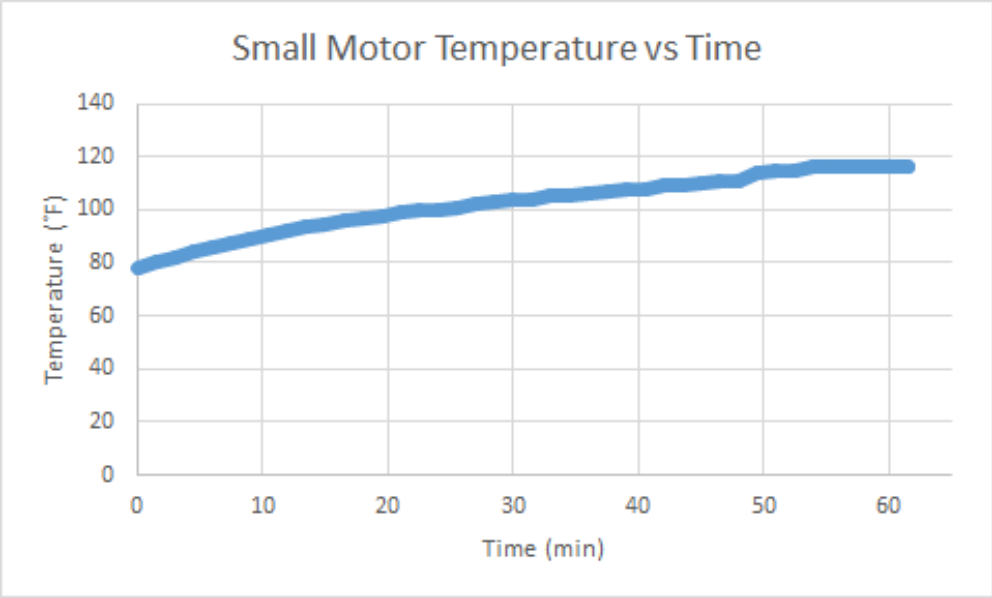


Figure 17. Small Motor Temp

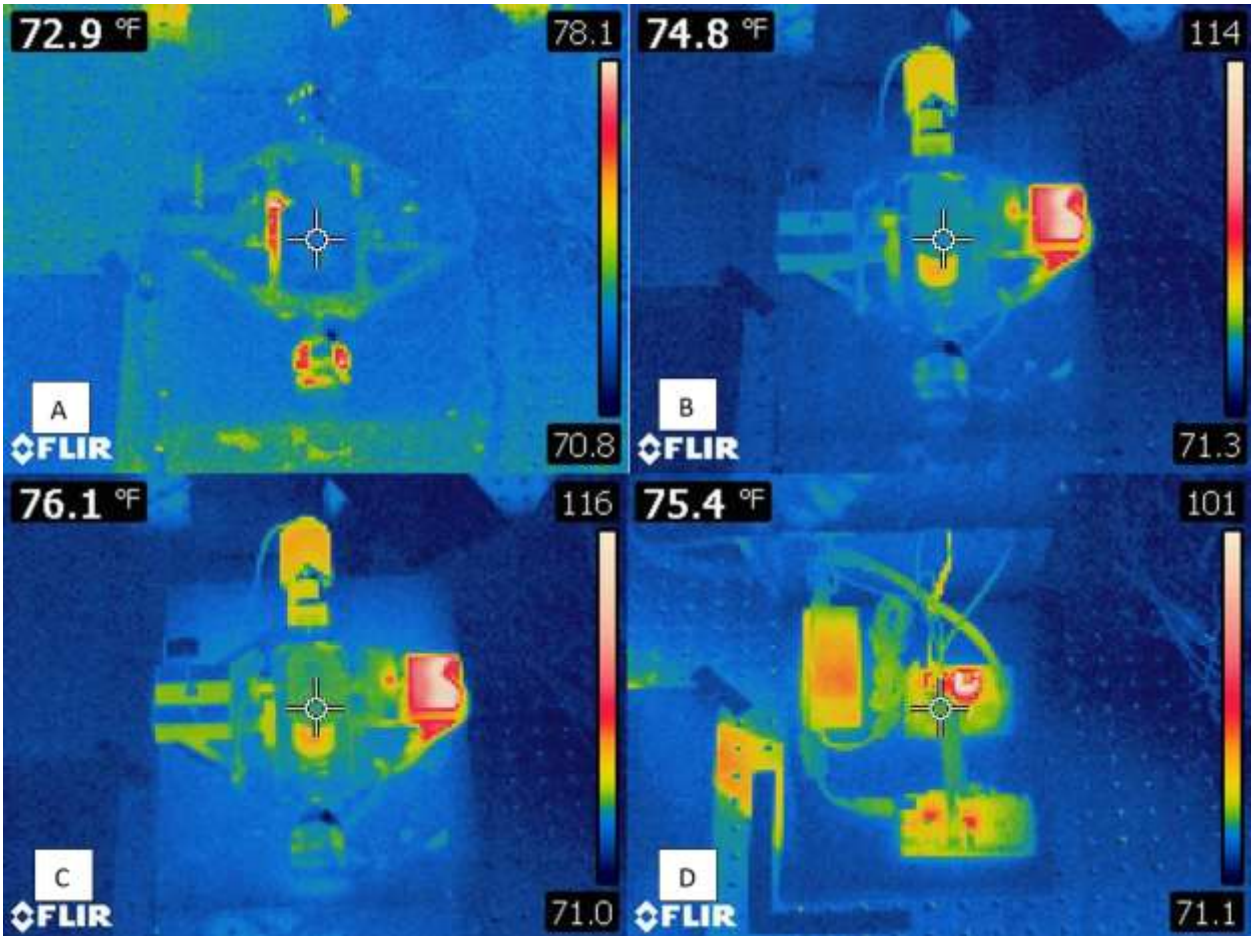


Figure 18. FLIR Image

6.0 Acceptance Test Results

6.1 Test Results

6.1.1 System Requirement Verification Table

Table 3: Verification Table				
Requirement	**Method**	**Limit/Reference**	**Measured/Ref Value**	**Pass/Fail**
5.1. Interface Requirements	-	-	-	-
5.1.1 System Interface:	D	Connects to COTS PC	Connects to COTS Device	Pass
5.1.1.1 Detection Notification:	D	Sends Notification	Notifies user	Pass
5.1.1.2 Detection Distance:	D	Output Height	Height Displayed	Pass
5.1.1.3 Detection Bearing:	D	Output Bearing in Az/El	Bearing Displayed	Pass
5.1.2 Power Supply:	I	120 V Wall Outlet	Plugs into outlet	Pass
5.2 Mechanical Requirements	-	-	-	-
5.2.1 Scanning Subsystem:	T,A	<35 Seconds	31 seconds	Pass
5.3 Performance Requirements	-	-	-	-
5.3.1 Distance Performance:	T,A	> 400 m	160 m	Fail*
5.3.2. Field of Regard Performance:	T,A	Up to 15 degrees	30 degrees FOR	Pass
5.3.3 Bearing Accuracy:	T,A	0-2 degrees	±0.229 degrees	Pass
5.4 Customer Constraints	-	-	-	-
5.4.1 Cost Constraint:	A	< $4000	$2,772.43	Pass
5.4.2 Portable Constraint:	A	<15 lbs	9.36 lbs	Pass
5.4.3 Operating Constraint	A	Lasts 8 hours	Lasts pass 8 hours	Pass

6.1.2 Scanning Subsystem

Run	Time	Start	End	Dir	Run	Time	Start	End	Dir
1	25	0	25	U	1	30	0	30	U
2	25	25	50	D	2	32	30	62	D
3	26	50	76	U	3	31	62	93	U
4	25	76	101	D	4	31	93	124	D
5	26	101	127	U	5	30	124	154	U
6	25	127	152	D	6	31	154	185	D
7	26	152	178	U	7	32	185	217	U
8	23	178	201	D	8	31	217	248	D
9	27	201	228	U	9	30	248	278	U
10	25	228	253	D	10	31	278	309	D
11	26	253	279	U		Average	30.9	0.738	
12	22	279	301	D		U Avg	30.6	StDev^	
13	29	301	330	U		D Avg	31.2		
14	24	330	354	D		Type: Full-Load			
15	25	354	379	U		Total time 5 mins 9 seconds			
16	26	379	405	D		Average 30.9 seconds a scan			
17	25	405	430	U					
18	25	430	455	D					
19	25	455	480	U					
20	26	480	506	D					
	Average	25.3	1.418						
	Up Avg	26	StDev^						
	Down Avg	24.6							
Type: No drones									
Total time 8 mins 26 seconds									
Average 25.3 seconds a scan									

Figure 19. Scanning Time Performance Results

The table above is the results from the Scanning Time Performance Test. Two distinct test data sets were collected, the no-drone with 20 measurements and the full-load with 10. The no-drone data set was created using the system running at full speed with no Target Like Objects Present, and the Full load test was taken with all data points considered TLOs. This distinction was accounted for, as data transfer between the system and the PC GUI requires a finite system slowdown and measuring both extremes gives the best representation of the system passing the requirement. Included in the test data are the average scan time's and their standard deviations, 25.3 with 1.418 for no-drones, and 30.9 with 0.738 for full-load. In both cases the average fell well with the 35 second requirement, as well as the time at 3 standard deviations. Also included were the average up and down scan times, corresponding to the direction the system platform was moving during the scan.

6.1.3 Field of Regard Performance

Mechanical Maximum Degree		
Lidar		Platform
-20° to 20°		-30° to 34°
Software FOR Test		
	Lidar (±15.07)	Platform (±15.06)
-15°	PASS	PASS
15°	PASS	PASS

Figure 20. FOR Performance Results

The table above is the results from the Scanning Time Performance Test. Two distinct tests were conducted, the first being the required Mechanical Maximum FOR test, and the second being the software steering test. The mechanical maximum test yielded results far above requirements, with at least 5 degrees extra on all axes. The software steering test reinforced this result proving the system could steer itself to the required 15 degrees.

6.1.4 Distance Performance

Lidar Distance (m)	Lidar Distance 2 (m)	Lidar Distance Avg(m)	Actual Distance (yd)**	Actual Distance (m)
14.25	13.28	13.765	15	13.71616679
27.7	27.1	27.4	30	27.43233358
41.7	40.91	41.305	45	41.14850037
54.5	54.2	54.35	60	54.86466715
68.7	67.8	68.25	75	68.58083394
83.5	81.34	82.42	90	82.29700073
96.8	95.48	96.14	105	96.01316752
110.7	108.9	109.8	120	109.7293343
123.4	122.4	122.9	135	123.4455011
137.9	136.4	137.15	150	137.1616679
151.73	150.01	150.87	165	150.8778347
165.3	160	162.65	180	164.5940015
178.9	160	169.45	195	178.3101683
191.8	160	175.9	210	192.026335
160	160	160	225	205.7425018
160	160	160	240	219.4586686
160	160	160	255	233.1748354
160	160	160	270	246.8910022
160	160	160	285	260.607169
160	160	160	300	274.3233358
160	160	160	315	288.0395026
160	160	160	330	301.7556693
160	160	160	345	315.4718361
160	160	160	360	329.1880029
160	160	160	375	342.9041697
160	160	160	390	356.6203365
160	160	160	405	370.3365033
160	160	160	420	384.0526701
160	160	160	435	397.7688369
160	160	160	450	411.4850037

Figure 21. Lidar Distance Performance Results

In the table above is the results from the Lidar Distance Performance Test. As shown in the table 30 measurements were taken between 0-411 meters. Two data sets were recorded using the Lidar and a corner cube for a target. The two data sets were then averaged and compared to the “Actual Distance” which was measured using a measuring wheel.
**Actual Distance was initially measured in yards and then converted to meters. This is due to the fact that our measuring wheel measures in yards and our Lidar measures in meters.

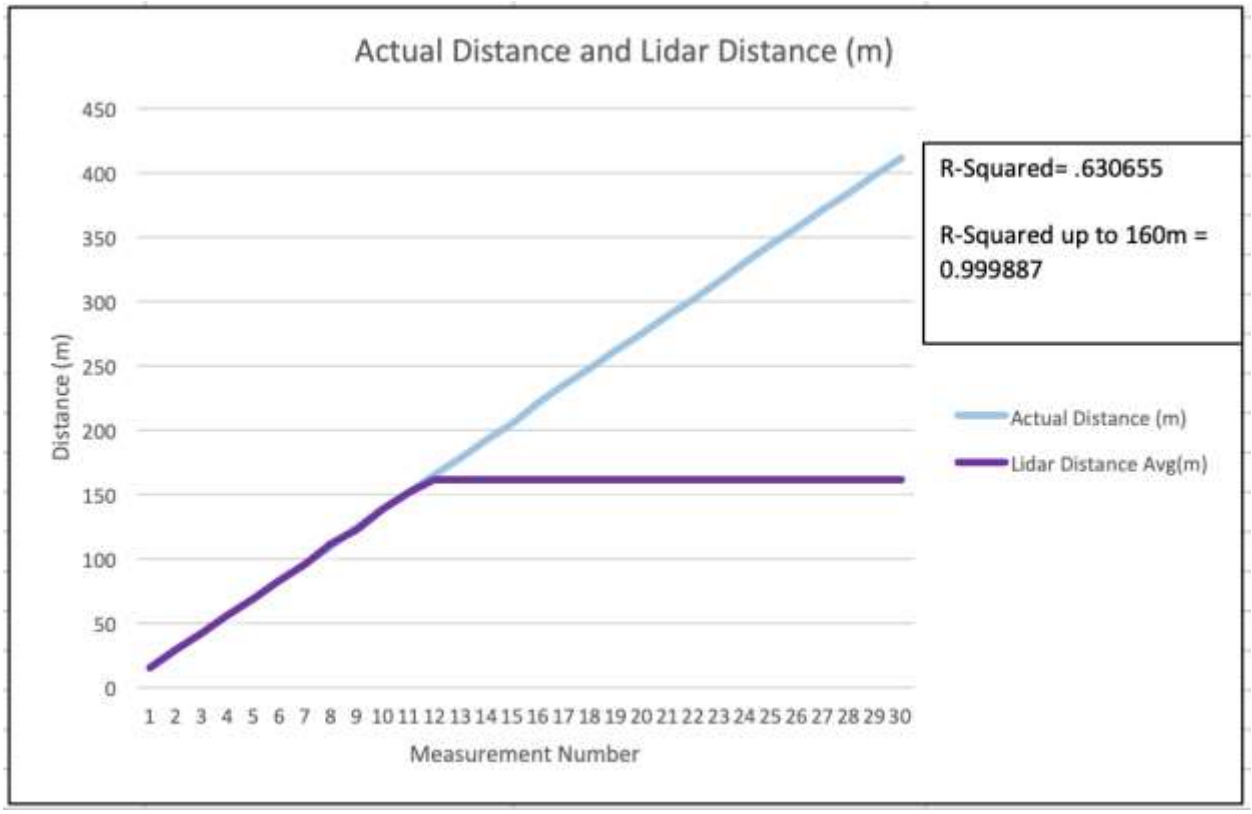


Figure 22. Lidar Distance Performance Graph

The graph above compares the actual distance measured (blue line) and the lidar distance read (purple line) for each measurement. As seen in the graph by the flat line, the lidar stops reporting correct distance measurements at 160 meters. This is due to the included firmware on the SF30-C, artificially caps this value at 160 meters, as the bit-width of the data transfer of the distance measurement allows it to report up to 164 meters at its true maximum. The calculated r-squared value is shown for the two sets of data. The r-squared value for all data is .630655. The r-squared value here is very low due to the lidars internal cap at 160m. The r-squared value for the two data sets was calculated for measurements up to 160m in order to gauge the accuracy of the lidar when compared to the actual distance. This r-squared value is 0.999887, which shows us that the lidar’s measured distances are accurate up to the 160 meter mark.

Purchasing a different Lidar Rangefinding unit would alleviate the distance performance. Acuity’s AR3000 Distance Measurement Sensor provides a 3km range for high-gain reflectors,

and a 300 meter range for other targets. The team is confident of the AR3000's performance against the distance specification of 400 meters. This unit achieves the high sampling rates required by the current system and communicates over Serial. Being heavier than the SF30-C, the system would just need a slightly stronger stepper motor. See attached Spec Sheet in Appendix 9.4, Figure 59.

6.1.5 Bearing Accuracy

X Actual (deg)	Y Actual (deg)	X (System measurement) (deg)	Y (System Measurement) (deg)	X Error (deg)	Y Error (deg)	Max Error (deg)
0	0	0	0	0	0	0
5.00	0.00	4.95	0	0.05	0	0.05
0.00	5.00	0	4.98	0	0.02	0.02
-5.00	0.00	-4.95	0	0.05	0	0.05
0.00	-5.00	0	-4.98	0	0.02	0.02
10	0	10.01	0	0.01	0	0.01
7.11	7.11	7.425	7.47	0.315	0.36	0.36
0	10	0	9.96	0	0.04	0.04
-7.11	7.11	-7.31	6.94	0.2	0.17	0.2
-10	0	-10.01	0	0.01	0	0.01
-7.11	-7.11	-6.9	-7.885	0.21	0.775	0.775
0	-10	0	-9.96	0	0.04	0.04
7.11	-7.11	7.93	-6.155	0.82	0.955	0.955
15	0	14.85	0	0.15	0	0.15
13.07	7.63	13.05	8.055	0.02	0.425	0.425
7.63	13.07	7.705	12.97	0.075	0.1	0.1
0	15	0	15.06	0	0.06	0.06
-7.63	13.07	-7.93	12.905	0.3	0.165	0.3
-13.07	7.63	-13.065	7.86	0.005	0.23	0.23
-15	0	-15.19	0	0.19	0	0.19
-13.07	-7.63	-13.67	-7.27	0.6	0.36	0.6
-7.63	-13.07	-7.485	-13.295	0.145	0.225	0.225
0	-15	0	-15.06	0	0.06	0.06
7.63	-13.07	7.595	-12.77	0.035	0.3	0.3
13.07	-7.63	12.995	-6.89	0.075	0.74	0.74

Table 4. Bearing Accuracy Results

To run the bearing accuracy test, a test board was made from posterboard with the pattern designated by figure 8.The values for the measurements were calculated with the distance between the system and ceiling set to 53.625”, while the measurements were performed with a dial caliper to ensure an accuracy of ±0.001”. A square of velcro was then attached to each mark on the board, while the corner cube mount shown in figure 8 had the opposite type of velcro attached to its base. After completion, the board was attached to the ceiling above an optical lab bench, so the system could then be placed underneath it with the corner cube attached to the center mark. After the system was verified to be in its neutral position, the center of the corner cube was found by moving the entire system. Upon finding the center position, the electrical box was taped down to ensure it did not move. The corner cube was then manually moved throughout all possible positions, stopping at each point to allow the system to find the cube and report its position. Table 4 above displays the results for each position on the measurement board. The average deviation was 0.17° with the maximum error reaching a value of 0.95°, both of which are well within the required ±2°.

