University of Colorado Boulder
Aerospace Engineering Sciences
Senior Projects - ASEN 4018

Project ATOMIC

Advanced Tracking of Orbital Motion for Inter-satellite Clusters

Project Final Report

Monday 4th May, 2020

1 General Information

1.1 Customer

Affiliation	**Customer Name**
General Atomics	Benjamin Kragt
General Atomics	Tommy Romano
General Atomics	Zach Koch

1.2 Team Members

Name: Andrew Dellsite Email: andrew.dellsite@colorado.edu Phone: 360-489-5574	Name: Adam Farmer Email: adfa8721@colorado.edu Phone: 303-941-9692
Name: Camilla Hallin Email: camilla.hallin@colorado.edu Phone: 720-323-8436	Name: Corey Huffman Email: cohu8717@colorado.edu Phone: 254-289-9973
Name: Nate Lee Email: nale2807@colorado.edu Phone: 425-999-0716	Name: Brendan Lutes Email: brlu7909@colorado.edu Phone: 720-256-8518
Name: Jamison McGinley Email: jamc3951@colorado.edu Phone: 719-304-6017	Name: Anastasia Muszynski Email: anmu1134@colorado.edu Phone: 719-425-0178
Name: Hunter Peery Email: hupe3689@colorado.edu Phone: 720-375-6422	Name: Jarrod Puseman Email: jarrod.puseman@colorado.edu Phone: 720-775-8163
Name: Erin Shimoda Email: erin.shimoda@colorado.edu Phone: 832-247-0129	Name: Emily Webb Email: emily.webb@colorado.edu Phone: 303-817-6418

Contents

List of Figures

List of Tables

Summary of Acronyms

Acronym	Definition
ADC	Analog-to-Digital Converter
AOA	Angle of Arrival
ATOMIC	Team Name: Advanced Tracking of Orbital Motion for Inter-satellite Clusters
CONOPS	Concept of Operations
CDMA	Code Division Multiple Access
CNC	Computer Numerical Control
CPR	Critical Project Elements
CTM	Coordinate Transformation Method
DSS	Distributed Space System
DR	Design Requirement
ESD	Electrostatic Discharge
FBD	Functional Block Diagram
FDMA	Frequency Division Multiple Access
FOV	Field of View
FPGA	Field Programmable Gate Arrays
FR	Functional Requirement
G.A.	General Atomics
GPS	Global Positioning System
KRC	Kinematic Range Comparison
MPH	Miles per Hour (speed)
PDD	Preliminary Design Document
PDDF	Pseudo-Doppler Direction Finding
PI	Phase Interferometry
R.C.	Remote-control
R.F., RF	Radio Frequency
RX, Rx	Receiving Antenna
SDR	Software-defined Radio
SNR	Signal-to-Noise Ratio
SMT	Sequential Multiple Transmission
SST	Satellite-to-satellite Tracking
TDOA	Time Difference Of Arrival
TX, Tx	Transmitting Antenna
U.S., US, USA	United States of America
USD	U.S. Dollar
WBS	Work Breakdown Structure

Nomenclature

$\hat{x}_k$	Estimated State at time k
λ	Wavelength
ν	Beam vertical displacement in the center
Φ	Azimuthal Angle
ρ	Density of Air
θ	Polar Angle
A	Process Model
A_x	Cross Sectional Area, Relative to Free Stream
c	Speed of light in a vacuum inertial frame
C_D	Coefficient of Drag
d	Distance
E	Young's Modulus of Elasticity
F_d	Force due to Drag
H	Sensor Model
h	Planck constant
I	Identity Matrix
I	Second moment of area for a beam
k	Discretized Timestep
k	Spring Stiffness
K_k	Kalman Gain at time k
P_k	Covariance Matrix at time k
Q	Process Noise Covariance Matrix
R	Range
R	Sensor Noise Covariance Matrix
Re	Reynolds Number
t	Time
V	Air Speed
z_k	Sensor Update at time k

2 Project Purpose

Author: Emily Webb

Small satellite constellations are taking the aerospace industry by storm [1]. Lightweight, low-cost, and easily manufacturable, they serve a variety of uses from a global communication system [2] to in-depth analysis of the Earth's magnetic field not possible with a single orbiter. As the number of these constellations grows, the aerospace industry, from defense contractors to academics, will need to investigate innovative methods to accurately determine the position of each satellite in the constellation. This information helps the cluster avoid collisions and preserve its shape for communication and science purposes. If a satellite loses GPS connectivity, relative position measurements from nearby satellites can enable it to regain knowledge of its location. One such position determination method is satellite-to-satellite tracking (SST), in which the position of a satellite in the constellation is determined relative to another in the constellation. The aim of this project was to examine a variety of heritage and emerging SST technologies and to design and test a prototype solution.

The industry is just beginning to explore software defined radios (SDRs) as a method for intersatellite communication, ranging, and heading determination. Though a space tether company called Tethers Unlimited uses specifically designed SDRs for satellite-to-satellite tracking in their SWIFT RelNav system [3], papers on the subject date back only to 2015. Thus, in itself, the use of SDRs to calculate relative position between satellites without ground communication is a fairly novel concept. However, the design and implementation of these systems is currently costly. NASA recently purchased a series of SDRs from Tethers Unlimited for just under one million dollars [5]. By constructing a similar system with off-the-shelf components for less than 5,000 USD. Success of the system would make SDR intersatellite communication more widely available within the industry, especially for low-cost cubesats from universities that wish to change the face of space communication.

3 Project Objectives and Functional Requirements

3.1 Functional Requirements

Author: Emily Webb

FR 1 The Beacon shall transmit RF signals between 2 and 4 GHz (S-Band).
Rationale: The Beacon will transmit radio frequencies in the S-band for legality and accuracy purposes. RF as a method for determining range and heading was determined through trade studies conducted early on in the project history.

FR 2 The Sensor shall receive RF signals transmitted by the Beacon.
Rationale: The Sensor and Beacon must be able to communicate with one another to determine the Beacon's position relative to the Sensor.

FR 3 The Sensor shall track relative position of the Beacon within a 2% error bound 66.7% of the time (1σ).
Rationale: The original 2% error bound on this requirement was defined by the customer. This was originally intended to be a 3σ error bound, but after investigation into timing errors, the team discovered they would not be able to meet this bound with equipment available. The requirement was revised and deemed feasible with the 1σ error bound.

FR 4 The tracking system shall alert the operator if the satellite drifts more than 10% of the range away from a predetermined notional path.
Rationale: The customer asked that the system be able to notify the operator/ground systems if a satellite drifted from its notional course by 10% of the range. The 10% bound was chosen because it is much larger than the error bound on position determination.

FR 5 The Beacon system shall operate between temperatures of -20°C to 50°C, inclusive.
Rationale: This was a customer defined requirement based on the temperature changes a satellite would undergo in an Earth-centric orbit.

FR 6 The Beacon system shall be powered by a 28±6 V unregulated power supply.
Rationale: This was a customer defined requirement based on the power that could be supplied from their satellite buses.

3.2 Levels of Success

Author: Jarrod Puseman

The ATOMIC project will develop a ground-based Sensor-Beacon package with accompanying software which will determine the relative position of the Beacon. This Beacon will act as a target for the Sensor to track and the software package to analyze and extrapolate position information. This information is to be stored locally for analysis. For all levels of success each of the aforementioned components will be developed and tested by ATOMIC on the ground for future use in satellite constellations.

Table 1 outlines the levels of success based on the functional requirements listed above for the project with Level 1 serving as the minimum successful implementation of the Beacon tracking system individual components. Level 2 includes the completion of intermediate goals which describe the customer's desires as opposed to requirements. Levels 3 and 4 outline the ideal form of the project at the end of development with total system integration. Each of the Critical Project Elements has its levels of success mapped below.

Table 1: Levels of Success

Success Level	**2-Way Range Finding**	**Phase Interferometry**	**Data Handling**	**Electrical Components**	**Testing**
Level 1	Sensor Transmits a Probe Signal. Beacon Receives the signal.	Sensor can measure phase difference on a set of antennas.	Beacon and Sensor individually complete all software processes.	Circuitry allows functionality of the Sensor and/or Beacon.	Preliminary Testing confirms power models, functionality, and operability.
Level 2	Sensor able to compute range within 2% of actual range.	Sensor able to multiplex the receiving antennas.	Sensor and Beacon communicate with each other (probe and reply). CAN bus functional for alert feature.	Circuitry functional at all temperatures in the desired range.	Stationary Test performed (see Testing section of the report.)
Level 3	Sensor computes relative position of the Beacon to 2% of the range.		Position update rate no slower than 1 Hz.	n/a	Relative Motion Test performed (see Testing section for more detail.)
Level 4	Sensor applies Kalman filter to relative position computation.		Position update no slower than 1 Hz. CAN bus fully operational.	n/a	Cluster Motion Tests performed (See Testing Section for more detail.)

3.3 Concept of Operations

Author: Camilla Hallin

On-orbit relative position data is useful in the context of satellite networking to enable formation flying in satellite clusters. Project ATOMIC will design and test a ground based Sensor system mimicking a two satellite network which allows the Sensor satellite analog to detect the relative position of the Beacon satellite analog.

The process begins when the Sensor sends a polling message via Radio Frequency transmission. This signal is received by the Beacon. The Beacon processes the incoming message and replies with a unique identifier reply sequence. Based on time of flight and phase off-set on the Sensor phased array, the relative position –consisting of range and heading– is computed. The Sensor then compares the relative position to the expected path and if the Beacon deviates by more than 10% of the range, the alarm flag is set. The Sensor outputs the relative position and alarm flag status at a rate of 1 Hz to the Flight Computer analog via

CAN bus serial communication protocol. The system will be tested at a range of 100-300 m with relative velocities from 0-10 m/s. The system will be characterized for performance at these various ranges and relative speeds.

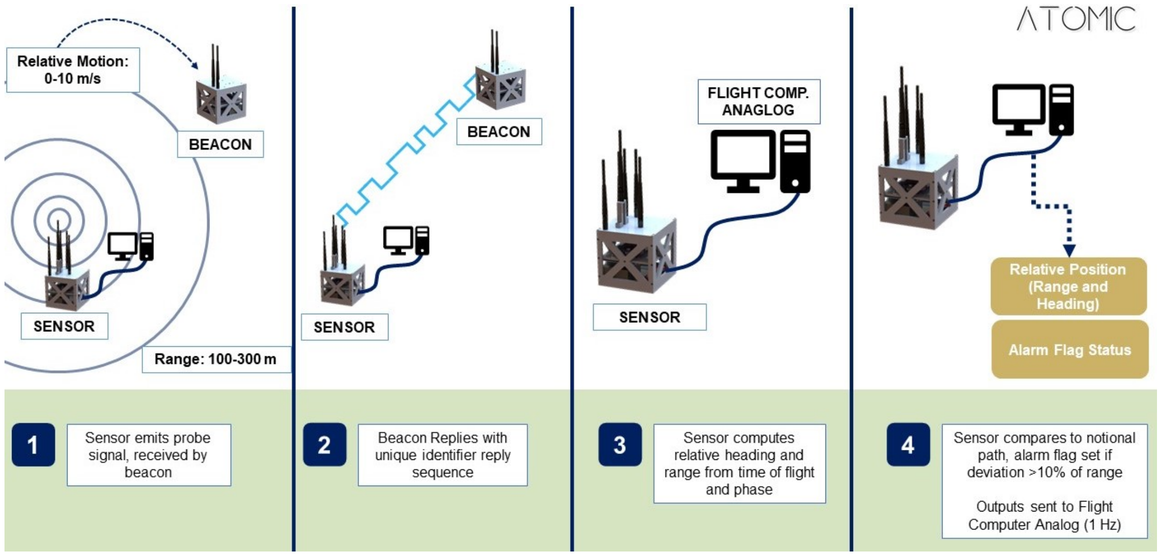

ATOMIC
Relative Motion: 0-10 m/s
BEACON
SENSOR
Range: 100-300 m
BEACON
SENSOR
FLIGHT COMP. ANAGLOG
SENSOR
Relative Position (Range and Heading)
Alarm Flag Status
1 Sensor emits probe signal, received by beacon
2 Beacon Replies with unique identifier reply sequence
3 Sensor computes relative heading and range from time of flight and phase
4 Sensor compares to notional path, alarm flag set if deviation >10% of range
Outputs sent to Flight Computer Analog (1 Hz)

Figure 1: Concept of Operations for Project Atomic

3.4 Functional Block Diagram

Author: Hunter Peery

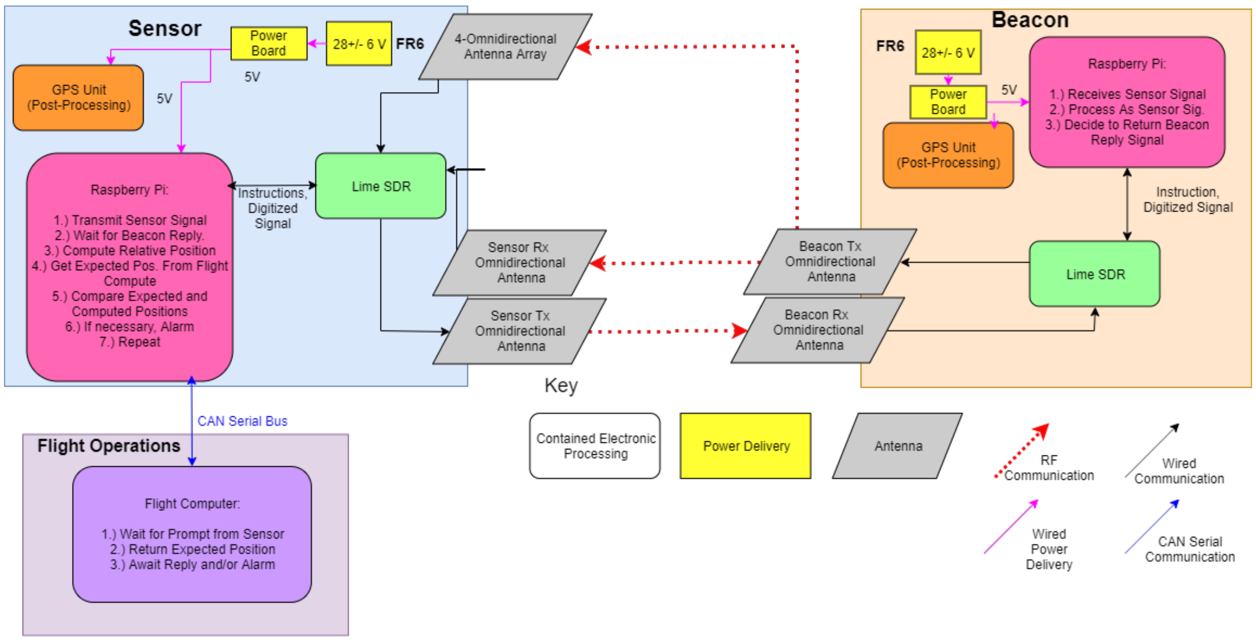

Sensor
Power Board
28+/- 6 V
FR6
4-Omnidirectional Antenna Array
GPS Unit (Post-Processing)
5V
5V
Raspberry Pi: 1.) Transmit Sensor Signal 2.) Wait for Beacon Reply. 3.) Compute Relative Position 4.) Get Expected Pos. From Flight Compute 5.) Compare Expected and Computed Positions 6.) If necessary, Alarm 7.) Repeat
Instructions, Digitized Signal
Lime SDR
Sensor Rx Omnidirectional Antenna
Sensor Tx Omnidirectional Antenna
CAN Serial Bus
Flight Operations
Flight Computer: 1.) Wait for Prompt from Sensor 2.) Return Expected Position 3.) Await Reply and/or Alarm
Beacon
FR6
28+/- 6 V
Power Board
5V
GPS Unit (Post-Processing)
Raspberry Pi: 1.) Receives Sensor Signal 2.) Process As Sensor Sig. 3.) Decide to Return Beacon Reply Signal
Instruction, Digitized Signal
Lime SDR
Beacon Tx Omnidirectional Antenna
Beacon Rx Omnidirectional Antenna
Key
Contained Electronic Processing
Power Delivery
Antenna
RF Communication
Wired Communication
Wired Power Delivery
CAN Serial Communication

Figure 2: Functional Block Diagram

The functional block diagram for Project ATOMIC is shown in Figure 2. The upper left blue box portrays the Sensor package. This is one of the two packages to be constructed. The yellow power board will receive power from a 28 ±6V Lithium Ion DC battery, which will also power the other components. The next important element is the Raspberry Pi, denoted by the purple box. This mini-computer undertakes several tasks for project success. It will receive the digitized signal, compute the position from the flight computer, compare the expected vs actual position, then communicate that an alarm is necessary to the flight computer if the Beacon is drifting off its expected path. The Raspberry Pi will be communicating with the flight operations computer (laptop) through CAN Bus, as specified by the client. The Lime SDR component, in green, processes the signal from the Pi. It then sends the signal to the transmit specific omni-directional antenna. Notice the separate receive and transmit antennas in order to minimize complexity for the SDR. From here, a radio frequency signal is sent to the Beacon package denoted in orange. The signal being sent from the Sensor to the Beacon will be uniquely coded to avoid confusion with possible other packages or multipath errors from other incoming RF signals. The Beacon receive antennas will receive the incoming transmission, passing the signal to the SDR which digitizes the signal for the Pi. The Raspberry Pi will then process the signal, and decide to reply with the unique Beacon identifier code. This signal is sent using the transmit antenna to be received by the Sensor. This entire process is iterative throughout the duration of the test. The GPS on both the Beacon and Sensor is denoted by the orange box. Data collected with this unit will be used in post processing to verify the accuracy of the measured position.

4 Design Process and Outcome

4.1 Trade Studies

Authors: Erin Shimoda and Hunter Peery

To begin the design process, the team conducted a trade study to analyze different forms of communication between the Beacon and Sensor packages. The options include RF ranging, GPS, image processing, and a hybrid method using both RF and GPS. Other methods such as laser interferometry were also considered but ultimately dropped due to complexity.

Table 2: Mission – Specific Trade Study

	Weighting	**RF Ranging**	**GPS**	**Image Processing**	**Hybrid (RF/GPS)**
Cost	14%	5	5	3	2
Mass	8%	4	4	3	3
Flight Heritage	5%	4	4	1	2
Range	10%	5	5	2	5
Accuracy	25%	4	4	3	5
Ground-Testable	20%	5	5	5	5
Adaptability	8%	5	1	5	1
Speed	10%	4	3	3	3
Overall Score	100%	90.4	86.2	67.2	76.0

All trade studies categories were graded on a scale of 1-5, with 1 being the lowest and 5 being the optimal score. As seen in the trade study in the Appendix, RF was chosen because of the combined score compared against the other methods. RF can be used in space, thus meeting the mission context and tied GPS for the highest flight heritage among the four major communication options studied. Also, RF provides the option for off-the-shelf components to be acquired, reducing cost and increasing ease of implementation. Based on the results on the trade study conduced in figure 2, RF Ranging won with a score of 90.4, The next closest was GPS with a hybrid method and image processing coming in a distant third and forth, respectively. Although GPS was considered as the primary ranging method due to relative simplicity and heritage in Earth orbits, it is not currently a viable method for GPS-denied environments, such as interplanetary missions. Thus, RF was chosen and the project requirements were built around this selection. Trade studies were also conducted for conceptual design in both software and testing. For software, SDR vs Integrated radio methods were investigated, as well as different methods of range finding and position determination techniques. Finally, multiple test methods were considered ranging from using cars to balloons.

Table 3 shows the pros and cons for the Software Defined Radio (SDR) and the Integrated Radio. SDRs use software to replace much of the hardware from the typical radio circuitry, such as frequency mixers, signal filters, wave-detection, signal amplifiers, and modulators. Integrated radio, on the other hand, uses more hardware and generally requires a greater knowledge of RF engineering to design.

Table 3: Pros and Cons of SDR and Integrated Radio

SDR or Integrated	**Statement**	**Pro**	**Con**
SDR	Requires external computer		X
	Signal latency (This is not acceptable for some ranging methods of RF)		X
	Variability in turning allows flexibility as project develops	X	
	Hardware choices remain a complex decision		X
	Need knowledge of digital signal processing		X
	Larger packages		X
	No way to measure signal strength if strength algorithms are used		X
Integrated	Circuitry more complex		X
	Smaller packages	X	
	Communication frequency fixed		X
	Relatively cheap	X	
	Code to interact with transceiver relatively simple	X	
	Signal processing happens through hardware	X	

Despite the results of the study, the SDR was selected. SDR's provide greater flexibility and required a software learning curve, as opposed to a hardware learning for integrated radio. These attributes made the use of SDR's more feasible than integrated radio for a team with little RF experience. Lastly, the ability to take phase measurements (the chosen position determination technique) is far simpler on an SDR.

Next, the table below trades the multiple range finding techniques. One-Way Range Finding involves sending a signal from one receiver to another and then uses highly accurate synchronized clocks to determine the time of arrival. Two-Way Time Transfer requires that each both the Sensor and Beacon has transmitting and receiving antennas. This method allows for a higher accuracy with a lower precision clock and eliminates the need for clock synchronization. Received Signal Strength Indication (RSSI) uses a well characterized antenna to transmit a signal which is then received by the “chase" vehicle. While this method will readily provide the magnitude of distance between the two objects, it cannot give the direction vector without additional receivers or vehicles.

Table 4: Pros and Cons of Range-finding Methods

Range Finding Technique	**Statement**	Pro	Con
One-Way Range Finding	Inexpensive	X	
	Simplicity	X	
	May require complex math		X
Two-way Time Transfer	Flight history	X	
	Accurate	X	
	No codependent synchronization necessary	X	
	More complex to implement than one-way		X
	Need two transmitters		X
	Reliable	X	
RSSI	Accuracy issues due to signal fading		X
	May require complex software		X
	Cheap	X	
	Established history	X	

A trade matrix was constructed with the aforementioned range finding methods. Following the results, two-way range finding was selected as the optimal range finding method. Because of this choice, the Sensor will need at least four antennas and the Beacon will need at least one. As seen later in this report, the final configuration ended up with 6 and 2 antennas on the Sensor and Beacon, respectively. Two-way range finding will provide an accurate method to calculate the relative range and simultaneously meet the project requirements.

The next trade study conducted was used to establish the Position Determining Technique. Multiple techniques were examined, varying greatly in their cost, simplicity, timing accuracy, and hardware requirements. The Kinematic Range Comparison Method uses multiple receivers and transmitters mounted on a single frame with predetermined rigid body components. However, this method tends to have high interference and more expensive hardware. The next method analyzed was Time Distance of Arrival which measures the distance in the arrival time of one emitted signal at multiple antennae on the chase vehicle. This method relies heavily on accurate timing between the vehicles which is difficult to implement given our budget. The following methods, Phase Interferometry (PI), Pseudo-Doppler Direction Finding (PDDF), and the Watson-Watt (Adcock Antenna) technique all utilize amplitude and phase comparison to calculate a direction solution. PI and PDDF both use phase comparison while the Watson-Watt technique uses amplitude comparison which is less reliable than phase comparison. Comparing the two techniques that use phase comparison, PDDF requires a much more complex software solution while PI has a faster software rate and a higher accuracy.

Table 5: Pros and Cons of Position Determining Techniques

Position Determination Technique	**Statement**	Pro	Con
Kinematic Range Comparison	More expensive		X
	Simplicity	X	
	High interference		X
TDOA	Accuracy tied to distance between antennas		X
	Estimation based		X
	Flight history	X	
Phase Interferometry	Phase comparison is less affected by interference	X	
	Requires omni-directional antenna		X
	Simultaneous sampling requires more processing complexity		X
	Simultaneous sampling takes less time	X	
PDDF	Omnidirectional antenna required		X
	Sequential sampling takes more time		X
	Estimation based		X
	Resistant to interference	X	
Watson-Watt (Adcock Antenna)	Amplitude comparison not accurate for close antenna spacing	X	
	Requires directional antenna		X
	Angular resolution increases with number of antennas		X

A trade matrix, found in the Appendix, analyzes the previously mentioned positioning techniques needed to find the exact position of the Beacon. Phase interferometry (PI) was selected for the final design. The reason for this is because it combines high accuracy with a high solution rate. With an overall score of 4.1, PI was the clear winner and thus selected.

An additional consideration for the initial design was the hardware selection. Because of the design requirements set forth by General Atomic, several hardware components had no flexibility in choice. The largest hardware selection that required a trade study was the type of antenna used for the radio frequency communication. The following table is a study conducted about the type of antenna needed and fueled the final design of the antenna and Beacon packages.

Table 6: Pros and Cons of Antennas

Antennas Selection	**Statement**	Pro	Con
Clover Leaf Antenna	Inexpensive	X	
	Mounting		X
	May move configuration during testing		X
Patch Antenna	Inexpensive	X	
	Accurate	X	
	Mounting	X	
Dipole Antennas	Accuracy	X	
	Mounting	X	
	Expensive		X
	Established history	X	

As shown above, the dipole antenna won the trade by a score of 4.5 over the patch and clover leaf. In the final design, the dipole antennas will be used because of their accuracy and ease in placement and mounting on the packages. The issue with the patch antenna is the size is too large. The antennas need to be be placed half a wavelength apart to correctly get the range and position. Therefore, the patch could not be accurately enough placed due to their size relative to the small Beacon and Sensor packages. In addition, clover leaf antennas have a very specific orientation they have to be in for accuracy. If the antenna moves, the position can not be correctly obtained. So, despite their price, dipole antennas make the most sense for the final design.

A wide variety of different testing methods were considered: un-motorized and motorized balloons, drones, cars, RC cars, and a custom track. The main pros and cons for testing related to the cost and the effectiveness of the test. An un-motorized balloon would be cheap but also provide little to no maneuverability. A motorized balloon would be better for testing purposes but would be significantly more expensive. A drone is along the same lines as a motorized balloon with a very high expense, but good maneuverability. Using personal vehicles would not be expensive but it is difficult to control the speed precisely to less than a km/s. An RC car is highly maneuverable and low cost. A custom track would be useful for repeating test but the actual construction of a track that is over 100 meters would be difficult and costly and bring along multi-path concerns.

Table 7: Pros and Cons of Proposed Testing Methodologies

Test Methodology and Fixture	**Statement**	Pro	Con
Un-Motorized Balloon	Cheap	X	
	High weight capacity	X	
	Little to no maneuverability		X
Motorized Balloon	Expensive		X
	High weight capacity	X	
	Requires additional development		X
Drone	Expensive		X
	Low weight capacity		X
	Highly maneuverable	X	
Car	Almost no cost	X	
	Highest weight capacity	X	
	Difficult to control speed at very low speeds		X
RC Car	Maneuverable	X	
	Relatively low cost	X	
	Relatively high weight capacity	X	
Custom Track	Requires additional development/cost		X
	Modular	X	
	Repeatability increases	X	

Seen above is a trade of the various testing methods. As seen in the results, the car is the clear winner with an overall score of 4.5. While it might not be the most accurate due to the uncontrollable nature of driving, it provides a low cost and easily implemented solution. In addition, most other methods would be costly and be needed to be preformed indoors. In doing this, the RF signals bouncing off the structure becomes a big discern for a project that requires precise signal acquiring. Mounting the Beacon and Sensor on a car makes the most since practically and physically and thus will be implemented for final testing.

In conclusion, all trades studies were conducted in order to best meet the project requirements. The final design, in further detail below, is the result of the research conducted. The SDR was selected because of its low cost and is more robust than the integrated radio. Two-way range finding was selected for its low cost solution that doesn't sacrifice accuracy. In addition, position will be found using phase interferometry for its low uncertainty and high solution rate. To test the validity of all these methods, the Beacon and Sensor will be mounted on cars for the freedom this entails along with saving money. An in depth analysis of all trade studies can be seen in the Conceptual Design Document.

4.2 Design Requirements

Author: Emily Webb

The requirements development process was a lengthy one and they continued to update through Critical Design Review. The six fundamental requirements have not changed since their initial revisions with the client on 13 September. From there, the team wrote design requirements to further specify the design.

FR 1 The Beacon shall transmit RF signals between 2 and 4 GHz (S-Band).
Rationale: The Beacon will transmit radio frequencies in the S-band for legality and accuracy purposes. RF as a method for determining range and heading was determined through trade studies conducted early on in the project history.

DR 1.1 The Beacon shall have one or more antennas to transmit RF signals.
Rationale: Antennas are needed to transmit and receive RF signals.

DR 1.1.1 The Beacon antenna(s) shall have a resonant frequency between 2 and 4 GHz.
Rationale: The antennas should be optimized for the project's chosen wavelength.

DR 1.2 The Beacon and its testing equipment shall be constructed from materials transparent to RF.
Rationale: RF transparent materials will not interfere with transmissions from the Beacon and will therefore not contribute errors to the position calculations.

FR 2 The Sensor shall receive RF signals transmitted by the Beacon.
Rationale: The Sensor and Beacon must be able to communicate with one another to determine the Beacon's position relative to the Sensor.

DR 2.1 The Sensor shall have one or more antennas to receive RF signals.
Rationale: Antennas are needed to transmit and receive RF signals.

DR 2.1.1 The Sensor antennas shall have a resonant frequency between 2 and 4 GHz.
Rationale: The antennas should be optimized for the project's chosen wavelength.

DR 2.2 The Sensor and its testing equipment shall be constructed from materials transparent to RF.
Rationale: RF transparent materials will not interfere with transmissions from the Beacon and will therefore not contribute errors to the position calculations.

FR 3 The Sensor shall track relative position of the Beacon within a 2% error bound 66.7% of the time (1σ).
Rationale: The original 2% error bound on this requirement was defined by the customer. This was originally intended to be a 3σ error bound, but after investigation into timing errors, the team discovered they would not be able to meet this bound with equipment available. The requirement was revised and deemed feasible with the 1σ error bound.

DR 3.1 The Beacon's true position relative to the Sensor shall be known to within 1 m in every direction.
Rationale: The Beacon's true relative position should be known more accurately than the desired position calculation accuracy for optimal comparison between “truth data" and calculated data.

DR 3.1.1 The true relative position of the Beacon to the Sensor shall be known at a rate no slower than 1 Hz.
Rationale: The position calculation rate boundary of 1 Hz is a balance between feasibility and accuracy. As the rate increases, so does the processing cost. However, given expected relative speeds in satellite clusters, the system will be able to determine position accurately at this given rate.

DR 3.2 The Sensor shall compute the position of the Beacon to 2% accuracy in x,y, and z axes 66% of the time.

Rationale: The Sensor should know the position of the Beacon to within 2% accuracy in each axis for the most accurate position data. The Beacon's position will be reported in Cartesian coordinates relative to the Sensor.

DR 3.2.1 The Sensor antennas shall be spaced at half a wavelength apart.
Rationale: The antennas are spaced as such to eliminate ambiguity in heading calculations.

DR 3.3 The Sensor shall compute position data at a rate no slower than 1 Hz.
Rationale: The position calculation rate boundary of 1 Hz is a balance between feasibility and accuracy. As the rate increases, so does the processing cost. However, given expected relative speeds in satellite clusters, the system will be able to determine position accurately at this given rate.

FR 4 The tracking system shall alert the operator if the satellite drifts more than 10% of the range away from a predetermined notional path.
Rationale: The customer asked that the system be able to notify the operator/ground systems if a satellite drifted from its expected course by some amount. The 10% bound was suggested by the customer given their experience with similar alert systems.

DR 4.1 The Sensor shall receive the expected path of the Beacon from a ground station.
Rationale: To determine whether the Beacon has deviated from an notional path, the Sensor will need to compare the position of the Beacon to its expected position along this path. The system is not responsible for computing the Beacon's expected path, as in the cooperative cluster context, ground systems should know the expected position based on orbital parameters.

DR 4.2 The Sensor shall communicate with the ground station using CAN Serial protocol.
Rationale: This was a customer defined requirement. General Atomics frequently uses CAN to interface between instruments and their spacecraft buses.

FR 5 The Beacon system shall operate between temperatures of -20°C to 50°C, inclusive.
Rationale: This was a customer defined requirement based on the temperature changes a satellite would undergo in an Earth-centric orbit.

DR 5.1 The project shall provide a heater for electronics/components whose operable temperature range has a minimum greater than -20°C.
Rationale: If any components are not able to operate within the desired temperature range because they are too cold, the project will provide a heater to keep those components within their operable temperature range so the project can function as desired.

DR 5.2 The project shall provide a cooling mechanism for electronics/components whose operable temperature range has a maximum less than 50°C.
Rationale: If any components are not able to operate within the desired temperature range because they are too warm, the project will provide a cooling mechanism to keep these components within their operable temperature range so the project can function as desired.

FR 6 The Beacon system shall be powered by a 28±6 V unregulated power supply.
Rationale: This was a customer defined requirement based on the power that could be supplied from their satellite buses.

DR 6.1 The Beacon shall be powered by a battery.
Rationale: The Beacon must be powered in a fashion such that it is mobile for relative motion tests.

DR 6.2 The Sensor shall be powered by a battery.
Rationale: The Sensor must be powered in a fashion such that it is mobile for relative motion tests.

DR 6.3 The power for the Sensor components and electronics shall be regulated.
Rationale: To avoid frying any electronics and to meet strict power requirements for some sensitive components, the power supply will need to be regulated.

DR 6.4 The power for the Beacon components and electronics shall be regulated.
Rationale: To avoid frying any electronics and to meet strict power requirements for some sensitive components, the power supply will need to be regulated.

4.3 Basic Design Overview

Authors: Jarrod Puseman, Nate Lee, & Corey Huffman

Given the requirements and FBD given above, this section outlines the top-level design for our system and how it satisfies this design. At large, the design consists of a Beacon package and a Sensor package. These are shown conceptually in Figure 3.

Figure 3: Beacon and Sensor Packages

Each of these packages uses the same structural housing, as described later in the Detailed Design section. All together, each of these housings without antennas is 6" x 6" x 6". The mass of the Beacon is about 2.036 kg, while the Sensor will have mass of about 2.390 kg. This is from the additional SDR and supporting antennas. With the antennas on top of the packages, the Beacon reaches to just over 10 inches tall. The Sensor antennas bring its height up to 12.3".

4.3.1 Subsystems Overview

The major components/subsystems are shown in Figure 4. The figure points out the locations of the components in each part of the project. The components of interest are the Lime SDRs, the Raspberry Pi for processing, the PiCAN 2 to achieve CAN protocol, the GPS with data logger and the custom power distribution board.

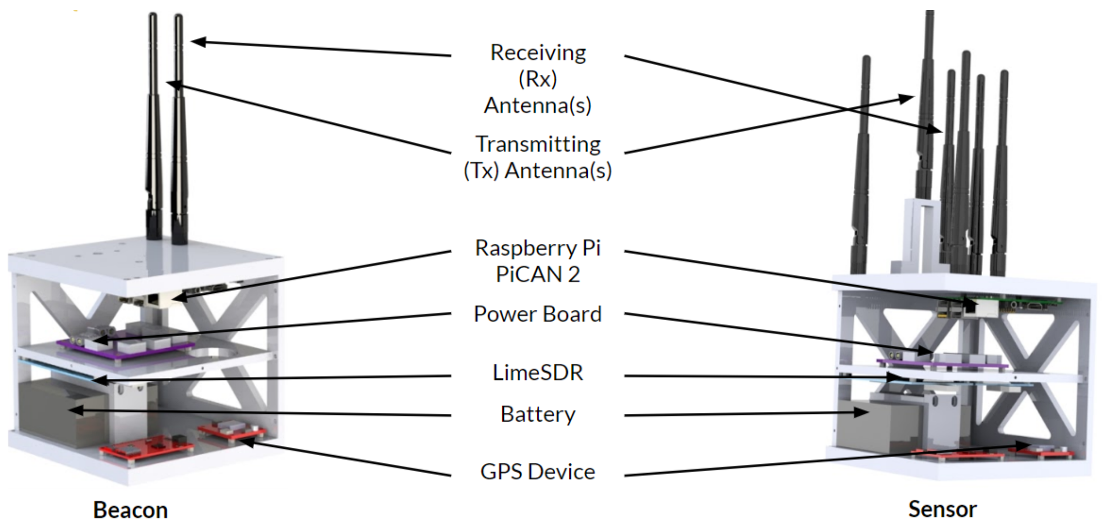


Figure 4: Notable Subsystems to the Project

Recall from the concept of operations and functional block diagram that the project is to build a tracking system for tracking the relative motion of a Beacon. To achieve this, the Sensor will send out probing signals to which the Beacon will reply. The Sensor will then receive the Beacon reply and use information about it to compute the relative position of the Beacon.

Specifically, the Sensor will use time of flight measurements (described below) on the probing signal to the reply to compute the range of the Beacon. At the same time, a single SDR will be multiplexing an array of four antennas to find the difference in phase of each received antenna. These phase differences can be used to solve a geometry problem to compute the heading (see phase interferometry below). With both the heading a range known, the relative position is thus computed. To accomplish this, the Lime SDR was selected to handle the RF communication aspect. This choice simplifies the amount of RF engineering that the team must do, and it has all the capabilities that we need from it. The Lime SDR can communicate at the specified 2.4 GHz range, has room for 6 RX antennas, 4 transmit antennas, and USB communication with a computer. Additionally, it is easily adjustable as a radio to make troubleshooting unforeseen issues less difficult than a traditional integrated radio. The Lime SDR communicates with most open-source SDR software packages, including Pothos, GNU Radio, and SoapySDR, some of the more popular options.

Because most SDRs communicate via universal serial bus (USB), the processing unit for both the Sensor and the Beacon was selected to be the Raspberry Pi 4. This particular model is capable of USB 3.0 protocols and speeds (as well as the SDR) to minimize communication time. The Raspberry Pi 4 is essentially a small computer with it's own Linux-based operating system, so it will be able to run python scripts to execute the Beacon and Sensor functions.

In order to make sure these subsystems are running as stand alone systems, a power board (described in greater detail below) was designed to handle the down conversation from the 28$\pm$6 V as defined by FR 6. This power distribution can be managed through switches on the board to help with testing and debugging. The board design can be seen in Figure 5.

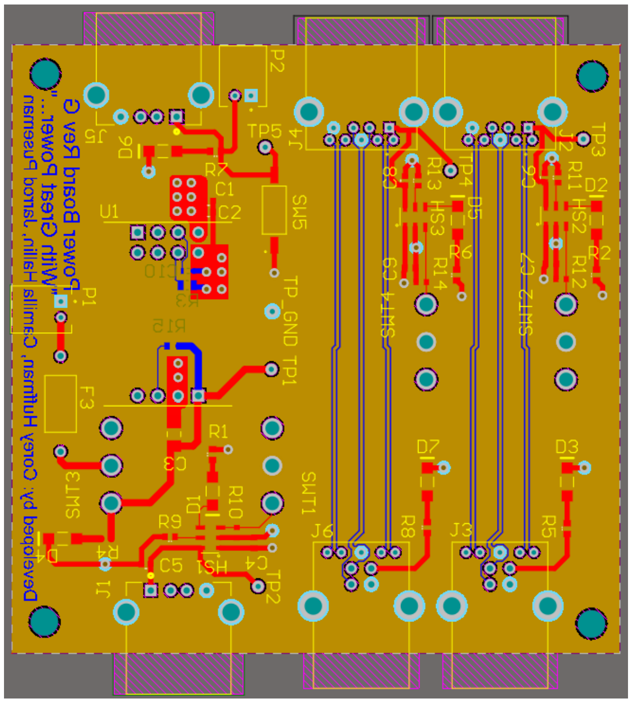


Figure 5: Power board design

4.3.2 Data Flow Diagrams

To understand how the project works at large, consider the data flow diagrams for each system. Shown in Figure 6 the data flow diagram for the Beacon, and Figure 7 the data flow diagram for the Sensor.

Beacon Data Flow

As aforementioned, the Beacon receives and replies to a probe signal from the Sensor. This process is outlined in Figure 6. To read this image, begin in the top left at the Sensor probe signal. This signal is in the form of RF communication at 2.4 GHz. This signal will get picked up by the Beacon receiving antenna and electrically picked up on the Lime SDR. On the SDR, the signal gets digitized according to code on the SDR. From here, the Lime SDR will packetize the code for communication to the processing unit in the Raspberry Pi 4.

The SDR then sends these data packets to the Raspberry Pi 4 via USB. The Raspberry Pi 4 then compares every data packet delivered to it against the pre-determined signal from the Sensor. When it finds a match, it then sends its unique reply signal to the SDR via USB to transmit in every direction.

The SDR receives this packetized information, converts it into the signal that needs to be communicated, and then mixes and transmits it out the transmit antenna. Note that this antenna is separate from the receiving antenna due to the dedicated Tx and Rx channels on the Lime SDR.

The data loop at the bottom of Figure 6 demonstrates how the Beacon will handle GPS data. Note that the GPS data is not part of the final Beacon product in the context of inter-satellite tracking. The GPS data will be used in Team ATOMIC's ground-based testing to verify functionality of the system. To do this, the true absolute position (GPS) of both the Sensor and Beacon are recorded during testing. After testing, the relative position between the two units will be compared against the data computed via the RF hardware.

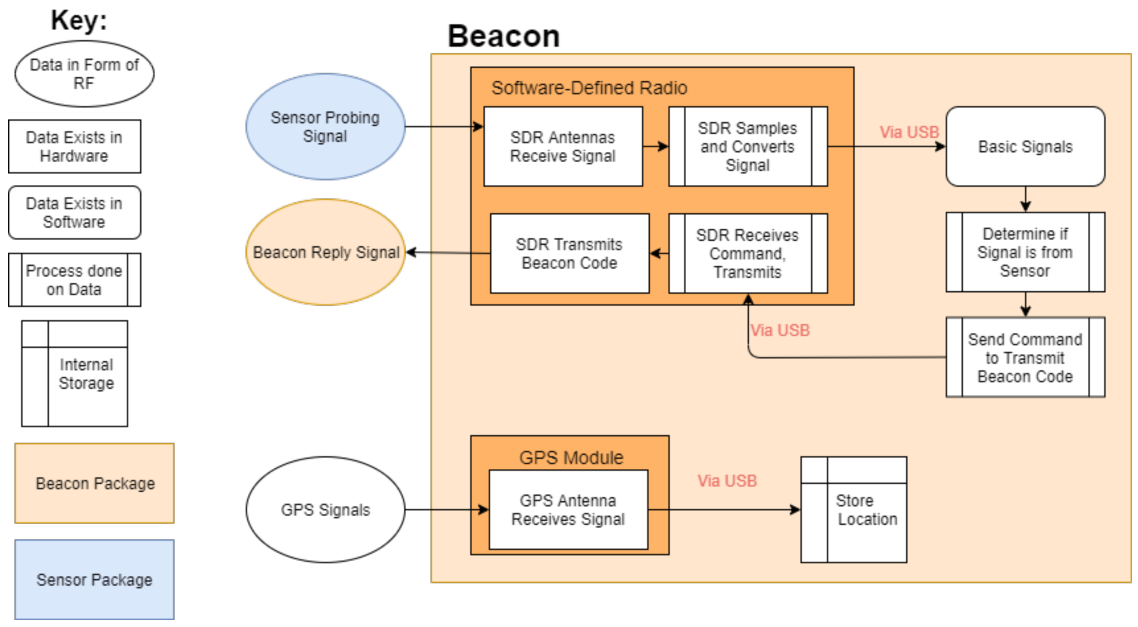


Figure 6: Beacon Data Flow Diagram

To achieve this, the Beacon (and Sensor) simply has a GPS module and antenna. The module will continually (up to 8 Hz) compute its position on the globe and send this to a data-logging device via I2C protocols. The data logger merely receives this GPS data and writes it to a microSD card, which can be retrieved and put into a computer for post-processing.

Sensor Data Flow

The flow of data through the Sensor is fairly similar to that of the Beacon, but more complicated. This is depicted in Figure 7.

To study the flow of data in the Sensor, begin again in the top left corner of Figure 7. The Beacon reply signal is being transmitted through the air or space in RF communication. The Sensor will receive this signal on 5 separate antennas (discussed below in Section 6 to the report). One antenna is used to measure the time of flight (this antenna is connected to the same SDR sending out the probing signal), and the other four are multiplexed to get the phase difference between each antenna. These numbers are similarly communicated the the processing unit, a Raspberry Pi 4 via USB.

The Raspberry Pi 4 will then store these measurements of time of flight and each pair of antennas' phase difference. When 30 measurements are made, the Raspberry Pi will then average all of the time of flight measurements and each pair of antennas' phase differences. With these new "measured" values for the time of flight and phase differences, the Sensor will compute the relative position of the Beacon in spherical coordinates, as detailed in Section 6 of this report. At this point, another simple conversion from spherical coordinates to Cartesian is performed.

This relative position is then fed into a Kalman filter to achieve better accuracy of the position estimates. This filter essentially uses knowledge of the dynamics of the system to predict the state at a certain time, and it updates this prediction according to the measurements from the antennas. This filter is described more in Section 6 as well.

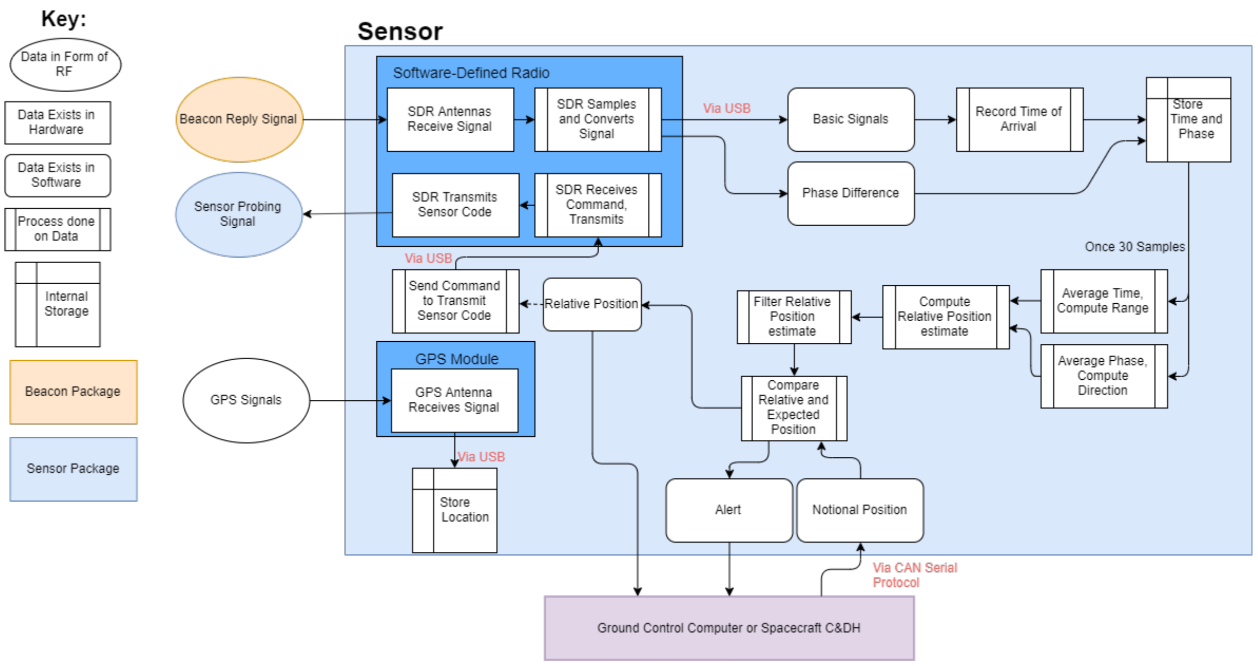


Figure 7: Data Flow Diagram for the Sensor

After the filter produces a state estimation, the Sensor then prompts the flight computer for a nominal path and position. This is, the flight computer will have knowledge of where the Beacon should be at any instant in time. When the Sensor prompts the flight computer for the position, the computer will provide the appropriate expected Cartesian position of the Beacon back to the Sensor. the Sensor then compares the expected position to the measured position. If the Beacon's measured position and the expected position are different by more than 10.0 meters, the Sensor will then alert the flight computer that the Beacon has strayed from its path. The Sensor then prompts the flight computer if it would like to know the computed position. If the computer does, then the Sensor reports the computed position to the flight computer.

With this loop complete, the Sensor goes back to probing for the Beacon and collecting replies from the Beacon, thus beginning the entire process over again, with the Sensor probe signal being sent to the SDRs via USB to be mixed and transmitted.

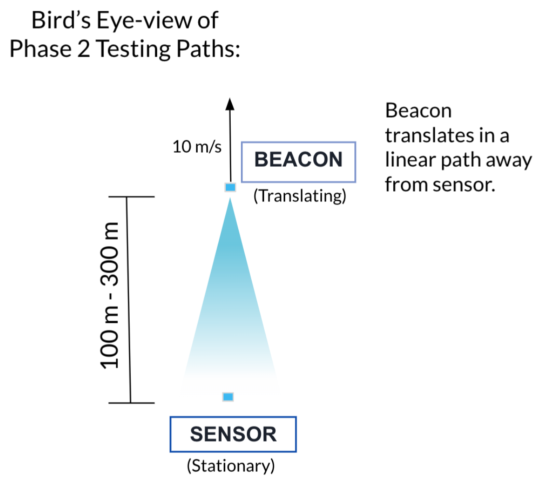
Just as in the case of the Beacon, the Sensor has a GPS data-logging loop separate from all other processes on the Sensor. The GPS unit will continuously (again up to 8 Hz) log the position of the Sensor on the globe. After the tests are run, then the truth data for the Sensor and the Beacon can be combined and the true relative position computed.

4.3.3 Basics of Testing

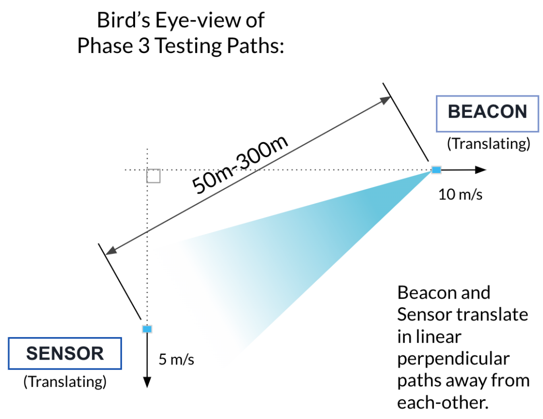
Testing is the main method by which the CPEs will be satisfied. The testing needs to show that all the components will function together properly, while meeting the CPEs. To accomplish this, the testing will be divided into six main phases: Preliminary, Phase 1: Stationary, Phase 2: Single Motion, Phase 3: Linear Cluster Motion, Phase 4: Non-Linear Cluster Motion, and Thermal Testing. The Phase 2-4 top view diagrams can be seen in figures 8a-9. The Phases 2 and 3 diagrams show the build up of testing as it gets more complex. The Phase 4 figure shows the path of the Beacon and the Sensor, along with the representation of the dynamics of the test. The Preliminary testing is designed to show that all of the components are able to operate individually, then together in the sub-systems of the Beacon and the Sensor. Phases 1,2, 3, and 4. of the testing are designed to meet the different levels of success. The thermal testing is designed to satisfy FR5. These levels include stationary ranging and positioning, Beacon moving ranging and positioning, and Sensor and Beacon moving ranging and positioning. Together, the six phases of testing satisfy the CPEs, and allow team ATOMIC to meet the different levels of success.

At a base level, the team does not have much experience with RF components or methodology. The Preliminary levels of testing will allow for the team to become more familiar with the concepts surrounding RF components while checking functionality of the purchased parts. The different levels of the Preliminary testing will allow for a gradual build-up of knowledge, instead of trying to build the final product from the start. This will allow the team to build up a knowledge base on replaceable parts. Not only does this reduce risk of damaging the final product, it ensure that proper handling procedures can be learned before the Sensor and Beacon packages are built.

The general testing levels, Phases 1-4, maintain the same buildup as the Preliminary testing. Instead of starting with the test that will satisfy all the CPEs, the team will start with a lower level test. This test, Phase 1, will show that the Beacon and Sensor can find each other while stationary. This test will show basic functionality and allow the team to meet a level of success before moving onto tests involving motion. Furthermore, the stationary test allows for troubleshooting before introducing the more complex motion tests, Phases 2-4. As each successive test is completed, higher levels of success are checked off. Successful completion of Phase 4 testing will result in the team meeting all of the levels of position finding success.



(a) Phase 2 Test Path



(b) Phase 3 Test Path

Figure 8: Birds -Eye View of Phases 2 and 3

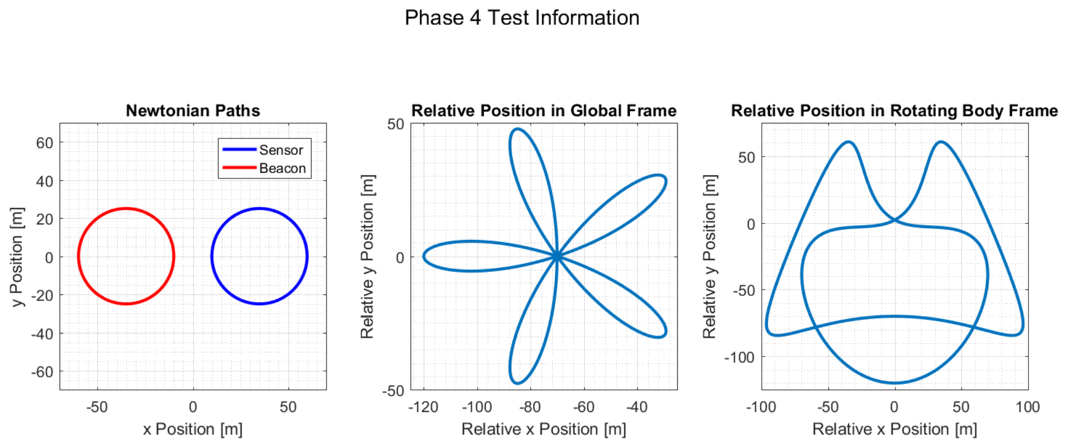


Figure 9: Phase 4 Test Paths

4.4 Software Design

The design below describes the software to be running on the Sensor package. The software on this package aims to compute the relative position of the Beacon from time and phase measurements. This design addresses FR3 and FR4 and their respective design requirements.

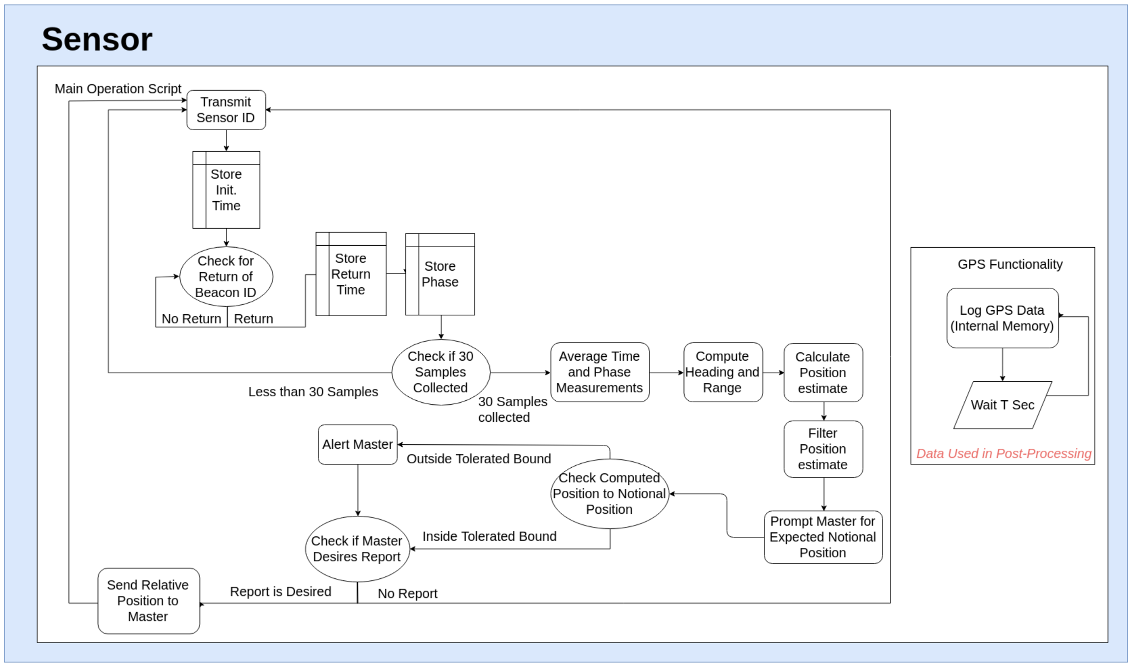

Sensor
Main Operation Script
Transmit Sensor ID
Store Init. Time
Check for Return of Beacon ID
No Return
Return
Store Return Time
Store Phase
Check if 30 Samples Collected
Less than 30 Samples
30 Samples collected
Average Time and Phase Measurements
Compute Heading and Range
Calculate Position estimate
Filter Position estimate
Prompt Master for Expected Notional Position
Check Computed Position to Notional Position
Outside Tolerated Bound
Inside Tolerated Bound
Alert Master
Check if Master Desires Report
Report is Desired
No Report
Send Relative Position to Master
GPS Functionality
Log GPS Data (Internal Memory)
Wait T Sec
Data Used in Post-Processing

Figure 10: Data flow diagram for software approach

Important software elements that the software design addresses are: averaging time and phase measurements, computing heading and range, computing a position estimate, and filtering the position estimate. In addition to providing the necessary modules for computation of range and heading, the flight software will also provide the architecture needed for the Raspberry Pi to communicate with peripherals and handle scheduling.

4.4.1 Time of Flight Measurements

Obtaining time of flight measurements is a major component of this project since the calculations for position heavily rely on accurate time of flight. In order to ensure accuracy, the signal from the Beacon to the Sensor will be encoded with a unique reply sequence. The Sensor will receive that signal and save the time of arrival. Then, the range can be calculated using the total time it takes for the signal to travel from the Sensor to the Beacon and back to the Sensor. There will be processing times during that process, so those will be subtracted out. The following equation shows how the range will be calculated:

$$R = \frac{c}{2}\left(\Delta t_{total} - (t_{receive,1} + t_{receive,2})\right) \tag{1}$$

where c is the speed of light, Δt_{total} is the total time the signal takes going from the Sensor to the Beacon and back, and $t_{receive,1}$ and $t_{receive,2}$ are the processing times. For post-processing, the estimated relative position calculated using developed algorithms will be compared to the true position. In order to implement this comparison, the estimated relative position and the true position's time stamp must be compared and synchronized. This will be done after data collection

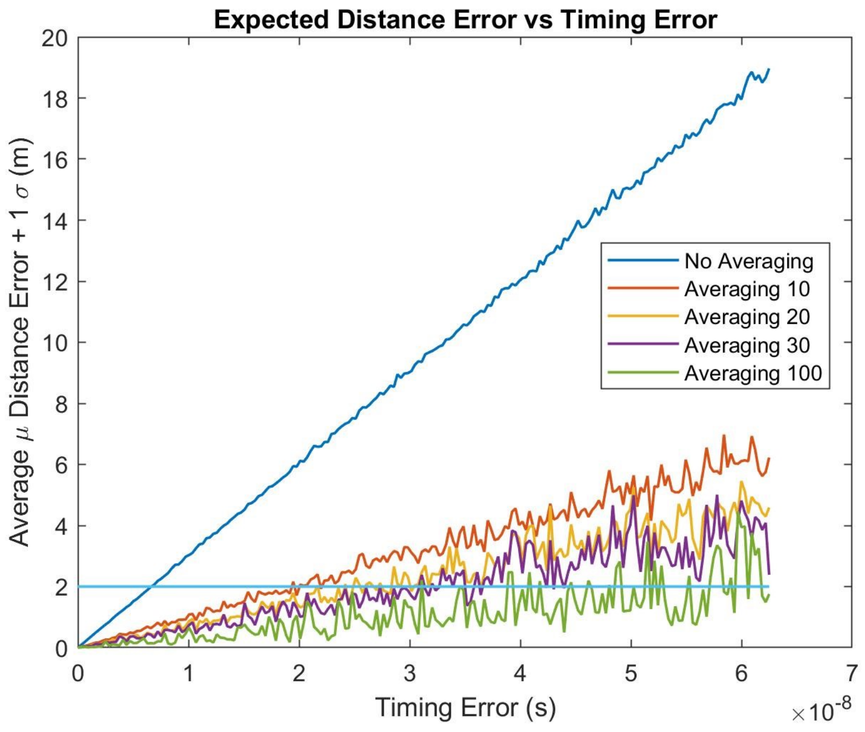


Figure 11: Expected distance error vs timing error

In order to determine the accuracy of this method for range, a Monte Carlo simulation was run using a calculated range. These simulations determined the maximum allowable timing error. In the figure below, each test was run using a different number of samples for averaging, from no averaging to 100 averaged points.

Figure 11 shows how the distance error increases dramatically with no averaging, but with each increase in the averaging scheme, the error decreases. The light blue horizontal line is the accepted error value for a 100m range. The best case scenario was calculated to be 30 averaged data points with a maximum allowable timing error of 36ns and a minimum clock speed of 30 MHz.

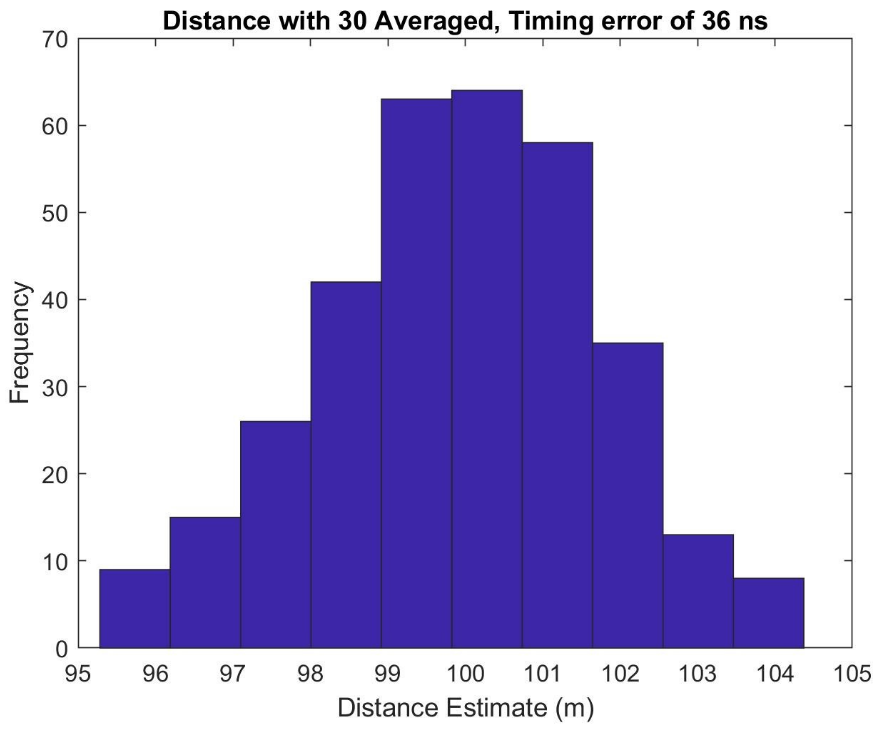


Figure 12: Monte Carlo for Range

Figure 12 shows the Monte Carlo simulation for that particular averaging scheme with 30 averaged data points. For this simulation, the mean is 99.9m with a standard deviation of 1.82m which meets the 2% error requirement. This yields a timing error of 36 ns which is feasible with the given components.

4.4.2 Phase Interferometry

Phase interferometry utilizes a difference in phase between antennas to determine the heading of the Beacon. First, it is assumed that the incoming signals are parallel to each other. This is be a satisfactory assumption as long as the distance between the antennas are much smaller than the range between the Beacon and the Sensor. In the case of ATOMIC, the antennas are separated by approximately .1 m, and the Beacon and Sensor are separated by approximately 100m. This is an order of 10^3 difference, so the assumption is a good one. Only if the Sensor and Beacon are less than a meter apart would this assumption cause significant error to the model. When this is the case, the satellites or vehicles on which the sensor and beacon are mounted will already have collapsed. To perform phase interferometry, a phase difference is determined between the antennas and the reference antenna for an incoming signal. For 3D tracking, this method requires 4 antennas. Using trigonometry, the following equations can be derived for azimuth (ψ) and elevation (θ), respectively:

$$\psi = tan^{-1}\left(\frac{\Delta\phi_{13}d_{14}}{\Delta\phi_{14}d_{13}}\right) \tag{2}$$

$$\theta = sin^{-1}\left(\frac{\Delta\phi_{12}\lambda}{2\pi d_{12}}\right) \tag{3}$$

where $\Delta\phi$ corresponds to the phase difference between the reference antenna (A1) and the other antennas (A2, A3, or A4). d corresponds to the distance between the antennas and λ is the wavelength. An illustration is shown below for representation:

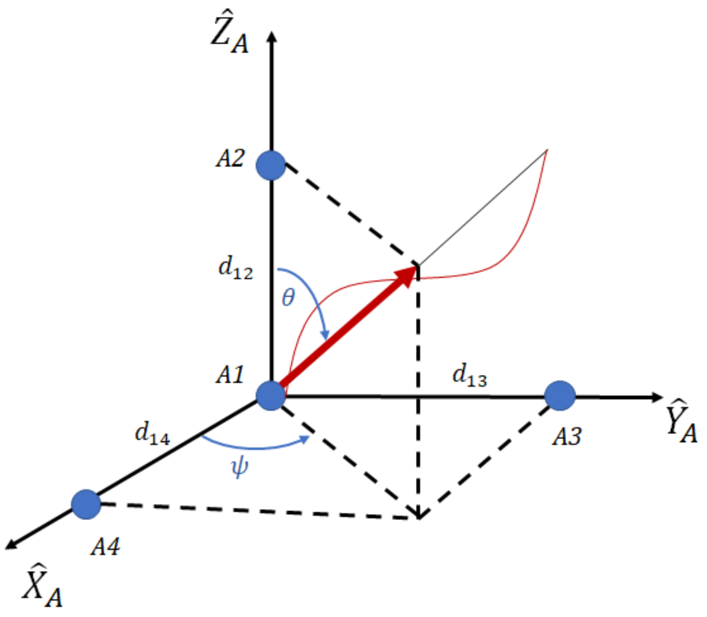


Figure 13: Phase interferometry diagram

In Figure 13, the four blue dots represent the antennas places in an orthogonal orientation to form coordinate axes, and the red vector represents the heading to the Beacon from the Sensor antenna origin. This is the path along which received signals from the Beacon travel, and the direction the Sensor is using phase interferometry to compute.

Then to find the position, a simple spherical coordinates to Cartesian coordinates transformation is utilized:

$$\vec{r} = \langle R\sin(\theta)cos(\psi), R\sin(\theta)\sin(\phi), R\cos(\theta)\rangle \tag{4}$$

where R is the range between the Sensor and the Beacon. This transformation is shown in Figure 14 below:

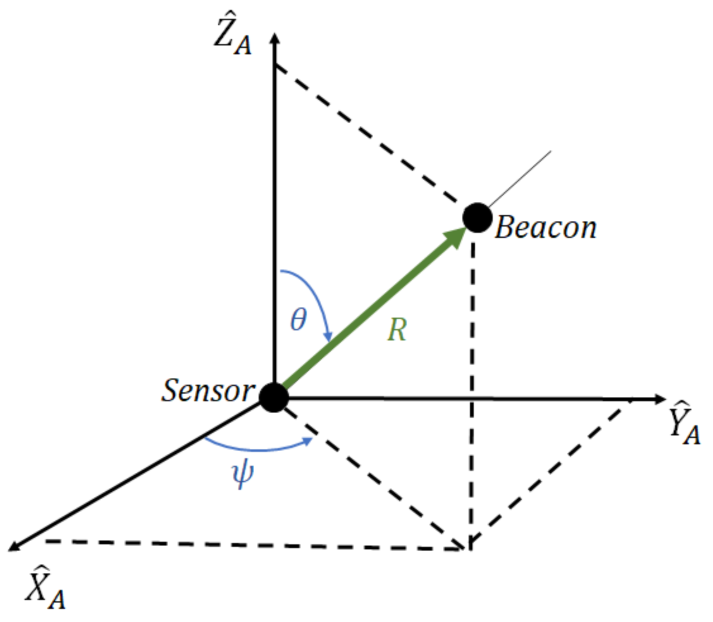


Figure 14: Coordinate Transformation

This method is the best for the given project since it reduces timing error using a unique reply sequence in the signal. Additionally, the antennas can be bought at relatively little expense and the calculations performed to get the relative position do not take an extreme amount of computational power.

A Monte Carlo simulation was also run for the heading much like for the range. This demonstrates feasibility with the phase error. Averaging 20 data points yielded a mean position error of 0.91m with a standard deviation error of 0.7m. This resulted in 91.4% of the cases within 2m which meets the 2% error requirement as set by Functional Requirement #3. Figure 15 shows the results from that Monte Carlo simulation. The next figure (Figure 16) shows a plot of the results from that simulation. As seen in the plot on the right, the position is shown with the error bound in the blue circle. Only a few cases were outside the error circle, proving feasibility further. The plot on the left shows that the error in computation decreases with the Beacon location. Based on the findings of this Monte Carlo analysis on just the timing measurement, the system would need to average 20 data samples to meet the accuracy requirement specified in Functional Requirement 3.

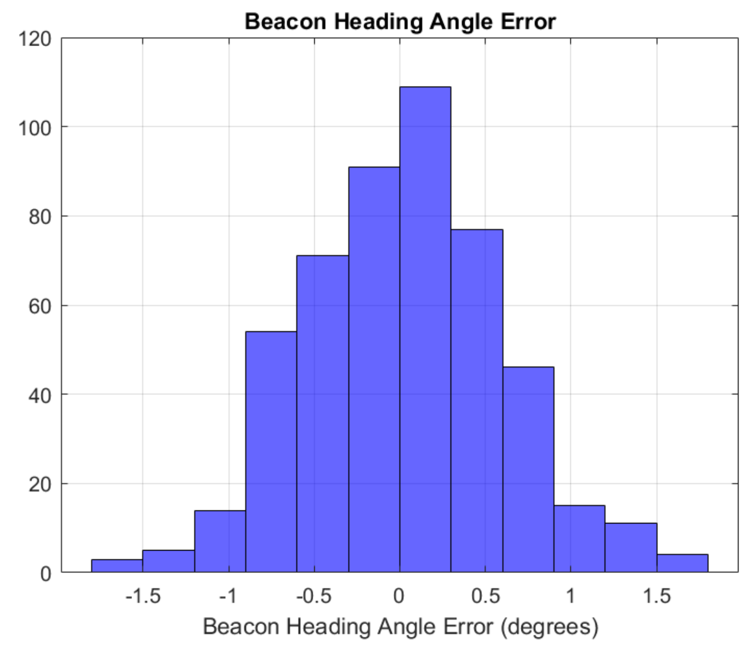


Figure 15: Monte Carlo Heading

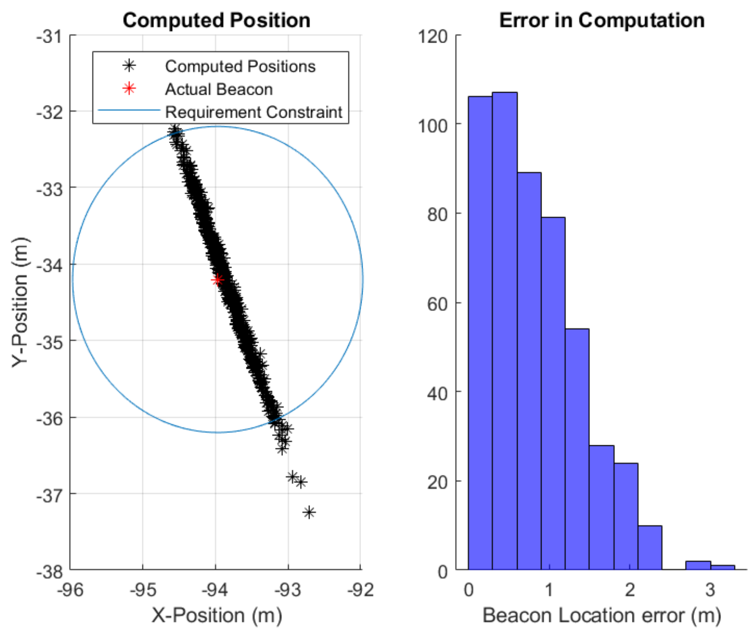


Figure 16: Monte Carlo Heading

4.4.3 Monte Carlo for Range and Position

Authors: Anastasia Muszynski, Camilla Hallin, Brendan Lutes, & Jarrod Puseman

Both the range and position Monte Carlo simulations deemed each method feasible. Then a Monte Carlo was run for both of those methods simultaneously.

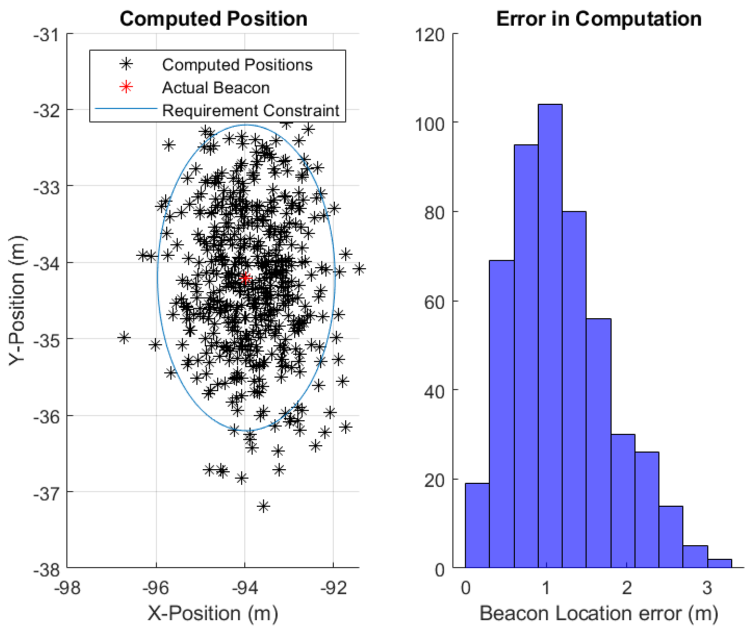


Figure 17: Position and Range Monte-Carlo Results

Results for averaging 20 data points for phase and range measurements, including normalized, random error scaled by 0.1 [rad] resulted in a mean error of 1.18 [m], with a standard deviation of 0.61 [m] 88.6% of cases successfully calculate relative position to within 2 meters, which meets the project's functional requirement, FR3: "The Sensor shall track relative position of the Beacon to within 2% of the true position 66.7% (1σ) of the time." These error estimates we found using a general error sensitivity analysis (the sum in quadrature of the sensitivities of the partial derivatives). The analysis for each part of the model is given here.

Range Equation The range, R is a function of the start time, t_1, end time, t_2, processing time, t_3, and speed of light, c. The errors associated with each of these respectively are denoted as: $\delta R, \sigma_{t_1}, \sigma_{t_2}, \sigma_{t_3}, \sigma_c$

$$R = \frac{c}{2}\left(t_2 - t_1 - t_3\right)$$

$$\delta R = \sqrt{\left(\frac{\partial R}{\partial t_1}\sigma_{t_1}\right)^2 + \left(\frac{\partial R}{\partial t_2}\sigma_{t_2}\right)^2 + \left(\frac{\partial R}{\partial t_3}\sigma_{t_3}\right)^2 + \left(\frac{\partial R}{\partial c}\sigma_c\right)^2}$$

where,

$$\frac{\partial R}{\partial t_1} = -\frac{c}{2}$$

$$\frac{\partial R}{\partial t_2} = \frac{c}{2}$$

$$\frac{\partial R}{\partial t_3} = -\frac{c}{2}$$

and

$$\frac{\partial R}{\partial c} = [t_2 - t_1 - t_3]/2$$

Note that σ_c is very small since the speed of light is well known from many experiments, therefore the term contributing from error in c can be considered negligible.

Azimuth Equation The azimuth is computed using phase interferometry, which measures the phase difference in a signal in pairs of antennas. Using the known geometry of the antennas, the azimuth can be computed using the following equation:

$$\psi = \arctan\left(\frac{\Delta\phi_{13}d_{14}}{\Delta\phi_{14}d_{13}}\right)$$

Now let

$$u = \left[\frac{\Delta\phi_{13}d_{14}}{\Delta\phi_{14}d_{13}}\right]$$
$$\gamma(x) = \frac{d}{dx}\arctan(x) = \frac{1}{1+x^2}$$

So that

$$\frac{\partial\psi}{\partial\Delta\phi_{13}} = \frac{d_{14}}{\Delta\phi_{14}d_{13}}\gamma(u) \qquad \frac{\partial\psi}{\partial d_{14}} = \frac{\Delta\phi_{13}}{\Delta\phi_{14}d_{13}}\gamma(u)$$

$$\frac{\partial\psi}{\partial\Delta\phi_{14}} = \frac{\Delta\phi_{13}d_{14}}{d_{13}}\frac{-1}{(\Delta\phi_{14})^2}\gamma(u) \qquad \frac{\partial\psi}{\partial d_{13}} = \frac{\Delta\phi_{13}d_{14}}{\Delta\phi_{14}}\frac{-1}{(d_{13})^2}\gamma(u)$$

The uncertainties can be modelled:

$$\sigma_{\Delta\phi_{13}} = \sigma_{\Delta\phi_{14}} = .05 \text{ rad}$$
$$\sigma_{d_{13}} = \sigma_{d_{14}} \leq .5 \text{ mm flexion of antennas}$$

Recall the general uncertainty formula

$$\partial\psi = \sqrt{\left(\frac{\partial\psi}{\partial\Delta\phi_{13}}\sigma_{\Delta\phi_{13}}\right)^2 + \left(\frac{\partial\psi}{\partial\Delta\phi_{14}}\sigma_{\Delta\phi_{14}}\right)^2 + \left(\frac{\partial\psi}{\partial d_{14}}\sigma_{d_{14}}\right)^2 + \left(\frac{\partial\psi}{\partial d_{13}}\sigma_{d_{13}}\right)^2}$$

Plugging in the expected values for each of these uncertainties gives an overall approximate azimuth uncertainty of about 1.5 degrees. At 100 yards, this is close to 2.6 m for a single measurement. By averaging several measurements, this uncertainty will go down as demonstrated in the Monte Carlo simulation.

Elevation Equation Computing the elevation of the Beacon relative to the Sensor axes is very similar to the azimuth:

$$\theta = \arcsin\left(\frac{\Delta\phi_{12}\lambda}{2\pi d_{12}}\right)$$

Now let

$$u = \frac{\Delta\phi_{12}\lambda}{2\pi d_{12}}$$
$$\gamma(x) = \frac{d}{dx}\arcsin(x) = \frac{1}{\sqrt{1-x^2}}$$

So that

$$\frac{\partial\theta}{\partial\Delta\phi_{12}} = \frac{\lambda}{2\pi d_{12}}\gamma(u),$$

$$\frac{\partial\theta}{\partial\lambda} = \frac{\Delta\phi_{12}}{2\pi d_{12}}\gamma(u),$$

$$\frac{\partial\theta}{\partial d_{12}} = \frac{\Delta\phi_{12}\lambda}{2\pi}\frac{(-1)}{(d_{12})^2}\gamma(u)$$

Use the following random uncertainties:

$$\sigma_{\Delta\phi_{12}} = .05 \text{ rad}$$
$$\sigma_{d_{12}} \leq .5 \text{ mm}$$
$$\sigma_{\lambda} = 1.365x10^{-9} \text{ m}$$

Which can be applied in the general uncertainty formula:

$$\delta\theta = \sqrt{\left(\frac{\partial\theta}{\partial\Delta\phi_{12}}\sigma_{\phi_{12}}\right)^2 + \left(\frac{\partial\theta}{\partial\lambda}\sigma_{\lambda}\right)^2 + \left(\frac{\partial\theta}{\partial d_{12}}\sigma_{d_{12}}\right)^2}$$

Due to a similar sensitivity spectrum (see that the sensitivity is dependent on the actual heading due to nonlinearity), the expected uncertainty and variance for the elevation computation will be also about 2.6 m. This gives an approximately conical “error bar” to the heading calculation, but the software is designed to bring the effects of these uncertainties out of the final reported position to an acceptable level.

4.4.4 Kalman Filter

Author: Jamison McGinley

Once a raw position estimate has been computed, the estimate is further filtered through a linear Kalman Filter. This is to further increase the accuracy of the Sensor and Beacon packages. Even though the predicted model of position computation has acceptable accuracy, as the implementation of the hardware deviates from the assumptions used to create the model, the position estimate is expected to worsen. This can be handled by increasing the accuracy of the estimate using a Kalman filter, which also lays the framework for the extension of this project to space applications where accuracy requirements will be greater. The motivation behind this is that at its core, the determination of the Beacon’s position is a state estimation problem with compound noise that can be predicted and corrected in order to produce a better estimate at large. This is a feasible algorithm to implement in this system for a number of reasons. First, the dynamics of the system are easily predictable as they are defined by each of the testing levels: stationary, relative motion, and cluster motion. Secondly, noise in the system is contributed to by the uncertainty in processes and Sensor uncertainty. For this system that is quantified and/or predictable from data sheets and can be inflated to provide robustness to the algorithm. Finally, the Kalman Filter is inherently limited to operation on systems with a discrete time step. Due to communications between the Sensor package and the Beacon operating on a pulse approach this requirement is met.

For a discrete time step, k, the linear Kalman Filter takes input variables: $A, Q, R, H, z_k, P_{k-1}, x_{k-1}$, which refer to the process model, process noise covariance, Sensor noise covariance, Sensor model, measurement update, previous covariance matrix, and previous state estimate, respectively. The filter will then produce the following output variables: P_k, x_k which refer to the new covariance matrix and new state estimate. The linear Kalman Filter operates in a two step process - The first step is to predict a new covariance matrix P_k and new state vector $\vec{x}_k$ purely based on the process model (dynamics) of the system:

$$P'_k = AP_{k-1}A^T + Q \quad \text{and} \quad \vec{x}'_k = A\vec{x}'_{k-1} + Bu_k \tag{5}$$

For the purposes of this investigation there is no forcing function to the input, and as such u_k is a zero vector making the relevant equation $\vec{x}'_k = A\vec{x}'_{k-1}$. In this case the state vector is $\vec{x}'_k = [x, y, z, V_x, V_y, V_z]$. For each of the testing phases, the dynamics of the system are necessarily different, these differences are reflected in A.

In the filter's second step it will then use a measurement, z_k, to correct the prediction into a final P_k and $\vec{x}_k$:

$$P_k = (I - K_kH)P'_k \quad \text{and} \quad \vec{x}_k = \vec{x}'_k + K_k(z_k - H\vec{x}'_k) \tag{6}$$

The unknown variable above, K_k, is the Kalman gain that is calculated at each time step via $K_k = P'_kH^T(HP'_kH^T + R)^{-1}$. This gain semantically represents how much the filter 'trusts' the measurement update in comparison to the predicted dynamics at a given time step.

In addition to A, Q, R, and P_0 are subject to change for different dynamical systems. Q, R, and P_0 serve as 'tuning knobs' in a similar fashion to control gains in a PID controller. The different dynamics for different testing scenarios will be implemented in the form of a configuration file that populates A, Q, R for the test at hand.