6.2 Data Sheets

<table>
<tr><th colspan="2">6.2.1 EitD Detection Distance Data Sheet</th></tr>
<tr><td colspan="2">Referenced: CDR Section 3.1.3.1</td></tr>
<tr><td colspan="2">Analysis referenced (for verification by test/analysis): FR 5.1.4</td></tr>
<tr><td colspan="2">Name of Test: EitD Distance Detection Test</td></tr>
<tr><td colspan="2">Unit Under Test
Name: Lidar
Model #: SF30-c
Part #: 28058
Serial #: S30-15023</td></tr>
<tr><td>Results (Pass/Fail): Fail</td><td>Date of Test: 04/03/2019</td></tr>
</table>

Recording of Test Measurement: 13.72m: 13.765m 27.43m: 27.4m 41.14m: 41.3m 54.86m: 54.35m 68.58m: 68.25m 82.30m: 82.42m 96.01m: 96.14m 109.73m: 109.8m 123.44m: 122.9m 137.16m: 137.15m 150.87m: 150.87m 164.59m: 160m 178.13m: 160m 192.03m: 160m	205.74m: 160m 219.46m: 160m 233.17m: 160m 246.89m: 160m 260.61m: 160m 274.33m: 160m 288.04m: 160m 301.75m: 160m 315.47m: 160m 329.18m: 160m 342.90m: 160m 356.62m: 160m 370.33m: 160m 384.05m: 160m 397.76m: 160m 411.49m: 160m	Requirement (SRD, with Tolerances): 400 meters Test Equipment Error: 0.25m	Adjusted Requirement: 400.25m
Computations: Conversion from Yards to Meters. 1 yd = 0 .9144 m In order to compare our measuring wheels distance with the Lidars distance we had to convert the measuring wheels measurements from yards to meters.			
Signatures		Tester: Sarah Rimsza	
		Customer:	

6.2.2 EitD Mechanical Performance Data Sheet
Referenced: CDR Requirements 3.1.2.1, 3.1.3.2
Analysis referenced (for verification by test/analysis): FR 5.1.1

Name of Test: EitD Mechanical Performance Tests			
Unit Under Test Name: Full System System Model #: N/A			
Results (Pass/Fail): Pass		Date of Tests: Scanning 4-14-2019 FOR 4-17-2019	
Recording of Test Measurement: Lidar ±θ = -20° +20° Platform ±θ = -30° +34°	Requirements (SRD, with Tolerances): Lidar ±θ = ±15° Platform ±θ = ±15°	Test Equipment Error: θ = ± 0.5° θ = ± 0.5°	Adjusted Requirement: Lidar ±θ = 15.5° Platform ± θ = 15.5°
Computations: No-Drone Test Time 1: 25 Time 11: 279 Time 2: 50 Time 12: 301 Time 3: 76 Time 13: 330 Time 4: 101 Time 14: 354 Time 5: 127 Time 15: 379 Time 6: 152 Time 16: 405 Time 7: 178 Time 17: 430 Time 8: 201 Time 18: 455 Time 9: 228 Time 19: 455 Time 10: 253 Time 20: 506 Computations: Full-Load Test Time 1: 30 Time 6: 185 Time 2: 62 Time 7: 217 Time 3: 93 Time 8: 248 Time 4: 124 Time 9: 278 Time 5: 154 Time 10:309		Total Time: 506 seconds Average Scan Time: 25.3 seconds (TT/20) Requirement: 35 seconds Total Time: 309 seconds Average Scan Time: 30.9 seconds (TT/10) Requirement: 35 seconds	

Signatures	Tester: Mark Sackett
	Customer:

6.2.3 EitD Bearing Accuracy Data Sheet			
Referenced CDR Requirement 3.1.3.3			
Analysis referenced (for verification by test/analysis): FR 5.1.1			
Name of Test: EitD Bearing Accuracy Test			
Unit Under Test Name: Full System Model #: N/A			
Results (Pass/Fail): Pass		Date of Test: 4/14/2019	
Recording of Test Measurement: Mean Error = 0.17° Max Error = 0.95°	Requirements (SRD, with Tolerances): Mean Error = ±2° Max Error = ±2°	Test Equipment Error: dθ = ±1° dθ = ±1°	Adjusted Requirement: Position 1 = ±1° Position 2 = ±1°

<table>
<tr>
<td>Computations:
Test 1: 0.05
Test 2: 0.02
Test 3: 0.00
Test 4: 0.00
Test 5: 0.00
Test 6: 0.32
Test 7: 0.04
Test 8: 0.2
Test 9: 0.01
Test 10: 0.078
Test 11: 0.0
Test 12: 0.95
Test 13: 0.15
Test 14: 0.02</td>
<td>Test 15: 0.1
Test 16: 0.0
Test 17: 0.3
Test 18: 0.005
Test 19: 0.19
Test 20: 0.6
Test 21: 0.23
Test 22: 0.06
Test 23: 0.035
Test 24: 0.075</td>
<td>*For each test point*
Measured Value(in degrees)
(x,y)
Known Value
(x0,y0) is converted to degrees
xA=atan(x0/L)
yA=atan(y0/L)

Test Result = Max[x-xo,y-y0]

A test was performed at every position on the board shown in figure 8</td>
</tr>
<tr>
<td colspan="2" rowspan="2">Signatures</td>
<td>Tester: Mark Sackett & Nathan Donovan</td>
</tr>
<tr>
<td>Customer: Raytheon</td>
</tr>
</table>

7.0 Final Budget

Figure 23. BOM

As shown above the project total was just under $3,000. This is well below the teams required maximum budget of $4,000. The system cost was also calculated and found to be $1,150.72.

8.0 Lessons Learned

Over the course of the past two semesters, the team has learned a lot about engineering design. One of the team's biggest takeaways from this course was to order parts early. Throughout the second semester the team ran into parts having long lead times or going missing from the Purchasing Office. This caused the team to be behind on testing individual parts. Another lesson learned by the team was to start building early and to prototype when possible. From prototyping the team was able to make adjustments on our motor stands since they were not the same as the spec sheets provided by the companies. On both our motor stands and our Lidar the actual unit did not match the spec sheets and so the team had to make quick adjustments and order more parts to replace them. A key lessoned learned for the team was to order duplicates of parts since the team had extra money in their budget. Two weeks out from design day we thought one of our Nema 11 had broken, but since we had purchased duplicates we were able to switch it out easily in time for Design Day. The software engineers on our team spent a lot of time working on optimizing the software to get the scan time of the system lower. From this they found that it is important to look at ways to optimize before

beginning. As the team went into the testing phase, we quickly discovered that while our testing plans look good on paper, they were not easy to recreate. The team had to adjust some of our testing plans because the procedure written was nearly impossible. From this the team learned to really think about the test and to try to make them as simple as possible.

9.0 Appendix

9.1 Software Design Documents

Table 5. EitD Controller Software Design Document
Eyes in the Dark (EitD) System Controller Software Design Document   EitD Team 18061 12 January 2019 rev (-)
Table of Contents 1.0 Scope 1.1 Identification 1.2 System Overview 1.3 Document Overview 2.0 Referenced Documents 3.0 Controller CSCI Design Decisions 4.0 Controller CSCI Architectural Design 4.1 CSCI Components 4.2 Concept of Execution 4.3 Interface Design 4.3.1 Identification and Diagrams 4.3.2 PC-Controller Serial Interface 4.3.3 Controller-Lidar Serial Interface 5.0 Controller CSCI Detailed Design 5.1 Startup_Handshake Protocol 5.2 Alignment Protocol 5.3 Absolute_Positioning Protocol 5.4 Detection_Scan Protocol 6.0 Requirements Traceability 7.0 Notes A. Appendices
1.0 Scope – 1.1 This Eyes in the Dark (EitD) rev (-) Controller Software Design Document gives the design of the software CSCI for the internal Microcontroller of the EitD System. 1.2 This document defines the control functionality modules required to operate the EitD system internally controlling the sensors and motors of the device, while

integrating with the PC software. This software began development in the fall in its preliminary architectural design phase along with the systems requirement definitions. Raytheon sponsored this software development, which will be the end acquirer and user of the software, developed by Team 18061. This software requires understanding of relevant documents including: 2.1 the datasheet for the 2.3 stepper motors, and 2.5 the datasheet for the SF-30C Lidar Rangefinder. The corresponding documents are shown in section 2.0.

1.3 This document's purpose is to fully define the requirements and functionalities required by the controller software for the development of the EitD System. There are no privacy or security considerations for this software.

2.0 Referenced Documents –

Number	Title	Revision	Source
2.3	11HS20-0674S-PG14 Stepper Motor Datasheet	N/A	Figure 41
2.4	QSH2818-32-07-006 Stepper Motor Datasheet	1.04	Figure 40
2.5	SF-30C Lidar Datasheet	8	SF-30 Laser Altimeter Manual

3.0 Controller CSCI Design Decisions –

The primary/majority of design decisions for functionality lie within the many software units this system implements, including the Startup_Handler Protocol, Alignment Protocol, Absolute Positioning Protocol, and Detection_Scan Protocol units. These units are all individual system states/modes and thus are all dependent on the systems current state for their functions to run.

The CSCI interfaces with the first input through the UART Module on the microcontroller, allowing direct communication with the second CSCI in the system, the PC 5.3.2 Matlab Code, through USB Serial communication. It takes state/mode commands from the PC as well as handshake information. The system also takes input from the SF-30C Lidar Range Finder, through a pin interrupt and a serial communication line to the microcontroller's secondary UART. The input provides detection/range data of targets from the Lidar. Additionally, two position sensors in the form of potentiometers are interpreted by the internal ADCs as inputs. These inputs are values mapped to platform positions within the system. The system has 4 outputs, including the first two being the aforementioned serial communication lines for the Lidar and PC CSCI interface. The Lidar output is startup command data, and the to-PC output is formatted detection data. The third and fourth outputs are the two control signal pairs sent to the stepper motor controllers in the form of a step and direction signal sent using digital pin interfaces. These are HI-LOW signals.

After the startup and handshake software unit passes control to the primary loop, the system will initially interpret commands sent from the PC CSCI as state/mode commands. Depending on the signal/character sent the system will enter the specified mode and send a confirmation signal. Once in a mode the inputs/outputs will be defined by the software unit, excluding the universal exit command, which ceases a mode immediately and resume the primary loop. If a command is received by the controller that is invalid or unknown, them the system will reject it and wait for another command. The primary loop automatically checks and complete actions within a 2ms response time of the command input. Additionally, while not inside one of the software units, the system will automatically reset the position of the platforms to level, interpreting the position inputs and comparing them to internally stored

“zeroed” values, if any deviation occurs from external stimulus then the position input will trigger this response. Invalid input values, or non-defined zero values will cause the inputs to be ignored. The lidar will be disabled during the handshake software unit, thus no inputs will occur during the primary loop.

The controller does not internally store data thus does not have a DBDD. Internally stored data only includes in transit buffered values for the two USARTS, and the zeroed position values if the user resets the zero positions. The data formatting for PC communication is ASCII data formatted in comma separated values of lidar low, lidar high, and time.

The system has no explicit requirements for security or privacy, the base level of system robustness in security is supported by the design. Implementing invalid input and command checks prevents the system from behaving incorrectly if the system is being tampered with. Invalid position and Lidar data is also ignored by the controller to prevent data tampering. For default safety requirements the system preserves both system integrity and safety along with user safety by ceasing activity if motor position does not change for an extended period of time (>2 seconds) after a command is sent for the motor to move to a new position.

The system’s liberal use of clearly distinct/separate software units that represent actions/modes was designed into the systems core to allow for more modes/actions to be added if needed, and to allow all of the specific software and system controls required for both the development and testing of the EitD system. An expandable feature set and a strong system for device setup greatly increases the robustness of the system, an overall system wide goal of the project.

4.0 CSCI Architectural Design –

4.1 CSCI Components

The CSCI is split in a mode/state-based structure, and as such each mode has its own protocol and functions/interactions. The system consists of one primary software unit, with 4 software units within it. The first software unit is the primary loop, which is the main code block that controls the mode selection and escaping/switching of software units after setup has completed. Setup occurs inside the Startup_Handshake protocol software unit, where initializations and connections are established and confirmed. The third software unit is the Alignment protocol, which allows the user to tell the system to move in 1 step intervals relatively for both alignment/zeroing and for exact position during tests. The fourth software unit is the Absolute_Positioning protocol, which is a mode that allows the user to tell the system to move to a specific position relative to its zero-position based on the user input. Finally, the fifth software unit is the Detection_Scan protocol, which is the protocol which instructs the system to run the final/primary scanning operation to detect targets.

The modes all share a common static connection to the primary loop, in that the various execution modes are all called and executed from the primary loop and on return, resume the primary loop. The sole exception of this is the Startup_Handshake loop that runs on system power on, which upon completion exits to the primary loop, but is never called by it, thus is not within the primary loop.

The Startup_Handshake Protocol’s purpose is to initialize the various components and software interfaces within the controller and establish 2-way communication with the PC CSCI once all components are ready. Once complete, the unit passes control on to the Primary Loop. The Primary Loop’s purpose is to control the

overall flow of the systems various modes and processes. Furthermore, it implements all basic universal logic, specifically, using potentiometer positions to determine if the motors have arrived at the correct location, closing the open loop system of the stepper motors, automatically adding additional steps to move the system to the correct position. Moreover, it adds the buffer clearing logic, that at any point in time information gathered from the sensors or user will be passed on to either the protocol (for commands), or the PC (for collected detection data) between the two serial interfaces. This means since the transfer is at the primary loop level, potential issues in a protocol will not affect the system's ability to read from and clear its input and output buffers. It implements the protocol switching and error checking/escaping so that the device will function correctly and quickly, while allowing for easy expansion and refinement of the specific protocols it runs. Additionally, it forwards all PC commands to the protocols while they are active, except for the escape signal and the other specifics detailed in section 3.0 it has redundancy and safety features as well as basic optimization features. The first sub-protocol, the Alignment protocol, functions to facilitate the setup and zeroing process of the device as well as in system testing. The controller will hold the two input motors in place, then user input commands from the pc to the controller passed from the primary loop will control the two motors. 4 commands will represent moving each motor 1 step positive or negative, allowing the user to precisely position the two motors, thus the lidars precise direction vector, wherever they would like. A fifth command will set the current motor positions to the zero position in the controller's internal memory, as to allow the platform system to be zeroed in a level position, in case the whole system placement is not on an exact level surface. The system will have enabled the Lidar in this protocol as to send back reading/detection data during this process, since this precise positioning will serve double duty allowing it to be used for debug and testing procedures. Once complete exiting this protocol will once again disable the lidar and move the system back to its zero position. The Absolute_Positioning Protocols purpose is primarily for testing purposes allowing the user to through the PC serial monitor input degree values for each of the two motors, and the system will automatically move and lock in that position from the zero position. Invalid inputs such as too large values are automatically ignored as described in the previous section. This mode will be used for a majority of the unit tests and functionality demos of the system. Additionally, this program will be used to find the optimal mechanical speed synchronized to the lidar's polling rate. It retains the same startup and end properties of the Alignment protocol detailed above. The final sub-software unit is the Detection_Scan Protocol, which fills the role of the final product software for the system. Upon entering the Detection_Scan protocol the system will begin scanning the full scope of the field of regard, using the two motors to form a progressive scan across the area. As with the other protocols the Lidar detection data will be passed back to the controller and forwarded to the PC any time a detection occurs. This data transfer is described in section 3.0. At any point the user can command the controller to exit the scanning process and exit back into the primary loop, where any remaining data will still be pushed through the PC buffer, so no data is lost. Error detection remains the same as described above.

All software included will be built on Arduino Core version 1.8.8, see section 2.0 for reference, library and core release notes.
The controller CSCI is the only software running on the microcontroller hardware and will attempt to use its full power at all times. However, as the power of the microcontroller is much greater than what is required by our system, margins are quite wide. In terms of memory the system has 32kbytes of flash and 2kbytes of SRAM. The system's code will encompass approximately 40% of the flash memory, and 30% of its SRAM which are average values for a project of this size, but depending one what the compiler deems is required by the system/what optimizations it can make, leaving a worst case scenario at an additional 25% usage, giving a total worst case of 65% and 55% for what is expected out of a software package this size. With 45% of the remaining SRAM available for dynamic storage and the system continuously clearing information out as it is sent to the PC CSCI, auxiliary memory will not be an issue. The system will be running much faster than the sample rate of the Lidar, meaning system speeds are bottlenecked at this point. But because of the built-in overhead and system built overhead, actions can still occur while waiting thus overall use of the systems total 16MHz clock speed and processing power, 60-80% usage is a safe estimate for the implementation. The system has 14 digital IO pins, 6 analog pins, and a USB connector. The system will use 7 digital IO pins, 2 analog pins, the USB connector and the default power rails. The system will use a Baud rate of 115200 for its PC and lidar serial communication lines, to quickly transfer data between the three discrete components. There are no utilization requirements or measurements for this system.
The Controller CSCI will be split into 4 software libraries:

Library Name	Required	Software Units Housed
Main.cpp	X	Startup_Handshake Protocol, Primary Loop
Alignment.cpp		Alignment Protocol
Absolute.cpp		Absolute_Positioning Protocol

4.2 Concept of Execution

The Controller CSCI includes 5 software units, and their execution flow control, data flow, dynamic sequencing, and state transitions were described in section 4.1 in full detail but will be repeated here for ease of access. The primary flow of execution control is as follows: On system startup, the startup and handshake protocol run, which sets up any needed variables and software objects needed by the other protocols, it then waits for a handshake from the PC CSCI. Upon completion of the handshake control is passed on to the Primary Loop protocol. From here dynamic sequencing occurs, where the system can pass execution control over to any of the three reaming protocols depending on user input. Depending of the command sent from the PC CSCI, the primary loop will pass partial control to one of the three protocols, but the primary loop operations will always have priority over the protocols and will run regardless of protocol activity. Alignment, Absolute, and Detection Scan

Protocol state transitions will occur on the next primary loop cycle, after the primary loop has confirmed the input command is valid and buffers are empty. As described in 3.0 exceptions within protocols cause them to return immediately and the primary loop flushes all remaining buffered data and waits for further commands from the user. Within a protocol, dynamic calculations are only run when needed by the specific protocol, with true dynamic behavior stemming exclusively from user input commands, and mechanical inconsistency in the stepper motors causing software recalculations and operation changes. Below you can see a flowchart of execution control as the user uses the CSCI along with the PC CSCI. An important note being that protocols must exit fully, and the primary loop complete all required background tasks before entering another protocol, to ensure developmental robustness.