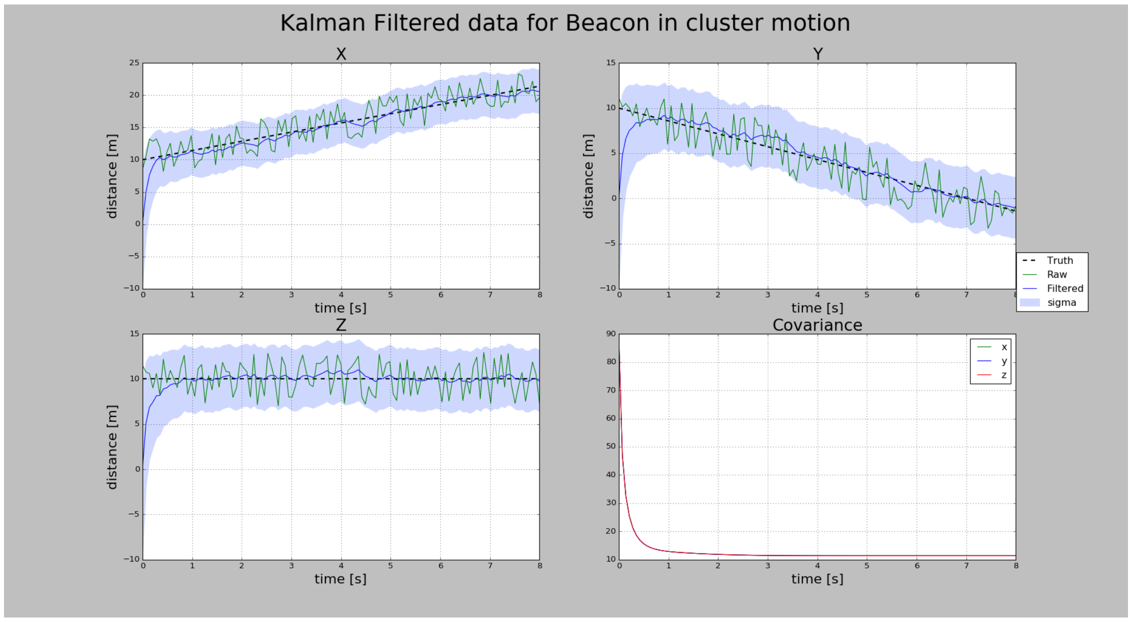


Figure 18: Linear Kalman Filter acting on simulated data for a Beacon traversing at constant velocity along the multiple axis, emulating distinct relative motion between the packages

Fig. 18 shows the implementation of the filter for relative motion between the Sensor and Beacon packages along multiple axis. The filter continues to converge in the face of constant velocity motion based on the dynamics from the relative motion test.

4.5 Hardware Design

The project hardware subsystems consist of radio, processing, antennas, GPS system, power, housing structure, and test mounting structure. Each is described in more detail than in the baseline design here. Where models were used, they are reported here as well.

4.5.1 Radio

Author: Camilla Hallin

Since the project requires communication via RF signals, some radio equipment will be necessary to mix signals and convert the digital signal to an Rf signal. As detailed above, the choices for radio varied between integrated radio and software-defined radio. Recall the chosen SDR is the Lime SDR transceiver. The Lime is capable of transmitting and receiving in the S-band, has programmable low noise amplifiers built-in. It has a total of 6 RX and 4 TX channels with UF.L connectors.

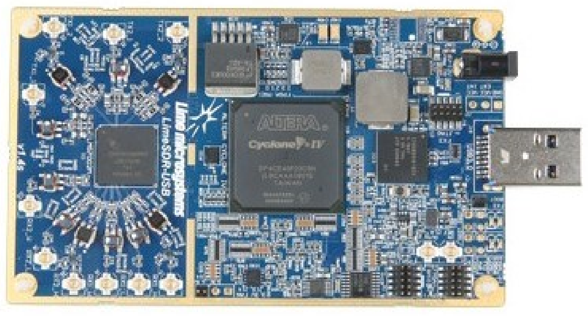


Figure 19: Lime SDR

The Lime SDR is shown in Figure 19. Note the board has on-board ADC and processing unit. This allows the SDR to function virtually autonomously and just report the received signals via USB 3.0.

The Lime is capable of 2x2 MIMO multiplexing, so two channels can be active at one time. To protect the SDR while it is not yet integrated into the final housing, it will be contained in an SDR case. This will help to mitigate potential damage to the electronics from ESD, water, or dropping damage. The case has feed-throughs for each of the connectors which will additionally protect the connectors from stress. Additional detail on the Lime SDR's can be found in the Hardware Archive.

4.5.2 Antenna Selection

Author: Camilla Hallin

The chosen antennas will interface with the Lime SDRs via SMA connector to minimize cycles on the more sensitive connectors. The primary antennas are Taoglas Monopole antennas. The antennas are omni-directional and linearly polarized. The 3-D gain pattern near the target frequency is shown in Figure 20.

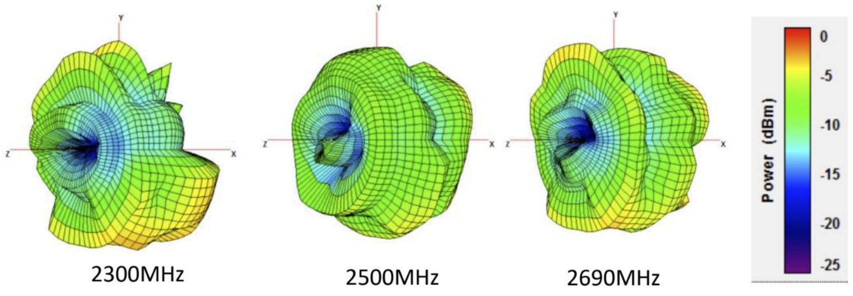


Figure 20: 3-D Antenna Gain Pattern [4]

There are additionally two back-up antenna designs selected. The back-up antennas are variations on the primary design including a clover leaf and chip antenna. Depending on the performance of the primary design antennas, these antennas can be swapped into the system to characterize any benefit from varying design.

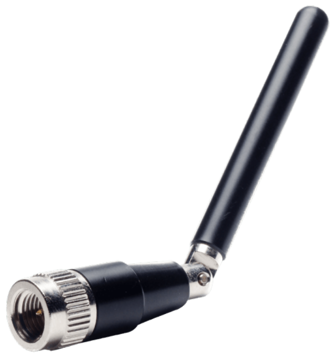


Figure 21: Taoglas Monopole Antenna

The primary antenna selection is shown in Figure 21. Note that in the final configuration (described below), the antennas will be mounted in the vertical position, with zero articulation at this joint.

4.5.3 Antenna Configuration

Author: Jarrod Puseman

To complete the phase interferometry, an antenna array of four antennas needs to create a set of orthogonal axes with their phase centers. In general, knowing where the phase center on an antenna resides is difficult. Additionally, since antennas are often close to their reception wavelength over 2 in size already, placing them near each other is a tab tricky. To get around the issues of nebulous phase centers and physical coincidence, the following antenna configuration was devised. The antenna array comprise a coordinate axes with the $< 1,1,1 >$ direction pointing directly downward. That is, the antenna at the origin is suspended above the box lid, and the three antennas placed on the lid will be the antennas at unit spacing on each coordinate axis. This allows the antennas to be mounted on the same plane, so the antenna phase centers will be ideally all shifted up or down the same amount. Additionally, only a single bracket is needed to create the antenna array. The antenna spacing is derived here. See Figure 22 for reference, which shows how the antennas will be configured.

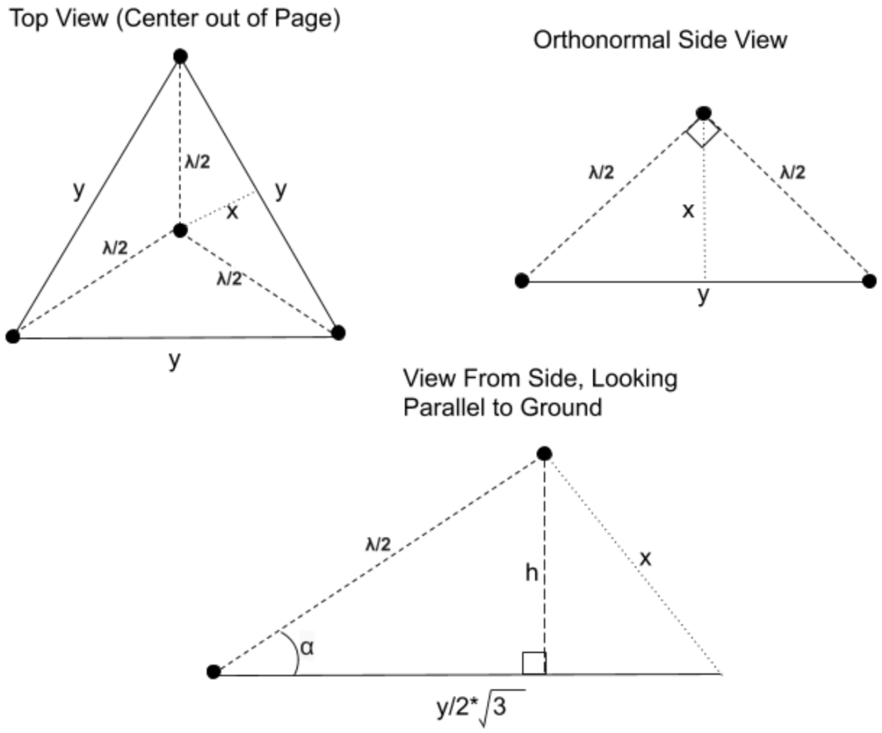

Top View (Center out of Page)
y
y
λ/2
x
λ/2
λ/2
y
Orthonormal Side View
λ/2
λ/2
x
y
View From Side, Looking Parallel to Ground
λ/2
h
x
α
y/2* √3

Figure 22: Geometry of the Antennas

From Figure 22 and using the law of cosines, one can then use simple trigonometry to compute h. Doing this gives

$$h = \frac{\lambda}{2}\sin(\alpha)$$

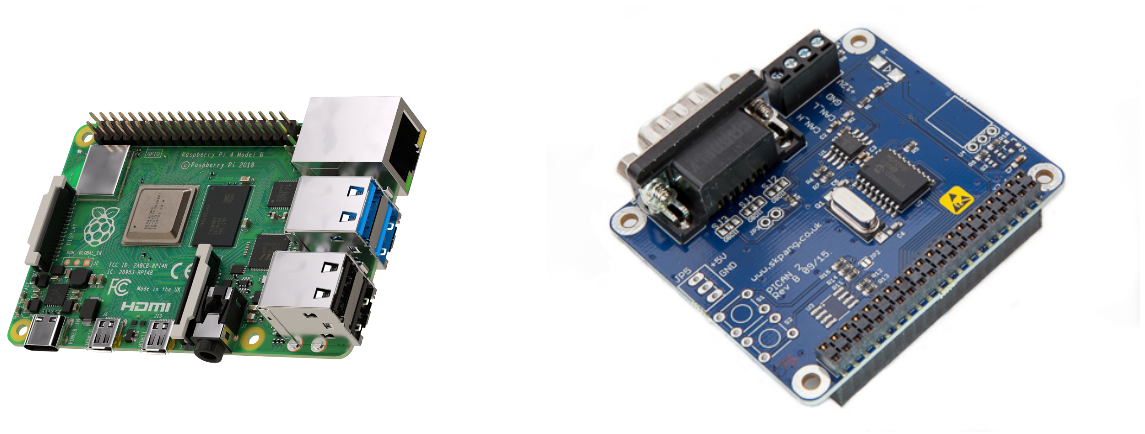
For transmitting at 3.3 GHz then with a wavelength $\lambda = 12.491$ cm, the distances of interest are computed to be $y = \frac{\lambda\sqrt{2}}{2} = 6.422$ cm and $h = 2.622$ cm.

4.5.4 Processing

Authors: Camilla Hallin A central processing unit is required to manage all the peripherals and perform calculations. Since the unit must interface with the Lime SDR, the processor must be capable of USB protocol. Additionally, the processing unit must be programmable more than once to allow for alterations to software, though most micro-controllers and microprocessors already have this capability. Additionally, the processor may need to interface with the SDR using some open-source firmware, so using a Linux-based or Linux-capable unit may make interfacing with the SDR easier.

To accomplish all of this, the Raspberry Pi 4 Model B was chosen to meet these needs. These will be the main computing units. The Pi 4 was chosen for its quick clock speed and ability to run its own OS, as well as the USB C connectors. More detail on the plan for software can be found in the Software Design section. The interface between the Pi and the 'Flight Computer' is facilitated by the PiCAN2 Bus interface. This meets the requirement of interfacing with a CAN Bus as requested by General Atomics. This is a shield which is fully compatible with the Raspberry Pi 4 Model B.

The Raspberry Pi 4 B and PiCAN2 are shown in Figure 23.



Raspberry Pi 4 B Central Processing Unit PiCAN 2 CAN Serial Interface Bus and Shield

Figure 23: Central Processing Electronics for ATOMIC

4.5.5 GPS Subsystem

Author: Camilla Hallin

The GPS Module is integrated separate from the main Sensor and Beacon systems. The purpose of the GPS is to log truth data to evaluate the performance of the main system. For this to be possible, the accuracy of the GPS module has to be maximized, otherwise it is impossible to compare Sensor data to less accurate truth data. The GPS RTK board NEO-M8P-2 incorporates Real-time Kinematic Corrections. This technique incorporates correction data from base stations. The RTK data will be streamed from UNAVCO station P041, located near Boulder. Details for this connection are reported below:

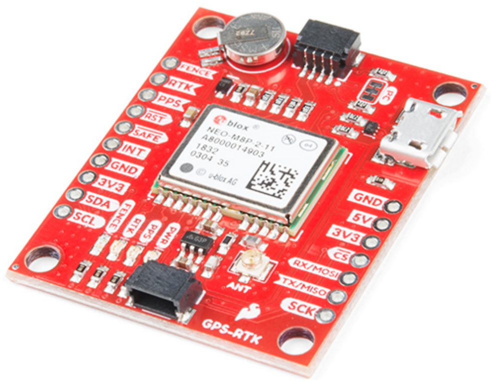
username: challin

password: 5HrAmz0e

The caster: rtgpsout.unavco.org

The port: 2101 (RTCM 3.1), 2105 (BINEX), and 2110 (Processed PPP)

Using this data increases the reported GPS accuracy from 2.5 m to 0.025 m accuracy at a rate of 5 Hz. The truth data will be communicated via I2C to the data logger and stored on a 128 GB SD card. This will be used in post-processing. These units are shown in Figure 24.


GPS 15005 GPS Module

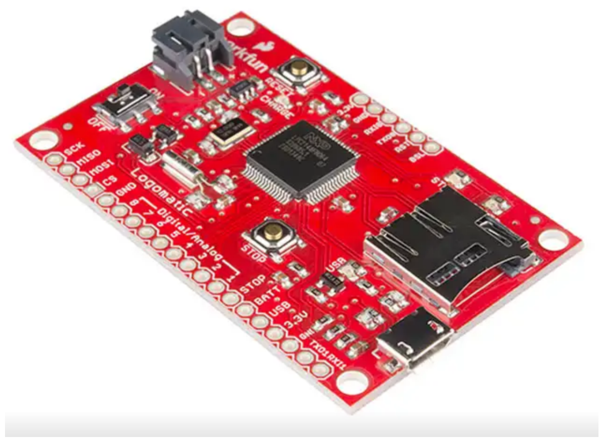

WIG-12772 Data Logger

Figure 24: GPS Data Logging System

4.5.6 Housing Design

Author: Jarrod Puseman

For the design of the housing, a simple, multi-functional design has been developed. The design serves as mounting as well as protection for all of the components of the system. The design is outlined below, piece by piece. No part drawings are included in the body of this report, but all drawings are available in the Appendix of the report.

Each component of the housing design is specified to be constructed from ABS plastic. This material was selected for its heritage being used as housings for electronic equipment. It's rigidity and flexibility offer a good compromise between protecting components and absorbing energy from impacts. Additionally, ABS plastic is commonly used in RF systems as a radome. That is, ABS is mostly transparent to RF signals, so the structure should interfere very little with the operation of the Sensor and Beacon in terms of the RF communication. In addition to this, the fastening method for the entire structure is to use glass-filled nylon socket head cap screws. The socket head cap screw is industry standard, but making them out of glass-filled nylon gives them a similar transparency to RF.

Before going into how all of the components mount to the housing structure, we'll look at the structure at large. The housing for the Sensor and the housing for the Beacon are the same design. The Beacon has a few unused features on the housing structure, but using the same design keeps things simple during the manufacture process. The housing consists of 7 structural pieces. These include 4 walls, a floor, a mid-section, and a roof. Together, these pieces form a cubical shape measuring 6 inches (15.24 cm) in every dimension. The dimensions of each piece are provided in the Appendix to the report in technical drawings for all pieces to be manufactured by Team ATOMIC, and described in Section 5.

Floor The floor of the housing structure has the greatest variety of features. The floor piece has four threaded holes for mounting to the test rack. The floor to the housing additionally has holes in the corners through which the walls will mount as well as has tapped holes for standoffs for the data logger and GPS unit. The final feature of the floor piece are counter-bored slots for socket-head cap screws mounting the battery. These provide a little slop in the fastening to accommodate battery dimensions not exactly as reported.

Mid-Level The middle level of the housing serves as a little bit of structure to the overall design. Like an internal rib, it serves as support for the walls in the middle of the height. Additionally, the middle level serves as the mounting location for many components. On the side facing the battery, the two Lime SDRs (if the Sensor, one on the Beacon) will mount, again on standoffs from the mid level. On the other side of the middle level, the custom-designed power board (discussed below) will mount. Note, since cabling to and from the battery and SDRs will need to go through this layer, a hole in one corner is called out to allow cable pass through. On top of the cable clearance, there are called out 6 holes for panel-mount SMA connectors. These will be used to pass signals from the antennas through the middle level to the SDR's on the bottom half of the housing.

Top Piece The top piece likewise serves two purposes: strength and mounting locations. The top bolts onto the top of the walls using the holes in the corners of the piece. On the inside of the top piece mounts the Raspberry Pi and the shield on it the PiCAN 2. The other holes in the top piece are all through-holes, but serve the hardware on top of the box. In the case of the Beacon, this is merely 2 panel-mount SMA connector for the Rx and Tx antennas. In the case of the Sensor, there are 5 SMA connectors that mount to it, and an antenna bracket. This bracket suspends a sixth antenna off the top surface of the box to make the phase interferometry (see below) work.

Walls The walls of the housing screw to the floor, middle piece, and the top piece, as well as each other. The walls also have holes cut out of each side. This is so that the SDRs can get adequate airflow to cool them. The FPGAs tend to run hot to the touch, and creating a hot-box for them would damage them. As a result, it is important to keep the SDR cold when performing benchtop tests. Another reason for the holes is cable pass-through to the PiCAN2 for communication with the flight computer. The cable will enter the housing enclosure through the holes in the walls. Finally, the holes serve to lightweight the enclosure a small amount. The lighter the enclosure, the less stress it can induce and energy it can receive during testing.

Brackets To complete the housing, a few brackets are also used. Most have been mentioned previously, but this section recaps them. Every bracket is constructed from ABS plastic, just like the rest of the housing. The first bracket is the antenna bracket. As discussed, this bracket mounts to the lid to hold one antenna off of the lid surface a specific amount (see antenna configuration below). This bracket takes a basic U-shape. The next few brackets mount the battery to the bottom piece of the housing. Since the battery cannot be punctured, these brackets wrap around the battery and press it into place. The front one serves as a bumper to keep the battery pressed against the “back wall". Another bracket stands on the side of the battery and reaches over it. Another bracket piece then sits on top of the battery and connects to the wall on one end, the side bracket on the other. This is the piece to hold the battery down.

4.5.7 Roof Rack Design

Author: Jarrod Puseman

To test the Beacon and Sensor packages, vehicles were selected to serve as the providers of relative motion at the appropriate scale of the project. Keeping the packages inside the vehicles might work, but team ATOMIC is wary of high signal loss and attenuation through windows and other car parts. Additionally, multipath error would increase tenfold. As a result, a roof mounting rack has been designed to mount the Sensor and Beacon to the top of separate vehicles. This will allow the Sensor and the Beacon to communicate in direct line of sight through only air. Additionally, the multipath errors should be reduces significantly in this way. This section details the design of this mounting system.

Mounting Plate Both the Sensor and Beacon have mounting hole patterns on the bottom piece designed to match the mounting plate. This plate will be attached the the roof rack mount system and the Sensor and Beacon removable from this plate using four socket-head cap screws. These thread into each science package, allowing simple assembly and disassembly. The material for this plate is ABS plastic, just like the rest of the housing, to avoid cutting RF signals and transmission.

Cross Beam The mounting plate is bolted on top of a beam spanning the car. This beam is set to be a structural fiberglass 2-in square tube. This dimension was picked according to the structural analysis performed below. This beam tube shape and material offered a stiff enough beam to be functional and minimally cut down RF signals.

4.5.8 Timing Analysis

Author: Camilla Hallin

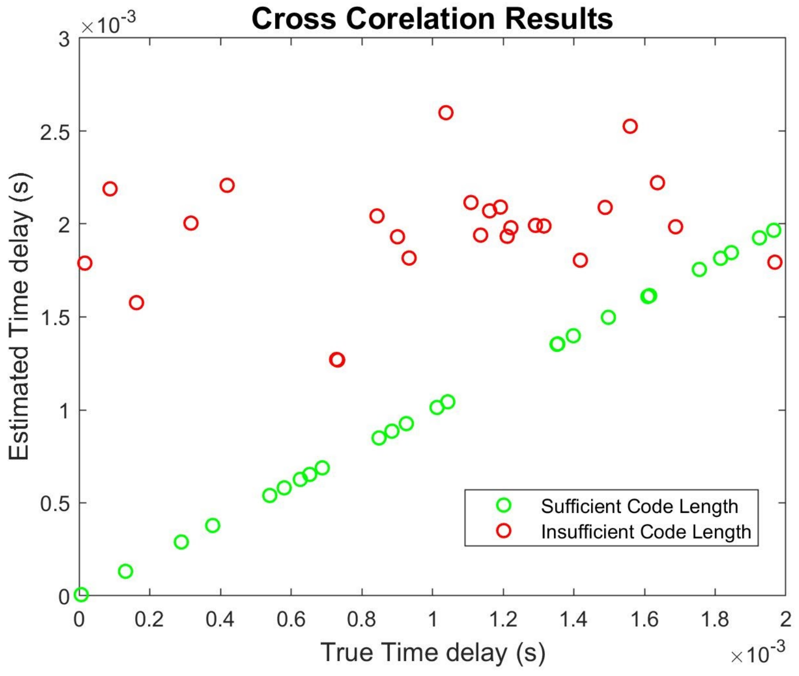


Figure 25: Results Comparing Code Length

To meet the requirement of a 1 Hz update rate the complete software run time including CAN transmission must not exceed 1 second. To determine the timing restraints a model for the link was created to determine the minimum reply sequence code length needed to recover accurate timing information. If the code is too short, the results of the cross correlation is not sufficient to create a peak in the the lags. This is shown in the red dots in Figure 25. The model uses the expected Eb/No and Pr/No values from the antenna link budget to simulate random noise to two signals. Then, barker codes of varying lengths are added to the two signals with some time delay Δt the two signals are then cross-correlated and the estimated time delay is compared to the true time delay. When the barker code is sufficiently long, the estimated and true delays match up as seen in the green dots of Figure 25.

The single point where the timing is not able to be recovered shows that even when the code is long enough, this method is subject to occasional errors. This helps motivate the averaging scheme as well as the Kalman filter because it will help to reduce the effect of outliers such as this. Additionally, it is a consideration to include in software a fail-safe that checks for outliers before averaging. This would prevent the mean from being skewed from the true position.

Additionally, to assess the timing requirements, the time to compute the cross correlation and peak lag is taken into account. This is achieved by using the timing feature in MATLAB. This is only an estimate because the SDR will likely be faster than the MATLAB version of the algorithm.

Message Signal Length:

$$9.1667E-8*30*4 \text{ (send and receive) } = 1.1E^{-5} \text{ sec}$$

300 m Transit Time:

$$2.0E-6*30 = 6.0E^{-55} \text{ sec}$$

Signal Related Timing Total:

$$7.1E^{-5}\text{sec}$$

Correlation Length:

$$XCOREL()*30*2 = 0.2994 \text{ sec}$$

Total Time: 0.2995 seconds

$$1-0.29995 = 0.7005 \text{ sec}$$

Processing Time Allowable:

$$0.7005/30 = 0.0233 \text{ sec per send/receive iteration}$$

The conclusions of this are that we currently estimate that the software run time must be ≤ 0.7 seconds. Given the Raspberry Pi clock speed of 1.5 GHz, this corresponds to $\approx 1E^9$ clock cycles.

4.5.9 Calibration Plan

Author: Camilla Hallin

There are multiple aspects of the hardware software interface that must be calibrated to accurately reflect the system. These are the phase delay through each of the Rx channels, the true antenna spacing, and the processing time of the Beacon. The results from these calibrations will be used to update constants in software.

Phase Delay

Purpose:
Calculate phase delay caused by RF components in the system to make sure there are no offsets that are not accounted for
Frequency:
Only needs to be performed once. Ideally, the RF phase delays are the same for each of the Rx ports on the Sensor

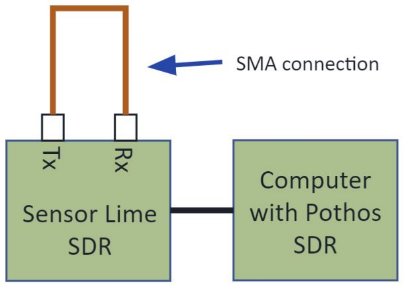


Figure 26: Phase Delay Calibration

Method:

1. Set Pothos to produce a wave with known phase and frequency
2. Feed into the Rx channel of one of the Sensor Phased Array antennas
3. Compute phase difference
4. Repeat for each antenna
5. If there is a discrepancy in phase difference, update software values to account for the difference

Antenna Spacing
Purpose:
Calibrate system for inaccuracies in phase center location of antennas Maximum sensitivity occurs at 90 degrees incidence
Frequency:
Each time antennas are mounted or re-installed, and if noticeable bias occurs in heading measurement

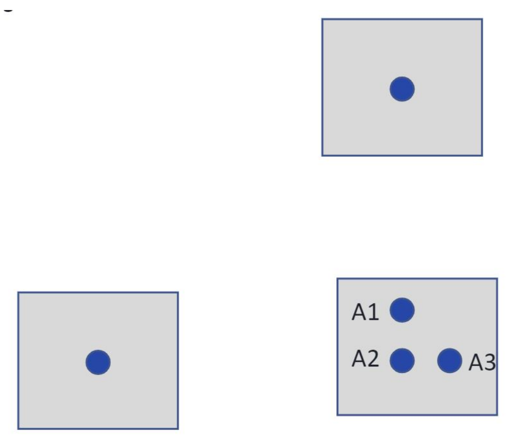


Figure 27: Antenna Spacing Calibration

Method:

1. In Lab, place Beacon at 2m 90 degrees from the Beacon for each pair of antennas
2. Back calculate distance between two antennas using the calculated phase difference
3. Repeat for each side of A1,A2 10 times each side and take average update software with values for r12, r23 etc.

Processing Time
Purpose: Get accurate as possible value for **Frequency:** Any time the software is updated

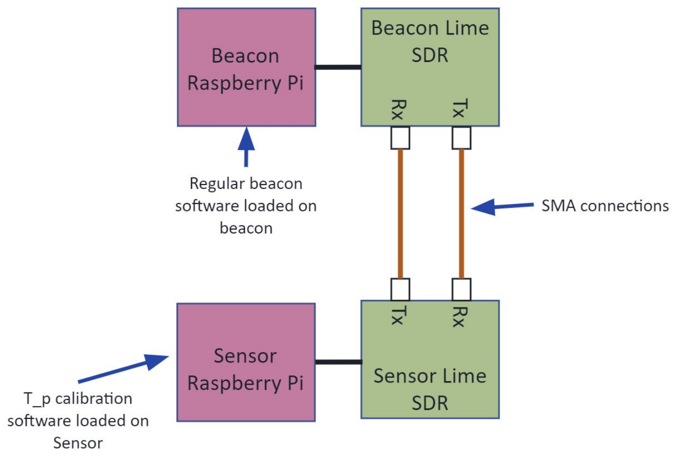


Figure 28: Antenna Spacing Calibration

Method:

1. Connect the Tx of the Sensor directly to the Rx of the Beacon via SMA cables and vice-a-versa

2. Using $t_{process}$ calibration software loaded on the Sensor, send multiple poll messages, allow Beacon to run standard processing software and respond with the response message
3. Calculate average time delay over 500 poll/reply cycles
4. Report average and standard deviation
5. Update Sensor software value $t_{process}$ with average

4.6 Power System Design

Author: Corey Huffman

In the context of the mission, the Sensor and Beacon will be powered from a satellite bus, and therefore will need to be able to handle an unregulated 28 ± 6-volt source. This is reflected directly in the functional requirement 6. In order to do this, it was understood that some sort of circuitry would have to be designed to convert the unregulated higher voltage to a regulated voltage that the electronics can operate at. All the electronics that have been picked for the Beacon and Sensor packages operate with 5 volts. In order to get the 26 volts to the 5 volts needed for the electronics some sort of converter needs to be used. The first consideration was a simple linear converter, however there were some concerns of the heat that would be dissipated. A simple linear regulator was found, namely the MC7805 from ON Semiconductor, with a power to heat ratio of 10°C/W as defined in the data sheet. Using a worst case estimate for amperage used of 2 Amps, representing the Sensor package, the total temperature dissipated would be 125° C which would still be large temperature differential to handle. After showing that a linear regulator would not work, a DC/DC converter was found. This DC/DC converter is the mEZDPD3603A from Monolithic Power Systems. With this high efficiency converter, the temperature rise would only be around 30° C. This temperature differential is much more manageable and may not even be an issue for the Sensor or Beacon.

After finding a converter that can be used reliably within the Beacon and Sensor, the over all layout of the circuitry can be built and a schematic can be designed. One of the major components that had to be considered was how the LimeSDR would be powered. While the LimeSDR could be plugged in and powered directly through the Raspberry Pi, there are concerns of over current or inconsistent power to the SDR. So, a data bypass was designed and the power to the LimeSDR will be provided directly from the power board. This lead to a vast array of revisions in order to maintain the USB3.0 connection speeds between the Raspberry Pi and the LimeSDR. All other components of the Sensor and Beacon will also be powered directly from the power board. The battery will be plugged into the board through a Lipo battery connector and will run through the converter to create a 5-volt line that will be distributed to all the components. The data logger will also be connected through a Lipo battery connection. The LimeSDR, GPS module, and the Raspberry Pi will all be connected through a USB port.

While the connections and power distribution are enough for a schematic to be built and a board to be designed, it is also important to be able to test the power system in a modular manner. Along with the need to test, over current protections should also be implemented to avoid potential damage to other components or the batteries. In order to incorporate testing, there will be test points added at each connection point leading to a component. Along with a test point there will be a switch to toggle which components receive power to locate any potential issues that may arise. For ease of visual inspection there will also be indicator LEDs and fault LEDs added to show which components are receiving power or those that have had a current overload. This will aid in circuitry and power debugging. In order to avoid over-current, there will be fuse placed after the main power line and power hotswaps will be paced before each component to help mitigate any damage that could come from over current. With these considerations in mind, Fig. 29 shows the final schematic design.

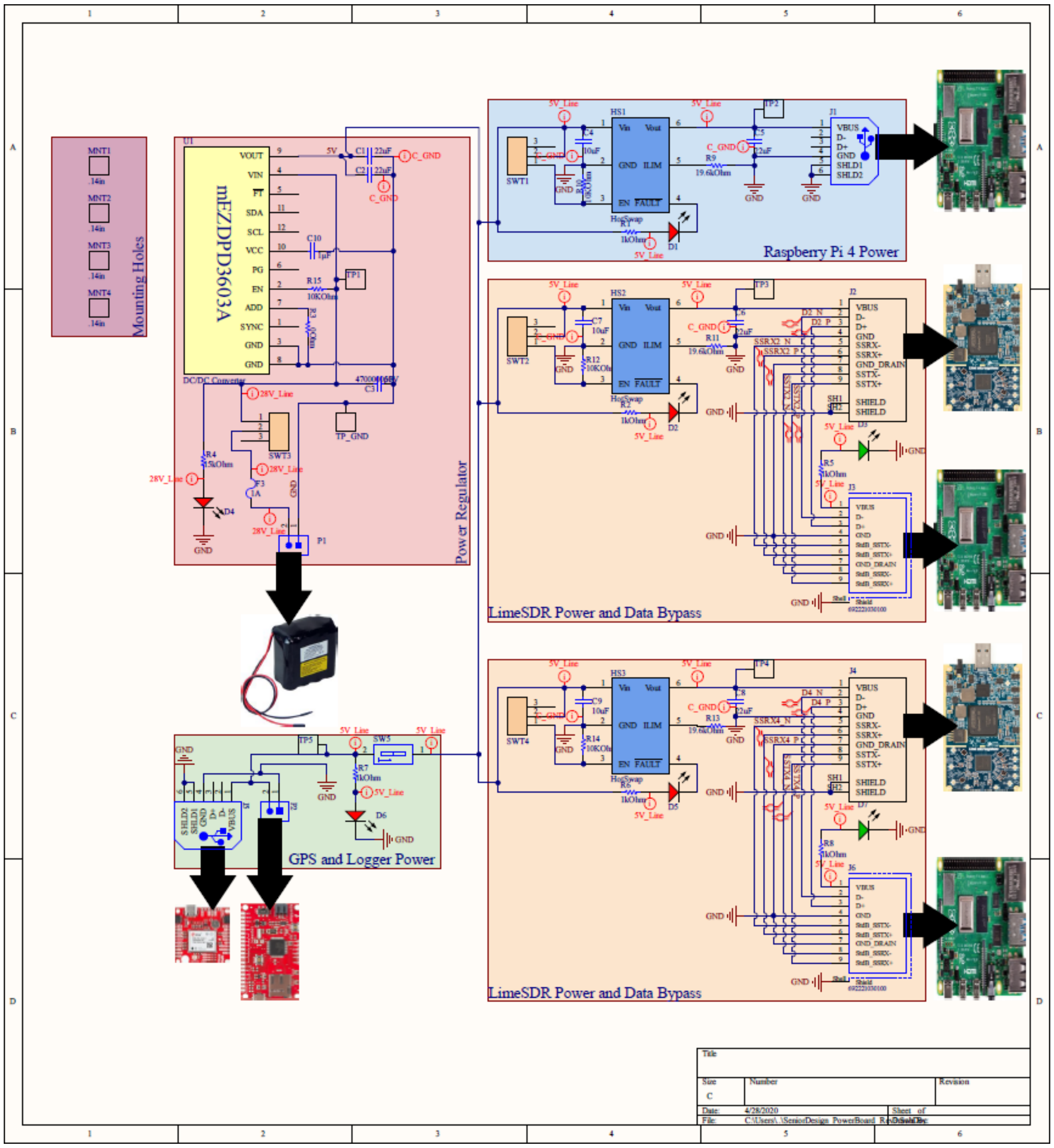


Figure 29: Schematic for power board design

This schematic was designed after the current draws were estimated from the selected components and through the converter and onto the batteries. Knowing that the current draw should not be above 2 amps, the converter will be capable of handling the circuit. Along with the current draw, the output voltage can range from 0.6 volts to 12 volts which covers the 5 volts that is required for the selected components and the input is range is 4.5 volts to 36 volts which covers the 28 $\pm$ 6 volts that a bus would require. This converter also falls within the temperature range that functional requirement 5 defines. Next the batteries had to be picked based on expected current draw and temperature range as well. The battery that was found to best fit the expected draws is the Li-Ion 18650 28.8-volt 3000mAH battery from AA Portable Power Corp. The discharge operating range is from -20° C to 75° C which falls within the functional requirement 5. The lower end of that bound is matching that of the functional requirement, however as this is power that would be provided by a bus, and not a part of the system, it is not as dire that it will be within that range. This battery pack has a current discharge range of 600 mA to 4 A which will cover any current pulls that we would expect from the Sensor or Beacon. Also included in the battery pack is a pre-installed protective circuit to protect from overcharge, over discharge, over drain, short circuit, and over temperature. This will help with protecting the battery and the Beacon and Sensor from damages pertaining to power. Pictures of these two components can be seen in Fig. 30.

Figure 30: Battery and DC/DC converter

5 Manufacture

The manufacture tasks of this project included mechanical parts, electrical components, and software. Since project ATOMIC's main objectives were to prove the feasibility of a cheap tracking system and less on specific strength or material properties, the largest manufacture element was software. The various manufacture elements are described below.

5.1 Mechanical Components

Author: Jarrod Puseman

5.1.1 Mechanical Parts Description

The mechanical components of ATOMIC were all built in the Aerospace Machine Shop. These parts consist of the housing materials listed in Section 4, namely the four walls that comprise the housing of each unit, the floor piece, the mid-level, and the top piece. Additionally, the test mount racks were also machined in the Aerospace Machine Shop. These included the clamps, beams, and the adapter plate on the test mount.

To manufacture the four walls, the parts were rendered into SolidCam, the machine shop's tool for generating G-code for the computer-numerical control (CNC) mills. Once the tool paths were programmed, the parts and tools were installed into one of the CNC mills in the shop and the program executed. This program served to cut all the triangular slots in the walls. After these holes were cut, the walls were removed and the holes on the sides and top drilled and tapped manually on the mill. The manufacture of the floors, wall, and ceiling were all done similarly. That is, the parts were opened in SolidCam and the paths and tools programmed. From here, the parts were installed on the mill, the tool paths run, and any holes that needed to be tapped were done by hand. An example of a piece being cut on the mill is shown in Figure 31.

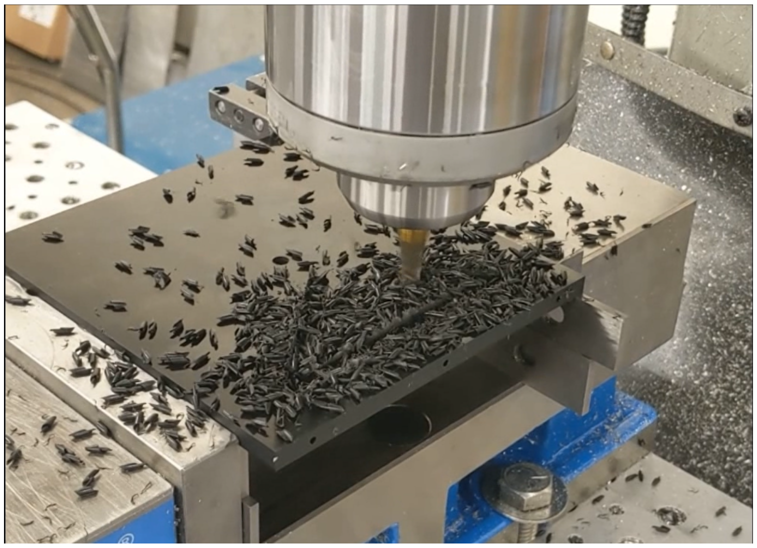


Figure 31: Picture of a Wall Section on the CNC Mill

In the floor, midlevel, and top, the mounting holes for all of the electrical components were a size small enough to require helicoil inserts in the threads to strengthen them. These helicoils were inserted using the tools in the Aerospace Machine shop. To do this, the holes are first drilled to the precise depth and width reported by helicoil tap and drill charts. The holes are then tapped using a helicoil tap ordered by the team. This tap was a bottoming tap since the holes in the parts are blind. Once the holes in the pieces were all drilled and tapped, the helicoils were inserted. For the size helicoil used on the project (2-56), the tool to do this is a small hand-held piece of hex stock with a modified threaded insert in the end of the tool. The helicoil threads onto this tool and catches at a small protrusion on the tip of the insertion tool. The helicoil is threaded into the part until the top thread of the helicoil is advanced a quarter turn beneath the surface of the part. The helicoil insertion tool is removed, and another spring loaded punch is used to break the tang off the bottom of the helicoil. These tangs were all counted and confirmed to have left the part.

Additionally machined for this project were several brackets. One such bracket was the bracket for the central antenna on the Sensor. This bracket stands the antenna a precise distance off the top of the Sensor to form the orthogonal coordinate axes used to measure heading for phase interferometry to compute relative heading. Since this calculation will be changed by small changes in position of this antenna's phase center, the precise placement of the antenna is essential to get accurate positions from the Sensor. To build this bracket, the block was placed on the mill, and the final height was cut first while the bracket was still the most rigid. After this, the hole for the antenna was drilled for the same reason (most stiff means more precise), and then and end mill was used to cut the remaining slots and features manually. The bracket also used a tight tolerance counter-bore for socket-head cap screws to minimize slop from bracket placement. All this allowed this bracket to be one of the most precise parts on the project (along with the precision drilled mounting holes for the remaining antennas in the top piece of the box).

The other bracketry constructed for the project holds the battery in both the Sensor and the Beacon. The parts for each battery consist of a side, top, and front bumper which keep it wedged into the corner of the frame. The front brackets were quickly machined by hand on the vertical mill. The side and top were also done on the manual mill, with the added complexities of drilling and tapping several holes. Additionally, the side bracket required two horizontal slots that allow the bracket to be tightened to the battery, and these were also cut on the manual mill without issue.

5.1.2 Mechanical Obstacles Faced

Throughout the manufacture process of the mechanical structure, a few issues were encountered that had to be overcome. These included remaking parts, improper hole locations, a cable routing issue, and working in available shop hours.

The parts that had to be remade where the top bracketry to the battery. In transcribing a drawing from SolidWorks, a dimension was miscopied and the first revision of the parts did not fit. This issue was easily overcome by making two new parts of the correct size with the extra material purchased from McMaster Carr. The impact to the overall project schedule was negligible as the team was able to take on an extra shift in the machine shop to fix the parts right away.

Another issue that befell the project was an improper hole pattern in for the GPS unit in the floor piece of the project. The reason this occurred is that there was no data sheet for the breakout board for the GPS (just the GPS chip itself). This led the team to use online pictures of the board and measuring with a ruler on a computer monitor. This dimension was then scaled according to the dimensions given next to the picture of the board, and these dimensions used in SolidWorks. At the time of this design step, the decision was made to modify the CAD and drawings upon the arrival of the GPS unit, when measurements with calipers could be performed on the actual hardware. Needless to say, this step was forgotten, and the parts were manufactured according to the rough dimensions from online photos. The GPS unit did not fit the existing hole pattern. After some quick geometric analyses, it was determined that none of the existing holes could be re-used in a new hole pattern. The correct hole pattern was then drilled and tapped in a slightly new location that allowed the GPS to fit. The delay to the schedule was minimal. While planning the corrective action required a day after the issue was found, this was able to be eaten up by an extra day in the hardware manufacture schedule. There were also additional helicoils for this, so no extra hardware needed to be purchased.

On top of these issues, a cable routing issue became apparent when integrating all of the hardware components. When designing the box, thinner and more flexible cables were in mind when computing necessary clearance for the bend radii of the cables. To communicate via USB 3.0 However, cables with more wires are required, which makes the wires thicker. In turn, this increases the bend radius, so the purchased cables would not fit into the structure of either the Sensor or the Beacon. To solve this issue, the top parts of two opposing walls on each structure were removed to allow the cables to travel exterior to the box. This does not exceed the minimum bend radius of the cables and still provides enough structural support to the Sensor. This issue did not affect the schedule much. While the modification did occur after the budgeted manufacturing window, a team member was able to modify them quickly and not allow the schedule to slip.

The final issue faced when manufacturing the mechanical/structural components was finding sufficient time to work in the shop. Only one team member had taken all of the shop courses, so that person was mostly responsible for manufacturing the hardware. With classes to take during the day and the shop following a business hours schedule, only a few hours each day were able to be devoted to hardware manufacturing. To get around this issue, several of the team helped out in the various ways that everyone could. By effective schedule management and employing the help of several team members, the project was able to be completed primarily in the budgeted machining window.

5.1.3 Mechanical Integration Plan and Results

The integration plan for the mechanical components was straightforward since the project was not too complicated mechanically. The plan for integration first tested all of the structure and it's ability to mount up into the cube shapes that house all the internal components. The reason the entire structure was tested at once (rather than progressively) was the result of confidence in the team's ability to manufacture. One member had confidence that the relatively simple parts could be machined to precision without fitment checks. When this integration was performed, the structures all mounted together just fine, though some shorter fasteners were ordered after realizing initially some of the wrong size had been requested on the procurement spreadsheet.

The next step of the integration was to make sure the hardware fit into the box. This step required mounting all the hardware into the structures to make sure that components would clear each other and fasten securely to the frame. The critical part of this integration was making sure the non-prismatic battery could be held securely in place. When performing this integration, the team found that the GPS unit did not mount correctly to the floor of the project, as described in Section 5.1.2. Recall that this issue required modification to the structure of the Sensor and the Beacon. However, the rest of the components fit as designed into the structure, and the power board switches were confirmed to clear the PiCAN 2 module. Additionally, the battery was found to be held securely by angling one of the clamps as allowed for in the design using slotted counterbores.

After the hardware mechanical interfacing into the frame, the next stage of integration focused on testing hardware fitment. After the test hardware was confirmed to bolt and screw together (checked in the first hardware integration step), it needed to integrate with the specific test vehicles. This simple process involved bringing it to a test vehicle and mounting it to make sure it fit. The testing rack was confirmed to fit to one team member's vehicle, though the second rack was never integrated in this way due to the work-halt order.

The final mechanical integration step mechanically was a cable fit check. This integration merely extends the hardware mount check to now all hardware and physical objects. This includes cables, wires, and antennas. When performing this integration step, it was found that the acquired USB 3.0 cables would not fit into the structure as designed. They could not bend in the tight space originally planned. This led to the modification of the wall. The rest of the wires all mounted well and were appropriate lengths.

5.2 Electrical

Author: Corey Huffman

5.2.1 Electrical Parts Description

The main point of focus for the electrical parts is the power board. For the board to be made, the design was sent to OSHPark for manufacturing. Then, there components that were to be placed on the board where ordered from DigiKey and Mouser. Finally, the overall assembly of the board was done in house. There are a few key pieces of the board to highlight outside of an array of capacitors and resisters.

One of the major components is the previously mentioned (Section 4) DC-DC converter from Monolithic Power Systems. This component is a programable DC-DC converter that has a range of output voltages and can handle an input above what FR 6 requires. The model used on this board is preset to output 5 volts, which is exactly what is needed for all our subsystems.

Another major component is the hot-swaps used for over-current projection for the Raspberry Pi and the LimeSDRs. These hot-swaps are the Richtek RT9728BHGE and are specifically used for USB connections. They are also programmable for a range of currents but were set to a current of 1.5 amps as we were not expecting to have anything above that. However, if that were to go against expectations a different resister could be put into place changing the current the hot-swap would allow. A key function of this hot-swap is the fault line, which would light an LED should an over-current instance happen.

A major set of components is the various connections on the board. There our two Li-po battery connection, 2 USB-B connections (3.0), 2 USB-A connections (3.0), and 2 USB-A connection (2.0). The two Li-po connections are used for the battery and the connection to the data logger. Since the data logger was meant to be used through a Li-Po battery and was therefore the primary connection to provide power. The USB-B connectors are used for the data bypass to the Rasberry Pi. USB-B was chosen for its rigid connection as well as the availability of a USB-B to USB-A connection since USB-A to USB-A is not a common connector. The USB-A 3.0 were used for the other side of the data bypass to the LimeSDRs. USB-A is what the LimeSDR would be used it if were plugged directly into the board, so this was the logical choice for the connection as a simple USB extender could be used. The USB-A 2.0 was chosen as it was a simple power output for the Raspberry Pi and GPS module, and could be used with a USB-A to USB-Micro connector.

The final noteworthy set of components on the board is the switches. There are four toggle switches and one slide switch on the board used to control power flow on the board. The toggle switches are used on the main power and to the major subsystems of the board. They were chosen for their high voltage and current rating as well as their rigidity for holding their state. The slide switch activates the GPS module and the data logger. The slide switch was chosen as it was enough to handle the expected load from this subsystem and would take up less space on the board.

5.2.2 Electrical Obstacles Faced

As most of the components were of the 0603-size variety and surface mounted, the main obstacle faced was the time it took to hand solder everything onto the board. However, although the first board took some time, the second board came together in a much sorter time span. There was talks of building up a means to surface mount with an oven and stencil, but it was deemed unnecessary as hand soldering could be done and there was no need for a mass production of boards. If more boards were to be made, a stencil-oven combination would be advised.

Another obstacle faced was shorts on the board. Since the components were 0603 and close together, there was a short on the first board put together. This was a major obstacle as it was a short of the 5-volt line to ground and since these two lines were also tied to power and ground planes on the board there was no easy way to narrow down the location of the short. This prompted a component by component check which located a trouble area which was fixed. This process did eventually lead to a re-flow of the whole board to ensure that there were no other shorts.

5.2.3 Electrical Integration Plan and Results

Since the electrical systems was focused around the power board, the main part of the integration was the manufacturing and build up of the board. Once the board was built up and tested it was time to begin plugging in the pieces.

The first step was to plug in the battery and make sure that the toggle switch allowed for flow control. This was a success and was proven through the indicator LED that lit up when the switch was on and turned off when off. The next phase of plugging the battery in was to test that the input was as expected and the output from the DC-DC convertor was nominal. Through the built-in test point the output from the battery was as defined by FR 6 and the output from the DC-DC convertor was 4.96-volts (tested at 5 different test points) which was enough for the subsystems to use. After the check on the battery it was time to start plugging in the other subsystems to ensure that they would turn on and run as expected.

The first subsystem to be plugged in was the Raspberry Pi through its power port. This was a simple ‘on’ test, which was done through the toggle switch inhibiting that area of the board. Upon toggling the switch to an on state, the Raspberry Pi booted up as expected and had no issues while running. This subsystem was turned off for the next one to be tested.

The next subsystem was the power to the LimeSDRs. This was tested at both connections individually and then both at the same time (to separate SDRs). Like the Raspberry Pi, the toggle switch was switched on and the LimeSDR turned on and when into its nominal state with no issues. This held true when two SDRs where been ran at the same time. This checked off the power to this subsystem and was turned off for the next.

The final power test was that to the data logger and GPS module. Prior to this test a configuration file had to be placed on a microSD card and placed in the logger for the data to bed passively logged. Then a wired connection directly between the TX of the GPS and the RX of the logger, as well as a GND between both, had to be made. Then the two were plugged into the board and the slide switch was turn to its on state, and both subsystems turn on. To check that the functionality was also nominal the microSD was checked to see that the GPS outputted data, which it did so this subsystem was also turned on and running nominally through the power board.

For the power board to be finished with integration and testing the USB3.0 connection had to be tested from the SDR to a computer. A laptop was used in leu of the Raspberry Pi as the Pi had not yet been programmed to use the SDR at the time. However, since the Raspberry Pi is a computer, this test was enough to prove the boards isolation of the power output from a computer and the data bypass. The computer was connected to the USB-B side of the board and the LimeSDR was connected to the USB-A (3.0) side. The computer was able to run two LimeSDRs and recognized both connections as super speed connections, proving that the USB3.0 bypass was working.

With everything running on an individual bases it was important to make sure they would all run together while being powered from the battery. So all subsystems where plugged in and turned on through the power board verifying that the board can handle power conversion and distribution as well as a data bypass between the SDR and a computer. With this verified, the integration of the power board and the subsystems was completed.

5.3 Software

Authors: Camilla Hallin & Jamison McGinley

High level operation of the Sensor and Beacon systems was facilitated through the use of a Raspberry Pi 4 mounted directly to each of the packages. The Pi's each had a 32 GB SD card which contained its operating system and all software required for operation. The Pi's were loaded with Raspian, a proprietary Linux based operating system. Furthermore, the Pi's were programmed to host their own wireless networks to allow direct line SSH into either the Beacon or Sensor from another computer to allow for rapid debugging of software issues.

Communication with the Lime SDR's and processing tasks such as averaging time and phase measurements, heading and range computation, and Kalman filtering would occur locally on the Raspberry Pi's in operation.

The SDR is initialized with the Lime driver. The Lime driver and complete SoapySDR API are built from cmake files on the Pi. The necessary drivers and files are in the project archive, along with instructions for installing the software. Once the proper SDR has been identified (using device serial numbers) the SDR is configured to transmit and receive on 3.1 GHz with a sample rate of 61.44e5 Hz. Although this seems like a mis match that would produce aliasing, the sample rate is sufficient for the mixed signal (baseband offset and binary bit rate), which is what is important to the data, not the carrier frequency itself. The baseband offset is approximately 19.6 kHz which is sampled at far greater than twice that, which prevents any aliasing that would occur.

Interfacing with the SDR is enabled via the API using three major elements: filling the Tx buffer, reading the Rx buffer, and obtaining time stamps. Each of these routines is reused on the Sensor and Beacon as needed to complete the software flow. The software is modular with many methods being used more than once and for multiple uses. This cuts down on the complexity of the overall software significantly because the routines can be reused and are guaranteed to be the same for the code comparisons. Filling the transmission buffer is done by first producing binary data for the code to be transmitted. The number of data points per code symbol is a function of the sampling frequency and the code bit rate. Then, the baseband offset signal is phase modulated by the binary data. This produces complex Inphase Quadrature (IQ) data to fill the Tx buffer. This is also the routine run to create the comparison message for the expected response message. This expected message is housed in the memory of the Beacon and the Sensor and used on the receive routine to compare and recognize the poll and response.

Routine	**Packages**	**Tunable Parameters**	**Summary**
Initialize	Sensor & Beacon	Gains, Lime Device Serial Number	1. Create Soapy Device Object using Lime Drivers 2. Set Frequency, Rx/Tx Channel, Bandwidth, Gains, Antenna, Sample Frequency 3. Create Complex 32 bit Tx/ Rx Streams
Generate Complex Message	Sensor & Beacon	Code length, "Amplitude"	1. Create Binary Data with correct samples per bit 2. Create time vector for transmit length 3. Create IQ data: exp(f 2 π t i + BIN · π i)
Transmit Poll/Identifier	Sensor & Beacon	Scheduled time difference	1. Get current time from SDR 2. Write to Tx Buffer 3. Stream Scheduled to start at t_start= t0+10us 4. Activate Tx Stream
Beacon Receive	Beacon	Buffer Length	1. Activate Rx Buffer 2. Read buffer 3. Get ns Time stamps
Identify Poll Message	Beacon	Threshold	1. Use numpy.correlate to correlate Poll Msg (generated) to rx_buffer 2. Find max argument 3. Compare with threshold 4. If met, send response
Find TOF	Sensor	Processing Time	1. Use numpy.correlate to correlate Poll Msg (generated) to rx_buffer 2. Find max argument 3. Get time of max arg 4. Subtract Processing Time
Switch Antenna	Sensor	Switch settle time	1. Specified antenna mode input 2. Adjust GPIO pins to match antenna mode 3. Let switch settle
Get Phase Difference	Sensor	None (unless phase offset is uncovered during calibration)	1. From Rx Buffer data, subtract phase for current antenna 2. Call Switch antenna 3. Receive new buffer/phase 4. Call Switch Antenna 5. Receive new buffer/phase
Find and Send Position	Sensor	Notional Path	1. Using TOF find range 2. Using Phase Differences find heading 3. Compare to notional path 4. Output to PiCAN

After receiving data from the SDR's, computation is required to get a filtered position estimate of the Beacon. This takes the form of simple averaging of the SDR data, computation of heading and range, and a further step to get an unfiltered position estimate. All of these elements were implemented as python functions to be run during system operation.

Filtering the position estimate is accomplished using a linear Kalman Filter for testing Phases 1-3, and by an Extended Kalman Filter (EKF) for testing Phase 4. For a discrete time step, k, the linear Kalman Filter takes input variables: A, Q, R, H, z_k, P_{k-1}, x_{k-1}, as indicated in Section 4.4.4. The filter will then produce the following output variables: P_k, x_k which refer to the new covariance matrix and new state estimate. The linear Kalman Filter operates in a two step process: The first step is to predict a new covariance matrix P_k

and new state vector $\vec{x}_k$ purely based on the process model (dynamics) of the system, The second step is correct the covariance estimate and state estimate using a measurements. The python code on the Pi will supply constant input variables such as A, Q, R, and H from a configuration file specific to the test while other required elements such as the measurements are found using the previous code blocks. The EKF was never implemented in code due to the limited amount of testing that was able to be completed. However, in terms of software implementation the EKF would behave in largely the same way as the linear Kalman filter with differences in elements computed live in test versus loaded from a configuration file.

5.3.1 Software Obstacles Faced

The most difficult part of getting the software built was understanding exactly what could and could not be done by each component in the chain of data flow. For instance, when we first began designing and building the software, we were under the impression that the SDR would be able to run GNU scripts on its on board computer. We assumed that a script could be loaded to the SDR, much like a micro controller and that the data output would be whatever we had programmed in GNU even after separating it from the programming device. If this were possible, we would receive only the processed data such as phase difference or time of flight over USB to the Pi. However, this is not possible. In reality, the SDR will always output IQ data and the GNU is just a script that tells the computer, in this case the Pi, what do do with the IQ samples received by the SDR. Understanding this took some time, and for the first level of SDR testing we were still using GNU radio. GNU radio was a great tool to help understand the RF characteristics of our system. Seeing physically what a base band offset is when you send a signal, was really interesting. Mixing signals in GNU is very easy to do compared to mathematically producing the complex data, so this was very much the prefered method for learning. We were also able to use GNU GUI's such as the FFT or Quad plots to get a feel for how our radios were behaving. Looking back I would not have changed using GNU for the preliminary software development such as determining best sample rates and offset frequencies. However, as soon as we learned about the SoapySDR API this became the preferred method.

Another interesting problem that emerged in software but became a hardware/software problem was the antennas on the phased array. Originaly we thought the multiplexing worked for all six antenna channels, however we soon discovered there were only 2 physical Rx channels. Having this restraint built in to the SDR meant that we needed to include an external switch. The switch had to be RF rated to ensure signal integrity across the input to output. The switch state is controlled by 3.3 V across selector pins, for which we wrote a simple script to control the GPIO pins on the Pi. This added some minor complexity because in the receive script the Pi had to both receive, calculate phase difference then switch the antenna reading. The center antenna stayed stagnant as the reference for all.

6 Verification and Validation

As indicated in the Design Process and Outcome section, there were an number of preliminary and system level tests planned to verify the system as a whole against requirements. However, due to the halt on manufacturing it is more apt to describe each CPE and its planned validation against requirements as well as plans if it was found that requirements were not met. Because of this failure to verify or validate our results, we will go through what this procedure would have looked like, and and what steps would have been taken to verify and validate our results had we been given the opportunity to do so. We will explain this process with respect to each critical project element.

6.1 CPE 1 - Two-Way Range Finding

Author(s): Brendan Lutes

In order to verify the two-way range finding method, we needed to be able to demonstrate that we could successfully calculate the range between the Sensor and Beacon 66% of the time with an accuracy of 2% of the range. All four testing levels would have allowed us to validate this capability, and characterize the accuracy of our system in different ways.

We expected to collect two-way time of flight data for each poll and response cycle. These time of flight measurements would then have been fed into the standard radar range calculation given by Equation 7 below. This simply multiplies the RF wave propagation speed by the time of flight to give the distance between the Sensor and the Beacon.

$$x = \frac{c}{2}(\Delta T_{total} - (t_{receive1} - t_{receive2})) \tag{7}$$

In order to validate the collected range values, the on-board GPS would have collected truth data stored in GPS coordinates with time stamps. The difference between these time stamped coordinates then gives relative position versus time. Taking the vector norm for distance would allow us to compare the average accuracy of our system during our tests.

The models laid out in the Project Planning section show the feasibility of this solution method. Monte Carlo simulations were run for the range calculations, injecting normal random noise into our system scaled by an estimate of the timing error in the system. This simulation yielded the expected variance of our time of flight measurements, which would have been used in our verification and validation of the solution. The estimated scale of the error was calculated from a combination of oscillator clock jitter, sample rate, clock speed, and cross-correlation fidelity. After testing the system, this value can be updated with an experimentally determined value.

After collecting our test data and truth data, we would then have started to characterize the success of our system in various ways. Phase 1 testing allows for simple accuracy testing - are we able to accurately calculate the distance between the stationary Sensor and Beacon? Phase 2, which incorporates motion of the Beacon, allows for the characterization of accuracy versus range. This will help us understand how well our system functions as the distance between our Sensor and Beacon change. Phase 3, in which both the Sensor and the Beacon are moving allows us the characterize our systems accuracy against the relative speed between the Sensor and Beacon. Phase 4, which incorporates nonlinear dynamics, allows us to characterize our system in the most complete way. This phase of testing also requires the implementation of the Kalman filter. In order to validate this phase of testing, we would generate expected range calculations while injecting normal random noise into our sample data set, and filter this "expected data". Then we would compare the filtered expected data against the filtered collected data in order to verify our systems functioning.

Once the data had been collected and our accuracy has been characterized as previously outlined, the means of the range predictions would be compared against the measured variance and estimated variance. From here we would be able to see if the system is capable of meeting our 2% accuracy requirement outlined in Functional Requirement 3. If the measured variance is greater than predicted, that would indicate that our model had not characterized the errors and losses in our system with high enough accuracy. From here it would be important to revisit our assumptions and find where in our system the extra errors are being introduced. There errors could be in the hardware, like line losses, or SDR clock errors, or there could be errors in our model, such as processing time or Kalman filter errors. Once the error had been found, we would go back and either solve the system error if possible, or update our predictive models and the Kalman filter with new uncertainties. Finally if a bias in the range mean exists (a systematic tendency to either over or under estimate the range) that would indicate a poor calibration of our equipment and a failure to fully address known constant errors, such as SDR processing time or antenna switch timing. If this happens, re-calibration of our instruments and timing errors would need to be done to return the mean bias back to zero. This process is shown below in Figure 32 as a validation and verification flow diagram.

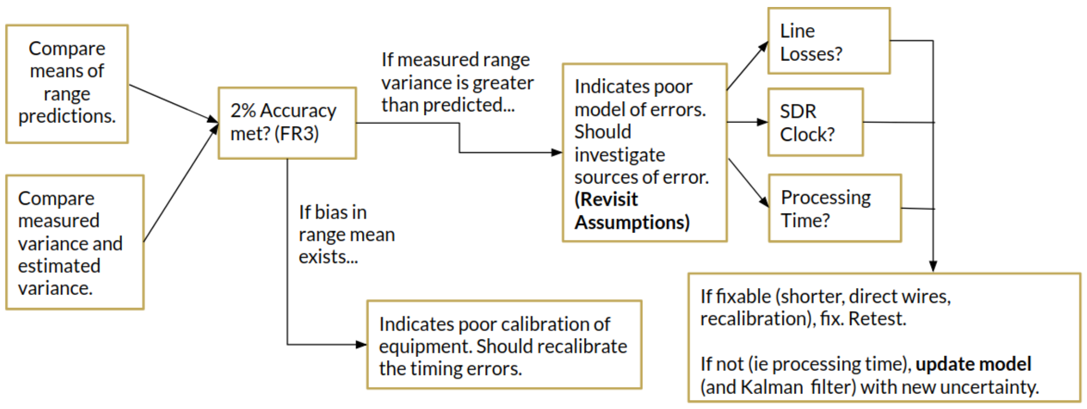

Compare means of range predictions.
Compare measured variance and estimated variance.
2% Accuracy met? (FR3)
If measured range variance is greater than predicted...
Indicates poor model of errors. Should investigate sources of error. (Revisit Assumptions)
Line Losses?
SDR Clock?
Processing Time?
If bias in range mean exists...
Indicates poor calibration of equipment. Should recalibrate the timing errors.
If fixable (shorter, direct wires, recalibration), fix. Retest.
If not (ie processing time), update model (and Kalman filter) with new uncertainty.

Figure 32: CPE1 Test Validation and Verification Decision Flowchart

6.2 CPE 2 - Phase Interferometry

Author(s): Brendan Lutes

A very similar process would be undertaken to validate and verify our second critical project element which directly relates to functional requirements one, two, and three. In order to verify the correct heading determination using phase interferometry, we needed to demonstrate that we could successfully calculate the correct heading from the Sensor to the Beacon 66% of the time. Once again all four levels of testing would have allowed us to validate this capability and characterize the accuracy of our system in the same ways as were outlined in the verification and validation of our first critical project element.

We expected to collect state estimates stored in matrix form, collected at our sample rate. Each vector of data would store the time, the position in rectangular coordinates, and the relative calculated velocities, also stored in rectangular coordinates. A data file set from each test would be compiled in order to allow for comparison between trials and for consistency and redundancy. Truth data would once again be stored from our GPS units, which stores time stamped GPS coordinates. The difference in these coordinates yields the relative position between the Sensor and Beacon for each time stamp.

Once again we had models to predict our expected position values for each phase of the test. Phases 1 and 2 utilized the same model as outlined in the validation of our first critical project element. Phase 3 again must incorporate non zero relative velocities between the Sensor and Beacon, which introduces a unique error for phase interferometry: Doppler shift. Utilizing the Doppler equation for frequency and the equation for artificial wavelength shift, it was calculated that at worse case we would expect a frequency shift of about 100Hz. This shift corresponds to a heading error of .0051 degrees, or about 9mm position error at 100 meters. This is fairly insignificant and is not expected to impact our ability to calculate position within our 2% allowable error. Combining the range and heading errors, we get a total combined position error of 1.68cm at 100 meters, which is once again fairly negligible. The Doppler shift Equation 8 and artificial wavelength shift Equation 9 are given below.

$$f_{observed} = \frac{v+v_0}{v-v_s} f \tag{8}$$

$$\Delta\lambda = \frac{c}{f} - \frac{c}{f_{observed}} \tag{9}$$

Phase 4 once again incorporates the nonlinear dynamics, and therefor the nonlinear Kalman filter. Test data would have once again been generated with normal random error injected into the expected heading calculations to produce a sample expected data set. This expected data set would then be run through our nonlinear Kalman filter to give our our modeled data for Phase 4.

An example graph of this modeled and filtered data for Phase 4 is shown to the right.

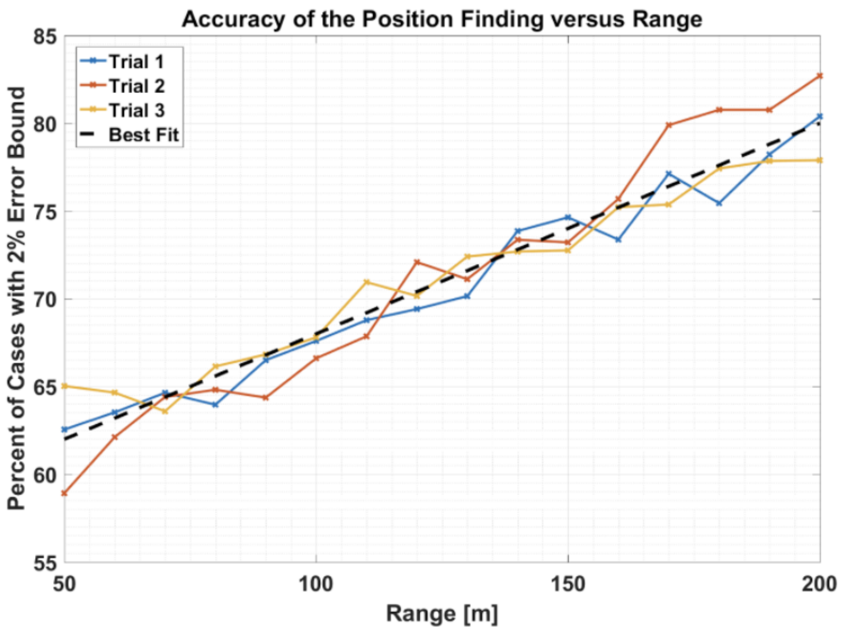


Figure 33: Modeled Data From Phase 4 Testing

Once the experimental and truth data had been collected, we would one again compare the means of position predictions to see if we met our 2% accuracy requirement form Functional Requirement 3. We would also compare the measured variance and estimated variances in our data sets to get a deeper understanding of our full position determination system. Again if measured variances are greater than predicted or if correlation exists, that would indicate a poor model of errors, which once again we would have to revisit, either solving identified hardware issues, or revising our expected errors in the models. If bias in the position mean exists, this once again would indicate a poor calibration of our instruments and known latencies, leading to a re-calibration of our instruments. If errors were found specific to the heading calculation and not within the range calculation, this would indicate poor mounting of some of our antennas, which could be adjusted using kinematic transformation matrices. This process is shown below in Figure 34 as a validation and verification flow diagram.

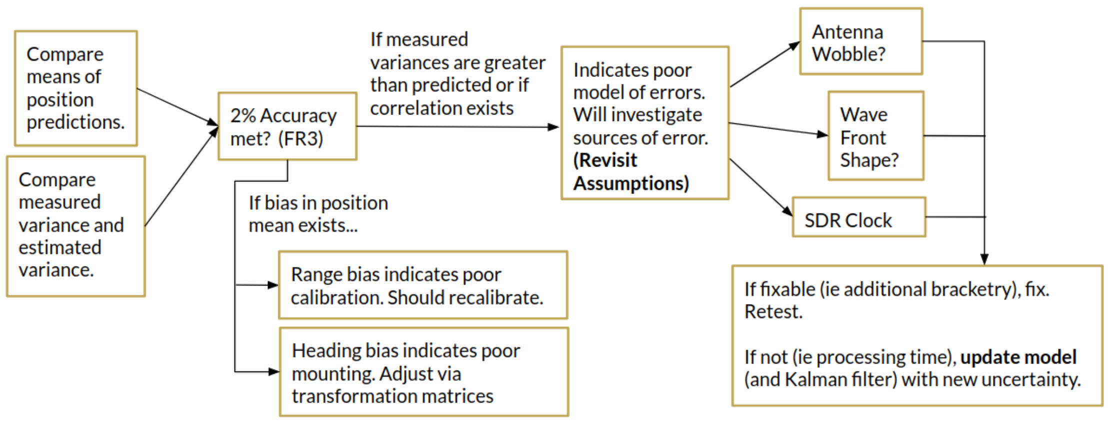

Compare means of position predictions.
Compare measured variance and estimated variance.
2% Accuracy met? (FR3)
If measured variances are greater than predicted or if correlation exists
Indicates poor model of errors. Will investigate sources of error. (Revisit Assumptions)
Antenna Wobble?
Wave Front Shape?
SDR Clock
If bias in position mean exists...
Range bias indicates poor calibration. Should recalibrate.
Heading bias indicates poor mounting. Adjust via transformation matrices
If fixable (ie additional bracketry), fix. Retest.
If not (ie processing time), update model (and Kalman filter) with new uncertainty.

Figure 34: CPE2 Test Validation and Verification Decision Flowchart

6.3 CPE 3 - Data Flow

Author(s): Jamison McGinley, Camilla Hallin, Brendan Lutes

Data flow is an important element of the project for both the time of flight calculation and the overall system update rate. The key models that need to be validated are the system latency and timing budget models. Inconsistent processing time will negatively impact the range calculation. This processing time can be modeled as a Gaussian distribution centered around some mean value. This mean value is calculated with the calibration routine by repeatedly measuring the time for the Beacon to process the incoming poll message. The sample mean and sample standard deviation from this routine are unbiased estimators of the true mean and standard deviation by the law of large numbers. If the standard deviation is too great, the system will not be able to accurately subtract the processing time in the time of flight calculation. Furthermore, each of the process elements will be timed to characterize the complete timing profile of the system. Particularly this is used to determine how the execution time of each subroutine compounds to slow the overall system response rate. FR 3.1.1 indicates that the relative position of the Beacon needs to update at a minimum of 1 Hz to satisfy FR 3.

During all phases of testing, the system tracks overall update rate through the CAN bus. This data is compared to the requirement in FR 3. The additional data captured in the calibration routines to time specific subroutines on the Sensor and Beacon separately is used to identify any system bottlenecks. Identifying bottlenecks will allow for specific targeted action to be taken if FR3 is not initially met in testing.

In the case that the system refresh rate is faster than predicted, it is then possible to add more points into the averaging scheme such that the system would yield higher fidelity state estimates. However, if the system is slower than allowable, several steps could be taken to reduce strain on processing. To solve this issue, the Beacon and Sensor could communicate via a shorter message or fewer points could be used in the averaging scheme. However, as shown in Fig. 25 there is minimum required code length to be able to recover timing from the cross correlation routine. If the transmit code is of insufficient length, the cross correlation peak is not well defined, so the timing is not recoverable.

The GPS system data flow is required to compare the performance of the Sensor and Beacon to an expected path as well as to truth data. In the mission context, the expected path would be known by the flight computer and passed to the Sensor when determining whether the path deviation flag should be set. In the ground based system, the expected path is first generated by driving the GPS units along the test path. The GPS data from each of the GPS units is then converted to a coordinate system of the Beacon relative to the Sensor and loaded to the Flight Computer analog. After loading this expected path, the real system test can be performed. The Sensor and Beacon are again driven on the test path. Meanwhile, the GPS system logs truth data to the data logger. This truth data is collected by the GPS unit independent from the rest of the system and is then compared to the Sensor outputs in post processing. This data flow is demonstrated fully in 35.

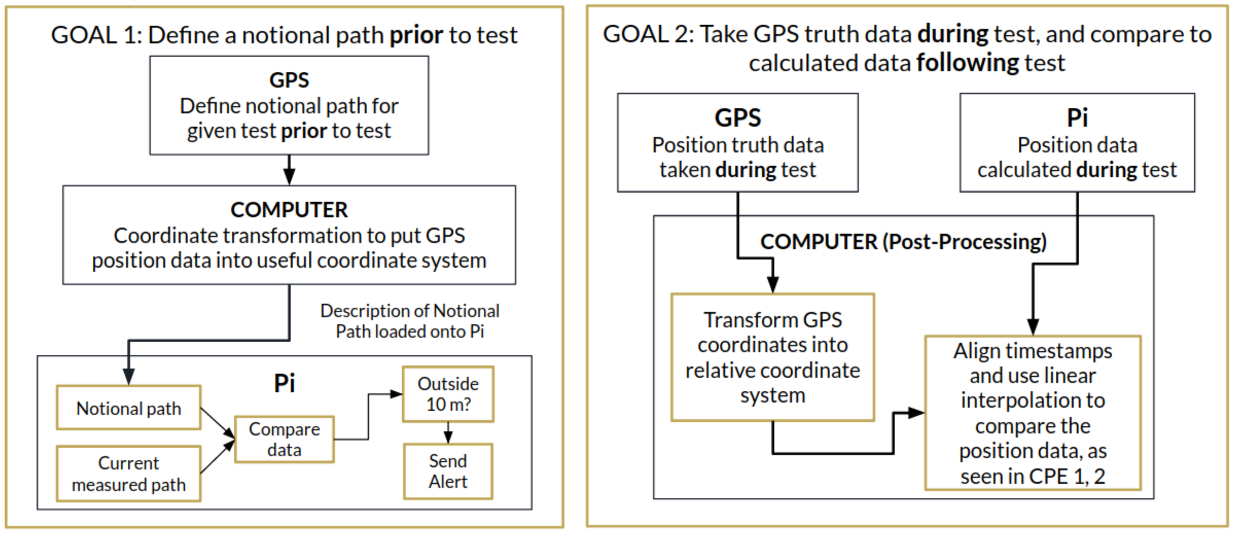

GOAL 1: Define a notional path prior to test
GPS
Define notional path for given test prior to test
COMPUTER
Coordinate transformation to put GPS position data into useful coordinate system
Description of Notional Path loaded onto Pi
Pi
Notional path
Current measured path
Compare data
Outside 10 m?
Send Alert
GOAL 2: Take GPS truth data during test, and compare to calculated data following test
GPS
Position truth data taken during test
Pi
Position data calculated during test
COMPUTER (Post-Processing)
Transform GPS coordinates into relative coordinate system
Align timestamps and use linear interpolation to compare the position data, as seen in CPE 1, 2

Figure 35: GPS system data flow diagram indicating its uses for the project

6.4 CPE 4 - Electrical Components

Author(s): Hunter Peery

Critical project element 4 was designed in order to meet functional requirement 5, all electrical components must operate within the defined temperature constraints of -20^oC to 50^oC. This requirement was due to be tested using General Atomic's thermal chamber during the last testing period. Due to the work stoppage, this test was subsequently cancelled. The test would have allowed a system wide verification that each component operates within the given temperature restraints. A variety of different temperatures were due to be tested, including both outside bounds of the requirement. Although this test never occurred, another method can be used to satisfy the requirement. Thus, each critical project component's data-sheet was sifted through to find the manufacturer's recommended temperature bounds. Those results are conglomerated below.

	Temperature Range	Satisfies FR5?
SDR	-40 to 85 °C	Yes
RaspberryPi	0 to 50°C ambient	No (but demonstrated to function outside datasheet range)
Antennas	Unknown	Unknown
GPS	-40 to 85 °C	Yes
Battery	-20 to 70 °C discharged, -5 to 50 °C charged	Yes for discharged, no for charged
PiCAN/Converter	Unknown	Yes (compatible with Pi)
Data Logger	-40 to 85 °C	Yes
Custom PCB	-25 to 85 °C	Yes
Cables	-65°C to 165°C (estimate)	Yes
Housing	-20 (estimate) to 60°C	Yes

Table 8: Tabulated summary of major components' temperature constraints

The verification of FR5 is confirmed by the manufactures for the SDR, GPS, Data Logger, PCB and electronics. On the other hand, the RaspberryPi does not meet the lower constraint of -20^oC. However, there are several valid tests online demonstrating its feasibility at temperatures well below -20^oC. This would have been confirmed in the thermal chamber. Next, the linear antennas used for RF communication did not have a data-sheet in which their temperature constraints were confirmed. These omni-directional dipole antennas were chosen because they transmitted and received within the S band at a low cost. Not many commercial grade antennas have information available on their appropriate temperature ability, so this would have been another item tested along with the full system test. The battery did not meet the lower temperature bound when charged. Due to safety hazards, this would not have been tested below its lower limit of -5^oC. In addition to the antennas, the PiCAN and Can to USB converter did not information available. It is assumed, however, that these would satisfy the requirement because they are fully compatible with the RaspberryPi which does meet the requirement. In addition, because many different cables were purchased, each individual cable did not have a data-sheet to validate. So, an estimate was created for similar products by the same manufactures. This estimate is well within the governing temperature limits. Finally, the housing of the Beacon and Sensor packages (ABS plastic) did meet the upper bounds of the temperature requirement based on the recommendation of the manufacturer. The lower limit was not defined, but this is well versed material that other suppliers have confirmed is operable to -20^oC therefore meeting the requirement. In summary, every major project element does meet the requirements, except for the unknowns that would have been confirmed in thermal testing.

6.5 CPE 5 - Testing

Author(s): Nate Lee

Although testing has been outlined as a way to verify and validate the rest of the CPEs, it is a CPE by itself. The testing design needs to be able to provide all of the data necessary to satisfy the project requirements. As previously mentioned, there were two main phases to the testing: preliminary tests and field tests. The preliminary tests were initially meant as a simple check on the components, but evolved into part of the entire plan for verification and validation along with the field tests.

The preliminary testing was designed to verify that the components were up to the specifications defined. These tests were designed to build on each other, from the component level to the system level, ensuring that functionality was checked at every step along the way. As seen in the figure below, Figure 36, each part of the preliminary testing levels verified different levels of the component specifications. The preliminary testing in the level one testing that was completed verified all of the individual components to the required specifications.

With the verification steps laid out for the preliminary tests, it was important for the team to validate the tests as well. The level two and three preliminary test were more focused toward validation. Through the level two testing, the sub-systems were able to be validated, including the GPS/Data logger and the Tx/Rx software. The Raspberry Pi/SDR integration testing was nearing completion when testing was suspended. The third test would have validated FR1, FR2, and FR5, as the Beacon and Sensor packages would have sent and received RF signals in the laboratory environment. During this test, the Sensor and Beacon would have been running off of the internal battery, dealing with FR5. To verify and validate the remainder of the requirements, the field testing protocol was developed.

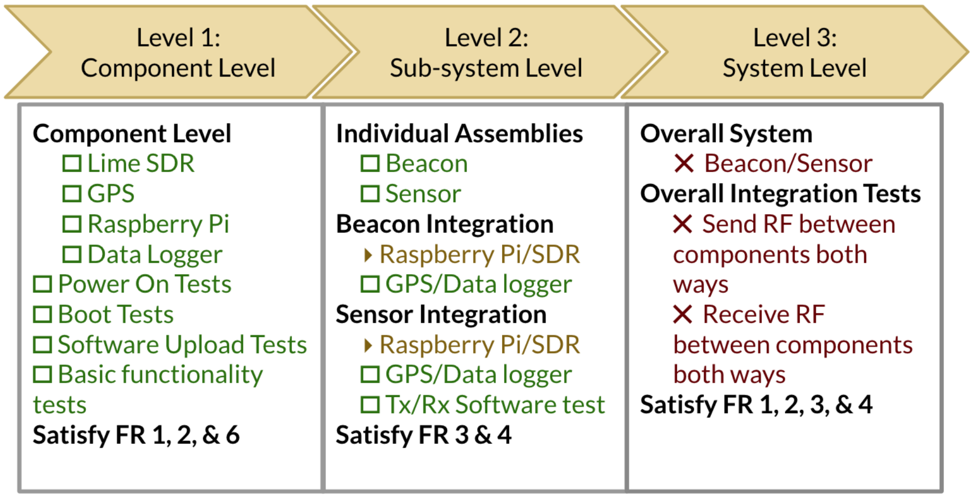
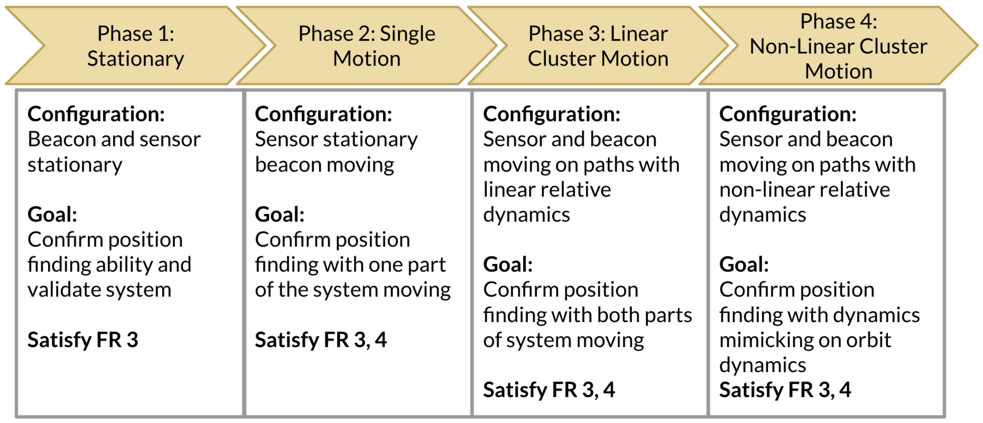


Figure 36: Preliminary Test Flow

The next phase of tests planned after the preliminary tests were the field tests. These tests were designed to verify and validate the functional requirements dealing with the actual position finding. As mentioned in previous sections, the main method of verification for the position finding requirements was through the software models. The field tests were established to validate these models, and gain real-world data for the customer. Similar to the preliminary tests, the field tests were broken up into levels of ascending complexity, with the simplest test performed first. The Phase 1-4 of the field test be seen below in Figure 37.

The field tests were to start by validating stationary position finding, allowing the team to manually control for variables. Each successive field test, through Phase 4, was to become more complex, ending with the Phase 4 test including Beacon and Sensor motion on paths with non-linear dynamics. All of the field tests, Phases 1-4, were designed to validate FR3 and FR4, which were the position finding requirements.