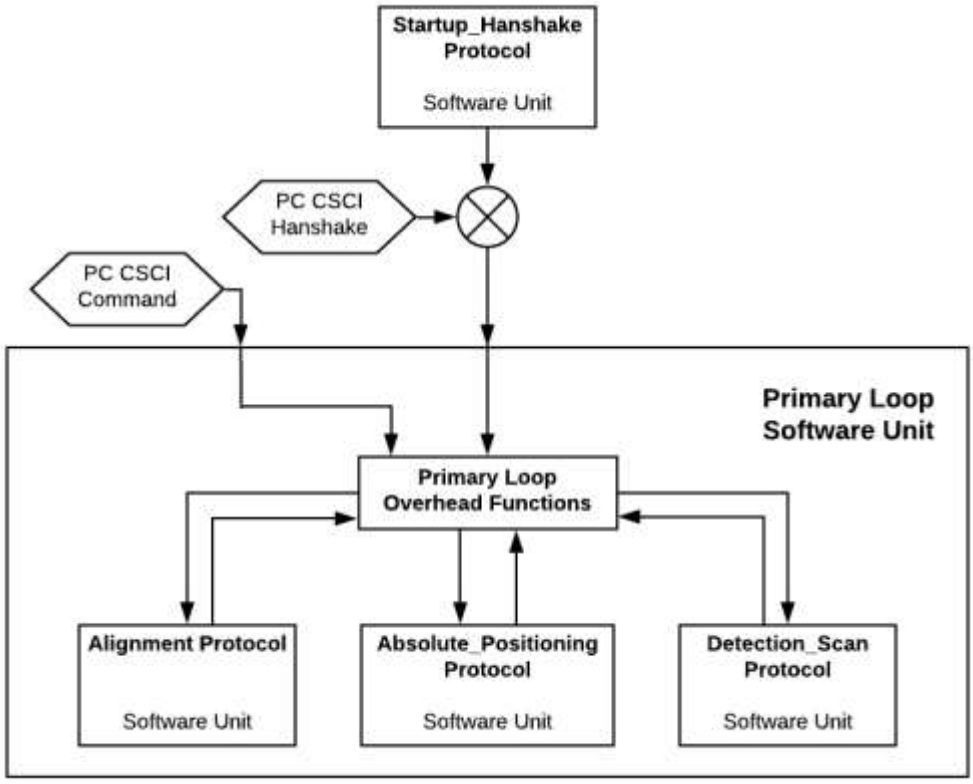


Figure 24. Controller CSCI

4.3 Interface Design

4.3.1 Identification and Diagrams

Two interfaces are present for the Controller CSCI, the first being the PC-Controller Serial interface, and the second being the Controller-Lidar Serial interface. The PC-Controller Serial interface interfaces the PC CSCI with the Controller CSCI over a USB Serial communication line, both share the same static interfacing characteristics, using the same serial, 115200 baud rate,

CR/LF terminated, USB 2.0 connection line, communication interface.

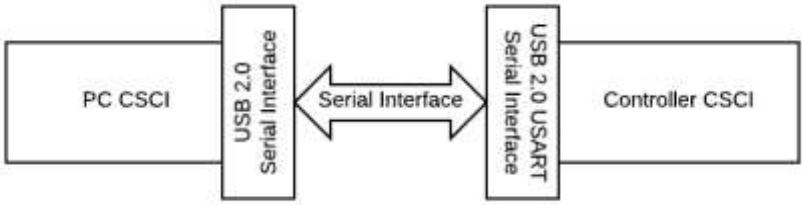


Figure 25. Lidar Serial Interface

The Controller-Lidar Serial interface interfaces the Controller CSCI with the Lidar module through a USART module. The communication occurs between the RX and TX pins of both components. Additionally, the Lidar's SYNC pin is attached to IOPin9 which has a rising interrupt watching the pin, which tells indicates the start time of a collection, part of the data package collected by the Controller CSCI and effectively paired to the serial connection as a supplement. The Lidar SF-30C component holds the static interfacing requirements, requiring a baud rate of 115200 for its required throughput to function, it also requires the serial connection as no other interface method is available on the system.

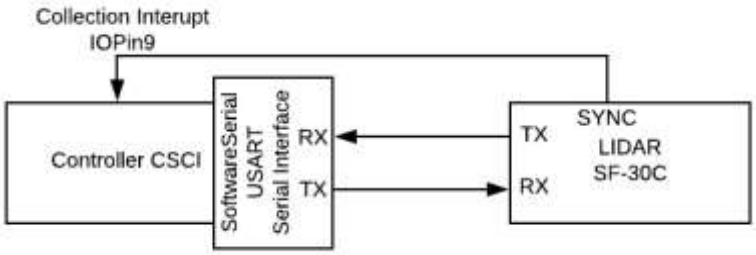


Figure 26. PC-Controller Serial Interface

4.3.2 PC-Controller Serial Interface

This interface is a real-time data transfer system with a buffer system in place, making it act as more of a storage and retrieval system. It will send from PC to Controller: PC-Commands, called Commands or User Input Commands, in the form of ASCII characters and symbols, which is 1 byte long, typically sent in a series that valid commands can vary from mode selection characters, 1-3, to numbers terminated by line feed, and numbers representing directional movement. Will be received by the Primary Loop unit and depending on system state, passed on to a protocol to be used or implemented. From Controller to PC, this interface will send a 40byte ASCII formatted character string terminated with a CR/LF character. The data in the packet will include: Distance-Float, called Distance, or Detection Distance, represented by 5 ASCII characters which represent a floating-point number for the detection distance in meters accurate to 2 decimal places. Time-Mils, called Time, or Detection Time, represented by a 16byte string of ASCII characters, that represent the time in milliseconds since system boot, that the measurement was taken. Then the next16 bytes represent the positions of the two motors at the time of measurement finally the last 2byte flag determine whether or not a detection had assuredly occurred or whether or not the data should be ignored. This

data will be received by the PC CSCI application and distributed from its buffer. This communication is over USB serial, at 115200 baud-rate, formatted in ASCII bytes, with a first in first out buffer flow control system.

4.3.3 Controller-Lidar Serial Interface

This interface is a real-time data transfer system with a virtual buffer system in place, making it act as more of a storage and retrieval system. It will send from Controller to Lidar: Measurement-Toggle, called Toggling Lidar Measurements, in the form of an 8byte string ASCII characters and symbols, in the format '#Annnnn:', where '#:' indicate start and end of commands, 'A' indicates the command, and 'nnnnn' is the value to be set. Will be received by the Lidar unit and settings will be updated. From Lidar to Controller: This interface will send a 2byte ASCII character string. The data in the packet will include: Byte_L, or Lower Byte or Detection decimal, represented by 1 ASCII character which represent an integer for the detection distance's low byte. Byte_H, or Upper Byte or Detection Integer, represented by 1 ASCII character which represent an integer for the detection distance's high byte. Sync-Int, called Sync or Sync Interrupt, or Detection Time, which is a rising and falling signal where rising indicates a measurement has just begun and is going to be sent over serial soon. This communication is over TXRX serial, at 115200 baud rate, formatted in ASCII bytes, with a first in first out buffer flow control system, with a separate warning interrupt pin.

5.0 Controller CSCI Detailed Design

5.1 Startup_Handshake Protocol

The primary design decision in this protocol was to, like its namesake, force a handshake between the PC CSCI and the Controller before exiting this protocol after all startup processes have been completed. The handshake is not required for a serial communication to be established normally, but the handshake assures that both side of the communication lines have their buffers open and are ready to proceed before continuing, increasing the robustness of system during reboots and startups in the case of issues or repairs of the system in the future. This handshake is limited to simply endlessly looping the same message from one side to the other meaning the device is filling its own send buffer and wasting power during this process and cannot perform other actions while it is occurring, this is both by design but also a technical limitation, albeit an intentional one. This unit will be programmed in C++ as all units will be. The handshake process exchanges data with the PC CSCI in the form of an ASCII formatted byte, the '~' character, by both sending and receiving said byte. Logically speaking, while waiting for the character the controller loops, once character is seen control is passed to the primary loop.

5.2 Alignment Protocol

The primary design decision in this protocol was to allow users direct control of the precise position of the two motors, as described in section 4.1 above. With this control the user can position the device in any orientation they desire, and with a press of a button zero the system to that position. This would set internal registers to now hold the current position thus dynamically adjusting calculations used in the general positioning logic for where the motors default position is as described in 4.1 and in the detection scanning process as described in 5.4. This design comes with inherent constraints in that the designer must first test the upper limits of the systems maximum

angles, then hard code these limits into the primary loops checks for invalid position requests. Additionally, these values must be considered when attempting to zero the device as past a certain threshold the systems FOR performance must be dropped to prevent system damage. This limits the protocols practicality as only an experience user can hard code these values in the case of future hardware revision, or mechanical repairs/failures. This unit will be programmed in C++ as all units will be. Logically if the user attempts to position or zero the system to a location that could harm the integrity of the device or its performance, then the inputs will be ignored. When the user wants to switch to a different mode control is passed back to the primary loop when the escape command is sent.

5.3 Absolute_Positioning Protocol

The primary design decision in this protocol was to allow users to input angle coordinates and have the system automatically position itself at that location/vector as described in 4.1. This allows moving to exact locations for testing or development purposes and is used in the calibration of the potentiometer to motor step algorithm. This algorithm takes in the Potentiometer value, as well as the sought-after angular position, and mutates the value based off of the determined value from this test. This constrains the use of this mode as it limits the exactness of any result using angular input method, as the angular detection output is only as accurate as the original calibration mutators. This unit will be programmed in C++ as all units will be. Logically if the user requests an angular position that would result in a post mutated step position or potentiometer position that is outside the defined maximum the invalid input will be ignored as described in 3.0. When the user wants to switch to a different mode control is passed back to the primary loop when the escape command is sent.

5.4 Detection_Scan Protocol

The primary design decision in this protocol was to create the most cycle efficient algorithm for scanning the device across the FOR to maximize available time to prevent the USART buffers from filling up. The chosen methodology for this protocol is as follows. The system upon control being passed to the protocol positions the two motors at their most negative position as defined by the Alignment protocol testing. The system will then employ low level control commands on digital output pins to send step commands synced to the Lidar's polling rate. Upon reaching the end of the scan line the system will iterate the additional motor 1 step, this will ensure that all space in the FOR will be thoroughly covered by the progressive pattern. This system is limited by the Arduino's ADC speeds as the position data from the potentiometer is the only source of closed loop data we have on the true position of the motors at any given moment in time. This means information on whether or not the system has fully reached the destination location will be potentially delayed. This means extra time wasted waiting at the end of scan lines, or potential backtracking to fix lost areas, but either way this delay is inherent to the potentiometer design and must be addressed. Upon completion of a FOR scan, the system will then reverse its pattern and scan the FOR again on the way back to its starting position. Throughout this operation data will be passed on from the Lidar to the PC CSCI, as described in 3.0. This unit will be programmed in C++ as all units will be. Logically, the calculations running each frame will follow a hierarchy of checks. First if the system has reached the potentiometers value for the system edge as defined in the internal memory then the secondary motor is iterated and the flag is set to start moving for the inverse position. If a scan line is

complete but the position is not correct, calculate number of extras steps and call new steps until position read correctly. If ongoing scan line is not complete and position is not correct but also stationary, escape from action back to primary loop and disable both motors, send error data to indicate mechanical issue as described in 3.0 and 4.1. When the user wants to switch to a different mode control is passed back to the primary loop when the escape command is sent.
6.0 Requirements Traceability No SDD/CSCI requirements in system
7.0 Notes N/A

Table 6. **EitD GUI/Main Software Design Document**
Eyes in the Dark (EitD) System GUI/Main Software Design Document EitD Team 18061 15 January 2019 rev (-)
Table of Contents 1.0 Scope 1.1 Identification 1.2 System Overview 1.3 Privacy Overview 2.0 Referenced Documents 3.0 GUI/Main CSCI Design Decisions 4.0 GUI/Main CSCI Architectural Design 4.1 CSCI Components 4.2 Concept of Execution 4.3 Interface Design 4.3.1 Identification and Diagrams 4.3.2 User Interface 4.3.3 Software Development Status 5.0 Controller CSCI Detailed Design 6.0 Requirements Traceability 7.0 Notes A. Appendices – Pseudocode
1.0 Scope – 1.1 This Eyes in the Dark (EitD) GUI/Main Software Design Document specifies the design of the PC portion of the Software Package of the EitD System. 1.2 This software is a portion of the overall Software Design Package that serves as deliverable to Raytheon Missile Systems, the Sponsor in

accordance to the Proposal. It has been developed by Senior Design Team 18061.
The MATLAB Portion serves as the front end of the Software Design Package and will support the GUI informing the Operator of the EitD of drone location in the form of Azimuth and Elevation (Az/ El) with respect to the EItD physical Counterweight. Many of the GUI libraries, functions, and logic are Open Source and available through the Matlab Central knowledge base. MATLAB 2018b is the supported version to run the GUI_Debug variant of the GUI.

1.3 No public user information is requested in this software. No privacy considerations are necessary for use.

2.0 Referenced Documents –

Number	Title	Revision	Date	Source
2.1	Reference for EitD SRD	(-)	XXX	

3.0 Controller CSCI Design Decisions –

Because the Arduino connects to the PC via a USB cable, the serial connection is the most appropriate data transfer method to use.
MATLAB affords an easy interface to Arduino microcontrollers through Arduino Drivers (found on mathworks.com) and provides a robust framework to execute the computations.

4.0 CSCI Architectural Design –

4.1 CSCI Components

The CSCI has the Back and Front-end portion. They will be denoted as GMB and GMF respectively. The GMB is shown below and details the high-level flow of actions while the system is turned on.

Function Name	Required	Software Units Housed
initializeGUI()	X	Start GUI and send Start Flag to Hardware
readSerial()	X	Receive the Data “frame”
convertToAzEl()	X	Parse input into final AZ/El data
updateGUI()	X	Update the GUI to the User
updateDB()	X	Store the latest data in a table

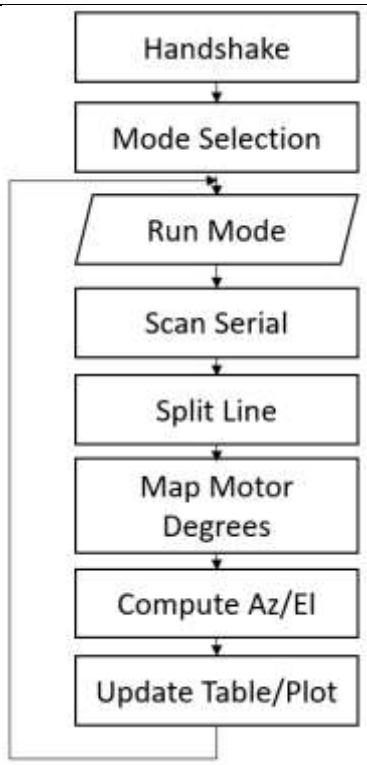


Figure 27. shows Back-End high-level sequence of events while the system on

4.2 Concept of Execution

The program should access MATLAB's serial interface, initialize the GUI, and send a command to start reading. When the start command is sent, the program will enter a while loop to continuously read the data coming in via USB. This data should then be parsed into useable data-types so that the Az/El conversion computations can take place. It is important to differentiate between different types of "hits". A quick succession of positive readings may just be the result of the drone travelling within the FOV even as the EitD system scans across the FOR. This target discrimination logic is executed in the back-end and a single target-like object data point is passed to the GUI. Finally, all "hits" should be recorded with their corresponding timestamp to serve as a history.

4.3 Interface Design

4.3.1 Identification and Diagrams

The main interface exists between the hardware and the PC via the USB cable.

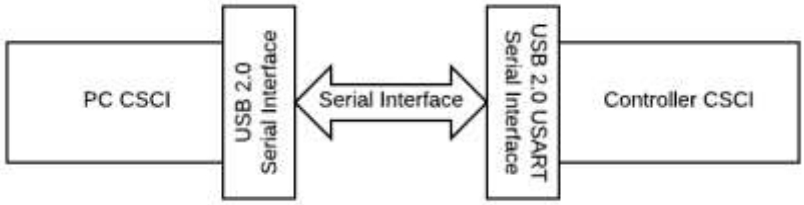


Figure 28. Shows the PC CSCI and Controller Interface. Taken from EitD Controller Software Design Document.

4.3.2 User Interface

The CSCI will inform the user of a detection via the Graphical User Interface (GUI). In addition to providing the timestamp, Az/El, and distance of a detection, the GUI should also supply a visual representation akin to a radar interface.

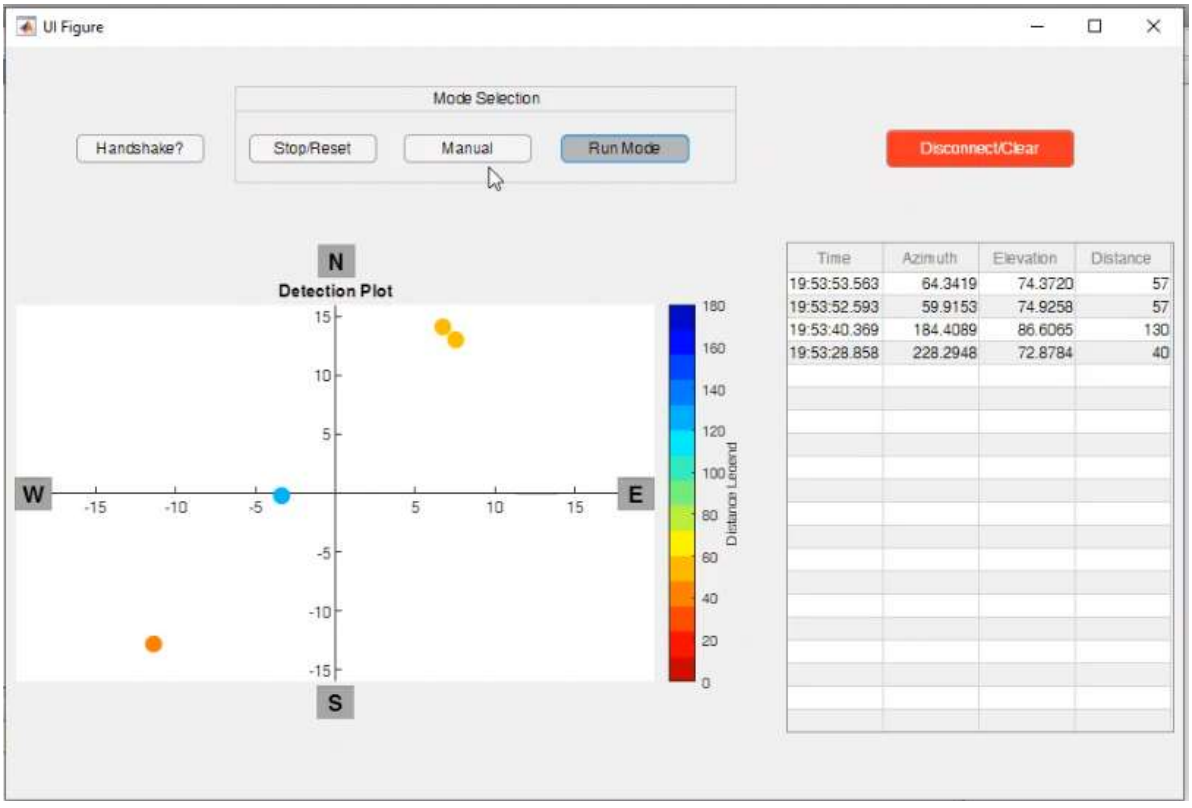


Figure 29. EitD GUI

The remainder of the interfaces are internal to the program and are just parameter calls among the functions of the program.