The final functional requirement, FR5 specified the thermal operability requirements. The final field test was a thermal test to take place in the General Atomics thermal chamber. FR5 was verified when components were selected, through the information provided on the data sheets. The thermal field test was designed to validate the thermal performance of the system by ensuring operability of the overall system in the chamber. The successful completion of the field test would have allowed for all of the requirements to be satisfied, indicating overall project success



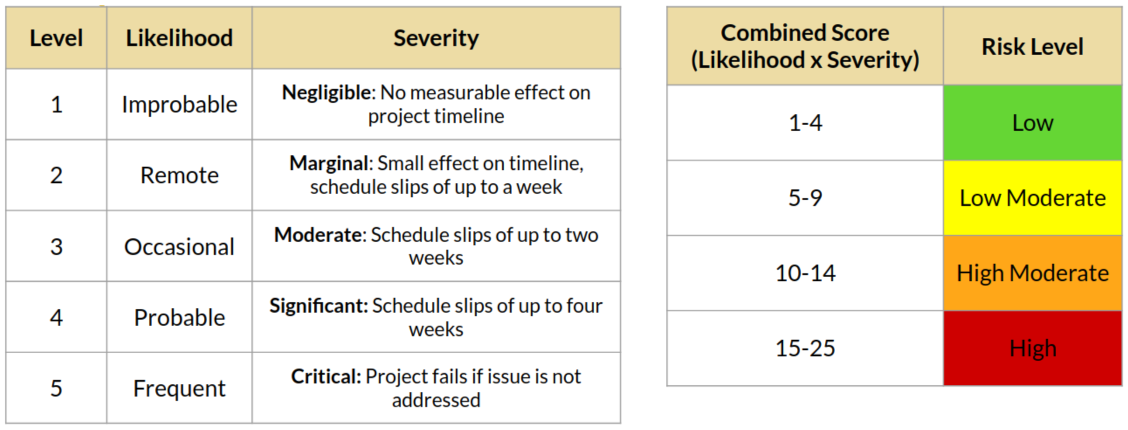
Phase 1: Stationary	Phase 2: Single Motion	Phase 3: Linear Cluster Motion	Phase 4: Non-Linear Cluster Motion
Configuration: Beacon and sensor stationary	**Configuration:** Sensor stationary beacon moving	**Configuration:** Sensor and beacon moving on paths with linear relative dynamics	**Configuration:** Sensor and beacon moving on paths with non-linear relative dynamics
Goal: Confirm position finding ability and validate system	**Goal:** Confirm position finding with one part of the system moving	**Goal:** Confirm position finding with both parts of system moving	**Goal:** Confirm position finding with dynamics mimicking on orbit dynamics
Satisfy FR 3	**Satisfy FR 3, 4**	**Satisfy FR 3, 4**	**Satisfy FR 3, 4**

Figure 37: Field Test Flow

7 Risk Assessment and Mitigation

Author(s): Adam Farmer, Nate Lee

Shown below in Figure 38 is a legend for reading the risk matrices included in this section.



Level	Likelihood	Severity
1	Improbable	**Negligible**: No measurable effect on project timeline
2	Remote	**Marginal**: Small effect on timeline, schedule slips of up to a week
3	Occasional	**Moderate**: Schedule slips of up to two weeks
4	Probable	**Significant**: Schedule slips of up to four weeks
5	Frequent	**Critical**: Project fails if issue is not addressed

Combined Score (Likelihood x Severity)	Risk Level
1-4	Low
5-9	Low Moderate
10-14	High Moderate
15-25	High

Figure 38: Risk Matrix Legend

Major risks and their effects on the project were listed for each sub-team. Mitigation techniques for each of these major risks are discussed for each subteam, then a new risk matrix shows the effects of risk mitigation. Successful risk mitigation will reduce the risk severity and likelihood of each risk in a subteam to a necessary or tolerable level.

Following each subteam's risk mitigation, a short discussion on which of these risks were realized and to what extent they affected the project is presented.

7.1 Software Risks

Major risks for the software sub-team included timing errors, software run time, failure to synchronize data from GPS to measured RF data and sluggish performance from a bloated OS on the Beacon and Sensor.

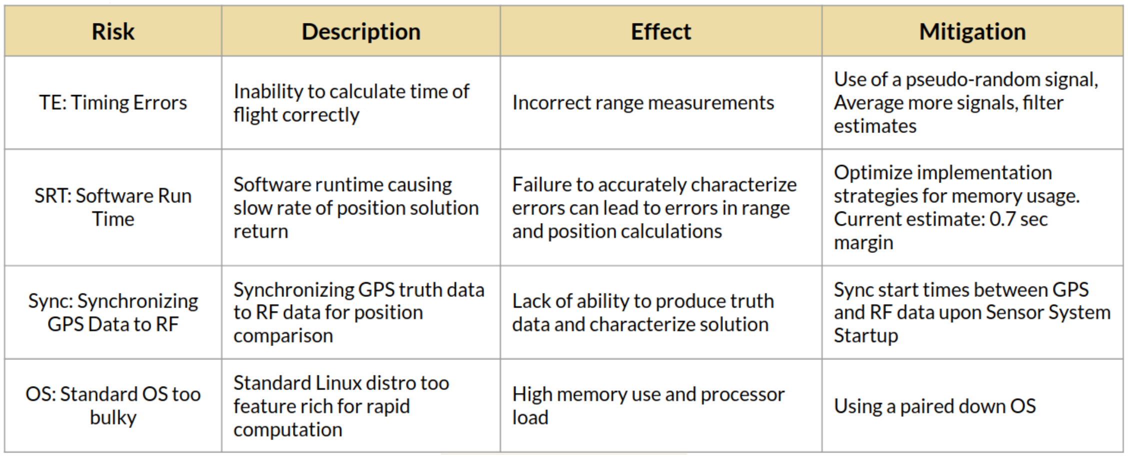
Timing errors in the system were the most probable and most detrimental risk to the software sub-team. Timing errors would propagate directly to large errors in the calculations of range and heading. This is due to the speed of light being so large; any small uncertainty in the time that a measurement of electromagnetic radiation would lead to a large uncertainty in the distance traveled or phase measurement of that radiation.

Software runtime was also considered to be one of the major concerns for the software sub-team. Averaging position and heading solutions is an integral part of the computation of relative Beacon position. Optimization of software runtime would allow for a greater number of averaged solutions and therefore higher confidence in relative position solutions. This would have been done by using optimized algorithms techniques and performing runtime tests both before and after systems integration.

Synchronization of GPS and computed relative position data provided a potential challenge to the software sub-team due to discrepancies in time data between datasets. Since the GPS saves position data to memory for post processing and performance evaluation of the relative position calculation, a miss-match in the time vectors of those datasets would render post-test analysis of the performance of the system useless. In order to mitigate this risk, start times for Sensor package RF data and the GPS module will be marked upon system start-up. Additionally, uncertainties in the timestamps could be mitigated by pulling timestamps further up the RF chain. This was accomplished by switching from GNURadio, an open source SDR interface software, to SoapySDR, another open-source SDR software. The switch allowed for the acquisition of timestamps immediately after analogue-to-digital conversion, or just before digital-to-analogue conversion.

A bulky OS on the flight computers can reduce performance of the system overall due to unnecessary computational overhead. This was to be mitigated by choosing a distribution of the Raspberry Pi OS which does not have a graphical user interface, or any other unnecessary software which is not needed.

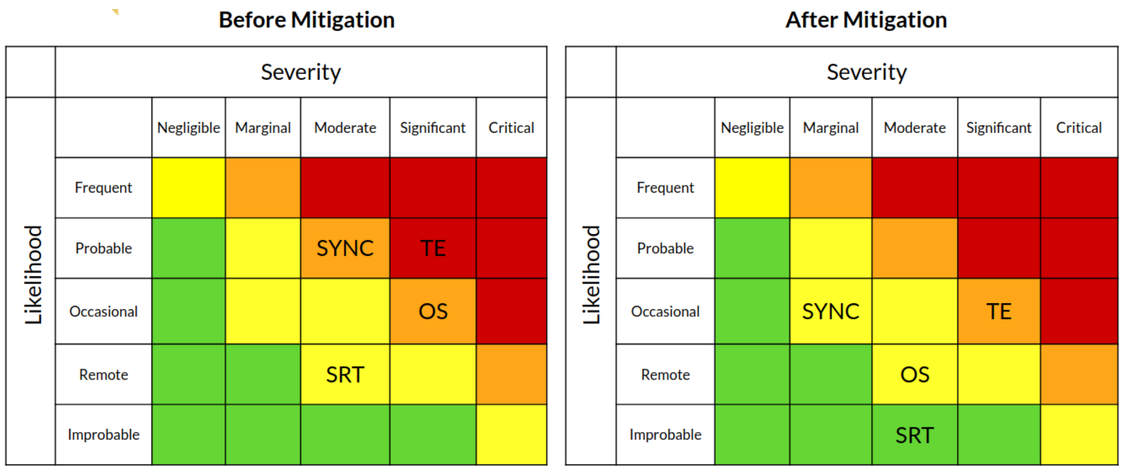
Figure 39 summarizes the major risks associated with the software sub-team of Project ATOMIC.



Risk	Description	Effect	Mitigation
TE: Timing Errors	Inability to calculate time of flight correctly	Incorrect range measurements	Use of a pseudo-random signal, Average more signals, filter estimates
SRT: Software Run Time	Software runtime causing slow rate of position solution return	Failure to accurately characterize errors can lead to errors in range and position calculations	Optimize implementation strategies for memory usage. Current estimate: 0.7 sec margin
Sync: Synchronizing GPS Data to RF	Synchronizing GPS truth data to RF data for position comparison	Lack of ability to produce truth data and characterize solution	Sync start times between GPS and RF data upon Sensor System Startup
OS: Standard OS too bulky	Standard Linux distro too feature rich for rapid computation	High memory use and processor load	Using a paired down OS

Figure 39: Tabulated summary of major risk associated with the software sub-team.

Figure 40 shows the software risk matrix before and after risk mitigation. As can be seen, timing errors in the system were still the largest concern for the software sub-team. Synchronization in timing between computed relative position and GPS position data, bulky OS and software runtime risks were successfully mitigated. However, it is recommended that any future students monitor these risks going forward, especially in the systems integration phase as that is where software optimization could mean the difference between success or failure of the project.



Before Mitigation

Likelihood \ Severity	Negligible	Marginal	Moderate	Significant	Critical
Frequent	🟨	🟧	🟥	🟥	🟥
Probable	🟩	🟨	🟧 SYNC	🟥 TE	🟥
Occasional	🟩	🟨	🟨	🟧 OS	🟥
Remote	🟩	🟩	🟨 SRT	🟨	🟧
Improbable	🟩	🟩	🟩	🟩	🟨

After Mitigation

Likelihood \ Severity	Negligible	Marginal	Moderate	Significant	Critical
Frequent	🟨	🟧	🟥	🟥	🟥
Probable	🟩	🟨	🟧	🟥	🟥
Occasional	🟩	🟨 SYNC	🟨	🟧 TE	🟥
Remote	🟩	🟩	🟨 OS	🟨	🟧
Improbable	🟩	🟩	🟩 SRT	🟩	🟨

Figure 40: Software sub-team risk matrix before and after risk mitigation.

Throughout the component level and sub-system level testing, the risks outlined above did not, at least in theory, hinder the project in any way, see the Verification and Validation section. However, as stated above, these risk should be monitored closely should any future work be pursued, especially during the systems testing phase.

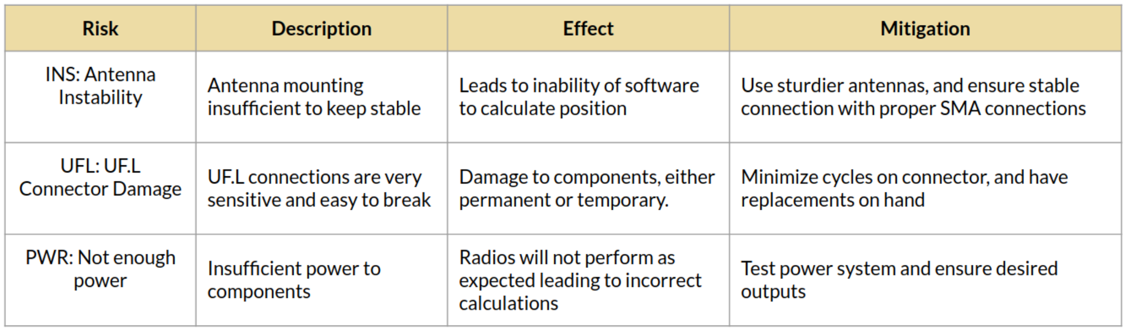
7.2 Hardware Risks

Major risks for the hardware sub-team included: Antenna instability, UFL:UFL connector damage and not getting enough power to peripherals.

Stability of the antennas was important to multiple aspects of the mission. Firstly, characterization of the phase centers of each antenna, and their relative position to each other on the Sensor package is extremely important for the measurement of relative position of the Beacon package. Small displacements in the phase centers of antenna used to calculate range and heading would have lead to errors in relative position of the Beacon package on the order of tens of meters. In order to mitigate this risk, project ATOMIC used sturdy antennas with well characterized phase centers for all transmit and receive operations.

Wear and tear on UFL connectors was largely a function of the number of cycles each connector goes through during subsystem integration, testing and system integration. A broken UFL connector would likely lead to loss of communications between the Sensor and the Beacon package, or loss of heading determination. Luckily, broken UFL connectors will be easy to diagnose and replace. As such, multiple replacement UFL connectors were to remain on-hand. Furthermore, cycling on installed UFL connectors was done minimally and according to manufacture's procedure.

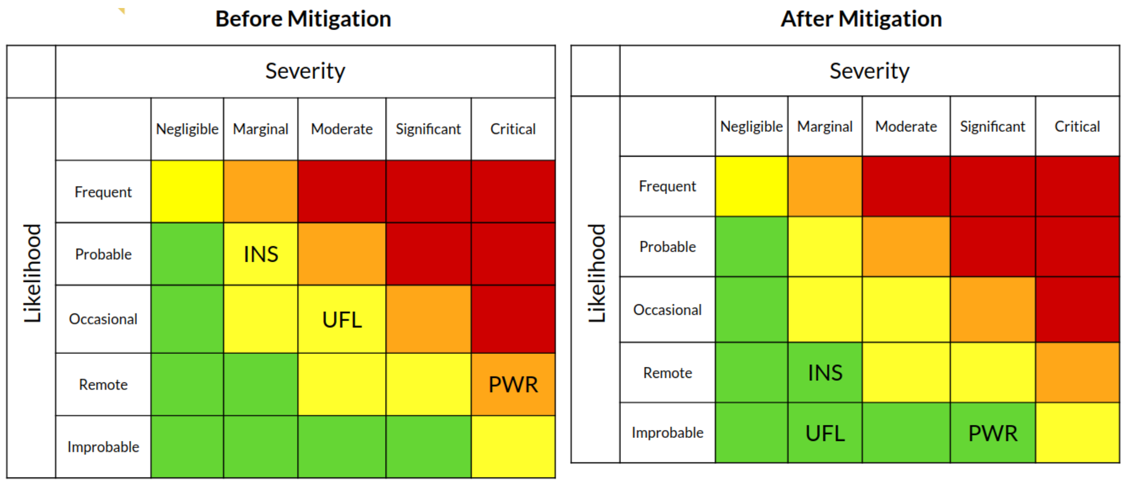
Providing power to all peripherals is important for stable and predictable function on both the Sensor and Beacon package. Failure to provide sufficient power to peripherals could lead to system failure, sporadic behavior, or possible damage to components. In order to mitigate this risk, subsystem and systems testing will be implemented to unsure proper function of both the Sensor and Beacon packages. Testing will include edge-case operations such as max power draw, minimal power draw, and effects due to environmental factors.



Risk	Description	Effect	Mitigation
INS: Antenna Instability	Antenna mounting insufficient to keep stable	Leads to inability of software to calculate position	Use sturdier antennas, and ensure stable connection with proper SMA connections
UFL: UF.L Connector Damage	UF.L connections are very sensitive and easy to break	Damage to components, either permanent or temporary.	Minimize cycles on connector, and have replacements on hand
PWR: Not enough power	Insufficient power to components	Radios will not perform as expected leading to incorrect calculations	Test power system and ensure desired outputs

Figure 41: Tabulated summary of major risk associated with the hardware sub-team.

Figure 42 shows the hardware risk matrix before and after risk mitigation. All risks for this subteam have been successfully mitigated below the "low moderate" risk level as per Figure 38. Further testing will provide insight on additional focus areas for risk mitigation for the hardware subteam.



Before Mitigation

Likelihood \ Severity	Negligible	Marginal	Moderate	Significant	Critical
Frequent	🟨	🟧	🟥	🟥	🟥
Probable	🟩	🟨 INS	🟧	🟥	🟥
Occasional	🟩	🟨	🟨 UFL	🟧	🟥
Remote	🟩	🟩	🟨	🟨	🟧 PWR
Improbable	🟩	🟩	🟩	🟩	🟨

After Mitigation

Likelihood \ Severity	Negligible	Marginal	Moderate	Significant	Critical
Frequent	🟨	🟧	🟥	🟥	🟥
Probable	🟩	🟨	🟧	🟥	🟥
Occasional	🟩	🟨	🟨	🟧	🟥
Remote	🟩	🟩 INS	🟨	🟨	🟧
Improbable	🟩	🟩 UFL	🟩	🟩 PWR	🟨

Figure 42: Hardware sub-team risk matrix before and after risk mitigation.

Throughout the course of the project, there were no major issues of flexing or structural instability in the antennas. During the component level and subsystem level testing that was done, these antennas performed as expected.

Issues of wear and tear on UFL connectors did not provide a significant issue to the project throughout the course of manufacturing, component level or sub-system level testing. Very minor miscellaneous items needed replacement during testing. Namely, one of the miniature SMA wires on one of the GPS units came unattached. However this was an easy fix and required only minor soldering. Due to a significant margin in the budget and careful handling of all hardware, normal wear and tear of components was not a significant risk to the project in general.

Lastly, power to peripherals was provided by a custom built power board which served to fulfill functional requirement 6. Component level testing was performed on the power board, revealing that there was indeed improbable likelihood that peripheral components would be in danger of receiving insufficient power.

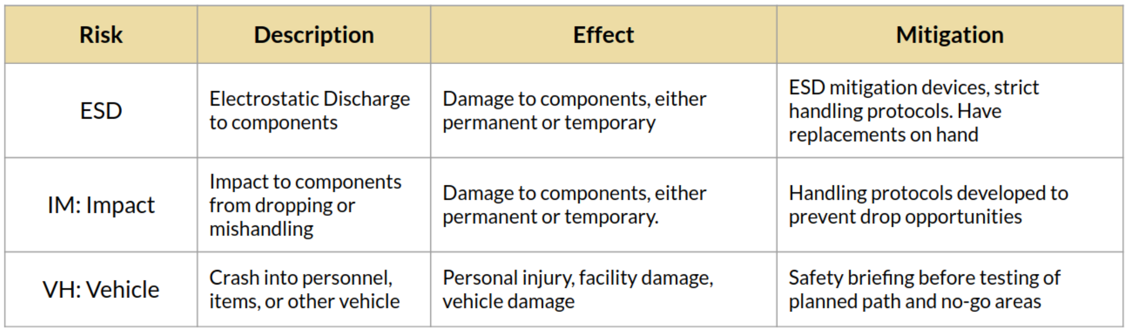
7.3 Testing Risks

The three main risks associated with testing for Project ATOMIC were as follows; Electrostatic Discharge (ESD), Impact (IM) and Vehicle damage (VH). To ensure project success, these risks were to be mitigated. A summary of these risks can be found in Figure 43.

One of the biggest risks to testing was ESD. A number of the components chosen for the project are ESD sensitive, such as the GPS devices and the Lime SDR. Damage to these components from ESD is irreversible and would have caused the loss of the component. To mitigate this, ESD safety equipment was be used while handling these components. This included grounding wristbands and grounding matts. When parts were not actively being used, they were stored in ESD shielded bags. In addition, procedures were developed which formalize the process of handling these components so so as to minimize ESD risk throughout testing and integration. In the case that a component were to be damaged beyond use, at least one replacement was to remain on hand to ensure that the project timeline was not set back significantly.

Components sustaining damage during testing was another potential risk of the testing regime. None of the chosen parts are rugged, and were therefore susceptible to damage from integration cycling or impact due to being handled roughly. This risk was mitigated through the use of formalized handling procedures which ensured proper handling of components during transport and installation.

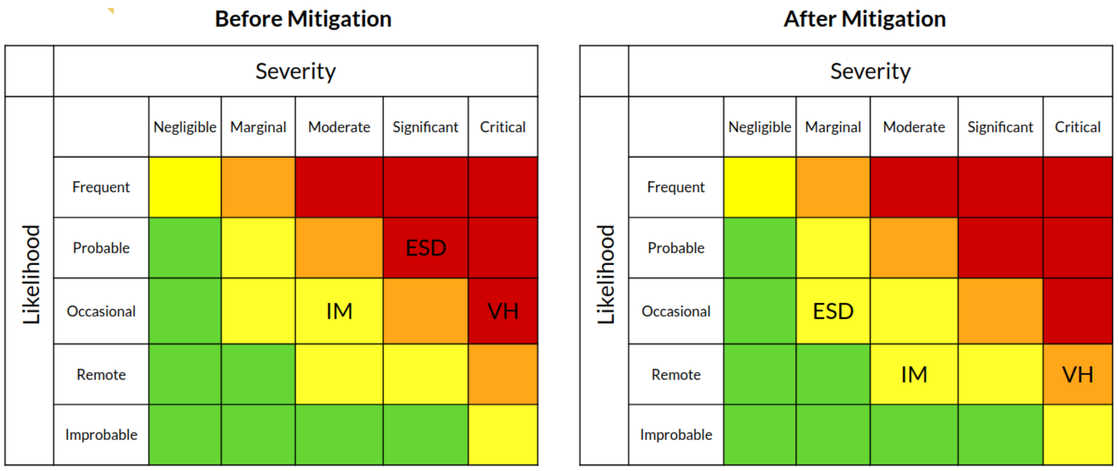
The last, and perhaps most potentially damaging risk, is that of vehicle damage or collision. Testing Phases 2 and 3 involved the use of vehicles to provide the motion needed for the test. Testing with vehicles is inherently risky, as the risk of injury is high when operating machinery in close proximity. Though components can be replaced, injury to a team member is not tolerable under any circumstances. For this reason, the mitigation steps needed to be extremely effective in reducing this risk in particular. The baseline mitigation procedures involved a safety briefing before the tests occurred, with the safety lead laying out the no-go areas and ensuring the entire team was familiar with the test procedure. Phase 3 testing involved motion with two vehicles and therefore would have required more planning and team preparation prior to testing. With a strong safety plan in place, the risk of injury to a team member or vehicle could be reduced to almost zero. That being said, it is important that all team members stay alert and cautious during tests of this nature.



Risk	Description	Effect	Mitigation
ESD	Electrostatic Discharge to components	Damage to components, either permanent or temporary	ESD mitigation devices, strict handling protocols. Have replacements on hand
IM: Impact	Impact to components from dropping or mishandling	Damage to components, either permanent or temporary.	Handling protocols developed to prevent drop opportunities
VH: Vehicle	Crash into personnel, items, or other vehicle	Personal injury, facility damage, vehicle damage	Safety briefing before testing of planned path and no-go areas

Figure 43: Tabulated summary of major risk associated with the testing sub-team.

As seen below in the testing risk matrix, Figure 44, The testing risks were reduced through the mitigation steps outlined above. The first risk, ESD, falls from significant to marginal, and from probable to occasional. The reduction in severity is due to having had replacement parts on hand, and the likelihood reduction was due to the mitigation procedures. Similarly, the likelihood of the other two risks, IM and VH, goes down with the mitigation procedures. The result of this mitigation was that the VH risk became the most significant risk. With the outlined mitigation procedures, the test will be accomplished in the safest manner possible.



Before Mitigation

Likelihood \ Severity	Negligible	Marginal	Moderate	Significant	Critical
Frequent	🟨	🟧	🟥	🟥	🟥
Probable	🟩	🟨	🟧	🟥 ESD	🟥
Occasional	🟩	🟨	🟨 IM	🟧	🟥 VH
Remote	🟩	🟩	🟨	🟨	🟧
Improbable	🟩	🟩	🟩	🟩	🟨

After Mitigation

Likelihood \ Severity	Negligible	Marginal	Moderate	Significant	Critical
Frequent	🟨	🟧	🟥	🟥	🟥
Probable	🟩	🟨	🟧	🟥	🟥
Occasional	🟩	🟨 ESD	🟨	🟧	🟥
Remote	🟩	🟩	🟨 IM	🟨	🟧 VH
Improbable	🟩	🟩	🟩	🟩	🟨

Figure 44: Testing sub-team risk matrix before and after risk mitigation.

During manufacturing, component testing and subsystem testing, no major components were destroyed. Due to being unable to perform the systems tests, the risk associated with VH was not imposed on the team.

7.4 Additional Risks

In addition to the main concerns listed above for each subteam, there where additional miscellaneous risks associated with the project which the team monitored throughout process of manufacturing and component testing. It was difficult to estimate the probability or impact of some of these risks, due to either their complexity or nebulosity. Nevertheless, these were risk which were considered impactful enough to take into consideration and are discussed here for completeness. Such risks include, but are not limited to:

- Charging of LiIon batteries.
- Electromagnetic interference due to peripheral's current draw causing perturbations in the electric field near the phase centers of receiving antennas.
- Battery proximity to software defined radios.
- Sensor antenna array self inteactions causing impedance missmatch.
- Underperformance of peripherals when integrated with the system.

Since LiIon batteries can be dangerous and even cause fires when over charged. The team was sure to handle and charge LiIon batteries according to manufactures specifications and thus this risk will be mitigated. Electromagnetic self interference from on-board electronics could have lead to issues ranging from errors in phase measurements to inability for the Sensor and Beacon to communicate with each other. This problem would be extremely difficult or perhaps impossible to accurately model analytically and therefore would be better dealt with emperically. That is, if interference was found to be an issue, then the package hardware will be rearranged in order to mitigate this risk. Since the Sensor package contains several antenna in close proximity to each other, it is possible that this could negatively and unpredictably affect the ability to measure signals from the Beacon package due to multipathing and perterbations of the electromagnetic near field. These perturbations could lead to a loss of individual antenna gain and efficiency due to missmatched impedance. Underperfomance of electronics such as the microcontrollers, SDRs, or GPS module could have a large affect the estimation of relative position. This risk is probably the hardest to characterize actually having the hardware interacting with each other. Additionally, this risk could be the hardest to mitigate, depending on the specific performance issue.

Unfortunately, the team was unable to perform systems testing and was not able to determine if electromagnetic interference, antenna electromagnetic field interactions, or underperformance of peripherals when integrated into the system was going to be a problem. It is highly advisable that future work on this project focus on the last two items of this list as they are 1. hard to model and 2. have the potential to ruin the project.

8 Project Planning

Authors: Andrew Dellsite, Hunter Peery

8.1 Team Organization Chart

The organization chart of how Project ATOMIC is broken down into subsystems is shown in Fig. 45. Overseeing Project ATOMIC are the team advisor, Dr. Sanghamitra Neogi, and customer, General Atomics. At the top level, the Project Manager is in charge of the organization of the project as a whole and deals with communicating with stakeholders. The next level includes the rest of the administrative team, including the Safety and Finance Officers as well as the Systems Engineer. Under the Systems Engineer, ATOMIC is divided into three main sub-teams: Manufacturing and Testing, Software, and Hardware. The Systems Engineer is in charge of communication between each sub-team and tracking all design changes. Each sub-team is then broken down into subsystems with a team member as the system lead. The manufacturing and Testing sub-team has Manufacturing and Testing subsystems. The Software sub-team has Command and Data Handling, Advanced Development, Position Determination, and RF Modeling subsystems. The Hardware sub-team includes RF Modeling, Hardware, Power Systems subsystems. Each member of ATOMIC, including the administrative leads, has a technical position and contributes directly to the design and development of the project.

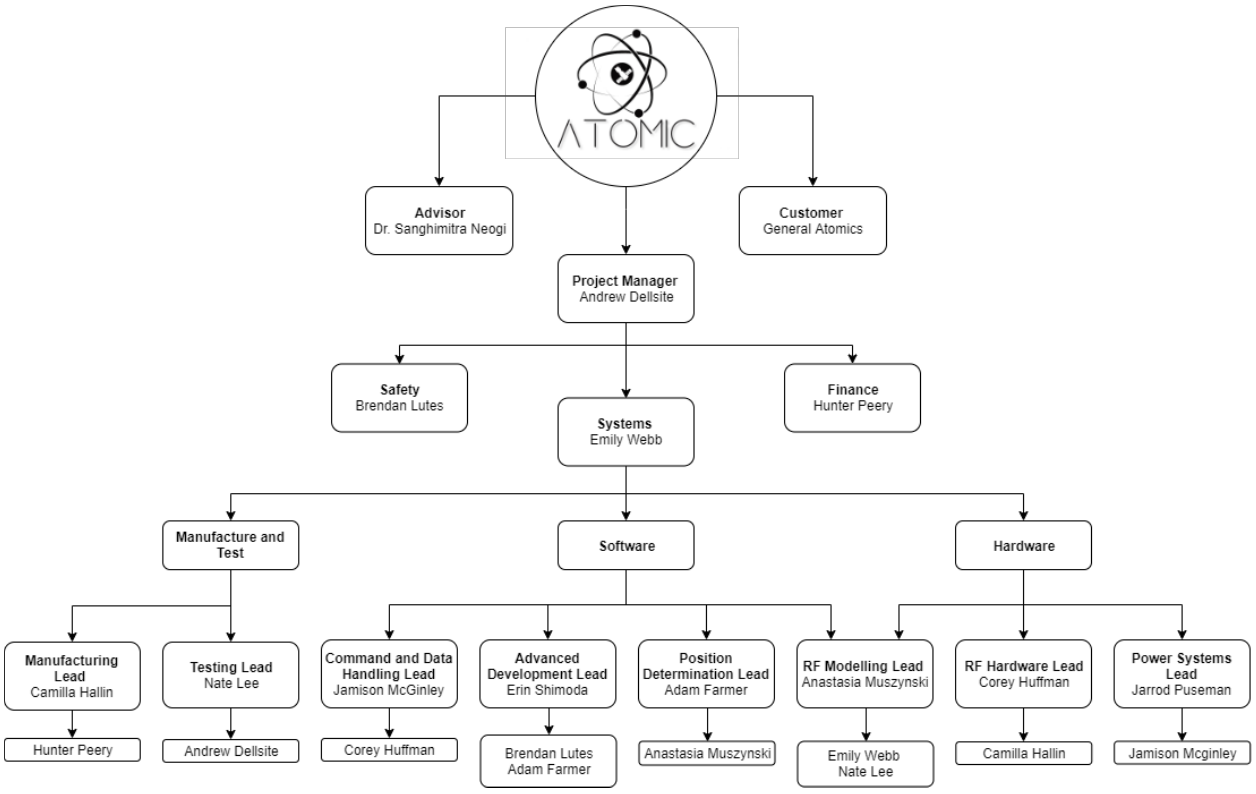


Figure 45: Project ATOMIC Organization Chart

8.2 Work Breakdown

Table 9 is the Work Breakdown Structure (WBS) for the spring semester of Project ATOMIC. A flowdown of each of the tasks is shown in the first column. The tasks marked with * are milestones, marking the completion of a phase within the project process. The duration of each task as well as the expected start and end dates for each task are also shown in the WBS. The last column of the WBS shows the predecessors of each task and how they are related.

The tasks included in the work breakdown structure are the major elements of the project that must be completed to reach the next milestone. Highlighting these major elements for each subteam keeps the project on schedule.

WBS	Name	Duration	Start	Finish	Predecessors
9	Design PCB	21 days	1/6/20	1/27/20	
10	Make PCB	3 days	1/27/20	1/30/20	39FS+2days
11	Purchase Hardware	3 days	12/19/19	12/22/19	
12	Characterize Timing in SDR	10 days	3/06/20	3/16/20	41FS+21days
13	Integrate GPS with RTK	2 days	1/12/20	1/14/20	41FS+21days
14	Develop Heading Calc Code	2 days	1/13/20	1/15/20	
15	Develop Range Finding Code	2 days	1/15/20	1/17/20	

Continued on next page

Table 9 – *Continued from previous page*

WBS	**Name**	**Duration**	**Start**	**Finish**	**Predecessors**
16 *	Finalize Position Finding Code	1 day	1/17/20	1/18/20	44,45
17	Integrate GPS and Data Logger	2 days	1/20/20	1/22/20	43
18	Finalize Preliminary Tests Procedures	5 days	1/6/20	1/11/20	
19	Make Safety Procedures for Testing	5 days	1/13/20	1/18/20	
20 *	CDR at General Atomics	1 day	1/17/20	1/18/20	
21 *	*Manufacturing Status Review*	21 days	1/13/20	2/3/20	
22	Construct Beacon and Sensor Structures	4 days	1/26/20	1/30/20	
23	Preliminary Test 1	2 days	1/27/20	1/29/20	48,42,47,49
24 *	Integrate Pi and SDR	28 days	2/2/20	3/01/20	53FS+4days,52
25	Preliminary Test 2	28 days	2/10/20	3/09/20	54FS+5days,46
26	Preliminary Test 3	4 days	3/10/20	3/14/20	55FS+5days
27	Finalize Field Tests Procedures	5 days	1/20/20	1/25/20	
28 *	*Test Readiness Review*	28 days	2/3/20	3/2/20	
29	Construct Testing Equipment	7 days	2/10/20	2/17/20	
30 *	*Spring Final Review*	49 days	3/2/20	4/20/20	
31 *	Phase One - Test	3 days	3/14/20	3/17/20	56FS+7days,57
32	Phase Two - Test	3 days	3/24/20	3/27/20	61FS+7days
33	Phase Three - Test	3 days	4/3/20	4/6/20	62FS+7days
34	Phase Four - Test	3 days	4/13/20	4/16/20	63FS+7days
35	GA Thermal Test	1 day	4/13/20	4/14/20	61
36 *	*Senior Design Symposium*	1 day	4/23/20	4/24/20	
37 *	*Project Final Report*	14 days	4/20/20	5/4/20	

Table 9: Spring Work Breakdown Structure

The main tasks to accomplish for Project ATOMIC from the work breakdown structure are displayed in the Fall, Spring, and Testing Gantt Charts, Figure 46, Figure 47, and Figure 49 respectively. Each chart shows the scheduling for the tasks that make up Project ATOMIC. Milestones are marked throughout the schedule by diamonds. Each task is allocated to a sub-team, marked by the color of the individual task. Tasks assigned to Hardware, Software, and Manufacturing and Testing sub-teams are marked in orange, purple, and green, respectively. Tasks that are blue require the entire team to contribute to their completion.

The critical path for the Fall schedule, Figure 46, is marked in red and is defined as the path with the least amount of float time. Along this path includes the tasks necessary to complete the course assignments, including the Project Definition Document, Conceptual Design Document, Preliminary Design Review, and Critical Design Review. The total float time along the critical path for the fall was 6 days. There was not a lot of slack given to tasks during the fall semester because of the low risks associated with these tasks.

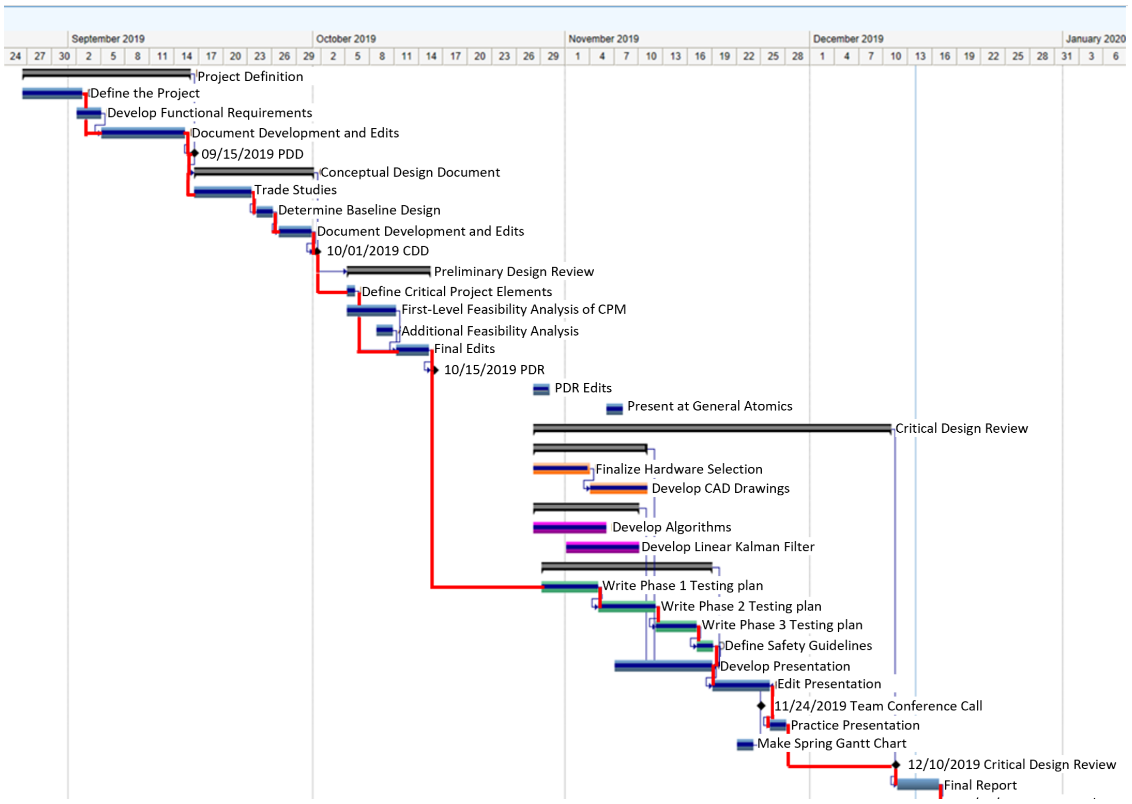


Figure 46: Project ATOMIC Fall Gantt Chart

The critical path for the Spring schedule, Figure 47, is marked in red. Tasks along the critical path have the greatest impact to the schedule given any time delays. The tasks that were to be completed during the spring semester involved many risks and were given more float time accordingly. The critical path includes each phase of the software development and the integration of all the scripts. This process was given 4 days of float time to allow for additional debugging if necessary. Then, each hardware component was tested for functionality, with 5 days of float time to ensure the team understood how each component operated individually. Integrating the Pi and SDR was the main task worked on during the spring semester. This critical task was was allotted 4 weeks with an additional week of float time to complete. Next, the components were to be assembled into the Sensor and Beacon packages. Each package would have been individually tested to verify they can send and receive signals as well as verify proper data flow within each system. Once each package was operational, 4 field tests and a thermal test would have been conducted in succession, only moving on to the next testing phase after the completion of the previous phase, to verify each of the functional requirements. Each testing phase was given 5 days of float time to insure completion before the next phase commences. Additionally, all course assigned tasks are marked along the critical path.

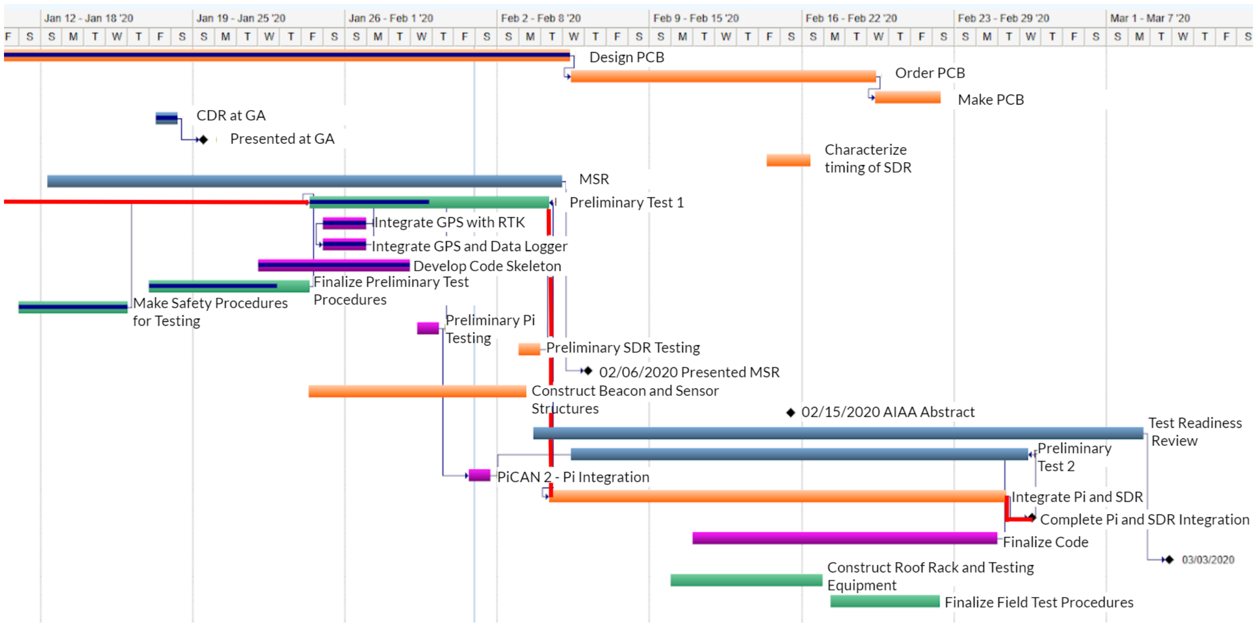


Figure 47: Project ATOMIC Spring Gantt Chart

8.3 Cost Plan

The available budget for this project was the allotted $5,000 for the Aerospace department plus an additional $1,000 from the Engineering Excellence Fund. $4,750 was spent before the project was halted, with an additional $200 projected to complete the project. The only items left to complete testing would have been safety gear for the vehicular tests. With these numbers, the project would have cost a total of $4,950. From the initial budget estimate of $5,000 developed at CDR, Project ATOMIC was almost 99% accurate to this value. The extra money available allowed for a 15% margin from the initial $6,000 should any unforeseen complication have arisen. To avoid setting back the critical path, almost ever major component had a backup. For example, only 3 SDRs were required to complete the project; however, an additional was purchased in case of failure. This helped the project stay on schedule and avoid waiting for long lead and shipping times. The only project component without a backup was the custom PCB board. Due to the extended fabrication time, only one was purchased. Had this broken, testing would have been set back at least two weeks. Overall, the project stayed very close to its budget with few problems arising.