4.3.3 Software Developmental Status

Virtually every portion of the GMB and GMF will be developed by the team. The “Scan Serial” and convertToAzEl() portions of the GMB are

detailed in pseudocode and only need to be transcribed into the MATLAB scripting language. Furthermore, the routine will execute inside a MATLAB “app” using Appdesigner.

The GMF portion will be based on open-sourced documentation from MATLAB Central and is software to be developed.

5.0 Controller CSCI Detailed Design

The most detailed design of the GMB is presented as pseudocode in Appendix A.
After using MATLAB’s serial function to access the data stream to come from the Arduino, the Handshake Item must be the first process to execute so that the data stream has a proper place to terminate. The process will depend on a keystroke (‘g’) from the User to prepare for data streaming. The button labelled “Run Mode” initiates the data stream from the hardware. The Matlab Support Package for Arduino allows direct communication from the keyboard to the Arduino board. Keystrokes will also be used to issue commands to the controller in the case of certain control modes like the Alignment and Absolute_Positioning Protocols, for further information see the Controller CSCI SDD.

The “Scan Serial” Item is the first action in the loop and receives time, LRF distance measurement, and two motor positions. The two motor positions are required to make an Az/El conversion and the processed data will be split into distinct fields to be processed in the following function.

convertToAzEl() is the computationally heavy portion of the GMB. After receiving the appropriate data, this function discerns the location of the received signal and projects it onto one of the four quadrants of the Cartesian Coordinate System. The discreet motor positions can relate to a distance in the Cartesian Coordinate System’s ‘x’ or ‘y’ direction. Trigonometry is then used to discern the quadrant, and thus, an azimuth value. Because the motor positions are mapped to degree within +/- 15° Field of Regard with 0° being normal to the system, Elevation is just the RMS of both motor degree positions with respect to the horizon.

updateGUI() will then plot the newest data on the graph and notify the user of the detection via the new entry on the table.

updateDB() is a straightforward export of the newest data into a table to serve as a record of the last twenty detections.

6.0 Requirements Traceability

No SDD/CSCI requirements in system

7.0 Notes

The MATLAB Support Package will need to be installed in the developer's version of MATLAB to achieve successful communication via the I2C and SPI port. Please refer to Mathworks: https://www.mathworks.com/hardware-support/arduino-matlab.html

MATLAB version 2018b must be used in order to run the GUI inside MATLAB. Alternatively, a standalone executable is available.

Appendix A

MATLAB GUI MAIN

```
Serial create toArduino                 //Serial Stream Object
Set Serial Connection Settings //Port, BaudRate, Terminator etc.
Serial.open toArduino                                   //Open connection
Initialize GUI                  //Start GUI functions and display
Define CollecMutate = '18'
Define ScansPerLine = '167'
counter = 0
loop{
        Wait Until Serial Connection is Ready
        if( User Pressed a Key/Button ){   //flag from GUI handler
                Serial.send(Key)
        }

        if( Serial.Available ){
                Serial.ReadLine
                Split/Format Input Data
                convertToAzEl(New Data)  //Single scan passed in
                computeTarget(New Data, Past Data) //Past = 2 arrays
                updateGUI(Updated Data)
                Past Data += Data
                if (counter == (CollecMutate * ScansPerLine * 3)){
                        counter = 0
                }
                else{
                        counter++
                }
        }
} //End of main loop
```

Convert to AZ/EL()

```
//Replaces XY coordinates with az and el
convertToAzEL(New Data){        //pass in x, y, dist, cycle
        Convert Coordinates to Elevation
        Convert Coordinates to Azimuth
        convertedData = [Az, El, Distance, Cycle]
        return convertedData
}       //pass through dist & cycle with calculated data
```

UpdateGUI()

```
//Touches three GUI handlers to refresh screen with new data
updateGUI(Data){      //pass in current detection data
        Check Data Set for newly Addressed Targets
        Update the XY Map
        Update the Alerts
}
```

ComputeTarget()

```
computeTarget(New Data, Past Data){
  //past data includes recent and all-time detection data
        for (all past detections){
                if( New Data within 2 degrees of Past Data){
                        Detected Data(match)    avg(PastData(match), New Data)
                        return Detected Data //avg position replaces old data
                }
        }
        for (all recent values){ //not previously occurred
                if( Data within 2 degrees){
                        matched counter++  //counts recent close values
                }
        }
        if(matched counter > CollecMutate){
          //must be > the threshold to be determined a detection
                Add Data to Past Detected Data
                return Detected Data
        }
}
```

9.2 Electronic Design Drawing

9.2.1 Circuit Diagram

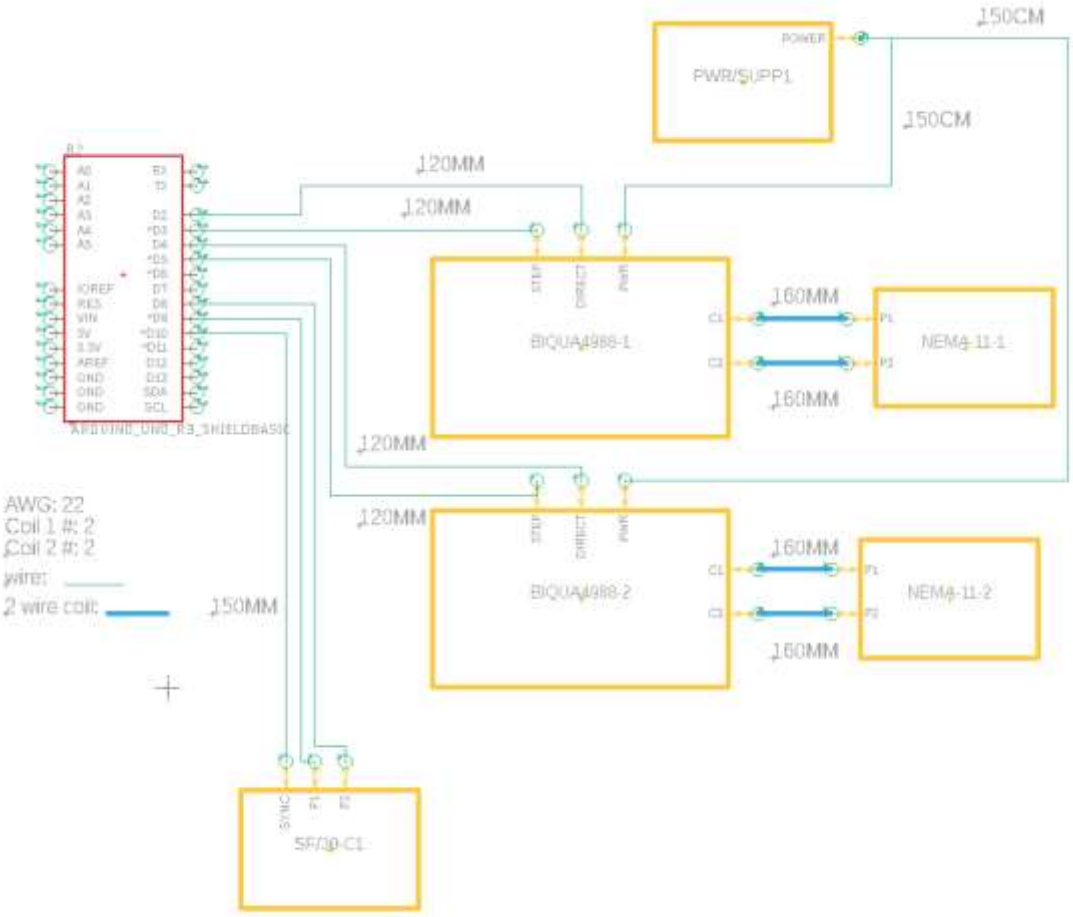


Figure 30. Electrical Circuit

Build Instructions:

- Using the holes mount the Arduino Uno to the electrical base (shown in figure 5.1.1)
    - Plastic screws provided in Arduino uno kit
- Solder the motor drivers to the vector board (be sure that the power pins line up on either driver)
- Attach the vector board using the remaining holes in the electrical base
- Wire down potentiometers down the shaft support through the hole in the top of the base and connect into the Arduino Uno
- Connect 150mm 22 AWG wire from the Lidar unit to three serial ports located on the Arduino Uno

- Connect 120mm 22 AWG wires step and direct port on the motor controller to the programmed Lidar ports
- Repeat step for both of the controllers
- From the motor controller connect two 160mm double wire coils to the corresponding motors
- Connect your 150cm power supply cable to the power ports on both of the controllers
- Connect the Arduino Uno to the computer via USB cable

9.2.2 Wire List

PIN	Connection	Description/location
A0	potentiometer	through side hole to main shaft
A1	potentiometer	through side hole to Lidar shaft
5v	motor driver slot 1	vectorboard slots
gnd	motor driver slot 2	vectorboard slots
Digital 8	motor driver slot 3	vectorboard slots
Digital 9	motor driver slot 4	vectorboard slots
Digital 10	motor driver slot 5	vectorboard slots
Digital 11	motor driver slot 6	vectorboard slots
tx 3	tx Lidar	through center hole to lidar
rx 3	rx Lidar	through center hole to lidar
Arduino Power	Arduino Power	corner of Arduino
Motor power	Motorpower	side of vectorboard

Figure 31. Wiring Table

9.3 Part Drawings

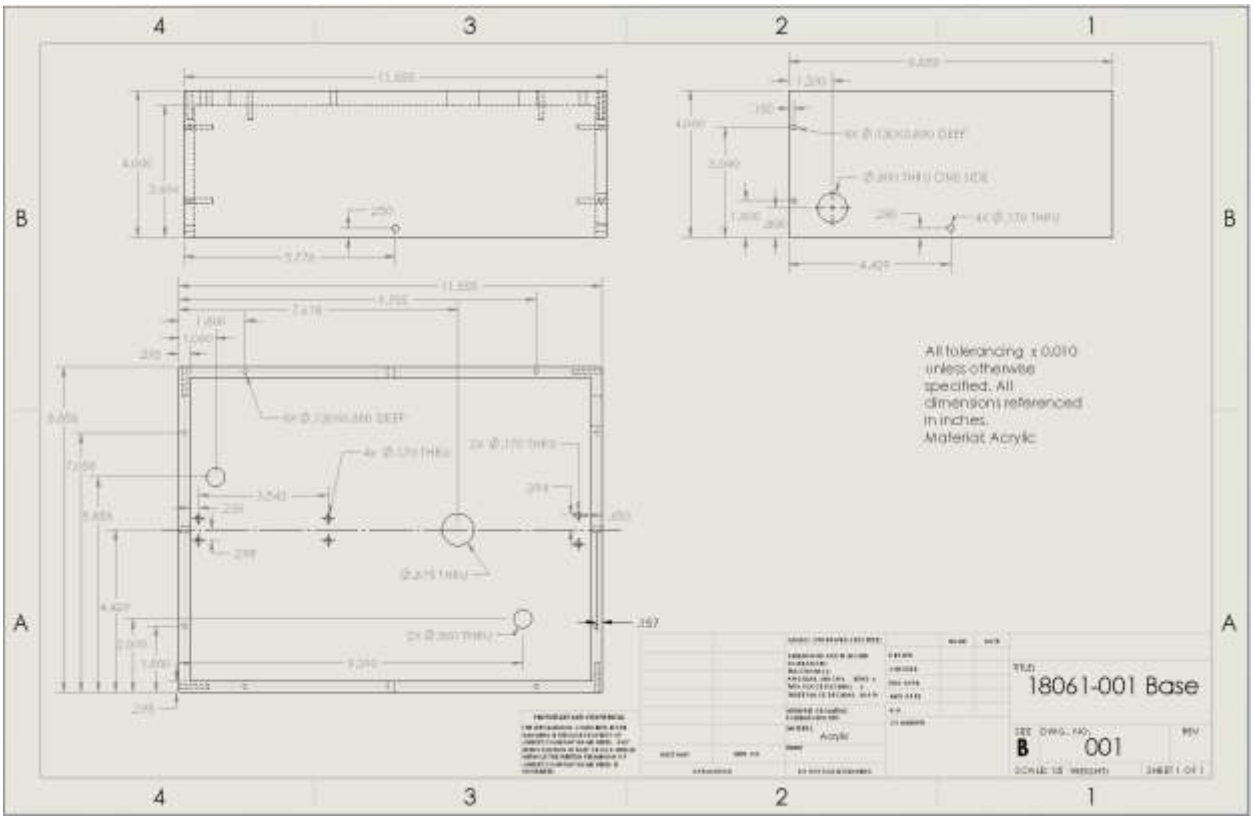


Figure 32. Drawing #001 Part 18061-001 Base

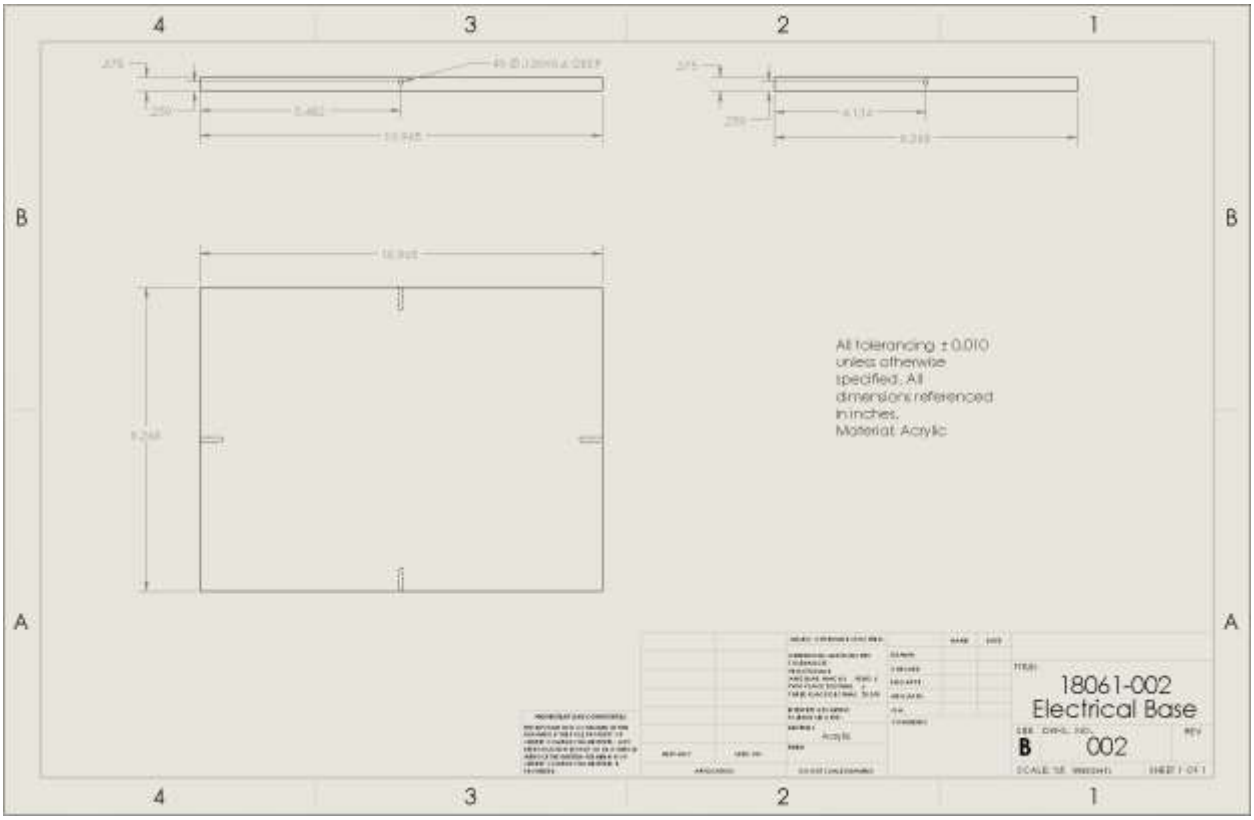


Figure 33. Drawing #002 Part 18061-002 Electrical Base

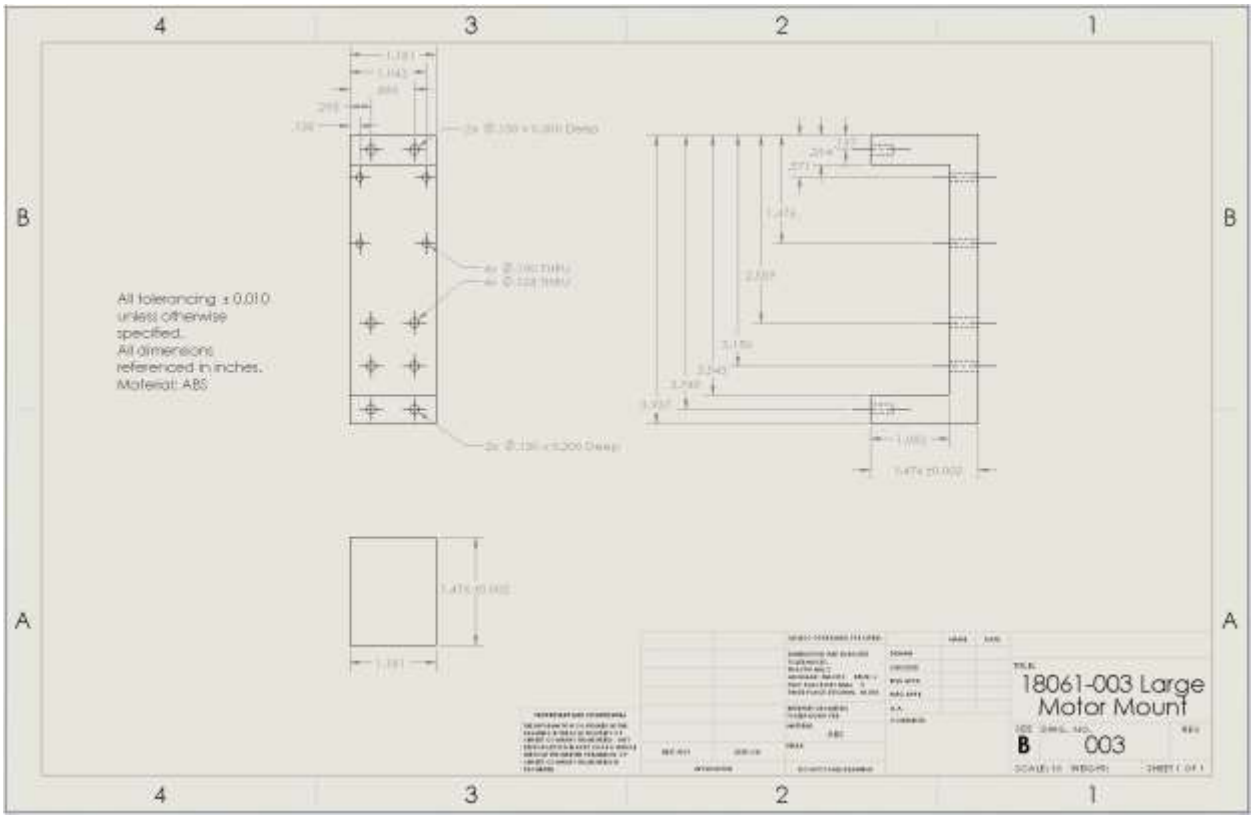


Figure 34. Drawing #003 Part 18061-003 Large Motor Mount

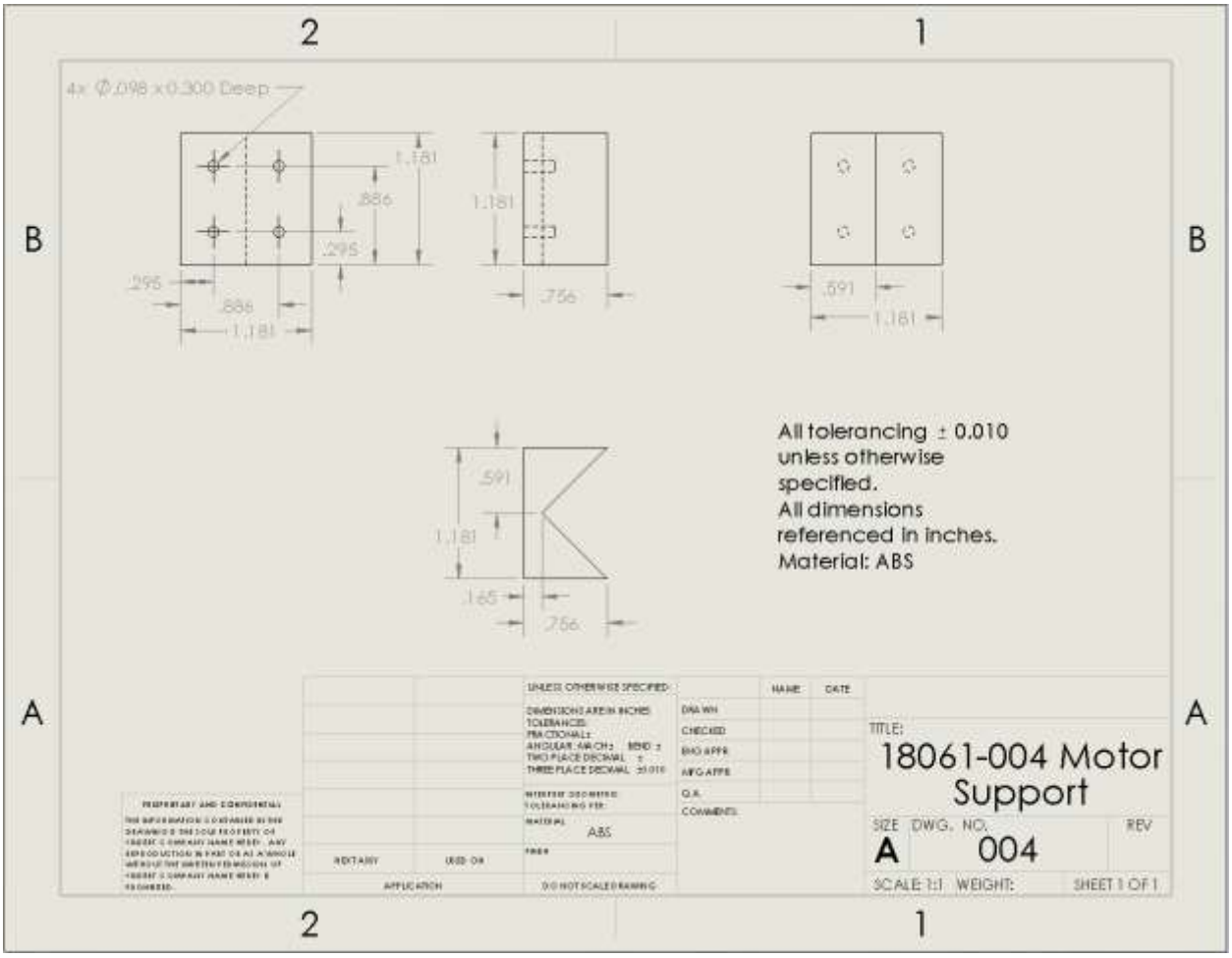


Figure 35. Drawing #004 Part 18061-004 Motor Support

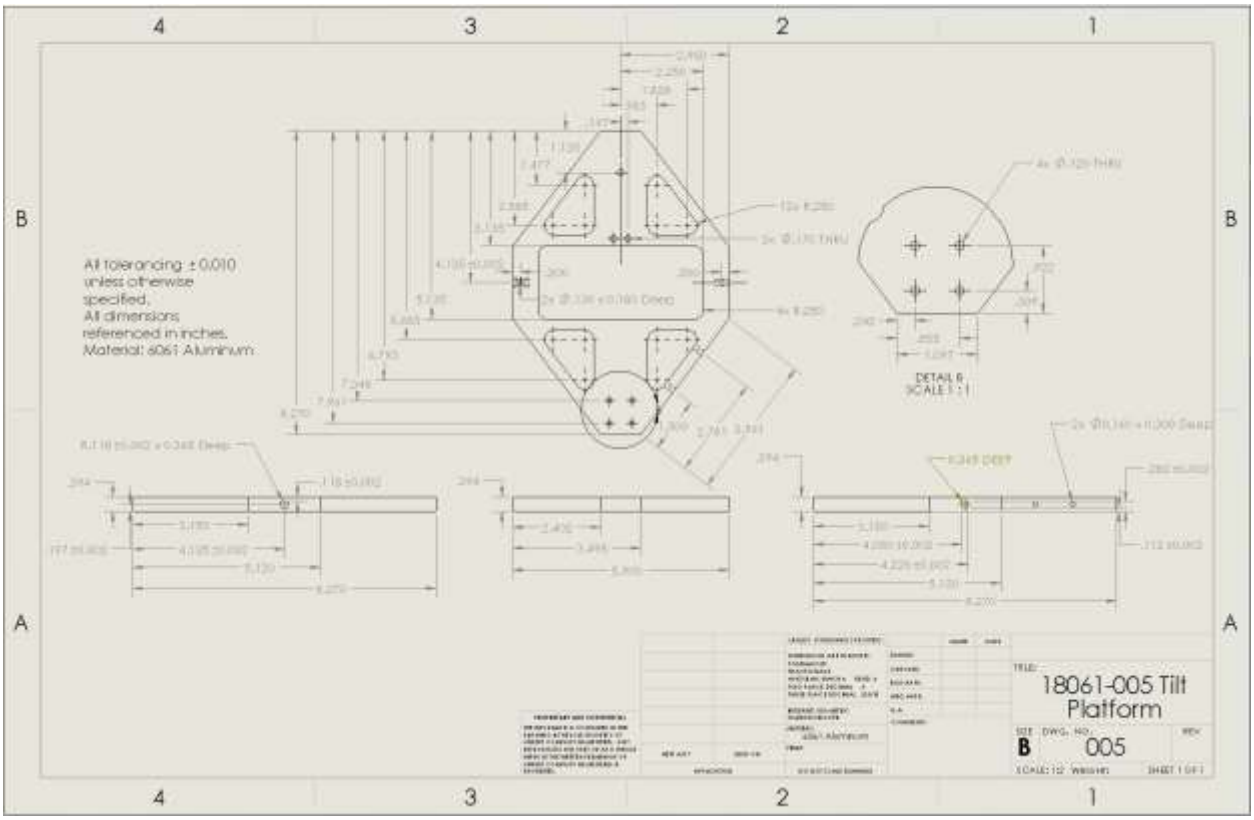


Figure 36. Drawing #005 Part 18061-005 Tilt Platform

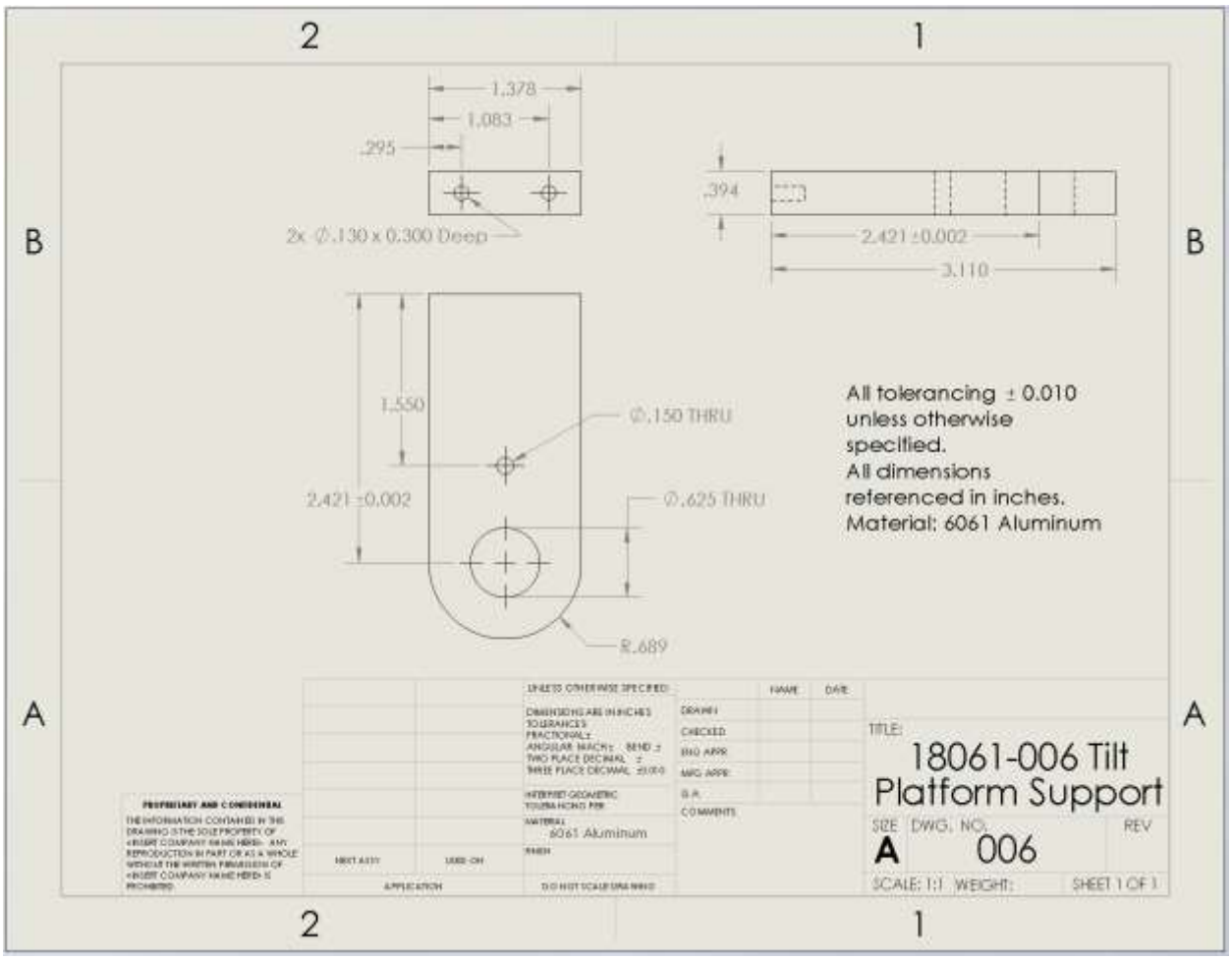


Figure 37. Drawing #006 Part 18061-006 Tilt Platform Support