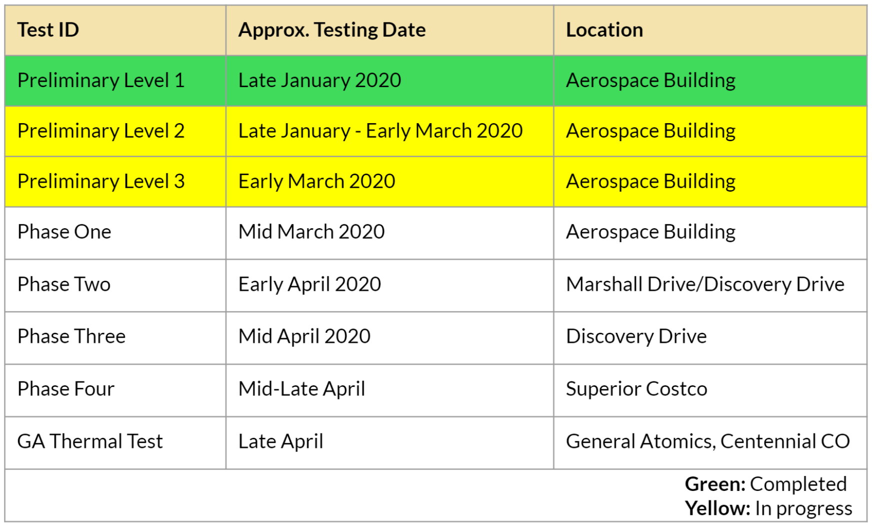
	Cost per Item	Quantity	Total Cost	Lead/Shipping Time
SDR	$299	4	$1,196	3 Weeks
Raspberry Pi	$55	3	$176.95	1 Week
Antennas	$5.20	9	$46.79	1-2 Weeks
GPS	$200	2	$400	2 Days
Battery	$207	3	$621.26	2-3 Weeks
PiCAN/Converter	$124.95	3	$374.85	1-2 Weeks
Data Logger	$58.95	2	$177.90	2 Days
Custom PCB	$98.30	1	$98.30	3-4 weeks
Cables	N/A	N/A	$310	1 Week
Housing	N/A	N/A	$612.69	2-3 Days
Testing Equipment	N/A	N/A	$62.35	1 Week
Electronics	N/A	N/A	$142	1 Week
Miscellaneous	N/A	N/A	$731	1 Week
Total			$4,950	

Table 10: Cost Plan for major components

Note, several items above have a N/A symbol for their cost per item. The reason for this is they are a conglomeration of different parts from different manufactures, so price varied with each individual purchase. Instead of listing every specific part, similar are contained within one category. For example, all the resistors, capacitors, stitches, etc are contained within the electronics section while cables have their own. The final cost of each category represents the additive total of all parts contained within, giving an accurate representation of the total money spent.

8.4 Test Plan

The Test Plan (TP) for ATOMIC is shown in Figure 48. The tests highlighted in green or in yellow are complete or were in progress when testing and manufacturing stopped, respectively. Once the three preliminary levels tests were completed, verifying functionality of the Beacon and Sensor packages, the field tests would have then been conducted in order. These field tests were scheduled to be conducted from mid-March through mid-April at various locations outlined in the third column of 48. Each testing locations the team had access to and provided the proper working space necessary to complete each test phase. The team contacted Costco and presented safety and testing plans to gain access for the Phase 4 testing location. A thermal test at General Atomics, the team's customer, was secluded to be conducted once all of the field tests were completed.



Test ID	Approx. Testing Date	Location
Preliminary Level 1	Late January 2020	Aerospace Building
Preliminary Level 2	Late January - Early March 2020	Aerospace Building
Preliminary Level 3	Early March 2020	Aerospace Building
Phase One	Mid March 2020	Aerospace Building
Phase Two	Early April 2020	Marshall Drive/Discovery Drive
Phase Three	Mid April 2020	Discovery Drive
Phase Four	Mid-Late April	Superior Costco
GA Thermal Test	Late April	General Atomics, Centennial CO

Green: Completed
Yellow: In progress

Figure 48: Project ATOMIC Test Plan

The testing schedule for the project is shown in Figure 49. The red line marks the critical path through each of the testing phases. The thermal test at General Atomics is not along the critical path as this test could be conducted at anytime, though it was best to be conducted after the field test were completed. This Gantt chart also displays the float time between each testing phase.

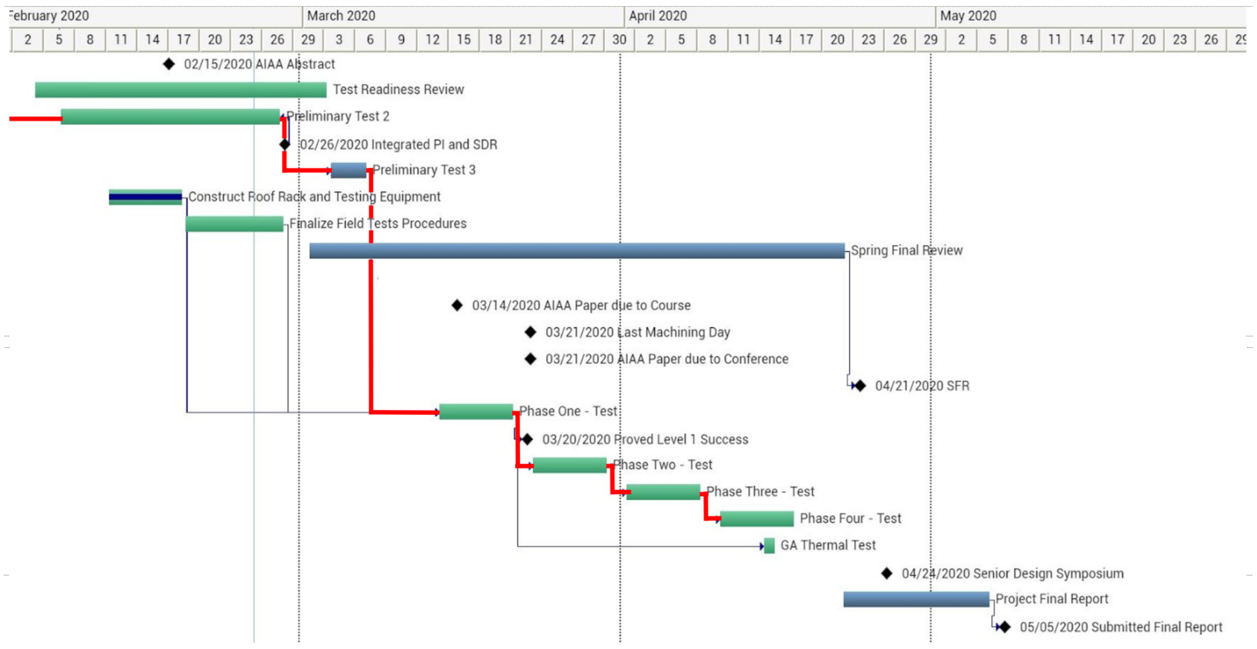


Figure 49: Project ATOMIC Testing Schedule

9 Lessons Learned

“In every job that must be done, there’s an element of fun. Find the fun, and snap: The job’s a game!"
– Julie Andrews as Mary Poppins

This project was almost entirely comprised of lessons to be learned: not only would we be delving into electrical engineering and RF system design, but we would also be running an industry-level project for the first time, complete with scheduling and budgetary concerns. In all, our general engineering knowledge increased tremendously throughout the course of our senior project, and all graduating engineers will be better for the experience. Below we've listed some of the key teachings we as a team will take away from this project:

– *Communication is one of the most important facets of a working relationship.* While we each had individual jobs, and different areas in which we specialized, the team as a whole had to know the status of the project at each step of the way. Communicating between team members, and setting up a structure to do this effectively, makes each member more efficient.

– Documentation goes a long way to help team members communicate and keep everyone up-to-date on the status of the project. *For future senior projects teams:* document everything. When it comes time for presentations, reports, and archives, you'll be glad you did.

– *Management style is important.* This team worked well in part because Andrew provided a management style that worked well with the team dynamics. He let people do their work independently, which aided the production of the team.

– *Procurement delays are to be expected.* Several items in this project were flagged by the government and had to be validated for their purpose before acquiring. This led to long logistical delays and communication issues. Also, different companies has varying customer service representation leading to long delays for simple issues like ridding of sales tax. In addition, sometimes manufactures ran out of a product after purchase, and won't have it re-stocked for months. The lessons learned here are organization is vital for communication with different vendors, otherwise the project could be delayed. These issues should be included in the WBS and should allow for some float time.

– *Make the work enjoyable.* We found ourselves initially on a project that wasn't anyone's first choice, but while defining the scope of the project, it became a project we could all enjoy.

– *Don't be afraid to ask for help.* As aerospace engineers, we had little experience with RF and other electrical engineering topics. As such, we reached out to professors and fellow students who were more knowledgeable in those areas to help us, and we wouldn't have gotten nearly as much done if we hadn't.

– *Team composition matters.* This team was very successful in part because of the strong interpersonal dynamics. Any project will have its ups and downs, but what will bring you through them is the team members you work with, and the support they provide each other.

– *Don't forget to have fun.* Stressing out unnecessarily about the success of the project won't help anyone, so make sure to try and have fun with it. Whether it's weekend lab trips or late-night report writing sessions, involve all members of the team and try to make it an enjoyable experience.

– *No one is perfect.* It is important to remember that even the smartest of us will sometimes make mistakes. At the end of the day, teams are comprised of people, and no one is perfect.

- *Assume the best intentions.* You never truly know what someone is going through at any point in time. Therefore, working to remain in a state where one assumes the best intentions from others can make for a much less stressful endeavor.

10 Contributions

Andrew Dellsite: Project Planning, Editing

Adam Farmer: Risk Analysis Section, Lessons learned.

Camilla Hallin: Con-ops, Manufacturing, Verification and validation

Corey Huffman: Basic Design Overview, Power System Design, Electrical

Nate Lee: Baseline Design, Verification and Validation, Risk Assessment, Lessons Learned.

Brendan Lutes: Verification and Validation, Lessons Learned

Jamison McGinley: Kalman Filter, Manufacturing Software, Verification and Validation

Anastasia Muszynski: Design Process and Trade Studies, Verification and Validation, General Editing and formatting.

Hunter Peery: Verification and Validation, Project Planning, Lessons Learned

Jarrod Puseman: Design Process and Overview, Manufacturing

Erin Shimoda: Design Process and Outcome, Lessons Learned, general editing and formatting

Emily Webb: Project Purpose, Fundamental Requirements, Design Requirements, Lessons Learned, general editing and formatting

References

[1] Werner, D. (Aug 2018). "Small Satellites are at the Center of a Space Industry Transformation." *Space News*. Online. Accessed: `spacenews.com/small-satellites-are-at-the-center-of-a-space-industry-transformation/`

[2] Harris, M. (Jan 2019) "Swarm Wants to Send Hundreds of Tiny CubeSats into Orbit." *IEEE Spectrum: Technology, Engineering, and Science News*. Online. Accessed: `spectrum.ieee.org/tech-talk/aerospace/satellites/swarm-wants-to-fly-the-sky-with-tiny-cubesats`

[3] Ethier, Vince, Voronka, Nestor, Newton, Tyrel, Gagnon, Peter, Chandler, Alan, Hicks, Mat, and Hoyt, Rob. "Enabling Software Defined Radio Technology for High Performance Coordinated Constellations." Tethers Unlimited, Inc. Retrieved September 15, 2019 from `mstl.atl.calpoly.edu/~workshop/archive/2013/Summer/Day%202/1545-Ethier-EnablingSDRTech.pdf`

[4] Taoglas Limited. Digikey. "TG.09.0113W Datasheet" https://www.digikey.com/product-detail/en/taoglas-limited/TG.09.0113W/931-1039-ND/2332666&?gclid=Cj0KCQiAz53vBRCpARIsAPPsz8VHkwpQ
t1nDRmj8H5vjO2OL-wNJG8xKCoegTNdzGSPIQ8QaAlk8EALw_wcB. Accessed Dec. 15, 2019.

[5] "OpenSWIFT-SDR for STRS." (2017) *National Aeronautics and Space Administration.* Retrieved 14 December 2019. `sbir.gov/sbirsearch/detail/1425793`

[6] Scott, Jeff. "Drag of Cylinders & Cones" (2005) *aerospaceweb.org*. Retrieved 10 November 2019. `http://www.aerospaceweb.org/question/aerodynamics/q0231.shtml`

11 Appendix

11.1 Trade Tables

Table 11: Integrated Radio vs SDR Trade Matrix

	Weighting	SDR	Integrated
Cost	10%	1	4
Weight	15%	3	4
Complexity	30%	3	5
Power Consumption	15%	1	5
Capability	30%	5	3
Overall	100%	3.1	4.2

Table 12: Trading Different Antennas

	Weighting	Patch	Clover Leaf	Dipole
Price	25%	5	5	3
Software Complexity	15%	5	5	5
Mounting Ease	20%	1	2	5
Accuracy	40%	4	3	5
Overall	100%	3.4	3.6	4.5

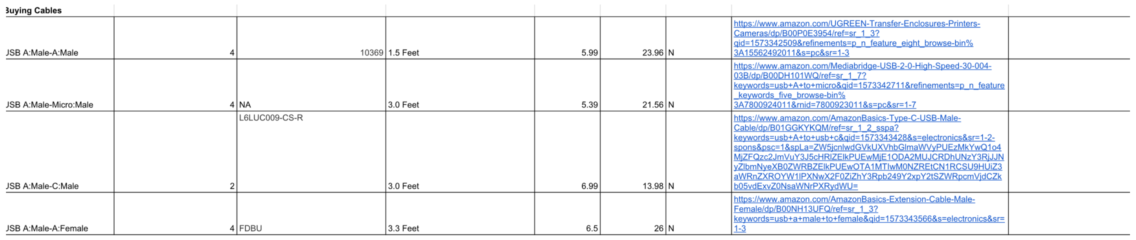
Table 13: Testing Trade Matrix

	Cost	Speed and Range	Control Ability	Restrictions and Feasibility	System Capability	Weight Capability	Totals
Motorized Balloon	1	3	3	3	4	3	2.6
Un-Motorized Balloon	5	2	2	4	2.5	4	3.4
Drone	3	5	4.5	4	4	2	3.6
Car	5	5	4	3	4	5	4.5
RC Car	4.5	4	3	4	4	2.5	3.7
Custom Track Solution	2	2	5	4	5	1-5	3.8

Table 14: Trading Position Determination Methods

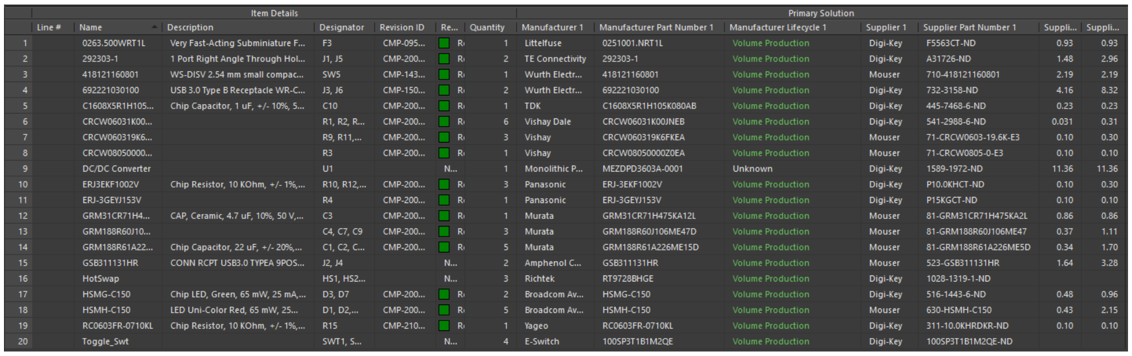
Weighting	KRC	TDOA	PDDF	PI	Watson Watt
Solution Rate	25%	5	5	5	1
Software Complexity	15%	3	3	3	1
Hardware Complexity	20%	3	3	3	3
Accuracy	30%	1	1	5	5
Interference Concerns	10%	3	3	3	5
Overall	100%	2.9	2.9	4.1	2.5

11.2 Power Board Purchasing List



Buying Cables								
USB A:Male-A:Male	4	10369	1.5 Feet	5.99	23.96	N	https://www.amazon.com/UGREEN-Transfer-Enclosures-Printers-Cameras/dp/B00P0E3954/ref=sr_1_3?qid=1573342509&refinements=p_n_feature_eight_browse-bin%3A15562492011&s=pc&sr=1-3	
USB A:Male-Micro:Male	4	NA	3.0 Feet	5.39	21.56	N	https://www.amazon.com/Mediabridge-USB-2-0-High-Speed-30-004-03B/dp/B00DH101WQ/ref=sr_1_7?keywords=usb+A+to+micro&qid=1573342711&refinements=p_n_feature_keywords_five_browse-bin%3A7800924011&rnid=7800923011&s=pc&sr=1-7	
USB A:Male-C:Male	2	L6LUC009-CS-R	3.0 Feet	6.99	13.98	N	https://www.amazon.com/AmazonBasics-Type-C-USB-Male-Cable/dp/B01GGKYKQM/ref=sr_1_2_sspa?keywords=usb+A+to+usb+c&qid=1573343428&s=electronics&sr=1-2-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzMkYwQ1o4MjZFQzc2JmVuY3J5cHRlZElkPUEwMjE1ODA2MUJCRDhUNzY3RjJJNyZlbmNyeXB0ZWRBZElkPUEwOTA1MTIwM0NZREtCN1RCSU9HUiZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=	
USB A:Male-A:Female	4	FDBU	3.3 Feet	6.5	26	N	https://www.amazon.com/AmazonBasics-Extension-Cable-Male-Female/dp/B00NH13UFQ/ref=sr_1_3?keywords=usb+a+male+to+female&qid=1573343566&s=electronics&sr=1-3	

Figure 50: Purchasing list for cables needed for connections



	Item Details							Primary Solution						
	Line #	Name	Description	Designator	Revision ID	Re...	Quantity	Manufacturer 1	Manufacturer Part Number 1	Manufacturer Lifecycle 1	Supplier 1	Supplier Part Number 1	Suppli...	Suppli...
1		0263.500WRT1L	Very Fast-Acting Subminiature F...	F3	CMP-095...	🟩 R	1	Littelfuse	0251001.NRT1L	Volume Production	Digi-Key	F5563CT-ND	0.93	0.93
2		292303-1	1 Port Right Angle Through Hol...	J1, J5	CMP-200...	🟩 R	2	TE Connectivity	292303-1	Volume Production	Digi-Key	A31726-ND	1.48	2.96
3		418121160801	WS-DISV 2.54 mm small compac...	SW5	CMP-143...	🟩 R	1	Wurth Electr...	418121160801	Volume Production	Mouser	710-418121160801	2.19	2.19
4		692221030100	USB 3.0 Type B Receptacle WR-C...	J3, J6	CMP-150...	🟩 R	2	Wurth Electr...	692221030100	Volume Production	Digi-Key	732-3158-ND	4.16	8.32
5		C1608X5R1H105...	Chip Capacitor, 1 uF, +/- 10%, 5...	C10	CMP-200...	🟩 R	1	TDK	C1608X5R1H105K080AB	Volume Production	Digi-Key	445-7468-6-ND	0.23	0.23
6		CRCW06031K00...		R1, R2, R...	CMP-200...	🟩 R	6	Vishay Dale	CRCW06031K00JNEB	Volume Production	Digi-Key	541-2988-6-ND	0.031	0.31
7		CRCW060319K6...		R9, R11,...	CMP-200...	🟩 R	3	Vishay	CRCW060319K6FKEA	Volume Production	Mouser	71-CRCW0603-19.6K-E3	0.10	0.30
8		CRCW08050000...		R3	CMP-200...	🟩 R	1	Vishay	CRCW08050000Z0EA	Volume Production	Mouser	71-CRCW0805-0-E3	0.10	0.10
9		DC/DC Converter		U1		N...	1	Monolithic P...	MEZDPD3603A-0001	Unknown	Digi-Key	1589-1972-ND	11.36	11.36
10		ERJ3EKF1002V	Chip Resistor, 10 KOhm, +/- 1%,...	R10, R12,...	CMP-200...	🟩 R	3	Panasonic	ERJ-3EKF1002V	Volume Production	Digi-Key	P10.0KHCT-ND	0.10	0.30
11		ERJ-3GEYJ153V		R4	CMP-200...	🟩 R	1	Panasonic	ERJ-3GEYJ153V	Volume Production	Digi-Key	P15KGCT-ND	0.10	0.10
12		GRM31CR71H4...	CAP, Ceramic, 4.7 uF, 10%, 50 V,...	C3	CMP-200...	🟩 R	1	Murata	GRM31CR71H475KA12L	Volume Production	Mouser	81-GRM31CR71H475KA2L	0.86	0.86
13		GRM188R60J10...		C4, C7, C9	CMP-200...	🟩 R	3	Murata	GRM188R60J106ME47D	Volume Production	Mouser	81-GRM188R60J106ME47	0.37	1.11
14		GRM188R61A22...	Chip Capacitor, 22 uF, +/- 20%,...	C1, C2, C...	CMP-200...	🟩 R	5	Murata	GRM188R61A226ME15D	Volume Production	Mouser	81-GRM188R61A226ME5D	0.34	1.70
15		GSB311131HR	CONN RCPT USB3.0 TYPEA 9POS...	J2, J4		N...	2	Amphenol C...	GSB311131HR	Volume Production	Mouser	523-GSB311131HR	1.64	3.28
16		HotSwap		HS1, HS2...		N...	3	Richtek	RT9728BHGE	Volume Production	Digi-Key	1028-1319-1-ND		
17		HSMG-C150	Chip LED, Green, 65 mW, 25 mA,...	D3, D7	CMP-200...	🟩 R	2	Broadcom Av...	HSMG-C150	Volume Production	Digi-Key	516-1443-6-ND	0.48	0.96
18		HSMH-C150	LED Uni-Color Red, 65 mW, 25...	D1, D2,...	CMP-200...	🟩 R	5	Broadcom Av...	HSMH-C150	Volume Production	Mouser	630-HSMH-C150	0.43	2.15
19		RC0603FR-0710KL	Chip Resistor, 10 KOhm, +/- 1%,...	R15	CMP-210...	🟩 R	1	Yageo	RC0603FR-0710KL	Volume Production	Digi-Key	311-10.0KHRDKR-ND	0.10	0.10
20		Toggle_Swt		SWT1, S...		N...	4	E-Switch	100SP3T1B1M2QE	Volume Production	Digi-Key	100SP3T1B1M2QE-ND		

Figure 51: Bill of materials for power board

11.3 Drawings

Included here are all technical drawings for each part needed to be manufactured in this project. Since the part number and purpose are given on each document as well as the simplicity of the project hardware, no further descriptions are given of each part. To get a better understanding of where each part goes, refer to CAD renderings of the project throughout the report.

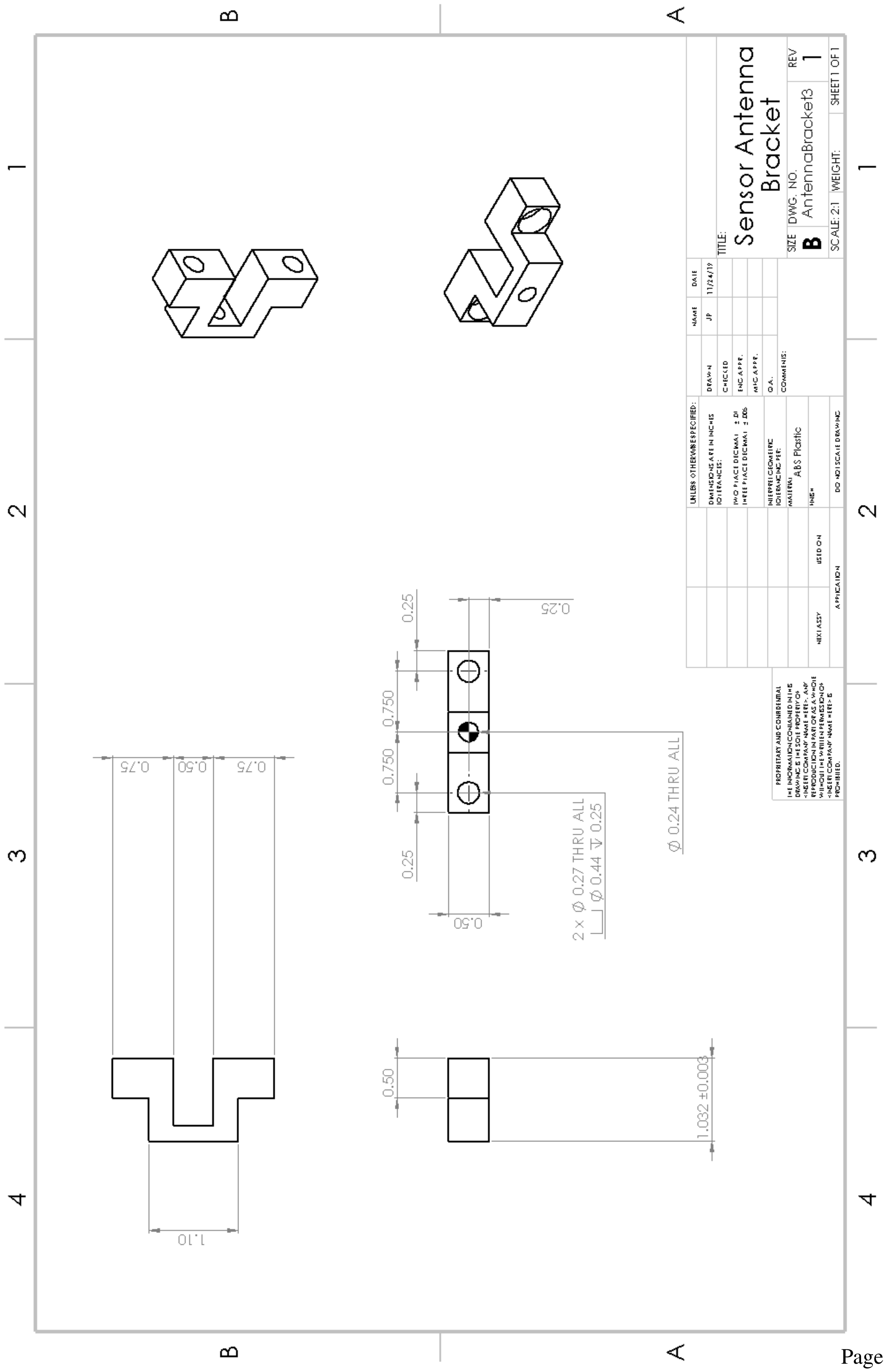


Figure 52: Antenna Bracket on Top of the Sensor

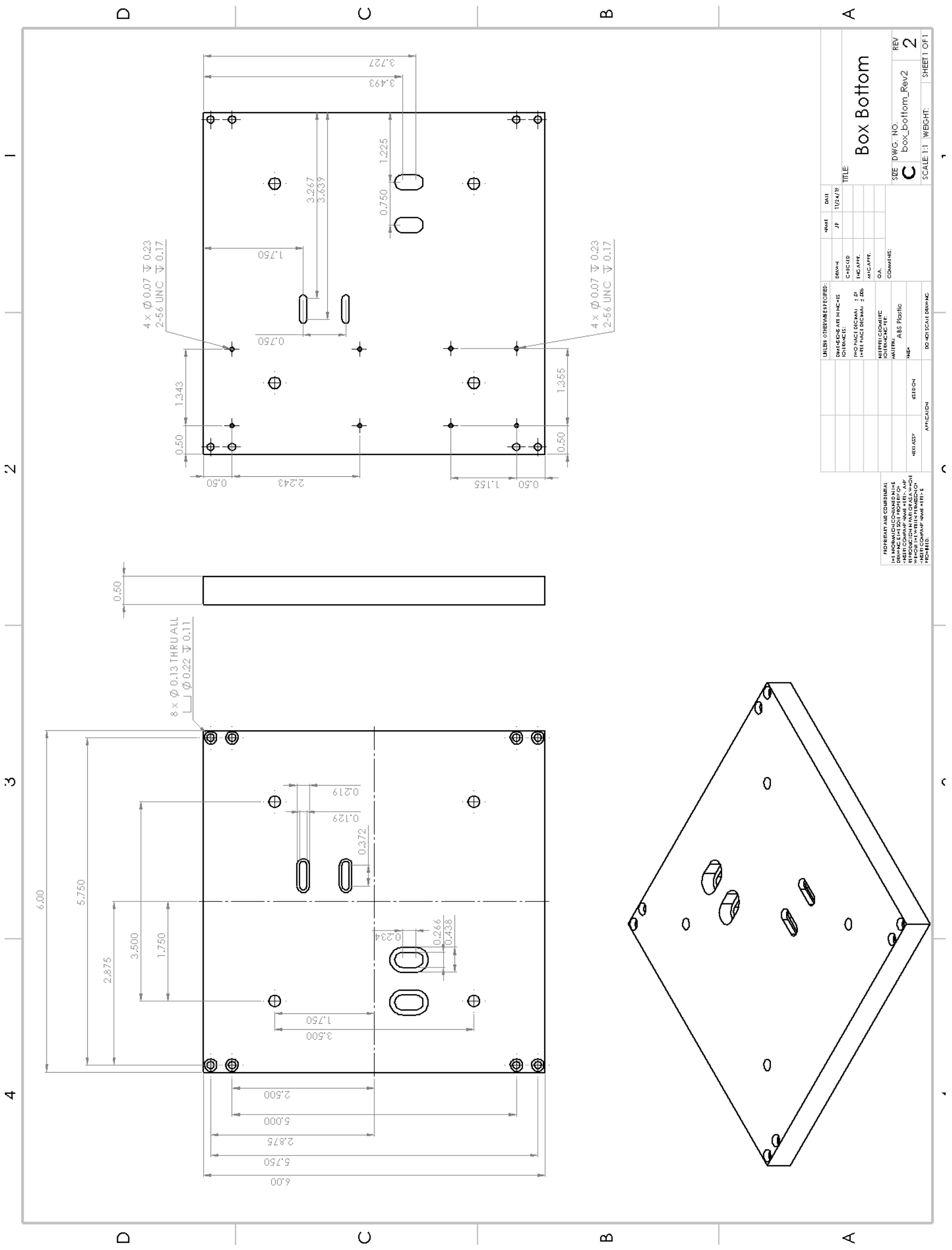


Figure 53: Box Bottom Piece

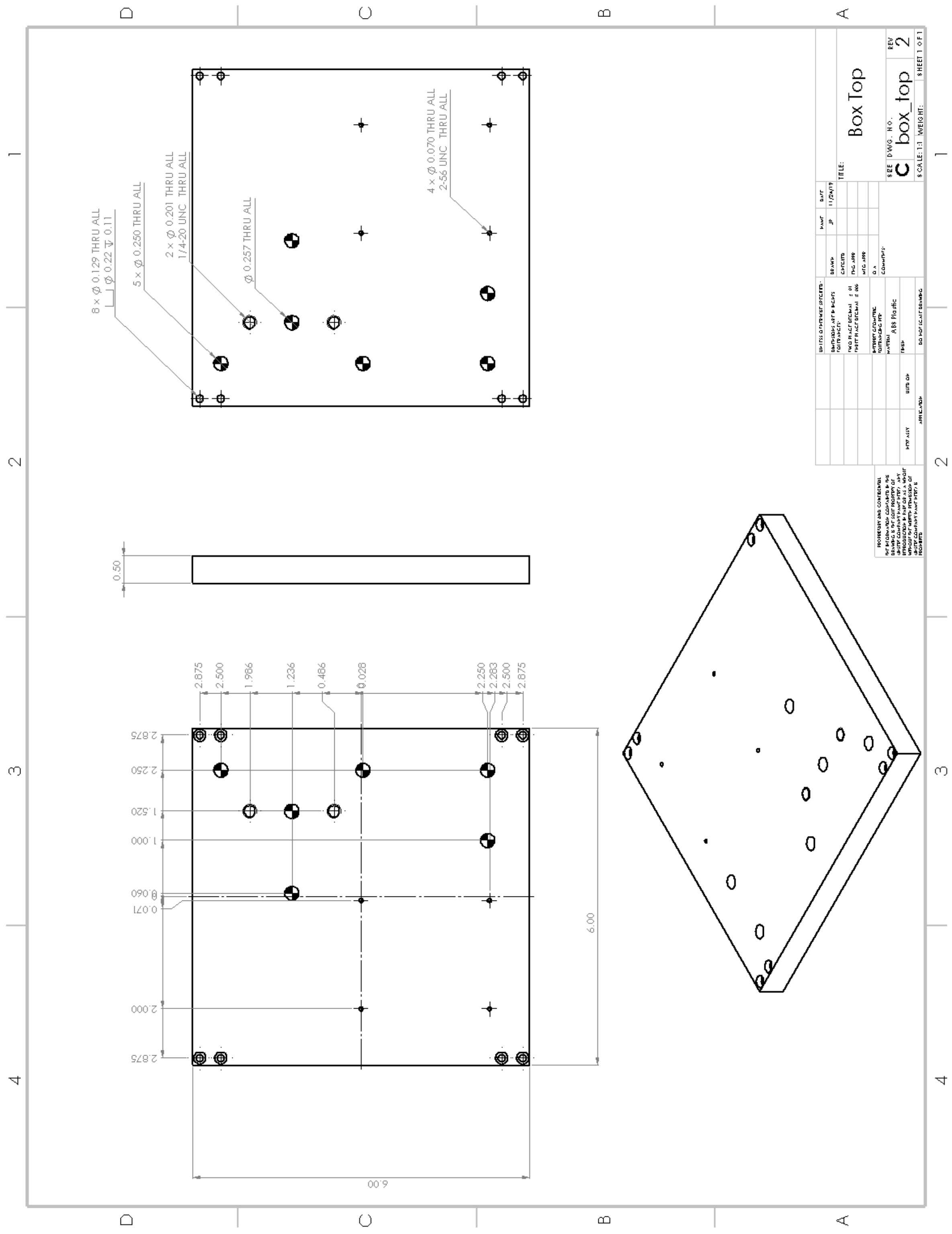
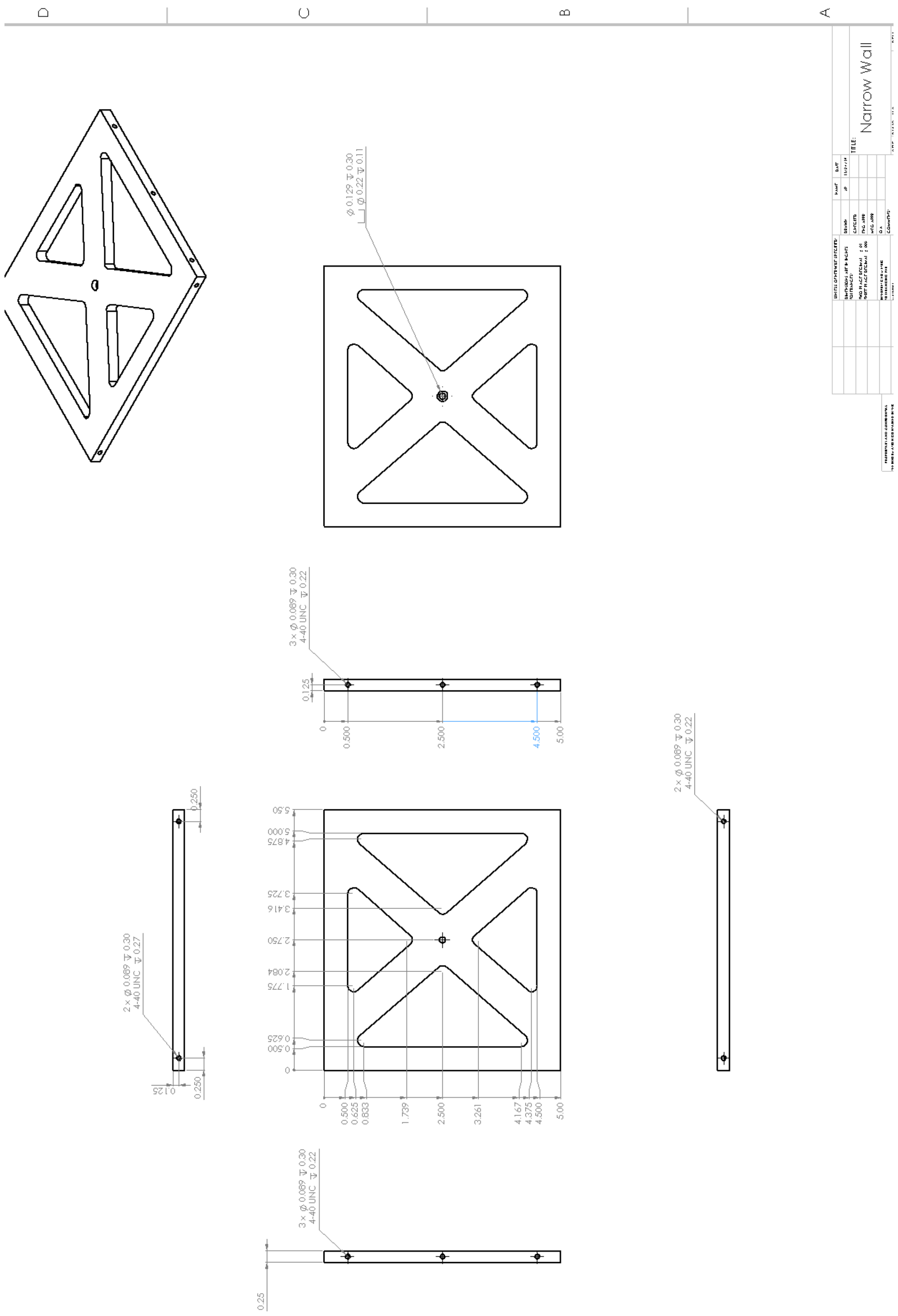

Box Top
C box_top
2
ABS Plastic
8 × ⌀ 0.129 THRU ALL ⌴ ⌀ 0.22 ↧ 0.11
5 × ⌀ 0.250 THRU ALL
2 × ⌀ 0.201 THRU ALL 1/4-20 UNC THRU ALL
⌀ 0.257 THRU ALL
4 × ⌀ 0.070 THRU ALL 2-56 UNC THRU ALL
0.50
6.00
6.00

Figure 54: Box Top Piece


Narrow Wall
TITLE:
D
C
B
A
⌀ 0.129 ↧ 0.30
⌀ 0.22 ↧ 0.11
3 × ⌀ 0.089 ↧ 0.30
4-40 UNC ↧ 0.22
2 × ⌀ 0.089 ↧ 0.30
4-40 UNC ↧ 0.22
2 × ⌀ 0.089 ↧ 0.30
4-40 UNC ↧ 0.27
0.125
0.250
0.25
0
0.500
0.625
0.833
1.739
2.500
3.261
4.167
4.375
4.500
5.00
5.50
5.000
4.875
3.725
3.416
2.750
2.084
1.775

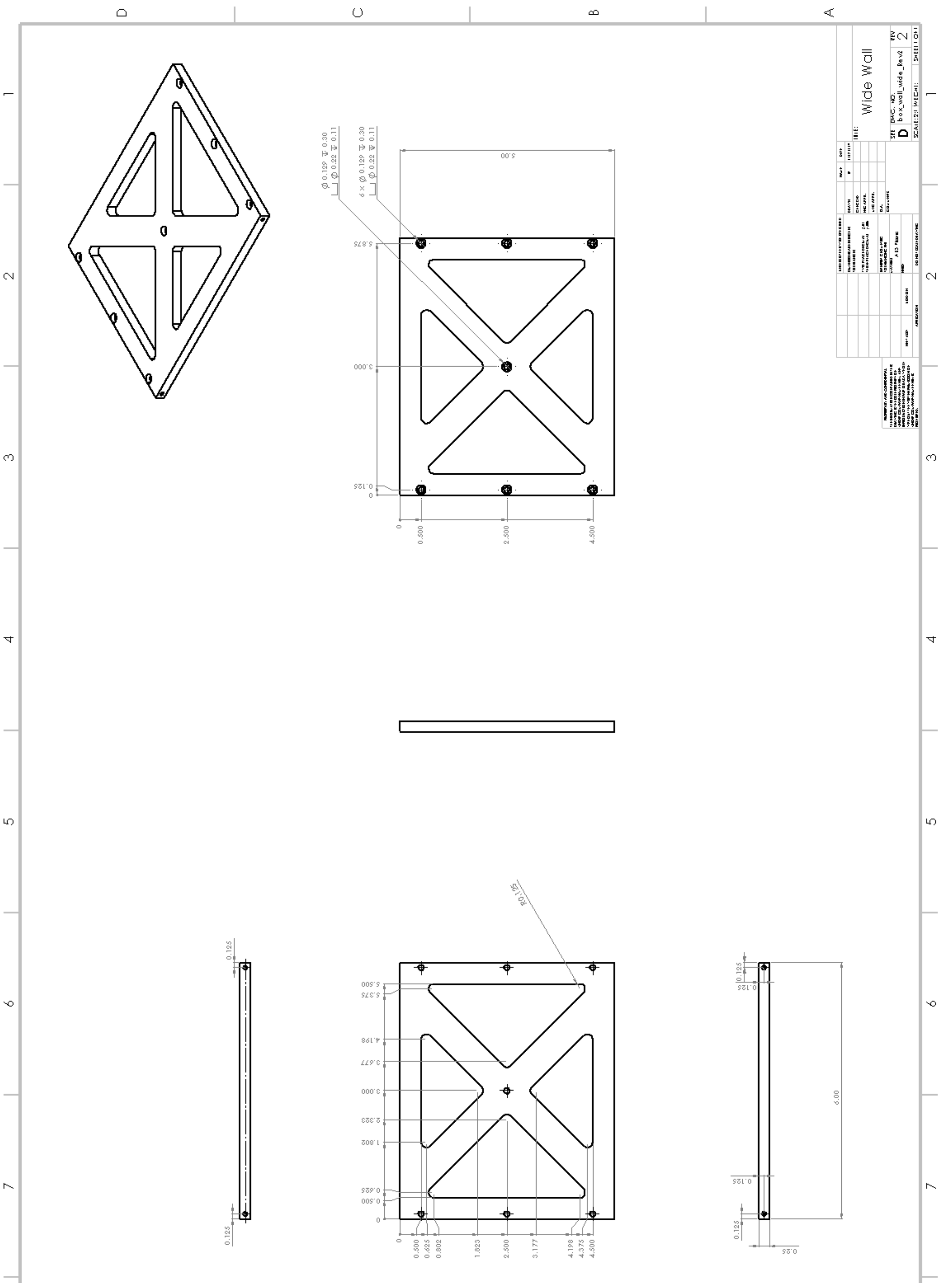
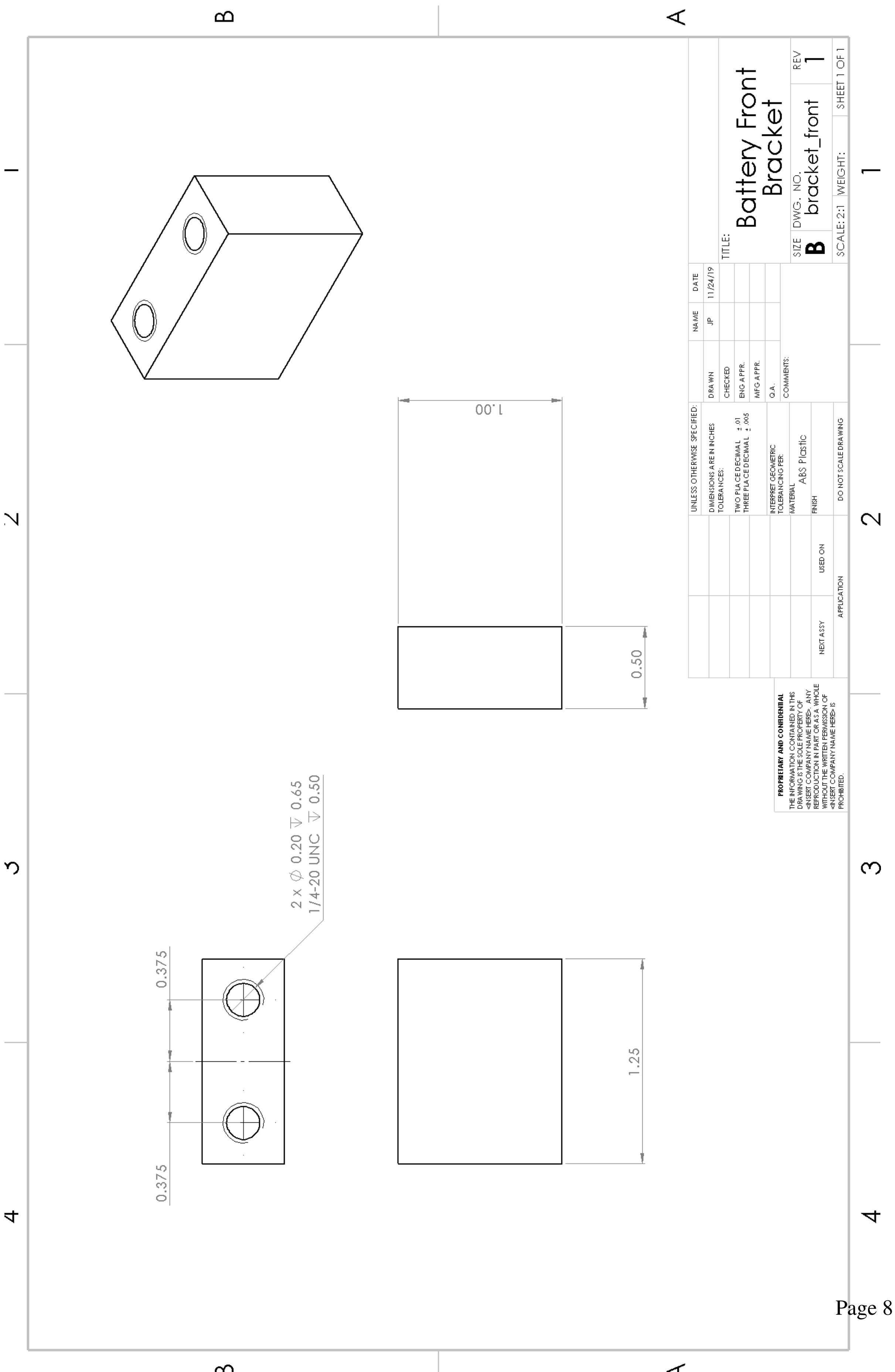