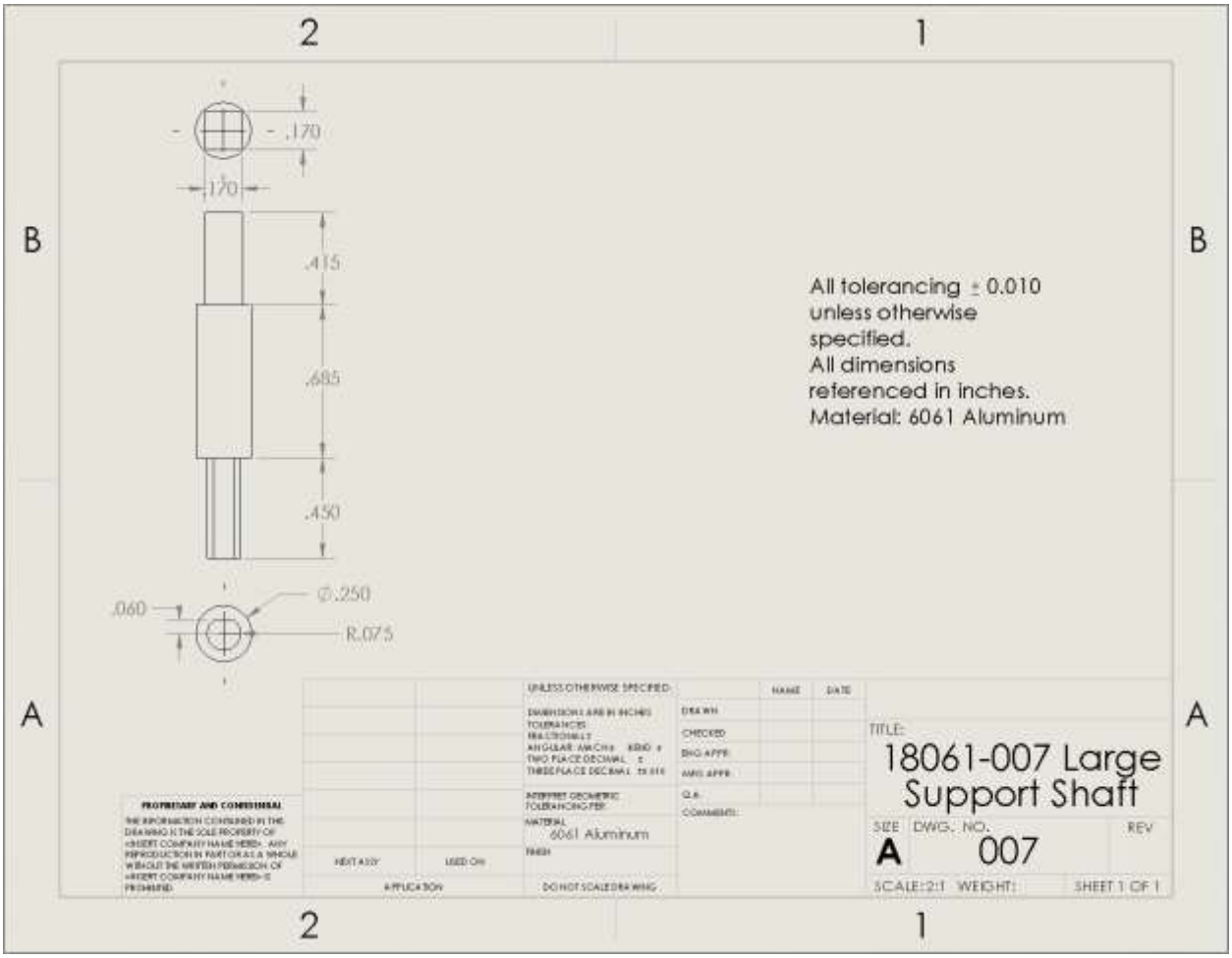


Figure 38. Drawing #007 Part 18061-007 Large Support Shaft

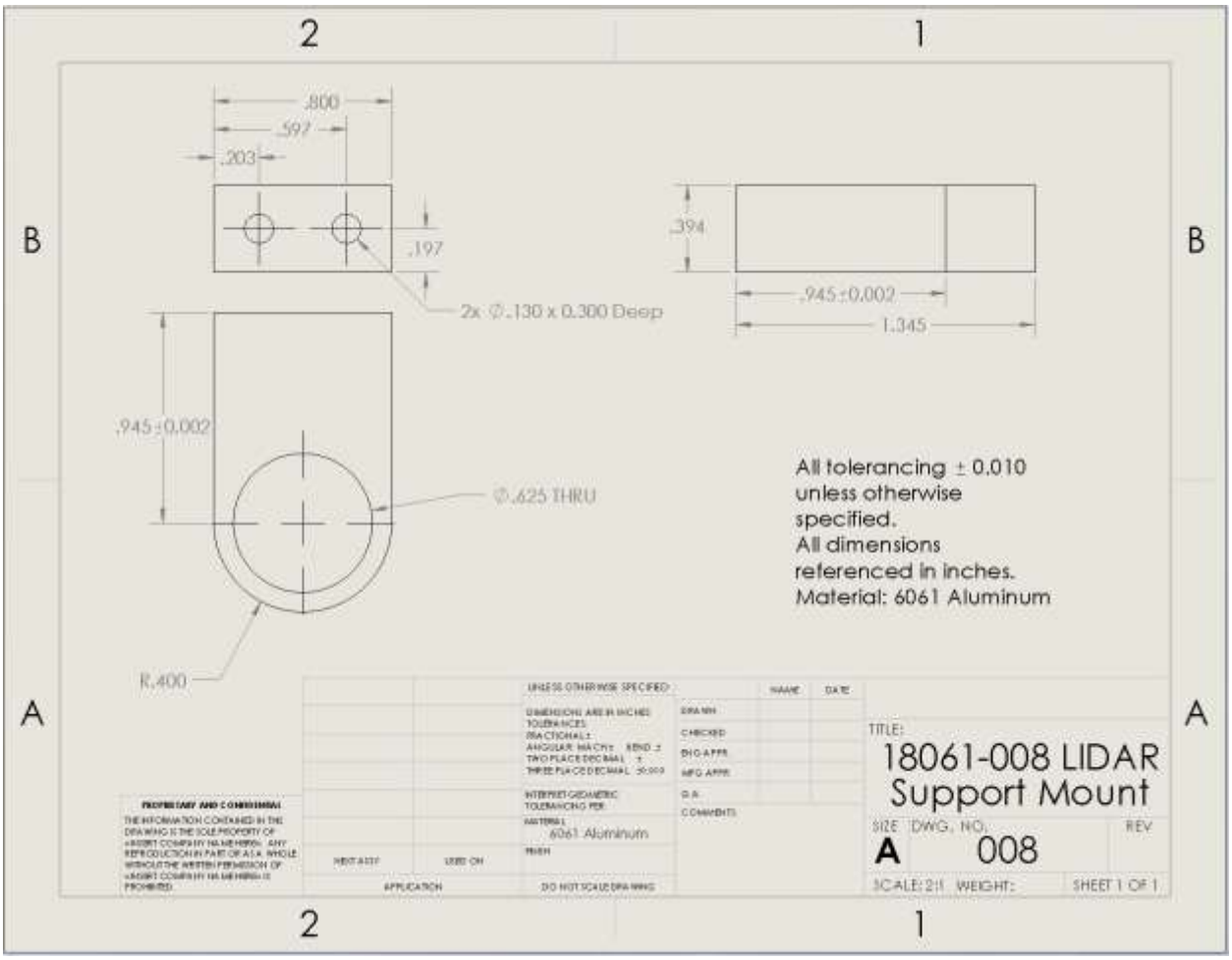


Figure 39. Drawing #008 Part 18061-008 LIDAR Support Mount

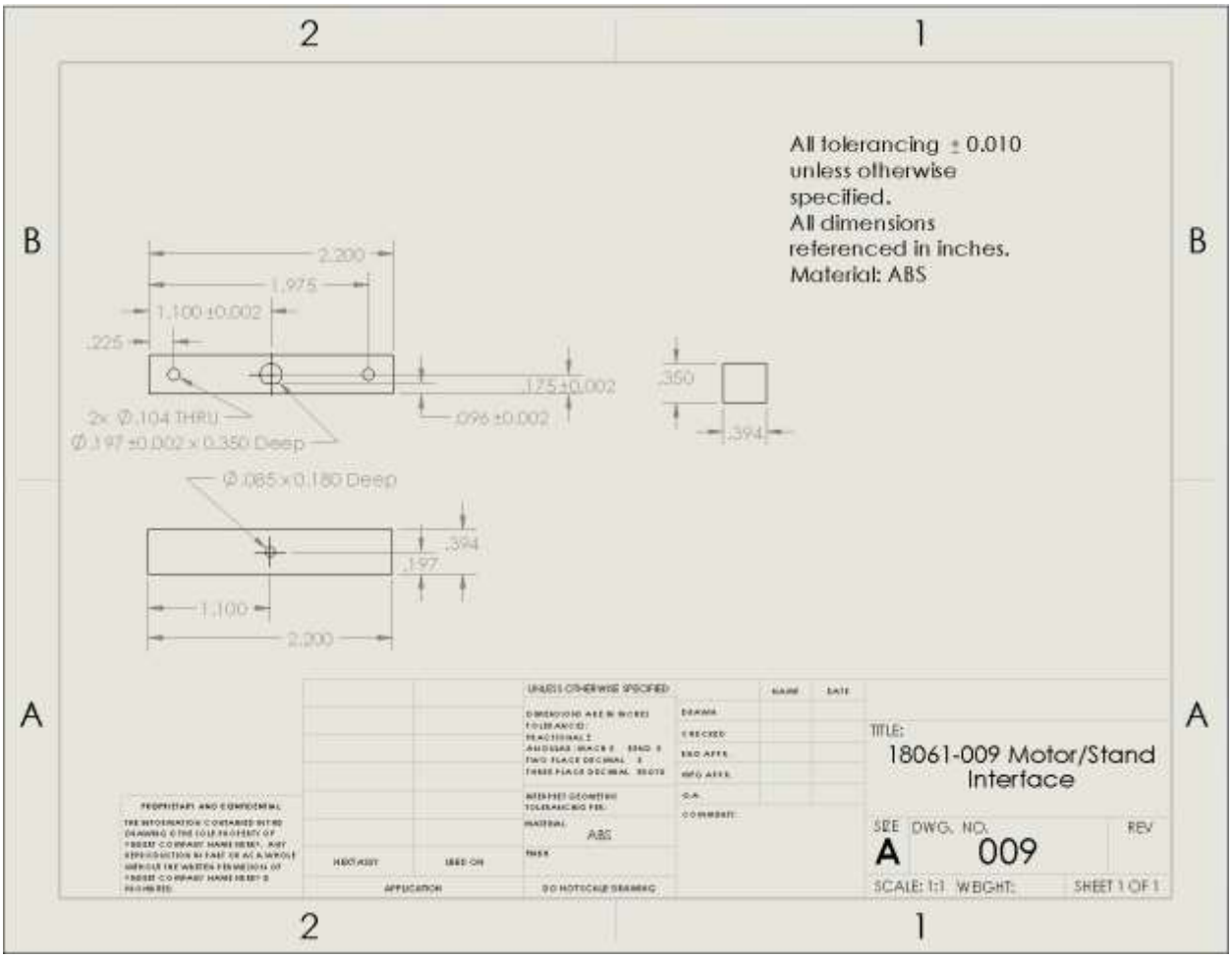


Figure 40. Drawing #009 Part 18061-009 Motor/Stand Interface

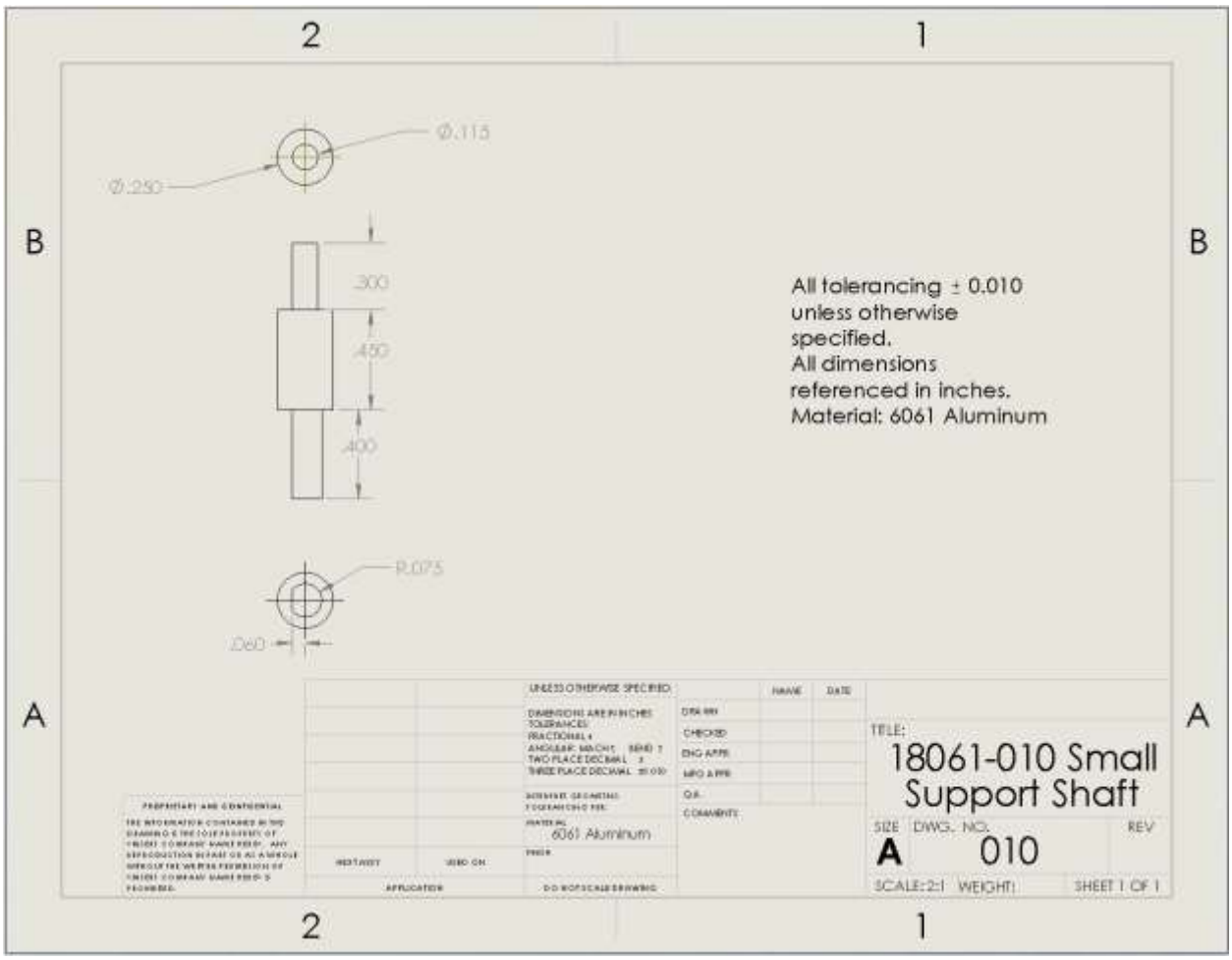


Figure 41. Drawing #010 Part 18061-010 Small Support Shaft

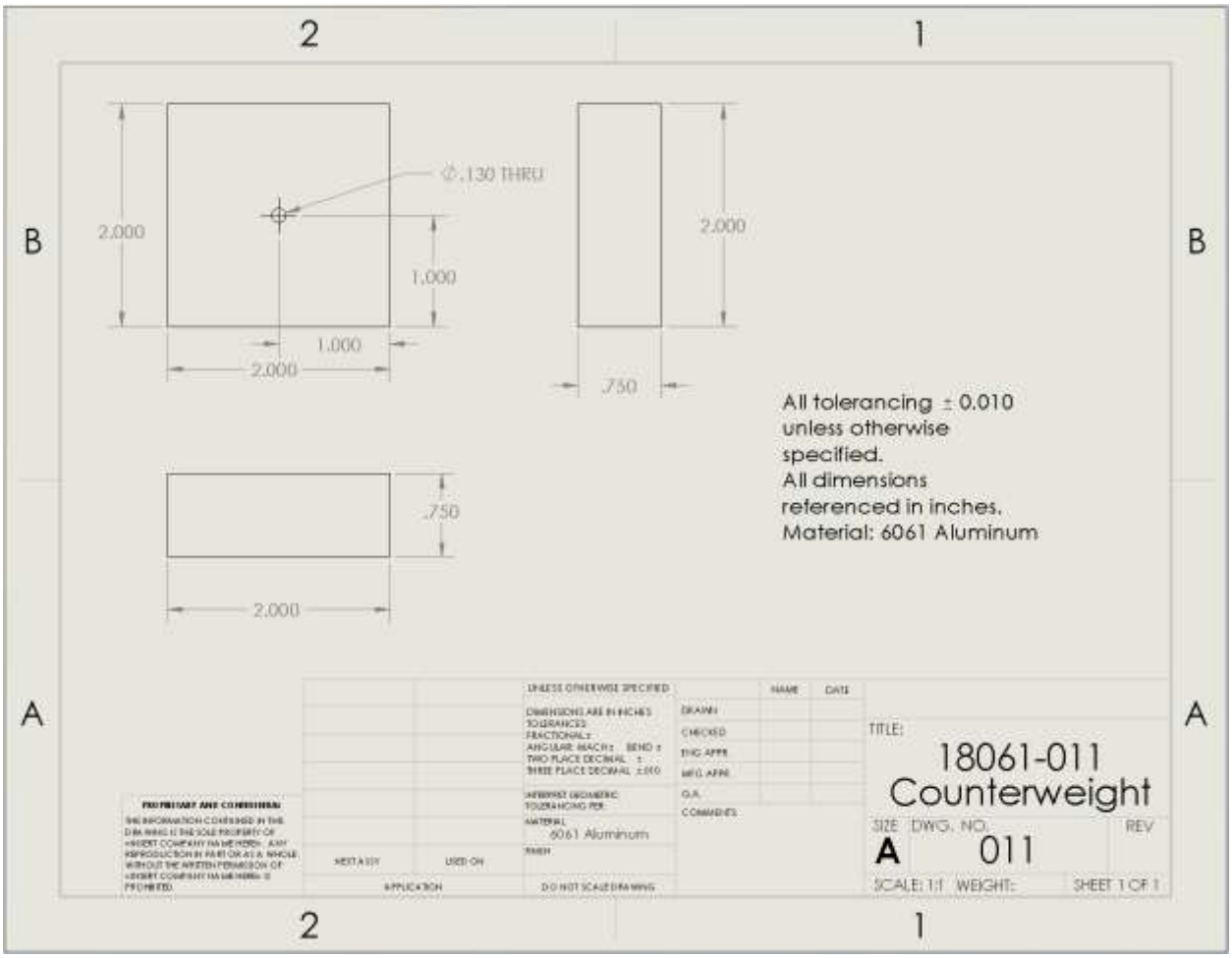


Figure 42. Drawing #011 Part 18061-011 Counterweight

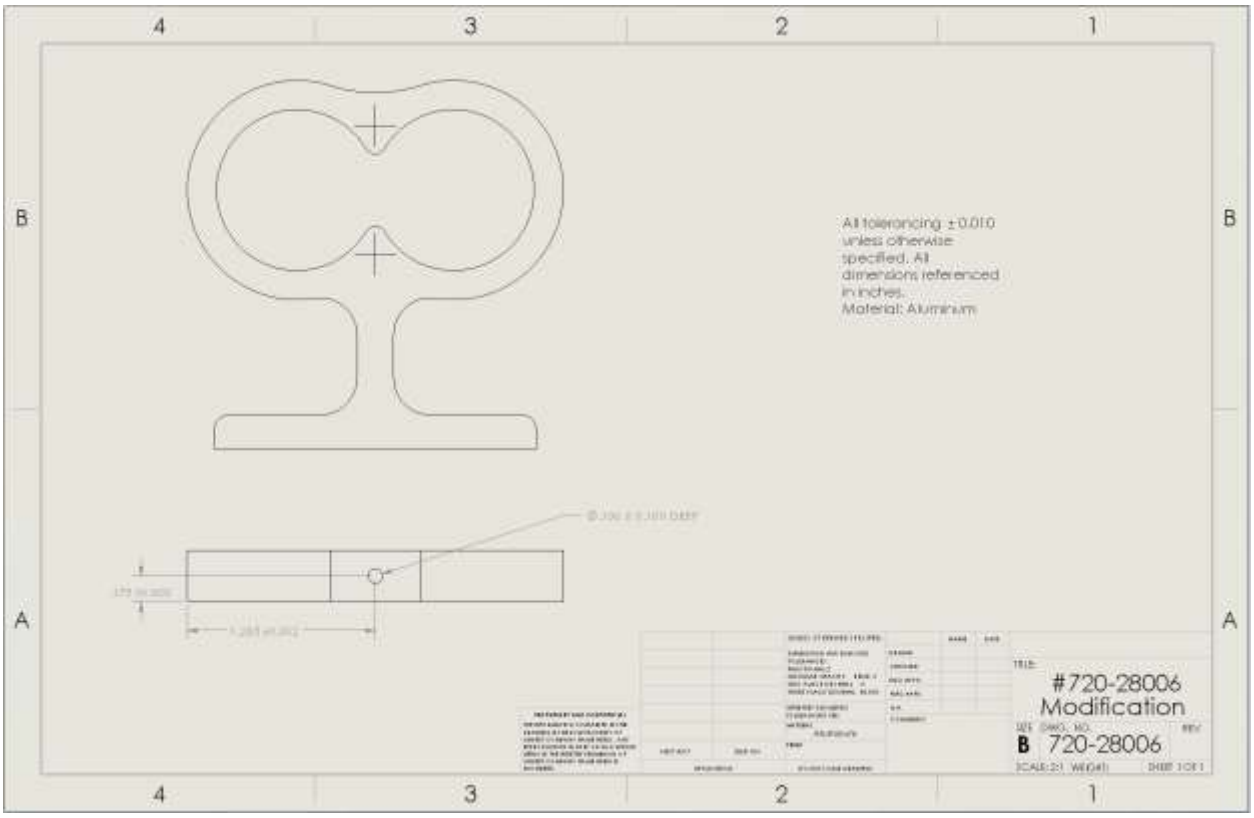


Figure 43. Drawing #720-28006 Part #720-28006 Modification

Assembly Drawings:

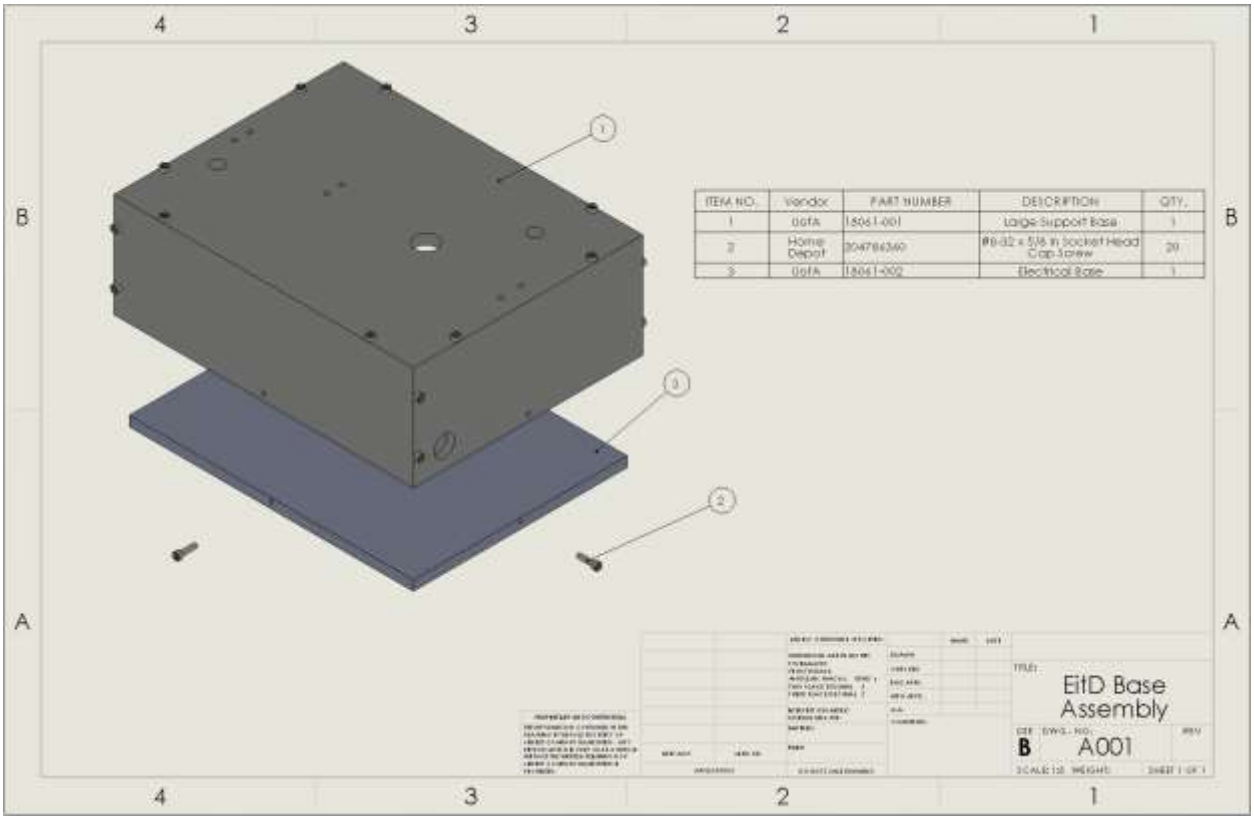


Figure 44. Drawing #A001 EitD Base Assembly

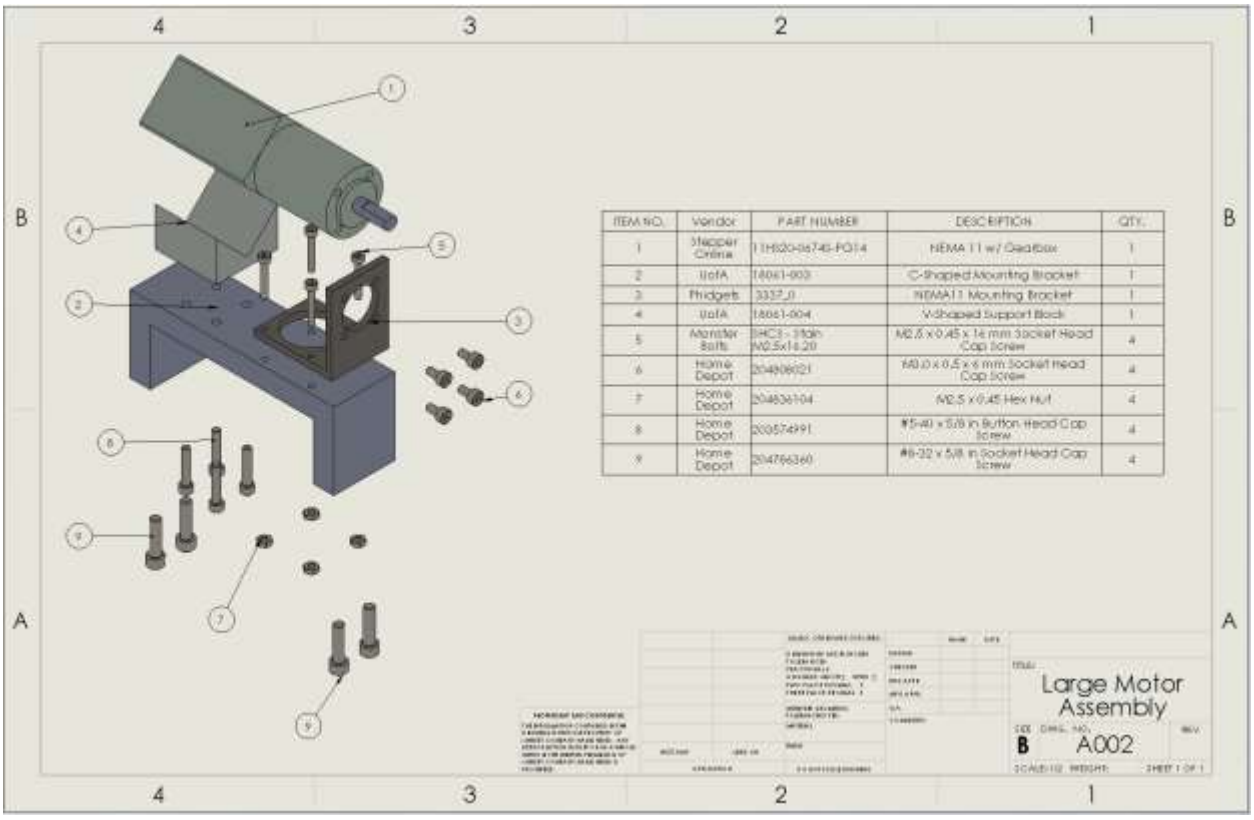


Figure 45. Drawing #A002 Large Motor Assembly

9.4 Other Drawings and Specification Sheets

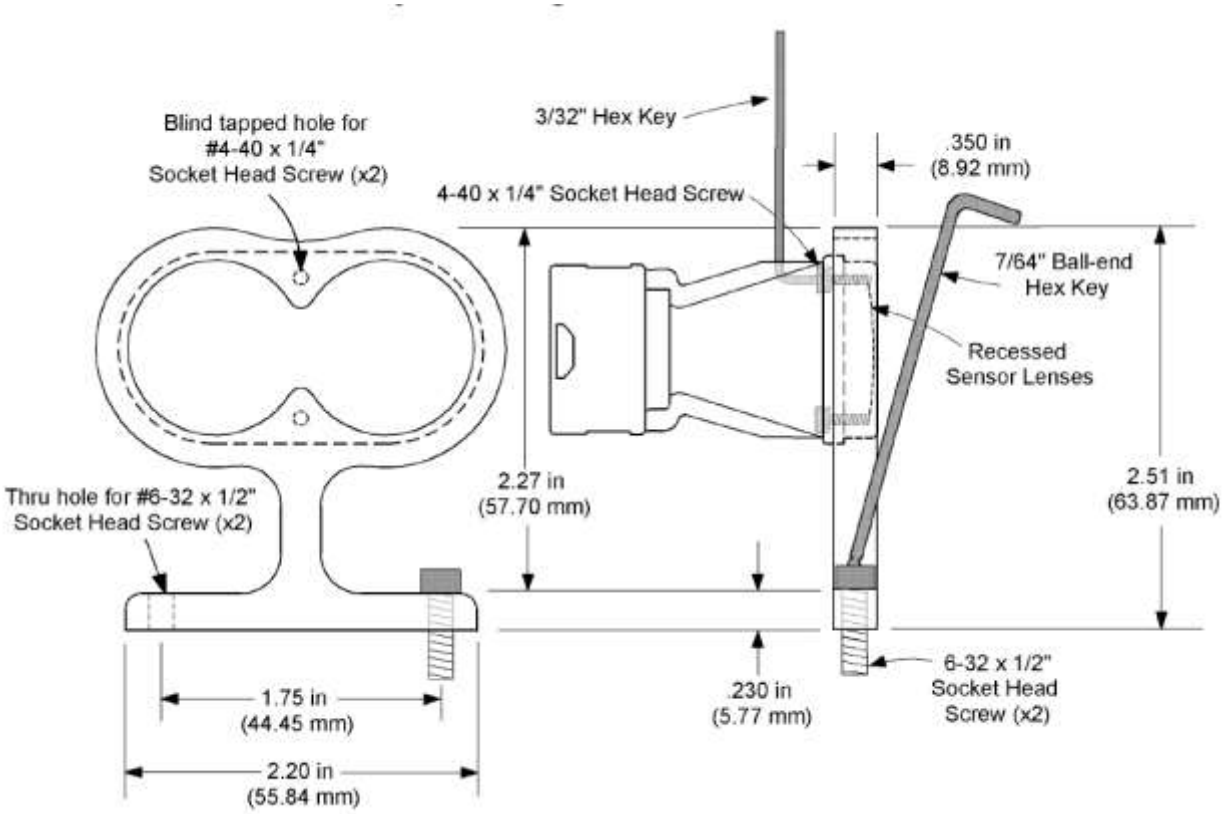


Figure 46. Aluminum Stand Kit for SF Laser Rangefinders - Dimensional/Assembly Drawing

Figure 47. Aluminum Stand Kit for SF Laser Rangefinders - Kit Contents and Assembled View

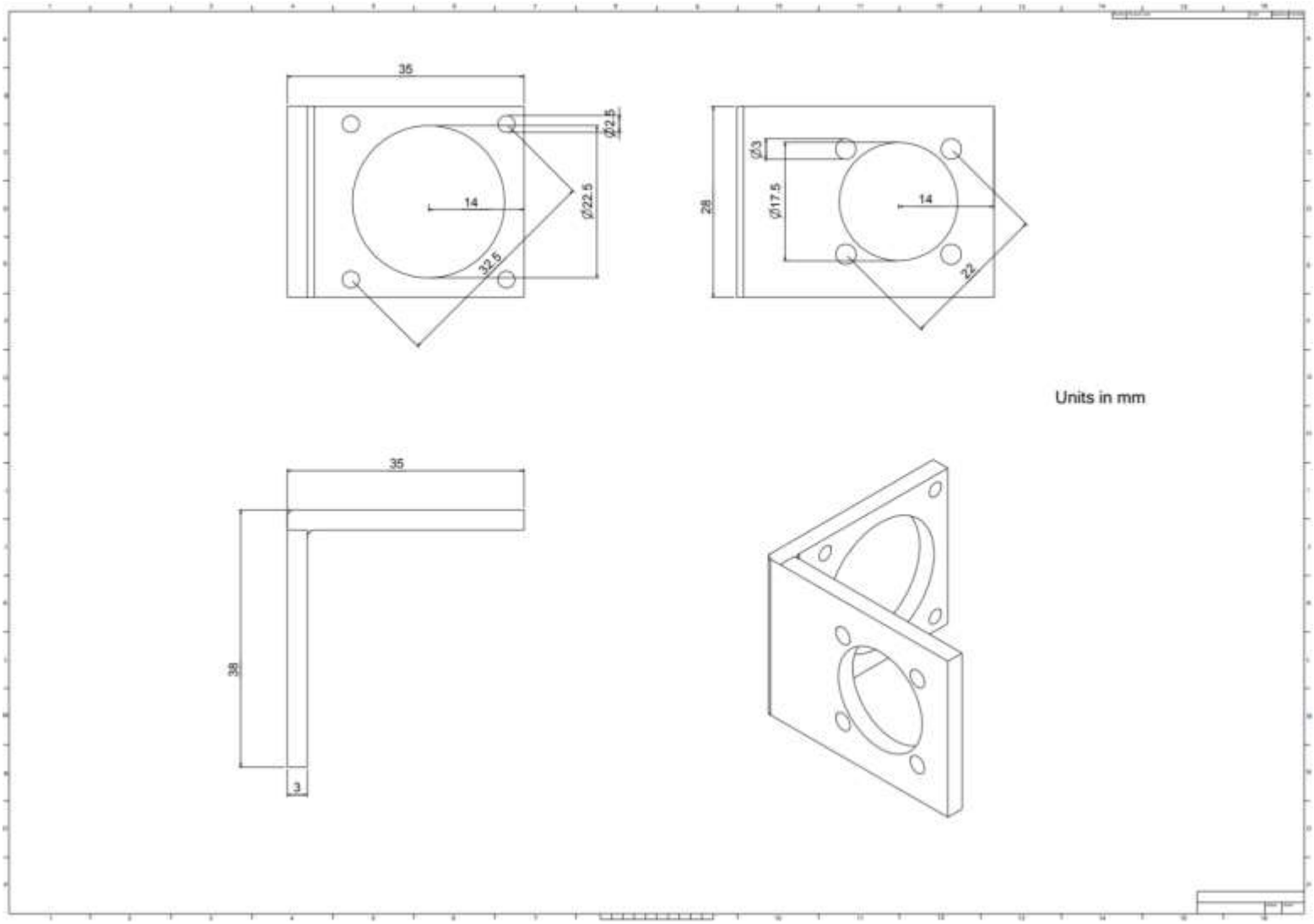


Figure 48. Stepper Mounting Bracket (NEMA 11) - Dimensional Drawing

Physical Properties	
Mounting Plate Size	NEMA - 11
Material	Mild Steel (Coated)

Figure 49. Stepper Mounting Bracket (NEMA 11) - Product Specifications

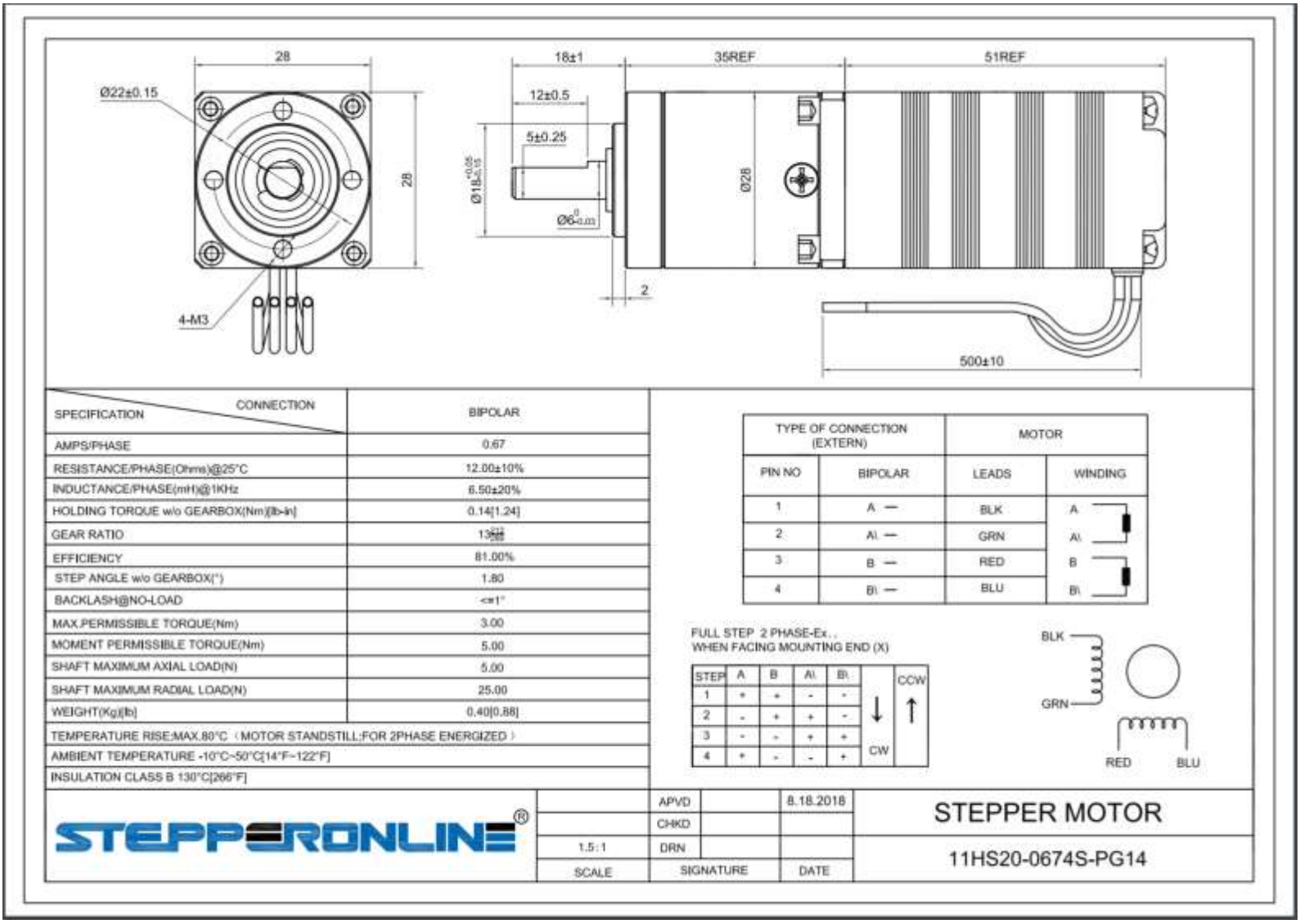
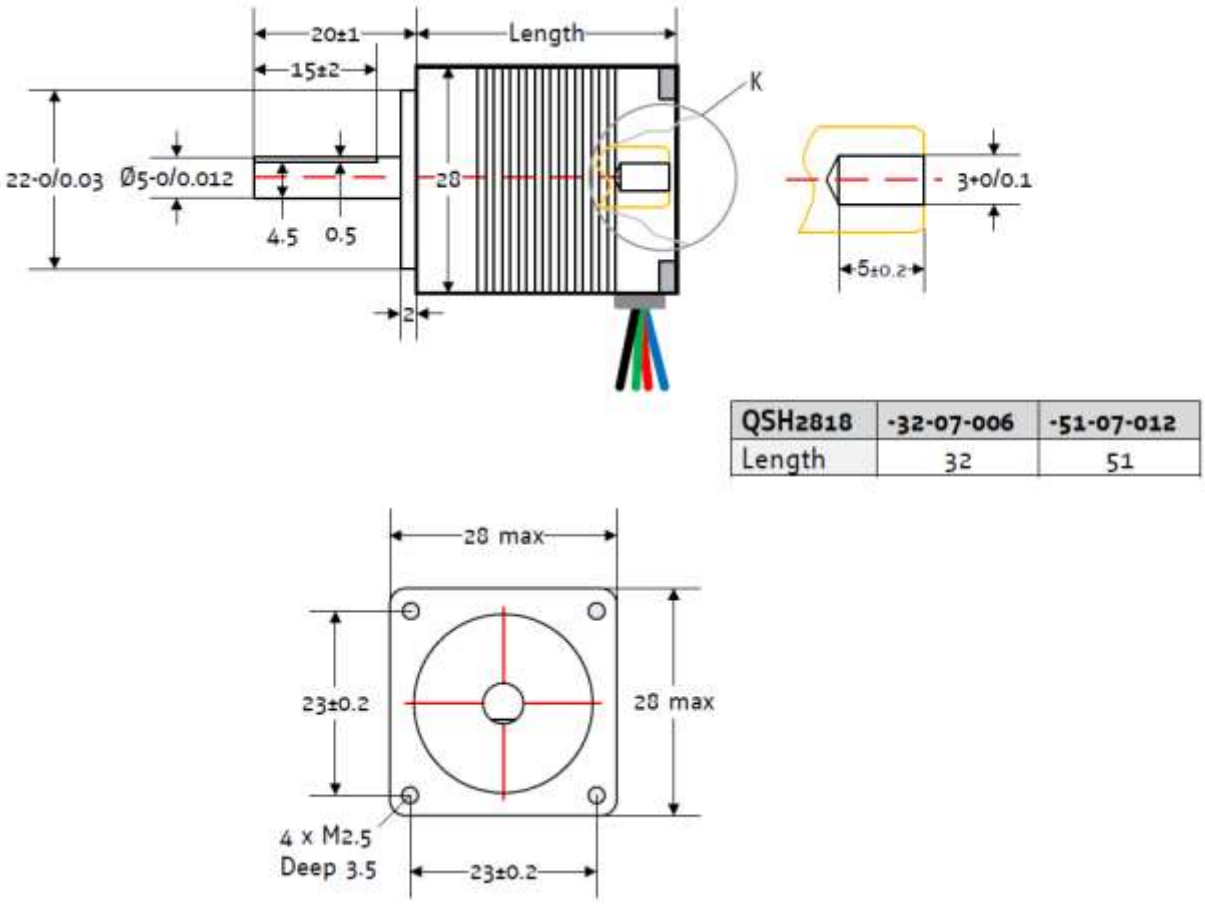