Figure 56: Wide Wall Piece of the Box


Battery Front Bracket
DWG. NO. bracket_front
SIZE B
REV 1
SCALE: 2:1
WEIGHT:
SHEET 1 OF 1
TITLE:
DRAWN JP 11/24/19
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL ABS Plastic
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
2 x ∅ 0.20 ↧ 0.65
1/4-20 UNC ↧ 0.50
0.375
0.375
1.25
0.50
1.00
A
B
1
2
3
4

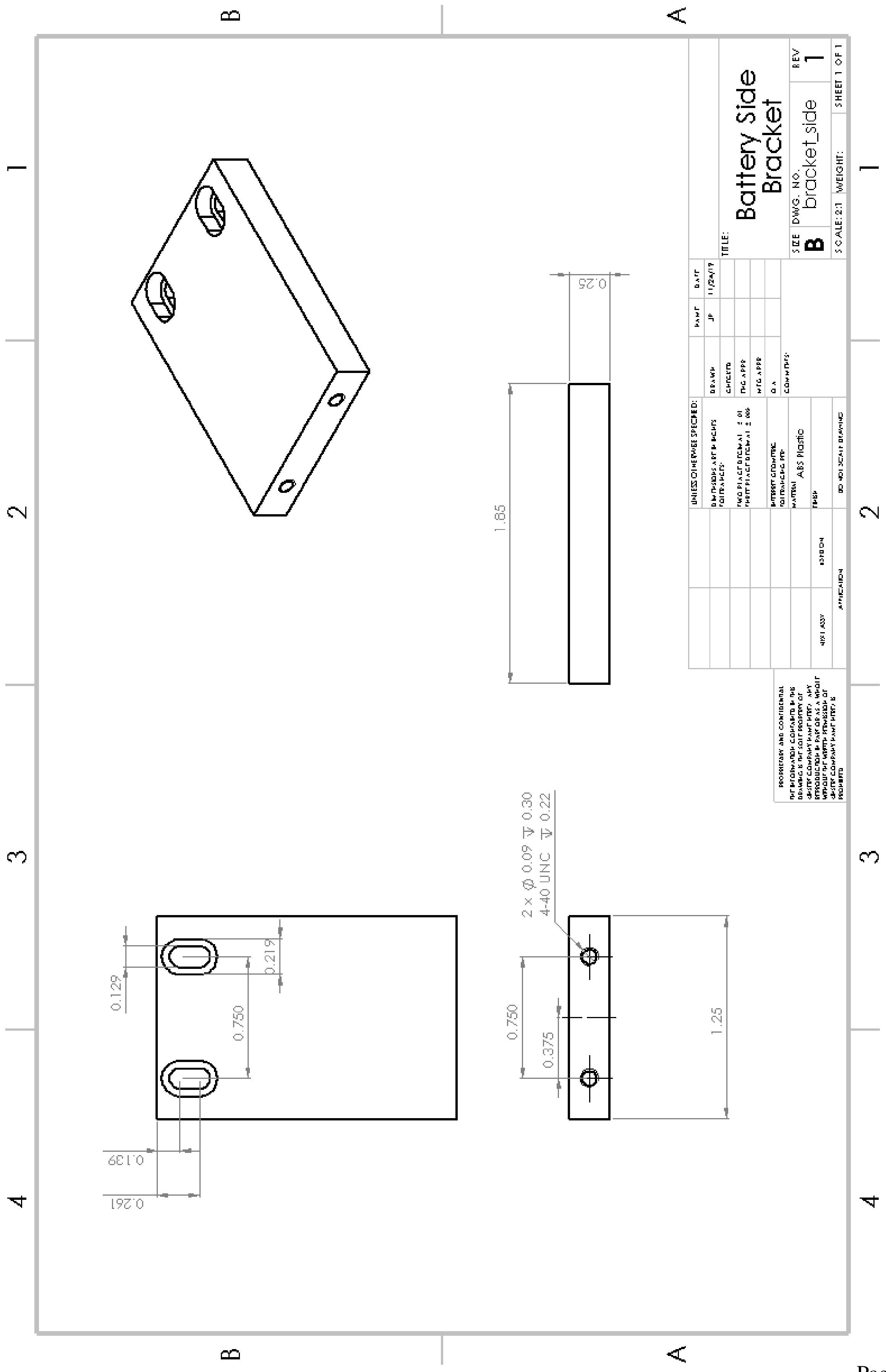
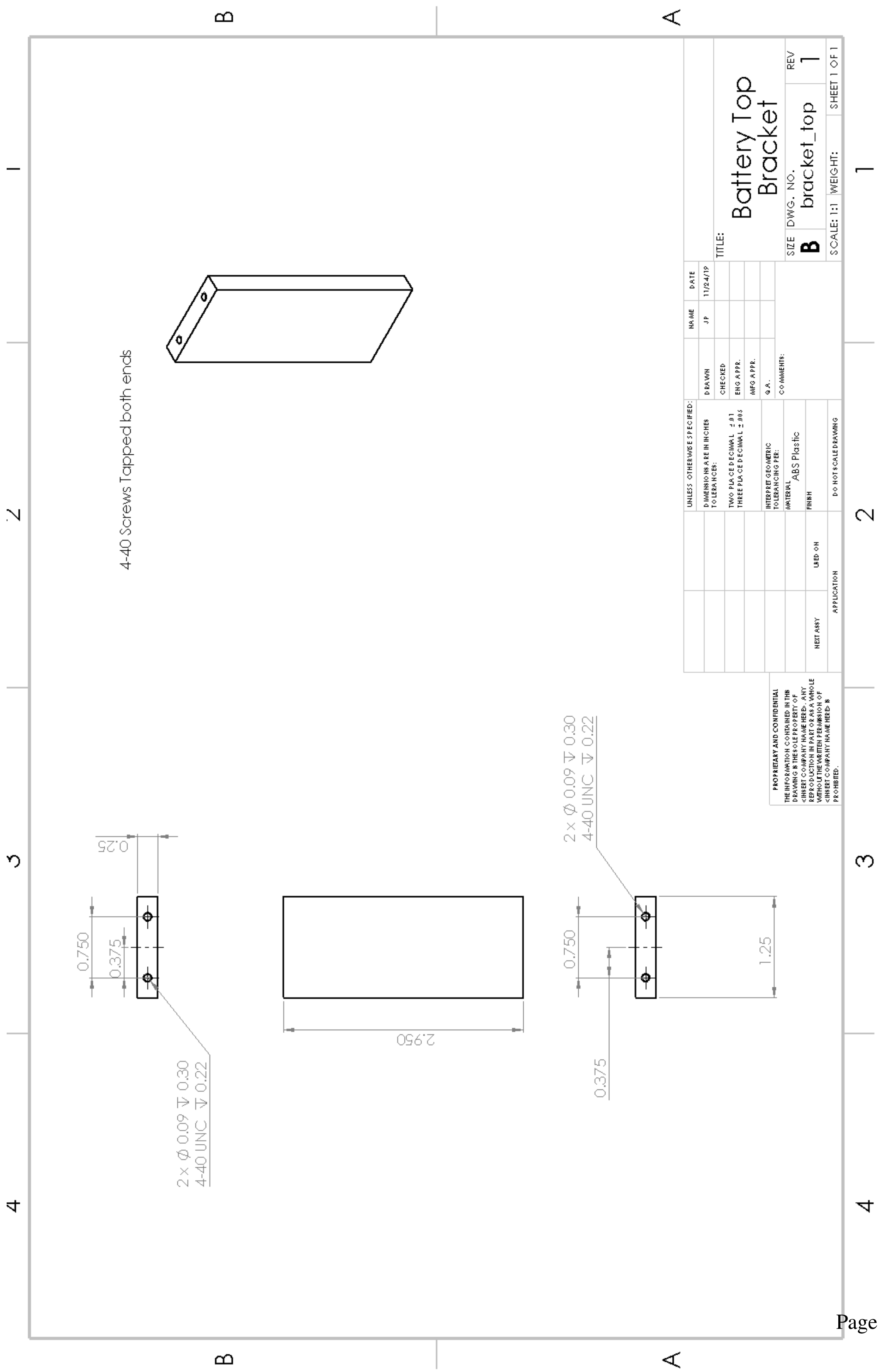
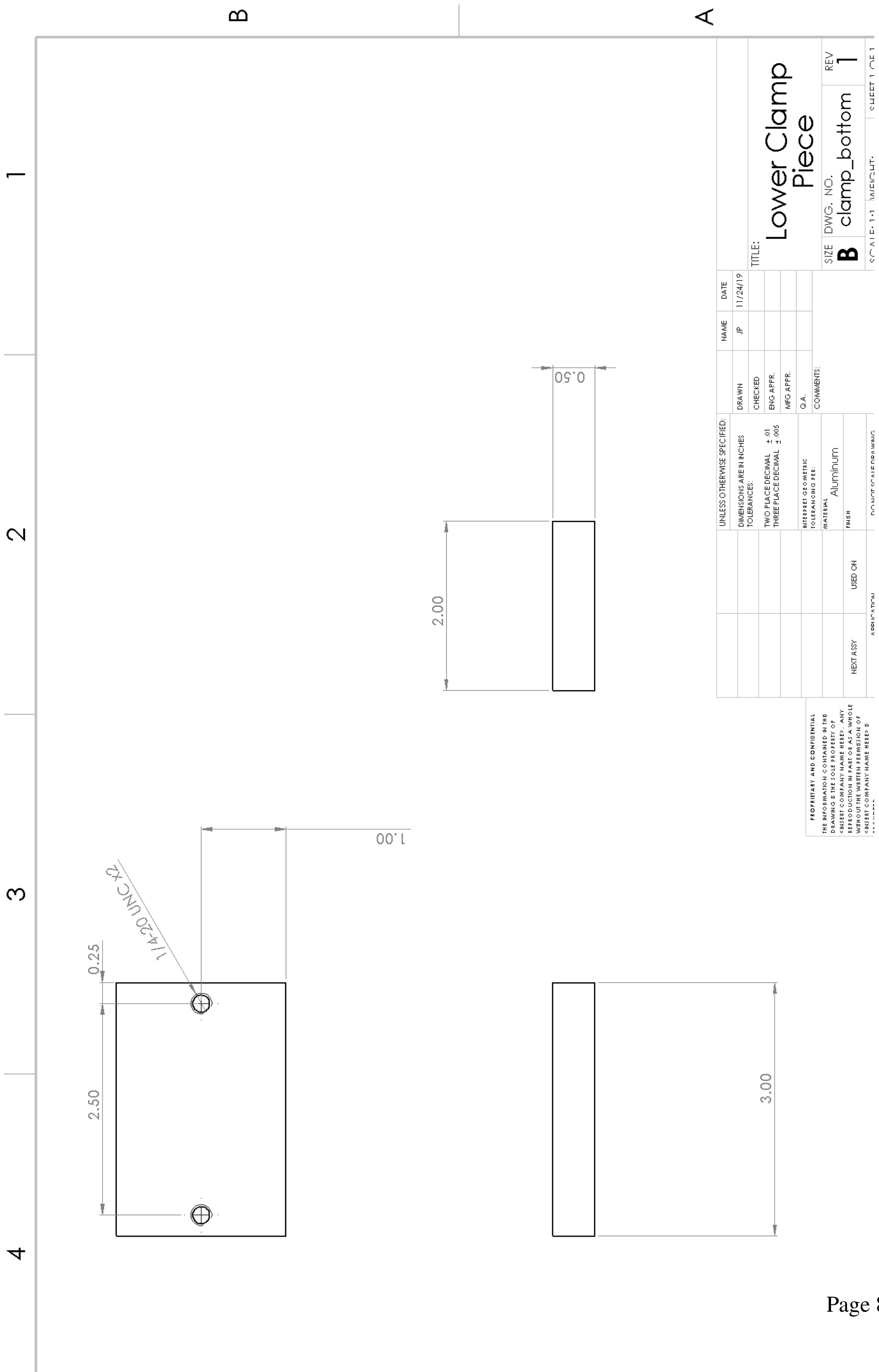


Figure 58: Side Mounting Bracket for the Battery


4-40 Screws Tapped both ends
0.25
0.750
0.375
2 × ⌀ 0.09 ↧ 0.30
4-40 UNC ↧ 0.22
2.950
0.750
0.375
1.25
2 × ⌀ 0.09 ↧ 0.30
4-40 UNC ↧ 0.22
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL ABS Plastic
FINISH
DO NOT SCALE DRAWING
NAME DATE
DRAWN JP 11/24/19
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
Battery Top Bracket
SIZE B
DWG. NO. bracket_top
REV 1
SCALE: 1:1 WEIGHT: SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
A
B
1
2
3
4


B
A
1
2
3
4
0.50
2.00
1.00
1/4-20 UNC x2
0.25
2.50
3.00
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
Aluminum
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
JP
11/24/19
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
Lower Clamp Piece
SIZE
B
DWG. NO.
clamp_bottom
REV
1
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

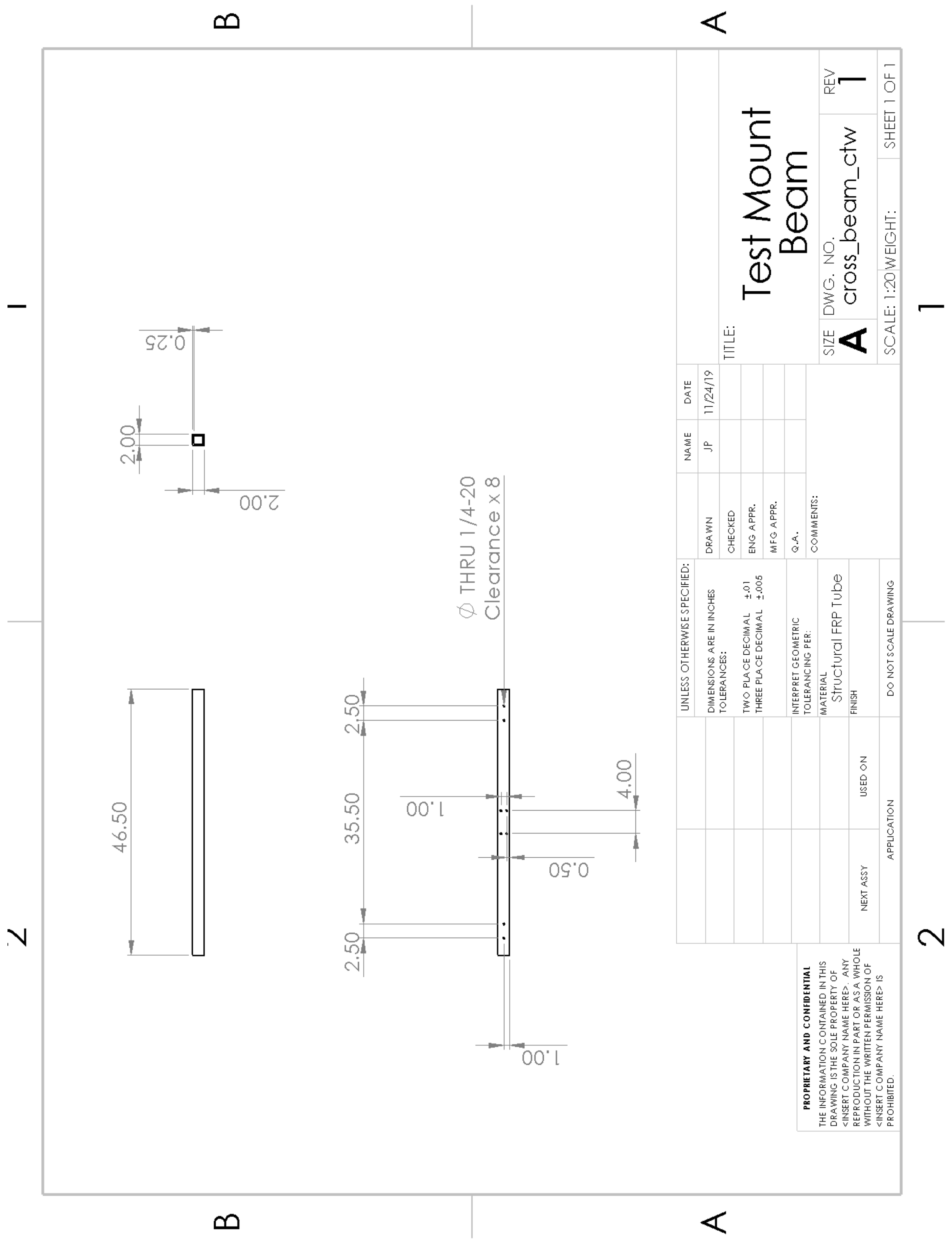

Test Mount Beam
DWG. NO. cross_beam_ctw
SIZE A
REV 1
SCALE: 1:20 WEIGHT:
SHEET 1 OF 1
TITLE:
NAME DATE
DRAWN JP 11/24/19
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
TWO PLACE DECIMAL ±.01
THREE PLACE DECIMAL ±.005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
Structural FRP Tube
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
∅ THRU 1/4-20 Clearance x 8
0.25
2.00
2.00
46.50
2.50
35.50
2.50
1.00
4.00
0.50
1.00
A
B
1
2

Figure 61: Test Mounting Cross Beam

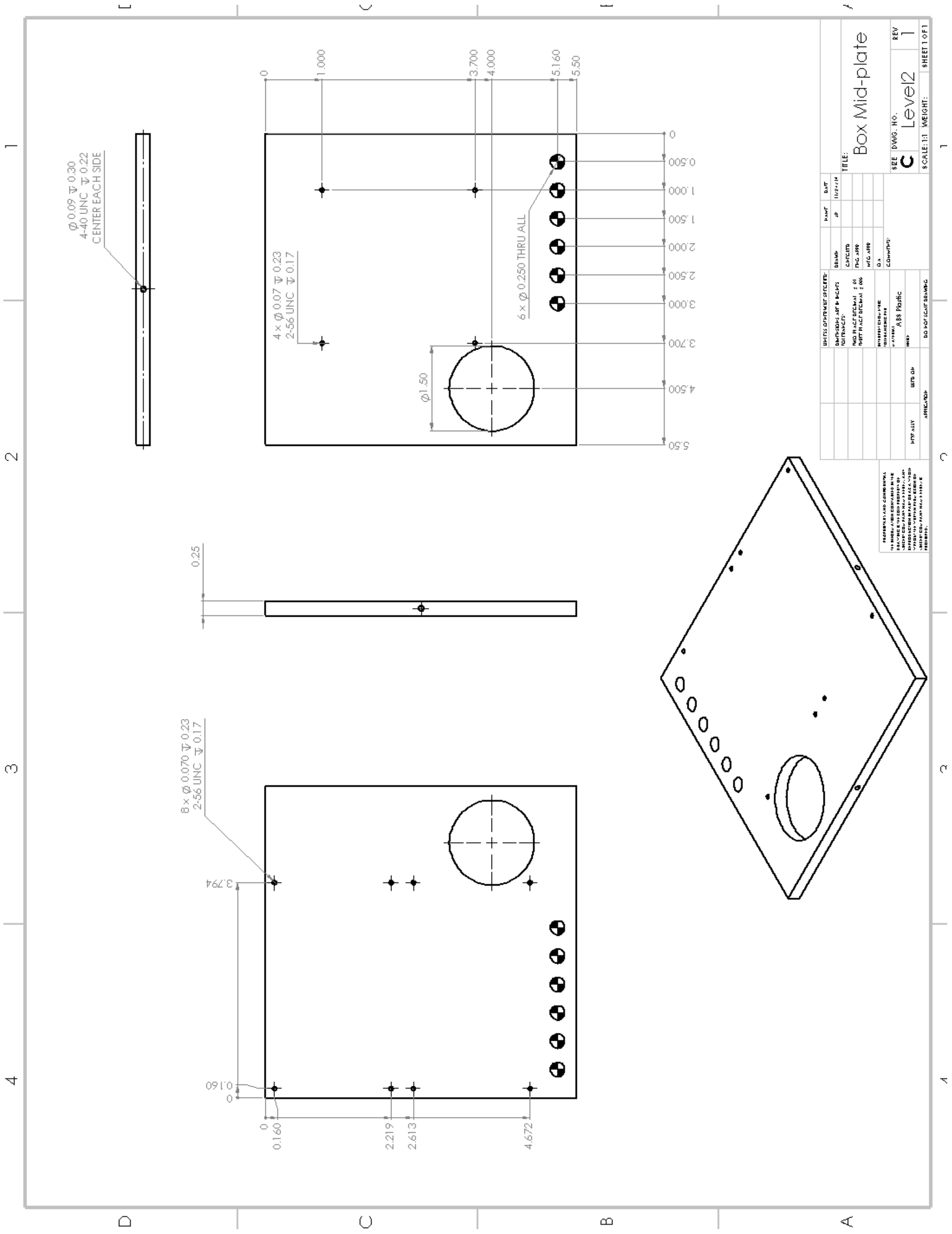


Figure 62: Mid-level Mounting Plate in the Sensor and Beacon

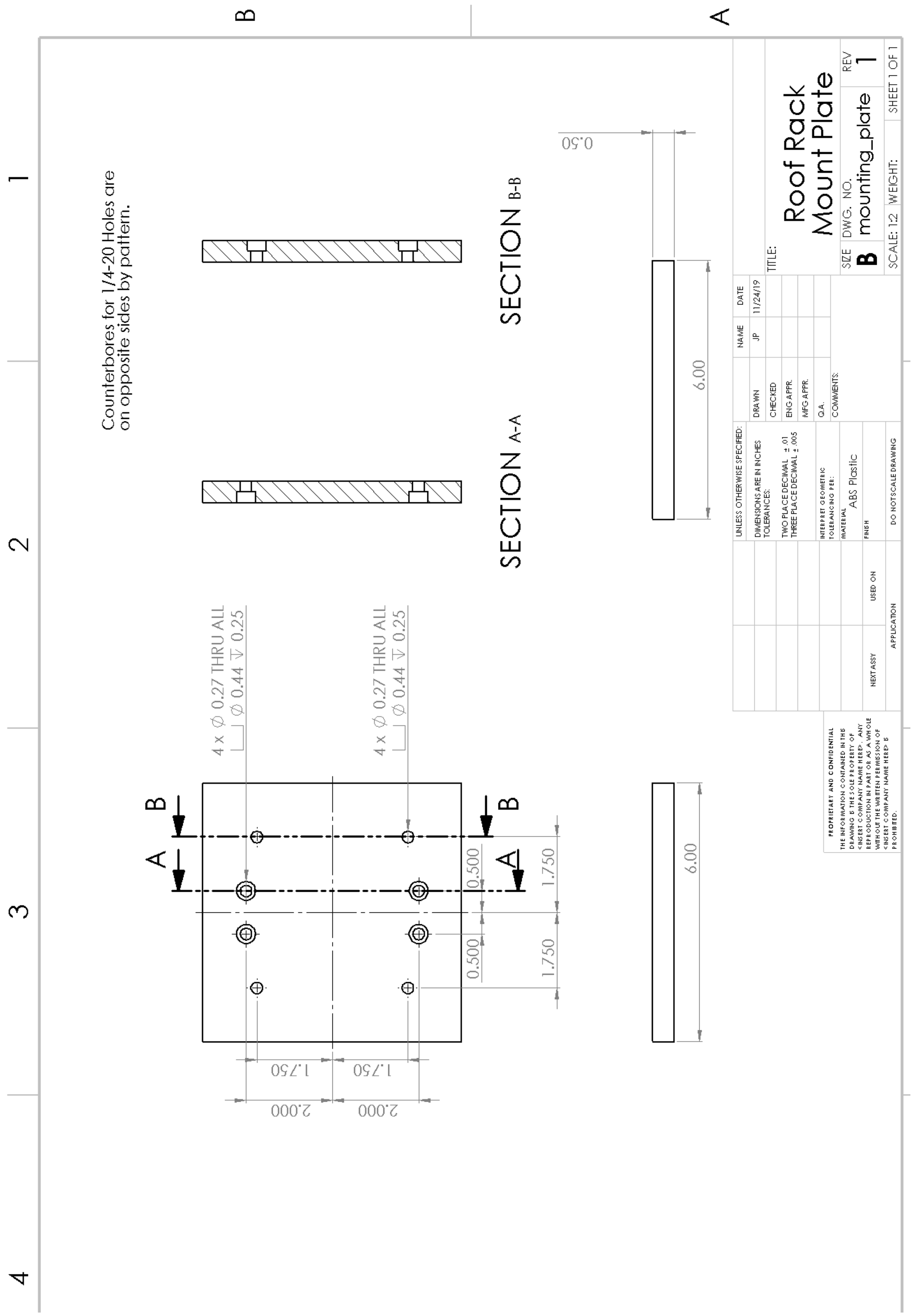


Figure 63: Test Mount Mounting Plate

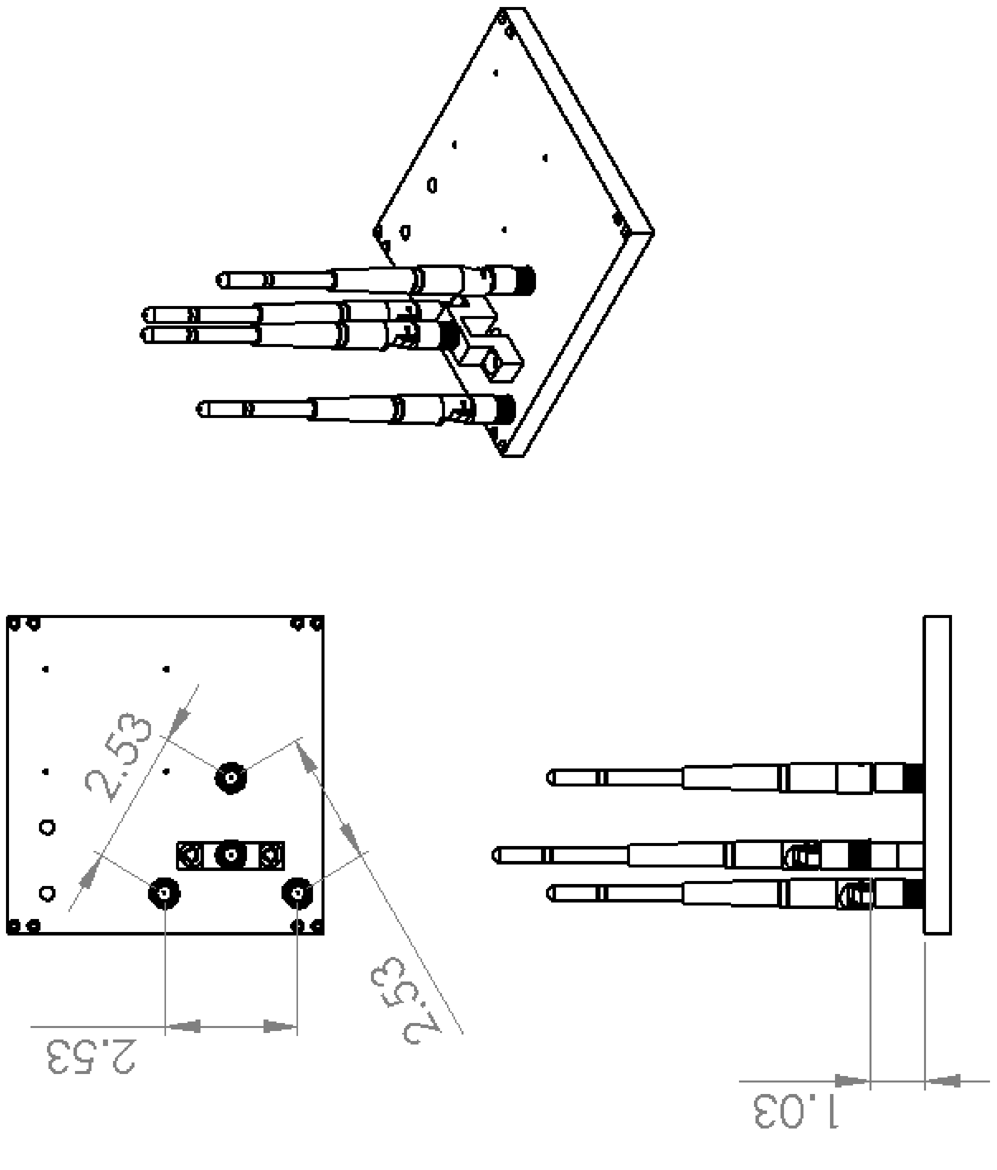


Figure 64: Sensor Antenna Spacing

11.4 Project Planning

Presented here is an expanded Work Breakdown Structure for the entire project, including both fall and spring semesters. This WBS follows the same format as the one presented in the body of this report.

WBS	Name	Duration	Start	Finish	Predecessors
1	*Project Definition*	20 days	8/26/19	9/15/19	
1.1	Define the Problem	7 days	8/26/19	9/2/19	
1.2	Develop Functional Requirements	3 days	9/2/19	9/5/19	
1.3	Document Development and Edits	10 days	9/5/19	9/15/19	2,3
1.4 *	PDD	0.25 day	9/15/19	9/15/19	4
2	*Conceptual Design Document*	14 days	9/16/19	10/1/19	1
2.1	Trade Studies	7 days	9/16/19	9/23/19	
2.2	Determine Baseline Design	2 days	9/24/19	9/26/19	7
2.3	Document Development and Edits	4 days	9/26/19	9/30/19	8
2.4 *	CDD	0.5 day	9/30/19	10/1/19	9
3	*Preliminary Design Review*	10 days	10/5/19	10/15/19	6
3.1	Define Critical Project Elements	1 day	10/5/19	10/6/19	
3.2	First-Level Feasibility Analysis of CPM	6 days	10/5/19	10/11/19	
3.3	Additional Feasibility Analysis	2 days	10/8/19	10/10/19	
3.4	Final Edits	4 days	10/11/19	10/15/19	12,13,14
3.5 *	PDR	0.25 day	10/15/19	10/15/19	15
4	PDR Edits	2 days	10/28/19	10/30/19	
5 *	Present at General Atomics	2 days	11/6/19	11/8/19	
6	*Critical Design Review*	44 days	10/28/19	12/11/19	
6.1.1	Finalize Hardware Selection	7 days	10/28/19	11/4/19	
6.1.2	Develop CAD Drawings	7 days	11/4/19	11/11/19	21
6.2.1	Develop Algorithms	9 days	10/28/19	11/6/19	
6.2.2	Apply Kalman Filter	9 days	11/1/19	11/10/19	
6.3.1	Stage 1 Testing	7 days	10/29/19	11/5/19	
6.3.2	Stage 2 Testing	7 days	11/5/19	11/12/19	27
6.3.3	Stage 3 Testing	5 days	11/12/19	11/17/19	28
6.3.4	Define Safety Guidelines	2 days	11/17/19	11/19/19	29
6.4	Presentation Development	12 days	11/7/19	11/19/19	30FF,23FF,26FF
6.5	Presentation Edits	7 days	11/19/19	11/26/19	31,35FF
6.6	Team Conference Call	0.5 day	11/24/19	11/24/19	
6.7	Practice Presentation	2 days	11/26/19	11/28/19	32
6.8	Make Spring Gantt Chart	2 days	11/22/19	11/24/19	
6.9 *	Critical Design Review	1 days	12/10/19	12/11/19	
7	*Final Report*	5 days	12/11/19	12/16/19	19

Continued on next page

Table 15 – *Continued from previous page*

WBS	**Name**	**Duration**	**Start**	**Finish**	**Predecessors**
8 *	Final Report Submission	1 day	12/16/19	12/17/19	37
9	Design PCB	21 days	1/6/20	1/27/20	
10	Make PCB	3 days	1/27/20	1/30/20	39FS+2days
11	Purchase Hardware	3 days	12/19/19	12/22/19	
12	Characterize Timing in SDR	10 days	3/06/20	3/16/20	41FS+21days
13	Integrate GPS with RTK	2 days	1/12/20	1/14/20	41FS+21days
14	Develop Heading Calc Code	2 days	1/13/20	1/15/20	
15	Develop Range Finding Code	2 days	1/15/20	1/17/20	
16 *	Finalize Position Finding Code	1 day	1/17/20	1/18/20	44,45
17	Integrate GPS and Data Logger	2 days	1/20/20	1/22/20	43
18	Finalize Preliminary Tests Procedures	5 days	1/6/20	1/11/20	
19	Make Safety Procedures for Testing	5 days	1/13/20	1/18/20	
20 *	CDR at General Atomics	1 day	1/17/20	1/18/20	
21 *	*Manufacturing Status Review*	21 days	1/13/20	2/3/20	
22	Construct Beacon and Sensor Structures	4 days	1/26/20	1/30/20	
23	Preliminary Test 1	2 days	1/27/20	1/29/20	48,42,47,49
24 *	Integrate Pi and SDR	28 days	2/2/20	3/01/20	53FS+4days,52
25	Preliminary Test 2	28 days	2/10/20	3/09/20	54FS+5days,46
26	Preliminary Test 3	4 days	3/10/20	3/14/20	55FS+5days
27	Finalize Field Tests Procedures	5 days	1/20/20	1/25/20	
28 *	*Test Readiness Review*	28 days	2/3/20	3/2/20	
29	Construct Testing Equipment	7 days	2/10/20	2/17/20	
30 *	*Spring Final Review*	49 days	3/2/20	4/20/20	
31 *	Phase One - Test	3 days	3/14/20	3/17/20	56FS+7days,57
32	Phase Two - Test	3 days	3/24/20	3/27/20	61FS+7days
33	Phase Three - Test	3 days	4/3/20	4/6/20	62FS+7days
34	Phase Four - Test	3 days	4/13/20	4/16/20	63FS+7days
35	GA Thermal Test	1 day	4/13/20	4/14/20	61
36 *	*Senior Design Symposium*	1 day	4/23/20	4/24/20	
37 *	*Project Final Report*	14 days	4/20/20	5/4/20	

Table 15: Work Breakdown Structure

11.5 Code

11.5.1 Kalman Filter Code

The code below takes position data for the beacon and runs it through a linear Kalman Filter in order to get a better estimate for the beacon's position.

```
#!/usr/bin/env python2
import rospy
import numpy as np
import sys,os, os.path
import math
import matplotlib.pyplot as plt
import yaml

def main():
        #dynamics = sys.argv[1] #Next semester AKA deployment
        #with open(dynamics, 'r') as fp:
        #       dynamics = yaml.load(fp)

        #x = dynamics['initial_state']
        #t = dynamics['timestep']

        a = np.loadtxt('total_motion.txt', delimiter=',') #x,y,z
        #t = np.loadtxt('timestamp.txt', delimiter=', ')
        t = np.linspace(0,8,115)
        w_noise = np.random.rand(1,6)*0.1
        m_noise = np.random.rand(1,6)*0.1
        ts = t[1]
        position = np.array(a[0,:]*ts*ts)
        velocity = np.array(a[0,:]*ts)
        #B = [0.1 , 0 , 0 , 0]
        x = [0,0,0,0,0,0]
        P = np.eye(6)*np.random.rand(1,1)*100 #Initial uncertainty
        A = np.array([[1, 0, 0, ts, 0, 0],
                [0, 1, 0, 0, ts, 0],
                [0, 0, 1, 0, 0, ts],
                [0, 0, 0, 1, 0, 0],
                [0, 0, 0, 0, 1, 0],
                [0, 0, 0, 0, 0, 1]])
        Q = np.eye(6)*0.75 #actuator type uncertainy 1%
```

```
H = np.eye(6)

R = np.array([[100, 0, 0, 0, 0, 0],
        [0, 100, 0, 0, 0, 0],
        [0, 0, 100, 0, 0, 0],
        [0, 0, 0, 400, 0, 0],
        [0, 0, 0, 0, 400, 0],
        [0, 0, 0, 0, 0, 400]])/30

pose, P = kf(x,t,P,A,Q,H,R,a,w_noise,m_noise)

stdx = np.sqrt(P[:,0,0])
stdy = np.sqrt(P[:,1,1])
stdz = np.sqrt(P[:,2,2])



fig, axs = plt.subplots(2, 2)
fig.suptitle("Kalman␣Filtered␣data␣for␣Beacon␣in␣cluster␣motion", fontsize=
#fig.legend('x', 'y', 'z')
#Truth data
y = np.zeros((len(t),1))
z = np.zeros((len(t),1))
for i in range(0,len(t)):
        y[i] = 10 +0.1*i
        z[i] = 10 -0.1*i
axs[0,0].plot(t,y, linewidth = 2,linestyle = '--', color='black')
axs[0,1].plot(t,z, linewidth = 2,linestyle = '--', color='black')
truth = axs[1,0].hlines(10,0,8, linewidth = 2,linestyle = '--', color='blac



filtered, = axs[0,0].plot(t,pose[:,0], color='blue')
#plt.plot(t[:,0],P[:,0,0])
axs[0,1].plot(t,pose[:,1], color='blue', label='std')
axs[1,0].plot(t,pose[:,2], color='blue')


raw, = axs[0,0].plot(t,a[:,0], color='green')
axs[0,1].plot(t,a[:,1], color='green')
axs[1,0].plot(t,a[:,2], color='green')
```

```
        axs[1,1].plot(t,P[:,0,0], color='green')
        axs[1,1].plot(t,P[:,1,1], color='blue')
        axs[1,1].plot(t,P[:,2,2], color='red')

        #2 sigma bounds
        sigma = axs[0,0].fill_between(t,stdx + pose[:,0],pose[:,0] - stdx, color='#
        axs[0,1].fill_between(t,stdy + pose[:,1],pose[:,1] - stdx, color='#CED8FF')
        axs[1,0].fill_between(t,stdz + pose[:,2],pose[:,2] - stdx, color='#CED8FF')

        labels = ["Truth", "Raw", "Filtered", "sigma"]
        fig.legend([truth, raw, filtered, sigma], labels, loc = (0.9, 0.5))

        axs[0, 0].set_title('X',fontsize=24)
        axs[0, 1].set_title('Y',fontsize=24)
        axs[1, 0].set_title('Z',fontsize=24)
        axs[1, 1].set_title('Covariance', fontsize=24)

        axs[0,0].set_xlabel('time␣[s]', fontsize=20)
        axs[0,0].set_ylabel('distance␣[m]', fontsize=20)
        axs[0,1].set_xlabel('time␣[s]', fontsize=20)
        axs[0,1].set_ylabel('distance␣[m]', fontsize=20)
        axs[1,0].set_xlabel('time␣[s]', fontsize=20)
        axs[1,0].set_ylabel('distance␣[m]', fontsize=20)
        axs[1,1].set_xlabel('time␣[s]', fontsize=20)


        axs[1,1].legend(('x', 'y', 'z'))
        axs[0,0].grid()
        axs[0,1].grid()
        axs[1,0].grid()
        axs[1,1].grid()

        plt.show()



def kf(xp,t,Pp,A,Q,H,R,a, w,m):
        dt = t[1] # discrete
        x = np.zeros((len(t),6))
        x[0,:] = xp
```

```
    P = np.zeros((len(t),6,6))
    P[0] = Pp

    for k in range(0,len(t)-1):
        #Predict
        xkp = np.dot(A,(x[k]))

        pkp = np.dot(A,np.dot(P[k],np.transpose(A))) + Q

        #Correct
        K = np.dot(np.dot(pkp, np.transpose(H)), np.linalg.inv(np.dot(np.d


        '''try:
            Vx = (a[k+1][0] - a[k][0])/dt
            Vy = (a[k+1][1] - a[k][1])/dt
            Vz = (a[k+1][2] - a[k][2])/dt
        except:
            Vx = 0
            Vy = 0
            Vz = 0
        z = [a[k+1,0],a[k+1,1],a[k+1,2],Vx, Vy, Vz]'''

        z = [a[k+1,0],a[k+1,1],a[k+1,2],0,0,0]


        x[k+1] = xkp + np.dot(K,(z-np.dot(H,xkp)))

        P[k+1] = np.dot((np.eye(6) - np.dot(K,H)),pkp)

    return x, P


if __name__ == '__main__':
    try:
        main()
    except rospy.ROSInterruptException:
        pass

clear;close;clc;

n = 115;
```

```
noise = 6;
for i = 1:n
    x(i) =  10+rand*noise -noise/2;
    y(i) =  10+rand*noise -noise/2;
    z(i) =  10+rand*noise -noise/2;
end

mat = [x' y' z'];

dlmwrite('stationary.txt',mat,',')


for i = 1:n
    x(i) =  10+rand*noise -noise/2 +0.1*i;
    y(i) =  10+rand*noise -noise/2;
    z(i) =  10+rand*noise -noise/2;

end

mat = [x' y' z'];

dlmwrite('motion.txt',mat,',')

for i = 1:n
    x(i) =  10+rand*noise -noise/2 +0.1*i;
    y(i) =  10+rand*noise -noise/2 -0.1*i;
    z(i) =  10+rand*noise -noise/2;
end

mat = [x' y' z'];

dlmwrite('total_motion.txt',mat,',')
```

11.5.2 Road Test Arduino Code

The code included in this section was designed to help understand the fundamental vibrational excitations which the testing apparatus would be exposed to during systems testing. See Section 7 for info regarding systems testing. This analysis is done by acquiring z-axis acceleration data while driving at testing speeds on a road which is representative of actual testing conditions, then Fourier transforming this data to see what the prominent spectral peaks are in the data, if any.

```
/* #######################################################
```

```
 *
 *   Title: IMU Road is Bumpy Boi Test
 *
 *   Author: Adam Farmer
 *   Date Created: 11/17/19
 *   Date Modified: 11/18/19
 *
 *     READ ME
 *
 *     For serial port access on Adam's HP Spectre:
 *     sudo chmod a+rw /dev/ttyACM0
 *
 *     Program prints to serial the following packet:
 *     time [ms], grav_x [m/s^2], grav_y [m/s^2], grav_z [m/s^2], acc_x [m/s^2], acc
 *
 * ######################################################
 */


// Included libraries:
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <Adafruit_MPL3115A2.h>
#include <utility/imumaths.h>

Adafruit_BNO055 bno = Adafruit_BNO055(55);
Adafruit_MPL3115A2 baro = Adafruit_MPL3115A2();

// Define globals, if necessary:


void setup() {

  // Initialize serial rate:
  Serial.begin(9600);

  // Check to see if IMU is alive:
  if(!bno.begin())
  {
    // There was a problem detecting the BNO055 ... check your connections
    Serial.print("Ooops, no BNO055 detected ... Check wiring and restart!");
    while(1);
  }

  // Check to see if barometer is alive:
  if (! baro.begin()) {
    Serial.println("Couldn't find sensor. Check wiring and restart!");
    while(1);
  }

  delay(1000);
  bno.setExtCrystalUse(true);
  delay(1000);

  // Call to funciton which calibrates the IMU
  CalibrateIMU();
}

/* Program main loop */
void loop() {

  /* Get a new sensor event */
  sensors_event_t event;
  bno.getEvent(&event);
  imu::Quaternion quat = bno.getQuat();
  imu::Vector<3> gravityVec = bno.getVector(Adafruit_BNO055::VECTOR_GRAVITY);
  imu::Vector<3> accVec = bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);

  // Print time:
  Serial.print(millis()); Serial.print(",");

  // Print gravity vector:
  Serial.print(-gravityVec.x()); Serial.print(",");
  Serial.print(-gravityVec.y()); Serial.print(",");
  Serial.print(-gravityVec.z()); Serial.print(",");

  // Print acceleration vector
  Serial.print(-accVec.x()); Serial.print(",");
  Serial.print(-accVec.y()); Serial.print(",");
  Serial.println(-accVec.z());
  // Rough loop rate control:
  delay(10); // [ms]

}

/* Function defnintions used in this program */
void CalibrateIMU(void)
{
  // Initiation of local variables:
  uint8_t system, gyro, accel, mag, gyroLast, accelLast, magLast;
  system = gyro = accel = mag = gyroLast = accelLast = magLast = 0;
  bool calibrated = false;

  Serial.print("Please start calibration \n");

  // Loop until fully calibrated:
  while(calibrated == false)
  {
    // Does black magic to get calibration:
    bno.getCalibration(&system, &gyro, &accel, &mag);

    /* The data should be ignored until the system calibration is > 0 */
    if (system != 0) // If the system values are non-zero then print their values
    {
      // Display the individual values if they changed since last itteration:
      if ((gyro != gyroLast) || (accel != accelLast) || (mag != magLast))
      {
        Serial.print("Sys:");
        Serial.print(system, DEC);

        Serial.print(" Gyro:");
        Serial.print(gyro, DEC);

        Serial.print(" Accelerometer:");
        Serial.print(accel, DEC);

        Serial.print(" Magnetometer:");
        Serial.println(mag, DEC);

        Serial.print("\n");
      }
    }

    // Condition for full calibration:
    if(gyro == 3 && accel == 3 && mag == 3)
    {
      calibrated = true;
    }
    // Set current values to last values to compare in next itteration:
    else
    {
      gyroLast = gyro;
      accelLast = accel;
      magLast = mag;
    }
  }
}

float rad2deg(float angRad)
{
  float angDeg = angRad*180/PI;
  return angDeg;
}

##########################################################################################
#
# Title:
#
# Purpose:
#
# Author: Adam Farmer
#
# Date Written:  11/17/19
# Date Modified: 11/18/19
#
##########################################################################################

## Import libraries.

import math as m
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import mpl_toolkits
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.axes as ax
import csv
from scipy.fftpack import fft

## Make file names to import

# Test cases:
calibrationFileName = 'roadTest_0_0Hz_test2.csv'
testFileName = 'roadTest_0_5Hz_test2.csv'


def readCsvFile(filename):

    # Read in IMU data from .csv file:
    try:
        with open(filename, "r") as csvfile:
            csvread = csv.reader(csvfile, delimiter=',')
            Data = list(csvread)

    except FileNotFoundError:
        print('File␣not␣found')

    # turn the list into float
    imuData = np.zeros((len(Data[:]),7))

    i=0
    j=0
    for row in Data:
        for itemInRow in row:
            imuData[i,j] = float(itemInRow)
            j=j+1
        j=0
        i=i+1

    return imuData


def integrator(yVec, xVec):
    sumVec = np.zeros(len(xVec))

    for i in range(0, len(xVec)-1):

        sumVec[i+1] = yVec[i]*(xVec[i+1]-xVec[i]) + 0.5*(yVec[i+1]-yVec[i])*(xVec[i

    return sumVec

imuCalData = readCsvFile(calibrationFileName)
imuData = readCsvFile(testFileName)

## Do some necessary math

# Define time vector and uncalibrated acceleration data:
tVec = (imuData[:,0] - imuData[0,0])/1000
rddot = (imuData[:,4:] - imuData[:,1:4])

# Testing
# tVec = np.linspace(0, 30, 1000)
# rddot = np.zeros((len(tVec),3))
# omega = 2*np.pi/10
# rddot[:,0] = 0.1
# rddot[:,1] = -0.2
# rddot[:,2] = np.cos(omega*tVec) + 1

# Prealocation:
delta_rddot = np.zeros((len(rddot[:,0]),3))
rdot = np.zeros((len(rddot[:,0]),3))
r = np.zeros((len(rdot[:,0]),3))

# Get constant offset in acceleration from calibration data:
for i in range(0, 3):
    #delta_rddot[:,i] = np.mean(imuCalData[:,4+i] - imuCalData[:,1+i])
    delta_rddot[:, i] = np.mean(rddot[:,i])

# Remove the constant offset bias:
rddot = rddot - delta_rddot

rddot_sp = np.real(np.fft.rfft(rddot[:,2]))
# rddot_f = np.real(np.fft.rfftfreq(tVec.shape[-1],30/1000))
rddot_f = np.real(np.fft.rfftfreq(tVec.shape[-1],tVec[-1]/1000))
```

```
# for i in range(0, 3):
#     rdot[:,i] = integrator(rddot[:,i],tVec)
#     r[:,i] = integrator(rdot[:,i],tVec)

for i, ti in enumerate(tVec[:-1]):

    rdot[i,0] = np.trapz(rddot[:i,0],tVec[:i])
    rdot[i,1] = np.trapz(rddot[:i,1],tVec[:i])
    rdot[i,2] = np.trapz(rddot[:i,2],tVec[:i])

    r[i,0] = np.trapz(rdot[:i,0],tVec[:i])
    r[i,1] = np.trapz(rdot[:i,1],tVec[:i])
    r[i,2] = np.trapz(rdot[:i,2],tVec[:i])

## Plotting acceleration and position data

plt.figure(1)
plt.minorticks_on()
# Acceleration in time:
plt.subplot(3, 1, 1)
plt.plot(tVec, rddot)
#plt.plot(tVec, delta_rddot)
plt.title('Acc,␣Vel␣and␣Pos␣Data␣in␣time')
plt.legend(['x-axis','y-axis','z-axis','x␣error', 'y␣error', 'z␣error'])
plt.ylabel('Acceleration␣[m/s^2]')
plt.grid(which='major', color='k', linestyle='-', linewidth=.1)
plt.grid(which='minor', color='k', linestyle=':', linewidth=.1)
# Velocity in time:
plt.subplot(3, 1, 2)
plt.plot(tVec, rdot)
plt.legend(['x-axis','y-axis','z-axis'])
plt.ylabel('Velocity␣[m/s]')
plt.grid(which='major', color='k', linestyle='-', linewidth=.1)
plt.grid(which='minor', color='k', linestyle=':', linewidth=.1)
# Position in time:
plt.subplot(3, 1, 3)
plt.plot(tVec, r)
plt.legend(['x-axis','y-axis','z-axis'])
plt.xlabel('Time␣[seconds]')
plt.ylabel('Position␣[m]')
plt.grid(which='major', color='k', linestyle='-', linewidth=.1)
plt.grid(which='minor', color='k', linestyle=':', linewidth=.1)

## FFT plotting:
plt.figure(2)
plt.minorticks_on()
plt.plot(rddot_f, rddot_sp)
plt.title('Amplitude␣VS␣Frequency')
plt.xlabel('Frequency␣(Hz)')
plt.ylabel('Rotational␣Energy␣(Joules)')
plt.grid(which='major', color='k', linestyle='-', linewidth=.1)
plt.grid(which='minor', color='k', linestyle=':', linewidth=.1)

plt.show()
```

11.5.3 Link Budget Code

The following code calculates the link margin for an RF transmitter/receiver using inputs such as frequency, bit energy to noise ratio, design margin, data rate, noise figure, receiver gain, and transmitter gain. This code is used to determine an estimate for the link margin which proved feasibility for the scope of the project.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RF Antenna Link Budget Calculator
%
% This calculates the Link Margin for an RF Transmitter/Reciever
% Inputs are: Frequency, Bit energy to noise rati
% Design Margin, Data Rate, Noise Figure, Receiver Gain
% Transmitter Gain
%
% Author: Camilla Hallin
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Housekeeping
 clear all
 close all
 clc

%%  Setup
D=[100,200,300,400]/1000;
 for i=1:numel(D)
```

```
c=3e8;
f=2.5;
lamb= c/(f*1e9);
Boltzman=1.380e-23;
BER=10e-7;
EbNo=10; % Bit energy to noise ratio
Z=50e6; % Data Rate
PrNo=EbNo+10*log10(Z); % Carrier to Noise Ratio Density
DM=10; % Design Margin

MinPrNo=PrNo+DM;

RxTemp=8; % Kelvin for a 60 degree path
RxLosses=-.5; % Typical Value
RxNF=2.5; % Can vary, this is based on LIME
RxF=290*10^(RxNF/10);
RxTs=RxTemp+((290*(1-0.89)/0.89))+((290*(RxF-1)/0.89));
RxNoisePower=10*log10(RxTs*Boltzman); %dBW-Hz

Gr=20; % Not sure what antenna gain to use

RxFOM=Gr/RxTs;


% Propagation Loss
Dist=D(i); %km
Ls=10*log10((lamb/(4*pi()*Dist*10^3))^2);
La=-5; % Atmospheric absorption
Lp=-.2; % Polarization Loss typical value


Gt=20; % Not sure what antenna gain to use
TxLosses=-.5;
TxWatts=linspace(.00001,.005,500);
TxPower=10*log10(TxWatts);
EIRP=TxPower+Gt;


RxPower=TxPower+Gt+Gr+Ls+La+Lp+RxLosses;
PrNoActual=RxPower-RxNoisePower;
```

```
TxNeed=MinPrNo-(Gt+Gr+Ls+La+Lp+RxLosses)+RxNoisePower;

LinkMargin=PrNoActual-MinPrNo;

%% Plot The Results
figure(1), plot(TxWatts*1000,LinkMargin,'LineWidth',1.1)
hold on
xlabel('Transmit Power (mW)')
ylabel('Link Margin (dB)')
title('Link Margin vs Transmit Power at 100 m','FontSize',14)
ltext{i}=strcat('D = ',num2str(round(1000*D(i))),' m');
end
plot(TxWatts*1000,zeros(1,numel(LinkMargin)),'g--','LineWidth',1.4)
legend(ltext,'Location','EastOutside')


%% Timing Accuracy Analysis
c=3e8;
dist=100; % Start with 100m
data_rate=32e6; % 32 MHz processor (INPUT)
t_trav=dist/0.5/c; % "truth travel time"
t_start=3/data_rate; % Asuumed time to send start bits
t_buff=4/data_rate; % Assumed buffer time

packetSize=8; % 16 bit packet to decode
t_DM= packetSize*1/data_rate; % Assumed time to Demodulate/ recieve packet
ac=linspace(0,5,50); % Accuracy of clock (in fractions of a clock cycle)

% for j=1:50
% for i=1:1000
% t_startR=t_start+normrnd(0,1)*;
% t_DMR=t_DM+normrnd(0,1)*.13e-12;
% t_buffR=t_buff+normrnd(0,1)*.13e-12;
%
% T_sum(i)=t_trav+2*t_startR+2*t_DMR+t_buffR;
% DistCal(i)=.5*c*(T_sum(i)-2*t_start-2*t_DMR-t_buff);
%
% end
% var(j)=std(DistCal);
% end
%
```

```
% figure(5), hist(DistCal)
%
% figure(6)
% plot(ac./data_rate,var)
% title('Distance Accuracy vs Timing Accuracies')
% xlabel('Timing Accuracy')
% ylabel('1 \sigma error on Distance (m)')
% hold on


%% Averaging Benefit

%figure
TErr=linspace(1e-12,2/data_rate,200);
for j=1:10000
for i=1:numel(TErr)
t_startR=t_start+normrnd(0,1)*TErr(i);
t_DMR=t_DM+normrnd(0,1)*0*TErr(i);
t_buffR=t_buff+normrnd(0,1)*0*TErr(i);

T_sum2(i)=t_trav+2*t_startR+2*t_DMR+t_buffR;
DistCal2(i,j)=.5*c*(T_sum2(i)-2*t_start-2*t_DMR-t_buff);

end
if mod(j,10)==0
    avdata10(:,j/10)=mean(DistCal2(:,j/10:j/10+9),2);
end
if mod(j,20)==0
    avdata20(:,j/20)=mean(DistCal2(:,j/20:j/20+19),2);
end
if mod(j,30)==0
    avdata30(:,j/30)=mean(DistCal2(:,j/30:j/30+29),2);
end
if mod(j,100)==0
    avdata100(:,j/100)=mean(DistCal2(:,j/100:j/100+99),2);
end



% scatter(TErr,abs(DistCal2(:,j)),'b.')
%hold on
```

```
end
% hold on,scatter(1:numel(saveDist),saveDist),ylim([99.5, 100.5])
%% Standard Deviations

figure
plot(TErr,abs(mean(100-DistCal2,2))+abs(std(100-DistCal2,1,2)),'LineWidth',1.1)
hold on
plot(TErr,abs(mean(100-avdata10,2))+abs(std(100-avdata10,1,2)),'LineWidth',1.1)
plot(TErr,abs(mean(100-avdata20,2))+abs(std(100-avdata20,1,2)),'LineWidth',1.1)
plot(TErr,abs(mean(100-avdata30,2))+abs(std(100-avdata30,1,2)),'LineWidth',1.1)
plot(TErr,abs(mean(100-avdata100,2))+abs(std(100-avdata100,1,2)),'LineWidth',1.1)
plot(TErr,2*ones(1,numel(TErr)),'LineWidth',1.2)
title('Expected Distance Error vs Timing Error')
xlabel('Timing Error (s)')
ylabel('Average \mu Distance Error + 1 \sigma (m)')
legend('No Averaging','Averaging 10','Averaging 20','Averaging 30','Averaging 100')
ylim([0,10])

%% No averaging Histogram
index1=find(TErr>19e-9,1);
figure, hist(DistCal2(index1,1:1000))
title('Distance Estimate, Timing error of 20 ns')
xlabel('Distance Estimate (m)')
ylabel('Frequency')
mean(DistCal2(index1,1:1000))
std(DistCal2(index1,1:1000))



%% Histogram For Averaging 10 with timing error ~20 ns
index10=find(TErr>19e-9,1);
figure, hist(avdata10(index10,:))
title('Distance with 10 Averaged, Timing error of 20 ns')
xlabel('Distance Estimate (m)')
ylabel('Frequency')
mean(avdata10(index10,:))

%%
index30=find(TErr>31.3e-9,1);
figure, hist(avdata30(index30,1:333))
title('Distance with 30 Averaged, Timing error of 33 ns')
```

```
xlabel('Distance Estimate (m)')
ylabel('Frequency')
mean(avdata30(index30,1:333))
std(avdata30(index30,1:333))
```

11.5.4 Position Code

The position code calculates the position of the beacon according to a set range and specified antenna distances. Incorporated in the code is a Monte Carlo simulation that proved feasibility for this part of the project.

```
%% Senior Projects Position Modeling Take 3
% Created: Brendan Lutes 10/8/19 based upon posModel by  Emily Webb anad
% Anastasia Muzinski
% Modified: Brendan Lutes 10/9/19
% A model for finding position of beacon in 3 directions
% based on posModel and information from Ambiguity Resolution in
% Interferometry (5 Antenna Direction finding)
%% Housekeeping!
clear all; close all; clc;

%% Initialize Position of Antennas and Beacon
A1 = [0 -0.5 0]; %[m]
A2 = [0 0.5 0]; %[m]
A3 = [-0.5 0 0]; %[m]
A4 = [0.5 0 0]; %[m]
A5 = [0 0 0]; %[m]



% Spherical Coordinates Whoo!
alpha = 70; % Azimuth angle, degrees
beta = 45; % Elvation angle, degrees
r = 200;    % Distance between receiver origin and transmitter origin (m)

% Center of Beacon
Bee = [r*sind(beta)*cosd(alpha), r*sind(beta)*sind(alpha), r*cosd(beta)];

%% Location of 3 transmitters with respect to beacon origin
% Add for attitude finding, TODO
% T1 = [0 0 0]+ Bee;
% T2 = [0 0 0.05] + Bee;
% T3 = [0 0.05 0] + Bee;
% Transforming Beacon to different attitude
% Creating transformation matrix

% Creating quaternion from transformation matrix


%% Running and plotting test case

%% Function to simulate converting recieved data to beacon location
err= test3D5A(A1, A2, A3, A4, A5, Bee, 1)


function [err] = test3D5A(A1, A2, A3, A4, A5, Bee, variation)
% Inputs 3 reciever  and 1 beacon locations in cartesian coordinates, as
% well as variation term (for playing around with error, multiply by
% variable inside function to apply an error)

    % Separation distances between receiving antennas
    s12 = norm(abs(A1 - A2));
    s13 = norm(abs(A1 - A3));
    s14 = norm(abs(A1 - A4));
    s15 = norm(abs(A1 - A5));
    s23 = norm(abs(A2 - A3));
    s24 = norm(abs(A2 - A4));
    s25 = norm(abs(A2 - A5));
    s34 = norm(abs(A3 - A4));
    s35 = norm(abs(A3 - A5));
    s45 = norm(abs(A4 - A5));


    % Quick Plot for sanity
    figure(1);
    plot3(A1(1),A1(2), A1(3),'*','MarkerSize',8);
    hold on
    plot3(A2(1),A2(2),A2(3),'*','MarkerSize',8);
    plot3(A3(1),A3(2), A3(3),'*','MarkerSize',8);
    plot3(Bee(1),Bee(2), Bee(3),'*','MarkerSize',8);
    grid on
    grid minor
    % Sanity checked!
    % Find the ranges. (Assumed to be known)
    r1 = norm(abs(Bee-A1));
    r2 = norm(abs(Bee-A2));
    r3 = norm(abs(Bee-A3));
    r4 = norm(abs(Bee-A4));
    r5 = norm(abs(Bee-A5));


    %% Initialize other Parameters
    f = 10e6; % Hz, frequency of sine wave
    c = 299792458; % m/s, the speed of light
    lambda = c/f; % m, wavelength
    dspec = lambda/(2*pi); % "specific distance"

    %% Actual Calculations
    r12 = abs(r2-r1); % distance difference (m)
    r34 = abs(r4-r3); %


    dphi12 = r12/dspec; % calculate phase difference between antennas 1 and 2
    dphi34 = r34/dspec;

    % These are the calculations we will be doing to find

    theta = asind((dphi12*lambda/(2*pi*s12))^2+(dphi34*lambda/(2*pi*s34))^2);
    psi = atand(dphi12*s34/(dphi34*s12));




    %% Actual Position, using A1 as reference antenna
    Wasp = [r1*sind(theta)*cosd(psi), r1*sind(theta)*sind(psi), r1*cosd(theta)];

    % New plotting sanity check
    figure(2);
    set(gcf,'Color','White');
    plot3(Bee(1),Bee(2),Bee(3),'*','MarkerSize',8);
```

```
    hold on
    plot3(Wasp(1),Wasp(2),Wasp(3),'*','MarkerSize',8);
    legend('Actual Beacon Position','Predicted Beacon Position');
    grid on
    grid minor
    %xlim([Bee(1)-1 Bee(1)+1]); ylim([Bee(2)-1 Bee(2)+1]);zlim([Bee(3)-1 Bee(3)+1]
    % title('Actual and Predicted Beacon Position');

    % Absolute error
    err = abs(Bee - Wasp);
    % Relative error
    %err = abs(Bee - Wasp)./abs(Bee);
    fprintf('The X direction error is: %.3f.\n',err(1));
    fprintf('The Y direction error is: %.3f.\n',err(2));
    fprintf('The Z direction error is: %.3f.\n',err(3));

    figure(1)
     plot3(Wasp(1),Wasp(2),Wasp(3),'*','MarkerSize',8);
    legend('Receiver 1', 'Receiver 2', 'Receiver 3','Actual Beacon Position','Pred
end

%% Senior Projects Position Modeling Take 2
% Created: Emily Webb National Garlic Lovers Day
% Modified: Anastasia Muszynski 10/7/19

%% Housekeeping!
clear all; close all; clc;

%% Initialize Position of Antennas and Beacon
A1 = [0 0];
A3 = [0 0.5];
Bee = [100*cosd(35) 100*sind(35)];

%% Testing Different Antenna Separations (holding A1 constant, moving A2), no rando

N =50;    % Number of samples
err1 = zeros(N,2);  % Preallocate space fo vector of error values
thetaerr1 = zeros(N,1);  % Preallocate space fo vector of error values
% Currently changing x location of A2
A2vec(:,2) = linspace(0.05, 0.5, N);

% Iterating through A2 values
```

```
for i = 1:N
    [err1(i,:), thetaerr1(i,:)] = test2D(A1,A2vec(i,:), Bee, 0);
end
% New plotting sanity check
    figure
    set(gcf,'Color','White');
    hold on
    plot(A2vec(:,2), abs(err1(:,1)) ,'b.','MarkerSize',8);
    plot(A2vec(:,2), abs(err1(:,2)) ,'r.','MarkerSize',8)
    legend('Error in X Position','Error in Y Position');
    xlabel('Separation Distance of Reciever A2 from A1 (m)')
    ylabel('Absolute Position Error (m)')
    grid on
    title('Error in Beacon Position Based on Separation');

%% Testing Different Antenna Separations, Monte Carlo Simulation

N =5000;    % Number of samples
err1 = zeros(N,2);  % Preallocate space fo vector of error values
thetaerr1 = zeros(N,1);  % Preallocate space fo vector of error values
A2 = [0 0.06];

for i = 1:N
    [err1(i,:), thetaerr1(i,:)] = test2D(A1, A2, Bee,1);
end
% Histograms of x and y error
   figure
   sgtitle({'Beacon Position Error for Normally Distributed Range Errors of 1mm',..
       ' for Antenna Separation of 6 cm, Range of 100m, Heading Angle of 35 degrees
   subplot(1,2,1)
   grid on
   histogram(err1(:,1))
   xlabel('X-Position Error (m)')
   subplot(1,2,2)
   grid on
   histogram(err1(:,2))
   xlabel('Y-Position Error (m)')

   figure
   histogram(thetaerr1(:,1))
   grid on
```

```
    xlabel('Beacon Heading Angle Error (degrees)')
    title({'Beacon Position Error for Normally Distributed Range Errors of 1mm',...
        ' for Antenna Separation of 6 cm, Range of 100m, Heading Angle of 35 degrees

% Reset A2 location
A2 = [0 0.2];
%% Testing different ranges
N = 500;
rvec = linspace(10,1000,N);
err2 = zeros(N,2);  % Preallocate space fo vector of error values
thetaerr2 = zeros(N,1);  % Preallocate space fo vector of error values

for i = 1:N
    Bee = [rvec(i)*cosd(35) rvec(i)*sind(35)];
    [err2(i,:), thetaerr2(i,:)] = test2D(A1, A2, Bee, 0);
end

% New plotting sanity check
    figure
    set(gcf,'Color','White');
    hold on
        plot(rvec, abs(err2(:,1)) ,'b.');
        plot(rvec, abs(err2(:,2)) ,'r.');
    legend('Error in X Position','Error in Y Position');
    xlabel('Distance of Reciever from Beacon (m)')
    ylabel('Absolute Position Error (m)')
    grid on
    title('Error in Beacon Position Based on Beacon/ Transmitter Separation');
%% Testing Different Ranges, Monte Carlo Simulation

N =5000;    % Number of samples
err2 = zeros(N,2);  % Preallocate space fo vector of error values
thetaerr2 = zeros(N,1);  % Preallocate space fo vector of error values

r1 = 100;
Bee = [r1*cosd(35) r1*sind(35)];

for i = 1:N
    [err2(i,:), thetaerr2(i,:)] = test2D(A1, A2, Bee,1);
end
% Histograms of x and y error
```

```
    figure
    sgtitle({'Beacon Position Error for Normally Distributed Ranging Errors of 1mm'
        ' for Antenna Separation of 20 cm, Range of 100m, Heading Angle of 35 degree
    subplot(1,2,1)
    histogram(err2(:,1))
    grid on
    xlabel('X-Position Error (m)')
    subplot(1,2,2)
    histogram(err2(:,2))
    grid on
    xlabel('X-Position Error (m)')

    figure
    histogram(thetaerr2(:,1))
    grid on
    xlabel('Beacon Heading Angle Error (degrees)')
    title({'Beacon Position Error for Normally Distributed Range Errors of 1mm',...
        ' for Antenna Separation of 20 cm, Range of 100m, Heading Angle of 35 degree
      fprintf('Mean: %d   Standard deviation: %d', mean(err3), std(err3))

    %% Testing different theta values
N = 50;
r = 100;
thetavec = linspace(0, 88,N);
err3 = zeros(N,2);  % Preallocate space fo vector of error values
thetaerr3 = zeros(N,1);  % Preallocate space fo vector of error values

for i = 1:N
    Bee = [r*cosd(thetavec(i)) r*sind(thetavec(i))];
    [err3(i,:), thetaerr3(i,:)] = test2D(A1, A2, Bee, 0);
end
% New plotting sanity check
     figure
     set(gcf,'Color','White');
     hold on
     plot(thetavec, err3(:,1) ,'b.');
     plot(thetavec, err3(:,2) ,'r.');
      %ylim([-.1, .2])
     legend('Error in X Position','Error in Y Position');
     xlabel('Heading Angle (from X axis) (degrees)')
     ylabel('Absolute Position Error (m)')
    grid on
    title('Error in Beacon Position Based on Theta Angle');

%% Testing Different Ranges, Monte Carlo Simulation

N =5000;    % Number of samples
err3 = zeros(N,2);  % Preallocate space fo vector of error values
thetaerr3 = zeros(N,1);  % Preallocate space fo vector of error values
theta = 10;
Bee = [r1*cosd(theta) r1*sind(theta)];

for i = 1:N
    [err3(i,:), thetaerr3(i,:)] = test2D(A1, A2, Bee,1);
end
% Histograms of x and y error
   figure
    sgtitle({'Beacon Position Error for Normally Distributed Range Errors of 1mm',
        ' for Antenna Separation of 20m, Range of 100m, Heading Angle of 10 degrees'
   subplot(1,2,1)
   histogram(err3(:,1))
   grid on
   xlabel('Beacon x-location error')
   subplot(1,2,2)
   histogram(err3(:,2))
   grid on
   xlabel('Beacon y-location error')

   figure
   histogram(thetaerr3(:,1))
   grid on
   xlabel('Beacon Heading Angle Error (degrees)')
   title({'Beacon Position Error for Normally Distributed Range Errors of 1mm',...
        ' for Antenna Separation of 6 cm, Range of 100m, Heading Angle of 10 degrees
   fprintf('Mean: %d   Standard deviation: %d', mean(err3), std(err3))



function [err, thetaErr] = test2D(A1, A2, Bee, useError)
    % Separation distances between receiving antennas
    s12 = norm(abs(A1 - A2));
```

```
% Applying Systematic Error to r1 and r2 measurements
r1 = norm(abs(Bee-A1));
r2 = norm(abs(Bee-A2));

% Applying Systematic Error to r1 and r2 measurements, if useError flag
% set to 1. Otherwise, assume no normal error

if useError == 1
    r1 = r1+normrnd(0,2)*0.001; % Normally distributed error (0.001 = 1mm erro
    r2 = r2+normrnd(0,2)*0.001; % Normally distributed error
end

%% Initialize other Parameters
f = 2.4e9; % Hz, frequency of sine wave
c = 299792458; % m/s, the speed of light
lambda = c/f; % m, wavelength
dspec = lambda/(2*pi); % "specific distance"

%% Actual Calculations
r12 = abs(r1-r2); % distance difference (m)

dphi12 = r12/dspec; % calculate phase difference

theta12 = asind(lambda*dphi12/(2*pi*s12)); % calculate heading angle

%% Actual Position, using A1 as reference antenna
Wasp = real([r1*cosd(theta12) r1*sind(theta12)]);

% New plotting sanity check
% figure(2);
% set(gcf,'Color','White');
% plot(Bee(1),Bee(2),'*','MarkerSize',8);
% hold on
% plot(Wasp(1),Wasp(2),'*','MarkerSize',8);
% legend('Actual Beacon Position','Predicted Beacon Position');
% xlim([Bee(1)-1 Bee(1)+1]); ylim([Bee(2)-1 Bee(2)+1]);
% title('Actual and Predicted Beacon Position');

% Absolute position error
err = Wasp-Bee;
```

```
    % Absolute heading error
     theta = asind(Bee(2)/norm(Bee));
     thetaErr = real(theta12 -theta);
    % Relative error
    %err = abs(Bee - Wasp)./abs(Bee);
%     fprintf('The X direction error is: %.3f.\n',err(1));
%     fprintf('The Y direction error is: %.3f.\n',err(2));
end
```

11.5.5 Attitude Code

The attitude code finds the position of the beacon in three directions using Euler angles to calculate the attitude. It also computes the error and was used to prove that the feasibility for incorporating attitude in this project was not possible.

```
%% Senior Projects Position Modeling Take 2
% Created: Anastasia Muszynski 10/7/19 based upon posModel by  Emily Webb
% Modified: Anastasia Muszynski 10/7/19
% A model for finding position of beacon in 3 directions
% based on posModel and information from Ambiguity Resolution in
% Interferometry (5 Antenna Direction finding)
%% Housekeeping!
 clear all; close all; clc;

%% Initialize Position of Antennas and Beacon
A1 = [0 0 0];
A2 = [0 0 0.2];
A3 = [0 0.2 0];
A4 = [0.2 0 0];


% Spherical Coordinates Whoo!
 psi = 45; % Azimuth angle, degrees
 theta = 45; % Elvation angle, degrees
 r = 200;    % Distance between receiver origin and transmitter origin (m)

% Center of Beacon
 Bee = [r*sind(theta)*cosd(psi), r*sind(theta)*sind(psi), r*cosd(theta)];
%% Attitude Part:
%Transmitters in same configuration as Recievers
T1 = A1; T2 = A2; T3 = A3; T4 = A4;
% Angles to rotate by (euler angle sequence ZYX)
```

```
eul = [pi/2 pi/3 pi/4]; % Convert to radians
rotM= eul2rotm(eul);
T2 = rotM*T2';
T3 = rotM*T3';
T4 = rotM*T4';
rotm2eul([T4 T3 T2]);


% Translate to Beacon location
T1 = T1 +Bee;
T2 = T2' +Bee;
T3 = T3' +Bee;
T4 = T4' +Bee;

%% Testing Angle calculations (just position in 3d)

thetavec = linspace(0,179, 360);
psivec = linspace(0, 359, 360);

% Iterate and check elevation test cases
theta2 = zeros(length(thetavec));
psi2 = zeros(length(thetavec));
psi = 200;
    for i = 1:length(thetavec)
        theta = thetavec(i);
        Bee = [r*sind(theta)*cosd(psi), r*sind(theta)*sind(psi), r*cosd(theta)];
        [BeeCalc, theta2(i), psi2(i)]= test3D(A1, A2, A3,A4, Bee, 0, 0);
    end

% Iterate and check azimuth test cases
theta3 = zeros( length(psivec));
psi3 = zeros( length(psivec));
    theta = 30;
    for i = 1:length(psivec)
        psi = psivec(i);
        Bee = [r*sind(theta)*cosd(psi), r*sind(theta)*sind(psi), r*cosd(theta)];
        [BeeCalc, theta3(i), psi3(i)]= test3D(A1, A2, A3, A4, Bee, 0, 0);
    end

  figure
     sgtitle({'Changing Elevation Angle Holding Azimuth Angle Constant at 200 degrees
     subplot(1,2,1)
     plot(thetavec, theta2)
     grid on
     xlabel('Actual Beacon Theta Angle')
     ylabel('Calculated Beacon Theta Angle')
     subplot(1,2,2)
     plot(thetavec, psi2)
     grid on
     xlabel('Actual Beacon Psi Angle')
     ylabel('Calculated Beacon Psi Angle')

     figure
     sgtitle({'Changing Azimuth Angle Holding Elevation Angle Constant at 30 Degrees'
     subplot(1,2,1)
     plot(psivec, theta3)
     grid on
     xlabel('Actual Beacon Theta Angle')
     ylabel('Calculated Beacon Theta Angle')
     subplot(1,2,2)
     plot(psivec, psi3)
     grid on
     xlabel('Actual Beacon Psi Angle')
     ylabel('Calculated Beacon Psi Angle')


  %% Testing Attitude  with phase errors

  N = 500; %Number of points
err = zeros(N,3);
for i = 1:N
    [T1calc, theta1, psi1] = test3D(A1, A2, A3, A4, T1, .0001, 0);
    [T2calc, theta2, psi2] = test3D(A1, A2, A3, A4, T2, .0001, 0);
    [T3calc, theta3, psi3] = test3D(A1, A2, A3, A4, T3, .0001, 0);
    [T4calc, theta4, psi4] = test3D(A1, A2, A3, A4, T4, .0001, 0);

    [eul2] = findAttitude(T1calc, T2calc, T3calc, T4calc);
    err(i,:) = (eul2-eul)*180/pi;
end

fprintf('Euler Angle Error- Range and Phase Error\n')
MakePlots(err)
```

```
   %% Testing Attitude  with  range errors

 N = 500; %Number of points
err = zeros(N,3);
for i = 1:N
    [T1calc, theta1, psi1] = test3D(A1, A2, A3, A4, T1, 0, 0.001);
    [T2calc, theta2, psi2] = test3D(A1, A2, A3, A4, T2, 0, 0.001);
    [T3calc, theta3, psi3] = test3D(A1, A2, A3, A4, T3, 0, 0.001);
    [T4calc, theta4, psi4] = test3D(A1, A2, A3, A4, T4, 0, 0.001);


    [eul2] = findAttitude(T1calc, T2calc, T3calc, T4calc);
    err(i,:) = (eul2-eul)*180/pi;
end
fprintf('Euler Angle Error- Phase Error\n')
 MakePlots(err)


      %% Testing Attitude  with  phase and range errors

 N = 500; %Number of points
err = zeros(N,3);
for i = 1:N
    [T1calc, theta1, psi1] = test3D(A1, A2, A3, A4, T1, 0.0001, 0.001);
    [T2calc, theta2, psi2] = test3D(A1, A2, A3, A4, T2, 0.0001, 0.001);
    [T3calc, theta3, psi3] = test3D(A1, A2, A3, A4, T3, 0.0001, 0.001);
    [T4calc, theta4, psi4] = test3D(A1, A2, A3, A4, T4, 0.0001, 0.001);

    [eul2] = findAttitude(T1calc, T2calc, T3calc, T4calc);
    err(i,:) = (eul2-eul)*180/pi;
end
fprintf('Euler Angle Error- Range and Phase Error\n')
MakePlots(err)

%% Functions to Find Position and attitude

function [BeeCalc, theta, psi] = test3D(A1, A2, A3, A4, Bee, phaseErr, rangeErr)
% Inputs 3 reciever  and 1 beacon locations in cartesian coordinates, as
% well as variation term (for playing around with error, multiply by
% variable inside function to apply an error)
% Separation distances between receiving antennas
s12 = norm(abs(A1 - A2));
s13 = norm(abs(A1 - A3));
s14 = norm(abs(A1 - A4));

% Find the ranges. (Assumed to be known)
r1 = norm(abs(Bee-A1));
r2 = norm(abs(Bee-A2));
r3 = norm(abs(Bee-A3));
r4 = norm(abs(Bee-A4));

%% Initialize other Parameters
f = 2.4e9; % Hz, frequency of sine wave
c = 299792458; % m/s, the speed of light
lambda = c/f; % m, wavelength
dspec = lambda/(2*pi); % "specific distance"

%% Actual Calculations
r12 = r1-r2; % distance difference (m)
r13 = r1-r3;
r14 = r1-r4;

% Error will only be applied if phaseErr is nonzero. phaseErr gives
% order of magnitude for error.
dphi12 = r12/dspec+normrnd(0,2)*phaseErr; % calculate phase difference between
dphi13 = r13/dspec+normrnd(0,2)*phaseErr; % calculate phase difference between
dphi14 = r14/dspec+normrnd(0,2)*phaseErr; % calculate phase difference between
% These values are what we will be measuring

% These are the calculations we will be doing to find
theta = acosd(lambda*dphi12/(2*pi*s12)); % Elevation
psi = atan2d(dphi13*s14,(dphi14*s13)); % Azimuth

if psi <0
  psi = 360+psi;
end

% Error will only be applied if rangeErr is nonzero. rangeErr gives
% order of magnitude for error.
r1 = r1+normrnd(0,2)*rangeErr;
```

```
    %% Finding Position, using A1 as reference antenna
    BeeCalc= [r1*sind(theta)*cosd(psi), r1*sind(theta)*sind(psi), r1*cosd(theta)];

end

function [eul] = findAttitude(Tloc1, Tloc2, Tloc3, Tloc4)
    % Transmitter Coordinate Frame unit vectors
    Xt = Tloc4 - Tloc1;
    Xt = Xt/norm(Xt);
    Yt = Tloc3 - Tloc1;
    Yt = Yt/norm(Yt);
    Zt = Tloc2 - Tloc1;
    Zt = Zt/norm(Zt);

    % Matrix describing transmitter attitude in receiver coordinate frame
    T = [Xt' Yt' Zt'];

    % Finding Euler angles for comparison (Could also find quaternion
    % here)
    eul = rotm2eul(real(T));
    % We will want to calculate the quaternion, but that is a later us
    % problem. Comparing the error of the euler angles will be more intuitive
    % Matlab also has built in quaternion functions, but we will need to
    % write these if we're coding in C

end

function [] = MakePlots(err)
figure
   hold on
   %Scatter plot of locations
   subplot(1,3,1)
   hold on
   histogram(err(:,1))
   title('Error in Computation')
   xlabel('Angle 1 Error (degrees)')
   hold off

   subplot(1,3,2)
   %Histogram of errors'
   hold on
```

```
    title('Error in Computation')
    histogram(err(:,2))
    xlabel('Angle 2 Error (degrees)')
    hold off

     subplot(1,3,3)
   %Histogram of errors'
    hold on
    title('Error in Computation')
    histogram(err(:,3))
    xlabel('Angle 3 Error (degrees)')
    hold off

    hold off
    fprintf(' Angle 1: Mean: %d degrees, St.Dev: %d degrees\n', mean(err(:,1)), std(
    fprintf(' Angle 2: Mean: %d degrees, St.Dev: %d degrees\n', mean(err(:,2)), std(
    fprintf(' Angle 3: Mean: %d degrees, St.Dev: %d degrees\n', mean(err(:,3)), std(
end
```