Figure 50. 11HS20-0674S-PG14 NEMA 11 Stepper Motor - Dimensions and Specifications



QSH2818	-32-07-006	-51-07-012
Length	32	51

Figure 51. QSH2818-32-07-006 Nema 11 Stepper Motor - Dimensional Drawing

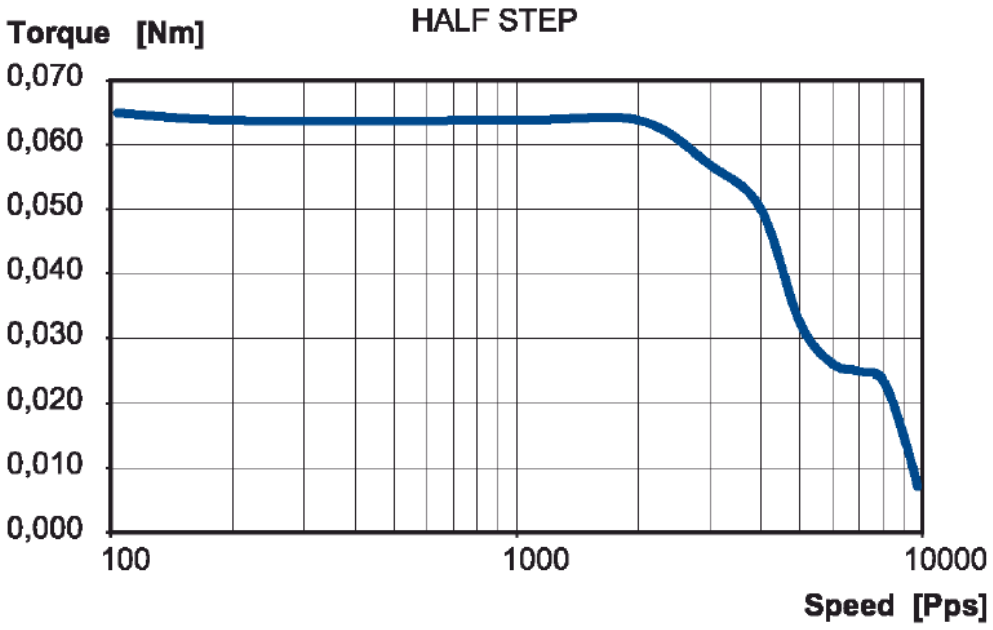


Figure 52. QSH2818-32-07-006 Nema 11 Stepper Motor - Torque Curve

Percentage of rated current	Percentage of motor torque	Percentage of static motor power dissipation	Comment
150%	≤150%	225%	Limit operation to a few seconds
125%	125%	156%	Operation possible for a limited time
100%	100%	100% $= 2 * I_{RMS\ RATED} * R_{COIL}$	Normal operation
85%	85%	72%	Normal operation
75%	75%	56%	Normal operation
50%	50%	25%	Reduced microstep exactness due to torque reducing in the magnitude of detent torque
38%	38%	14%	-"-
25%	25%	6%	-"-
0%	see detent torque	0%	Motor might lose position if the application's friction is too low

Figure 53. QSH2818-32-07-006 NEMA 11 Stepper Motor - Motor Current Settings

Specifications	Parameter	Units	QSH2818	
			-32-07-006	-51-07-012
Rated Voltage	V_{RATED}	V	3.8	6.2
Rated Phase Current	$I_{RMS\ RATED}$	A	0.67	0.67
Phase Resistance at 20°C	R_{COIL}	Ω	5.6	9.2
Phase Inductance (typ.)		mH	3.4	7.2
Holding Torque (typ.)		Ncm	6	12
		oz in	8.5	17.0
Detent Torque		Ncm		
Rotor Inertia		g cm²	9	18
Weight (Mass)		Kg	0.11	0.2
Insulation Class			B	B
Insulation Resistance		Ω	100M	100M
Dialectic Strength (for one minute)		VAC	500	500
Connection Wires		N°	4	4
Max applicable Voltage		V	tbd	tbd
Step Angle		°	1.8	1.8
Step angle Accuracy (max.)		%	5	5
Flange Size (max.)		mm	28.0	28.0
Motor Length (max.)	L_{MAX}	mm	32	51
Axis Diameter		mm	5.0	5.0
Axis Length (typ.)		mm	20.0	20.0
Shaft Radial Play (450g load)		mm	0.02	0.02
Shaft Axial Play (450g load)		mm	0.08	0.08
Maximum Radial Force (20 mm from front flange)		N	28	28
Maximum Axial Force		N	10	10
Ambient Temperature		°C	-20...+50	-20...+50
Temp Rise (rated current, 2phase on)		°C	max. 80	max. 80
Related Trinamic PANdrive		type	PD1-108-28	PD3-108-28

Figure 54. QSH2818-32-07-006 NEMA 11 Stepper Motor - Specifications

Cable type	Gauge	Coil	Function
Black	UL1007 AWG26	A	Motor coil A pin 1
Green	UL1007 AWG26	A-	Motor coil A pin 2
Red	UL1007 AWG26	B	Motor coil B pin 1
Blue	UL1007 AWG26	B-	Motor coil B pin 2

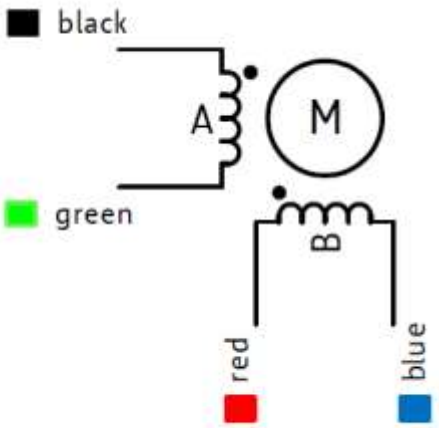


Figure 55. QSH2818-32-07-006 NEMA 11 Stepper Motor - Lead Wire Configuration

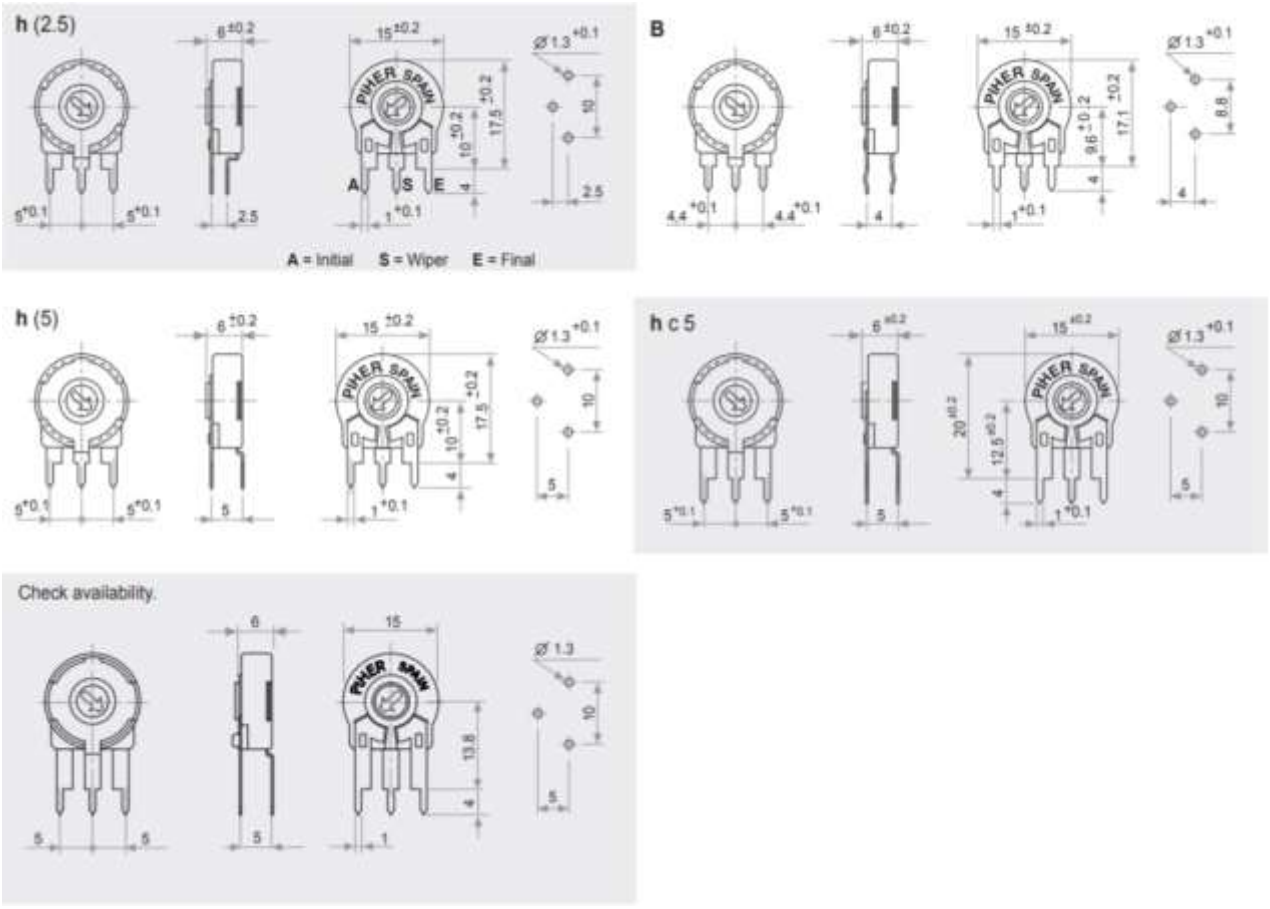
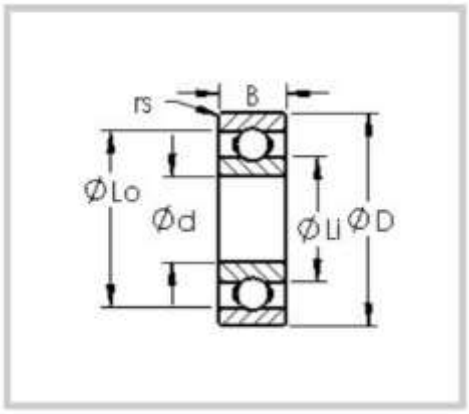
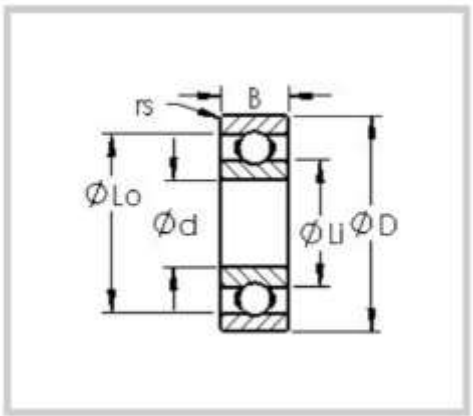


Figure 56. 531-PT15RV18-103A1010 Potentiometer - General Dimensions for Vertical Mount Version



Bearing Type	Open	
Bore Dia (d)	0.2500	in
Outer Dia (D)	0.6250	in
Width (B)	0.1960	in
Radius (min) (rs)	0.002	in
Dynamic Load Rating (Cr)	283	lbs
Static Load Rating (Cor)	112	lbs
Max Speed (Grease)	38,000	rpm
Max Speed (Oil)	45,000	rpm
Max. Shaft Shoulder Dia. Inner (Li)	0.4	in
Min. Housing Shoulder Dia., Outer (Lo)	0.5	in
Ball Qty	8	
Ball Dia (Dw)	0.0937	in
Weight (g)	4.46	grams
Precision	A1	
Standard Clearance	K25	
Material	Martensitic Stainless Steel	

Figure 57. SR4 Instrument Ball Bearing - Drawing and Specification



Bearing Type	Open	
Bore Dia (d)	0.1250	in
Outer Dia (D)	0.3750	in
Width (B)	0.1094	in
Radius (min) (rs)	0.006	in
Dynamic Load Rating (Cr)	122	lbs
Static Load Rating (Cor)	41	lbs
Max Speed (Grease)	53,000	rpm
Max Speed (Oil)	63,000	rpm
Max. Shaft Shoulder Dia. Inner (Li)	0.2	in
Min. Housing Shoulder Dia., Outer (Lo)	0.3	in
Ball Qty	7	
Ball Dia (Dw)	0.0625	in
Weight (g)	0.96	grams
Precision	A1	
Standard Clearance	K25	
Material	Martensitic Stainless Steel	

Figure 58. SR2-6 Instrument Ball Bearing - Drawing and Specifications

AR3000 Standard Model Specifications

	Standard AR3000 (2mrad divergence) 	AR3000 (10 mrad divergence) 
Span		
to 90% reflectance targets (white)	0.5 - 300 m [20 in.- 980 ft.]	0.5 - 50 m [20 in.- 165 ft.]
to 10% reflectance targets (dark)	8 - 200 m [26 - 650 ft.]	0.5 - 50 m [20 in.- 165 ft.]
to high-gain reflectors *	3 km [1.9 mi.] max	NA
Accuracy	+/- 20 mm [0.79 in.] at 100 Hz +/- 60 mm [2.36 in.] at 2000 Hz	
Resolution	1 mm [0.04 in.]	
Sample rates	2000 Hz maximum, or sample trigger (serial command and analog)	
Weight (less cable)	850 grams [1.9 lbs.]	650 grams [1.4 lbs.]
Laser (measuring)	905 nm, Infrared, Class 1, IEC/EN60825-1:2001	
Laser (aiming)	635 nm, Visible Red, Class 2, Complies with 21 CFR 1040.10 with Laser Notice 50, IEC/EN60825-1:2001 Aiming laser can be disabled	
Laser divergence	1.7 mrad	10 mrad
Power	10 - 30 Volts DC, 170 - 500 mA draw Heater operation: 24 Volts DC, 11.5 W	
Operating Temp	-40 to 60 °C [-40 to 140 °F]	
Environmental	NEMA – 4, IP67. Keep lenses clean for best performance. Aluminum case.	
Shock & Vibration	Shock (single): 500g / 1ms, DIN ISO-9022-30-08-1 Shock (continuous): 10g / 6ms / 1000x in all 6 directions, DIN ISO-9022-31-01-1 Vibration: 10 Hz ... 2000 Hz ... 10 Hz / 0.075 mm / 1g / 2 cycles in 3 axes, DIN ISO-9022-36-02-1	
Outputs serial	RS232 full duplex, RS422 (optional output) unterminated and terminated	
analog	4-20 mA, limit switch	
Cable	2 m (6.6 ft.) length, 12 conductor, Binder series 723 flange-mount connector, soldertail wire termination	

	Red – no connection	**Pink** - unassigned (RS232), Tx+ (RS422)	– 4-20 mA Out
	Black – Ground	– unassigned (RS232), Tx- (RS422)	**Green** – trigger input
	– TxD (RS232), RX+ (RS422)	**Red/Blue** – switching output Q1	**Blue** – 10-30 Volt DC IN
	/**Pink** – Ground	**Brown** – RxD(RS232), RX- (RS422)	**Violet** – switching output Q2

* Contact Acuity for these targets

Figure 59. AR3000 Spec Sheet

11.0 References

[1] *Aluminum Stand Kit for Laser Rangefinders (#725-28920)*. Parallax Inc, May 2018.

[2] *Stepper Mounting Bracket (NEMA 11) as_1088_6.* Phidgets Inc, N.D.

[3] *11HS20-0674S-PG14 Stepper Motor Full Data Sheet*. OMC-StepperOnline, August 2018.

[4] *QMot QSH2818 family Manual.* Trinamic Motion Control GmbH & Co, October 2010.

[5] *PT15RV18-103A1010E-S Datasheet.* Piher, N.D.

[6] *Part Number: SR4 Miniature & Instrument Series, Stainless Steel Ball Bearing.* AST Bearings and Related Products & Services, N.D.

[7] *Part Number: SR2-6 Miniature & Instrument Series, Stainless Steel Ball Bearing.* AST Bearings and Related Products & Services, N.D.

12.0 Honors Justification

The EitD project detailed above was a collaborative effort between 6 people, however there were several areas in the project where I took sole ownership of required tasks. In our design process we went through several iterations of the concept. For each of the four concepts developed, I was required to analyze the mechanical and optical systems to verify they adhered to the system requirements. The most involved analysis was the scan time. There were two different scanning patterns and two different modes of scanning to be analyzed. The first method was using a Risley Prism to steer a laser through the FOR. This required determining the maximum angular velocity the prisms could be physically moved as well as how long it would take to get a reliable return signal. These calculations were then the inputs into a script I wrote to determine how long it would take to scan the FOR with a cyclical and line-by-line scan pattern approach. The second method was steering a Lidar with stepper motors. Again, the analysis on the systems angular inertia was needed so I could determine a maximum speed within the motor's torque limit. These calculations would be easy to experimentally determine; however, they were difficult to find in published research. In the first semester of senior design we are not allowed to purchase any materials as the class is in a design phase. I was able to determine all the input parameters through research on both the mechanical components I was familiar with, and the optical components I had no prior knowledge with. Based on values found in publications, my calculated scan time of 31 seconds matched the experimentally verified time of 30.9 seconds.

In the initial description of the project, the scan time requirement was initially set at 4 seconds. However, based on our analysis this did not seem plausible. To get the requirement changed our team was required to write a summary of our analysis and present it to our sponsors at Raytheon. I took the onus to write the report and provide an alternate path to achieve the desired 4 second scan time through changing the requirement on the FOR size. After presenting my analysis and alternate model to our sponsors, they decided the system's FOR size was more valuable and agreed to the modified scan time requirement of 35 seconds.

Beyond the analysis for our system, I had to create the test procedures and equipment necessary to verify the EITD system met its requirements. The majority of senior design projects are continuations of past projects or modifications to current products. Unfortunately, in our case, we were tasked with creating an entirely new system, which led to the need for custom test equipment. In order to do this, I again looked to current research on similar devices to create modified procedures and fixtures to test our system. The devices created were two adjustable lidar mounts for a camera tripod, a six-inch mount for a corner cube, and a bearing accuracy board that needed a precision of at least ±0.001 inches. The system test fixtures were not the only thing constructed for the project. Due to long lead times in the machine shop, I decided to craft our project by hand. In the image on the first page of the report, the rotating platform is the only piece not created manually. This was done to ensure a quick turn around on the system components and to both learn and gain experience crafting precision parts. Because of this process, our team was able to test and optimize our system for 4 weeks prior to design day.